Cloud Container Engine

Guia de usuário

Edição 01

Data 2024-11-27

HUAWEI CLOUD COMPUTING TECHNOLOGIES CO., LTD.

Marcas registadas e permissões

e outras marcas registadas da Huawei são marcas registadas da Huawei Technologies Co., Ltd. Todos as outras marcas registadas e os nomes registados mencionados neste documento são propriedade dos seus respectivos detentores.

Aviso

Os produtos, os serviços e as funcionalidades adquiridos são estipulados pelo contrato estabelecido entre a Huawei Cloud e o cliente. Os produtos, os serviços e as funcionalidades descritos neste documento, no todo ou em parte, podem não estar dentro do âmbito de aquisição ou do âmbito de uso. Salvo especificação em contrário no contrato, todas as declarações, informações e recomendações neste documento são fornecidas "TAL COMO ESTÃO" sem garantias ou representações de qualquer tipo, sejam expressas ou implícitas.

As informações contidas neste documento estão sujeitas a alterações sem aviso prévio. Foram feitos todos os esforços na preparação deste documento para assegurar a exatidão do conteúdo, mas todas as declarações, informações e recomendações contidas neste documento não constituem uma garantia de qualquer tipo, expressa ou implícita.

Huawei Cloud Computing Technologies Co., Ltd.

Endereço: Huawei Cloud Data Center, Rua Jiaoxinggong
Avenida Qianzhong
Novo Distrito de Gui'an
Guizhou 550029
República Popular da China

Site: https://www.huaweicloud.com/intl/pt-br/

Índice

1 Operações de alto risco e soluções

Durante a implementação ou execução do serviço, você pode acionar operações de alto risco em diferentes níveis, causando falhas ou interrupção do serviço. Para ajudá-lo a estimar melhor e evitar riscos de operação, esta seção apresenta as consequências e soluções de operações de alto risco de várias dimensões, como clusters, nós, rede, balanceamento de carga, logs e discos EVS.

Clusters e nós

Tabela 1-1 Operações de alto risco e soluções

Categoria	Operação	Impacto	Solução
Nó principal	Modificar o grupo de segurança de um nó em um cluster	O nó principal pode estar indisponível. **NOTA** Regra de nomeação de um nó principal: *Cluster name*-**cce-control**-*Random number*	Restaure o grupo de segurança referindo-se à "Criação de um cluster" e permitir que o tráfego do grupo de segurança passe.
	Deixar o nó expirar ou destruir o nó	O nó principal estará indisponível.	Esta operação não pode ser desfeita.
	Reinstalar o SO	Os componentes no nó principal serão excluídos.	Esta operação não pode ser desfeita.
	Atualizar componentes no nó principal ou etcd	O cluster pode estar indisponível.	Volte para a versão original.
	Excluir ou formatando dados do diretório principal, como **/etc/kubernetes** no nó	O nó principal estará indisponível.	Esta operação não pode ser desfeita.
	Alterar o endereço IP do nó	O nó principal estará indisponível.	Altere o endereço IP de volta para o original.

Categoria	Operação	Impacto	Solução
	Modificar parâmetros de componentes principais (como etcd, kube-apiserver e docker)	O nó principal pode estar indisponível.	Restaure as configurações de parâmetro para os valores recomendados. Para mais detalhes, consulte **Gerenciamento de configuração de cluster**.
	Substituir o principal ou o certificado do etcd	O cluster pode estar indisponível.	Esta operação não pode ser desfeita.
Nó de trabalho	Modificar o grupo de segurança de um nó em um cluster	O nó pode estar indisponível. **NOTA** Regra de nomeação de um nó de trabalho: *Cluster name*-**cce-node**-*Random number*	Restaure o grupo de segurança e permita que o tráfego do grupo de segurança passe completamente.
	Excluir o nó	O nó ficará indisponível.	Esta operação não pode ser desfeita.
	Reinstalar o SO	Os componentes do nó são excluídos e o nó fica indisponível.	Redefina o nó. Para mais detalhes, consulte **Redefinição de um nó**.
	Atualizar o kernel do nó	O nó pode estar indisponível ou a rede pode estar anormal. **NOTA** A execução do nó depende da versão do kernel do sistema. Não use o comando **yum update** para atualizar ou reinstalar o kernel do sistema operacional de um nó, a menos que seja necessário. (Reinstalar o kernel do sistema operacional usando a imagem original ou outras imagens é uma operação arriscada.)	Se o sistema operacional for EulerOS 2.2, restaure o nó ou a conectividade de rede consultando **O que fazer se a rede do contêiner ficar indisponível depois que yum update for usado para atualizar o sistema operacional?** Se o sistema operacional não é EulerOS 2.2, você pode redefinir o nó. Para mais detalhes, consulte **Redefinição de um nó**.
	Alterar o endereço IP do nó	O nó ficará indisponível.	Altere o endereço IP de volta para o original.
	Modificar parâmetros de componentes principais (como kubelet e kube-proxy)	O nó pode ficar indisponível e os componentes podem ficar inseguros se as configurações relacionadas à segurança forem modificadas.	Restaure as configurações de parâmetro para os valores recomendados. Para mais detalhes, consulte **Configuração de um pool de nós**.

Categoria	Operação	Impacto	Solução
	Modificar a configuração do SO	O nó pode estar indisponível.	Restaure os itens de configuração ou redefina o nó. Para mais detalhes, consulte **Redefinição de um nó**.
	Excluindo ou modificando os diretórios **/opt/cloud/cce** e **/var/paas** e excluindo o disco de dados	O nó ficará despreparado.	Redefina o nó. Para mais detalhes, consulte **Redefinição de um nó**.
	Modificar a permissão do diretório de nó e a permissão do diretório de contêiner	As permissões serão anormais.	Não é aconselhado a modificar as permissões. Restaure as permissões se elas forem modificadas.
	Formatar ou particionar de discos do sistema, discos de Docker e discos de kubelet em nós.	O nó pode estar indisponível.	Redefina o nó. Para mais detalhes, consulte **Redefinição de um nó**.
	Instalar outros softwares nos nós	Isso pode causar exceções nos componentes do Kubernetes instalados no nó e tornar o nó indisponível.	Desinstale o software que foi instalado e restaure ou redefina o nó. Para mais detalhes, consulte **Redefinição de um nó**.
	Modificar configurações do NetworkManager	O nó ficará indisponível.	Redefina o nó. Para mais detalhes, consulte **Redefinição de um nó**.
	Exclua imagens do sistema, como **cce-pause**, do nó.	Os contêineres não podem ser criados e as imagens do sistema não podem ser extraídas.	Copie a imagem de outro nó normal para restauração.

Rede e balanceamento de carga

Tabela 1-2 Operações de alto risco e soluções

Operação	Impacto	Como evitar/corrigir
Alterar o valor do parâmetro do kernel **net.ipv4.ip_forward** para **0**	A rede se torna inacessível.	Altere o valor para **1**.
Alterando o valor do parâmetro do kernel **net.ipv4.tcp_tw_recycle** para **1**	O serviço de NAT torna-se anormal.	Altere o valor para **0**.
Alterar o valor do parâmetro do kernel **net.ipv4.tcp_tw_reuse** para **1**	A rede torna-se anormal.	Altere o valor para **0**.
Não configurar o grupo de segurança do nó para permitir que pacotes UDP passem pela porta 53 do bloco CIDR do contêiner	O DNS no cluster não pode funcionar corretamente.	Restaure o grupo de segurança referindo-se a **Compra de um cluster do CCE** e permitir que o tráfego do grupo de segurança passe completamente.
Criar um ouvinte personalizado no console do ELB para o balanceador de carga gerenciado pelo CCE	Os itens modificados são redefinidos pelo CCE ou a entrada é defeituosa.	Use o arquivo YAML do Serviço para criar automaticamente um ouvinte.
Vincular um back-end definido pelo usuário no console do ELB ao balanceador de carga gerenciado pelo CCE.		Não vincule manualmente qualquer back-end.
Alterar certificado ELB no console do ELB para o balanceador de carga gerenciado pelo CCE.		Use o arquivo YAML da entrada para gerenciar certificados automaticamente.
Alterar o nome do ouvinte no console do ELB para o ouvinte do ELB gerenciado pelo CCE.		Não altere o nome do ouvinte do ELB gerenciado pelo CCE.
Alterar a descrição de balanceadores de carga, ouvintes e políticas de encaminhamento gerenciadas pelo CCE no console do ELB.		Não modifique a descrição de balanceadores de carga, ouvintes ou políticas de encaminhamento gerenciadas pela CCE.

Operação	Impacto	Como evitar/corrigir
Apagar recursos de CRD de network-attachment-definitions de default-network.	A rede de contêineres é desconectada ou o cluster não é excluído.	Se os recursos forem excluídos por engano, use as configurações corretas para criar os recursos de rede padrão.

Logs

Tabela 1-3 Operações de alto risco e soluções

Operação	Impacto	Solução
Excluir o diretório **/tmp/ccs-log-collector/pos** na máquina de host	Os logs são coletados repetidamente.	Nenhuma
Excluir o diretório **/tmp/ccs-log-collector/buffer** na máquina de host	Logs são perdidos.	Nenhuma

Discos EVS

Tabela 1-4 Operações de alto risco e soluções

Operação	Impacto	Solução	Observações
Desmontar manualmente um disco EVS no console	Um erro de I/O ocorre quando os dados são gravados em um pod.	Exclua o caminho de montagem do nó e programe o pod novamente.	O arquivo no pod registra o local onde os arquivos devem ser coletados.
Desmontar o caminho de montagem do disco no nó	Os dados do pod são gravados em um disco local.	Volte a montar o caminho correspondente para o pod.	O buffer contém arquivos de cache de log a serem consumidos.
Operar discos EVS no nó	Os dados do pod são gravados em um disco local.	Nenhuma	Nenhuma

2 Clusters

2.1 Visão geral de cluster

2.1.1 Informações básicas do cluster

Kubernetes é um mecanismo de orquestração de contêineres de código aberto para automatizar a implementação, o dimensionamento e o gerenciamento de aplicações em contêineres.

Para os desenvolvedores, o Kubernetes é um sistema operacional de cluster. O Kubernetes fornece descoberta de serviços, escalabilidade, balanceamento de carga, auto-reparo e até eleição de líderes, liberando os desenvolvedores de configurações relacionadas à infraestrutura.

Rede de cluster

Uma rede de cluster pode ser dividida em três tipos de rede:

- Rede de nó: os endereços IP são atribuídos aos nós em um cluster.
- Rede de contêiner: os endereços IP são atribuídos a contêineres em um cluster para comunicação. Atualmente, vários modelos de rede de contêineres são suportados e cada modelo tem seu próprio mecanismo de trabalho.
- Rede de Serviço: um Serviço é um objeto do Kubernetes usado para acessar contêineres. Cada Serviço tem um endereço IP estático.

Ao criar um cluster, selecione um bloco CIDR adequado para cada rede. Certifique-se de que os blocos CIDR não entram em conflito entre si e têm endereços IP disponíveis suficientes. **Você não pode alterar o modelo de rede de contêiner após a criação do cluster.** Planeje o modelo de rede de contêineres adequadamente com antecedência.

É aconselhável aprender sobre a rede de cluster e modelos de rede de contêiner antes de criar um cluster. Para mais detalhes, consulte **Modelos de rede de contêineres**.

Nós principais e escala de cluster

Quando você cria um cluster no CCE, você pode ter um ou três nós principais. Três nós principais podem criar um cluster no modo HA.

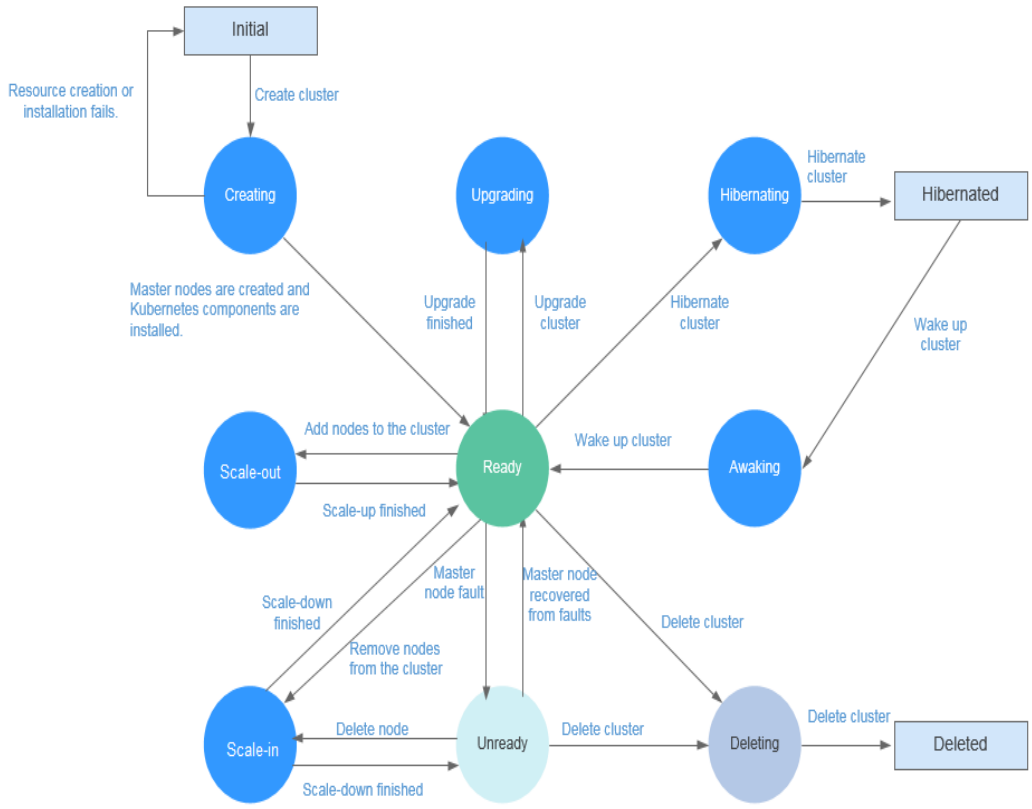
As especificações do nó principal decidem o número de nós que podem ser gerenciados por um cluster. Você pode selecionar a escala de gerenciamento de cluster, por exemplo, 50 ou 200 nós.

Ciclo de vida do cluster

Tabela 2-1 Status do cluster

Estado	Descrição
Creating	Um cluster está sendo criado e solicitando recursos de nuvem.
Running	Um cluster está sendo executado corretamente.
Hibernating	Um cluster está hibernando.
Awaking	Um aglomerado está sendo acordado.
Upgrading	Um cluster está sendo atualizado.
Unavailable	Um cluster não está disponível.
Deleting	Um cluster está sendo excluído.

Figura 2-1 Transição de status do cluster



2.1.2 Observações de versão do Kubernetes

2.1.2.1 Observações de versão Kubernetes 1.25

CCE foi aprovado pelo Programa certificado de conformidade com Kubernetes e é uma oferta certificada do Kubernetes. Esta seção descreve as atualizações no CCE de Kubernetes 1.25.

Alterações e depreciações de recursos

Observações de versão Kubernetes 1.25

- PodSecurityPolicy é substituída pelo Pod Security Admission. Para obter detalhes sobre a migração, consulte **Migração de PodSecurityPolicy para o controlador de admissão da PodSecurity**.
- SeccompDefault está em Beta. Para ativar esse recurso, adicione o parâmetro de inicialização **--seccomp-default=true** ao kubelet. Desta forma, seccomp é definido como **RuntimeDefault** por padrão, melhorando a segurança do sistema. Clusters da v1.25 não usam mais **seccomp.security.alpha.kubernetes.io/pod** e **container.seccomp.security.alpha.kubernetes.io/annotation** para usar seccomp. Substitua-os pelo campo **securityContext.seccompProfile** no pod ou contêiner. Para obter detalhes, consulte **Configuração de um contexto de segurança para um pod ou contêiner**.

  NOTA

  Depois que o recurso é ativado, as chamadas de sistema exigidas pela aplicação podem ser restringidas pelo tempo de execução. Certifique-se de que a depuração seja executada no ambiente de teste e que a aplicação não seja afetada.
- EndPort na Política de rede está estável. Esse recurso é incorporado na versão 1.21. EndPort é adicionada a NetworkPolicy para que você especifique um intervalo de portas.
- A partir da v1.25, o Kubernetes não oferece mais suporte à autenticação de certificado gerada usando o algoritmo SHA1WithRSA ou ECDSAWithSHA1. É aconselhável usar o algoritmo SHA256.

Observações de versão Kubernetes 1.24

- O Dockershim foi marcado como obsoleto no Kubernetes 1.20 e oficialmente removido do código do kubelet no Kubernetes 1.24. Se você quiser usar o contêiner Docker, alterne para cri-dockerd ou outros tempos de execução que suportem CRI, como containerd e CRI-O.

  Para obter detalhes sobre como alternar do mecanismo Docker para containerd, consulte **Migração de nós do Docker para containerd**.

  NOTA

  Verifique se há agentes ou aplicações que dependem do Docker. Por exemplo, se **docker ps**, **docker run** e **docker inspect** forem usados, certifique-se de que vários tempos de execução sejam compatíveis e alterne para CRI padrão.
- As APIs beta são desabilitadas por padrão. Quando algumas APIs beta de longo prazo são removidas do Kubernetes, 90% dos administradores de cluster não se importam com as APIs beta. Recursos beta não são recomendados no ambiente de produção. No entanto, devido à política de ativação padrão, essas APIs são ativadas no ambiente de produção, incorrendo em riscos. Portanto, na v1.24 e versões posteriores, as APIs beta são desativadas por padrão, exceto para as APIs beta ativadas.

- LegacyServiceAccountTokenNoAutoGeneration vai para a versão beta. Por padrão, esse recurso é ativado, onde nenhum token secreto é gerado automaticamente para uma conta de serviço. Para usar um token que nunca expira, crie um segredo e monte-o. Para obter detalhes, consulte **Segredos de token da conta de serviço**.
- **service.alpha.kubernetes.io/tolerate-unready-endpoints** é substituído por **Service.spec.publishNotReadyAddresses**.
- A tag **Service.Spec.LoadBalancerIP** está obsoleta e pode ser removida em versões posteriores. Use uma anotação personalizada.

Referências

Para obter mais detalhes sobre a comparação de desempenho e a evolução da função entre o Kubernetes 1.25 e outras versões, consulte os seguintes documentos:

- **Observações de versão Kubernetes 1.25**
- **Observações de versão Kubernetes 1.24**

2.1.2.2 Observações de versão Kubernetes 1.23

CCE foi aprovado pelo Programa certificado de conformidade com Kubernetes e é uma oferta certificada do Kubernetes. Esta seção descreve as atualizações no CCE de Kubernetes 1.23.

Alterações e depreciações de recursos

Observações de versão Kubernetes 1.23

- FlexVolume está obsoleto. Use CSI.
- HorizontalPodAutoscaler v2 é promovido a GA, e a API do HorizontalPodAutoscaler v2 é gradualmente estável na versão 1.23. A API de HorizontalPodAutoscaler v2beta2 não é recomendada. Use a API v2.
- **PodSecurity** passa para a versão beta, substituindo a PodSecurityPolicy obsoleta. PodSecurity é um controlador de admissão que aplica padrões de segurança de pods no namespace com base em rótulos específicos que definem o nível de imposição. PodSecurity é habilitado por padrão na versão 1.23.

Observações de versão Kubernetes 1.22

- Ingresses não suportam mais as APIs networking.k8s.io/v1beta1 e extensions/v1beta1. Se você usar a API de uma versão anterior para gerenciar ingresses, uma aplicação não poderá ser exposta a serviços externos. Use networking.k8s.io/v1.
- CustomResourceDefinitions não suporta mais a API apiextensions.k8s.io/v1beta1. Se você usar a API de uma versão anterior para criar um CRD, a criação falhará, o que afeta o controlador que reconcilia esse CRD. Use apiextensions.k8s.io/v1.
- ClusterRoles, ClusterRoleBindings, Roles e RoleBindings não suportam mais a API rbac.authorization.k8s.io/v1beta1. Se você usar a API de uma versão anterior para gerenciar recursos do RBAC, o controle de permissões de aplicação será afetado e até mesmo não poderá funcionar no cluster. Use rbac.authorization.k8s.io/v1.
- O ciclo de lançamento do Kubernetes é alterado de quatro lançamentos por ano para três lançamentos por ano.
- StatefulSets suportam **minReadySeconds**.
- Durante o ajuste de escala, os pods são selecionados e excluídos aleatoriamente com base no UID do pod por padrão (LogarithmicScaleDown). Esse recurso aumenta a

aleatoriedade dos pods a serem excluídos e alivia os problemas causados pelas restrições de propagação da topologia do pod. Para obter mais informações, consulte **KEP-2185** e **edição 96748**.

- O recurso **BoundServiceAccountTokenVolume** é estável. Esse recurso melhora a segurança do token da conta de serviço e altera o método de montagem de tokens em pods. Os clusters do Kubernetes v1.21 e posteriores ativam essa abordagem por padrão.

Referências

Para obter mais detalhes sobre a comparação de desempenho e a evolução da função entre o Kubernetes 1.23 e outras versões, consulte os seguintes documentos:

- **Observações de versão Kubernetes 1.23**
- **Observações de versão Kubernetes 1.22**

2.1.2.3 Observações de versão Kubernetes 1.21

CCE foi aprovado pelo Programa certificado de conformidade com Kubernetes e é uma oferta certificada do Kubernetes. Esta seção descreve as atualizações no CCE de Kubernetes 1.21.

Alterações e depreciações de recursos

Observações de versão Kubernetes 1.21

- CronJob está agora no estado estável e o número da versão é alterado para batch/v1.
- O imutável Segredo e ConfigMap foram atualizados para o estado estável. Um novo campo imutável é adicionado a esses objetos para rejeitar alterações. A rejeição protege os clusters de atualizações acidentais que podem causar interrupções na aplicação. Como esses recursos são imutáveis, o kubelet não monitora ou pesquisa mudanças. Isso reduz a carga do kube-apiserver e melhora a escalabilidade e o desempenho de seus clusters. Para obter mais informações, consulte **ConfigMaps imutáveis**.
- O desligamento gracioso do nó foi atualizado para o estado de teste. Com esta atualização, o kubelet pode detectar que um nó está desligado e terminar graciosamente os pods no nó. Antes dessa atualização, quando o nó era desligado, seu pod não seguia o ciclo de vida de encerramento esperado, o que causava problemas de carga de trabalho. Agora, o kubelet pode usar o systemd para detectar os sistemas que estão prestes a ser desligados e notificar os pods em execução para finalizá-los graciosamente.
- Para um pod com vários containers, você pode usar **kubectl.kubernetes.io/** para pré-selecionar containers.
- PodSecurityPolicy está preterida. Para mais detalhes, consulte **https://kubernetes.io/blog/2021/04/06/podsecuritypolicy-deprecation-past-present-and-future/**.
- O recurso **BoundServiceAccountTokenVolume** entrou no teste beta. Esse recurso melhora a segurança do token da conta de serviço e altera o método de montagem de tokens em pods. Os clusters do Kubernetes v1.21 e posteriores habilitam essa abordagem por padrão.

Observações de versão Kubernetes 1.20

- A prioridade e a equidade da API atingiram o estado de teste e estão habilitadas por padrão. Isso permite que o kube-apiserver classifique as requisições recebidas por prioridade. Para obter mais informações, consulte **Prioridade e equidade da API**.

- O bug de **exec probe timeouts** foi corrigido. Antes que esse bug seja corrigido, a sonda exec não considera o campo **timeoutSeconds**. Em vez disso, a sonda será executada indefinidamente, mesmo após o data limite configurado. Ela irá parar até que o resultado seja retornado. Agora, se nenhum valor for especificado, o valor padrão é usado, ou seja, um segundo. Se o tempo de detecção exceder um segundo, a verificação de integridade da aplicação pode falhar. Atualize o campo **timeoutSeconds** para as aplicações que usam esse recurso durante a atualização. O reparo fornecido pelo recurso ExecProbeTimeout recém-introduzido permite que o operador de cluster restaure o comportamento anterior, mas esse comportamento será bloqueado e removido em versões posteriores.
- RuntimeClass entra no estado estável. RuntimeClass fornece um mecanismo para suportar vários tempos de execução em um cluster e expor informações sobre o tempo de execução do contêiner ao plano de controle.
- A depuração do kubectl atingiu o estado de teste. A depuração do kubectl fornece suporte para fluxos de trabalho de depuração comuns.
- Dockershim foi marcado como preterido no Kubernetes 1.20. Atualmente, você pode continuar a usar o Docker no cluster. Essa alteração é irrelevante para a imagem de contêiner usada pelos clusters. Você ainda pode usar o Docker para criar suas imagens. Para obter mais informações, consulte **Perguntas frequentes sobre a depreciação de Dockershim**.

Referências

Para obter mais detalhes sobre a comparação de desempenho e a evolução da função entre o Kubernetes 1.21 e outras versões, consulte os seguintes documentos:

- **Observações de versão Kubernetes 1.21**
- **Observações de versão Kubernetes 1.20**

2.1.2.4 Observações de versão Kubernetes 1.19

CCE foi aprovado pelo Programa certificado de conformidade com Kubernetes e é uma oferta certificada do Kubernetes. Esta seção descreve as atualizações no CCE de Kubernetes 1.19.

Alterações e depreciações de recursos

Observações de versão Kubernetes v1.19

- Os volumes na árvore do vSphere podem ser migrados para os drivers de CSI do vSphere. O plug-in vSphere Volume na árvore não é mais usado e será excluído em versões posteriores.
- **apiextensions.k8s.io/v1beta1** foi preterido. É aconselhável usar **apiextensions.k8s.io/v1**.
- **apiregistration.k8s.io/v1beta1** foi preterido. É aconselhável usar **apiregistration.k8s.io/v1**.
- **authentication.k8s.io/v1beta1** e **authorization.k8s.io/v1beta1** foram preteridos e serão removidos do Kubernetes 1.22. É aconselhável que você use **authentication.k8s.io/v1** e **authorization.k8s.io/v1**.
- **autoscaling/v2beta1** foi preterido. Você é aconselhado a usar **autoscaling/v2beta2**.
- **coordination.k8s.io/v1beta1** foi preterido no Kubernetes 1.19 e será removido da versão 1.22. É aconselhável usar **coordination.k8s.io/v1**.

- kube-apiserver: a API **componentstatus** foi preterida.
- kubeadm: o comando **kubeadm config view** foi preterido e será excluído em versões posteriores. Use **kubectl get cm -o yaml -n kube-system kubeadm-config** para obter diretamente a configuração do kubeadm.
- kubeadm: o comando **kubeadm alpha kubelet config enable-dynamic** foi preterido.
- kubeadm: o sinalizador **--use-api** no comando **kubeadm alpha certs renew** foi preterido.
- O Kubernetes não suporta mais a criação de imagens **hyperkube**.
- O sinalizador **--export** é removido do comando **kubectl get**.
- O recurso alpha **ResourceLimitsPriorityFunction** foi excluído.
- **storage.k8s.io/v1beta1** foi preterido. É aconselhável usar **storage.k8s.io/v1**.

Observações do Kubernetes v1.18

- kube-apiserver
  - Todos os recursos nas versões da API **apps/v1beta1** e **apps/v1beta2** não são mais veiculados. Você pode usar a versão da API **apps/v1**.
  - DaemonSets e Implementações e ReplicaSets na versão da API **extensions/v1beta1** não estão mais disponíveis. Você pode usar a versão da API **apps/v1**.
  - NetworkPolicies na versão da API **extensions/v1beta1** não são mais disponíveis. Você pode usar a versão da API **networking.k8s.io/v1**.
  - PodSecurityPolicies na versão da API **extensions/v1beta1** não são mais disponíveis. Migre para usar a versão da API **policy/v1beta1**.
- kubelet
  - **--redirect-container-streaming** não é recomendado e será depreciado na v1.20.
  - O ponto final de medição de recursos **/metrics/resource/v1alpha1** e todos os padrões de medição sob esse ponto final foram preteridos. Em vez disso, use os padrões de medição no ponto final **/metrics/resource**:
    - scrape_error --> scrape_error
    - node_cpu_usage_seconds_total --> node_cpu_usage_seconds
    - node_memory_working_set_bytes --> node_memory_working_set_bytes
    - container_cpu_usage_seconds_total --> container_cpu_usage_seconds
    - container_memory_working_set_bytes --> container_memory_working_set_bytes
    - scrape_error --> scrape_error
  - Em versões futuras, o kubelet não criará mais o diretório de destino **CSI NodePublishVolume** de acordo com as especificações de CSI. Talvez seja necessário atualizar o driver CSI de acordo para criar e processar corretamente o caminho de destino.
- kube-proxy
  - Não é aconselhável usar os sinalizadores **--healthz-port** e **--metrics-port**. Use **--healthz-bind-address** e **--metrics-bind-address** em vez disso.
  - A opção de função **EndpointSliceProxying** é adicionada para controlar o uso de EndpointSlices no kube-proxy. Esta função está desativada por padrão.
- kubeadm

  - O sinalizador **--kubelet-version** do **kubeadm upgrade node** foi preterido e será excluído em versões posteriores.
  - O sinalizador **--use-api** no comando **kubeadm alpha certs renew** foi preterido.
  - O kube-dns foi preterido e não será mais suportado em versões futuras.
  - A estrutura ClusterStatus no kubeadm-config de ConfigMap foi preterida e será excluída em versões posteriores.
- kubectl
  - Não é aconselhável usar valores booleanos e não definidos para **--dry-run**. **server|client|none** é usado na nova versão.
  - **--server-dry-run** foi preterido para **kubectl apply** e substituído por **--dry-run=server**.
- Complementos

O complemento de monitoramento de clusters é excluído.

- kube-scheduler
  - A métrica **scheduling_duration_seconds** foi preterida.
  - As métricas **scheduling_algorithm_predicate_evaluation_seconds** e **scheduling_algorithm_priority_evaluation_seconds counters** não são mais usadas e são substituídas por **framework_extension_point_duration_seconds[extension_point="Filter"]** e **framework_extension_point_duration_seconds[extension_point="Score"]**.
  - A política do agendador AlwaysCheckAllPredictes foi preterida.
- Outras alterações
  - O componente k8s.io/node-api não é mais atualizado. Em vez disso, você pode usar o tipo **RuntimeClass** em **k8s.io/api** e os clientes gerados em **k8s.io/client-go**.
  - O rótulo **client** foi excluído de **apiserver_request_total**.

Referências

Para obter mais detalhes sobre a comparação de desempenho e a evolução da função entre o Kubernetes 1.19 e outras versões, consulte os seguintes documentos:

- **Observações de versão Kubernetes v1.19.0**
- **Observações de versão Kubernetes v1.18.0**

2.1.3 Observações de lançamento para versões de cluster do CCE

Versão 1.27

AVISO

No CCE v1.27 e versões posteriores, todos os nós oferecem suporte apenas ao mecanismo de contêiner. Para migrar nós do Docker para containerd, siga as operações descritas em **Migração de nós do Docker para containerd**.

Tabela 2-2 Observações de lançamento para o patch de v1.27

Versão do patch do cluster do CCE	Versão do Kubernetes	Atualizações de funcionalidades	Optimização	Fixação de vulnerabilidades
v1.27.2-r0	v1.27.2	● Volcano suporta agendamento de afinidade de conjunto de nós. ● Volcano suporta reagendamento de carga de trabalho.	-	Correção de alguns problemas de segurança.
v1.27.1-r10	v1.27.2	-	Os eventos gerados durante o dimensionamento do pool de nós foram otimizados.	Correção de alguns problemas de segurança.
v1.27.1-r0	v1.27.2	Os clusters do CCE da v1.27 são liberados pela primeira vez. ● Tanto o despejo suave quanto o despejo rígido são compatíveis com as configurações de pool de nós. ● As tags do TMS podem ser adicionadas aos discos EVS criados automaticamente para facilitar o gerenciamento de custos.	Nenhuma	Nenhuma

Versão 1.25

AVISO

Todos os nós nos clusters do CCE da versão 1.25, exceto os que executam o EulerOS 2.5, usam containerd por padrão.

Tabela 2-3 Observações de lançamento para o patch de v1.25

Versão do patch do cluster do CCE	Versão do Kubernetes	Atualizações de funcionalidades	Optimização	Fixação de vulnerabilidades
v1.25.5-r0	v1.25.5	● Volcano suporta agendamento de afinidade de conjunto de nós. ● Volcano suporta reagendamento de carga de trabalho.	-	Correção de alguns problemas de segurança.
v1.25.4-r10	v1.25.5	-	Os eventos gerados durante o dimensionamento do pool de nós foram otimizados.	Correção de alguns problemas de segurança.
v1.25.4-r0	v1.25.5	● Tanto o despejo suave quanto o despejo rígido são compatíveis com as configurações de pool de nós. ● As tags do TMS podem ser adicionadas aos discos EVS criados automaticamente para facilitar o gerenciamento de custos.	Nenhuma	Correção de alguns problemas de segurança.
v1.25.3-r10	v1.25.5	● Os clusters do CCE suportam balanceadores de carga dedicados. ● O intervalo de tempo limite pode ser configurado para um balanceador de carga.	Os parâmetros de alta frequência do kube-apiserver são configuráveis.	Correção de alguns problemas de segurança.

Versão do patch do cluster do CCE	Versão do Kubernetes	Atualizações de funcionalidades	Optimização	Fixação de vulnerabilidades
v1.25.3-r0	**v1.25.5**	• ENIs do CCE Turbo suportam endereços IP fixos. Para obter detalhes, consulte **Configuração de um endereço IP estático para um pod**. • As ENIs do CCE Turbo suportam alocação automática e vinculação de EIPs. Para obter detalhes, consulte **Configuração de um EIP estático para um pod**. • Implementação híbrida aprimorada de clusters do CCE Turbo: a largura de banda da rede de saída é garantida pela prioridade da rede. Para obter detalhes, consulte **Garantia de largura de banda da rede de saída**. • Os clusters do CCE Turbo suportam a associação entre namespaces e blocos CIDR de contêiner. Para detalhes, veja **NetworkAttachmentDefinition**. • O CPU Burst é suportado para evitar que a limitação de tráfego da CPU afete os serviços sensíveis à latência. Para obter detalhes, consulte **CPU Burst**.	Estabilidade de rede aprimorada dos clusters do CCE Turbo quando suas especificações são modificadas.	Correção de alguns problemas de segurança.
v1.25.1-r0	**v1.25.5**	Os clusters do CCE da v1.25 são liberados pela primeira vez. Para obter mais informações, consulte **Observações de versão do Kubernetes 1.25**.	Nenhuma	Nenhuma

Versão 1.23

Tabela 2-4 Observações de lançamento para o patch de v1.23

Versão do patch do cluster do CCE	Versão do Kubernetes	Atualizações de funcionalidades	Optimização	Fixação de vulnerabilidades
v1.23.10-r0	v1.23.11	● Volcano suporta agendamento de afinidade de conjunto de nós. ● Volcano suporta reagendamento de carga de trabalho.	-	Correção de alguns problemas de segurança.
v1.23.9-r10	v1.23.11	-	Os eventos gerados durante o dimensionamento do pool de nós foram otimizados.	Correção de alguns problemas de segurança.
v1.23.9-r0	v1.23.11	● Tanto o despejo suave quanto o despejo rígido são compatíveis com as configurações de pool de nós. ● As tags do TMS podem ser adicionadas aos discos EVS criados automaticamente para facilitar o gerenciamento de custos.	Nenhuma	Correção de alguns problemas de segurança.
v1.23.8-r10	v1.23.11	● Os clusters do CCE suportam balanceadores de carga dedicados. ● O intervalo de tempo limite pode ser configurado para um balanceador de carga.	Os parâmetros de alta frequência do kube-apiserver são configuráveis.	Correção de alguns problemas de segurança.

Versão do patch do cluster do CCE	Versão do Kubernetes	Atualizações de funcionalidades	Optimização	Fixação de vulnerabilidades
v1.23.8-r0	v1.23.11	• ENIs do CCE Turbo suportam endereços IP fixos. Para obter detalhes, consulte **Configuração de um endereço IP estático para um pod**. • As ENIs do CCE Turbo suportam alocação automática e vinculação de EIPs. Para obter detalhes, consulte **Configuração de um EIP estático para um pod**. • Implementação híbrida aprimorada de clusters do CCE Turbo: a largura de banda da rede de saída é garantida pela prioridade da rede. Para obter detalhes, consulte **Garantia de largura de banda da rede de saída**. • Os clusters do CCE Turbo suportam a associação entre namespaces e blocos CIDR de contêiner. Para detalhes, veja **NetworkAttachment-Definition**. • O CPU Burst é suportado para evitar que a limitação de tráfego da CPU afete os serviços sensíveis à latência. Para obter detalhes, consulte **CPU Burst**.	• Confiabilidade aprimorada do Docker durante as atualizações. • Sincronização de tempo de nó otimizada.	Correção de alguns problemas de segurança.

Versão do patch do cluster do CCE	Versão do Kubernetes	Atualizações de funcionalidades	Optimização	Fixação de vulnerabilidades
v1.23.7-r20	v1.23.11	Nenhuma	• A estabilidade aprimorada da interconexão entre Serviços/ingresses e balanceadores de carga. • Aumento da confiabilidade dos nós com vários discos de dados anexados.	Correção de alguns problemas de segurança.
v1.23.7-r10	v1.23.11	Nenhuma	• Confiabilidade aprimorada do Docker durante as atualizações. • Confiabilidade aprimorada do contêiner após desconexões. • Segurança reforçada para cenários onde ocorreu um erro durante a otimização do parâmetro do kernel.	Correção de alguns problemas de segurança.

Versão do patch do cluster do CCE	Versão do Kubernetes	Atualizações de funcionalidades	Optimização	Fixação de vulnerabilidades
v1.23.7-r0	v1.23.11		• Estabilidade de rede aprimorada dos clusters do CCE Turbo quando suas especificações são modificadas. • Estabilidade aprimorada da rede do nginx-ingress-controller quando o cluster é atualizado. • Sincronização de tempo de nó otimizada.	Correção de alguns problemas de segurança.
v1.23.6-r0	v1.23.11	• As portas TCP/UDP podem ser configuradas para os Serviços LoadBalancer. • Pod portão de prontidão é suportado.	• Confiabilidade aprimorada da tabela de fluxo quando a rede subjacente funciona mal. • Estabilidade aprimorada do sistema operacional com uma versão posterior do kernel em cenários de reinicialização, por exemplo, devido ao desligamento inesperado. • Métricas otimizadas de GPU/NPU do cAdvisor.	Correção de alguns problemas de segurança.

Versão do patch do cluster do CCE	Versão do Kubernetes	Atualizações de funcionalidades	Optimização	Fixação de vulnerabilidades
v1.23.5-r0	v1.23.11	• Os contêineres suportam o SFS 3.0 para armazenamento. • A detecção e o isolamento de falhas são suportados nos nós da GPU. • Os grupos de segurança podem ser personalizados por cluster. • Os clusters do CCE Turbo suportam pré-vinculação de ENIs por nó. • Logs do avião de controle podem ser coletados. • O Huawei Cloud EulerOS 2.0 desenvolvido pela Huawei é suportado. • containerd é suportado. • Os clusters do CCE Turbo suportam implementação híbrida e afinidade de maré da CPU.	• A versão ETCD do nó principal foi atualizada para a versão 3.5 do Kubernetes. • O desempenho de acesso ao Serviço nos nós EulerOS 2.8 é otimizado. • O agendamento é otimizado para que os pods sejam distribuídos uniformemente pelas AZs depois que os pods são reduzidos. • Otimização do uso de memória do kube-apiserver quando os CRDs são atualizados com frequência.	Correção de alguns problemas de segurança e das seguintes vulnerabilidades CVE: • **CVE-2022-3294** • **CVE-2022-3162** • **CVE-2022-3172** • **CVE-2021-25749**
v1.23.4-r10	v1.23.4	Nenhuma	Otimização do uso de memória do kube-apiserver quando os CRDs são atualizados com frequência.	Correção de alguns problemas de segurança.
v1.23.4-r0	v1.23.4	Os nós de Arm são suportados.	Nenhuma	Correção de alguns problemas de segurança.

Versão do patch do cluster do CCE	Versão do Kubernetes	Atualizações de funcionalidades	Optimização	Fixação de vulnerabilidades
1.23.3-r0	v1.23.4	● As métricas de monitoramento dos nós principais são disponibilizadas aos locatários. ● A pré-vinculação de sub-ENI é suportada para acelerar a inicialização das sub-ENI em clusters do CCE Turbo. ● Os pesos do servidor back-end podem ser configurados para os Serviços LoadBalancer. ● Os clusters do CCE suportam a implementação de clusters cruzados. ● Os clusters do CCE Turbo oferecem suporte à implementação híbrida quando os nós de VM são usados.	Confiabilidade aprimorada quando contêineres Kata são frequentemente criados ou excluídos.	Correção de alguns problemas de segurança.
1.23.1-r1	v1.23.4	A reserva de recursos do nó foi otimizada para que a exaustão de recursos possa ser detectada para melhorar a estabilidade do nó.	A compatibilidade de instalação do nó foi aprimorada.	Correção de alguns problemas de segurança.
v1.23.1-r0	v1.23.4	Os clusters do CCE da v1.23 são liberados pela primeira vez. Para obter mais informações, consulte **Observações de versão do Kubernetes 1.23**.	Nenhuma	Nenhuma

Versão 1.21

Tabela 2-5 Observações de lançamento para o patch de v1.21

Versão do patch do cluster do CCE	Versão do Kubernetes	Atualizações de funcionalidades	Optimização	Fixação de vulnerabilidades
v1.21.11-r20	v1.21.14	● Volcano suporta agendamento de afinidade de conjunto de nós. ● Volcano suporta reagendamento de carga de trabalho.	-	Correção de alguns problemas de segurança.
v1.21.11-r10	v1.21.14	-	Os eventos gerados durante o dimensionamento do pool de nós foram otimizados.	Correção de alguns problemas de segurança.
v1.21.11-r0	v1.21.14	● Tanto o despejo suave quanto o despejo rígido são compatíveis com as configurações de pool de nós. ● As tags do TMS podem ser adicionadas aos discos EVS criados automaticamente para facilitar o gerenciamento de custos.	Nenhuma	Correção de alguns problemas de segurança.
v1.21.10-r10	v1.21.14	● Os clusters do CCE suportam balanceadores de carga dedicados. ● O intervalo de tempo limite pode ser configurado para um balanceador de carga.	Os parâmetros de alta frequência do kube-apiserver são configuráveis.	Correção de alguns problemas de segurança.

Versão do patch do cluster do CCE	Versão do Kubernetes	Atualizações de funcionalidades	Optimização	Fixação de vulnerabilidades
v1.21.10-r0	v1.21.14	• ENIs do CCE Turbo suportam endereços IP fixos. Para obter detalhes, consulte **Configuração de um endereço IP estático para um pod**. • As ENIs do CCE Turbo suportam alocação automática e vinculação de EIPs. Para obter detalhes, consulte **Configuração de um EIP estático para um pod**.	• Confiabilidade aprimorada do Docker durante as atualizações. • Sincronização de tempo de nó otimizada. • Estabilidade aprimorada do tempo de execução do Docker para puxar imagens depois que os nós são reiniciados.	Correção de alguns problemas de segurança.
v1.21.9-r0	v1.21.14		Estabilidade de rede aprimorada dos clusters do CCE Turbo quando suas especificações são modificadas.	Correção de alguns problemas de segurança.
v1.21.8-r0	v1.21.14	• As portas TCP/UDP podem ser configuradas para os Serviços LoadBalancer. • Pod portão de prontidão é suportado.	• Confiabilidade aprimorada da tabela de fluxo quando a rede subjacente funciona mal. • Estabilidade aprimorada do sistema operacional com uma versão posterior do kernel em cenários de reinicialização, por exemplo, devido ao desligamento inesperado. • Métricas otimizadas de GPU/NPU do cAdvisor.	Correção de alguns problemas de segurança.

Versão do patch do cluster do CCE	Versão do Kubernetes	Atualizações de funcionalidades	Optimização	Fixação de vulnerabilidades
v1.21.7-r0	v1.21.14	• Os contêineres suportam o SFS 3.0 para armazenamento. • A detecção e o isolamento de falhas são suportados nos nós da GPU. • Os grupos de segurança podem ser personalizados por cluster. • Os clusters do CCE Turbo suportam pré-vinculação de ENIs por nó. • Logs do avião de controle podem ser coletados.	Estabilidade melhorada dos Serviços/ingresses de LoadBalancer com um grande número de conexões.	Correção de alguns problemas de segurança e das seguintes vulnerabilidades CVE: • CVE-2022-3294 • CVE-2022-3162 • CVE-2022-3172
1.21.6-r0	v1.21.7	Os nós de Arm são suportados.	Nenhuma	Correção de alguns problemas de segurança.
v1.21.5-r10	v1.21.7	Nenhuma	• Estabilidade de alocação aprimorada de uma rede de túneis quando os nós no plano de controle são intermitentemente desconectados. • Fortalecido para vulnerabilidades OVS.	Correção de alguns problemas de segurança.

Versão do patch do cluster do CCE	Versão do Kubernetes	Atualizações de funcionalidades	Optimização	Fixação de vulnerabilidades
v1.21.5-r0	v1.21.7	Nenhuma	• Estabilidade aprimorada das redes de túnel de contêiner durante as atualizações de cluster. • Restrições adicionadas na distribuição de topologia de pod. • Estabilidade aprimorada na desconexão de links quando um nó no plano de controle de cluster é desligado. • Estabilidade aprimorada quando os volumes são montados simultaneamente para pods.	Correção de alguns problemas de segurança.
v1.21.4-r10	v1.21.7	Nenhuma	Estabilidade aprimorada do NetworkManager de EulerOS 2.9.	Correção de alguns problemas de segurança.
v1.21.4-r0	v1.21.7	Nenhuma	• Estabilidade aprimorada dos contêineres Kata quando um driver da ENI é alterado. • Estabilidade aprimorada de acesso aos Serviços LoadBalancer durante a atualização da carga de trabalho e o dimensionamento dos nós.	Correção de alguns problemas de segurança.

Versão do patch do cluster do CCE	Versão do Kubernetes	Atualizações de funcionalidades	Optimização	Fixação de vulnerabilidades
v1.21.3-r10	v1.21.7	Nenhuma	Estabilidade aprimorada dos contêineres Kata quando um driver da ENI é alterado.	Correção de alguns problemas de segurança.
v1.21.3-r0	v1.21.7	Nenhuma	Os clusters do CCE Turbo suportam blocos CIDR de SNAT.	Correção de alguns problemas de segurança.
v1.21.2-r10	v1.21.7	Nenhuma	Estabilidade aprimorada da interconexão entre Serviços e balanceadores de carga.	Correção de alguns problemas de segurança.
v1.21.2-r0	v1.21.7	A reserva de recursos do nó foi otimizada para que a exaustão de recursos possa ser detectada para melhorar a estabilidade do nó.	● Capacidade aprimorada de atualização de cluster e confiabilidade. ● Armazenamento local de contêineres otimizado para melhorar a estabilidade.	Correção de alguns problemas de segurança.
1.21.1-r2	v1.21.7	● O armazenamento em contêiner suporta PVs locais. ● Os servidores de EulerOS 2.9 Kunpeng podem ser gerenciados. ● Tanto as redes de túnel de contêiner quanto as redes da VPC suportam ampla correspondência de uma versão do kernel do sistema operacional.	● Processo otimizado de instalação do nó para aumentar a confiabilidade da criação do nó. ● Otimização dos parâmetros do kernel do CentOS e EulerOS 2.5 para melhorar o desempenho do sistema operacional.	Correção de alguns problemas de segurança.

Versão do patch do cluster do CCE	Versão do Kubernetes	Atualizações de funcionalidades	Optimização	Fixação de vulnerabilidades
v1.21.1-r1	v1.21.7	Nenhuma	As redes de contêineres suportam ampla correspondência de uma versão do kernel do sistema operacional.	Correção de alguns problemas de segurança.
v1.21.1-r0	v1.21.7	Os clusters do CCE da v1.21 são liberados pela primeira vez. Para obter mais informações, consulte **Observações de versão do Kubernetes 1.21**.	Nenhuma	Nenhuma

Versão 1.19

Tabela 2-6 Observações de lançamento do patch de v1.19

Versão do patch do cluster do CCE	Versão do Kubernetes	Atualizações de funcionalidades	Optimização	Fixação de vulnerabilidades
v1.19.16-r60	v1.19.16	• Volcano suporta agendamento de afinidade de conjunto de nós. • Volcano suporta reagendamento de carga de trabalho.	-	Correção de alguns problemas de segurança.
v1.19.16-r50	v1.19.16	-	Os eventos gerados durante o dimensionamento do pool de nós foram otimizados.	Correção de alguns problemas de segurança.

Versão do patch do cluster do CCE	Versão do Kubernetes	Atualizações de funcionalidades	Optimização	Fixação de vulnerabilidades
v1.19.16-r40	v1.19.16	• Tanto o despejo suave quanto o despejo rígido são compatíveis com as configurações de pool de nós. • As tags do TMS podem ser adicionadas aos discos EVS criados automaticamente para facilitar o gerenciamento de custos.	Nenhuma	Correção de alguns problemas de segurança.
v1.19.16-r30	v1.19.16	• Os clusters do CCE suportam balanceadores de carga dedicados. • O intervalo de tempo limite pode ser configurado para um balanceador de carga.	Os parâmetros de alta frequência do kube-apiserver são configuráveis.	Correção de alguns problemas de segurança.
v1.19.16-r20	v1.19.16	• ENIs do CCE Turbo suportam endereços IP fixos. Para obter detalhes, consulte **Configuração de um endereço IP estático para um pod**. • As ENIs do CCE Turbo suportam alocação automática e vinculação de EIPs. Para obter detalhes, consulte **Configuração de um EIP estático para um pod**.	• Cloud Native 2.0 Networks permite especificar sub-redes para um namespace. • Estabilidade aprimorada do tempo de execução do Docker para puxar imagens depois que os nós são reiniciados. • Desempenho otimizado dos clusters do CCE Turbo na alocação de ENIs se nem todas as ENIs estiverem pré-vinculadas.	Correção de alguns problemas de segurança.
v1.19.16-r10	v1.19.16	Nenhuma	A estabilidade aprimorada da interconexão entre Serviços/ingresses e balanceadores de carga.	Correção de alguns problemas de segurança.

Versão do patch do cluster do CCE	Versão do Kubernetes	Atualizações de funcionalidades	Optimização	Fixação de vulnerabilidades
v1.19.16-r7	v1.19.16	Nenhuma	• Confiabilidade aprimorada do Docker durante as atualizações. • Sincronização de tempo de nó otimizada. • Confiabilidade aprimorada dos clusters do CCE Turbo quando as ENIs são pré-vinculadas.	Correção de alguns problemas de segurança.
v1.19.16-r6	v1.19.16		• Estabilidade aprimorada do contêiner configurado com QoS. • As políticas de reescrita de URL podem ser configuradas e modificadas para ingresses. • O uso de memória do kube-controller-manager é otimizado quando os recursos de CRD são atualizados com frequência.	Correção de alguns problemas de segurança.

Versão do patch do cluster do CCE	Versão do Kubernetes	Atualizações de funcionalidades	Optimização	Fixação de vulnerabilidades
v1.19.16-r5	v1.19.16	• As portas TCP/UDP podem ser configuradas para os Serviços LoadBalancer. • Pod portão de prontidão é suportado.	• Confiabilidade aprimorada da tabela de fluxo quando a rede subjacente funciona mal. • Estabilidade aprimorada do sistema operacional com uma versão posterior do kernel em cenários de reinicialização, por exemplo, devido ao desligamento inesperado.	Correção de alguns problemas de segurança.
v1.19.16-r4	v1.19.16	• Os contêineres suportam o SFS 3.0 para armazenamento. • A detecção e o isolamento de falhas são suportados nos nós da GPU. • Os grupos de segurança podem ser personalizados por cluster. • Os clusters do CCE Turbo suportam pré-vinculação de ENIs por nó.	• O agendamento é otimizado em nós de mancha. • Estabilidade aprimorada de execução a longo prazo do containerd quando os núcleos são vinculados. • Estabilidade melhorada dos Serviços/ingresses de LoadBalancer com um grande número de conexões. • Otimização do uso de memória do kube-apiserver quando os CRDs são atualizados com frequência.	Correção de alguns problemas de segurança e das seguintes vulnerabilidades CVE: • CVE-2022-3294 • CVE-2022-3162 • CVE-2022-3172

Versão do patch do cluster do CCE	Versão do Kubernetes	Atualizações de funcionalidades	Optimização	Fixação de vulnerabilidades
v1.19.16-r3	v1.19.16	Nenhuma	• O parâmetro de inicialização image-pull-progress-deadline pode ser reservado após uma atualização. • Os clusters do CCE Turbo suportam a pré-vinculação personalizada de ENI. • Correção do problema de alocação de rede de túnel inconsistente causada por desconexão intermitente entre os nós principais. • Estabilidade aprimorada dos clusters em execução em uma rede de túneis quando os nós no plano de controle são intermitentemente desconectados.	Correção de alguns problemas de segurança.
v1.19.16-r2	v1.19.16	Nenhuma	• Estabilidade aprimorada na desconexão de links quando um nó no plano de controle de cluster é desligado. • Estabilidade aprimorada quando os volumes são montados simultaneamente para pods.	Correção de alguns problemas de segurança.
v1.19.16-r1	v1.19.16	Nenhuma	Estabilidade aprimorada do NetworkManager de EulerOS 2.9.	Correção de alguns problemas de segurança.

Versão do patch do cluster do CCE	Versão do Kubernetes	Atualizações de funcionalidades	Optimização	Fixação de vulnerabilidades
v1.19.16-r0	v1.19.16	Nenhuma	Estabilidade aprimorada na atualização dos Serviços LoadBalancer quando as cargas de trabalho são atualizadas e os nós são expandidos ou reduzidos.	Correção de alguns problemas de segurança e das seguintes vulnerabilidades CVE: ● CVE-2021-25741 ● CVE-2021-25737
v1.19.10-r0	v1.19.10	Os clusters do CCE da v1.19 são liberados pela primeira vez. Para obter mais informações, consulte **Observações de versão do Kubernetes 1.19**.	Nenhuma	Nenhuma

2.2 Compra de um cluster

2.2.1 Clusters do CCE Turbo e clusters do CCE

Comparação entre clusters do CCE Turbo e clusters do CCE

A tabela a seguir lista as diferenças entre clusters do CCE Turbo e clusters do CCE:

Tabela 2-7 Tipos de cluster

Dimensão	Subdimensão	Cluster do CCE Turbo	Cluster do CCE
Cluster	Posicionamento	Cluster de container de última geração para Cloud Native 2.0 com computação acelerada, rede e agendamento	Cluster padrão para uso comercial comum
	Tipo de nó	Implementação híbrida de VMs e bare-metal servers	Implementação híbrida de VMs e bare-metal servers
Redes	Modelo	**Cloud Native Network 2.0**: aplica-se a cenários de grande escala e alto desempenho. Escala de rede: 2000 nós	**Cloud-native network 1.0** para cenários que não exigem alto desempenho ou envolvem implementação em larga escala. ● Modelo da rede de túnel ● Modelo da rede VPC
	Desempenho	A rede VPC e a rede de container são agrupadas em uma, atingindo zero perda de desempenho.	A rede VPC é sobreposta à rede de container, causando certa perda de desempenho.
	Isolamento da rede de container	Os pods podem ser associados diretamente a grupos de segurança para configurar políticas de isolamento para recursos dentro e fora de um cluster.	● Modelo da rede de túnel: as políticas de isolamento de rede são suportadas para comunicação intra-cluster (configurando políticas de rede). ● Modelo da rede VPC: o isolamento não é suportado.
Segurança	Isolamento	● Máquina física: Kata containers, fornecendo isolamento no nível da VM. ● VM: containers comuns são implementados.	Containers comuns são implementados e isolados por Cgroups.

Desempenho na criação de pods em lote em um cluster do CCE Turbo

Pods em um cluster do CCE Turbo solicitam interfaces de rede elásticas (ENIs) ou sub-ENIs da VPC. Atualmente, os pods são vinculados a ENIs ou sub-ENIs após a conclusão do agendamento do pod. A velocidade de criação do pod é limitada pela rapidez com que as NICs são criadas e vinculadas. A tabela a seguir descreve as restrições.

Tabela 2-8 Duração da criação da ENI

Tipo de nó	Tipo de ENI	Número máximo de ENI suportados	Vincular ENI ao nó	Disponibilidade da ENI	Controle de concorrência	Configuração de pré-vinculação
ECS	Sub-ENI	256	Especifique a ENI do nó para criar uma sub-ENI.	Dentro de um segundo	Nível do locatário: 600/minuto	Para clusters de versões anteriores a 1.19.16-r2, 1.21.5-r0 ou 1.23.3-r0, a pré-vinculação da ENI não é suportada. Para clusters de versões entre 1.19.16-r2, 1.21.5-r0 ou 1.23.3-r0 e 1.19.16-r4, 1.21.7-r0 ou 1.23.5-r0, a pré-vinculação dinâmica de ENI é suportada (nic-minimum-target=10; nic-warm-target=2). Para clusters das versões 1.23.5-r0, 1.19.16-r4, 1.21.7-r0 ou 1.25.1-r0 e suas versões posteriores, a pré-vinculação dinâmica de ENI é suportada (nic-minimum-target=10; nic-maximum-target=2; nic-warm-target=2; nic-max-above-warm-target=2).
BMS	ENI	128	Vincular uma ENI a um nó.	20s-30s	Nível de nó: 3 em simultâneo	Para clusters anteriores a 1.19.16-r4, 1.21.7-r0 e 1.23.5-r0, o número total de ENIs é baseado na a taxa de limite (nic-threshold=0.3:0.6). Para clusters de 1.19.16-r4, 1.21.7-r0, 1.23.5-r0, 1.25.1-r0 e posteriores, a pré-vinculação dinâmica é suportada (nic-minimum-target=10; nic-maximum-target=2; nic-warm-target=2; nic-max-above-warm-target=2).

Criar pods em nós do ECS (usando sub-ENIs)

- Se nenhuma ENI pré-vinculada estiver disponível no nó para o qual o pod está programado, a API para criar uma sub-ENI é chamada para criar uma sub-ENI em uma ENI do nó e alocar a sub-ENI ao pod.
- Se uma ENI pré-vinculada estiver disponível no nó para o qual o pod está programado, a sub-ENI não utilizada mais antiga será alocada ao pod.
- Limitado pela velocidade de criação simultânea de sub-ENIs, um máximo de 600 pods podem ser criados por minuto sem pré-vinculação. Se for necessária uma criação em maior escala, é possível configurar a pré-vinculação para sub-ENIs.

Criar pods em nós do BMS (usando sub-ENIs)

- Se nenhuma ENI pré-vinculada estiver disponível no nó para o qual o pod está programado, a API para vincular uma ENI ao nó é chamada para vincular e alocar uma ENI ao pod. Demora cerca de 20 a 30 segundos para vincular uma ENI a um nó do BMS.
- Se uma ENI pré-vinculada estiver disponível no nó para o qual o pod está programado, a ENI mais antiga não utilizada será alocado ao pod.
- Limitada pela velocidade de ligação de ENIs a nós do BMS, a velocidade de inicialização de pods no mesmo nó é de 3/20 segundos sem pré-vinculação. Portanto, é aconselhável pré-vincular ENIs para nós do BMS.

2.2.2 Compra de um cluster do CCE

No console do CCE, você pode facilmente criar clusters do Kubernetes. Depois que um cluster é criado, o nó principal é hospedado pelo CCE. Você só precisa criar nós de trabalho. Dessa forma, você pode implementar O&M econômica e implantação eficiente de serviços.

Restrições

- Durante a criação do nó, os pacotes de software são baixados do OBS usando o nome de domínio. Use um servidor DNS privado para resolver o nome de domínio do OBS e configure o endereço do servidor DNS da sub-rede em que o nó reside com um **endereço de servidor DNS privado** . Quando você cria uma sub-rede, o servidor DNS privado é usado por padrão. Se alterar o DNS de sub-rede, certifique-se de que o servidor DNS em utilização pode resolver o nome de domínio do OBS.
- Você pode criar no máximo 50 clusters em uma única região. Se mais clusters forem necessários, você pode clicar **aqui** para aumentar sua cota.
- Depois que um cluster é criado, os seguintes itens não podem ser alterados:
  - Tipo de cluster
  - Número de nós principais no cluster
  - AZ de um nó principal
  - Configuração de rede do cluster, como VPC, sub-rede, bloco CIDR de contêiner, bloco CIDR de serviço, configurações IPv6 e configurações de kube-proxy (**encaminhamento de solicitações**).
  - Modelo da rede. Por exemplo, altere **Tunnel network** para **VPC network**.

Procedimento

Passo 1 Efetue logon no **console do CCE** .

Passo 2 Escolha **Clusters**. Na página exibida, selecione o tipo de cluster a ser criado e clique em Buy .

Passo 3 Especificar parâmetros de cluster.

Basic Settings

- **Billing Mode**: permite que você escolha um modo de cobrança para o cluster, conforme necessário.
  - **Yearly/Monthly**: especifica um modo de cobrança pré-pago. Ele economiza dinheiro quando você tem uma boa idéia de quanto tempo você vai precisar dos discos. Você será cobrado antecipadamente com base na duração da assinatura.

    Se você escolher esse modo de cobrança, configure a duração necessária e determine se deseja renovar automaticamente a assinatura. (Se você comprar uma assinatura mensal, o período de renovação automática é de um mês. Se você comprar uma assinatura anual, o período de renovação automática é de um ano.)
  - Pay-per-use: é um modo de cobrança pós-pago. É adequado em cenários em que os recursos serão cobrados com base na frequência e duração do uso. Você pode provisionar ou excluir recursos a qualquer momento.
- **Cluster Name**: indica o nome do cluster a ser criado. O nome do cluster deve ser exclusivo na mesma conta.
- **Enterprise Project**:

  este parâmetro é exibido somente para usuários empresariais que ativaram a função de projeto empresarial.

  Depois que um projeto da empresa (por exemplo, **default**) for selecionado, o cluster, os nós do cluster, os grupos de segurança do cluster, os grupos de segurança do nó e os IPs elásticos (EIPs) dos nós criados automaticamente serão criados neste projeto empresarial. Depois que um cluster é criado, é recomendável não modificar os projetos empresariais de nós, grupos de segurança de cluster e grupos de segurança de nó no cluster.

  Os projetos empresariais facilitam o gerenciamento em nível de projeto e o agrupamento de recursos e usuários em nuvem. Para obter mais informações, consulte **Enterprise Management**.
- **Cluster Version**: selecione a versão do Kubernetes usada pelo cluster.
- **Cluster Scale**: selecione o número máximo de nós que podem ser gerenciados pelo cluster. O cluster recém-criado só suporta expansão. Para obter detalhes, consulte **Alteração de escala do cluster**.
- HA: modo de distribuição dos nós principais. Por padrão, os nós principais são distribuídos aleatoriamente em diferentes AZs para melhorar os recursos de DR.

  Você também pode expandir configurações avançadas e personalizar o modo de distribuição do nó principal. Os seguintes modos são suportados:
  - **Random**: os nós principais são criados em diferentes AZs para DR.
  - **Custom**: você pode determinar a localização de cada nó mestre.
    - **Host**: Os nós principais são criados em hosts diferentes na mesma AZ.
    - **Custom**: você pode determinar a localização de cada nó mestre.

Network Settings

As configurações de rede do cluster abrangem nós, contêineres e serviços. Para obter detalhes sobre os modelos de rede de cluster e de contêiner, consulte **Visão geral**.

- Modelo da rede: os clusters do CCE suportam **rede da VPC** e **rede de túnel**. Os clusters do CCE Turbo suportam **Cloud Native Network 2.0** . Para mais detalhes, consulte **Visão geral**.

- **VPC**: selecione a VPC à qual o cluster pertence. Se nenhuma VPC estiver disponível, clique em **Create VPC** para criar uma. O valor não pode ser alterado após a criação.

  NOTA

  – A pilha dupla IPv4/IPv6 está disponível para os clusters do CCE (v1.15 e posteriores) que usam o modelo de rede de túnel de contêiner e mais tarde estará disponível para clusters de v1.23.
  – Os conjuntos do CCE Turbo de v1.23.8-r0, v1.25.3-r0 e de versões mais posteriores apoiam a pilha dupla IPv4/IPv6.

  Para obter detalhes, consulte **Criação de um cluster IPv4/IPv6 de pilha dupla no CCE**.
- **Master Node Subnet**: selecione a sub-rede em que o nó principal é implementado. Se nenhuma sub-rede estiver disponível, clique em **Create Subnet** para criar uma. A sub-rede não pode ser alterada após a criação.
- **Container CIDR Block** (cluster do CCE): especifique o bloco CIDR usado pelos contêineres, que determina o número máximo de contêineres no cluster. Vários blocos CIDR de contêiner podem ser adicionados ao modelo de rede da VPC após a criação de um cluster. Para obter detalhes, consulte **Adição de um bloco CIDR de contêiner para um cluster**.
- **Default Pod Subnet** (cluster do CCE Turbo): selecione a sub-rede onde o contêiner está localizado. Se nenhuma sub-rede estiver disponível, clique em **Create Subnet**. A sub-rede do pod determina o número máximo de contêineres no cluster. Você pode adicionar sub-redes de pods depois de criar o cluster.
- **Service CIDR Block**: bloco CIDR para Serviços usados por contêineres no mesmo cluster para acessar uns aos outros. O valor determina o número máximo de Serviços que você pode criar. O valor não pode ser alterado após a criação.

Advanced Settings

- **Request Forwarding**: os modos IPVS e iptables são suportados. Para mais detalhes, consulte **Comparação entre iptables e IPVS**.
- **CPU Manager**: quando ativado, os núcleos de CPU serão alocados exclusivamente aos pods de carga de trabalho. Para mais detalhes, consulte **Política de CPU**.
- Resource Tag:

  você pode adicionar tags de recursos para classificar recursos.

  Você pode criar **tags predefinidas** no console do TMS. As tags predefinidas estão disponíveis para todos os recursos que suportam tags. Você pode usar tags predefinidas para melhorar a criação de tags e a eficiência da migração de recursos. Para obter detalhes, consulte **Criação de tags predefinidas**.

  Especificações da chave

  – Não pode ser vazia. Contém de 1 a 128 caracteres de byte único.
  – Não insira rótulos começando com **_sys_**, que são rótulos do sistema.
  – Pode conter letras UTF-8, dígitos, espaços e os seguintes caracteres: _. : / = + - @

    Expressão regular recomendada: ^((?!_sys_)[\\p{L}\\p{Z}\\p{N}_.:\\/=+\\-@]*)$

  Especificações do valor

  – Pode conter até 255 caracteres.
  – Pode conter letras UTF-8, dígitos, espaços e os seguintes caracteres: _. : / = + - @

    Expressão regular recomendada: ^([\p{L}\p{Z}\p{N}_.:\/=+\-@]*)$

  - O valor pode ser vazio ou nulo.
  - O valor de uma tag predefinida não pode ser vazio ou nulo.
- **Default Node Security Group**: você pode usar o grupo de segurança gerado automaticamente pelo CCE ou selecionar um existente.

  AVISO

  O grupo de segurança de nó padrão precisa permitir o acesso de determinadas portas para garantir a comunicação normal. Caso contrário, o nó não pode ser criado. Para obter detalhes, consulte **Configuração das regras do grupo de segurança do cluster do CCE**.

- **Certificate Authentication**:
  - **Default**: o modo de autenticação baseado em X509 é ativado por padrão. X509 é um formato de certificado comumente usado.
  - **Custom:** o cluster pode identificar usuários com base no cabeçalho no corpo da solicitação para autenticação.

    Carregue **certificado raiz de AC**, **certificado do cliente** e **chave privada** do certificado do cliente.

    ⚠ CUIDADO

    - Carregue um arquivo **menor que 1 MB**. O certificado de CA e o certificado do cliente podem estar no formato **.crt** ou **.cer**. A chave privada do certificado de cliente só pode ser carregada **não criptografada**.
    - O período de validade do certificado de cliente deve ser superior a cinco anos.
    - O certificado de CA carregado é usado para o proxy de autenticação e a configuração da camada de agregação de kube-apiserver. **Se o certificado for inválido, o cluster não poderá ser criado**.
    - A partir da v1.25, o Kubernetes não oferece mais suporte à autenticação de certificado gerada usando o algoritmo SHA1WithRSA ou ECDSAWithSHA1. É aconselhável usar o algoritmo SHA256.
- **Description**: a descrição não pode exceder 200 caracteres.

Passo 4 Clique em **Next: Add-on Configuration**.

Domain Name Resolution:

- **Domain Name Resolution**: o complemento **coredns** é instalado por padrão para resolver nomes de domínio e conectar-se ao servidor DNS da nuvem.

Container Storage: o complemento **everest** é instalado por padrão para fornecer armazenamento em contêiner baseado em CSI e se conectar a serviços de armazenamento em nuvem, como o EVS.

Service Logs

- Usar o ICAgent:

Um coletor de logs fornecido pelo Application Operations Management (AOM), gerando relatórios de logs para o AOM e o Log Tank Service (LTS) de acordo com as regras de coleta de logs configuradas.

Você pode coletar logs de stdout conforme necessário.

Overload Control: se ativado, as solicitações simultâneas são controladas dinamicamente com base na pressão de recursos dos nós mestres para mantê-los e o cluster disponíveis. Para mais detalhes, consulte **Controle de sobrecarga do cluster**.

Passo 5 Depois que os parâmetros forem especificados, clique em **Next: Confirm**. A lista de recursos do cluster é exibida. Confirme as informações e clique em **Submit**.

Demora cerca de 6 a 10 minutos para criar um cluster. Você pode clicar em **Back to Cluster List** para executar outras operações no cluster ou clicar em **Go to Cluster Events** para exibir os detalhes do cluster.

----Fim

Operações relacionadas

- Depois de criar um cluster, você pode usar a ferramenta de linha de comando (CLI) do Kubernetes kubectl para se conectar ao cluster. Para mais detalhes, consulte **Conexão a um cluster usando o kubectl**.
- Adicione nós ao cluster. Para mais detalhes, consulte **Criação de um nó**.
- Crie uma pilha dupla IPv4/IPv6. Para obter detalhes, consulte **Criação de um cluster IPv4/IPv6 de pilha dupla no CCE**.
- Conecte-se a vários clusters usando o kubectl. Para obter detalhes, consulte **Conexão a vários clusters usando o kubectl**.

2.2.3 Comparação entre iptables e IPVS

kube-proxy é um componente importante de um cluster do Kubernetes. Ele é usado para balanceamento de carga e encaminhamento de dados entre um serviço e seus pods de back-end.

O CCE suporta os modos de encaminhamento iptables e IPVS.

- O IPVS permite maior taxa de transferência e encaminhamento mais rápido. Esse modo se aplica a cenários em que a escala de cluster é grande ou o número de serviços é grande.
- iptables é o modo kube-proxy tradicional. Esse modo se aplica ao cenário em que o número de serviços é pequeno ou há um grande número de conexões simultâneas curtas no cliente. Quando há mais de 1.000 Serviços no cluster, pode ocorrer atraso de rede.

Restrições

- Em um cluster usando o modo de proxy IPVS, se o ingress e o Serviço usarem o mesmo balanceador de carga do ELB, o ingress não pode ser acessado dos nós e contêineres no cluster porque o kube-proxy monta o endereço do Serviço LoadBalancer na ponte ipvs-0. Essa ponte intercepta o tráfego do balanceador de carga conectado ao ingress. Use balanceadores de carga do ELB diferentes para o ingress e o Serviço.
- No modo iptables, o ClusterIP não pode ser pingado. No modo IPVS, o ClusterIP pode ser pingado.

iptables

iptables é uma função do kernel do Linux para processar e filtrar um grande número de pacotes de dados. Ele permite sequências flexíveis de regras para ser anexado a vários ganchos no pipeline de processamento de pacotes. Quando o iptables é usado, o kube-proxy implementa a NAT e o balanceamento de carga no gancho de pré-roteamento da NAT.

kube-proxy é um algoritmo O(n), no qual *n* aumenta com a escala de cluster. A escala de cluster refere-se ao número de serviços e pods de back-end.

IPVS

O IP Virtual Server (IPVS) é construído sobre o Netfilter e equilibra as cargas da camada de transporte como parte do kernel do Linux. O IPVS pode direcionar solicitações de serviços baseados em TCP ou UDP para os servidores reais e fazer com que os serviços dos servidores reais apareçam como serviços virtuais em um único endereço IP.

No modo IPVS, o kube-proxy usa o balanceamento de carga do IPVS em vez do iptables. O IPVS foi projetado para equilibrar cargas para um grande número de Serviços. Ele possui um conjunto de APIs otimizadas e usa algoritmos de pesquisa otimizados em vez de simplesmente procurar regras em uma lista.

A complexidade do processo de conexão do kube-proxy baseado em IPVS é O(1). Na maioria dos casos, a eficiência de processamento da conexão é irrelevante para a escala do cluster.

O IPVS envolve vários algoritmos de balanceamento de carga, como round-robin, menor atraso esperado, menor quantidade de conexões e vários métodos de hash. No entanto, o iptables tem apenas um algoritmo para seleção aleatória.

Comparado com o iptables, o IPVS apresenta as seguintes vantagens:

1. Oferece melhor escalabilidade e desempenho para grandes clusters.
2. Suporta algoritmos de balanceamento de carga melhores que o iptables.
3. Suporta funções, incluindo verificação de integridade do servidor e tentativas de conexão.

2.3 Conexão a um cluster

2.3.1 Conexão a um cluster usando o kubectl

Cenário

Esta seção usa um cluster do CCE como um exemplo para descrever como conectar a um cluster do CCE usando kubectl.

Permissões

Quando você acessa um cluster usando kubectl, o CCE usa **kubeconfig.json** gerado no cluster para autenticação. Esse arquivo contém informações do usuário, com base no qual o CCE determina quais recursos do Kubernetes podem ser acessados pelo kubectl. As permissões registradas em um arquivo **kubeconfig.json** variam de usuário para usuário.

Para obter detalhes sobre permissões de usuário, consulte **Permissões de cluster (baseadas no IAM) e permissões de namespace (baseadas no Kubernetes RBAC)**.

Usar o kubectl

Para se conectar a um cluster do Kubernetes a partir de um PC, você pode usar o kubectl, uma ferramenta de linha de comando do Kubernetes. Você pode fazer logon no console do CCE, clicar no nome do cluster a ser conectado e exibir o endereço de acesso e o procedimento de conexão de kubectl na página de detalhes do cluster, conforme mostrado na **Figura 2-2**.

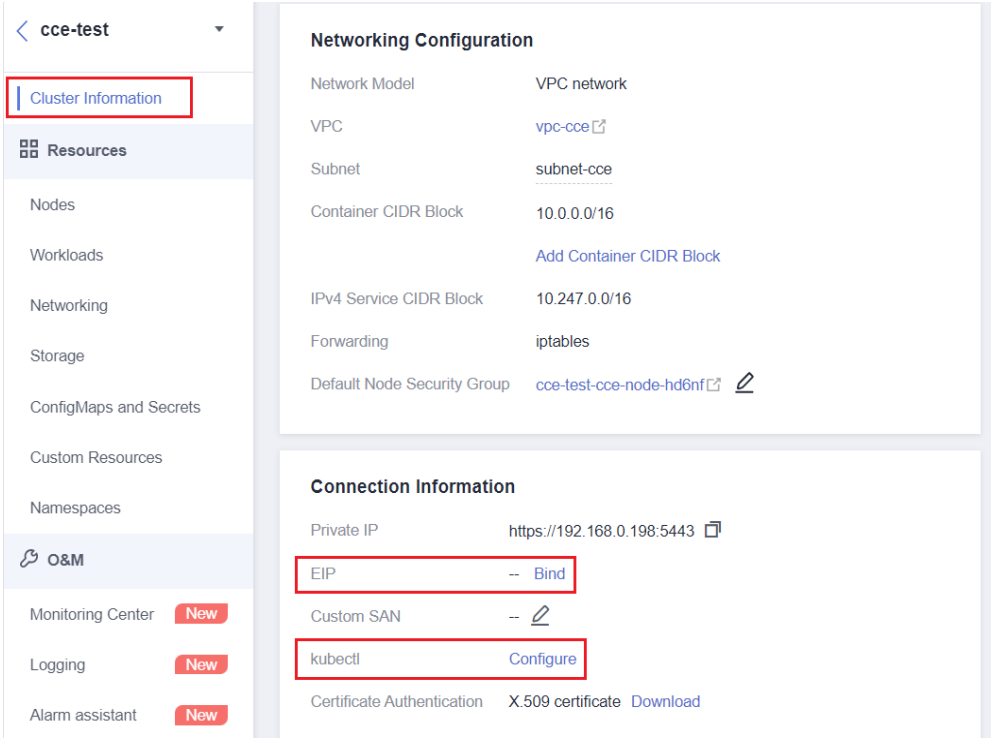
O CCE permite que você acesse um cluster por meio de uma rede privada ou uma rede pública.

- Acesso de intranet: o cliente que acessa o cluster deve estar na mesma VPC que o cluster.
- Acesso público: o cliente que acessa o cluster deve ser capaz de acessar redes públicas e o cluster foi vinculado a um IP de rede pública.

AVISO

Para vincular um IP público (EIP) ao cluster, vá para a página de detalhes do cluster e clique em **Bind** ao lado de **EIP** no painel de **Connection Information**, conforme mostrado na **Figura 2-2**. Em um cluster com um EIP vinculado, o kube-apiserver será exposto a redes públicas e poderá ser atacado. É aconselhável configurar o Advanced Anti-DDoS (AAD) para o EIP do nó onde o kube-apiserver reside.

Figura 2-2 Informações de conexão do cluster



Baixe o kubectl e o arquivo de configuração. Copie o arquivo para o seu cliente e configure o kubectl. Após a conclusão da configuração, você pode acessar seus clusters do Kubernetes. Procedimento:

Passo 1 Baixe kubectl.

Prepare um computador que possa acessar a rede pública e instale o kubectl no modo CLI. Você pode executar o comando **kubectl version** para verificar se o kubectl foi instalado. Se o kubectl tiver sido instalado, pule esta etapa.

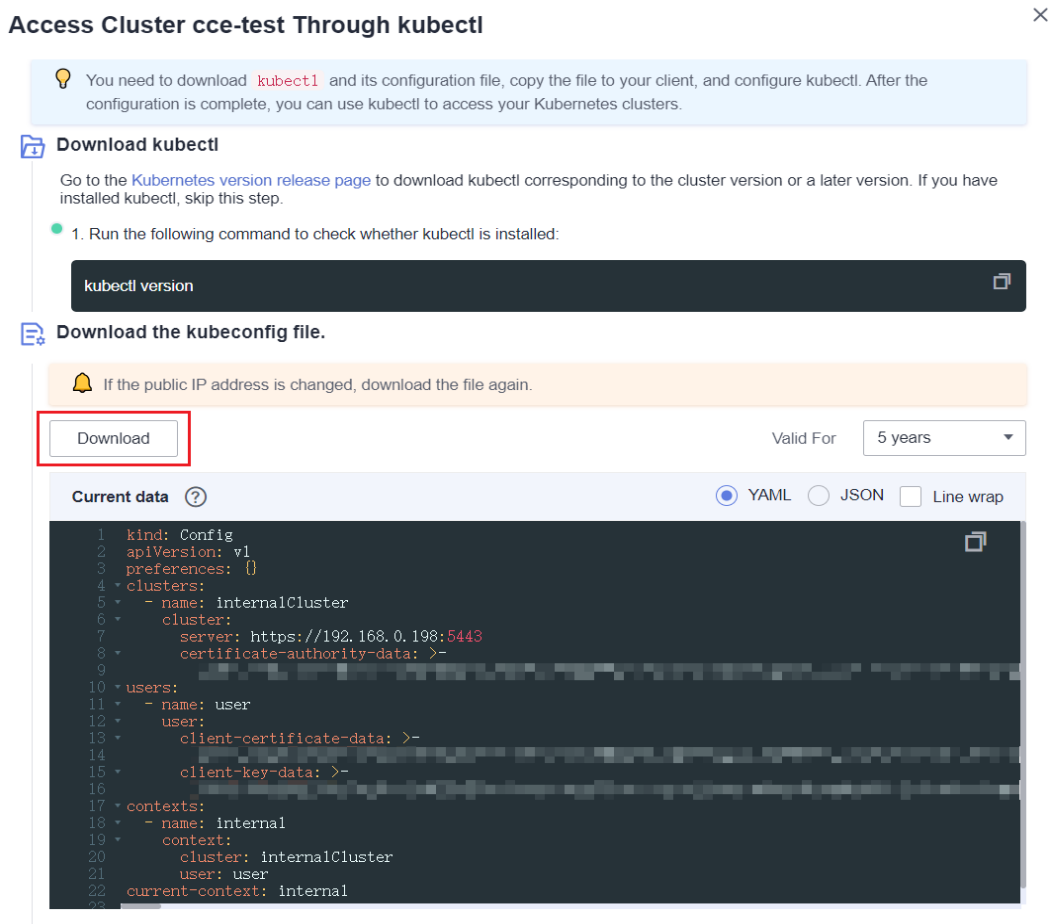
Esta seção usa o ambiente Linux como um exemplo para descrever como instalar e configurar o kubectl. Para obter detalhes, consulte **Instalação do kubectl**.

1. Faça logon no seu cliente e baixe o kubectl.
   ```
   cd /home
   curl -LO https://dl.k8s.io/release/{v1.25.0}/bin/linux/amd64/kubectl
   ```
   {v1.25.0} especifica o número da versão. Substitua-o conforme necessário.
2. Instale o kubectl.
   ```
   chmod +x kubectl
   mv -f kubectl /usr/local/bin
   ```

Passo 2 Obtenha o arquivo de configuração de kubectl (kubeconfig).

No painel de **Connection Information** na página de detalhes do cluster, clique em **Configure** ao lado de **kubectl**. Na janela exibida, baixe o arquivo de configuração.

Figura 2-3 Baixar arquivo de configuração



NOTA

- O arquivo de configuração de kubectl **kubeconfig.json** é usado para autenticação de cluster. Se o arquivo for vazado, seus clusters podem ser atacados.
- Por padrão, a autenticação bidirecional é desabilitada para nomes de domínio no cluster atual. Você pode executar o comando **kubectl config use-context externalTLSVerify** para habilitar a autenticação bidirecional. Para obter detalhes, consulte **Autenticação bidirecional para nomes de domínio**. Para um cluster vinculado a um EIP, se a autenticação falhar (x509: certificate is valid) quando a autenticação bidirecional for usada, vincule o EIP novamente e faça o download do **kubeconfig.json** novamente.
- As permissões do Kubernetes atribuídas pelo arquivo de configuração baixado pelos usuários do IAM são as mesmas atribuídas aos usuários do IAM no console do CCE.
- Se a variável de ambiente KUBECONFIG estiver configurada no sistema operacional Linux, o kubectl carregará preferencialmente a variável de ambiente KUBECONFIG em vez de **$home/.kube/config**.

Passo 3 Configure o kubectl.

Configure o kubectl (um sistema operacional Linux é usado).

1. Faça logon no seu cliente e copie o arquivo de configuração kubeconfig.json baixado em **Passo 2** para o diretório **/home** no seu cliente.
2. Configure o arquivo de autenticação kubectl.
   ```
   cd /home
   mkdir -p $HOME/.kube
   mv -f kubeconfig.json $HOME/.kube/config
   ```
3. Alterne o modo de acesso kubectl com base em cenários de serviço.
   - Execute este comando para habilitar o acesso dentro da VPC:
     ```
     kubectl config use-context internal
     ```
   - Execute este comando para habilitar o acesso público (EIP necessário):
     ```
     kubectl config use-context external
     ```
   - Execute este comando para habilitar o acesso público e a autenticação bidirecional (EIP necessário):
     ```
     kubectl config use-context externalTLSVerify
     ```
     Para obter detalhes sobre a autenticação bidirecional do cluster, consulte **Autenticação bidirecional para nomes de domínio**.

----Fim

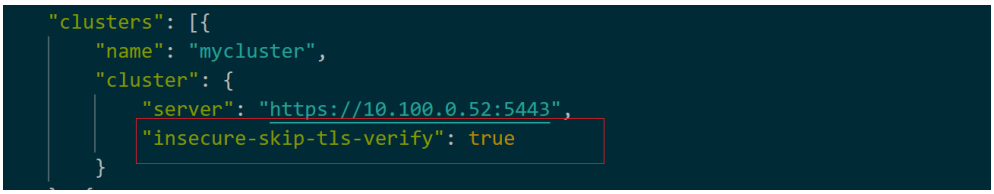
Autenticação bidirecional para nomes de domínio

O CCE suporta autenticação bidirecional para nomes de domínio.

- A autenticação bidirecional está desabilitada para nomes de domínio por padrão. Você pode executar o comando **kubectl config use-context externalTLSVerify** para alternar para o contexto de externalTLSVerify para ativá-lo.
- Quando um EIP estiver vinculado ou não vinculado a um cluster, ou um nome de domínio personalizado for configurado ou atualizado, o certificado do servidor de cluster será adicionado ao endereço de acesso de cluster mais recente (incluindo o EIP vinculado ao cluster e todos os nomes de domínio personalizados configurados para o cluster).
- A sincronização assíncrona do cluster demora cerca de 5 a 10 minutos. Você pode exibir o resultado da sincronização em **Synchronize Certificate** de **Operation Records**.

- Para um cluster vinculado a um EIP, se a autenticação falhar (x509: certificate is valid) quando a autenticação bidirecional for usada, vincule o EIP novamente e faça o download do **kubeconfig.json** novamente.
- Se a autenticação bidirecional do nome de domínio não for suportada, **kubeconfig.json** contém o campo **"insecure-skip-tls-verify": true**, como mostrado em **Figura 2-4**. Para usar a autenticação bidirecional, você pode baixar o arquivo **kubeconfig.json** novamente e ativar a autenticação bidirecional para os nomes de domínio.

Figura 2-4 Autenticação bidirecional desabilitada para nomes de domínio



Perguntas frequentes

- **Erro do servidor proibido**

  Quando você usa o kubectl para criar ou consultar recursos do Kubernetes, a seguinte saída é retornada:

  ```
  # kubectl get deploy Error from server (Forbidden): deployments.apps is
  forbidden: User "0c97ac3cb280f4d91fa7c0096739e1f8" cannot list resource
  "deployments" in API group "apps" in the namespace "default"
  ```

  A causa é que o usuário não tem permissões para operar os recursos do Kubernetes. Para obter detalhes sobre como atribuir permissões, consulte **Permissões de namespace (com base no RBAC do Kubernetes)**.

- **A conexão com o servidor localhost:8080 foi recusada**

  Quando você usa o kubectl para criar ou consultar recursos do Kubernetes, a seguinte saída é retornada:

  ```
  The connection to the server localhost:8080 was refused - did you specify the
  right host or port?
  ```

  A causa é que a autenticação de cluster não está configurada para o cliente de kubectl. Para mais detalhes, consulte **Passo 3**.

Operações relacionadas

- **Conexão a vários clusters usando o kubectl**.
- **Configuração de kubeconfig para gerenciamento refinado dos recursos do cluster**

2.3.2 Conexão a um cluster usando o CloudShell

Cenário

Esta seção usa um cluster do CCE como um exemplo para descrever como conectar a um cluster do CCE usando CloudShell.

Permissões

Ao usar o kubectl no CloudShell, as permissões do kubectl são determinadas pelo usuário que efetua o logon.

Usar o CloudShell

O CloudShell é um shell da Web usado para gerenciar e manter os recursos da nuvem. O CCE permite usar o CloudShell para se conectar a clusters e usar o kubectl no CloudShell para acessar clusters (clicando no ícone da ferramenta de linha de comando em **Figura 2-5**).

NOTA

- O certificado de kubectl no CloudShell é válido por um dia. Você pode redefinir o período de validade acessando o CloudShell no console do CCE.
- O CloudShell é implementado com base no VPCEP. Para usar o kubectl para acessar um cluster, configure o grupo de segurança (*Cluster name*-cce-control-*Random number*) no nó principal do cluster para permitir que os seguintes blocos CIDR acessem a porta 5443. Por padrão, a porta 5443 permite acesso de todos os blocos CIDR. Se você tiver grupos de segurança reforçados e nenhum cluster não puder ser acessado no CloudShell, verifique se a porta 5443 permite acesso de **198.19.0.0/16**.
- O CloudShell pode ser usado somente após a instalação do CoreDNS em um cluster.
- Atualmente, você pode usar CloudShell para fazer logon em contêineres apenas nas regiões CN North-Beijing1, CN North-Beijing4, CN East-Shanghai1, CN East-Shanghai2, CN South-Guangzhou e CN North-Ulanqab1.

Figura 2-5 CloudShell

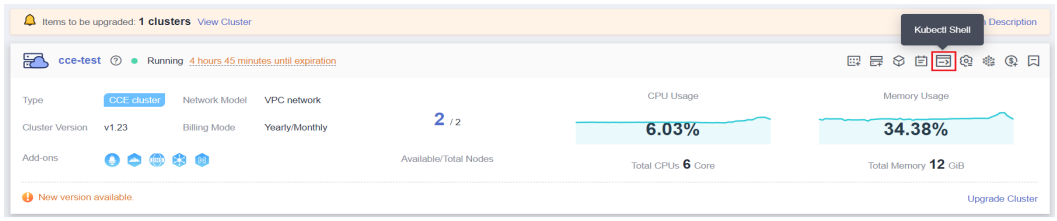
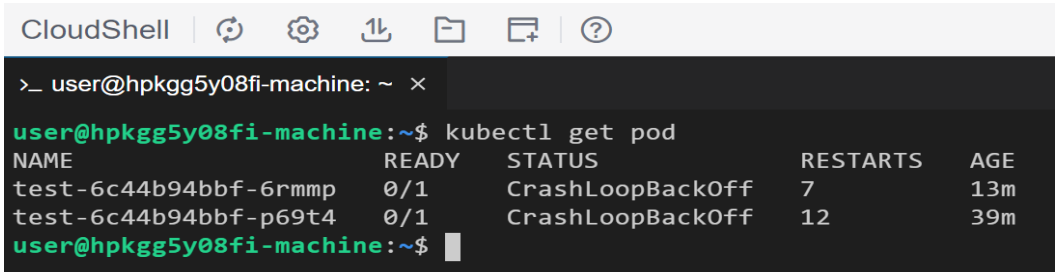


Figura 2-6 Usar o kubectl no CloudShell



2.3.3 Conexão a um cluster usando um certificado X.509

Cenário

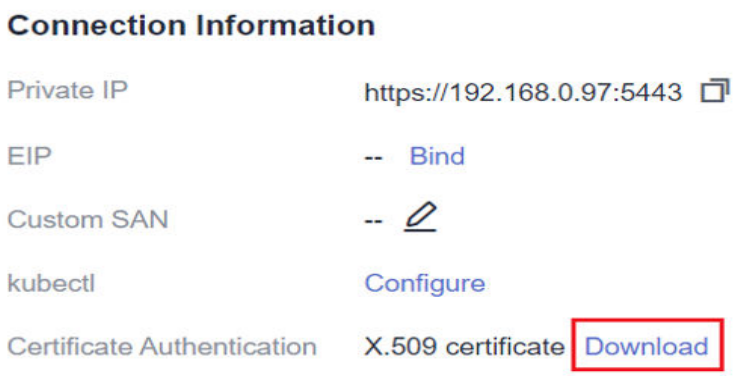
Esta seção descreve como obter o certificado de cluster do console e usá-lo para acessar os clusters do Kubernetes.

Procedimento

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster.

Passo 2 Escolha **Cluster Information** no painel de navegação e clique em **Download** ao lado de **Authentication Mode** na área **Connection Information**.

Figura 2-7 Baixar um certificado de cluster



Passo 3 Na caixa de diálogo **Download X.509 Certificate** exibida, selecione o tempo de expiração do certificado e baixe o certificado X.509 do cluster conforme solicitado.

AVISO

- O certificado baixado contém três arquivos: **client.key**, **client.crt** e **ca.crt**. Mantenha estes arquivos em segurança.
- Certificados não são necessários para acesso mútuo entre contêineres em um cluster.

Passo 4 Chame APIs do Kubernetes nativo usando o certificado de cluster.

Por exemplo, execute o comando **curl** para chamar uma API para visualizar as informações do pod. Nas informações a seguir, *192.168.0.18:5443* indica o endereço IP do servidor da API no cluster.

```
curl --cacert ./ca.crt --cert ./client.crt --key ./client.key  https://
192.168.0.18:5443/api/v1/namespaces/default/pods/
```

Para obter mais APIs de cluster, consulte **APIs do Kubernetes**.

----Fim

2.3.4 Acesso a um cluster usando um nome de domínio personalizado

Cenário

Um **Nome alternativo do assunto (SAN)** pode ser conectado a um certificado de servidor de cluster. Um SAN é normalmente usado pelo cliente para verificar a validade do servidor em handshakes TLS. Especificamente, a verificação de validade inclui se o certificado de servidor é emitido por uma AC confiável pelo cliente e se o SAN no certificado corresponde ao endereço IP ou nome de domínio do DNS que o cliente realmente acessa.

Se o cliente não conseguir acessar diretamente o IP privado ou EIP do cluster, você poderá assinar o endereço IP ou o nome de domínio do DNS que pode ser acessado diretamente pelo

cliente no certificado do servidor de cluster para habilitar a autenticação bidirecional no cliente, o que melhora a segurança. Casos de uso típicos incluem acesso à DNAT e acesso ao nome de domínio.

Cenários típicos de acesso a nomes de domínio:

- Adicione o mapeamento de nome de domínio de resposta ao especificar o endereço de nome de domínio do DNS na configuração de nome de domínio do host no cliente ou ao configurar **/etc/hosts** no host do cliente.
- Use o acesso de nome de domínio na intranet. O DNS permite configurar mapeamentos entre EIPs de cluster e nomes de domínio personalizados. Depois que um EIP for atualizado, você poderá continuar usando a autenticação bidirecional e o nome de domínio para acessar o cluster sem baixar o arquivo **kubeconfig.json** novamente.
- Adicione registros A em um servidor DNS autoconstruído.

Restrições

Esse recurso está disponível apenas para clusters de v1.19 e posteriores.

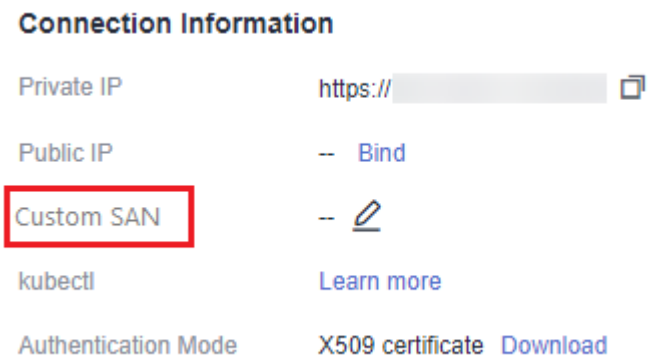
Personalizar um SAN

Passo 1 Efetue logon no console do CCE.

Passo 2 Clique no cluster de destino na lista de clusters para ir para a página de detalhes do cluster.

Passo 3 Na área **Connection Information**, clique em ao lado de **Custom SAN**. Na caixa de diálogo exibida, adicione o endereço IP ou o nome do domínio e clique em **Save**.

Figura 2-8 Personalizar SAN



NOTA

1. Esta operação irá reiniciar o kube-apiserver e atualizar o arquivo **kubeconfig.json** por um curto período de tempo. Não execute operações no cluster durante esse período.
2. Um máximo de 128 nomes de domínio ou endereços IP, separados por vírgulas (,), são permitidos.
3. Se um nome de domínio personalizado precisar ser vinculado a um EIP, verifique se um EIP foi configurado.

----Fim

2.4 Atualização de um cluster

2.4.1 Visão geral de atualização

CCE foi aprovado pelo Certified Kubernetes Conformance Program e é uma oferta certificada do Kubernetes. Para habilitar a interoperabilidade de uma instalação do Kubernetes para a próxima, você deve atualizar seus clusters do Kubernetes antes que o período de manutenção termine.

Depois que a versão mais recente do Kubernetes estiver disponível no CCE, o CCE descreverá as alterações nessa versão.

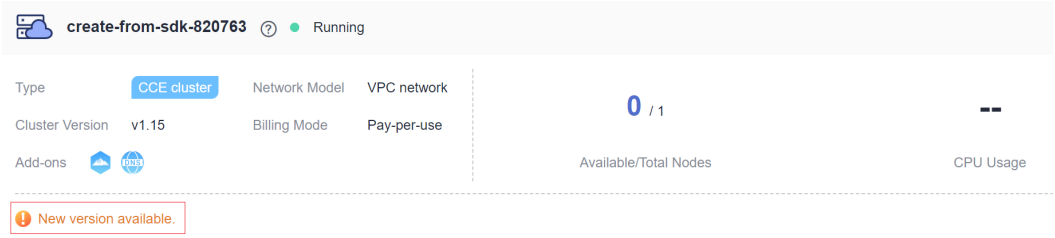
Você pode usar o console do CCE para atualizar a versão do Kubernetes de um cluster.

Uma tag de upgrade será exibida na exibição de placa de cluster se houver uma nova versão para o cluster a ser atualizado.

Como verificar:

Faça logon no console do CCE e verifique se a mensagem "New version available" é exibida no canto inferior esquerdo do cluster. Se sim, o cluster pode ser atualizado. Veja as notas de lançamento para a versão mais recente. Para mais detalhes, consulte **Observações de lançamento para versões de cluster do CCE**. Se nenhuma mensagem for exibida, o cluster é da versão mais recente.

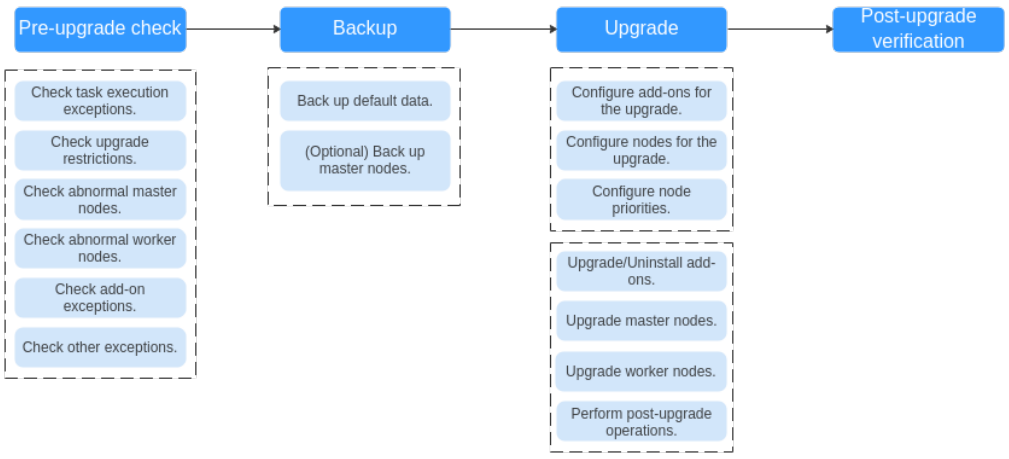
Figura 2-9 Cluster com a tag de atualização



Processo de atualização do cluster

O processo de atualização de cluster envolve verificação pré-atualização, backup, atualização e verificação pós-atualização.

Figura 2-10 Processo de atualização de um cluster



Depois de determinar a versão de destino do cluster, leia atentamente as **precauções** e evite a incompatibilidade de função durante a atualização.

1. **Verificação pré-atualização**

   Antes de uma atualização de cluster, o CCE verifica a compatibilidade de nós, complementos e cargas de trabalho no cluster para reduzir a probabilidade de falhas de atualização para a melhor extensão. Se alguma exceção for detectada, retifique a falha conforme solicitado no console.

2. **Backup**

   Durante a atualização, o backup dos dados do cluster é feito por padrão. Você também pode fazer backup de nós principais inteiros conforme necessário. O Cloud Backup and Recovery (CBR) será usado para backup de nó completo. Demora cerca de 20 minutos para fazer backup de um nó.

3. **Atualização**

   Durante a atualização, configure os parâmetros de atualização, como a etapa para atualização de complementos ou atualização de rolamento de nó. Depois que os parâmetros de atualização são configurados, os complementos e nós serão atualizados um por um.

4. **Verificação pós-atualização**

   Após a atualização, verifique manualmente os serviços e assegure-se de que os serviços não sejam interrompidos pela atualização.

Atualização do cluster

A tabela a seguir descreve a versão de destino para a qual cada versão do cluster pode ser atualizada e os modos de atualização suportados.

Tabela 2-9 Atualização do cluster

Versão atual	Versão de destino	Modo de atualização
v1.23	v1.25	Atualização in-loco

Versão atual	Versão de destino	Modo de atualização
v1.21	v1.25 v1.23	Atualização in-loco
v1.19	v1.23 v1.21	Atualização in-loco
v1.17	v1.19	Atualização in-loco
v1.15	v1.19	Atualização in-loco
v1.13	v1.15	Atualização contínua
v1.11 v1.9 v1.7	Última versão que pode ser criada no console	Migração

Modo de atualização

Diferentes modos de atualização têm diferentes vantagens e desvantagens.

Tabela 2-10 Diferenças entre os modos de atualização e suas vantagens e desvantagens

Modo de atualização	Método	Vantagens	Desvantagens
Atualização in-loco	Os componentes do Kubernetes, os componentes de rede e os componentes de gerenciamento do CCE são atualizados no nó. Durante a atualização, os pods de serviço e as redes não são afetados. O rótulo **SchedulingDisabled** será adicionado a todos os nós existentes. Após a conclusão da atualização, você pode usar corretamente os nós existentes.	Você não precisa migrar serviços, garantindo a continuidade do serviço.	A atualização in-loco não atualiza o sistema operacional de um nó. Se você deseja atualizar o sistema operacional, limpe os dados correspondentes do nó após a conclusão da atualização do nó e redefina o nó para atualizar o sistema operacional para uma nova versão.

Modo de atualização	Método	Vantagens	Desvantagens
Atualização contínua	Somente os componentes do Kubernetes e determinados componentes de rede são atualizados no nó. O rótulo **SchedulingDisabled** será adicionado a todos os nós existentes para garantir que as aplicações em execução não sejam afetados. **AVISO** ● **Após a conclusão da atualização, crie manualmente os nós e libere gradualmente os nós antigos**, migrando assim suas aplicações para os novos nós. Neste modo, você pode controlar o processo de atualização.	Os serviços não são interrompidos.	● **Após a conclusão da atualização, crie manualmente os nós e libere gradualmente os nós antigos.** Os novos nós são cobrados adicionalmente. Depois que os serviços são migrados para os novos nós, os nós anteriores podem ser excluídos. ● Após a conclusão da atualização contínua, se você quiser continuar a atualização para uma versão posterior, redefina os nós anteriores primeiro. Caso contrário, a verificação pré-atualização não pode ser aprovada. Os serviços podem ser interrompidos durante a atualização.

2.4.2 Antes de começar

Antes da atualização, você pode verificar se o cluster pode ser atualizado e quais versões estão disponíveis no console do CCE. Para mais detalhes, consulte **Visão geral de atualização**.

Precauções

Antes de atualizar um cluster, preste atenção aos seguintes pontos:

- **A atualização de um cluster não pode ser revertida. Realize uma atualização no momento adequado para minimizar o impacto em seus serviços.** Para garantir a segurança dos dados, faça backup de seus dados antes de uma atualização.
- Antes de atualizar um cluster, **garanta que nenhuma operação de alto risco seja executada no cluster**. Caso contrário, a atualização de cluster poderá falhar ou a configuração poderá ser perdida após a atualização. As operações comuns de alto risco incluem a modificação local das configurações de nós do cluster e a modificação das configurações dos ouvintes gerenciados pelo CCE no console do ELB. Em vez disso, modifique configurações no console do CCE para que as modificações possam ser herdadas automaticamente durante a atualização.
- Antes de atualizar um cluster, verifique se o cluster está funcionando corretamente. Se o cluster não estiver funcionando corretamente, corrija a falha. Se a falha persistir, **envie um tíquete de serviço** para obter assistência.
- Antes de atualizar um cluster, saiba mais sobre os recursos e as diferenças de cada versão de cluster em **Observações de versão do Kubernetes** para evitar exceções devido ao uso de uma versão de cluster incompatível. Por exemplo, verifique se alguma API preterida na versão de destino é usada no cluster. Caso contrário, chamar as APIs pode falhar após a atualização. Para mais detalhes, consulte **APIs preteridas**.

Durante uma atualização de cluster, preste atenção aos seguintes pontos que podem afetar seus serviços:

- Durante uma atualização de cluster, não execute nenhuma operação no cluster. Não **interrompa, reinicie ou exclua nós** durante a atualização do cluster. Caso contrário, a atualização falhará.
- Durante uma atualização de cluster, as cargas de trabalho em execução não serão interrompidas, mas o acesso ao servidor da API será temporariamente interrompido.
- Durante uma atualização de cluster, a mancha **node.kubernetes.io/upgrade** (equivalente a **NoSchedule**) será adicionada aos nós no cluster. A mancha será removida depois que o cluster for atualizado. Não adicione manchas com o mesmo nome de chave em um nó. Mesmo que as manchas tenham efeitos diferentes, elas podem ser excluídas pelo sistema por engano após a atualização.

Restrições

- Os clusters do CCE e os clusters do CCE Turbo com nós de VM podem ser atualizados.
- Se houver nós criados usando uma imagem privada, o cluster não poderá ser atualizado.
- Depois que o cluster for atualizado, se a vulnerabilidade em contêiner do mecanismo de contêiner for corrigida nas **Observações de versão do Kubernetes**, reinicie manualmente o contêiner para que a atualização entre em vigor. O mesmo se aplica aos pods existentes.
- Se você montar o arquivo **docker.sock** em um nó em um pod usando o modo hostPath, ou seja, o Docker no cenário Docker, o Docker será reiniciado durante a atualização, mas o arquivo **docker.sock** não será alterado. Como resultado, seus serviços podem não funcionar corretamente. É aconselhável montar o arquivo **docker.sock** montando o diretório.
- Quando os clusters que usam o modelo de rede de túnel são atualizados para v1.19.16-r4, v1.21.7-r0, v1.23.5-r0, v1.25.1-r0 ou posterior, a regra SNAT cujo endereço de destino é o bloco CIDR do contêiner, mas o endereço de origem não é o bloco CIDR do contêiner será removido. Se você configurou as rotas da VPC para acessar diretamente todos os pods fora do cluster, somente os pods nos nós correspondentes poderão ser acessados diretamente após a atualização.

APIs preteridas

Com a evolução das APIs do Kubernetes, as APIs são periodicamente reorganizadas ou atualizadas, e as APIs anteriores são preteridas e finalmente excluídas. As tabelas a seguir listam as APIs preteridas em cada versão da comunidade do Kubernetes. Para obter detalhes sobre mais APIs preteridas, consulte **Guia de migração de APIs preteridas**.

- **APIs preteridas no Kubernetes v1.25**
- **APIs preteridas no Kubernetes v1.22**
- **APIs preteridas no Kubernetes v1.16**

NOTA

Quando uma API é preterida, os recursos existentes não são afetados. No entanto, quando você criar ou editar os recursos, a versão da API será interceptada.

Tabela 2-11 APIs preteridas no Kubernetes v1.25

Nome do recurso	Versão da API preterida	Versão da API substituta	Descrição de mudança
CronJob	batch/v1beta1	batch/v1 (Esta API está disponível desde a v1.21.)	Nenhuma
EndpointSlice	discovery.k8s.io/v1beta1	discovery.k8s.io/v1 (Esta API está disponível desde a v1.21.)	Preste atenção às seguintes mudanças: ● Em cada ponto de extremidade, o campo **topology["kubernetes.io/hostname"]** foi preterido. Substitua-o pelo campo **nodeName**. ● Em cada ponto de extremidade, o campo **topology["kubernetes.io/zone"]** foi preterido. Substitua-o pelo campo **zone**. ● O campo **topology** é substituído por **deprecatedTopology** e não pode ser gravado em v1.

Nome do recurso	Versão da API preterida	Versão da API substituta	Descrição de mudança
Event	events.k8s.io/v1beta1	events.k8s.io/v1 (Esta API está disponível desde a v1.19.)	Preste atenção às seguintes mudanças: ● O campo **type** só pode ser definido como **Normal** ou **Warning**. ● O campo **involvedObject** é renomeado a **regarding**. ● Os campos **action**, **reason**, **reportingController** e **reportingInstance** são obrigatórios para criar um novo evento **events.k8s.io/v1**. ● Use **eventTime** em vez do campo **firstTimestamp** obsoleto (este campo foi renomeado como **deprecatedFirstTimestamp** e não tem permissão para aparecer no novo objeto de evento **events.k8s.io/v1**). ● Use **series.lastObservedTime** em vez do campo obsoleto **lastTimestamp** (este campo foi renomeado como **deprecatedLastTimestamp** e não pode aparecer no novo objeto de evento **events.k8s.io/v1**). ● Use **series.count** em vez do campo obsoleto **count** (este campo foi renomeado como **deprecatedCount** e não pode aparecer no novo objeto de evento **events.k8s.io/v1**). ● Use **reportingController** em vez do campo obsoleto **source.component** (esse campo foi renomeado como **deprecatedSource.component** e não tem permissão para aparecer no novo objeto de evento **events.k8s.io/v1**). ● Use **reportingInstance** em vez do campo obsoleto **source.host** (este campo foi renomeado como **deprecatedSource.host**

Nome do recurso	Versão da API preterida	Versão da API substituta	Descrição de mudança
			e não pode aparecer no novo objeto de evento **events.k8s.io/v1**).
HorizontalPodAutoscaler	autoscaling/v2beta1	autoscaling/v2 (Esta API está disponível desde a v1.23.)	Nenhuma
PodDisruptionBudget	policy/v1beta1	policy/v1 (Esta API está disponível desde a v1.21.)	Se **spec.selector** for definido como nulo ({}) em **PodDisruptionBudget** de **policy/v1**, todos os pods no namespace serão selecionados. (Em **policy/v1beta1**, um **spec.selector** vazio significa que nenhum pod será selecionado.) Se **spec.selector** não for especificado, pod será selecionado em nenhuma das versões da API.
PodSecurityPolicy	policy/v1beta1	Nenhuma	Desde a v1.25, o recurso PodSecurityPolicy não fornece mais APIs da versão **policy/v1beta1**, e o controlador de acesso de PodSecurityPolicy é excluído. Substitua-o por **Configuração de admissão de segurança do pod**.
RuntimeClass	node.k8s.io/v1beta1	node.k8s.io/v1 (Esta API está disponível desde a v1.20.)	Nenhuma

Tabela 2-12 APIs preteridas no Kubernetes v1.22

Nome do recurso	Versão da API preterida	Versão da API substituta	Descrição de mudança
MutatingWebhookConfiguration ValidatingWebhookConfiguration	admissionregistration.k8s.io/v1beta1	admissionregistration.k8s.io/v1 (Esta API está disponível desde a v1.16.)	• O valor padrão de **webhooks[*].failurePolicy** é alterado de **Ignore** para **Fail** na v1. • O valor padrão de **webhooks[*].matchPolicy** é alterado de **Exact** para **Equivalent** na v1. • O valor padrão de **webhooks[*].timeoutSeconds** é alterado de **30s** para **10s** na v1. • O valor padrão de **webhooks[*].sideEffects** é excluído e esse campo deve ser especificado. Na v1, o valor só pode ser **None** ou **NoneOnDryRun**. • O valor padrão de **webhooks[*].admissionReviewVersions** é excluído. Na v1, este campo deve ser especificado. (Há suporte para **AdmissionReview** v1 e v1beta1.) • **webhooks[*].name** deve ser único na lista de objetos criados por meio de **admissionregistration.k8s.io/v1**.

Nome do recurso	Versão da API preterida	Versão da API substituta	Descrição de mudança
CustomResourceDefinition	apiextensions.k8s.io/v1beta1	apiextensions/v1 (Esta API está disponível desde a v1.16.)	• O valor padrão de **spec.scope** não é mais **Namespaced**. Este campo deve ser explicitamente especificado. • **spec.version** é excluído da v1. Use **spec.versions** em vez disso. • **spec.validation** é excluído da v1. Use **spec.versions[*].schema** em vez disso. • **spec.subresources** é excluído da v1. Use **spec.versions[*].subresources** em vez disso. • **spec.additionalPrinterColumns** é excluído da v1. Use **spec.versions[*].additionalPrinterColumns** em vez disso. • **spec.conversion.webhookClientConfig** é movido para **spec.conversion.webhook.clientConfig** na v1. • **spec.conversion.conversionReviewVersions** é movido para **spec.conversion.webhook.conversionReviewVersions** na v1. • **spec.versions[*].schema.openAPIV3Schema** torna-se um campo obrigatório quando o objeto **CustomResourceDefinition** da versão v1 é criado, e seu valor deve ser um **esquema estrutural**. • **spec.preserveUnknownFields: true** não pode ser especificado quando o objeto **CustomResourceDefinition** da versão v1 é criado. Essa configuração deve ser especificada usando **x-kubernetes-preserve-unknown-fields: true** na definição do esquema. • Na v1, o campo **JSONPath** na entrada de **additionalPrinter-**

Nome do recurso	Versão da API preterida	Versão da API substituta	Descrição de mudança
			Columns é renomeado como **jsonPath** (patch **#66531**).
APIService	apiregistration/v1beta1	apiregistration.k8s.io/v1 (Esta API está disponível desde a v1.10.)	Nenhuma
TokenReview	authentication.k8s.io/v1beta1	authentication.k8s.io/v1 (Esta API está disponível desde a v1.6.)	Nenhuma
LocalSubjectAccessReview SelfSubjectAccessReview SubjectAccessReview SelfSubjectRulesReview	authorization.k8s.io/v1beta1	authorization.k8s.io/v1 (Esta API está disponível desde a v1.16.)	**spec.group** foi renomeado **spec.groups** na v1 (patch **#32709**).

Nome do recurso	Versão da API preterida	Versão da API substituta	Descrição de mudança
CertificateSigningRequest	certificates.k8s.io/v1beta1	certificates.k8s.io/v1 (Esta API está disponível desde a v1.19.)	Preste atenção às seguintes alterações em **certificates.k8s.io/v1**: ● Para um cliente de API que solicita um certificado: – **spec.signerName** se torna um campo obrigatório (consulte **Signatários conhecidos do Kubernetes**). Além disso, a API **certificates.k8s.io/v1** não pode ser usada para criar solicitações cujo signatário seja **kubernetes.io/legacy-unknown**. – **spec.usages** agora se torna um campo obrigatório, que não pode conter valores de cadeia duplicados e pode conter apenas strings de uso conhecidas. ● Para um cliente de API que precisa aprovar ou assinar um certificado: – **status.conditions** não pode conter tipos duplicados. – O campo **status.conditions[*].status** agora é obrigatório. – O **status.certificate** deve ser codificado pelo PEM e pode conter apenas o bloco de dados **CERTIFICATE**.
Lease	coordination.k8s.io/v1beta1	coordination.k8s.io/v1 (Esta API está disponível desde a v1.14.)	Nenhuma

Nome do recurso	Versão da API preterida	Versão da API substituta	Descrição de mudança
Ingress	networking.k8s.io/v1beta1 extensions/v1beta1	networking.k8s.io/v1 (Esta API está disponível desde a v1.19.)	● O campo **spec.backend** é renomeado **spec.defaultBackend**. ● O campo **serviceName** do back-end é renomeado **service.name**. ● O campo **servicePort** de back-end representado por um número é renomeado **service.port.number**. ● O campo **servicePort** de back-end representado por uma cadeia é renomeado **service.port.name**. ● O campo **pathType** é obrigatório para que todos os caminhos sejam especificados. As opções são **Prefix**, **Exact** e **ImplementationSpecific**. Para corresponder ao comportamento de não definir o tipo de caminho em v1beta1, use **ImplementationSpecific**.
IngressClass	networking.k8s.io/v1beta1	networking.k8s.io/v1 (Esta API está disponível desde a v1.19.)	Nenhuma
ClusterRole ClusterRoleBinding Role RoleBinding	rbac.authorization.k8s.io/v1beta1	rbac.authorization.k8s.io/v1 (Esta API está disponível desde a v1.8.)	Nenhuma
PriorityClass	scheduling.k8s.io/v1beta1	scheduling.k8s.io/v1 (Esta API está disponível desde a v1.14.)	Nenhuma

Nome do recurso	Versão da API preterida	Versão da API substituta	Descrição de mudança
CSIDriver CSINode StorageClass VolumeAttachment	storage.k8s.io/v1beta1	storage.k8s.io/v1	● CSIDriver está disponível em **storage.k8s.io/v1** desde a v1.19. ● CSINode está disponível em **storage.k8s.io/v1** desde a v1.17. ● StorageClass está disponível em **storage.k8s.io/v1** desde a v1.6. ● VolumeAttachment está disponível em **storage.k8s.io/v1** desde a v1.13.

Tabela 2-13 APIs preteridas no Kubernetes v1.16

Nome do recurso	Versão da API preterida	Versão da API substituta	Descrição de mudança
NetworkPolicy	extensions/v1beta1	networking.k8s.io/v1 (Esta API está disponível desde a v1.8.)	Nenhuma
DaemonSet	extensions/v1beta1 apps/v1beta2	apps/v1 (Esta API está disponível desde a v1.9.)	● O campo **spec.templateGeneration** é excluído. ● **spec.selector** agora é um campo obrigatório e não pode ser alterado após a criação do objeto. O rótulo de um modelo existente pode ser usado como um seletor para migração perfeita. ● O valor padrão de **spec.updateStrategy.type** é alterado para **RollingUpdate** (o valor padrão na versão da API **extensions/v1beta1** é **OnDelete**).

Nome do recurso	Versão da API preterida	Versão da API substituta	Descrição de mudança
Deployment	extensions/v1beta1 apps/v1beta1 apps/v1beta2	apps/v1 (Esta API está disponível desde a v1.9.)	● O campo **spec.rollbackTo** é excluído. ● **spec.selector** agora é um campo obrigatório e não pode ser alterado após a criação da Implementação. O rótulo de um modelo existente pode ser usado como um seletor para migração perfeita. ● O valor padrão de **spec.progressDeadlineSeconds** é alterado para 600 segundos (o valor padrão em **extensions/v1beta1** é ilimitado). ● O valor padrão de **spec.revisionHistoryLimit** é alterado para **10**. (Na versão da API **apps/v1beta1**, o valor padrão desse campo é **2**. Na versão da API **extensions/v1beta1**, todos os registros históricos são mantidos por padrão.) ● Os valores padrão de **maxSurge** e **maxUnavailable** são alterados para **25%**. (Na versão da API **extensions/v1beta1**, esses campos têm como padrão **1**.)
StatefulSet	apps/v1beta1 apps/v1beta2	apps/v1 (Esta API está disponível desde a v1.9.)	● **spec.selector** agora é um campo obrigatório e não pode ser alterado após a criação do StatefulSet. O rótulo de um modelo existente pode ser usado como um seletor para migração perfeita. ● O valor padrão de **spec.updateStrategy.type** é alterado para **RollingUpdate** (O valor padrão na versão da API **apps/v1beta1** é **OnDelete**).

Nome do recurso	Versão da API preterida	Versão da API substituta	Descrição de mudança
ReplicaSet	extensions/v1beta1 apps/v1beta1 apps/v1beta2	apps/v1 (Esta API está disponível desde a v1.9.)	**spec.selector** agora é um campo obrigatório e não pode ser alterado após a criação do objeto. O rótulo de um modelo existente pode ser usado como um seletor para migração perfeita.
PodSecurityPolicy	extensions/v1beta1	policy/v1beta1 (Esta API está disponível desde a v1.10.)	PodSecurityPolicy para a versão da API **policy/v1beta1** será removida na v1.25.

Diferenças entre versões

Caminho de atualização	Diferença entre versões	Auto-verificação
v1.19 a v1.21	O bug de **exec probe timeouts** foi corrigido no Kubernetes 1.21. Antes que esse bug seja corrigido, a sonda exec não considera o campo **timeoutSeconds**. Em vez disso, a sonda será executada indefinidamente, mesmo após o data limite configurado. Ela irá parar até que o resultado seja retornado. Se este campo não for especificado, o valor padrão **1** será usado. Este campo entra em vigor após a atualização. Se a sonda for executada por mais de 1 segundo, a verificação de integridade da aplicação poderá falhar e a aplicação poderá ser reiniciada com frequência.	Antes do upgrade, verifique se o tempo limite está configurado corretamente para a sonda exec.

Caminho de atualização	Diferença entre versões	Auto-verificação
	O kube-apiserver do CCE 1.19 ou posterior requer que o campo Subject Alternative Names (SANs) esteja configurado para o certificado do servidor webhook. Caso contrário, kube-apiserver falha ao chamar o servidor webhook após a atualização, e os contêineres não podem ser iniciados corretamente. Causa raiz: X.509 **CommonName** é descartado em Go 1.15. kube-apiserver do CCE 1.19 é compilado usando Go 1.15. Se o seu certificado webhook não tiver SANs, kube-apiserver não processa o campo **CommonName** do certificado X.509 como o nome de host por padrão. Como resultado, a autenticação falha.	Antes da atualização, verifique se o campo SAN está configurado no certificado do servidor webhook. ● Se você não tem seu próprio servidor webhook, você pode pular esta verificação. ● Se o campo não estiver definido, é aconselhável usar o campo SAN para especificar o endereço IP e o nome de domínio suportados pelo certificado.
v1.15 a v1.19	O plano de controle de clusters do CCE de v1.19 é incompatível com kubelet v1.15. Se um nó não for atualizado ou o nó a ser atualizado for reiniciado após o nó principal ser atualizado com sucesso, há uma alta probabilidade de que o nó esteja no status **NotReady**. Isso ocorre porque o nó falhou ao ser atualizado reinicia o kubelet e aciona o registro do nó. Em clusters de v1.15, as tags de registro padrão (**failure-domain.beta.kubernetes.io/is-baremetal** e **kubernetes.io/availablezone**) são consideradas tags inválidas pelo cluster da v1.19. As tags válidas nos clusters da v1.19 são **node.kubernetes.io/baremetal** e **failure-domain.beta.kubernetes.io/zone**.	1. Em casos normais, esse cenário não é acionado. 2. Depois que o nó principal for atualizado, não suspenda a atualização para que o nó possa ser atualizado rapidamente. 3. Se um nó não for atualizado e não puder ser restaurado, expulse aplicações no nó o mais rápido possível. Entre em contato com o suporte técnico e pule a atualização do nó. Após a conclusão da atualização, redefina o nó.

Caminho de atualização	Diferença entre versões	Auto-verificação
	Nos clusters do CCE 1.15 e 1.19, o sistema de arquivos do driver de armazenamento Docker é alterado de XFS para Ext4. Como resultado, a sequência de pacote de importação nos pods da aplicação Java atualizado pode ser anormal, causando exceções de pod.	Antes da atualização, verifique o arquivo de configuração do Docker **/etc/docker/daemon.json** no nó. Verifique se o valor de **dm.fs** é **xfs**. ● Se o valor for **ext4** ou o driver de armazenamento for Overlay, você poderá ignorar as próximas etapas. ● Se o valor for **xfs**, é aconselhável implementar aplicações no cluster da nova versão com antecedência para testar se os aplicativos são compatíveis com a nova versão do cluster. `{ "storage-driver": "devicemapper", "storage-opts": [ "dm.thinpooldev=/dev/mapper/vgpaas-thinpool", "dm.use_deferred_removal=true", "dm.fs=xfs", "dm.use_deferred_deletion=true" ] }`
	O kube-apiserver do CCE 1.19 ou posterior requer que o campo Subject Alternative Names (SANs) esteja configurado para o certificado do servidor webhook. Caso contrário, kube-apiserver falha ao chamar o servidor webhook após a atualização, e os contêineres não podem ser iniciados corretamente. Causa raiz: X.509 **CommonName** é descartado em Go 1.15. kube-apiserver do CCE 1.19 é compilado usando Go 1.15. O campo **CommonName** é processado como o nome do host. Como resultado, a autenticação falha.	Antes da atualização, verifique se o campo SAN está configurado no certificado do servidor webhook. ● Se você não tem seu próprio servidor webhook, você pode pular esta verificação. ● Se o campo não estiver definido, é aconselhável usar o campo SAN para especificar o endereço IP e o nome de domínio suportados pelo certificado. **AVISO** Para mitigar o impacto das diferenças de versão na atualização do cluster, o CCE executa um processamento especial durante a atualização de 1.15 para 1.19 e ainda oferece suporte a certificados sem SANs. No entanto, nenhum processamento especial é necessário para atualizações subsequentes. Aconselha-se que retifique o seu certificado o mais rapidamente possível.

Caminho de atualização	Diferença entre versões	Auto-verificação
	Em clusters da v1.17.17 e posteriores, o CCE cria automaticamente políticas de segurança de pods (PSPs) para você, o que restringe a criação de pods com configurações inseguras, por exemplo, pods para os quais **net.core.somaxconn** sob um sysctl é configurado no contexto de segurança.	Após uma atualização, você pode permitir configurações inseguras do sistema, conforme necessário. Para mais detalhes, consulte **Configuração de uma política de segurança de pod**.
	Se initContainer ou Istio for usado na atualização in-loco de um cluster v1.15, preste atenção às seguintes restrições: No kubelet 1.16 e versões posteriores, as **classes de QoS** são diferentes daquelas em versões anteriores. No kubelet 1.15 e versões anteriores, somente contêineres em **spec.containers** são contados. No kubelet 1.16 e versões posteriores, os contêineres em **spec.containers** e **spec.initContainers** são contados. A classe de QoS de um pod mudará após a atualização. Como resultado, o contêiner no pod é reiniciado.	É aconselhável modificar a classe de QoS do contêiner de serviço antes da atualização para evitar esse problema. Para mais detalhes, consulte **Tabela 2-14**.
v1.13 a v1.15	Depois que um cluster de rede da VPC é atualizado, o nó principal ocupa um bloco CIDR extra devido à atualização dos componentes de rede. Se nenhum bloco CIDR de contêiner estiver disponível para o novo nó, o pod agendado para o nó não poderá ser executado.	Geralmente, esse problema ocorre quando os nós no cluster estão prestes a ocupar totalmente o bloco CIDR do contêiner. Por exemplo, o bloco CIDR do contêiner é 10.0.0.0/16, o número de endereços IP disponíveis é 65.536 e a rede da VPC recebe um bloco CIDR com o tamanho fixo (usando a máscara para determinar o número máximo de endereços IP de contêiner alocados para cada nó). Se o limite superior for 128, o cluster suportará um máximo de 512 (65536/128) nós, incluindo os três nós principais. Depois que o cluster é atualizado, cada um dos três nós principais ocupa um bloco CIDR. Como resultado, 506 nós são suportados.

Tabela 2-14 Mudanças de classe de QoS antes e depois da atualização

Init contêiner (calculado com base em spec.initContainers)	Contêiner de serviço (calculado com base em spec.containers)	Pod (calculado com base em spec.containers e spec.initContainers)	Impactado ou não
Guaranteed	Besteffort	Burstable	Sim
Guaranteed	Burstable	Burstable	Não
Guaranteed	Guaranteed	Guaranteed	Não
Besteffort	Besteffort	Besteffort	Não
Besteffort	Burstable	Burstable	Não
Besteffort	Guaranteed	Burstable	Sim
Burstable	Besteffort	Burstable	Sim
Burstable	Burstable	Burstable	Não
Burstable	Guaranteed	Burstable	Sim

Atualizar backup

Como fazer backup de um nó:

- Backup do banco de dados etcd: o CCE faz backup automaticamente do banco de dados etcd durante a atualização do cluster.
- Backup de nó principal (recomendado, **confirmação manual necessária**): na página de confirmação da atualização, clique em **Backup** para fazer backup de todo o nó principal do cluster. O processo de backup usa o serviço Cloud Backup and Recovery (CBR) e leva cerca de 20 minutos. Se houver muitas tarefas de backup de nuvem no site atual, o tempo de backup pode ser prolongado

2.4.3 Atualização in-loco

Você pode atualizar seus clusters para uma versão mais recente no console do CCE.

Antes da atualização, saiba mais sobre a versão de destino para a qual cada cluster do CCE pode ser atualizado de que maneiras e os impactos da atualização. Para mais detalhes, veja **Visão geral de atualização** e **Antes de começar**.

Descrição

- Uma atualização in-loco atualiza os componentes do Kubernetes em nós de cluster, sem alterar a versão do sistema operacional.
- Os nós do plano de dados são atualizados em lotes. Por padrão, eles são priorizados com base em sua CPU, memória e **PodDisruptionBudgets (PDBs)**. Você também pode definir as prioridades de acordo com seus requisitos de serviço.

Precauções

- Durante a atualização do cluster, o sistema atualizará automaticamente os complementos para uma versão compatível com a versão do cluster de destino. Não desinstale ou reinstale complementos durante a atualização do cluster.
- Antes da atualização, certifique-se de que todos os complementos estejam em execução. Se um complemento não for atualizado, corrija a falha e tente novamente.
- Durante a atualização, o CCE verifica o status de execução do complemento. Alguns complementos (como CoreDNS) exigem pelo menos dois nós para serem executados normalmente. Nesse caso, pelo menos dois nós devem estar disponíveis para a atualização.
- Se uma mensagem de falha de atualização for exibida durante a atualização do cluster, corrija a falha conforme solicitado e tente novamente. Se as tentativas de atualização falharem novamente, **envie um tíquete de serviço** para obter assistência.

Para obter mais informações, consulte **Antes de começar**.

Procedimento

A atualização do cluster passa por verificação, backup, configuração e atualização e verificação.

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster.

Passo 2 No painel de navegação, escolha **Cluster Upgrade**.

Passo 3 Execute a verificação de pré-atualização. Clique em **Start Check** e confirme a verificação. Se houver itens anormais ou arriscados no cluster, trate as exceções com base nos resultados da verificação exibidos na página e verifique novamente.

- **Exceptions**: visualize a solução exibida na página, trate as exceções e verifique novamente.
- **Risk Items**: podem afetar a atualização do cluster. Verifique a descrição do risco e veja se você pode ser afetado. Se não houver risco, clique em **OK** ao lado do item de risco para ignorar manualmente esse item de risco e verifique novamente.

Depois que a verificação for aprovada, clique em **Next: Back Up**.

Passo 4 (Opcional) Faça backup manual dos dados do cluster. O backup dos dados é feito durante a atualização seguindo uma política padrão. Você pode clicar em **Back Up** para fazer backup manual dos dados. Se você não precisar fazer backup dos dados manualmente, clique em **Next: Configure & Upgrade**.

O backup manual fará backup de todo o nó principal. O processo de backup usa o serviço Cloud Backup and Recovery (CBR) e leva cerca de 20 minutos. Se houver muitas tarefas de backup em nuvem no site atual, o backup pode demorar mais tempo. O cluster não pode ser atualizado durante o backup.

Passo 5 Configure os parâmetros de upgrade.

- **Add-on Upgrade Configuration**: os complementos que foram instalados no cluster são listados. Durante a atualização do cluster, o sistema atualiza automaticamente os complementos para serem compatíveis com a versão do cluster de destino. Você pode clicar em **Set** para redefinir os parâmetros do complemento.

NOTA

Se um ponto vermelho for exibido à direita de um complemento, o complemento é incompatível com a versão do cluster de destino. Durante a atualização, o complemento será desinstalado e, em seguida, reinstalado. Certifique-se de que os parâmetros do complemento estejam configurados corretamente.

- **Node Upgrade Configuration:** você pode definir o número máximo de nós a serem atualizados em um lote.
- **Node Priority:** você pode definir prioridades para os nós a serem atualizados. Se você não definir esse parâmetro, o sistema determinará os nós a serem atualizados em lotes com base em condições específicas. Antes de definir a prioridade de atualização do nó, selecione um pool de nós. Os nós e os pools de nós serão atualizados de acordo com as prioridades que você especificar.
  - **Add Upgrade Priority**: adicionar prioridades de atualização para pools de nós.
  - **Add Node Priority**: depois de adicionar uma prioridade de pool de nós, você pode definir a sequência de upgrade de nós no pool de nós. O sistema atualiza os nós na sequência especificada. Se você ignorar essa configuração, o sistema atualizará os nós com base na política padrão.

Passo 6 Após a conclusão da configuração, clique em **Upgrade** e confirme a atualização. O cluster começa a ser atualizado. Você pode ver o processo na parte inferior da página.

Durante a atualização, você pode clicar em **Suspend** à direita para suspender a atualização do cluster. Para continuar a atualização, clique em **Continue**. Quando a barra de progresso atinge 100%, a atualização do cluster está concluída.

NOTA

Se uma mensagem de falha de atualização for exibida durante a atualização do cluster, corrija a falha conforme solicitado e tente novamente.

Passo 7 Após a conclusão da atualização, clique em **Next: Verify**. Verifique a atualização com base nos itens de verificação exibidos. Depois de confirmar que todos os itens de verificação estão normais, clique em **Complete** e confirme se a verificação pós-atualização está concluída.

Você pode verificar a versão do Kubernetes do cluster na página **Clusters**.

----Fim

Perguntas frequentes

- **O que fazer se um complemento de cluster falhar ao ser atualizado durante atualização de cluster do CCE?**

2.4.4 Execução de verificação pós-atualização

2.4.4.1 Verificação de pod

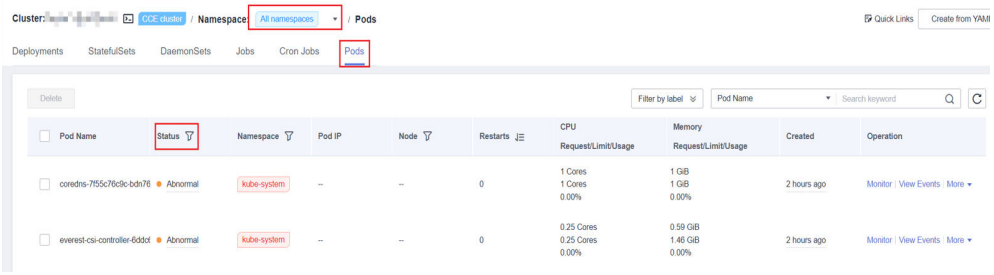
Item de verificação

- Verifique se há pods inesperados no cluster.
- Verifique se há algum pod que foi executado corretamente originalmente na reinicialização do cluster inesperadamente.

Procedimento

Efetue logon no console do CCE e acesse o console do cluster. Escolha **Workloads** no painel de navegação. Na página exibida, alterne para a página da guia **Pods**. Selecione todos os namespaces, clique em **Status** e verifique se há algum pod anormal.

Figura 2-11 Consultar o status do pod



Veja a coluna **Restarts** para verificar se há pods que são reiniciados de forma anormal.

Figura 2-12 Verificar o número de vezes de reinicialização do pod

Solução

Se houver pods anormais no cluster após a atualização do cluster, entre em contato com o suporte técnico.

2.4.4.2 Verificação de rede de nó e contêiner

Item de verificação

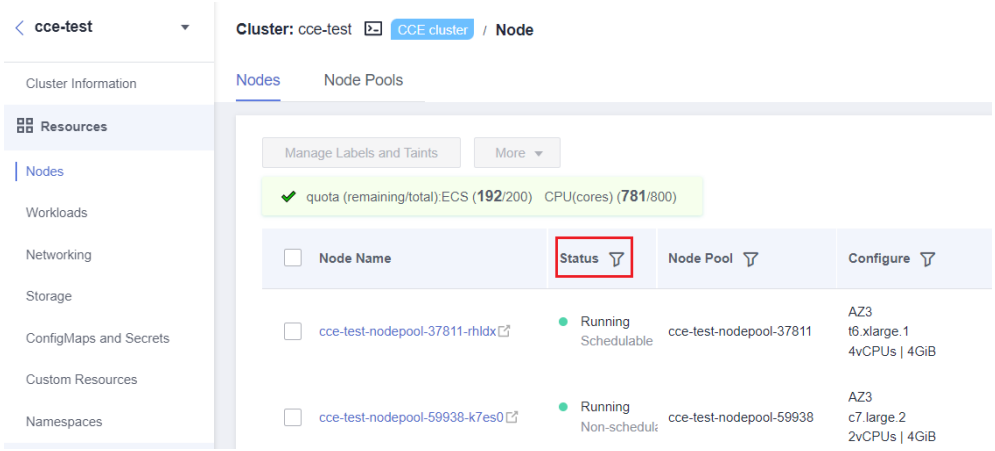
- Verifique se os nós estão funcionando corretamente.
- Verifique se a rede do nó está normal.
- Verifique se a rede de contêineres está normal.

Procedimento

O status do nó reflete se o componente do nó ou a rede é normal.

Vá para console do CCE e acesse o console do cluster. Escolha **Nodes** no painel de navegação. Você pode filtrar o status do nó por status para verificar se há nós anormais.

Figura 2-13 Consultar o status do nó



A rede de contêineres afeta os serviços. Verifique se os seus serviços estão disponíveis.

Solução

Se o status do nó for anormal, entre em contato com o suporte técnico.

Se a rede de contêineres estiver anormal e seus serviços forem afetados, entre em contato com o suporte técnico e confirme o caminho de acesso à rede anormal.

Fonte	Destino	Tipo de destino	Possível falha
• Pods (dentro de um cluster) • Nós (dentro de um cluster) • Nós na mesma VPC (fora de um cluster) • Nuvens de terceiros	Endereço IP público do Serviço do ELB	Entrada de balanceamento de carga de tráfego de cluster	Sem registro.
	Endereço IP privado do Serviço do ELB	Entrada de balanceamento de carga de tráfego de cluster	Sem registro.
	Endereço IP público do ingress do ELB	Entrada de balanceamento de carga de tráfego de cluster	Sem registro.
	Endereço IP privado do ingress do ELB	Entrada de balanceamento de carga de tráfego de cluster	Sem registro.
	Endereço IP público do Serviço NodePort	Entrada de tráfego do cluster	A configuração do proxy do kube é sobrescrita. Esta falha foi corrigida no processo de atualização.

Fonte	Destino	Tipo de destino	Possível falha
	Endereço IP privado do Serviço NodePort	Entrada de tráfego do cluster	Sem registro.
	Serviço ClusterIP	Plano de rede de serviço	Sem registro.
	Porta de Serviço não NodePort	Rede de contêiner	Sem registro.
	Pods entre nós	Plano de rede de contêiner	Sem registro.
	Pods no mesmo nó	Plano de rede de contêiner	Sem registro.
	Os nomes de domínio de serviços e pods são resolvidos pelo CoreDNS.	Resolução do nome de domínio	Sem registro.
	Os nomes de domínio externos são resolvidos com base na configuração dos hosts CoreDNS.	Resolução do nome de domínio	Depois que o CoreDNS é atualizado, a configuração é sobrescrita. Esta falha foi corrigida no processo de atualização do complemento.
	Os nomes de domínio externos são resolvidos com base no servidor CoreDNS upstream.	Resolução do nome de domínio	Depois que o CoreDNS é atualizado, a configuração é sobrescrita. Esta falha foi corrigida no processo de atualização do complemento.
	Os nomes de domínio externos não são resolvidos pelo CoreDNS.	Resolução do nome de domínio	Sem registro.

2.4.4.3 Verificação de rótulo e mancha do nó

Item de verificação

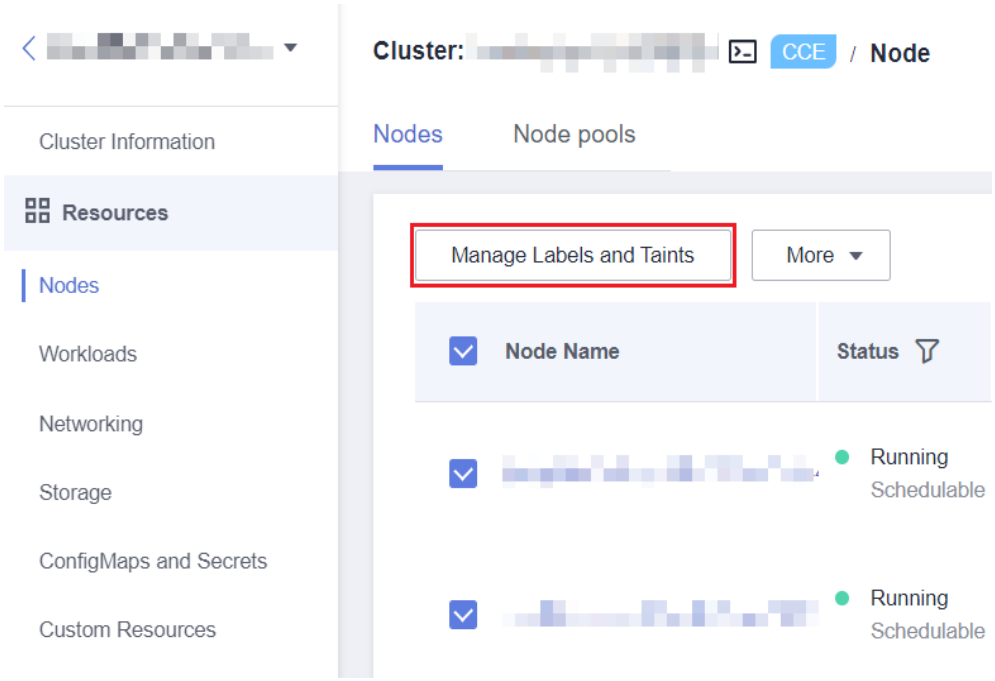
- Verifique se os rótulos de nó personalizados são perdidos.

- Verifique se há quaisquer manchas inesperadas recém-adicionadas no nó, o que afetará o agendamento da carga de trabalho.

Procedimento

Vá ao console do CCE, alcance o console do conjunto e escolha **Nodes** no painel de navegação. Na página exibida, clique na guia **Nodes**, selecione todos os nós e clique em **Manage Labels and Taints** para exibir os rótulos e manchas do nó atual.

Figura 2-14 Exibir rótulos e manchas de nó



Solução

Os rótulos personalizados não serão alterados durante uma atualização de cluster. Se você achar que os rótulos foram perdidos ou adicionados inesperadamente, entre em contato com o suporte técnico.

Se você encontrar uma nova mancha (**node.kubernetes.io/upgrade**) em um nó, o nó pode ser ignorado durante a atualização. Para mais detalhes, consulte **Verificação de ignoração de nó para redefinição**.

Se você achar que outras manchas foram adicionadas ao nó, entre em contato com o suporte técnico.

2.4.4.4 Verificação de ignoração de nó para redefinição

Item de verificação

Depois que o cluster for atualizado, redefina os nós que não forem atualizados.

Procedimento

Volte para a etapa anterior ou visualize os detalhes da atualização na página do histórico de atualização para visualizar os nós que são ignorados durante a atualização.

Figura 2-15 Detalhes da atualização

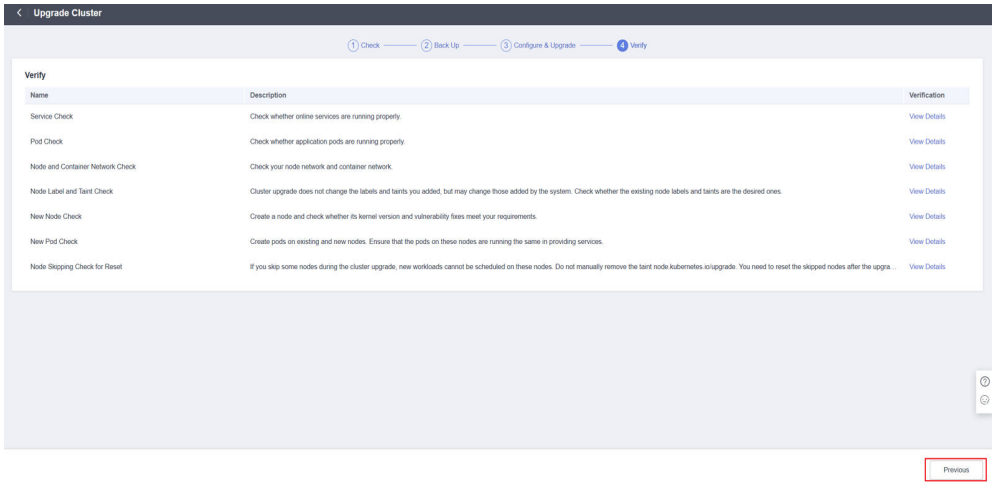
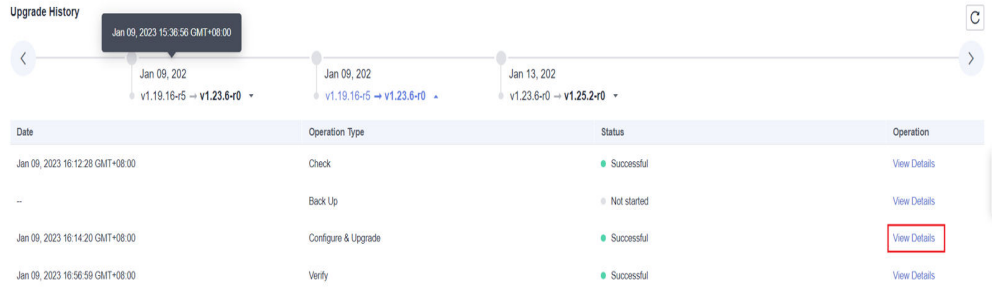


Figura 2-16 Visualizar detalhes



Os nós ignorados são exibidos na página de detalhes da atualização. Redefina os nós ignorados após a conclusão da atualização. Para obter detalhes sobre como redefinir um nó, consulte **Redefinição de um nó**.

NOTA

A redefinição de um nó redefinirá todos os rótulos de nó, o que pode afetar o agendamento da carga de trabalho. Antes de redefinir um nó, verifique e retenha os rótulos que você adicionou manualmente ao nó.

2.4.4.5 Verificação do serviço

Item de verificação

Depois que o cluster for atualizado, verifique se os serviços estão sendo executados normalmente.

Procedimento

Diferentes serviços têm diferentes modos de verificação. Selecione um adequado e verifique o serviço antes e depois da atualização.

Você pode verificar o serviço a partir dos seguintes aspectos:

- A página de serviço está disponível.
- Nenhum alarme ou evento é gerado na plataforma normal.
- Nenhum registro de erro é gerado para processos principal.
- O teste de discagem da API é normal.

Solução

Se os seus serviços online estiverem anormais após a atualização do cluster, entre em contato com o suporte técnico.

2.4.4.6 Verificação de novo nó

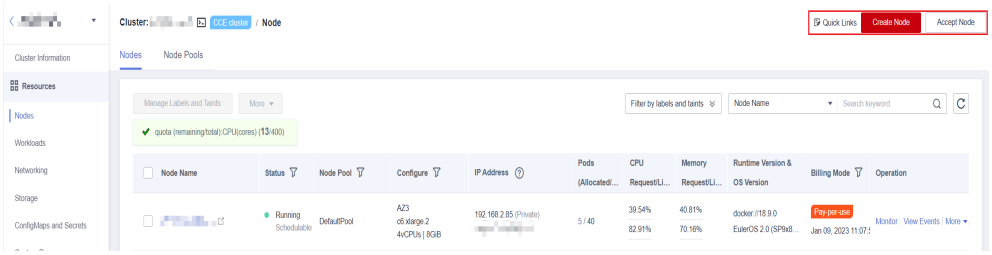
Item de verificação

Verifique se os nós podem ser criados no cluster.

Procedimento

Efetue logon no console do CCE e acesse o console do cluster. Escolha **Nodes** no painel de navegação e clique em **Create Node**. Para obter detalhes sobre configurações de nó, consulte **Criação de um nó**.

Figura 2-17 Criar um nó



Solução

Se os nós não puderem ser criados no cluster após a atualização do cluster, entre em contato com o suporte técnico.

2.4.4.7 Verificação de pod novo

Item de verificação

- Verifique se os pods podem ser criados nos nós existentes após a atualização do cluster.
- Verifique se os pods podem ser criados em novos nós após a atualização do cluster.

Procedimento

Depois de criar um nó baseado em **Verificação de novo nó**, crie uma carga de trabalho de DaemonSet para criar pods em cada nó.

Vá para o console do CCE, acesse o console do cluster e escolha **Workloads** no painel de navegação. Na página exibida, alterne para a página de guia **DaemonSets** e clique em **Create Workload** ou **Create from YAML** no canto superior direito. Para mais detalhes, consulte **Criação de um DaemonSet**.

Figura 2-18 Criar um DaemonSet

É aconselhável usar a imagem para testes de rotina como a imagem de base. Você pode implementar um pod consultando o seguinte arquivo YAML.

NOTA

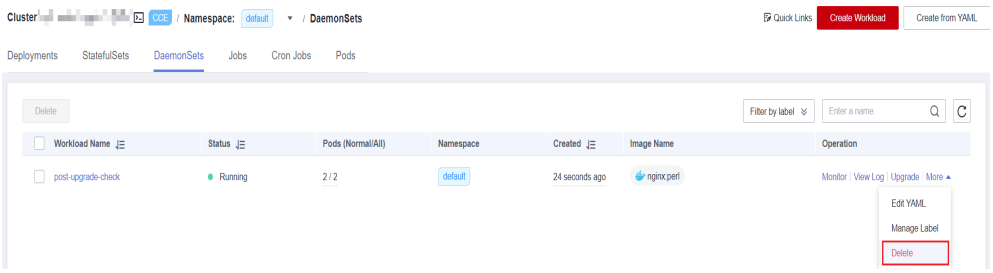
Neste teste, o YAML implementa o DaemonSet no namespace padrão, usa **ngxin:perl** como imagem base, solicita 10 MB de CPU e 10 MB de memória e limita 100 MB de CPU e 50 MB de memória.

```
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: post-upgrade-check
  namespace: default
spec:
  selector:
    matchLabels:
      app: post-upgrade-check
      version: v1
  template:
    metadata:
      labels:
        app: post-upgrade-check
        version: v1
    spec:
      containers:
        - name: container-1
          image: nginx:perl
          imagePullPolicy: IfNotPresent
          resources:
            requests:
              cpu: 10m
              memory: 10Mi
            limits:
              cpu: 100m
              memory: 50Mi
```

Depois que a carga de trabalho for criada, verifique se o status do pod da carga de trabalho é normal.

Depois que a verificação for concluída, vá para o console do CCE e acesse o console do cluster. Escolha **Workloads** no painel de navegação. Na página exibida, alterne para a página de guia **DaemonSets**, escolha **More** > **Delete** na coluna **Operation** da carga de trabalho **post-upgrade-check** para excluir a carga de trabalho de teste.

Figura 2-19 Excluir um DaemonSet



Solução

Se o pod não puder ser criado ou se o status do pod for anormal, entre em contato com o suporte técnico e especifique se a exceção ocorre em nós novos ou em nós existentes.

2.4.5 Migração de serviços em clusters de versões diferentes

Cenários de aplicações

Esta seção descreve como migrar serviços de um cluster de uma versão anterior para um cluster de uma versão posterior no CCE.

Esta operação é aplicável quando uma atualização de cluster entre versões é necessária (por exemplo, atualização de v1.7.* ou v1.9.* para 1.17.*) e novos clusters podem ser criados para a migração de serviços.

Pré-requisitos

Tabela 2-15 Lista de verificação antes da migração

Categoria	Descrição
Cluster	Relacionados ao NodeIP: verifique se os endereços IP do nó (incluindo EIPs) do cluster antes da migração foram usados em outras configurações ou listas brancas.
Cargas de trabalho	Registre o número de cargas de trabalho para verificação pós-migração.
Armazenamento	1. Verifique se os recursos de armazenamento em uso são provisionados pela nuvem ou pela sua organização. 2. Altere o armazenamento criado automaticamente para o armazenamento existente no novo cluster.
Rede	1. Preste atenção especial ao ELB e ao ingress. 2. Clusters de uma versão anterior suportam apenas o balanceador de carga clássico. Para migrar serviços para um novo cluster, altere o tipo de balanceador de carga para balanceador de carga compartilhado. Em seguida, o serviço ELB correspondente será restabelecido.

Categoria	Descrição
O&M	Configuração privada: verifique se os parâmetros do kernel ou os dados do sistema foram configurados nos nós do cluster.

Procedimento

Passo 1 Crie um cluster do CCE.

Crie um cluster com as mesmas especificações e configurações do cluster da versão anterior. Para mais detalhes, consulte **Compra de um cluster do CCE**.

Passo 2 Adicione um nó.

Adicione nós com as mesmas especificações e itens de configuração manual. Para mais detalhes, consulte **Criação de um nó**.

Passo 3 Crie um volume de armazenamento no novo cluster.

Use um volume de armazenamento existente para criar uma PVC no novo cluster. O nome da PVC permanece inalterado. Para obter detalhes, consulte **Uso de um bucket do OBS existente através de um PV estático** ou **Uso de um sistema de arquivos do SFS Turbo existente por meio de um PV estático**.

NOTA

A comutação de armazenamento suporta apenas buckets do OBS e sistemas de arquivos do SFS Turbo. Se o armazenamento não compartilhado for usado, suspenda as cargas de trabalho no cluster antigo para alternar os recursos de armazenamento. Como resultado, os serviços ficarão indisponíveis.

Passo 4 Crie uma carga de trabalho no novo cluster.

O nome e as especificações da carga de trabalho permanecem inalterados. Para obter detalhes sobre como criar uma carga de trabalho, consulte **Criação de uma Implantação** ou **Criação de um StatefulSet**.

Passo 5 Monte o armazenamento novamente.

Monte o armazenamento existente na carga de trabalho novamente. Para obter detalhes, consulte **Uso de um bucket do OBS existente através de um PV estático** ou **Uso de um sistema de arquivos do SFS Turbo existente por meio de um PV estático**.

Passo 6 Crie um Serviço no novo cluster.

O nome e as especificações do Serviço permanecem inalterados. Para obter detalhes sobre como criar um Serviço, consulte **Serviço**.

Passo 7 Serviços da comissão.

Depois que todos os recursos forem criados, comissione os serviços em contêiner. Se o comissionamento for bem-sucedido, migre os serviços para o novo cluster.

Passo 8 Exclua ou cancele a assinatura do cluster antigo.

Quando todas as funções do novo cluster estiverem estáveis, cancele a assinatura ou exclua o cluster antigo. Para obter detalhes sobre como excluir um cluster, consulte **Exclusão de um cluster**.

----Fim

2.4.6 Solução de problemas para exceções de verificação de pré-atualização

2.4.6.1 Verificação de pré-atualização

O sistema executa uma verificação abrangente de pré-atualização antes da atualização do cluster. Se o cluster não atender às condições de verificação de pré-atualização, a atualização não poderá continuar. Para evitar riscos de atualização, você pode executar a verificação de pré-atualização de acordo com os itens de verificação fornecidos por esta seção.

Tabela 2-16 Itens de verificação

N.º	Item de verificação	Descrição
1	**Restrições de nó**	● Verifique se o nó está disponível. ● Verifique se o sistema operacional do nó suporta a atualização. ● Verifique se o nó está marcado com rótulos de pool de nós inesperados. ● Verifique se o nome do nó do Kubernetes é o mesmo que o nome do ECS.
2	**Gerenciamento de atualização**	Verifique se o cluster de destino está em gerenciamento de upgrade.
3	**Complementos**	● Verifique se o status do complemento é normal. ● Verifique se o complemento suporta a versão de destino.
4	**Gráficos do Helm**	Verifique se o registro HelmRelease atual contém APIs de Kubernetes descartadas que não são suportadas pela versão de cluster de destino. Se sim, o gráfico de Helm pode estar indisponível após a atualização.
5	**Conectividade SSH de nós principais**	Verifique se o CCE pode se conectar aos nós principais.
6	**Pools de nós**	Verifique o status do pool do nó.
7	**Grupos de segurança**	Verifique se o grupo de segurança permite que o nó principal acesse nós usando ICMP.
8	**Restrições de nó Arm**	● Verifique se o cluster contém nós Arm. ● Verifique se o cluster é um cluster de Kunpeng ou se o nó principal de um cluster híbrido é da arquitetura ARM.

N.º	Item de verificação	Descrição
9	**Nós a serem migrados**	Verifique se o nó precisa ser migrado.
10	**Recursos do Kubernetes descartados**	Verifique se há recursos descartados nos clusters.
11	**Riscos de compatibilidade**	Leia as diferenças de compatibilidade de versão e certifique-se de que elas não sejam afetadas. A atualização de patch não envolve diferenças de compatibilidade de versão.
12	**Versões do agente do CCE de nó**	Verifique se o cce-agent no nó atual é da versão mais recente.
13	**Uso da CPU do nó**	Verifique se o uso da CPU do nó excede 90%.
14	**CRDs**	● Verifique se um CRD principal **packageversions.version.cce.io** é excluído. ● Verifique se o CRD principal do cluster **network-attachment-definitions.k8s.cni.cncf.io** é excluído.
15	**Discos de nó**	● Verifique se os discos de dados principais no nó atendem aos requisitos de atualização. ● Verifique se o diretório **/tmp** tem 500 MB de espaço disponível.
16	**DNS do nó**	● Verifique se a configuração de DNS do nó atual pode resolver o endereço do OBS. ● Verifique se o nó atual pode acessar o endereço do OBS do pacote de componente de atualização de armazenamento.
17	**Permissões de arquivo de diretório principal de nó**	Verifique se o diretório principal **/var/paas** nos nós contém arquivos com proprietários anormais ou grupos de proprietários.
18	**Kubelet**	Verifique se o kubelet no nó está sendo executado corretamente.
19	**Memória do nó**	Verifique se o uso da memória do nó excede 90%.
20	**Servidor de sincronização de relógio de nó**	Verifique se o servidor de sincronização de relógio ntpd ou chronyd do nó está sendo executado corretamente.
21	**Sistema operacional do nó**	Verifique se a versão do kernel do sistema operacional do nó é suportada pelo CCE.
22	**CPUs do nó**	Verifique se o número de CPUs no nó principal é maior que 2.
23	**Comandos Python do nó**	Verifique se os comandos Python estão disponíveis em um nó.

N.º	Item de verificação	Descrição
24	**Versão do ASM**	● Verifique se o ASM é usado pelo cluster. ● Verifique se a versão atual do ASM oferece suporte à versão do cluster de destino.
25	**Prontidão do nó**	Verifique se os nós no cluster estão prontos.
26	**Nó journald**	Verifique se o journald de um nó é normal.
27	**containerd.sock**	Verifique se o arquivo containerd.sock existe no nó. Esse arquivo afeta a inicialização do tempo de execução do container no Euler OS.
28	**Erros internos**	Antes da atualização, verifique se ocorre um erro interno.
29	**Pontos de montagem do nó**	Verifique se existem pontos de montagem inacessíveis no nó.
30	**Manchas de nós do Kubernetes**	Verifique se a mancha necessário para a atualização do cluster existe no nó.
31	**Restrições de everest**	Verifique se há restrições de compatibilidade no complemento everest atual.
32	**Restrições de cce-hpa-controller**	Verifique se o atual complemento cce-controller-hpa tem restrições de compatibilidade.
33	**Políticas de CPU aprimorada**	Verifique se a versão atual do cluster e a versão de destino oferecem suporte à **política de CPU aprimorada**.
34	**Integridade dos componentes do nó de trabalho**	Verifique se o tempo de execução do contêiner e os componentes de rede nos nós de trabalho estão íntegros.
35	**Integridade dos componentes do nó principal**	Verifique se o Kubernetes, o tempo de execução do contêiner e os componentes de rede dos nós principais estão íntegros.
36	**Limite de recursos de memória dos componentes do Kubernetes**	Verifique se os recursos dos componentes do Kubernetes, como etcd e kube-controller-manager, excedem o limite superior.
37	**APIs do Kubernetes descartadas**	O sistema verifica os logs de auditoria do dia anterior para verificar se o usuário chama as APIs preteridas da versão de destino do Kubernetes. **NOTA** Devido ao intervalo de tempo limitado dos logs de auditoria, esse item de verificação é apenas um método auxiliar. As APIs a serem preteridas podem ter sido usadas no cluster, mas seu uso não está incluído nos logs de auditoria do dia anterior. Verifique o uso da API com cuidado.
38	**Capacidades de IPv6 de um cluster do CCE Turbo**	Se o IPv6 é permitido para um cluster do CCE Turbo, verifique se a versão do conjunto de destino apoia o IPv6.

N.º	Item de verificação	Descrição
39	**NetworkManager de nó**	Verifique se o NetworkManager de um nó é normal.
40	**Arquivo de ID do nó**	Verifique o formato do arquivo de ID.
41	**Consistência da configuração do nó**	Quando você atualiza um cluster para v1.19 ou posterior, o sistema verifica se os seguintes arquivos de configuração foram modificados no back-end:
42	**Arquivo da configuração de nó**	Verifique se os arquivos de configuração dos componentes-chave existem no nó.
43	**Consistência da configuração de CoreDNS**	Verifique se o Corefile de configuração principal de CoreDNS atual é diferente do registro de lançamento do Helm. A diferença pode ser substituída durante a atualização do complemento, **afetando a resolução de nomes de domínio no cluster**.
44	**Comandos sudo de um nó**	Se os comandos sudo e arquivos relacionados ao sudo do nó estão funcionando
45	**Comandos principais dos nós**	Se alguns comandos principais dos quais a atualização do nó depende estão funcionando
46	**A montagem do arquivo sock em um nó**	O arquivo **docker/containerd.sock** no nó é montado no pod por meio de um hostPath. Durante a atualização, o Docker/containerd é reiniciado, mas o arquivo sock no contêiner não é alterado. Como resultado, pode ocorrer um erro em seus serviços.
47	**Consistência do certificado do balanceador de carga de HTTPS**	Verifique se o certificado usado por um balanceador de carga de HTTPS foi modificado no ELB.
48	**Montagem do nó**	Verifique se o diretório de montagem padrão e o link suave no nó foram montados ou modificados manualmente.
49	**Permissões de logon de usuário paas em um nó**	Verifique se o usuário **paas** tem permissão para fazer logon em um nó.
50	**Endereços IPv4 privados de balanceadores de carga**	Verifique se o balanceador de carga associado a um Serviço está alocado com um endereço IPv4 privado.
51	**Registros históricos de atualização**	Verifique se a versão de origem do cluster é anterior à v1.11 e se a versão de destino é posterior à v1.23.
52	**Bloco CIDR do plano de gerenciamento do cluster**	Verifique se o bloco CIDR do plano de gerenciamento do cluster é o mesmo que o configurado na rede backbone.

N.º	Item de verificação	Descrição
53	**Complemento da GPU**	O complemento da GPU está envolvido na atualização, o que pode afetar a instalação do driver da GPU durante a criação de um nó da GPU.
54	**Configurações de parâmetro do sistema dos nós**	Verifique se as configurações de parâmetros padrão do sistema em seus nós foram modificadas.
55	**Versões de pacotes residuais**	Verifique se há dados residuais da versão do pacote no cluster atual.
56	**Comandos do nó**	Verifique se os comandos necessários para a atualização estão disponíveis no nó.
57	**Troca de nó**	Verifique se a troca foi ativada nos nós de cluster.

2.4.6.2 Restrições de nó

Itens de verificação

Verifique os seguintes itens:

- Verifique se o nó está disponível.
- Verifique se o sistema operacional do nó suporta a atualização.
- Verifique se o nó está marcado com rótulos de pool de nós inesperados.
- Verifique se o nome do nó do Kubernetes é o mesmo que o nome do ECS.

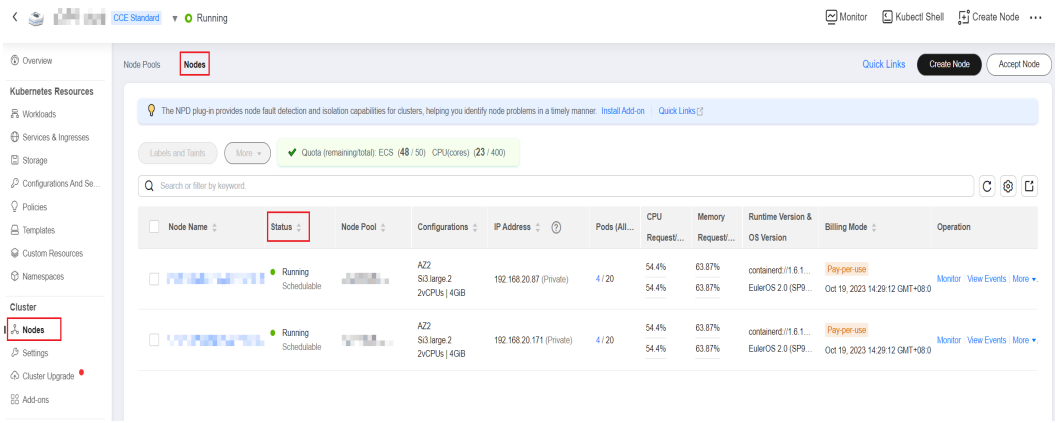
Solução

1. **O nó não está disponível. Recupere preferencialmente o nó.**

   Se um nó não estiver disponível, faça logon no console do CCE e clique no nome do cluster para acessar o console do cluster. Em seguida, escolha **Nodes** no painel de navegação e clique na guia **Nodes**. Assegure-se de que o nó esteja no estado **Running**. Um nó no estado **Installing** ou **Deleting** não pode ser atualizado.

   Se um nó não estiver disponível, recupere o nó e tente novamente a tarefa de verificação. Para obter detalhes, consulte **O que devo fazer se um cluster estiver disponível, mas alguns nós não estiverem disponíveis?**

Figura 2-20 Verificar status de nó



2. **O SO de nó não suporta a atualização.**

   A tabela a seguir lista os sistemas operacionais de nó que suportam a atualização. Você pode redefinir o sistema operacional do nó para um sistema operacional disponível na lista.

   Tabela 2-17 SOs que suportam a atualização

SO	Restrição
EulerOS 2.x	Se a versão de destino for anterior à v1.27, não há restrições. Se a versão alvo for v1.27 ou posterior, apenas o EulerOS 2.9 e o EulerOS 2.10 suportam a atualização.
CentOS 7.x	Nenhuma.
Ubuntu	Se o resultado da verificação mostrar que a atualização não é suportada devido a restrições regionais, entre em contato com o suporte técnico. **NOTA** Se a versão de destino for a v1.27 ou posterior, apenas o Ubuntu 22.04 suporta a atualização.
Huawei Cloud EulerOS	Se o resultado da verificação mostrar que a atualização não é suportada devido a restrições regionais, entre em contato com o suporte técnico.

3. **O nó afetado pertence ao pool de nós padrão, mas é configurado com um rótulo de pool de nós não padrão, que afetará a atualização.**

   Se um nó for migrado de um pool de nós para o pool de nós padrão, o rótulo do pool de nós **cce.cloud.com/cce-nodepool** será retido, afetando a atualização do cluster. Verifique se o agendamento de carga no nó depende do rótulo.

   – Se não, exclua o rótulo.

   – Se sim, modifique a política de balanceamento de carga, remova a dependência e, em seguida, exclua o rótulo.

4. **O nó é marcado com uma mancha CNIProblem. Recupere preferencialmente o nó.**

   O nó contém uma mancha cuja chave é **node.cloudprovider.kubernetes.io/cni-problem** e o efeito é **NoSchedule**. A mancha é adicionada pelo complemento npd. Atualize o

complemento npd para a versão mais recente e verifique novamente. Se o problema persistir, entre em contato com o suporte técnico.

5. **O nó do Kubernetes correspondente ao nó afetado não existe.**

   É possível que o nó esteja sendo excluído. Verifique novamente mais tarde.

2.4.6.3 Gerenciamento de atualização

Itens de verificação

Verifique se o cluster de destino está em gerenciamento de upgrade.

Solução

O CCE pode restringir temporariamente a atualização do cluster devido aos seguintes motivos:

- O cluster é identificado como o núcleo de produção.
- Outras tarefas de O&M estão sendo ou serão executadas, por exemplo, reconstrução 3-AZ em nós mestres.

Para resolver este problema, contacte o suporte técnico.

2.4.6.4 Complementos

Item de verificação

Verifique os seguintes itens:

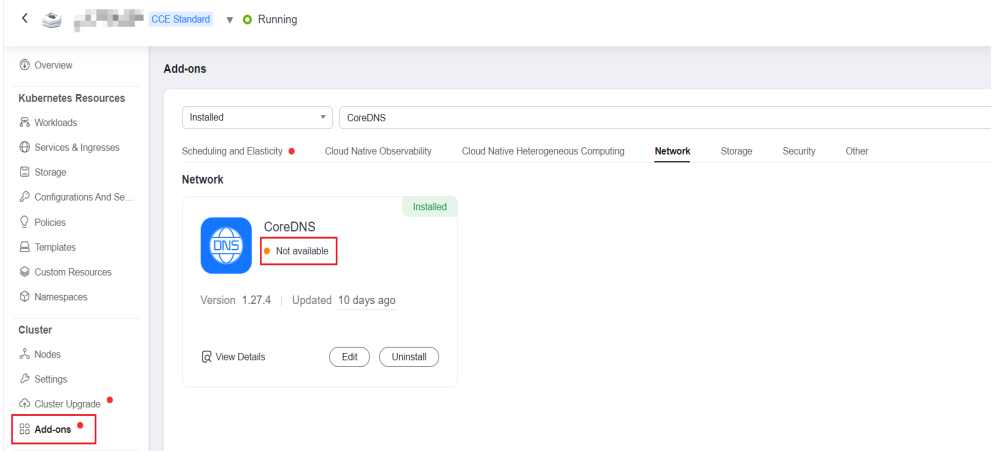
- Verifique se o status do complemento é normal.
- Verifique se o complemento suporta a versão de destino.

Solução

- **Cenário 1: o complemento funciona mal.**

  Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster. Escolha **Add-ons** no painel de navegação e obter complementos. Em seguida, lidar com mau funcionamento complementos.

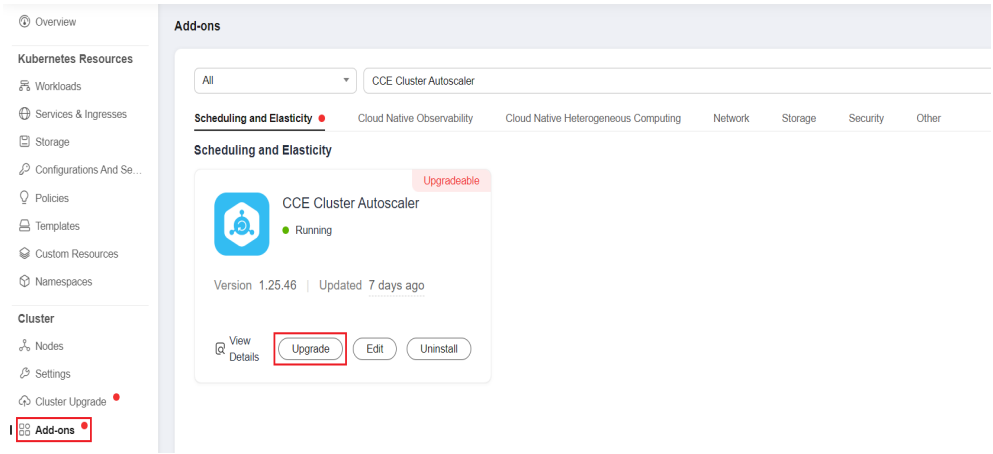
  Figura 2-21 Verificar status de complementos



- **Cenário 2: a versão do cluster de destino não suporta a versão atual do complemento.**

  O complemento não pode ser atualizado automaticamente com o cluster devido a problemas de compatibilidade. Nesse caso, efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster. Escolha **Add-ons** no painel de navegação e atualize manualmente o complemento.

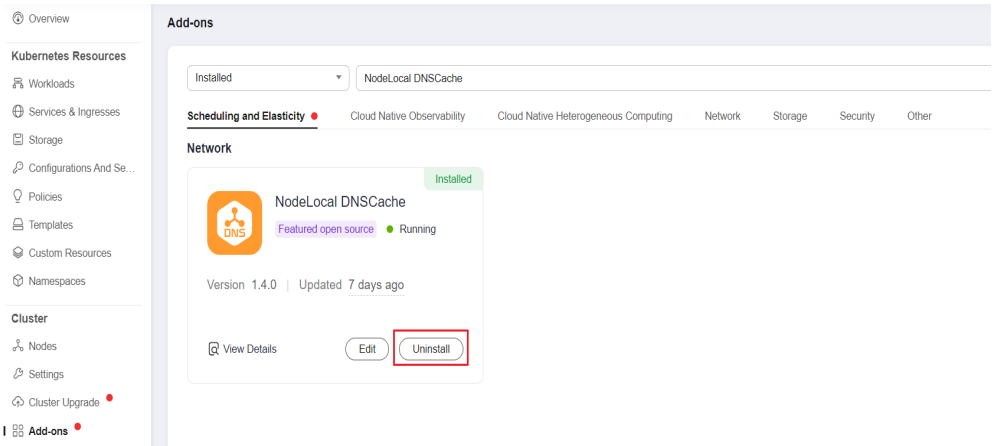
  Figura 2-22 Atualizar um complemento

- **Cenário 3: depois que o complemento é atualizado para a versão mais recente, ele ainda não é suportado pela versão do cluster de destino.**

  Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster. Escolha **Add-ons** no painel de navegação e desinstale manualmente o complemento. Para obter detalhes sobre as versões e substituições de complementos suportadas, consulte o documento **Ajuda**.

  Figura 2-23 Desinstalar um complemento

- **Cenário 4: a configuração do complemento não atende aos requisitos de atualização. Atualize o complemento e tente novamente.**

  As seguintes informações de erro são exibidas durante a verificação de pré-atualização:

  ```
  please upgrade addon [ ] in the page of addon managecheck and try again
  ```

  Nesse caso, efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster. Escolha **Add-ons** no painel de navegação e atualize manualmente o complemento.

2.4.6.5 Gráficos do Helm

Itens de verificação

Verifique se o registro HelmRelease atual contém APIs de Kubernetes descartadas que não são suportadas pela versão de cluster de destino. Se sim, o gráfico de Helm pode estar indisponível após a atualização.

Solução

Converta as APIs do Kubernetes descartadas em APIs compatíveis com as versões de origem e de destino.

NOTA

Este item foi processado automaticamente no processo de atualização. Você pode ignorar este item.

2.4.6.6 Conectividade SSH de nós principais

Itens de verificação

Verifique se o CCE pode se conectar aos nós principais.

Solução

Entre em contato com o suporte técnico.

2.4.6.7 Pools de nós

Itens de verificação

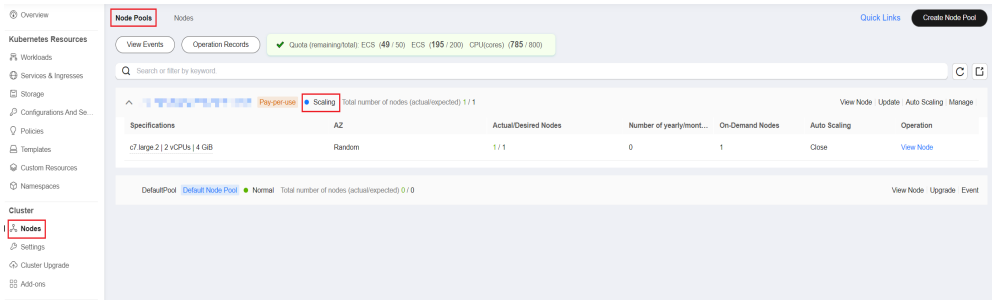
- Verifique o status do pool de nós.

Solução

- **Cenário: o pool de nós funciona mal.**

  Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster. Escolha **Nodes** no painel de navegação e visualize o status do pool de nós afetado na guia **Node Pools**. Se o pool de nós estiver sendo dimensionado, aguarde até que o dimensionamento do pool de nós seja concluído.

  Figura 2-24 Verificar status do pool de nós



2.4.6.8 Grupos de segurança

Itens de verificação

Verifique se o grupo de segurança permite que o nó principal acesse nós usando ICMP.

NOTA

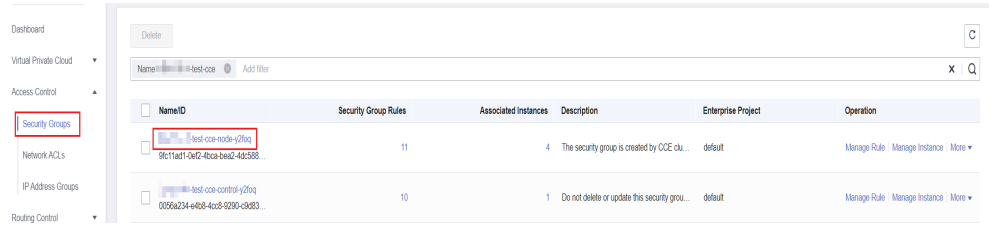
Esse item de verificação é executado somente para clusters que usam redes da VPC. Para clusters que usam outras redes, ignore este item de verificação.

Solução

Faça logon no console da VPC, escolha **Access Control** > **Security Groups** e insira o nome do cluster de destino na caixa de pesquisa. Espera-se que dois grupos de segurança sejam exibidos:

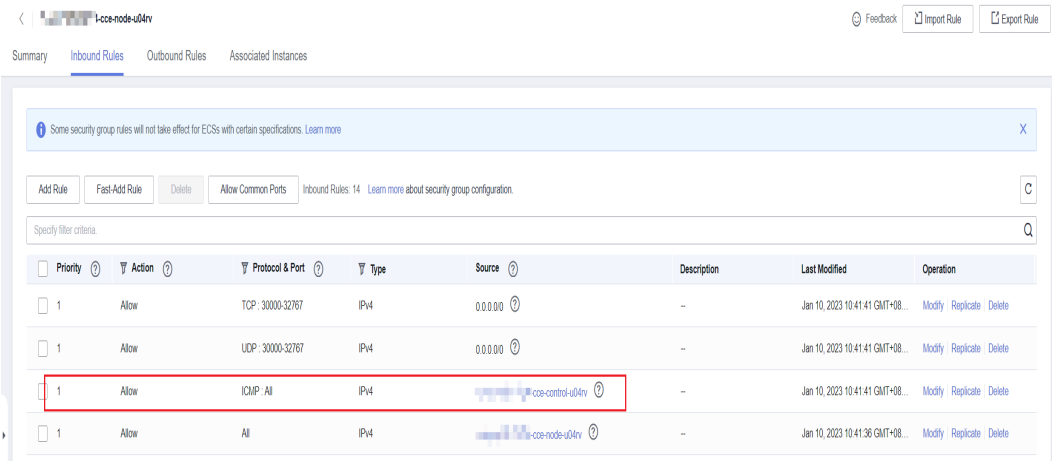
- O nome do grupo de segurança é **cluster name-node-*xxx*** . Esse grupo de segurança está associado aos nós de trabalho.
- O nome do grupo de segurança é **cluster name-control-*xxx*** . Esse grupo de segurança está associado aos nós principais.

Figura 2-25 Grupos de segurança de cluster



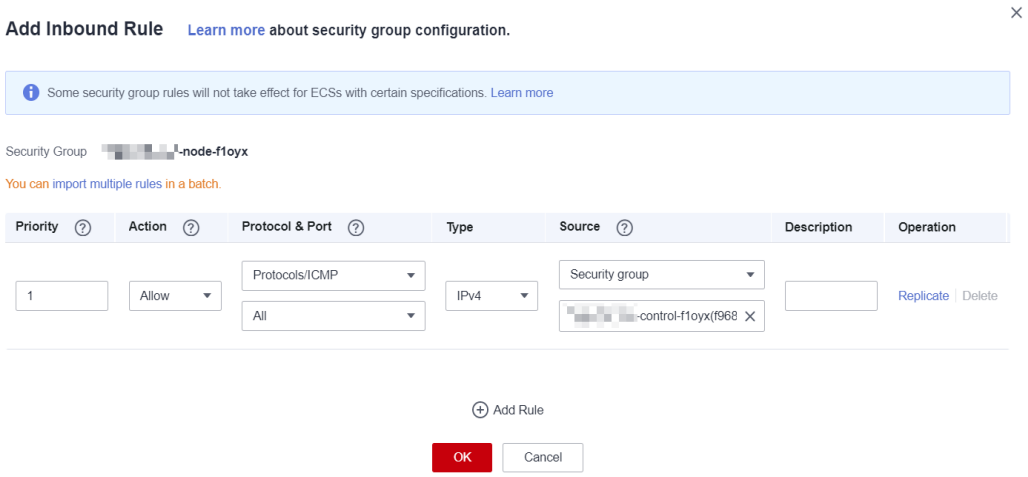
Clique no grupo de segurança do usuário do nó e verifique se as regras a seguir estão configuradas para permitir que o nó principal acesse o nó usando **ICMP**.

Figura 2-26 Regras de grupo de segurança de nó



Se a regra de grupo de segurança anterior não estiver disponível, adicione a regra com as seguintes configurações ao grupo de segurança de nó: defina **Protocol & Port** para **Protocols/ICMP** e **All** e **Source** para **Security group** e o grupo de segurança principal. Descreva a regra como "Created by CCE,please don't modify! Used by the master node to access the worker node."

Figura 2-27 Permitir ICMP para o grupo de segurança principal



2.4.6.9 Restrições de nó Arm

Itens de verificação

Verifique os seguintes itens:

- Verifique se o cluster contém nós Arm.
- Verifique se o cluster é um cluster de Kunpeng ou se o nó principal de um cluster híbrido é da arquitetura ARM.

Solução

- **Cenário 1: o cluster contém nós Arm.**

  Exclua nós Arm.
- **Cenário 2: o cluster é um cluster Kunpeng ou o nó principal de um cluster híbrido é da arquitetura Arm.**

  A versão mais recente para a qual os clusters de Kunpeng podem ser atualizados é a v1.25.

2.4.6.10 Nós a serem migrados

Itens de verificação

Verifique se o nó precisa ser migrado.

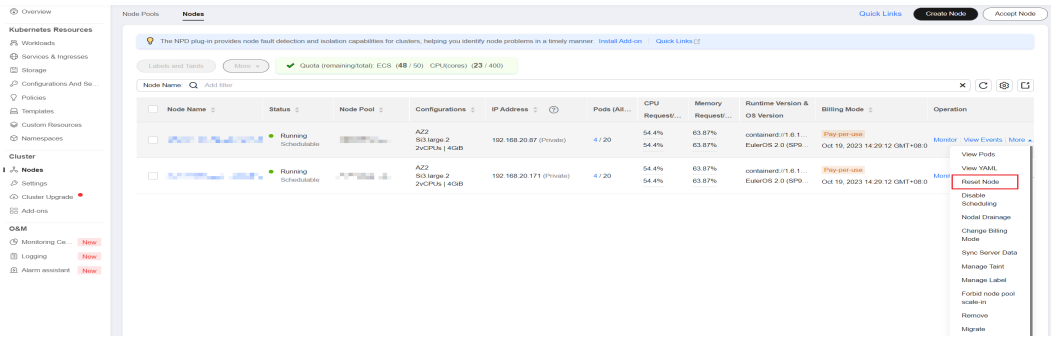
Solução

Para o cluster 1.15 que é atualizado do 1.13 no modo contínuo, migre (reinicie ou crie e substitua) todos os nós antes de executar a atualização novamente.

Solução 1

Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster. Escolha **Nodes** no painel de navegação. Localize a linha que contém o nó de destino e escolha **More** > **Reset Node** na coluna **Operation**. Para mais detalhes, consulte **Redefinição de um nó**. Depois que o nó for redefinido, tente novamente a tarefa de verificação.

Figura 2-28 Redefinição de um nó



NOTA

A redefinição de um nó redefinirá todos os rótulos de nó, o que pode afetar o agendamento da carga de trabalho. Antes de redefinir um nó, verifique e retenha os rótulos que você adicionou manualmente ao nó.

Solução 2

Após criar um nó, exclua o nó defeituoso.

2.4.6.11 Recursos do Kubernetes descartados

Itens de verificação

Verifique se há recursos descartados nos clusters.

Solução

Cenário 1: o objeto de recurso PodSecurityPolicy foi descartado desde clusters de 1.25.

Log de erro:

```
some check failed in cluster upgrade: PodSecurityPolicy (PSP)API has been removed
in k8s 1.25 and uc observed that your cluster has PSP obiects: [ psp-global psp-
system], you need to check if these PSP objects should be removed
```

Execute o comando **kubectl get psp -A** no cluster para obter o objeto PSP existente.

Se esses dois objetos não forem usados, pule a verificação. Caso contrário, atualize as funções correspondentes para PodSecurity consultando **Segurança de pod**.

Cenário 2: o Serviço nos clusters de 1.25 ou posterior descartou a anotação: tolerate-unready-endpoints.

Log de erro:

```
some check failed in cluster upgrade: this cluster has deprecated service list:
map[***] with deprecated annotation list [tolerate-unready-endpoints]
```


Verifique se o Serviço fornecido nas informações de log contém a anotação de **tolerate-unready-endpoints**. Em caso afirmativo, substitua a anotação pelos seguintes campos:

```
publishNotReadyAddresses: true
```

2.4.6.12 Riscos de compatibilidade

Itens de verificação

Leia as diferenças de compatibilidade de versão e certifique-se de que elas não sejam afetadas. A atualização de patch não envolve diferenças de compatibilidade de versão.

Compatibilidade de versões

Caminho de atualização	Diferença entre versões	Auto-verificação
v1.19 a v1.21	O bug de **exec probe timeouts** foi corrigido no Kubernetes 1.21. Antes que esse bug seja corrigido, a sonda exec não considera o campo **timeoutSeconds**. Em vez disso, a sonda será executada indefinidamente, mesmo após o data limite configurado. Ela irá parar até que o resultado seja retornado. Se este campo não for especificado, o valor padrão **1** será usado. Este campo entra em vigor após a atualização. Se a sonda for executada por mais de 1 segundo, a verificação de integridade da aplicação poderá falhar e a aplicação poderá ser reiniciada com frequência.	Antes do upgrade, verifique se o tempo limite está configurado corretamente para a sonda exec.

Caminho de atualização	Diferença entre versões	Auto-verificação
	O kube-apiserver do CCE 1.19 ou posterior requer que o campo Subject Alternative Names (SANs) esteja configurado para o certificado do servidor webhook. Caso contrário, kube-apiserver falha ao chamar o servidor webhook após a atualização, e os contêineres não podem ser iniciados corretamente. Causa raiz: X.509 **CommonName** é descartado em Go 1.15. kube-apiserver do CCE 1.19 é compilado usando Go 1.15. Se o seu certificado webhook não tiver SANs, kube-apiserver não processa o campo **CommonName** do certificado X.509 como o nome de host por padrão. Como resultado, a autenticação falha.	Antes da atualização, verifique se o campo SAN está configurado no certificado do servidor webhook. ● Se você não tem seu próprio servidor webhook, você pode pular esta verificação. ● Se o campo não estiver definido, é aconselhável usar o campo SAN para especificar o endereço IP e o nome de domínio suportados pelo certificado.
v1.15 a v1.19	O plano de controle de clusters do CCE de v1.19 é incompatível com kubelet v1.15. Se um nó não for atualizado ou o nó a ser atualizado for reiniciado após o nó principal ser atualizado com sucesso, há uma alta probabilidade de que o nó esteja no status **NotReady**. Isso ocorre porque o nó falhou ao ser atualizado reinicia o kubelet e aciona o registro do nó. Em clusters de v1.15, as tags de registro padrão (**failure-domain.beta.kubernetes.io/is-baremetal** e **kubernetes.io/availablezone**) são consideradas tags inválidas pelo cluster da v1.19. As tags válidas nos clusters da v1.19 são **node.kubernetes.io/baremetal** e **failure-domain.beta.kubernetes.io/zone**.	1. Em casos normais, esse cenário não é acionado. 2. Depois que o nó principal for atualizado, não suspenda a atualização para que o nó possa ser atualizado rapidamente. 3. Se um nó não for atualizado e não puder ser restaurado, expulse aplicações no nó o mais rápido possível. Entre em contato com o suporte técnico e pule a atualização do nó. Após a conclusão da atualização, redefina o nó.

Caminho de atualização	Diferença entre versões	Auto-verificação
	Nos clusters do CCE 1.15 e 1.19, o sistema de arquivos do driver de armazenamento Docker é alterado de XFS para Ext4. Como resultado, a sequência de pacote de importação nos pods da aplicação Java atualizado pode ser anormal, causando exceções de pod.	Antes da atualização, verifique o arquivo de configuração do Docker **/etc/docker/daemon.json** no nó. Verifique se o valor de **dm.fs** é **xfs**. ● Se o valor for **ext4** ou o driver de armazenamento for Overlay, você poderá ignorar as próximas etapas. ● Se o valor for **xfs**, é aconselhável implementar aplicações no cluster da nova versão com antecedência para testar se os aplicativos são compatíveis com a nova versão do cluster. `{ "storage-driver": "devicemapper", "storage-opts": [ "dm.thinpooldev=/dev/mapper/vgpaas-thinpool", "dm.use_deferred_removal=true", "dm.fs=xfs", "dm.use_deferred_deletion=true" ] }`
	O kube-apiserver do CCE 1.19 ou posterior requer que o campo Subject Alternative Names (SANs) esteja configurado para o certificado do servidor webhook. Caso contrário, kube-apiserver falha ao chamar o servidor webhook após a atualização, e os contêineres não podem ser iniciados corretamente. Causa raiz: X.509 **CommonName** é descartado em Go 1.15. kube-apiserver do CCE 1.19 é compilado usando Go 1.15. O campo **CommonName** é processado como o nome do host. Como resultado, a autenticação falha.	Antes da atualização, verifique se o campo SAN está configurado no certificado do servidor webhook. ● Se você não tem seu próprio servidor webhook, você pode pular esta verificação. ● Se o campo não estiver definido, é aconselhável usar o campo SAN para especificar o endereço IP e o nome de domínio suportados pelo certificado. **AVISO** Para mitigar o impacto das diferenças de versão na atualização do cluster, o CCE executa um processamento especial durante a atualização de 1.15 para 1.19 e ainda oferece suporte a certificados sem SANs. No entanto, nenhum processamento especial é necessário para atualizações subsequentes. Aconselha-se que retifique o seu certificado o mais rapidamente possível.

Caminho de atualização	Diferença entre versões	Auto-verificação
	Em clusters da v1.17.17 e posteriores, o CCE cria automaticamente políticas de segurança de pods (PSPs) para você, o que restringe a criação de pods com configurações inseguras, por exemplo, pods para os quais **net.core.somaxconn** sob um sysctl é configurado no contexto de segurança.	Após uma atualização, você pode permitir configurações inseguras do sistema, conforme necessário. Para mais detalhes, consulte **Configuração de uma política de segurança de pod**.
	Se initContainer ou Istio for usado na atualização in-loco de um cluster v1.15, preste atenção às seguintes restrições: No kubelet 1.16 e versões posteriores, as **classes de QoS** são diferentes daquelas em versões anteriores. No kubelet 1.15 e versões anteriores, somente contêineres em **spec.containers** são contados. No kubelet 1.16 e versões posteriores, os contêineres em **spec.containers** e **spec.initContainers** são contados. A classe de QoS de um pod mudará após a atualização. Como resultado, o contêiner no pod é reiniciado.	É aconselhável modificar a classe de QoS do contêiner de serviço antes da atualização para evitar esse problema. Para mais detalhes, consulte **Tabela 2-14**.
v1.13 a v1.15	Depois que um cluster de rede da VPC é atualizado, o nó principal ocupa um bloco CIDR extra devido à atualização dos componentes de rede. Se nenhum bloco CIDR de contêiner estiver disponível para o novo nó, o pod agendado para o nó não poderá ser executado.	Geralmente, esse problema ocorre quando os nós no cluster estão prestes a ocupar totalmente o bloco CIDR do contêiner. Por exemplo, o bloco CIDR do contêiner é 10.0.0.0/16, o número de endereços IP disponíveis é 65.536 e a rede da VPC recebe um bloco CIDR com o tamanho fixo (usando a máscara para determinar o número máximo de endereços IP de contêiner alocados para cada nó). Se o limite superior for 128, o cluster suportará um máximo de 512 (65536/128) nós, incluindo os três nós principais. Depois que o cluster é atualizado, cada um dos três nós principais ocupa um bloco CIDR. Como resultado, 506 nós são suportados.

Tabela 2-18 Mudanças de classe de QoS antes e depois da atualização

Init contêiner (calculado com base em spec.initContainers)	Contêiner de serviço (calculado com base em spec.containers)	Pod (calculado com base em spec.containers e spec.initContainers)	Impactado ou não
Guaranteed	Besteffort	Burstable	Sim
Guaranteed	Burstable	Burstable	Não
Guaranteed	Guaranteed	Guaranteed	Não
Besteffort	Besteffort	Besteffort	Não
Besteffort	Burstable	Burstable	Não
Besteffort	Guaranteed	Burstable	Sim
Burstable	Besteffort	Burstable	Sim
Burstable	Burstable	Burstable	Não
Burstable	Guaranteed	Burstable	Sim

2.4.6.13 Versões do agente do CCE de nó

Itens de verificação

Verifique se o cce-agent no nó atual é da versão mais recente.

Solução

- **Cenário 1: a mensagem de erro "you cce-agent no update, please restart it" é indicada.**

  O cce-agent não precisa ser atualizado, mas não é reiniciado. Nesse caso, faça logon no nó e reinicie manualmente o cce-agent.

  Solução: faça logon no nó e execute o seguinte comando:

  ```
  systemctl restart cce-agent
  ```

  Execute a verificação de pré-atualização novamente.

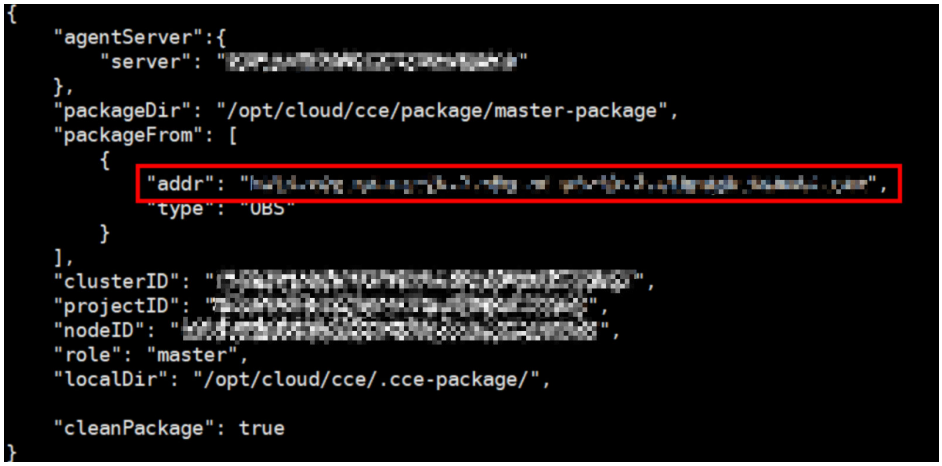
- **Cenário 2: a mensagem de erro "your cce-agent is not the latest version" é exibida.**

  O cce-agent não é da versão mais recente e a atualização automática falhou. Esse problema geralmente é causado por um caminho inválido do OBS ou a versão do componente está desatualizada.

  Solução

  a. Efetue logon em um nó em que a verificação tenha sido bem-sucedida, obtenha o caminho do arquivo de configuração de cce-agent e obtenha o endereço do OBS.
  ```
  cat `ps aux | grep cce-agent | grep -v grep | awk -F '-f ' '{print $2}'`
  ```
  O campo de endereço de configuração do OBS no arquivo de configuração é **packageFrom.addr**.

Figura 2-29 Endereço do OBS



b. Efetue logon em um local em que a verificação falhou, obtenha o endereço do OBS novamente consultando a etapa anterior e verifique se os endereços do OBS são os mesmos. Se eles forem diferentes, altere o endereço OBS do nó anormal para o endereço correto.

c. Execute os seguintes comandos para baixar o arquivo binário mais recente:

- x86
  ```
  curl -k "https://{OBS address you have obtained}/cluster-versions/
  base/cce-agent" > /tmp/cce-agent
  ```
- Arm
  ```
  curl -k "https://{OBS address you have obtained}/cluster-versions/
  base/cce-agent-arm" > /tmp/cce-agent-arm
  ```

d. Substitua o arquivo binário cce-agent original.

- x86
  ```
  mv -f /tmp/cce-agent /usr/local/bin/cce-agent
  chmod 750 /usr/local/bin/cce-agent
  chown root:root /usr/local/bin/cce-agent
  ```
- Arm
  ```
  mv -f /tmp/cce-agent-arm /usr/local/bin/cce-agent-arm
  chmod 750 /usr/local/bin/cce-agent-arm
  chown root:root /usr/local/bin/cce-agent-arm
  ```

e. Reinicie o cce-agent.
```
systemctl restart cce-agent
```

Se você tiver alguma dúvida sobre as operações anteriores, entre em contato com o suporte técnico.

2.4.6.14 Uso da CPU do nó

Itens de verificação

Verifique se o uso da CPU do nó excede 90%.

Solução

- **Atualize o cluster durante as horas de pico.**
- Verifique se muitos pods estão implementados no nó. Em caso afirmativo, reagende os pods para outros nós ociosos.

2.4.6.15 CRDs

Itens de verificação

Verifique os seguintes itens:

- Verifique se um CRD principal **packageversions.version.cce.io** é excluído.
- Verifique se o CRD principal do cluster **network-attachment-definitions.k8s.cni.cncf.io** é excluído.

Solução

Se os resultados da verificação forem anormais, entre em contato com o suporte técnico.

2.4.6.16 Discos de nó

Itens de verificação

Verifique os seguintes itens:

- Verifique se os discos de dados principais no nó atendem aos requisitos de atualização.
- Verifique se o diretório **/tmp** tem 500 MB de espaço disponível.

Solução

Durante a atualização do nó, os discos principais armazenam o pacote do componente de atualização e o diretório **/tmp** armazena arquivos temporários.

- **Cenário 1: cerificar se o disco atende aos requisitos de atualização.**

  Execute o seguinte comando para verificar o uso de cada disco principais. Depois de garantir que o espaço disponível atenda aos requisitos e verifique novamente. Se o espaço do nó principal for insuficiente, entre em contato com o suporte técnico.

  – Partição de disco do Docker: 2 GB para nós principais e 1 GB para nós de trabalho
  ```
  df -h /var/lib/docker
  ```
  – Partição de disco do containerd: 2 GB para nós principais e 1 GB para nós de trabalho
  ```
  df -h /var/lib/containerd
  ```
  – Partição de disco do kubelet: 2 GB para nós principais e 1 GB para nós de trabalho
  ```
  df -h /mnt/paas/kubernetes/kubelet
  ```
  – Disco do sistema: 10 GB para nós principais e 2 GB para nós de trabalho
  ```
  df -h /
  ```

- **Cenário 2: o espaço do diretório /tmp é insuficiente.**

  Execute o seguinte comando para verificar o uso do sistema de arquivos onde o diretório **/tmp** está localizado. Verifique se o espaço é maior que 500 MB e verifique novamente.

  ```
  df -h /tmp
  ```


2.4.6.17 DNS do nó

Itens de verificação

Verifique os seguintes itens:

- Verifique se a configuração de DNS do nó atual pode resolver o endereço do OBS.
- Verifique se o nó atual pode acessar o endereço do OBS do pacote de componente de atualização de armazenamento.

Solução

Durante a atualização do nó, obtenha o pacote do componente de atualização do OBS. Se essa verificação falhar, entre em contato com o suporte técnico.

2.4.6.18 Permissões de arquivo de diretório principal de nó

Itens de verificação

Verifique se o diretório principal **/var/paas** nos nós contém arquivos com proprietários anormais ou grupos de proprietários.

Solução

- **Cenário 1: a mensagem de erro "xx file permission has been changed!" é exibida.**

  Solução: habilite o CCE para usar o diretório **/var/paas** para gerenciar nós e armazenar dados de arquivos cujo proprietário e grupo proprietário são ambos **paas**.

  Durante a atualização do cluster atual, o proprietário e o grupo de proprietários dos arquivos no diretório **/var/paas** são redefinidos para paas.

  Verifique se os dados do arquivo estão armazenados no diretório **/var/paas**. Se sim, não use este diretório, remova arquivos anormais deste diretório e verifique novamente. Caso contrário, a atualização é proibida.

- **Cenário 2: a mensagem de erro "user paas must have at least read and execute permissions on the root directory" é exibida.**

  Solução: altere a permissão no diretório raiz para a permissão padrão 555. Se a permissão no diretório raiz do nó for modificada, o usuário **paas** não terá a permissão de leitura no diretório raiz. Como resultado, a reinicialização do componente falhou durante a atualização.

2.4.6.19 Kubelet

Itens de verificação

Verifique se o kubelet no nó está sendo executado corretamente.

Solução

- **Cenário 1: o status do kubelet é anormal.**

  Se o kubelet funcionar mal, o nó não estará disponível. Restaure o nó e verifique novamente. Para obter detalhes, consulte **O que devo fazer se um cluster estiver disponível, mas alguns nós não estiverem disponíveis?**

- **Cenário 2: a versão de cce-pause está incorreta.**

  A versão da imagem do contêiner de pausa da qual o kubelet depende não é cce-pause:3.1. Se você continuar a atualização, os pods serão reiniciados em lotes. Atualmente, a atualização não é suportada. Entre em contato com o suporte técnico.

2.4.6.20 Memória do nó

Itens de verificação

Verifique se o uso da memória do nó excede 90%.

Solução

- **Atualize o cluster durante as horas de pico.**
- Verifique se muitos pods estão implementados no nó. Em caso afirmativo, reagende os pods para outros nós ociosos.

2.4.6.21 Servidor de sincronização de relógio de nó

Itens de verificação

Verifique se o servidor de sincronização de relógio ntpd ou chronyd do nó está sendo executado corretamente.

Solução

- **Cenário 1: ntpd está sendo executado anormalmente.**

  Faça logon no nó e execute o comando **systemctl status ntpd** para obter o status de execução de ntpd. Se a saída do comando for anormal, execute o comando **systemctl restart ntpd** e obtenha o status novamente.

  A saída normal do comando é a seguinte:

  Figura 2-30 Status de execução do ntpd

  ```
  [root@[redacted] paas]# systemctl status ntpd
  ● ntpd.service - Network Time Service
     Loaded: loaded (/usr/lib/systemd/system/ntpd.service; enabled; vendor preset: disabled)
     Active: active (running) since Tue 2022-12-06 14:52:30 CST; 4 days ago
   Main PID: 8587 (ntpd)
      Tasks: 2
     Memory: 1.6M
     CGroup: /system.slice/ntpd.service
             └─8587 /usr/sbin/ntpd -u ntp:ntp -g -x
  ```

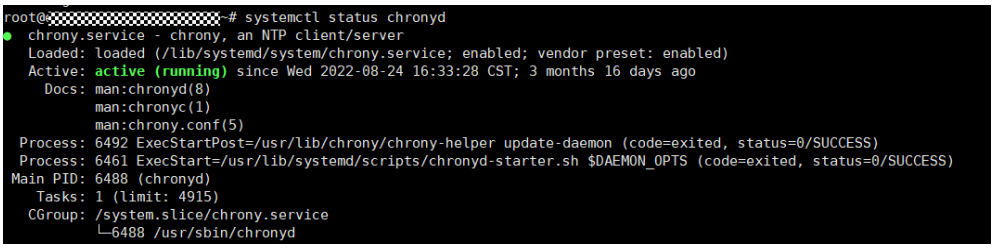
  Se o problema persistir após a reinicialização de ntpd, entre em contato com o suporte técnico.

- **Cenário 2: chronyd está sendo executado anormalmente.**

  Efetue logon no nó e execute o comando **systemctl status chronyd** para obter o status de execução de chronyd. Se a saída do comando for anormal, execute o comando **systemctl restart chronyd** e obtenha o status novamente.

  A saída normal do comando é a seguinte:

Figura 2-31 Status de execução de chronyd



Se o problema persistir após a reinicialização de chronyd, entre em contato com o suporte técnico.

2.4.6.22 Sistema operacional do nó

Itens de verificação

Verifique se a versão do kernel do sistema operacional do nó é suportada pelo CCE.

Solução

Os nós em execução dependem da versão padrão inicial do kernel quando são criados. O CCE realizou testes de compatibilidade abrangentes com base nesta versão do kernel. Uma versão de kernel não padrão pode causar problemas de compatibilidade inesperados durante a atualização do nó e a atualização do nó pode falhar. Para mais detalhes, consulte **Operações de alto risco e soluções**.

Este tipo de nós não deve ser atualizado. Redefina o nó para a versão padrão do kernel antes da atualização seguindo as instruções em **Redefinição de um nó**.

2.4.6.23 CPUs do nó

Itens de verificação

Verifique se o número de CPUs no nó principal é maior que 2.

Solução

Se o número de CPUs no nó principal for 2, entre em contato com o suporte técnico para expandir o número para 4 ou mais.

2.4.6.24 Comandos Python do nó

Itens de verificação

Verifique se os comandos Python estão disponíveis em um nó.

Método de verificação

```
/usr/bin/python --version
echo $?
```

Se a saída do comando não for 0, a verificação falhará.

Solução

Instale Python antes da atualização.

2.4.6.25 Versão do ASM

Itens de verificação

Verifique os seguintes itens:

- Verifique se o ASM é usado pelo cluster.
- Verifique se a versão atual do ASM oferece suporte à versão do cluster de destino.

Solução

- Atualize o ASM e, em seguida, atualize o cluster. As regras de adaptação entre as versões do ASM e do cluster são as seguintes:

  Tabela 2-19 Regras de adaptação entre versões do ASM e do cluster

Versão do ASM	Versão do cluster
1.3	v1.13, v1.15, v1.17 ou v1.19
1.6	v1.15, v1.17, v1.19 ou v1.21
1.8	v1.15, v1.17, v1.19 ou v1.21
1.13	v1.21 ou v1.23
1.15	v1.21, v1.23 ou v1.25

- Se o ASM não for necessário, exclua-o antes da atualização. Após o upgrade, o cluster não pode ser vinculado ao ASM que não corresponde à tabela. Por exemplo, se você deseja atualizar um cluster de v1.21 e ASM de v1.8 para v1.23, atualize o ASM primeiro.

2.4.6.26 Prontidão do nó

Itens de verificação

Verifique se os nós no cluster estão prontos.

Solução

- **Cenário 1: os nós estão no status indisponível.**

  Efetue logon no console do CCE e acesse o console do cluster. Escolha **Nodes** no painel de navegação e filtre os nós indisponíveis, retifique os nós defeituosos consultando as sugestões fornecidas pelo console e verifique novamente.

- **Cenário 2: o status do nó exibido é inconsistente com o status real.**

  As possíveis causas são as seguintes:

  a. O status do nó é normal na página de nós, mas o resultado da verificação mostra que o nó não está pronto. Verifique novamente.

b. O nó não é encontrado na página nós, mas o resultado da verificação mostra que o nó está no cluster. Entre em contato com o suporte técnico.

2.4.6.27 Nó journald

Itens de verificação

Verifique se o journald de um nó é normal.

Solução

Faça logon no nó e execute o comando **systemctl is-active systemd-journald** para obter o status de execução do journald. Se a saída do comando for anormal, execute o comando **systemctl restart systemd-journald** e obtenha o status novamente.

A saída normal do comando é a seguinte:

Figura 2-32 Status de execução do journald

```
[root@ paas]# systemctl is-active systemd-journald
active
```

Se o problema persistir depois que o journald for reiniciado, entre em contato com o suporte técnico.

2.4.6.28 containerd.sock

Itens de verificação

Verifique se o arquivo containerd.sock existe no nó. Esse arquivo afeta a inicialização do tempo de execução do container no Euler OS.

Solução

Cenário: o Docker usado pelo nó é o Euler-docker personalizado.

Passo 1 Efetue logon no nó.

Passo 2 Execute o comando **rpm -qa | grep docker | grep euleros**. Se a saída do comando não estiver vazia, o Docker usado no nó é Euler-docker.

Passo 3 Execute o comando **ls /run/containerd/containerd.sock**. Se o arquivo existir, a inicialização do Docker falhará.

Passo 4 Execute o comando **rm -rf /run/containerd/containerd.sock** e execute a verificação de atualização de cluster novamente.

----Fim

2.4.6.29 Erros internos

Itens de verificação

Antes da atualização, verifique se ocorre um erro interno.

Solução

Se essa verificação falhar, entre em contato com o suporte técnico.

2.4.6.30 Pontos de montagem do nó

Itens de verificação

Verifique se existem pontos de montagem inacessíveis no nó.

Solução

Cenário: existem pontos de montagem inacessíveis no nó.

Se o NFS de rede (como OBS, SFS e SFS) for usado pelo nó e o nó for desconectado do servidor de NFS, o ponto de montagem ficará inacessível e todos os processos que acessarem esse ponto de montagem serão suspensos.

Passo 1 Efetue logon no nó.

Passo 2 Execute os seguintes comandos no nó em sequência:

```
- df -h
- for dir in `df -h | grep -v "Mounted on" | awk "{print \\$NF}"`;do cd $dir;
done && echo "ok"
```

Passo 3 Se **ok** for retornado, não ocorrerá nenhum problema.

Caso contrário, inicie outro terminal e execute o seguinte comando para verificar se o comando anterior está no estado D:

```
- ps aux | grep "D "
```

Passo 4 Se um processo estiver no estado D, o problema ocorre. Você só pode redefinir o nó para resolver o problema. Redefina o nó e atualize o cluster novamente. Para obter detalhes sobre como redefinir um nó, consulte**Redefinição de um nó**.

NOTA

A redefinição de um nó redefinirá todos os rótulos de nó, o que pode afetar o agendamento da carga de trabalho. Antes de redefinir um nó, verifique e retenha os rótulos que você adicionou manualmente ao nó.

----Fim

2.4.6.31 Manchas de nós do Kubernetes

Itens de verificação

Verifique se a mancha necessário para a atualização do cluster existe no nó.

Tabela 2-20 Lista de verificação de mancha

Nome da mancha	Impacto
node.kubernetes.io/upgrade	NoSchedule

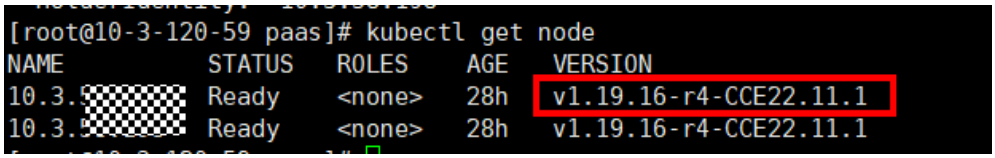
Solução

Cenário 1: o nó é ignorado durante a atualização do cluster.

Passo 1 Para detalhes sobre como configurar o kubectl, veja **Conexão a um cluster usando o kubectl**.

Passo 2 Verifique a versão do kubelet do nó correspondente. As seguintes informações são esperadas:

Figura 2-33 Versão de kubelet



Se a versão do nó for diferente da de outros nós, o nó será ignorado durante a atualização. Redefina o nó e atualize o cluster novamente. Para obter detalhes sobre como redefinir um nó, consulte **Redefinição de um nó**.

NOTA

A redefinição de um nó redefinirá todos os rótulos de nó, o que pode afetar o agendamento da carga de trabalho. Antes de redefinir um nó, verifique e retenha os rótulos que você adicionou manualmente ao nó.

----Fim

2.4.6.32 Restrições de everest

Itens de verificação

Verifique se há restrições de compatibilidade no complemento everest atual.

Tabela 2-21 Lista de versões complemento everest com restrições de compatibilidade

Nome do complemento	Versões envolvidas
everest	v1.0.2-v1.0.7 v1.1.1-v1.1.5

Solução

Há restrições de compatibilidade no complemento everest atual e ele não pode ser atualizado com a atualização do cluster. Entre em contato com o suporte técnico.

2.4.6.33 Restrições de cce-hpa-controller

Itens de verificação

Verifique se o atual complemento cce-controller-hpa tem restrições de compatibilidade.

Solução

O atual complemento cce-controller-hpa tem restrições de compatibilidade. Um complemento que pode fornecer APIs de métrica, por exemplo, metric-server, deve ser instalado no cluster.

2.4.6.34 Políticas de CPU aprimorada

Itens de verificação

Verifique se a versão atual do cluster e a versão de destino oferecem suporte à **política de CPU aprimorada**.

Solução

Cenário: somente a versão atual do cluster suporta a função de política de CPU aprimorada. A versão de destino não suporta a função de política de CPU aprimorada.

Atualize para uma versão de cluster que suporte a função de política de CPU aprimorada. A tabela a seguir lista as versões de cluster que oferecem suporte à função de política de CPU aprimorada.

Tabela 2-22 Lista de versões de cluster que suportam a função de política de CPU aprimorada

Versão do cluster	Política de CPU aprimorada
Clusters da v1.17 ou anterior	Não compatível
Clusters da v1.19	Não compatível
Clusters da v1.21	Não compatível
Clusters da v1.23 e posterior	Compatível

2.4.6.35 Integridade dos componentes do nó de trabalho

Itens de verificação

Verifique se o tempo de execução do contêiner e os componentes de rede nos nós de trabalho estão íntegros.

Solução

Se um componente de nó de trabalho funcionar mal, faça logon no nó para verificar o status do componente e corrigir a falha.

2.4.6.36 Integridade dos componentes do nó principal

Itens de verificação

Verifique se o Kubernetes, o tempo de execução do contêiner e os componentes de rede dos nós principais estão íntegros.

Solução

Se um componente de nó principal funcionar mal, entre em contato com o suporte técnico.

2.4.6.37 Limite de recursos de memória dos componentes do Kubernetes

Itens de verificação

Verifique se os recursos dos componentes do Kubernetes, como etcd e kube-controller-manager, excedem o limite superior.

Solução

- Solução 1: reduzir os recursos do Kubernetes necessários.
- Solução 2: estender as especificações do cluster. Para mais detalhes, consulte **Alteração de escala do cluster**.

2.4.6.38 APIs do Kubernetes descartadas

Itens de verificação

O sistema verifica os logs de auditoria do dia anterior para verificar se o usuário chama as APIs preteridas da versão de destino do Kubernetes.

NOTA

Devido ao intervalo de tempo limitado dos logs de auditoria, esse item de verificação é apenas um método auxiliar. As APIs a serem preteridas podem ter sido usadas no cluster, mas seu uso não está incluído nos logs de auditoria do dia anterior. Verifique o uso da API com cuidado.

Solução

Verificar descrição

Com base no resultado da verificação, é detectado que seu cluster chama uma API preterida da versão do cluster de destino usando kubectl ou outras aplicações. Você pode corrigir a falha antes da atualização. Caso contrário, a API será interceptada pelo kube-apiserver após a atualização. Para obter detalhes sobre cada API preterida, consulte **APIs preteridas**.

Estudo de caso

Ingresses das APIs extensions/v1beta1 e networking.k8s.io/v1beta1 estão preteridos no Kubernetes v1.22. Se você atualizar um cluster de v1.19 ou v1.21 para v1.23, os recursos existentes não serão afetados, mas a API v1beta1 poderá ser interceptada nos cenários de criação e edição.

Para obter detalhes sobre as alterações na estrutura de configuração do YAML, consulte **Uso do kubectl para criar um ingress do ELB**.

2.4.6.39 Capacidades de IPv6 de um cluster do CCE Turbo

Itens de verificação

Se o IPv6 é permitido para um cluster do CCE Turbo, verifique se a versão do conjunto de destino apoia o IPv6.

Solução

Os clusters do CCE Turbo suportam IPv6 desde v1.23. Esse recurso está disponível nas seguintes versões:

- v1.23: 1.23.8-r0 ou mais recente
- v1.25: 1.25.3-r0 ou mais recente
- v1.25 ou mais recente

Se o IPv6 tiver sido habilitado no cluster antes da atualização, a versão do cluster de destino também deverá oferecer suporte ao IPv6. Selecione uma versão de cluster adequada.

2.4.6.40 NetworkManager de nó

Itens de verificação

Verifique se o NetworkManager de um nó é normal.

Solução

Efetue logon no nó e execute o comando **systemctl is-active NetworkManager** para obter o status de execução do NetworkManager. Se a saída do comando for anormal, execute o comando **systemctl restart NetworkManager** e obtenha o status novamente.

Se o problema persistir depois que o NetworkManager for reiniciado, entre em contato com o suporte técnico.

2.4.6.41 Arquivo de ID do nó

Itens de verificação

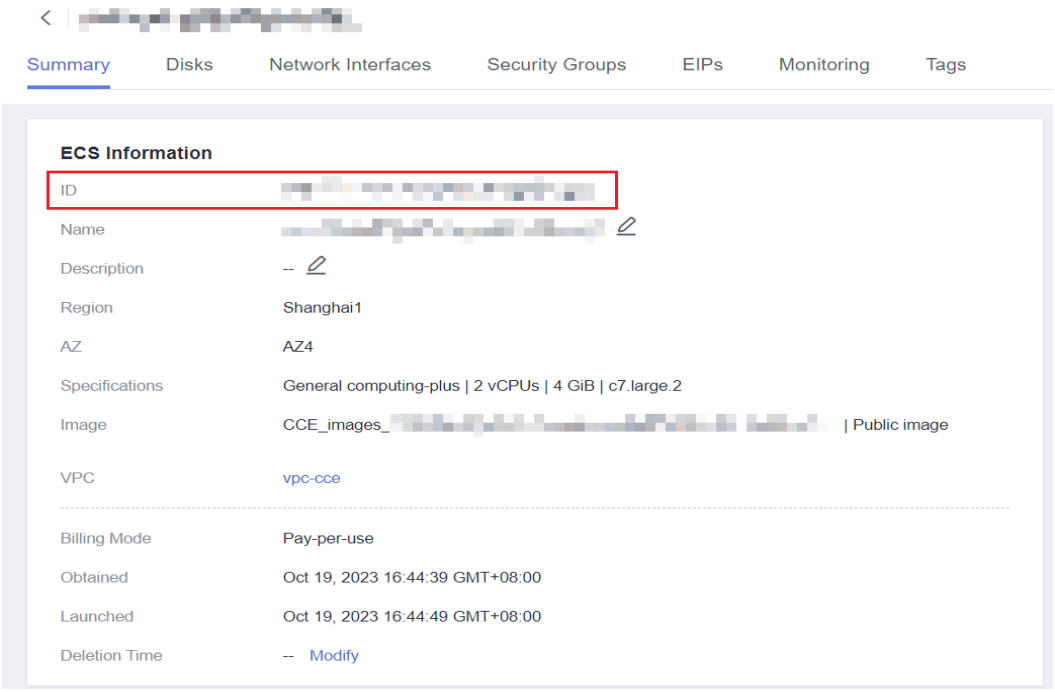
Verifique o formato do arquivo de ID.

Solução

Passo 1 Na página **Nodes** do console do CCE, clique no nome do nó anormal para ir para a página do ECS.

Passo 2 Copie o ID do nó e salve-o no host local.

Figura 2-34 Copiar um ID de nó



Passo 3 Efetue logon no nó anormal e faça backup dos arquivos.

```
cp /var/lib/cloud/data/instance-id /tmp/instance-id
cp /var/paas/conf/server.conf /tmp/server.conf
```

Passo 4 Efetue logon no nó anormal e grave o ID do nó obtido no arquivo.

```
echo "Node ID" > /var/lib/cloud/data/instance-id
echo "Node ID" > /var/paas/conf/server.conf
```

----Fim

2.4.6.42 Consistência da configuração do nó

Itens de verificação

Quando você atualiza um cluster para v1.19 ou posterior, o sistema verifica se os seguintes arquivos de configuração foram modificados no back-end:

- /opt/cloud/cce/kubernetes/kubelet/kubelet
- /opt/cloud/cce/kubernetes/kubelet/kubelet_config.yaml
- /opt/cloud/cce/kubernetes/kube-proxy/kube-proxy
- /etc/containerd/default_runtime_spec.json
- /etc/sysconfig/docker
- /etc/default/docker
- /etc/docker/daemon.json

Se você modificar alguns parâmetros nesses arquivos, a atualização de cluster pode falhar ou serviços podem ser anormais após a atualização. Se você confirmar que a modificação não afeta os serviços, continue a atualização.

> **NOTA**
>
> O CCE usa o script de imagem padrão para verificar a consistência da configuração do nó. Se você usar outras imagens personalizadas, a verificação pode falhar.

A modificação esperada não será interceptada. A tabela a seguir lista os parâmetros que podem ser modificados.

Tabela 2-23 Parâmetros que podem ser modificados

Componente	Arquivo de configuração	Parâmetro	Upgrade Version
kubelet	/opt/cloud/cce/kubernetes/kubelet/kubelet_config.yaml	cpuManagerPolicy	Mais recente de que v1.19
kubelet	/opt/cloud/cce/kubernetes/kubelet/kubelet_config.yaml	maxPods	Mais recente de que v1.19
kubelet	/opt/cloud/cce/kubernetes/kubelet/kubelet_config.yaml	kubeAPIQPS	Mais recente de que v1.19
kubelet	/opt/cloud/cce/kubernetes/kubelet/kubelet_config.yaml	kubeAPIBurst	Mais recente de que v1.19
kubelet	/opt/cloud/cce/kubernetes/kubelet/kubelet_config.yaml	podPidsLimit	Mais recente de que v1.19
kubelet	/opt/cloud/cce/kubernetes/kubelet/kubelet_config.yaml	topologyManager-Policy	Mais recente de que v1.19
kubelet	/opt/cloud/cce/kubernetes/kubelet/kubelet_config.yaml	resolvConf	Mais recente de que v1.19
kubelet	/opt/cloud/cce/kubernetes/kubelet/kubelet_config.yaml	eventRecordQPS	Mais recente de que v1.21
kubelet	/opt/cloud/cce/kubernetes/kubelet/kubelet_config.yaml	topologyManager-Scope	Mais recente de que v1.21
kubelet	/opt/cloud/cce/kubernetes/kubelet/kubelet_config.yaml	allowedUnsafeSysctls	Mais recente de que v1.19
Docker	/etc/docker/daemon.json	dm.basesize	Mais recente de que v1.19

Solução

Se você modificar alguns parâmetros nesses arquivos, exceções podem ocorrer após a atualização. Se você não tiver certeza se os parâmetros modificados afetarão a atualização, entre em contato com o suporte técnico.

2.4.6.43 Arquivo da configuração de nó

Itens de verificação

Verifique se os arquivos de configuração dos componentes-chave existem no nó.

A tabela a seguir lista os arquivos a serem verificados.

Nome do arquivo	Conteúdo do arquivo	Observações
/opt/cloud/cce/kubernetes/kubelet/kubelet	Parâmetros de inicialização da linha de comando do kubelet	Nenhuma
/opt/cloud/cce/kubernetes/kubelet/kubelet_config.yaml	Parâmetros de inicialização do kubelet	Nenhuma
/opt/cloud/cce/kubernetes/kube-proxy/kube-proxy	Parâmetros de inicialização da linha de comando do kube-proxy	Nenhuma
/etc/sysconfig/docker	Arquivo de configuração do Docker	Não é verificado quando a máquina de containerd ou Debain-Group é usada.
/etc/default/docker	Arquivo de configuração do Docker	Não é verificado quando a máquina de containerd ou Centos-Group é usada.

Solução

Entre em contato com o suporte técnico para restaurar o arquivo de configuração e, em seguida, executar a atualização.

2.4.6.44 Consistência da configuração de CoreDNS

Itens de verificação

Verifique se o Corefile de configuração principal de CoreDNS atual é diferente do registro de lançamento do Helm. A diferença pode ser substituída durante a atualização do complemento, **afetando a resolução de nomes de domínio no cluster**.

Solução

Você pode atualizar o CoreDNS separadamente depois de confirmar as diferenças de configuração.

Passo 1 Configure o kubectl. Para mais detalhes, consulte **Conexão a um cluster usando o kubectl**.

Passo 2 Obtenha o Corefile que entra em vigor atualmente.

```
kubectl get cm -nkube-system coredns -o jsonpath='{.data.Corefile}' >
corefile_now.txt
cat corefile_now.txt
```

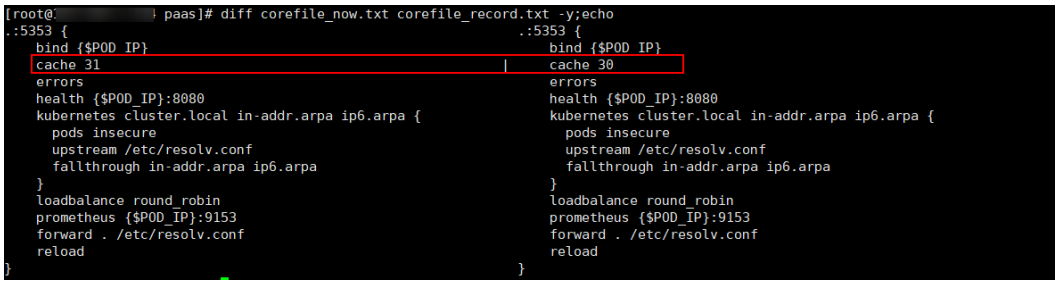
Passo 3 Obtenha o Corefile no registro de lançamento do Helm (dependendo do Python 3).

```
latest_release=`kubectl get secret -nkube-system -l owner=helm -l name=cceaddon-
coredns --sort-by=.metadata.creationTimestamp | awk 'END{print $1}'`
kubectl get secret -nkube-system $latest_release -o jsonpath='{.data.release}' |
base64 -d | base64 -d | gzip -d | python -m json.tool | python -c "
import json,sys,re,yaml;
manifests = json.load(sys.stdin)['manifest']
files = re.split('(?:^|\s*\n)---\s*',manifests)
for file in files:
    if 'coredns/templates/configmap.yaml' in file and 'Corefile' in file:
      corefile = yaml.safe_load(file)['data']['Corefile']
      print(corefile,end='')
      exit(0);
print('error')
exit(1);
" > corefile_record.txt
cat corefile_record.txt
```

Passo 4 Compare as diferenças de saída entre **Passo 2** e **Passo 3**.

```
diff corefile_now.txt corefile_record.txt -y;
```

Figura 2-35 Visualizar diferenças de saída



Passo 5 Volte para console do CCE e clique no nome do cluster para acessar o console do cluster. Escolha **Add-ons** no painel de navegação, selecione CoreDNS e clique em **Edit**.

Para manter configurações personalizadas, use um dos seguintes métodos:

- Defina **parameterSyncStrategy** para **force**. Insira manualmente a configuração diferencial. Para mais detalhes, consulte **CoreDNS**.
- Se **parameterSyncStrategy** estiver definido para **inherit**, as configurações personalizadas serão automaticamente herdadas. O sistema analisa, identifica e herda automaticamente parâmetros personalizados.

Passo 6 Clique em **OK**. Após a conclusão da atualização do complemento, verifique se todas as instâncias do CoreDNS estão disponíveis e se o Corefile atende à expectativa.

```
kubectl get cm -nkube-system coredns -o jsonpath='{.data.Corefile}'
```

Passo 7 Altere o valor do **parameterSyncStrategy** para **ensureConsistent** para ativar a verificação de consistência da configuração.

Além disso, é uma boa prática usar a função de configuração de parâmetros dos complementos do CCE para modificar a configuração do Corefile para obter consistência.

----Fim

2.4.6.45 Comandos sudo de um nó

Itens de verificação

Se os comandos sudo e arquivos relacionados ao sudo do nó estão funcionando

Solução

- Cenário 1: o comando sudo falha ao ser executado.

  Durante a atualização de cluster no local, o comando sudo deve estar disponível. Faça logon no nó e execute o seguinte comando para verificar se o comando sudo está disponível:

  ```
  sudo echo hello
  ```

- Cenário 2: arquivos de chave não podem ser modificados.

  Durante a atualização do cluster in-loco, os arquivos **/etc/sudoers** e **/etc/sudoers.d/sudoerspaas** são modificados para obter a permissão sudo e atualizar os componentes (como Docker e kubelet), cujo proprietário e grupo são arquivos de configuração **root** e relacionados no nó. Faça logon no nó e execute o seguinte comando para verificar se o arquivo pode ser modificado:

  ```
  lsattr -l /etc/sudoers.d/sudoerspaas /etc/sudoers
  ```

  Se **immutable** for exibido na saída do comando, o arquivo será bloqueado pelo bloqueio **i** e não poderá ser modificado. É aconselhável remover o bloqueio **i**.

  ```
  chattr -i /etc/sudoers.d/sudoerspaas /etc/sudoers
  ```

2.4.6.46 Comandos principais dos nós

Itens de verificação

Se alguns comandos principais dos quais a atualização do nó depende estão funcionando

Solução

- Cenário 1: o comando do gerenciador de pacotes falha ao ser executado.

  O comando **rpm** ou **dpkg** falha ao ser executado. Faça logon no nó e verifique se os seguintes comandos estão disponíveis:

  – rpm:
  ```
  rpm -qa
  ```
  – dpkg:
  ```
  dpkg -l
  ```

- Cenário 2: o comando **systemctl status** falha ao ser executado.

  Se o comando **systemctl status** em um nó não estiver disponível, muitos itens de verificação serão afetados. Efetue logon no nó e verifique a disponibilidade dos seguintes comandos:

  ```
  systemctl status kubelet
  ```


2.4.6.47 A montagem do arquivo sock em um nó

Itens de verificação

O arquivo **docker/containerd.sock** no nó é montado no pod por meio de um hostPath. Durante a atualização, o Docker/containerd é reiniciado, mas o arquivo sock no contêiner não é alterado. Como resultado, pode ocorrer um erro em seus serviços.

Solução

- Cenário 1: esse problema ocorreu em uma aplicação.

  Monte o arquivo sock montando um diretório.
- Cenário 2: esse problema ocorreu em alguns complementos do CCE de versões anteriores.

  Atualize os complementos do CCE para a versão mais recente. Por exemplo, se esse problema ocorreu no complemento Dolphin de versões anteriores a 1.2.2, atualize o complemento para 1.2.2 ou posterior.

2.4.6.48 Consistência do certificado do balanceador de carga de HTTPS

Itens de verificação

Verifique se o certificado usado por um balanceador de carga de HTTPS foi modificado no ELB.

Solução

Substitua o certificado usado por um ingresso por aquele usado pelo balanceador de carga. Em seguida, você pode criar ou editar o certificado no console do ELB.

Passo 1 Faça logon no console do ELB, escolha **Elastic Load Balance** > **Certificates**, localize o certificado e encontre o **secret_id** na descrição do certificado.

Figura 2-36 Visualizar um certificado

O **secret_id** é o **metadata.uid** do Segredo no cluster. Utilize este UID para obter o nome do Segredo no cluster.

Execute o seguinte comando kubectl para obter o nome do Segredo (substitua *<secret_id>* pelo valor real):

```
kubectl get secret --all-namespaces -o jsonpath='{range .items[*]}{"uid:"}
{.metadata.uid}{" namespace:"}{.metadata.namespace}{" name:"}{.metadata.name}
{"\n"}{end}' | grep <secret_id>
```

Passo 2 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster. Escolha **Services & Ingresses** no painel de navegação, clique a guia **Ingresses**, localize a linha que contém o ingress que usa o certificado, e escolha **More** > **Update** na coluna da **Operation**. Se vários Ingresses estiverem usando esse certificado, atualize o

certificado para todas essas Ingresses. Para verificar quais Ingresses estão usando um certificado, use o parâmetro **secertName** em **spec.tls** dos arquivos YAML de Ingress.

Execute o seguinte comando kubectl para obter os Ingresses usando um certificado (substitua <*secret_id*> pelo valor real):

```
kubectl get ingress --all-namespaces -o jsonpath='{range .items[*]}{"namespace:"}
{.metadata.namespace}{" name:"}{.metadata.name}{" tls:"}{.spec.tls[*]}{"\n"}
{end}' | grep <secret_name>
```

Passo 3 Ao configurar um ouvinte, selecione **ELB server certificate** para **Certificate Source** e clique em **OK**. Dessa forma, o certificado pode ser criado ou editado no console do ELB.

Passo 4 Na página **ConfigMaps and Secrets**, exclua o Segredo de destino. Antes da exclusão, faça backup dos dados.

----Fim

2.4.6.49 Montagem do nó

Itens de verificação

Verifique se o diretório de montagem padrão e o link suave no nó foram montados ou modificados manualmente.

- Disco não compartilhado
  - Por padrão, **/var/lib/docker**, **containerd** ou **/mnt/paas/kubernetes/kubelet** é montado em nós do CCE. Verifique se **/var**, **/var/lib**, **/mnt**, **/mnt/paas** e **/mnt/paas/kubernetes** foram montados manualmente.
  - O link soft de **/var/lib/kubelet** para **/mnt/paas/kubernetes/kubelet** é criado para CCE por padrão. Verifique se foi modificado manualmente.
- Disco compartilhado
  - Por padrão, /mnt/paas/ é montado em nós CCE. Verifique se o **/mnt** foi montado manualmente.
  - O link soft de **/var/lib/kubelet** para **/mnt/paas/kubernetes/kubelet**, ou **/var/lib/docker** ou **containerd** para **/mnt/paas/runtime** é criado para CCE por padrão. Verifique se os soft links foram modificados manualmente.

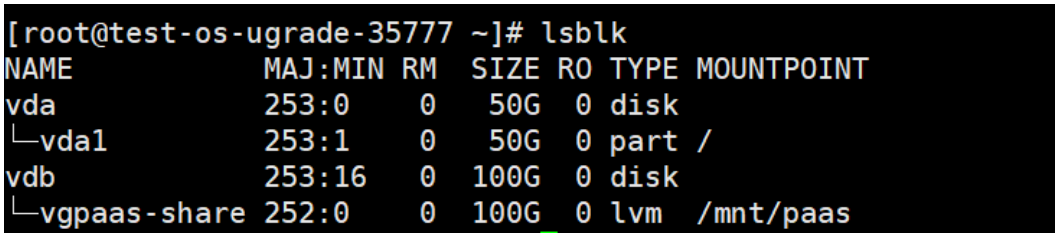
Solução

Como verificar se um disco é compartilhado?

Passo 1 Efetue logon no nó de destino com base nas informações de verificação.

Passo 2 Execute o comando **lsblk** para verificar se **vgpaas-share** está montado em **/mnt/paas**. Se sim, um disco compartilhado é usado.

Figura 2-37 Verificar se um disco compartilhado é usado



----Fim

O que fazer se um erro ocorreu em uma verificação de montagem do nó?

1. Cancele o ponto de montagem modificado manualmente.
2. Cancele a modificação no soft link padrão.

2.4.6.50 Permissões de logon de usuário paas em um nó

Itens de verificação

Verifique se o usuário **paas** tem permissão para fazer logon em um nó.

Solução

Execute o seguinte comando para verificar se o usuário**paas** tem permissão para fazer logon em um nó:

```
sudo grep "paas" /etc/passwd
```

Se as permissões atribuídas ao usuário **paas** contiverem **nologin** ou **false**, o usuário não terá a permissão de logon. Nesse caso, restaure a permissão de logon do usuário **paas**.

Execute o seguinte comando para restaurar a permissão de logon do usuário **paas**:

```
usermod -s /bin/bash paas
```

2.4.6.51 Endereços IPv4 privados de balanceadores de carga

Itens de verificação

Verifique se o balanceador de carga associado a um Serviço está alocado com um endereço IPv4 privado.

Solução

Solução 1: exclua o Serviço associado a um balanceador de carga sem um endereço IPv4 privado.

Solução 2: vincule um endereço IP privado ao balanceador de carga sem um endereço IPv4 privado. O procedimento é o seguinte:

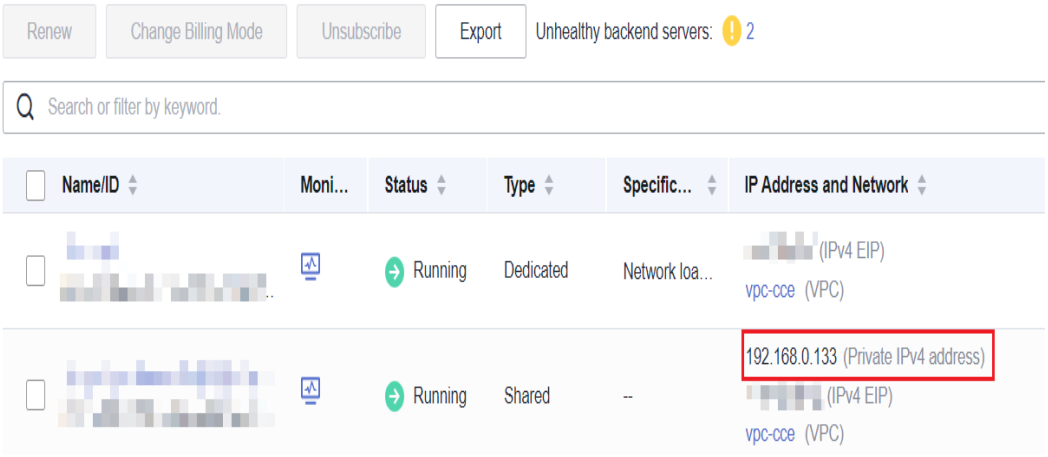
Passo 1 Obtenha o balanceador de carga associado ao Serviço de destino.

- Método 1: obtenha o ID do balanceador de carga com base no log de verificação de pré-atualização. Vá para o console do ELB e filtre os balanceadores de carga por ID do balanceador de carga.


```
elbs (ids: [*****]) without ipv4 private ip, please bind private ip tothese
elbs and try again
```

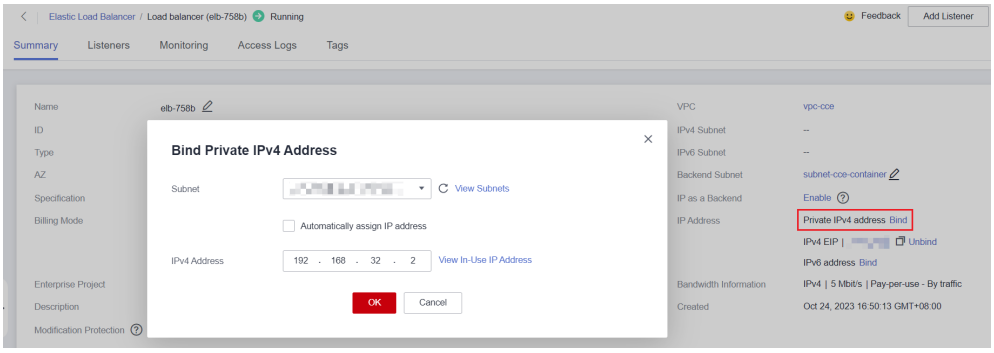
- Método 2: efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster. Em seguida, escolha **Services & Ingresses** no painel de navegação e clique no nome do balanceador de carga de destino para acessar a página do ELB.

Passo 2 Verifique se o balanceador de carga tem um endereço IPv4 privado.



Passo 3 Vincule um endereço IP privado ao balanceador de carga sem um endereço IPv4 privado.

1. Faça logon no console do CCE e clique no nome do balanceador de carga de destino.
2. Na guia **Summary**, clique em **Bind** ao lado de **Private IPv4 address**.
3. Configure a sub-rede e o endereço IPv4 e clique em **OK**.



----Fim

2.4.6.52 Registros históricos de atualização

Itens de verificação

Verifique se a versão de origem do cluster é anterior à v1.11 e se a versão de destino é posterior à v1.23.

Solução

Se a versão de origem do cluster for anterior à v1.11, é arriscado atualizar o cluster para uma versão posterior à v1.23. Nesse caso, entre em contato com o suporte técnico.

2.4.6.53 Bloco CIDR do plano de gerenciamento do cluster

Itens de verificação

Verifique se o bloco CIDR do plano de gerenciamento do cluster é o mesmo que o configurado na rede backbone.

Solução

Se o bloco CIDR do plano de gerenciamento de cluster for diferente daquele configurado na rede backbone, entre em contato com o suporte técnico.

2.4.6.54 Complemento da GPU

Itens de verificação

O complemento da GPU está envolvido na atualização, o que pode afetar a instalação do driver da GPU durante a criação de um nó da GPU.

Solução

O driver do complementar da GPU precisa ser configurado por você mesmo. Verifique a compatibilidade entre o complemento da GPU e o driver da GPU. É uma boa prática verificar a atualização do driver da GPU para a versão de destino no ambiente de teste, configurar o driver da GPU atual e verificar se o nó da GPU criado pode ser executado corretamente.

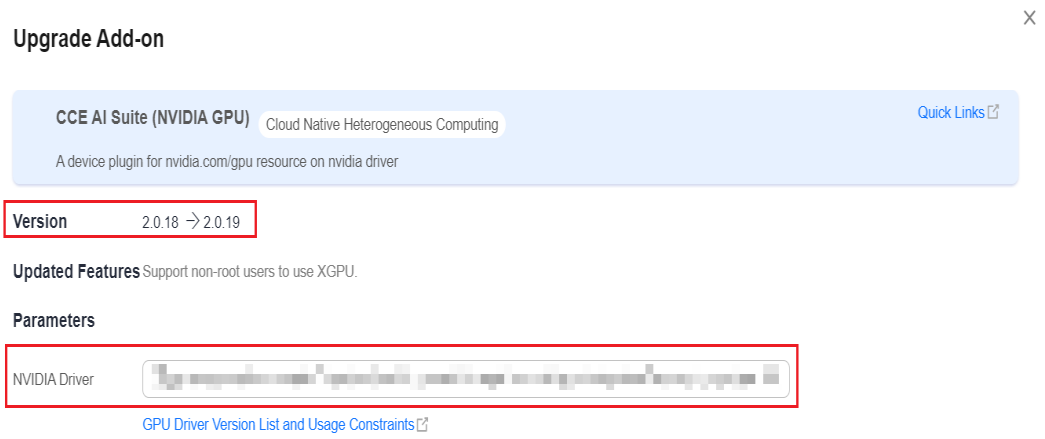
Execute as seguintes operações para verificar a atualização do driver da GPU para a versão de destino e a configuração atual de driver do complemento da GPU:

Passo 1 Efetue logon no console do CCE e clique em **Add-ons** para exibir o complemento da GPU.

NOTA

gpu-beta é o mesmo que **gpu-device-plugin**. **gpu-beta** é renomeado **gpu-device-plugin** em versões posteriores a 2.0.0.

Passo 2 Clique em **Upgrade** do complemento para exibir a versão de destino e a configuração do complemento driver.



Passo 3 Verifique a atualização do driver da GPU para a versão de destino no ambiente de teste, configure o driver da GPU atual e verifique se o nó da GPU criado pode ser executado corretamente.

Se o complemento da GPU e o driver da GPU forem incompatíveis, instale o driver de uma versão posterior. Se necessário, entre em contato com o suporte técnico.

----Fim

2.4.6.55 Configurações de parâmetro do sistema dos nós

Itens de verificação

Verifique se as configurações de parâmetros padrão do sistema em seus nós foram modificadas.

Solução

Se o valor de MTU da rede bond0 em seu nó do BMS não for o valor padrão 1500, este item de verificação falhou.

Configurações de parâmetros não padrão podem levar à perda de pacotes de serviço. Altere-os de volta para os valores padrão.

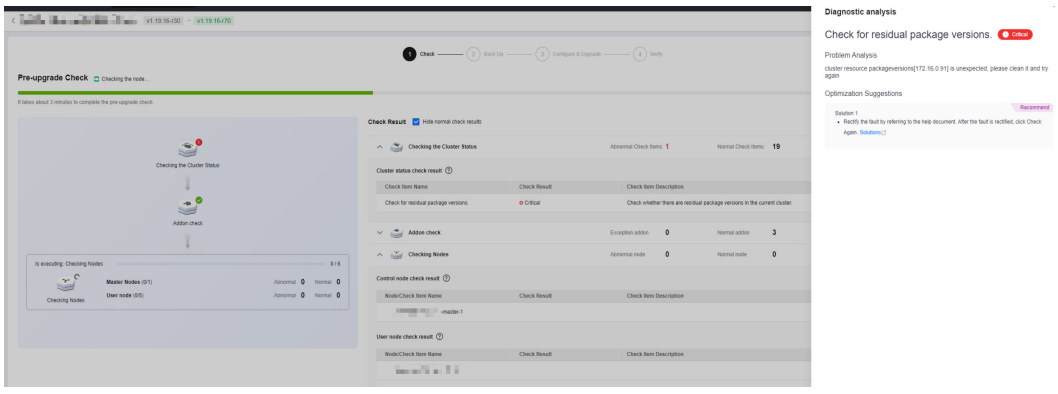
2.4.6.56 Versões de pacotes residuais

Itens de verificação

Verificar se há versões residuais de pacotes no cluster atual.

Solução

Uma mensagem é exibida indicando que há recursos de CRD residuais (10.12.1.109) no seu cluster. Esse problema ocorre porque os recursos de CRD não são limpos depois que os nós de versões anteriores do CCE são excluídos.



Execute manualmente as seguintes operações para limpar os recursos residuais:

Passo 1 Faça backup dos recursos residuais de CRD. Tome o recurso de CRD 10.12.1.109 como um exemplo. Substitua-o pelo recurso exibido na mensagem de erro.

```
kubectl get packageversion 10.12.1.109 -oyaml > /tmp/packageversion-109.bak
```


Passo 2 Limpe os recursos de CRD residuais.

```
kubectl delete packageversion 10.12.1.109
```

Passo 3 Verifique as versões de pacotes residuais novamente.

----Fim

2.4.6.57 Comandos do nó

Itens de verificação

Verifique se os comandos necessários para a atualização estão disponíveis no nó.

Solução

A falha de atualização de cluster normalmente é causada pela falta de comandos de nó principais que são necessários na atualização de cluster.

Mensagens de erro:

```
__error_code#ErrorCommandNotExist#chage command is not exists#__
__error_code#ErrorCommandNotExist#chown command is not exists#__
__error_code#ErrorCommandNotExist#chmod command is not exists#__
__error_code#ErrorCommandNotExist#mkdir command is not exists#__
__error_code#ErrorCommandNotExist#in command is not exists#__
__error_code#ErrorCommandNotExist#touch command is not exists#__
__error_code#ErrorCommandNotExist#pidof command is not exists#__
```

As mensagens de erro precedentes indicam a falta de comandos do nó tais como **chage**, **chown** e **chmod**. Adicione estes comandos e verifique os comandos de nó novamente.

2.4.6.58 Troca de nó

Itens de verificação

Verifique se a troca foi ativada nos nós de cluster.

Solução

Por padrão, a troca é desativada nos nós do CCE. Verifique a necessidade de ativar a troca manualmente e determine o impacto de desativar essa função. Execute o comando **swapoff -a** para desativar a troca.

2.5 Gerenciamento de um cluster

2.5.1 Gerenciamento de configuração de cluster

Cenário

O CCE permite que você gerencie parâmetros de cluster, por meio dos quais você pode permitir que os componentes principais funcionem sob suas próprias necessidades.

Restrições

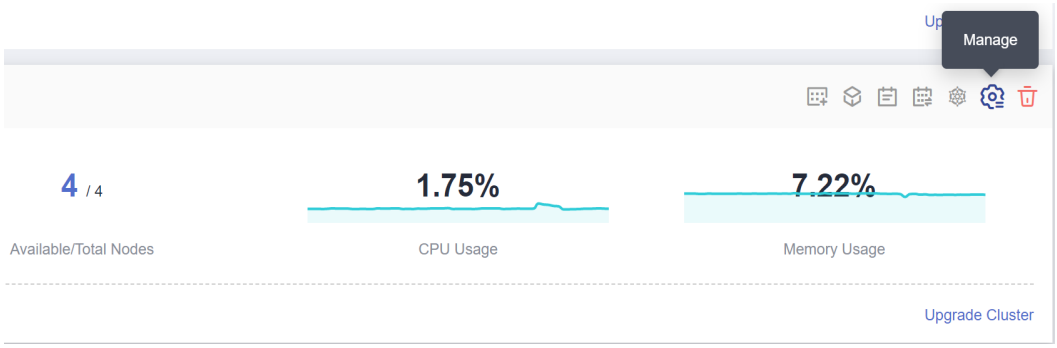
Esta função é suportada apenas em clusters de **v1.15 e posterior**. Ela não é exibida para versões anteriores à v1.15.

Procedimento

Passo 1 Efetue logon no console do CCE. No painel de navegação, escolha **Clusters**.

Passo 2 Clique em  ao lado do cluster de destino.

Figura 2-38 Configuração



Passo 3 Na página **Manage Components** à direita, altere os valores dos parâmetros do Kubernetes listados na tabela a seguir.

Tabela 2-24 Parâmetros de kube-apiserver

Parâmetro	Descrição	Valor
default-not-ready-toleration-seconds	Tempo de tolerância quando um nó está no estado **NotReady**. Por padrão, essa tolerância é adicionada a cada pod.	Padrão: 300s
default-unreachable-toleration-seconds	Tempo de tolerância quando um nó está no estado **unreachable**. Por padrão, essa tolerância é adicionada a cada pod.	Padrão: 300s

Parâmetro	Descrição	Valor
max-mutating-requests-inflight	Número máximo de solicitações de mutação simultâneas. Quando o valor deste parâmetro é excedido, o servidor rejeita solicitações. O valor **0** indica que não há limitação. Este parâmetro está relacionado à escala do cluster. Recomenda-se que o valor não seja alterado.	A configuração manual não é mais suportada desde o cluster v1.21. O valor é especificado automaticamente com base na escala do cluster. ● **200** para clusters com 50 ou 200 nós ● **500** para clusters com 1.000 nós ● **1000** para clusters com 2.000 nós
max-requests-inflight	Número máximo de solicitações simultâneas sem mutação. Quando o valor deste parâmetro é excedido, o servidor rejeita solicitações. O valor **0** indica que não há limitação. Este parâmetro está relacionado à escala do cluster. Recomenda-se que o valor não seja alterado.	A configuração manual não é mais suportada desde o cluster v1.21. O valor é especificado automaticamente com base na escala do cluster. ● **400** para clusters com 50 ou 200 nós ● **1000** para clusters com 1.000 nós ● **2000** para clusters com 2.000 nós
service-node-port-range	Intervalo de portas de NodePort. Depois de alterar o valor, vá para a página de grupo de segurança e altere o intervalo de porta TCP/UDP de grupos de segurança de nó 30000 para 32767. Caso contrário, portas diferentes da porta padrão não podem ser acessadas externamente.	Padrão: 30000-32767 Intervalo de valores: Mín. > 20105 Máx. < 32768
support-overload	Controle de sobrecarga do cluster. Se for ativado, as solicitações simultâneas são controladas dinamicamente com base na pressão de recursos dos nós mestres para mantê-los e o cluster disponíveis. Este parâmetro é suportado apenas por clusters de v1.23 ou posterior.	● false: o controle de sobrecarga está desativado. ● true: o controle de sobrecarga está ativado.

Tabela 2-25 Parâmetros do kube-scheduler

Parâmetro	Descrição	Valor
kube-api-qps	Consulta por segundo (QPS) para usar enquanto conversa com kube-apiserver.	● Se o número de nós em um cluster for menor que 1000, o valor padrão será **100**. ● Se um cluster contiver 1000 ou mais nós, o valor padrão será **200**.
kube-api-burst	Intermitência para usar enquanto fala com kube-apiserver.	● Se o número de nós em um cluster for menor que 1000, o valor padrão será **100**. ● Se um cluster contiver 1000 ou mais nós, o valor padrão será **200**.

Tabela 2-26 Parâmetros de kube-controller-manager

Parâmetro	Descrição	Valor
concurrent-deployment-syncs	Número de Implementações que têm permissão para sincronizar simultaneamente.	Padrão: 5
concurrent-endpoint-syncs	Número de pontos de extremidade finais que são permitidos para sincronizar simultaneamente.	Padrão: 5
concurrent-gc-syncs	Número de trabalhadores do coletor de lixo que têm permissão para sincronizar simultaneamente.	Padrão: 20
concurrent-job-syncs	Número de tarefas que podem ser sincronizadas ao mesmo tempo.	Padrão: 5
concurrent-namespace-syncs	Número de namespaces permitidos para sincronizar simultaneamente.	Padrão: 10
concurrent-replicaset-syncs	Número de ReplicaSets que podem ser sincronizados simultaneamente.	Padrão: 5
concurrent-resource-quota-syncs	Número de cotas de recursos que podem ser sincronizadas simultaneamente.	Padrão: 5
concurrent-service-syncs	Número de serviços que têm permissão para sincronizar simultaneamente.	Padrão: 10

Parâmetro	Descrição	Valor
concurrent-serviceaccount-token-syncs	Número de tokens de conta de serviço que podem ser sincronizados simultaneamente.	Padrão: 5
concurrent-ttl-after-finished-syncs	Número de trabalhadores do controlador TTL-after-finished que são permitidos sincronizar simultaneamente.	Padrão: 5
concurrent_rc_syncs	Número de controladores de replicação que têm permissão para sincronizar simultaneamente. **NOTA** Este parâmetro é usado somente em clusters de v1.19 ou anterior.	Padrão: 5
concurrent-rc-syncs	Número de controladores de replicação que têm permissão para sincronizar simultaneamente. **NOTA** Este parâmetro é usado somente em clusters de v1.21 a v1.23. Em clusters da v1.25 e posteriores, esse parâmetro está obsoleto (oficialmente obsoleto a partir da v1.25.3-r0).	Padrão: 5
horizontal-pod-autoscaler-sync-period	Com que frequência HPA audita métricas em um cluster.	Padrão: 15 segundos
kube-api-qps	Consulta por segundo (QPS) para usar enquanto conversa com kube-apiserver.	● Se o número de nós em um cluster for menor que 1000, o valor padrão será **100**. ● Se um cluster contiver 1000 ou mais nós, o valor padrão será **200**.
kube-api-burst	Intermitência para usar enquanto fala com kube-apiserver.	● Se o número de nós em um cluster for menor que 1000, o valor padrão será **100**. ● Se um cluster contiver 1000 ou mais nós, o valor padrão será **200**.
terminated-pod-gc-threshold	Número de pods terminados que podem existir antes que o coletor de lixo de pod terminado comece a excluir pods encerrados. Se for <= 0, o coletor de lixo de pod terminado será desabilitado.	Padrão: 1000

Tabela 2-27 Parâmetros de eni (suportados apenas por clusters do CCE Turbo)

Parâmetro	Descrição	Valor
nic-minimum-target	Número mínimo de ENIs vinculadas a um nó no nível do cluster	Padrão: 10
nic-maximum-target	Número máximo de ENIs pré-vinculadas a um nó no nível do cluster	Padrão: 0
nic-warm-target	Número de ENIs pré-vinculadas a um nó no nível do cluster	Padrão: 2
nic-max-above-warm-target	Número de recuperação de ENIs pré-vinculadas a um nó no nível do cluster	Padrão: 2
prebound-subeni-percentage	Limite baixo do número de ENIs vinculadas: limite elevado do número de ENIs vinculadas **NOTA** Esse parâmetro está sendo descartado. Use os parâmetros dinâmicos de pré-vinculação dos outras quatro ENIs.	Padrão: 0:0

Tabela 2-28 Parâmetros de configuração estendida do controlador (suportados apenas por clusters de v1.21 e posteriores)

Parâmetro	Descrição	Valor
enable-resource-quota	Se criar automaticamente um objeto de quota de recurso ao criar um namespace. ● **false**: não há criação automática ● **true**: criação automática ativada. Para obter detalhes sobre os padrões de cota de recursos, consulte **Configuração de uma cota de recurso**.	Padrão: false

Passo 4 Clique em **OK**.

----Fim

Referências

- **kube-apiserver**
- **kube-controller-manager**
- **kube-scheduler**

2.5.2 Controle de sobrecarga do cluster

Cenário

Se ativado, as solicitações simultâneas são controladas dinamicamente com base na pressão de recursos dos nós mestres para mantê-los e o cluster disponíveis.

Restrições

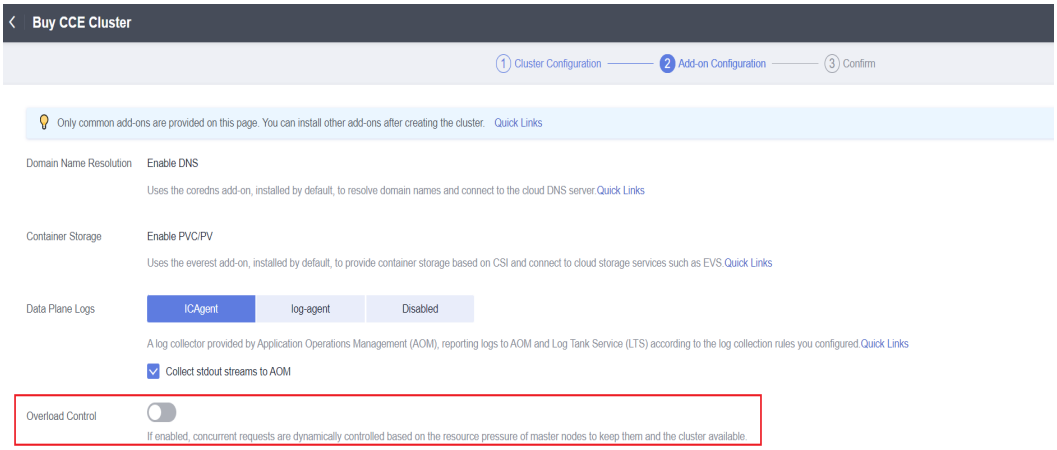
A versão do cluster deve ser 1.23 ou posterior.

Ativar o controle de sobrecarga

Método 1: habilitá-lo ao criar um cluster

Ao criar um cluster v1.23 ou posterior, você pode ativar o controle de sobrecarga durante a criação do cluster.

Figura 2-39 Ativar o controle de sobrecarga durante a criação do cluster

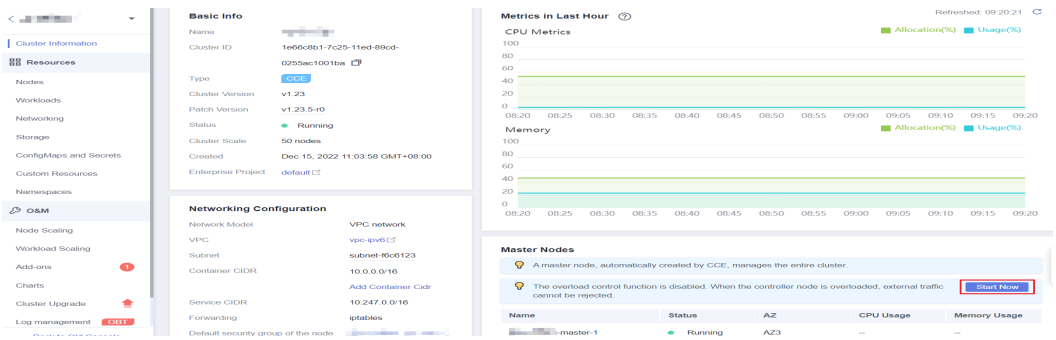


Método 2: habilitá-lo em um cluster existente

Passo 1 Faça logon no console do CCE e vá para um cluster existente cuja versão seja v1.23 ou posterior.

Passo 2 Na página de informações do cluster, exiba as informações do nó principal. Se o controle de sobrecarga não estiver ativado, uma mensagem será exibida. Você pode clicar em **Enable** para ativar a função.

Figura 2-40 Ativar o controle de sobrecarga para um cluster existente



----Fim

Desativar o controle de sobrecarga do cluster

Passo 1 Faça logon no console do CCE e vá para um cluster existente cuja versão seja v1.23 ou posterior.

Passo 2 Na página **Cluster Information**, clique em **Manage** no canto superior direito.

Passo 3 Defina **support-overload** como **false kube-apiserver**.

Passo 4 Clique em **OK**.

----Fim

2.5.3 Alteração de escala do cluster

Cenário

O CCE permite que você altere o número de nós gerenciados em um cluster.

Restrições

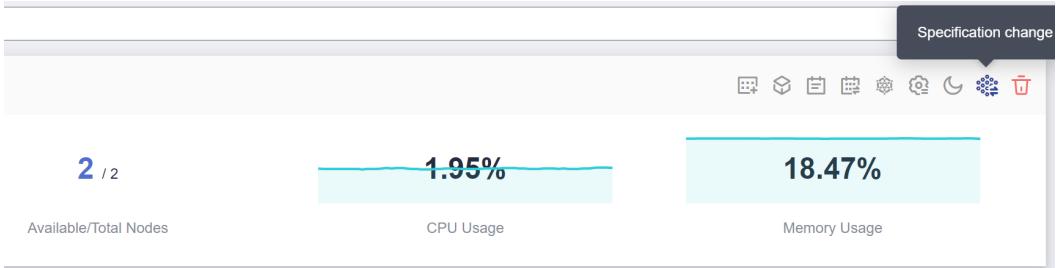
- Esta função é suportada para clusters de v1.15 e versões posteriores.
- A partir da v1.15.11, o número de nós em um cluster pode ser alterado para 2000. O número de nós em um único nó principal não pode ser alterado para 1000 ou mais.
- Atualmente, um cluster só pode ser expandido para uma especificação maior, mas não pode ser diminuído.
- Durante a alteração das especificações, os nós principais serão desligados e ligados e o cluster não poderá ser executado corretamente. Realize a atualização fora dos horários de pico.
- Alterar a escala de cluster não afeta os serviços em execução no cluster. No entanto, o painel de controle (nós principais) será interrompido por um curto período de tempo. Recomenda-se que você não execute nenhuma outra operação (como criar cargas de trabalho) durante a alteração.
- Falhas de alteração acionarão uma reversão de cluster para o estado normal. Se a reversão falhar, submeta um tíquete de serviço.

Procedimento

Passo 1 Efetue logon no console do CCE. No painel de navegação, escolha **Clusters**.

Passo 2 Clique em  ao lado do cluster cujas especificações precisam ser modificadas.

Figura 2-41 Modificar especificações

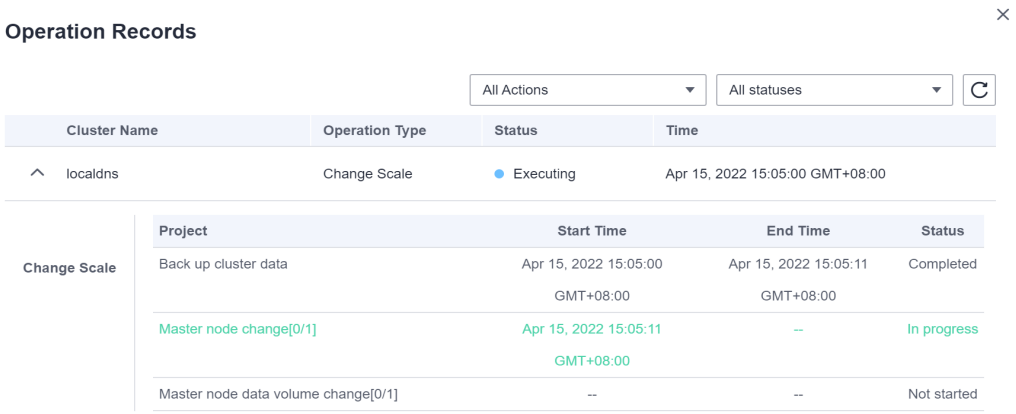


Passo 3 Na página exibida, selecione uma nova escala de cluster.

Passo 4 Clique em **Next** para confirmar as especificações e clique em **OK**.

Você pode clicar em **Operation Records** no canto superior esquerdo para exibir o histórico de alterações do cluster. O status muda de **Executing** para **Successful**, indicando que as especificações do cluster foram alteradas com êxito.

Figura 2-42 Registros da operação



----Fim

2.5.4 Alteração do grupo de segurança padrão de um nó

Cenário

Ao criar um cluster, você pode personalizar um grupo de segurança de nó para gerenciar centralmente as políticas de segurança de rede. Para um cluster criado, você pode alterar seu grupo de segurança de nó padrão.

Restrições

- Não adicione mais de 1000 pods ao mesmo grupo de segurança. Caso contrário, o desempenho do grupo de segurança pode ser afetado. Para obter mais restrições em grupos de segurança, consulte **Restrições de grupo de segurança**.
- O grupo de segurança do nó principal não pode ser especificado. Tenha cuidado ao modificar as regras do grupo de segurança do nó principal. Para obter detalhes, consulte **Configuração das regras do grupo de segurança do cluster do CCE**.

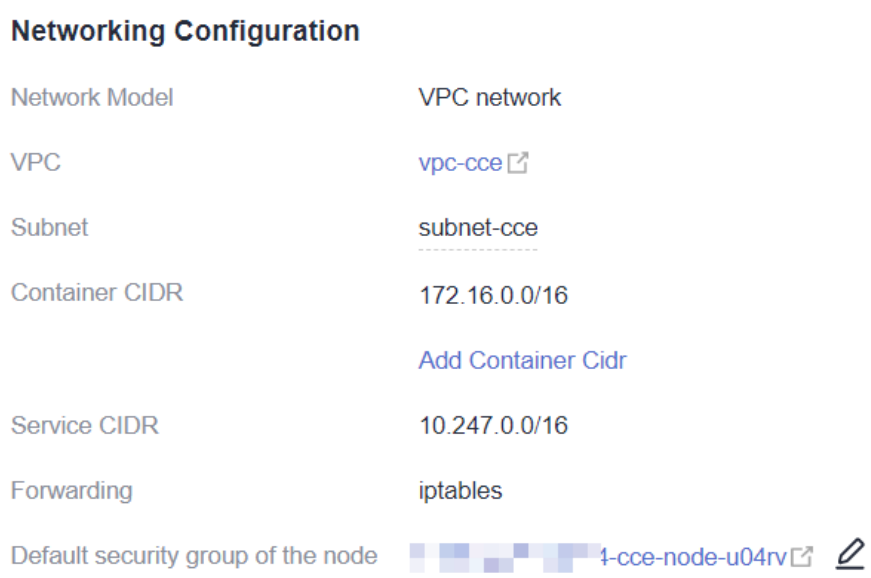
Procedimento

Passo 1 Efetue logon no console do CCE. No painel de navegação, escolha **Clusters**.

Passo 2 Clique no nome do cluster para acessar a página de informações do cluster.

Passo 3 Na área **Network Configuration**, clique em ao lado do **Default security group of the node**.

Figura 2-43 Grupo de segurança de nó padrão

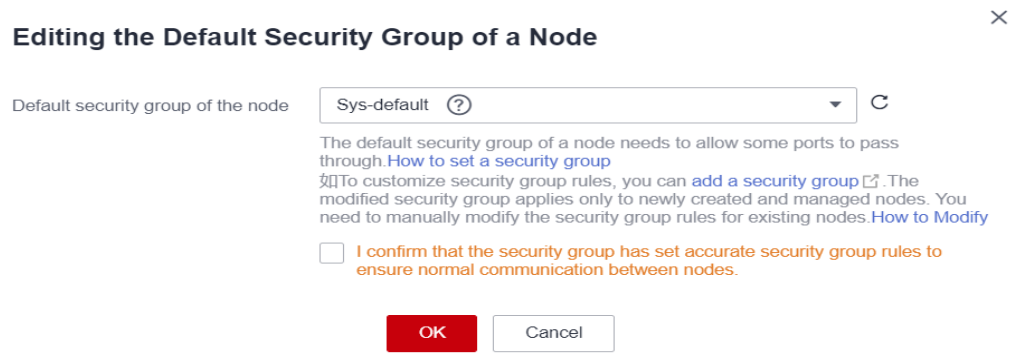


Passo 4 Selecione um grupo de segurança existente, confirme se as regras do grupo de segurança atendem aos requisitos do cluster e clique em **OK**.

AVISO

- Certifique-se de que as regras de porta corretas estejam configuradas para o grupo de segurança selecionado. Caso contrário, o nó não pode ser criado. As regras de porta que um grupo de segurança deve cumprir variam com o tipo de cluster. Para obter detalhes, consulte **Configuração das regras do grupo de segurança do cluster do CCE**.
- O novo grupo de segurança tem efeito somente para nós recém-criados ou gerenciados. Para nós existentes, modifique as regras do grupo de segurança e redefina os nós em tempo real. O grupo de segurança original ainda é usado. Para obter detalhes sobre como modificar as configurações de grupo de segurança dos nós existentes em lotes, consulte **Como alterar o grupo de segurança dos nós em um cluster em lotes?**

Figura 2-44 Editar grupo de segurança de nó padrão



----Fim

2.5.5 Exclusão de um cluster

Cenário

- Você pode excluir diretamente clusters de pagamento por uso. Para mais detalhes, consulte **Exclusão de um cluster**.
- Clusters anuais/mensais não podem ser excluídos diretamente. Cancelar assinatura de clusters que não expiraram ou liberar clusters que expiraram e não foram renovados. Para mais detalhes, consulte **Cancelar a assinatura de ou liberar um cluster**.

Precauções

- Quando um cluster é excluído, os nós gerenciados e anuais/mensais serão removidos do cluster e o sistema será reinstalado. As senhas de logon originais dos nós se tornarão inválidas. Para obter detalhes, consulte **Redefinição da senha para fazer logon em um ECS no console de gerenciamento**.
- A exclusão de um cluster não excluirá os recursos de faturamento anual/mensal no cluster e sua cobrança continuará.
- A exclusão de um cluster excluirá os nós do cluster (excluindo nós aceitos), discos de dados conectados aos nós, cargas de trabalho e Serviços. Os serviços relacionados não podem ser restaurados. Antes de executar essa operação, verifique se os dados foram copiados ou migrados. Os dados excluídos não podem ser restaurados.

  Os recursos que não forem criados no CCE não serão excluídos:

  – Nós aceites (apenas os nós criados no CCE são excluídos);
  – Balanceadores de carga do ELB associados a Serviços e ingresses (apenas os balanceadores de carga criados automaticamente são excluídos);
  – Recursos de armazenamento em nuvem criados manualmente associados aos PVs ou recursos de armazenamento em nuvem importados (somente os recursos de armazenamento em nuvem criados automaticamente pelas PVCs são excluídos)
- Se você excluir um cluster que não esteja em execução (por exemplo, congelado ou indisponível), os recursos associados, como recursos de armazenamento e rede, permanecerão.
- Se a versão do cluster for v1.13.10 ou anterior, não altere manualmente o nome do ouvinte e o nome do servidor de back-end no console do ELB. Caso contrário, os recursos residuais existirão quando você excluir o cluster.

Exclusão de um cluster

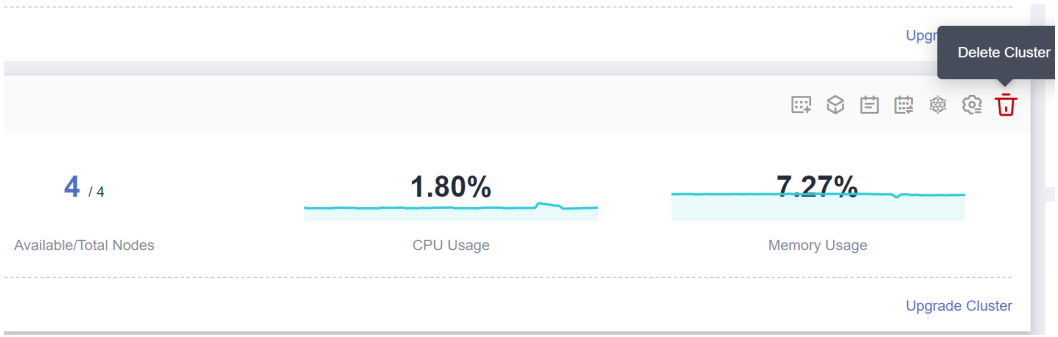
AVISO

Um cluster hibernado não pode ser excluído. Desperte o cluster e tente novamente.

Passo 1 Efetue logon no console do CCE. No painel de navegação, escolha **Clusters**.

Passo 2 Clique em ao lado do cluster a ser excluído.

Figura 2-45 Excluir um cluster



Passo 3 Na caixa de diálogo **Delete Cluster** exibida, selecione os recursos a serem liberados.

- Exclua recursos de armazenamento em nuvem associados a cargas de trabalho no cluster.

  NOTA

  Ao excluir recursos de armazenamento em nuvem subjacentes vinculados a volumes de armazenamento em um cluster, preste atenção às seguintes restrições:

  – Os recursos de armazenamento subjacentes são excluídos de acordo com a política de recuperação definida para os volumes de armazenamento. Por exemplo, se a política de recuperação de volumes de armazenamento for **Retain**, os recursos de armazenamento subjacentes serão retidos após a exclusão do cluster.

  – Se houver mais de 1.000 arquivos no bucket do OBS, limpe manualmente os arquivos e exclua o cluster.

- Exclua recursos de rede, como balanceadores de carga em um cluster. (Somente balanceadores de carga criados automaticamente serão excluídos).
- Exclua o fluxo de log principal do LTS (somente fluxos de log criados automaticamente podem ser excluídos).

  NOTA

  Se você não excluir o fluxo de log, os logs existentes não serão excluídos e você poderá ser cobrado pelos logs do LTS. Para obter detalhes, consulte **Calculadora de preço**.

Passo 4 Clique em **Yes** para iniciar a exclusão do cluster.

A operação de excluir leva de 1 a 3 minutos para ser concluída.

----Fim

Cancelar a assinatura de ou liberar um cluster

AVISO

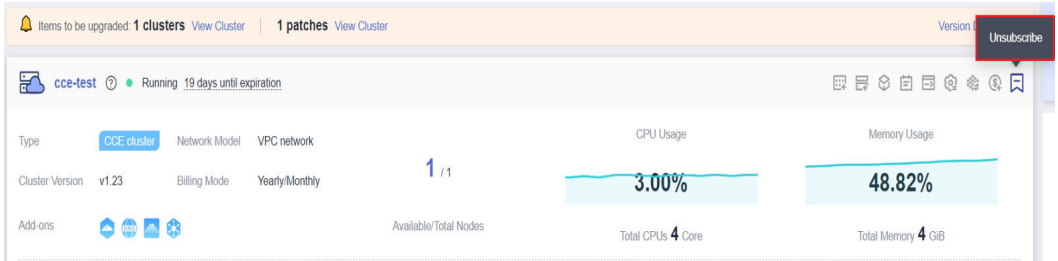
- Quando você cancela a assinatura ou libera um cluster, somente os recursos associados ao pedido são cancelados. Os recursos não associados são retidos e sua cobrança continua.
- Para um cluster anual/mensal, se a retenção expirar, o cluster será liberado automaticamente. Para os nós no cluster, se eles expirarem ao mesmo tempo, eles também serão liberados. Caso contrário, o CCE não executará nenhuma operação em seus nós. Os dados do nó são retidos e o faturamento continua. Preste atenção aos clusters expirados em sua conta e renove-os em tempo hábil para evitar a perda de dados causada pela reinstalação do nó.
- Se um pedido contiver recursos em um relacionamento primário-secundário, cancele a assinatura dos recursos separadamente.
- Para obter detalhes sobre as regras de cancelamento de assinatura, consulte **Regras de cancelar assinatura**.
  - Se você estiver cancelando a assinatura de um recurso que está sendo usado, revise as informações do recurso e as informações de reembolso com cuidado. O recurso não pode ser restaurado após o cancelamento da assinatura. Você pode cancelar a assinatura de um período de renovação que ainda não entrou em vigor na página **Unsubscribe from Renewal Period**.
  - O cancelamento da assinatura de um recurso associado a outros recursos de cobrança anual/mensal pode afetar o uso normal desses recursos.

    O cancelamento da assinatura de um recurso associado a outros recursos de pagamento por uso não afetará o uso normal desses recursos. Eles serão cobrados normalmente.
  - Se a sua operação não for um cancelamento incondicional de cinco dias, você será cobrado pela taxa de manuseio e pelo valor usado. Cupons de dinheiro e cupons de desconto usados não serão reembolsados.

Passo 1 Efetue logon no console do CCE. No painel de navegação, escolha **Clusters**.

Passo 2 Clique em ao lado do cluster de destino.

Figura 2-46 Cancelar a assinatura de um cluster



Passo 3 Na página exibida, selecione os recursos a serem liberados.

- Exclua recursos de armazenamento em nuvem associados a cargas de trabalho no cluster.

> **NOTA**
>
> Ao excluir recursos de armazenamento em nuvem subjacentes vinculados a volumes de armazenamento em um cluster, preste atenção às seguintes restrições:
>
> – Os recursos de armazenamento subjacentes são excluídos de acordo com a política de recuperação definida para os volumes de armazenamento. Por exemplo, se a política de recuperação de volumes de armazenamento for **Retain**, os recursos de armazenamento subjacentes serão retidos após a exclusão do cluster.
>
> – Se houver mais de 1.000 arquivos no bucket do OBS, limpe manualmente os arquivos e exclua o cluster.

- Exclua recursos de rede, como balanceadores de carga em um cluster. (Somente balanceadores de carga criados automaticamente serão excluídos).
- Exclua o fluxo de log principal do LTS (somente fluxos de log criados automaticamente podem ser excluídos).

> **NOTA**
>
> Se você não excluir o fluxo de log, os logs existentes não serão excluídos e você poderá ser cobrado pelos logs do LTS. Para obter detalhes, consulte **Calculadora de preço**.

Passo 4 Clique em **Yes**. O cancelamento ou liberação leva de 1 a 3 minutos para ser concluído.

----Fim

2.5.6 Hibernação e despertar de um cluster (pagamento por uso)

Cenário

Se você não precisar usar um cluster temporariamente, é aconselhável hibernar o cluster.

Depois que um cluster é hibernado, recursos como cargas de trabalho não podem ser criados ou gerenciados no cluster.

Um aglomerado hibernado pode ser rapidamente desperto e usado normalmente.

Restrições

Os clusters cobrados anualmente/mensalmente não podem ser hibernados.

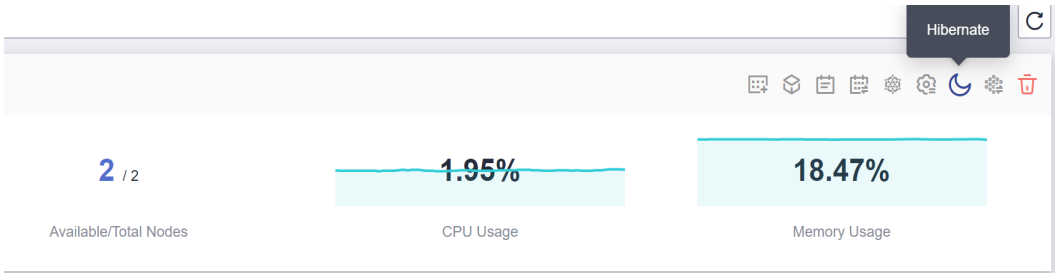
Durante o despertar do cluster, o nó principal poderá falhar ao ser iniciado devido a recursos insuficientes. Como resultado, o cluster não consegue ser acordado. Espere um pouco e acorde o aglomerado novamente.

Hibernar um cluster

Passo 1 Efetue logon no console do CCE. No painel de navegação, escolha **Clusters**.

Passo 2 Clique em ao lado do cluster a ser hibernado.

Figura 2-47 Hibernar um cluster

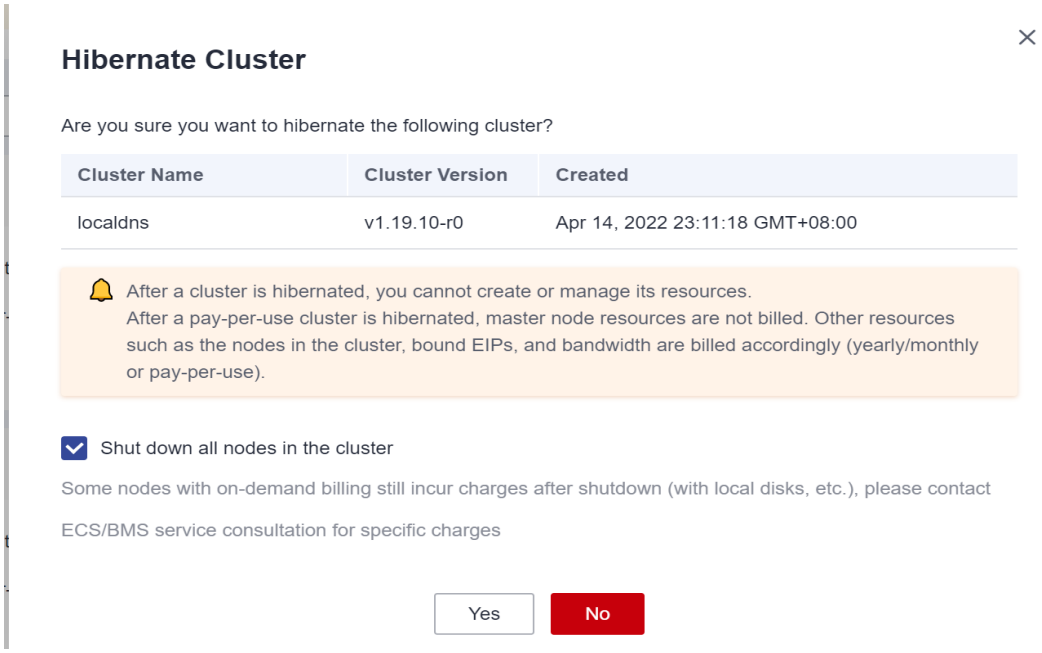


Passo 3 Na caixa de diálogo exibida, verifique as precauções e clique em **Yes**. Aguarde até que o cluster esteja hibernado.

Depois que um cluster é hibernado, o faturamento de recursos de nó principal será interrompido. Recursos, como nós de trabalho (ECSs), EIPs vinculados e largura de banda, ainda são cobrados com base em seus próprios modos de cobrança. Para encerrar os nós, selecione **Stop all nodes in the cluster** na caixa de diálogo ou consulte **Interrupção de um nó**.

A maioria dos nós não é mais faturada após a interrupção, excluindo determinados tipos de ECSs (aqueles com discos locais anexados, como ECSs de I/O ultra-alta e com uso intenso de disco). Para obter detalhes, consulte **Cobrança do ECS**.

Figura 2-48 Informações imediatas



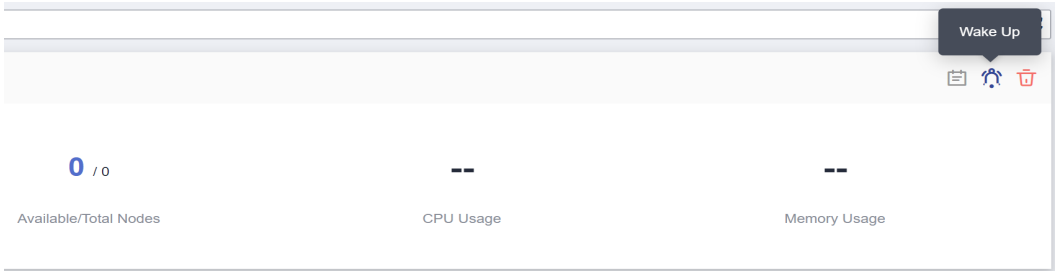
----Fim

Despertar um cluster

Passo 1 Efetue logon no console do CCE. No painel de navegação, escolha **Clusters**.

Passo 2 Clique em ao lado do cluster para ser desperto.

Figura 2-49 Despertar um cluster



Passo 3 Quando o status do cluster muda de **Waking up** para **Running**, o cluster é desperto. Demora cerca de 3 a 5 minutos para despertar o cluster.

NOTA

Depois que o cluster é desperto, a taxa de gerenciamento do cluster continua a ser cobrada.

----Fim

2.5.7 Renovação de um cluster de cobrança anual/mensal

Você pode renovar um cluster de cobrança anual/mensal.

Procedimento

Esta seção descreve como renovar um cluster do CCE de **cobrança anual/mensal**.

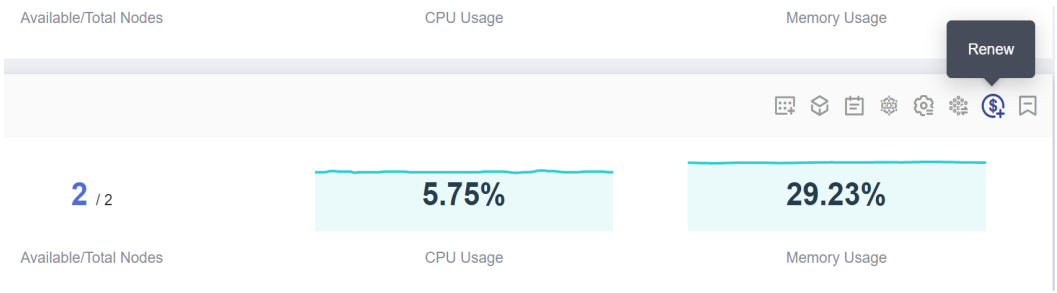
AVISO

Um cluster de cobrança anual/mensal será excluído se não for renovado após a expiração, e todos os nós e os serviços em execução no cluster serão destruídos. O CCE recomenda enfaticamente que você renove o cluster antes que ele expire ou **ative a renovação automática**.

Passo 1 Efetue logon no console do CCE. No painel de navegação, escolha **Clusters**.

Passo 2 Clique em ao lado do cluster a ser renovado.

Figura 2-50 Renovar um cluster



Passo 3 Na página exibida, renove o serviço conforme solicitado.

NOTA

- Se o recurso selecionado (destacado) estiver associado a outros recursos, você poderá decidir se deseja executar a operação em todos esses recursos ao mesmo tempo.
- Se você alterar as especificações do recurso antes que seu período de renovação entre em vigor, o período de renovação não poderá ser cancelado.
- Recursos renovados não são qualificados para o cancelamento de assinatura em 5 dias sem condicionantes.

Passo 4 Clique em **Pay**. Na página exibida, revise o valor do pedido, selecione um método de pagamento e clique em **Pay**.

Passo 5 Após a conclusão do pagamento, você pode voltar para a página **Orders** ou **Renewals** para visualizar e gerenciar seu pedido.

----Fim

2.5.8 Alteração do modo de cobrança de pagamento por uso para anual/mensal

Atualmente, os clusters suportam modos de cobrança **pagamento por uso** e **anual/mensal**. Um cluster de pagamento por uso pode ser convertido em um cluster de cobrança anual/mensal.

Restrições

- Não é possível alterar os nós de pagamento por uso para anual/mensal no console do ECS.
- Somente os nós no pool de nós padrão **DefaultPool** podem ser alterados para o modo de cobrança anual/mensal.
- Os nós cujo modo de cobrança é alterado para anual/mensal não suportam o escalonamento automático.

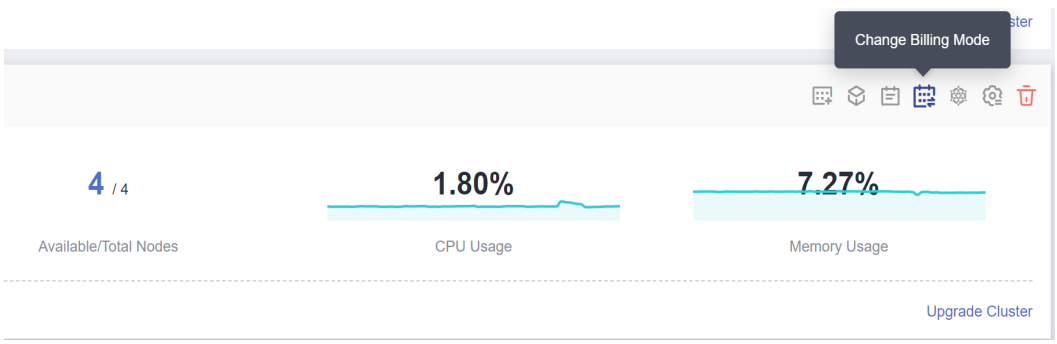
Alterar para a cobrança anual/mensal

Para alterar o modo de cobrança dos clusters comprados de pagamento por uso para anual/mensal, execute as seguintes etapas:

Passo 1 Efetue logon no console do CCE. No painel de navegação, escolha **Clusters**.

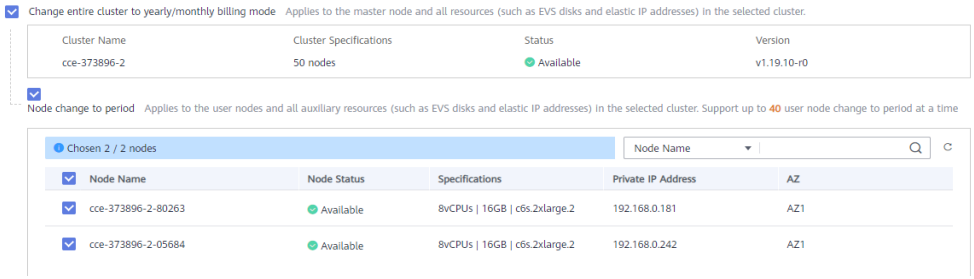
Passo 2 Clique em  ao lado do cluster de destino.

Figura 2-51 Mudar para o modo de cobrança anual/mensal



Passo 3 Na página **Change Billing Mode**, escolha os nós principal e de trabalho que serão alterados para o modo de cobrança anual/mensal.

Figura 2-52 Alterar o modo de cobrança para nós principal e de trabalho



Passo 4 Clique em **OK**. Aguarde até que o pedido seja processado e o pagamento seja concluído.

Durante o pagamento, se uma mensagem for exibida indicando **you do not have the permission to access the resource API**, volte para a página anterior e execute a operação novamente.

----Fim

3 Nós

3.1 Visão geral do nó

Introdução

Um cluster de contêiner consiste em um conjunto de máquinas de trabalho, chamadas de nós, que executam aplicações conteinerizadas. Um nó pode ser uma máquina virtual (VM) ou uma máquina física (PM), dependendo dos seus requisitos de serviço. Os componentes em um nó incluem kubelet, tempo de execução de contêiner e kube-proxy.

NOTA

Um cluster do Kubernetes consiste em nós principais e nós de trabalho. Os nós descritos nesta seção referem-se aos **nós de trabalho**, os nós de computação de um cluster que executa aplicações em contêiner.

O CCE usa Elastic Cloud Servers (ECSs) ou Bare Metal Servers (BMSs) de alto desempenho como nós para criar clusters do Kubernetes altamente disponíveis.

Especificações do nó suportado

Diferentes regiões suportam diferentes flavors de nó, e os flavors de nó podem ser alterados ou esgotados. Faça logon no console do CCE e verifique se os flavors de nó necessários são suportados na página para a criação de nós.

Sistema de armazenamento de arquivos subjacente do Docker

- Em clusters v1.15.6 ou anterior, o sistema de armazenamento de arquivos subjacente usa o formato XFS.
- Em clusters v1.15.11 ou posterior, depois que um nó é criado ou redefinido, o sistema de armazenamento de arquivos subjacente usa o formato ext4.

Para aplicações em contêiner que usam o formato XFS, preste atenção ao impacto da alteração do formato de armazenamento de arquivos subjacente. (A sequência de arquivos em diferentes sistemas de arquivos é diferente. Por exemplo, algumas aplicações Java fazem referência a um pacote JAR, mas o diretório contém várias versões do pacote JAR. Se a versão não for especificada, o pacote referenciado real é determinado pelo arquivo de sistema.)

Execute o comando **docker info | grep "Backing Filesystem"** para verificar o formato do arquivo de armazenamento subjacente do Docker usado pelo nó atual.

Usuários e grupo de usuários de paas

Quando você cria um nó em um cluster do CCE, um usuário ou grupo de usuários de paas é criado no nó por padrão. Os componentes do CCE e os complementos do CCE em um nó são executados como um usuário não raiz (usuário/grupo de usuários de paas) para minimizar a permissão de execução. Se o usuário ou grupo de usuários de paas for modificado, os componentes e pods do CCE podem não funcionar corretamente.

AVISO

O funcionamento normal dos componentes do CCE depende do usuário ou grupo de usuários de paas. Preste atenção aos seguintes requisitos:

- Não modifique a permissão de diretório e a permissão de diretório de contêiner em um nó.
- Não altere o GID e o UID do usuário ou grupo de usuários do paas.
- Não use diretamente o usuário ou o grupo de usuários de paas para definir o usuário e o grupo ao qual o arquivo de serviço pertence.

Ciclo de vida do nó

Um ciclo de vida indica os status do nó registrados a partir do momento em que o nó é criado até o momento em que o nó é excluído ou liberado.

Tabela 3-1 Status do nó

Estado	Atributo de status	Descrição
Running	Estado estável	O nó está em execução corretamente e está conectado ao cluster. Nós nesse estado pode fornecer serviços.
Unavailable	Estado estável	O nó não está funcionando corretamente. As instâncias neste estado não fornecem mais serviços. Neste caso, execute as operações em **Redefinição de um nó**.
Creating	Estado intermediário	O nó foi criado, mas não está em execução.
Installing	Estado intermediário	O software Kubernetes está sendo instalado no nó.
Deleting	Estado intermediário	O nó está sendo excluído. Se esse estado permanece por muito tempo, ocorre uma exceção.

Estado	Atributo de status	Descrição
Stopped	Estado estável	O nó está parado corretamente. Um nó neste estado não pode fornecer serviços. Você pode iniciar o nó no console do ECS.
Error	Estado estável	O nó está anormal. As instâncias neste estado não fornecem mais serviços. Neste caso, execute as operações em **Redefinição de um nó**.

3.2 Mecanismo de contêiner

Introdução aos mecanismos de contêineres

Os mecanismos de contêineres, um dos componentes mais importantes do Kubernetes, gerenciam o ciclo de vida de imagens e contêineres. O kubelet interage com um tempo de execução de contêiner por meio da Interface de tempo de execução do contêiner (CRI).

CCE suporta contêiner e Docker. **containerd é recomendado por seus traços mais curtos, menos componentes, maior estabilidade e menor consumo de recursos de nó**.

O Kubernetes removeu o dockershim da v1.24 e não suporta o Docker por padrão. Para obter detalhes, consulte **Kubernetes está se movendo do Dockershim: compromissos e próximas etapas**. Para garantir a experiência do usuário, o CCE continuará a suportar o Docker na v1.25, mas apenas até a v1.27. Migre nós do Docker para contêineres antes disso. Para mais detalhes, consulte **Migração de nós do Docker para containerd**.

Mapeamento entre sistemas operacionais de nó e mecanismos de contêiner

Tabela 3-2 Sistemas operacionais de nó e mecanismos de contêiner em clusters do CCE

SO	Versão de kernel	Mecanismo de contêiner	Rootfs de armazenamento de contêiner	Tempo de execução do contêiner
CentOS 7.x	3.x	Docker Clusters de v1.23 e posterior suportam containerd.	Clusters de v1.19.16 e anteriores usam Device Mapper. Clusters de v1.19.16 e posterior usam OverlayFS.	runC
EulerOS 2.3	3.x	Docker	Device Mapper	runC
EulerOS 2.5	3.x	Docker	Device Mapper	runC

SO	Versão de kernel	Mecanismo de contêiner	Rootfs de armazenamento de contêiner	Tempo de execução do contêiner
EulerOS 2.9	4.x	Docker Clusters de v1.23 e posterior suportam containerd.	OverlayFS	runC
Ubuntu 18.04	4.x	Docker Clusters de v1.23 e posterior suportam containerd.	OverlayFS	runC
Huawei Cloud EulerOS 1.1	3.x	Docker containerd	OverlayFS	runC
Huawei Cloud EulerOS 2.0	5.x	Docker containerd	OverlayFS	runC

Tabela 3-3 Sistemas operacionais de nó e mecanismos de contêiner em clusters do CCE Turbo

Tipo de nó	SO	Versão de kernel	Mecanismo de contêiner	Rootfs de armazenamento de contêiner	Tempo de execução do contêiner
Elastic Cloud Server (VM)	CentOS 7.6	3.x	Docker containerd	OverlayFS	runC
	Ubuntu 18.04	4.x			
	EulerOS 2.9	4.x			
	Huawei Cloud EulerOS 1.1	3.x			
	Huawei Cloud EulerOS 2.0	5.x			
Elastic Cloud Server (máquina física)	EulerOS 2.9	4.x	containerd	Device Mapper	Kata

Tabela 3-4 Sistemas operacionais de nó e mecanismos de contêiner em CCE Kunpeng

SO	Versão de kernel	Mecanismo de contêiner	Rootfs de armazenamento de contêiner	Tempo de execução do contêiner
EulerOS 2.8	4.x	Docker	OverlayFS	runC

Comandos comuns de containerd e Docker

containerd não suporta as APIs e a CLI do Docker, mas você pode executar comandos crictl para implementar funções semelhantes.

Tabela 3-5 Comandos relacionados à imagem

N.º	Comando de Docker	Comando de containerd	Observações
1	docker images [Option] [Image name[:Tag]]	crictl images [Option] [Image name[:Tag]]	Listar imagens locais.
2	docker pull [Option] *Image name*[:Tag|@DIGEST]	crictl pull [Option] *Image name*[:Tag|@DIGEST]	Extrair imagens.
3	docker push	Nenhum	Enviar imagens por push.
4	docker rmi [Option] *Image*...	crictl rmi [Option] *Image ID*...	Excluir uma imagem local.
5	docker inspect *Image ID*	crictl inspecti *Image ID*	Verificar as imagens.

Tabela 3-6 Comandos relacionados ao contêiner

N.º	Comando de Docker	Comando de containerd	Observações
1	docker ps [Option]	crictl ps [Option]	Listar contêineres.
2	docker create [Option]	crictl create [Option]	Criar um contêiner.
3	docker start [Option] *Container ID*...	crictl start [Option] *Container ID*...	Iniciar um contêiner.
4	docker stop [Option] *Container ID*...	crictl stop [Option] *Container ID*...	Parar um contêiner.
5	docker rm [Option] *Container ID*...	crictl rm [Option] *Container ID*...	Excluir um contêiner.

N.º	Comando de Docker	Comando de containerd	Observações
6	docker attach [Option] *Container ID*	crictl attach [Option] *Container ID*	Conectar-se a um contêiner.
7	docker exec [Option] *Container ID Startup command* [*Parameter*...]	crictl exec [Option] *Container ID Startup command* [*Parameter*...]	Acessar o contêiner.
8	docker inspect [Option] *Container name|ID*...	crictl inspect [Option] *Container ID*...	Consultar detalhes do contêiner.
9	docker logs [Option] *Container ID*	crictl logs [Option] *Container ID*	Exibir logs do contêiner.
10	docker stats [Option] *Container ID*...	crictl stats [Option] *Container ID*	Verificar o uso de recursos do contêiner.
11	docker update [Option] *Container ID*...	crictl update [Option] *Container ID*...	Atualizar limites de recursos do contêiner.

Tabela 3-7 Comandos relacionados ao pod

N.º	Comando de Docker	Comando de containerd	Observações
1	Nenhum	crictl pods [Option]	Listar pods.
2	Nenhum	crictl inspectp [Option] *Pod ID*...	Ver detalhes do pod.
3	Nenhum	crictl start [Option] *Pod ID*...	Iniciar um pod.
4	Nenhum	crictl runp [Option] *Pod ID*...	Executar um pod.
5	Nenhum	crictl stopp [Option] *Pod ID*...	Parar um pod.
6	Nenhum	crictl rmp [Option] *Pod ID*...	Excluir um pod.

NOTA

Os contêineres criados e iniciados pelo containerd são imediatamente excluídos pelo kubelet. containerd não suporta suspender, retomar, reiniciar, renomear e aguardar contêineres, nem criar, importar, exportar, comparar, enviar, pesquisar e rotular imagens do Docker. containerd não suporta cópia de arquivo. Você pode efetuar logon no repositório de imagens modificando o arquivo de configuração de containerd.

Diferenças no rastreamento

- Docker(Kubernetes 1.23 e versões anteriores):

kubelet --> docker shim (no processo de kubelet) --> docker --> containerd

- Docker(solução de comunidade para Kubernetes v1.24 ou posterior):

  kubelet --> cri-dockerd (o kubelet usa a CRI para se conectar a cri-dockerd) --> docker--> containerd

- containerd:

  kubelet --> plug-in cri (no processo de containerd) --> containerd

Embora o Docker tenha adicionado funções como swarm cluster, docker build e APIs do Docker, ele também introduz bugs. Comparado com containerd, Docker tem mais uma camada de chamadas. **Portanto, containerd economiza mais recursos e é mais seguro.**

Descrição da versão do mecanismo de contêiner

- Docker
  - EulerOS/CentOS: docker-engine 18.9.0, uma versão do Docker customizada para CCE. As vulnerabilidades de segurança serão corrigidas em tempo hábil.
  - Ubuntu: docker-ce 18.9.9 (versão comunitária). É aconselhável usar o mecanismo containerd para nós do Ubuntu.

    NOTA

    O docker-ce de código aberto do Ubuntu pode disparar bugs quando operações exec simultâneas são executadas (por exemplo, várias sondas exec são configuradas). É aconselhável usar testes HTTP/TCP.

- containerd: 1.6.14

3.3 Sistema operacional do nó

Mapeamentos entre versões de cluster e versões de sistema operacional

A tabela a seguir lista os mapeamentos entre as versões de cluster e as versões de SO lançadas.

Tabela 3-8 Mapeamentos entre versões de SO de nó de VM e versões de cluster

SO	Versão do cluster	Kernel mais recente
Huawei Cloud EulerOS 2.0	v1.25	5.10.0-60.18.0.50.r865_35.hce2.x86_64
	v1.23	5.10.0-60.18.0.50.r865_35.hce2.x86_64
Huawei Cloud EulerOS 2.0 (Arm)	v1.25	5.10.0-60.18.0.50.r865_35.hce2.aarch64
Huawei Cloud EulerOS 2.0 (Arm)	v1.25	5.10.0-60.18.0.50.r865_35.hce2.aarch64
	v1.23	5.10.0-60.18.0.50.r865_35.hce2.aarch64
Ubuntu 22.04	1.25	5.15.0-60-generic
	1.23	5.15.0-60-generic

SO	Versão do cluster	Kernel mais recente
Huawei Cloud EulerOS 1.1	v1.25	3.10.0-1160.76.2.hce1c.x86_64
	v1.23	3.10.0-1160.76.2.hce1c.x86_64
	v1.21	3.10.0-1160.76.2.hce1c.x86_64
CentOS Linux release 7.6	v1.25	3.10.0-1160.66.1.el7.x86_64
	v1.23	3.10.0-1160.66.1.el7.x86_64
	v1.21	3.10.0-1160.66.1.el7.x86_64
	v1.19.16	3.10.0-1160.66.1.el7.x86_64
	v1.19.10	3.10.0-1160.25.1.el7.x86_64
	v1.19.8	3.10.0-1160.15.2.el7.x86_64
	v1.17.17 (fim da manutenção)	3.10.0-1160.15.2.el7.x86_64
	v1.17.9 (fim da manutenção)	3.10.0-1062.12.1.el7.x86_64
	v1.15.11 (fim da manutenção)	3.10.0-1062.12.1.el7.x86_64
	v1.15.6-r1 (fim da manutenção)	3.10.0-1062.1.1.el7.x86_64
	v1.13.10-r1 (fim da manutenção)	3.10.0-957.21.3.el7.x86_64
	v1.13.7-r0 (fim da manutenção)	3.10.0-957.21.3.el7.x86_64
EulerOS release 2.9	v1.25	4.18.0-147.5.1.6.h1017.eulerosv2r9.x86_64
	v1.23	4.18.0-147.5.1.6.h1017.eulerosv2r9.x86_64
	v1.21	4.18.0-147.5.1.6.h1017.eulerosv2r9.x86_64
	v1.19	4.18.0-147.5.1.6.h1017.eulerosv2r9.x86_64
EulerOS release 2.9 (Arm)	v1.25	4.19.90-vhulk2103.1.0.h990.eulerosv2r9.aarch64
	v1.23	4.19.90-vhulk2103.1.0.h990.eulerosv2r9.aarch64
	v1.21	4.19.90-vhulk2103.1.0.h990.eulerosv2r9.aarch64

SO	Versão do cluster	Kernel mais recente
	v1.19	4.19.90-vhulk2103.1.0.h990.eulerosv2r9.aarch64
EulerOS release 2.10	v1.25	4.18.0-147.5.2.10.h933.eulerosv2r10.x86_64
	v1.23	4.18.0-147.5.2.10.h933.eulerosv2r10.x86_64
EulerOS release 2.10 (Arm)	v1.25	4.19.90-vhulk2204.1.0.h1160.eulerosv2r10.aarch64
	v1.23	4.19.90-vhulk2204.1.0.h1160.eulerosv2r10.aarch64
EulerOS release 2.8 (Arm)	v1.25	4.19.36-vhulk1907.1.0.h1350.eulerosv2r8.aarch64
	v1.23	4.19.36-vhulk1907.1.0.h1350.eulerosv2r8.aarch64
	v1.21	4.19.36-vhulk1907.1.0.h1350.eulerosv2r8.aarch64
	v1.19.16	4.19.36-vhulk1907.1.0.h1350.eulerosv2r8.aarch64
	v1.19.10	4.19.36-vhulk1907.1.0.h962.eulerosv2r8.aarch64
	v1.17.17 (end of maintenance)	4.19.36-vhulk1907.1.0.h962.eulerosv2r8.aarch64
	v1.15.11 (end of maintenance)	4.19.36-vhulk1907.1.0.h702.eulerosv2r8.aarch64
EulerOS release 2.5	v1.25	3.10.0-862.14.1.5.h687.eulerosv2r7.x86_64
	v1.23	3.10.0-862.14.1.5.h687.eulerosv2r7.x86_64
	v1.21	3.10.0-862.14.1.5.h687.eulerosv2r7.x86_64
	v1.19.16	3.10.0-862.14.1.5.h687.eulerosv2r7.x86_64

SO	Versão do cluster	Kernel mais recente
	v1.19.10	3.10.0-862.14.1.5.h520.eulerosv2r7.x86_64
	v1.19.8	3.10.0-862.14.1.5.h520.eulerosv2r7.x86_64
	v1.17.17 (fim da manutenção)	3.10.0-862.14.1.5.h470.eulerosv2r7.x86_64
	v1.17.9 (fim da manutenção)	3.10.0-862.14.1.5.h428.eulerosv2r7.x86_64
	v1.15.11 (fim da manutenção)	3.10.0-862.14.1.5.h428.eulerosv2r7.x86_64
	v1.15.6-r1 (fim da manutenção)	3.10.0-862.14.1.5.h328.eulerosv2r7.x86_64
	v1.13.10-r1 (fim da manutenção)	3.10.0-862.14.1.2.h249.eulerosv2r7.x86_64
	v1.13.7-r0 (fim da manutenção)	3.10.0-862.14.1.0.h197.eulerosv2r7.x86_64
Ubuntu 18.04 server 64-bit (fim da manutenção)	v1.25	4.15.0-171-generic
	v1.23	4.15.0-171-generic
	v1.21	4.15.0-171-generic
	v1.19.16	4.15.0-171-generic
	v1.19.8	4.15.0-136-generic
	v1.17.17	4.15.0-136-generic

Tabela 3-9 Mapeamentos entre versões de SO de nó de BMS e versões de cluster

SO	Versão do cluster	Informações do Kernel
EulerOS release 2.10 (servidor bare-metal elástico)	v1.25 v1.23 v1.21 v1.19.16	4.18.0-147.5.2.15.h1109.eulerosv2r10.x86_64
EulerOS release 2.9 (servidor bare metal)	v1.25 v1.23 v1.21 v1.19	4.18.0-147.5.1.6.h841.eulerosv2r9.x86_64 Para obter detalhes sobre as especificações do servidor, consulte **Família de instância**.

SO	Versão do cluster	Informações do Kernel
EulerOS release 2.3 (servidor bare metal, fim da manutenção)	v1.15.11 ou mais recente	3.10.0-514.41.4.28.h62.x86_64 Para obter detalhes sobre as especificações do servidor, consulte **Família de instância**.

Mapeamentos entre tipos de cluster e versões de sistema operacional

Tabela 3-10 Mapeamento entre SOs de nó e tipos de cluster

SO	Versão do cluster	Modelo de rede da VPC	Modelo de rede de túnel de contêiner	Cloud Native Network 2.0 (cluster do CCE Turbo)
Huawei Cloud EulerOS 2.0	v1.25	√	×	√
	v1.23	√	×	√
Huawei Cloud EulerOS 2.0 (Arm)	v1.25	√	×	√
	v1.23	√	×	√
Ubuntu 22.04	1.25	√	×	√
	1.23	√	×	√
Huawei Cloud EulerOS 1.1	v1.25	√	√	√
	v1.23	√	√	√
	v1.21	√	√	√
CentOS Linux release 7.6	v1.25	√	√	√
	v1.23	√	√	√
	v1.21	√	√	√
	v1.19.16	√	√	√
	v1.19.10	√	√	√
	v1.19.8	√	√	√
	v1.17.17 (fim da manutenção)	√	√	√
	v1.17.9 (fim da manutenção)	√	√	√
	v1.15.11 (fim da manutenção)	√	√	√

SO	Versão do cluster	Modelo de rede da VPC	Modelo de rede de túnel de contêiner	Cloud Native Network 2.0 (cluster do CCE Turbo)
	v1.15.6-r1 (fim da manutenção)	√	√	√
	v1.13.10-r1 (fim da manutenção)	√	√	√
	v1.13.7-r0 (fim da manutenção)	√	√	√
EulerOS release 2.9	v1.25	√	√	√
	v1.23	√	√	√
	v1.21	√	√	√
	v1.19	√	√	√
EulerOS release 2.9 (Arm)	v1.25	√	√	√
	v1.23	√	√	√
	v1.21	√	√	√
	v1.19	√	√	√
EulerOS release 2.10	v1.25	√	√	√
	v1.23	√	√	√
EulerOS release 2.10 (Arm)	v1.25	√	√	√
	v1.23	√	√	√
EulerOS release 2.8 (Arm)	v1.25	√	√	√
	v1.23	√	√	√
	v1.21	√	√	√
	v1.19.16	√	√	√
	v1.19.10	√	√	√
	v1.17.17 (fim da manutenção)	√	√	√
	v1.15.11 (end of maintenance)	√	√	√
EulerOS release 2.5	v1.25	√	√	√
	v1.23	√	√	√
	v1.21	√	√	√

SO	Versão do cluster	Modelo de rede da VPC	Modelo de rede de túnel de contêiner	Cloud Native Network 2.0 (cluster do CCE Turbo)
	v1.19.16	√	√	√
	v1.19.10	√	√	√
	v1.19.8	√	√	√
	v1.17.17 (fim da manutenção)	√	√	√
	v1.17.9 (fim da manutenção)	√	√	√
	v1.15.11 (fim da manutenção)	√	√	√
	v1.15.6-r1 (fim da manutenção)	√	√	√
	v1.13.10-r1 (fim da manutenção)	√	√	√
	v1.13.7-r0 (fim da manutenção)	√	√	√
Ubuntu 18.04 server 64-bit (fim da manutenção)	v1.25	√	×	√
	v1.23	√	×	√
	v1.21	√	×	√
	v1.19.16	√	×	√
	v1.19.8	√	×	√
	v1.17.17	√	×	√

3.4 Criação de um nó

Pré-requisitos

- Pelo menos um cluster foi criado.
- Um par de chaves foi criado para autenticação de identidade no logon do nó remoto.

  Se você usar uma senha para fazer logon em um nó, pule esta etapa. Para obter detalhes, consulte **Criação de um par de chaves**.

Restrições

- O nó tem pelo menos 2 vCPUs e 4 GiB de memória.

- Para garantir a estabilidade do nó, um certo número de recursos de nó do CCE será reservado para componentes do Kubernetes (como kubelet, kube-proxy e docker) com base nas especificações do nó. Portanto, o número total de recursos de nó e o número de recursos de nó alocáveis para o cluster são diferentes. Quanto maiores as especificações do nó, mais os contêineres implementados no nó. Portanto, mais recursos de nó precisam ser reservados para executar componentes do Kubernetes. Para mais detalhes, consulte **Política de reserva de recursos de nó**.
- Redes, incluindo redes de VM e redes de contêineres de nós, são todas gerenciadas pelo CCE. Não adicione ou exclua ENIs ou altere rotas. Caso contrário, os serviços podem estar indisponíveis. Por exemplo, a NIC denominada **gw_11cbf51a@eth0** no nó é o gateway de rede do contêiner e não pode ser modificada.
- Se quiser modificar as especificações de um nó comprado, interrompa o nó e execute as operações descritas em **Operações gerais para modificar especificações**. Você também pode comprar um novo nó e excluir o antigo.
- Durante a criação do nó, os pacotes de software são baixados do OBS usando o nome de domínio. Use um servidor DNS privado para resolver o nome de domínio do OBS e configure o endereço do servidor DNS da sub-rede em que o nó reside com um **endereço de servidor DNS privado** . Quando você cria uma sub-rede, o servidor DNS privado é usado por padrão. Se alterar o DNS de sub-rede, certifique-se de que o servidor DNS em utilização pode resolver o nome de domínio do OBS.
- Depois que um nó é criado, sua AZ não pode ser alterada.
- Os nós comprados no modo de faturamento de pagamento por uso serão excluídos depois que você os excluir na página **Nodes** no console do CCE. **Nós de faturamento anual/mensal** em um cluster não podem ser excluídos no console do CCE. Você pode escolher **Billing Center** > **My Orders** no canto superior direito da página para cancelar a assinatura nos nós.
- Os serviços podem ser comprometidos por limites de ID de processo de nó. Avalie se o número máximo de IDs de processo deve ser ajustado. Para mais detalhes, consulte **Alteração de limites de ID de processo (kernel.pid_max)**.
- Quando a pilha dual IPv4/IPv6 está ativada, a concessão ilimitada de DHCP não pode ser ativada para a sub-rede de nó selecionada.

Procedimento

Depois que um cluster é criado, você pode criar nós para o cluster.

Passo 1 Efetue logon no **console do CCE** .

Passo 2 No painel de navegação do console do CCE, escolha **Clusters**. Clique no nome do cluster de destino para acessar sua página de detalhes.

Passo 3 No painel de navegação à esquerda, escolha **Nodes**. Na página exibida, clique em **Create Node**. Na etapa **Node Settings**, defina os parâmetros do nó consultando a tabela a seguir.

Compute Settings

Você pode configurar as especificações e o sistema operacional de um servidor de nuvem, no qual suas aplicações em contêiner são executadas.

Tabela 3-11 Parâmetros de configuração

Parâmetro	Descrição
Billing Mode	Os seguintes modos de cobrança são suportados: ● Yearly/Monthly Você deve especificar a duração necessária se **Yearly/Monthly** estiver selecionado. Você pode escolher se deseja selecionar **Auto-renew** com base nos requisitos do local. Seu pedido será renovado automaticamente mensalmente ou anualmente, dependendo se você comprou por mês ou por ano. ● Pay-per-use Os recursos serão cobrados com base na duração do uso. Você pode provisionar ou excluir recursos a qualquer momento.
AZ	AZ onde o nó está localizado. Os nós em um cluster podem ser criados em diferentes AZs para maior confiabilidade. O valor não pode ser alterado após a criação do nó. Selecione **Random** para implementar seu nó em uma AZ aleatória com base na variação de nó selecionada. Uma AZ é uma região física onde os recursos usam fontes de alimentação e redes independentes. As AZs são fisicamente isoladas, mas interconectadas por meio de uma rede interna. Para melhorar a disponibilidade da carga de trabalho, crie nós em diferentes AZs.
Node Type	Cluster do CCE: ● ECS (VM): os contêineres são executados em ECSs. ● ECS (físico): os contêineres são executados em servidores usando a arquitetura QingTian. ● BMS: os contêineres são executados em BMSs. Anexe discos locais ou discos EVS. Cluster do CCE Turbo: ● ECS (VM): os contêineres são executados em ECSs. Somente os ECSs que podem ser vinculados a várias NICs são suportados. ● ECS (físico): os contêineres são executados em servidores usando a arquitetura QingTian.
Container Engine	Clusters do CCE suportam Docker e containerd em alguns cenários. ● Nós que executam CentOS, Ubuntu ou EulerOS 2.9 suportam containerd. Nós Arm executando EulerOS 2.5 ou EulerOS 2.8 não suportam containerd. ● Os clusters de rede da VPC da v1.23 e versões posteriores suportam containerd. Clusters de rede de túnel de v1.23.2-r0 e versões posteriores suportam containerd. ● Para um cluster do CCE Turbo, **Docker** e o **containerd** são suportados. Para obter detalhes, consulte **Mapeamento entre sistemas operacionais de nó e mecanismos de contêiner**.

Parâmetro	Descrição
Specifications	Selecione as especificações de nó que melhor atendam às suas necessidades de serviço. Os flavors de nó disponíveis variam dependendo das AZs. Obtenha os flavors exibidos no console. **NOTA** Os nós de Kunpeng (de computação geral aprimorada e memória otimizada) podem ser adicionados a um cluster do CCE. As especificações reais exibidas na página de criação do nó são usadas.
OS	Selecione um tipo de SO. Diferentes tipos de nós suportam diferentes sistemas operacionais. ● **Public image**: selecione uma imagem pública para o nó. ● **Private image**: selecione uma imagem privada para o nó. Para obter detalhes sobre como criar uma imagem privada, consulte **Criação de uma imagem de nó do CCE personalizada**. **NOTA** ● Os tempos de execução do contêiner de serviço compartilham o kernel e as chamadas subjacentes dos nós. Para garantir a compatibilidade, selecione uma versão de distribuição do Linux que seja igual ou próxima da imagem final do contêiner de serviço para o sistema operacional do nó.
Node Name	Nome do nó. Quando os nós (ECSs) são criados em lotes, o valor desse parâmetro é usado como o prefixo de nome de cada ECS. O sistema gera um nome padrão para você, que pode ser modificado. O nome do nó deve começar com uma letra minúscula e não pode terminar com um hífen (-). Somente dígitos, letras minúsculas e hifens (-) são permitidos.
Login Mode	● **Password** O nome do usuário padrão é **root**. Digite a senha para efetuar logon no nó e confirme a senha. Certifique-se de lembrar a senha, pois você precisará dela quando fizer logon no nó. ● **Key Pair** Selecione o par de chaves usado para efetuar logon no nó. Você pode selecionar uma chave compartilhada. Um par de chaves é usado para autenticação de identidade quando você entra remotamente em um nó. Se nenhum par de chaves estiver disponível, clique em **Create Key Pair**. Para obter detalhes sobre como criar um par de chaves, consulte **Criação de par de chaves**.

Storage Settings

Configure recursos de armazenamento em um nó para os contêineres em execução nele. Defina o tamanho do disco de acordo com os requisitos do site.

Tabela 3-12 Parâmetros de configuração

Parâmetro	Descrição
System Disk	Disco do sistema usado pelo sistema operacional do nó. O valor varia de 40 GB a 1.024 GB. O valor padrão é 50 GB. **Encryption**: a criptografia de disco do sistema protege seus dados. Os snapshots gerados a partir de discos criptografados e discos criados usando esses snapshots herdam automaticamente a configuração de criptografia. **Esta função está disponível apenas em determinadas regiões.** ● **Encryption** não está selecionada por padrão. ● Depois de selecionar **Encryption**, você pode selecionar uma chave existente na caixa de diálogo exibida. Se nenhuma chave estiver disponível, clique em **View Key List** e crie uma chave. Depois que a chave for criada, clique no ícone de atualização ao lado da caixa de texto **Encryption**.

Parâmetro	Descrição
Data Disk	**Pelo menos um disco de dados é necessário** para o tempo de execução do contêiner e o kubelet. **O disco de dados não pode ser excluído ou desinstalado. Caso contrário, o nó ficará indisponível.** ● Primeiro disco de dados: usado para o tempo de execução do contêiner e componentes do kubelet. O valor varia de 20 GB a 32.768 GB. O valor padrão é 100 GB. ● Outros discos de dados: você pode definir o tamanho do disco de dados para um valor que varia de 10 GB a 32.768 GB. O valor padrão é 100 GB. **NOTA** Se o flavor de nó for de I/O intensiva em disco ou ultra-alta, um disco de dados pode ser um disco local. Discos locais podem quebrar e não garantir a confiabilidade dos dados. Armazene seus dados de serviço em discos EVS, que são mais confiáveis do que os discos locais. **Configurações avançadas** Clique em **Expand** para configurar os seguintes parâmetros: ● **Alocação de espaço em disco de dados**: depois de selecionar **Set Container Engine Space**, você pode especificar a proporção do espaço para o mecanismo de contêiner, a imagem e o armazenamento temporário no disco de dados. O espaço do mecanismo de contêiner é usado para armazenar o diretório de trabalho, os dados da imagem do contêiner e os metadados da imagem para o tempo de execução do contêiner. O espaço restante do disco de dados é usado para arquivos de configuração, chaves e EmptyDir do pod. Para obter detalhes sobre como alocar espaço em disco de dados, consulte **Alocação de espaço em disco de dados**. ● **Encryption**: a criptografia do disco de dados protege os seus dados. Os snapshots gerados a partir de discos criptografados e discos criados usando esses snapshots herdam automaticamente a configuração de criptografia. **Esta função está disponível apenas em determinadas regiões.** – **Encryption** não está selecionada por padrão. – Depois de selecionar **Encryption**, você pode selecionar uma chave existente na caixa de diálogo exibida. Se nenhuma chave estiver disponível, clique em **View Key List** e crie uma chave. Depois que a chave for criada, clique no ícone de atualização ao lado da caixa de texto **Encryption**. **Adicionar vários discos de dados** Um máximo de quatro discos de dados podem ser adicionados. Por padrão, os discos brutos são criados sem qualquer processamento. Você também pode clicar em **Expand** e selecionar qualquer uma das seguintes opções: ● **Default**: por padrão, um disco bruto é criado sem qualquer processamento. ● **Mount Disk**: o disco de dados é anexado a um diretório especificado.

Parâmetro	Descrição
	● **Use as PV**: aplicável a cenários em que há um requisito de alto desempenho em PVs. O rótulo **node.kubernetes.io/local-storage-persistent** é adicionado ao nó com o PV configurado. O valor é **linear** ou **striped**. ● **Use as ephemeral volume**: aplicável a cenários em que há uma exigência de alto desempenho em EmptyDir. **NOTA** ● Os PVs locais são suportados apenas quando a versão do cluster é v1.21.2-r0 ou posterior e a versão do complemento everest é 2.1.23 ou posterior. Recomenda-se a versão 2.1.23 ou posterior. ● Os EVs locais são suportados apenas quando a versão do cluster é v1.21.2-r0 ou posterior e a versão do complemento everest é 2.1.23 ou posterior. **Volumes persistentes locais** e **EVs locais** suportam os seguintes modos de gravação: ● **Linear**: um volume lógico linear integra um ou mais volumes físicos. Os dados são gravados no próximo volume físico quando o anterior é usado. ● **Striped**: um volume lógico distribuído distribui dados em blocos do mesmo tamanho e os armazena em vários volumes físicos em sequência, permitindo que os dados sejam lidos e gravados simultaneamente. Um pool de armazenamento que consiste em volumes distribuídos não pode ser dimensionado. Essa opção só pode ser selecionada quando existirem vários volumes.

Configurações da rede

Configure os recursos de rede para permitir o acesso de nós e aplicações em contêiner.

Tabela 3-13 Parâmetros de configuração

Parâmetro	Descrição
Node Subnet	A sub-rede de nó selecionada durante a criação do cluster é usada por padrão. Você pode escolher outra sub-rede em vez disso.
Node IP Address	Endereço IP do nó especificado. Por padrão, o valor é alocado aleatoriamente.
EIP	Um ECS sem um EIP vinculado não pode acessar a Internet ou ser acessado por redes públicas. O valor padrão é **Do not use**. **Use existing** e **Auto create** são suportados.

Configurações avançadas

Configure recursos avançados de nó, como rótulos, manchas e comando de inicialização.

Tabela 3-14 Parâmetros de configuração avançadas

Parâmetro	Descrição
Kubernetes Label	Um par de chave e valor adicionado a um objeto do Kubernetes (como um pod). Um máximo de 20 rótulos podem ser adicionados. Os rótulos podem ser usados para distinguir nós. Com as configurações de afinidade da carga de trabalho, os pods de contêiner podem ser agendados para um nó especificado. Para obter mais informações, consulte **Rótulos e seletores**.
Resource Tag	Você pode adicionar tags de recursos para classificar recursos. Você pode criar **tagspredefinidas** no console do TMS. As tags predefinidas estão disponíveis para todos os recursos que suportam tags. Você pode usar tags predefinidas para melhorar a criação de tags e a eficiência da migração de recursos. Para obter detalhes, consulte **Criação de tags predefinidas**. O CCE criará automaticamente a tag "CCE-Dynamic-Provisioning-Node=*node id*".
Taint	Este parâmetro é deixado em branco por padrão. Você pode adicionar manchas para configurar a antiafinidade para o nó. Um máximo de 20 manchas são permitidas para cada nó. Cada mancha contém os seguintes parâmetros: ● **Key**: uma chave deve conter de 1 a 63 caracteres, começando com uma letra ou dígito. Apenas letras, dígitos, hifens (-), sublinhados (_) e pontos (.) são permitidos. Um nome de subdomínio do DNS pode ser usado como prefixo de uma chave. ● **Value**: um valor deve começar com uma letra ou dígito e pode conter no máximo 63 caracteres, incluindo letras, dígitos, hifens (-) e pontos (.). ● **Effect**: as opções disponíveis são **NoSchedule**, **PreferNoSchedule** e **NoExecute**. Para mais detalhes, consulte **Gerenciamento de manchas de nó**. **NOTA** Para um cluster v1.19 ou anterior, a carga de trabalho pode ter sido agendada para um nó antes de a mancha ser adicionada. Para evitar tal situação, selecione um cluster v1.19 ou posterior.
Max. Pods	Número máximo de pods que podem ser executados no nó, incluindo os pods padrão do sistema. Intervalo de valor: 16 a 256 Esse limite impede que o nó seja sobrecarregado com pods. Esse número também é decidido por outros fatores. Para mais detalhes, consulte **Número máximo de pods que podem ser criados em um nó**.

Parâmetro	Descrição
ECS Group	Um grupo de ECS agrupa logicamente os ECS. Os ECSs do mesmo grupo de ECS cumprem a mesma política associada ao grupo de ECS. **Anti-affinity**: os ECSs em um grupo de ECS são implementados em hosts físicos diferentes para melhorar a confiabilidade do serviço. Selecione um grupo de ECS existente ou clique em **Add ECS Group** para criar um. Depois que o grupo de ECS for criado, clique no botão de atualizar.
Pre-installation Command	Insira comandos. Um máximo de 1.000 caracteres são permitidos. O script será executado antes da instalação do software Kubernetes. Observe que, se o script estiver incorreto, o software Kubernetes pode falhar ao ser instalado.
Post-installation Command	Insira comandos. Um máximo de 1.000 caracteres são permitidos. O script será executado após a instalação do software Kubernetes e não afetará a instalação. **NOTA** Não execute o comando **reboot** no script de pós-instalação para reiniciar o sistema imediatamente. Para reiniciar o sistema, execute o comando **shutdown -r 1** para reiniciar com um atraso de um minuto.
Agency	Uma agência é criada pelo administrador do locatário no console do IAM. Ao criar uma agência, você pode compartilhar seus recursos de servidor em nuvem com outra conta ou confiar a uma pessoa ou equipe mais profissional para gerenciar seus recursos. Se nenhuma agência estiver disponível, clique em **Create Agency** à direita para criar uma.

Passo 4 Configure o número de nós a serem comprados. Em seguida, clique em **Next: Confirm**. Confirme os parâmetros configurados, o preço e as especificações. Assegure-se de que você leu e compreendeu a **Declaração do serviço de gerenciamento de imagens**.

Passo 5 Clique em **Submit**.

Se o nó for cobrado anual/mensalmente, clique em **Pay Now** e siga as instruções na tela para pagar o pedido.

A página de lista de nós é exibida. Se o status do nó for **Running**, o nó é criado com êxito. Demora cerca de 6 a 10 minutos para criar um nó.

Passo 6 Clique em **Back to Node List**. O nó é criado com sucesso se ele muda para o estado **Running**.

----Fim

Operações relacionadas

Criação de um script de injeção de nó.

3.5 Adição de nós para gerenciamento

Cenário

No CCE, você pode criar um nó (**Criação de um nó**) ou adicionar nós existentes (ECSs ou/ BMSs) ao cluster. Esses nós podem ser cobrados **no modo anual/mensal ou pagamento por uso**.

AVISO

- Enquanto um ECS estiver sendo aceito em um cluster, o sistema operacional do ECS será redefinido para a imagem padrão do sistema operacional fornecida pelo CCE para garantir a estabilidade do nó. O console do CCE solicita que você selecione o sistema operacional e o modo de logon durante a reinicialização.
- As informações do LVM, incluindo grupos de volumes (VGs), volumes lógicos (LVs) e volumes físicos (PVs), serão excluídas dos discos do sistema e dos discos de dados anexados aos ECSs selecionados durante o gerenciamento. Certifique-se de que o backup das informações tenha sido feito.
- Enquanto um ECS estiver sendo aceito em um cluster, não execute nenhuma operação no ECS por meio do console do ECS.

Restrições

- A versão do cluster deve ser 1.15 ou posterior.
- Os nós do Kunpeng podem ser gerenciados apenas por clusters de v1.19 a v1.23.
- Você pode gerenciar nós do ECS, BMS e DeH, mas não nós do HECS.
- Se **IPv6** estiver habilitado para um cluster, apenas os nós em uma sub-rede com IPv6 habilitado poderão ser aceitos e gerenciados. Se **IPv6** não estiver habilitado para o cluster, somente os nós em uma sub-rede sem IPv6 habilitado poderão ser aceitos.
- Se uma senha ou chave tiver sido definida quando o nó original da VM foi criado, redefina a senha ou a chave durante o gerenciamento. A senha ou chave original se tornará inválida.
- Os nós em um cluster do CCE Turbo devem suportar sub-ENIs ou estar vinculados a pelo menos 16 ENIs. Para obter detalhes sobre as especificações do nó, consulte os nós que podem ser selecionados no console quando você cria um nó.
- O sistema operacional Ubuntu não é suportado quando os nós do BMS são gerenciados.
- Discos de dados que foram particionados serão ignorados durante o gerenciamento do nó. Certifique-se de que haja pelo menos um disco de dados não particionado que atenda às especificações esteja anexado ao nó.

Pré-requisitos

Um servidor de nuvem que atenda às seguintes condições pode ser aceito:

- O nó a ser aceito deve estar no estado **Running** e não ser usado por outros clusters. Além disso, o nó a ser aceite não transporta a tag CCE-Dynamic-Provisioning-Node.

- O nó a ser aceito e o cluster devem estar na mesma VPC. (Se a versão do cluster for anterior à v1.13.10, o nó a ser aceito e o cluster do CCE devem estar na mesma sub-rede.)
- Os discos de dados devem ser anexados aos nós a serem gerenciados. Um disco local (disco intensivo) ou um disco de dados de pelo menos 20 GiB pode ser anexado ao nó, e quaisquer discos de dados já anexados não podem ser menores que 10 GiB. Para obter detalhes sobre como anexar um disco de dados, consulte **Adição de um disco a um ECS**.
- O nó a ser aceito tem CPU de 2 núcleos ou mais, 4 GiB ou mais de memória e apenas uma NIC.
- Se um projeto empresarial for usado, o nó a ser aceito e o cluster devem estar no mesmo projeto empresarial. Caso contrário, os recursos não podem ser identificados durante a gestão. Como resultado, o nó não pode ser aceito.
- Somente servidores em nuvem com as mesmas especificações, AZ e configuração de disco de dados podem ser adicionados em lotes.

Procedimento

Passo 1 Efetue logon no console do CCE e vá para o cluster onde reside o nó a ser aceito.

Passo 2 No painel de navegação, escolha **Nodes**. Na página exibida, clique em **Accept Node** no canto superior direito.

Passo 3 Especifique os parâmetros do nó.

Compute Settings

Tabela 3-15 Parâmetros de configuração

Parâmetro	Descrição
Specifications	Clique em **Select Cloud Server** e selecione os servidores a serem aceitos. Você pode selecionar vários servidores de nuvem para gerenciamento de lotes. No entanto, apenas os servidores em nuvem com as mesmas especificações, AZ e configuração de disco de dados podem ser adicionados em lotes. Se um servidor de nuvem contiver vários discos de dados, selecione um deles para o tempo de execução do contêiner e o kubelet.
Container engine	Clusters do CCE suportam Docker e containerd em alguns cenários. ● Nós que executam CentOS, Ubuntu ou EulerOS 2.9 suportam containerd. Nós Arm executando EulerOS 2.5 ou EulerOS 2.8 não suportam containerd. ● Os clusters de rede da VPC da v1.23 e versões posteriores suportam containerd. Clusters de rede de túnel de v1.23.2-r0 e versões posteriores suportam containerd. ● Para um cluster do CCE Turbo, **Docker** e o **containerd** são suportados. Para obter detalhes, consulte **Mapeamento entre sistemas operacionais de nó e mecanismos de contêiner**.

Parâmetro	Descrição
OS	**Public image**: selecione um SO para o nó. **Private image**: você pode usar imagens privadas. Para obter detalhes sobre como criar uma imagem privada, consulte **Criação de uma imagem personalizada de nó do CCE**.
Login mode	● **Password** O nome do usuário padrão é **root**. Digite a senha para efetuar logon no nó e confirme a senha. Certifique-se de lembrar a senha, pois você precisará dela quando fizer logon no nó. ● **Key Pair** Selecione o par de chaves usado para efetuar logon no nó. Você pode selecionar uma chave compartilhada. Um par de chaves é usado para autenticação de identidade quando você entra remotamente em um nó. Se nenhum par de chaves estiver disponível, clique em **Create Key Pair**. Para obter detalhes sobre como criar um par de chaves, consulte **Criação de par de chaves**.

Storage Settings

Configure recursos de armazenamento em um nó para os contêineres em execução nele.

Tabela 3-16 Parâmetros de configuração

Parâmetro	Descrição
System disk	Use diretamente o disco do sistema do servidor de nuvem.
Data disk	**Pelo menos um disco de dados é necessário** para o tempo de execução do contêiner e o kubelet. **O disco de dados não pode ser excluído ou desinstalado. Caso contrário, o nó ficará indisponível.** Clique em **Expand** e selecione **Allocate Disk Space** para definir o espaço em disco ocupado pelo tempo de execução do contêiner para armazenar os diretórios de trabalho, os dados da imagem do contêiner e os metadados da imagem. Para obter detalhes sobre como alocar espaço em disco de dados, consulte **Alocação de espaço em disco de dados**. Para outros discos de dados, um disco bruto é criado sem qualquer processamento por padrão. Você também pode clicar em **Expand** e selecionar **Mount Disk** para montar o disco de dados em um diretório especificado. Discos de dados também podem ser usados como **PVs locais** e **EVs locais**.

Advanced Settings

Tabela 3-17 Parâmetros de configuração avançadas

Parâmetro	Descrição
Kubernetes label	Clique em **Add** para definir o par chave-valor anexado aos objetos do Kubernetes (como pods). Um máximo de 20 rótulos podem ser adicionados. Os rótulos podem ser usados para distinguir nós. Com as configurações de afinidade da carga de trabalho, os pods de contêiner podem ser agendados para um nó especificado. Para obter mais informações, consulte **Rótulos e seletores**.
Resource tag	Você pode adicionar tags de recursos para classificar recursos. Você pode criar **tags predefinidas** no console do TMS. As tags predefinidas estão disponíveis para todos os recursos que suportam tags. Você pode usar tags predefinidas para melhorar a criação de tags e a eficiência da migração de recursos. Para obter detalhes, consulte **Criação de tags predefinidas**. O CCE criará automaticamente a tag "CCE-Dynamic-Provisioning-Node=*node id*".
Taint	Este campo é deixado em branco por padrão. Você pode adicionar manchas para configurar a antiafinidade para o nó. Um máximo de 20 manchas são permitidas para cada nó. Cada mancha contém os seguintes parâmetros: ● **Key**: uma chave deve conter de 1 a 63 caracteres, começando com uma letra ou dígito. Apenas letras, dígitos, hifens (-), sublinhados (_) e pontos (.) são permitidos. Um nome de subdomínio do DNS pode ser usado como prefixo de uma chave. ● **Value**: um valor deve começar com uma letra ou dígito e pode conter no máximo 63 caracteres, incluindo letras, dígitos, hifens (-) e pontos (.). ● **Effect**: as opções disponíveis são **NoSchedule**, **PreferNoSchedule** e **NoExecute**. **AVISO** ● Se manchas forem usadas, você deverá configurar tolerâncias nos arquivos YAML dos pods. Caso contrário, o aumento de escala pode falhar ou os pods não podem ser programados nos nós adicionados. ● Depois que um pool de nós é criado, você pode clicar em **Edit** para modificar sua configuração. A modificação será sincronizada com todos os nós no pool de nós.
Max. pod	Número máximo de pods que podem ser executados no nó, incluindo os pods padrão do sistema. Intervalo de valor: 16 a 256 Esse limite impede que o nó seja sobrecarregado com pods.
Pre-installation command	Insira comandos. Um máximo de 1.000 caracteres são permitidos. O script será executado antes da instalação do software Kubernetes. Observe que, se o script estiver incorreto, o software Kubernetes pode falhar ao ser instalado.

Parâmetro	Descrição
Post-installation command	Insira comandos. Um máximo de 1.000 caracteres são permitidos. O script será executado após a instalação do software Kubernetes e não afetará a instalação.

Passo 4 Clique em **Next: Confirm**. Certifique-se de que leu e entendeu a **Image Management Service Statement**. Clique em **Submit**.

----Fim

3.6 Logon em um nó

Restrições

- Se você usar o SSH para efetuar logon em um nó (um ECS), certifique-se de que o ECS já tenha um EIP (um endereço IP público).
- Somente o logon em um ECS em execução é permitido.
- Somente o usuário **root** pode fazer logon em um servidor Linux.

Modos de logon

Você pode efetuar logon em um ECS em um dos seguintes modos:

- Console de gerenciamento (VNC)

  Se um ECS não tiver EIP, efetue logon no console do ECS e clique em **Remote Login** na mesma linha do ECS.

  Para obter detalhes, consulte **Logon usando o VNC**.
- SSH

  Esse modo se aplica somente aos ECSs que executam o Linux. Normalmente, você pode usar uma ferramenta de logon remoto, como PuTTY, Xshell e SecureCRT, para fazer logon no seu ECS. Se nenhuma das ferramentas de logon remoto puder ser usada, faça logon no console do ECS e clique em **Remote Login** na mesma linha do ECS para exibir o status da conexão e o status de execução do ECS.

  NOTA

  - Você pode usar uma chave SSH ou uma senha SSH para logon. Para obter detalhes, consulte **Logon usando uma chave SSH** e **Logon usando uma senha SSH**
  - Quando você usa o sistema operacional Windows para efetuar logon em um nó Linux, defina **Auto-login username** como **root**.
  - O console do CCE não oferece suporte à atualização do sistema operacional do nó. Não atualize o sistema operacional do nó usando o comando **yum update**. Caso contrário, os componentes de rede de contêiner estarão indisponíveis. Para detalhes sobre como restaurar manualmente a rede de contêineres, veja **O que fazer se a rede de contêineres se tornar indisponível após a atualização de yum ser usada para atualizar o SO?**

Tabela 3-18 Modos de logon do ECS de Linux

Vinculação de EIP	SO local	Método de conexão
Sim	Windows	Use uma ferramenta de logon remoto, como PuTTY ou Xshell. ● Autenticação de senha SSH: **logon usando uma senha SSH** ● Autenticação da chave SSH: **logon usando uma chave SSH**
Sim	Linux	Executar comandos. ● Autenticação de senha SSH: **logon usando uma senha SSH** ● Autenticação da chave SSH: **logon usando uma chave SSH**
Sim/Não	Windows/Linux	Logon remoto usando o console de gerenciamento: **logon usando o VNC**

3.7 Nós de gerenciamento

3.7.1 Gerenciamento de rótulos de nó

Você pode adicionar rótulos diferentes aos nós e definir atributos diferentes para rótulos. Usando esses rótulos de nó, você pode entender rapidamente as características de cada nó.

Cenário de uso de rótulo de nó

Os rótulos de nó são usados principalmente nos seguintes cenários:

- Gerenciamento de nó: os rótulos de nó são usados para classificar nós.
- Afinidade e antiafinidade entre uma carga de trabalho e nó:
  - Diferentes cargas de trabalho têm diferentes requisitos de recursos, como CPU, memória e I/O. Se uma carga de trabalho consome muitos recursos em um cluster, outras cargas de trabalho no mesmo cluster podem não ser executadas corretamente. Nesse caso, é aconselhável adicionar rótulos diferentes aos nós. Ao implantar uma carga de trabalho, você pode selecionar nós com rótulos especificados para implantação de afinidade para garantir a operação normal do sistema. Caso contrário, a implantação de antiafinidade de nó pode ser usada.
  - Um sistema pode ser dividido em vários módulos. Cada módulo é composto por vários microsserviços. Para garantir uma O&M eficiente, você pode adicionar um rótulo de módulo a cada nó para que cada módulo possa ser implantado no nó correspondente. Dessa forma, os módulos não interferem uns com os outros e os microsserviços podem ser facilmente mantidos em seus nós.

Rótulo inerente de um nó

Depois que um nó é criado, alguns rótulos fixos existem e não podem ser excluídos. Para obter detalhes sobre esses rótulos, consulte **Tabela 3-19**.

NOTA

Não altere manualmente os rótulos inerentes que são adicionados automaticamente a um nó. Se o valor alterado manualmente entrar em conflito com o valor do sistema, o valor do sistema prevalece.

Tabela 3-19 Rótulos inerentes de um nó

Chave	Descrição
Novo: topology.kubernetes.io/region Antigo: failure-domain.beta.kubernetes.io/region	Região onde o nó está localizado
Novo: topology.kubernetes.io/zone Antigo: failure-domain.beta.kubernetes.io/zone	AZ onde o nó está localizado
Novo: node.kubernetes.io/baremetal Antigo: failure-domain.beta.kubernetes.io/is-baremetal	Se o nó é um nó bare metal **false** indica que o nó não é um nó bare metal.
node.kubernetes.io/instance-type	Especificações do nó
kubernetes.io/arch	Arquitetura do processador de nó
kubernetes.io/hostname	Nome do nó
kubernetes.io/os	Tipo de SO
node.kubernetes.io/subnetid	ID da sub-rede onde o nó está localizado.
os.architecture	Arquitetura do processador de nó Por exemplo, **amd64** indica um processador AMD64-bit.
os.name	Nome do OS do nó
os.version	Versão do kernel do OS do nó
node.kubernetes.io/container-engine	Mecanismo de contêiner usado pelo nó.
accelerator/huawei-npu	Rótulos de nó de NPU.
accelerator	Rótulos de nó de GPU.
cce.cloud.com/cce-nodepool	O rótulo dedicado de um nó em um pool de nós.

Adicionar ou excluir um rótulo de nó

Passo 1 Efetue logon no console do CCE.

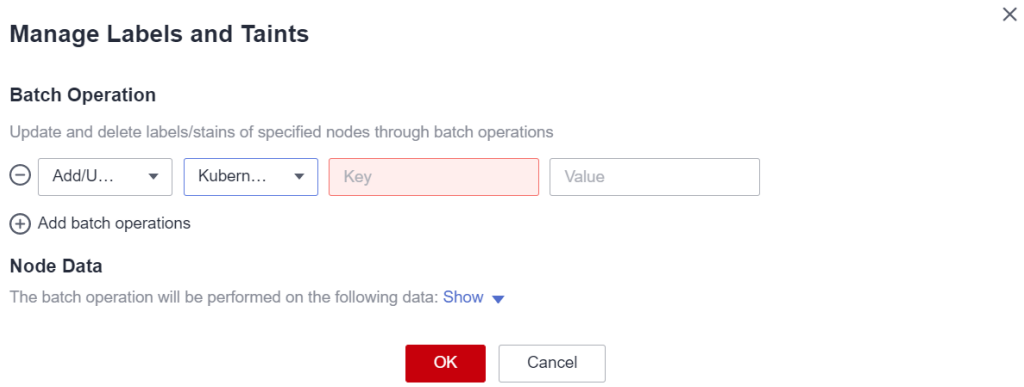
Passo 2 Clique no nome do cluster, acesse a página de detalhes do cluster e escolha **Nodes** no painel de navegação. Na página exibida, selecione um nó e clique em **Manage Labels and Taints**.

Passo 3 Na caixa de diálogo exibida, clique em **Add batch operations** em **Batch Operation** e escolha **Add/Update** ou **Delete**.

Insira a chave e o valor do rótulo a ser adicionado ou excluído e clique em **OK**.

Por exemplo, a chave é **deploy_qa** e o valor é **true**, indicando que o nó é usado para implementar o ambiente de QA (teste).

Figura 3-1 Adicionar um rótulo de nó



Passo 4 Depois que o rótulo for adicionado, verifique o rótulo adicionado nos dados do nó.

----Fim

3.7.2 Gerenciamento de manchas de nó

Manchas permitem que um nó repela pods específicos para evitar que esses pods sejam agendados para o nó.

Manchas

Uma mancha é um par de chave-valor associado a um efeito. Os seguintes efeitos estão disponíveis:

- NoSchedule: nenhum pod será agendado para o nó, a menos que tenha uma tolerância correspondente. Os pods existentes não serão despejados do nó.
- PreferNoSchedule: o Kubernetes impede que pods que não podem tolerar essa mancha sejam agendados no nó.
- NoExecute: se o pod estiver sendo executado em um nó, o pod será despejado do nó. Se o pod não estiver sendo executado em um nó, o pod não será agendado para o nó.

Para adicionar uma mancha a um nó, execute o comando **kubectl taint node** *nodename* da seguinte forma:

```
$ kubectl get node
NAME             STATUS   ROLES    AGE    VERSION
192.168.10.170   Ready    <none>   73d    v1.19.8-r1-CCE21.4.1.B003
192.168.10.240   Ready    <none>   4h8m   v1.19.8-r1-CCE21.6.1.2.B001
$ kubectl taint node 192.168.10.240 key1=value1:NoSchedule
node/192.168.10.240 tainted
```


Para visualizar a configuração do mancha, execute os comandos **describe** e **get** da seguinte forma:

```
$ kubectl describe node 192.168.10.240
Name:               192.168.10.240
...
Taints:             key1=value1:NoSchedule
...
$ kubectl get node 192.168.10.240 -oyaml
apiVersion: v1
...
spec:
  providerID: 06a5ea3a-0482-11ec-8e1a-0255ac101dc2
  taints:
  - effect: NoSchedule
    key: key1
    value: value1
...
```

Para remover uma mancha, execute o seguinte comando com um hífen (-) adicionado após **NoSchedule**:

```
$ kubectl taint node 192.168.10.240 key1=value1:NoSchedule-
node/192.168.10.240 untainted
$ kubectl describe node 192.168.10.240
Name:               192.168.10.240
...
Taints:             <none>
...
```

No console do CCE, você também pode gerenciar manchas de um nó em lotes.

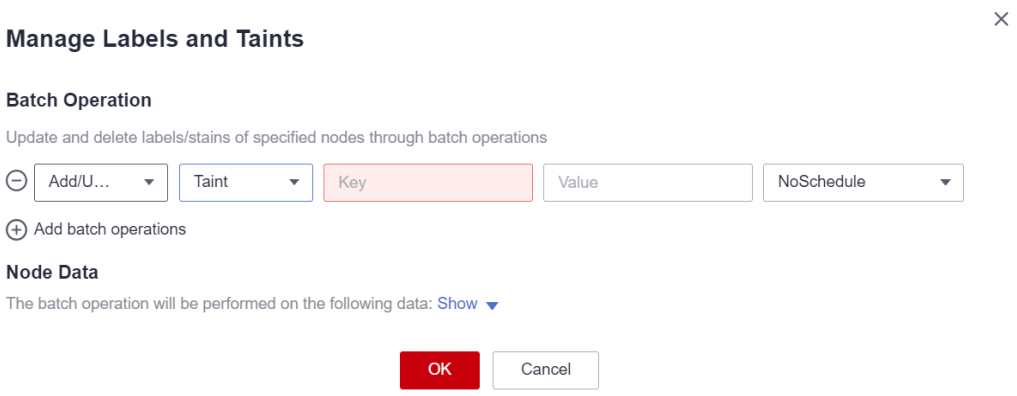
Passo 1 Efetue logon no console do CCE.

Passo 2 Clique no nome do cluster, acesse a página de detalhes do cluster e escolha **Nodes** no painel de navegação. Na página exibida, selecione um nó e clique em **Manage Labels and Taints**.

Passo 3 Na caixa de diálogo exibida, clique em **Add batch operations** em **Batch Operation**, escolha **Add/Update** e selecione **Taint**.

Digite a chave e o valor da mancha a ser adicionado, selecione o efeito de mancha e clique em **OK**.

Figura 3-2 Adicionar uma mancha



Passo 4 Depois que a mancha for adicionada, verifique a mancha adicionada nos dados do nó.

----Fim

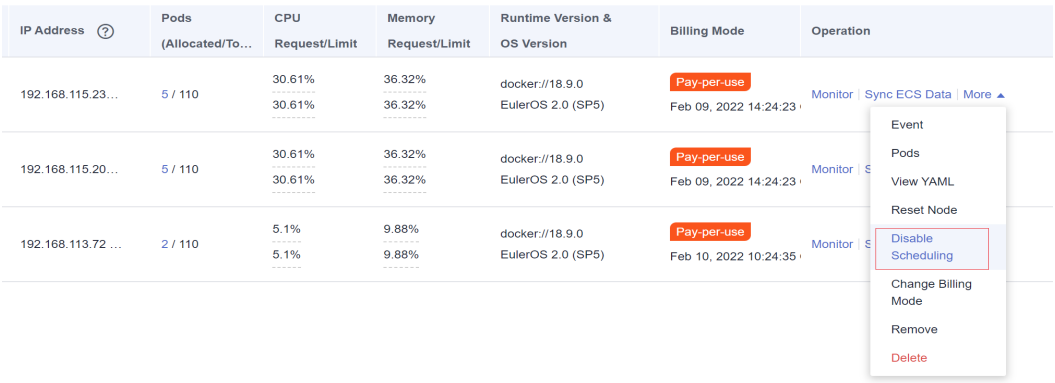
Manchas do sistema

Quando alguns problemas ocorrem em um nó, o Kubernetes adiciona automaticamente uma mancha ao nó. As manchas embutidas são as seguintes:

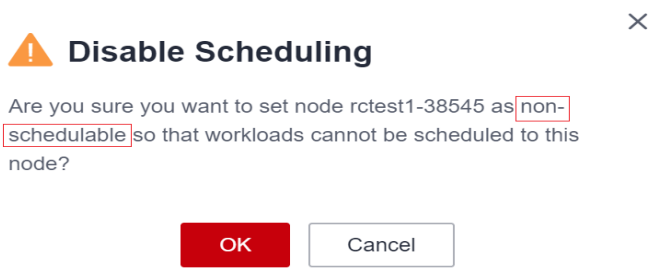
- node.kubernetes.io/not-ready: o nó não está pronto. O valor **Ready** do nó é **False**.
- node.kubernetes.io/unreachable: o controlador de nó não pode acessar o nó. O valor de nó **Ready** é **Unknown**.
- node.kubernetes.io/memory-pressure: a memória do nó está se aproximando do limite superior.
- node.kubernetes.io/disk-pressure: o espaço em disco do nó está se aproximando do limite superior.
- node.kubernetes.io/pid-pressure: os PIDs do nó estão se aproximando do limite superior.
- node.kubernetes.io/network-unavailable: o nó da rede não está indisponível.
- node.kubernetes.io/unschedulable: o nó não pode ser agendado.
- node.cloudprovider.kubernetes.io/uninitialized: se um driver de plataforma de nuvem externa for especificado quando o kubelet for iniciado, o kubelet adicionará um taint ao nó atual e o marcará como indisponível. Depois que um controlador do **cloud-controller-manager** inicializar o nó, o kubelet excluirá a mancha.

Configurações de agendamento de nó

Para configurar a programação, faça logon no console do CCE, clique no cluster, escolha **Nodes** no painel de navegação e clique em **More** > **Disable Scheduling** na coluna **Operation** de um nó na lista de nós.



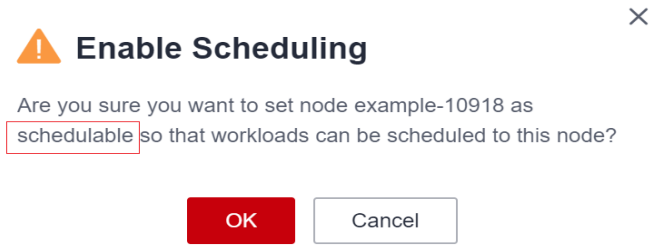
Na caixa de diálogo exibida, clique em **OK** para definir o nó a ser unschedulable.



Esta operação adicionará uma mancha ao nó. Você pode usar o kubectl para visualizar o conteúdo da mancha.

```
$ kubectl describe node 192.168.10.240
...
Taints:             node.kubernetes.io/unschedulable:NoSchedule
...
```

No console do CCE, escolha **More** > **Enable scheduling** para remover a mancha e ajustar o nó para ser programável.



Tolerâncias

Tolerâncias são aplicadas aos pods e permitem (mas não exigem) que os pods se programem em nós com manchas correspondentes.

Manchas e tolerâncias trabalham juntas para garantir que os pods não sejam agendados para nós inadequados. Uma ou mais manchas são aplicadas a um nó. Isso marca que o nó não deve aceitar quaisquer vagens que não toleram as manchas.

Exemplo:

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  labels:
    env: test
spec:
  containers:
  - name: nginx
    image: nginx
    imagePullPolicy: IfNotPresent
  tolerations:
  - key: "key1"
    operator: "Equal"
    value: "value1"
    effect: "NoSchedule"
```

No exemplo anterior, o rótulo de tolerância do pod é key1=value1 e o efeito de mancha é NoSchedule. Portanto, o pod pode ser programado no nó correspondente.

Você também pode configurar tolerâncias semelhantes às informações a seguir, o que indica que o pod pode ser agendado em um nó quando o nó tiver a mancha key1:

```
tolerations:
- key: "key1"
  operator: "Exists"
  effect: "NoSchedule"
```


3.7.3 Redefinição de um nó

Cenário

Você pode redefinir um nó para modificar a configuração do nó, como o sistema operacional do nó e o modo de logon.

A redefinição de um nó reinstalará o sistema operacional do nó e o software Kubernetes no nó. Se um nó não estiver disponível porque você modifica a configuração do nó, você poderá redefinir o nó para corrigir a falha.

Restrições

- Para clusters do CCE e clusters do CCE Turbo, a versão deve ser v1.13 ou posterior para oferecer suporte à redefinição de nó.
- Para clusters de Kunpeng, a versão deve ser v1.15 ou posterior para oferecer suporte à redefinição de nó.

Precauções

- Somente os nós de trabalho podem ser redefinidos. Se o nó ainda não estiver disponível após a redefinição, exclua o nó e crie um novo.
- **A redefinição de um nó reinstalará o sistema operacional do nó e interromperá os serviços de carga de trabalho em execução no nó. Portanto, realize a operação fora dos horários de pico.**
- **Os dados no disco do sistema e nos discos de dados do Docker serão limpos. Faça backup de dados importantes antes de redefinir o nó.**
- **Quando um disco de dados extra é montado em um nó, os dados nesse disco serão apagados se o disco não tiver sido desmontado antes da reinicialização do nó. Para evitar a perda de dados, faça backup dos dados com antecedência e monte o disco de dados novamente após a conclusão da redefinição do nó.**
- Os endereços IP dos pods de carga de trabalho no nó serão alterados, mas o acesso à rede de contêiner não será afetado.
- Existe uma quota de disco EVS remanescente.
- Enquanto o nó está sendo excluído, o back-end irá definir o nó para o estado não programável.
- A redefinição de um nó causará perda de dados de PVC/PV para o **PV local** associado ao nó. Esses PVCs e PVs não podem ser restaurados ou usados novamente. Nesse cenário, o pod que usa o PV local é expulso do nó de reinicialização. Um novo pod é criado e permanece no estado pendente. Isso ocorre porque a PVC usada pelo pod tem um rótulo de nó, devido ao qual o pod não pode ser programado. Depois que o nó é redefinido, o pod pode ser agendado para o nó de redefinição. Nesse caso, o pod está sempre no estado de criação porque o volume lógico subjacente correspondente à PVC não existe.

Procedimento

O novo console permite que você redefina os nós em lotes. Você também pode usar uma imagem privada para redefinir nós em lotes.

Passo 1 Efetue logon no console do CCE.

Passo 2 Clique no nome do cluster para acessar a página de detalhes do cluster, escolha **Nodes** no painel de navegação e selecione um ou vários nós a serem redefinidos na lista à direita. Escolha **More** > **Reset**.

Passo 3 Na caixa de diálogo exibida, clique em **Next**.

- Para nós no pool de nós do DefaultPool, a página de configuração de parâmetros é exibida. Defina os parâmetros referindo-se a **Passo 4**.
- Para um nó que você cria em um pool de nós, a redefinição do nó não suporta a configuração de parâmetros. Você pode usar diretamente a imagem de configuração do pool de nós para redefinir o nó.

Passo 4 Especifique os parâmetros do nó.

Compute Settings

Tabela 3-20 Parâmetros de configuração

Parâmetro	Descrição
Specifications	As especificações não podem ser modificadas quando você redefine um nó.
Container Engine	Clusters do CCE suportam Docker e containerd em alguns cenários. ● Nós que executam CentOS, Ubuntu ou EulerOS 2.9 suportam containerd. Nós Arm executando EulerOS 2.5 ou EulerOS 2.8 não suportam containerd. ● Os clusters de rede da VPC da v1.23 e versões posteriores suportam containerd. Clusters de rede de túnel de v1.23.2-r0 e versões posteriores suportam containerd. ● Para um cluster do CCE Turbo, **Docker** e o **containerd** são suportados. Para obter detalhes, consulte **Mapeamento entre sistemas operacionais de nó e mecanismos de contêiner**.
OS	**Public image**: selecione um SO para o nó. **Private image**: você pode usar imagens privadas. Para obter detalhes sobre como criar uma imagem privada, consulte **Criação de uma imagem personalizada de nó do CCE**.
Login Mode	● **Password** O nome do usuário padrão é **root**. Digite a senha para efetuar logon no nó e confirme a senha. Certifique-se de lembrar a senha, pois você precisará dela quando fizer logon no nó. ● **Key Pair** Selecione o par de chaves usado para efetuar logon no nó. Você pode selecionar uma chave compartilhada. Um par de chaves é usado para autenticação de identidade quando você entra remotamente em um nó. Se nenhum par de chaves estiver disponível, clique em **Create Key Pair**. Para obter detalhes sobre como criar um par de chaves, consulte **Criação de par de chaves**.

Storage Settings

Configure recursos de armazenamento em um nó para os contêineres em execução nele.

Tabela 3-21 Parâmetros de configuração

Parâmetro	Descrição
System disk	Use diretamente o disco do sistema do servidor de nuvem.
Data disk	**Pelo menos um disco de dados é necessário** para o tempo de execução do contêiner e o kubelet. **O disco de dados não pode ser excluído ou desinstalado. Caso contrário, o nó ficará indisponível.** Clique em **Expand** e selecione **Allocate Disk Space** para definir o espaço em disco ocupado pelo tempo de execução do contêiner para armazenar os diretórios de trabalho, os dados da imagem do contêiner e os metadados da imagem. Para obter detalhes sobre como alocar espaço em disco de dados, consulte **Alocação de espaço em disco de dados**. Para outros discos de dados, um disco bruto é criado sem qualquer processamento por padrão. Você também pode clicar em **Expand** e selecionar **Mount Disk** para montar o disco de dados em um diretório especificado. Discos de dados também podem ser usados como **PVs locais** e **EVs locais**.

Advanced Settings

Tabela 3-22 Parâmetros de configuração avançadas

Parâmetro	Descrição
Kubernetes label	Clique em **Add** para definir o par chave-valor anexado aos objetos do Kubernetes (como pods). Um máximo de 20 rótulos podem ser adicionados. Os rótulos podem ser usados para distinguir nós. Com as configurações de afinidade da carga de trabalho, os pods de contêiner podem ser agendados para um nó especificado. Para obter mais informações, consulte **Rótulos e seletores**.
Resource tag	Você pode adicionar tags de recursos para classificar recursos. Você pode criar **tags predefinidas** no console do TMS. As tags predefinidas estão disponíveis para todos os recursos que suportam tags. Você pode usar tags predefinidas para melhorar a criação de tags e a eficiência da migração de recursos. Para obter detalhes, consulte **Criação de tags predefinidas**. O CCE criará automaticamente a tag "CCE-Dynamic-Provisioning-Node=*node id*".

Parâmetro	Descrição
Taint	Este campo é deixado em branco por padrão. Você pode adicionar manchas para configurar a antiafinidade para o nó. Um máximo de 20 manchas são permitidas para cada nó. Cada mancha contém os seguintes parâmetros: ● **Key**: uma chave deve conter de 1 a 63 caracteres, começando com uma letra ou dígito. Apenas letras, dígitos, hifens (-), sublinhados (_) e pontos (.) são permitidos. Um nome de subdomínio do DNS pode ser usado como prefixo de uma chave. ● **Value**: um valor deve começar com uma letra ou dígito e pode conter no máximo 63 caracteres, incluindo letras, dígitos, hifens (-) e pontos (.). ● **Effect**: as opções disponíveis são **NoSchedule**, **PreferNoSchedule** e **NoExecute**. **AVISO** ● Se manchas forem usadas, você deverá configurar tolerâncias nos arquivos YAML dos pods. Caso contrário, o aumento de escala pode falhar ou os pods não podem ser programados nos nós adicionados. ● Depois que um pool de nós é criado, você pode clicar em **Edit** para modificar sua configuração. A modificação será sincronizada com todos os nós no pool de nós.
Max. pod	Número máximo de pods que podem ser executados no nó, incluindo os pods padrão do sistema. Intervalo de valor: 16 a 256 Esse limite impede que o nó seja sobrecarregado com pods.
Pre-installation command	Insira comandos. Um máximo de 1.000 caracteres são permitidos. O script será executado antes da instalação do software Kubernetes. Observe que, se o script estiver incorreto, o software Kubernetes pode falhar ao ser instalado.
Post-installation command	Insira comandos. Um máximo de 1.000 caracteres são permitidos. O script será executado após a instalação do software Kubernetes e não afetará a instalação.

Passo 5 Clique em **Next: Confirm**. Certifique-se de que leu e entendeu a **Declaração do serviço de gerenciamento de imagens**.

Passo 6 Clique em **Submit**.

----Fim

3.7.4 Remoção de um nó

Cenário

A remoção de um nó de um cluster reinstalará o sistema operacional do nó e limpará os componentes do CCE no nó.

Remover um nó não excluirá o servidor correspondente ao nó. Você é aconselhado a remover os nós fora do horário de pico para evitar impactos em seus serviços.

Depois que um nó é removido do cluster, o nó ainda está em execução e incorre em taxas.

Restrições

- Os nós só podem ser removidos quando o cluster estiver no status **Available** ou **Unavailable**.
- Um nó do CCE pode ser removido somente quando estiver no status **Active**, **Abnormal** ou **Error**.
- Um nó do CCE no estado **Active** pode ter seu sistema operacional reinstalado e os componentes do CCE cancelados depois que ele é removido.
- Se o sistema operacional não conseguir ser reinstalado depois que o nó for removido, reinstale manualmente o sistema operacional. Após a reinstalação, faça logon no nó e execute o script de liberação para limpar os componentes do CCE. Para mais detalhes, consulte **Manipular a reinstalação do sistema operacional com falha**.
- A remoção de um nó causará perda de dados de PVC/PV para o **PV local** associado ao nó. Esses PVCs e PVs não podem ser restaurados ou usados novamente. Nesse cenário, o pod que usa o PV local é despejado do nó. Um novo pod é criado e permanece no estado pendente. Isso ocorre porque a PVC usada pelo pod tem um rótulo de nó, devido ao qual o pod não pode ser programado.

Precauções

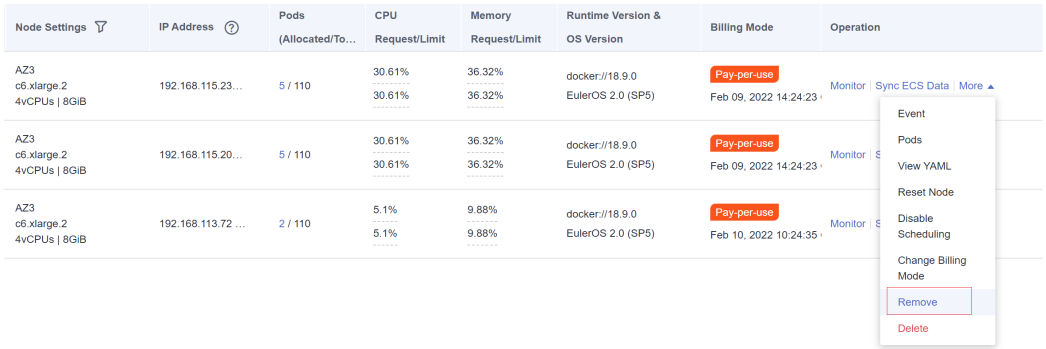
- A remoção de um nó levará à migração de pods, o que pode afetar os serviços. Portanto, realize esta operação fora dos horários de pico.
- Riscos inesperados podem ocorrer durante a operação. Faça backup dos dados com antecedência.
- Enquanto o nó está sendo excluído, o back-end irá definir o nó para o estado não programável.
- Depois de remover o nó e reinstalar o sistema operacional, as partições LVM originais serão limpas e os dados gerenciados pelo LVM serão limpos. Portanto, faça backup dos dados com antecedência.

Procedimento

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster.

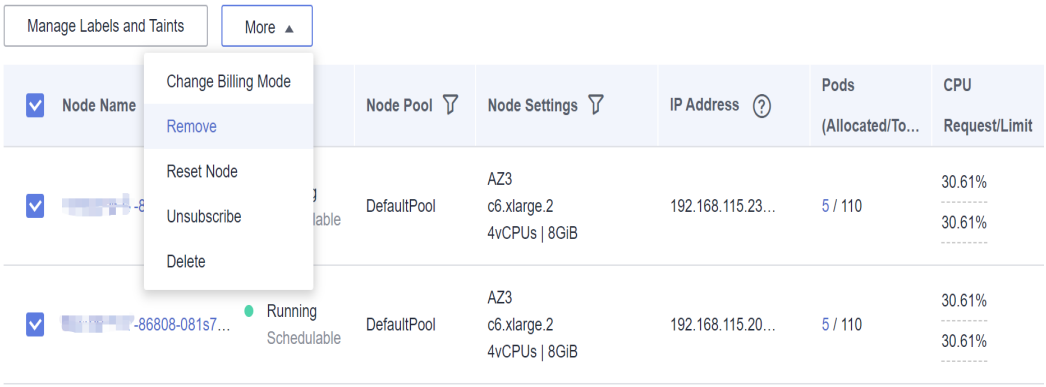
Passo 2 Escolha **Nodes** no painel de navegação e escolha **More** > **Remove** na coluna **Operation** do nó de destino.

Figura 3-3 Remover um nó



Você também pode selecionar vários nós e removê-los de uma vez.

Figura 3-4 Remover vários nós de uma vez



Passo 3 Na caixa de diálogo exibida, configure as informações de logon necessárias para reinstalar o sistema operacional e clique em **Yes**. Aguarde até que o nó seja removido.

Depois que o nó é removido, os pods de carga de trabalho no nó são migrados automaticamente para outros nós disponíveis.

----Fim

Manipular a reinstalação do sistema operacional com falha

Você pode executar as seguintes etapas para reinstalar o sistema operacional e limpar os componentes do CCE no nó se as tentativas anteriores falharem:

Passo 1 Faça logon no console de gerenciamento do servidor e reinstale o sistema operacional. Para obter detalhes, consulte **Alteração do SO**.

Passo 2 Faça logon no servidor e execute os seguintes comandos para limpar os componentes do CCE e os dados do LVM:

Escreva os seguintes scripts no ficheiro **clean.sh**:

```
lsblk
vgs --noheadings | awk '{print $1}' | xargs vgremove -f
pvs --noheadings | awk '{print $1}' | xargs pvremove -f
lvs --noheadings | awk '{print $1}' | xargs -i lvremove -f --select {}
function init_data_disk() {
    all_devices=$(lsblk -o KNAME,TYPE | grep disk | grep -v nvme | awk '{print
```



```
$1}' | awk '{ print "/dev/"$1}')
    for device in ${all_devices[@]}; do
        isRootDisk=$(lsblk -o KNAME,MOUNTPOINT $device 2>/dev/null| grep -E
'[[:space:]]/$' | wc -l )
        if [[ ${isRootDisk} != 0 ]]; then
            continue
        fi
        dd if=/dev/urandom of=${device} bs=512 count=64
        return
    done
    exit 1
}
init_data_disk
lsblk
```

Execute o seguinte comando:

bash clean.sh

----Fim

3.7.5 Sincronização de dados com servidores em nuvem

Cenário

Cada nó em um cluster é um servidor de nuvem ou máquina física. Depois que um nó de cluster é criado, você pode alterar o nome ou as especificações do servidor de nuvem conforme necessário. A modificação das especificações do nó afetará os serviços. Execute a operação nos nós um por um.

Algumas informações dos nós do CCE são mantidas independentemente do console do ECS. Depois de alterar o nome, o EIP, o modo de cobrança ou as especificações de um ECS no console do ECS, sincronize o ECS com o nó de destino no console do CCE. Após a sincronização, as informações em ambos os consoles são consistentes.

Informações comuns do ECS a alterar:

- Nome do ECS (nó): **alteração de um nome de ECS**
- Especificações do nó: **operações gerais para modificar especificações**

Restrições

- Os dados, incluindo o status da VM, os nomes do ECS, o número de CPUs, o tamanho da memória, as especificações do ECS e os endereços IP públicos, podem ser sincronizados.
- Os dados, como o SO e o ID da imagem, não podem ser sincronizados. (Esses parâmetros não podem ser modificados no console do ECS.)

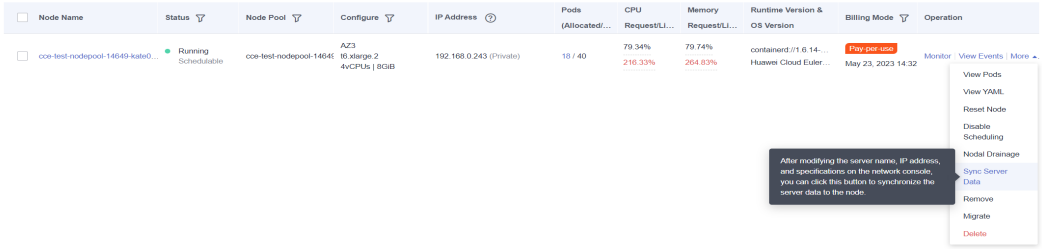
Procedimento

Passo 1 Efetue logon no console do CCE.

Passo 2 Clique no nome do cluster para acessar o console do cluster. Escolha **Nodes** no painel de navegação.

Passo 3 Escolha **More** > **Sync Server Data** ao lado do nó.

Figura 3-5 Sincronizar dados do servidor



Após a conclusão da sincronização, a mensagem **ECS data synchronization requested** é exibida no canto superior direito.

----Fim

3.7.6 Drenagem de um nó

Cenário

Depois de ativar a função de drenagem de nó no console, o sistema define o nó como não programável e despeja com segurança todos os pods que estejam em conformidade com **Regras de drenagem de nó** no nó. Novos pods subsequentes não serão programados para o nó.

Quando um nó está defeituoso, essa função ajuda a isolar rapidamente o nó defeituoso. Os pods despejados serão transferidos do controlador de carga de trabalho para outro nó que pode ser programado corretamente.

Restrições

- Somente os clusters das seguintes versões suportam a função de drenagem de nó:
  - v1.21: v1.21.10-r0 ou mais recente
  - v1.23: v1.23.8-r0 ou mais recente
  - v1.25: v1.25.3-r0 ou mais recente
  - v1.25 ou mais recente
- Para usar a função de drenagem de nó, um usuário do IAM deve ter pelo menos uma das seguintes permissões. Para mais detalhes, consulte **Permissões de namespace (com base no RBAC do Kubernetes)**.
  - cluster-admin (administrador): permissões de leitura e gravação em todos os recursos em todos os namespaces.
  - drainage-editor: drenar um nó.
  - drainage-viewer: visualizar o status de drenagem de nodal, mas não pode drenar um nó.
- Se um **orçamento de interrupção** não for especificado para a carga de trabalho, a função de carga de trabalho poderá ficar indisponível durante o reagendamento do pod.

Regras de drenagem de nó

A função de drenagem de nó expulsa com segurança os pods em um nó. No entanto, para pods que atendem aos seguintes critérios de filtragem, o sistema executa as operações correspondentes:

Critério do filtro	Drenagem forçada ativada	Drenagem forçada desativada
O campo **status.phase** do pod é **Succeeded** ou **Failed**.	Exclusão	Exclusão
O pod não é gerenciado pelo controlador de carga de trabalho.	Exclusão	Cancelamento da drenagem
O pod é gerenciado pelo DaemonSet.	Nenhuma	Cancelamento da drenagem
Um volume do tipo emptyDir é montado no pod.	Despejo	Cancelamento da drenagem
O pod é um **pod estático** gerenciado diretamente pelo kubelet	Nenhum	Nenhum

NOTA

As seguintes operações podem ser realizadas em pods durante a drenagem de nó:

- Exclusão: o pod é excluído do nó atual e não será agendado para outros nós.
- Despejo: o pod é excluído do nó atual e reagendado para outro nó.
- Nenhuma: o pod não será despejado ou excluído.
- Cancelamento da drenagem: se um pod em um nó cancela a drenagem, o processo de drenagem do nó é encerrado e nenhum pod é despejado ou excluída.

Procedimento

Passo 1 Efetue logon no console do CCE e acesse o console do cluster.

Passo 2 No painel de navegação à esquerda, escolha **Nodes**. Na coluna **Operation** do nó de destino, escolha **More** > **Node Drainage**.

Passo 3 Na janela **Nodal Drainage** exibida, defina os parâmetros.

- **Timeout (s)**: a tarefa de drenagem falha automaticamente após o período de tempo limite predefinido. O valor 0 indica que o período de tempo limite não está definido.
- **Forced Drainage**: se esta função estiver ativada, pods gerenciados pelo DaemonSet serão ignorados e pods com volumes de emptyDir e pods não gerenciados pelos controladores serão excluídos. Para mais detalhes, consulte **Regras de drenagem de nó**.

Passo 4 Clique em **OK** e aguarde até que a drenagem de nó esteja completa.

----Fim

3.7.7 Exclusão de um nó

Cenário

Quando um nó em um cluster do CCE é excluído, os serviços em execução no nó também serão excluídos. Tenha cuidado ao realizar esta operação.

Restrições

- Se os nós forem cobrados anualmente/mensalmente, eles não poderão ser excluídos diretamente. Você pode escolher **Billing Center** > **My Orders** para cancelar a assinatura dos nós.
- Os nós de VM que estão sendo usados pelo CCE não suportam cancelamento de assinatura ou exclusão na página do ECS.
- Para um cluster v1.17.11 ou posterior, depois que um servidor bare metal é cancelado ou excluído no console do BMS, o nó correspondente no cluster é excluído automaticamente.
- Excluir um nó causará perda de dados de PVC/PV para os **PVs locais** associado ao nó. Esses PVCs e PVs não podem ser restaurados ou usados novamente. Nesse cenário, o pod que usa o PV local é despejado do nó. Um novo pod é criado e permanece no estado pendente. Isso ocorre porque a PVC usada pelo pod tem um rótulo de nó, devido ao qual o pod não pode ser programado.

Precauções

- A exclusão de um nó levará à migração de pods, o que pode afetar os serviços. Execute esta operação fora dos horários de pico.
- Riscos inesperados podem ocorrer durante a operação. Faça backup dos dados relacionados com antecedência.
- Durante a operação, o back-end definirá o nó para o estado não programável.
- Somente os nós de trabalho podem ser excluídos.

Procedimento

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster.

Passo 2 No painel de navegação, escolha **Nodes**. Na mesma linha do nó que você excluirá, escolha **More** > **Delete**.

Passo 3 Na caixa de diálogo **Delete Node**, clique em **Yes**.

NOTA

- Depois que o nó é excluído, os pods nele são migrados automaticamente para outros nós disponíveis.
- Se os discos e EIPs vinculados ao nó forem recursos importantes, desvincule-os primeiro. Caso contrário, eles serão excluídos com o nó.

----Fim

3.7.8 Alteração de pagamento por uso para anual/mensal

O CCE suporta cobrança de **pagamento por uso** e **anual / mensal**. Um nó de pagamento por uso pode ser alterado para cobrança anual/mensal.

Restrições

- Não é possível alterar os nós de pagamento por uso para anual/mensal no console do ECS.
- Os nós cujo modo de cobrança é alterado para anual/mensal não suportam o escalonamento automático.

Alterar para a cobrança anual/mensal

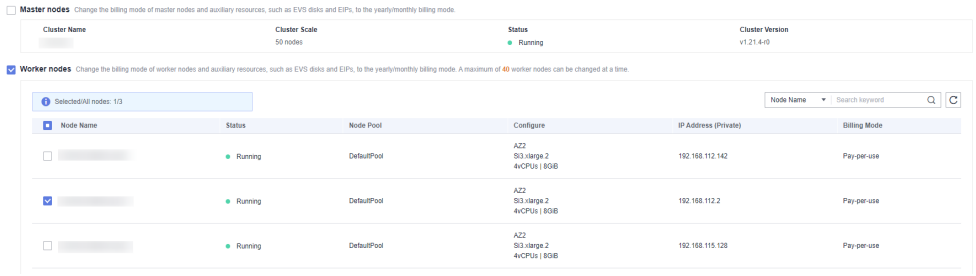
Para fazer isso, execute as seguintes operações:

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster.

Passo 2 No painel de navegação à esquerda, escolha **Nodes**. Na linha do nó de destino, escolha **More** > **Change Billing Mode**.

Passo 3 Na página **Change Billing Mode**, escolha os nós que serão alterados para anual/mensal.

Figura 3-6 Alterar os modos de cobrança para nós principal e de trabalho



Passo 4 Clique em **OK**. Aguarde até que o pedido seja processado e o pagamento seja concluído.

Durante o pagamento, se uma mensagem for exibida indicando **you do not have the permission to access the resource API**, volte para a página anterior e execute a operação novamente.

----Fim

3.7.9 Interrupção de um nó

Cenário

Depois que um nó no cluster é interrompido, os serviços no nó também são interrompidos. Antes de interromper um nó, verifique se a descontinuidade dos serviços no nó não resultará em impactos adversos.

A maioria dos nós não é mais faturada após a interrupção, excluindo determinados tipos de ECSs (aqueles com discos locais anexados, como ECSs de I/O ultra-alta e com uso intenso de disco). Para obter detalhes, consulte **Cobrança do ECS**.

Restrições

- A exclusão de um nó levará à migração de pods, o que pode afetar os serviços. Portanto, exclua os nós durante o horário de pico.
- Riscos inesperados podem ocorrer durante a exclusão de nó. Faça backup dos dados relacionados com antecedência.
- Enquanto o nó está sendo excluído, o back-end irá definir o nó para o estado não programável.
- Somente nós de trabalho podem ser interrompidos.

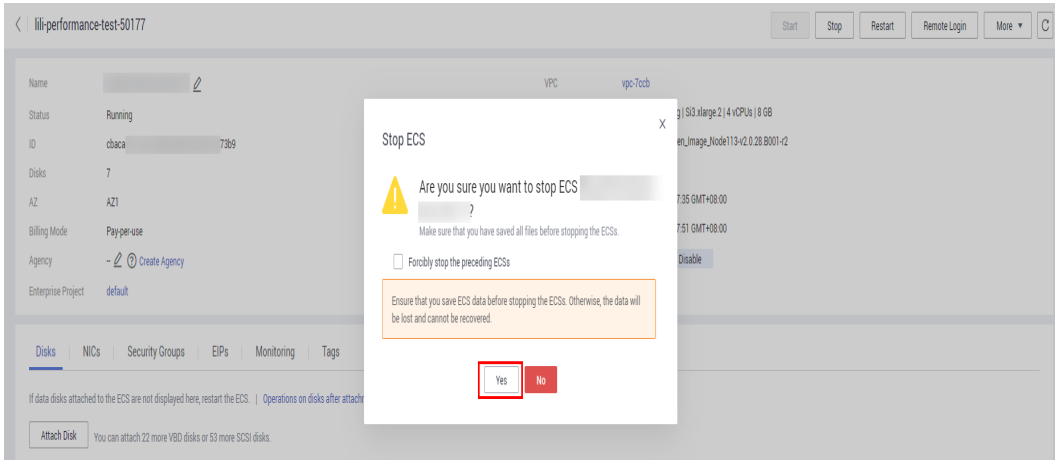
Procedimento

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster.

Passo 2 No painel de navegação, escolha **Nodes**. No painel direito, clique no nome do nó a ser interrompido.

Passo 3 No canto superior direito da página de detalhes do ECS, clique em **Stop**. Na caixa de diálogo exibida, clique em **Yes**.

Figura 3-7 Página de detalhes do ECS



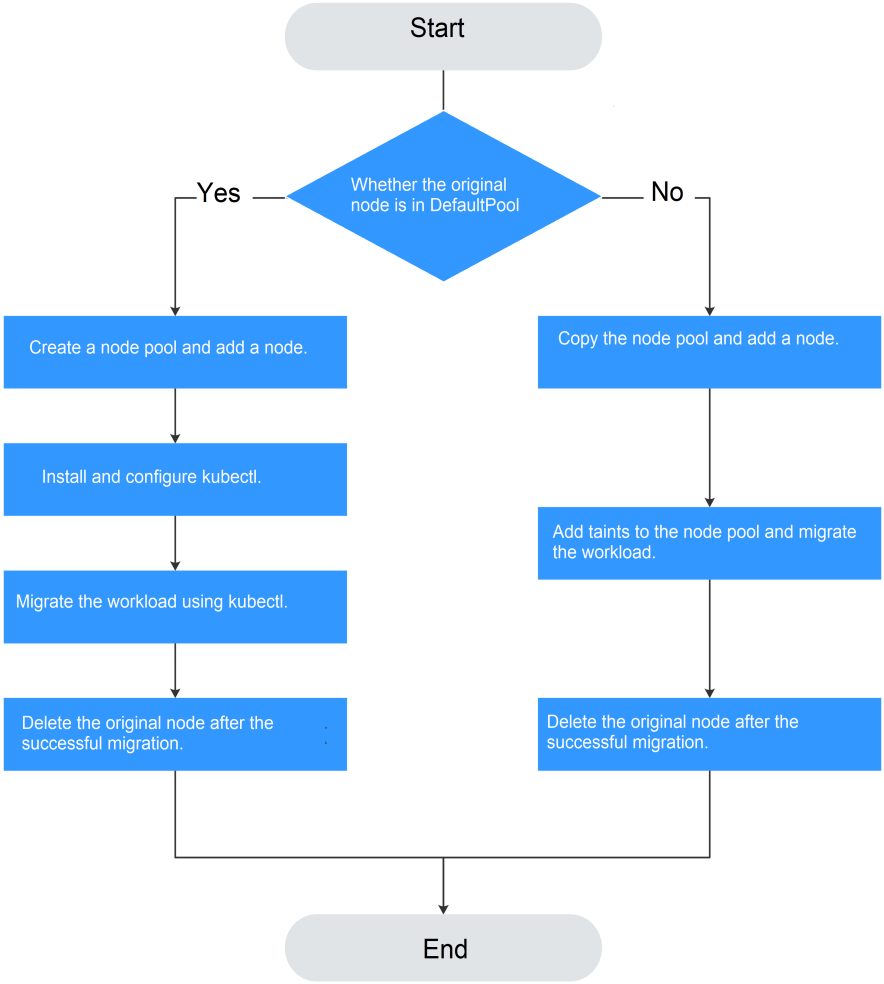
----Fim

3.7.10 Execução de atualização contínua para nós

Cenário

Em uma atualização contínua, um novo nó é criado, as cargas de trabalho existentes são migradas para o novo nó e, em seguida, o nó antigo é excluído. **Figura 3-8** mostra o processo de migração.

Figura 3-8 Migração da carga de trabalho



Restrições

- O nó original e o nó de destino para o qual a carga de trabalho será migrada devem estar no mesmo cluster.
- O cluster deve ser v1.13.10 ou posterior.
- O pool de nós padrão DefaultPool não oferece suporte a essa configuração.

Cenário 1: o nó original está em DefaultPool

Passo 1 Crie um pool de nós. Para mais detalhes, consulte **Criação de um pool de nós**.

Passo 2 Na página da lista de pool de nós, clique em **View Node** na coluna **Operation** do pool de nós de destino. O endereço IP do novo nó é exibido na lista de nós.

Passo 3 Instale e configure o kubectl. Para mais detalhes, consulte **Conexão a um cluster usando o kubectl**.

Passo 4 Migre a carga de trabalho.

1. Adicione uma mancha ao nó para onde a carga de trabalho precisa ser migrada.

   kubectl taint node *[node]* **key=value:***[effect]*

No comando anterior, *[node]* indica o endereço IP do nó onde a carga de trabalho a ser migrada está localizada. O valor de *[effect]* pode ser **NoSchedule**, **PreferNoSchedule** ou **NoExecute**. Neste exemplo, defina este parâmetro como **NoSchedule**.

- **NoSchedule**: os pods que não toleram essa mancha não são agendados no nó; os pods existentes não são despejados do nó.
- **PreferNoSchedule**: o Kubernetes tenta evitar o agendamento de pods que não toleram essa mancha no nó.
- **NoExecute**: um pod é despejado do nó se já estiver em execução no nó e não está agendado para o nó se ainda não estiver em execução no nó.

NOTA

Para redefinir uma mancha, execute o comando **kubectl taint node *[node]* key:*[effect]*- command** para remover a mancha.

2. Despeja com segurança a carga de trabalho no nó.

   kubectl drain *[node]*

   No comando anterior, *[node]* indica o endereço IP do nó onde a carga de trabalho a ser migrada está localizada.

3. No painel de navegação do console do CCE, escolha **Workloads** > **Deployments**. Na lista de cargas de trabalho, o status da carga de trabalho a ser migrada é alterado de **Running** para **Unready**. Se o status da carga de trabalho mudar para **Running** novamente, a migração será bem-sucedida.

NOTA

Durante a migração da carga de trabalho, se a afinidade do nó estiver configurada para a carga de trabalho, a carga de trabalho continuará exibindo uma mensagem indicando que a carga de trabalho não está pronta. Nesse caso, clique no nome da carga de trabalho para ir para a página de detalhes da carga de trabalho. Na página **Scheduling Policies**, exclua a configuração de afinidade do nó original e configure as políticas de afinidade e antiafinidade do novo nó. Para mais detalhes, consulte **Política de agendamento (afinidade/antiafinidade)**.

Depois que a carga de trabalho for migrada com êxito, você poderá exibir que a carga de trabalho foi migrada para o nó criado em **Passo 1** na página de guia **Pods** da página de detalhes da carga de trabalho.

Passo 5 Exclua o nó original.

Depois que a carga de trabalho for migrada com êxito e executada corretamente, exclua o nó original.

----Fim

Cenário 2: o nó original não está em DefaultPool

Passo 1 Copie o pool de nós e adicione nós a ele. Para mais detalhes, consulte **Cópia de um pool de nós**.

Passo 2 Clique em **View Node** na coluna **Operation** do pool de nós. O endereço IP do novo nó é exibido na lista de nós.

Passo 3 Migre a carga de trabalho.

1. Clique em **Edit** à direita do pool de nós original e configure **Taints**.
2. Insira a chave e o valor de uma mancha. As opções de **Effect** são **NoSchedule**, **PreferNoSchedule** e **NoExecute**. Selecione **NoExecute** e clique em **Add**.

- **NoSchedule**: os pods que não toleram essa mancha não são agendados no nó; os pods existentes não são despejados do nó.
- **PreferNoSchedule**: o Kubernetes tenta evitar o agendamento de pods que não toleram essa mancha no nó.
- **NoExecute**: um pod é despejado do nó se já estiver em execução no nó e não está agendado para o nó se ainda não estiver em execução no nó.

NOTA

Para redefinir a mancha, exclua a configurada.

3. Clique em **OK**.
4. No painel de navegação do console do CCE, escolha **Workloads** > **Deployments**. Na lista de cargas de trabalho, o status da carga de trabalho a ser migrada é alterado de **Running** para **Unready**. Se o status da carga de trabalho mudar para **Running** novamente, a migração será bem-sucedida.

NOTA

Durante a migração da carga de trabalho, se a afinidade do nó estiver configurada para a carga de trabalho, a carga de trabalho continuará exibindo uma mensagem indicando que a carga de trabalho não está pronta. Nesse caso, clique no nome da carga de trabalho para ir para a página de detalhes da carga de trabalho. Na página **Scheduling Policies**, exclua a configuração de afinidade do nó original e configure as políticas de afinidade e antiafinidade do novo nó. Para mais detalhes, consulte **Política de agendamento (afinidade/antiafinidade)**.

Depois que a carga de trabalho é migrada, você pode exibir que a carga de trabalho é migrada para o nó criado em **Passo 1** na página de guia **Pods** da página de detalhes da carga de trabalho.

Passo 4 Exclua o nó original.

Depois que a carga de trabalho for migrada com êxito e executada corretamente, exclua o nó original.

----Fim

3.8 O&M do nó

3.8.1 Política de reserva de recursos de nó

Alguns recursos de nó são usados para executar componentes e recursos obrigatórios do sistema Kubernetes para tornar o nó parte do cluster. Portanto, o número total de recursos de nó e o número de recursos de nó alocáveis para o cluster são diferentes. Quanto maiores as especificações do nó, mais os contêineres implementados no nó. Portanto, mais recursos de nó precisam ser reservados para executar componentes do Kubernetes.

Para garantir a estabilidade do nó, um certo número de recursos de nó do CCE será reservado para componentes do Kubernetes (como kubelet, kube-proxy e docker) com base nas especificações do nó.

O CCE calcula os recursos que podem ser alocados aos nós do usuário da seguinte forma:

Allocatable resources = Total amount - Reserved amount - Eviction threshold

O limite de remoção de memória é fixado em 100 MB.

NOTA

Total amount indica a memória disponível do ECS, excluindo a memória usada pelos componentes do sistema. Portanto, a quantidade total é um pouco menor que a memória do flavor do nó. Para obter detalhes, consulte **Por que a memória de um ECS obtida pela execução do comando free é inconsistente com a memória real?**

Quando a memória consumida por todos os pods em um nó aumenta, os seguintes comportamentos podem ocorrer:

1. Quando a memória disponível do nó é menor que o limite da remoção, o kubelet é acionado para remover o pod. Para obter detalhes sobre o limite de despejo no Kubernetes, consulte **Node-pressure Eviction**.
2. Se um nó acionar um evento de insuficiência de memória (OOM) antes de o kubelet recuperar a memória, o sistema termina o contêiner. No entanto, diferente da remoção do pod, o kubelet reinicia o contêiner com base no RestartPolicy do pod.

Regras v1 para reservar memória de nó

NOTA

Para clusters de versões anteriores a **v1.21.4-r0** e **v1.23.3-r0**, o modelo v1 é usado para reserva de memória de nó. Para clusters de **v1.21.4-r0**, **v1.23.3-r0** ou posterior, o modelo de reserva de memória de nó é otimizado para v2. Para mais detalhes, consulte **Regras para reservar memória de nó v2**.

Você pode usar a seguinte fórmula para calcular a quantidade de memória que você deve reservar para executar contêineres em um nó:

Quantidade total reservada = **memória reservada para os componentes do sistema** + **memória reservada para kubelet para gerenciar pods**

Tabela 3-23 Regras de reserva para componentes do sistema

Memória total (TM)	Memória reservada para componentes do sistema
TM ≤ 8 GB	0 MB
8 GB < TM ≤ 16 GB	[(TM – 8 GB) x 1024 x 10%] MB
16 GB < TM ≤ 128 GB	[8 GB x 1024 x 10% + (TM – 16 GB) x 1024 x 6%] MB
TM > 128 GB	(8 GB x 1024 x 10% + 112 GB x 1024 x 6% + (TM – 128 GB) x 1024 x 2%) MB

Tabela 3-24 Regras de reserva para kubelet

Memória total (TM)	Número de pods	Memória reservada para kubelet
TM ≤ 2 GB	Nenhum	TM x 25%
TM > 2 GB	0 < Máx. pods em um nó≤ 16	700 MB

Memória total (TM)	Número de pods	Memória reservada para kubelet
	16 < Máx. pods em um nó≤ 32	[700 + (Máx. pods em um nó – 16) x 18,75] MB
	32 < Máx. pods em um nó≤ 64	[1024 + (Máx. pods em um nó – 32) x 6,25] MB
	64 < Máx. pods em um nó≤ 128	[1230 + (Máx. pods em um nó – 64) x 7,80] MB
	Máx. pods em um nó > 128	[1740 + (Máx. pods em um nó – 128) x 11,20] MB

AVISO

Para um nó de pequena capacidade, ajuste o número máximo de instâncias com base nos requisitos do site. Alternativamente, ao criar um nó no console do CCE, você pode ajustar o número máximo de instâncias para o nó com base nas especificações do nó.

Regras para reservar memória de nó v2

Para clusters de **v1.21.4-r0**, **v1.23.3-r0** ou posterior, o modelo de reserva de memória de nó é otimizado para v2 e pode ser ajustado dinamicamente usando os parâmetros **kube-reserved-mem** e **system-reserved-mem**. Para mais detalhes, consulte **Gerenciamento de um pool de nós**.

A memória total de nó reservada do modelo v2 é igual à soma da reservada para o sistema operacional e da reservada para o CCE gerenciar pods.

A memória reservada inclui partes básicas e flutuantes. Para o sistema operacional, a memória flutuante depende das especificações do nó. Para CCE, a memória flutuante depende do número de pods em um nó.

Tabela 3-25 Regras para reservar memória de nó v2

Reservada para	Básica/flutuante	Reservas	Usada por
SO	Básica	400 MB (fixa)	Componentes de serviço do SO, como sshd e systemd-journald.
	Flutuante (dependendo da memória do nó)	25 MB/GB	Kernel

Reservada para	Básica/flutuante	Reservas	Usada por
CCE	Básica	500 MB (fixa)	Componentes do mecanismo de contêiner, como kubelet e kube-proxy, quando o nó é descarregado
	Flutuante (dependendo do número de pods no nó)	Docker: 20 MB/pod containerd: 5 MB/pod	Componentes do motor de contêiner quando o número de pods aumenta **NOTA** Quando o modelo v2 reserva memória para um nó por padrão, o número máximo padrão de pods é estimado com base na memória. Para mais detalhes, consulte **Tabela 3-29**.

Regras para reservar CPU de nó

Tabela 3-26 Regras de reserva de CPU de nó

Total de núcleos de CPU (total)	Núcleos de CPU reservados
Total ≤ 1 núcleo	Total x 6%
1 núcleo < total ≤ 2 núcleos	1 núcleo x 6% + (total – 1 núcleo) x 1%
2 núcleos < total ≤ 4 núcleos	1 núcleo x 6% + 1 núcleo x 1% + (total – 2 núcleos) x 0,5%
Total > 4 núcleos	1 núcleo x 6% + 1 núcleo x 1% + 2 núcleos x 0,5% + (total – 4 núcleos) x 0,25%

Regras para o CCE reservar discos de dados em nós

O CCE usa o Logical Volume Manager (LVM) para gerenciar discos. O LVM cria uma área de metadados em um disco para armazenar volumes lógicos e físicos, ocupando espaço de 4 MiB. Portanto, o espaço em disco disponível real de um nó é igual ao tamanho do disco menos 4 MiB.

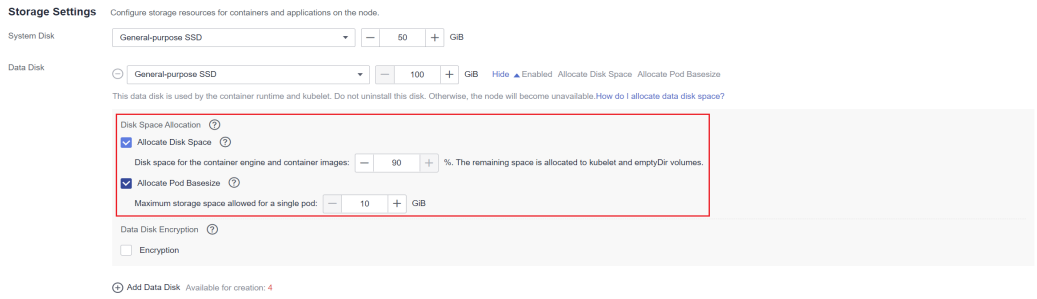
3.8.2 Alocação de espaço em disco de dados

Esta seção descreve como alocar espaço em disco de dados aos nós para que você possa configurar o espaço em disco de dados de acordo.

Alocar espaço em disco de dados

Ao criar um nó, configure discos de dados para o nó. Você também pode clicar em **Expand** e personalizar a alocação de espaço em disco de dados para o nó.

Figura 3-9 Alocação de espaço em disco de dados



- **Allocate Disk Space**:

  O CCE divide o espaço em disco de dados por duas partes por padrão. Uma parte é usada para armazenar os diretórios de trabalho do Docker/containerd, imagens de contêiner e metadados de imagem. A outra é reservada para os volumes de kubelet e emptyDir. O espaço disponível do mecanismo de contêiner afeta os pulls de imagem e a inicialização e execução do contêiner.

  – Mecanismo de contêiner e espaço de imagem de contêiner (90% por padrão): armazena os diretórios de trabalho do tempo de execução do contêiner, os dados da imagem de contêiner e os metadados da imagem.

  – Espaço kubelet e emptyDir (10% por padrão): armazena arquivos de configuração do pod, segredos e armazenamento montado, como volumes de emptyDir.

- **Allocate Pod Basesize**: indica o tamanho da base de um pod. Você pode definir um limite superior para o espaço em disco ocupado por cada pod de carga de trabalho (incluindo o espaço ocupado por imagens de contêiner). Essa configuração impede que os pods ocupem todo o espaço em disco disponível, o que pode causar exceções de serviço. Recomenda-se que o valor seja menor ou igual a 80% do espaço do mecanismo do contêiner. Este parâmetro está relacionado ao sistema operacional do nó e rootfs de armazenamento de contêiner e não é suportado em alguns cenários.

Alocação de espaço em disco

Para um nó usando um disco de dados não compartilhado (100 GB, por exemplo), a divisão do espaço em disco varia de acordo com o tipo Rootfs de armazenamento de contêiner **Device Mapper** ou **OverlayFS**. Para obter detalhes sobre os Rootfs de armazenamento de contêineres correspondentes a diferentes sistemas operacionais, consulte **Mapeamento entre SO e Rootfs de armazenamento de contêiner**.

- **Rootfs (Device Mapper)**

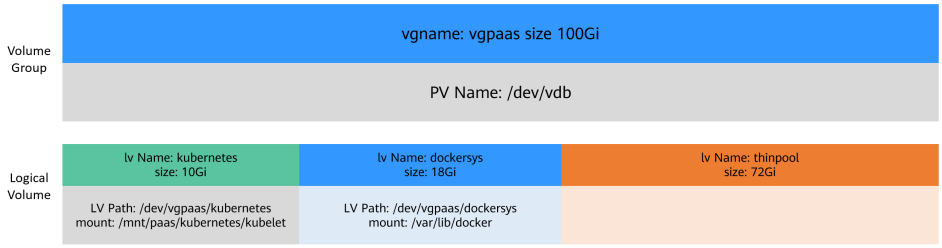
  Por padrão, o mecanismo de contêiner e o espaço de imagem, ocupando 90% do disco de dados, podem ser divididos nas duas partes a seguir:

  – O diretório **/var/lib/docker** é usado como o diretório de trabalho do Docker e ocupa 20% do mecanismo de contêiner e do espaço da imagem de contêiner por padrão. (Tamanho do espaço do diretório **/var/lib/docker** = **espaço em disco de dados x 90% x 20%**)

– O thin pool é usado para armazenar dados de imagem de contêiner, metadados de imagem e dados de contêiner e ocupa 80% do mecanismo de contêiner e do espaço de imagem de contêiner por padrão. (Espaço de thin pool = **espaço em disco de dados x 90% x 80%**)

  O thin pool é montado dinamicamente. Você pode visualizá-lo executando o comando **lsblk** em um nó, mas não o comando **df -h**.

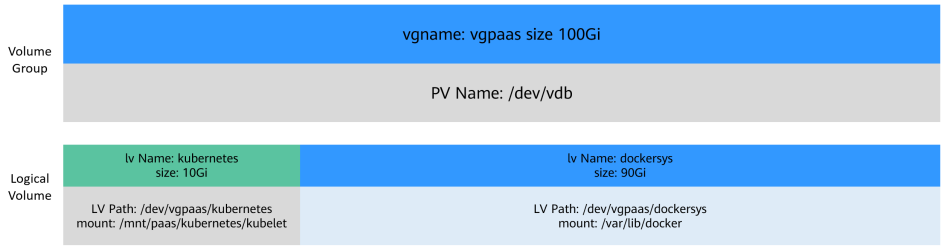
Figura 3-10 Alocação de espaço para mecanismos de contêineres do Device Mapper



- **Rootfs (OverlayFS)**

  Não há thin pool separado. Todo o mecanismo de contêiner e o espaço de imagem de contêiner (90% do disco de dados por padrão) estão no diretório **/var/lib/docker**.

Figura 3-11 Alocação de espaço para motores de contêineres do OverlayFS



Alocação de tamanho de base para pods

O espaço de contêiner de pod personalizado (tamanho de base) está relacionado ao sistema operacional do nó e ao Rootfs de armazenamento do contêiner. Para obter detalhes sobre o Rootfs de armazenamento de contêineres, consulte **Mapeamento entre SO e Rootfs de armazenamento de contêiner**.

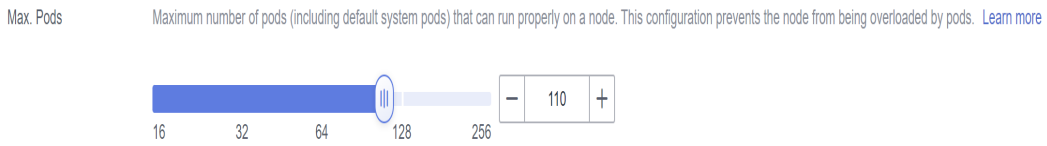
- O Device Mapper suporta o tamanho de bases de pod personalizado. O valor padrão é 10 GB.
- No modo OverlayFS, o espaço do contêiner do pod não é limitado por padrão.

  NOTA

  No caso de usar o Docker em nós do EulerOS 2.9, **basesize** não terá efeito se **CAP_SYS_RESOURCE** ou **privileged** estiver configurado para um contêiner.

Ao configurar **basesize**, considere o número máximo de pods em um nó. O espaço do mecanismo de contêiner deve ser maior que o espaço total em disco usado pelos contêineres. Fórmula: **o espaço do mecanismo de contêiner e o espaço da imagem do contêiner (90% por padrão)** > **número de contêineres** x **tamanho de base**. Caso contrário, o espaço do mecanismo de contêiner alocado para o nó pode ser insuficiente e o contêiner não pode ser iniciado.

Figura 3-12 Número máximo de pods



Para nós que suportam **basesize**, quando o Device Mapper é usado, embora você possa limitar o tamanho do diretório **/home** de um único contêiner (a 10 GB por padrão), todos os contêineres no nó ainda compartilham o thin pool do nó para armazenamento. Eles não estão completamente isolados. Quando a soma do espaço do thin pool usado por determinados contêineres atinge o limite superior, outros contêineres não podem ser executados corretamente.

Além disso, depois que um arquivo é excluído no diretório **/home** do contêiner, o espaço do thin pool ocupado pelo arquivo não é liberado imediatamente. Portanto, mesmo que o **basesize** seja definido como 10 GB, o espaço de thin pool ocupado pelos arquivos continua aumentando até 10 GB quando os arquivos são criados no contêiner. O espaço liberado após a exclusão do arquivo será reutilizado, mas depois de um tempo. Se **o número de contêineres no nó multiplicado pelo tamanho de base** for maior do que o tamanho do espaço do thin pool do nó, existe a possibilidade de que o espaço do thin pool tenha sido usado.

Mapeamento entre SO e Rootfs de armazenamento de contêiner

Tabela 3-27 Sistemas operacionais de nó e mecanismos de contêiner em clusters do CCE

SO	Rootfs de armazenamento de contêiner	Tamanho da base personalizado
CentOS 7.x	Clusters de v1.19.16 e anteriores usam Device Mapper. Clusters de v1.19.16 e posterior usam OverlayFS.	Suportado quando Rootfs é definido como Device Mapper e o mecanismo de contêiner é Docker. O valor padrão é 10 GB. Não é suportado quando Rootfs está definido como OverlayFS.
EulerOS 2.3	Device Mapper	Suportado somente quando o mecanismo de contêiner é Docker. O valor padrão é 10 GB.
EulerOS 2.5	Device Mapper	Suportado somente quando o mecanismo de contêiner é Docker. O valor padrão é 10 GB.
EulerOS 2.8	Clusters de v1.19.16-r2 e anteriores usam Device Mapper. Clusters de v1.19.16-r2 e posterior usam OverlayFS.	Suportado quando Rootfs é definido como Device Mapper e o mecanismo de contêiner é Docker. O valor padrão é 10 GB. Não é suportado quando Rootfs está definido como OverlayFS.

SO	Rootfs de armazenamento de contêiner	Tamanho da base personalizado
EulerOS 2.9	OverlayFS	Suportado apenas por clusters de v1.19.16, v1.21.3, v1.23.3 e posteriores. O tamanho da base do contêiner não é limitado por padrão. Não há suporte se as versões de cluster forem anteriores a v1.19.16, v1.21.3 e v1.23.3.
EulerOS 2.10	OverlayFS	Suportado somente quando o mecanismo de contêiner é Docker. O tamanho da base do contêiner não é limitado por padrão.
Ubuntu 18.04	OverlayFS	Não compatível.
Huawei Cloud EulerOS 1.1	OverlayFS	Não compatível.
Huawei Cloud EulerOS 2.0	OverlayFS	Suportado somente quando o mecanismo de contêiner é Docker. O tamanho da base do contêiner não é limitado por padrão.

Tabela 3-28 Sistemas operacionais de nó e mecanismos de contêiner em clusters do CCE Turbo

SO	Rootfs de armazenamento de contêiner	Tamanho da base personalizado
CentOS 7.x	OverlayFS	Não compatível.
Ubuntu 18.04	OverlayFS	Não compatível.
EulerOS 2.9	As VMs do ECS usam OverlayFS. As PMs do ECS usam Device Mapper.	Suportado somente quando Rootfs é definido como OverlayFS e o mecanismo de contêiner é Docker. O tamanho da base do contêiner não é limitado por padrão. Suportado quando Rootfs é definido como Device Mapper e o mecanismo de contêiner é Docker. O valor padrão é 10 GB.
Huawei Cloud EulerOS 1.1	OverlayFS	Não compatível.

SO	Rootfs de armazenamento de contêiner	Tamanho da base personalizado
Huawei Cloud EulerOS 2.0	OverlayFS	Suportado somente quando o mecanismo de contêiner é Docker. O tamanho da base do contêiner não é limitado por padrão.

Políticas de coleta de lixo para imagens de contêiner

Quando o espaço do mecanismo de contêiner é insuficiente, a coleta de lixo de imagem é acionada.

A política de coleta de imagens de lixo leva dois fatores em consideração: **HighThresholdPercent** e **LowThresholdPercent**. O uso do disco acima do limite alto (padrão: 85%) acionará a coleta de lixo. A coleta de lixo excluirá as imagens menos usadas recentemente até que o limite baixo (padrão: 80%) seja atingido.

Configuração recomendada para o espaço do mecanismo de contêiner

- O espaço do mecanismo de contêiner deve ser maior que o espaço total em disco usado pelos contêineres. Fórmula: **espaço do mecanismo do contêiner** > **número de contêineres** x **tamanho de base**
- É aconselhável criar e excluir arquivos de serviços em contêiner em volumes de armazenamento local (como volumes emptyDir e hostPath) ou diretórios de armazenamento em nuvem montados nos contêineres. Desta forma, o espaço da piscina fina não é ocupado. Os volumes emptyDir ocupam o espaço de kubelet. Portanto, planeje adequadamente o tamanho do espaço do kubelet.
- Você pode implantar serviços em nós que usam o OverlayFS (para obter detalhes, consulte **Mapeamento entre SO e Rootfs de armazenamento de contêiner**) para que o espaço em disco ocupado por arquivos criados ou excluídos em contêineres possa ser liberado imediatamente.

Problemas comuns

Como expandir a capacidade de armazenamento de um contêiner?

Expansão da capacidade de disco de um nó em um cluster do CCE

3.8.3 Número máximo de pods que podem ser criados em um nó

Cálculo do número máximo de pods em um nó

O número máximo de pods que podem ser criados em um nó é calculado com base no tipo de cluster:

- Para um cluster que usa o modelo de rede de túnel de contêiner, o valor depende apenas **do número máximo de pods em um nó**.
- Para clusters que usam o modelo de rede VPC, o valor depende do **número máximo de pods em um nó** e **do número mínimo de endereços IP de contêiner que podem ser**

alocados a um nó. Recomenda-se que o número máximo de pods em um nó seja menor ou igual ao número de endereços IP do contêiner que podem ser alocados ao nó. Caso contrário, os pods podem não ser agendados.

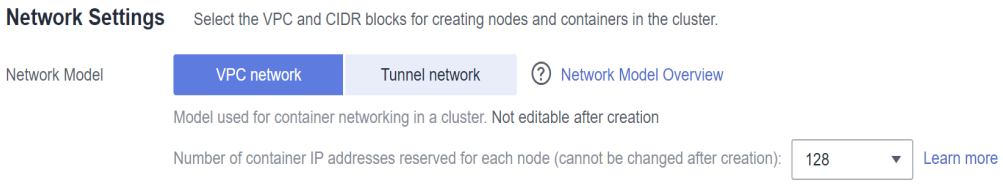
- Para clusters do CCE Turbo usando o modelo Cloud Native Network 2.0, o valor depende do **número máximo de pods em um nó** e do **número mínimo de ENIs em um nó de cluster do CCE Turbo**. Recomenda-se que o número máximo de pods em um nó seja menor ou igual ao número de ENIs no nó. Caso contrário, os pods podem não ser agendados.

Número de endereços IP de contêiner que podem ser alocados em um nó

Se você selecionar **VPC network** para **Network Model** ao criar um cluster do CCE, também precisará definir o número de endereços IP de contêiner que podem ser alocados para cada nó (alpha.cce/fixPoolMask). Se o pod usar a rede host (**hostNetwork: true**), o pod não ocupará o endereço IP da rede de contêineres alocável. Para mais detalhes, consulte **Rede de contêiner vs. rede host**.

Este parâmetro afeta o número máximo de pods que podem ser criados em um nó. Cada pod ocupa um endereço IP (quando a **rede de contêineres** é usada). Se o número de endereços IP disponíveis for insuficiente, os pods não poderão ser criados. Se o pod usar a rede host (**hostNetwork: true**), o pod não ocupará o endereço IP da rede de contêineres alocável.

Figura 3-13 Especificação do número de endereços IP de contêiner alocáveis em um nó no modelo de rede da VPC



Por padrão, um nó ocupa três endereços IP de contêiner (endereço de rede, endereço de gateway e endereço de transmissão). Portanto, o número de endereços IP de contêiner que podem ser alocados a um nó é igual ao número de endereços IP de contêiner selecionados menos 3. Por exemplo, na figura anterior, o número de endereços IP de contêiner que podem ser alocados a um nó é 125 (128 – 3).

Número máximo de pods em um nó

Ao criar um nó, você pode configurar o número máximo de pods (maxPods) que podem ser criados no nó. Este parâmetro é um item de configuração do kubelet e determina o número máximo de pods que podem ser criados pelo kubelet.

AVISO

Para nós no pool de nós padrão (**DefaultPool**), o número máximo de pods não pode ser alterado após a criação dos nós.

Depois que um nó em um pool de nós personalizado for criado, você poderá modificar o parâmetro **max-pods** na configuração do pool de nós para alterar o número máximo de pods no nó. Para obter detalhes, consulte **Configuração de um pool de nós**.

Figura 3-14 Especificar o número máximo de pods em um nó

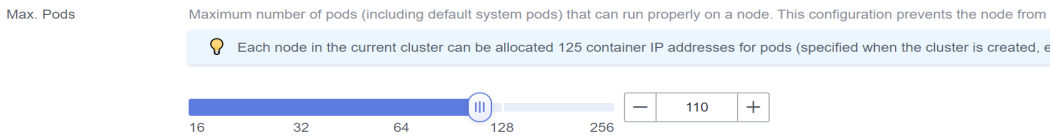


Tabela 3-29 lista o número máximo padrão de pods em um nó com base nas especificações do nó.

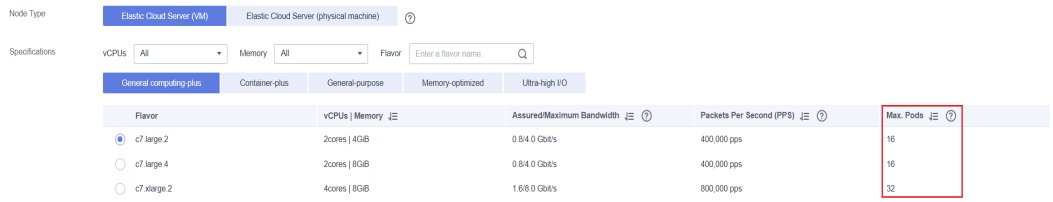
Tabela 3-29 Número máximo padrão de pods em um nó

Memória	Máx. Pods
4 GB	20
8 GB	40
16 GB	60
32 GB	80
64 GB ou mais	110

Número de ENIs de nó (clusters do CCE Turbo)

Em um cluster do CCE Turbo, os nós do ECS usam sub-ENIs e os nós do BMS usam ENIs. O número máximo de pods que podem ser criados em um nó depende do número de ENIs que podem ser usados pelo nó.

Figura 3-15 ENIs de nó



Rede de contêiner vs. rede host

Ao criar um pod, você pode selecionar a rede de contêiner ou a rede host para o pod.

- Rede de contêineres (padrão): **cada pod recebe um endereço IP pelos complementos de rede do cluster, que ocupam os endereços IP da rede do contêiner**.
- Rede host: o pod usa a rede host (**hostNetwork: true** precisa ser configurado para o pod) e ocupa a porta do host. O endereço IP do pod é o endereço IP do host. O pod não ocupa os endereços IP da rede de contêineres. Para usar a rede host, você deve confirmar se as portas do contêiner entram em conflito com as portas do host. Não use a rede host a menos que você saiba exatamente qual porta de host é usada por qual contêiner.

3.8.4 Migração de nós do Docker para containerd

Contexto

O Kubernetes removeu o dockershim da v1.24 e não suporta o Docker por padrão. O CCE continuará a suportar o Docker na v1.25, mas apenas até a v1.27. As etapas a seguir mostram como migrar nós do Docker para containerd.

Pré-requisitos

- Pelo menos um cluster que suporta nós em contêiner foi criado. Para mais detalhes, consulte **Mapeamento entre sistemas operacionais de nó e mecanismos de contêiner**.
- Há um nó do Docker ou um pool de nós do Docker no cluster.

Precauções

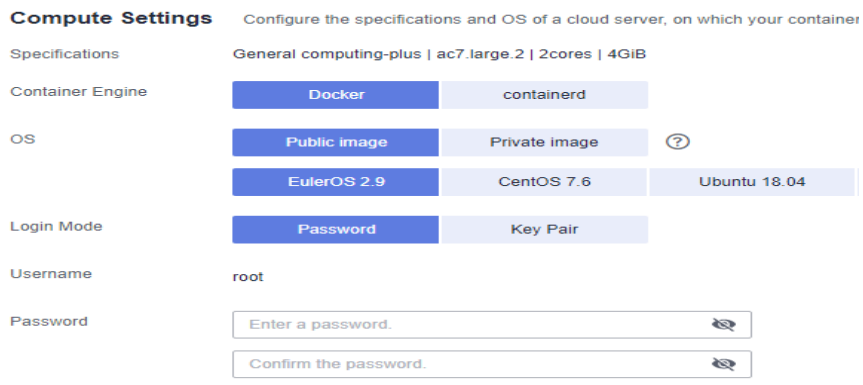
- Teoricamente, a migração durante a execução do contêiner interromperá os serviços por um curto período de tempo. Portanto, é altamente recomendável que os serviços a serem migrados tenham sido implementados como várias instâncias. Além disso, é aconselhável testar o impacto da migração no ambiente de teste para minimizar os riscos potenciais.
- containerd não pode construir imagens. Não use o comando **docker build** para criar imagens em nós em contêineres. Para outras diferenças entre Docker e containerd, veja **Mecanismo de contêiner**.

Migrar um nó

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster.

Passo 2 No painel de navegação, escolha **Nodes**. Na lista de nós, selecione um ou mais nós a serem redefinidos e escolha **More** > **Reset Node**.

Passo 3 Defina **Container Engine** como **containerd**. É possível ajustar outros parâmetros conforme necessário ou retê-los conforme definido durante a criação.



Passo 4 Se o status do nó for **Installing**, o nó está sendo redefinido.

Quando o status do nó é **Running**, você pode ver que a versão do nó é alterada para containerd. Você pode efetuar logon no nó e executar comandos containerd, como **crictl**, para exibir informações sobre os contêineres em execução no nó.

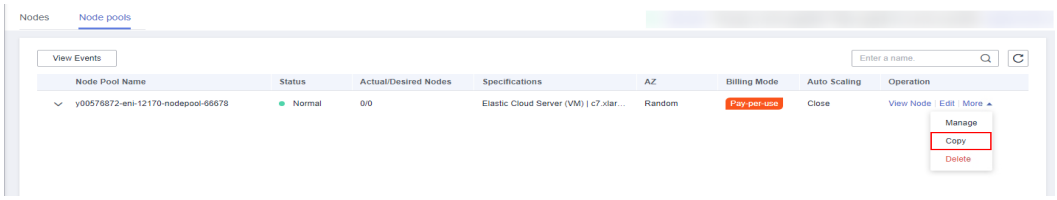
----Fim

Migrar um pool de nós

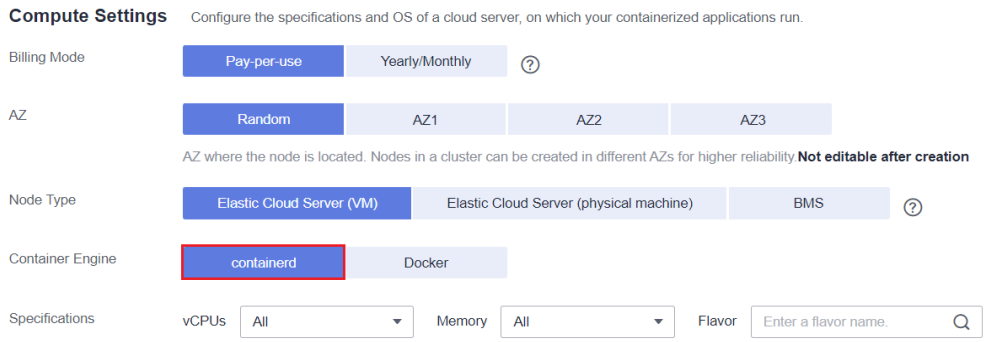
Você pode **copiar um pool de nós**, definir o mecanismo de contêiner do novo pool de nós como containerd e manter outras configurações iguais às do pool de nós original do Docker.

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster.

Passo 2 No painel de navegação, escolha **Nodes**. Na página de guia **Node Pools**, localize o pool de nós do Docker a ser copiado e escolha **More** > **Copy** na coluna **Operation**.



Passo 3 Na área **Compute Settings**, defina o **Container Engine** como **containerd** e modifique outros parâmetros conforme necessário.



Passo 4 Dimensione o número de pools de nós de containerd criados para o número de pools de nós originais do Docker e exclua os nós dos pools de nós do Docker, um por um.

A migração rolante é preferida. Ou seja, adicione alguns nós de containerd e, em seguida, exclua alguns nós do Docker até que o número de nós no novo pool de nós de containerd seja o mesmo do pool de nós original do Docker.

NOTA

Se você tiver definido a afinidade de nó para as cargas de trabalho implementadas nos nós originais do Docker ou no pool de nós, defina políticas de afinidade para as cargas de trabalho a serem executadas nos novos nós de containerd ou no pool de nós.

Passo 5 Após a migração, exclua o pool de nós original do Docker.

----Fim

3.8.5 Otimização de parâmetros do sistema de nó

3.8.5.1 Lista de parâmetros do sistema de nós que podem ser otimizados

O CCE fornece parâmetros de sistema de nó padrão, o que pode causar gargalos de desempenho em alguns cenários. Portanto, você pode personalizar e otimizar alguns parâmetros do sistema de nó. **Lista de parâmetros do sistema de nós que podem ser otimizados** descreve os parâmetros do sistema de nó.

AVISO

- A modificação tem certos riscos. Familiarize-se com os comandos do Linux e o sistema operacional Linux.
- Os parâmetros listados em **Tabela 3-30** foram testados e verificados. **Não modifique outros parâmetros**. Caso contrário, podem ocorrer falhas de nó.
- Os comandos para modificar parâmetros do sistema de nó são válidos somente quando imagens públicas são usadas. Os comandos fornecidos neste documento são apenas para referência quando imagens privadas são usadas.
- Depois que o nó for reiniciado, execute o comando **sysctl -p** para atualizar o valor do parâmetro.

Tabela 3-30 Lista de parâmetros do sistema que podem ser otimizados

Parâmetro	Localização do parâmetro	Descrição	Referência
kernel.pid_max	/etc/sysctl.conf	Número máximo de IDs de processo em um nó Obtenção do parâmetro: `sysctl kernel.pid_max`	**Alteração de limites de ID de processo (kernel.pid_max)**
RuntimeMaxUse	/etc/systemd/journald.conf	Limite superior da memória ocupada pelo cache de log do nó. Se este parâmetro não for definido, uma grande quantidade de memória será ocupada depois que o sistema for executado por um longo tempo. Obtenção do parâmetro: `cat /etc/systemd/journald.conf | grep RuntimeMaxUse`	**Alteração de RuntimeMaxUse da memória usada pelo cache de log em um nó**

Parâmetro	Localização do parâmetro	Descrição	Referência
Openfiles	/etc/security/limits.conf	Número máximo de manipuladores de arquivo para um único processo em um nó, que pode ser ajustado conforme necessário. Obtenção do parâmetro: `ulimit -n`	**Alterar o número máximo de identificadores de arquivo para um único processo em um nó**

(inside the Openfiles container)
LimitNOFILE
LimitNPROC
	● CentOS/EulerOS:
	– Nós do Docker: /usr/lib/systemd/system/docker.service
	– Nós do Docker: /usr/lib/systemd/system/containerd.service
	● Ubuntu:
	– Nós do Docker: /lib/systemd/system/docker.service
	– Nós do Docker: /lib/systemd/system/containerd.service
		Número máximo de alças de arquivo para um único processo em um contêiner, que pode ser ajustado conforme necessário.
		Obtenção do parâmetro:
		Nós do Docker:
		`cat /proc/`pidof dockerd`/limits | grep files`
		Nós de contêiner:
		`cat /proc/`pidof containerd`/limits | grep files`
			Alterar o número máximo de alças de arquivo para um único processo de contêiner

Parâmetro	Localização do parâmetro	Descrição	Referência
file-max	/etc/sysctl.conf	Número máximo de alças de arquivo no sistema, que podem ser ajustadas conforme necessário. Obtenção do parâmetro: `sysctl fs.file-max`	**Alterar o número máximo de identificadores de arquivos em nível de sistema em um nó**

nf_conntrack_buckets
nf_conntrack_max
	/etc/sysctl.conf	Capacidade da tabela de rastreamento de conexão, que pode ser ajustada conforme necessário.
		Uso do bucket = [nf_conntrack_count]/ [nf_conntrack_buckets]
		Ajuste o valor dos buckets para garantir que o uso do bucket seja menor que 0,7.
		Obtenção do parâmetro:
		`sysctl net.netfilter.nf_conntrack_count`
		`sysctl net.netfilter.nf_conntrack_buckets`
		`sysctl net.netfilter.nf_conntrack_max`
			Modificação de parâmetros do nó de kernel

net.netfilter.nf_conntrack_tcp_timeout_close	/etc/sysctl.conf	Tempo de expiração da entrada da conexão no estado de fechamento na tabela de rastreamento de conexão. Diminuir o tempo de expiração pode acelerar a reciclagem. Obtenção do parâmetro: `sysctl net.netfilter.nf_conntrack_tcp_timeout_close`	

Parâmetro	Localização do parâmetro	Descrição	Referência
net.netfilter.nf_conntrack_tcp_be_liberal	/etc/sysctl.conf	O valor do parâmetro é **0** ou **1**. ● **0**: a função está desativada. Todos os pacotes RST que não estão na janela TCP são marcados como inválidos. ● **1**: a função está ativada. Somente os pacotes RST que não estão na janela TCP são marcados como inválidos. Nos contêineres, habilitar esse parâmetro pode impedir que a largura de banda das conexões TCP que foram convertidas usando NAT seja limitada. Obtenção do parâmetro: `sysctl net.netfilter.nf_conntrack_tcp_be_liberal`	
tcp_keepalive_time	/etc/sysctl.conf	Intervalo no qual uma mensagem de Keepalive de TCP é enviada. Se esse parâmetro for definido com um valor grande, as conexões TCP podem ser suspensas na fase **Close_wait** por um longo tempo, esgotando os recursos do sistema. Obtenção do parâmetro: `sysctl net.ipv4.tcp_keepalive_time`	
tcp_max_syn_backlog	/etc/sysctl.conf	Número máximo de semiconexões TCP, ou seja, o número máximo de conexões na fila **SYN_RECV**. Obtenção do parâmetro: `sysctl net.ipv4.tcp_max_syn_backlog`	
tcp_max_tw_buckets	/etc/sysctl.conf	Especifica o número máximo de soquetes no estado **time-wait** que pode existir a qualquer momento. Se o valor do parâmetro for muito grande, os recursos do nó podem estar esgotados. Obtenção do parâmetro: `sysctl net.ipv4.tcp_max_tw_buckets`	

Parâmetro	Localização do parâmetro	Descrição	Referência
net.core.somaxconn	/etc/sysctl.conf	Número máximo de conexões TCP. Este parâmetro controla o número de conexões TCP em uma fila. Se esse parâmetro for definido como um valor pequeno, o número de conexões TCP é propenso a insuficiência. Se este parâmetro estiver definido como um valor grande, os recursos do sistema poderão ser desperdiçados porque cada cliente à espera de ligação na fila de ligação ocupa determinados recursos de memória. Obtenção do parâmetro: `sysctl net.core.somaxconn`	
max_user_instances	/etc/sysctl.conf	Número máximo de instâncias de inotify permitidas para cada usuário. Se o valor do parâmetro for muito pequeno, o número de instâncias de inotify pode ser insuficiente nos contêineres. Obtenção do parâmetro: `sysctl fs.inotify.max_user_instances`	
max_user_watches	/etc/sysctl.conf	Número máximo de diretórios de todas as instâncias de monitoramento. Se o valor do parâmetro for muito pequeno, o número de diretórios pode ser insuficiente nos contêineres. Obtenção do parâmetro: `sysctl fs.inotify.max_user_watches`	
netdev_max_backlog	/etc/sysctl.conf	Tamanho da fila de recebimento de pacotes da pilha de protocolos de rede. Se o valor do parâmetro for muito pequeno, o tamanho da fila pode ser insuficiente. Obtenção do parâmetro: `sysctl net.core.netdev_max_backlog`	

Parâmetro	Localização do parâmetro	Descrição	Referência

net.core.wmem_max
net.core.rmem_max
	/etc/sysctl.conf	Tamanho da memória (bytes) do buffer de envio e recebimento. Se este parâmetro estiver definido como um valor pequeno, o tamanho da memória poderá ser insuficiente em cenários de ficheiros grandes.
		Obtenção do parâmetro:
		`sysctl net.core.wmem_max`
		`sysctl net.core.rmem_max`	

net.ipv4.neigh.default.gc_thresh1
net.ipv4.neigh.default.gc_thresh2
net.ipv4.neigh.default.gc_thresh3
	/etc/sysctl.conf	Otimização da coleta de lixo de entradas ARP.
		Obtenção do parâmetro:
		`sysctl net.ipv4.neigh.default.gc_thresh1`
		`sysctl net.ipv4.neigh.default.gc_thresh2`
		`sysctl net.ipv4.neigh.default.gc_thresh3`	

vm.max_map_count	/etc/sysctl.conf	Se este parâmetro é ajustado a um valor pequeno, uma mensagem é indicada indicando que o espaço é insuficiente durante a instalação de ELK. Obtenção do parâmetro: `sysctl vm.max_map_count`	

3.8.5.2 Alteração de RuntimeMaxUse da memória usada pelo cache de log em um nó

Journald é um sistema de log em Linux. Ele grava informações de log em arquivos binários e usa o diretório **/run/log/journal** como o diretório de cache de log por padrão. O arquivo de configuração de Journald é armazenado no diretório **/etc/systemd/journald.conf** no nó. O parâmetro **RuntimeMaxUse** indica o uso máximo de memória do cache de log. Se **RuntimeMaxUse** não estiver definido, uma grande quantidade de memória será ocupada depois que o sistema for executado por um longo tempo.

AVISO

Os comandos para modificar parâmetros do sistema de nó são válidos somente quando imagens públicas são usadas. Os comandos fornecidos neste documento são apenas para referência quando imagens privadas são usadas.

Alterar RuntimeMaxUse

Passo 1 Faça logon no nó e visualize o arquivo **/etc/systemd/journald.conf**.

```
cat /etc/systemd/journald.conf
```

Passo 2 Modifique **RuntimeMaxUse**. O valor recomendado é **100M**.

- Se **RuntimeMaxUse** tiver sido definido no arquivo **journald.conf**, execute o seguinte comando para alterar o valor:

  ```
  sed -i "s/RuntimeMaxUse=[0-9]*M/RuntimeMaxUse=100M/g" /etc/systemd/
  journald.conf && systemctl restart systemd-journald
  ```

- Se **RuntimeMaxUse** não estiver definido no arquivo **journald.conf**, execute o seguinte comando para adicioná-lo:

  ```
  echo RuntimeMaxUse=100M >> /etc/systemd/journald.conf && systemctl restart
  systemd-journald
  ```

Passo 3 Se o valor retornado for o mesmo que o valor modificado, a modificação está correta.

```
cat /etc/systemd/journald.conf | grep RuntimeMaxUse
```

----Fim

Configurar automaticamente RuntimeMaxUse ao criar um nó ou pool de nós

Você pode definir o script a ser executado após a instalação de um nó ou pool de nós. Ao criar um nó ou pool de nós, você pode usar o script para configurar o tamanho do **RuntimeMaxUse**.

Passo 1 Confirme o sistema operacional do nó ou pool de nós a ser criado, por exemplo, CentOS 7.6.

Passo 2 Teste manualmente os comandos de script em nós **no mesmo cluster e executando o mesmo sistema operacional**. Para obter detalhes sobre como executar manualmente o script, consulte **Alterar RuntimeMaxUse**.

Passo 3 Ao criar um nó ou um pool de nós, escolha **Advanced Settings** > **Post-installation Command** para adicionar comandos. **(Os comandos a seguir devem ser configurados após a verificação ser bem-sucedida.)**

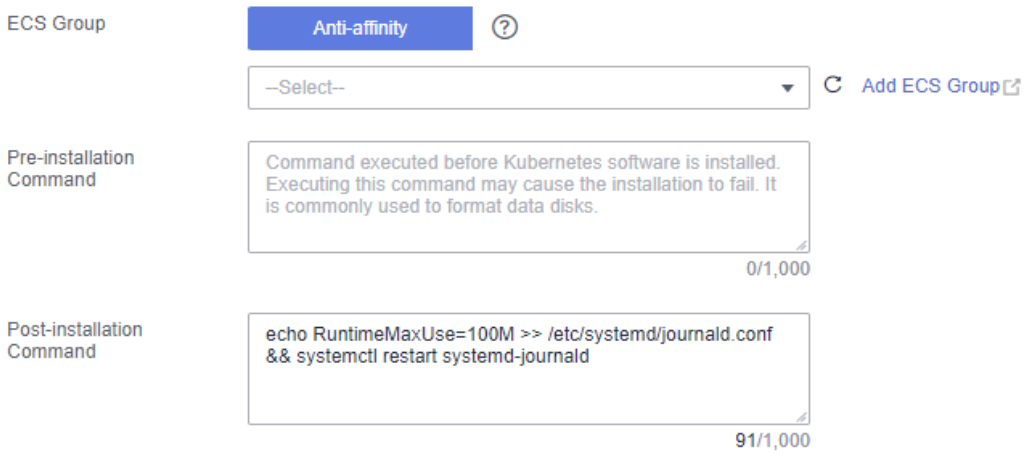
- Faça logon no nó e visualize o arquivo **/etc/systemd/journald.conf**. Se **RuntimeMaxUse** tiver sido definido, execute o seguinte comando para alterar o valor:

  ```
  sed -i "s/RuntimeMaxUse=[0-9]*M/RuntimeMaxUse=100M/g" /etc/systemd/
  journald.conf && systemctl restart systemd-journald
  ```

- Faça logon no nó e visualize o arquivo **/etc/systemd/journald.conf**. Se **RuntimeMaxUse** não estiver definido, execute o seguinte comando para adicioná-lo:

  ```
  echo RuntimeMaxUse=100M >> /etc/systemd/journald.conf && systemctl restart
  systemd-journald
  ```

O comando na figura a seguir é usado apenas como exemplo. Altere-o conforme necessário.



Passo 4 Depois que o nó for criado, efetue logon no nó para verificar se os parâmetros foram modificados com êxito.

```
cat /etc/systemd/journald.conf | grep RuntimeMaxUse
```

----Fim

3.8.5.3 Alteração do número máximo de identificadores de arquivo

O número máximo de identificadores de arquivo é o número máximo de arquivos que podem ser abertos. No Linux, há duas restrições de identificadores de arquivos. Uma é a restrição no nível do sistema, onde o número máximo de arquivos que podem ser abertos por todos os processos do usuário ao mesmo tempo. A outra é a restrição em nível de usuário, onde o número máximo de arquivos que podem ser abertos por um único processo de usuário. Os contêineres têm a terceira restrição de identificadores de arquivo, ou seja, o número máximo de identificadores de arquivo de um único processo no contêiner.

AVISO

Os comandos para modificar parâmetros do sistema de nó são válidos somente quando imagens públicas são usadas. Os comandos fornecidos neste documento são apenas para referência quando imagens privadas são usadas.

Alterar o número máximo de identificadores de arquivos em nível de sistema em um nó

Passo 1 Faça logon no nó e verifique o arquivo **/etc/sysctl.conf**.

```
cat /etc/sysctl.conf
```

Passo 2 Modifique o parâmetro **fs.file-max**. **fs.file-max=1048576** indica o nome do parâmetro do kernel e o valor recomendado.

- Se o valor de fs.file-max tiver sido definido no arquivo **sysctl.conf**, execute o seguinte comando para alterar o valor:

  ```
  sed -i "s/fs.file-max=[0-9]*$/fs.file-max=1048576/g" /etc/sysctl.conf && sysctl -p
  ```

- Se **fs.file-max** não estiver definido no arquivo **sysctl.conf**, execute o seguinte comando para adicioná-lo:


```
echo fs.file-max=1048576 >> /etc/sysctl.conf && sysctl -p
```

Passo 3 Execute os comandos a seguir para verificar se a alteração foi bem-sucedida (se o valor retornado é o mesmo que você configurou).

```
# sysctl fs.file-max
fs.file-max = 1048576
```

----Fim

Alterar o número máximo de identificadores de arquivo para um único processo em um nó

Passo 1 Faça logon no nó e visualize o arquivo **/etc/security/limits.conf**.

```
cat /etc/security/limits.conf
```

O número máximo de identificadores de arquivo para um único processo de um nó é especificado pelos seguintes parâmetros:

```
...
root soft nofile 65535
root hard nofile 65535
* soft nofile 65535
* hard nofile 65535
```

Passo 2 Execute o comando **sed** para alterar o número máximo de identificadores de arquivo. No comando, **65535** é o número máximo recomendado de identificadores de arquivo. O arquivo **/etc/security/limits.conf** no nó EulerOS 2.3 não contém a configuração padrão relacionada a nofile. Portanto, você não pode executar o comando **sed** para modificar a configuração.

```
sed -i "s/nofile.[0-9]*$/nofile 65535/g" /etc/security/limits.conf
```

Passo 3 **Faça logon no nó novamente** e execute o seguinte comando para verificar se a modificação foi bem-sucedida. Se o valor retornado for o mesmo que o valor modificado, a modificação será bem-sucedida.

```
# ulimit -n
65535
```

----Fim

Alterar o número máximo de alças de arquivo para um único processo de contêiner

Passo 1 Faça logon no nó e visualize o arquivo **/usr/lib/systemd/system/docker.service**.

- CentOS/EulerOS:
  - Nós do Docker:
    ```
    cat /usr/lib/systemd/system/docker.service
    ```
  - Nós de contêiner:
    ```
    cat /usr/lib/systemd/system/containerd.service
    ```
- Ubuntu:
  - Nós do Docker:
    ```
    cat /lib/systemd/system/docker.service
    ```
  - Nós de containerd:
    ```
    cat /lib/systemd/system/containerd.service
    ```


> **NOTA**
>
> Se **LimitNOFILE** ou **LimitNPROC** estiver definido como **infinity**, o número máximo de identificadores de arquivo suportados por um único processo de um contêiner é **1,048,576**.

O número máximo de identificadores de arquivo para um único processo de um contêiner é especificado pelos seguintes parâmetros:

```
...
LimitNOFILE=1048576
LimitNPROC=1048576
...
```

Passo 2 Execute os seguintes comandos para modificar os dois parâmetros. No comando, **1048576** é o valor recomendado para o número máximo de identificadores de arquivo.

AVISO

Alterar o número máximo de identificadores de arquivos de um contêiner reiniciará o processo docker/containerd.

- CentOS/EulerOS:
  - Nós do Docker:
    ```
    sed -i "s/LimitNOFILE=[0-9a-Z]*$/LimitNOFILE=1048576/g" /usr/lib/systemd/
    system/docker.service;sed -i "s/LimitNPROC=[0-9a-Z]*$/
    LimitNPROC=1048576/g" /usr/lib/systemd/system/docker.service &&
    systemctl daemon-reload && systemctl restart docker
    ```
  - Nós de containerd:
    ```
    sed -i "s/LimitNOFILE=[0-9a-Z]*$/LimitNOFILE=1048576/g" /usr/lib/systemd/
    system/containerd.service;sed -i "s/LimitNPROC=[0-9a-Z]*$/
    LimitNPROC=1048576/g" /usr/lib/systemd/system/containerd.service &&
    systemctl daemon-reload && systemctl restart containerd
    ```
- Ubuntu:
  - Nós do Docker:
    ```
    sed -i "s/LimitNOFILE=[0-9a-Z]*$/LimitNOFILE=1048576/g" /lib/systemd/
    system/docker.service;sed -i "s/LimitNPROC=[0-9a-Z]*$/
    LimitNPROC=1048576/g" /lib/systemd/system/docker.service && systemctl
    daemon-reload && systemctl restart docker
    ```
  - Nós de containerd:
    ```
    sed -i "s/LimitNOFILE=[0-9a-Z]*$/LimitNOFILE=1048576/g" /usr/lib/systemd/
    system/containerd.service;sed -i "s/LimitNPROC=[0-9a-Z]*$/
    LimitNPROC=1048576/g" /usr/lib/systemd/system/containerd.service &&
    systemctl daemon-reload && systemctl restart containerd
    ```

Passo 3 Verifique o número máximo de identificadores de arquivo de um único processo no contêiner. Se o valor retornado for o mesmo que o valor modificado, a modificação será bem-sucedida.

- Nós do Docker:
  ```
  # cat /proc/`pidof dockerd`/limits | grep files
  Max open files            1048576              1048576              files
  ```
- Nós de containerd:
  ```
  # cat /proc/`pidof containerd`/limits | grep files
  Max open files            1048576              1048576              files
  ```

----Fim

Configurar automaticamente o número máximo de manipuladores de arquivos ao criar um nó ou pool de nós

Você pode definir o script a ser executado após a criação de um nó ou pool de nós. Ao criar um nó ou um pool de nós, você pode usar o script para configurar o número máximo de identificadores de arquivo.

Passo 1 Confirme o sistema operacional do nó ou pool de nós a ser criado, por exemplo, CentOS 7.6.

Passo 2 Teste manualmente os comandos de script em nós **no mesmo cluster e executando o mesmo sistema operacional**.

- **Alterar o número máximo de identificadores de arquivos em nível de sistema em um nó**
- **Alterar o número máximo de identificadores de arquivo para um único processo em um nó**
- **Alterar o número máximo de alças de arquivo para um único processo de contêiner**

Passo 3 Ao criar um nó ou um pool de nós, escolha **Advanced Settings** > **Post-installation Command** para adicionar comandos. **(Os comandos a seguir devem ser configurados após a verificação ser bem-sucedida.)**

- Alterar o número máximo de identificadores de arquivo no nível do sistema em um nó.
  - Faça logon no nó e verifique o arquivo **/etc/sysctl.conf**. Se o valor de **fs.file-max** tiver sido definido no arquivo, execute o seguinte comando para alterá-lo:
    ```
    sed -i "s/fs.file-max=[0-9]*$/fs.file-max=1048576/g" /etc/sysctl.conf && sysctl -p
    ```
  - Faça logon no nó e verifique o arquivo **/etc/sysctl.conf**. Se o valor de **fs.file-max** não estiver definido no arquivo, execute o seguinte comando para adicioná-lo:
    ```
    echo fs.file-max=1048576 >> /etc/sysctl.conf && sysctl -p
    ```

  No comando anterior, **fs.file-max=1048576** indica o nome do parâmetro do kernel e o valor recomendado.
- Execute o seguinte comando para alterar o número máximo de identificadores de arquivo para um único processo em um nó:
  ```
  sed -i "s/nofile.[0-9]*$/nofile 65535/g" /etc/security/limits.conf
  ```

  No comando anterior, **65535** é o número máximo recomendado de identificadores de arquivo.
- Alterar o número máximo de identificadores de arquivo para um único processo de um contêiner.
  - CentOS/EulerOS:
    ```
    sed -i "s/LimitNOFILE=[0-9a-Z]*$/LimitNOFILE=1048576/g" /usr/lib/systemd/system/docker.service;sed -i "s/LimitNPROC=[0-9a-Z]*$/LimitNPROC=1048576/g" /usr/lib/systemd/system/docker.service && systemctl daemon-reload && systemctl restart docker
    ```
  - Ubuntu:
    ```
    sed -i "s/LimitNOFILE=[0-9a-Z]*$/LimitNOFILE=1048576/g" /lib/systemd/system/docker.service;sed -i "s/LimitNPROC=[0-9a-Z]*$/LimitNPROC=1048576/g" /lib/systemd/system/docker.service && systemctl daemon-reload && systemctl restart docker
    ```

  No comando anterior, **1048576** é o número máximo recomendado de identificadores de arquivo.

O comando na figura a seguir é usado apenas como exemplo. Altere-o conforme necessário.

Passo 4 Depois que o nó for criado, efetue logon no nó para verificar se os parâmetros foram modificados com êxito.

----Fim

3.8.5.4 Modificação de parâmetros do nó de kernel

Os parâmetros padrão do kernel do Linux podem não satisfazer todos os usuários. Você pode modificar o arquivo de configuração **/etc/sysctl.conf** no nó para modificar os parâmetros do kernel.

AVISO

- Os comandos para modificar parâmetros do sistema de nó são válidos somente quando imagens públicas são usadas. Os comandos fornecidos neste documento são apenas para referência quando imagens privadas são usadas.
- Depois que o nó for reiniciado, execute o comando **sysctl -p** para atualizar o valor do parâmetro.

Tabela 3-31 Parâmetros do kernel de um nó

Parâmetro	Localização do parâmetro	Descrição	Valor recomendado
file-max	/etc/sysctl.conf	Número máximo de alças de arquivo no sistema, que podem ser ajustadas conforme necessário. Obtenção do parâmetro: `sysctl fs.file-max`	fs.file-max=1048576

Parâmetro	Localização do parâmetro	Descrição	Valor recomendado

nf_conntrack_buckets
nf_conntrack_max
	/etc/sysctl.conf	Capacidade da tabela de rastreamento de conexão, que pode ser ajustada conforme necessário.
		Uso do bucket =[nf_conntrack_count]/[nf_conntrack_buckets]
		Se o uso da CPU for maior que 0,7 por um longo período de tempo, aumente o valor dos buckets para diminuir o uso da CPU para menos de 0,7.
		Obtenção do parâmetro:
		`sysctl net.netfilter.nf_conntrack_count`
		`sysctl net.netfilter.nf_conntrack_buckets`
		`sysctl net.netfilter.nf_conntrack_max`
		NOTA
		Observação: **net.netfilter.nf_conntrack_buckets** no EulerOS 2.3, EulerOS 2.5 e CentOS 7.6 não podem ser modificados editando **/etc/sysctl.conf**, você pode modificar buckets modificando **/sys/module/nf_conntrack/parameters/hashsize**.
			O valor padrão é definido com base no tamanho da memória do nó. Para alterar o valor, consulte a seguinte fórmula:
			● net.netfilter.nf_conntrack_buckets = [nf_conntrack_count] / 0.7
			● net.netfilter.nf_conntrack_max = 4* [nf_conntrack_buckets]

net.netfilter.nf_conntrack_tcp_timeout_close	/etc/sysctl.conf	Tempo de expiração da entrada da conexão no estado de fechamento na tabela de rastreamento de conexão. Diminuir o tempo de expiração pode acelerar a reciclagem. Obtenção do parâmetro: `sysctl net.netfilter.nf_conntrack_tcp_timeout_close`	net.netfilter.nf_conntrack_tcp_timeout_close=3

Parâmetro	Localização do parâmetro	Descrição	Valor recomendado
net.netfilter.nf_conntrack_tcp_be_liberal	/etc/sysctl.conf	O valor do parâmetro é **0** ou **1**. ● **0**: a função está desativada. Todos os pacotes RST que não estão na janela TCP são marcados como inválidos. ● **1**: a função está ativada. Somente os pacotes RST que não estão na janela TCP são marcados como inválidos. Nos contêineres, habilitar esse parâmetro pode impedir que a largura de banda das conexões TCP que foram convertidas usando NAT seja limitada. Obtenção do parâmetro: `sysctl net.netfilter.nf_conntrack_tcp_be_liberal`	net.netfilter.nf_conntrack_tcp_be_liberal=1
tcp_keepalive_time	/etc/sysctl.conf	Intervalo para enviar mensagens de detecção de Keepalive através de TCP. Se esse parâmetro for definido com um valor grande, as conexões TCP podem ser suspensas na fase **Close_wait** por um longo tempo, esgotando os recursos do sistema. Obtenção do parâmetro: `sysctl net.ipv4.tcp_keepalive_time`	net.ipv4.tcp_keepalive_time=600
tcp_max_syn_backlog	/etc/sysctl.conf	Número máximo de semiconexões TCP, ou seja, o número máximo de conexões na fila **SYN_RECV**. Obtenção do parâmetro: `sysctl net.ipv4.tcp_max_syn_backlog`	net.ipv4.tcp_max_syn_backlog=8096

Parâmetro	Localização do parâmetro	Descrição	Valor recomendado
tcp_max_tw_buckets	/etc/sysctl.conf	Especifica o número máximo de soquetes no estado de espera de tempo que podem existir a qualquer momento. Se o valor do parâmetro for muito grande, os recursos do nó podem estar esgotados. Obtenção do parâmetro: `sysctl net.ipv4.tcp_max_tw_buckets`	net.ipv4.tcp_max_tw_buckets=5000
net.core.somaxconn	/etc/sysctl.conf	Número máximo de conexões TCP e tamanho máximo da fila ESTABLISHED. Se o valor do parâmetro for muito pequeno, o valor pode ser insuficiente. Obtenção do parâmetro: `sysctl net.core.somaxconn`	net.core.somaxconn=32768
max_user_instances	/etc/sysctl.conf	Número máximo de instâncias de inotify permitidas para cada usuário. Se o valor do parâmetro for muito pequeno, o número de instâncias de inotify pode ser insuficiente nos contêineres. Obtenção do parâmetro: `sysctl fs.inotify.max_user_instances`	fs.inotify.max_user_instances=8192
max_user_watches	/etc/sysctl.conf	Número máximo de diretórios de todas as instâncias de monitoramento. Se o valor do parâmetro for muito pequeno, o número de diretórios pode ser insuficiente nos contêineres. Obtenção do parâmetro: `sysctl fs.inotify.max_user_watches`	fs.inotify.max_user_watches=524288

Parâmetro	Localização do parâmetro	Descrição	Valor recomendado
netdev_max_backlog	/etc/sysctl.conf	Tamanho da fila de recebimento de pacotes da pilha de protocolos de rede. Se o valor do parâmetro for muito pequeno, o tamanho da fila pode ser insuficiente. Obtenção do parâmetro: `sysctl net.core.netdev_max_backlog`	net.core.netdev_max_backlog=16384

net.core.wmem_max
net.core.rmem_max
	/etc/sysctl.conf	Tamanho da memória (bytes) do buffer de envio e recebimento. Se este parâmetro estiver definido como um valor pequeno, o tamanho da memória poderá ser insuficiente em cenários de arquivos grandes.
		Obtenção do parâmetro:
		`sysctl net.core.wmem_max`
		`sysctl net.core.rmem_max`
			net.core.wmem_max=16777216
			net.core.rmem_max=16777216

net.ipv4.neigh.default.gc_thresh1
net.ipv4.neigh.default.gc_thresh2
net.ipv4.neigh.default.gc_thresh3
	/etc/sysctl.conf	Otimização da coleta de lixo de entradas ARP.
		Obtenção do parâmetro:
		`sysctl net.ipv4.neigh.default.gc_thresh1`
		`sysctl net.ipv4.neigh.default.gc_thresh2`
		`sysctl net.ipv4.neigh.default.gc_thresh3`
			net.ipv4.neigh.default.gc_thresh1=0
			net.ipv4.neigh.default.gc_thresh2=4096
			net.ipv4.neigh.default.gc_thresh3=8192

vm.max_map_count	/etc/sysctl.conf	Se este parâmetro é ajustado a um valor pequeno, uma mensagem é indicada indicando que o espaço é insuficiente durante a instalação de ELK. Obtenção do parâmetro: `sysctl vm.max_map_count`	vm.max_map_count=262144

Modificação de parâmetros de kernel de um nó

Tabela 3-31 lista os parâmetros do kernel dos nós. A seguir, descrevemos como alterar o valor de **tcp_keepalive_time**, que indica o intervalo para enviar mensagens de detecção de Keepalive por TCP.

Passo 1 Faça logon no nó e verifique o arquivo **/etc/sysctl.conf**.

```
cat /etc/sysctl.conf
```

Passo 2 Modifique o parâmetro **net.ipv4.tcp_keepalive_time**. **net.ipv4.tcp_keepalive_time=600** indica o nome do parâmetro do kernel e o valor recomendado. Para obter detalhes sobre o valor recomendado, consulte **Tabela 3-31**.

Para modificar outros parâmetros do kernel, substitua os nomes e valores de parâmetros nos comandos, referindo-se a **Tabela 3-31**.

- Se **net.ipv4.tcp_keepalive_time** tiver sido definido no arquivo **sysctl.conf**, execute o seguinte comando para alterar o valor:
  ```
  sed -i "s/net.ipv4.tcp_keepalive_time=[0-9]*$/
  net.ipv4.tcp_keepalive_time=600/g" /etc/sysctl.conf && sysctl -p
  ```
- Se **net.ipv4.tcp_keepalive_time** não estiver definido no arquivo **sysctl.conf**, execute o seguinte comando para o adicionar:
  ```
  echo net.ipv4.tcp_keepalive_time=600 >> /etc/sysctl.conf && sysctl -p
  ```

Passo 3 Execute o comando em **Tabela 3-31** para verificar se a modificação foi bem-sucedida. Se o valor retornado for o mesmo que o modificado, a modificação será bem-sucedida.

```
# sysctl net.ipv4.tcp_keepalive_time
net.ipv4.tcp_keepalive_time = 600
```

----Fim

Configurar automaticamente os parâmetros do kernel ao criar um nó ou um pool de nós

Você pode definir o script a ser executado após a criação de um nó ou pool de nós. Ao criar um nó ou um pool de nós, você pode usar o script para configurar os parâmetros do kernel.

O parâmetro **tcp_keepalive_time** é usado como um exemplo para descrever como alterar o intervalo para enviar mensagens de detecção de Keepalive por TCP. O valor é o valor recomendado em **Tabela 3-31**.

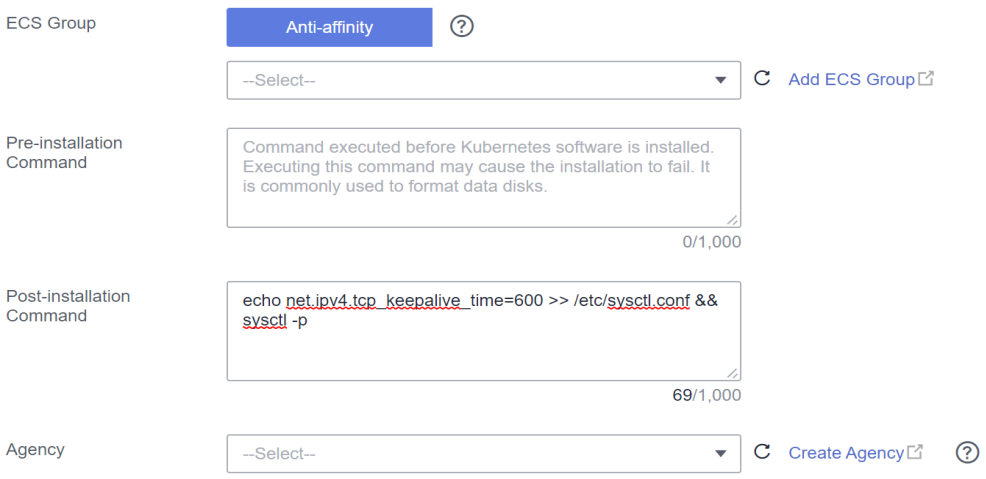
Passo 1 Confirme o sistema operacional do nó ou pool de nós a ser criado, por exemplo, CentOS 7.6.

Passo 2 Teste manualmente os comandos de script em nós **no mesmo cluster e executando o mesmo sistema operacional**. Para obter detalhes sobre como executar manualmente o script, consulte **Modificação de parâmetros de kernel de um nó**.

Passo 3 Ao criar um nó ou um pool de nós, escolha **Advanced Settings** > **Post-installation Command** para adicionar comandos. **(Os comandos a seguir devem ser configurados após a verificação ser bem-sucedida.)** Para modificar outros parâmetros do kernel, substitua os nomes e valores de parâmetros nos comandos, referindo-se a **Tabela 3-31**.

- Faça logon no nó e verifique o arquivo **/etc/sysctl.conf**. Se **net.ipv4.tcp_keepalive_time** tiver sido definido no arquivo, execute o seguinte comando para alterá-lo:
  ```
  sed -i "s/net.ipv4.tcp_keepalive_time=[0-9]*$/
  net.ipv4.tcp_keepalive_time=600/g" /etc/sysctl.conf && sysctl -p
  ```
- Faça logon no nó e verifique o arquivo **/etc/sysctl.conf**. Se **net.ipv4.tcp_keepalive_time** não estiver definido no arquivo, execute o seguinte comando para adicioná-lo:
  ```
  echo net.ipv4.tcp_keepalive_time=600 >> /etc/sysctl.conf && sysctl -p
  ```

O comando na figura a seguir é usado apenas como exemplo. Altere-o conforme necessário.



Passo 4 Depois que o nó for criado, faça logon no nó e execute o comando em **Tabela 3-31** para verificar se a modificação foi bem-sucedida.

----Fim

3.8.5.5 Alteração de limites de ID de processo (kernel.pid_max)

Contexto

Os IDs de processos (PIDs) são um recurso fundamental nos nós. É trivial atingir o limite de tarefas sem atingir nenhum outros limites de recursos, o que pode causar instabilidade em uma máquina host.

Você pode ajustar o limite de PID (kernel.pid_max) de acordo com os requisitos de serviço.

Padrões do kernel.pid_max

A partir de janeiro de 2022, o CCE altera o valor padrão de kernel.pid_max para **4194304** para os nós de EulerOS 2.5, CentOS 7.6 e Ubuntu 18.04 em clusters de v1.17 ou posterior. Condições específicas:

- Versão do cluster: v1.17.17 ou posterior
- Criação de nó: após 30 de janeiro de 2022

Se as duas condições anteriores não forem atendidas, **kernel.pid_max** nos nós de EulerOS 2.5, CentOS 7.6 e Ubuntu 18.04 o padrão é **32768**.

Tabela 3-32 Padrões do kernel.pid_max

SO	Clusters de 1.17.9 e anteriores	Clusters de 1.17.17 e posteriores	
		Nós criados em ou antes de 30 de janeiro de 2022	Nós criados após 30 de janeiro de 2022
EulerOS 2.5	32768	32768	4194304
CentOS 7.6	32768	32768	4194304

Ubuntu 18.04	N/D	32768	4194304
EulerOS 2.3	57344	57344	57344
EulerOS 2.9	N/D	4194304	4194304

Sugestão de alteração

- EulerOS 2.3: altere o padrão para **4194304** para todos os nós. Para mais detalhes, consulte **Alterar o kernel.pid_max de um nó**. Use um script de pré-instalação para fazer isso para novos nós e pools de nós. Para mais detalhes, veja **Configurar kernel.pid_max ao criar um pool de nós** e **Configurar kernel.pid_max ao criar um nó**.
- EulerOS 2.5, CentOS 7.6 e Ubuntu 18.04:
  - altere o valor de **kernel.pid_max** para **4194304** para nós criados em 30 de janeiro de 2022 ou anterior em clusters de v1.17.17 ou posterior. Para mais detalhes, consulte **Alterar o kernel.pid_max de um nó**.
  - Para clusters de 1.17.9 e anteriores:
    - altere o valor de **kernel.pid_max** para **4194304** para nós existentes. Para mais detalhes, consulte **Alterar o kernel.pid_max de um nó**.
    - Use um script de pré-instalação para fazer isso para novos nós e pools de nós. Para mais detalhes, veja **Configurar kernel.pid_max ao criar um pool de nós** e **Configurar kernel.pid_max ao criar um nó**.

Exibir kernel.pid_max

Faça logon no nó e execute o seguinte comando para obter o valor de **kernel.pid_max**:

sysctl kernel.pid_max

```
# sysctl kernel.pid_max
kernel.pid_max = 32768
```

Altere **kernel.pid_max**, se necessário, conforme instruído em **Alterar o kernel.pid_max de um nó**.

Verificar os PIDs de nó

Faça logon no nó e execute o seguinte comando para verificar quantos PIDs estão em uso:

ps -eflL | wc -l

```
# ps -eflL | wc -l
691
```

Alterar o kernel.pid_max de um nó

Faça logon no nó e execute o seguinte comando: **4194304** indica o valor de **kernel.pid_max** e é usado como um exemplo aqui.

echo kernel.pid_max = *4194304* >> /etc/sysctl.conf && sysctl -p

echo *4194304* > /sys/fs/cgroup/pids/kubepods/pids.max

Execute os seguintes comandos para verificar se o valor retornado é o mesmo que você configurou:

```
# sysctl kernel.pid_max
kernel.pid_max = 4194304
# cat /sys/fs/cgroup/pids/kubepods/pids.max
4194304
```

Configurar kernel.pid_max ao criar um pool de nós

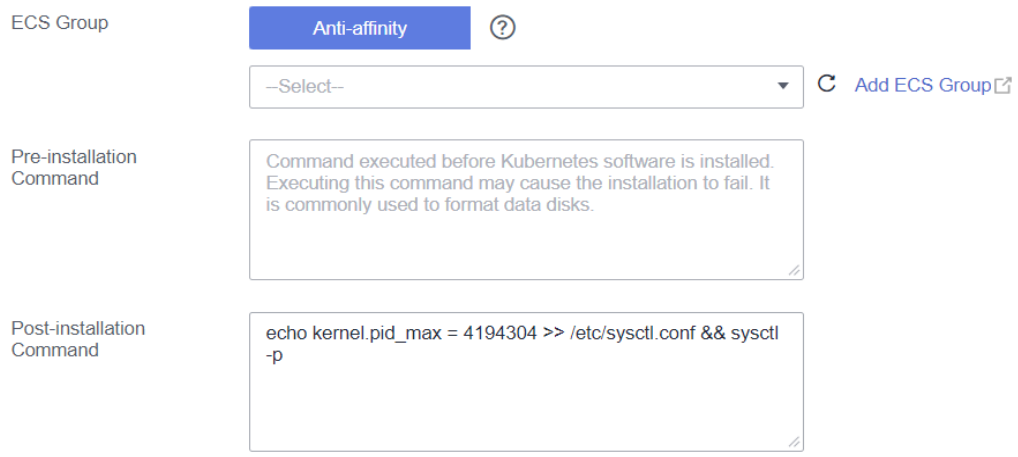
EulerOS 2.3: configuração necessária.

EulerOS 2.5, CentOS 7.6 e Ubuntu 18.04: **configuração necessária** para clusters v1.17.9 e anteriores. **Configuração desnecessária** para clusters de v1.17.17 e posterior porque o valor foi alterado.

Você pode configurar **kernel.pid_max** no script de pré-instalação para criar um nó de um pool de nós.

Ao criar um pool de nós, escolha **Advanced Settings** > **Post-installation Command** e adicione o seguinte comando:

echo kernel.pid_max = *4194304* >> /etc/sysctl.conf && sysctl -p



Configurar kernel.pid_max ao criar um nó

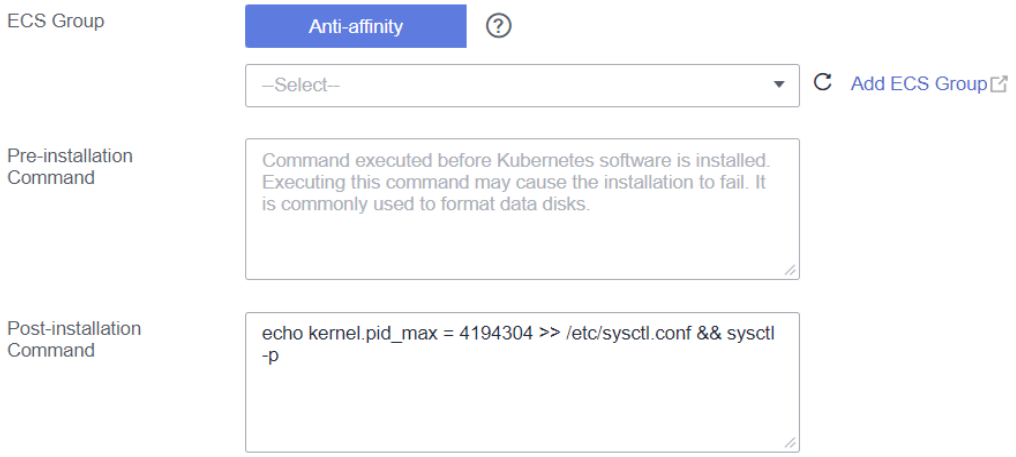
EulerOS 2.3: configuração necessária.

EulerOS 2.5, CentOS 7.6 e Ubuntu 18.04: **configuração necessária** para clusters v1.17.9 e anteriores. **Configuração desnecessária** para clusters de v1.17.17 e posterior porque o valor foi alterado.

Você pode configurar **kernel.pid_max** usando o script de pré-instalação ao criar um nó.

Escolha**Advanced Settings** > **Post-installation Command** e adicione o seguinte comando:

echo kernel.pid_max = *4194304* >> /etc/sysctl.conf && sysctl -p



3.8.6 Política de detecção de falhas de nó

A função de detecção de falha de nó depende do complemento **node-problem-detector (npd)**. As instâncias complementares são executadas em nós e monitoram nós. Esta seção descreve como ativar a detecção de falha de nó.

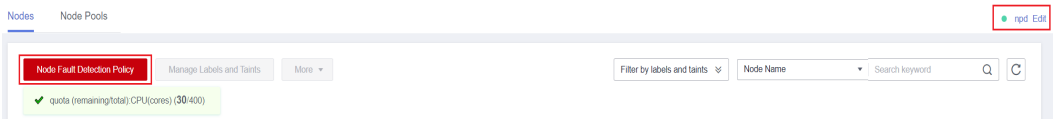
Pré-requisitos

O complemento **Detector de problema de nó do CCE** foi instalado no cluster.

Ativar a detecção de falhas de nó

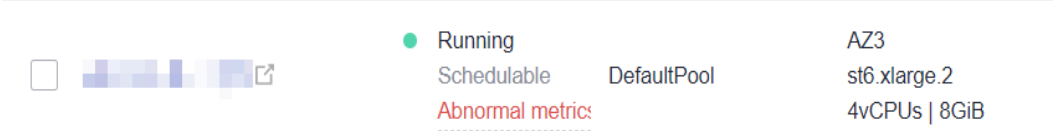
Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster.

Passo 2 No painel de navegação à esquerda, escolha **Nodes**. Verifique se o complemento npd foi instalado no cluster ou se o complemento foi atualizado para a versão mais recente. Depois que o complemento npd for instalado, você poderá usar a função de detecção de falhas.

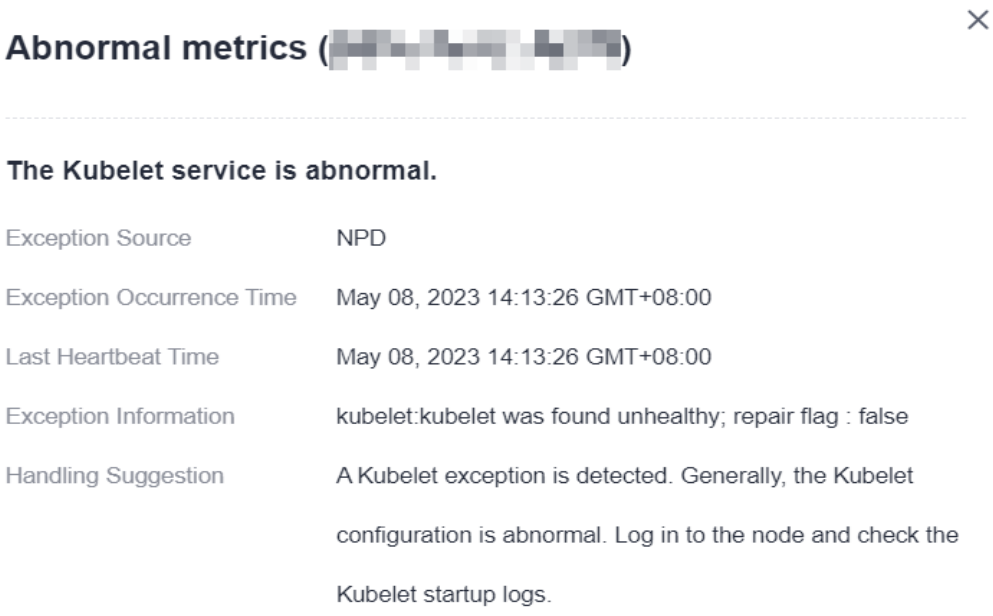


Passo 3 Se o complemento npd estiver sendo executado corretamente, clique em **Node Fault Detection Policy** para exibir os itens atuais de detecção de falhas. Para obter detalhes sobre a lista de itens de verificação do npd, consulte **Itens de verificação de NPD**.

Passo 4 Se o resultado da verificação do nó atual for anormal, uma mensagem será exibida na lista de nós, indicando que a métrica é anormal.



Passo 5 Você pode clicar em **Abnormal metrics** e corrigir a falha conforme solicitado.



----Fim

Itens de verificação personalizados

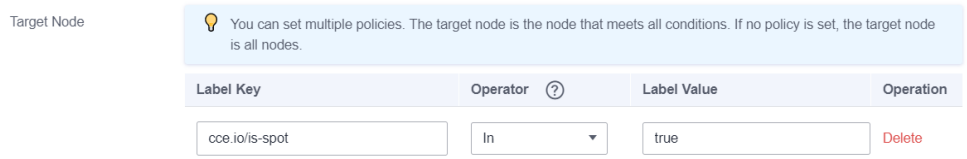
Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster.

Passo 2 Escolha Node Management à esquerda e clique em **Node Fault Detection Policy**.

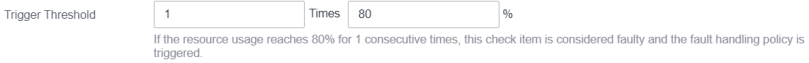
Passo 3 Na página exibida, visualize os itens de verificação atuais. Clique em **Edit** na coluna **Operation** e edite as verificações.

Atualmente, as seguintes configurações são suportadas:

- **Enable/Disable**: ativar ou desativar um item de verificação.
- **Target Node**: por padrão, verifique os itens executados em todos os nós. Você pode alterar o limite de falhas com base em cenários especiais. Por exemplo, a verificação de recuperação de interrupção do ECS de preço à vista é executada apenas no nó do ECS de preço à vista.



- **Trigger Threshold**: os limites padrão correspondem a cenários de falha comuns. Você pode personalizar e modificar os limites de falha conforme necessário. Por exemplo, altere o limite para disparar a exaustão da tabela de rastreamento de conexão de 90% para 80%.



- **Check Period**: o período de verificação padrão é de 30 segundos. Você pode modificar esse parâmetro conforme necessário.

- **Troubleshooting Strategy**: após ocorrer uma falha, você pode selecionar as estratégias listadas na tabela a seguir.

  Tabela 3-33 Estratégias de solução de problemas

Estratégia de solução de problemas	Efeito
Exceção de solicitação	Os eventos do Kubernetes são relatados.
Desativação de agendamento	Os eventos do Kubernetes são relatados e a mancha do **NoSchedule** é adicionada ao nó.
Evicção de carga de nós	Os eventos do Kubernetes são relatados e a mancha do **NoExecute** é adicionada ao nó. Essa operação eliminará cargas de trabalho no nó e interromperá os serviços. Tenha cuidado ao realizar esta operação.

----Fim

Itens de verificação de NPD

NOTA

Os itens de verificação são suportados apenas em 1.16.0 e versões posteriores.

Verifique os itens de cobertura de eventos e status.

- Relacionados a eventos

  Para itens de verificação relacionados a eventos, quando ocorre um problema, o NPD relata um evento para o servidor da API. O tipo de evento pode ser **Normal** (evento normal) ou **Warning** (evento anormal).

  Tabela 3-34 Itens de verificação relacionados a eventos

Item de verificação	Função	Descrição
OOMKilling	Ouvir os logs do kernel e verifique se os eventos OOM ocorrem e são relatados. Cenário típico: quando o uso de memória de um processo em um contêiner excede o limite, a OOM é acionada e o processo é encerrado.	Evento de aviso Objeto de escuta: **/dev/kmsg** Regra de correspondência: "Killed process \\d+ (.+) total-vm:\\d+kB, anon-rss:\\d+kB, file-rss:\\d +kB.*"

Item de verificação	Função	Descrição
TaskHung	Ouvir os logs do kernel e verifique se os eventos taskHung ocorrem e são relatados. Cenário típico: a suspensão de I/O de disco causa a suspensão do processo.	Evento de aviso Objeto de escuta: **/dev/kmsg** Regra de correspondência: "task \\S+:\\w+ blocked for more than \\w+ seconds\\."
ReadonlyFilesystem	Verificar se o erro **Remount root filesystem read-only** ocorre no kernel do sistema ouvindo os logs do kernel. Cenário típico: um usuário separa um disco de dados de um nó por engano no ECS, e as aplicações gravam dados continuamente no ponto de montagem do disco de dados. Como resultado, ocorre um erro de I/O no kernel e o disco é montado novamente como um disco de somente leitura. **NOTA** Se o rootfs dos node pods for do tipo mapeador de dispositivos, ocorrerá um erro no thin pool se um disco de dados for desanexado. Isso afetará o NPD e o NPD não poderá detectar falhas de nó.	Evento de aviso Objeto de escuta: **/dev/kmsg** Regra de correspondência: **Remounting filesystem read-only**

- Status-related

  Para itens de verificação relacionados ao status, quando ocorre um problema, o NPD relata um evento para o servidor da API e altera o status do nó de forma síncrona. Esta função pode ser usada em conjunto com o **isolamento de falhas do Node-problem-controller** para isolar nós.

  Se o período de verificação não for especificado nos seguintes itens de verificação, o período padrão será de 30 segundos.

  Tabela 3-35 Verificar componentes do sistema

Item de verificação	Função	Descrição
Erro no componente de rede do contêiner CNIProblem	Verificar o status dos componentes da CNI (componentes da rede do contêiner).	Nenhuma

Item de verificação	Função	Descrição
Erro no componente de tempo de execução do contêiner CRIProblem	Verificar o status do Docker e do contêiner dos componentes de CRI (componentes de tempo de execução do contêiner).	Objeto de verificar: Docker ou containerd
Reinícios frequentes do Kubelet FrequentKubeletRestart	Retroceder periodicamente os logs do sistema para verificar se o componente importante de Kubelet reinicia com frequência.	● Limite padrão: 10 reinícios em 10 minutos Se o Kubelet for reiniciado 10 vezes em 10 minutos, isso indica que o sistema é reiniciado com frequência e um alarme de falha é gerado. ● Objeto de escuta: logs no diretório **/run/log/journal** **NOTA** Os sistemas operacionais Ubuntu e HCE 2.0 não suportam os itens de verificação anteriores devido a formatos de log incompatíveis.
Reinícios frequentes do Docker FrequentDockerRestart	Retroceder periodicamente os logs do sistema para verificar se o tempo de execução do contêiner Docker é reiniciado com frequência.	
Reinícios frequentes do containerd FrequentContainerdRestart	Retroceder periodicamente os logs do sistema para verificar se o tempo de execução do contêiner containerd reinicia com frequência.	
erro de kubelet KubeletProblem	Verificar o status do componente importante de Kubelet.	Nenhuma
erro de kube-proxy KubeProxyProblem	Verificar o status do componente importante de kube-proxy.	Nenhuma

Tabela 3-36 Verificar métricas do sistema

Item de verificação	Função	Descrição
Tabela conntrack cheia ConntrackFullProblem	Verificar se a tabela conntrack está cheia.	● Limite padrão: 90% ● Utilização: **nf_conntrack_count** ● Valor máximo: **nf_conntrack_max**

Item de verificação	Função	Descrição
Recursos de disco insuficientes DiskProblem	Verificar o uso do disco do sistema e dos discos de dados do CCE (incluindo o disco lógico de CRI e o disco lógico de kubelet) no nó.	● Limite padrão: 90% ● Origem: `df -h` Atualmente, discos de dados adicionais não são suportados.
Manipuladores de arquivo insuficientes FDProblem	Verificar se os manipuladores do arquivo FD estão esgotados.	● Limite padrão: 90% ● Utilização: o primeiro valor em **/proc/sys/fs/file-nr** ● Valor máximo: o terceiro valor em **/proc/sys/fs/file-nr**
Memória de nó insuficiente MemoryProblem	Verificar se a memória está gasta.	● Limite padrão: 80% ● Uso: **MemTotal-MemAvailable** em **/proc/meminfo** ● Valor máximo: **MemTotal** em **/proc/meminfo**
Recursos de processo insuficientes PIDProblem	Verificar se os recursos do processo PID estão esgotados.	● Limite padrão: 90% ● Utilização: **nr_threads in /proc/loadavg** ● Valor máximo: menor valor entre **/proc/sys/kernel/pid_max** e **/proc/sys/kernel/threads-max**.

Tabela 3-37 Verificar o armazenamento

Item de verificação	Função	Descrição
Disco de somente leitura DiskReadonly	Execute periodicamente testes de gravação no disco do sistema e nos discos de dados do CCE (incluindo o disco lógico de CRI e o disco lógico de Kubelet) do nó para verificar a disponibilidade dos discos importantes.	Caminhos de detecção: ● /mnt/paas/kubernetes/kubelet/ ● /var/lib/docker/ ● /var/lib/containerd/ ● /var/paas/sys/log/cceaddon-npd/ O arquivo temporário **npd-disk-write-ping** é gerado no caminho de detecção. Atualmente, discos de dados adicionais não são suportados.

Item de verificação	Função	Descrição
Erro do pool de armazenamento emptyDir EmptyDirVolumeGroupStatusError	Verificar se o grupo de volume efêmero no nó é normal. Impacto: o pod que depende do pool de armazenamento não pode gravar dados no volume temporário. O volume temporário é montado novamente como um sistema de arquivos somente leitura pelo kernel devido a um erro de I/O. Cenário típico: ao criar um nó, um usuário configura dois discos de dados como um pool de armazenamento de volume temporário. O usuário exclui alguns discos de dados por engano. Como resultado, o pool de armazenamento se torna anormal.	● Período da detecção: 30s ● Origem: `vgs -o vg_name, vg_attr` ● Princípio: verifique se o VG (storage pool) está no estado P. Se sim, alguns PVs (discos de dados) são perdidos. ● Agendamento conjunto: o agendador pode identificar automaticamente um erro de pool de armazenamento PV e impedir que pods que dependem do pool de armazenamento sejam gendados para o nó. ● Cenário excepcional: o complemento NPD não pode detectar a perda de todos os PVs (discos de dados), resultando na perda de VGs (pools de armazenamento). Nesse caso, o kubelet isola automaticamente o nó, detecta a perda de VGs (pools de armazenamento) e atualiza os recursos correspondentes em **nodestatus.allocatable** para **0**. Isso impede que os pods que dependem do pool de armazenamento sejam agendados para o nó. O dano de um único PV não pode ser detectado por este item de verificação, mas pelo item de verificação ReadonlyFilesystem.
Erro do pool de armazenamento PV LocalPvVolumeGroupStatusError	Verificar o grupo PV no nó. Impacto: os pods que dependem do pool de armazenamento não podem gravar dados no volume persistente. O volume persistente é remontado como um sistema de arquivos somente leitura pelo kernel devido a um erro de I/O. Cenário típico: ao criar um nó, um usuário configura dois discos de dados como um pool de armazenamento de volume persistente. Alguns discos de dados são apagados por engano.	

Item de verificação	Função	Descrição
Erro do ponto de montagem MountPointProblem	Verificar o ponto de montagem no nó. Definição excepcional: você não pode acessar o ponto de montagem executando o comando **cd**. Cenário típico: Network File System (NFS), por exemplo, obsfs e s3fs é montado em um nó. Quando a conexão é anormal devido a exceções de servidor do NFS de rede ou de par, todos os processos que acessam o ponto de montagem são suspensos. Por exemplo, durante uma atualização de cluster, um kubelet é reiniciado e todos os pontos de montagem são verificados. Se o ponto de montagem anormal for detectado, a atualização falhará.	Alternativamente, você pode executar o seguinte comando: `for dir in `df -h | grep -v "Mounted on" | awk "{print \\$NF}"`;do cd $dir; done && echo "ok"`
I/O de disco suspensa DiskHung	Verificar se a suspensão de I/O ocorre em todos os discos no nó, ou seja, se as operações de leitura e gravação de I/O não são respondidas. Definição de suspensão de I/O: o sistema não responde a solicitações de I/O de disco e alguns processos estão no estado D. Cenário típico: os discos não podem responder devido a drivers de disco rígido do sistema operacional anormais ou falhas graves na rede subjacente.	● Objeto de verificar: todos os discos de dados ● Origem: /proc/diskstat Alternativamente, você pode executar o seguinte comando: `iostat -xmt 1` ● Limite: – Uso médio: ioutil >= 0,99 – Comprimento médio da fila de I/O: avgqu-sz >= 1 – Volume médio de transferência de I/O: iops (w/s) + ioth (wMB/s) <= 1 **NOTA** Em alguns sistemas operacionais, nenhum dado é alterado durante a I/O. Nesse caso, calcule o uso do tempo de I/O da CPU. O valor de iowait deve ser maior que 0,8.

Item de verificação	Função	Descrição
I/O de disco lenta DiskSlow	Verificar se todos os discos no nó têm I/Os lentas, ou seja, se as I/Os respondem lentamente. Cenário típico: os discos EVS têm I/Os lentas devido à flutuação da rede.	● Objeto de verificar: todos os discos de dados ● Origem: /proc/diskstat Alternativamente, você pode executar o seguinte comando: `iostat -xmt 1` ● Limite padrão: Latência média de I/O: await >= 5000 ms **NOTA** Se as solicitações de I/O não forem respondidas e os dados **await** não forem atualizados, esse item de verificação será inválido.

Tabela 3-38 Outros itens de verificação

Item de verificação	Função	Descrição
NTP anormal NTPProblem	Verificar se o serviço de sincronização de relógio de nó ntpd ou chronyd está sendo executado corretamente e se um desvio de tempo do sistema é causado.	Limite de deslocamento de relógio padrão: 8000 ms
Erro no processo D ProcessD	Verificar se há um processo D no nó.	Limite padrão: 10 processos anormais detectados por três vezes consecutivas Origem: ● /proc/{PID}/stat ● Como alternativa, você pode executar o comando **ps aux**. Cenário excepcional: o item de verificação ProcessD ignora os processos D residentes (heartbeat e update) dos quais o driver SDI no nó do BMS depende.
Erro de processo Z ProcessZ	Verificar se o nó tem processos no estado Z.	

Item de verificação	Função	Descrição
Erro de ResolvConf ResolvConfFileProblem	Verificar se o arquivo ResolvConf foi perdido. Verificar se o arquivo ResolvConf está normal. Definição excepcional: nenhum servidor de resolução de nome de domínio upstream (nameserver) está incluído.	Objeto: **/etc/resolv.conf**
Evento agendado existente ScheduledEvent	Verificar se existem eventos de migração ao vivo agendados no nó. Um evento de plano de migração ao vivo geralmente é acionado por uma falha de hardware e é um método de retificação automática de falhas na camada IaaS. Cenário típico: o host está defeituoso. Por exemplo, o ventilador está danificado ou o disco tem setores defeituosos. Como resultado, a migração ao vivo é acionada para VMs.	Origem: ● http://169.254.169.254/meta-data/latest/events/scheduled Este item de verificação é um recurso Alpha e está desativado por padrão.

O componente de kubelet tem os seguintes itens de verificação padrão, que têm bugs ou defeitos. Você pode corrigi-los atualizando o cluster ou usando o NPD.

Tabela 3-39 Itens de verificação padrão do kubelet

Item de verificação	Função	Descrição
Recursos de PID insuficientes. PIDPressure	Verificar se os PIDs são suficientes.	● Intervalo: 10 segundos ● Limite: 90% ● Defeito: na versão 1.23.1 da comunidade e versões anteriores, este item de verificação torna-se inválido quando mais de 65535 PIDs são usados. Para obter detalhes, consulte **issue 107107**. Na versão 1.24 da comunidade e versões anteriores, thread-max não é considerado neste item de verificação.
Memória insuficiente MemoryPressure	Verificar se a memória alocável para os contêineres é suficiente.	● Intervalo: 10 segundos ● Limite: máx. 100 MiB ● Alocável = memória total de um nó – memória reservada de um nó ● Defeito: esse item de verificação verifica apenas a memória consumida pelos contêineres e não considera a consumida por outros elementos no nó.
Recursos de disco insuficientes DiskPressure	Verificar o uso do disco e o uso de inodes dos discos do kubelet e do Docker.	● Intervalo: 10 segundos ● Limite: 90%

4 Pools de nós

4.1 Visão geral do pool de nós

Introdução

O CCE apresenta pools de nós para ajudá-lo a gerenciar melhor os nós em clusters do Kubernetes. Um pool de nós contém um nó ou um grupo de nós com configuração idêntica em um cluster.

Você pode criar pools de nós personalizados no console do CCE. Com pools de nós, você pode criar, gerenciar e destruir nós rapidamente sem afetar o cluster. Todos os nós em um pool de nós personalizados têm parâmetros e tipo de nó idênticos. Você não pode configurar um único nó em um pool de nós; quaisquer alterações de configuração afetam todos os nós no pool de nós.

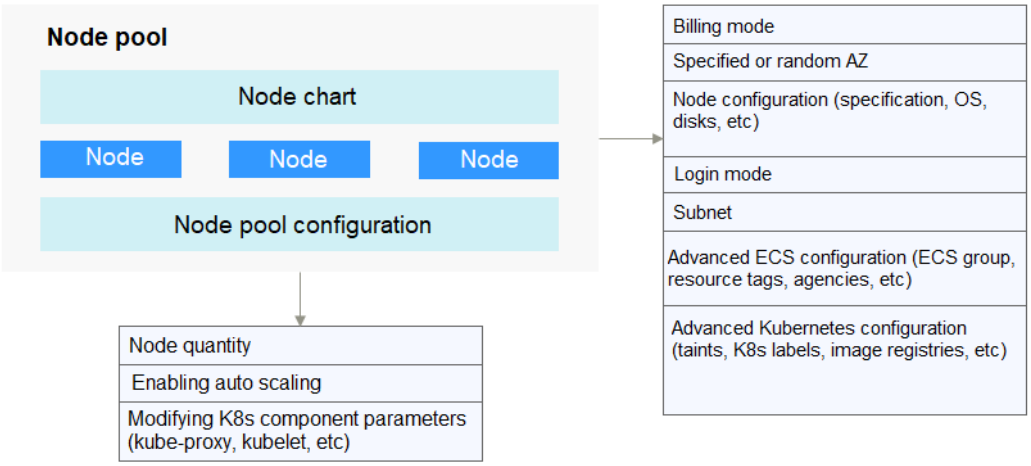
Você também pode usar pools de nós para dimensionamento automático (suportado apenas por pools de nós de pagamento por uso).

- Quando um pod em um cluster não pode ser agendado devido a recursos insuficientes, a expansão pode ser acionada automaticamente.
- Quando há um nó ocioso ou um limite de métrica de monitoramento é atingido, a redução pode ser acionada automaticamente.

Esta seção descreve como os pools de nós funcionam no CCE e como criar e gerenciar pools de nós.

Arquitetura do pool de nós

Figura 4-1 Arquitetura geral de um pool de nós



Geralmente, todos os nós em um pool de nós têm os mesmos atributos a seguir:

- Sistema operacional do nó
- Especificações do nó
- Modo de logon do nó
- Tempo de execução do contêiner do nó
- Parâmetros de inicialização de componentes do Kubernetes em um nó
- Script de inicialização definido pelo usuário de um nó
- **Rótulos e manchas** do **Kubernetes**

O CCE fornece os seguintes atributos estendidos para pools de nós:

- SO do pool de nós
- Número máximo de pods em cada nó em um pool de nós

Descrição de DefaultPool

O DefaultPool não é um pool de nós real. Ele **classifica** apenas os nós que não estão nos pools de nós criados pelo usuário. Esses nós são criados diretamente no console ou chamando APIs. O DefaultPool não oferece suporte a funções de pool de nós criadas pelo usuário, incluindo dimensionamento e configuração de parâmetros. O DefaultPool não pode ser editado, excluído, expandido ou dimensionado automaticamente, e os nós nele não podem ser migrados.

Cenários aplicáveis

Quando um cluster de grande escala é necessário, é aconselhável usar pools de nós para gerenciar nós.

A tabela a seguir descreve vários cenários de gerenciamento de cluster em larga escala e as funções dos pools de nós em cada cenário.

Tabela 4-1 Usar pools de nós para diferentes cenários de gerenciamento

Cenário	Função
Vários nós heterogêneos (com diferentes modelos e configurações) no cluster	Os nós podem ser agrupados em diferentes pools para gerenciamento.
dimensionado de nó frequente necessário em um cluster	Os pools de nós suportam o escalonamento automático para adicionar ou reduzir nós dinamicamente.
Regras complexas de agendamento de aplicações em um cluster	As tags de pool de nós podem ser usadas para especificar rapidamente as regras de agendamento de serviços.

Funções e precauções

Função	Descrição	Precaução
Criar um pool de nós	Adicionar um pool de nós.	Recomenda-se que um cluster não contenha mais de que 100 conjuntos de nós.
Excluir um pool de nós	Quando um pool de nós é excluído, os nós no pool de nós são excluídos primeiro. Quando um pool de nós anual/mensal é excluído, os nós são migrados para o pool de nós padrão primeiro. As cargas de trabalho nos nós originais são migradas automaticamente para os nós disponíveis em outros pools de nós.	Se os pods no pool de nós tiverem um seletor de nó específico e nenhum dos outros nós no cluster satisfizer o seletor de nó, os pods se tornarão não escalonáveis.
Ativar o dimensionamento automático para um pool de nós	Depois que o dimensionamento automático for ativado, os nós serão criados ou excluídos automaticamente no pool de nós com base nas cargas do cluster.	Não armazene dados importantes em nós em um pool de nós, pois os nós podem ser excluídos após o dimensionamento. Os dados nos nós excluídos não podem ser restaurados.
Ativar o dimensionamento automático para um pool de nós	Depois que o dimensionamento automático for desativado, o número de nós em um pool de nós não será alterado automaticamente com as cargas do cluster.	Nenhuma

Função	Descrição	Precaução
Ajustar o tamanho de um pool de nós	O número de nós em um pool de nós pode ser ajustado diretamente. Se o número de nós for reduzido, os nós serão removidos aleatoriamente do pool de nós atual.	Depois que o dimensionamento automático estiver ativado, não é aconselhável ajustar manualmente o tamanho do pool de nós.
Alterar configurações de pool de nós	Você pode modificar o nome do pool de nós, a quantidade de nós, os rótulos do Kubernetes (e sua quantidade), as tags de recursos e as manchas e ajustar as configurações de disco, sistema operacional e mecanismo de contêiner do pool de nós.	Os rótulos e manchas do Kubernetes excluídos ou adicionados (bem como sua quantidade) serão aplicados a todos os nós no pool de nós, o que pode causar o reagendamento do pod. Portanto, tenha cuidado ao realizar esta operação.
Remover um nó de um pool de nós	Os nós em um pool de nós podem ser migrados para o pool de nós padrão do mesmo cluster.	Os nós no pool de nós padrão não podem ser migrados para outros pools de nós, e os nós em um pool de nós criado pelo usuário não podem ser migrados para outros pools de nós criados pelo usuário.
Clonar um pool de nós	Você pode copiar a configuração de um pool de nós existente para criar um novo pool de nós.	Nenhuma
Configurar parâmetros do Kubernetes	Você pode configurar os componentes principais com granularidade fina.	• Esta função é suportada apenas em clusters de v1.15 e posteriores. Ela não é exibida para versões anteriores à v1.15. • O pool de nós padrão DefaultPool não oferece suporte a esse tipo de configuração.

Implementar uma carga de trabalho em um pool de nós especificado

Ao criar uma carga de trabalho, você pode restringir pods para serem executados em um pool de nós especificado.

Por exemplo, no console do CCE, você pode definir a afinidade entre a carga de trabalho e o nó na página de guia **Scheduling Policies** na página de detalhes da carga de trabalho para implementar forçosamente a carga de trabalho em um pool de nós específico. Dessa forma, a carga de trabalho é executada apenas em nós no pool de nós. Para controlar melhor onde a carga de trabalho deve ser agendada, você pode usar políticas de afinidade ou antiafinidade entre cargas de trabalho e nós descritos em **Política de agendamento (afinidade/antiafinidade)**.

Por exemplo, você pode usar a solicitação de recurso do contêiner como um nodeSelector para que as cargas de trabalho sejam executadas apenas nos nós que atendem à solicitação de recurso.

Se o arquivo de definição de carga de trabalho definir um contêiner que exija quatro CPUs, o agendador não escolherá os nós com duas CPUs para executar cargas de trabalho.

Operações relacionadas

Você pode fazer logon no console do CCE e consultar as seguintes seções para executar operações em pools de nós:

- **Criação de um pool de nós**
- **Gerenciamento de um pool de nós**
- **Criação de uma Implantação**
- **Política de agendamento (afinidade/antiafinidade)**

4.2 Criação de um pool de nós

Cenário

Esta seção descreve como criar um pool de nós e executar operações no pool de nós. Para obter detalhes sobre como um pool de nós funciona, consulte **Visão geral do pool de nós**.

Restrições

- O complemento autoscaler precisa ser instalado para o dimensionamento automático do nó. Para obter detalhes sobre a instalação do complemento e a configuração de parâmetros, consulte **Autoscaler de cluster do CCE**.
- O dimensionamento automático está disponível apenas para pools de nós de pagamento por uso, não para aqueles faturados por ano ou mês.
- Somente clusters de v1.19 ou posterior oferecem suporte a grupos de segurança personalizados.

Procedimento

Passo 1 Efetue logon no **console do CCE** .

Passo 2 Clique no nome do cluster para acessar o console do cluster. Escolha **Nodes** no painel de navegação e clique na guia **Node Pools** à direita.

Passo 3 No canto superior direito da página, clique em **Create Node Pool**.

Basic Settings

Tabela 4-2 Configurações básicas

Parâmetro	Descrição
Node Pool Name	Nome de um pool de nós. Por padrão, o nome está no formato de *Cluster name*-nodepool-*Random number*. Se não quiser usar o formato de nome padrão, você pode personalizar o nome.

Parâmetro	Descrição
Nodes	Número de nós a serem criados neste pool de nós.

Compute Settings

Você pode configurar o flavor e o sistema operacional de um servidor de nuvem, no qual suas aplicações em contêiner são executadas.

Tabela 4-3 Parâmetros de configuração

Parâmetro	Descrição
Billing Mode	Os seguintes modos de cobrança são suportados: ● Yearly/Monthly Você deve especificar a duração necessária se **Yearly/Monthly** estiver selecionado. Você pode escolher se deseja selecionar **Auto-renew** com base nos requisitos do local. Seu pedido será renovado automaticamente mensalmente ou anualmente, dependendo se você comprou por mês ou por ano. ● Pay-per-use Os recursos serão cobrados com base na duração do uso. Você pode provisionar ou excluir recursos a qualquer momento.
Node Type	Cluster do CCE: ● ECS (VM): os contêineres são executados em ECSs. ● ECS (físico): os contêineres são executados em servidores usando a arquitetura QingTian. ● BMS: os contêineres são executados em BMSs. Anexe discos locais ou discos EVS. Cluster do CCE Turbo: ● ECS (VM): os contêineres são executados em ECSs. Somente os ECSs que podem ser vinculados a várias NICs são suportados. ● ECS (físico): os contêineres são executados em servidores usando a arquitetura QingTian.
Container engine	Clusters do CCE suportam Docker e containerd em alguns cenários. ● Nós que executam CentOS, Ubuntu ou EulerOS 2.9 suportam containerd. Nós Arm executando EulerOS 2.5 ou EulerOS 2.8 não suportam containerd. ● Os clusters de rede da VPC da v1.23 e versões posteriores suportam containerd. Clusters de rede de túnel de v1.23.2-r0 e versões posteriores suportam containerd. ● Para um cluster do CCE Turbo, **Docker** e o **containerd** são suportados. Para obter detalhes, consulte **Mapeamento entre sistemas operacionais de nó e mecanismos de contêiner**.

Parâmetro	Descrição
Specifications	Selecione um flavor de nó com base nos requisitos de serviço. Os flavors de nó disponíveis variam dependendo das regiões ou AZs. Para obter detalhes, veja o console do CCE.
OS	Selecione um tipo de SO. Diferentes tipos de nós suportam diferentes sistemas operacionais. Para mais detalhes, consulte **Especificações do nó suportado**. **Public image**: selecione um SO para o nó. **Private image**: você pode usar imagens privadas. Para obter detalhes sobre como criar uma imagem privada, consulte **Criação de uma imagem personalizada de nó do CCE**.
Login Mode	● **Password** O nome do usuário padrão é **root**. Digite a senha para efetuar logon no nó e confirme a senha. Certifique-se de lembrar a senha, pois você precisará dela quando fizer logon no nó. ● **Key Pair** Selecione o par de chaves usado para efetuar logon no nó. Você pode selecionar uma chave compartilhada. Um par de chaves é usado para autenticação de identidade quando você entra remotamente em um nó. Se nenhum par de chaves estiver disponível, clique em **Create Key Pair**. Para obter detalhes sobre como criar um par de chaves, consulte **Criação de par de chaves**.

Storage Settings

Configure recursos de armazenamento em um nó para os contêineres em execução nele. Defina o tamanho do disco de acordo com os requisitos do site.

Tabela 4-4 Parâmetros de configuração

Parâmetro	Descrição
System Disk	Disco do sistema usado pelo sistema operacional do nó. O valor varia de 40 GB a 1.024 GB. O valor padrão é 50 GB. **Encryption**: a criptografia de disco do sistema protege seus dados. Os snapshots gerados a partir de discos criptografados e discos criados usando esses snapshots herdam automaticamente a configuração de criptografia. **Esta função está disponível apenas em determinadas regiões.** ● **Encryption** não está selecionada por padrão. ● Depois de selecionar **Encryption**, você pode selecionar uma chave existente na caixa de diálogo exibida. Se nenhuma chave estiver disponível, clique em **View Key List** e crie uma chave. Depois que a chave for criada, clique no ícone de atualização ao lado da caixa de texto **Encryption**.

Parâmetro	Descrição
Data Disk	**Pelo menos um disco de dados é necessário** para o tempo de execução do contêiner e o kubelet. **O disco de dados não pode ser excluído ou desinstalado. Caso contrário, o nó ficará indisponível.** ● Primeiro disco de dados: usado para o tempo de execução do contêiner e componentes do kubelet. O valor varia de 20 GB a 32.768 GB. O valor padrão é 100 GB. ● Outros discos de dados: você pode definir o tamanho do disco de dados para um valor que varia de 10 GB a 32.768 GB. O valor padrão é 100 GB. **NOTA** Se o flavor de nó for de I/O intensiva em disco ou ultra-alta, um disco de dados pode ser um disco local. Discos locais podem quebrar e não garantir a confiabilidade dos dados. Armazene seus dados de serviço em discos EVS, que são mais confiáveis do que os discos locais. **Configurações avançadas** Clique em **Expand** para configurar os seguintes parâmetros: ● **Alocação de espaço em disco de dados**: depois de selecionar **Set Container Engine Space**, você pode especificar a proporção do espaço para o mecanismo de contêiner, a imagem e o armazenamento temporário no disco de dados. O espaço do mecanismo de contêiner é usado para armazenar o diretório de trabalho, os dados da imagem do contêiner e os metadados da imagem para o tempo de execução do contêiner. O espaço restante do disco de dados é usado para arquivos de configuração, chaves e EmptyDir do pod. Para obter detalhes sobre como alocar espaço em disco de dados, consulte **Alocação de espaço em disco de dados**. ● **Encryption**: a criptografia do disco de dados protege os seus dados. Os snapshots gerados a partir de discos criptografados e discos criados usando esses snapshots herdam automaticamente a configuração de criptografia. **Esta função está disponível apenas em determinadas regiões.** – **Encryption** não está selecionada por padrão. – Depois de selecionar **Encryption**, você pode selecionar uma chave existente na caixa de diálogo exibida. Se nenhuma chave estiver disponível, clique em **View Key List** e crie uma chave. Depois que a chave for criada, clique no ícone de atualização ao lado da caixa de texto **Encryption**. **Adicionar vários discos de dados** Um máximo de quatro discos de dados podem ser adicionados. Por padrão, os discos brutos são criados sem qualquer processamento. Você também pode clicar em **Expand** e selecionar qualquer uma das seguintes opções: ● **Default**: por padrão, um disco bruto é criado sem qualquer processamento. ● **Mount Disk**: o disco de dados é anexado a um diretório especificado.

Parâmetro	Descrição
	● **Use as PV**: aplicável a cenários em que há um requisito de alto desempenho em PVs. O rótulo **node.kubernetes.io/local-storage-persistent** é adicionado ao nó com o PV configurado. O valor é **linear** ou **striped**. ● **Use as ephemeral volume**: aplicável a cenários em que há uma exigência de alto desempenho em EmptyDir. **NOTA** ● Os PVs locais são suportados apenas quando a versão do cluster é v1.21.2-r0 ou posterior e a versão do complemento everest é 2.1.23 ou posterior. Recomenda-se a versão 2.1.23 ou posterior. ● Os EVs locais são suportados apenas quando a versão do cluster é v1.21.2-r0 ou posterior e a versão do complemento everest é 2.1.23 ou posterior. **Volumes persistentes locais** e **EVs locais** suportam os seguintes modos de gravação: ● **Linear**: um volume lógico linear integra um ou mais volumes físicos. Os dados são gravados no próximo volume físico quando o anterior é usado. ● **Striped**: um volume lógico distribuído distribui dados em blocos do mesmo tamanho e os armazena em vários volumes físicos em sequência, permitindo que os dados sejam lidos e gravados simultaneamente. Um pool de armazenamento que consiste em volumes distribuídos não pode ser dimensionado. Essa opção só pode ser selecionada quando existirem vários volumes.

Network Settings

Configure os recursos de rede para permitir o acesso de nós e aplicações em contêiner.

Tabela 4-5 Parâmetros de configuração

Parâmetro	Descrição
Node Subnet	A sub-rede de nó selecionada durante a criação do cluster é usada por padrão. Você pode escolher outra sub-rede em vez disso.
Node IP	A alocação aleatória é suportada.
Associate Security Group	Grupo de segurança usado pelos nós criados no pool de nós. Um máximo de 5 grupos de segurança podem ser selecionados. Quando um cluster é criado, um grupo de segurança de nó chamado **{*Cluster name*}-cce-node-{*Random ID*}** é criado e usado por padrão. O tráfego precisa passar por determinadas portas no grupo de segurança do nó para garantir as comunicações do nó. Assegure-se de que você habilitou estas portas se você selecionar um outro grupo de segurança. Para obter detalhes, consulte **Configuração das regras do grupo de segurança do cluster**.

Advanced Settings

Configure recursos avançados de nó, como rótulos, manchas e comando de inicialização.

Tabela 4-6 Parâmetros de configuração avançadas

Parâmetro	Descrição
Kubernetes Label	Um par de chave e valor adicionado a um objeto do Kubernetes (como um pod). Um máximo de 20 rótulos podem ser adicionados. Os rótulos podem ser usados para distinguir nós. Com as configurações de afinidade da carga de trabalho, os pods de contêiner podem ser agendados para um nó especificado. Para obter mais informações, consulte **Rótulos e seletores**.
Resource Tag	Você pode adicionar tags de recursos para classificar recursos. Você pode criar **tagspredefinidas** no console do TMS. As tags predefinidas estão disponíveis para todos os recursos que suportam tags. Você pode usar tags predefinidas para melhorar a criação de tags e a eficiência da migração de recursos. Para obter detalhes, consulte **Criação de tags predefinidas**. O CCE criará automaticamente a tag "CCE-Dynamic-Provisioning-Node=*node id*".
Taint	Este parâmetro é deixado em branco por padrão. Você pode adicionar manchas para configurar a antiafinidade para o nó. Um máximo de 20 manchas são permitidas para cada nó. Cada mancha contém os seguintes parâmetros: ● **Key**: uma chave deve conter de 1 a 63 caracteres, começando com uma letra ou dígito. Apenas letras, dígitos, hifens (-), sublinhados (_) e pontos (.) são permitidos. Um nome de subdomínio do DNS pode ser usado como prefixo de uma chave. ● **Value**: um valor deve começar com uma letra ou dígito e pode conter no máximo 63 caracteres, incluindo letras, dígitos, hifens (-) e pontos (.). ● **Effect**: as opções disponíveis são **NoSchedule**, **PreferNoSchedule** e **NoExecute**. Para mais detalhes, consulte **Gerenciamento de manchas de nó**. **NOTA** Para um cluster v1.19 ou anterior, a carga de trabalho pode ter sido agendada para um nó antes de a mancha ser adicionada. Para evitar tal situação, selecione um cluster v1.19 ou posterior.
Max. Pods	Número máximo de pods que podem ser executados no nó, incluindo os pods padrão do sistema. Intervalo de valor: 16 a 256 Esse limite impede que o nó seja sobrecarregado com pods. Esse número também é decidido por outros fatores. Para mais detalhes, consulte **Número máximo de pods que podem ser criados em um nó**.

Parâmetro	Descrição
ECS Group	Um grupo de ECS agrupa logicamente os ECS. Os ECSs do mesmo grupo de ECS cumprem a mesma política associada ao grupo de ECS. **Anti-affinity**: os ECSs em um grupo de ECS são implementados em hosts físicos diferentes para melhorar a confiabilidade do serviço. Selecione um grupo de ECS existente ou clique em **Add ECS Group** para criar um. Depois que o grupo de ECS for criado, clique no botão de atualizar.
Pre-installation Command	Insira comandos. Um máximo de 1.000 caracteres são permitidos. O script será executado antes da instalação do software Kubernetes. Observe que, se o script estiver incorreto, o software Kubernetes pode falhar ao ser instalado.
Post-installation Command	Insira comandos. Um máximo de 1.000 caracteres são permitidos. O script será executado após a instalação do software Kubernetes e não afetará a instalação. **NOTA** Não execute o comando **reboot** no script de pós-instalação para reiniciar o sistema imediatamente. Para reiniciar o sistema, execute o comando **shutdown -r 1** para reiniciar com um atraso de um minuto.
Agency	Uma agência é criada pelo administrador do locatário no console do IAM. Ao criar uma agência, você pode compartilhar seus recursos de servidor em nuvem com outra conta ou confiar a uma pessoa ou equipe mais profissional para gerenciar seus recursos. Se nenhuma agência estiver disponível, clique em **Create Agency** à direita para criar uma.

Passo 4 Clique em **Next: Auto Scaling Configuration**.

Tabela 4-7 Configurações de dimensionamento automático

Parâmetro	Descrição
Auto Scaling	Por padrão, essa função está desabilitada. O dimensionamento automático está disponível apenas para pools de nós de pagamento por uso, não para aqueles cobrados anualmente ou mensalmente. Para habilitar o dimensionamento automático, instale o complemento **autoscaler**. Depois que você habilitar o dimensionamento automático ativando  , os nós no pool de nós serão criados ou excluídos automaticamente com base nas cargas do cluster. ● **Maximum Nodes** e **Minimum Nodes**: você pode definir o número máximo e mínimo de nós para garantir que o número de nós a serem escalados esteja dentro de um intervalo adequado. ● **Priority**: defina esse parâmetro com base nos requisitos do serviço. Um valor maior indica uma prioridade mais alta. Por exemplo, se esse parâmetro for definido como **1** e **4** respectivamente para os pools de nós A e B, B terá uma prioridade maior que A. Se as prioridades de vários pools de nós forem definidas com o mesmo valor, por exemplo, **2**, os pools de nós não serão priorizados e o sistema executará o dimensionamento com base no princípio do desperdício mínimo de recursos. **NOTA** O CCE seleciona um pool de nós para dimensionamento automático com base nas seguintes políticas: 1. O CCE usa algoritmos para determinar se um pool de nós atende às condições para permitir o agendamento de um pod em estado pendente, incluindo se os recursos de nó são maiores do que os solicitados pelo pod e se o nodeSelect, nodeAffinity e taints atendem às condições. Além disso, os pools de nós que não conseguem ser escalados (devido a recursos insuficientes ou outros motivos) e ainda estão no intervalo de resfriamento de 15 minutos são filtrados. 2. Se vários pools de nós atenderem aos requisitos de dimensionamento, o sistema verificará a prioridade de cada pool de nós e selecionará o pool de nós com a prioridade mais alta para dimensionamento. O valor varia de 0 a 100 e a prioridade padrão é 0. O valor 100 indica a prioridade mais alta e o valor 0 indica a prioridade mais baixa. 3. Se vários pools de nós tiverem a mesma prioridade ou se nenhuma prioridade estiver configurada para eles, o sistema selecionará o pool de nós que consumirá menos recursos com base no flavor de VM configurada. 4. Se os flavors de VM de vários pools de nós forem as mesmas, mas os pools de nós forem implementados em AZs diferentes, o sistema selecionará aleatoriamente um pool de nós para acionar o dimensionamento. ● **Cooldown Period**: insira um período, em minutos. Esse campo indica o período durante o qual os nós adicionados no pool de nós atual não podem ser dimensionados. Os intervalos de resfriamento de dimensionamento podem ser configurados nas configurações do pool de nós e nas configurações do **complemento do autoscaler**.

Parâmetro	Descrição
	Scale-in cooling interval configured in a node pool Esse intervalo indica o período durante o qual os nós adicionados ao pool de nós atual após uma operação de expansão não podem ser excluídos. Essa configuração entra em vigor em todo o pool de nós. **Scale-in cooling interval configured in the autoscaler add-on** O intervalo após uma expansão indica o período durante o qual todo o cluster não pode ser dimensionado após o complemento autoscaler disparar a expansão (devido aos pods, métricas e políticas de dimensionamento não programáveis). Essa configuração entra em vigor em todo o cluster. O intervalo depois que um nó é excluído indica o período durante o qual o cluster não pode ser dimensionado após o complemento autoscaler disparar a redução. Essa configuração entra em vigor em todo o cluster. O intervalo após uma falha de redução indica o período durante o qual o cluster não pode ser dimensionado após o complemento autoscaler disparar a redução. Essa configuração entra em vigor em todo o cluster. **NOTA** Não armazene dados importantes em nós em um pool de nós, pois os nós podem ser excluídos após o dimensionamento. Os dados nos nós excluídos não podem ser restaurados.

Passo 5 Clique em **Next: Confirm**. Certifique-se de que leu e entendeu a **Declaração do serviço de gerenciamento de imagens**.

Passo 6 Clique em **Submit**.

----Fim

4.3 Gerenciamento de um pool de nós

4.3.1 Atualização de um pool de nós

Restrições

- Ao editar as tags de recurso do pool de nós. A configuração modificada só tem efeito para novos nós. Para sincronizar a configuração com os nós existentes, redefina manualmente os nós existentes.
- Atualizações de rótulos e manchas do kubernetes são automaticamente sincronizadas com os nós existentes. Você não precisa redefinir os nós.

Atualizar um pool de nós

Passo 1 Efetue logon no console do CCE.

Passo 2 Clique no nome do cluster para acessar o console do cluster. Escolha **Nodes** no painel de navegação e clique na guia **Node Pools** à direita.

Passo 3 Clique em **Update** ao lado do nome do pool de nós que você editará. Configure os parâmetros na página **Update Node Pool**.

Basic Settings

Tabela 4-8 Configurações básicas

Parâmetro	Descrição
Node pool name	Nome do pool de nós.

Advanced Settings

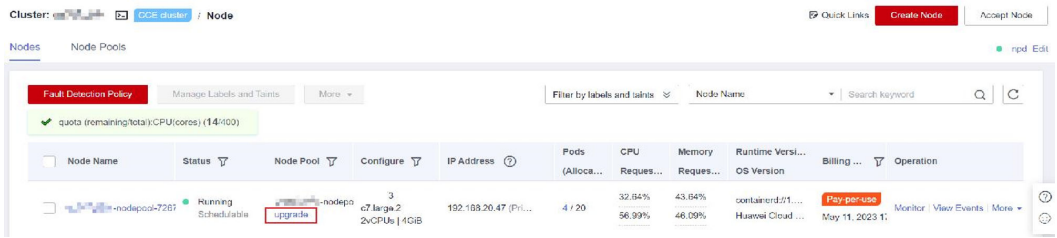
Tabela 4-9 Configurações avançadas

Parâmetro	Descrição
Kubernetes label	Um rótulo de Kubernetes é um par chave-valor adicionado a um objeto de Kubernetes (como um pod). Depois de especificar um rótulo, clique em **Add**. Um máximo de 20 rótulos podem ser adicionados. Os rótulos podem ser usados para distinguir nós. Com as configurações de afinidade da carga de trabalho, os pods de contêiner podem ser agendados para um nó especificado. Para obter mais informações, consulte **Rótulos e seletores**. **NOTA** Depois que um **Kubernetes label** é modificado, os nós de inventário no pool de nós são atualizados de forma síncrona.
Resource tag	Você pode adicionar tags de recursos para classificar recursos. Você pode criar **tags predefinidas** no console do TMS. As tags predefinidas estão disponíveis para todos os recursos que suportam tags. Você pode usar tags predefinidas para melhorar a criação de tags e a eficiência da migração de recursos. Para obter detalhes, consulte **Criação de tags predefinidas**. O CCE criará automaticamente a tag "CCE-Dynamic-Provisioning-Node=*node id*". **NOTA** Depois que uma tag de recurso é modificada, a modificação entra em vigor automaticamente nos nós recém-adicionados. Para nós existentes, redefina manualmente os nós para que a modificação entre em vigor.

Parâmetro	Descrição
Taint	Este campo é deixado em branco por padrão. Você pode adicionar manchas para configurar a antiafinidade do nó. Um máximo de 20 manchas são permitidas para cada nó. Cada mancha contém os seguintes parâmetros: ● **Key**: uma chave deve conter de 1 a 63 caracteres, começando com uma letra ou dígito. Apenas letras, dígitos, hifens (-), sublinhados (_) e pontos (.) são permitidos. Um nome de subdomínio do DNS pode ser usado como prefixo de uma chave. ● **Value**: um valor deve começar com uma letra ou dígito e pode conter no máximo 63 caracteres, incluindo letras, dígitos, hifens (-) e pontos (.). ● **Effect**: as opções disponíveis são **NoSchedule**, **PreferNoSchedule** e **NoExecute**. Para mais detalhes, consulte **Gerenciamento de manchas de nó**. **NOTA** Depois que uma **mancha** é modificada, os nós existentes no pool de nós são atualizados de forma síncrona.
Edit key pair	Somente os pools de nós que usam pares de chaves para logon suportam a edição de pares de chaves. Você pode selecionar outro par de chaves. **NOTA** O par de chaves editado automaticamente entra em vigor nos nós recém-adicionados. Para nós existentes, redefina manualmente os nós para que a modificação entre em vigor.

Passo 4 Após a configuração, clique em **OK**.

Depois que os parâmetros do pool de nós forem atualizados, vá para a página **Nodes** para verificar se o nó ao qual o pool de nós pertence está atualizado. Você pode redefinir o nó para sincronizar as atualizações de configuração para o pool de nós.



----Fim

4.3.2 Atualização de uma configuração de AS

O Auto Scaling (AS) permite o escalonamento elástico de nós em um pool de nós com base em políticas de escalonamento. Sem essa função, você precisa ajustar manualmente o número de nós em um pool de nós.

Restrições

Para habilitar o AS, o complemento **autoscaler** deve ser instalado no cluster de destino.

Procedimento

Passo 1 Efetue logon no console do CCE.

Passo 2 Clique no nome do cluster para acessar o console do cluster. Escolha **Nodes** no painel de navegação e clique na guia **Node Pools** à direita.

Passo 3 Localize a linha que contém o pool de nós de destino. Escolha **More** na coluna **Operation** e escolha **Configure Auto Scaling**. Na página exibida, configure os parâmetros de AS.

Tabela 4-10 Configuração de AS

Parâmetro	Descrição
Auto Scaling	Por padrão, essa função está desabilitada. Depois que você habilitar AS clicando em  , os nós no pool de nós serão criados ou excluídos automaticamente com base nas políticas de dimensionamento. Para garantir a execução correta do AS, instale o **autoscaler**.
Max. Nodes and Min. Nodes	O número máximo ou mínimo de nós assegurados em um pool de nós para garantir que os nós em um pool de nós sejam dimensionados dentro de um intervalo adequado.
Node Pool Priority	A prioridade de um pool de nós para uma expansão. Um valor maior indica uma prioridade mais alta. Por exemplo, o pool de nós com prioridade **4** é expandido antes do conjunto com prioridade **1**. Se as prioridades de vários pools de nós forem definidas com o mesmo valor, esses pools de nós não serão priorizados e serão expandidos seguindo a regra de maximizar a utilização de recursos. Depois que a prioridade é alterada, a modificação entra em vigor dentro de 1 minuto.
Cooldown Period	Um período, em minutos, durante o qual os nós adicionados no pool de nós atual não podem ser reduzidos.

Passo 4 Clique em **OK**.

----Fim

4.3.3 Configuração de um pool de nós

Restrições

O pool de nós padrão DefaultPool não suporta as seguintes operações de gestão.

Gerenciamento da configuração

O CCE permite que você personalize altamente as configurações de parâmetros do Kubernetes em componentes principais em um cluster. Para obter mais informações, consulte **kubelet**.

Esta função é suportada apenas em clusters de **v1.15 e posterior**. Ela não é exibida para versões anteriores à v1.15.

Passo 1 Efetue logon no console do CCE.

Passo 2 Clique no nome do cluster para acessar o console do cluster. Escolha **Nodes** no painel de navegação e clique na guia **Node Pools** à direita.

Passo 3 Escolha **More** > **Manage** na coluna **Operation** do pool de nós de destino.

Passo 4 Na página **Manage Components** à direita, altere os valores dos seguintes parâmetros do Kubernetes:

Tabela 4-11 kubelet

Parâmetro	Descrição	Valor padrão	Modificação	Observações
cpu-manager-policy	Configuração da política de gerenciamento de CPU. Para mais detalhes, consulte **Agendamento de CPU**. ● **none**: desativa pods de ocupar exclusivamente CPUs. Selecione esse valor se quiser um grande pool de núcleos de CPU compartilháveis. ● **static**: permite que pods ocupem exclusivamente CPUs. Selecione esse valor se sua carga de trabalho for sensível à latência no cache e no agendamento da CPU. ● **enhanced-static**: permite que pods intermitentes usem preferencialmente núcleos de CPU. Selecione esse valor se sua carga de trabalho tiver uma enorme diferença de pico-vale e estiver no estado de vale a maior parte do tempo.	none	Nenhuma	Nenhuma
kube-api-qps	Consulta por segundo (QPS) para comunicação com kube-apiserver.	100	Nenhuma	Nenhuma

Parâmetro	Descrição	Valor padrão	Modificação	Observações
	Intermitência para usar enquanto conversa com kube-apiserver.	100	Nenhuma	Nenhuma
max-pods	Número máximo de pods gerenciados pelo kubelet.	• Para um cluster do CCE, o número máximo de pods é determinado com base **no número máximo de pods em um nó**. • Para um cluster do CCE Turbo, o número máximo de pods é determinado com base **no número de NICs em um nó do cluster do CCE Turbo**.	Nenhuma	Nenhuma
pod-pids-limit	Número limitado de PIDs no Kubernetes	-1	Nenhuma	Nenhuma
	Se usar o endereço IP local como o ClusterDNS do nó.	false	Nenhuma	Nenhuma
event-qps	Limite do QPS para criação de eventos	5	Nenhuma	Nenhuma

Parâmetro	Descrição	Valor padrão	Modificação	Observações
allowed-unsafe-sysctls	Configuração insegura do sistema permitida. A partir da **v1.17.17**, o CCE habilita as políticas de segurança do pod para o kube-apiserver. Adicione as configurações correspondentes a **allowedUnsafeSysctls** de uma política de segurança de pod para que a política entre em vigor. (Esta configuração não é necessária para clusters anteriores à v1.17.17.) Para mais detalhes, consulte **Exemplo de ativação de Sysctls inseguros na política de segurança do pod**.	[]	Nenhuma	Nenhuma
over-subscription-resource	Se deve ser ativada a sobreassinatura de nó. Se esse parâmetro for definido como **true**, a sobreassinatura de nó será ativada. Para mais detalhes, consulte **Excesso de assinaturas de recursos dinâmicos**.	true	Nenhuma	Nenhuma
colocation	Se ativar a implementação híbrida em nós. Se esse parâmetro for definido como **true**, a implementação híbrida será ativada nos nós. Para mais detalhes, consulte **Excesso de assinaturas de recursos dinâmicos**.	true	Nenhuma	Nenhuma

Parâmetro	Descrição	Valor padrão	Modificação	Observações
kube-reserved-mem system-reserved-mem	Memória de nó reservada.	Depende das especificações do nó. Para mais detalhes, consulte **Política de reserva de recursos de nó**.	Nenhuma	A soma de **kube-reserved-mem** e **system-reserved-mem** é menor que a metade da memória.
topology-manager-policy	Defina a política de gerenciamento de topologia. Os valores válidos são os seguintes: ● **restricted**: o kubelet aceita apenas pods que alcançam o alinhamento NUMA ideal nos recursos solicitados. ● **best-effort**: o kubelet seleciona preferencialmente pods que implementam o alinhamento NUMA nos recursos da CPU e do dispositivo. ● **none** (padrão): a política de gerenciamento de topologia está desabilitada. ● **single-numa-node**: o kubelet permite apenas pods alinhados ao mesmo nó NUMA em termos de recursos de CPU e dispositivo.	none	Nenhuma	**AVISO** Modificar **topology-manager-policy** e **topology-manager-scope** reiniciará o kubelet e a alocação de recursos dos pods será recalculada com base na política modificada. Nesse caso, a execução de pods pode reiniciar ou até mesmo deixar de receber quaisquer recursos.

Parâmetro	Descrição	Valor padrão	Modificação	Observações
topology-manager-scope	Defina a granularidade de alinhamento de recursos da política de gerenciamento de topologia. Os valores válidos são os seguintes: ● **container** (padrão) ● **pod**	container		
resolv-conf	Arquivo de configuração de resolução de DNS especificado pelo contêiner	O valor padrão é null.	Nenhuma	Nenhuma

Tabela 4-12 kube-proxy

Parâmetro	Descrição	Valor padrão	Modificação
conntrack-min	Número máximo de entradas de rastreamento de conexão Para obter o valor, execute o seguinte comando: `sysctl -w net.nf_conntrack_max`	131072	Nenhuma
conntrack-tcp-timeout-close-wait	Tempo de espera de uma conexão TCP fechada Para obter o valor, execute o seguinte comando: `sysctl -w net.netfilter.nf_conntrack_tcp_timeout_close_wait`	1h0m0s	Nenhuma

Tabela 4-13 Componentes de rede (disponível apenas para clusters do CCE Turbo)

Parâmetro	Descrição	Valor padrão	Modificação
nic-threshold	Limite baixo do número de ENIs vinculadas: limite elevado do número de ENIs vinculadas	Padrão: 0:0	**NOTA** Esse parâmetro está sendo descartado. Use os parâmetros dinâmicos de pré-vinculação dos outras quatro ENIs.
nic-minimum-target	Número mínimo de ENIs vinculadas aos nós no pool de nós	Padrão: 10	Nenhuma
nic-maximum-target	Número máximo de ENIs pré-vinculadas a um nó no nível do pool	Padrão: 0	Nenhuma
nic-warm-target	Número de ENIs pré-vinculadas a um nó no nível do pool de nós	Padrão: 2	Nenhuma
nic-max-above-warm-target	Recupere o número de ENIs pré-vinculadas a um nó no nível do pool de nós	Padrão: 2	Nenhuma

Tabela 4-14 Grupo de segurança de pods em um pool de nós (disponível apenas para clusters do CCE Turbo)

Parâmetro	Descrição	Valor padrão	Modificação
security_groups_for_nodepool	● Grupo de segurança padrão usado por pods em um pool de nós. Você pode inserir o ID do grupo de segurança. Se esse parâmetro não for definido, o grupo de segurança padrão da rede de contêineres de cluster será usado. Um máximo de cinco IDs de grupo de segurança podem ser especificados ao mesmo tempo, separados por ponto e vírgula (;). ● A prioridade do grupo de segurança é menor do que a do grupo de segurança configurado para **Grupos de segurança**.	Nenhuma	Nenhuma

Tabela 4-15 Docker (disponível apenas para grupos de nós que usam Docker)

Parâmetro	Descrição	Valor padrão	Modificação
native-umask	`--exec-opt native.umask	normal	Não pode ser alterado.
docker-base-size	`--storage-opts dm.basesize	0	Não pode ser alterado.
insecure-registry	Endereço de um registro de imagem insegura	false	Não pode ser alterado.
limitcore	Tamanho máximo de um arquivo principal em um contêiner. A unidade é byte. Se não especificado, o valor é **infinity**.	5368709120	Nenhuma

Parâmetro	Descrição	Valor padrão	Modificação
default-ulimit-nofile	Limite no número de identificadores em um contêiner	{soft}:{hard}	O valor não pode exceder o valor do parâmetro do kernel **nr_open** e não pode ser um número negativo. Você pode executar o seguinte comando para obter o parâmetro do kernel **nr_open**: `sysctl -a | grep nr_open`

Tabela 4-16 containerd (disponível somente para pools de nós que usam containerd)

Parâmetro	Descrição	Valor padrão	Modificação
devmapper-base-size	Espaço de dados disponível de um único contêiner	Nenhum	Não pode ser alterado.
limitcore	Tamanho máximo de um arquivo principal em um contêiner. A unidade é byte. Se não especificado, o valor é **infinity**.	5368709120	Nenhuma
default-ulimit-nofile	Limite no número de identificadores em um contêiner	1048576	O valor não pode exceder o valor do parâmetro do kernel **nr_open** e não pode ser um número negativo. Você pode executar o seguinte comando para obter o parâmetro do kernel **nr_open**: `sysctl -a | grep nr_open`

Passo 5 Clique em **OK**.

----Fim

4.3.4 Cópia de um pool de nós

Você pode copiar a configuração de um pool de nós existente para criar um novo pool de nós no console do CCE.

Passo 1 Efetue logon no console do CCE.

Passo 2 Clique no nome do cluster para acessar o console do cluster. Escolha **Nodes** no painel de navegação e clique na guia **Node Pools** à direita.

Passo 3 Escolha **More > Copy** na coluna **Operation** do pool de nós de destino.

Passo 4 As configurações do pool de nós selecionado são replicadas para a página **Clone Node Pool**. Você pode editar as configurações conforme necessário. Para obter detalhes sobre itens de configuração, consulte **Criação de um pool de nós**. Após confirmar a configuração, clique em **Next: Confirm**.

Passo 5 Na página **Confirm**, confirme a configuração do pool de nós e clique em **Submit**. Em seguida, um novo pool de nós é criado com base na configuração editada.

----Fim

4.3.5 Sincronização de pools de nós

Depois que a configuração de um pool de nós é atualizada, algumas configurações não podem ser sincronizadas automaticamente para os nós existentes. Você pode sincronizar manualmente as configurações desses nós.

AVISO

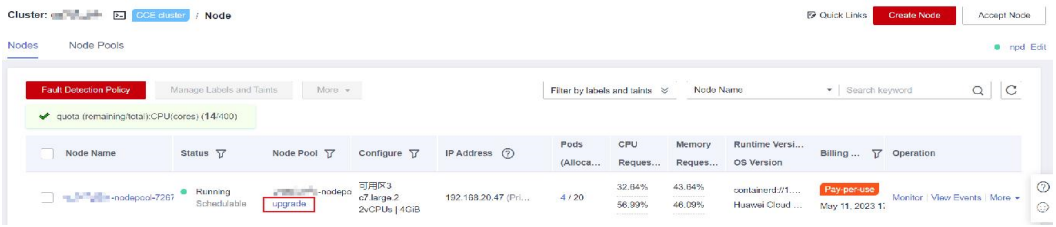
- Não exclua ou reinicialize nós durante a sincronização em lote. Caso contrário, a sincronização da configuração do pool de nós pode falhar.
- Essa operação envolve a reinicialização de nós. **As cargas de trabalho em execução em um nó podem ser interrompidas devido à implementação autônoma ou a recursos programáveis insuficientes.** Avalie os riscos de upgrade e execute a atualização fora do horário de pico. Como alternativa, **especifique um orçamento de interrupção para seus principais aplicações** para garantir a disponibilidade dessas aplicações durante a atualização.
- Durante a sincronização de configuração para nós existentes, os nós serão reiniciados e os discos do sistema e os discos de dados serão limpos. Faça backup de dados importantes antes da sincronização.
- Apenas alguns parâmetros do pool de nós podem ser sincronizados reinicializando nós. As restrições são as seguintes:
  - Ao editar as tags de recurso do pool de nós. A configuração modificada só tem efeito para novos nós. Para sincronizar a configuração com os nós existentes, redefina manualmente os nós existentes.
  - Atualizações de rótulos e manchas do kubernetes são automaticamente sincronizadas com os nós existentes. Você não precisa redefinir os nós.

Sincronizar um nó único

Passo 1 Efetue logon no console do CCE.

Passo 2 Clique no nome do cluster para acessar o console do cluster. Escolha **Nodes** no painel de navegação e clique na guia **Nodes** à direita.

Passo 3 Localize **upgrade** na coluna **Node Pool** dos nós existentes no pool de nós.



Passo 4 Clique em **Update**. Na caixa de diálogo exibida, confirme se deseja reiniciar o nó imediatamente.

----Fim

Sincronização em lotes

Passo 1 Efetue logon no console do CCE.

Passo 2 Clique no nome do cluster para acessar o console do cluster. Escolha **Nodes** no painel de navegação e clique na guia **Node Pools** à direita.

Passo 3 Escolha **More > Synchronize** na coluna **Operation** do pool de nós de destino.

Passo 4 Na janela **Batch synchronization**, configure os parâmetros.

- **OS**: mostra a imagem da versão de destino. Você não precisa configurar esse parâmetro.
- **Synchronization Policy**: **Node Reset** é suportada.
- **Max. Nodes for Batch Synchronize**: número máximo de nós que estarão indisponíveis durante a sincronização do nó. Os nós estarão indisponíveis durante a sincronização reiniciando os nós. Configure adequadamente esse parâmetro para evitar falhas de agendamento de pods causadas por muitos nós indisponíveis no cluster.
- **Node List**: selecione os nós que exigem a sincronização de configurações de pool de nós.

Passo 5 Clique em **OK**.

----Fim

4.3.6 Atualização de um sistema operacional

Quando o CCE libera uma nova imagem do sistema operacional, os nós existentes não podem ser atualizados automaticamente. Você pode atualizá-los manualmente em lotes.

AVISO

Esta seção descreve como atualizar um sistema operacional redefinindo o nó de destino. **As cargas de trabalho em execução em um nó podem ser interrompidas devido à implementação autônoma ou a recursos agendáveis insuficientes.** Avalie os riscos de atualização e execute a atualização fora do horário de pico. Como alternativa, **especifique um orçamento de interrupção para seus principais aplicações** para garantir a disponibilidade dessas aplicações durante a atualização.

Restrições

- Os nós que executam imagens privadas não podem ser atualizados.
- Problemas de compatibilidade podem ocorrer quando o SO do nó de uma versão anterior é atualizado. Nesse caso, redefina manualmente o nó para a atualização do sistema operacional.

Procedimento para pools de nós padrão

Passo 1 Efetue logon no console do CCE.

Passo 2 Clique no nome do cluster para acessar o console do cluster. Escolha **Nodes** no painel de navegação e clique na guia **Node Pools** à direita.

Passo 3 Clique em **Upgrade** ao lado do pool de nós padrão.

Passo 4 Na janela **Operating System Upgrade** exibida, configure os parâmetros.

- **Target Operating System**: mostra a imagem da versão de destino. Você não precisa configurar esse parâmetro.
- **Upgrade Policy**: **Node Reset** é suportada.
- **Max. Nodes for Batch Upgrade**: número máximo de nós que estarão indisponíveis durante a atualização do nó. Os nós estarão indisponíveis durante a atualização, reiniciando os nós. Configure adequadamente esse parâmetro para evitar falhas de agendamento de pods causadas por muitos nós indisponíveis no cluster.
- **View Node**: selecione os nós a serem atualizados.
- Login Mode:
  - **Password**

    O nome do usuário padrão é **root**. Digite a senha para efetuar logon no nó e confirme a senha.

    Certifique-se de lembrar a senha, pois você precisará dela quando fizer logon no nó.
  - **Key Pair**

    Selecione o par de chaves usado para efetuar logon no nó. Você pode selecionar uma chave compartilhada.

    Um par de chaves é usado para autenticação de identidade quando você entra remotamente em um nó. Se nenhum par de chaves estiver disponível, clique em **Create Key Pair**. Para obter detalhes sobre como criar um par de chaves, consulte **Criação de par de chaves**.
- **Pre-installation Command**: insira um máximo de 1.000 caracteres.

  O script será executado antes da instalação do software Kubernetes. Observe que, se o script estiver incorreto, o software Kubernetes pode falhar ao ser instalado.

- **Post-installation Command**: insira um máximo de 1.000 caracteres.

  O script será executado após a instalação do software Kubernetes e não afetará a instalação.

Passo 5 Clique em **OK**.

----Fim

Procedimento para pools de nós não padrão

Passo 1 Efetue logon no console do CCE.

Passo 2 Clique no nome do cluster para acessar o console do cluster. Escolha **Nodes** no painel de navegação e clique na guia **Node Pools** à direita.

Passo 3 Escolha **More > Synchronize** na coluna **Operation** do pool de nós de destino.

Passo 4 Na janela **Batch synchronization**, configure os parâmetros.

- **OS**: mostra a imagem da versão de destino. Você não precisa configurar esse parâmetro.
- **Synchronization Policy**: **Node Reset** é suportada.
- **Max. Nodes for Batch Synchronize**: número máximo de nós que estarão indisponíveis durante a sincronização do nó. Os nós estarão indisponíveis durante a sincronização reiniciando os nós. Configure adequadamente esse parâmetro para evitar falhas de agendamento de pods causadas por muitos nós indisponíveis no cluster.
- **Node List**: selecione os nós que exigem a sincronização de configurações de pool de nós.

Passo 5 Clique em **OK**.

----Fim

4.3.7 Migração de um nó

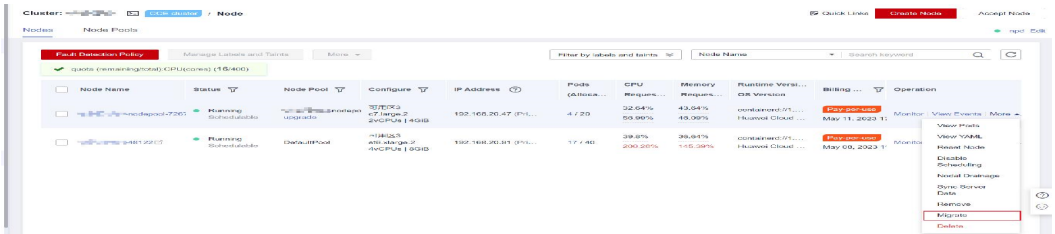
Os nós em um pool de nós podem ser migrados. Atualmente, os nós em um pool de nós podem ser migrados somente para o pool de nós padrão (pool padrão) no mesmo cluster.

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster.

Passo 2 No painel de navegação, escolha **Nodes** e alterne para a página de guia **Node Pools**.

Passo 3 Clique em **View Node** na coluna **Operation** do pool de nós a ser migrado.

Passo 4 Clique em **More** > **Migrate** na coluna **Operation** do nó de destino para migrar o nó.



Passo 5 Na janela **Migrate Node** exibida, confirme as informações.

NOTA

A migração não tem impacto nas tags de recursos originais, nos rótulos do Kubernetes e nas manchas do nó.

----Fim

4.3.8 Exclusão de um pool de nós

A exclusão de um pool de nós excluirá os nós no pool. Os pods nesses nós serão migrados automaticamente para nós disponíveis em outros pools de nós.

Restrições

- Um pool de nós de faturamento anual/mensal não pode ser excluído antes que todos os nós nele sejam removidos primeiro.
- Excluir um nó causará perda de dados de PVC/PV para os **PVs locais** associado ao nó. Esses PVCs e PVs não podem ser restaurados ou usados novamente. Nesse cenário, o pod que usa o PV local é despejado do nó. Um novo pod é criado e permanece no estado pendente. Isso ocorre porque a PVC usada pelo pod tem um rótulo de nó, devido ao qual o pod não pode ser programado.

Precauções

- A exclusão de um pool de nós excluirá todos os nós no pool de nós. Faça backup dos dados em tempo hábil para evitar a perda de dados.
- A exclusão de um nó levará à migração de pods, o que pode afetar os serviços. Portanto, realize esta operação fora dos horários de pico. Se os pods no pool de nós tiverem um seletor de nó específico e nenhum dos outros nós no cluster satisfizer o seletor de nó, os pods se tornarão não escalonáveis.
- Ao excluir um pool de nós, o sistema define todos os nós no pool de nós atual para o estado não-agendável.

Procedimento

Passo 1 Efetue logon no console do CCE.

Passo 2 Clique no nome do cluster para acessar o console do cluster. Escolha **Nodes** no painel de navegação e clique na guia **Node Pools** à direita.

Passo 3 Escolha **More > Delete** na coluna **Operation** do pool de nós de destino.

Passo 4 Leia as precauções na caixa de diálogo **Delete Node Pool**.

Passo 5 Na caixa de texto, clique em **Yes** para confirmar que você deseja continuar a exclusão.

----Fim

5 Cargas de trabalho

5.1 Visão geral

Uma carga de trabalho é uma aplicação executada no Kubernetes. Não importa quantos componentes existam em sua carga de trabalho, você pode executá-la em um grupo de pods do Kubernetes. Uma carga de trabalho é um modelo abstrato de um grupo de pods no Kubernetes. As cargas de trabalho classificadas no Kubernetes incluem Implementações, StatefulSets, DaemonSets, tarefas e tarefas cronometradas.

O CCE fornece implementação e gerenciamento de contêiner nativos do Kubernetes e suporta o gerenciamento do ciclo de vida de cargas de trabalho de contêiner, incluindo criação, configuração, monitoramento, escalonamento automático, atualização, desinstalação, descoberta de serviços e balanceamento de carga.

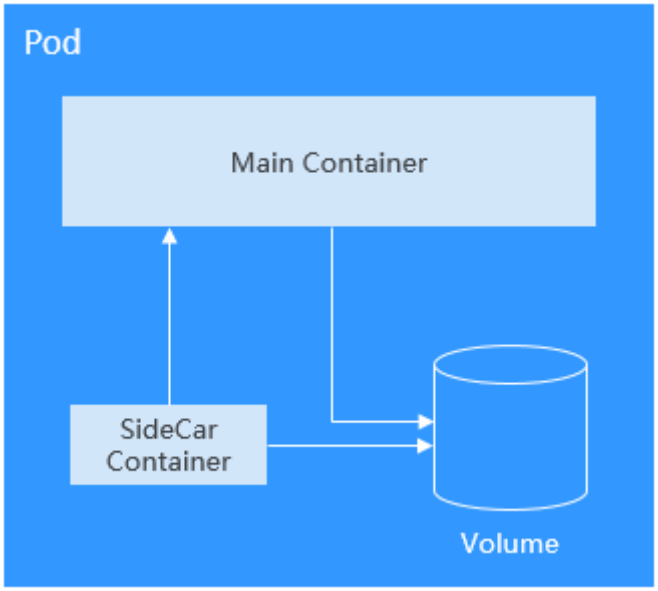
Visão geral do pod

Um pod é a unidade menor e mais simples no modelo de objeto do Kubernetes que você cria ou implementa. Um pod é um grupo de um ou mais contêineres, com recursos de rede e armazenamento compartilhados, e uma especificação de como executar os contêineres. Cada pod tem um endereço IP separado.

Os pods podem ser usados de uma das seguintes maneiras:

- Um pod executa apenas um contêiner. Esse é o uso mais comum de pods no Kubernetes. Você pode considerar um pod como um contêiner, mas o Kubernetes gerencia diretamente pods em vez de contêineres.
- Um pod executa vários contêineres que precisam ser fortemente acoplados. Nesse cenário, um pod contém um contêiner principal e vários contêineres de sidecar, conforme mostrado em **Figura 5-1**. Por exemplo, o contêiner principal é um servidor da Web que fornece serviços de arquivos a partir de um diretório fixo, e contêineres de sidecar baixam arquivos periodicamente para esse diretório fixo.

Figura 5-1 Pod executando vários contêineres

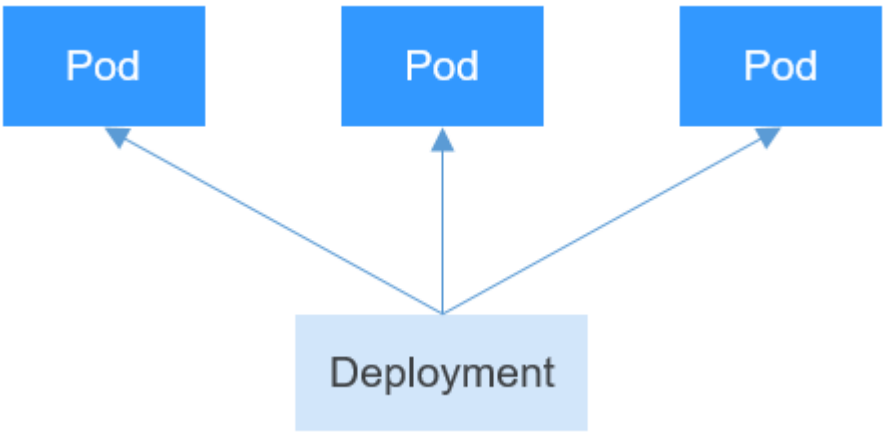


No Kubernetes, os pods são raramente criados diretamente. Em vez disso, o controlador do Kubernetes gerencia pods por meio de instâncias de pods, como Deployments e jobs. Um controlador normalmente usa um modelo de pod para criar pods. O controlador também pode gerenciar vários pods e fornecer funções como gerenciamento de réplicas, atualização contínua e autocorreção.

Visão geral de Deployment

Um pod é a unidade menor e mais simples que você cria ou implementa em Kubernetes. Ele foi projetado para ser uma entidade efêmera e única. Um pod pode ser removido quando os recursos do nó são insuficientes e desaparece junto com uma falha no nó do cluster. O Kubernetes fornece controladores para gerenciar pods. Os controladores podem criar e gerenciar pods e fornecer recursos de gerenciamento de réplicas, atualização contínua e autocorreção. O controlador mais comumente usado é o Deployment.

Figura 5-2 Relacionamento entre um Deployment e pods



Um Deployment pode conter um ou mais pods. Esses pods têm a mesma função. Portanto, o sistema distribui solicitações automaticamente para vários pods de um Deployment.

Um Deployment integra muitas funções, incluindo implementação on-line, atualização contínua, criação de réplicas e restauração de tarefas on-line. Até certo ponto, os Deployments podem ser usados para realizar a implementação autônoma, o que reduz bastante as dificuldades e os riscos de operação no processo de implementação.

Visão geral de StatefulSet

Todos os pods em um Deployment têm as mesmas características, exceto o nome e o endereço IP. Se necessário, um Deployment pode usar um modelo de pod para criar novos pods. Se não for necessário, o Deployment pode excluir qualquer um dos pods.

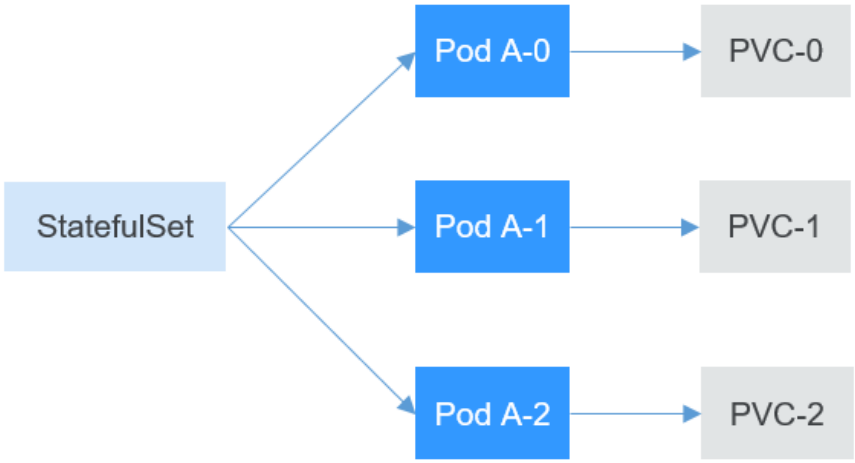
No entanto, os Deployments não podem atender aos requisitos em alguns cenários distribuídos quando cada pod requer seu próprio status ou em um banco de dados distribuído onde cada pod requer armazenamento independente.

As aplicações distribuídas com estado envolvem diferentes funções para diferentes responsabilidades. Por exemplo, os bancos de dados funcionam no modo ativo/em espera, e os pods dependem uns dos outros. Para implementar aplicações com estado no Kubernetes, certifique-se de que os pods atendam aos seguintes requisitos:

- Cada pod deve ter um identificador fixo para que possa ser reconhecido por outros pods.
- Recursos de armazenamento separados devem ser configurados para cada pod. Dessa forma, os dados originais podem ser recuperados depois que um pod é excluído e restaurado. Caso contrário, o status do pod será alterado depois que o pod for reconstruído.

Para atender aos requisitos anteriores, o Kubernetes fornece StatefulSets.

1. Os StatefulSets fornecem um nome fixo para cada pod seguindo um número fixo que varia de 0 a N. Depois que um pod é reprogramado, o nome do pod e o nome do host permanecem inalterados.
2. Os StatefulSets usam um Serviço sem periféricos para alocar um nome de domínio fixo para cada pod.
3. Os StatefulSets criam PersistentVolumeClaims (PVCs) com identificadores fixos para garantir que os pods acessem os mesmos dados persistentes após serem reprogramados.



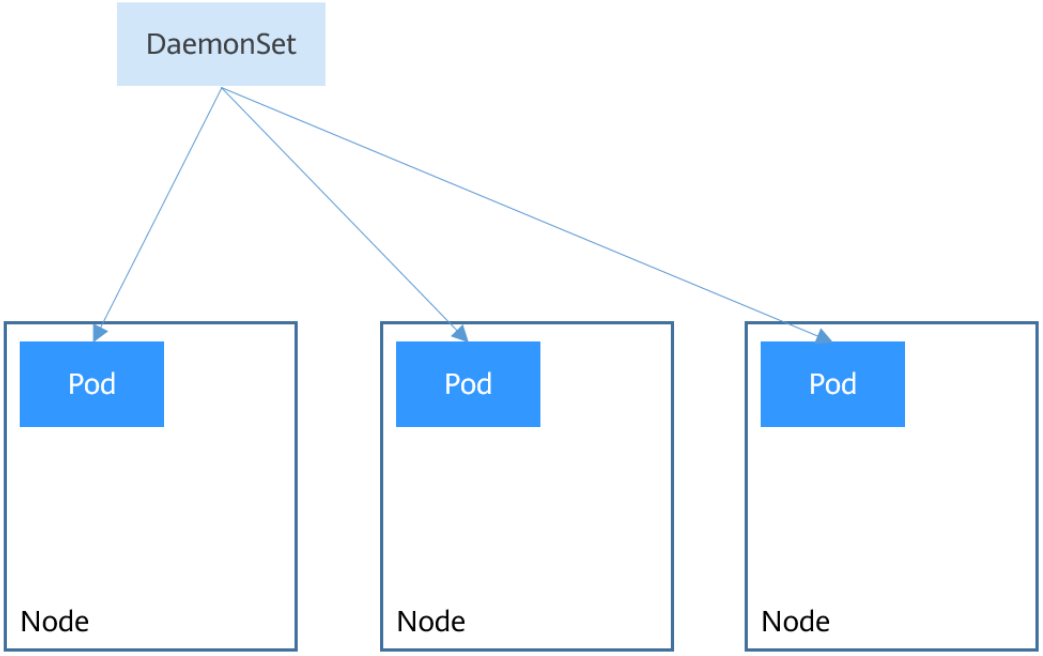
Visão geral do DaemonSet

Um DaemonSet executa um pod em cada nó em um cluster e garante que haja apenas um pod. Isso funciona bem para determinadas configurações no nível do sistema, como coleta de logs

e monitoramento de recursos, pois elas devem ser executadas em cada nó e precisam apenas de alguns pods. Um bom exemplo é o kube-proxy.

Os DaemonSets estão intimamente relacionados aos nós. Se um nó se tornar defeituoso, o DaemonSet não criará os mesmos pods em outros nós.

Figura 5-3 DaemonSet



Visão geral do job e do CronJob

Jobs e CronJobs permitem que você execute tarefas únicas e de curta duração em lote. Eles garantem que os pods de tarefas sejam executados até a conclusão.

- Um job é um objeto de recurso usado pelo Kubernetes para controlar tarefas em lote. Jobs são diferentes das tarefas servo de longo prazo (como Deployments e StatefulSets). O primeiro é iniciado e encerrado em horários específicos, enquanto o segundo é executado ininterruptamente, a menos que seja encerrado. Os pods gerenciados por um job serão removidos automaticamente após a conclusão bem-sucedida das tarefas com base nas configurações do usuário.
- Um CronJob executa um job periodicamente em uma programação especificada. Um objeto CronJob é semelhante a uma linha de um arquivo crontab no Linux.

Esse recurso de execução até a conclusão dos jobs é especialmente adequado para tarefas pontuais, como integração contínua (CI).

Ciclo de vida da carga de trabalho

Tabela 5-1 Descrição de status

Estado	Descrição
Running	Todos os pods estão em execução ou o número de pods é 0.

Estado	Descrição
Unready	O contêiner funciona mal e o pod sob a carga de trabalho não está funcionando.
Processing	A carga de trabalho não está em execução, mas nenhum erro é relatado.
Available	Para uma Implementação de vários pods, alguns pods são anormais, mas pelo menos um pod está disponível.
Completed	A tarefa foi executada com sucesso. Esse status está disponível somente para tarefas comuns.
Stopped	A carga de trabalho é interrompida e o número de pods é alterado para 0. Esse status está disponível para cargas de trabalho anteriores à v1.13.
Deleting	A carga de trabalho está sendo excluída.

5.2 Criação de uma carga de trabalho

5.2.1 Criação de uma Implantação

Cenário

As Implantações são cargas de trabalho (por exemplo, Nginx) que não armazenam dados ou status. Você pode criar Implantações no console do CCE ou executando comandos kubectl.

Pré-requisitos

- Antes de criar uma carga de trabalho, você deve ter um cluster disponível. Para obter detalhes sobre como criar um cluster, consulte **Compra de um cluster do CCE**.
- Para habilitar o acesso público a uma carga de trabalho, verifique se um EIP ou balanceador de carga foi vinculado a pelo menos um nó no cluster.

  NOTA

  Se um pod tiver vários contêineres, certifique-se de que as portas usadas pelos contêineres não entrem em conflito entre si. Caso contrário, a criação da Implantação falhará.

Usar o console do CCE

Passo 1 Efetue logon no console do CCE.

Passo 2 Clique no nome do cluster para acessar o console do cluster, escolha **Workloads** no painel de navegação e clique em **Create Workload** no canto superior direito.

Passo 3 Defina informações básicas sobre a carga de trabalho.

Basic Info

- **Workload Type**: selecione **Deployment**. Para obter detalhes sobre os tipos de carga de trabalho, consulte **Visão geral**.

- **Workload Name**: insira o nome da carga de trabalho. Digite de 1 a 63 caracteres começando com uma letra minúscula e terminando com uma letra minúscula ou dígito. Somente letras minúsculas, dígitos e hifens (-) são permitidos.
- **Namespace**: selecione o namespace da carga de trabalho. O valor padrão é **default**. Você também pode clicar em **Create Namespace** para criar um. Para mais detalhes, consulte **Criação de um namespace**.
- **Pods**: digite o número de pods da carga de trabalho.
- **Container Runtime**: um cluster do CCE usa runC por padrão, enquanto um cluster do CCE Turbo suporta runC e Kata. Para obter detalhes sobre as diferenças, consulte **Tempo de execução do Kata e tempo de execução comum**.
- **Time Zone Synchronization**: especifique se deseja ativar a sincronização de fuso horário. Depois que a sincronização de fuso horário estiver ativada, o contêiner e o nó usarão o mesmo fuso horário. A função de sincronização de fuso horário depende do disco local montado no contêiner. Não modifique nem exclua o fuso horário. Para mais detalhes, consulte **Configuração da sincronização de fuso horário**.

Container Settings

- Informações sobre o contêiner

  Vários contêineres podem ser configurados em um pod. Você pode clicar em **Add Container** à direita para configurar vários contêineres para o pod.

  – **Basic Info**: configure informações básicas sobre o contêiner.

Parâmetro	Descrição
Container Name	Nomeie o contêiner.
Pull Policy	Política de atualização ou extração de imagem. Se você selecionar **Always**, a imagem será extraída do repositório de imagens a cada vez. Se você não selecionar **Always**, a imagem existente do nó é usada preferencialmente. Se a imagem não existir, a imagem será extraída do repositório de imagens.
Image Name	Clique em **Select Image** e selecione a imagem usada pelo contêiner. Para usar uma imagem de terceiros, consulte **Uso de imagens de terceiros**.
Image Tag	Selecione a tag de imagem a ser implementada.
CPU Quota	■ **Request**: número mínimo de núcleos de CPU exigidos por um contêiner. O valor padrão é 0,25 núcleos. ■ **Limit**: número máximo de núcleos de CPU disponíveis para um contêiner. Não deixe **Limit** não especificado. Caso contrário, ocorrerá um uso intensivo de recursos de contêiner e sua carga de trabalho poderá exibir um comportamento inesperado. Se **Request** e **Limit** não forem especificados, a cota não é limitada. Para obter mais informações e sugestões sobre **Request** e **Limit**, consulte **Configuração de especificações do contêiner**.

Parâmetro	Descrição
Memory Quota	■ **Request**: quantidade mínima de memória requerida por um contêiner. O valor padrão é 512 MiB. ■ **Limit**: quantidade máxima de memória disponível para um contêiner. Quando o uso de memória exceder o limite de memória especificado, o contêiner será encerrado. Se **Request** e **Limit** não forem especificados, a cota não é limitada. Para obter mais informações e sugestões sobre **Request** e **Limit**, consulte **Configuração de especificações do contêiner**.
(Optional) GPU Quota	Configurável somente quando o cluster contém nós de GPU e o complemento **Suíte IA do CCE (GPU NVIDIA)** está instalado. ■ **All**: nenhuma GPU será usada. ■ **Dedicated**: os recursos da GPU são dedicados ao contêiner. ■ **Shared**: porcentagem de recursos de GPU usados pelo contêiner. Por exemplo, se esse parâmetro for definido como **10%**, o contêiner usará 10% dos recursos da GPU. Para obter detalhes sobre como usar GPUs no cluster, consulte **Agendamento de GPU padrão no Kubernetes**.
(Optional) NPU Quota	Número de chips de Ascend 310 necessários para o contêiner. O valor deve ser um número inteiro e o complemento **Suíte de IA do CCE (Ascend NPU)** deve ser instalado. Para obter detalhes sobre como usar NPUs no cluster, consulte **Agendamento de NPU**.
(Optional) Privileged Container	Programas em um contêiner privilegiado têm certos privilégios. Se **Privileged Container** estiver habilitado, os privilégios serão atribuídos ao contêiner. Por exemplo, contêineres privilegiados podem manipular dispositivos de rede na máquina host e modificar parâmetros do kernel.
(Optional) Init Container	Se usar o contêiner como um contêiner init. Um contêiner init não suporta verificação de integridade. Um contêiner init é um contêiner especial que é executado antes que outros contêineres de aplicações em um pod sejam iniciados. Cada pod pode conter vários contêineres. Além disso, um pod pode conter um ou mais contêineres init. Os contêineres de aplicações em um pod são iniciados e executados somente após a conclusão da execução de todos os contêineres init. Para obter detalhes, consulte **Contêineres init**.

– (Opcional) **Lifecycle**: configure as operações a serem executadas em uma fase específica do ciclo de vida do contêiner, como Startup Command, Post-Start e Pre-

Stop. Para mais detalhes, consulte **Definição dos parâmetros do ciclo de vida do contêiner**.

- (Opcional) **Health Check**: configure a sonda de vivacidade, a sonda pronta e a sonda de inicialização conforme necessário. Para mais detalhes, consulte **Configuração da verificação de integridade para um contêiner**.
- (Opcional) **Environment Variables**: configure variáveis para o ambiente em execução do contêiner usando pares chave-valor. Essas variáveis transferem informações externas para contêineres executados em pods e podem ser modificadas de forma flexível após a implementação da aplicação. Para mais detalhes, consulte **Configuração de uma variável de ambiente**.
- (Opcional) **Data Storage**: monte armazenamento local ou armazenamento em nuvem no contêiner. Os cenários de aplicações e os modos de montagem variam de acordo com o tipo de armazenamento. Para mais detalhes, consulte **Armazenamento**.

  NOTA

  Se a carga de trabalho contiver mais de um pod, os volumes do EVS não poderão ser montados.
- (Opcional) **Security Context**: atribua permissões de contêiner para proteger o sistema e outros contêineres de serem afetados. Insira o ID do usuário para atribuir permissões de contêiner e impedir que sistemas e outros contêineres sejam afetados.
- (Opcional) **Logging**: relate os logs de saída de contêiner padrão para o AOM por padrão, sem a necessidade de configurações manuais. Você pode configurar manualmente o caminho da coleção de logs. Para mais detalhes, consulte **Uso do ICAgent para coletar logs de contêiner**.

  Para desativar a saída padrão da carga de trabalho atual, adicione a anotação **kubernetes.AOM.log.stdout: []** em **Rótulos e anotações**. Para obter detalhes sobre como usar essa anotação, consulte **Tabela 5-16**.

- **Image Access Credential**: selecione a credencial usada para acessar o repositório de imagens. O valor padrão é **default-secret**. Você pode usar default-secret para acessar imagens no SWR. Para obter detalhes sobre **default-secret**, consulte **default-secret**.
- (Opcional) **GPU**: **All** é selecionado por padrão. A instância da carga de trabalho será agendada para o nó do tipo de GPU especificado.

(Opcional) Service Settings

Um Serviço fornece acesso externo para pods. Com um endereço IP estático, um Serviço encaminha o tráfego de acesso aos pods e equilibra automaticamente a carga desses pods.

Você também pode criar um Serviço depois de criar uma carga de trabalho. Para obter detalhes sobre Serviços de diferentes tipos, consulte **Visão geral**.

(Opcional) Advanced Settings

- **Upgrade:** especifique o modo de atualizar e os parâmetros de upgrade da carga de trabalho. **Rolling upgrade** e **Replace upgrade** são suportados. Para mais detalhes, consulte **Configuração da política de atualização da carga de trabalho**.
- **Scheduling**: configure políticas de afinidade e antiafinidade para agendamento flexível de carga de trabalho. Afinidade de nó, afinidade de pod e antiafinidade de pod são suportadas. Para mais detalhes, consulte **Política de agendamento (afinidade/antiafinidade)**.
- **Toleration**: o uso de ambas manchas e tolerâncias permite (não forçosamente) que o pod seja agendado para um nó com as manchas correspondentes e controla as políticas de

despejo de pod depois que o nó onde o pod está localizado é manchado. Para mais detalhes, consulte **Manchas e tolerâncias**.

- **Labels and Annotations**: adicione rótulos ou anotações para pods usando pares chave-valor. Depois de inserir a chave e o valor, clique em **Confirm**. Para obter detalhes sobre como usar e configurar rótulos e anotações, consulte **Rótulos e anotações**.
- **DNS**: configure uma política DNS separada para a carga de trabalho. Para mais detalhes, consulte **Configuração de DNS**.
- **APM Settings**: use o Application Performance Management (APM) para fornecer uma análise de problemas e localização mais precisas para programas Java. Para mais detalhes, consulte **Configuração de configurações de APM para análise de gargalo de desempenho**.
- **Network configuration:**
  - limitação de largura de banda de entrada/saída do pod: Você pode definir a limitação de largura de banda de entrada/saída para pods. Para mais detalhes, consulte **Configuração da limitação da taxa de QoS para acesso entre pods**.

Passo 4 Clique em **Create Workload** no canto inferior direito.

----Fim

Usar kubectl

O procedimento a seguir usa o Nginx como exemplo para descrever como criar uma carga de trabalho usando o kubectl.

Passo 1 Use o kubectl para se conectar ao cluster. Para mais detalhes, consulte **Conexão a um cluster usando o kubectl**.

Passo 2 Crie e edite o arquivo **nginx-deployment.yaml**. **nginx-deployment.yaml** é um nome de arquivo de exemplo. Você pode renomeá-lo conforme necessário.

vi nginx-deployment.yaml

O seguinte é um exemplo de arquivo YAML. Para obter mais informações sobre Implementações, consulte a **documentação do Kubernetes**.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  strategy:
    type: RollingUpdate
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - image: nginx    # If you use an image from an open-source image registry,
enter the image name. If you use an image in My Images, obtain the image path
from SWR.
        imagePullPolicy: Always
        name: nginx
      imagePullSecrets:
      - name: default-secret
```


Para obter detalhes sobre os parâmetros, consulte **Tabela 5-2**.

Tabela 5-2 Parâmetros de Implementação YAML

Parâmetro	Descrição	Obrigatório/opcional
apiVersion	Versão da API. **NOTA** Defina este parâmetro com base na versão do cluster. ● Para clusters v1.17 ou posterior, o formato de apiVersion de Implementações é **apps/v1**. ● Para clusters de v1.15 ou anterior, o formato de apiVersion de Implementações é **extensions/v1beta1**.	Obrigatório
kind	Tipo de um objeto criado.	Obrigatório
metadata	Metadados de um objeto de recurso.	Obrigatório
name	Nome da Implementação.	Obrigatório
spec	Descrição detalhada da Implementação.	Obrigatório
replicas	Número de pods.	Obrigatório
selector	Determina os pods de contêiner que podem ser gerenciados pela Implementação.	Obrigatório
strategy	Modo de atualização. Valores possíveis: ● RollingUpdate ● ReplaceUpdate Por padrão, a atualização rolante é usada.	Opcional
template	Descrição detalhada de um pod de contêiner criado.	Obrigatório
metadata	Metadados.	Obrigatório
labels	**metadata.labels**: rótulos de contêineres.	Opcional

Parâmetro	Descrição	Obrigatório/opcional
spec: containers	● **image** (obrigatório): nome de uma imagem de contêiner. ● **imagePullPolicy** (opcional): política para obtenção de uma imagem. As opções incluem **Always** (tentando baixar imagens a cada vez), **Never** (usando apenas imagens locais) e **IfNotPresent** (usando imagens locais se estiverem disponíveis; baixando imagens se imagens locais não estiverem disponíveis). O valor padrão é **Always**. ● **name** (obrigatório): nome do contêiner.	Obrigatório
imagePullSecrets	Nome do segredo usado durante a extração da imagem. Se for utilizada uma imagem privada, este parâmetro é obrigatório. ● Para extrair uma imagem do Software Repository for Container (SWR), defina esse parâmetro como **default-secret**. ● Para extrair uma imagem de um repositório de imagens de terceiros, defina esse parâmetro como o nome do segredo criado.	Opcional

Passo 3 Crie uma Implementação.

kubectl create -f nginx-deployment.yaml

Se as informações a seguir forem exibidas, isso indicará que a Implementação está sendo criada.

```
deployment "nginx" created
```

Passo 4 Consulte o status da Implementação.

kubectl get deployment

Se as informações a seguir forem exibidas, a Implementação está sendo executada.

```
NAME           READY     UP-TO-DATE   AVAILABLE   AGE
nginx          1/1       1            1           4m5s
```

Descrição do parâmetro

- **NAME**: nome da aplicação em execução no pod.
- **READY**: indica o número de cargas de trabalho disponíveis. O valor é exibido como "the number of available pods/the number of expected pods".

- **UP-TO-DATE**: indica o número de réplicas que foram atualizadas.
- **AVAILABLE**: indica o número de pods disponíveis.
- **AGE**: período em que a Implementação continua funcionando

Passo 5 Se a Implementação for acessada por meio de um Serviço ClusterIP ou NodePort, adicione o Serviço correspondente. Para mais detalhes, consulte **Rede**.

----Fim

Documentação

- **Atualização de pods sem interromper serviços**
- **Configuração da resolução de nomes de domínio para contêineres do CCE**

5.2.2 Criação de um StatefulSet

Cenário

StatefulSets são um tipo de cargas de trabalho cujos dados ou status são armazenados enquanto estão em execução. Por exemplo, MySQL é um StatefulSet porque ele precisa armazenar novos dados.

Um contêiner pode ser migrado entre hosts diferentes, mas os dados não são armazenados nos hosts. Para armazenar dados de StatefulSet de forma persistente, anexar volumes de armazenamento HA fornecidos pelo CCE para o recipiente.

Restrições

- Quando você exclui ou escala um StatefulSet o sistema não exclui os volumes de armazenamento associados ao StatefulSet para garantir a segurança dos dados.
- Ao excluir um StatefulSet, reduza o número de réplicas para **0** antes de excluir o StatefulSet para que os pods no StatefulSet possam ser interrompidos em ordem.
- Quando você cria um StatefulSet, um Serviço headless é necessário para o acesso ao pod. Para mais detalhes, consulte **Serviços headless**.
- Quando um nó está indisponível, os pods se tornam **Unready**. Nesse caso, exclua manualmente os pods do StatefulSet para que os pods possam ser migrados para um nó normal.

Pré-requisitos

- Antes de criar uma carga de trabalho, você deve ter um cluster disponível. Para obter detalhes sobre como criar um cluster, consulte **Compra de um cluster do CCE**.
- Para habilitar o acesso público a uma carga de trabalho, verifique se um EIP ou balanceador de carga foi vinculado a pelo menos um nó no cluster.

  NOTA

  Se um pod tiver vários contêineres, certifique-se de que as portas usadas pelos contêineres não entrem em conflito entre si. Caso contrário, a criação do StatefulSet falhará.

Usar o console do CCE

Passo 1 Efetue logon no console do CCE.

Passo 2 Clique no nome do cluster para acessar o console do cluster, escolha **Workloads** no painel de navegação e clique em **Create Workload** no canto superior direito.

Passo 3 Defina informações básicas sobre a carga de trabalho.

Basic Info

- **Workload Type**: selecione **StatefulSet**. Para obter detalhes sobre os tipos de carga de trabalho, consulte **Visão geral**.
- **Workload Name**: insira o nome da carga de trabalho. Digite de 1 a 63 caracteres começando com uma letra minúscula e terminando com uma letra minúscula ou dígito. Somente letras minúsculas, dígitos e hifens (-) são permitidos.
- **Namespace**: selecione o namespace da carga de trabalho. O valor padrão é **default**. Você também pode clicar em **Create Namespace** para criar um. Para mais detalhes, consulte **Criação de um namespace**.
- **Pods**: digite o número de pods da carga de trabalho.
- **Container Runtime**: um cluster do CCE usa runC por padrão, enquanto um cluster do CCE Turbo suporta runC e Kata. Para obter detalhes sobre as diferenças, consulte **Tempo de execução do Kata e tempo de execução comum**.
- **Time Zone Synchronization**: especifique se deseja ativar a sincronização de fuso horário. Depois que a sincronização de fuso horário estiver ativada, o contêiner e o nó usarão o mesmo fuso horário. A função de sincronização de fuso horário depende do disco local montado no contêiner. Não modifique nem exclua o fuso horário. Para mais detalhes, consulte **Configuração da sincronização de fuso horário**.

Container Settings

- Informações sobre o contêiner

  Vários contêineres podem ser configurados em um pod. Você pode clicar em **Add Container** à direita para configurar vários contêineres para o pod.

  – **Basic Info**: configure informações básicas sobre o contêiner.

Parâmetro	Descrição
Container Name	Nomeie o contêiner.
Pull Policy	Política de atualização ou extração de imagem. Se você selecionar **Always**, a imagem será extraída do repositório de imagens a cada vez. Se você não selecionar **Always**, a imagem existente do nó é usada preferencialmente. Se a imagem não existir, a imagem será extraída do repositório de imagens.
Image Name	Clique em **Select Image** e selecione a imagem usada pelo contêiner. Para usar uma imagem de terceiros, consulte **Uso de imagens de terceiros**.
Image Tag	Selecione a tag de imagem a ser implementada.

Parâmetro	Descrição
CPU Quota	■ **Request**: número mínimo de núcleos de CPU exigidos por um contêiner. O valor padrão é 0,25 núcleos. ■ **Limit**: número máximo de núcleos de CPU disponíveis para um contêiner. Não deixe **Limit** não especificado. Caso contrário, ocorrerá um uso intensivo de recursos de contêiner e sua carga de trabalho poderá exibir um comportamento inesperado. Se **Request** e **Limit** não forem especificados, a cota não é limitada. Para obter mais informações e sugestões sobre **Request** e **Limit**, consulte **Configuração de especificações do contêiner**.
Memory Quota	■ **Request**: quantidade mínima de memória requerida por um contêiner. O valor padrão é 512 MiB. ■ **Limit**: quantidade máxima de memória disponível para um contêiner. Quando o uso de memória exceder o limite de memória especificado, o contêiner será encerrado. Se **Request** e **Limit** não forem especificados, a cota não é limitada. Para obter mais informações e sugestões sobre **Request** e **Limit**, consulte **Configuração de especificações do contêiner**.
(Optional) GPU Quota	Configurável somente quando o cluster contém nós de GPU e o complemento **Suíte IA do CCE (GPU NVIDIA)** está instalado. ■ **All**: nenhuma GPU será usada. ■ **Dedicated**: os recursos da GPU são dedicados ao contêiner. ■ **Shared**: porcentagem de recursos de GPU usados pelo contêiner. Por exemplo, se esse parâmetro for definido como **10%**, o contêiner usará 10% dos recursos da GPU. Para obter detalhes sobre como usar GPUs no cluster, consulte **Agendamento de GPU padrão no Kubernetes**.
(Optional) NPU Quota	Número de chips de Ascend 310 necessários para o contêiner. O valor deve ser um número inteiro e o complemento **Suíte de IA do CCE (Ascend NPU)** deve ser instalado. Para obter detalhes sobre como usar NPUs no cluster, consulte **Agendamento de NPU**.
(Optional) Privileged Container	Programas em um contêiner privilegiado têm certos privilégios. Se **Privileged Container** estiver habilitado, os privilégios serão atribuídos ao contêiner. Por exemplo, contêineres privilegiados podem manipular dispositivos de rede na máquina host e modificar parâmetros do kernel.

Parâmetro	Descrição
(Optional) Init Container	Se usar o contêiner como um contêiner init. Um contêiner init não suporta verificação de integridade. Um contêiner init é um contêiner especial que é executado antes que outros contêineres de aplicações em um pod sejam iniciados. Cada pod pode conter vários contêineres. Além disso, um pod pode conter um ou mais contêineres init. Os contêineres de aplicações em um pod são iniciados e executados somente após a conclusão da execução de todos os contêineres init. Para obter detalhes, consulte **Contêineres init**.

- (Opcional) **Lifecycle**: configure as operações a serem executadas em uma fase específica do ciclo de vida do contêiner, como Startup Command, Post-Start e Pre-Stop. Para mais detalhes, consulte **Definição dos parâmetros do ciclo de vida do contêiner**.
- (Opcional) **Health Check**: configure a sonda de vivacidade, a sonda pronta e a sonda de inicialização conforme necessário. Para mais detalhes, consulte **Configuração da verificação de integridade para um contêiner**.
- (Opcional) **Environment Variables**: configure variáveis para o ambiente em execução do contêiner usando pares chave-valor. Essas variáveis transferem informações externas para contêineres executados em pods e podem ser modificadas de forma flexível após a implementação da aplicação. Para mais detalhes, consulte **Configuração de uma variável de ambiente**.
- (Opcional) **Armazenamento de dados**: monte armazenamento local ou armazenamento em nuvem no contêiner. Os cenários de aplicações e os modos de montagem variam de acordo com o tipo de armazenamento. Para mais detalhes, consulte **Armazenamento**.

  NOTA

  - StatefulSets oferecem suporte à conexão dinâmica de discos EVS. Para obter mais detalhes, consulte **Montagem dinâmica de um disco EVS para um StatefulSet** e **Montagem dinâmica de um PV local para um StatefulSet**.

    A montagem dinâmica é obtida usando o campo **volumeClaimTemplates** e depende da capacidade de criação dinâmica da StorageClass. Um StatefulSet associa cada pod a uma PVC usando o campo **volumeClaimTemplates**, e a PVC é vinculada ao PV correspondente. Portanto, depois que o pod é reprogramado, os dados originais ainda podem ser montados com base no nome da PVC.
  - Depois que uma carga de trabalho é criada, o armazenamento montado dinamicamente não pode ser atualizado.
- (Opcional) **Security Context**: atribua permissões de contêiner para proteger o sistema e outros contêineres de serem afetados. Insira o ID do usuário para atribuir permissões de contêiner e impedir que sistemas e outros contêineres sejam afetados.
- (Opcional) **Logging**: relate os logs de saída de contêiner padrão para o AOM por padrão, sem a necessidade de configurações manuais. Você pode configurar manualmente o caminho da coleção de logs. Para mais detalhes, consulte **Uso do ICAgent para coletar logs de contêiner**.

  Para desativar a saída padrão da carga de trabalho atual, adicione a anotação **kubernetes.AOM.log.stdout: []** em **Rótulos e anotações**. Para obter detalhes sobre como usar essa anotação, consulte **Tabela 5-16**.

- **Image Access Credential**: selecione a credencial usada para acessar o repositório de imagens. O valor padrão é **default-secret**. Você pode usar default-secret para acessar imagens no SWR. Para obter detalhes sobre **default-secret**, consulte **default-secret**.
- (Opcional) **GPU**: **All** é selecionado por padrão. A instância da carga de trabalho será agendada para o nó do tipo de GPU especificado.

Parâmetros de Serviço headless

Um Serviço headless é usado para resolver o problema de acesso mútuo entre pods em um StatefulSet. O Serviço headless fornece um nome de domínio de acesso fixo para cada pod. Para mais detalhes, consulte **Serviços headless**.

(Opcional) Service Settingso

Um Serviço fornece acesso externo para pods. Com um endereço IP estático, um Serviço encaminha o tráfego de acesso aos pods e equilibra automaticamente a carga desses pods.

Você também pode criar um Serviço depois de criar uma carga de trabalho. Para obter detalhes sobre Serviços de diferentes tipos, consulte **Visão geral**.

(Opcional) Advanced Settings

- **Upgrade:** especifique o modo de atualizar e os parâmetros de upgrade da carga de trabalho. **Rolling upgrade** e **Replace upgrade** são suportados. Para mais detalhes, consulte **Configuração da política de atualização da carga de trabalho**.
- **Políticas de gerenciamento de pods**

  Para alguns sistemas distribuídos, a sequência de StatefulSet é desnecessária e/ou não deve ocorrer. Esses sistemas exigem apenas exclusividade e identificadores.

  – **OrderedReady**: o StatefulSet implantará, excluirá ou dimensionará os pods em ordem e um por um. (O StatefulSet só continua depois que o pod anterior estiver pronto ou excluído.) Esta é a política padrão.

  – **Parallel**: o StatefulSet criará pods em paralelo para corresponder à escala desejada sem esperar e excluirá todos os pods de uma só vez.
- **Scheduling**: configure políticas de afinidade e antiafinidade para agendamento flexível de carga de trabalho. Afinidade de nó, afinidade de pod e antiafinidade de pod são suportadas. Para mais detalhes, consulte **Política de agendamento (afinidade/antiafinidade)**.
- **Toleration**: o uso de ambas manchas e tolerâncias permite (não forçosamente) que o pod seja agendado para um nó com as manchas correspondentes e controla as políticas de despejo de pod depois que o nó onde o pod está localizado é manchado. Para mais detalhes, consulte **Manchas e tolerâncias**.
- **Labels and Annotations**: adicione rótulos ou anotações para pods usando pares chave-valor. Depois de inserir a chave e o valor, clique em **Confirm**. Para obter detalhes sobre como usar e configurar rótulos e anotações, consulte **Rótulos e anotações**.
- **DNS**: configure uma política DNS separada para a carga de trabalho. Para mais detalhes, consulte **Configuração de DNS**.
- **APM Settings**: use o Application Performance Management (APM) para fornecer uma análise de problemas e localização mais precisas para programas Java. Para mais detalhes, consulte **Configuração de configurações de APM para análise de gargalo de desempenho**.

Passo 4 Clique em **Create Workload** no canto inferior direito.

----Fim

Usar kubectl

Neste exemplo, uma carga de trabalho de nginx é usada e o volume do EVS é montado dinamicamente nela usando o campo **volumeClaimTemplates**.

Passo 1 Use o kubectl para se conectar ao cluster. Para mais detalhes, consulte **Conexão a um cluster usando o kubectl**.

Passo 2 Crie e edite o arquivo **nginx-statefulset.yaml**.

nginx-statefulset.yaml é um nome de arquivo de exemplo, e você pode alterá-lo conforme necessário.

vi nginx-statefulset.yaml

A seguir é apresentado um exemplo do conteúdo do arquivo. Para obter mais informações sobre o StatefulSet consulte a **documentação do Kubernetes**.

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: nginx
spec:
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: container-1
          image: nginx:latest
          imagePullPolicy: IfNotPresent
          resources:
            requests:
              cpu: 250m
              memory: 512Mi
            limits:
              cpu: 250m
              memory: 512Mi
          volumeMounts:
            - name: test
              readOnly: false
              mountPath: /usr/share/nginx/html
              subPath: ''
      imagePullSecrets:
        - name: default-secret
      dnsPolicy: ClusterFirst
      volumes: []
  serviceName: nginx-svc
  replicas: 2
  volumeClaimTemplates:  # Dynamically mounts the EVS volume to the workload.
    - apiVersion: v1
      kind: PersistentVolumeClaim
      metadata:
        name: test
        namespace: default
        annotations:
          everest.io/disk-volume-type: SAS  # SAS EVS volume type.
        labels:
          failure-domain.beta.kubernetes.io/region: ap-southeast-1  # region
where the EVS volume is created.
          failure-domain.beta.kubernetes.io/zone: ap-southeast-1a   # AZ where
the EVS volume is created. It must be the same as the AZ of the node.
      spec:
```



```
        accessModes:
          - ReadWriteOnce  # The value must be ReadWriteOnce for the EVS volume.
        resources:
          requests:
            storage: 10Gi
        storageClassName: csi-disk # Storage class name. The value is csi-disk
for the EVS volume.
  updateStrategy:
    type: RollingUpdate
```

vi nginx-headless.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-svc
  namespace: default
  labels:
    app: nginx
spec:
  selector:
    app: nginx
    version: v1
  clusterIP: None
  ports:
    - name: nginx
      targetPort: 80
      nodePort: 0
      port: 80
      protocol: TCP
  type: ClusterIP
```

Passo 3 Crie uma carga de trabalho e o serviço headless correspondente.

kubectl create -f nginx-statefulset.yaml

Se as informações a seguir forem exibidas, o StatefulSet foi criado com sucesso.

```
statefulset.apps/nginx created
```

kubectl create -f nginx-headless.yaml

Se as seguintes informações forem exibidas, o serviço headless foi criado com êxito.

```
service/nginx-svc created
```

Passo 4 Se a carga de trabalho for acessada por meio de um Serviço ClusterIP ou NodePort, defina o tipo de acesso à carga de trabalho correspondente. Para mais detalhes, consulte **Rede**.

----Fim

5.2.3 Criação de um DaemonSet

Cenário

O CCE fornece recursos de implementação e gerenciamento para vários tipos de contêineres e oferece suporte a recursos de cargas de trabalho de contêiner, incluindo criação, configuração, monitoramento, escalabilidade, atualização, desinstalação, descoberta de serviços e balanceamento de carga.

DaemonSet garante que apenas um pod seja executado em todos ou em alguns nós. Quando um nó é adicionado a um cluster, um novo pod também é adicionado para o nó. Quando um nó é removido de um cluster, o pod também é recuperado. Se um DaemonSet for excluído, todos os pods criados por ele serão excluídos.

Os cenários de aplicação típicos de um DaemonSet são os seguintes:

- Execute o daemon de armazenamento de cluster, como glusterd ou Ceph, em cada nó.
- Execute o daemon de coleta de logs, como Fluentd ou Logstash, em cada nó.
- Execute o daemon de monitoramento, como Prometheus Node Exporter, Collectd, Datadog Agent, New Relic Agent ou Ganglia (gmond), em cada nó.

Você pode implementar um DaemonSet para cada tipo de daemons em todos os nós ou implementar vários DaemonSets para o mesmo tipo de daemons. No segundo caso, os DaemonSets têm diferentes flags e diferentes requisitos de memória e CPU para diferentes tipos de hardware.

Pré-requisitos

Antes de criar um DaemonSet, é necessário ter um cluster disponível. Para obter detalhes sobre como criar um cluster, consulte **Compra de um cluster do CCE**.

Usar o console do CCE

Passo 1 Efetue logon no console do CCE.

Passo 2 Clique no nome do cluster para acessar o console do cluster, escolha **Workloads** no painel de navegação e clique em **Create Workload** no canto superior direito.

Passo 3 Defina informações básicas sobre a carga de trabalho.

Basic Info

- **Workload Type**: selecione **DaemonSet**. Para obter detalhes sobre os tipos de carga de trabalho, consulte **Visão geral**.
- **Workload Name**: insira o nome da carga de trabalho. Digite de 1 a 63 caracteres começando com uma letra minúscula e terminando com uma letra minúscula ou dígito. Somente letras minúsculas, dígitos e hifens (-) são permitidos.
- **Namespace**: selecione o namespace da carga de trabalho. O valor padrão é **default**. Você também pode clicar em **Create Namespace** para criar um. Para mais detalhes, consulte **Criação de um namespace**.
- **Container Runtime**: um cluster do CCE usa runC por padrão, enquanto um cluster do CCE Turbo suporta runC e Kata. Para obter detalhes sobre as diferenças, consulte **Tempo de execução do Kata e tempo de execução comum**.
- **Time Zone Synchronization**: especifique se deseja ativar a sincronização de fuso horário. Depois que a sincronização de fuso horário estiver ativada, o contêiner e o nó usarão o mesmo fuso horário. A função de sincronização de fuso horário depende do disco local montado no contêiner. Não modifique nem exclua o fuso horário. Para mais detalhes, consulte **Configuração da sincronização de fuso horário**.

Container Settings

- Informações sobre o contêiner

  Vários contêineres podem ser configurados em um pod. Você pode clicar em **Add Container** à direita para configurar vários contêineres para o pod.

  – **Basic Info**: configure informações básicas sobre o contêiner.

Parâmetro	Descrição
Container Name	Nomeie o contêiner.
Pull Policy	Política de atualização ou extração de imagem. Se você selecionar **Always**, a imagem será extraída do repositório de imagens a cada vez. Se você não selecionar **Always**, a imagem existente do nó é usada preferencialmente. Se a imagem não existir, a imagem será extraída do repositório de imagens.
Image Name	Clique em **Select Image** e selecione a imagem usada pelo contêiner. Para usar uma imagem de terceiros, consulte **Uso de imagens de terceiros**.
Image Tag	Selecione a tag de imagem a ser implementada.
CPU Quota	■ **Request**: número mínimo de núcleos de CPU exigidos por um contêiner. O valor padrão é 0,25 núcleos. ■ **Limit**: número máximo de núcleos de CPU disponíveis para um contêiner. Não deixe **Limit** não especificado. Caso contrário, ocorrerá um uso intensivo de recursos de contêiner e sua carga de trabalho poderá exibir um comportamento inesperado. Se **Request** e **Limit** não forem especificados, a cota não é limitada. Para obter mais informações e sugestões sobre **Request** e **Limit**, consulte **Configuração de especificações do contêiner**.
Memory Quota	■ **Request**: quantidade mínima de memória requerida por um contêiner. O valor padrão é 512 MiB. ■ **Limit**: quantidade máxima de memória disponível para um contêiner. Quando o uso de memória exceder o limite de memória especificado, o contêiner será encerrado. Se **Request** e **Limit** não forem especificados, a cota não é limitada. Para obter mais informações e sugestões sobre **Request** e **Limit**, consulte **Configuração de especificações do contêiner**.
(Optional) GPU Quota	Configurável somente quando o cluster contém nós de GPU e o complemento **Suíte IA do CCE (GPU NVIDIA)** está instalado. ■ **All**: nenhuma GPU será usada. ■ **Dedicated**: os recursos da GPU são dedicados ao contêiner. ■ **Shared**: porcentagem de recursos de GPU usados pelo contêiner. Por exemplo, se esse parâmetro for definido como **10%**, o contêiner usará 10% dos recursos da GPU. Para obter detalhes sobre como usar GPUs no cluster, consulte **Agendamento de GPU padrão no Kubernetes**.

Parâmetro	Descrição
(Optional) NPU Quota	Número de chips de Ascend 310 necessários para o contêiner. O valor deve ser um número inteiro e o complemento **Suíte de IA do CCE (Ascend NPU)** deve ser instalado. Para obter detalhes sobre como usar NPUs no cluster, consulte **Agendamento de NPU**.
(Optional) Privileged Container	Programas em um contêiner privilegiado têm certos privilégios. Se **Privileged Container** estiver habilitado, os privilégios serão atribuídos ao contêiner. Por exemplo, contêineres privilegiados podem manipular dispositivos de rede na máquina host e modificar parâmetros do kernel.
(Optional) Init Container	Se usar o contêiner como um contêiner init. Um contêiner init não suporta verificação de integridade. Um contêiner init é um contêiner especial que é executado antes que outros contêineres de aplicações em um pod sejam iniciados. Cada pod pode conter vários contêineres. Além disso, um pod pode conter um ou mais contêineres init. Os contêineres de aplicações em um pod são iniciados e executados somente após a conclusão da execução de todos os contêineres init. Para obter detalhes, consulte **Contêineres init**.

- (Opcional) **Lifecycle**: configure as operações a serem executadas em uma fase específica do ciclo de vida do contêiner, como Startup Command, Post-Start e Pre-Stop. Para mais detalhes, consulte **Definição dos parâmetros do ciclo de vida do contêiner**.
- (Opcional) **Health Check**: configure a sonda de vivacidade, a sonda pronta e a sonda de inicialização conforme necessário. Para mais detalhes, consulte **Configuração da verificação de integridade para um contêiner**.
- (Opcional) **Environment Variables**: configure variáveis para o ambiente em execução do contêiner usando pares chave-valor. Essas variáveis transferem informações externas para contêineres executados em pods e podem ser modificadas de forma flexível após a implementação da aplicação. Para mais detalhes, consulte **Configuração de uma variável de ambiente**.
- (Opcional) **Data Storage**: monte armazenamento local ou armazenamento em nuvem no contêiner. Os cenários de aplicações e os modos de montagem variam de acordo com o tipo de armazenamento. Para mais detalhes, consulte **Armazenamento**.
- (Opcional) **Security Context**: atribua permissões de contêiner para proteger o sistema e outros contêineres de serem afetados. Insira o ID do usuário para atribuir permissões de contêiner e impedir que sistemas e outros contêineres sejam afetados.
- (Opcional) **Logging**: relate os logs de saída de contêiner padrão para o AOM por padrão, sem a necessidade de configurações manuais. Você pode configurar manualmente o caminho da coleção de logs. Para mais detalhes, consulte **Uso do ICAgent para coletar logs de contêiner**.

Para desativar a saída padrão da carga de trabalho atual, adicione a anotação **kubernetes.AOM.log.stdout: []** em **Rótulos e anotações**. Para obter detalhes sobre como usar essa anotação, consulte **Tabela 5-16**.

- **Image Access Credential**: selecione a credencial usada para acessar o repositório de imagens. O valor padrão é **default-secret**. Você pode usar default-secret para acessar imagens no SWR. Para obter detalhes sobre **default-secret**, consulte **default-secret**.
- (Opcional) **GPU**: **All** é selecionado por padrão. A instância da carga de trabalho será agendada para o nó do tipo de GPU especificado.

(Opcional) Service Settings

Um Serviço fornece acesso externo para pods. Com um endereço IP estático, um Serviço encaminha o tráfego de acesso aos pods e equilibra automaticamente a carga desses pods.

Você também pode criar um Serviço depois de criar uma carga de trabalho. Para obter detalhes sobre Serviços de diferentes tipos, consulte **Visão geral**.

(Opcional) Advanced Settings

- **Upgrade:** especifique o modo de atualizar e os parâmetros de upgrade da carga de trabalho. **Rolling upgrade** e **Replace upgrade** são suportados. Para mais detalhes, consulte **Configuração da política de atualização da carga de trabalho**.
- **Scheduling**: configure políticas de afinidade e antiafinidade para agendamento flexível de carga de trabalho. Afinidade de nó, afinidade de pod e antiafinidade de pod são suportadas. Para mais detalhes, consulte **Política de agendamento (afinidade/antiafinidade)**.
- **Toleration**: o uso de ambas manchas e tolerâncias permite (não forçosamente) que o pod seja agendado para um nó com as manchas correspondentes e controla as políticas de despejo de pod depois que o nó onde o pod está localizado é manchado. Para mais detalhes, consulte **Manchas e tolerâncias**.
- **Labels and Annotations**: adicione rótulos ou anotações para pods usando pares chave-valor. Depois de inserir a chave e o valor, clique em **Confirm**. Para obter detalhes sobre como usar e configurar rótulos e anotações, consulte **Rótulos e anotações**.
- **DNS**: configure uma política DNS separada para a carga de trabalho. Para mais detalhes, consulte **Configuração de DNS**.
- **APM Settings**: use o Application Performance Management (APM) para fornecer uma análise de problemas e localização mais precisas para programas Java. Para mais detalhes, consulte **Configuração de configurações de APM para análise de gargalo de desempenho**.
- **Network configuration:**
  - limitação de largura de banda de entrada/saída do pod: Você pode definir a limitação de largura de banda de entrada/saída para pods. Para mais detalhes, consulte **Configuração da limitação da taxa de QoS para acesso entre pods**.

Passo 4 Clique em **Create Workload** no canto inferior direito.

----Fim

Usar o kubectl

O procedimento a seguir usa o Nginx como exemplo para descrever como criar uma carga de trabalho usando o kubectl.

Passo 1 Use o kubectl para se conectar ao cluster. Para mais detalhes, consulte **Conexão a um cluster usando o kubectl**.

Passo 2 Crie e edite o arquivo **nginx-daemonset.yaml**. **nginx-daemonset.yaml** é um nome de arquivo de exemplo, e você pode alterá-lo conforme necessário.

vi nginx-daemonset.yaml

O conteúdo do arquivo de descrição é o seguinte: o seguinte fornece um exemplo. Para obter mais informações sobre DaemonSets, consulte **Documentos do Kubernetes**.

```
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: nginx-daemonset
  labels:
    app: nginx-daemonset
spec:
  selector:
    matchLabels:
      app: nginx-daemonset
  template:
    metadata:
      labels:
        app: nginx-daemonset
    spec:
      nodeSelector:                # Node selection. A pod is created on a node
only when the node meets daemon=need.
        daemon: need
      containers:
      - name: nginx-daemonset
        image: nginx:alpine
        resources:
          limits:
            cpu: 250m
            memory: 512Mi
          requests:
            cpu: 250m
            memory: 512Mi
      imagePullSecrets:
      - name: default-secret
```

O parâmetro **replicas** usado na definição de uma Implementação ou StatefulSet não existe na configuração acima para um DaemonSet porque cada nó tem apenas uma réplica. Ele é fixo.

O nodeSelector no modelo de pod anterior especifica que um pod é criado somente nos nós que atendem a **daemon=need**, conforme mostrado na figura a seguir. Se você quiser criar um pod em cada nó, exclua o rótulo.

Passo 3 Crie um DaemonSet.

kubectl create -f nginx-daemonset.yaml

Se as informações a seguir forem exibidas, o DaemonSet está sendo criado.

```
daemonset.apps/nginx-daemonset created
```

Passo 4 Consulte o status do DaemonSet.

kubectl get ds

```
$ kubectl get ds
NAME              DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE
SELECTOR   AGE
nginx-daemonset   1         1         0       1            0
daemon=need     116s
```


Passo 5 Se a carga de trabalho for acessada por meio de um Serviço ClusterIP ou NodePort, defina o tipo de acesso à carga de trabalho correspondente. Para mais detalhes, consulte **Rede**.

----Fim

5.2.4 Criação de uma tarefa

Cenário

As tarefas são de curta duração e correm por um certo tempo até a conclusão. Elas podem ser executados imediatamente após serem implementados. É concluído depois de sair normalmente (exit 0).

Uma tarefa é um objeto de recurso usado para controlar tarefas em lote. É diferente de uma carga de trabalho servo de longo prazo (como Implementação e StatefulSet).

Uma tarefa é iniciada e terminada em horários específicos, enquanto uma carga de trabalho servo de longo prazo é executada incessantemente, a menos que seja terminada. Os pods gerenciados por uma tarefa saem automaticamente após a conclusão bem-sucedida da tarefa com base nas configurações do usuário. O sinalizador de sucesso varia de acordo com a política spec.completions.

- Tarefas únicas: um único pod é executado uma vez até o encerramento bem-sucedido.
- Empregos com uma contagem fixa de sucesso: os N pods são executados até o término bem-sucedido.
- Uma tarefa de fila é considerada concluída com base no sucesso global confirmado pela aplicação.

Pré-requisitos

Os recursos foram criados. Para mais detalhes, consulte **Criação de um nó**. Se clusters e nós estiverem disponíveis, não será necessário criá-los novamente.

Usar o console do CCE

Passo 1 Efetue logon no console do CCE.

Passo 2 Clique no nome do cluster para acessar o console do cluster, escolha **Workloads** no painel de navegação e clique em **Create Workload** no canto superior direito.

Passo 3 Defina informações básicas sobre a carga de trabalho.

Basic Info

- **Workload Type**: selecione **Job**. Para obter detalhes sobre os tipos de carga de trabalho, consulte **Visão geral**.
- **Workload Name**: insira o nome da carga de trabalho. Digite de 1 a 63 caracteres começando com uma letra minúscula e terminando com uma letra minúscula ou dígito. Somente letras minúsculas, dígitos e hifens (-) são permitidos.
- **Namespace**: selecione o namespace da carga de trabalho. O valor padrão é **default**. Você também pode clicar em **Create Namespace** para criar um. Para mais detalhes, consulte **Criação de um namespace**.
- **Pods**: digite o número de pods da carga de trabalho.

- **Container Runtime**: um cluster do CCE usa runC por padrão, enquanto um cluster do CCE Turbo suporta runC e Kata. Para obter detalhes sobre as diferenças, consulte **Tempo de execução do Kata e tempo de execução comum**.

Container Settings

- Informações sobre o contêiner

  Vários contêineres podem ser configurados em um pod. Você pode clicar em **Add Container** à direita para configurar vários contêineres para o pod.

  – **Basic Info**: configure informações básicas sobre o contêiner.

Parâmetro	Descrição
Container Name	Nomeie o contêiner.
Pull Policy	Política de atualização ou extração de imagem. Se você selecionar **Always**, a imagem será extraída do repositório de imagens a cada vez. Se você não selecionar **Always**, a imagem existente do nó é usada preferencialmente. Se a imagem não existir, a imagem será extraída do repositório de imagens.
Image Name	Clique em **Select Image** e selecione a imagem usada pelo contêiner. Para usar uma imagem de terceiros, consulte **Uso de imagens de terceiros**.
Image Tag	Selecione a tag de imagem a ser implementada.
CPU Quota	■ **Request**: número mínimo de núcleos de CPU exigidos por um contêiner. O valor padrão é 0,25 núcleos. ■ **Limit**: número máximo de núcleos de CPU disponíveis para um contêiner. Não deixe **Limit** não especificado. Caso contrário, ocorrerá um uso intensivo de recursos de contêiner e sua carga de trabalho poderá exibir um comportamento inesperado. Se **Request** e **Limit** não forem especificados, a cota não é limitada. Para obter mais informações e sugestões sobre **Request** e **Limit**, consulte **Configuração de especificações do contêiner**.
Memory Quota	■ **Request**: quantidade mínima de memória requerida por um contêiner. O valor padrão é 512 MiB. ■ **Limit**: quantidade máxima de memória disponível para um contêiner. Quando o uso de memória exceder o limite de memória especificado, o contêiner será encerrado. Se **Request** e **Limit** não forem especificados, a cota não é limitada. Para obter mais informações e sugestões sobre **Request** e **Limit**, consulte **Configuração de especificações do contêiner**.

Parâmetro	Descrição
(Optional) GPU Quota	Configurável somente quando o cluster contém nós de GPU e o complemento **Suíte IA do CCE (GPU NVIDIA)** está instalado. ■ **All**: nenhuma GPU será usada. ■ **Dedicated**: os recursos da GPU são dedicados ao contêiner. ■ **Shared**: porcentagem de recursos de GPU usados pelo contêiner. Por exemplo, se esse parâmetro for definido como **10%**, o contêiner usará 10% dos recursos da GPU. Para obter detalhes sobre como usar GPUs no cluster, consulte **Agendamento de GPU padrão no Kubernetes**.
(Optional) NPU Quota	Número de chips de Ascend 310 necessários para o contêiner. O valor deve ser um número inteiro e o complemento **Suíte de IA do CCE (Ascend NPU)** deve ser instalado. Para obter detalhes sobre como usar NPUs no cluster, consulte **Agendamento de NPU**.
(Optional) Privileged Container	Programas em um contêiner privilegiado têm certos privilégios. Se **Privileged Container** estiver habilitado, os privilégios serão atribuídos ao contêiner. Por exemplo, contêineres privilegiados podem manipular dispositivos de rede na máquina host e modificar parâmetros do kernel.
(Optional) Init Container	Se usar o contêiner como um contêiner init. Um contêiner init não suporta verificação de integridade. Um contêiner init é um contêiner especial que é executado antes que outros contêineres de aplicações em um pod sejam iniciados. Cada pod pode conter vários contêineres. Além disso, um pod pode conter um ou mais contêineres init. Os contêineres de aplicações em um pod são iniciados e executados somente após a conclusão da execução de todos os contêineres init. Para obter detalhes, consulte **Contêineres init**.

– (Opcional) **Lifecycle**: configure as operações a serem executadas em uma fase específica do ciclo de vida do contêiner, como Startup Command, Post-Start e Pre-Stop. Para mais detalhes, consulte **Definição dos parâmetros do ciclo de vida do contêiner**.

– (Opcional) **Environment Variables**: configure variáveis para o ambiente em execução do contêiner usando pares chave-valor. Essas variáveis transferem informações externas para contêineres executados em pods e podem ser modificadas de forma flexível após a implementação da aplicação. Para mais detalhes, consulte **Configuração de uma variável de ambiente**.

– (Opcional) **Data Storage**: monte armazenamento local ou armazenamento em nuvem no contêiner. Os cenários de aplicações e os modos de montagem variam de

acordo com o tipo de armazenamento. Para mais detalhes, consulte **Armazenamento**.

NOTA

Se a carga de trabalho contiver mais de um pod, os volumes do EVS não poderão ser montados.

- (Opcional) **Logging**: relate os logs de saída de contêiner padrão para o AOM por padrão, sem a necessidade de configurações manuais. Você pode configurar manualmente o caminho da coleção de logs. Para mais detalhes, consulte **Uso do ICAgent para coletar logs de contêiner**.

  Para desativar a saída padrão da carga de trabalho atual, adicione a anotação **kubernetes.AOM.log.stdout: []** em **Rótulos e anotações**. Para obter detalhes sobre como usar essa anotação, consulte **Tabela 5-16**.

- **Image Access Credential**: selecione a credencial usada para acessar o repositório de imagens. O valor padrão é **default-secret**. Você pode usar default-secret para acessar imagens no SWR. Para obter detalhes sobre **default-secret**, consulte **default-secret**.
- (Opcional) **GPU**: **All** é selecionado por padrão. A instância da carga de trabalho será agendada para o nó do tipo de GPU especificado.

(Opcional) Advanced Settings

- **Labels and Annotations**: adicione rótulos ou anotações para pods usando pares chave-valor. Depois de inserir a chave e o valor, clique em **Confirm**. Para obter detalhes sobre como usar e configurar rótulos e anotações, consulte **Rótulos e anotações**.
- **Job Settings**
  - **Parallel Pods**: número máximo de pods que podem ser executados em paralelo durante a execução da tarefa. O valor não pode ser maior que o número total de pods na tarefa.
  - **Timeout (s)**: uma vez que uma tarefa atinge este tempo, o status da tarefa torna-se com falha e todos os pods nesta tarefa serão excluídos. Se você deixar esse parâmetro em branco, a tarefa nunca expirará.
  - Modo de conclusão
    - **Non-indexed**: uma tarefa é considerada concluída quando todos os pods são executados com sucesso. Cada complementação de vagem é homóloga uma à outra.
    - **Indexed**: cada pod recebe um índice de conclusão associado de 0 para o número de pods menos 1. A tarefa é considerada concluída quando cada pod alocado com um índice é executado com êxito. Para uma tarefa indexada, os pods são nomeados no formato $(job-name)-$(index).
  - **Suspend Job**: por padrão, uma tarefa é executada imediatamente após ser criada. A execução da tarefa será suspensa se você ativar essa opção e retomada após desativá-la.
- **Network configuration:**
  - limitação de largura de banda de entrada/saída do pod: Você pode definir a limitação de largura de banda de entrada/saída para pods. Para mais detalhes, consulte **Configuração da limitação da taxa de QoS para acesso entre pods**.

Passo 4 Clique em **Create Workload** no canto inferior direito.

----Fim

Usar o kubectl

Uma tarefa tem os seguintes parâmetros de configuração:

- **spec.template**: tem o mesmo esquema que um pod.
- **RestartPolicy**: só pode ser definido como **Never** ou **OnFailure**.
- Para uma tarefa de pod único, a tarefa termina depois que o pod é executado com êxito por padrão.
- **.spec.completions**: indica o número de pods que precisam ser executados com êxito para finalizar uma tarefa. O valor padrão é **1**.
- **.spec.parallelism**: indica o número de pods que são executados simultaneamente. O valor padrão é **1**.
- **spec.backoffLimit**: indica o número máximo de tentativas realizadas se um pod falhar. Quando o limite for atingido, o pod não tentará novamente.
- **.spec.activeDeadlineSeconds**: indica o tempo de execução dos pods. Uma vez que o tempo é alcançado, todos os pods do trabalho são encerrados. A prioridade de .spec.activeDeadlineSeconds é maior que a de .spec.backoffLimit. Ou seja, se uma tarefa atingir o .spec.activeDeadlineSeconds, o spec.backoffLimit será ignorado.

Com base nas configurações **.spec.completions** e **.spec.Parallelism**, as tarefas são classificadas nos seguintes tipos.

Tabela 5-3 Tipo da tarefa

Tipo da tarefa	Descrição	Exemplo
Tarefas únicas	Um único pod é executado uma vez até o encerramento bem-sucedido.	Migração de banco de dados
Tarefas com uma contagem de conclusão fixa	Um pod é executado até atingir a contagem de **completions** especificada.	Pod de processamento de fila de tarefa
Tarefas paralelas com uma contagem de conclusão fixa	Vários pods são executados até atingir a contagem de **completions** especificada.	Vários pods para processamento simultâneo de filas de tarefa
Tarefas paralelas	Um ou mais pods são executados até a rescisão bem-sucedida.	Vários pods para processamento simultâneo de filas de tarefa

O seguinte é um exemplo de tarefa, que calcula o número até o 2000º dígito e imprime a saída.

```
apiVersion: batch/v1
kind: Job
metadata:
  name: myjob
spec:
  completions: 50        # 50 pods need to be run to finish a job. In this
example, π is printed for 50 times.
  parallelism: 5        # 5 pods are run in parallel.
  backoffLimit: 5        # The maximum number of retry times is 5.
```



```
  template:
    spec:
      containers:
      - name: pi
        image: perl
        command: ["perl",  "-Mbignum=bpi", "-wle", "print bpi(2000)"]
      restartPolicy: Never
      imagePullSecrets:
        - name: default-secret
```

Descrição

- **apiVersion: batch/v1** indica a versão da tarefa atual.
- **kind: Job** indica que o recurso atual é uma tarefa.
- **restartPolicy: Never** indica a política de reinicialização atual. Para tarefas, esse parâmetro só pode ser definido como **Never** ou **OnFailure**. Para outros controladores (por exemplo, Implementações), você pode ajustar este parâmetro a **Always**.

Execute a tarefa.

Passo 1 Comece a tarefa.

```
[root@k8s-master k8s]# kubectl apply -f myjob.yaml
job.batch/myjob created
```

Passo 2 Visualize os detalhes da tarefa.

kubectl get job

```
[root@k8s-master k8s]# kubectl get job
NAME    COMPLETIONS   DURATION   AGE
myjob   50/50         23s        3m45s
```

Se o valor de **COMPLETIONS** for **50/50**, a tarefa será executada com êxito.

Passo 3 Consulte o status do pod.

kubectl get pod

```
[root@k8s-master k8s]# kubectl get pod
NAME          READY   STATUS      RESTARTS   AGE
myjob-29qlw   0/1     Completed   0          4m5s
...
```

Se o status for **Completed**, a tarefa será concluída.

Passo 4 Veja os logs do pod.

kubectl logs

```
# kubectl logs myjob-29qlw
3.14159265358979323846264338327950288419716939937510582097494459230781640628620899
86280348253421170679821480865132823066470938446095505822317253594081284811174502 84
10270193852110555964462294895493038196442881097566593344612847564823378678316527 12
01909145648566923460348610454326648213393607260249141273724587006606315588174881 52
09209628292540917153643678925903600113305305488204665213841469519415116094330572 70
36575959195309218611738193261179310511854807446237996274956735188575272489122793 81
83011949129833673362440656643086021394946395224737190702179860943702770539217176 29
31767523846748184676694051320005681271452635608277857713427577896091736371787214 68
44090122495343014654958537105079227968925892354201995611212902196086403441815981 36
29774771309960518707211349999998372978049951059731732816096318595024459455346908 30
26425223082533446850352619311881710100031378387528865875332083814206171776691473 03
59825349042875546873115956286388235378759375195778185778053217122680661300192787 66
11195909216420198938095257201065485863278865936153381827968230301952035301852968 99
57736225994138912497217752834791315155748572424541506959508295331168617278558890 75
09838175463746493931925506040092770167113900984882401285836160356370766010471018 19
42955596198946767837449448255379774726847104047534646208046684259069491293313677 02
```



```
898915210475216205696602405803815019351125338243003558764024749647326391419927260 4
269922796782354781636009341721641219924586315030286182974555706749838505494588586 9
269956909272107975093029553211653449872027559602364806654991198818347977535663698 0
742654252786255181841757467289097777279380008164706001614524919217321721477235014 1
441973568548161361157352552133475741849468438523323907394143334547762416862518983 5
694855620992192221842725502542568876717904946016534668049886272327917860857843838 2
796797668145410095388378636095068006422512520511739298489608412848862694560424196 5
285022210661186306744278622039194945047123713786960956364371917287467764657573962 4
138908658326459958133904780275901
```

----Fim

Operações relacionadas

Depois que uma tarefa única é criada, você pode executar as operações listadas em **Tabela 5-4**.

Tabela 5-4 Outras operações

Operação	Descrição
Editar um arquivo YAML	Clique em **More** > **Edit YAML** ao lado do nome da tarefa para editar o arquivo YAML correspondente à tarefa atual.
Excluir uma tarefa	1. Selecione uma tarefa a ser excluída e clique em **Delete** na coluna **Operation**. 2. Clique em **Yes**. Tarefas excluídas não podem ser recuperadas. Tenha cuidado ao excluir uma tarefa.

5.2.5 Criação de uma tarefa cronometrada

Cenário

Uma tarefa cronometrada é executada em um cronograma repetido. Você pode executar a sincronização de tempo para todos os nós ativos em um ponto de tempo fixo.

Uma tarefa cronometrada é executada periodicamente no horário especificado. É semelhante com o crontab do Linux. Uma tarefa cronometrada tem as seguintes características:

- É executada apenas uma vez no horário especificado.
- É executada periodicamente no horário especificado.

O uso típico de uma tarefa cronometrada é o seguinte:

- Agenda as tarefas cronometradas no horário especificado.
- Cria tarefas para serem executadas periodicamente, por exemplo, backup de banco de dados e envio de e-mail.

Pré-requisitos

Os recursos foram criados. Para mais detalhes, consulte **Criação de um nó**.

Usar o console do CCE

Passo 1 Efetue logon no console do CCE.

Passo 2 Clique no nome do cluster para acessar o console do cluster, escolha **Workloads** no painel de navegação e clique em **Create Workload** no canto superior direito.

Passo 3 Defina informações básicas sobre a carga de trabalho.

Basic Info

- **Workload Type**: selecione **Cron Job**. Para obter detalhes sobre os tipos de carga de trabalho, consulte **Visão geral**.
- **Workload Name**: insira o nome da carga de trabalho. Digite de 1 a 63 caracteres começando com uma letra minúscula e terminando com uma letra minúscula ou dígito. Somente letras minúsculas, dígitos e hifens (-) são permitidos.
- **Namespace**: selecione o namespace da carga de trabalho. O valor padrão é **default**. Você também pode clicar em **Create Namespace** para criar um. Para mais detalhes, consulte **Criação de um namespace**.
- **Container Runtime**: um cluster do CCE usa runC por padrão, enquanto um cluster do CCE Turbo suporta runC e Kata. Para obter detalhes sobre as diferenças, consulte **Tempo de execução do Kata e tempo de execução comum**.

Container Settings

- Informações sobre o contêiner

  Vários contêineres podem ser configurados em um pod. Você pode clicar em **Add Container** à direita para configurar vários contêineres para o pod.

  – **Basic Info**: configure informações básicas sobre o contêiner.

Parâmetro	Descrição
Container Name	Nomeie o contêiner.
Pull Policy	Política de atualização ou extração de imagem. Se você selecionar **Always**, a imagem será extraída do repositório de imagens a cada vez. Se você não selecionar **Always**, a imagem existente do nó é usada preferencialmente. Se a imagem não existir, a imagem será extraída do repositório de imagens.
Image Name	Clique em **Select Image** e selecione a imagem usada pelo contêiner. Para usar uma imagem de terceiros, consulte **Uso de imagens de terceiros**.
Image Tag	Selecione a tag de imagem a ser implementada.

Parâmetro	Descrição
CPU Quota	■ **Request**: número mínimo de núcleos de CPU exigidos por um contêiner. O valor padrão é 0,25 núcleos. ■ **Limit**: número máximo de núcleos de CPU disponíveis para um contêiner. Não deixe **Limit** não especificado. Caso contrário, ocorrerá um uso intensivo de recursos de contêiner e sua carga de trabalho poderá exibir um comportamento inesperado. Se **Request** e **Limit** não forem especificados, a cota não é limitada. Para obter mais informações e sugestões sobre **Request** e **Limit**, consulte **Configuração de especificações do contêiner**.
Memory Quota	■ **Request**: quantidade mínima de memória requerida por um contêiner. O valor padrão é 512 MiB. ■ **Limit**: quantidade máxima de memória disponível para um contêiner. Quando o uso de memória exceder o limite de memória especificado, o contêiner será encerrado. Se **Request** e **Limit** não forem especificados, a cota não é limitada. Para obter mais informações e sugestões sobre **Request** e **Limit**, consulte **Configuração de especificações do contêiner**.
(Optional) GPU Quota	Configurável somente quando o cluster contém nós de GPU e o complemento **Suíte IA do CCE (GPU NVIDIA)** está instalado. ■ **All**: nenhuma GPU será usada. ■ **Dedicated**: os recursos da GPU são dedicados ao contêiner. ■ **Shared**: porcentagem de recursos de GPU usados pelo contêiner. Por exemplo, se esse parâmetro for definido como **10%**, o contêiner usará 10% dos recursos da GPU. Para obter detalhes sobre como usar GPUs no cluster, consulte **Agendamento de GPU padrão no Kubernetes**.
(Optional) NPU Quota	Número de chips de Ascend 310 necessários para o contêiner. O valor deve ser um número inteiro e o complemento **Suíte de IA do CCE (Ascend NPU)** deve ser instalado. Para obter detalhes sobre como usar NPUs no cluster, consulte **Agendamento de NPU**.
(Optional) Privileged Container	Programas em um contêiner privilegiado têm certos privilégios. Se **Privileged Container** estiver habilitado, os privilégios serão atribuídos ao contêiner. Por exemplo, contêineres privilegiados podem manipular dispositivos de rede na máquina host e modificar parâmetros do kernel.

Parâmetro	Descrição
(Optional) Init Container	Se usar o contêiner como um contêiner init. Um contêiner init não suporta verificação de integridade. Um contêiner init é um contêiner especial que é executado antes que outros contêineres de aplicações em um pod sejam iniciados. Cada pod pode conter vários contêineres. Além disso, um pod pode conter um ou mais contêineres init. Os contêineres de aplicações em um pod são iniciados e executados somente após a conclusão da execução de todos os contêineres init. Para obter detalhes, consulte **Contêineres init**.

  - (Opcional) **Lifecycle**: configure as operações a serem executadas em uma fase específica do ciclo de vida do contêiner, como Startup Command, Post-Start e Pre-Stop. Para mais detalhes, consulte **Definição dos parâmetros do ciclo de vida do contêiner**.
  - (Opcional) **Environment Variables**: configure variáveis para o ambiente em execução do contêiner usando pares chave-valor. Essas variáveis transferem informações externas para contêineres executados em pods e podem ser modificadas de forma flexível após a implementação da aplicação. Para mais detalhes, consulte **Configuração de uma variável de ambiente**.
- **Image Access Credential**: selecione a credencial usada para acessar o repositório de imagens. O valor padrão é **default-secret**. Você pode usar default-secret para acessar imagens no SWR. Para obter detalhes sobre **default-secret**, consulte **default-secret**.
- (Opcional) **GPU**: **All** é selecionado por padrão. A instância da carga de trabalho será agendada para o nó do tipo de GPU especificado.

Schedule

- **Concurrency Policy**: os seguintes três modos são suportados:
  - **Forbid**: uma nova tarefa não pode ser criada antes que a tarefa anterior seja concluída.
  - **Allow**: a tarefa cronometrada permite tarefas em execução simultânea, que antecipa recursos de cluster.
  - **Replace**: uma nova tarefa substitui a tarefa anterior quando é hora de criar uma tarefa, mas a tarefa anterior não é concluída.
- **Policy Settings**: especifica quando uma nova tarefa cronometrada é executada. As configurações de política no YAML são implementadas usando expressões cron.
  - Uma tarefa cronometrada é executada em um intervalo fixo. A unidade pode ser minuto, hora, dia ou mês. Por exemplo, se uma tarefa cronometrada é executada a cada 30 minutos e a expressão cron correspondente é ***/30 * * * ***, o tempo de execução começa a partir de 0 no intervalo de unidades, por exemplo, **00:00:00**, **00:30:00**, **01:00:00** e **...**.
  - A tarefa cronometrada é executada em um horário fixo (por mês). Por exemplo, se uma tarefa cronometrada é executada às 00:00 no primeiro dia de cada mês, a expressão cron é **0 0 1 */1 ***, e o tempo de execução é ******-01-01 00:00:00**, ******-02-01 00:00:00** e **...**.
  - A tarefa cronometrada é executada por semana. Por exemplo, se uma tarefa cronometrada é executada às 00:00 todas as segundas-feiras, a expressão cron é **0 0**

*** * 1**, e o tempo de execução é ******-**-01 00:00:00 on Monday**, ******-**-08 00:00:00 on Monday** e **...**.

– **Custom Cron Expression**: para obter detalhes sobre como usar expressões cron, consulte **CronJob**.

NOTA

– Se uma tarefa cronometrada for executada em um horário fixo (por mês) e o número de dias em um mês não existir, a tarefa cronometrada não será executada neste mês. Por exemplo, a execução saltará fevereiro se a data estiver definida como 30.

– Devido à definição de cron, o período fixo não é um período rigoroso. O intervalo de unidades de tempo é dividido de 0 por período. Por exemplo, se a unidade é minuto, o valor varia de 0 a 59. Se o valor não puder ser dividido exatamente, o último período será redefinido. Portanto, um período preciso pode ser representado somente quando o período pode ser dividido igualmente.

Tome como exemplo uma tarefa cronometrada que é executada por hora. Como **/2, /3, /4, /6, /8 e /12** podem dividir exatamente 24 horas, um período preciso pode ser representado. Se outro período for usado, o último período será reiniciado no início de um novo dia. Por exemplo, se a expressão cron é *** */12 * * ***, o tempo de execução é **00:00:00** e **12:00:00** todos os dias. Se a expressão cron é *** */13 * * ***, o tempo de execução é **00:00:00** e **13:00:00** todos os dias. Às 00:00 do dia seguinte, o tempo de execução é atualizado mesmo que o período não atinja 13 horas.

- **Job Records**: você pode definir o número de tarefas que são executadas com êxito ou que não são executadas. Definir um limite para **0** corresponde a manter nenhum dos trabalhos depois de concluídos.

(Opcional) Advanced Settings

- **Labels and Annotations**: adicione rótulos ou anotações para pods usando pares chave-valor. Depois de inserir a chave e o valor, clique em **Confirm**. Para obter detalhes sobre como usar e configurar rótulos e anotações, consulte **Rótulos e anotações**.
- **Network configuration:**
  – limitação de largura de banda de entrada/saída do pod: Você pode definir a limitação de largura de banda de entrada/saída para pods. Para mais detalhes, consulte **Configuração da limitação da taxa de QoS para acesso entre pods**.

Passo 4 Clique em **Create Workload** no canto inferior direito.

----Fim

Usar o kubectl

Uma tarefa cronometrada tem os seguintes parâmetros de configuração:

- **.spec.schedule**: leva uma cadeia no formato **Cron**, por exemplo, **0 * * * *** ou **@hourly**, como horário de programação das tarefas a serem criadas e executadas.
- **.spec.jobTemplate**: especifica as tarefas a serem executadas e tem o mesmo esquema de quando você está em **Criação de uma tarefa usando kubectl**.
- **.spec.startingDeadlineSeconds**: especifica o prazo para iniciar uma tarefa.
- **.spec.concurrencyPolicy**: especifica como tratar execuções simultâneas de uma tarefa criada pela tarefa cronometrada. As seguintes opções são compatíveis:
  – **Allow** (valor padrão): permite tarefas em execução simultânea.
  – **Forbid**: proíbe corridas simultâneas, pulando a próxima execução se a anterior ainda não tiver terminado.

- **Replace**: cancela a tarefa em execução no momento e a substitui por uma nova.

Segue-se um exemplo de tarefa cronometrada, que é guardado no ficheiro **cronjob.yaml**.

NOTA

Em clusters v1.21 ou posterior, CronJob apiVersion é **batch/v1**.

Em clusters anteriores à v1.21, CronJob apiVersion é **batch/v1beta1**.

```
apiVersion: batch/v1
kind: CronJob
metadata:
  name: hello
spec:
  schedule: "*/1 * * * *"
  jobTemplate:
    spec:
      template:
        spec:
          containers:
          - name: hello
            image: busybox
            command:
            - /bin/sh
            - -c
            - date; echo Hello from the Kubernetes cluster
          restartPolicy: OnFailure
          imagePullSecrets:
            - name: default-secret
```

Execute a tarefa.

Passo 1 Crie uma tarefa cronometrada.

kubectl create -f cronjob.yaml

Informação semelhante à seguinte é exibida:

```
cronjob.batch/hello created
```

Passo 2 Consulte o status de execução da tarefa cronometrada:

kubectl get cronjob

```
NAME      SCHEDULE      SUSPEND   ACTIVE    LAST SCHEDULE   AGE
hello     */1 * * * *   False     0         <none>          9s
```

kubectl get jobs

```
NAME               COMPLETIONS   DURATION   AGE
hello-1597387980   1/1           27s        45s
```

kubectl get pod

```
NAME                             READY     STATUS      RESTARTS   AGE
hello-1597387980-tjv8f           0/1       Completed   0          114s
hello-1597388040-lckg9           0/1       Completed   0          39s
```

kubectl logs hello-1597387980-tjv8f

```
Fri Aug 14 06:56:31 UTC 2020
Hello from the Kubernetes cluster
```

kubectl delete cronjob hello

```
cronjob.batch "hello" deleted
```


AVISO

Quando uma CronJob é excluída, as tarefas e pods relacionados são excluídos de acordo.

----Fim

Operações relacionadas

Depois que uma CronJob é criada, você pode executar as operações listadas em **Tabela 5-5**.

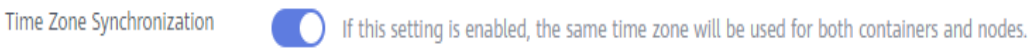
Tabela 5-5 Outras operações

Operação	Descrição
Editar um arquivo YAML	Clique em **More** > **Edit YAML** ao lado do nome da tarefa cronometrada para editar o arquivo YAML da tarefa cronometrada atual.
Parar uma CronJob	1. Selecione a tarefa a ser excluída e clique em **Stop** na coluna **Operation**. 2. Clique em **Yes**.
Excluir uma CronJob	1. Selecione a CronJob a ser excluída e clique em **More** > **Delete** na coluna **Operation**. 2. Clique em **Yes**. Tarefas excluídas não podem ser recuperadas. Portanto, tenha cuidado ao excluir uma tarefa.

5.3 Configuração de um contêiner

5.3.1 Configuração da sincronização de fuso horário

Ao criar uma carga de trabalho, você pode configurar contêineres para usar o mesmo fuso horário que o nó. Você pode habilitar a sincronização de fuso horário ao criar uma carga de trabalho.



A função de sincronização de fuso horário depende do disco local (hostPath) montado no contêiner. Depois que a sincronização de fuso horário é habilitada, **/etc/localtime** do nó é montado para **/etc/localtime** do contêiner em modo HostPath, desta forma, o nó e o contêiner usam o mesmo arquivo de configuração de fuso horário.

```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: test
  namespace: default
spec:
```



```
  replicas: 2
  selector:
    matchLabels:
      app: test
  template:
    metadata:
      labels:
        app: test
    spec:
      volumes:
        - name: vol-162979628557461404
          hostPath:
            path: /etc/localtime
            type: ''
      containers:
        - name: container-0
          image: 'nginx:alpine'
          volumeMounts:
            - name: vol-162979628557461404
              readOnly: true
              mountPath: /etc/localtime
          imagePullPolicy: IfNotPresent
      imagePullSecrets:
        - name: default-secret
```

5.3.2 Configuração de uma política de extração de imagem

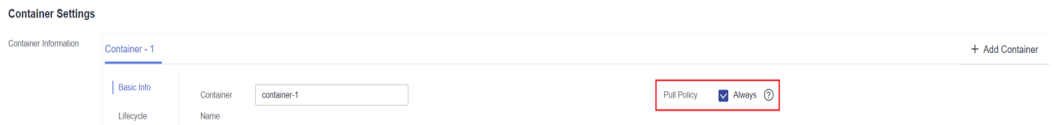
Quando uma carga de trabalho é criada, a imagem de contêiner é extraída do repositório de imagens para o nó. A imagem também é extraída quando a carga de trabalho é reiniciada ou atualizada.

Por padrão, **imagePullPolicy** é definido como **IfNotPresent** indicando que, se a imagem existir no nó, a imagem existente será usada. Se a imagem não existir no nó, a imagem será extraída do repositório de imagens.

A política de extração de imagem também pode ser definida como **Always**, indicando que a imagem é extraída do repositório de imagens e substitui a imagem no nó, independentemente de a imagem existir no nó.

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - image: nginx:alpine
    name: container-0
    resources:
      limits:
        cpu: 100m
        memory: 200Mi
      requests:
        cpu: 100m
        memory: 200Mi
    imagePullPolicy: Always
  imagePullSecrets:
  - name: default-secret
```

Você também pode definir a política de extração de imagem ao criar uma carga de trabalho no console do CCE. Se selecionar **Always**, a imagem é sempre extraída. Se você não selecioná-la, a política será **IfNotPresent**, o que significa que a imagem não é extraída.



AVISO

Recomendamos que você use uma nova tag toda vez que criar uma imagem. Se você não atualizar a tag, mas apenas atualizar a imagem, quando **Pull Policy** é definida como **IfNotPresent**, o CCE considera que uma imagem com a tag já existe no nó atual e não a extrairá novamente.

5.3.3 Uso de imagens de terceiros

Cenário

O CCE permite que você crie cargas de trabalho usando imagens extraídas de repositórios de imagens de terceiros.

Geralmente, um repositório de imagens de terceiros pode ser acessado somente após a autenticação (usando sua conta e senha). O CCE usa a autenticação baseada em segredo para extrair imagens. Portanto, crie um segredo para um repositório de imagens antes de extrair imagens do repositório.

Pré-requisitos

O nó onde a carga de trabalho está em execução é acessível a partir de redes públicas.

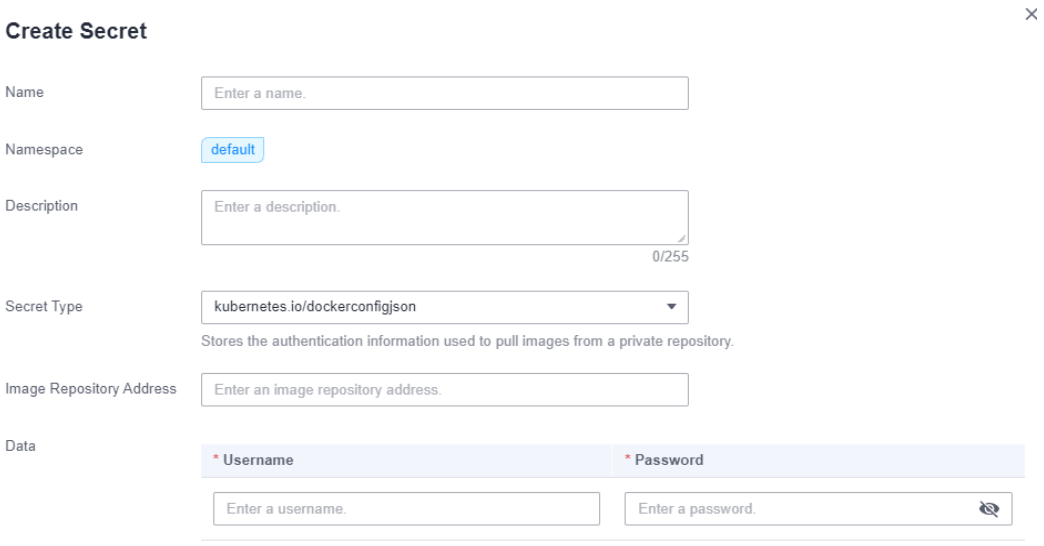
Usar o console

Passo 1 Crie um segredo para acessar um repositório de imagens de terceiros.

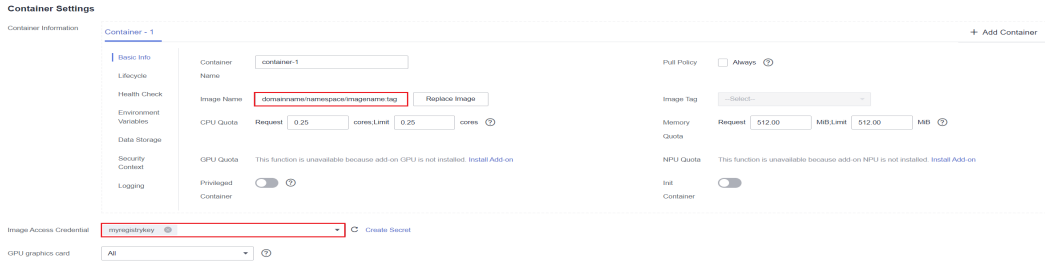
Clique no nome do cluster para acessar o console do cluster. No painel de navegação, escolha **ConfigMaps and Secrets**. Na página **Secrets**, clique em **Create Secret** no canto superior direito. Defina **Secret Type** como **kubernetes.io/dockerconfigjson**. Para mais detalhes, consulte **Criação de um segredo**.

Digite o nome de usuário e a senha usados para acessar o repositório de imagens de terceiros.

Figura 5-4 Criar um segredo



Passo 2 Ao criar uma carga de trabalho, você pode inserir um caminho de imagem privado no formato de *domainname/namespace/imagename:tag* em **Image Name** e selecionar a chave criada em **Passo 1**.



Passo 3 Defina outros parâmetros e clique em **Create Workload**.

----Fim

Usar o kubectl

Passo 1 Use o kubectl para se conectar ao cluster. Para mais detalhes, consulte **Conexão a um cluster usando o kubectl**.

Passo 2 Use o kubectl para criar um segredo do kubernetes.io/dockerconfigjson.

```
kubectl create secret docker-registry myregistrykey  -n default --docker-
server=DOCKER_REGISTRY_SERVER --docker-username=DOCKER_USER --docker-
password=DOCKER_PASSWORD --docker-email=DOCKER_EMAIL
```

No comando anterior, *myregistrykey* indica o nome da chave, *default* indica o namespace onde a chave está localizada e outros parâmetros são os seguintes:

- **DOCKER_REGISTRY_SERVER**: endereço de um repositório de imagens de terceiros, por exemplo **www.3rdregistry.com** ou **10.10.10.10:443**
- **DOCKER_USER**: conta usada para fazer logon em um repositório de imagens de terceiros
- **DOCKER_PASSWORD**: senha usada para fazer logon em um repositório de imagens de terceiros
- **DOCKER_EMAIL**: e-mail de um repositório de imagens de terceiros

Passo 3 Use uma imagem de terceiros para criar uma carga de trabalho.

Um segredo de kubernetes.io/dockerconfigjson é usado para autenticação quando você obtém uma imagem privada. Segue-se um exemplo de utilização da myregistrykey para autenticação.

```
apiVersion: v1
kind: Pod
metadata:
  name: foo
  namespace: default
spec:
  containers:
    - name: foo
      image: www.3rdregistry.com/janedoe/awesomeapp:v1
  imagePullSecrets:
    - name: myregistrykey              #Use the created secret.
```

----Fim

5.3.4 Configuração de especificações do contêiner

Cenário

O CCE permite que você defina requisitos e limites de recursos, como CPU e RAM, para contêineres adicionados durante a criação da carga de trabalho. O Kubernetes também permite o uso do YAML para definir requisitos de outros tipos de recursos.

Solicitação e limite

Para **CPU** e **Memory**, os significados de **Request** e **Limit** são os seguintes:

- **Request**: o sistema programa um pod para o nó que atende aos requisitos de implementação de carga de trabalho com base no valor da solicitação.
- **Limit**: o sistema limita os recursos utilizados pela carga de trabalho com base no valor de limite.

Se um nó tiver recursos suficientes, o pod nesse nó poderá usar mais recursos do que o solicitado, mas não mais do que limitado.

Por exemplo, se você definir a solicitação de memória de um contêiner para 1 GiB e o valor de limite para 2 GiB, um pod será programado para um nó com CPUs de 8 GiB sem nenhum outro pod em execução. Nesse caso, o pod pode usar mais de 1 GiB de memória quando a carga é pesada, mas o uso da memória não pode exceder 2 GiB. Se um processo em um contêiner tentar usar mais de 2 GiB de recursos, o kernel do sistema tentará encerrar o processo. Como resultado, ocorre um erro de falta de memória (OOM).

NOTA

Ao criar uma carga de trabalho, é aconselhável definir os limites superior e inferior dos recursos de CPU e memória. Se os limites de recursos superior e inferior não estiverem definidos para uma carga de trabalho, um vazamento de recursos dessa carga de trabalho tornará os recursos indisponíveis para outras cargas de trabalho implementadas no mesmo nó. Além disso, as cargas de trabalho que não possuem limites de recursos superiores e inferiores não podem ser monitoradas com precisão.

Descrição da configuração

Em cenários do mundo real, a proporção recomendada de **Request** para **Limit** é de cerca de 1:1,5. Para alguns serviços sensíveis, a proporção recomendada é de 1:1. Se a **Request** for muito pequena e o **Limit** for muito grande, os recursos de nó serão inscritos em excesso. Durante os picos de serviço, a memória ou a CPU de um nó podem ser usadas. Como resultado, o nó está indisponível.

- CPU quota: a unidade de recursos da CPU é o núcleo, que pode ser expresso por quantidade ou um inteiro com sufixo com a unidade (m). Por exemplo, 0,1 núcleo na expressão de quantidade é equivalente a 100m na expressão. No entanto, o Kubernetes não permite recursos de CPU cuja precisão seja inferior a 1m.

Tabela 5-6 Descrição das cotas de CPU

Parâmetro	Descrição
CPU request	Número mínimo de núcleos de CPU exigidos por um contêiner. Os recursos são agendados para o contêiner com base nesse valor. O contêiner pode ser programado para esse nó somente quando a CPU total disponível no nó for maior ou igual ao número de aplicações de CPU em contêiner.
CPU limit	Número máximo de núcleos de CPU disponíveis para um contêiner.

Configuração recomendada

CPU atual disponível de um nó ≥ soma dos limites de CPU de todos os contêineres no nó atual ≥ soma das solicitações de CPU de todos os contêineres no nó atual. Você pode ver as CPUs disponíveis reais de um nó no console do CCE (**Resource Management** > **Nodes** > **Allocatable**).

- Memory quota: a unidade padrão de recursos de memória é byte. Você também pode usar um inteiro com o sufixo unitário, por exemplo, 100 Mi. Observe que a unidade faz distinção entre maiúsculas e minúsculas.

Tabela 5-7 Descrição das cotas de memória

Parâmetro	Descrição
Memory request	Quantidade mínima de memória exigida por um contêiner. Os recursos são agendados para o contêiner com base nesse valor. O contêiner pode ser agendado para esse nó somente quando a memória total disponível no nó for maior ou igual ao número de aplicativos de memória em contêiner.
Memory Limit	Quantidade máxima de memória disponível para um contêiner. Quando o uso de memória excede o limite de memória configurado, a instância pode ser reiniciada, o que afeta o uso normal da carga de trabalho.

Configuração recomendada

Memória disponível real de um nó ≥ soma dos limites de memória de todos os contêineres no nó atual ≥ soma das solicitações de memória de todos os contêineres no nó atual. Você pode ver a memória disponível real de um nó no console do CCE (**Resource Management** > **Nodes** > **Allocatable**).

NOTA

Os recursos alocáveis são calculados com base no valor da solicitação de recurso (**Request**), que indica o limite superior de recursos que podem ser solicitados por pods neste nó, mas não indica os recursos reais disponíveis do nó (para obter detalhes, consulte **Exemplo de uso de cota de CPU e memória**). A fórmula de cálculo é a seguinte:

- CPU alocável = CPU total – CPU solicitada de todos os pods – CPU reservada para outros recursos
- Memória alocável = memória total – memória solicitada de todos os pods – memória reservada para outros recursos

Exemplo de uso de cota de CPU e memória

Suponha que um cluster contém um nó com 4 núcleos de CPU e 8 GiB de memória. Dois pods (pod 1 e pod 2) foram implementados no cluster. O pod 1 com excesso de assinaturas de recursos (ou seja, **Limit** > **Request**). As especificações dos dois pods são as seguintes.

Pod	Solicitação de CPU	Limite de CPU	Solicitação de memória	Limite de memória
Pod 1	1 núcleo	2 núcleos	1 GiB	4 GiB
Pod 2	2 núcleos	2 núcleos	2 GiB	2 GiB

O uso de CPU e memória do nó é o seguinte:

- CPUs alocáveis = 4 núcleos – (1 núcleo solicitado pelo pod 1 + 2 núcleos solicitados pelo pod 2) = 1 núcleo
- Memória alocável = 8 GiB – (1 GiB solicitado pelo pod 1 + 2 GiB solicitado pelo pod 2) = 5 GiB

Nesse caso, o 1 núcleo restante de 5 GiB pode ser usado pelo próximo novo pod.

Se o pod 1 estiver sob carga pesada durante as horas de pico, ele usará mais CPUs e memória dentro do limite. Portanto, os recursos reais alocáveis são menos de 1 núcleo de 5 GiB.

5.3.5 Definição dos parâmetros do ciclo de vida do contêiner

Cenário

O CCE fornece funções de retorno de chamada para o gerenciamento do ciclo de vida de aplicações conteinerizadas. Por exemplo, se você quiser que um contêiner execute uma determinada operação antes de parar, poderá registrar uma função de gancho.

O CCE fornece as seguintes funções de retorno de chamada do ciclo de vida:

- **Startup Command**: executado para iniciar um contêiner. Para mais detalhes, consulte **Comandos de inicialização**.
- **Post-Start**: executado imediatamente após o início de um contêiner. Para mais detalhes, consulte **Processamento pós-inicialização**.
- **Pre-Stop**: executado antes de um container ser parado. A função de processamento pré-parada ajuda a garantir que os serviços em execução nos pods possam ser concluídos com antecedência no caso de atualização ou exclusão do pod. Para mais detalhes, consulte **Processamento pré-parada**.

Comandos de inicialização

Por padrão, o comando padrão durante o início da imagem. Para executar um comando específico ou reescrever o valor de imagem padrão, você deve executar configurações específicas:

Uma imagem do Docker tem metadados que armazenam informações de imagem. Se os comandos e argumentos do ciclo de vida não estiverem definidos, o CCE executará os

comandos e argumentos padrão, ou seja, instruções do Docker **ENTRYPOINT** e **CMD**, fornecidas durante a criação da imagem.

Se os comandos e argumentos usados para executar um contêiner forem definidos durante a criação da aplicação, os comandos padrão **ENTRYPOINT** e **CMD** serão substituídos durante a criação da imagem. As regras são as seguintes:

Tabela 5-8 Comandos e argumentos usados para executar um contêiner

Imagem ENTRYPOINT	Imagem CMD	Comando para executar um contêiner	Parâmetros para executar um contêiner	Comando executado
[touch]	[/root/test]	Não definido	Não definido	[touch /root/test]
[touch]	[/root/test]	[mkdir]	Não definido	[mkdir]
[touch]	[/root/test]	Não definido	[/opt/test]	[touch /opt/test]
[touch]	[/root/test]	[mkdir]	[/opt/test]	[mkdir /opt/test]

Passo 1 Efetue logon no console do CCE. Ao criar uma carga de trabalho, configure as informações do contêiner e selecione **Lifecycle**.

Passo 2 Insira um comando e argumentos na página de guia **Startup Command**.

Tabela 5-9 Comando de inicialização do contêiner

Itens de configuração	Procedimento
Comando	Digite um comando executável, por exemplo, **/run/server**. Se houver vários comandos executáveis, escreva-os em linhas diferentes. **NOTA** No caso de vários comandos, você é aconselhado a executar **/bin/sh** ou outros comandos **shell**. Outros comandos são usados como parâmetros.
Args	Insira o argumento que controla o comando em execução do contêiner, por exemplo, **--port=8080**. Se houver vários argumentos, separe-os em linhas diferentes.

----Fim

Processamento pós-inicialização

Passo 1 Efetue logon no console do CCE. Ao criar uma carga de trabalho, configure as informações do contêiner e selecione **Lifecycle**.

Passo 2 Defina os parâmetros de processamento pós-inicialização na página de guia **Post-Start**.

Tabela 5-10 Parâmetros de processamento pós-inicialização

Parâmetro	Descrição

CLI	Defina os comandos a serem executados no contêiner para o processamento pós-inicialização. O formato de comando é **Command Args[1] Args[2]...**. **Command** é um comando do sistema ou um programa executável definido pelo usuário. Se nenhum caminho for especificado, um programa executável no caminho padrão será selecionado. Se vários comandos precisarem ser executados, escreva os comandos em um script para execução. **Os comandos que são executados em segundo plano ou de forma assíncrona não são suportados.**
	Exemplo de comando:
	`exec:`
	`  command:`
	`  - /install.sh`
	`  - install_agent`
	Digite **/install install_agent** no script. Este comando indica que **install.sh** será executado depois que o contêiner for criado com sucesso.

Solicitação HTTPS	Envie uma solicitação HTTP para processamento pós-inicialização. Os parâmetros relacionados são descritos a seguir: ● **Path**: (opcional) URL de solicitação. ● **Port**: (obrigatório) porta de solicitação. ● **Host**: (opcional) endereço IP do host solicitado. O valor padrão é o endereço IP do pod.

----Fim

Processamento pré-parada

Passo 1 Efetue logon no console do CCE. Ao criar uma carga de trabalho, configure as informações do contêiner e selecione **Lifecycle**.

Passo 2 Defina os parâmetros de processamento pré-inicialização na página de guia **Pre-Stop**.

Tabela 5-11 Parâmetros de processamento pré-parada

Parâmetro	Descrição

CLI	Definir comandos a serem executados no contêiner para processamento pré-parada. O formato de comando é **Command Args[1] Args[2]...**. **Command** é um comando do sistema ou um programa executável definido pelo usuário. Se nenhum caminho for especificado, um programa executável no caminho padrão será selecionado. Se vários comandos precisarem ser executados, escreva os comandos em um script para execução.
	Exemplo de comando:
	`exec:`
	`  command:`
	`  - /uninstall.sh`
	`  - uninstall_agent`
	Digite **/uninstall uninstall_agent** no script. Esse comando indica que **uninstall.sh** será executado antes que o contêiner conclua sua execução e pare de ser executado.

Solicitação HTTPS	Envie uma solicitação HTTP para pré-parada de processamento. Os parâmetros relacionados são descritos a seguir: ● **Path**: (opcional) URL de solicitação. ● **Port**: (obrigatório) porta de solicitação. ● **Host**: (opcional) endereço IP do host solicitado. O valor padrão é o endereço IP do pod.

----Fim

Exemplo YAML

Esta seção usa o Nginx como um exemplo para descrever como definir o ciclo de vida do contêiner.

No seguinte arquivo de configuração, o comando **postStart** é definido para executar o comando **install.sh** no diretório **/bin/bash**. **preStop** é definido para executar o comando **uninstall.sh**.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - image: nginx
        command:
        - sleep 3600                          # Startup command
        imagePullPolicy: Always
        lifecycle:
```



```
        postStart:
          exec:
            command:
            - /bin/bash
            - install.sh                  # Post-start command
        preStop:
          exec:
            command:
            - /bin/bash
            - uninstall.sh                 # Pre-stop command
      name: nginx
    imagePullSecrets:
    - name: default-secret
```

5.3.6 Configuração da verificação de integridade para um contêiner

Cenário

Verificação de integridade pode verificar regularmente o status de integridade dos contêineres durante o funcionamento do contêiner. Se a função de verificação de integridade não estiver configurada, um pod não poderá detectar exceções de aplicação ou reiniciar automaticamente a aplicação para restaurá-la. Isso resultará em uma situação em que o status do pod é normal, mas a aplicação no pod é anormal.

O Kubernetes fornece os seguintes testes de verificação de integridade:

- **Sonda de vivacidade** (livenessProbe): verifica se um contêiner ainda está ativo. É similar ao comando **ps** que verifica se um processo existe. Se a verificação de vivacidade de um contêiner falhar, o cluster reiniciará o contêiner. Se a verificação de vivacidade for bem-sucedida, nenhuma operação será executada.
- **Sonda de prontidão** (readinessProbe): verifica se um contêiner está pronto para processar solicitações do usuário. Quando o contêiner for detectado despreparado, o tráfego de serviço não será direcionado para o contêiner. Pode levar muito tempo para que algumas aplicações sejam iniciadas antes que elas possam fornecer serviços. Isso ocorre porque elas precisam carregar dados do disco ou depender da inicialização de um módulo externo. Nesse caso, o processo da aplicação está em execução, mas a aplicação não pode fornecer serviços. Para resolver esse problema, essa sonda de verificação de integridade é usada. Se verificação de prontidão do contêiner falhar, o cluster mascara todas as solicitações enviadas ao contêiner. Se a verificação de prontidão do contêiner for bem-sucedida, o contêiner poderá ser acessado.
- **Sonda de inicialização** (startupProbe): verifica quando uma aplicação em contêiner foi iniciada. Se tal sonda estiver configurada, ela desabilitará as verificações de vivacidade e prontidão até que seja bem-sucedida, garantindo que essas sondas não interfiram na inicialização da aplicação. Isso pode ser usado para adotar verificações de vivacidade em contêineres de partida lenta, evitando que eles sejam encerrados pelo kubelet antes de serem iniciados.

Método de verificação

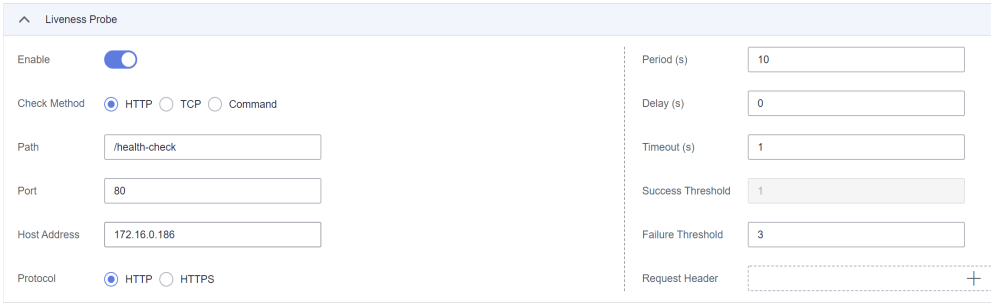
- **Solicitação HTTP**

  Esse modo de verificação de integridade se aplica a contêineres que fornecem serviços HTTP/HTTPS. O cluster inicia periodicamente uma solicitação GET HTTP/HTTPS para esses contêineres. Se o código de retorno da resposta HTTP/HTTPS estiver entre 200 e 399, o teste será bem-sucedido. Caso contrário, a sonda falha. Neste modo de verificação

de integridade, você deve especificar uma porta de escuta de contêiner e um caminho de solicitação HTTP/HTTPS.

Por exemplo, para um contêiner que fornece serviços HTTP, o caminho de verificação HTTP é **/health-check**, a porta é 80 e o endereço do host é opcional (que padrão é o endereço IP do contêiner). Aqui, 172.16.0.186 é usado como um exemplo, e podemos obter tal solicitação: GET http://172.16.0.186:80/health-check. O cluster inicia periodicamente essa solicitação ao contêiner. Você também pode adicionar um ou mais cabeçalhos a uma solicitação HTTP. Por exemplo, defina o nome do cabeçalho da solicitação como **Custom-Header** e o valor correspondente como **example**.

Figura 5-5 Verificação baseada em solicitação HTTP

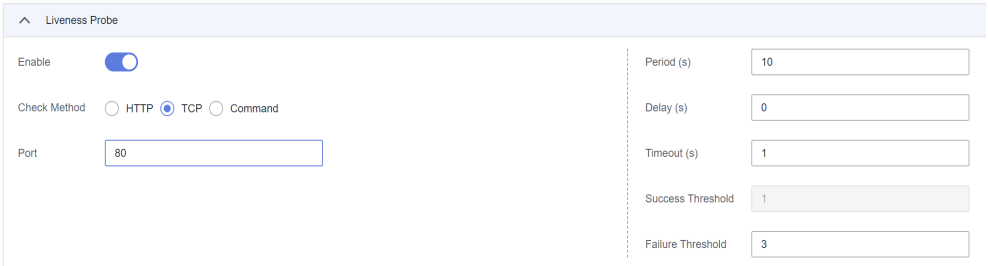


- **Porta TCP**

  Para um contêiner que fornece serviços de comunicação TCP, o cluster estabelece periodicamente uma conexão TCP com o contêiner. Se a conexão for bem-sucedida, a sonda será bem-sucedida. Caso contrário, a sonda falha. Neste modo de verificação de integridade, você deve especificar uma porta de escuta de contêiner.

  Por exemplo, se você tiver um contêiner Nginx com porta de serviço 80, depois de especificar a porta TCP 80 para escuta do contêiner, o cluster iniciará periodicamente uma conexão TCP com a porta 80 do contêiner. Se a conexão for bem-sucedida, a sonda será bem-sucedida. Caso contrário, a sonda falha.

Figura 5-6 Verificação baseada na porta TCP



- **CLI**

  CLI é uma ferramenta eficiente para verificação de integridade. Ao usar a CLI, você deve especificar um comando executável em um contêiner. O cluster executa periodicamente o comando no contêiner. Se a saída do comando for 0, a verificação de integridade será bem-sucedida. Caso contrário, a verificação de integridade falha.

  O modo CLI pode ser usado para substituir a verificação de integridade baseada em solicitação HTTP e baseada em porta TCP.

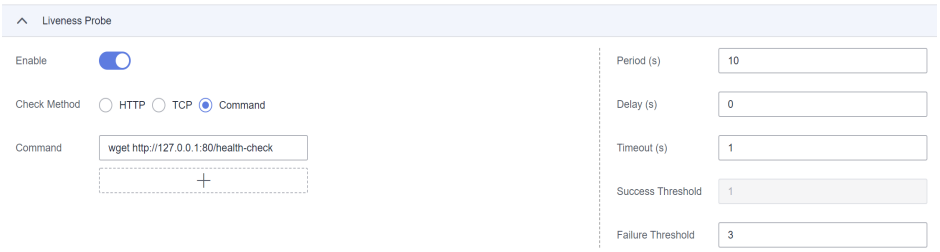
  – Para uma porta TCP, você pode usar um script de programa para se conectar a uma porta de contêiner. Se a conexão for bem-sucedida, o script retornará **0**. Caso contrário, o script retorna **–1**.

– Para uma solicitação HTTP, você pode usar o comando script para executar o comando **wget** para detectar o contêiner.

wget http://127.0.0.1:80/health-check

Verifique o código de retorno da resposta. Se o código de retorno estiver entre 200 e 399, o script retornará **0**. Caso contrário, o script retorna **–1**.

Figura 5-7 Verificação baseada na CLI



AVISO

- Coloque o programa a ser executado na imagem de contêiner para que o programa possa ser executado.
- Se o comando a ser executado for um script shell, não especifique diretamente o script como o comando, mas adicione um analisador de script. Por exemplo, se o script for **/data/scripts/health_check.sh**, você deve especificar **sh/data/scripts/health_check.sh** para a execução do comando. A razão é que o cluster não está no ambiente de terminal ao executar programas em um contêiner.

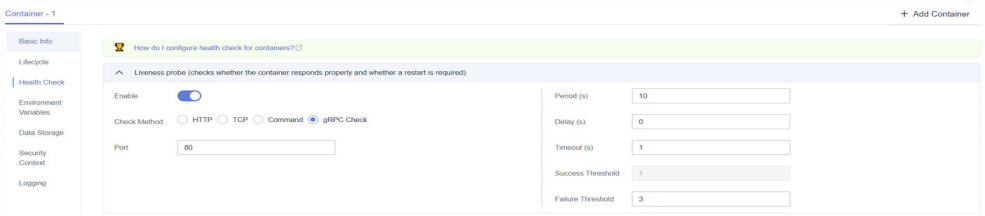
- **Verificação de gRPC**

  As verificações do gRPC podem configurar sondas de inicialização, vivacidade e prontidão para a aplicação gRPC sem expor nenhum ponto de extremidade HTTP, nem é necessário um executável. O Kubernetes pode se conectar à sua carga de trabalho via gRPC e obter seu status.

AVISO

- A verificação de gRPC é suportada apenas em clusters do CCE v1.25 ou posterior.
- Para usar gRPC para verificação, sua aplicação deve oferecer suporte ao **protocolo de verificação de integridade do gRPC**.
- Semelhante às sondas HTTP e TCP, se a porta estiver incorreta ou o aplicativo não oferecer suporte ao protocolo de verificação de integridade, a verificação falhará.

Figura 5-8 Verificação de gRPC



Parâmetros comuns

Tabela 5-12 Descrição do parâmetro de domínio

Parâmetro	Descrição
Period (periodSeconds)	Indica o período de detecção da sonda, em segundos. Por exemplo, se esse parâmetro for definido como **30**, a detecção será realizada a cada 30 segundos.
Delay (initialDelaySeconds)	Verifique o tempo de atraso em segundos. Defina este parâmetro de acordo com o tempo normal de inicialização dos serviços. Por exemplo, se esse parâmetro for definido como **30**, a verificação de saúde será iniciada 30 segundos após o início do contêiner. O tempo é reservado para o início dos serviços em contêiner.
Timeout (timeoutSeconds)	Número de segundos após os quais o tempo limite da sonda. Unidade: segundo. Por exemplo, se esse parâmetro for definido como **10**, o tempo de espera de tempo limite para a execução de uma verificação de integridade será de 10s. Se o tempo de espera passar, a verificação de integridade é considerada uma falha. Se o parâmetro for deixado em branco ou definido como **0**, o tempo de espera padrão é 1s.
Success Threshold (successThreshold)	Mínimo de sucessos consecutivos para que a sonda seja considerada bem sucedida depois de ter falhado. Por exemplo, se esse parâmetro for definido como **1**, o status da carga de trabalho será normal somente quando a verificação de integridade for bem-sucedida por um tempo consecutivo após a verificação de integridade falhar. O valor padrão é **1**, que também é o valor mínimo. O valor deste parâmetro é fixado em **1** em **Liveness Probe** e **Startup Probe**.
Failure Threshold (failureThreshold)	Número de tentativas repetidas vezes em que a detecção falha. Desistir em caso de sonda de vivacidade significa reiniciar o contêiner. Em caso de sonda de prontidão o pod será marcado como Unready. O valor padrão é **3**. O valor mínimo é **1**.

Exemplo de YAML

```
apiVersion: v1
kind: Pod
metadata:
  labels:
    test: liveness
  name: liveness-http
spec:
  containers:
  - name: liveness
```



```
    image: nginx:alpine
    args:
    - /server
    livenessProbe:
      httpGet:
        path: /healthz
        port: 80
        httpHeaders:
        - name: Custom-Header
          value: Awesome
      initialDelaySeconds: 3
      periodSeconds: 3
    readinessProbe:
      exec:
        command:
          - cat
          - /tmp/healthy
      initialDelaySeconds: 5
      periodSeconds: 5
    startupProbe:
      httpGet:
        path: /healthz
        port: 80
      failureThreshold: 30
      periodSeconds: 10
```

5.3.7 Configuração de uma variável de ambiente

Cenário

Uma variável de ambiente é uma variável cujo valor pode afetar a maneira como um contêiner em execução se comportará. Você pode modificar variáveis de ambiente mesmo depois que as cargas de trabalho são implantadas, aumentando a flexibilidade na configuração da carga de trabalho.

A função de definir variáveis de ambiente no CCE é a mesma de especificar **ENV** em um Dockerfile.

AVISO

Depois que um contêiner é iniciado, não modifique as configurações no contêiner. Se as configurações no contêiner forem modificadas (por exemplo, senhas, certificados e variáveis de ambiente de uma aplicação em contêiner são adicionados ao contêiner), as configurações serão perdidas depois que o contêiner for reiniciado e os serviços do contêiner se tornarão anormais. Um exemplo de cenário de reinicialização do contêiner é o reagendamento do pod devido a anomalias do nó.

As configurações devem ser importadas para um contêiner como argumentos. Caso contrário, as configurações serão perdidas após a reinicialização do contêiner.

As variáveis de ambiente podem ser definidas nos seguintes modos:

- **Custom**: insira o nome da variável de ambiente e o valor do parâmetro.
- **Added from ConfigMap key**: importe todas as chaves em um ConfigMap como variáveis de ambiente.
- **Added from ConfigMap**: importe uma chave em um ConfigMap como o valor de uma variável de ambiente. Como mostrado em **Figura 5-9**, se você importar **configmap_value** de **configmap_key** em um ConfigMap como o valor da variável de

ambiente **key1**, uma variável de ambiente chamada **key1** cujo valor é **configmap_value** existe no contêiner.

- **Added from secret**: importe todas as chaves em um segredo como variáveis de ambiente.
- **Added from secret key**: importe o valor de uma chave em um segredo como o valor de uma variável de ambiente. Como mostrado em **Figura 5-9**, se você importar **secret_value** de **secret_key** em segredo **secret-example** como o valor da variável de ambiente **key2**, uma variável de ambiente chamada **key2** cujo valor é **secret_value** existe no contêiner.
- **Variable value/reference**: use o campo definido por um pod como o valor da variável de ambiente. Conforme mostrado em **Figura 5-9**, se o nome do pod for importado como o valor da variável de ambiente **key3**, uma variável de ambiente chamada **key3** existirá no contêiner e seu valor será o nome do pod.
- **Resource Reference**: o valor de **Request** ou **Limit** definido pelo contêiner é usado como o valor da variável de ambiente. Conforme mostrado em **Figura 5-9**, se você importar o limite de CPU do contêiner-1 como o valor da variável de ambiente **key4**, uma variável de ambiente chamada **key4** existirá no contêiner e seu valor será o limite de CPU do contêiner-1.

Adicionar variáveis de ambiente

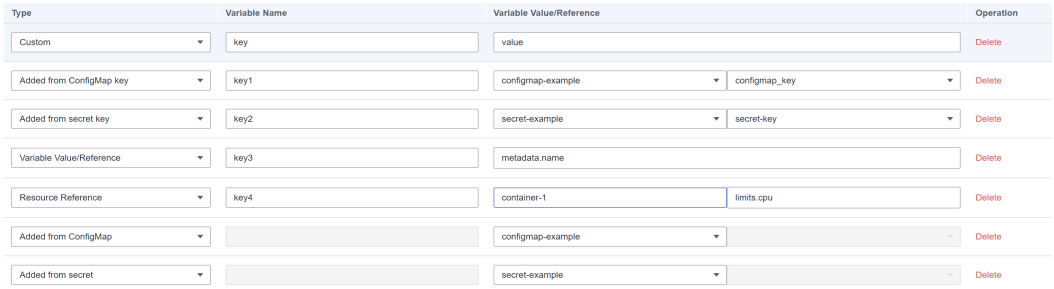
Passo 1 Efetue logon no console do CCE.

Passo 2 Clique no nome do cluster para acessar o console do cluster, escolha **Workloads** no painel de navegação e clique em **Create Workload** no canto superior direito.

Passo 3 Ao criar uma carga de trabalho, modifique as informações do contêiner em **Container Configuration** e clique na guia **Environment Variables**.

Passo 4 Configure variáveis de ambiente.

Figura 5-9 Configurar variáveis de ambiente



----Fim

Exemplo de YAML

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: env-example
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
```



```
      app: env-example
  template:
    metadata:
      labels:
        app: env-example
    spec:
      containers:
        - name: container-1
          image: nginx:alpine
          imagePullPolicy: Always
          resources:
            requests:
              cpu: 250m
              memory: 512Mi
            limits:
              cpu: 250m
              memory: 512Mi
          env:
            - name: key                     # Custom
              value: value
            - name: key1                    # Added from ConfigMap key
              valueFrom:
                configMapKeyRef:
                  name: configmap-example
                  key: key1
            - name: key2                    # Added from secret key
              valueFrom:
                secretKeyRef:
                  name: secret-example
                  key: key2
            - name: key3                    # Variable reference, which uses the
field defined by a pod as the value of the environment variable.
              valueFrom:
                fieldRef:
                  apiVersion: v1
                  fieldPath: metadata.name
            - name: key4                    # Resource reference, which uses the
field defined by a container as the value of the environment variable.
              valueFrom:
                resourceFieldRef:
                  containerName: container1
                  resource: limits.cpu
                  divisor: 1
          envFrom:
            - configMapRef:                 # Added from ConfigMap
                name: configmap-example
            - secretRef:                    # Added from secret
                name: secret-example
      imagePullSecrets:
        - name: default-secret
```

Exibii variáveis de ambiente

Se os conteúdos de **configmap-example** e **secret-example** são os seguintes:

```
$ kubectl get configmap configmap-example -oyaml
apiVersion: v1
data:
  configmap_key: configmap_value
kind: ConfigMap
...

$ kubectl get secret secret-example -oyaml
apiVersion: v1
data:
  secret_key: c2VjcmV0X3ZhbHVl              # c2VjcmV0X3ZhbHVl is the value of
secret_value in Base64 mode.
kind: Secret
...
```


As variáveis de ambiente no pod são as seguintes:

```
$ kubectl get pod
NAME                           READY   STATUS    RESTARTS   AGE
env-example-695b759569-lx9jp   1/1     Running   0          17m

$ kubectl exec env-example-695b759569-lx9jp  -- printenv
/ # env
key=value                              # Custom environment variable
ey1=configmap_value                   # Added from ConfigMap key
key2=secret_value                      # Added from secret key
key3=env-example-695b759569-lx9jp      # metadata.name defined by the pod
key4=1                                 # limits.cpu defined by container1. The
value is rounded up, in unit of cores.
configmap_key=configmap_value          # Added from ConfigMap. The key value in
the original ConfigMap key is directly imported.
secret_key=secret_value                # Added from key. The key value in the
original secret is directly imported.
```

5.3.8 Configuração de configurações de APM para análise de gargalo de desempenho

Cenário

O Application Performance Management (APM) permite que você monitore cargas de trabalho Java por meio de rastreamento e topologia. Você pode instalar testes de APM para localizar e analisar problemas para cargas de trabalho Java.

Você pode configurar o monitoramento da carga de trabalho Java quando e após a criação de uma carga de trabalho.

NOTA

- Conecte sua aplicação de Java no CCE ao APM através da sonda Pinpoint. Para obter detalhes, consulte **Conexão de uma aplicação conteinerizada da Huawei Cloud ao APM**.

Pré-requisitos

Se você não ativou o serviço APM, vá para o console do APM e ative-o conforme solicitado.

Procedimento

Passo 1 Efetue logon no console do CCE.

Passo 2 Clique no nome do cluster para acessar o console do cluster, escolha **Workloads** no painel de navegação e clique em **Create Workload** no canto superior direito.

Passo 3 Ao criar uma carga de trabalho, escolha **Advanced Settings** > **APM Settings**. Por padrão, a sonda está desativada. Você pode escolher **APM 2.0** ou **APM 1.0**, conforme necessário. Depois que o teste é ativado, o APM pode localizar e analisar problemas para programas Java.

NOTA

1. A sonda de APM 1.0 será inicializada em um contêiner init criado automaticamente chamado init-pinpoint. O contêiner init será alocado com CPU de 0,25 núcleo e memória de 250 MiB.
2. A adição de uma sonda do APM adicionará as variáveis de ambiente **PAAS_MONITORING_GROUP**, **JAVA_TOOL_OPTIONS** e **PAAS_CLUSTER_ID** a todos os contêineres de serviço.
3. A adição de uma sonda de APM montará um volume de armazenamento local denominado paas-apm (para sonda de APM 1.0) em todos os contêineres de serviço.

Passo 4 Definir parâmetros relacionados à sonda.

Sonda de APM 1.0

- **Monitoring Group**: digite um nome de grupo de monitoramento, por exemplo, **testapp**.
- **Probe Version**: selecione a versão da sonda.
- **Probe Upgrade Policy**: por padrão, **Auto upgrade upon restart** está selecionado.
  - **Auto upgrade upon restart**: o sistema baixa a imagem da sonda toda vez que o pod é reiniciado.
  - **Manual upgrade upon restart**: essa política significa que, se uma imagem local estiver disponível, a imagem local será usada. O sistema baixa a imagem da sonda somente quando uma imagem local não está disponível.

Passo 5 Três minutos após o início da aplicação, seus dados serão exibidos no console do APM. Você pode fazer logon no console do APM e otimizar o desempenho da aplicação por meio de topologia e rastreamento. Para mais detalhes, veja **Topologia**.

----Fim

Configurar configurações de APM

Passo 1 Efetue logon no console do CCE.

Passo 2 Clique no nome do cluster para ir para o console do cluster, escolha **Workloads** no painel de navegação e clique no nome da carga de trabalho desejada.

Passo 3 Na página exibida, alterne para a página de guia **APM Settings** e clique em **Edit** no canto inferior direito.

Para obter detalhes sobre os parâmetros, consulte **Passo 4**.

----Fim

5.3.9 Configuração da política de atualização da carga de trabalho

Em aplicações reais, a atualização é uma operação comum. Uma Implementação, o StatefulSet ou o DaemonSet podem suportar facilmente a atualização de aplicações.

Você pode definir diferentes políticas de atualização:

- **Atualização contínua**: novos pods são criados gradualmente e, em seguida, os pods anteriores são excluídos. Esta é a política padrão.
- **Atualização de substituição**: os pods atuais são excluídos e, em seguida, novos pods são criados.

Parâmetros de atualização

Parâmetro	Descrição	Restrição
Max. Surge (maxSurge)	Especifica o número máximo de pods que podem existir em comparação com **spec.replicas**. O valor padrão é **25%**. Por exemplo, se **spec.replicas** for definido como **4**, um máximo de cinco pods podem existir durante a atualização. Ou seja, a atualização é realizada em uma etapa de 1. Durante a atualização real, o valor é convertido em um número e arredondado para cima. O valor também pode ser definido como um número absoluto.	Esse parâmetro é suportado apenas por Implementações e DaemonSets.
Max. Unavailable Pods (maxUnavailable)	Especifica o número máximo de pods que podem estar indisponíveis em comparação com **spec.replicas**. O valor padrão é **25%** Por exemplo, se **spec.replicas** estiver definido como **4**, pelo menos três pods existirão durante a atualização. Ou seja, a exclusão é realizada em uma etapa de 1. O valor também pode ser definido como um número absoluto.	Esse parâmetro é suportado apenas por Implementações e DaemonSets.
Min. Ready Seconds (minReadySeconds)	Um pod é considerado disponível apenas quando o tempo mínimo de prontidão é excedido sem que nenhum de seus contêineres falhe. O valor padrão é **0** (o pod é considerado disponível imediatamente após estar pronto).	Nenhuma
Revision History Limit (revisionHistoryLimit)	Especifica o número de ReplicaSets antigos a serem retidos para permitir a reversão. Esses ReplicaSets antigos consomem recursos em etcd e lotam a saída de **kubectl get rs**. A configuração de cada revisão de Implementação é armazenada em seus ReplicaSets. Portanto, depois que o ReplicaSet anterior for excluído, você perderá a capacidade de reverter para essa revisão de Implementação. Por padrão, 10 ReplicaSets anteriores serão mantidos, mas o valor ideal depende da frequência e estabilidade das novas Implementações.	Nenhuma

Parâmetro	Descrição	Restrição
Max. Upgrade Duration (progressDeadline-Seconds)	Especifica o número de segundos que o sistema aguarda que uma Implementação progrida antes de comunicar uma falha de progresso da Implementação. Ele é exibido como uma condição com Type=Progressing, Status=False e Reason=ProgressDeadlineExceeded no status do recurso. O controlador de Implementação continuará tentando novamente a Implementação. No futuro, quando a reversão automática for implementada, o controlador de Implementação reverterá uma Implementação assim que observar tal condição. Se este parâmetro for especificado, o valor deste parâmetro deve ser maior que **.spec.minReadySeconds**.	Nenhuma
Scale-In Time Window (terminationGrace-PeriodSeconds)	Tempo de exclusão gracioso. O valor padrão é 30 segundos. Quando um pod é excluído, um sinal SIGTERM é enviado e o sistema aguarda que as aplicações no contêiner terminem. Se a aplicação não for encerrada dentro do tempo especificado por **terminationGracePeriodSeconds**, um sinal SIGKILL é enviado para terminar forçosamente o pod.	Nenhuma

Exemplo de atualização

A Implementação pode ser atualizada em um modo declarativo. Ou seja, você só precisa modificar a definição YAML da Implementação. Por exemplo, você pode executar o comando **kubectl edit** para alterar a imagem de Implementação para **nginx:alpine**. Após a modificação, consulte o ReplicaSet e o pod. O resultado da consulta mostra que um novo ReplicaSet é criado e o pod é recriado.

```
$ kubectl edit deploy nginx

$ kubectl get rs
NAME               DESIRED   CURRENT   READY     AGE
nginx-6f9f58dffd   2         2         2         1m
nginx-7f98958cdf   0         0         0         48m

$ kubectl get pods
NAME                     READY     STATUS    RESTARTS   AGE
nginx-6f9f58dffd-tdmqk   1/1       Running   0          1m
nginx-6f9f58dffd-tesqr   1/1       Running   0          1m
```

A Implementação pode usar os parâmetros **maxSurge** e **maxUnavailable** para controlar a proporção de pods a serem recriados durante a atualização, o que é útil em muitos cenários. A configuração é a seguinte:

```
spec:
  strategy:
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 0
    type: RollingUpdate
```


No exemplo anterior, o valor de **spec.replicas** é **2**. Se ambos **maxSurge** e **maxUnavailable** forem o valor padrão 25%, **maxSurge** permite que existam no máximo três pods (2 x 1,25 = 2,5, arredondado para 3), e **maxUnavailable** não permite que um máximo de dois pods estejam indisponíveis (2 x 0,75 = 1,5, arredondado para 2). Ou seja, durante o processo de atualização, sempre haverá dois pods em execução. Cada vez que um novo pod é criado, um pod antigo é excluído, até que todos os pods sejam novos.

Reversão

Reversão é reverter uma aplicação para a versão anterior quando ocorre uma falha durante a atualização. Uma Implementação pode ser facilmente revertida para a versão anterior.

Por exemplo, se a imagem atualizada estiver com defeito, você poderá executar o comando **kubectl rollout undo** desfazer para reverter a Implementação.

```
$ kubectl rollout undo deployment nginx
deployment.apps/nginx rolled back
```

Uma Implementação pode ser facilmente revertida porque usa um ReplicaSet para controlar um pod. Após a atualização, o ReplicaSet anterior ainda existe. A Implementação é revertida usando o ReplicaSet anterior para recriar o pod. O número de ReplicaSets armazenados em uma Implementação pode ser restrito pelo parâmetro **revisionHistoryLimit**. O valor padrão é **10**.

5.3.10 Política de agendamento (afinidade/antiafinidade)

O Kubernetes suporta afinidade de nó e afinidade/antiafinidade de pod. Você pode configurar regras personalizadas para obter o agendamento de afinidade e antiafinidade. Por exemplo, você pode implementar pods de front-end e pods de back-end juntos, implementar o mesmo tipo de aplicações em um nó específico ou implementar aplicações diferentes em nós diferentes.

A afinidade do Kubernetes se aplica a nós e pods.

- **nodeAffinity**: semelhante ao nodeSelector do pod, e ambos agendam pods apenas para os nós com rótulos especificados. A diferença entre nodeAffinity e nodeSelector reside no fato de que nodeAffinity apresenta uma expressão mais forte que nodeSelector e permite especificar restrições suaves selecionadas preferencialmente. Os dois tipos de afinidade de nó são os seguintes:
  - requiredDuringSchedulingIgnoredDuringExecution: restrição rígida que **deve ser atendida**. O agendador pode realizar o agendamento somente quando a regra é atendida. Essa função é semelhante ao nodeSelector, mas apresenta uma expressão de sintaxe mais forte. Para mais detalhes, consulte **Node Affinity (nodeAffinity)**.
  - preferredDuringSchedulingIgnoredDuringExecution: restrição suave que é **atendida o máximo possível**. O agendador tenta encontrar o nó que atende à regra. Se nenhum nó correspondente for encontrado, o agendador ainda agendará o pod. Para mais detalhes, consulte **Regra de preferência de nó**.
- **Afinidade da carga de trabalho (podAffinity)**/**Workload Anti-affinity (podAntiAffinity)**: os nós para os quais um pod pode ser agendado são determinados com base no rótulo do pod em execução em um nó, mas não no rótulo do nó. Semelhante à afinidade de nó, a afinidade de carga de trabalho e a antiafinidade também são dos tipos de requiredDuringSchedulingIgnoredDuringExecution e preferredDuringSchedulingIgnoredDuringExecution.

> **NOTA**
>
> A afinidade e a antiafinidade da carga de trabalho exigem uma certa quantidade de tempo de computação, o que retarda significativamente o agendamento em clusters de grande escala. Não ative afinidade e antiafinidade de carga de trabalho em um cluster que contém centenas de nós.

Você pode criar as políticas de afinidade anteriores no console. Para mais detalhes, consulte **Configurar políticas de agendamento**.

Configurar políticas de agendamento

Passo 1 Efetue logon no console do CCE.

Passo 2 Ao criar uma carga de trabalho, clique em **Scheduling** na área **Advanced Settings**.

Tabela 5-13 Configurações de afinidade de nó

Parâmetro	Descrição
Required	Restrição rígida, que corresponde requiredDuringSchedulingIgnoredDuringExecution para especificar as condições que devem ser atendidas. Se várias regras que **devem ser atendidas** forem adicionadas, o agendamento será realizado quando apenas uma regra for atendida.
Preferred	Restrição suave, que corresponde a preferredDuringSchedulingIgnoredDuringExecution para especificar as condições que devem ser atendidas o maior número possível. Se forem adicionadas várias regras que **devem ser cumpridas tanto quanto possível**, o agendamento será realizado mesmo que uma ou nenhuma das regras seja cumprida.

Passo 3 Clique em + em **Node Affinity**, **Workload Affinity** ou **Workload Anti-Affinity** para adicionar políticas de agendamento. Na caixa de diálogo exibida, adicione diretamente as políticas. Como alternativa, você pode especificar nós ou AZs a serem agendados no console.

A especificação de nós e AZs também é implementada por meio de rótulos. O console libera você da inserção manual de rótulos de nó. O rótulo **kubernetes.io/hostname** é usado quando você especifica um nó, e o rótulo **failure-domain.beta.kubernetes.io/zone** é usado quando você especifica uma AZ.

Figura 5-10 Adicionar uma política de agendamento

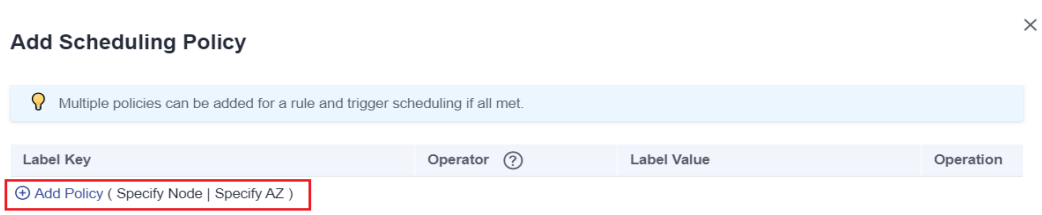


Tabela 5-14 Parâmetros para configurar a política de agendamento

Parâmetro	Descrição
Label	Rótulo do nó. Você pode usar o rótulo padrão ou personalizar um rótulo.
Operator	As seguintes relações são compatíveis: **In**, **NotIn**, **Exists**, **DoesNotExist**, **Gt** e **Lt**. ● **In**: o rótulo do objeto de afinidade ou antiafinidade está na lista de valores do rótulo (campo **values**). ● **NotIn**: o rótulo do objeto de afinidade ou antiafinidade não está na lista de valores do rótulo (campo **values**). ● **Exists**: o objeto de afinidade ou antiafinidade tem um nome de rótulo especificado. ● **DoesNotExist**: o objeto de afinidade ou antiafinidade não tem o nome de rótulo especificado. ● **Gt**: (disponível apenas para afinidade de nó) o valor do rótulo do nó agendado é maior que o valor da lista (comparação de cadeias). ● **Lt**: (disponível somente para afinidade de nó) o valor do rótulo do nó de agendamento é menor que o valor da lista (comparação de cadeias).
Label Value	Valor do rótulo.
Namespace	Esse parâmetro está disponível somente em uma política de agendamento de afinidade ou antiafinidade de carga de trabalho. Namespace para o qual a política de agendamento entra em vigor.
Topology Key	Esse parâmetro pode ser usado somente em uma política de agendamento de afinidade ou antiafinidade de carga de trabalho. Selecione o escopo especificado por **topologyKey** e, em seguida, selecione o conteúdo definido pela política.
Weight	Esse parâmetro pode ser definido apenas em uma política de agendamento **Preferred**.

----Fim

Node Affinity (nodeAffinity)

As regras de afinidade de nó de carga de trabalho são implementadas usando rótulos de nó. Quando um nó é criado em um cluster do CCE, determinados rótulos são adicionados automaticamente. Você pode executar o comando **kubectl describe node** para exibir os rótulos. O seguinte é um exemplo:

```
$ kubectl describe node 192.168.0.212
Name:               192.168.0.212
Roles:              <none>
Labels:             beta.kubernetes.io/arch=amd64
                    beta.kubernetes.io/os=linux
                    failure-domain.beta.kubernetes.io/is-baremetal=false
```



```
                    failure-domain.beta.kubernetes.io/region=******
                    failure-domain.beta.kubernetes.io/zone=******
                    kubernetes.io/arch=amd64
                    kubernetes.io/availablezone=******
                    kubernetes.io/eniquota=12
                    kubernetes.io/hostname=192.168.0.212
                    kubernetes.io/os=linux
                    node.kubernetes.io/subnetid=fd43acad-33e7-48b2-
a85a-24833f362e0e
                    os.architecture=amd64
                    os.name=EulerOS_2.0_SP5
                    os.version=3.10.0-862.14.1.5.h328.eulerosv2r7.x86_64
```

No agendamento de carga de trabalho, os rótulos de nó comuns são os seguintes:

- **failure-domain.beta.kubernetes.io/region**: região onde o nó está localizado.
- **failure-domain.beta.kubernetes.io/zone**: zona de disponibilidade à qual o nó pertence.
- **kubernetes.io/hostname**: nome do host do nó.

O Kubernetes fornece o campo **nodeSelector**. Ao criar uma carga de trabalho, você pode definir esse campo para especificar que o pod só pode ser implementado em um nó com o rótulo específico. O exemplo a seguir mostra como usar um nodeSelector para implantar o pod somente no nó com o rótulo **gpu=true**.

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  nodeSelector:                # Node selection. A pod is deployed on a node
only when the node has the gpu=true label.
    gpu: true
...
```

As regras de afinidade de nó podem alcançar os mesmos resultados. Em comparação com nodeSelector, as regras de afinidade de nó parecem mais complexas, mas com uma sintaxe mais expressiva. Você pode usar o campo **spec.affinity.nodeAffinity** para definir a afinidade do nó. Existem dois tipos de afinidade de nó:

- **requiredDuringSchedulingIgnoredDuringExecution**: o Kubernetes não pode agendar o pod a menos que a regra seja cumprida.
- **PreferredDuringSchedulingIgnoredDuringExecution**: o Kubernetes tenta encontrar um nó que atenda à regra. Se um nó correspondente não estiver disponível, o Kubernetes ainda agendará o pod.

NOTA

Nesses dois tipos de afinidade de nó, **requiredDuringScheduling** ou **preferredDuringScheduling** indica que o pod pode ser agendado para um nó somente quando todas as regras definidas forem atendidas (obrigatório). **IgnoredDuringExecution** indica que, se o rótulo do nó for alterado após o Kubernetes agendar o pod, o pod continuará em execução e não será reagendado.

Veja a seguir um exemplo de definição de afinidade de nó:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name:  gpu
  labels:
    app:  gpu
spec:
  selector:
    matchLabels:
      app: gpu
```



```
  replicas: 3
  template:
    metadata:
      labels:
        app:  gpu
    spec:
      containers:
      - image:  nginx:alpine
        name:  gpu
        resources:
          requests:
            cpu: 100m
            memory: 200Mi
          limits:
            cpu: 100m
            memory: 200Mi
      imagePullSecrets:
      - name: default-secret
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
              - key: gpu
                operator: In
                values:
                - "true"
```

Neste exemplo, o nó programado deve conter um rótulo com a chave chamada **gpu**. O valor do **operator** é **In**, indicando que o valor do rótulo deve estar na lista de **values**. Ou seja, o valor da chave do rótulo **gpu** do nó é **true**. Para obter detalhes sobre outros valores de **operator**, consulte **Descrição do valor do operador**. Observe que não existe nodeAntiAffinity porque os operadores **NotIn** e **DoesNotExist** fornecem a mesma função.

A seguir, descreve-se como verificar se a regra entra em vigor. Suponha que um cluster tenha três nós.

```
$ kubectl get node
NAME            STATUS   ROLES    AGE   VERSION
192.168.0.212   Ready    <none>   13m   v1.15.6-r1-20.3.0.2.B001-15.30.2
192.168.0.94    Ready    <none>   13m   v1.15.6-r1-20.3.0.2.B001-15.30.2
192.168.0.97    Ready    <none>   13m   v1.15.6-r1-20.3.0.2.B001-15.30.2
```

Adicione o rótulo **gpu=true** ao nó **192.168.0.212**.

```
$ kubectl label node 192.168.0.212 gpu=true
node/192.168.0.212 labeled

$ kubectl get node -L gpu
NAME            STATUS   ROLES    AGE   VERSION                              GPU
192.168.0.212   Ready    <none>   13m   v1.15.6-r1-20.3.0.2.B001-15.30.2   true
192.168.0.94    Ready    <none>   13m   v1.15.6-r1-20.3.0.2.B001-15.30.2
192.168.0.97    Ready    <none>   13m   v1.15.6-r1-20.3.0.2.B001-15.30.2
```

Crie a Implementação. Você pode descobrir que todos os pods estão implementados no nó **192.168.0.212**.

```
$ kubectl create -f affinity.yaml
deployment.apps/gpu created

$ kubectl get pod -o wide
NAME                   READY   STATUS    RESTARTS   AGE   IP
NODE
gpu-6df65c44cf-42xw4   1/1     Running   0          15s   172.16.0.37
192.168.0.212
gpu-6df65c44cf-jzjvs   1/1     Running   0          15s   172.16.0.36
192.168.0.212
gpu-6df65c44cf-zv5cl   1/1     Running   0          15s   172.16.0.38
192.168.0.212
```


Regra de preferência de nó

A regra **requiredDuringSchedulingIgnoredDuringExecution** é uma regra de seleção rígida. Existe outro tipo de regra de seleção, ou seja, **preferredDuringSchedulingIgnoredDuringExecution**. Ela é usada para especificar quais nós são preferidos durante o agendamento.

Para obter esse efeito, adicione um nó anexado com discos SAS ao cluster, adicione o rótulo **DISK=SAS** ao nó e adicione o rótulo **DISK=SSD** aos outros três nós.

```
$ kubectl get node -L DISK,gpu
NAME            STATUS   ROLES    AGE     VERSION
DISK     GPU
192.168.0.100   Ready    <none>   7h23m   v1.15.6-r1-20.3.0.2.B001-15.30.2
SAS
192.168.0.212   Ready    <none>   8h      v1.15.6-r1-20.3.0.2.B001-15.30.2
SSD      true
192.168.0.94    Ready    <none>   8h      v1.15.6-r1-20.3.0.2.B001-15.30.2
SSD
192.168.0.97    Ready    <none>   8h      v1.15.6-r1-20.3.0.2.B001-15.30.2   SSD
```

Defina uma Implementação. Use a regra **preferredDuringSchedulingIgnoredDuringExecution** para definir o peso dos nós com o disco SSD instalado como **80** e os nós com o rótulo **gpu=true** como **20**. Desta forma, os pods são preferencialmente implementados nos nós com o disco SSD instalado.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name:  gpu
  labels:
    app:  gpu
spec:
  selector:
    matchLabels:
      app: gpu
  replicas: 10
  template:
    metadata:
      labels:
        app:  gpu
    spec:
      containers:
      - image:  nginx:alpine
        name:  gpu
        resources:
          requests:
            cpu:  100m
            memory:  200Mi
          limits:
            cpu:  100m
            memory:  200Mi
      imagePullSecrets:
      - name: default-secret
      affinity:
        nodeAffinity:
          preferredDuringSchedulingIgnoredDuringExecution:
          - weight: 80
            preference:
              matchExpressions:
              - key: DISK
                operator: In
                values:
                - SSD
          - weight: 20
            preference:
              matchExpressions:
```



```
            - key: gpu
              operator: In
              values:
              - "true"
```

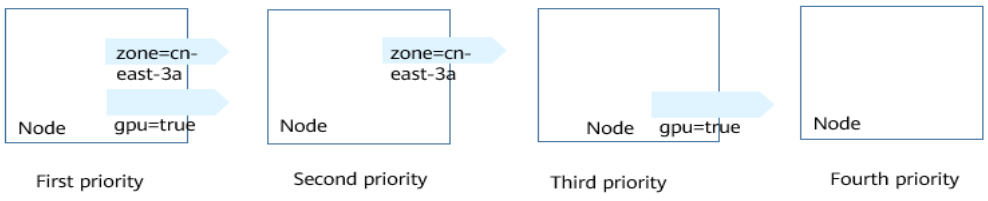
Após a implementação, há cinco pods implementados no nó **192.168.0.212** (rótulo: **DISK=SSD** e **GPU=true**), três pods implementados no nó **192.168.0.97** (rótulo: **DISK=SSD**) e dois pods implementados no nó **192.168.0.100** (rótulo: **DISK=SAS**).

A partir da saída anterior, você pode descobrir que nenhum pod da Implementação está agendado para o nó **192.168.0.94** (label: **DISK=SSD**). Isso ocorre porque o nó já possui muitos pods e seu uso de recursos é alto. Isso também indica que a regra **preferredDuringSchedulingIgnoredDuringExecution** define uma preferência em vez de um requisito rígido.

```
$ kubectl create -f affinity2.yaml
deployment.apps/gpu created

$ kubectl get po -o wide
NAME                   READY   STATUS    RESTARTS   AGE     IP
NODE
gpu-585455d466-5bmcz   1/1     Running   0          2m29s   172.16.0.44
192.168.0.212
gpu-585455d466-cg216   1/1     Running   0          2m29s   172.16.0.63
192.168.0.97
gpu-585455d466-f2bt2   1/1     Running   0          2m29s   172.16.0.79
192.168.0.100
gpu-585455d466-hdb5n   1/1     Running   0          2m29s   172.16.0.42
192.168.0.212
gpu-585455d466-hkgvz   1/1     Running   0          2m29s   172.16.0.43
192.168.0.212
gpu-585455d466-mngvn   1/1     Running   0          2m29s   172.16.0.48
192.168.0.97
gpu-585455d466-s26qs   1/1     Running   0          2m29s   172.16.0.62
192.168.0.97
gpu-585455d466-sxtzm   1/1     Running   0          2m29s   172.16.0.45
192.168.0.212
gpu-585455d466-t56cm   1/1     Running   0          2m29s   172.16.0.64
192.168.0.100
gpu-585455d466-t5w5x   1/1     Running   0          2m29s   172.16.0.41
192.168.0.212
```

No exemplo anterior, a prioridade de agendamento do nó é a seguinte. Os nós com rótulos **SSD** e **gpu=true** têm a prioridade mais alta. Os nós com o rótulo **SSD**, mas sem o rótulo **gpu=true**, têm a segunda prioridade (peso: 80). Os nós com o rótulo **gpu=true**, mas nenhum rótulo **SSD**, têm a terceira prioridade. Os nós sem qualquer um desses dois rótulos têm a prioridade mais baixa.

Figura 5-11 Prioridade de agendamento



Afinidade da carga de trabalho (podAffinity)

As regras de afinidade de nó afetam apenas a afinidade entre pods e nós. O Kubernetes também suporta a configuração de regras de afinidade entre pods. Por exemplo, o front-end e

o back-end de uma aplicação podem ser implementados juntos em um nó para reduzir a latência de acesso. Há também dois tipos de regras de afinidade entre pods: **requiredDuringSchedulingIgnoredDuringExecution** e **preferredDuringSchedulingIgnoredDuringExecution**.

NOTA

Para afinidade de carga de trabalho, a topologyKey não pode ser deixada em branco quando são usados requiredDuringSchedulingIgnoredDuringExecution e preferredDuringSchedulingIgnoredDuringExecution.

Suponha que o backend de uma aplicação tenha sido criado e tenha o rótulo **app=backend**.

```
$ kubectl get po -o wide
NAME                        READY   STATUS    RESTARTS   AGE     IP
NODE
backend-658f6cb858-dlrz8   1/1     Running   0          2m36s   172.16.0.67
192.168.0.100
```

Você pode configurar a seguinte regra de afinidade do pod para implementar os pods de front-end da aplicação no mesmo nó que os pods de back-end.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name:   frontend
  labels:
    app:  frontend
spec:
  selector:
    matchLabels:
      app: frontend
  replicas: 3
  template:
    metadata:
      labels:
        app:  frontend
    spec:
      containers:
      - image:  nginx:alpine
        name:  frontend
        resources:
          requests:
            cpu:  100m
            memory:  200Mi
          limits:
            cpu:  100m
            memory:  200Mi
      imagePullSecrets:
      - name: default-secret
      affinity:
        podAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
          - topologyKey: kubernetes.io/hostname
            labelSelector:
              matchExpressions:
              - key: app
                operator: In
                values:
                - backend
```

Implemente o front-end e você pode descobrir que o front-end é implantado no mesmo nó que o back-end.

```
$ kubectl create -f affinity3.yaml
deployment.apps/frontend created
```



```
$ kubectl get po -o wide
NAME                        READY   STATUS    RESTARTS   AGE     IP
NODE
backend-658f6cb858-dlrz8    1/1     Running   0          5m38s   172.16.0.67
192.168.0.100
frontend-67ff9b7b97-dsqzn   1/1     Running   0          6s      172.16.0.70
192.168.0.100
frontend-67ff9b7b97-hxm5t   1/1     Running   0          6s      172.16.0.71
192.168.0.100
frontend-67ff9b7b97-z8pdb   1/1     Running   0          6s      172.16.0.72
192.168.0.100
```

O campo **topologyKey** é usado para dividir domínios de topologia para especificar o intervalo de seleção. Se o rótulo chaves e valores de nós são os mesmos, os nós são considerados como estando no mesmo domínio topologia. Em seguida, os conteúdos definidos nas regras a seguir são selecionados. O efeito de **topologyKey** não é totalmente demonstrado no exemplo anterior porque todos os nós têm o rótulo **kubernetes.io/hostname**, ou seja, todos os nós estão dentro do intervalo.

Para ver como **topologyKey** funciona, suponha que o back-end do aplicação tenha dois pods, que estão sendo executados em nós diferentes.

```
$ kubectl get po -o wide
NAME                        READY   STATUS    RESTARTS   AGE     IP
NODE
backend-658f6cb858-5bpd6    1/1     Running   0          23m     172.16.0.40
192.168.0.97
backend-658f6cb858-dlrz8    1/1     Running   0          2m36s   172.16.0.67
192.168.0.100
```

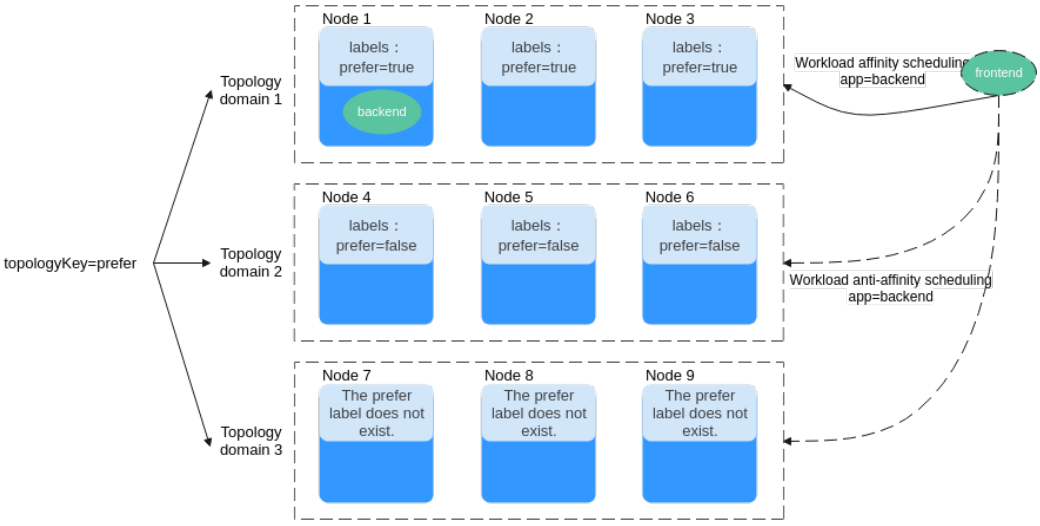
Adicione o rótulo **prefer=true** aos nós **192.168.0.97** e **192.168.0.94**.

```
$ kubectl label node 192.168.0.97 prefer=true
node/192.168.0.97 labeled
$ kubectl label node 192.168.0.94 prefer=true
node/192.168.0.94 labeled

$ kubectl get node -L prefer
NAME            STATUS   ROLES    AGE   VERSION                              PREFER
192.168.0.100   Ready    <none>   44m   v1.15.6-r1-20.3.0.2.B001-15.30.2
192.168.0.212   Ready    <none>   91m   v1.15.6-r1-20.3.0.2.B001-15.30.2
192.168.0.94    Ready    <none>   91m   v1.15.6-r1-20.3.0.2.B001-15.30.2   true
192.168.0.97    Ready    <none>   91m   v1.15.6-r1-20.3.0.2.B001-15.30.2   true
```

Se a **topologyKey** de **podAffinity** for definida como **prefer**, os domínios de topologia do nó serão divididos conforme mostrado em **Figura 5-12**.

```
      affinity:
        podAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
          - topologyKey: prefer
            labelSelector:
              matchExpressions:
              - key: app
                operator: In
                values:
                - backend
```


Figura 5-12 Domínios de topologia



Durante o agendamento, os domínios de topologia de nó são divididos com base no rótulo **prefer**. Neste exemplo, **192.168.0.97** e **192.168.0.94** são divididos no mesmo domínio de topologia. Se um pod com o rótulo **app=backend** for executado no domínio da topologia, mesmo que nem todos os nós no domínio da topologia executem o pod com o rótulo **app=backend** (neste exemplo, somente o nó **192.168.0.97** possui tal pod), **frontend** também é implementado nesse domínio de topologia (**192.168.0.97** ou **192.168.0.94**).

```
$ kubectl create -f affinity3.yaml
deployment.apps/frontend created

$ kubectl get po -o wide
NAME                        READY   STATUS    RESTARTS   AGE     IP
NODE
backend-658f6cb858-5bpd6    1/1     Running   0          26m     172.16.0.40
192.168.0.97
backend-658f6cb858-dlrz8    1/1     Running   0          5m38s   172.16.0.67
192.168.0.100
frontend-67ff9b7b97-dsqzn   1/1     Running   0          6s      172.16.0.70
192.168.0.97
frontend-67ff9b7b97-hxm5t   1/1     Running   0          6s      172.16.0.71
192.168.0.97
frontend-67ff9b7b97-z8pdb   1/1     Running   0          6s      172.16.0.72
192.168.0.97
```

Antiafinidade da carga de trabalho (podAntiAffinity)

Ao contrário dos cenários em que os pods são preferidos para serem agendados no mesmo nó, às vezes, pode ser exatamente o oposto. Por exemplo, se certos pods forem implementados juntos, eles afetarão o desempenho.

> **NOTA**
>
> Para antiafinidade de carga de trabalho, quando requiredDuringSchedulingIgnoredDuringExecution é usado, o controlador de acesso padrão LimitPodHardAntiAffinityTopology do Kubernetes exige que topologyKey só possa ser **kubernetes.io/hostname**. Para usar outra lógica de topologia personalizada, modifique ou desabilite o controlador de acesso.

O seguinte é um exemplo de definição de uma regra de antiafinidade. Essa regra divide os domínios de topologia de nó pelo rótulo **kubernetes.io/hostname**. Se um pod com o rótulo **app=frontend** já existir em um nó no domínio de topologia, os pods com o mesmo rótulo não poderão ser agendados para outros nós no domínio de topologia.


```
apiVersion: apps/v1
kind: Deployment
metadata:
  name:   frontend
  labels:
    app:  frontend
spec:
  selector:
    matchLabels:
      app: frontend
  replicas: 5
  template:
    metadata:
      labels:
        app:  frontend
    spec:
      containers:
      - image:  nginx:alpine
        name:  frontend
        resources:
          requests:
            cpu:  100m
            memory:  200Mi
          limits:
            cpu:  100m
            memory:  200Mi
      imagePullSecrets:
      - name: default-secret
      affinity:
        podAntiAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
          - topologyKey: kubernetes.io/hostname   # Topology domain of the node
            labelSelector:    # Pod label matching rule
              matchExpressions:
              - key: app
                operator: In
                values:
                - frontend
```

Crie uma regra de antiafinidade e visualize o resultado da implementação. No exemplo, os domínios de topologia de nó são divididos pelo rótulo **kubernetes.io/hostname**. Os valores de rótulo dos nós com o rótulo **kubernetes.io/hostname** são diferentes, portanto, há apenas um nó em um domínio de topologia. Se um pod de **frontend** já existir em um domínio de topologia, os pods com o mesmo rótulo não serão agendados para o domínio de topologia. Neste exemplo, existem apenas quatro nós. Portanto, há um pod que está no estado **Pending** e não pode ser agendado.

```
$ kubectl create -f affinity4.yaml
deployment.apps/frontend created

$ kubectl get po -o wide
NAME                        READY   STATUS    RESTARTS   AGE   IP
NODE
frontend-6f686d8d87-8dlsc   1/1     Running   0          18s   172.16.0.76
192.168.0.100
frontend-6f686d8d87-d6l8p   0/1     Pending   0          18s   <none>
<none>
frontend-6f686d8d87-hgcq2   1/1     Running   0          18s   172.16.0.54
192.168.0.97
frontend-6f686d8d87-q7cfq   1/1     Running   0          18s   172.16.0.47
192.168.0.212
frontend-6f686d8d87-xl8hx   1/1     Running   0          18s   172.16.0.23
192.168.0.94
```


Descrição do valor do operador

Você pode usar o campo **operator** para definir o relacionamento lógico da regra de uso. O valor do **operator** pode ser:

- **In**: o rótulo do objeto de afinidade ou antiafinidade está na lista de valores do rótulo (campo **values**).
- **NotIn**: o rótulo do objeto de afinidade ou antiafinidade não está na lista de valores do rótulo (campo **values**).
- **Exists**: o objeto de afinidade ou antiafinidade tem um nome de rótulo especificado.
- **DoesNotExist**: o objeto de afinidade ou antiafinidade não tem o nome de rótulo especificado.
- **Gt**: (disponível apenas para afinidade de nó) o valor do rótulo do nó agendado é maior que o valor da lista (comparação de cadeias).
- **Lt**: (disponível somente para afinidade de nó) o valor do rótulo do nó de agendamento é menor que o valor da lista (comparação de cadeias).

5.3.11 Manchas e tolerâncias

As tolerâncias permitem que o agendador agende pods para nós com manchas de destino. Tolerâncias trabalham com **manchas de nó**. Cada nó permite uma ou mais manchas. Se nenhuma tolerância for configurada para um pod, o agendador programará o pod com base nas políticas de mancha de nó para impedir que o pod seja programado para um nó inadequado.

A tabela a seguir mostra como as políticas e tolerâncias de manchas afetam a execução do pod.

Política de mancha	Nenhuma tolerância de mancha configurada	Tolerância de mancha configurada
NoExecute	• Pods em execução no nó serão despejados imediatamente. • Os pods inativos não serão agendados para o nó.	• Se a janela de tempo de tolerância **tolerationSeconds** não for especificada, os pods poderão ser executados nesse nó o tempo todo. • Se a tolerância da janela de tempo **tolerationSeconds** for especificada, os pods ainda serão executados no nó com manchas dentro da janela de tempo. Depois que o tempo expirar, os pods serão despejados.
PreferNoSchedule	• Pods em execução no nó não serão despejados. • Os pods inativos não serão programados para o nó **para a melhor extensão**.	Os pods podem ser executados nesse nó o tempo todo.

Política de mancha	Nenhuma tolerância de mancha configurada	Tolerância de mancha configurada
NoSchedule	● Pods em execução no nó não serão despejados. ● Os pods inativos não serão agendados para o nó.	Os pods podem ser executados nesse nó o tempo todo.

Configurar políticas de tolerância no console

Passo 1 Efetue logon no console do CCE.

Passo 2 Ao criar uma carga de trabalho, clique em **Toleration** na área **Advanced Settings**.

Passo 3 Adicione uma política de tolerância a manchas.

Tabela 5-15 Parâmetros para configurar uma política de tolerância a manchas

Parâmetro	Descrição
Taint key	Chave de uma mancha de nó
Operator	● **Equal**: **Exact match** para a chave de mancha especificada (obrigatória) e o valor de mancha. Se o valor de mancha for deixado em branco, todos as manchas com a chave igual à chave de mancha especificada serão correspondidos. ● **Exists**: **matches only** aos nós com a chave de mancha especificada. Neste caso, o valor de mancha não pode ser especificado. Se a chave de mancha for deixada em branco, todas as manchas serão toleradas.
Valor de mancha	Valor de mancha especificado se o operador for definido como **Equal**.
Taint Policy	● **All**: todas as políticas de mancha são correspondidas. ● **NoSchedule**: apenas a mancha de **NoSchedule** é correspondida. ● **PreferNoSchedule**: apenas a mancha de **PreferNoSchedule** é correspondida. ● **NoExecute**: apenas a mancha de **NoExecute** é correspondida.
Toleration Time Window	**tolerationSeconds**, que é configurável apenas quando **Taint Policy** é definida como **NoExecute**. Dentro da janela de tempo de tolerância, os pods ainda são executados no nó com manchas. Depois que o tempo expirar, os pods serão despejados.

----Fim

Política de tolerância padrão

O Kubernetes adiciona automaticamente tolerâncias para as manchas **node.kubernetes.io/not-ready** e **node.kubernetes.io/unreachable** aos pods, e define a janela de tempo de tolerância (**tolerationSeconds**) para 300s. Essas políticas de tolerância padrão indicam que, quando um dos pods anteriores é adicionado ao nó em que os pods estão sendo executados, os pods ainda podem ser executados no nó por 5 minutos.

NOTA

Quando um pod de DaemonSet é criado, nenhuma janela de tempo de tolerância será especificada para as tolerâncias adicionadas automaticamente para as manchas anteriores. Quando uma das manchas anteriores é adicionada ao nó onde o pod de DaemonSet está sendo executado, o pod de DaemonSet nunca será despejado.

```
tolerations:
  - key: node.kubernetes.io/not-ready
    operator: Exists
    effect: NoExecute
    tolerationSeconds: 300
  - key: node.kubernetes.io/unreachable
    operator: Exists
    effect: NoExecute
    tolerationSeconds: 300
```

5.3.12 Rótulos e anotações

Anotações de pod

O CCE permite que você adicione anotações a um arquivo YAML para realizar algumas funções avançadas do pod. A tabela a seguir descreve as anotações que você pode adicionar.

Tabela 5-16 Anotações do pod

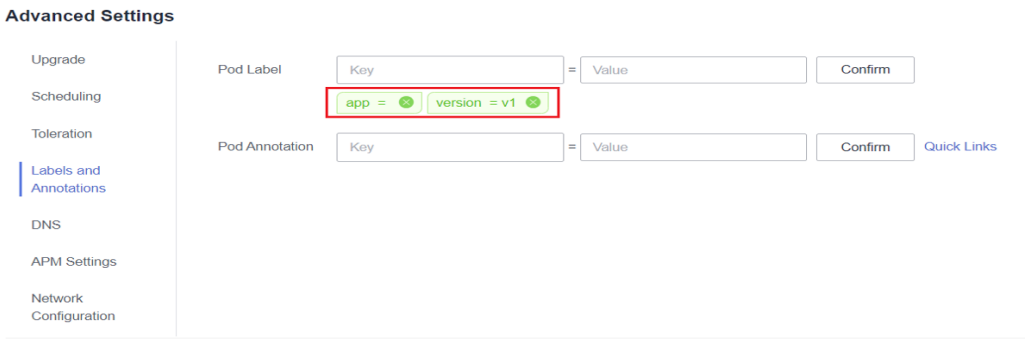
Anotação	Descrição	Valor padrão

kubernetes.AOM.log.stdout	Parâmetro de saída padrão. Se não for especificado, a saída de log padrão de todos os contêineres é informada ao AOM. Você pode coletar logs de stdout de determinados contêineres ou ignorá-los.
	Exemplo:
	● Coletar nenhum dos logs de stdout:
	`kubernetes.AOM.log.stdout: '[]'`
	● Coletar logs de stdout do container-1 e container-2:
	`kubernetes.AOM.log.stdout: '["container-1","container-2"]'`
		Nenhum

Anotação	Descrição	Valor padrão
metrics.alpha.kubernetes.io/custom-endpoints	Parâmetro para relatório de métricas de monitoramento do AOM que você especificar. Para mais detalhes, consulte **Monitoramento de métricas personalizadas no AOM**.	Nenhum
prometheus.io/scrape	Parâmetro para relatar métricas de Prometheus. Se o valor for **true**, a carga de trabalho atual relata as métricas de monitoramento. Para mais detalhes, consulte **Monitoramento de métricas personalizadas usando o Prometheus**.	Nenhum
prometheus.io/path	URL para Prometheus recolher dados. Para mais detalhes, consulte **Monitoramento de métricas personalizadas usando o Prometheus**.	/metrics
prometheus.io/port	Número da porta do ponto de extremidade para o Prometheus coletar dados. Para mais detalhes, consulte **Monitoramento de métricas personalizadas usando o Prometheus**.	Nenhum
prometheus.io/scheme	Protocolo usado pelo Prometheus para coletar dados. O valor pode ser **http** ou **https**. Para mais detalhes, consulte **Monitoramento de métricas personalizadas usando o Prometheus**.	Nenhum
kubernetes.io/ingress-bandwidth	Largura de banda de entrada de um pod. Para mais detalhes, consulte **Configuração da limitação da taxa de QoS para acesso entre pods**.	Nenhum
kubernetes.io/egress-bandwidth	Largura de banda de saída de um pod. Para mais detalhes, consulte **Configuração da limitação da taxa de QoS para acesso entre pods**.	Nenhum

Rótulos de pod

Quando você cria uma carga de trabalho no console, os seguintes rótulos são adicionados ao pod por padrão. O valor de **app** é o nome da carga de trabalho.



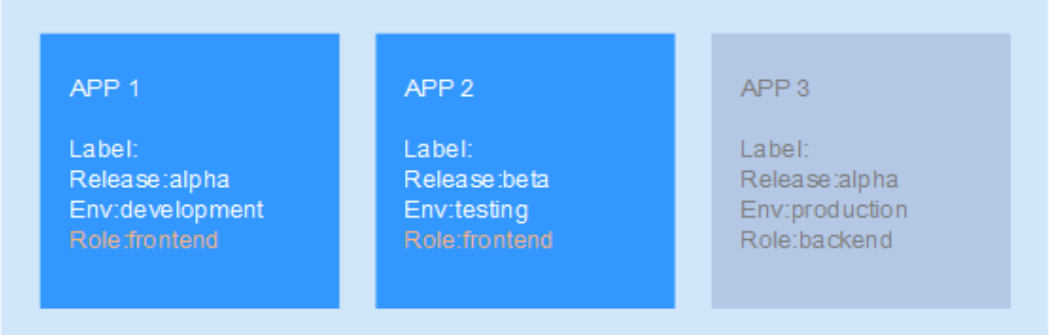
Exemplo YAML:

```
...
spec:
  selector:
    matchLabels:
      app: nginx
      version: v1
  template:
    metadata:
      labels:
        app: nginx
        version: v1
    spec:
      ...
```

Você também pode adicionar outros rótulos ao pod para agendamento de afinidade e antiafinidade. Na figura a seguir, três rótulos de pod (release, env e role) são definidos para a carga de trabalho APP 1, APP 2 e APP 3. Os valores desses rótulos variam com a carga de trabalho.

- APP 1: [release:alpha;env:development;role:frontend]
- APP 2: [release:beta;env:testing;role:frontend]
- APP 3: [release:alpha;env:production;role:backend]

Figura 5-13 Exemplo de rótulo



Por exemplo, se **key/value** estiver definido como **role/backend**, a APP 3 será selecionada para agendamento de afinidade. Para mais detalhes, consulte **Afinidade da carga de trabalho (podAffinity)**.

5.4 Acesso de um contêiner

Cenário

Se você encontrar problemas inesperados ao usar um contêiner, poderá efetuar logon no contêiner para depurá-lo.

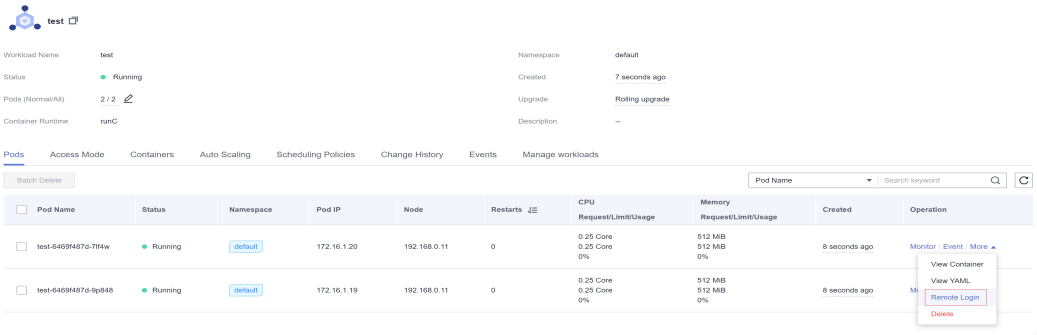
Fazer logon em um contêiner usando o CloudShell

AVISO

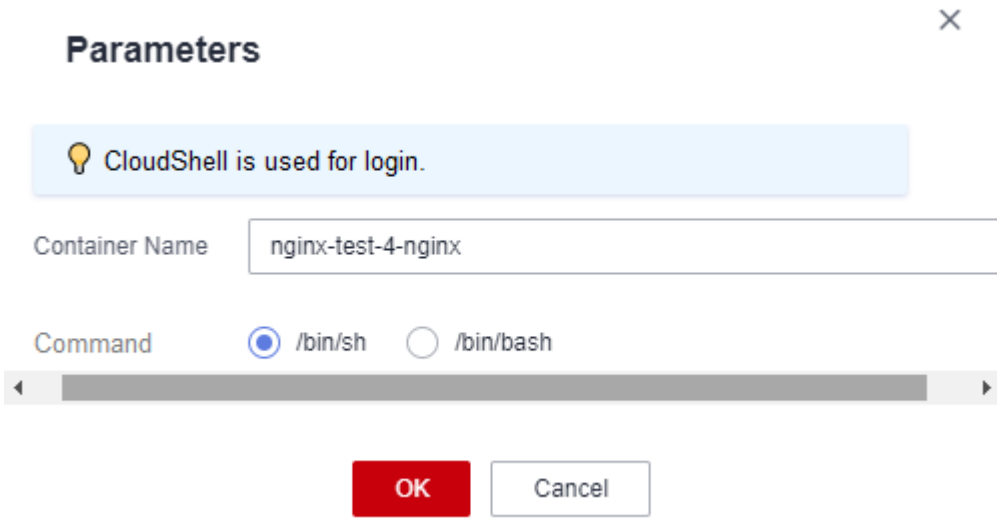
- O CloudShell é implementado com base no VPC Endpoint (VPCEP). Para usar o kubectl para acessar um cluster, configure o grupo de segurança (*Cluster name*-cce-control-*Random number*) no nó mestre do cluster para permitir o acesso à porta 5443. Por padrão, a porta 5443 permite acesso de todos os blocos CIDR. Se você tiver grupos de segurança reforçados e nenhum cluster não puder ser acessado no CloudShell, verifique se a porta 5443 permite acesso de 198.19.0.0/16.
- Atualmente, você pode usar CloudShell para fazer logon em contêineres apenas nas regiões CN North-Beijing1, CN North-Beijing4, CN East-Shanghai1, CN East-Shanghai2, CN South-Guangzhou e CN North-Ulanqab1.

Você pode encontrar a entrada de acesso na lista de pods de carga de trabalho, conforme mostrado na figura a seguir.

Figura 5-14 Acessar um contêiner



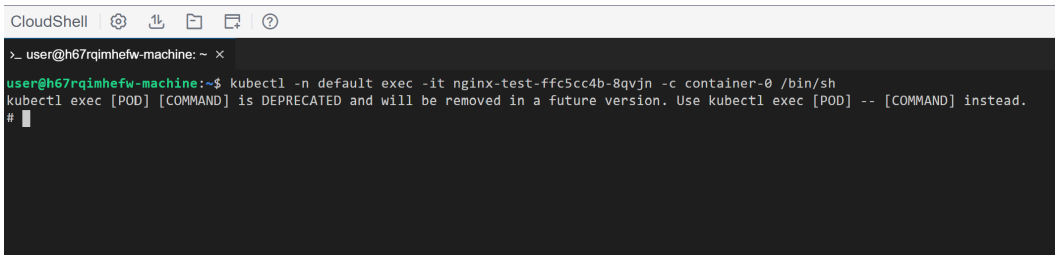
Depois de clicar em **Remote Login**, a seguinte caixa de diálogo é exibida. Selecione o contêiner que deseja acessar e o comando e clique em **OK**.



O CloudShell é iniciado, o kubectl é inicializado e o comando **kubectl exec** é executado automaticamente.

NOTA

Aguarde de 5 a 10 segundos até que o comando **kubectl exec** seja executado automaticamente.



Fazer logon em um contêiner usando o kubectl

Passo 1 Use o kubectl para se conectar ao cluster. Para mais detalhes, consulte **Conexão a um cluster usando o kubectl**.

Passo 2 Execute o seguinte comando para exibir o pod criado:

```
kubectl get pod
```

A saída de exemplo é a seguinte:

```
NAME                                READY   STATUS    RESTARTS        AGE
nginx-59d89cb66f-mhljr              1/1     Running   0               11m
```

Passo 3 Consulte o nome do contêiner no pod.

```
kubectl get po nginx-59d89cb66f-mhljr -o jsonpath='{range .spec.containers[*]}
{.name}{end}{"\n"}'
```

A saída de exemplo é a seguinte:

```
container-1
```

Passo 4 Execute o seguinte comando para efetuar logon no contêiner **container-1** no pod **nginx-59d89cb66f-mhljr**:

```
kubectl exec -it nginx-59d89cb66f-mhljr -c container-1 -- /bin/sh
```


Passo 5 Para sair do contêiner, execute o comando **exit**.

----Fim

5.5 Gerenciamento de cargas de trabalho e tarefas

Cenário

Depois que uma carga de trabalho é criada, você pode atualizar, monitorar, reverter ou excluir a carga de trabalho, bem como editar seu arquivo YAML.

Tabela 5-17 Gerenciamento de carga de trabalho/tarefas

Operação	Descrição
Monitorar	Você pode ver o uso de CPU e memória de cargas de trabalho e pods no console do CCE.
Exibir logs	Você pode exibir os logs de cargas de trabalho.
Atualizar	Você pode substituir imagens ou tags de imagem para atualizar rapidamente as Implementações, o StatefulSets e o DaemonSets sem interromper os serviços.
Editar YAML	Você pode modificar e baixar os arquivos YAML de Implementações, de StatefulSets e de pods no console do CCE. Arquivos YAML de tarefas e tarefas cronometradas só podem ser visualizados, copiados e baixados.
Reverter	Somente as Implementações podem ser revertidas.
Reimplementar	Você pode reimplementar uma carga de trabalho. Depois que a carga de trabalho for reimplementada, todos os pods na carga de trabalho serão reiniciados.
Ativar/desativar a atualização	Apenas as Implementações suportam esta operação.
Gerenciar rótulos	Os rótulos são anexados às cargas de trabalho como pares de chave-valor para gerenciar e selecionar cargas de trabalho. Tarefas e tarefas cronometradas não suportam esta operação.
Excluir	Você pode excluir uma carga de trabalho ou um trabalho que não seja mais necessário. Cargas de trabalho ou tarefas excluídas não podem ser recuperadas.
Visualizar eventos	Você pode visualizar nomes de eventos, tipos de eventos, número de ocorrências, eventos do Kubernetes, horário da primeira ocorrência e horário da última ocorrência.
Iniciar/parar	Você só pode iniciar ou parar uma tarefa cronometrada.

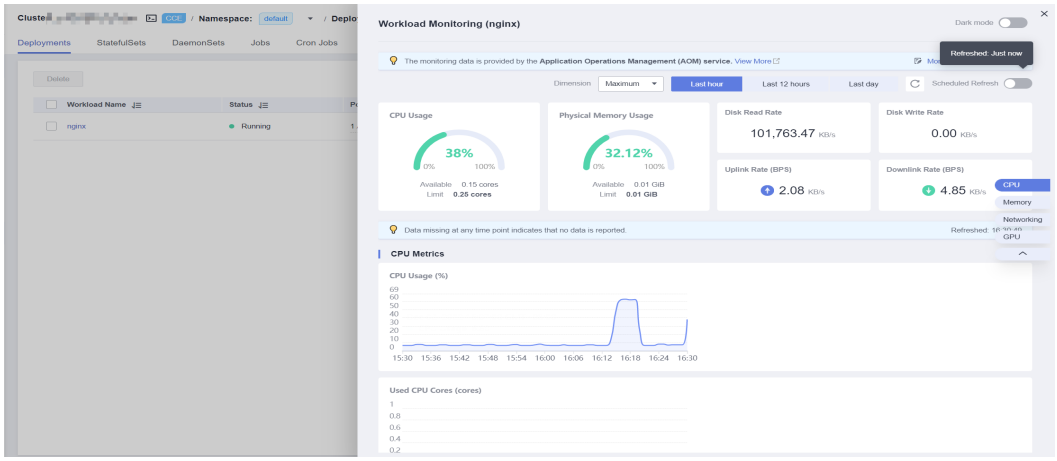
Monitorando uma carga de trabalho

Você pode visualizar o uso de CPU e memória de Implementações e pods no console do CCE para determinar as especificações de recursos que você pode precisar. Esta seção usa uma Implementação como exemplo para descrever como monitorar uma carga de trabalho.

Passo 1 Faça logon no console do CCE, vá para um cluster existente e escolha **Workloads** no painel de navegação.

Passo 2 Clique na guia **Deployments** e clique em **Monitor** da carga de trabalho de destino. Na página exibida, você pode exibir o uso da CPU e o uso da memória da carga de trabalho.

Figura 5-15 Visualizar informações de monitoramento



Passo 3 Clique no nome da carga de trabalho. Na página de guia **Pods**, clique em **Monitor** do pod de destino para visualizar o uso da CPU e da memória.

----Fim

Visualização de logs

Você pode exibir logs de Implementações, de StatefulSets e de tarefas. Esta seção usa uma Implementação como exemplo para descrever como exibir logs.

AVISO

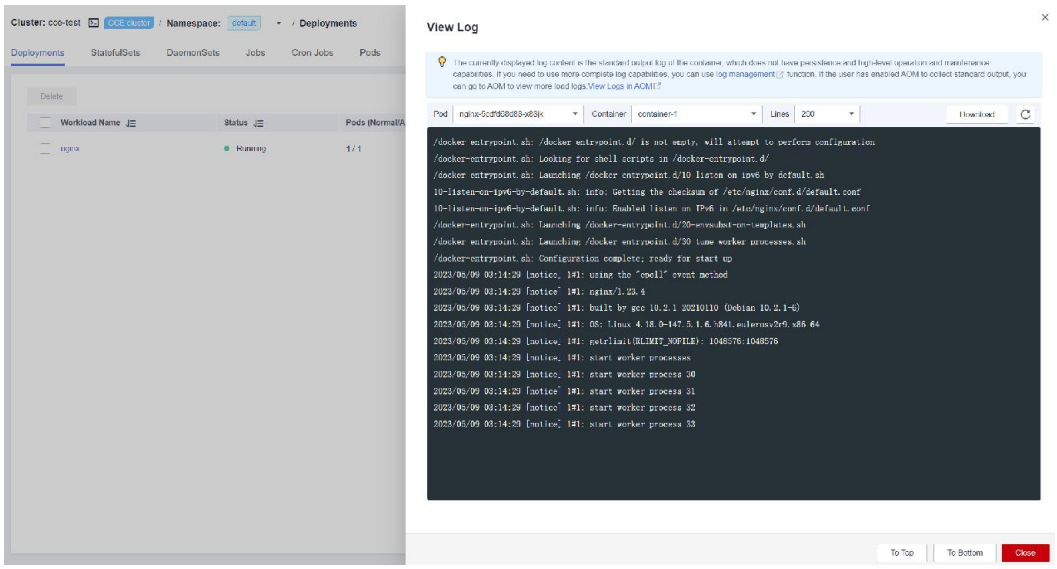
Antes de visualizar logs, certifique-se de que o tempo do navegador é o mesmo que o do servidor back-end.

Passo 1 Faça logon no console do CCE, vá para um cluster existente e escolha **Workloads** no painel de navegação.

Passo 2 Clique na guia **Deployments** e clique em **View Log** da carga de trabalho de destino.

Na janela **View Log** exibida, você pode exibir logs.

Figura 5-16 Exibir logs de uma carga de trabalho



NOTA

Os logs exibidos são logs de saída padrão de contêineres e não têm persistência e recursos avançados de O&M. Para usar recursos de log mais abrangentes, consulte **Logs**. Se a função de coleta de saída padrão estiver ativada para a carga de trabalho (ativada por padrão), você poderá acessar o AOM para exibir mais logs de carga de trabalho. Para mais detalhes, consulte **Uso do ICAgent para coletar logs de contêiner**.

----Fim

Atualizar uma carga de trabalho

Você melhora rapidamente as distribuições, o StatefulSets e o DaemonSets no console do CCE.

Esta seção usa uma Implementação como exemplo para descrever como atualizar uma carga de trabalho.

Antes de substituir uma imagem ou versão de imagem, carregue a nova imagem no serviço SWR. Para obter detalhes, consulte **Carregamento de uma imagem por meio de um cliente de mecanismo de contêiner**.

Passo 1 Faça logon no console do CCE, vá para um cluster existente e escolha **Workloads** no painel de navegação.

Passo 2 Clique na guia **Deployments** e clique em **Upgrade** da carga de trabalho de destino.

NOTA

- Cargas de trabalho não podem ser atualizadas em lotes.
- Antes de executar uma atualização de StatefulSet in-loco, você deve excluir manualmente os pods antigos. Caso contrário, o status de atualização é sempre exibido como **Processing**.

Passo 3 Atualize a carga de trabalho com base nos requisitos de serviço. O método para definir o parâmetro é o mesmo que para criar uma carga de trabalho.

Passo 4 Após a conclusão da atualização, clique em **Upgrade Workload**, confirme manualmente o arquivo YAML e submeta a atualização.

----Fim

Editar um arquivo YAML

Você pode modificar e baixar os arquivos YAML de Implementações, de StatefulSets e de pods no console do CCE. Arquivos YAML de tarefas e tarefas cronometradas só podem ser visualizados, copiados e baixados. Esta seção usa uma Implementação como exemplo para descrever como editar o arquivo YAML.

Passo 1 Faça logon no console do CCE, vá para um cluster existente e escolha **Workloads** no painel de navegação.

Passo 2 Clique na guia **Deployments** e escolha **More** > **Edit YAML** na coluna **Operation** da carga de trabalho de destino. Na caixa de diálogo exibida, modifique o arquivo YAML.

Passo 3 Clique em **OK**.

Passo 4 (Opcional) Na janela **Edit YAML**, clique em **Download** para baixar o arquivo YAML.

----Fim

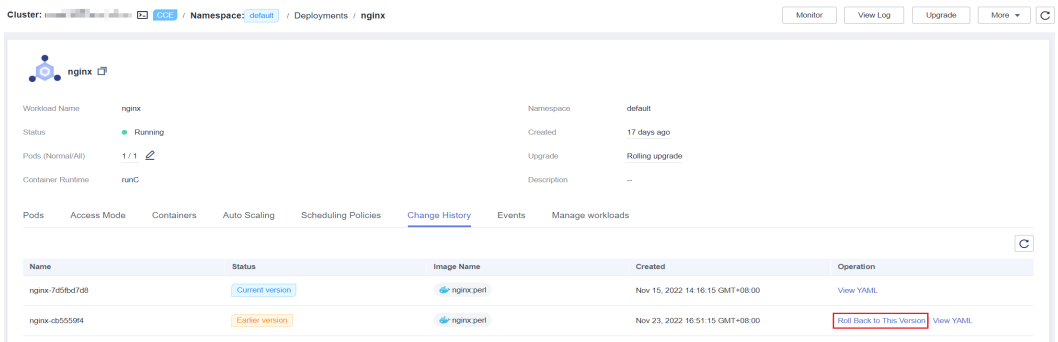
Reverter uma carga de trabalho (disponível somente para Implementações)

O CCE registra o histórico de lançamentos de todas as Implementações. Você pode reverter uma Implementação para uma versão especificada.

Passo 1 Faça logon no console do CCE, vá para um cluster existente e escolha **Workloads** no painel de navegação.

Passo 2 Clique na guia **Deployments**, escolha **More > Roll Back** na coluna **Operation** da carga de trabalho de destino.

Passo 3 Alterne para a página de guia **Change History**, clique em **Roll Back to This Version** da versão de destino, confirme manualmente o arquivo YAML e clique em **OK**.



----Fim

Reimplementar uma carga de trabalho

Depois de reimplementar uma carga de trabalho, todos os pods na carga de trabalho serão reiniciados. Esta seção usa Implementações como um exemplo para ilustrar como reimplementar uma carga de trabalho.

Passo 1 Faça logon no console do CCE, vá para um cluster existente e escolha **Workloads** no painel de navegação.

Passo 2 Clique na guia **Deployments** e escolha **More** > **Redeploy** na coluna **Operation** da carga de trabalho de destino.

Passo 3 Na caixa de diálogo exibida, clique em **Yes** para reimplementar a carga de trabalho.

----Fim

Desativar/ativar a atualização (disponível somente para Implementações)

Apenas as Implementações suportam esta operação.

- Após a atualização ser desativada, o comando de atualização pode ser entregue, mas não será aplicado aos pods.

  Se você estiver realizando uma atualização contínua, a atualização contínua será interrompida após o comando de atualização de desativação ser entregue. Neste caso, os pods novos e antigos coexistem.
- Se uma Implementação estiver sendo atualizada, ela poderá ser atualizada ou revertida. Seus pods herdarão as atualizações mais recentes da Implementação. Se forem inconsistentes, os pods são atualizados automaticamente de acordo com as informações mais recentes da Implementação.

AVISO

As Implementações no estado de atualização desativada não podem ser revertidas.

Passo 1 Faça logon no console do CCE, vá para um cluster existente e escolha **Workloads** no painel de navegação.

Passo 2 Clique na guia **Deployments** e escolha **More** > **Disable/Enable Upgrade** na coluna **Operation** da carga de trabalho.

Passo 3 Na caixa de diálogo exibida, clique em **Yes**.

----Fim

Gerenciar rótulos

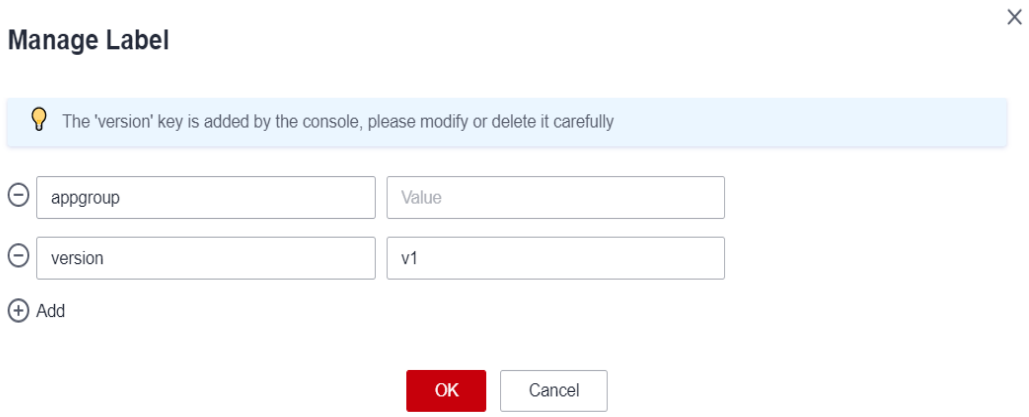
Os rótulos são pares de chave-valor e podem ser anexados a cargas de trabalho. Você pode gerenciar e selecionar cargas de trabalho por rótulos. Você pode adicionar rótulos a várias cargas de trabalho ou a uma carga de trabalho especificada.

Passo 1 Faça logon no console do CCE, vá para um cluster existente e escolha **Workloads** no painel de navegação.

Passo 2 Clique na guia **Deployments** e escolha **More** > **Manage Label** na coluna **Operation** da carga de trabalho de destino.

Passo 3 Clique em **Add**, insira uma chave e um valor e clique em **OK**.

Figura 5-17 Gerenciar rótulos



NOTA

Um par chave-valor deve conter de 1 a 63 caracteres começando e terminando com uma letra ou dígito. Apenas letras, dígitos, hifens (-), sublinhados (_) e pontos (.) são permitidos.

----Fim

Excluir uma carga de trabalho/tarefa

Você pode excluir uma carga de trabalho ou um trabalho que não seja mais necessário. Cargas de trabalho ou tarefas excluídas não podem ser recuperadas. Tenha cuidado ao realizar essa operação. Esta seção usa uma Implementação como exemplo para descrever como excluir uma carga de trabalho.

Passo 1 Faça logon no console do CCE, vá para um cluster existente e escolha **Workloads** no painel de navegação.

Passo 2 Na mesma linha da carga de trabalho que você excluirá, escolha **Operation** > **More** > **Delete**.

Leia atentamente as instruções do sistema. Uma carga de trabalho não pode ser recuperada após ser excluída. Tenha cuidado ao realizar esta operação.

Passo 3 Clique em **Yes**.

NOTA

- Se o nó onde o pod está localizado não estiver disponível ou estiver desligado e a carga de trabalho não puder ser excluída, você poderá excluir o pod à força da lista de pods na página de detalhes da carga de trabalho.
- Certifique-se de que os volumes de armazenamento a serem excluídos não sejam usados por outras cargas de trabalho. Se esses volumes forem importados ou tiverem snapshots, você só poderá desvinculá-los.

----Fim

Eventos

Esta seção usa Implementações como um exemplo para ilustrar como exibir eventos de uma carga de trabalho. Para exibir o evento de uma tarefa ou tarefa cronometrada, clique em **View Event** na coluna **Operation** da carga de trabalho de destino.

Passo 1 Faça logon no console do CCE, vá para um cluster existente e escolha **Workloads** no painel de navegação.

Passo 2 Na página de guia **Deployments**, clique na carga de trabalho de destino. Na página de guia **Pods**, clique em **View Events** para visualizar o nome do evento, o tipo de evento, o número de ocorrências, o evento do Kubernetes, a hora da primeira ocorrência e a hora da última ocorrência.

NOTA

Os dados do evento serão mantidos por uma hora e, em seguida, excluídos automaticamente.

----Fim

5.6 Tempo de execução do Kata e tempo de execução comum

A diferença mais significativa é que cada contêiner Kata (pod) é executado em uma micro-VM independente, tem um kernel de sistema operacional independente e é isolado de forma segura na camada de virtualização. Com o tempo de execução do Kata, os kernels, os recursos de computação e as redes são isolados entre os contêineres para proteger os recursos e os dados do pod de serem preempcionados e roubados por outros pods.

Os clusters do CCE Turbo permitem que você crie cargas de trabalho usando o tempo de execução comum ou o tempo de execução do Kata, conforme necessário. As diferenças entre eles são as seguintes.

Categoria	Tempo de execução do Kata	Tempo de execução comum
Tipo de nó usado para executar contêineres	Bare-metal server (BMS)	VM
Mecanismo de contêiner	containerd	Docker e containerd
Tempo de execução do contêiner	Kata	runC
Kernel do contêiner	Kernel exclusivo	Compartilhar o kernel com o host
Isolamento do contêiner	VMs leves	cgroups e namespaces
Driver de armazenamento do mecanismo de contêiner	Device Mapper	• Contêiner do Docker: OverlayFS2 • Contêiner de containerd: OverlayFS

Categoria	Tempo de execução do Kata	Tempo de execução comum
Sobrecarga do pod	Memória: 100 MiB CPU: 0,1 núcleos A sobrecarga do pod é um recurso para contabilizar os recursos consumidos pela infraestrutura do pod em cima das solicitações e limites do contêiner. Por exemplo, se **limits.cpu** for definido para 0,5 núcleos e **limits.memory** para 256 MiB para um pod, o pod solicitará CPUs de 0,6 núcleos e 356 MiB de memória.	Nenhuma
Especificações mínimas	Memória: 256 MiB CPU: 0,25 núcleos Recomenda-se que a proporção de CPU (unidade: núcleo) para a memória (unidade: GiB) estar na faixa de 1:1 a 1:8. Por exemplo, se a CPU tiver 0,5 núcleos, a memória deve variar de 512 MiB a 4 GiB.	Nenhuma
CLI do mecanismo de contêiner	crictl	● Contêiner de Docker: docker ● Contêiner de containerd: crictl
Recursos de computação de pod	Os valores de solicitação e limite devem ser os mesmos para CPU e memória.	Os valores de solicitação e limite podem ser diferentes para CPU e memória.
Rede host	Não compatível	Compatível

Para obter detalhes sobre como usar os comandos de containerd e Docker, consulte **Mecanismo de contêiner**.

6 Agendamento

6.1 Visão geral

O CCE suporta diferentes tipos de agendamento de recursos e agendamento de tarefas, melhorando o desempenho da aplicação e a utilização geral de recursos do cluster. Esta seção descreve as principais funções de agendamento de recursos de CPU, agendamento de recursos heterogêneos de GPU/NPU e agendamento de Volcano.

Agendamento da CPU

O CCE fornece políticas de CPU para alocar núcleos físicos completos de CPU para aplicações, melhorando o desempenho da aplicação e reduzindo a latência do agendamento de aplicações.

Função	Descrição	Documentação
Política de CPU	Quando muitos pods com uso intenso de CPU estão sendo executados em um nó, as cargas de trabalho podem ser migradas para diferentes núcleos de CPU. Muitas cargas de trabalho não são sensíveis a essa migração e, portanto, funcionam bem sem qualquer intervenção. Para aplicações sensíveis à CPU, você pode usar a política de CPU fornecida pelo Kubernetes para alocar núcleos dedicados a aplicações, melhorando o desempenho da aplicação e reduzindo a latência do agendamento da aplicação.	**Política de CPU**
Política de CPU aprimorada	Com base na política de vinculação de núcleo estático do Kubernetes, a política de CPU aprimorada (enhanced-static) oferece suporte a pods intermitentes (cujas solicitações e limites de CPU são inteiros positivos) e permite que determinadas CPUs priorizem esses pods, garantindo a estabilidade da aplicação.	**Política de CPU aprimorada**

Agendamento da GPU

O CCE programa recursos heterogêneos de GPU em clusters e permite que as GPUs sejam usadas em contêineres.

Função	Descrição	Documentação
Agendamento de GPU padrão no Kubernetes	Essa função permite especificar o número de GPUs que um pod solicita. O valor pode ser menor que 1 para que vários pods possam compartilhar uma GPU.	**Agendamento de GPU padrão no Kubernetes**

Agendamento da NPU

O CCE agenda recursos heterogêneos de NPU em um cluster para executar de forma rápida e eficiente a inferência e o reconhecimento de imagem.

Função	Descrição	Documentação
Agendamento de NPU	O agendamento de NPU permite que você especifique o número de NPUs que um pod solicita para fornecer recursos de NPU para cargas de trabalho.	**Agendamento de NPU**

Agendamento de Volcano

O Volcano é uma plataforma de processamento em lote baseada em Kubernetes que suporta aprendizado de máquina, aprendizado profundo, bioinformática, genômica e outros aplicativos de Big Data. Ele fornece recursos de computação de alto desempenho e de propósito geral, como agendamento de tarefas, gerenciamento de chips heterogêneos e gerenciamento de execução de tarefas.

Função	Descrição	Documentação
Agendamento de afinidade NUMA	Os alvos do Volcano para levantar a limitação para tornar a topologia NUMA do agendador consciente de modo que: ● Os pods não são agendados para os nós que a topologia NUMA não corresponde. ● Os pods são agendados para o melhor nó para a topologia NUMA.	**Agendamento de afinidade NUMA**

Implantação híbrida da nuvem nativa

A solução de implementação híbrida da nuvem nativa se concentra nos ecossistemas de Volcano e Kubernetes para ajudar os usuários a melhorar a utilização e a eficiência de recursos e reduzir custos.

Função	Descrição	Documentação
Excesso de assinaturas de recursos dinâmicos	Com base nos tipos de trabalhos on-line e off-line, o agendamento do Volcano é usado para utilizar os recursos que são solicitados, mas não usados no cluster (isto é, a diferença entre o número de recursos solicitados e o número de recursos utilizados) implementação de excesso de assinaturas de recursos e implementação híbrida e melhoria da utilização de recursos de cluster.	**Excesso de assinaturas de recursos dinâmicos**

6.2 Agendamento de CPU

6.2.1 Política de CPU

Cenários

Por padrão, o kubelet usa **cotas do CFS** para impor limites de CPU do pod. Quando o nó executa muitos pods ligados à CPU, a carga de trabalho pode mover-se para diferentes núcleos de CPU, dependendo se o pod é acelerado e quais núcleos de CPU estão disponíveis no momento do agendamento. Muitas cargas de trabalho não são sensíveis a essa migração e, portanto, funcionam bem sem qualquer intervenção. Algumas aplicações são sensíveis à CPU. Eles são sensíveis a:

- Limitação da CPU
- Comutação de contexto
- Falta cache do processador
- Acesso à memória entre soquetes
- Hyperthreads que devem ser executados na mesma placa de CPU física

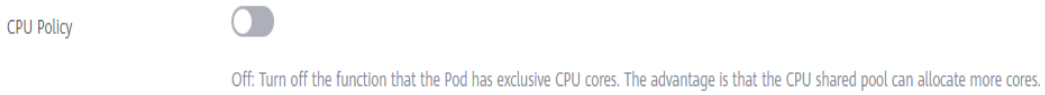
Se suas cargas de trabalho forem sensíveis a qualquer um desses itens e a afinidade de cache da CPU e a latência de agendamento afetarem significativamente o desempenho da carga de trabalho, o kubelet permitirá que políticas alternativas de gerenciamento de CPU (vinculação de CPU) determinem algumas preferências de posicionamento no nó. O gerenciador de CPU aloca preferencialmente recursos em um soquete e núcleos físicos completos para evitar interferências.

Ativar a política de gerenciamento da CPU

Uma **política de gerenciamento de CPU** é especificada pelo sinalizador de kubelet **--cpu-manager-policy**. Por padrão, o Kubernetes suporta as seguintes políticas:

- Desativada (**none**): a política padrão. A política **none** ativa explicitamente o esquema de afinidade da CPU padrão existente, não fornecendo nenhuma afinidade além do que o agendador do sistema operacional faz automaticamente.
- Ativada (**static**): a política **static** permite que os contêineres em pods **garantidos** com solicitações de GPU inteiras recebam maior afinidade e exclusividade da CPU no nó.

Ao criar um cluster, você pode configurar a política de gerenciamento da CPU em **Advanced Settings**.



Você também pode configurar a política em um pool de nós. A configuração mudará o sinalizador de kubelet **--cpu-manager-policy** no nó. Faça logon no console do CCE, clique no nome do cluster, acesse a página de detalhes do cluster e escolha **Nodes** no painel de navegação. Na página exibida, clique na guia **Node Pools**. Escolha **More** > **Manage** na coluna **Operation** do pool de nós de destino e altere o valor de **cpu-manager-policy** para **static**.

Permitir que os pods usem exclusivamente os recursos da CPU

Pré-requisitos:

- Ativar a política **static** no nó. Para mais detalhes, consulte **Ativar a política de gerenciamento da CPU**.
- Tanto as solicitações quanto os limites devem ser configurados em pods e seus valores devem ser o mesmo número inteiro.
- Se um contêiner init precisar usar exclusivamente CPUs, defina suas solicitações como as mesmas do contêiner de serviço. Caso contrário, o contêiner de serviço não herda o resultado de alocação de CPU do contêiner init, e o gerenciador de CPU reserva mais recursos de CPU do que o suposto.Para obter mais informações, consulte **Contêineres de aplicações não podem herdar CPUs de contêineres init - Política estática do gerenciador de CPU**.

Você pode usar **Política de agendamento (afinidade/antiafinidade)** para programar os pods configurados para os nós em que a política **static** está ativada. Desta forma, os pods podem usar exclusivamente os recursos da CPU.

Exemplo de YAML:

```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: test
spec:
  replicas: 1
  selector:
    matchLabels:
      app: test
  template:
    metadata:
      labels:
        app: test
    spec:
      containers:
        - name: container-1
          image: nginx:alpine
          resources:
```



```
            requests:
              cpu: 2           # The value must be an integer and must be the
same as that in limits.
              memory: 2048Mi
            limits:
              cpu: 2           # The value must be an integer and must be the
same as that in requests.
              memory: 2048Mi
      imagePullSecrets:
        - name: default-secret
```

6.2.2 Política de CPU aprimorada

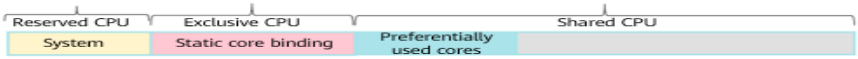
O Kubernetes fornece duas **políticas de CPU**: none e static.

- **none**: a política de CPU é desabilitada por padrão, indicando o comportamento de agendamento existente.
- **static**: a política de vinculação de núcleo da CPU estática está ativada. Essa política permite que pods com certas características de recurso recebam afinidade e exclusividade aprimoradas da CPU no nó.

Com base na política estática do Kubernetes, a política de CPU aprimorada (estática aprimorada) suporta pods intermitentes (cujas solicitações e limites de CPU devem ser inteiros positivos) e permite que eles usem preferencialmente determinadas CPUs, garantindo a estabilidade da aplicação. Exemplo:

```
...
spec:
  containers:
  - name: nginx
    image: nginx
    resources:
      limits:
        memory: "300Mi"
        cpu: "2"
      requests:
        memory: "200Mi"
        cpu: "1"
```

Esse recurso é construído sobre o agendamento otimizado da CPU no kernel do Huawei Cloud EulerOS 2.0. Quando o uso de CPU preferencialmente usado por um contêiner excede 85%, o contêiner é automaticamente alocado para outras CPUs com baixo uso para garantir a capacidade de resposta das aplicações.



NOTA

- Quando a política aprimorada de CPU está ativada, o desempenho da aplicação é melhor do que o da política **none**, mas pior do que o da política **static**.
- A CPU não seria usada exclusivamente por pods rebentáveis, ela ainda está no pool de CPUs compartilhado. Quando os pods estouráveis estão na maré baixa, outros pods podem compartilhar essa CPU.

Restrições

Para usar esse recurso, as seguintes condições devem ser atendidas:

- A versão do cluster é v1.23 ou posterior.
- O sistema operacional do nó é o Huawei Cloud EulerOS 2.0.

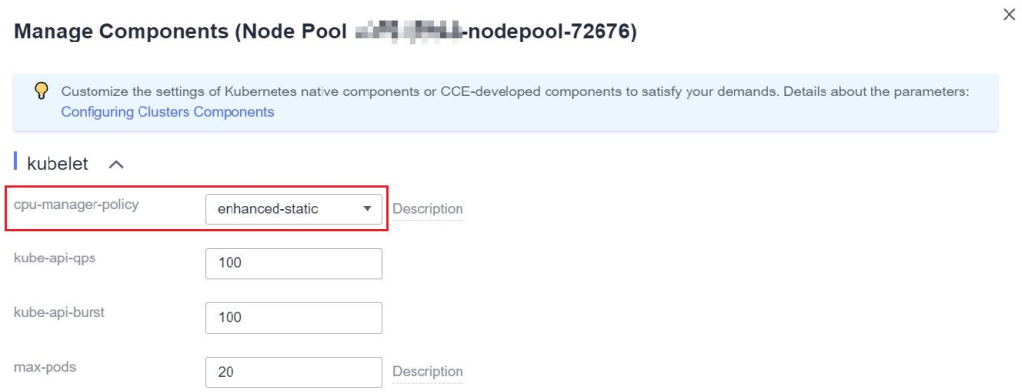
Procedimento

Passo 1 Efetue logon no console do CCE.

Passo 2 Clique no nome do cluster para acessar o console do cluster. Escolha **Nodes** no painel de navegação e clique na guia **Node Pools** à direita.

Passo 3 Selecione um pool de nós cujo SO seja o **Huawei Cloud EulerOS 2.0** e escolha **More** > **Manage** na coluna **Operation**.

Passo 4 Na janela**Manage Components** exibida, altere o valor de **cpu-manager-policy** do componente kubelet para **enhanced-static**.



Passo 5 Clique em **OK**.

----Fim

Verificação

Pegue um nó com 8 vCPUs e 32 GB de memória como exemplo. Implemente uma carga de trabalho cuja solicitação de CPU é 1 e o limite é 2 no cluster com antecedência.

Passo 1 Faça logon em um nó no pool de nós e visualize a saída **/var/lib/kubelet/cpu_manager_state**.

```
cat /var/lib/kubelet/cpu_manager_state
```

Saída do comando:

```
{"policyName":"enhanced-static","defaultCpuSet":"0,2-7","entries":{"6739f6f2-ebe5-48ae-945a-986d5d8919b9":{"container-1":"0-7,10001"}},"checksum":1638128523}
```

- Se o valor de **policyName** for **enhanced-static**, a política está configurada com êxito.

- 10000 é usado como base para o ID da CPU. Neste exemplo, 10001 indica que o ID de CPU de afinidade usado pelo contêiner é CPU 1, e 0-7 indica o conjunto de CPUs que pode ser usado pelo contêiner no pod.

Passo 2 Verifique a configuração cgroup de **cpuset.preferred_cpus** do contêiner. A saída é o ID da CPU que é usada preferencialmente.

```
cat /sys/fs/cgroup/cpuset/kubepods/burstable/pod {pod uid} / {Container ID} /
cpuset.preferred_cpus
```

- *{pod uid}* indica o UID do pod, que pode ser obtido executando o seguinte comando no host que foi conectado ao cluster usando kubectl:
  ```
  kubectl get po {pod name} -n {namespace} -ojsonpath='{.metadata.uid}{"\n"}'
  ```
  No comando anterior, *{pod name}* e *{namespace}* indicam o nome do pod e o namespace ao qual o pod pertence.
- *{Container id}* deve ser um ID de contêiner completo. Você pode executar o seguinte comando no nó em que o contêiner está sendo executado para obter o ID do contêiner:

  Pool de nós do Docker:
  ```
  docker ps --no-trunc | grep {pod name} | grep -v cce-pause | awk '{print $1}'
  ```
  Pool de nós de containerd:
  ```
  crictl ps --no-trunc | grep {pod name} | grep -v cce-pause | awk '{print $1}'
  ```

Um exemplo completo é o seguinte:

```
cat /sys/fs/cgroup/cpuset/kubepods/burstable/pod6739f6f2-
ebe5-48ae-945a-986d5d8919b9/5ba5603434b95fd22d36fba6a5f1c44eba83c18c2e1de9b52ac9b5
2e93547a13/cpuset.preferred_cpus
```

Se a saída do comando a seguir for exibida, a CPU 1 será usada preferencialmente.

```
1
```

----Fim

6.3 Agendamento de GPU

6.3.1 Agendamento de GPU padrão no Kubernetes

Você pode usar GPUs em contêineres do CCE.

Pré-requisitos

- Um nó de GPU foi criado. Para mais detalhes, consulte **Criação de um nó**.
- O gpu-device-plugin (anteriormente complemento gpu-beta) foi instalado. Durante a instalação, selecione o driver da GPU no nó. Para mais detalhes, consulte **Suíte IA do CCE (GPU NVIDIA)**.
- gpu-device-plugin monta o diretório do driver para **/usr/local/nvidia/lib64**. Para usar recursos de GPU em um contêiner, adicione **/usr/local/nvidia/lib64** à variável de ambiente **LD_LIBRARY_PATH**.

  Geralmente, você pode usar qualquer um dos seguintes métodos para adicionar um arquivo:

  a. Configure a variável de ambiente **LD_LIBRARY_PATH** no Dockerfile usada para criar uma imagem. (Recomendado)
  ```
  ENV LD_LIBRARY_PATH /usr/local/nvidia/lib64:$LD_LIBRARY_PATH
  ```


b. Configure a variável de ambiente **LD_LIBRARY_PATH** no comando de inicialização da imagem.

```
/bin/bash -c "export LD_LIBRARY_PATH=/usr/local/nvidia/
lib64:$LD_LIBRARY_PATH && ..."
```

c. Defina a variável de ambiente **LD_LIBRARY_PATH** ao criar uma carga de trabalho. (Certifique-se de que essa variável não esteja configurada no contêiner. Caso contrário, será sobrescrito.)

```
          env:
            - name: LD_LIBRARY_PATH
              value: /usr/local/nvidia/lib64
```

Usar GPUs

Crie uma carga de trabalho e solicite GPUs. Você pode especificar o número de GPUs da seguinte forma:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: gpu-test
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: gpu-test
  template:
    metadata:
      labels:
        app: gpu-test
    spec:
      containers:
      - image: nginx:perl
        name: container-0
        resources:
          requests:
            cpu: 250m
            memory: 512Mi
            nvidia.com/gpu: 1   # Number of requested GPUs
          limits:
            cpu: 250m
            memory: 512Mi
            nvidia.com/gpu: 1   # Maximum number of GPUs that can be used
      imagePullSecrets:
      - name: default-secret
```

nvidia.com/gpu especifica o número de GPUs a serem solicitadas. O valor pode ser inferior a **1**. Por exemplo, **nvidia.com/gpu: 0.5** indica que vários pods compartilham uma GPU. Nesse caso, todos os recursos de GPU solicitados vêm da mesma placa de GPU.

NOTA

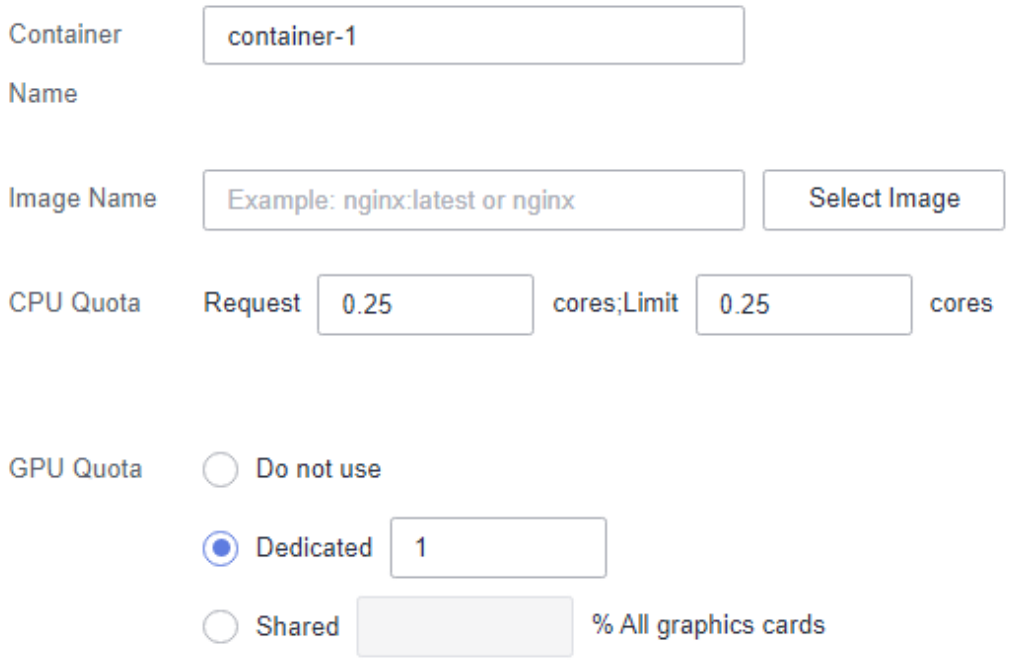
Quando você usa **nvidia.com/gpu** para especificar o número de GPUs, os valores de solicitações e limites devem ser os mesmos.

Depois que **nvidia.com/gpu** for especificado, as cargas de trabalho não serão agendadas para nós sem GPUs. Se o nó estiver carente de GPU, serão relatados eventos do Kubernetes semelhantes aos seguintes:

- 0/2 nodes are available: 2 Insufficient nvidia.com/gpu.
- 0/4 nodes are available: 1 InsufficientResourceOnSingleGPU, 3 Insufficient nvidia.com/gpu.

Para usar GPUs no console do CCE, selecione a cota de GPU e especifique a porcentagem de GPUs reservada para o contêiner ao criar uma carga de trabalho.

Figura 6-1 Usar GPUs



Rótulos de nó de GPU

O CCE rotulará os nós habilitados para GPU depois que eles forem criados. Diferentes tipos de nós habilitados para GPU têm rótulos diferentes.

```
$ kubectl get node -L accelerator
NAME           STATUS   ROLES    AGE
VERSION                                    ACCELERATOR
10.100.2.179   Ready    <none>   8m43s   v1.19.10-r0-
CCE21.11.1.B006-21.11.1.B006   nvidia-t4
```

Ao usar GPUs, você pode ativar a afinidade entre pods e nós com base em rótulos para que os pods possam ser agendados para os nós corretos.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: gpu-test
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: gpu-test
  template:
    metadata:
      labels:
        app: gpu-test
    spec:
      nodeSelector:
        accelerator: nvidia-t4
      containers:
      - image: nginx:perl
```



```
        name: container-0
        resources:
          requests:
            cpu: 250m
            memory: 512Mi
            nvidia.com/gpu: 1   # Number of requested GPUs
          limits:
            cpu: 250m
            memory: 512Mi
            nvidia.com/gpu: 1   # Maximum number of GPUs that can be used
      imagePullSecrets:
      - name: default-secret
```

6.4 Agendamento de NPU

Você pode usar NPUs em contêineres do CCE.

Pré-requisitos

- Um nó de NPU foi criado. Para mais detalhes, consulte **Criação de um nó**.
- O huawei-npu foi instalado. Para mais detalhes, consulte **Suíte de IA do CCE (Ascend NPU)**.

Usar NPUs

Crie uma carga de trabalho e solicite NPUs. Você pode especificar o número de NPUs da seguinte forma:

```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: npu-test
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: npu-test
  template:
    metadata:
      labels:
        app: npu-test
    spec:
      containers:
        - name: container-0
          image: nginx:perl
          resources:
            limits:
              cpu: 250m
              huawei.com/ascend-310: '1'
              memory: 512Mi
            requests:
              cpu: 250m
              huawei.com/ascend-310: '1'
              memory: 512Mi
      imagePullSecrets:
      - name: default-secret
```

Especifique o número de NPUs a serem solicitados em **huawei.com/ascend-310**.

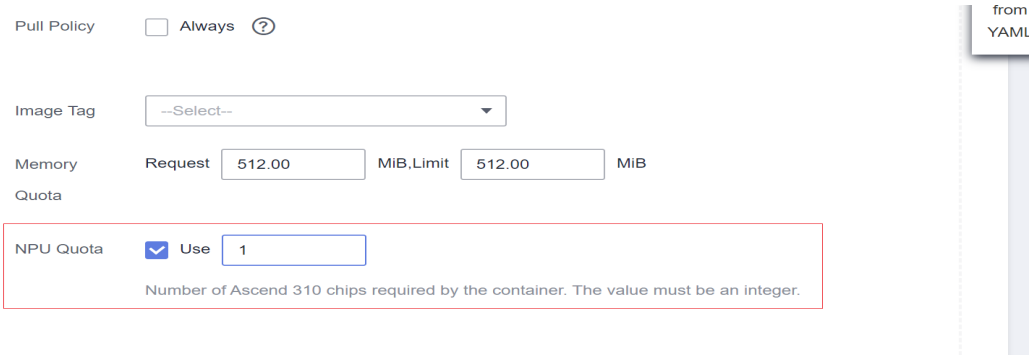
NOTA

Quando você usa **huawei.com/ascend-310** para especificar o número de NPUs, os valores de solicitações e limites devem ser os mesmos.

Depois que **huawei.com/ascend-310** for especificado, as cargas de trabalho serão agendadas apenas para nós com NPUs. Se as NPUs forem insuficientes, um evento do Kubernetes semelhante a "0/2 nodes are available: 2 Insufficient huawei.com/ascend-310." será relatado.

Para usar NPUs no console do CCE, selecione a cota de NPU e especifique o número de chips Ascend 310 a serem usados ao criar uma carga de trabalho.

Figura 6-2 Usar NPUs



Rótulos de nó de NPU

O CCE rotulará os nós habilitados para NPU que estão prontos para uso.

```
$ kubectl get node -L accelerator/huawei-npu
NAME           STATUS   ROLES    AGE
VERSION                                    HUAWEI-NPU
10.100.2.59    Ready    <none>   2m18s   v1.19.10-r0-
CCE21.11.1.B006-21.11.1.B006   ascend-310
```

Ao usar NPUs, você pode ativar a afinidade entre pods e nós com base em rótulos para que os pods possam ser programados para os nós corretos.

```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: npu-test
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: npu-test
  template:
    metadata:
      labels:
        app: npu-test
    spec:
      nodeSelector:
        accelerator/huawei-npu: ascend-310
      containers:
        - name: container-0
          image: nginx:perl
          resources:
            limits:
              cpu: 250m
              huawei.com/ascend-310: '1'
              memory: 512Mi
            requests:
              cpu: 250m
              huawei.com/ascend-310: '1'
              memory: 512Mi
      imagePullSecrets:
      - name: default-secret
```


6.5 Agendamento de Volcano

6.5.1 Binpack

Binpack é um complemento de agendamento de pods que permite que o agendador programe preferencialmente pods para nós com alta alocação de recursos. Isso reduz os fragmentos de recursos em cada nó e melhora a utilização dos recursos do cluster.

Pré-requisitos

- Um cluster de v1.19 ou posterior está disponível. Para mais detalhes, consulte **Compra de um cluster do CCE**.
- O complemento Volcano foi instalado. Para mais detalhes, consulte **Volcano scheduler**.

Recursos

Binpack visa preencher tantos nós existentes quanto possível (tente não alocar nós em branco). Na implementação concreta, o algoritmo de agendamento Binpack pontua os nós que podem ser entregues, e uma pontuação mais alta indica uma maior taxa de utilização de recursos dos nós. Binpack pretende agendar centralmente as cargas de trabalho da aplicação em alguns nós em um cluster, o que facilita o dimensionamento automático dos nós do cluster.

O complemento Binpack funciona com outros complementos de agendamento do agendador para marcar nós. Você pode personalizar o peso geral do complemento e o peso de cada recurso para modificar a influência na pontuação do nó. Ao usar o Binpack para calcular as pontuações dos nós, o agendador considera recursos estendidos, como CPUs, memória e GPUs solicitados pelos pods, e calcula as pontuações com base nos pesos configurados para cada recurso.

Implementação de algoritmo

Um nó é pontuado com base no peso geral do complemento Binpack e no peso de cada recurso. Cada tipo de recurso solicitado pelos pods a serem agendados é pontuado. Tomando CPUs como exemplo, a pontuação da CPU é calculada usando a seguinte fórmula:

CPU.weight x (Requested + Used)/Allocatable

Um peso maior da CPU leva a uma pontuação mais alta. Um maior uso de recursos de um nó leva a uma pontuação de nó mais alta. A mesma regra se aplica a recursos de memória e GPU. Os parâmetros na fórmula para marcar um recurso são os seguintes:

- **CPU.weight**: peso da CPU personalizado
- **Requested**: recursos de CPU solicitados pelos pods a serem agendados
- **Used**: recursos de CPU que foram usados no nó atual
- **Allocatable**: recursos totais da CPU disponíveis no nó atual

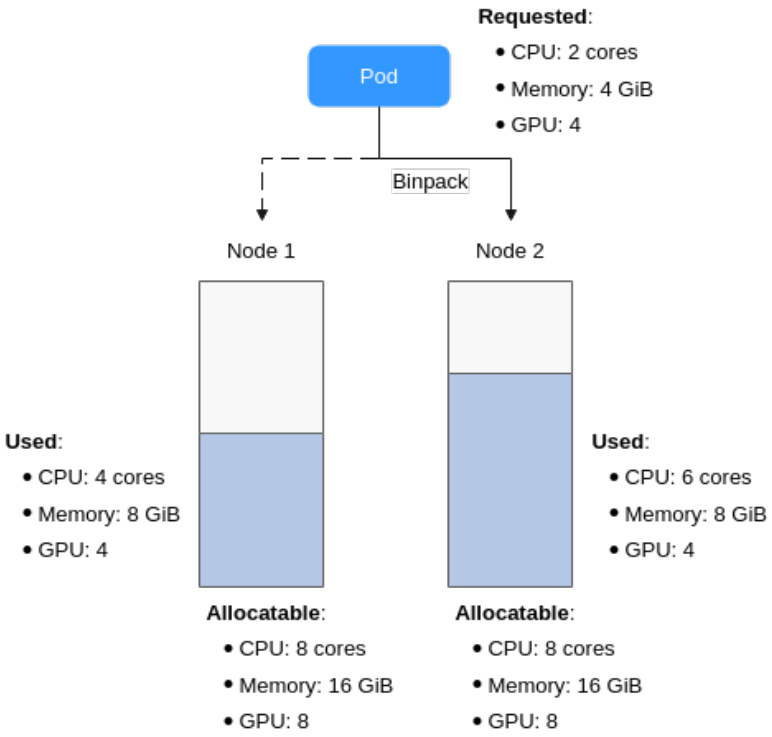
O complemento Binpack calcula a pontuação de um nó usando a seguinte fórmula:

Binpack.weight x (CPU.score + Memory.score + GPU.score)/(CPU.weight + Memory.weight + GPU.weight) x 100

Um peso de Binpack maior leva a uma pontuação mais alta. Um maior peso de recursos leva a uma maior influência na pontuação do nó. Os parâmetros na fórmula para marcar um nó são os seguintes:

- **Binpack.weight**: peso do Binpack
- **CPU.score**: pontuação de CPU calculada; **CPU.weight**: peso de CPU personalizado
- **Memory.score**: pontuação de memória calculada; **Memory.weight**: peso de memória personalizado
- **GPU.score**: pontuação calculada da GPU; **GPU.weight**: peso personalizado da GPU

Figura 6-3 Exemplo de Binpack



Como mostrado na figura, há dois nós no cluster. Quando os pods precisam ser agendados, a política de Binpack pontua os dois nós separadamente.

1. A pontuação para o nó 1 é a seguinte:

   Cada recurso é pontuado usando a seguinte fórmula: CPU.weight x (Requested + Used)/ Allocatable

   – Pontuação da CPU: 1 x (2 + 4)/8 = 0,75
   – Pontuação de memória: 1 x (4 + 8)/16 = 0,75
   – Pontuação da GPU: 2 x (4 + 4)/8 = 1

   A pontuação total de cada nó é calculada usando a seguinte fórmula: Binpack.weight x (CPU.score + Memory.score + GPU.score)/(CPU.weight + Memory.weight + GPU.weight) x 100

   Pontuação do nó 1: 5 x (0,75 + 0,75 + 1)/(1 + 1 + 2) x 100 = 312,5
2. A pontuação para o nó 2 é a seguinte:

- Pontuação da CPU: 1 x (2 + 6)/8 = 1
- Pontuação de memória: 1 x (4 + 8)/16 = 0,75
- Pontuação da GPU: 2 x (4 + 4)/8 = 1

Pontuação do nó 2: 5 x (1 + 0.75 + 1)/(1 + 1 + 2) x 100 = 343,75

Os resultados do cálculo mostram que a pontuação do nó 2 é maior que a do nó 1. De acordo com a política de Binpack, novos pods serão preferencialmente agendados para o nó 2.

Procedimento

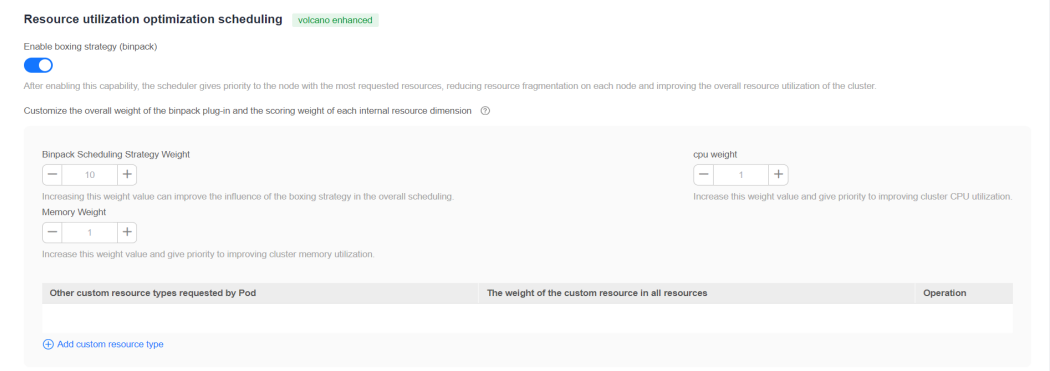
Depois que o Volcano é instalado, a política de Binpack entra em vigor por padrão. Se a configuração padrão não puder atender aos seus requisitos, você poderá personalizar o peso do complemento Binpack e o peso de cada recurso na página **Scheduling Configuration**. Para fazer isso, execute as seguintes operações:

Passo 1 Efetue logon no console do CCE.

Passo 2 Clique no nome do cluster para acessar o console do cluster. Escolha **Settings** no painel de navegação e clique na guia **Scheduling Configuration**.

Passo 3 Na área **Resource utilization optimization scheduling**, modifique as configurações de Binpack.

Figura 6-4 Agendamento da otimização da utilização de recursos



Passo 4 Clique em **Confirm**.

----Fim

6.5.2 Reagendador

O agendamento em um cluster é o processo de vinculação de pods pendentes a nós e é realizado por um componente chamado kube-scheduler ou Volcano Scheduler. O agendador usa uma série de algoritmos para calcular o nó ideal para executar pods. No entanto, os clusters do Kubernetes são dinâmicos e seu estado muda ao longo do tempo. Por exemplo, se um nó precisar ser mantido, todos os pods no nó serão despejados para outros nós. Após a conclusão da manutenção, os pods despejados não retornarão automaticamente ao nó, pois o agendamento não será acionado quando um pod for vinculado a um nó. Devido a essas alterações, a carga de um cluster pode ser desequilibrada após a execução do cluster por um período de tempo.

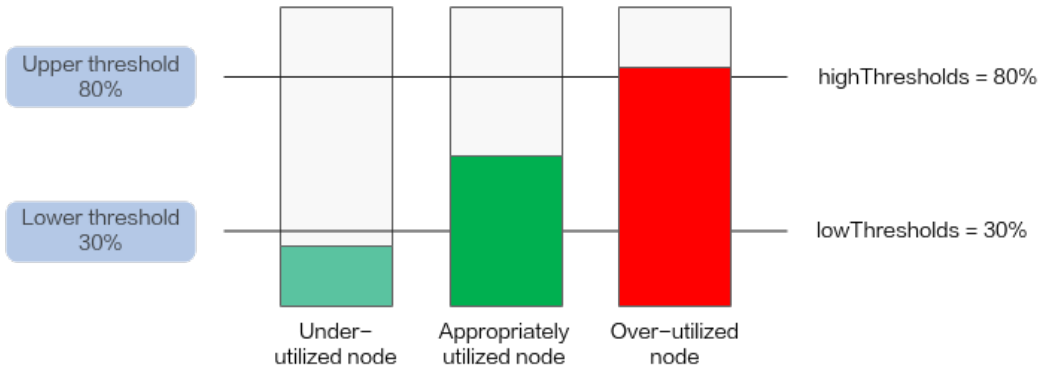
O CCE resolveu esse problema usando o Volcano scheduler para expulsar os pods que não estão em conformidade com a política configurada para que os pods possam ser reagendados. Dessa forma, a carga do cluster é balanceada e a fragmentação de recursos é minimizada.

Recursos de reagendamento

Reagendamento com reconhecimento de carga

Durante o gerenciamento de cluster do Kubernetes, os nós sobreutilizados são devido ao alto uso de CPU ou memória, o que afeta a execução estável de pods nesses nós e aumenta a probabilidade de falhas de nó. Para equilibrar dinamicamente o uso de recursos entre nós em um cluster, é necessária uma visualização de recursos de cluster com base nas métricas de monitoramento de nós. Durante o gerenciamento de cluster, o monitoramento em tempo real pode ser usado para detectar problemas como alto uso de recursos em um nó, falhas de nó e número excessivo de pods em um nó, para que o sistema possa tomar medidas prontamente, por exemplo, migrando alguns pods de um nó sobreutilizado para nós subutilizados.

Figura 6-5 Reagendamento com reconhecimento de carga



Ao usar esse complemento, verifique se o valor de **highThresholds** é maior que o valor de **lowThresholds**. Caso contrário, o reagendador não poderá funcionar.

- **Appropriately utilized node**: um nó cujo uso de recursos é maior ou igual a 30% e menor ou igual a 80%. O uso de recursos de nós adequadamente utilizados está dentro do intervalo esperado.
- **Over-utilized node**: um nó cujo uso de recursos é superior a 80%. Alguns pods serão despejados de nós sobreutilizados para reduzir seu uso de recursos para ser menor ou igual a 80%. O reagendador agendará os pods despejados para nós subutilizados.
- **Under-utilized node**: um nó cujo uso de recursos é inferior a 30%.

HighNodeUtilization

Essa política encontra nós que são subutilizados e despeja pods dos nós na esperança de que esses pods sejam agendados de forma compacta em menos nós. Esta política deve ser usada com a política binpack do Volcano scheduler ou a política MostAllocated do kube-scheduler scheduler. Os limites podem ser configurados para CPU e memória.

Pré-requisitos

- Um cluster de v1.19.16 ou posterior está disponível. Para mais detalhes, consulte **Compra de um cluster do CCE**.

- Volcano de v1.11.5 ou posterior foi instalado no cluster. Para mais detalhes, consulte **Volcano scheduler**.

Restrições

- Os pods precisam ser reprogramados usando um agendador, e nenhum agendador pode rotular pods ou nós. Portanto, um pod despejado pode ser reagendado para o nó original.
- O reagendamento não oferece suporte à antiafinidade entre pods. Um pod despejado está em relação à antiafinidade com outros pods em execução. Portanto, o agendador ainda pode agendar o pod de volta para o nó de onde o pod foi despejado.
- Ao configurar o reagendamento com reconhecimento de carga, é necessário habilitar o agendamento com reconhecimento de carga no Volcano scheduler. Ao configurar HighNodeUtilization, é necessário habilitar o reagendamento binpack no Volcano scheduler.

Configurar uma política de reagendamento com reconhecimento de carga

Ao configurar uma política do reagendador com reconhecimento de carga, faça o seguinte para habilitar o agendamento com reconhecimento de carga no Volcano scheduler:

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster. Escolha **Add-ons** no painel de navegação, localize **Volcano Scheduler** à direita e clique em **Install** ou **Edit**.

Passo 2 Na área **Parameters**, modifique **Advanced Settings** para configurar a política do reagendador com reconhecimento de carga e habilitar o complemento **usage** (um complemento de Volcano de código aberto).

```
{
  "colocation_enable": "",
  "default_scheduler_conf": {
    "actions": "allocate, backfill",
    "tiers": [
      {
        "plugins": [
          {
            "name": "priority"
          },
          {
            "enablePreemptable": false,
            "name": "gang"
          },
          {
            "name": "conformance"
          },
          {
            "name": "usage",
            "arguments": {
              "usage.weight": 5,
              "thresholds": {
                "CPUUsageAvg.5m": 60,
                "MEMUsageAvg.5m": 65
              }
            }
          }
        ]
      },
      {
        "plugins": [
          {
            "enablePreemptable": false,
            "name": "drf"
```



```
          },
          {
            "name": "predicates"
          },
          {
            "name": "nodeorder"
          }
        ]
      },
      {
        "plugins": [
          {
            "name": "cce-gpu-topology-predicate"
          },
          {
            "name": "cce-gpu-topology-priority"
          },
          {
            "name": "cce-gpu"
          }
        ]
      },
      {
        "plugins": [
          {
            "name": "nodelocalvolume"
          },
          {
            "name": "nodeemptydirvolume"
          },
          {
            "name": "nodeCSIscheduling"
          },
          {
            "name": "networkresource"
          }
        ]
      }
    ]
  },
  "deschedulerPolicy": {
    "profiles": [
      {
        "name": "ProfileName",
        "pluginConfig": [
          {
            "args": {
              "ignorePvcPods": true,
              "nodeFit": true,
              "priorityThreshold": {
                "value": 100
              }
            },
            "name": "DefaultEvictor"
          },
          {
            "args": {
              "duration": "2m",
              "evictableNamespaces": {
                "exclude": ["kube-system"]
              },
              "metrics": {
                "address": "**.**.**.**:9090",
                "type": "prometheus"
              },
              "targetThresholds": {
                "cpu": 80,
                "memory": 85
              },
```



```
              "thresholds": {
                "cpu": 30,
                "memory": 30
              }
            },
            "name": "LoadAware"
          }
        ],
        "plugins": {
          "balance": {
            "enabled": ["LoadAware"]
          }
        }
      }
    ]
  },
  "descheduler_enable": "true",
  "deschedulingInterval": "10m"
}
```

Tabela 6-1 Parâmetros-chave de uma política do reagendamento de cluster

Parâmetro	Descrição
descheduler_enable	Se deve ativar uma política de reagendador de clusters. ● **true**: a política do reagendador de cluster está ativada. ● **falso**: a política do reagendador de cluster está desativada.
deschedulingInterval	Período de reagendamento.
deschedulerPolicy	Política do reagendador do cluster. Para mais detalhes, consulte **Tabela 6-2**.

Tabela 6-2 Parâmetros de deschedulerPolicy

Parâmetro	Descrição
profiles.[].plugins.balance.enable.[]	Política do reagendador para um cluster. **LoadAware**: uma política de reagendamento com reconhecimento de carga é usada.
profiles.[].pluginConfig.[].name	Configuração de uma política do reagendador com reconhecimento de carga. Opções: ● **DefaultEvictor**: política de despejo padrão ● **LoadAware**: uma política de reagendamento com reconhecimento de carga

Parâmetro	Descrição

profiles.[].pluginConfig.[].args	Configuração da política do reagendador de um cluster.
	● Configurações para a política **DefaultEvictor**:
	– **ignorePvcPods**: se os pods de PVC devem ser ignorados ou despejados. O valor **true** indica que os pods são ignorados, e o valor **false** indica que os pods são despejados. Essa configuração não diferencia os tipos de PVC (PVs locais, SFS ou EVS).
	– **nodeFit**: se deve considerar as configurações de agendamento existentes, como afinidade de nó e mancha no nó durante o agendamento. O valor **true** indica que as configurações de agendamento existentes serão consideradas e o valor **false** indica que elas serão ignoradas.
	– **priorityThreshold**: definição de prioridade. Se a prioridade de um pod for maior ou igual ao valor deste parâmetro, o pod não será despejado. Exemplo:
	`{ "value": 100 }`
	● Configurações para a política **LoadAware**:
	– **duration**: período para obter o uso médio de CPU e memória do Prometheus.
	– **evictableNamespaces**: espaços de nomes onde a política de despejo entra em vigor. O valor padrão são os namespaces diferentes do kube-system. Exemplo:
	`{ "exclude": ["kube-system"] }`
	– **metrics**: monitorando a coleta de dados, onde o endereço e o tipo de coleta devem ser especificados. Apenas **Prometheus** é suportado. Exemplo:
	`{ "address": "10.247.119.103:9090", "type": "prometheus" }`
	– **targetThresholds**: limite para despejar pods de um nó. Quando o uso de CPU ou memória de um nó é maior que o limite, os pods no nó serão despejados. Exemplo:
	`{ "cpu": 60, "memory": 65 }`
	– **thresholds**: limite para um nó executar pods. Se o valor do nó for menor que o limite, o nó permitirá que pods despejados sejam executados. Exemplo:
	`{ "cpu": 30, "memory": 30 }`

Passo 3 Clique em **OK**.

----Fim

Configurar uma política HighNodeUtilization

Ao configurar uma política HighNodeUtilization, faça o seguinte para habilitar a política de agendamento binpack no Volcano scheduler:

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster. Escolha **Add-ons** no painel de navegação, localize **Volcano Scheduler** à direita e clique em **Install** ou **Edit**.

Passo 2 Na área **Parameters**, modifique **Advanced Settings** para configurar a política HighNodeUtilization.

```
{
  "colocation_enable": "",
  "default_scheduler_conf": {
    "actions": "allocate, backfill",
    "tiers": [
      {
        "plugins": [
          {
            "name": "priority"
          },
          {
            "enablePreemptable": false,
            "name": "gang"
          },
          {
            "name": "conformance"
          },
          {
            "arguments": {
              "binpack.weight": 5
            },
            "name": "binpack"
          }
        ]
      },
      {
        "plugins": [
          {
            "enablePreemptable": false,
            "name": "drf"
          },
          {
            "name": "predicates"
          },
          {
            "name": "nodeorder"
          }
        ]
      },
      {
        "plugins": [
          {
            "name": "cce-gpu-topology-predicate"
          },
          {
            "name": "cce-gpu-topology-priority"
          },
          {
            "name": "cce-gpu"
          }
        ]
```



```
      },
      {
        "plugins": [
          {
            "name": "nodelocalvolume"
          },
          {
            "name": "nodeemptydirvolume"
          },
          {
            "name": "nodeCSIscheduling"
          },
          {
            "name": "networkresource"
          }
        ]
      }
    ]
  },
  "deschedulerPolicy": {
    "profiles": [
      {
        "name": "ProfileName",
        "pluginConfig": [
          {
            "args": {
              "ignorePvcPods": true,
              "nodeFit": true,
              "priorityThreshold": {
                "value": 100
              }
            },
            "name": "DefaultEvictor"
          },
          {
            "args": {
              "evictableNamespaces": {
                "exclude": ["kube-system"]
              },
              "thresholds": {
                "cpu": 25,
                "memory": 25
              }
            },
            "name": "HighNodeUtilization"
          }
        ],
        "plugins": {
          "balance": {
            "enabled": ["HighNodeUtilization"]
          }
        }
      }
    ]
  },
  "descheduler_enable": "true",
  "deschedulingInterval": "10m"
}
```

Tabela 6-3 Parâmetros principais de uma política do reagendamento de cluster

Parâmetro	Descrição
descheduler_enable	Se deve ativar uma política de reagendador de clusters. ● **true**: a política do reagendador de cluster está ativada. ● **false**: a política do reagendador de cluster está desativada.

Parâmetro	Descrição
deschedulingInterval	Período de reagendamento.
deschedulerPolicy	Política do reagendador do cluster. Para mais detalhes, consulte **Tabela 6-4**.

Tabela 6-4 Parâmetros de deschedulerPolicy

Parâmetro	Descrição
profiles.[].plugins.balance.enable.[]	Política do reagendador para um cluster. **HighNodeUtilization**: a política para minimizar a CPU e fragmentos de memória é usada.
profiles.[].pluginConfig.[].name	Configuração de uma política do reagendador com reconhecimento de carga. Opções: ● **DefaultEvictor**: política de despejo padrão ● **HighNodeUtilization**: política para minimizar fragmentos de CPU e memória

Parâmetro	Descrição

profiles.[].pluginConfig.[].args	Configuração da política do reagendador de um cluster.
	● Configurações para a política **DefaultEvictor**:
	– **ignorePvcPods**: se os pods de PVC devem ser ignorados ou despejados. O valor **true** indica que os pods são ignorados, e o valor **false** indica que os pods são despejados.
	– **nodeFit**: se deve considerar as configurações de agendamento existentes, como afinidade de nó e mancha no nó durante o agendamento. O valor **true** indica que as configurações de agendamento existentes serão consideradas e o valor **false** indica que elas serão ignoradas.
	– **priorityThreshold**: definição de prioridade. Se a prioridade de um pod for maior ou igual ao valor deste parâmetro, o pod não será despejado. Exemplo:
	`{ "value": 100 }`
	● Configurações para a política **HighNodeUtilization**:
	– **evictableNamespaces**: espaços de nomes onde a política de despejo entra em vigor. O valor padrão são os namespaces diferentes do kube-system. Exemplo:
	`{ "exclude": ["kube-system"] }`
	– **thresholds**: limite para despejar pods de um nó. Quando o uso da CPU ou da memória de um nó é menor que o limite, os pods no nó serão despejados. Exemplo:
	`{ "cpu": 25, "memory": 25 }`

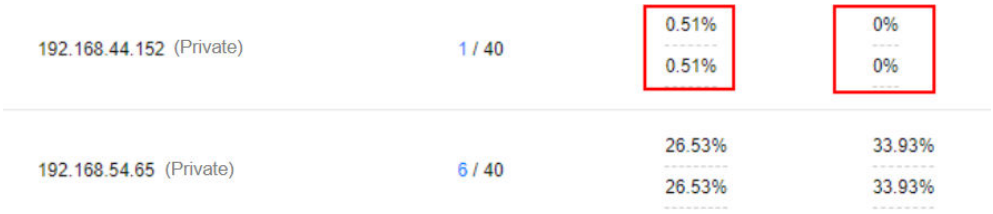
Passo 3 Clique em **OK**.

----Fim

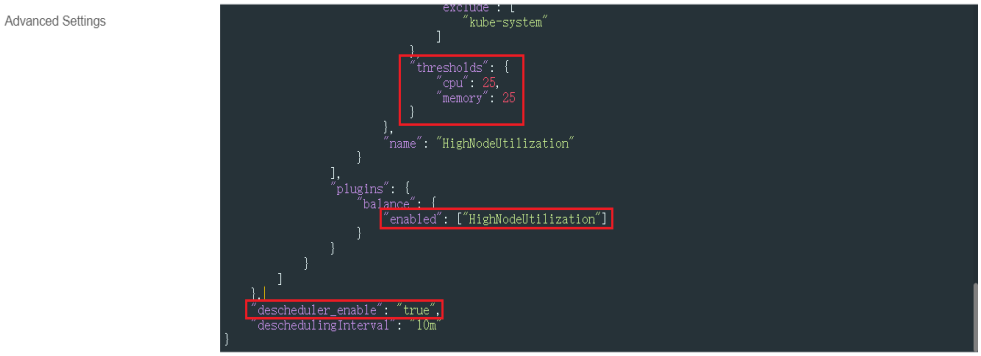
Casos de uso

HighNodeUtilization

1. Verifique os nós em um cluster. Verifica-se que alguns nós são subutilizados.



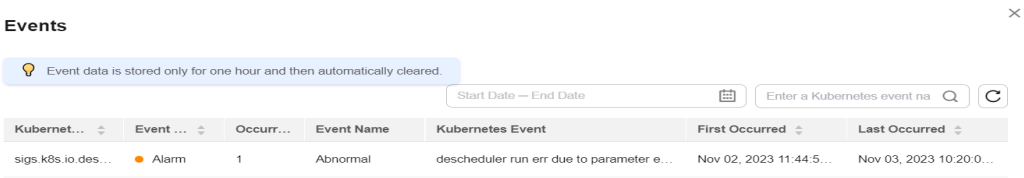
2. Edite os parâmetros do vulcão para ativar o reagendador e defina os limites de uso de CPU e memória para **25**. Quando o uso da CPU e da memória de um nó for menor que 25%, os pods no nó serão despejados.



3. Depois que a política entrar em vigor, os pods no nó com endereço IP 192.168.44.152 serão migrados para o nó com endereço IP 192.168.54.65 para minimizar fragmentos de recursos.

Problemas comuns

Se um parâmetro de entrada estiver incorreto, por exemplo, o valor inserido além do intervalo de valores aceitos ou estiver em um formato incorreto, um evento será gerado. Nesse caso, modifique a configuração do parâmetro conforme solicitado.



6.5.3 Afinidade do pool de nós

Em cenários como substituição de pool de nós e atualização contínua de nós, um pool de recursos antigo precisa ser substituído por um novo. Para impedir que a substituição do pool de nós afete os serviços, ative a afinidade suave para agendar pods de serviço para o novo pool de nós. O agendamento de afinidade suave tenta agendar pods recém-criados ou pods reagendados para o novo pool de nós. Se os pods não puderem ser agendados para o novo pool de nós, por exemplo, devido a recursos insuficientes, os pods também poderão ser agendados para o pool de nós anterior. Como a substituição de um pool de nós não deve afetar os serviços, a configuração de afinidade de nó não é declarada nas cargas de trabalho de serviço. Use afinidade suave no agendamento de cluster para agendar pods para novos pools de nós quando uma substituição de pool for acionada.

O Volcano tem como objetivo programar soft pods de serviço para nós especificados quando a afinidade suave do nó não estiver configurada em cargas de trabalho de serviço.

Prioridade de agendamento

O agendamento de afinidade suave de um pool de nós é implementado com base em rótulos no pool de nós. Cada nó no pool de nós é pontuado para selecionar o ideal para o agendamento de pods.

A regra é agendar pods para nós com rótulos especificados o máximo possível.

A fórmula para marcar um nó é a seguinte:

Pontuação do nó = Weight x MaxNodeScore x haveLabel

Parâmetros:

- **Weight**: peso do complemento de afinidade suave no pool de nós.
- **MaxNodeScore**: pontuação máxima (100) de um nó.
- **haveLabel**: se os rótulos configurados no complemento estão disponíveis em um nó. Se sim, o valor é **1**. Se não, o valor é **0**.

Pré-requisitos

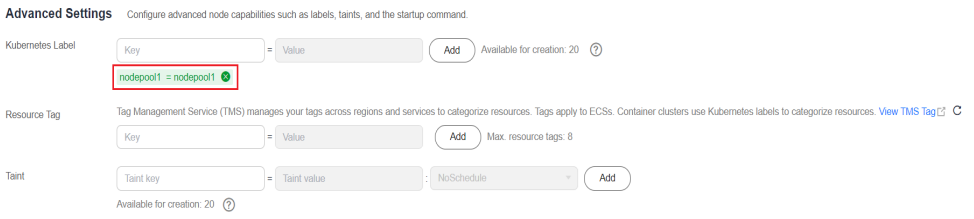
- Um cluster de v1.19.16 ou posterior está disponível. Para mais detalhes, consulte **Compra de um cluster do CCE**.
- Volcano da v1.11.5 ou posterior foi instalado no cluster. Para mais detalhes, consulte **Volcano scheduler**.

Habilitar o agendamento de afinidade suave para pools de nós de Volcano

Passo 1 Configure rótulos para agendamento de afinidade no pool de nós.

1. Efetue logon no console do CCE.
2. Clique no nome do cluster para acessar o console do cluster. Escolha **Nodes** no painel de navegação e clique na guia **Node Pools**.
3. Clique em **Update** do pool de nós de destino. Na página exibida, configure os rótulos na área **Kubernetes Label**.

   O seguinte é um exemplo.



Passo 2 Escolha **Add-ons** no painel de navegação, localize **Volcano Scheduler** à direita, clique em **Install** ou **Edit** e configure os parâmetros do Volcano scheduler na área **Parameters**.

```
{
    "ca_cert": "",
    "default_scheduler_conf": {
        "actions": "allocate, backfill",
        "tiers": [
            {
                "plugins": [
                    {
                        "name": "priority"
                    },
```



```
                    {
                        "name": "gang"
                    },
                    {
                        "name": "conformance"
                    }
                ]
            },
            {
                "plugins": [
                    {
                        "name": "drf"
                    },
                    {
                        "name": "predicates"
                    },
                    {
                        "name": "nodeorder"
                    }
                ]
            },
            {
                "plugins": [
                    {
                        "name": "cce-gpu-topology-predicate"
                    },
                    {
                        "name": "cce-gpu-topology-priority"
                    },
                    {
                        "name": "cce-gpu"
                    },
                    {
                        // Enable node pool affinity scheduling.
                        "name": "nodepoolaffinity",
                        // Configure the affinity scheduling weight and labels of
the node pool.
                        "arguments": {
                            "nodepoolaffinity.weight": 10000,
                            "nodepoolaffinity.label": "nodepool1=nodepool1"
                        }
                    }
                ]
            },
            {
                "plugins": [
                    {
                        "name": "nodelocalvolume"
                    },
                    {
                        "name": "nodeemptydirvolume"
                    },
                    {
                        "name": "nodeCSIscheduling"
                    },
                    {
                        "name": "networkresource"
                    }
                ]
            }
        ]
    },
    "server_cert": "",
    "server_key": ""
}
```

Passo 3 Clique em **OK**.

----Fim

6.5.4 Agendamento baseado em prioridade e preempção

Uma prioridade de pod indica a importância de um pod em relação a outros pods. Volcano suporta **PriorityClasses** de pod no Kubernetes. Depois que as PriorityClasses são configuradas, o agendador programa preferencialmente pods de alta prioridade. Quando os recursos de cluster são insuficientes, o agendador desalojará proativamente pods de baixa prioridade para possibilitar o agendamento de pods de alta prioridade pendentes.

Pré-requisitos

- Um cluster de v1.19 ou posterior está disponível. Para mais detalhes, consulte **Compra de um cluster do CCE**.
- O complemento Volcano foi instalado. Para mais detalhes, consulte **Volcano scheduler**.

Visão geral

Os serviços executados em um cluster são diversificados, incluindo serviços essenciais, serviços não essenciais, serviços on-line e serviços off-line. Você pode configurar prioridades para diferentes serviços com base na importância do serviço e nos requisitos de SLA. Por exemplo, configure uma alta prioridade para serviços principais e serviços online para que esses serviços obtenham preferencialmente recursos de cluster. Quando os recursos do cluster são ocupados por serviços não essenciais e os recursos restantes são insuficientes para novos serviços principais, o agendador antecipa ou despeja pods de baixa prioridade para liberar os recursos usados por serviços não essenciais para que os recursos possam ser usados para programar os pods dos serviços principais.

Tabela 6-5 lista o agendamento baseado em prioridade suportado pelos clusters do CCE.

Tabela 6-5 Agendamento baseado em prioridade

Tipo de agendamento	Descrição	Agendador
Agendamento baseado em prioridade	O agendador garante preferencialmente a execução de pods de alta prioridade, mas não expulsará pods de baixa prioridade que estão em execução. O agendamento baseado em prioridade é ativado por padrão e não pode ser desativado.	kube-scheduler ou Volcano scheduler
Preempção baseada em prioridade	Quando os recursos de cluster são insuficientes, o agendador desalojará proativamente pods de baixa prioridade para possibilitar o agendamento de pods de alta prioridade pendentes.	Volcano scheduler

Procedimento

Depois que o Volcano for instalado, você poderá ativar ou desativar o agendamento baseado em prioridade na página **Scheduling Configuration**.

Passo 1 Efetue logon no console do CCE.

Passo 2 Clique no nome do cluster para acessar o console do cluster. Escolha **Settings** no painel de navegação e clique na guia **Scheduling Configuration**.

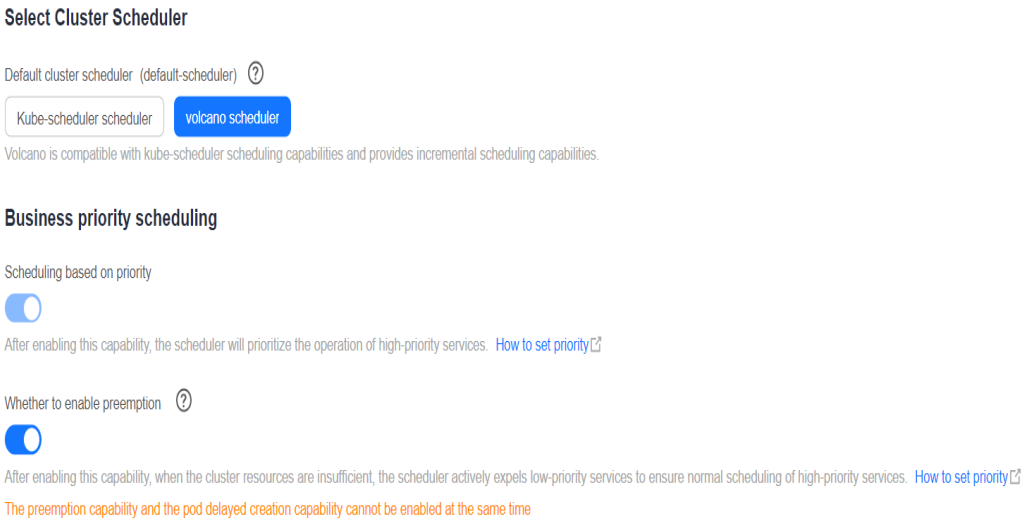
Passo 3 Na área **Business priority scheduling**, configure o agendamento baseado em prioridade.

- **Scheduling based on priority**: o agendador garante preferencialmente a execução de pods de alta prioridade, mas não despejará pods de baixa prioridade que estão em execução. O agendamento baseado em prioridade é ativado por padrão e não pode ser desativado.
- **Whether to enable preemption**: se o Volcano scheduler for usado como o agendador padrão do cluster, a preempção baseada em prioridade é suportada. Quando os recursos de cluster são insuficientes, o agendador despejará proativamente pods de baixa prioridade para possibilitar o agendamento de pods de alta prioridade pendentes.

  NOTA

  Depois que a preempção baseada em prioridade estiver ativada, a criação de pods atrasada não será permitida.

Figura 6-6 Agendamento baseado em prioridade



Passo 4 Clique em **Confirm**.

Passo 5 Após a configuração, você pode usar **PriorityClasses** para programar os pods de cargas de trabalho ou prioridades baseadas em tarefas do Volcano.

1. Crie uma ou mais **PriorityClasses**.

```
apiVersion: scheduling.k8s.io/v1
kind: PriorityClass
metadata:
  name: high-priority
value: 1000000
globalDefault: false
description: ""
```


2. Crie uma carga de trabalho ou uma tarefa de Volcanoe especifique seu nome de PriorityClass.
   - Carga de trabalho

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: high-test
  labels:
    app: high-test
spec:
  replicas: 5
  selector:
    matchLabels:
      app: test
  template:
    metadata:
      labels:
        app: test
    spec:
      priorityClassName: high-priority
      schedulerName: volcano
      containers:
      - name: test
        image: busybox
        imagePullPolicy: IfNotPresent
        command: ['sh', '-c', 'echo "Hello, Kubernetes!" && sleep 3600']
        resources:
          requests:
            cpu: 500m
          limits:
            cpu: 500m
```

   - Tarefa do Volcano

```
apiVersion: batch.volcano.sh/v1alpha1
kind: Job
metadata:
  name: vcjob
spec:
  schedulerName: volcano
  minAvailable: 4
  priorityClassName: high-priority
  tasks:
    - replicas: 4
      name: "test"
      template:
        spec:
          containers:
            - image: alpine
              command: ["/bin/sh", "-c", "sleep 1000"]
              imagePullPolicy: IfNotPresent
              name: running
              resources:
                requests:
                  cpu: "1"
          restartPolicy: OnFailure
```

----Fim

Casos de uso

Agendamento baseado em prioridade

Por exemplo, há dois nós ociosos e várias cargas de trabalho com três prioridades (alta prioridade, média prioridade e baixa prioridade). Execute a carga de trabalho de alta prioridade para esgotar todos os recursos de cluster e emita as cargas de trabalho de média e baixa prioridade. Em seguida, os dois tipos de cargas de trabalho estão pendentes devido a

recursos insuficientes. Quando a carga de trabalho de alta prioridade termina, os pods da carga de trabalho de prioridade média serão programados antes dos pods da carga de trabalho de baixa prioridade de acordo com a configuração de programação baseada em prioridade.

Passo 1 Adicione três **PriorityClasses** (**high-priority**, **med-priority** e **low-priority**) em **priority.yaml**.

Exemplo de configuração de **priority.yaml**:

```
apiVersion: scheduling.k8s.io/v1
kind: PriorityClass
metadata:
  name: high-priority
value: 100
globalDefault: false
description: "This priority class should be used for volcano job only."
---
apiVersion: scheduling.k8s.io/v1
kind: PriorityClass
metadata:
  name: med-priority
value: 50
globalDefault: false
description: "This priority class should be used for volcano job only."
---
apiVersion: scheduling.k8s.io/v1
kind: PriorityClass
metadata:
  name: low-priority
value: 10
globalDefault: false
description: "This priority class should be used for volcano job only."
```

Crie PriorityClasses.

```
kubectl apply -f priority.yaml
```

Passo 2 Verifique PriorityClasses.

```
kubectl get PriorityClass
```

Saída do comando:

```
NAME                      VALUE        GLOBAL-DEFAULT   AGE
high-priority             100          false            97s
low-priority              10           false            97s
med-priority              50           false            97s
system-cluster-critical   2000000000   false            6d6h
system-node-critical      2000001000   false            6d6h
```

Passo 3 Crie uma carga de trabalho de alta prioridade chamada **high-priority-job** para esgotar todos os recursos do cluster.

high-priority-job.yaml

```
apiVersion: batch.volcano.sh/v1alpha1
kind: Job
metadata:
  name: priority-high
spec:
  schedulerName: volcano
  minAvailable: 4
  priorityClassName: high-priority
  tasks:
    - replicas: 4
      name: "test"
      template:
        spec:
          containers:
            - image: alpine
              command: ["/bin/sh", "-c", "sleep 1000"]
```



```
              imagePullPolicy: IfNotPresent
              name: running
              resources:
                requests:
                  cpu: "1"
          restartPolicy: OnFailure
```

Execute o seguinte comando para emitir a tarefa:

```
kubectl apply -f high_priority_job.yaml
```

Execute o comando **kubectl get pod** para verificar os status do pod:

```
NAME                   READY   STATUS    RESTARTS   AGE
priority-high-test-0   1/1     Running   0          3s
priority-high-test-1   1/1     Running   0          3s
priority-high-test-2   1/1     Running   0          3s
priority-high-test-3   1/1     Running   0          3s
```

A saída do comando mostra que todos os recursos do cluster foram usados.

Passo 4 Crie uma carga de trabalho de média prioridade **med-priority-job** e uma carga de trabalho de baixa prioridade baixa **low-priority-job**.

med-priority-job.yaml

```
apiVersion: batch.volcano.sh/v1alpha1
kind: Job
metadata:
  name: priority-medium
spec:
  schedulerName: volcano
  minAvailable: 4
  priorityClassName: med-priority
  tasks:
    - replicas: 4
      name: "test"
      template:
        spec:
          containers:
            - image: alpine
              command: ["/bin/sh", "-c", "sleep 1000"]
              imagePullPolicy: IfNotPresent
              name: running
              resources:
                requests:
                  cpu: "1"
          restartPolicy: OnFailure
```

low-priority-job.yaml

```
apiVersion: batch.volcano.sh/v1alpha1
kind: Job
metadata:
  name: priority-low
spec:
  schedulerName: volcano
  minAvailable: 4
  priorityClassName: low-priority
  tasks:
    - replicas: 4
      name: "test"
      template:
        spec:
          containers:
            - image: alpine
              command: ["/bin/sh", "-c", "sleep 1000"]
              imagePullPolicy: IfNotPresent
              name: running
```



```
              resources:
                requests:
                  cpu: "1"
          restartPolicy: OnFailure
```

Execute os seguintes comandos para emitir as tarefas:

```
kubectl apply -f med_priority_job.yaml
kubectl apply -f low_priority_job.yaml
```

Execute o comando **kubectl get pod** para verificar os status dos pods das cargas de trabalho recém-criadas. A saída do comando mostra que os pods estão pendentes devido a recursos insuficientes:

```
NAME                      READY   STATUS    RESTARTS   AGE
priority-high-test-0      1/1     Running   0          3m29s
priority-high-test-1      1/1     Running   0          3m29s
priority-high-test-2      1/1     Running   0          3m29s
priority-high-test-3      1/1     Running   0          3m29s
priority-low-test-0       0/1     Pending   0          2m26s
priority-low-test-1       0/1     Pending   0          2m26s
priority-low-test-2       0/1     Pending   0          2m26s
priority-low-test-3       0/1     Pending   0          2m26s
priority-medium-test-0    0/1     Pending   0          2m36s
priority-medium-test-1    0/1     Pending   0          2m36s
priority-medium-test-2    0/1     Pending   0          2m36s
priority-medium-test-3    0/1     Pending   0          2m36s
```

Passo 5 Exclua a carga de trabalho **high_priority_job** para liberar recursos e verifique se os pods da carga de trabalho **med-priority-job** serão preferencialmente agendados.

Execute o comando **kubectl delete -f high_priority_job.yaml** para liberar recursos de cluster e verificar o agendamento de pods.

```
NAME                      READY   STATUS    RESTARTS   AGE
priority-low-test-0       0/1     Pending   0          5m18s
priority-low-test-1       0/1     Pending   0          5m18s
priority-low-test-2       0/1     Pending   0          5m18s
priority-low-test-3       0/1     Pending   0          5m18s
priority-medium-test-0    1/1     Running   0          5m28s
priority-medium-test-1    1/1     Running   0          5m28s
priority-medium-test-2    1/1     Running   0          5m28s
priority-medium-test-3    1/1     Running   0          5m28s
```

----Fim

Preempção baseada em prioridade

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster. Escolha **Settings** no painel de navegação e clique na guia **Scheduling Configuration**.

Passo 2 Modifique configurações.

1. Selecione **volcano scheduler** como o agendador de cluster padrão.
2. Ative **Scheduling based on priority**.

Passo 3 Emita a carga de trabalho **high_priority_job**. Em seguida, o agendador expulsará os pods da carga de trabalho **med_priority_job** para que os pods da carga de trabalho de alta prioridade possam ser agendados.

Execute o comando **kubectl apply -f high_priority_job.yaml** para emitir a carga de trabalho de alta prioridade. Em seguida, verifique os status dos pods.

```
NAME                      READY   STATUS        RESTARTS   AGE
priority-high-test-0      0/1     Pending       0          2s
priority-high-test-1      0/1     Pending       0          2s
```



```
priority-high-test-2     0/1     Pending       0          2s
priority-high-test-3     0/1     Pending       0          2s
priority-low-test-0      0/1     Pending       0          14s
priority-low-test-1      0/1     Pending       0          14s
priority-low-test-2      0/1     Pending       0          14s
priority-low-test-3      0/1     Pending       0          14s
priority-medium-test-0   1/1     Terminating   0          21s
priority-medium-test-1   1/1     Terminating   0          21s
priority-medium-test-2   1/1     Terminating   0          21s
priority-medium-test-3   1/1     Terminating   0          21s
```

Depois que os recursos usados pela carga de trabalho do recurso **med_priority_job resource** forem liberados, os pods da carga de trabalho **high_priority_job** poderão ser agendados.

```
NAME                     READY   STATUS    RESTARTS   AGE
priority-high-test-0     1/1     Running   0          70s
priority-high-test-1     1/1     Running   0          70s
priority-high-test-2     1/1     Running   0          70s
priority-high-test-3     1/1     Running   0          70s
priority-low-test-0      0/1     Pending   0          82s
priority-low-test-1      0/1     Pending   0          82s
priority-low-test-2      0/1     Pending   0          82s
priority-low-test-3      0/1     Pending   0          82s
priority-medium-test-0   0/1     Pending   0          37s
priority-medium-test-1   0/1     Pending   0          36s
priority-medium-test-2   0/1     Pending   0          37s
priority-medium-test-3   0/1     Pending   0          37s
```

----Fim

6.5.5 DRF

Dominant Resource Fairness (DRF) é um algoritmo de agendamento baseado no recurso dominante de um grupo de contêineres. O agendamento DRF pode ser usado para aprimorar a taxa de transferência de serviço de um cluster, reduzir o tempo geral de execução do serviço e melhorar o desempenho da execução do serviço. É adequado para treinamento em lote de IA e trabalhos de Big Data.

Pré-requisitos

- Um cluster de v1.19 ou posterior está disponível. Para mais detalhes, consulte **Compra de um cluster do CCE**.
- O complemento Volcano foi instalado. Para mais detalhes, consulte **Volcano scheduler**.

Contexto

Em serviços reais, recursos de cluster limitados são frequentemente alocados a vários usuários. Cada usuário tem os mesmos direitos para obter recursos, mas o número de recursos necessários pode ser diferente. É fundamental alocar recursos de forma justa para cada usuário. Um algoritmo de agendamento comum é o max-min fairness share, que aloca recursos para atender aos requisitos mínimos dos usuários, tanto quanto possível e, em seguida, aloca de forma justa os recursos restantes. As regras são as seguintes:

1. Os recursos são alocados em ordem de demanda crescente.
2. Nenhuma fonte obtém um compartilhamento de recursos maior do que sua demanda.
3. Fontes com demandas insatisfeitas obtêm uma parcela igual do recurso.

O algoritmo max-min fairness aplica-se ao cenário de recurso único, no qual todos os trabalhos estão solicitando os mesmos recursos. No entanto, em situações reais, vários recursos estão envolvidos. Por exemplo, recursos de CPU, memória e GPU são solicitados

para alocação. O DRF pode ser usado para resolver o problema anterior. O DRF pode ser considerado como uma versão geral do algoritmo max-min fairness e suporta alocação justa de vários tipos de recursos para que o recurso dominante de cada usuário atenda ao requisito de max-min fairness.

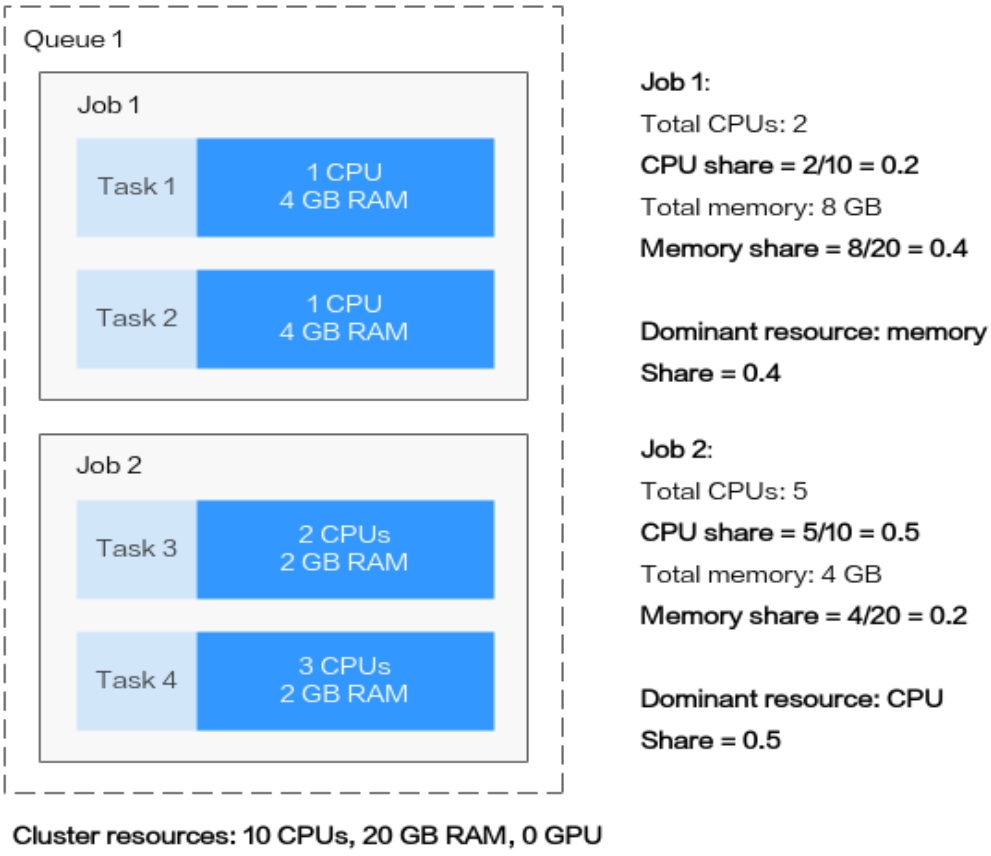
O valor de compartilhamento de cada recurso de tarefa é calculado usando a seguinte fórmula:

Compartilhamento = total de recursos solicitados/recursos do cluster

Se uma tarefa envolve vários recursos, o recurso com o maior valor de compartilhamento é o recurso dominante. O valor de compartilhamento do recurso dominante será usado no agendamento baseado em prioridade.

Por exemplo, existem duas cargas de trabalho, tarefa 1 e tarefa 2. A figura a seguir mostra os recursos solicitados pelas duas tarefas. Após o cálculo do DRF, o recurso dominante da tarefa 1 é a memória e seu valor de compartilhamento é 0,4; o recurso dominante da tarefa 2 é a CPU e seu valor de compartilhamento é 0,5. Como a parcela de recurso dominante da tarefa 1 é menor que a da tarefa 2, a tarefa 1 tem precedência sobre a tarefa 2 no agendamento de acordo com a política de max-min fairness.

Figura 6-7 Agendamento DRF



Procedimento

Após a instalação do Volcano, você pode ativar ou desativar o agendamento DRF na página **Scheduling Configuration**. Essa função está ativada por padrão.

Passo 1 Efetue logon no console do CCE.

Passo 2 Clique no nome do cluster para acessar o console do cluster. Escolha **Settings** no painel de navegação e clique na guia **Scheduling Configuration**.

Passo 3 No painel **AI task performance enhanced scheduling**, selecione se deseja ativar o DRF.

Essa função ajuda a aprimorar a taxa de transferência de serviço do cluster e melhorar o desempenho da execução do serviço.

Passo 4 Clique em **Confirm**.

----Fim

6.5.6 Gang

O agendamento Gang é um algoritmo de agendamento que agenda processos ou threads correlacionados para serem executados simultaneamente em diferentes processadores. Ele atende aos requisitos de agendamento de "Tudo ou nada" no processo de agendamento e evita o desperdício de recursos de cluster causados pelo agendamento arbitrário de pods. Gang é usado principalmente em cenários que exigem colaboração de vários processos, como cenários de IA e Big Data. O agendamento Gang resolve efetivamente pontos problemáticos, como impasses em trabalhos de treinamento distribuídos, melhorando significativamente a utilização de recursos de cluster.

Pré-requisitos

- Um cluster de v1.19 ou posterior está disponível. Para mais detalhes, consulte **Compra de um cluster do CCE**.
- O complemento Volcano foi instalado. Para mais detalhes, consulte **Volcano scheduler**.

Recursos

A política de agendamento Gang é um dos principais algoritmos de agendamento do volcano-scheduler. Ele atende aos requisitos de agendamento de "Tudo ou nada" no processo de agendamento e evita o desperdício de recursos de cluster causados pelo agendamento arbitrário de pods. O algoritmo Gang scheduler verifica se o número de pods agendados em uma tarefa atende aos requisitos mínimos para a execução da tarefa. Se sim, todos os pods na tarefa serão agendados. Se não, os pods não serão agendados.

O algoritmo de agendamento Gang baseado em grupos de contêineres é adequado para cenários em que a colaboração multiprocesso é necessária. Os cenários de IA geralmente envolvem processos complexos. A ingestão de dados, os analistas de dados, a divisão de dados, os treinadores, o serviço e o registro que exigem que um grupo de contêineres trabalhem juntos são adequados para o agendamento de gangues baseado em contêiner. Cenários de comunicação de computação paralela multi-thread sob a estrutura de computação MPI também são adequados para o agendamento Gang porque os processos principal e secundário precisam trabalhar juntos. Os contêineres em um grupo de pods são altamente correlacionados e pode haver contenção de recursos. A alocação geral de agendamento pode efetivamente resolver impasses. Se os recursos de cluster forem insuficientes, o agendamento de grupos pode melhorar significativamente a utilização dos recursos de cluster.

Procedimento

Depois que o Volcano for instalado, você poderá ativar ou desativar o agendamento Gang na página **Scheduling Configuration**. Essa função está ativada por padrão.

Passo 1 Efetue logon no console do CCE.

Passo 2 Clique no nome do cluster para acessar o console do cluster. Escolha **Settings** no painel de navegação e clique na guia **Scheduling Configuration**.

Passo 3 No painel **AI task performance enhanced scheduling**, selecione se deseja ativar o Gang.

Essa função ajuda a aprimorar a taxa de transferência de serviço do cluster e melhorar o desempenho da execução do serviço.

Passo 4 Clique em **Confirm**.

Passo 5 Após a configuração, use o agendamento Gang em cargas de trabalho ou tarefas de Volcano.

- Crie uma carga de trabalho usando o agendamento Gang.
  a. Crie um grupo de pods e especifique **minMember** e **minResources** da seguinte forma:

```
apiVersion: scheduling.volcano.sh/v1beta1
kind: PodGroup
metadata:
  name: pg-test1
spec:
  minMember: 3
  minResources:
    cpu: 3
    memory: 3Gi
```

  - **minMember**: especifica o requisito mínimo sobre o número de pods para executar uma carga de trabalho. Quando o número de pods no grupo de pods atual atende ao requisito, esses pods podem ser agendados centralmente.
  - **minResources**: especifica o requisito mínimo de recursos para executar uma carga de trabalho. Quando os recursos disponíveis em um cluster atendem ao requisito, o grupo de pods pode ser agendado centralmente.

  b. Ao criar uma carga de trabalho, use **schedulerName** para especificar o Volcano scheduler e **annotation** para especificar o grupo de pods no qual o Volcano scheduler é executado.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: podgroup-test
  labels:
    app: podgroup-test
spec:
  replicas: 6
  selector:
    matchLabels:
      app: podgroup-test
  template:
    metadata:
      annotations:
        scheduling.k8s.io/group-name: pg-test1
      labels:
        app: podgroup-test
    spec:
      schedulerName: volcano
      containers:
      - name: test
        image: busybox
        imagePullPolicy: IfNotPresent
        command: ['sh', '-c', 'echo "Hello, Kubernetes!" && sleep 3600']
        resources:
          requests:
            cpu: 500m
          limits:
            cpu: 500m
```


  - **schedulerName**: defina este parâmetro para **volcano**, indicando que o vulcão será usado para programar pods para a carga de trabalho.
  - **scheduling.k8s.io/group-name**: especifica o grupo de pods criado na etapa anterior, por exemplo, **pg-test1**.

- Crie uma tarefa de Volcano usando o agendamento Gang.

  Ao criar uma tarefa de Volcano, você só precisa configurar **minAvailable** e definir o **schedulerName** para **volcano**. O Volcano scheduler criará automaticamente um grupo de pods e o gerenciará. O seguinte mostra um exemplo:

```
apiVersion: batch.volcano.sh/v1alpha1
kind: Job
metadata:
  name: vcjob
spec:
  schedulerName: volcano
  minAvailable: 2
  tasks:
    - replicas: 4
      name: "test"
      template:
        spec:
          containers:
            - image: alpine
              command: ["/bin/sh", "-c", "sleep 1000"]
              imagePullPolicy: IfNotPresent
              name: running
              resources:
                requests:
                  cpu: "1"
          restartPolicy: OnFailure
```

----Fim

6.5.7 Agendamento de afinidade NUMA

Conhecimento de fundo

Quando o nó executa muitos pods vinculados à CPU, a carga de trabalho pode mover-se para diferentes núcleos de CPU, dependendo se o pod é acelerado e quais núcleos de CPU estão disponíveis no momento do agendamento. Muitas cargas de trabalho não são sensíveis a essa migração e, portanto, funcionam bem sem qualquer intervenção. No entanto, em cargas de trabalho nas quais a afinidade de cache da CPU e a latência de agendamento afetam significativamente o desempenho da carga de trabalho, o kubelet permite que políticas alternativas de gerenciamento da CPU determinem algumas preferências de posicionamento no nó.

Ambos CPU Manager e Topology Manager são componentes do kubelet, mas têm as seguintes limitações:

- O agendador não está ciente da topologia. Portanto, a carga de trabalho pode ser agendada em um nó e, em seguida, falhar no nó devido ao Topology Manager. Isso é inaceitável para tarefas TensorFlow. Se algum trabalhador ou ps falhou no nó, a tarefa falhará.
- Os gerentes são nível de nó que resulta em uma incapacidade de combinar o melhor nó para a topologia NUMA em todo o cluster.

Para obter mais informações, consulte **https://github.com/volcano-sh/volcano/blob/master/docs/design/numa-aware.md**.

Metas do Volcano para resolver a limitação para fazer a topologia NUMA de agendador consciente, de modo a alcançar o seguinte:

- Não agende pods para os nós que a topologia NUMA não combina.
- Agende pods para o melhor nó para topologia NUMA.

Escopo da aplicação

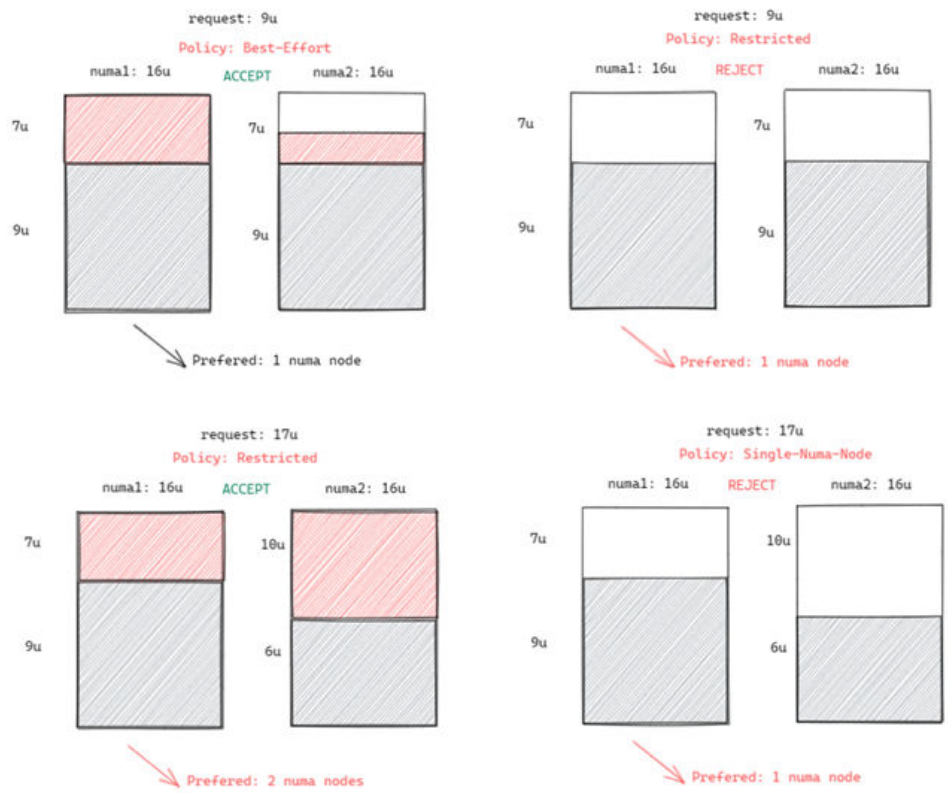
- Suporte ao agendamento de topologia de recursos da CPU
- Suporte às políticas de topologia em nível de pod

Previsão de agendamento

Para pods com a política de topologia, predique a lista de nós correspondentes.

Política	Ação
none	1. Nenhuma ação de filtro
best-effort	1. Filtre o nó com a política de topologia **best-effort**.
restricted	1. Filtre o nó com a política de topologia **restricted**. 2. Filtre o nó que a topologia de CPU cumpre os requisitos de CPU para **restricted**.
single-numa-node	1. Filtre o nó com a política de topologia **single-numa-node**. 2. Filtre o nó que a topologia da CPU atende aos requisitos de CPU para **single-numa-node**.

Figura 6-8 Comparação das políticas de agendamento de NUMA



Prioridade de agendamento

A política de topologia visa agendar pods para o nó ideal. Neste exemplo, cada nó é pontuado para classificar o nó ideal.

Princípio: agende pods para os nós de trabalho que exigem o menor número de nós NUMA.

A fórmula de pontuação é a seguinte:

score = weight * (100 - 100 * numaNodeNum / maxNumaNodeNum)

Descrição do parâmetro:

- **weight**: indica o peso de NUMA Aware Plugin.
- **numaNodeNum**: indica o número de nós NUMA necessários para executar o pod no nó de trabalho.
- **maxNumaNodeNum**: indica o número máximo de nós NUMA em um pod de todos os nós de trabalho.

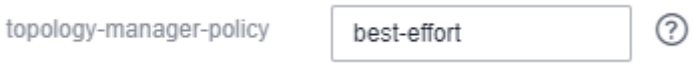
Habilitar o Volcano para suportar agendamento de afinidade NUMA

Passo 1 Habilite a política de gerenciamento da CPU. Para mais detalhes, consulte **Ativar a política de gerenciamento da CPU**.

Passo 2 Configure uma política de topologia de CPU.

1. Faça logon no console do CCE, clique no nome do cluster, acesse a página de detalhes do cluster e escolha **Nodes** no painel de navegação. Na página exibida, clique na guia **Node Pools**. Escolha **More** > **Manage** na coluna **Operation** do pool de nós de destino.
2. Altere o valor de **topology-manager-policy** em **kubelet** para a política de topologia de CPU necessária. Conforme mostrado na figura a seguir, a política de topologia da CPU é **best-effort**.

   As políticas de topologia válidas são **none**, **best-effort**, **restricted** e **single-numa-node**. Para obter detalhes sobre essas políticas, consulte **Previsão de agendamento**.



Passo 3 Habilite o complemento numa-aware e a função **resource_exporter**.

volcano 1.7.1 ou mais tarde

1. Efetue logon no console do CCE e acesse o console do cluster. No painel de navegação, escolha **Add-ons**. À direita da página, localize o complemento do **volcano** e clique em **Edit**. Na área **Parameters**, configure os parâmetros do agendador de Volcano.

```
{
    "ca_cert": "",
    "default_scheduler_conf": {
        "actions": "allocate, backfill",
        "tiers": [
            {
                "plugins": [
                    {
                        "name": "priority"
                    },
                    {
                        "name": "gang"
                    },
                    {
                        "name": "conformance"
                    }
                ]
            },
            {
                "plugins": [
                    {
                        "name": "drf"
                    },
                    {
                        "name": "predicates"
                    },
                    {
                        "name": "nodeorder"
                    }
                ]
            },
            {
                "plugins": [
                    {
                        "name": "cce-gpu-topology-predicate"
                    },
                    {
                        "name": "cce-gpu-topology-priority"
                    },
                    {
                        "name": "cce-gpu"
                    },
                    {
                        // add this also enable resource_exporter
                        "name": "numa-aware",
                        // the weight of the NUMA Aware Plugin
```



```
                        "arguments": {
                           "weight": "10"
                        }
                    }
                ]
            },
            {
                "plugins": [
                    {
                        "name": "nodelocalvolume"
                    },
                    {
                        "name": "nodeemptydirvolume"
                    },
                    {
                        "name": "nodeCSIscheduling"
                    },
                    {
                        "name": "networkresource"
                    }
                ]
            }
        ]
    },
    "server_cert": "",
    "server_key": ""
}
```

volcano anterior a 1.7.1

1. O parâmetro **resource_exporter_enable** está ativado para o complemento de volcano coletar informações de nó NUMA.

   ```
   {
      "plugins": {
         "eas_service": {
            "availability_zone_id": "",
            "driver_id": "",
            "enable": "false",
            "endpoint": "",
            "flavor_id": "",
            "network_type": "",
            "network_virtual_subnet_id": "",
            "pool_id": "",
            "project_id": "",
            "secret_name": "eas-service-secret"
         }
      },
      "resource_exporter_enable": "true"
   }
   ```

   Depois que esta função é permitida, você pode ver a informação da topologia NUMA do nó atual.

   ```
   kubectl get numatopo
   NAME              AGE
   node-1            4h8m
   node-2            4h8m
   node-3            4h8m
   ```

2. Habilite o complemento do algoritmo numa-aware do volcano.

   kubectl edit cm -n kube-system volcano-scheduler-configmap

   ```
   kind: ConfigMap
   apiVersion: v1
   metadata:
     name: volcano-scheduler-configmap
     namespace: kube-system
   data:
     default-scheduler.conf: |-
       actions: "allocate, backfill"
       tiers:
   ```



```
    - plugins:
      - name: priority
      - name: gang
      - name: conformance
    - plugins:
      - name: overcommit
      - name: drf
      - name: predicates
      - name: nodeorder
    - plugins:
      - name: cce-gpu-topology-predicate
      - name: cce-gpu-topology-priority
      - name: cce-gpu
    - plugins:
      - name: nodelocalvolume
      - name: nodeemptydirvolume
      - name: nodeCSIscheduling
      - name: networkresource
        arguments:
          NetworkType: vpc-router
      - name: numa-aware # add it to enable numa-aware plugin
        arguments:
          weight: 10 # the weight of the NUMA Aware Plugin
```

----Fim

Usar o Volcano para dar suporte ao agendamento de afinidade NUMA

Passo 1 Configure afinidade NUMA para Implementações. O seguinte é um exemplo:

```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: numa-tset
spec:
  replicas: 1
  selector:
    matchLabels:
      app: numa-tset
  template:
    metadata:
      labels:
        app: numa-tset
      annotations:
        volcano.sh/numa-topology-policy: single-numa-node    # set the topology
policy
    spec:
      containers:
        - name: container-1
          image: nginx:alpine
          resources:
            requests:
              cpu: 2           # The value must be an integer and must be the
same as that in limits.
              memory: 2048Mi
            limits:
              cpu: 2           # The value must be an integer and must be the
same as that in requests.
              memory: 2048Mi
      imagePullSecrets:
      - name: default-secret
```

Passo 2 Crie uma tarefa de volcano e use a afinidade NUMA.

```
apiVersion: batch.volcano.sh/v1alpha1
kind: Job
metadata:
  name: vj-test
spec:
  schedulerName: volcano
```



```
  minAvailable: 1
  tasks:
    - replicas: 1
      name: "test"
      topologyPolicy: best-effort   # set the topology policy for task
      template:
        spec:
          containers:
            - image: alpine
              command: ["/bin/sh", "-c", "sleep 1000"]
              imagePullPolicy: IfNotPresent
              name: running
              resources:
                limits:
                  cpu: 20
                  memory: "100Mi"
          restartPolicy: OnFailure
```

Passo 3 Verifique o uso de NUMA.

```
# Check the CPU usage of the current node.
lscpu
...
CPU(s):              32
NUMA node(s):        2
NUMA node0 CPU(s):   0-15
NUMA node1 CPU(s):   16-31

# Check the CPU allocation of the current node.
cat /var/lib/kubelet/cpu_manager_state
{"policyName":"static","defaultCpuSet":"0,10-15,25-31","entries":{"777870b5-
c64f-42f5-9296-688b9dc212ba":{"container-1":"16-24"},"fb15e10a-
b6a5-4aaa-8fcd-76c1aa64e6fd":{"container-1":"1-9"}},"checksum":318470969}
```

----Fim

6.6 Implementação híbrida da nuvem nativa

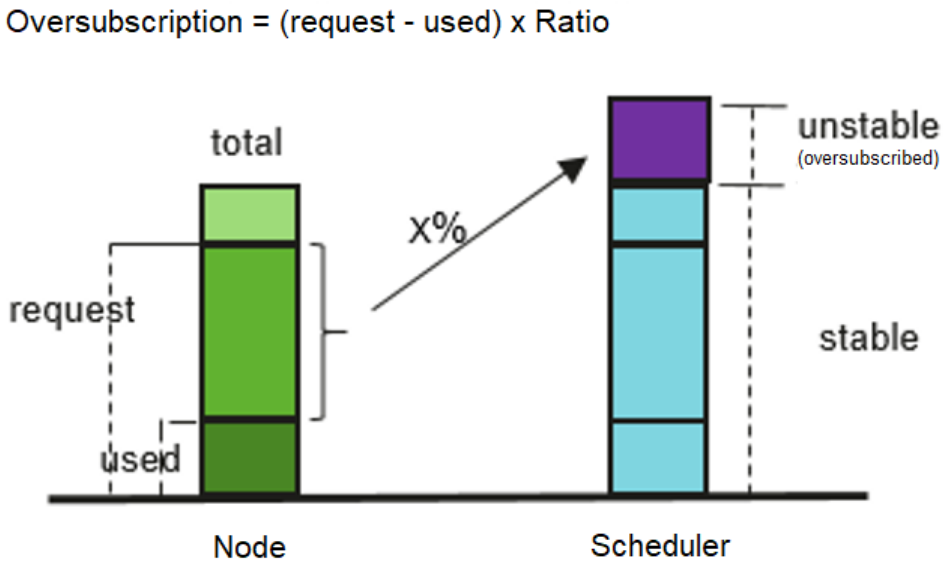
6.6.1 Excesso de assinaturas de recursos dinâmicos

Muitos serviços apresentam picos de tráfego. Para garantir o desempenho e a estabilidade, os recursos são frequentemente solicitados no máximo necessário. No entanto, os surtos podem diminuir muito em breve e os recursos, se não forem liberados, são desperdiçados em horas fora do horário de pico. Especialmente para trabalhos on-line que solicitam uma grande quantidade de recursos para garantir SLA, a utilização de recursos pode ser tão baixa quanto possível.

Excesso de assinaturas de recursos é o processo de fazer uso de recursos solicitados ociosos. Os recursos com excesso de assinatura são adequados para a implementação de tarefas off-line, que se concentram na taxa de transferência, mas têm baixos requisitos de SLA e podem tolerar certas falhas.

A implementação híbrida de tarefas on-line e off-line em um cluster pode utilizar melhor os recursos do cluster.

Figura 6-9 Excesso de assinaturas de recursos



Recursos

NOTA

Depois que o excesso de assinaturas de recursos dinâmicos e o dimensionamento elástico são habilitados em um pool de nós, os recursos com excesso de assinaturas mudam rapidamente porque o uso de recursos de aplicações de alta prioridade muda em tempo real. Para evitar reduções e expansões frequentes do nó, não considere recursos com excesso de assinatura ao avaliar as reduções do nó.

A implementação híbrida é suportada e os recursos da CPU e da memória podem ser excedidos. As principais características são as seguintes:

- Tarefas off-line são executadas preferencialmente em nós com excesso de assinatura.

  Se ambos os nós com excesso de assinatura e sem excesso de assinatura existirem, o primeiro terá uma pontuação maior do que o último e as tarefas off-line são preferencialmente programadas para nós com excesso de assinatura.
- As tarefas on-line podem usar somente recursos não de excesso de assinatura se programadas para um nó com assinatura em excesso.

  As tarefas off-line podem usar recursos com e sem assinatura em excesso de um nó com assinatura em excesso.
- No mesmo período de agendamento, as tarefas on-line têm precedência sobre as tarefas off-line.

  Se houver tarefas on-line e off-line, as tarefas on-line serão agendadas primeiro. Quando o uso de recursos do nó exceder o limite superior e as solicitações do nó excederem 100%, as tarefas off-line serão despejadas.
- O isolamento da CPU/memória é fornecido pelos kernels.

  Isolamento da CPU: as tarefas on-line podem rapidamente antecipar os recursos da CPU de tarefas off-line e suprimir o uso da CPU das tarefas off-line.

  Isolamento da memória: quando os recursos de memória do sistema são usados e o OOM Kill é acionado, o kernel despeja as tarefas off-line primeiro.

- Regras de admissão das tarefas off-line do kubelet:

  Depois que o pod for programado para um nó, o kubelet iniciará o pod somente quando os recursos do nó puderem atender à solicitação do pod (predicateAdmitHandler.Admit). O kubelet iniciará o pod quando ambas as condições a seguir forem atendidas:

  – A solicitação total de pods a serem iniciados e tarefas em execução on-line < nós alocáveis

  – A solicitação total de pods a serem iniciados e tarefas on-line/off-line em execução < nós alocáveis + nós com excesso de assinatura

- Excesso de assinatura de recursos e implementação híbrida:

  Se apenas a implementação híbrida for usada, configure o rótulo **volcano.sh/colocation=true** para o nó e exclua o rótulo do nó **volcano.sh/oversubscription** ou defina seu valor como **false**.

  Se o rótulo **volcano.sh/colocation=true** estiver configurado para um nó, a implementação híbrida será ativada. Se o rótulo **volcano.sh/oversubscription=true** estiver configurado, o excesso de assinatura de recursos será ativado. A tabela a seguir lista as combinações de recursos disponíveis após a implementação híbrida ou excesso de assinatura de recursos estar ativado.

Implementação híbrida ativada (volcano.sh/colocation=true)	Excesso de assinatura de recurso ativado (volcano.sh/oversubscription=true)	Usar recursos de excesso de assinatura	Condições para despejar pods off-line
Não	Não	Não	Nenhuma
Sim	Não	Não	O uso de recursos do nó excede o limite alto.
Não	Sim	Sim	O uso de recursos do nó excede o limite alto e a solicitação do nó excede 100%.
Sim	Sim	Sim	O uso de recursos do nó excede o limite alto.

Excesso de assinatura de kubelet

AVISO

Especificações

- Versão do cluster
  - v1.19: v1.19.16-r4 ou mais recente
  - v1.21: v1.21.7-r0 ou mais recente
  - v1.23: v1.23.5-r0 ou mais recente
  - v1.25 ou mais recente
- Tipo do cluster: CCE ou CCE Turbo
- Sistema operacional do nó: EulerOS 2.9 (kernel-4.18.0-147.5.1.6.h729.6.eulerosv2r9.x86_64) ou Huawei Cloud EulerOS 2.0
- Tipo do nó: ECS
- A versão do complemento volcano: 1.7.0 ou mais recente

Restrições

- Antes de ativar excesso de assinaturas, certifique-se de que o complemento overcommit não está ativado no volcano.
- Modificar o rótulo de um nó com excesso de assinatura não afeta os pods em execução.
- Os pods em execução não podem ser convertidos entre serviços on-line e off-line. Para converter serviços, você precisa reconstruir pods.
- Se o rótulo **volcano.sh/oversubscription=true** estiver configurado para um nó no cluster, a configuração de **oversubscription** deverá ser adicionada ao complemento volcano. Caso contrário, o agendamento de nós com excesso de demanda será anormal. Assegure-se de que você tenha configurado corretamente as etiquetas porque o agendador não verifica o complemento e as configurações do nó. Para obter detalhes sobre os rótulos, consulte **Tabela 6-6**.
- Para desativar o excesso de assinatura, execute as seguintes operações:
  - Remova o rótulo **volcano.sh/oversubscription** do nó com excesso de assinatura.
  - Defina **over-subscription-resource** como **false**.
  - Modifique o configmap do volcano scheduler chamado **volcano-scheduler-configmap** e remova o complemento de excesso de assinatura.
- Se **cpu-manager-policy** estiver definida para a vinculação de núcleo estática em um nó, não atribua a classe de QoS Guaranteed aos pods off-line. Se a vinculação do núcleo for necessária, altere os pods para pods on-line. Caso contrário, os pods off-line podem ocupar as CPUs dos pods on-line, causando falhas de inicialização dos pods on-line e os pods off-line não podem ser iniciados, embora tenham sido programados com êxito.
- Se **cpu-manager-policy** estiver definida como vinculação de núcleo estática em um nó, não vincule núcleos a todos os pods on-line. Caso contrário, os pods on-line ocupam todos os recursos de CPU ou memória, deixando um pequeno número de recursos com excesso de assinatura.

Se o rótulo **volcano.sh/oversubscription=true** estiver configurado para um nó no cluster, a configuração de **oversubscription** deverá ser adicionada ao complemento volcano. Caso contrário, o agendamento de nós com excesso de demanda será anormal. Para obter detalhes sobre a configuração relacionada, consulte **Tabela 6-6**.

Assegure-se de que você tenha configurado corretamente as etiquetas porque o agendador não verifica o complemento e as configurações do nó.

Tabela 6-6 Configurar rótulos de excesso de assinatura para agendamento

Excesso de assinatura no complemento	Rótulo de excesso de assinatura no nó	Agendamento
Sim	Sim	Acionado por excesso de assinatura
Sim	Não	Acionado
Não	Não	Acionado
Não	Sim	Não foi acionado ou falhou. Evite essa configuração.

Passo 1 Configure o complemento volcano.

1. Use o kubectl para se conectar ao cluster.
2. Instale o complemento volcano e adicione o complemento de excesso de assinatura para **volcano-scheduler-configmap**. Certifique-se de que a configuração do complemento não contém o complemento overcommit. Se **- name: overcommit** existir, exclua esta configuração. Além disso, defina **enablePreemptable** e **enableJobStarving** do complementogang como **false** e configure uma ação de preempção.

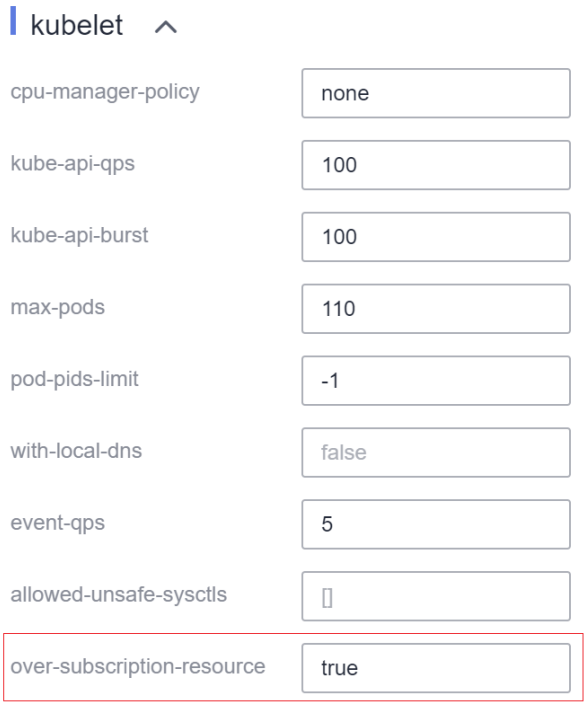
```
# kubectl edit cm volcano-scheduler-configmap -n kube-system
apiVersion: v1
data:
  volcano-scheduler.conf: |
    actions: "enqueue, allocate, preempt"   # Configure a preemption action.
    tiers:
    - plugins:
      - name: gang
        enablePreemptable: false
        enableJobStarving: false
      - name: priority
      - name: conformance
      - name: oversubscription
    - plugins:
      - name: drf
      - name: predicates
      - name: nodeorder
      - name: binpack
    - plugins:
      - name: cce-gpu-topology-predicate
      - name: cce-gpu-topology-priority
      - name: cce-gpu
```

Passo 2 Ative o recurso de excesso de assinatura do nó.

Um rótulo pode ser configurado para usar recursos com excesso de assinatura somente depois que o recurso de excesso de assinatura estiver ativado para um nó. Os nós relacionados podem ser criados somente em um pool de nós. Para ativar o recurso de excesso de assinatura, execute as seguintes etapas:

1. Crie um pool de nós.
2. Escolha **More** > **Manage** na coluna **Operation** do pool de nós criado.

3. Na janela **Manage Components** exibida, defina **over-subscription-resource** em **kubelet** como **true** e clique em **OK**.



Passo 3 Defina o rótulo de excesso de assinatura do nó.

O rótulo **volcano.sh/oversubscription** precisa ser configurado para um nó com excesso de assinatura. Se esse rótulo for definido para um nó e o valor for **true**, o nó é um nó com excesso de assinatura. Caso contrário, o nó não é um nó com excesso de assinatura.

```
kubectl label node 192.168.0.0 volcano.sh/oversubscription=true
```

Um nó com excesso de assinatura também suporta os limites de excesso de assinatura, conforme listado em **Tabela 6-7**. Por exemplo:

```
kubectl annotate node 192.168.0.0 volcano.sh/evicting-cpu-high-watermark=70
```

Consultar as informações do nó

```
# kubectl describe node 192.168.0.0
Name:              192.168.0.0
Roles:               <none>
Labels:            ...
                   volcano.sh/oversubscription=true
Annotations:       ...
                   volcano.sh/evicting-cpu-high-watermark: 70
```

Tabela 6-7 Anotações de excesso de assinatura de nó

Nome	Descrição
volcano.sh/evicting-cpu-high-watermark	Quando o uso da CPU de um nó excede o valor especificado, o despejo de trabalho off-line é acionado e o nó se torna não programável. O valor padrão é **80**, indicando que a remoção de tarefas off-line é acionada quando o uso da CPU de um nó excede 80%.

Nome	Descrição
volcano.sh/evicting-cpu-low-watermark	Depois que o despejo é acionado, o agendamento começa novamente quando o uso da CPU de um nó é menor do que o valor especificado. O valor padrão é **30**, indicando que o agendamento é iniciado novamente quando o uso da CPU de um nó é inferior a 30%.
volcano.sh/evicting-memory-high-watermark	Quando o uso de memória de um nó excede o valor especificado, a remoção de trabalho off-line é acionada e o nó se torna não programável. O valor padrão é **60**, indicando que a remoção de tarefas off-line é acionada quando o uso de memória de um nó excede 60%.
volcano.sh/evicting-memory-low-watermark	Depois que o despejo é acionado, o agendamento é iniciado novamente quando o uso de memória de um nó é menor do que o valor especificado. O valor padrão é **30**, indicando que o agendamento é iniciado novamente quando o uso de memória de um nó é inferior a 30%.
volcano.sh/oversubscription-types	Tipo de recurso de excesso de assinatura. As opções são as seguintes: ● CPU (CPU de excesso excesso de assinatura) ● memory (memória de excesso de assinatura) ● cpu,memory (CPU e memória de excesso de assinatura) O valor padrão é **cpu,memory**.

Passo 4 Crie recursos em uma classe de alta e baixa prioridade, respectivamente.

```
cat <<EOF | kubectl apply -f -

apiVersion: scheduling.k8s.io/v1
description: Used for high priority pods
kind: PriorityClass
metadata:
  name: production
preemptionPolicy: PreemptLowerPriority
value: 999999
---
apiVersion: scheduling.k8s.io/v1
description: Used for low priority pods
kind: PriorityClass
metadata:
  name: testing
preemptionPolicy: PreemptLowerPriority
value: -99999

EOF
```

Passo 5 Implemente tarefas on-line e off-line e configure priorityClasses para essas tarefas.

O rótulo **volcano.sh/qos-level** precisa ser adicionado à anotação para distinguir tarefas off-line. O valor é um número inteiro que varia de -7 a 7. Se o valor for menor que 0, a tarefa é uma tarefa off-line. Se o valor for maior ou igual a 0, a tarefa é uma tarefa de alta prioridade, ou seja, tarefa on-line. Você não precisa definir esse rótulo para tarefas on-line. Para ambos as tarefas on-line e off-line, defina **schedulerName** para **volcano** para ativar o volcano scheduler.

> **NOTA**
>
> As prioridades das tarefas on-line/on-line e off-line/off-line não são diferenciadas e a validade do valor não é verificada. Se o valor de **volcano.sh/qos-level** de uma tarefa off-line não for um número inteiro negativo variando de -7 a 0, a tarefa é processada como uma tarefa on-line.

Para uma tarefa off-line:

```
kind: Deployment
apiVersion: apps/v1
spec:
  replicas: 4
  template:
    metadata:
      annotations:
        metrics.alpha.kubernetes.io/custom-endpoints:
'[{"api":"","path":"","port":"","names":""}]'
        volcano.sh/qos-level: "-1"       # Offline job label
    spec:
      schedulerName: volcano             # The volcano scheduler is used.
      priorityClassName: testing         # Configure the testing priorityClass.
      ...
```

Para uma tarefa on-line:

```
kind: Deployment
apiVersion: apps/v1
spec:
  replicas: 4
  template:
    metadata:
      annotations:
        metrics.alpha.kubernetes.io/custom-endpoints:
'[{"api":"","path":"","port":"","names":""}]'
    spec:
      schedulerName: volcano          # The volcano scheduler is used.
      priorityClassName: production   # Configure the production priorityClass.
      ...
```

Passo 6 Execute o seguinte comando para verificar o número de recursos de excesso de assinatura e o uso de recursos:

kubectl describe node <*nodeIP*>

```
# kubectl describe node 192.168.0.0
Name:             192.168.0.0
Roles:              <none>
Labels:           ...
                  volcano.sh/oversubscription=true
Annotations:      ...
                  volcano.sh/oversubscription-cpu: 2335
                  volcano.sh/oversubscription-memory: 341753856
Allocatable:
  cpu:               3920m
  memory:            6263988Ki
Allocated resources:
  (Total limits may be over 100 percent, i.e., overcommitted.)
  Resource           Requests      Limits
  --------           --------      ------
  cpu                 4950m (126%)  4950m (126%)
  memory             1712Mi (27%)  1712Mi (27%)
```

----Fim

Exemplo de implementação

O exemplo a seguir é usado para descrever como implementar tarefas on-line e off-line no modo híbrido.

Passo 1 Suponha que um cluster tem dois nós: um nó com excesso de assinatura e um nó sem excesso de assinatura.

```
# kubectl get node
NAME           STATUS   ROLES    AGE    VERSION
192.168.0.173   Ready    <none>   4h58m   v1.19.16-r2-CCE22.5.1
192.168.0.3     Ready    <none>   148m    v1.19.16-r2-CCE22.5.1
```

- 192.168.0.173 é um nó com excesso de assinatura (com o rótulo **volcano.sh/oversubscription=true**).
- 192.168.0.3 é um nó sem excesso de assinatura (sem o rótulo **volcano.sh/oversubscription=true**).

```
# kubectl describe node 192.168.0.173
Name:               192.168.0.173
Roles:              <none>
Labels:             beta.kubernetes.io/arch=amd64
                    ...
                    volcano.sh/oversubscription=true
```

Passo 2 Envie solicitações de criação de tarefa off-line. Se os recursos forem suficientes, todos os tarefas off-line serão agendados para o nó com excesso de assinatura.

O modelo de tarefa off-line é o seguinte:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: offline
  namespace: default
spec:
  replicas: 2
  selector:
    matchLabels:
      app: offline
  template:
    metadata:
      labels:
        app: offline
      annotations:
        volcano.sh/qos-level: "-1"       # Offline job label
    spec:
      schedulerName: volcano             # The volcano scheduler is used.
      priorityClassName: testing         # Configure the testing priorityClass.
      containers:
        - name: container-1
          image: nginx:latest
          imagePullPolicy: IfNotPresent
          resources:
            requests:
              cpu: 500m
              memory: 512Mi
            limits:
              cpu: "1"
              memory: 512Mi
      imagePullSecrets:
        - name: default-secret
```

As tarefas off-line são agendadas para o nó com excesso de assinatura.

```
# kubectl get pod -o wide
NAME                       READY   STATUS   RESTARTS  AGE     IP             NODE
offline-69cdd49bf4-pmjp8   1/1     Running   0         5s    192.168.10.178
192.168.0.173
```



```
offline-69cdd49bf4-z8kxh   1/1    Running   0         5s    192.168.10.131
192.168.0.173
```

Passo 3 Envie solicitações de criação de tarefa on-line. Se os recursos forem suficientes, as tarefas on-line serão agendadas para o nó sem excesso de assinatura.

O modelo de tarefa on-line é o seguinte:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: online
  namespace: default
spec:
  replicas: 2
  selector:
    matchLabels:
      app: online
  template:
    metadata:
      labels:
        app: online
    spec:
      schedulerName: volcano                # The volcano scheduler is used.
      priorityClassName: production         # Configure the production
priorityClass.
      containers:
        - name: container-1
          image: resource_consumer:latest
          imagePullPolicy: IfNotPresent
          resources:
            requests:
              cpu: 1400m
              memory: 512Mi
            limits:
              cpu: "2"
              memory: 512Mi
      imagePullSecrets:
        - name: default-secret
```

As tarefas on-line são agendadas para o nó sem excesso de assinatura.

```
# kubectl get pod -o wide
NAME                      READY   STATUS   RESTARTS  AGE     IP              NODE
online-ffb46f656-4mwr6  1/1    Running   0         5s    192.168.10.146
192.168.0.3
online-ffb46f656-dqdv2   1/1    Running   0         5s    192.168.10.67
192.168.0.3
```

Passo 4 Melhore o uso de recursos do nó com excesso de assinatura e observe se o despejo de tarefas off-line é acionado.

Implemente tarefas on-line no nó com excesso de assinatura (192.168.0.173).

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: online
  namespace: default
spec:
  replicas: 2
  selector:
    matchLabels:
      app: online
  template:
    metadata:
      labels:
        app: online
    spec:
       affinity:                            # Submit an online job to an
oversubscribed node.
        nodeAffinity:
```



```
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
              - key: kubernetes.io/hostname
                operator: In
                values:
                - 192.168.0.173
      schedulerName: volcano                # The volcano scheduler is used.
      priorityClassName: production         # Configure the production
priorityClass.
      containers:
        - name: container-1
          image: resource_consumer:latest
          imagePullPolicy: IfNotPresent
          resources:
            requests:
              cpu: 700m
              memory: 512Mi
            limits:
              cpu: 700m
              memory: 512Mi
      imagePullSecrets:
        - name: default-secret
```

Submeta as tarefas on-line ou off-line ao nó com excesso de assinatura (192.168.0.173) ao mesmo tempo.

```
# kubectl get pod -o wide
NAME                       READY   STATUS   RESTARTS  AGE     IP              NODE
offline-69cdd49bf4-pmjp8  1/1     Running    0      13m   192.168.10.178
192.168.0.173
offline-69cdd49bf4-z8kxh  1/1     Running     0      13m   192.168.10.131
192.168.0.173
online-6f44bb68bd-b8z9p  1/1     Running     0     3m4s   192.168.10.18
192.168.0.173
online-6f44bb68bd-g6xk8  1/1     Running     0     3m12s   192.168.10.69
192.168.0.173
```

Observe o nó com excesso de assinatura (192.168.0.173). Você pode descobrir que existem recursos de excesso de assinatura e a taxa de alocação de CPU excede 100%.

```
# kubectl describe node 192.168.0.173
Name:              192.168.0.173
Roles:              <none>
Labels:              …
                    volcano.sh/oversubscription=true
Annotations:         …
                    volcano.sh/oversubscription-cpu: 2343
                    volcano.sh/oversubscription-memory: 3073653200
                    …
Allocated resources:
  (Total limits may be over 100 percent, i.e., overcommitted.)
  Resource               Requests      Limits
  --------               --------        ------
  cpu                    4750m (121%)  7350m (187%)
  memory                 3760Mi (61%)  4660Mi (76%)
                         …
```

Aumentar o uso da CPU de tarefas on-line no nó. Remoção de tarefa off-line é acionada.

```
# kubectl get pod -o wide
NAME                       READY   STATUS   RESTARTS  AGE     IP              NODE
offline-69cdd49bf4-bwdm7  1/1    Running   0       11m   192.168.10.208
192.168.0.3
offline-69cdd49bf4-pmjp8   0/1    Evicted    0       26m   <none>
192.168.0.173
offline-69cdd49bf4-qpdss   1/1     Running   0       11m   192.168.10.174
192.168.0.3
offline-69cdd49bf4-z8kxh   0/1     Evicted    0       26m   <none>
192.168.0.173
online-6f44bb68bd-b8z9p   1/1     Running   0       24m   192.168.10.18
192.168.0.173
```



```
online-6f44bb68bd-g6xk8   1/1     Running   0       24m   192.168.10.69
192.168.0.173
```

----Fim

Sugestões de manipulação

- Depois que o kubelet do nó com excesso de assinatura é reiniciado, a visualização de recursos do Volcano scheduler não é sincronizada com a do kubelet. Como resultado, OutOfCPU ocorre em algumas tarefas recém-agendados, o que é normal. Depois de um período de tempo, o Volcano scheduler pode programar corretamente tarefas on-line e off-line.
- Depois que as tarefas on-line e off-line forem enviadas, não é aconselhável alterar dinamicamente o tipo de tarefa (adicionando ou excluindo a anotação volcano.sh/qos-level: "-1") porque o kernel atual não suporta a mudança de uma tarefa offline para uma tarefa online.
- O CCE coleta o uso de recursos (CPU/memória) de todos os pods em execução em um nó com base nas informações de status no sistema cgroups. O uso de recursos pode ser diferente do uso de recursos monitorados, por exemplo, as estatísticas de recursos exibidas executando o comando **top**.
- Você pode adicionar recursos de excesso de assinatura (como CPU e memória) a qualquer momento.

  Você pode reduzir os tipos de recursos de excesso de assinatura somente quando a taxa de alocação de recursos não exceder 100%.
- Se uma tarefa off-line for implementada em um nó antes de uma tarefa on-line e a tarefa on-line não puder ser agendada devido a recursos insuficientes, configure uma classe de prioridade mais alta para a tarefa on-line do que para a tarefa off-line.
- Se houver apenas tarefas on-line em um nó e o limite de despejo for alcançado, as tarefas off-line agendadas para o nó atual serão despejadas em breve. Isto é normal.

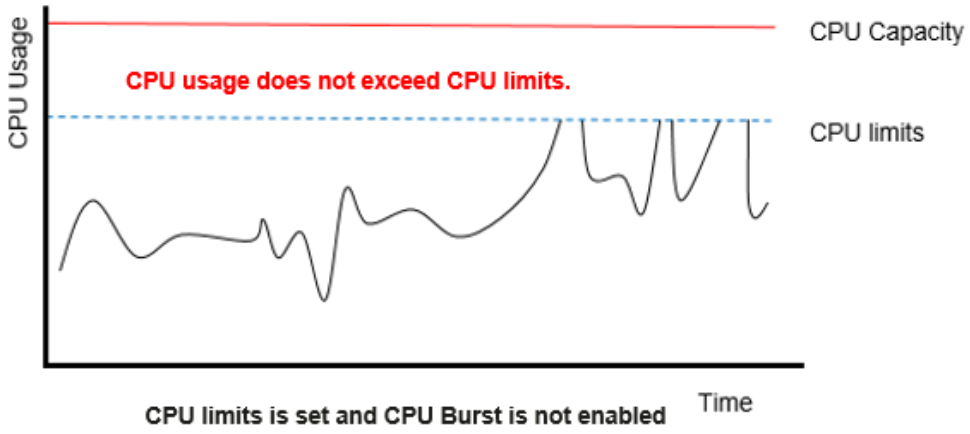
6.6.2 Intermitência de CPU

Se um limite de CPU for definido para um contêiner no pod, o uso da CPU do contêiner não poderá exceder o limite. A limitação frequente do tráfego de CPU afeta o desempenho do serviço e aumenta a latência da resposta de cauda longa, especialmente para serviços sensíveis à latência.

A intermitência da CPU é um mecanismo de limitação de tráfego elástico que permite exceder temporariamente o limite da CPU para reduzir o tempo de resposta de cauda longa dos serviços. Quando a cota de CPU para um serviço em cada período de agendamento de CPU permanece, o sistema acumula a cota de CPU. Se o limite de CPU precisar ser excedido em períodos de agendamento subsequentes, a cota de CPU acumulada poderá ser usada.

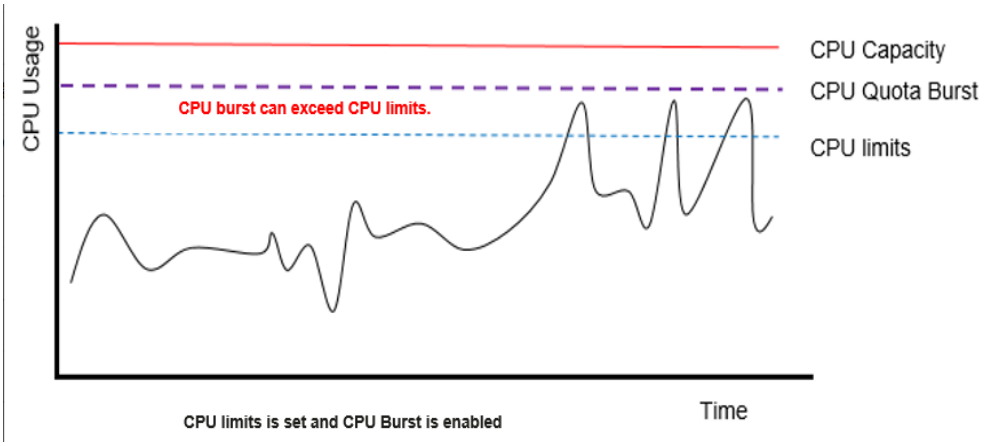
- Se a intermitência de CPU não estiver ativada, a cota de CPU de um contêiner não poderá exceder o limite e os recursos de intermitência acumulados não poderão ser usados.

Figura 6-10 Intermitência da CPU não ativada



- Depois que a intermitência de CPU é ativada, a cota de CPU de um contêiner pode exceder o limite para usar os recursos de intermitência acumulados.

Figura 6-11 Intermitência da CPU ativada



Restrições

- Versão do cluster: cluster do CCE Turbo v1.23.5-r0 ou posterior
- Versão de SO: Huawei Cloud EulerOS 2.0
- O complemento volcano de 1.9.0 ou posterior deve ser instalado no cluster e a função de implementação híbrida deve estar habilitada (isto é, **colocation_enable** nas configurações avançadas deve ser definido como **true**).

Procedimento

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster.

Passo 2 No painel de navegação à esquerda, escolha **Nodes**. Clique na guia **Node Pools**. Ao criar ou atualizar um pool de nós, ative a implementação híbrida de serviços on-line e offline em **Advanced Settings**.

- volcano.sh/oversubscription=true

- volcano.sh/colocation=true

Figura 6-12 Configurações de rótulo do nó

Passo 3 No painel de navegação à esquerda, escolha **Add-ons**. Clique em **Install** sob volcano. Na área **Advanced Settings**, defina **colocation_enable** como **true** para permitir a implementação híbrida de serviços on-line e off-line. Para obter detalhes sobre a instalação, consulte **Volcano scheduler**.

Se o complemento vulcão tiver sido instalado, clique em **Edit** para exibir ou modificar o parâmetro **colocation_enable**.

Figura 6-13 Ativar a implementação híbrida de serviços on-line e off-line

Passo 4 Ativar intermitência da CPU.

Depois de confirmar que o complemento volcano está funcionando, execute o seguinte comando para editar o parâmetro **configmap** de **volcano-agent-configuration** no namespace de **kube-system**. Se a opção **enable** estiver definida como **true**, a intermitência da CPU será ativada. Se **enable** for definido como **false**, a intermitência da CPU será desativada.

```
kubectl edit configmap -nkube-system volcano-agent-configuration
```

Exemplo:

```
cpuBurstConfig:
  enable: true
```

NOTA

Depois que a intermitência da CPU é desativada, essa função ainda é ativada nos pods existentes onde a intermitência da CPU foi ativada. A desativação da intermitência da CPU só tem efeito em novos pods.

Passo 5 Implemente uma carga de trabalho em um pool de nós onde a implementação híbrida foi ativada. Tome Nginx como exemplo. Defina **requests** para **2** e **limits** para **4** e crie um Serviço que possa ser acessado no cluster para a carga de trabalho.

```
apiVersion: apps/v1
kind: Deployment
```



```
metadata:
  name: nginx
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
      annotations:
        volcano.sh/enable-quota-burst: "true"
        volcano.sh/quota-burst-time: "200000"
    spec:
      containers:
      - name: container-1
        image: nginx:latest
        resources:
          limits:
            cpu: "4"
          requests:
            cpu: "2"
      imagePullSecrets:
        - name: default-secret
---
apiVersion: v1
kind: Service
metadata:
  name: nginx
  namespace: default
  labels:
    app: nginx
spec:
  selector:
    app: nginx
  ports:
    - name: cce-service-0
      targetPort: 80
      nodePort: 0
      port: 80
      protocol: TCP
  type: ClusterIP
```

Anotação	Obrigatório	Descrição
volcano.sh/enable-quota-burst	Sim	A intermitência da CPU está ativada para a carga de trabalho.

Anotação	Obrigatório	Descrição
volcano.sh/quota-burst-time	Não	Para garantir a estabilidade do agendamento da CPU e reduzir a contenção quando vários contêineres enfrentam intermitências de CPU ao mesmo tempo, o valor padrão de **CPU Burst** é o mesmo que o valor de **CPU Quota**. Ou seja, um contêiner pode usar no máximo duas vezes o valor do **CPU Limit**. Por padrão, **CPU Burst** é definido para todos os contêineres de serviço em um pod. Neste exemplo, o **CPU Limit** do contêiner é **4**, ou seja, o valor padrão é **400,000** (1 núcleo = 100.000), indicando que um máximo de quatro núcleos adicionais podem ser usados após o valor de **CPU Limit** ser atingido.

Passo 6 Verificar intermitência da CPU.

Você pode usar a ferramenta wrk para aumentar a carga da carga de trabalho e observar a latência do serviço, a limitação de tráfego e o limite de CPU excedendo quando a Intermitência de CPU é ativada e desativada, respectivamente.

1. Execute o seguinte comando para aumentar a carga do pod. <*service_ip*> indica o endereço IP de serviço associado ao pod.
   ```
   # Download and install the wrk tool on the node.
   # The Gzip compression module is enabled in the Apache configuration to simulate the computing logic for the server to process requests.
   # Run the following command to increase the load. Note that you need to change the IP address of the target application.
   wrk -H "Accept-Encoding: deflate, gzip" -t 4 -c 28 -d 120  --latency --timeout 2s http://<service_ip>
   ```
2. Obtenha o ID do pod.
   ```
   kubectl get pod -n <namespace> <pod_name> -o jsonpath='{.metadata.uid}'
   ```
3. Você pode executar os seguintes comandos no nó para exibir o status de limitação de tráfego e o limite de CPU excedendo o status. No comando, <*pod_id*> indica o ID do pod.
   ```
   cat /sys/fs/cgroup/cpu/kubepods/burstable/pod<pod_id>/cpu.stat
   ```
   Informação semelhante à seguinte é exibida.
   ```
   nr_periods 0  # Number of scheduling periods
   nr_throttled 0  # Traffic limiting times
   throttled_time 0  # Traffic limiting duration (ns)
   nr_bursts 0 # CPU Limit exceeding times
   burst_time 0 # Total Limit exceeding duration
   ```


Tabela 6-8 Resumo do resultado neste exemplo

Intermitência de CPU	Latência P99	nr_throttled Tempos de limitação de tráfego	throttled_time Duração do limite de tráfego	nr_bursts Limite de tempos excedentes	bursts_time Limite total que excede a duração
Não ativada	2,96 ms	986	14,3s	0	0
Ativada	456 µs	0	0	469	3,7s

----Fim

6.6.3 Garantia de largura de banda de rede de egress

A garantia de largura de banda da rede de egress é implementada definindo as prioridades da rede. Ela possui as seguintes vantagens:

- A largura de banda da rede de egress usada por serviços on-line e off-line é balanceada para garantir largura de banda de rede suficiente para serviços on-line. Quando o limite for atingido para serviços on-line, a utilização da largura de banda dos serviços off-line será reduzida.
- Quando os serviços on-line ocupam um pequeno número de recursos de rede, os serviços off-line podem usar mais largura de banda. Quando os serviços on-line ocupam um grande número de recursos de rede, o uso de recursos dos serviços off-line será reduzido para garantir que mais largura de banda de rede priorize os serviços on-line.

Restrições

Para usar a garantia de largura de banda de rede de egress, os seguintes requisitos devem ser atendidos:

- Somente nós que executam o Huawei Cloud EulerOS 2.0 são suportados.
- Apenas os clusters do CCE Turbo de v1.23 ou posterior são suportados.
- O complemento volcano de 1.9.0 ou posterior deve ser instalado no cluster e a função de implementação híbrida deve estar habilitada (**colocation_enable** nas configurações avançadas deve ser definido como **true**).
- Antes de ativar, modificar ou desativar a garantia de largura de banda de rede de egress, certifique-se de que o complemento volcano esteja funcionando.
- Para pods que foram executados no nó antes do complemento volcano ser instalado, reinicie manualmente os pods depois de ativar a garantia de largura de banda da rede para que o recurso possa entrar em vigor.
- Desinstalar o complemento volcano ou desativar a função de implementação híbrida (isto é, definindo **colocation_enable** nas configurações avançadas para **false**) não afeta a garantia de largura de banda de rede de egress existente no nó. Para desativar esse recurso, consulte **Desativação da garantia de largura de banda de rede de egress**.
- Se o limite de largura de banda estiver ativado, o cache da pilha de protocolo poderá ser empilhado. Para protocolos sem mecanismos de contrapressão, como UDP, perda de pacotes e ENOBUFS pode ocorrer.

- O limite de largura de banda aumenta o risco de que os serviços off-line não consigam obter largura de banda. Os serviços podem até passar fome devido à largura de banda insuficiente ou a verificação de saúde do pod pode falhar.
- A garantia de largura de banda de rede de egress não é priorizada nos seguintes cenários:
  - Quando o **limite de largura de banda da rede** é usado para pods híbridos on-line ou offline, a prioridade do limite de largura de banda da rede é maior do que a da função atual.
  - Quando um pod usa a rede de nó (**hostNetwork**), a função de garantia de largura de banda da rede de egress não entra em vigor.

Procedimento

O seguinte descreve como ativar ou desativar a garantia de largura de banda de rede de egress.

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster.

Passo 2 No painel de navegação à esquerda, escolha **Nodes**. Clique na guia **Node Pools**. Ao criar ou atualizar um pool de nós, ative a implementação híbrida de serviços on-line e offline em **Advanced Settings**.

- volcano.sh/oversubscription=true
- volcano.sh/colocation=true

Figura 6-14 Configurações de rótulo do nó

Passo 3 No painel de navegação à esquerda, escolha **Add-ons**. Clique em **Install** sob volcano. Na área **Advanced Settings**, defina **colocation_enable** como **true** para permitir a implementação híbrida de serviços on-line e off-line. Para obter detalhes sobre a instalação, consulte **Volcano scheduler**.

Se o complemento vulcão tiver sido instalado, clique em **Edit** para exibir ou modificar o parâmetro **colocation_enable**.

Figura 6-15 Ativar a implementação híbrida de serviços on-line e off-line

Passo 4 Ative ou modifique parâmetros para garantia de largura de banda de rede de egress.

Depois de confirmar que o complemento volcano está funcionando, execute o seguinte comando para editar o parâmetro **configmap** de **volcano-agent-configuration** no namespace de **kube-system**. Se **enable** for definido como **true**, a garantia de largura de banda de rede de egress será ativada e os parâmetros relacionados poderão ser modificados.

```
kubectl edit configmap -nkube-system volcano-agent-configuration
```

Exemplo:

```
networkQosConfig:
  enable: true
  onlineBandwidthWatermarkPercent: 80
  offlineLowBandwidthPercent: 10
  offlineHighBandwidthPercent: 40
```

NOTA

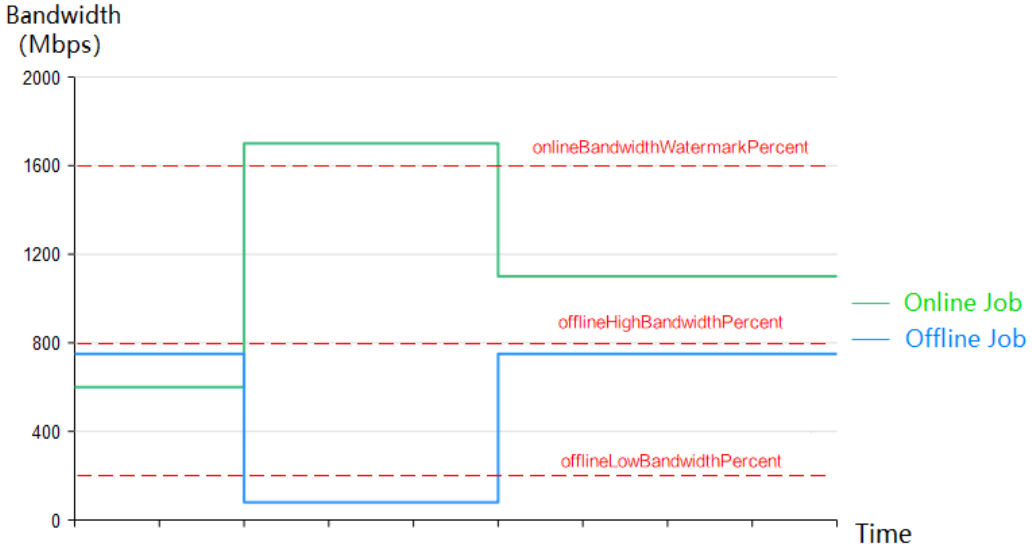
Os parâmetros modificados entram em vigor para todos os nós que executam o Huawei Cloud EulerOS 2.0 no cluster.

Tabela 6-9 Parâmetros de networkQosConfig

Nome	Descrição
enable	Especifica se deve ativar a funcionalidade de garantia de largura de banda da rede de egress.
onlineBandwidthWatermarkPercent	Relação entre o limite de largura de banda total dos serviços online e a largura de banda de base do modelo, ou seja: *Total bandwidth threshold of online services = Baseline bandwidth of the model x onlineBandwidthWatermarkPercent/100* Intervalo de valores: 1–1000 Valor padrão: **80**
offlineLowBandwidthPercent	Relação entre o uso total máximo de largura de banda dos serviços off-line e a largura de banda de base do modelo quando o uso de largura de banda dos serviços on-line excede o limite. Se o uso total da largura de banda dos serviços on-line no mesmo nó exceder o valor de *Baseline bandwidth of the model x onlineBandwidthWatermarkPercent/100*, o uso total da largura de banda dos serviços off-line no mesmo nó não pode exceder o valor de *Baseline bandwidth of the model x offlineLowBandwidthPercent/100*. Intervalo de valores: 1–1000 Valor padrão: **10**

Nome	Descrição
offlineHighBand-widthPercent	Proporção entre o uso total máximo de largura de banda dos serviços off-line e a largura de banda de base do modelo quando o uso de largura de banda dos serviços on-line não excede o limite. Se o uso total da largura de banda dos serviços on-line no mesmo nó não exceder o valor de *Baseline bandwidth of the model x onlineBandwidthWatermarkPercent/100*, o uso total da largura de banda dos serviços off-line no mesmo nó não pode exceder o valor de *Baseline bandwidth of the model x offlineHighBand-widthPercent/100*. Intervalo de valores: 1–1000 Valor padrão: **40**

Figura 6-16 Exemplo de garantia de largura de banda de rede de egress



Passo 5 Para desabilitar a garantia de largura de banda de rede de egress, depois de confirmar que o complemento volcano está funcionando, execute o seguinte comando para editar o parâmetro **configmap** de **volcano-agent-configuration** no namespace de **kube-system**. Defina **enable** como **false**.

```
kubectl edit configmap -nkube-system volcano-agent-configuration
```

Modifique os seguintes parâmetros:

```
networkQosConfig:
  enable: false
  onlineBandwidthWatermarkPercent: 80
  offlineLowBandwidthPercent: 10
  offlineHighBandwidthPercent: 40
```

----Fim

7 Rede

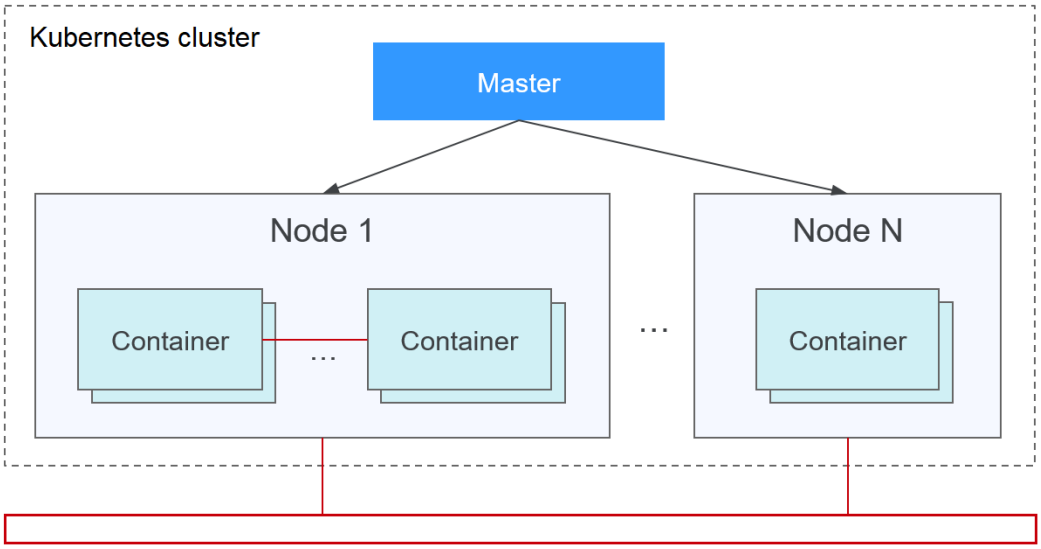
7.1 Visão geral

Você pode aprender sobre uma rede de cluster a partir dos dois aspectos a seguir:

- Como é uma rede de cluster? Um cluster consiste em vários nós, e os pods (ou contêineres) estão sendo executados nos nós. Os nós e os contêineres precisam se comunicar uns com os outros. Para obter detalhes sobre os tipos de rede de cluster e suas funções, consulte **Estrutura de rede do cluster**.
- Como o acesso ao pod é implementado em um cluster? Acessar um pod ou contêiner é um processo de acesso aos serviços de um usuário. O Kubernetes fornece **Serviço** e resolve **Ingress** problemas de acesso ao pod. Esta seção resume cenários comuns de acesso à rede. Você pode selecionar o cenário adequado com base nos requisitos do site. Para obter detalhes sobre os cenários de acesso à rede, consulte **Cenários de acesso**.

Estrutura de rede do cluster

Todos os nós no cluster estão localizados em uma VPC e usam a rede da VPC. A rede de contêineres é gerenciada por complementos de rede dedicados.



- **Rede de nó**

  Uma rede de nó atribui endereços IP a hosts (nós na figura acima) em um cluster. Selecione uma sub-rede da VPC como a rede de nó do cluster do CCE. O número de endereços IP disponíveis em uma sub-rede determina o número máximo de nós (incluindo nós mestres e nós de trabalho) que podem ser criados em um cluster. Essa quantidade também é afetada pela rede de contêineres. Para obter detalhes, consulte o modelo de rede de contêiner.

- **Rede de contêiner**

  Uma rede de contêineres atribui endereços IP a contêineres em um cluster. O CCE herda o modelo de rede IP-Por-Pod-Por-Rede do Kubernetes. Ou seja, cada pod tem um endereço IP independente em um plano de rede e todos os contêineres em um pod compartilham o mesmo namespace de rede. Todos os pods em um cluster existem em uma rede plana conectada diretamente. Eles podem acessar uns aos outros através de seus endereços IP sem usar NAT. O Kubernetes fornece apenas um mecanismo de rede para pods, mas não configura diretamente redes de pods. A configuração de redes de pods é implementada por complementos de rede de contêineres específicos. Os complementos de rede de contêineres são responsáveis por configurar redes para pods e gerenciar endereços IP de contêiner.

  Atualmente, o CCE suporta os seguintes modelos de rede de contêineres:

  – Rede de túneis de contentores: a rede de túnel de contêiner é construída em, mas independente da rede de nó através do encapsulamento de túnel. Este modelo de rede usa VXLAN para encapsular pacotes Ethernet em pacotes UDP e transmiti-los em túneis. Open vSwitch serve como o comutador virtual de back-end.
  – Rede da VPC: a rede da VPC usa o roteamento da VPC para se integrar à rede subjacente. Esse modelo de rede se aplica a cenários de alto desempenho. O número máximo de nós permitidos em um cluster depende da cota de rota em uma rede da VPC. Cada nó é atribuído a um bloco CIDR de um tamanho fixo. Esse modelo de rede é livre de sobrecarga de encapsulamento de túnel e supera o modelo de rede de túnel de contêiner. Além disso, como o roteamento de VPC inclui rotas para endereços IP de nó e o bloco CIDR de contêiner, os pods de contêiner em um cluster podem ser acessados diretamente de fora do cluster.
  – Desenvolvido pelo CCE, o Cloud Native Network 2.0 integra profundamente as interfaces de rede elásticas (ENIs) e as interfaces de sub-rede (sub-ENIs) da VPC. Os endereços IP do contêiner são alocados a partir do bloco CIDR da VPC. A rede de passagem do ELB é suportada para direcionar solicitações de acesso a contêineres. Grupos de segurança e IPs elásticos (EIPs) são obrigados a oferecer alto desempenho.

  O desempenho, a escala de rede e os cenários de aplicações de uma rede de contêineres variam de acordo com o modelo da rede de contêineres. Para obter detalhes sobre as funções e recursos de diferentes modelos de rede de contêineres, consulte **Visão geral**.

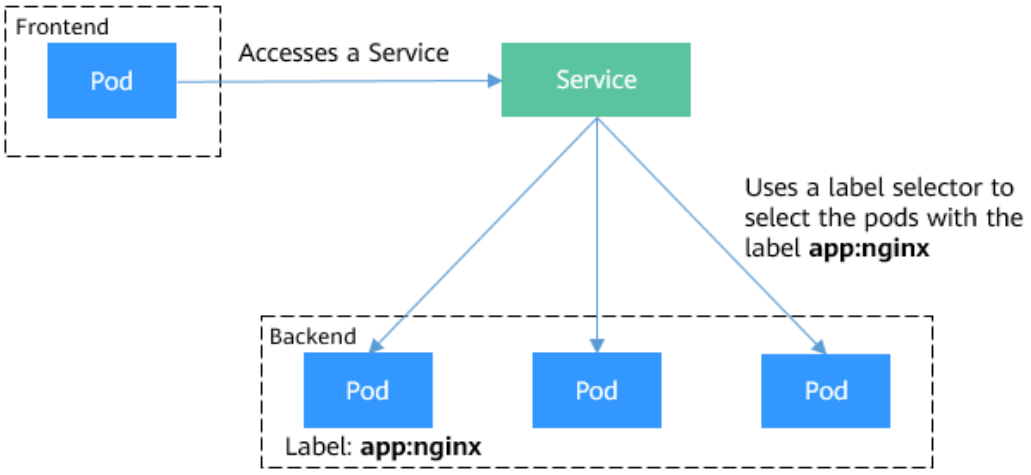
- **Rede de serviço**

  O Serviço também é um objeto do Kubernetes. Cada Serviço tem um endereço IP estático. Ao criar um cluster no CCE, você pode especificar o bloco CIDR do Serviço. O bloco CIDR de Serviço não pode se sobrepor ao bloco CIDR de nó ou contêiner. O bloco CIDR de Serviço pode ser usado somente dentro de um cluster.

Serviço

Um Serviço é usado para acesso ao pod. Com um endereço IP estático, um Serviço encaminha o tráfego de acesso aos pods e executa o balanceamento de carga para esses pods.

Figura 7-1 Acessar pods por meio de um serviço



Você pode configurar os seguintes tipos de Serviços:

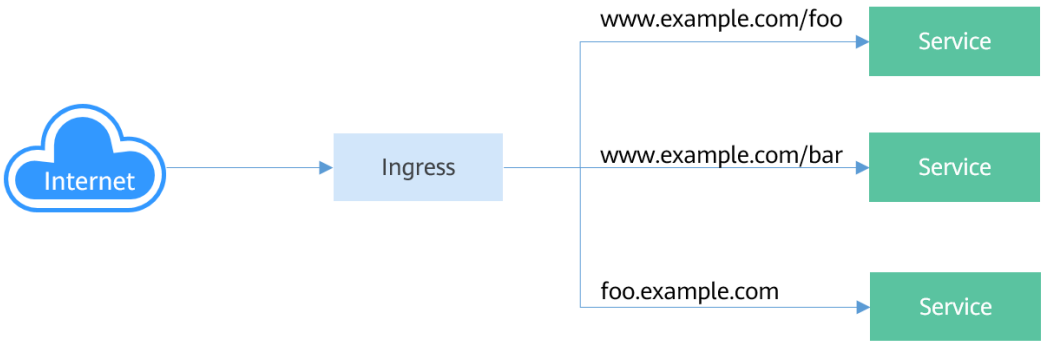
- ClusterIP: usado para tornar o Serviço acessível apenas de dentro de um cluster.
- NodePort: usado para acesso de fora de um cluster. Um Serviço NodePort é acessado através da porta no nó.
- LoadBalancer: usado para acesso de fora de um cluster. É uma extensão de NodePort para o qual um balanceador de carga roteia e os sistemas externos só precisam acessar o balanceador de carga.
- DNAT: usada para acesso de fora de um cluster. Ela converte endereços para nós do cluster e permite que vários nós do cluster compartilhem um EIP.

Para obter detalhes sobre o Serviço, consulte **Visão geral**.

Ingress

Serviços encaminham solicitações usando os protocolos TCP e UDP da camada 4. Ingresses encaminham solicitações usando os protocolos HTTP e HTTPS de camada 7. Nomes de domínio e caminhos podem ser usados para obter granularidades mais finas.

Figura 7-2 Ingress e Serviço



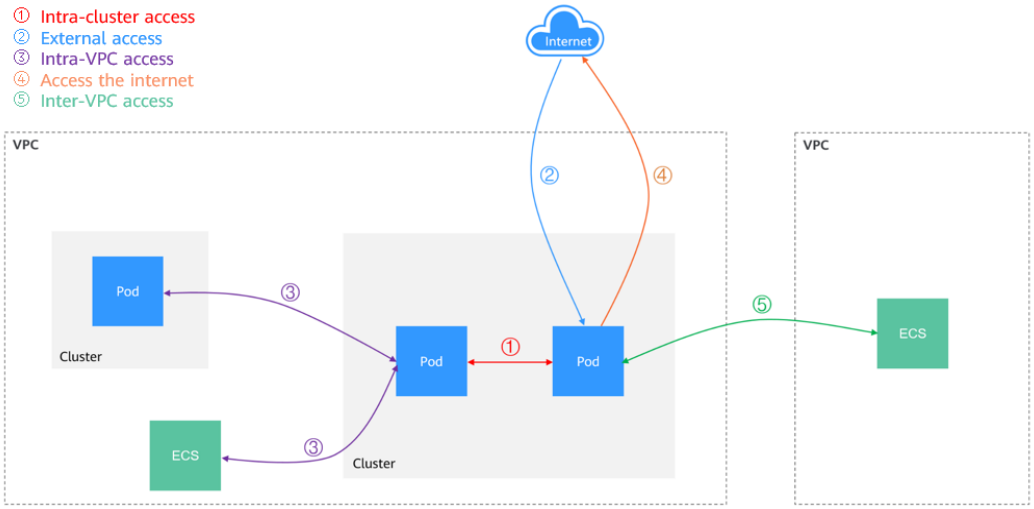
Para obter detalhes sobre o ingress, consulte **Visão geral**.

Cenários de acesso

Os cenários de acesso à carga de trabalho podem ser categorizados da seguinte forma:

- Acesso intra-cluster: um Serviço ClusterIP é usado para cargas de trabalho no mesmo cluster para acessar uns aos outros.
- Acesso de fora de um cluster: um Serviço (tipo NodePort ou LoadBalancer) ou um ingress é recomendado para uma carga de trabalho fora de um cluster para acessar cargas de trabalho no cluster.
  - Acesso através da rede pública: um EIP deve ser vinculado ao nó ou ao balanceador de carga.
  - Acesso através da rede privada: a carga de trabalho pode ser acessada por meio do endereço IP interno do nó ou do balanceador de carga. Se as cargas de trabalho estiverem localizadas em VPCs diferentes, é necessária uma conexão de emparelhamento para permitir a comunicação entre VPCs diferentes.
- A carga de trabalho pode acessar a rede externa da seguinte forma:
  - Acessar uma intranet: a carga de trabalho acessa o endereço de intranet, mas o método de implementação varia dependendo dos modelos de rede de contêiner. Certifique-se de que o grupo de segurança de par permita as solicitações de acesso do bloco CIDR do contêiner. Para mais detalhes, consulte **Configuração do acesso dentro da VPC**.
  - Acessar uma rede pública: atribuir um EIP ao nó em que a carga de trabalho é executada (quando a rede da VPC ou o modelo de rede de túnel é usado) vincular um EIP ao endereço IP do pod (quando o modelo Cloud Native Network 2.0 é usado) ou configurar regras SNAT por meio do gateway NAT. Para mais detalhes, consulte **Acesso a redes públicas a partir de um contêiner**.

Figura 7-3 Diagrama de acesso à rede



7.2 Modelos de rede de contêineres

7.2.1 Visão geral

A rede de contêineres atribui endereços IP a pods em um cluster e fornece serviços de rede. No CCE, você pode selecionar os seguintes modelos de rede para o cluster:

- **Rede de túneis**
- **Rede da VPC**
- **Cloud Native Network 2.0**

Comparação de modelos de rede

Tabela 7-1 descreve as diferenças de modelos de rede suportados pelo CCE.

⚠ CUIDADO

Depois que um cluster é criado, o modelo de rede não pode ser alterado.

Tabela 7-1 Comparação de modelos de rede

Rede de túneis	Rede de túneis	Rede da VPC	Cloud Native Network 2.0
Cenários de aplicações	• Cenários comuns de serviço de contêiner • Cenários que não têm altos requisitos de latência de rede e largura de banda	• Cenários que têm altos requisitos de latência de rede e largura de banda • Os contêineres podem se comunicar com VMs usando uma estrutura de registro de microsserviços, como Dubbo e CSE.	• Cenários que têm altos requisitos de latência de rede, largura de banda e desempenho • Os contêineres podem se comunicar com VMs usando uma estrutura de registro de microsserviços, como Dubbo e CSE.
Tecnologia principal	OVS	IPvlan e rota da VPC	ENI da VPC/sub-ENI
Clusters aplicáveis	Cluster padrão do CCE	Cluster padrão do CCE	Cluster do CCE Turbo
Isolamento de rede	NetworkPolicy do Kubernetes nativo para pods	Não	Os pods suportam o isolamento do grupo de segurança.
Rede de passagem	Não	Não	Sim

Rede de túneis	Rede de túneis	Rede da VPC	Cloud Native Network 2.0
Gerenciamento de endereços IP	● O bloco CIDR de contêiner é alocado separadamente. ● Os blocos CIDR são divididos por nó e podem ser alocados dinamicamente (os blocos CIDR podem ser adicionados dinamicamente após serem alocados)	● O bloco CIDR de contêiner é alocado separadamente. ● Os blocos CIDR são divididos por nó e alocados estaticamente (o bloco CIDR não pode ser alterado após a criação de um nó).	O bloco CIDR do contêiner é dividido da sub-rede VPC e não precisa ser alocado separadamente.
Desempenho de rede	Perda de desempenho devido ao encapsulamento VXLAN	Sem encapsulamento de túnel. Os pacotes de nó cruzado são encaminhados por meio de roteadores da VPC, proporcionando desempenho equivalente ao da rede host.	A rede de contêineres é integrada à rede da VPC, eliminando a perda de desempenho.
Escala de rede	Um máximo de 2000 nós são suportados.	Adequado para redes de pequena e média escala devido à limitação nas tabelas de roteamento da VPC. Recomenda-se que o número de nós seja menor ou igual a 1000. Cada vez que um nó é adicionado ao cluster, uma rota é adicionada às tabelas de roteamento da VPC (incluindo as padrão e as personalizadas). Portanto, a escala de cluster é limitada pelas tabelas de roteamento da VPC. Para obter detalhes sobre tabelas de roteamento, consulte **Observações e restrições**.	Um máximo de 2000 nós são suportados.

AVISO

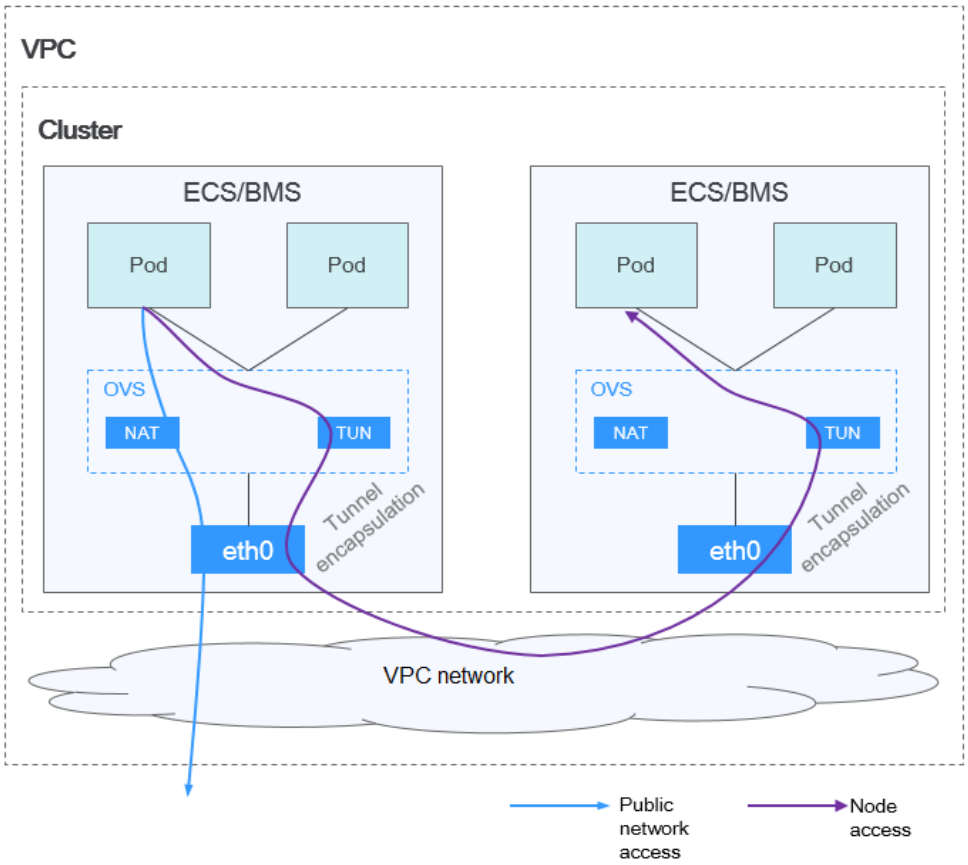
1. A escala de um cluster que usa o modelo de rede da VPC é limitada pelas rotas personalizadas da VPC. Portanto, estime o número de nós necessários antes de criar um cluster.
2. A escala de um cluster que usa o modelo Cloud Native Network 2.0 depende do tamanho do bloco CIDR de sub-rede da VPC selecionado para a definição de anexo de rede. Antes de criar um cluster, avalie a escala do cluster.
3. Por padrão, a rede de roteamento da VPC oferece suporte à comunicação direta entre contêineres e hosts na mesma VPC. Se uma política de conexão de emparelhamento for configurada entre a VPC e outra VPC, os contêineres poderão se comunicar diretamente com hosts na VPC de par. Além disso, em cenários de rede híbrida, como Direct Connect e VPN, a comunicação entre contêineres e hosts na extremidade do peer também pode ser alcançada com um planejamento adequado.
4. Não altere a máscara do bloco CIDR primário na VPC após a criação de um cluster. Caso contrário, a rede será anormal.

7.2.2 Rede de túneis de contêineres

Modelo de rede de túnel de contêiner

A rede de túnel de contêiner é construída em, mas independente da rede de nó através do encapsulamento de túnel. Este modelo de rede usa VXLAN para encapsular pacotes Ethernet em pacotes UDP e transmiti-los em túneis. Open vSwitch serve como o comutador virtual de back-end. Embora com alguns custos de desempenho, o encapsulamento de pacotes e a transmissão de túnel permitem maior interoperabilidade e compatibilidade com recursos avançados (como isolamento baseado em políticas de rede) para os cenários mais comuns.

Figura 7-4 Rede de túneis de contêineres



Comunicação pod a pod

- No mesmo nó: os pacotes são encaminhados diretamente através da ponte OVS no nó.
- Entre os nós: os pacotes são encapsulados na ponte OVS e encaminhados para o nó de par.

Vantagens e desvantagens

Vantagens

- A rede de contêineres é desacoplada da rede de nó e não é limitada pelas cotas de VPC e pela velocidade de resposta (como o número de rotas da VPC, o número de ENIs elásticos e a velocidade de criação).
- O isolamento de rede é suportado. Para mais detalhes, consulte **Políticas de rede**.
- Limites de largura de banda são suportados.
- Redes em grande escala são suportadas.

Desvantagens

- Alta sobrecarga de encapsulamento, rede complexa e baixo desempenho
- Os pods não podem usar diretamente funcionalidades como EIPs e grupos de segurança.
- As redes externas não podem ser conectadas diretamente aos endereços IP do contêiner.

Cenários aplicáveis

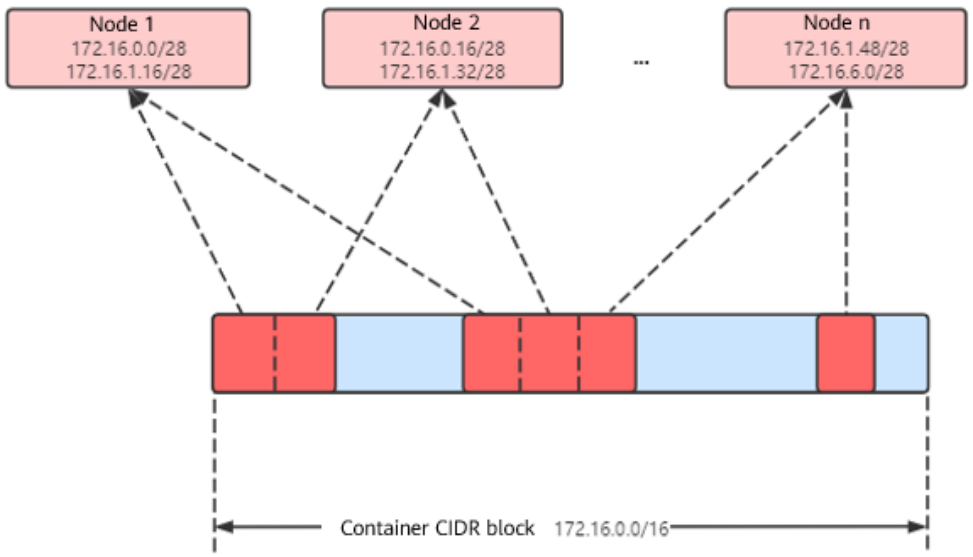
- Baixos requisitos de desempenho: como a rede de túnel de contêiner requer encapsulamento de túnel VXLAN adicional, ela tem cerca de 5% a 15% de perda de desempenho quando comparada com os outros dois modelos de rede de contêiner. Portanto, a rede de túnel de contêiner aplica-se aos cenários que não têm requisitos de alto desempenho, como aplicações da Web e serviços de middle-end e back-end com um pequeno número de solicitações de acesso.
- Rede em grande escala: diferentemente da rede da VPC que é limitada pela cota de rota da VPC, a rede de túnel de contêiner não tem nenhuma restrição na infraestrutura. Além disso, a rede de túnel de contêiner controla o domínio de transmissão até o nível do nó. A rede de túnel de contêiner suporta um máximo de 2000 nós.

Gerenciamento de endereços IP de contêiner

A rede de túnel de contêiner aloca endereços IP de contêiner de acordo com as seguintes regras:

- O bloco CIDR do contêiner é alocado separadamente, o que é irrelevante para o bloco CIDR do nó.
- Os endereços IP são alocados por nó. Um ou mais blocos CIDR com um tamanho fixo (16 por padrão) são alocados para cada nó em um cluster a partir do bloco CIDR do contêiner.
- Quando os endereços IP em um nó são usados, você pode solicitar um novo bloco CIDR.
- O bloco CIDR do contêiner aloca ciclicamente blocos CIDR para novos nós ou nós existentes em sequência.
- Os pods agendados para um nó são endereços IP alocados ciclicamente de um ou mais blocos CIDR alocados para o nó.

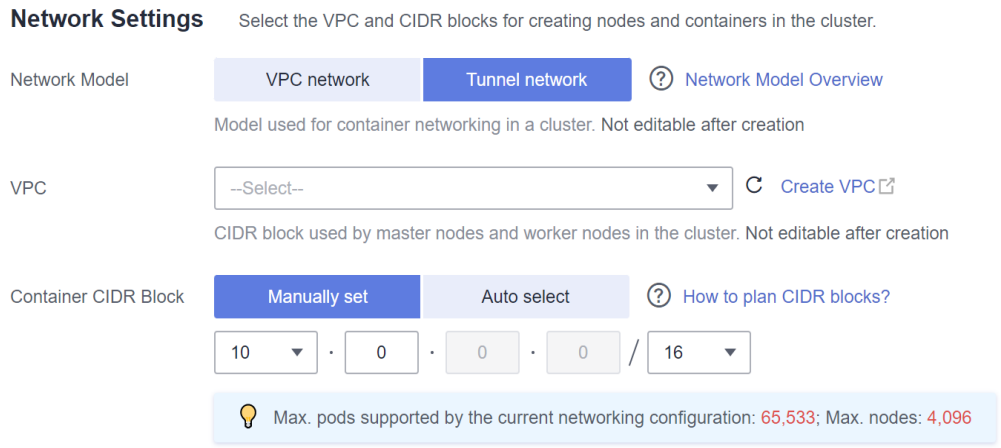
Figura 7-5 Alocação de endereço IP da rede de túnel de contêiner



Número máximo de nós que podem ser criados no cluster usando a rede de túnel de contêiner = número de endereços IP no bloco CIDR do contêiner / tamanho do bloco CIDR de IP alocado ao nó pelo bloco CIDR do contêiner por vez (16 por padrão)

Por exemplo, se o bloco CIDR do contêiner for 172.16.0.0/16, o número de endereços IP será 65536. Se 16 endereços IP são atribuídos a um nó de cada vez, um máximo de 4096 (65536/16) nós podem ser criados no cluster. Este é um caso extremo. Se 4096 nós forem criados, um máximo de 16 pods podem ser criados para cada nó, pois apenas 16 blocos CIDR IP são alocados para cada nó. Além disso, o número de nós que podem ser criados em um cluster também depende da rede do nó e da escala do cluster.

Figura 7-6 Selecionar um modelo de rede (ao criar o cluster)



Recomendação para planejamento de blocos CIDR

Conforme descrito em **Estrutura de rede do cluster**, os endereços de rede em um cluster podem ser divididos em três partes: rede de nó, rede de contêineres e rede de serviço. Ao planejar endereços de rede, considere os seguintes aspectos:

- Os três blocos CIDR não podem se sobrepor. Caso contrário, ocorre um conflito. Todas as sub-redes (incluindo aquelas criadas a partir do bloco CIDR secundário) na VPC em que o cluster reside não podem entrar em conflito com o contêiner e os blocos CIDR de serviço.
- Certifique-se de que cada bloco CIDR tenha endereços IP suficientes.
  - Os endereços IP no bloco CIDR do nó devem corresponder à escala do cluster. Caso contrário, os nós não podem ser criados devido a endereços IP insuficientes.
  - Os endereços IP no bloco CIDR do contêiner devem corresponder à escala de serviço. Caso contrário, os pods não podem ser criados devido a endereços IP insuficientes. O número de pods que podem ser criados em cada nó também depende de outras configurações de parâmetros. Para mais detalhes, consulte **Número máximo de pods que podem ser criados em um nó**.

Exemplo de acesso à rede de túnel de contêiner

Crie um cluster que use o modelo de rede de túnel de contêiner. Crie uma Implementação no cluster.

```
kind: Deployment
apiVersion: apps/v1
```



```
metadata:
  name: example
  namespace: default
spec:
  replicas: 4
  selector:
    matchLabels:
      app: example
  template:
    metadata:
      labels:
        app: example
    spec:
      containers:
        - name: container-0
          image: 'nginx:perl'
          resources:
            limits:
              cpu: 250m
              memory: 512Mi
            requests:
              cpu: 250m
              memory: 512Mi
      imagePullSecrets:
        - name: default-secret
```

Visualize o pod criado.

```
$ kubectl get pod -owide
NAME                         READY   STATUS    RESTARTS   AGE     IP
NODE           NOMINATED NODE   READINESS GATES
example-5bdc5699b7-5rvq4   1/1     Running   0          3m28s   10.0.0.20
192.168.0.42   <none>           <none>
example-5bdc5699b7-984j9   1/1     Running   0          3m28s   10.0.0.21
192.168.0.42   <none>           <none>
example-5bdc5699b7-lfxkm   1/1     Running   0          3m28s   10.0.0.22
192.168.0.42   <none>           <none>
example-5bdc5699b7-wjcmg   1/1     Running   0          3m28s   10.0.0.52
192.168.0.64   <none>           <none>
```

Nesse caso, o endereço IP do pod não pode ser acessado diretamente fora do cluster na mesma VPC. Essa é uma característica da rede de túnel de contêineres.

No entanto, o pod pode ser acessado a partir de um nó no cluster ou no pod. Como mostrado na figura a seguir, o pod pode ser acessado diretamente do contêiner.

```
$ kubectl exec -it example-5bdc5699b7-5rvq4 -- curl 10.0.0.21
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>
```



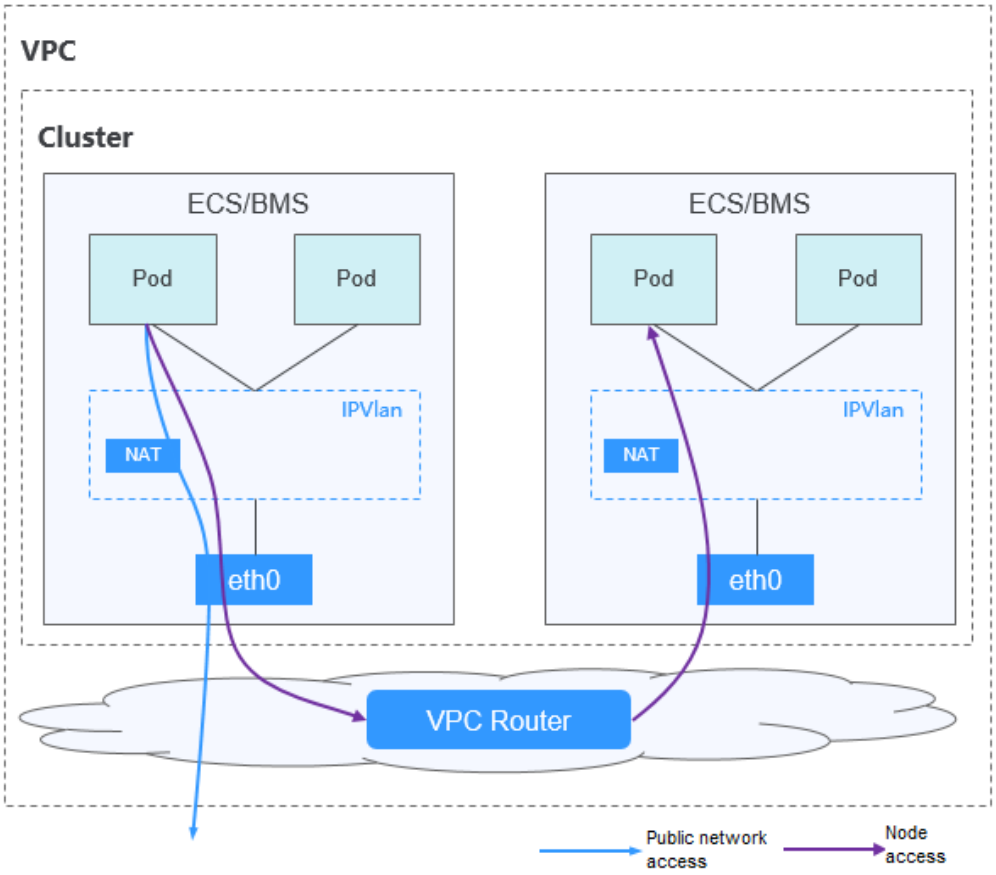
```
<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

7.2.3 Rede da VPC

Definição do modelo

A rede da VPC usa o roteamento da VPC para se integrar à rede subjacente. Esse modelo de rede é adequado para cenários de alto desempenho. O número máximo de nós permitidos em um cluster depende da cota de rota da VPC. Cada nó é atribuído a um bloco CIDR de um tamanho fixo. Esse modelo de rede é livre de sobrecarga de encapsulamento de túnel e supera o modelo de rede de túnel de contêiner. Além disso, como o roteamento de VPC inclui rotas para endereços IP de nó e o bloco CIDR de contêiner, os pods de contêiner em um cluster podem ser acessados diretamente de ECSs na mesma VPC fora do cluster.

Figura 7-7 Modelo de rede da VPC



Comunicação pod a pod

- No mesmo nó: os pacotes são encaminhados diretamente através de IPvlan.
- Entre os nós: os pacotes são encaminhados para o gateway padrão por meio de rotas padrão e, em seguida, para o nó do par por meio das rotas da VPC.

Vantagens e desvantagens

Vantagens

- Nenhum encapsulamento de túnel é necessário, de modo que os problemas de rede são fáceis de localizar e o desempenho é alto.
- Na mesma VPC, a rede externa do cluster pode ser conectada diretamente ao endereço IP do contêiner.

Desvantagens

- O número de nós é limitado pela cota de rota da VPC.
- Cada nó é atribuído a um bloco CIDR de um tamanho fixo, o que leva a um desperdício de endereços IP no bloco CIDR do contêiner.
- Os pods não podem usar diretamente funcionalidades como EIPs e grupos de segurança.

Cenários aplicáveis

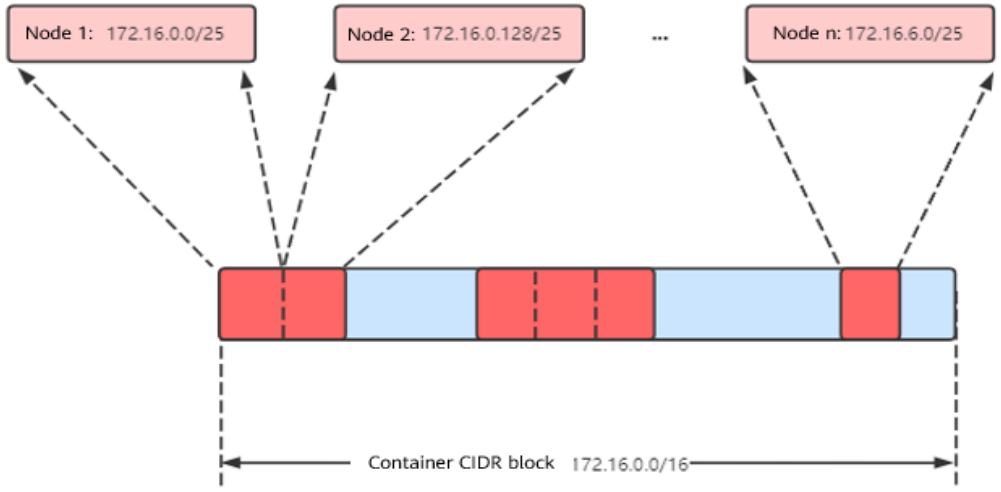
- Requisitos de alto desempenho: como nenhum encapsulamento de túnel é necessário, o modelo de rede da VPC oferece um desempenho próximo ao de uma rede da VPC quando comparado ao modelo de rede de túnel de contêiner. Portanto, o modelo de rede da VPC se aplica a cenários que possuem altos requisitos de desempenho, como computação de IA e computação de Big Data.
- Redes de pequena e média escala: devido à limitação nas tabelas de roteamento da VPC, é recomendável que o número de nós em um cluster seja menor ou igual a 1000.

Gerenciamento de endereços IP de contêiner

A rede da VPC aloca endereços IP de contêiner de acordo com as seguintes regras:

- O bloco CIDR de contêiner é alocado separadamente.
- Os endereços IP são alocados por nó. Um bloco CIDR com um tamanho fixo (que é configurável) é alocado para cada nó em um cluster a partir do bloco CIDR do contêiner.
- O bloco CIDR do contêiner aloca ciclicamente blocos CIDR para novos nós em sequência.
- Os pods agendados para um nó são endereços IP alocados ciclicamente de blocos CIDR alocados para o nó.

Figura 7-8 Gerenciamento de endereços IP da rede da VPC



Número máximo de nós que podem ser criados no cluster usando a rede VPC = número de endereços IP no bloco CIDR do contêiner /número de endereços IP no bloco CIDR alocados ao nó pelo bloco CIDR do contêiner

Por exemplo, se o bloco CIDR do contêiner for 172.16.0.0/16, o número de endereços IP será 65536. A máscara do bloco CIDR de contêiner alocado para o nó é 25. Ou seja, o número de endereços IP de contêiner em cada nó é 128. Portanto, um máximo de 512 (65536/128) nós podem ser criados. Além disso, o número de nós que podem ser criados em um cluster também depende da rede do nó e da escala do cluster.

Recomendação para planejamento de blocos CIDR

Conforme descrito em **Estrutura de rede do cluster**, os endereços de rede em um cluster podem ser divididos em três partes: rede de nó, rede de contêineres e rede de serviço. Ao planejar endereços de rede, considere os seguintes aspectos:

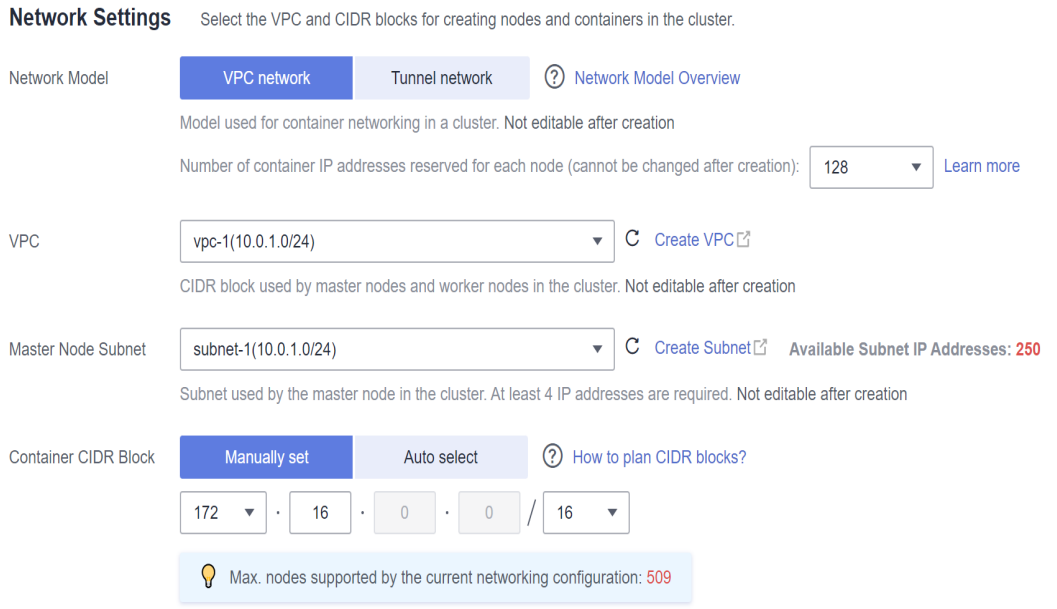
- Os três blocos CIDR não podem se sobrepor. Caso contrário, ocorre um conflito. Todas as sub-redes (incluindo aquelas criadas a partir do bloco CIDR secundário) na VPC em que o cluster reside não podem entrar em conflito com o contêiner e os blocos CIDR de serviço.
- Certifique-se de que cada bloco CIDR tenha endereços IP suficientes.
  - Os endereços IP no bloco CIDR do nó devem corresponder à escala do cluster. Caso contrário, os nós não podem ser criados devido a endereços IP insuficientes.
  - Os endereços IP no bloco CIDR do contêiner devem corresponder à escala de serviço. Caso contrário, os pods não podem ser criados devido a endereços IP insuficientes. O número de pods que podem ser criados em cada nó também depende de outras configurações de parâmetros. Para mais detalhes, consulte **Número máximo de pods que podem ser criados em um nó**.

Suponha que um cluster contenha 200 nós e o modelo de rede seja a rede da VPC.

Nesse caso, o número de endereços IP disponíveis na sub-rede de nó selecionada deve ser maior que 200. Caso contrário, os nós não podem ser criados devido a endereços IP insuficientes.

O bloco CIDR do contêiner é 10.0.0.0/16 e o número de endereços IP disponíveis é 65536. Conforme descrito em **Gerenciamento de endereços IP de contêiner**, a rede da VPC é alocada um bloco CIDR com o tamanho fixo (usando a máscara para determinar o número máximo de endereços IP de contêiner alocados para cada nó). Por exemplo, se o limite superior for 128, o cluster suportará um máximo de 512 (65536/128) nós, incluindo os três nós principais.

Figura 7-9 Configurar o bloco CIDR do contêiner (ao criar o cluster)



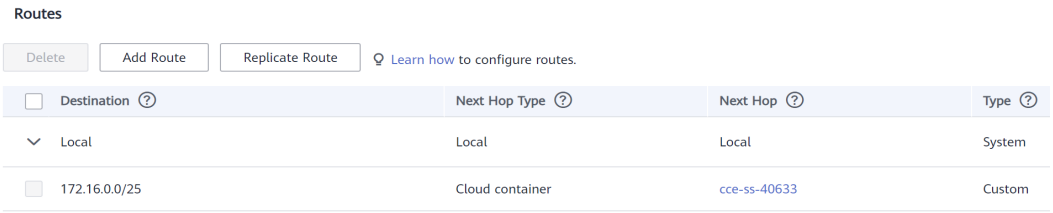
Exemplo de acesso à rede da VPC

Crie um cluster usando o modelo de rede da VPC. O cluster contém um nó.

```
$ kubectl get node
NAME            STATUS   ROLES    AGE   VERSION
192.168.0.99    Ready    <none>   9d    v1.17.17-r0-CCE21.6.1.B004-17.37.5
```

Verifique a tabela de roteamento da VPC. O endereço de destino 172.16.0.0/25 é o bloco CIDR de contêiner alocado para o nó, e o próximo salto é o nó correspondente. Quando o endereço IP do contêiner é acessado, a rota da VPC encaminha a solicitação de acesso para o nó de próximo salto. Isso indica que o modelo de rede da VPC usa rotas da VPC.

Figura 7-10 Rotas



Crie uma Implementação no cluster.

```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: example
  namespace: default
spec:
  replicas: 4
  selector:
    matchLabels:
      app: example
  template:
    metadata:
```



```
      labels:
        app: example
    spec:
      containers:
        - name: container-0
          image: 'nginx:perl'
      imagePullSecrets:
        - name: default-secret
```

Verifique o pod.

```
$ kubectl get pod -owide
NAME                       READY   STATUS    RESTARTS   AGE   IP
NODE           NOMINATED NODE   READINESS GATES
example-86b9779494-l8qrw   1/1     Running   0          14s   172.16.0.6
192.168.0.99   <none>           <none>
example-86b9779494-svs8t   1/1     Running   0          14s   172.16.0.7
192.168.0.99   <none>           <none>
example-86b9779494-x8kl5   1/1     Running   0          14s   172.16.0.5
192.168.0.99   <none>           <none>
example-86b9779494-zt627   1/1     Running   0          14s   172.16.0.8
192.168.0.99   <none>           <none>
```

Nesse caso, se você acessar o endereço IP do pod de um ECS (fora do cluster) na mesma VPC, o acesso será bem-sucedido. Esse é um recurso da rede da VPC. Os pods podem ser acessados diretamente de qualquer nó localizado fora do cluster e na mesma VPC dos pods usando os endereços IP dos pods.

Os pods podem ser acessados a partir de nós ou pods no mesmo cluster. No exemplo a seguir, você pode acessar diretamente os pods no contêiner.

```
$ kubectl exec -it example-86b9779494-l8qrw -- curl 172.16.0.7
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

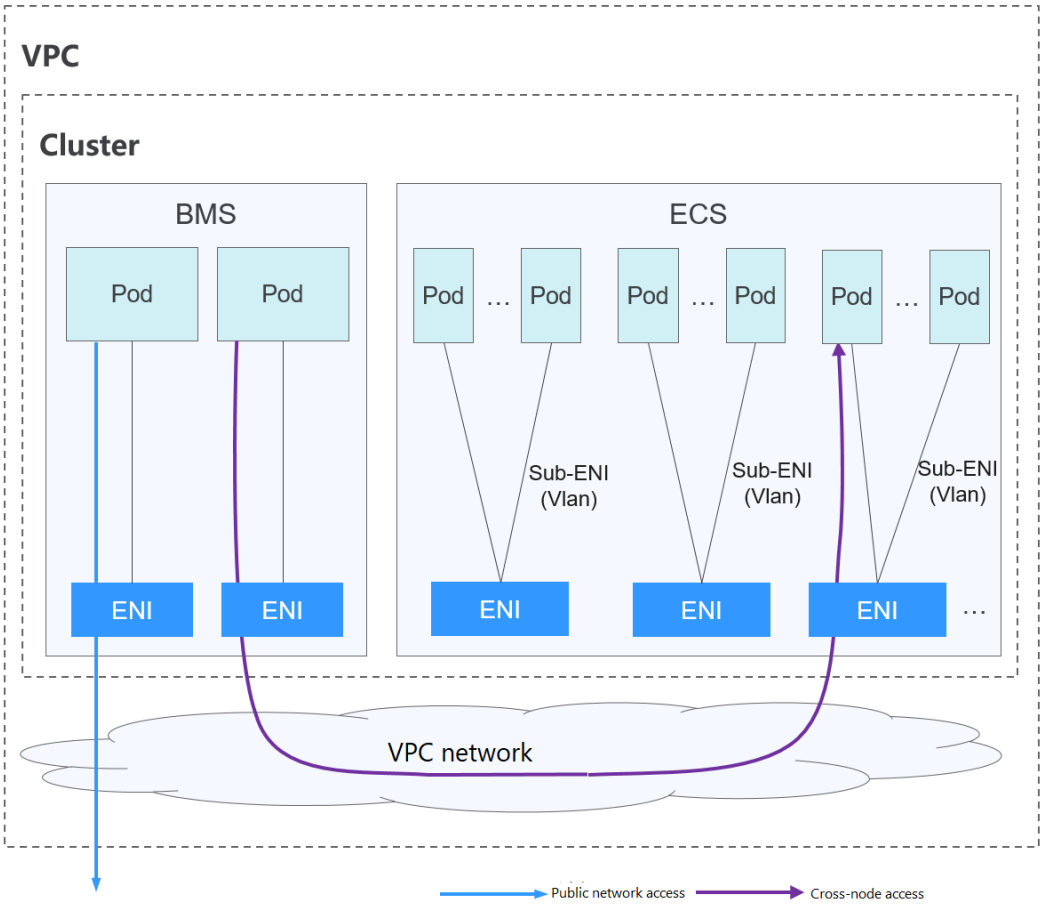
7.2.4 Cloud Native Network 2.0

Definição do modelo

Desenvolvido pelo CCE, o Cloud Native Network 2.0 integra profundamente Interfaces de rede elásticas (ENIs) e sub-ENIs da Virtual Private Cloud (VPC). Os endereços IP do contêiner são alocados a partir do bloco CIDR da VPC. A rede de passagem do ELB é

suportada para direcionar solicitações de acesso a contêineres. Grupos de segurança e IPs elásticos (EIPs) são obrigados a oferecer alto desempenho.

Figura 7-11 Cloud Native Network 2.0



Comunicação pod a pod

- Pods em nós do BMS usam ENIs, enquanto pods em nós do ECS usam sub-ENIs. Sub-ENIs são anexadas a ENIs através de sub-interfaces de VLAN.
- No mesmo nó: os pacotes são encaminhados por meio da ENI ou sub-ENI da VPC.
- Entre os nós: os pacotes são encaminhados por meio da ENI ou sub-ENI da VPC.

Observações e restrições

Este modelo de rede está disponível apenas para clusters do CCE Turbo.

Vantagens e desvantagens

Vantagens

- Como a rede de contêineres usa diretamente a VPC, é fácil localizar problemas de rede e fornecer o mais alto desempenho.
- As redes externas em numa VPC não podem ser conectadas diretamente aos endereços IP do contêiner.

- Os recursos de balanceamento de carga, grupo de segurança e EIP fornecidos pela VPC podem ser usados diretamente pelos pods.

Desvantagens

A rede de contêineres usa diretamente a VPC, que ocupa o espaço de endereço da VPC. Portanto, você deve planejar corretamente o bloco CIDR do contêiner antes de criar um cluster.

Cenários de aplicações

- Requisitos de alto desempenho e uso de outros recursos de rede da VPC: o Cloud Native Network 2.0 usa diretamente a VPC, que oferece quase o mesmo desempenho que a rede da VPC. Portanto, aplica-se a cenários com altos requisitos de largura de banda e latência, como transmissão ao vivo e venda em flash de comércio eletrônico.
- Rede em grande escala: o Cloud Native Network 2.0 é compatível com um máximo de 2.000 nós do ECS e 100.000 contêineres.

Gerenciamento de endereços IP de contêiner

No modelo Cloud Native Network 2.0, os nós do BMS usam ENIs e os nós do ECS usam sub-ENIs.

- O endereço IP do pod é alocado diretamente da sub-rede da VPC configurada para a rede de contêineres. Você não precisa alocar um pequeno segmento de rede independente ao nó.
- Para adicionar um nó do ECS a um cluster, vincule primeiro a ENI que transporta a sub-ENI. Depois que a ENI é vinculado, você pode vincular a sub-ENI.
- Número de ENIs vinculadas a um nó ECS: **número máximo de sub-ENIs que podem ser vinculadas ao node/64**. O valor é arredondado para cima.
- ENI vinculadas a um nó do ECS = **número de ENIs utilizadas para suportar sub-ENIs + número de sub-ENIs utilizadas atualmente por pods + número de sub-ENIs**
- ENIs vinculadas a um nó do BMS = **búmero de ENIs atualmente usadas por pods + número de ENIs pré-vinculadas**
- Quando um pod é criado, uma ENI disponível é alocada aleatoriamente do pool de ENI pré-vinculada do nó.
- Quando o pod é excluído, a ENI é liberada de volta para o pool de ENI do nó.
- Quando um nó é excluído, as ENIs são liberadas de volta para o pool e as sub-ENIs são excluídas.

O Cloud Native Network 2.0 é compatível com políticas de pré-vinculação da ENI **dinâmica** e **baseada em limites**. A tabela a seguir lista os cenários.

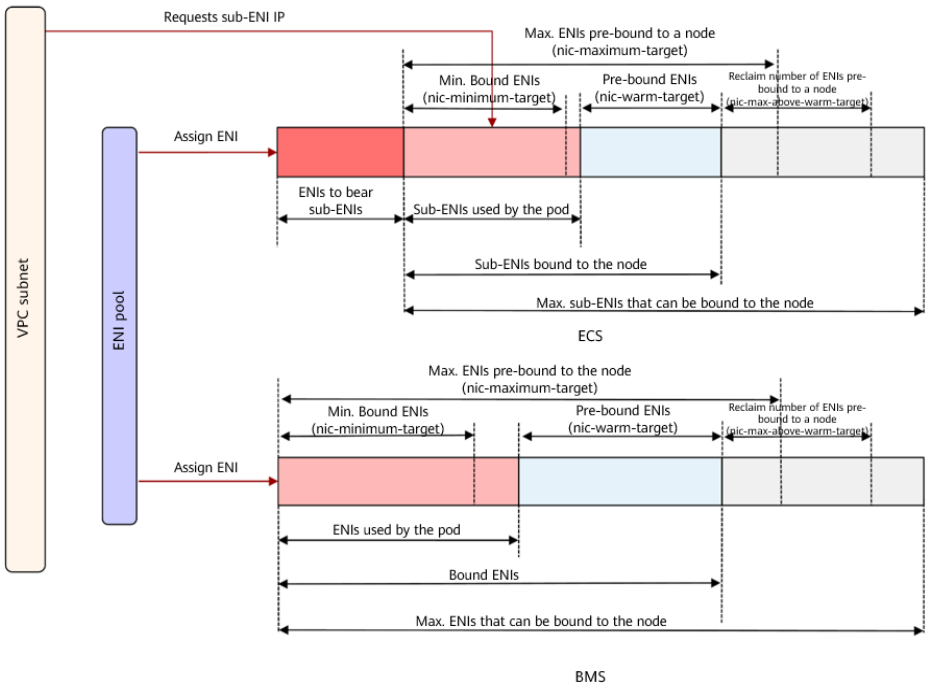
Tabela 7-2 Comparação entre as políticas de pré-vinculação da ENI

Política	Política de pré-vinculação da ENI dinâmica (padrão)	Política de pré-vinculação da ENI baseada em limites
Política de gerenciamento	**nic-minimum-target**: número mínimo de ENIs (não utilizadas+ utilizadas) vinculadas a um nó **nic-maximum-target**: se o número de ENIs vinculadas a um nó exceder o valor deste parâmetro, o sistema não faz a pré-vinculação proativa de ENIs. **Pre-bound ENIs**: ENIs extras que serão pré-vinculadas a um nó **nic-max-above-warm-target**: as ENI são desacopladas e recuperadas apenas quando o número de ENI ociosas menos o número de **nic-warm-target** for superior ao limite.	**Low threshold of the number of bound ENIs**: número mínimo de ENIs (não utilizadas) vinculadas a um nó **High threshold of the number of bound ENIs**: número máximo de ENIs que podem ser vinculadas a um nó. Se o número de ENIs vinculadas a um nó exceder o valor desse parâmetro, o sistema desvinculará as ENIs ociosas.
Cenário de aplicação	Acelera a inicialização do pod enquanto melhora a utilização de recursos IP. Esse modo se aplica a cenários em que o número de endereços IP no segmento de rede de contêiner é insuficiente. Para obter detalhes sobre os parâmetros anteriores, consulte **Pré-vinculação de ENIs aos clusters do CCE Turbo**.	Aplica-se a cenários em que o número de endereços IP no bloco CIDR do contêiner é suficiente e o número de pods nos nós muda drasticamente, mas é fixo em um determinado intervalo.

NOTA

- Para clusters de 1.19.16-r2, 1.21.5-r0, 1.23.3-r0 to 1.19.16-r4, 1.21.7-r0 e 1.23.5-r0, somente os parâmetros **nic-minimum-target** e **nic-warm-target** são suportados. A política de pré-vinculação baseada em limiares tem prioridade sobre a política de pré-vinculação dinâmica da ENI.
- Para conjuntos de 1.19.16-r4, 1.21.7-r0, 1.23.5-r0, 1.25.1-r0 ou posterior, os parâmetros anteriores são suportados. A política dinâmica de pré-vinculação da ENI tem prioridade sobre a política de pré-vinculação baseada em limites.

Figura 7-12 Política dinâmica de pré-vinculação da ENI



O CCE fornece quatro parâmetros para a política de pré-vinculação dinâmica da ENI. Defina esses parâmetros corretamente.

Tabela 7-3 Parâmetros da política dinâmica de pré-vinculação da ENI

Parâmetro	Valor padrão	Descrição	Sugestão
nic-minimum-target	10	Número mínimo de ENIs vinculadas a um nó. O valor pode ser um número ou uma porcentagem. ● Value: o valor deve ser um número inteiro positivo. Por exemplo, 10 indica que pelo menos 10 ENIs estão vinculadas a um nó. Se a cota de ENI de um nó for excedida, a cota de ENI será usada. ● Percentage: o valor varia de 1% a 100%. Por exemplo, 10%. Se a cota de ENI de um nó for 128, pelo menos 12 ENIs (arredondadas para baixo) serão vinculadas ao nó. Defina ambos **nic-minimum-target** e **nic-maximum-target** para o mesmo valor ou porcentagem.	Defina esses parâmetros com base no número de pods.

Parâmetro	Valor padrão	Descrição	Sugestão
nic-maximum-target	0	Se o número de ENIs vinculadas a um nó exceder o valor de **nic-maximum-target**, o sistema não fará pré-vinculação proativa das ENIs. Se o valor deste parâmetro for maior ou igual ao valor de **nic-minimum-target**, a verificação do número máximo de ENIs pré-vinculados é ativada. Caso contrário, a verificação é desativada. O valor pode ser um número ou uma porcentagem. ● Value: o valor deve ser um número inteiro positivo. Por exemplo, 0. A verificação do número máximo de ENIs pré-vinculadas está desativada. Se a cota de ENI de um nó for excedida, a cota de ENI será usada. ● Percentage: o valor varia de 1% a 100%. Por exemplo, 50%. Se a cota de ENI de um nó for 128, o número máximo da ENI pré-vinculada é 64 (arredondado para baixo). Defina ambos **nic-minimum-target** e **nic-maximum-target** para o mesmo valor ou porcentagem.	Defina esses parâmetros com base no número de pods.
nic-warm-target	2	As ENIs extras serão pré-vinculada após o **nic-minimum-target** ser usado em um pod. O valor só pode ser um número. Quando o valor de **nic-warm-target** + o número de ENIs vinculadas for superior ao valor de **nic-maximum-target**, o sistema pré-vinculará as ENIs com base na diferença entre o valor do **nic-maximum-target** e o número de ENIs vinculadas.	Defina este parâmetro para o número de pods que podem ser expandidos instantaneamente dentro de 10 segundos.

Parâmetro	Valor padrão	Descrição	Sugestão
nic-max-above-warm-target	2	Apenas quando o número de ENIs ociosas em um nó menos o valor de **nic-warm-target** for maior que o limiar, as ENIs pré-vinculadas serão desacopladas e recuperadas. O valor só pode ser um número. ● Definir um valor maior desse parâmetro desacelera a reciclagem de ENIs ociosas e acelera a inicialização do pod. No entanto, o uso do endereço IP diminui, especialmente quando os endereços IP são insuficientes. Portanto, **tenha cuidado ao aumentar o valor desse parâmetro**. ● Definir um valor menor desse parâmetro acelera a reciclagem de ENIs ociosas e melhora o uso do endereço IP. No entanto, quando um grande número de vagens aumenta instantaneamente, a inicialização de algumas vagens diminui.	Defina esse parâmetro com base na diferença entre o número de pods que são frequentemente escalonados na maioria dos nós em minutos e o número de pods que são escalonados instantaneamente na maioria dos nós em 10 segundos.

NOTA

Os parâmetros anteriores oferecem suporte à configuração global no nível do cluster e às configurações personalizadas no nível do pool de nós. Este último tem prioridade sobre o primeiro.

O componente de rede de contêineres mantém um pool de ENI pré-vinculada escalável para cada nó. O componente verifica e calcula o número de ENIs pré-vinculadas ou ENIs ociosas a cada 10 segundos.

- **Número de ENIs pré-vinculadas = min(nic-maximum-target - número de ENIs vinculadas, max(nic-minimum-target - número de ENIs vinculadas, nic-warm-target - número de ENIs ociosas)**
- **número de ENIs a serem vinculadas = min(número de ENIs ociosas - nic-warm-target-nic-max-above-warm-target, número de ENIs vinculadas - nic-minimum-target)**

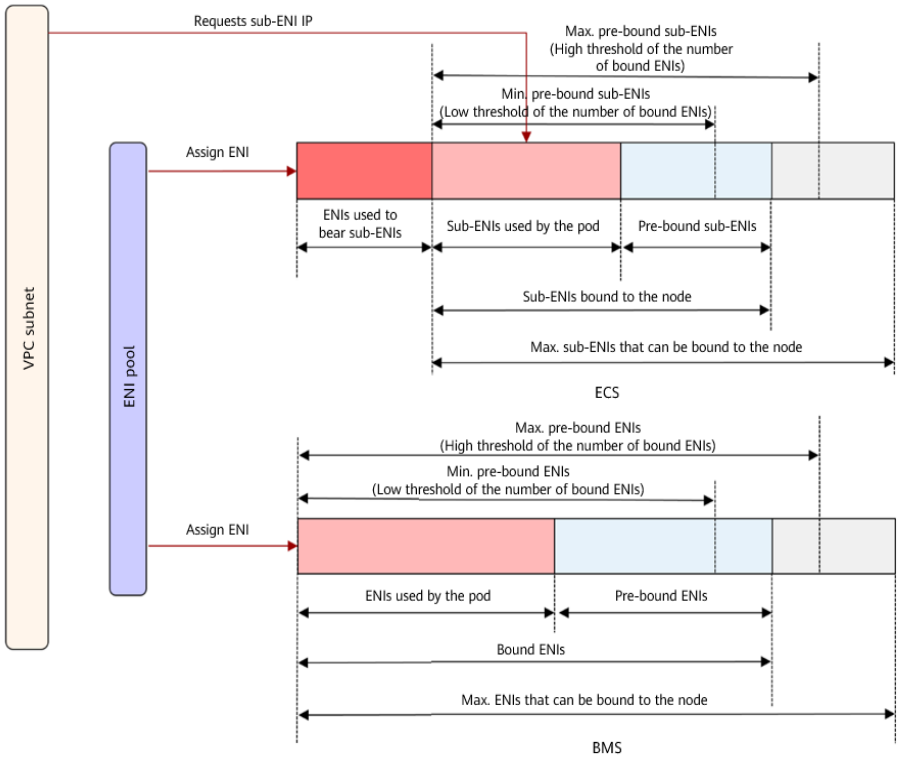
O número de ENIs de pré-vinculação no nó permanece no seguinte intervalo:

- **Número mínimo de ENIs a serem pré-vinculadas = min(max(nic-minimum-target - número de ENIs vinculadas, nic-warm-target), nic-maximum-target - número de ENIs vinculadas)**
- **Número máximo de ENIs a serem pré-vinculadas = max(nic-warm-target+ nic-max-above-warm-target, número de ENIs vinculadas - nic-minimum-target)**

Quando um pod é criado, uma ENI ociosa (a mais antiga não utilizada) é preferencialmente alocado do pool. Se nenhuma ENI ociosa estiver disponível, uma newsub-ENI será vinculada ao pod.

Quando o pod é excluído, a ENI correspondente é liberada de volta ao pool de ENI vinculada do nó, entra em um período de resfriamento de 2 minutos e pode ser vinculada a outro pod. Se a ENI não estiver vinculada a nenhum pod dentro de 2 minutos, ela será liberada.

Figura 7-13 Política baseada em limites



O CCE fornece um parâmetro de configuração para os algoritmos de limite. Você pode definir esse parâmetro com base no plano de serviço, na escala do cluster e no número de ENIs que podem ser vinculadas a um nó.

- **Limite baixo do número de ENIs vinculadas**: o padrão é **0**, indicando o número mínimo de ENIs (não utilizadas) vinculadas a um nó. Número mínimo de ENIs pré-vinculadas num nó ECS = número de ENIs vinculadas ao nó no limite baixo x número de sub-ENIs no nó. Número mínimo de ENIs pré-vinculadas num nó do BMS = número de ENIs vinculadas ao nó no limite baixo x número de ENIs no nó.
- **Limite elevado do número de ENIs vinculadas**: o padrão é **0**, indicando o número máximo de ENIs que podem ser vinculadas a um nó. Se o número de ENIs vinculadas a um nó exceder o valor desse parâmetro, o sistema desvinculará as ENIs ociosas. Número máximo de ENIs vinculadas num nó do ECS = número de ENIs vinculada no limite elevado x número de sub-ENIs no nó. Número máximo de ENIs pré-vinculadas num nó do BMS = número de ENIs vinculadas no limite elevado x número de ENIs no nó.

O componente de rede de contêineres mantém um pool de ENI escalável para cada nó.

- Se **o número de ENIs vinculadas (ENIs pré-vinculadas usadas) for menor que o número de ENIs pré-vinculadas no limite baixo**, as ENIs serão vinculadas até que **os dois números sejam iguais**.
- Se **o número de ENIs vinculadas (ENIs pré-vinculadas utilizadas) for superior ao número de ENIs pré-vinculadas no limite elevado** e **o número de ENIs pré-vinculadas for superior a 0**, as ENI pré-vinculadas que não são utilizadas durante mais de 2 minutos serão libertas periodicamente até que **o número de ENIs vinculadas = número de ENIs pré-vinculadas no limite elevado** ou **o número de ENIs utilizadas seja superior ao número de ENIs pré-vinculadas no limite elevado** e **o número de ENIs pré-vinculadas no nó é 0**.

Recomendação para planejamento de blocos CIDR

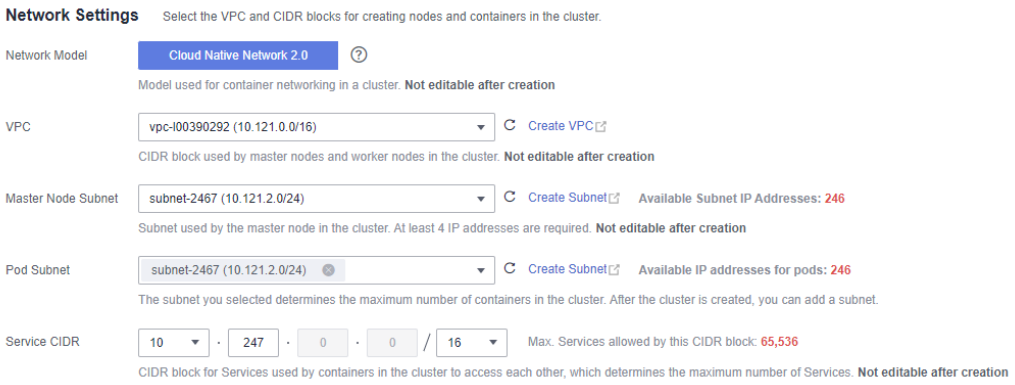
Conforme descrito em **Estrutura de rede do cluster**, os endereços de rede em um cluster podem ser divididos em três partes: rede de nó, rede de contêineres e rede de serviço. Ao planejar endereços de rede, considere os seguintes aspectos:

- Os três blocos CIDR não podem se sobrepor. Caso contrário, ocorre um conflito. Todas as sub-redes (incluindo aquelas criadas a partir do bloco CIDR secundário) na VPC em que o cluster reside não podem entrar em conflito com o contêiner e os blocos CIDR de Serviço.
- Certifique-se de que cada bloco CIDR tenha endereços IP suficientes.
  - Os endereços IP no bloco CIDR do nó devem corresponder à escala do cluster. Caso contrário, os nós não podem ser criados devido a endereços IP insuficientes.
  - Os endereços IP no bloco CIDR do contêiner devem corresponder à escala de serviço. Caso contrário, os pods não podem ser criados devido a endereços IP insuficientes.

No modelo Cloud Native Network 2.0, o bloco CIDR do contêiner e o bloco CIDR do nó compartilham os endereços de rede em uma VPC. Recomenda-se que a sub-rede do contêiner e a sub-rede do nó não usem a mesma sub-rede. Caso contrário, contêineres ou nós podem não ser criados devido a recursos IP insuficientes.

Além disso, uma sub-rede pode ser adicionada ao bloco CIDR do contêiner depois que um cluster é criado para aumentar o número de endereços IP disponíveis. Nesse caso, certifique-se de que a sub-rede adicionada não entra em conflito com outras sub-redes no bloco CIDR do contêiner.

Figura 7-14 Configurar os blocos CIDR (ao criar o cluster)

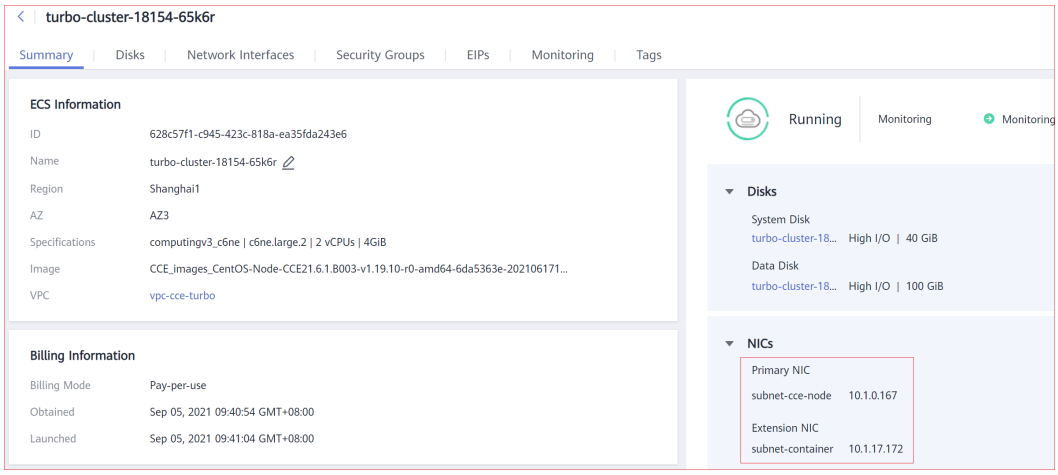


Exemplo de acesso ao Cloud Native Network 2.0

Crie um cluster do CCE Turbo, que contenha três nós do ECS.

Acesse a página de detalhes de um nó. Você pode ver que o nó tem uma ENI primária e uma ENI estendida, e ambas são ENIs. A ENI estendida pertence ao bloco CIDR do contêiner e é usado para montar uma sub-ENI no pod.

Figura 7-15 ENIs de nó



Crie uma Implementação no cluster.

```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: example
  namespace: default
spec:
  replicas: 6
  selector:
    matchLabels:
      app: example
  template:
    metadata:
      labels:
        app: example
    spec:
      containers:
        - name: container-0
          image: 'nginx:perl'
          resources:
            limits:
              cpu: 250m
              memory: 512Mi
            requests:
              cpu: 250m
              memory: 512Mi
      imagePullSecrets:
        - name: default-secret
```

Visualize o pod criado.

```
$ kubectl get pod -owide
NAME                       READY   STATUS    RESTARTS   AGE   IP
NODE          NOMINATED NODE   READINESS GATES
example-5bdc5699b7-54v7g   1/1     Running   0          7s    10.1.18.2
10.1.0.167    <none>           <none>
example-5bdc5699b7-6dzx5   1/1     Running   0          7s    10.1.18.216
10.1.0.186    <none>           <none>
example-5bdc5699b7-gq7xs   1/1     Running   0          7s    10.1.16.63
10.1.0.144    <none>           <none>
example-5bdc5699b7-h9rvb   1/1     Running   0          7s    10.1.16.125
10.1.0.167    <none>           <none>
example-5bdc5699b7-s9fts   1/1     Running   0          7s    10.1.16.89
10.1.0.144    <none>           <none>
example-5bdc5699b7-swq6q   1/1     Running   0          7s    10.1.17.111
10.1.0.167    <none>           <none>
```


Os endereços IP de todos os pods são sub-ENIs, que são montados na ENI (ENI estendida) do nó.

Por exemplo, a ENI estendida do nó 10.1.0.167 é 10.1.17.172. Na página **Network Interfaces** do Console de rede, você pode ver que três sub-ENIs estão montadas na ENI estendida 10.1.17.172, que é o endereço IP do pod.

Figura 7-16 ENIs de pod

Na VPC, o endereço IP do pod pode ser acessado com êxito.

7.3 Serviço

7.3.1 Visão geral

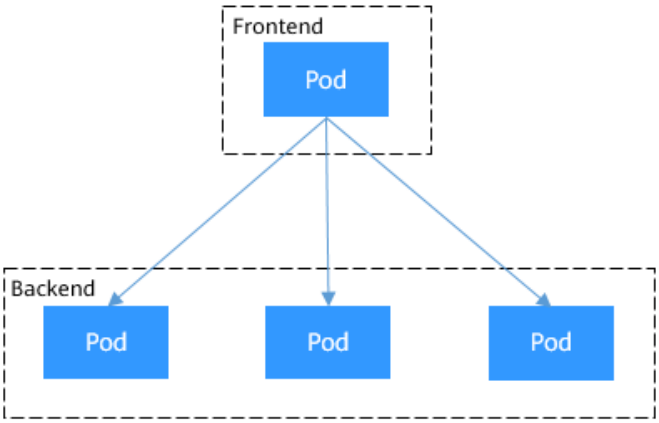
Acesso direto a um pod

Depois que um pod é criado, os seguintes problemas podem ocorrer se você acessar diretamente o pod:

- O pod pode ser excluído e recriado a qualquer momento por um controlador, como um Deployment, e o resultado do acesso ao pod se torna imprevisível.
- O endereço IP do pod é alocado somente depois que o pod é iniciado. Antes de o pod ser iniciado, o endereço IP do pod é desconhecido.
- Uma aplicação é geralmente composta de vários pods que executam a mesma imagem. Acessar pods um por um não é eficiente.

Por exemplo, uma aplicação usa Deployments para criar o front-end e o back-end. O front-end chama o back-end para computação, conforme mostrado em **Figura 7-17**. Três pods estão sendo executados no back-end, que são independentes e substituíveis. Quando um pod de back-end é recriado, o novo pod é atribuído com um novo endereço IP, do qual o pod de front-end não tem conhecimento.

Figura 7-17 Acesso entre pods

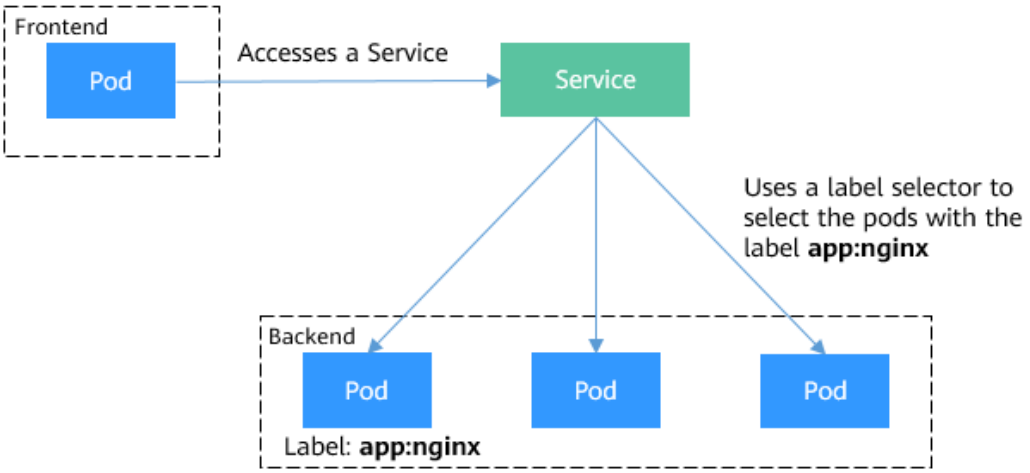


Usar Serviços para acesso a pods

Serviços do Kubernetes são usados para resolver os problemas anteriores de acesso ao pod. Um Serviço tem um endereço IP fixo. (Quando um cluster do CCE é criado, um bloco CIDR de Serviço é definido, que é usado para alocar endereços IP aos Serviços.) Um Serviço encaminha solicitações de acesso ao Serviço para pods com base em rótulos e, ao mesmo tempo, executa o balanceamento de carga para esses pods.

No exemplo anterior, um Serviço é adicionado para o pod de front-end acessar os pods de back-end. Dessa forma, o pod de front-end não precisa estar ciente das alterações nos pods de back-end, como mostrado em **Figura 7-18**.

Figura 7-18 Acessar pods por meio de um serviço



Tipos de Serviço

O Kubernetes permite que você especifique um Serviço de um tipo obrigatório. Os valores e ações dos diferentes tipos de Serviços são os seguintes:

- **ClusterIP**

  Um Serviço ClusterIP permite que cargas de trabalho no mesmo cluster usem seus nomes de domínio internos do cluster para acessar uns aos outros.

- **NodePort**

  Um Serviço é exposto no endereço IP de cada nó em uma porta estática (NodePort). Um Serviço ClusterIP, para o qual o Serviço NodePort será roteado, é criado automaticamente. Solicitando <NodeIP>:<NodePort>, você pode acessar um Serviço NodePort de fora do cluster.

- **LoadBalancer**

  Os Serviços LoadBalancer podem acessar cargas de trabalho da rede pública por meio de um balanceador de carga, que é mais confiável do que o acesso baseado em EIP. Os Serviços LoadBalancer são recomendados para acessar cargas de trabalho de fora do cluster.

- **DNAT**

  Um gateway DNAT traduz endereços para nós de cluster e permite que vários nós de cluster compartilhem um EIP. Os Serviços NodePort fornecem maior confiabilidade do que os Serviços DNAT baseados em EIP. Você não precisa vincular um EIP a um único nó e as solicitações ainda podem ser distribuídas à carga de trabalho, mesmo que qualquer um dos nós internos esteja desativado.

externalTrafficPolicy (afinidade de serviço)

Para um Serviço NodePort e LoadBalancer, as solicitações são enviadas primeiro para a porta do nó, depois para o Serviço e, finalmente, para o pod que suporta o Serviço. O pod de apoio pode não estar localizado no nó que recebe as solicitações. Por padrão, a carga de trabalho de back-end pode ser acessada a partir de qualquer endereço IP de nó e porta de serviço. Se o pod não estiver no nó que recebe a solicitação, a solicitação será redirecionada para o nó onde o pod está localizado, o que pode causar perda de desempenho.

externalTrafficPolicy é um parâmetro de configuração do Serviço.

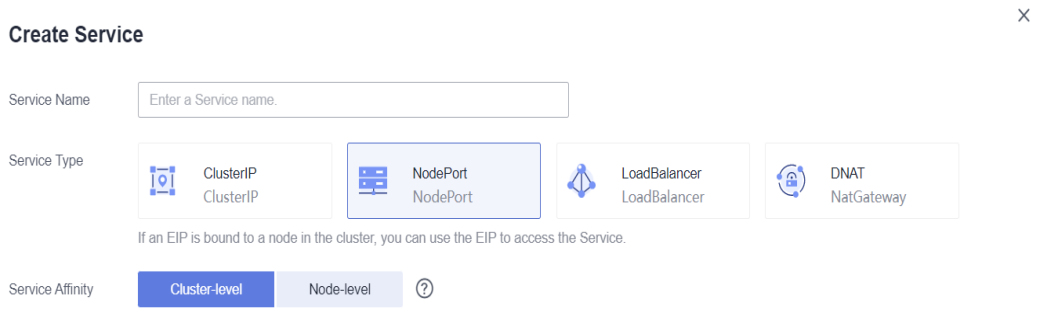
```
apiVersion: v1
kind: Service
metadata:
  name: nginx-nodeport
spec:
  externalTrafficPolicy: Local
  ports:
  - name: service
    nodePort: 30000
    port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: nginx
  type: NodePort
```

Se o valor de **externalTrafficPolicy** for **Local**, as solicitações enviadas de *Node IP address:Service port* serão encaminhadas apenas para o pod no nó local. Se o nó não tiver um pod, as solicitações serão suspensas.

Se o valor de **externalTrafficPolicy** for **Cluster**, as solicitações serão encaminhadas dentro do cluster e a carga de trabalho de back-end poderá ser acessada a partir de qualquer endereço IP de nó e porta de serviço.

Se **externalTrafficPolicy** não estiver definido, o valor padrão **Cluster** será usado.

Você pode definir esse parâmetro ao criar um Serviço do tipo NodePort no console do CCE.



Os valores de **externalTrafficPolicy** são os seguintes:

- **Cluster**: os endereços IP e as portas de acesso de todos os nós em um cluster podem acessar a carga de trabalho associada ao Serviço. O acesso ao Serviço causará perda de desempenho devido ao redirecionamento de rota e o endereço IP de origem do cliente não pode ser obtido.
- **Local**: somente o endereço IP e a porta de acesso do nó onde a carga de trabalho está localizada podem acessar a carga de trabalho associada ao Serviço. O acesso ao Serviço não causará perda de desempenho devido ao redirecionamento de rota e o endereço IP de origem do cliente pode ser obtido. Neste cenário, os Serviços poderão não ser acedidos a partir do cluster. Para mais detalhes, consulte **Por que um Serviço falha ao ser acessado de dentro do cluster**.

Por que um Serviço falha ao ser acessado de dentro do cluster

Se a afinidade de serviço de um Serviço for definida para o nível do nó, ou seja, o valor de **externalTrafficPolicy** for **Local**, o Serviço pode não ser acessado de dentro do cluster (especificamente, nós ou contêineres). Informação semelhante à seguinte é exibida:

```
upstream connect error or disconnect/reset before headers. reset reason:
connection failure
Or
curl: (7) Failed to connect to 192.168.10.36 port 900: Connection refused
```

É comum que um balanceador de carga em um cluster não possa ser acessado. A razão é a seguinte: quando o Kubernetes cria um erviço, kube-proxy adiciona o endereço de acesso do balanceador de carga como um endereço IP externo (IP externo, como mostrado na saída do comando a seguir) ao iptables ou ao IPVS. Se um cliente dentro do cluster iniciar uma solicitação para acessar o balanceador de carga, o endereço será considerado como o endereço IP externo do Serviço e a solicitação será encaminhada diretamente pelo kube-proxy sem passar pelo balanceador de carga fora do cluster.

```
# kubectl get svc nginx
NAME    TYPE           CLUSTER-IP      EXTERNAL-IP
PORT(S)        AGE
nginx   LoadBalancer   10.247.76.156   123.**.**.**,192.168.0.133
80:32146/TCP   37s
```

Quando o valor de **externalTrafficPolicy** é **Local**, as falhas de acesso em diferentes modelos de rede de contêiner e modos de encaminhamento de serviço são as seguintes:

NOTA

- Para uma carga de trabalho com vários pods, certifique-se de que todos os pods estejam acessíveis. Caso contrário, existe a possibilidade de que o acesso à carga de trabalho falhe.
- Os clusters do CCE Turbo que usam a rede do Cloud Native 2.0 não oferecem suporte à afinidade de serviço no nível do nó.

Tipo de Serviço liberado no servidor	Tipo de acesso	Local de início da solicitação no cliente	Cluster de rede de túnel (IPVS)	Cluster de rede da VPC (IPVS)	Cluster de rede de túnel (iptables)	Cluster de rede da VPC (iptables)
Serviço NodePort	Rede pública / privada	Mesmo nó que o pod de serviço	Acesse o endereço IP e o NodePort no nó onde o servidor está localizado: o acesso é bem sucedido. Acesse o endereço IP e o NodePort em um nó diferente do nó onde o servidor está localizado: o acesso falhou.	Acesse o endereço IP e o NodePort no nó onde o servidor está localizado: o acesso é bem sucedido. Acesse o endereço IP e o NodePort em um nó diferente do nó onde o servidor está localizado: o acesso falhou.	Acesse o endereço IP e o NodePort no nó onde o servidor está localizado: o acesso é bem sucedido. Acesse o endereço IP e o NodePort em um nó diferente do nó onde o servidor está localizado: o acesso falhou.	Acesse o endereço IP e o NodePort no nó onde o servidor está localizado: o acesso é bem sucedido. Acesse o endereço IP e o NodePort em um nó diferente do nó onde o servidor está localizado: o acesso falhou.
		Diferentes nós do pod de serviço	Acesse o endereço IP e o NodePort no nó onde o servidor está localizado: o acesso é bem sucedido. Acesse o endereço IP e o NodePort em um nó diferente do nó onde o servidor está localizado: o acesso falhou.	Acesse o endereço IP e o NodePort no nó onde o servidor está localizado: o acesso é bem sucedido. Acesse o endereço IP e o NodePort em um nó diferente do nó onde o servidor está localizado: o acesso falhou.	O acesso é bem sucedido.	O acesso é bem sucedido.

Tipo de Serviço liberado no servidor	Tipo de acesso	Local de início da solicitação no cliente	Cluster de rede de túnel (IPVS)	Cluster de rede da VPC (IPVS)	Cluster de rede de túnel (iptables)	Cluster de rede da VPC (iptables)
		Outros contêineres no mesmo nó do pod de serviço	Acesse o endereço IP e o NodePort no nó onde o servidor está localizado: o acesso é bem sucedido. Acesse o endereço IP e o NodePort em um nó diferente do nó onde o servidor está localizado: o acesso é bem sucedido.	O acesso falhou.	Acesse o endereço IP e o NodePort no nó onde o servidor está localizado: o acesso é bem sucedido. Acesse o endereço IP e o NodePort em um nó diferente do nó onde o servidor está localizado: o acesso falhou.	O acesso falhou.
		Outros contêineres em diferentes nós do pod de serviço	Acesse o endereço IP e o NodePort no nó onde o servidor está localizado: o acesso é bem sucedido. Acesse o endereço IP e o NodePort em um nó diferente do nó onde o servidor está localizado: o acesso falhou.	Acesse o endereço IP e o NodePort no nó onde o servidor está localizado: o acesso é bem sucedido. Acesse o endereço IP e o NodePort em um nó diferente do nó onde o servidor está localizado: o acesso falhou.	Acesse o endereço IP e o NodePort no nó onde o servidor está localizado: o acesso é bem sucedido. Acesse o endereço IP e o NodePort em um nó diferente do nó onde o servidor está localizado: o acesso falhou.	Acesse o endereço IP e o NodePort no nó onde o servidor está localizado: o acesso é bem sucedido. Acesse o endereço IP e o NodePort em um nó diferente do nó onde o servidor está localizado: o acesso falhou.

Tipo de Serviço liberado no servidor	Tipo de acesso	Local de início da solicitação no cliente	Cluster de rede de túnel (IPVS)	Cluster de rede da VPC (IPVS)	Cluster de rede de túnel (iptables)	Cluster de rede da VPC (iptables)
Serviço LoadBalancer usando um balanceador de carga dedicado	Rede privada	Mesmo nó que o pod de serviço	O acesso falhou.	O acesso falhou.	O acesso falhou.	O acesso falhou.
		Outros contêineres no mesmo nó do pod de serviço	O acesso falhou.	O acesso falhou.	O acesso falhou.	O acesso falhou.
Serviço de gateway de DNAT	Rede pública	Mesmo nó que o pod de serviço	O acesso falhou.	O acesso falhou.	O acesso falhou.	O acesso falhou.
		Diferentes nós do pod de serviço	O acesso falhou.	O acesso falhou.	O acesso falhou.	O acesso falhou.
		Outros contêineres no mesmo nó do pod de serviço	O acesso falhou.	O acesso falhou.	O acesso falhou.	O acesso falhou.
		Outros contêineres em diferentes nós do pod de serviço	O acesso falhou.	O acesso falhou.	O acesso falhou.	O acesso falhou.

Tipo de Serviço liberado no servidor	Tipo de acesso	Local de início da solicitação no cliente	Cluster de rede de túnel (IPVS)	Cluster de rede da VPC (IPVS)	Cluster de rede de túnel (iptables)	Cluster de rede da VPC (iptables)
Complemento nginx-ingress conectado a um balanceador de carga dedicado (**Local**)	Rede privada	Mesmo nó que pod de cceaddon-nginx-ingress-controller	O acesso falhou.	O acesso falhou.	O acesso falhou.	O acesso falhou.
		Outros contêineres no mesmo nó do pod de cceaddon-nginx-ingress-controller	O acesso falhou.	O acesso falhou.	O acesso falhou.	O acesso falhou.

Os seguintes métodos podem ser usados para resolver este problema:

- (**Recomendado**) No cluster, use o Serviço ClusterIP ou o nome de domínio do serviço para acesso.
- Defina **externalTrafficPolicy** do Serviço como **Cluster**, o que significa afinidade de serviço no nível do cluster. Observe que isso afeta a persistência do endereço de origem.

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    kubernetes.io/elb.class: union
    kubernetes.io/elb.autocreate: '{"type":"public","bandwidth_name":"cce-
bandwidth","bandwidth_chargemode":"bandwidth","bandwidth_size":5,"bandwidth_sh
aretype":"PER","eip_type":"5_bgp","name":"james"}'
  labels:
    app: nginx
  name: nginx
spec:
  externalTrafficPolicy: Cluster
  ports:
  - name: service0
    port: 80
    protocol: TCP
    targetPort: 80
```



```
  selector:
    app: nginx
  type: LoadBalancer
```

- Aproveitar o recurso de passagem do Serviço, o kube-proxy é ignorado quando o endereço do ELB é usado para acesso. O balanceador de carga do ELB é acessado primeiro e, em seguida, a carga de trabalho. Para mais detalhes, consulte **Configuração de rede de passagem para um Serviço de LoadBalancer**.

  NOTA

  - Em um cluster padrão do CCE, depois que a rede de passagem é configurada para um balanceador de carga dedicado, o endereço IP privado do balanceador de carga não pode ser acessado do nó onde o pod de carga de trabalho reside ou de outros contêineres no mesmo nó da carga de trabalho.
  - A rede de passagem não é suportada para clusters de v1.15 ou anterior.
  - No modo de rede IPVS, as configurações de passagem dos Serviços conectados ao mesmo balanceador de carga devem ser as mesmas.
  - Se a afinidade de serviço em nível de nó (local) for usada, **kubernetes.io/elb.pass-through** será automaticamente definido como **onlyLocal** para ativar a passagem.

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    kubernetes.io/elb.pass-through: "true"
    kubernetes.io/elb.class: union
    kubernetes.io/elb.autocreate: '{"type":"public","bandwidth_name":"cce-
bandwidth","bandwidth_chargemode":"bandwidth","bandwidth_size":5,"bandwidth_sh
aretype":"PER","eip_type":"5_bgp","name":"james"}'
  labels:
    app: nginx
  name: nginx
spec:
  externalTrafficPolicy: Local
  ports:
  - name: service0
    port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: nginx
  type: LoadBalancer
```

7.3.2 ClusterIP

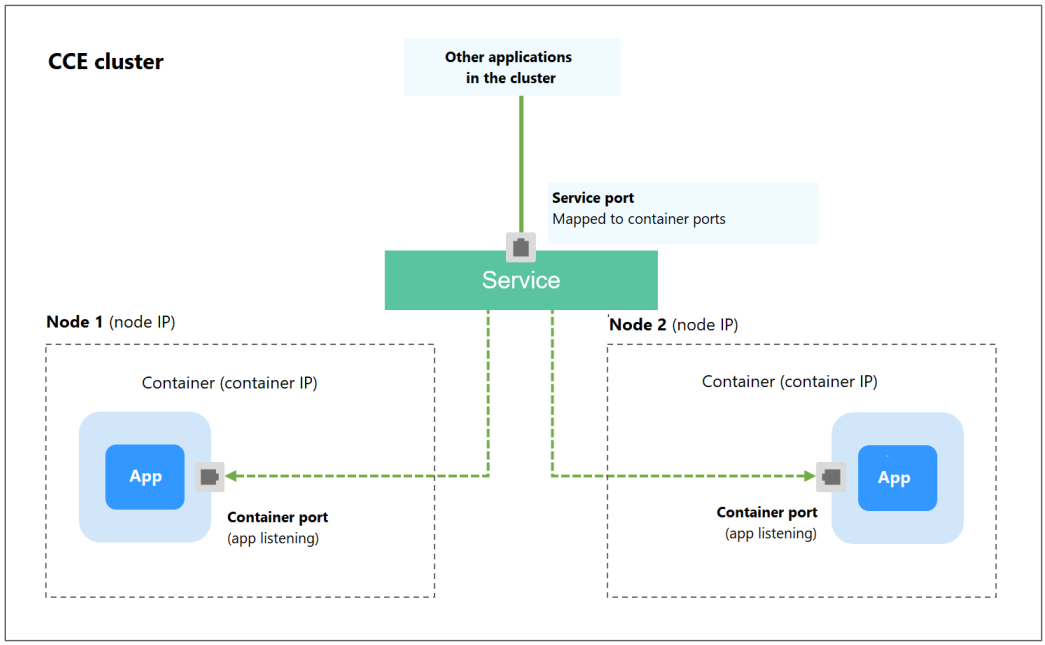
Cenário

Os Serviços ClusterIP permitem que cargas de trabalho no mesmo cluster usem seus nomes de domínio internos do cluster para acessar uns aos outros.

O formato de nome de domínio interno do cluster é *<Service name>.<Namespace of the workload>***.svc.cluster.local:***<Port>*, por exemplo, **nginx.default.svc.cluster.local:80**.

Figura 7-19 mostra os relacionamentos de mapeamento entre canais de acesso, portas de contêiner e portas de acesso.

Figura 7-19 Acesso dentro do cluster (ClusterIP)



Criar um Serviço ClusterIP

Passo 1 Efetue logon no console do CCE e acesse o console do cluster.

Passo 2 Escolha **Networking** no painel de navegação e clique em **Create Service** no canto superior direito.

Passo 3 Defina os parâmetros de acesso dentro do cluster.

- **Service Name**: nome do serviço, que pode ser o mesmo que o nome da carga de trabalho.
- **Service Type**: selecione **ClusterIP**.
- **Namespace**: namespace ao qual a carga de trabalho pertence.
- **Selector**: adicione um rótulo e clique em **Confirm**. Um Serviço seleciona um pod com base no rótulo adicionado. Você também pode clicar em **Reference Workload Label** para fazer referência ao rótulo de uma carga de trabalho existente. Na caixa de diálogo exibida, selecione uma carga de trabalho e clique em **OK**.
- **IPv6**: esta função está desativada por padrão. Depois que essa função é ativada, o endereço IP do cluster do Serviço muda para um endereço IPv6. Para obter detalhes, consulte **Criação de um cluster IPv4/IPv6 de pilha dupla no CCE**. **Este parâmetro está disponível apenas em clusters de v1.15 ou posterior com IPv6 habilitado (definido durante a criação do cluster).**
- **Port Settings**
  - **Protocol**: protocolo utilizado pelo Serviço.
  - **Service Port**: porta utilizada pelo Serviço. O número da porta varia de 1 a 65535.
  - **Container Port**: porta na qual a carga de trabalho escuta. Por exemplo, o Nginx usa a porta 80 por padrão.

Passo 4 Clique em **OK**.

----Fim

Configurar o tipo de acesso usando o kubectl

Você pode executar comandos kubectl para definir o tipo de acesso (Serviço). Esta seção usa uma carga de trabalho do Nginx como exemplo para descrever como implementar o acesso dentro do cluster usando kubectl.

Passo 1 Use o kubectl para se conectar ao cluster. Para mais detalhes, consulte **Conexão a um cluster usando o kubectl**.

Passo 2 Crie e edite os arquivos **nginx-deployment.yaml** e **nginx-clusterip-svc.yaml**.

Os nomes dos arquivos são definidos pelo usuário. **nginx-deployment.yaml** e **nginx-clusterip-svc.yaml** são apenas nomes de arquivo de exemplo.

vi nginx-deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - image: nginx:latest
        name: nginx
      imagePullSecrets:
      - name: default-secret
```

vi nginx-clusterip-svc.yaml

```
apiVersion: v1
kind: Service
metadata:
  labels:
    app: nginx
  name: nginx-clusterip
spec:
  ports:
  - name: service0
    port: 8080                # Port for accessing a Service.
    protocol: TCP             # Protocol used for accessing a Service. The value
can be TCP or UDP.
    targetPort: 80            # Port used by a Service to access the target
container. This port is closely related to the applications running in a
container. In this example, the Nginx image uses port 80 by default.
  selector:                   # Label selector. A Service selects a pod based on
the label and forwards the requests for accessing the Service to the pod. In this
example, select the pod with the app:nginx label.
    app: nginx
  type: ClusterIP             # Type of a Service. ClusterIP indicates that a
Service is only reachable from within the cluster.
```

Passo 3 Crie uma carga de trabalho.

kubectl create -f nginx-deployment.yaml

Se forem exibidas informações semelhantes às seguintes, a carga de trabalho foi criada.

```
deployment "nginx" created
```

kubectl get po

Se informações semelhantes às seguintes forem exibidas, a carga de trabalho está em execução.

```
NAME                       READY     STATUS              RESTARTS   AGE
nginx-2601814895-znhbr     1/1       Running             0          15s
```

Passo 4 Crie um Serviço.

kubectl create -f nginx-clusterip-svc.yaml

Se forem exibidas informações semelhantes às seguintes, o Serviço foi criado.

```
service "nginx-clusterip" created
```

kubectl get svc

Se informações semelhantes às seguintes forem exibidas, o Serviço foi criado e um endereço IP interno do cluster foi atribuído ao Serviço.

```
# kubectl get svc
NAME              TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)    AGE
kubernetes        ClusterIP   10.247.0.1     <none>        443/TCP    4d6h
nginx-clusterip   ClusterIP   10.247.74.52   <none>        8080/TCP   14m
```

Passo 5 Acesse um serviço.

Um Serviço pode ser acessado a partir de contêineres ou nós em um cluster.

Crie um pod, acesse o pod e execute o comando **curl** para acessar *IP address:Port* ou o nome de domínio do Serviço, conforme mostrado na figura a seguir.

O sufixo do nome de domínio pode ser omitido. No mesmo namespace, você pode usar diretamente **nginx-clusterip:8080** para acesso. Em outros namespaces, você pode usar **nginx-clusterip.default:8080** para acesso.

```
# kubectl run -i --tty --image nginx:alpine test --rm /bin/sh
If you do not see a command prompt, try pressing Enter.
/ # curl 10.247.74.52:8080
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
/ # curl nginx-clusterip.default.svc.cluster.local:8080
...
<h1>Welcome to nginx!</h1>
...
```



```
/ # curl nginx-clusterip.default:8080
...
<h1>Welcome to nginx!</h1>
...
/ # curl nginx-clusterip:8080
...
<h1>Welcome to nginx!</h1>
...
```

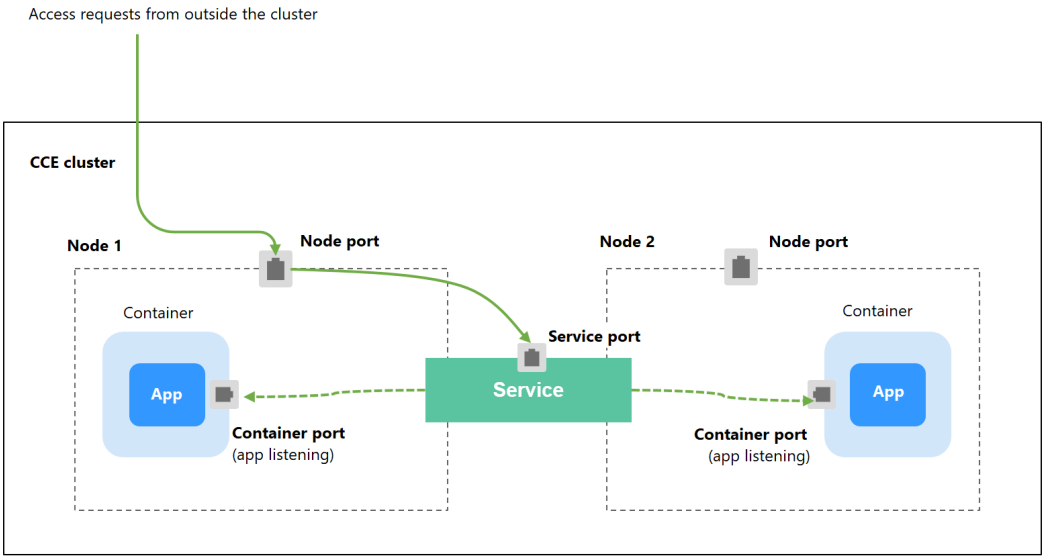
----Fim

7.3.3 NodePort

Cenário

Um Serviço é exposto no endereço IP de cada nó em uma porta estática (NodePort). Quando você cria um Serviço NodePort, o Kubernetes aloca automaticamente um endereço IP interno (ClusterIP) do cluster. Quando os clientes fora do cluster acessarem <NodeIP>:<NodePort>, o tráfego será encaminhado para o pod de destino por meio do ClusterIP do Serviço NodePort.

Figura 7-20 Acesso a NodePort



Restrições

- Por padrão, um Serviço NodePort é acessado dentro de uma VPC. Para usar um EIP para acessar um Serviço NodePort por meio de redes públicas, vincule um EIP ao nó do cluster com antecedência.
- Depois que um Serviço é criado, se a configuração de afinidade for alternada do nível do cluster para o nível do nó, a tabela de rastreamento de conexão não será limpa. Não modifique a configuração de afinidade de Serviço após a criação do Serviço. Para modificá-lo, crie um Serviço novamente.
- Os clusters do CCE Turbo suportam apenas afinidade de serviço em nível de cluster.
- No modo de rede da VPC, quando o contêiner A é publicado por meio de um serviço NodePort e a afinidade do serviço é definida no nível do nó (ou seja, **externalTrafficPolicy** é definido como **local**), o contêiner B implementado no mesmo nó não pode acessar o contêiner A por meio do endereço IP do nó e do serviço NodePort.

- Quando um serviço NodePort é criado em um cluster v1.21.7 ou posterior, a porta no nó não é exibida usando **netstat** por padrão. Se o modo de encaminhamento de cluster for **iptables**, execute o comando **iptables -t nat -L** para exibir a porta. Se o modo de encaminhamento de cluster for **IPVS**, execute o comando **ipvsadm -Ln** para exibir a porta.

Criar um Serviço NodePort

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster.

Passo 2 Escolha **Networking** no painel de navegação e clique em **Create Service** no canto superior direito.

Passo 3 Defina os parâmetros de acesso dentro do cluster.

- **Service Name**: especifique um nome do Serviço, que pode ser o mesmo que o nome da carga de trabalho.
- **Service Type**: selecione **NodePort**.
- **Namespace**: namespace ao qual a carga de trabalho pertence.
- **Service Affinity**: para mais detalhes, consulte **externalTrafficPolicy (afinidade de serviço)**.
  - **Cluster level**: os endereços IP e as portas de acesso de todos os nós em um cluster podem acessar a carga de trabalho associada ao Serviço. O acesso ao Serviço causará perda de desempenho devido ao redirecionamento de rota e o endereço IP de origem do cliente não pode ser obtido.
  - **Node level**: somente o endereço IP e a porta de acesso do nó onde a carga de trabalho está localizada podem acessar a carga de trabalho associada ao Serviço. O acesso ao Serviço não causará perda de desempenho devido ao redirecionamento de rota e o endereço IP de origem do cliente pode ser obtido.
- **Selector**: adicione um rótulo e clique em **Confirm**. Um Serviço seleciona um pod com base no rótulo adicionado. Você também pode clicar em **Reference Workload Label** para fazer referência ao rótulo de uma carga de trabalho existente. Na caixa de diálogo exibida, selecione uma carga de trabalho e clique em **OK**.
- **IPv6**: esta função está desativada por padrão. Depois que essa função é ativada, o endereço IP do cluster do Serviço muda para um endereço IPv6. Para obter detalhes, consulte **Criação de um cluster IPv4/IPv6 de pilha dupla no CCE**. **Este parâmetro está disponível apenas em clusters de v1.15 ou posterior com IPv6 habilitado (definido durante a criação do cluster).**
- **Port Settings**
  - **Protocol**: protocolo utilizado pelo Serviço.
  - **Service Port**: porta utilizada pelo Serviço. O número da porta varia de 1 a 65535.
  - **Container Port**: porta na qual a carga de trabalho escuta. Por exemplo, o Nginx usa a porta 80 por padrão.
  - **Node Port**: é aconselhável selecionar **Auto**. Você também pode especificar uma porta. A porta padrão varia de 30000 a 32767.

Passo 4 Clique em **OK**.

----Fim

Usar o kubectl

Você pode executar comandos kubectl para definir o tipo de acesso. Esta seção usa uma carga de trabalho do Nginx como exemplo para descrever como definir um Serviço NodePort usando kubectl.

Passo 1 Use o kubectl para se conectar ao cluster. Para mais detalhes, consulte **Conexão a um cluster usando o kubectl**.

Passo 2 Crie e edite os arquivos **nginx-deployment.yaml** e **nginx-nodeport-svc.yaml**.

Os nomes dos arquivos são definidos pelo usuário. **nginx-deployment.yaml** e **nginx-nodeport-svc.yaml** são apenas nomes de arquivo de exemplo.

vi nginx-deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - image: nginx:latest
        name: nginx
      imagePullSecrets:
      - name: default-secret
```

vi nginx-nodeport-svc.yaml

```
apiVersion: v1
kind: Service
metadata:
  labels:
    app: nginx
  name: nginx-nodeport
spec:
  ports:
  - name: service
    nodePort: 30000     # Node port. The value ranges from 30000 to 32767.
    port: 8080          # Port for accessing a Service.
    protocol: TCP       # Protocol used for accessing a Service. The value can be
TCP or UDP.
    targetPort: 80      # Port used by a Service to access the target container.
This port is closely related to the applications running in a container. In this
example, the Nginx image uses port 80 by default.
  selector:             # Label selector. A Service selects a pod based on the
label and forwards the requests for accessing the Service to the pod. In this
example, select the pod with the app:nginx label.
    app: nginx
  type: NodePort        # Service type. NodePort indicates that the Service is
accessed through a node port.
```

Passo 3 Crie uma carga de trabalho.

kubectl create -f nginx-deployment.yaml

Se forem exibidas informações semelhantes às seguintes, a carga de trabalho foi criada.

```
deployment "nginx" created
```


kubectl get po

Se informações semelhantes às seguintes forem exibidas, a carga de trabalho está em execução.

```
NAME                     READY     STATUS              RESTARTS   AGE
nginx-2601814895-qhxqv   1/1       Running             0          9s
```

Passo 4 Crie um Serviço.

kubectl create -f nginx-nodeport-svc.yaml

Se forem exibidas informações semelhantes às seguintes, o Serviço foi criado.

```
service "nginx-nodeport" created
```

kubectl get svc

Se forem apresentadas informações semelhantes às seguintes, o Serviço foi criado.

```
# kubectl get svc
NAME             TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)          AGE
kubernetes       ClusterIP   10.247.0.1     <none>        443/TCP          4d8h
nginx-nodeport   NodePort    10.247.30.40   <none>        8080:30000/TCP   18s
```

Passo 5 Acesse o Serviço.

Por padrão, um Serviço NodePort pode ser acessado usando *Any node IP address:Node port*.

O Serviço pode ser acessado de um nó em outro cluster na mesma VPC ou em outro pod no cluster. Se um endereço IP público estiver vinculado ao nó, você também poderá usar o endereço IP público para acessar o Serviço. Crie um contêiner no cluster e acesse o contêiner usando *Node IP address:Node port*.

```
# kubectl get node -owide
NAME           STATUS   ROLES    AGE    INTERNAL-IP    EXTERNAL-IP   OS-
IMAGE               KERNEL-VERSION               CONTAINER-RUNTIME
10.100.0.136   Ready    <none>   152m   10.100.0.136   <none>        CentOS Linux
7 (Core)   3.10.0-1160.25.1.el7.x86_64   docker://18.9.0
10.100.0.5     Ready    <none>   152m   10.100.0.5     <none>        CentOS Linux
7 (Core)   3.10.0-1160.25.1.el7.x86_64   docker://18.9.0
# kubectl run -i --tty --image nginx:alpine test --rm /bin/sh
If you do not see a command prompt, try pressing Enter.
/ # curl 10.100.0.136:30000
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
```



```
</html>
/ #
```

----Fim

7.3.4 LoadBalancer

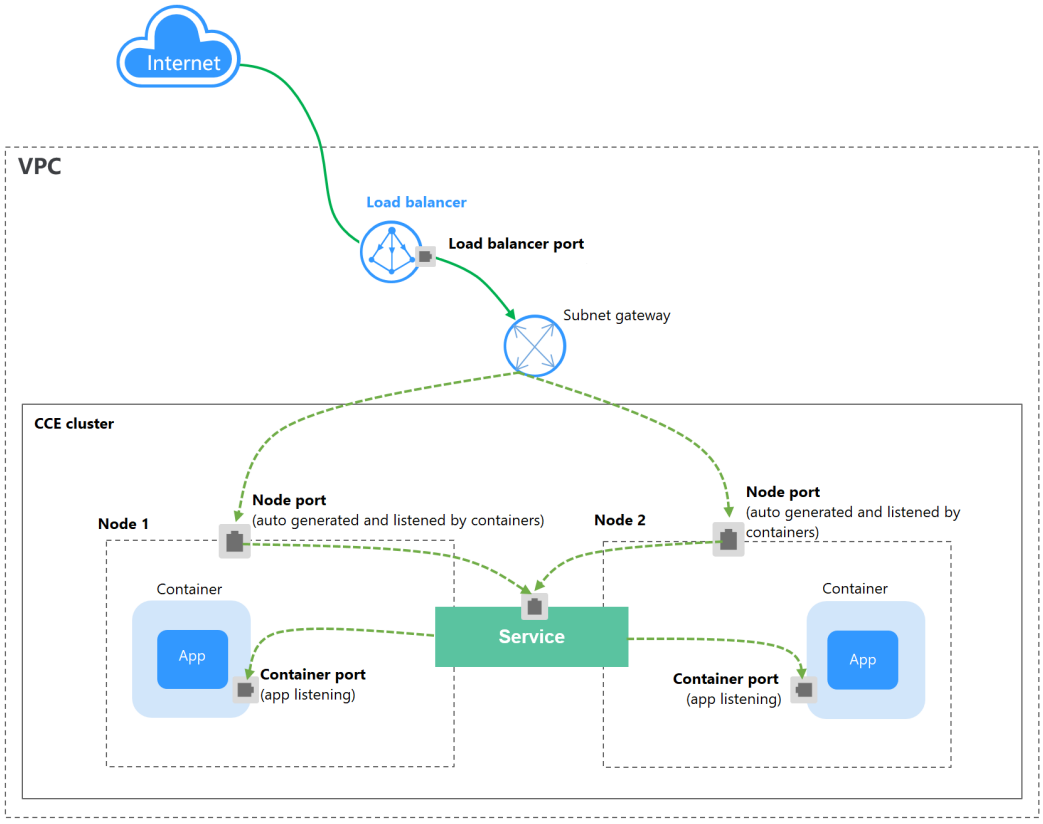
7.3.4.1 Criação de um Serviço LoadBalancer

Cenário

Os Serviços LoadBalancer podem acessar cargas de trabalho da rede pública por meio do ELB, que é mais confiável do que o acesso baseado em EIP. O endereço de acesso de LoadBalancer está no formato de *IP address of public network load balancer*:*Access port*, por exemplo, **10.117.117.117:80**.

Nesse modo de acesso, as solicitações são transmitidas por meio de um balanceador de carga do ELB para um nó e, em seguida, encaminhadas para o pod de destino por meio do Serviço.

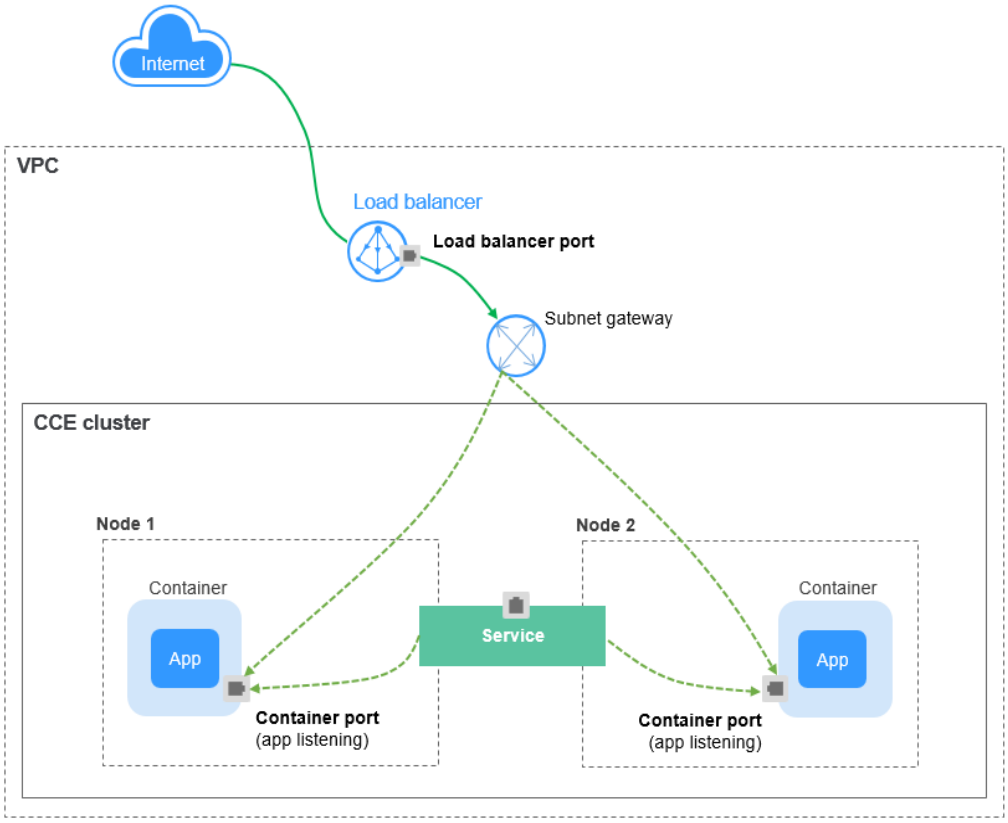
Figura 7-21 LoadBalancer



Quando **clusters do CCE Turbo e balanceadores de carga dedicados** são usados, a rede de passagem é suportada para reduzir a latência do serviço e garantir perda de desempenho zero.

As solicitações de acesso externo são encaminhadas diretamente de um balanceador de carga para os pods. As solicitações de acesso interno podem ser encaminhadas para um pod por meio de um Serviço.

Figura 7-22 Rede de passagem



Restrições

- Os Serviços LoadBalancer permitem que as cargas de trabalho sejam acessadas de redes públicas por meio do ELB. Este modo de acesso tem as seguintes restrições:
  - Balanceadores de carga criados automaticamente não devem ser usados por outros recursos. Caso contrário, esses balanceadores de carga não poderão ser completamente excluídos.
  - Não altere o nome do ouvinte do balanceador de carga em clusters v1.15 e anteriores. Caso contrário, o balanceador de carga não poderá ser acessado.
- Depois que um Serviço é criado, se a configuração de afinidade for alternada do nível do cluster para o nível do nó, a tabela de rastreamento de conexão não será limpa. Recomendamos que você não modifique a configuração de afinidade de Serviço após a criação do Serviço. Para modificá-lo, crie um Serviço novamente.
- Se a afinidade de serviço estiver definida para o nível do nó (ou seja, **externalTrafficPolicy** estiver definida como **Local**), o cluster poderá não conseguir aceder ao Serviço utilizando o endereço do ELB. Para mais detalhes, consulte **Por que um Serviço falha ao ser acessado de dentro do cluster**.
- Os clusters do CCE Turbo suportam apenas afinidade de serviço em nível de cluster.
- Balanceadores de carga ELB dedicados podem ser usados somente em clusters v1.17 e posteriores.
- Os balanceadores de carga dedicados devem ser do tipo de rede (TCP/UDP) com suporte a redes privadas (com um IP privado). Se o Serviço precisar oferecer suporte a HTTP, as especificações dos balanceadores de carga dedicados devem usar HTTP/HTTPS

(balanceamento de carga de aplicação) além do TCP/UDP (balanceamento de carga de rede).

- Em um cluster do CCE, se a afinidade no nível do cluster estiver configurada para um Serviço LoadBalancer, as solicitações serão distribuídas para as portas de cada nó usando SNAT ao entrar no cluster. O número de portas de nó não pode exceder o número de portas de nó disponíveis no nó. Se a afinidade de serviço estiver no nível do nó (Local), não há tal restrição. Em um cluster do CCE Turbo, essa restrição se aplica aos balanceadores de carga do ELB compartilhados, mas não aos dedicados. Use balanceadores de carga do ELB dedicados em clusters do CCE Turbo.
- Quando o modo de encaminhamento (proxy) do serviço de cluster é IPVS, o IP do nó não pode ser configurado como o IP externo do serviço. Caso contrário, o nó não está disponível.
- Em um cluster usando o modo de proxy IPVS, se o ingress e o Serviço usarem o mesmo balanceador de carga do ELB, o ingress não pode ser acessado dos nós e contêineres no cluster porque o kube-proxy monta o endereço do Serviço LoadBalancer na ponte ipvs-0. Essa ponte intercepta o tráfego do balanceador de carga conectado ao ingress. Use balanceadores de carga do ELB diferentes para o ingress e o Serviço.

Criação de um Serviço LoadBalancer

Passo 1 Efetue logon no console do CCE e acesse o console do cluster.

Passo 2 Escolha **Networking** no painel de navegação e clique em **Create Service** no canto superior direito.

Passo 3 Configure parâmetros.

- **Service Name**: especifique um nome do Serviço, que pode ser o mesmo que o nome da carga de trabalho.
- **Service Type**: selecione **LoadBalancer**.
- **Namespace**: namespace ao qual a carga de trabalho pertence.
- **Service Affinity**: para mais detalhes, consulte **externalTrafficPolicy (afinidade de serviço)**.
  - **Cluster level**: os endereços IP e as portas de acesso de todos os nós em um cluster podem acessar a carga de trabalho associada ao Serviço. O acesso ao Serviço causará perda de desempenho devido ao redirecionamento de rota e o endereço IP de origem do cliente não pode ser obtido.
  - **Node level**: somente o endereço IP e a porta de acesso do nó onde a carga de trabalho está localizada podem acessar a carga de trabalho associada ao Serviço. O acesso ao Serviço não causará perda de desempenho devido ao redirecionamento de rota e o endereço IP de origem do cliente pode ser obtido.
- **Selector**: adicione um rótulo e clique em **Confirm**. Um Serviço seleciona um pod com base no rótulo adicionado. Você também pode clicar em **Reference Workload Label** para fazer referência ao rótulo de uma carga de trabalho existente. Na caixa de diálogo exibida, selecione uma carga de trabalho e clique em **OK**.
- **IPv6**: esta função está desativada por padrão. Depois que essa função é ativada, o endereço IP do cluster do Serviço muda para um endereço IPv6. Para obter detalhes, consulte **Criação de um cluster IPv4/IPv6 de pilha dupla no CCE**. **Este parâmetro está disponível apenas em clusters de v1.15 ou posterior com IPv6 habilitado (definido durante a criação do cluster).**
- **Load Balancer**

Selecione o balanceador de carga para interconexão. Somente balanceadores de carga na mesma VPC que o cluster são compatíveis. Se nenhum balanceador de carga estiver disponível, clique em **Create Load Balancer** para criar um no console do ELB.

O console do CCE suporta a criação automática de balanceadores de carga. Selecione **Auto create** na caixa de listagem suspensa e defina os seguintes parâmetros:

- **Instance Name**: insira o nome de um balanceador de carga.
- **Public Access**: se ativado, um EIP com largura de banda de 5 Mbit/s será criado. Por padrão, é cobrado pelo tráfego.
- **Subnet**, **AZ**, e **Specifications** (disponíveis apenas para balanceadores de carga dedicados): configure a sub-rede, a AZ e as especificações. Somente balanceadores de carga dedicados do tipo de rede (TCP/UDP) podem ser criados automaticamente.

Você pode clicar em **Edit** na área **Set ELB** e configurar os parâmetros do balanceador de carga na caixa de diálogo **Set ELB**.

- **Algorithm**: três algoritmos estão disponíveis: weighted round robin, weighted least connections algorithm ou source IP hash.

  NOTA

  - **Weighted round robin**: as solicitações são encaminhadas para servidores diferentes com base em seus pesos, que indicam o desempenho do processamento do servidor. Servidores back-end com pesos mais altos recebem proporcionalmente mais solicitações, enquanto servidores com pesos iguais recebem o mesmo número de solicitações. Este algoritmo é normalmente usado para conexões curtas, como serviços HTTP.
  - **Weighted least connections**: além do peso atribuído a cada servidor, o número de conexões processadas por cada servidor back-end também é considerado. As solicitações são encaminhadas ao servidor com a menor relação entre conexões e peso. Desenvolvido com base no algoritmo de **menos conexões**, o algoritmo **weighted least connections** atribui um peso a cada servidor de acordo com sua capacidade de processamento. Este algoritmo é normalmente usado para conexões persistentes, como conexões de bancos de dados.
  - **Source IP hash**: o endereço IP de origem de cada solicitação é calculado usando-se o algoritmo de hash para obter uma chave de hash única, e todos os servidores back-end são numerados. A chave gerada aloca o cliente a um servidor particular. Isso permite que solicitações de clientes diferentes sejam distribuídas no modo de balanceamento de carga e garante que as solicitações do mesmo cliente sejam encaminhadas para o mesmo servidor. Este algoritmo aplica-se a conexões TCP sem cookies.

- **Type**: esta função está desativada por padrão. Você pode selecionar **Source IP address**. Sessão persistente baseada em endereço IP de origem significa que as solicitações de acesso do mesmo endereço IP são encaminhadas para o mesmo servidor back-end.

  NOTA

  Quando a **política de distribuição** utiliza o hash do IP de origem, a sessão adesiva não pode ser definida.

- **Health Check**: configure a verificação de integridade do balanceador de carga.
  - **Global health check**: aplica-se apenas a portas que utilizam o mesmo protocolo. É aconselhável selecionar **Custom health check**.
  - **Custom health check**: aplica-se a **portas** que usam protocolos diferentes. Para obter detalhes sobre a definição YAML para verificação de integridade personalizada, consulte **Configuração da verificação de integridade para várias portas**.

Tabela 7-4 Parâmetros de verificação de integridade

Parâmetro	Descrição
Protocolo	Quando o protocolo de **Port** é definido como TCP, o TCP e o HTTP são suportados. Quando o protocolo de **Port** é definido como UDP, o UDP é suportado. – **Check Path** (suportado apenas por HTTP para verificação de integridade): especifica o URL de verificação de integridade. O caminho de verificação deve começar com uma barra (/) e pode conter de 1 a 80 caracteres.
Port	Por padrão, a porta de serviço (Node Port e Container Port do Serviço) é usada para verificação de integridade. Você também pode especificar uma porta para verificação de integridade. Depois que a porta for especificada, uma porta de serviço chamada **cce-healthz** será adicionada ao Serviço. – **Node Port**: se um balanceador de carga compartilhado for usado ou nenhuma instância de ENI estiver associada, a porta do nó será usada como a porta de verificação de integridade. Se este parâmetro não for especificado, uma porta aleatória será usada. O valor varia de 30000 a 32767. – **Container Port**: quando um balanceador de carga dedicado é associado a uma instância de ENI, a porta do contêiner é usada para verificação de integridade. O valor varia de 1 a 65535.
Check Period (s)	Especifica o intervalo máximo entre as verificações de integridade. O valor varia de 1 a 50.
Timeout (s)	Especifica a duração máxima do tempo limite para cada verificação de integridade. O valor varia de 1 a 50.
Max. Retries	Especifica o número máximo de tentativas de verificação de integridade. O valor varia de 1 a 10.

- **Port**
  - **Protocol**: protocolo utilizado pelo Serviço.
  - **Service Port**: porta utilizada pelo Serviço. O número da porta varia de 1 a 65535.
  - **Container Port**: porta na qual a carga de trabalho escuta. Por exemplo, o Nginx usa a porta 80 por padrão.
  - **Health Check**: se **Health Check** estiver definida como **Custom health check**, você poderá configurar a verificação de integridade para portas usando protocolos diferentes. Para mais detalhes, consulte **Tabela 7-4**.

  NOTA

  Quando um Serviço LoadBalancer é criado, um número de porta de nó aleatório (NodePort) é gerado automaticamente.
- **Annotation**: o Serviço LoadBalancer tem algumas funções avançadas de CCE, que são implementadas por anotações. Para mais detalhes, consulte **Uso de anotações para configurar o balanceamento de carga**.

Passo 4 Clique em **OK**.

----Fim

Usar o kubectl para criar um Serviço (usando um balanceador de carga existente)

Você pode definir o Serviço ao criar uma carga de trabalho usando o kubectl. Esta seção usa uma carga de trabalho do Nginx como exemplo para descrever como adicionar um Serviço NodePort usando kubectl.

Passo 1 Use o kubectl para se conectar ao cluster. Para mais detalhes, consulte **Conexão a um cluster usando o kubectl**.

Passo 2 Crie os arquivos denominados **nginx-deployment.yaml** e **nginx-elb-svc.yaml** e edite-os.

Os nomes dos arquivos são definidos pelo usuário. **nginx-deployment.yaml** e **nginx-elb-svc.yaml** são apenas exemplo de nomes de arquivo.

vi nginx-deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - image: nginx
        name: nginx
      imagePullSecrets:
      - name: default-secret
```

vi nginx-elb-svc.yaml

NOTA

Antes de ativar a sessão persistente, certifique-se de que as seguintes condições sejam atendidas:

- O protocolo de carga de trabalho é TCP.
- A antiafinidade foi configurada entre os pods da carga de trabalho. Ou seja, todos os pods da carga de trabalho são implementados em nós diferentes. Para mais detalhes, consulte **Política de agendamento (afinidade/antiafinidade)**.

```
apiVersion: v1
kind: Service
metadata:
  name: nginx
  annotations:
    kubernetes.io/elb.id: <your_elb_id>                          # ELB ID. Replace
it with the actual value.
    kubernetes.io/elb.class: performance                      # Load balancer type
    kubernetes.io/elb.lb-algorithm: ROUND_ROBIN                  # Load balancer
algorithm
    kubernetes.io/elb.session-affinity-mode: SOURCE_IP          # The sticky
session type is source IP address.
    kubernetes.io/elb.session-affinity-option: '{"persistence_timeout":
"30"}'     # Stickiness duration (min)
```



```
    kubernetes.io/elb.health-check-flag: 'on'                  # Enable the ELB
health check function.
    kubernetes.io/elb.health-check-option: '{
      "protocol":"TCP",
      "delay":"5",
      "timeout":"10",
      "max_retries":"3"
    }'
spec:
  selector:
     app: nginx
  ports:
  - name: service0
    port: 80     # Port for accessing the Service, which is also the listener
port on the load balancer.
    protocol: TCP
    targetPort: 80  # Port used by a Service to access the target container. This
port is closely related to the applications running in a container.
    nodePort: 31128  # Port number of the node. If this parameter is not
specified, a random port number ranging from 30000 to 32767 is generated.
  type: LoadBalancer
```

O exemplo anterior usa anotações para implementar algumas funções avançadas de balanceamento de carga, como sessão persistente e verificação de integridade. Para mais detalhes, consulte **Tabela 7-5**.

Para obter mais anotações e exemplos relacionados a funções avançadas, consulte **Uso de anotações para configurar o balanceamento de carga**.

Tabela 7-5 Parâmetros de anotações

Parâmetro	Obrigatório	Tipo	Descrição
kubernetes.io/elb.id	Sim	String	ID de um balanceador de carga aprimorado. Obrigatório quando um balanceador de carga existente deve ser associado. **Como obter**: No console de gerenciamento, clique em **Service List** e escolha **Networking** > **Elastic Load Balance**. Clique no nome do balanceador de carga de destino. Na página de guia **Summary**, localize e copie o ID. **NOTA** O sistema se conecta preferencialmente ao balanceador de carga com base no campo **kubernetes.io/elb.id**. Se este campo não for especificado, o campo **spec.loadBalancerIP** é usado (opcional e disponível apenas em 1.23 e versões anteriores). Não use o campo **spec.loadBalancerIP** para se conectar ao balanceador de carga. Este campo será descartado pelo Kubernetes. Para obter detalhes, consulte **Deprecação**.

Parâmetro	Obrigatório	Tipo	Descrição
kubernetes.io/elb.class	Sim	String	Selecione um tipo adequado de balanceador de carga. O valor pode ser: ● **union**: balanceador de carga compartilhado ● **performance**: balanceador de carga dedicado, que pode ser usado apenas em clusters de v1.17 e posteriores. Para obter detalhes, consulte **Diferenças entre balanceadores de carga compartilhado e dedicado**. **NOTA** Se um Serviço LoadBalancer acessar um balanceador de carga dedicado existente, o balanceador de carga dedicado deve suportar redes TCP/UDP.
kubernetes.io/elb.lb-algorithm	Não	String	Especifica o algoritmo de balanceamento de carga do grupo de servidores back-end. O valor padrão é **ROUND_ROBIN**. Opções: ● **ROUND_ROBIN**: algoritmo round robin ponderado ● **LEAST_CONNECTIONS**: algoritmo de menor conexão ponderada ● **SOURCE_IP**: algoritmo de hash do IP de origem **NOTA** Se esse parâmetro for definido como **SOURCE_IP**, a configuração de peso (campo **weight**) dos servidores back-end vinculados ao grupo de servidores back-end será inválida e a sessão persistente não poderá ser ativada.
kubernetes.io/elb.session-affinity-mode	Não	String	A sessão persistente baseada em endereço IP de origem é suportada. Ou seja, as solicitações do mesmo endereço IP sejam encaminhadas para o mesmo servidor back-end. ● Disabling sticky session: não configure este parâmetro. ● Enabling sticky session: defina esse parâmetro como **SOURCE_IP**, indicando que a sessão persistente é baseada no endereço IP de origem. **NOTA** Quando **kubernetes.io/elb.lb-algorithm** é definido como **SOURCE_IP** (source IP hash), sessão persistente não pode ser ativada.

Parâmetro	Obrigatório	Tipo	Descrição
kubernetes.io/elb.session-affinity-option	Não	Tabela 7-6 object	Tempo limite de sessão persistente expira.
kubernetes.io/elb.health-check-flag	Não	String	Se deve ativar a verificação de integridade do ELB. ● Enabling health check: deixe este parâmetro em branco ou defina-o como **on**. ● Disabling health check: defina este parâmetro como **off**. Se esse parâmetro estiver ativado, o campo **kubernetes.io/elb.health-check-option** também deve ser especificado ao mesmo tempo.
kubernetes.io/elb.health-check-option	Não	Tabela 7-7 object	Itens de configuração de verificação de integridade do ELB.

Tabela 7-6 Estrutura de dados de elb.session-affinity-option

Parâmetro	Obrigatório	Tipo	Descrição
persistence_timeout	Sim	String	Tempo limite de sessão persistente, em minutos. Esse parâmetro é válido somente quando **elb.session-affinity-mode** é definido como **SOURCE_IP**. Intervalo de valores: 1 a 60. Valor padrão: **60**

Tabela 7-7 Estrutura de dados de elb.health-check-option

Parâmetro	Obrigatório	Tipo	Descrição
delay	Não	String	Tempo de espera inicial (em segundos) para iniciar a verificação de integridade. Intervalo de valores: 1 a 50. Valor padrão: **5**
timeout	Não	String	Tempo limite de verificação de integridade, em segundos. Intervalo de valores: 1 a 50. Valor padrão: **10**

Parâmetro	Obrigatório	Tipo	Descrição
max_retries	Não	String	Número máximo de tentativas de verificação de integridade. Intervalo de valores: 1 a 10. Valor padrão: **3**
protocol	Não	String	Protocolo de verificação de integridade. Opções de valor: TCP ou HTTP
path	Não	String	URL de verificação de integridade. Este parâmetro precisa ser configurado quando o protocolo é **HTTP**. Valor padrão: / O valor pode conter de 1 a 10.000 caracteres.

Passo 3 Crie uma carga de trabalho.

kubectl create -f nginx-deployment.yaml

Se forem exibidas informações semelhantes às seguintes, a carga de trabalho foi criada.

```
deployment/nginx created
```

kubectl get pod

Se informações semelhantes às seguintes forem exibidas, a carga de trabalho está em execução.

```
NAME                     READY     STATUS              RESTARTS   AGE
nginx-2601814895-c1xhw   1/1       Running             0          6s
```

Passo 4 Crie um Serviço.

kubectl create -f nginx-elb-svc.yaml

Se forem exibidas informações semelhantes às seguintes, o Serviço foi criado.

```
service/nginx created
```

kubectl get svc

Se informações semelhantes às seguintes forem exibidas, o tipo de acesso foi definido e a carga de trabalho está acessível.

```
NAME         TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)         AGE
kubernetes   ClusterIP      10.247.0.1       <none>        443/TCP         3d
nginx        LoadBalancer   10.247.130.196   10.78.42.242   80:31540/TCP   51s
```

Passo 5 Digite o URL na caixa de endereço do navegador, por exemplo, **10.78.42.242:80**. **10.78.42.242** indica o endereço IP do balanceador de carga e **80** indica a porta de acesso exibida no console do CCE.

O Nginx é acessível.

Figura 7-23 Acessar o Nginx por meio do Serviço LoadBalancer

----Fim

Usar o kubectl para criar um Serviço (criação automática de um balanceador de carga)

Você pode definir o Serviço ao criar uma carga de trabalho usando o kubectl. Esta seção usa uma carga de trabalho do Nginx como exemplo para descrever como adicionar um Serviço NodePort usando kubectl.

Passo 1 Use o kubectl para se conectar ao cluster. Para mais detalhes, consulte **Conexão a um cluster usando o kubectl**.

Passo 2 Crie os arquivos denominados **nginx-deployment.yaml** e **nginx-elb-svc.yaml** e edite-os.

Os nomes dos arquivos são definidos pelo usuário. **nginx-deployment.yaml** e **nginx-elb-svc.yaml** são apenas exemplo de nomes de arquivo.

vi nginx-deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - image: nginx
        name: nginx
      imagePullSecrets:
      - name: default-secret
```

vi nginx-elb-svc.yaml

NOTA

Antes de ativar a sessão persistente, certifique-se de que as seguintes condições sejam atendidas:

- O protocolo de carga de trabalho é TCP.
- A antiafinidade foi configurada entre os pods da carga de trabalho. Ou seja, todos os pods da carga de trabalho são implementados em nós diferentes. Para mais detalhes, consulte **Política de agendamento (afinidade/antiafinidade)**.

Exemplo de um Serviço usando um balanceador de carga compartilhado de rede pública:

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    kubernetes.io/elb.class: union
    kubernetes.io/elb.autocreate: '{
      "type": "public",
      "bandwidth_name": "cce-bandwidth-1551163379627",
      "bandwidth_chargemode": "bandwidth",
      "bandwidth_size": 5,
      "bandwidth_sharetype": "PER",
      "eip_type": "5_bgp"
    }'
    kubernetes.io/elb.enterpriseID: 0       # ID of the enterprise project to
which the load balancer belongs
    kubernetes.io/elb.lb-algorithm: ROUND_ROBIN                   # Load balancer
algorithm
    kubernetes.io/elb.session-affinity-mode: SOURCE_IP          # The sticky
session type is source IP address.
    kubernetes.io/elb.session-affinity-option: '{"persistence_timeout":
"30"}'     # Stickiness duration (min)
    kubernetes.io/elb.health-check-flag: 'on'                   # Enable the ELB
health check function.
    kubernetes.io/elb.health-check-option: '{
      "protocol":"TCP",
      "delay":"5",
      "timeout":"10",
      "max_retries":"3"
    }'
  labels:
    app: nginx
  name: nginx
spec:
  ports:
  - name: service0
    port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: nginx
  type: LoadBalancer
```

Exemplo de Serviço usando um balanceador de carga dedicado de rede pública (somente para clusters de v1.17 e posterior):

```
apiVersion: v1
kind: Service
metadata:
  name: nginx
  labels:
    app: nginx
  namespace: default
  annotations:
    kubernetes.io/elb.class: performance
    kubernetes.io/elb.autocreate: '{
      "type": "public",
      "bandwidth_name": "cce-bandwidth-1626694478577",
      "bandwidth_chargemode": "bandwidth",
      "bandwidth_size": 5,
```



```
      "bandwidth_sharetype": "PER",
      "eip_type": "5_bgp",
      "available_zone": [
         ""
      ],
      "l4_flavor_name": "L4_flavor.elb.s1.small"
    }'
    kubernetes.io/elb.enterpriseID: 0       # ID of the enterprise project to
which the load balancer belongs
    kubernetes.io/elb.lb-algorithm: ROUND_ROBIN                   # Load balancer
algorithm
    kubernetes.io/elb.session-affinity-mode: SOURCE_IP          # The sticky
session type is source IP address.
    kubernetes.io/elb.session-affinity-option: '{"persistence_timeout":
"30"}'     # Stickiness duration (min)
    kubernetes.io/elb.health-check-flag: 'on'                   # Enable the ELB
health check function.
    kubernetes.io/elb.health-check-option: '{
      "protocol":"TCP",
      "delay":"5",
      "timeout":"10",
      "max_retries":"3"
    }'
spec:
  selector:
    app: nginx
  ports:
  - name: cce-service-0
    targetPort: 80
    nodePort: 0
    port: 80
    protocol: TCP
  type: LoadBalancer
```

O exemplo anterior usa anotações para implementar algumas funções avançadas de balanceamento de carga, como sessão persistente e verificação de integridade. Para mais detalhes, consulte **Tabela 7-8**.

Para obter mais anotações e exemplos relacionados a funções avançadas, consulte **Uso de anotações para configurar o balanceamento de carga**.

Tabela 7-8 Parâmetros de anotações

Parâmetro	Obrigatório	Tipo	Descrição
kubernetes.io/elb.class	Sim	String	Selecione um tipo adequado de balanceador de carga. O valor pode ser: ● **union**: balanceador de carga compartilhado ● **performance**: balanceador de carga dedicado, que pode ser usado apenas em clusters de v1.17 e posteriores. Para obter detalhes, consulte **Diferenças entre balanceadores de carga compartilhado e dedicado**.

Parâmetro	Obrigatório	Tipo	Descrição
kubernetes.io/elb.autocreate	Sim	**elb.autocreate** object	Se criar automaticamente um balanceador de carga associado ao Serviço. **Exemplo** ● Se um balanceador de carga de rede pública for criado automaticamente, defina esse parâmetro com o seguinte valor: {"type":"public","bandwidth_name":"cce-bandwidth-1551163379627","bandwidth_chargemode":"bandwidth","bandwidth_size":5,"bandwidth_sharetype":"PER","eip_type":"5_bgp","name":"james"} ● Se um balanceador de carga de rede privada for criado automaticamente, defina esse parâmetro com o seguinte valor: {"type":"inner","name":"A-location-d-test"}
kubernetes.io/elb.subnet-id	Nenhuma	String	ID da sub-rede onde o cluster está localizado. O valor pode conter de 1 a 100 caracteres. ● Obrigatório quando um cluster v1.11.7-r0 ou anterior deve ser criado automaticamente. ● Opcional para clusters posteriores à v1.11.7-r0. Para obter detalhes sobre como obter o valor, consulte**Qual é a diferença entre a API de sub-rede da VPC e a API de sub-rede do OpenStack Neutron**.
kubernetes.io/elb.enterpriseID	Não	String	**Clusters de v1.15 e versões posteriores suportam este campo. Nos clusters anteriores à v1.15, os balanceadores de carga são criados no projeto padrão por padrão.** Esse parâmetro indica o ID do projeto empresarial no qual o balanceador de carga do ELB será criado. Se esse parâmetro não for especificado ou for definido como **0**, os recursos serão vinculados ao projeto corporativo padrão. **Como obter**: Faça logon no console de gerenciamento e escolha **Enterprise** > **Project Management** na barra de menu superior. Na lista exibida, clique no nome do projeto empresarial de destino e copie o ID na página de detalhes do projeto empresarial.

Parâmetro	Obrigatório	Tipo	Descrição
kubernetes.io/elb.lb-algorithm	Não	String	Especifica o algoritmo de balanceamento de carga do grupo de servidores back-end. O valor padrão é **ROUND_ROBIN**. Opções: ● **ROUND_ROBIN**: algoritmo round robin ponderado ● **LEAST_CONNECTIONS**: algoritmo de menor conexão ponderada ● **SOURCE_IP**: algoritmo de hash do IP de origem **NOTA** Se esse parâmetro for definido como **SOURCE_IP**, a configuração de peso (campo **weight**) dos servidores back-end vinculados ao grupo de servidores back-end será inválida e a sessão persistente não poderá ser ativada.
kubernetes.io/elb.session-affinity-mode	Não	String	A sessão persistente baseada em endereço IP de origem é suportada. Ou seja, as solicitações do mesmo endereço IP sejam encaminhadas para o mesmo servidor back-end. ● Disabling sticky session: não configure este parâmetro. ● Enabling sticky session: defina esse parâmetro como **SOURCE_IP**, indicando que a sessão persistente é baseada no endereço IP de origem. **NOTA** Quando **kubernetes.io/elb.lb-algorithm** é definido como **SOURCE_IP** (source IP hash), sessão persistente não pode ser ativada.
kubernetes.io/elb.session-affinity-option	Não	**Tabela 7-6** object	Tempo limite de sessão persistente expira.
kubernetes.io/elb.health-check-flag	Não	String	Se deve ativar a verificação de integridade do ELB. ● Enabling health check: deixe este parâmetro em branco ou defina-o como **on**. ● Disabling health check: defina este parâmetro como **off**. Se esse parâmetro estiver ativado, o campo **kubernetes.io/elb.health-check-option** também deve ser especificado ao mesmo tempo.
kubernetes.io/elb.health-check-option	Não	**Tabela 7-7** object	Itens de configuração de verificação de integridade do ELB.

Tabela 7-9 Estrutura de dados de elb.autocreate

Parâmetro	Obrigatório	Tipo	Descrição
name	Não	String	Nome do balanceador de carga criado automaticamente. O valor pode conter de 1 a 64 caracteres. Somente letras, dígitos, sublinhados (_), hifens (-) e pontos (.) são permitidos. Padrão: **cce-lb+service.UID**
type	Não	String	Tipo de rede do balanceador de carga. ● **public**: balanceador de carga de rede pública ● **inner**: balanceador de carga de rede privada Padrão: **inner**
bandwidth_name	Sim para balanceadores de carga de rede pública	String	Nome da largura de banda. O valor padrão é **cce-bandwidth-******. O valor pode conter de 1 a 64 caracteres. Somente letras, dígitos, sublinhados (_), hifens (-) e pontos (.) são permitidos.
bandwidth_chargemode	Não	String	Modo de cobrança de largura de banda. ● **bandwidth**: cobrado pela largura de banda ● **traffic**: cobrado pelo tráfego Padrão: **bandwidth**
bandwidth_size	Sim para balanceadores de carga de rede pública	Integer	Tamanho da largura de banda. O valor padrão é de 1 a 2000 Mbit/s. Configure este parâmetro com base na faixa de largura de banda permitida em sua região. O incremento mínimo para ajuste de largura de banda varia dependendo do intervalo de largura de banda. ● O incremento mínimo é de 1 Mbit/s se a largura de banda permitida não exceder 300 Mbit/s. ● O incremento mínimo é de 50 Mbit/s se a largura de banda permitida varia de 300 Mbit/s a 1000 Mbit/s. ● O incremento mínimo é de 500 Mbit/s se a largura de banda permitida exceder 1000 Mbit/s.
bandwidth_sharetype	Sim para balanceadores de carga de rede pública	String	Modo de compartilhamento de largura de banda. ● **PER**: largura de banda dedicada

Parâmetro	Obrigatório	Tipo	Descrição
eip_type	Sim para balanceadores de carga de rede pública	String	Tipo de EIP. ● **5_telcom**: China Telecom ● **5_union**: China Unicom ● **5_bgp**: BGP dinâmico ● **5_sbgp**: BGP estático
vip_subnet_cidr_id	Não	String	Sub-rede onde o balanceador de carga está localizado. Este campo é suportado por clusters de v1.21 ou posterior. Se esse parâmetro não for especificado, o balanceador de carga do ELB e o cluster estarão na mesma sub-rede.
available_zone	Sim	Array of strings	AZ onde o balanceador de carga está localizado. Você pode obter todas as AZs suportadas **consultando a lista de AZ**. Esse parâmetro está disponível apenas para balanceadores de carga compartilhados.
l4_flavor_name	Sim	String	Nome do flavor do balanceador de carga da camada 4. Você pode obter todos os tipos suportados **consultando a lista de flavors**. Esse parâmetro está disponível apenas para balanceadores de carga compartilhados.
l7_flavor_name	Não	String	Nome do flavor do balanceador de carga da camada-7. Você pode obter todos os tipos suportados **consultando a lista de flavors**. Esse parâmetro está disponível apenas para balanceadores de carga compartilhados. O valor deste parâmetro deve ser o mesmo de **l4_flavor_name**, ou seja, ambos são especificações elásticas ou especificações fixas.

Parâmetro	Obrigatório	Tipo	Descrição
elb_virsubnet_ids	Não	Array of strings	Sub-rede onde o servidor back-end do balanceador de carga está localizado. Se esse parâmetro for deixado em branco, a sub-rede do cluster padrão será usada. Os balanceadores de carga ocupam um número diferente de endereços IP de sub-rede com base em suas especificações. Não use os blocos CIDR de sub-rede de outros recursos (como clusters e nós) como o bloco CIDR do balanceador de carga. Esse parâmetro está disponível apenas para balanceadores de carga compartilhados. Exemplo: `"elb_virsubnet_ids": [ "14567f27-8ae4-42b8-ae47-9f847a4690dd" ]`

Passo 3 Crie uma carga de trabalho.

kubectl create -f nginx-deployment.yaml

Se forem exibidas informações semelhantes às seguintes, a carga de trabalho está sendo criada.

```
deployment/nginx created
```

kubectl get pod

Se informações semelhantes às seguintes forem exibidas, a carga de trabalho está em execução.

```
NAME                     READY     STATUS              RESTARTS   AGE
nginx-2601814895-c1xhw   1/1       Running             0          6s
```

Passo 4 Crie um Serviço.

kubectl create -f nginx-elb-svc.yaml

Se forem exibidas informações semelhantes às seguintes, o Serviço foi criado.

```
service/nginx created
```

kubectl get svc

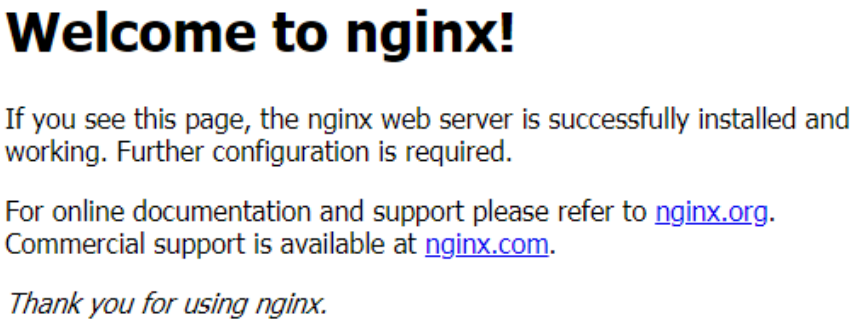
Se informações semelhantes às seguintes forem exibidas, o tipo de acesso foi definido e a carga de trabalho está acessível.

```
NAME         TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)         AGE
kubernetes   ClusterIP      10.247.0.1       <none>        443/TCP         3d
nginx        LoadBalancer   10.247.130.196   10.78.42.242   80:31540/TCP   51s
```

Passo 5 Digite o URL na caixa de endereço do navegador, por exemplo, **10.78.42.242:80**. **10.78.42.242** indica o endereço IP do balanceador de carga e **80** indica a porta de acesso exibida no console do CCE.

O Nginx é acessível.

Figura 7-24 Acessar o Nginx por meio do Serviço LoadBalancer



----Fim

7.3.4.2 Uso de anotações para configurar o balanceamento de carga

Você pode adicionar anotações a um arquivo YAML para usar algumas funções avançadas do CCE. Esta seção descreve as anotações disponíveis quando um serviço LoadBalancer é criado.

- **Interligação com o ELB**
- **Sessão persistente**
- **Verificação de integridade**
- **Protocolo HTTP**
- **Ajuste dinâmico do peso do ECS de back-end**
- **Capacidade de passagem**
- **Lista branca**
- **Rede host**

Interligação com o ELB

Tabela 7-10 Anotações para interconexão com o ELB

Parâmetro	Tipo	Descrição	Versão do cluster suportada
kubernetes.io/elb.class	String	Selecione um tipo adequado de balanceador de carga. O valor pode ser: ● **union**: balanceador de carga compartilhado ● **performance**: balanceador de carga dedicado, que pode ser usado apenas em clusters de v1.17 e posteriores. Para obter detalhes, consulte **Diferenças entre balanceadores de carga compartilhado e dedicado**.	v1.9 ou mais recente

Parâmetro	Tipo	Descrição	Versão do cluster suportada
kubernetes.io/elb.id	String	Obrigatório **quando um balanceador de carga existente deve ser associado**. ID de um balanceador de carga. **Como obter**: No console de gerenciamento, clique em **Service List** e escolha **Networking** > **Elastic Load Balance**. Clique no nome do balanceador de carga de destino. Na página de guia **Summary**, localize e copie o ID. **NOTA** O sistema se conecta preferencialmente ao balanceador de carga com base no campo **kubernetes.io/elb.id**. Se este campo não for especificado, o campo **spec.loadBalancerIP** é usado (opcional e disponível apenas em 1.23 e versões anteriores). Não use o campo **spec.loadBalancerIP** para se conectar ao balanceador de carga. Este campo será descartado pelo Kubernetes. Para obter detalhes, consulte **Deprecação**.	v1.9 ou mais recente
kubernetes.io/elb.autocreate	**Tabela 7-18**	Obrigatório **quando os balanceadores de carga são criados automaticamente**. **Exemplo:** ● Se um balanceador de carga de rede pública for criado automaticamente, defina esse parâmetro com o seguinte valor: {"type":"public","bandwidth_name":"cce-bandwidth-1551163379627","bandwidth_chargemode":"bandwidth","bandwidth_size":5,"bandwidth_sharetype":"PER","eip_type":"5_bgp","name":"james"} ● Se um balanceador de carga de rede privada for criado automaticamente, defina esse parâmetro com o seguinte valor: {"type":"inner","name":"A-location-d-test"}	v1.9 ou mais recente

Parâmetro	Tipo	Descrição	Versão do cluster suportada
kubernetes.io/elb.enterpriseID	String	Opcional **quando balanceadores de carga são criados automaticamente**. **Clusters de v1.15 e versões posteriores suportam este campo. Nos clusters anteriores à v1.15, os balanceadores de carga são criados no projeto padrão por padrão.** Esse parâmetro indica o ID do projeto empresarial no qual o balanceador de carga do ELB será criado. Se esse parâmetro não for especificado ou for definido como **0**, os recursos serão vinculados ao projeto corporativo padrão. **Como obter**: Faça logon no console de gerenciamento e escolha **Enterprise** > **Project Management** na barra de menu superior. Na lista exibida, clique no nome do projeto empresarial de destino e copie o ID na página de detalhes do projeto empresarial.	v1.15 ou mais recente
kubernetes.io/elb.subnet-id	String	Opcional **quando balanceadores de carga são criados automaticamente**. ID da sub-rede onde o cluster está localizado. O valor pode conter de 1 a 100 caracteres. ● Obrigatório quando um cluster v1.11.7-r0 ou anterior deve ser criado automaticamente. ● Opcional para clusters posteriores à v1.11.7-r0.	Obrigatório para versões anteriores à v1.11.7-r0 Descartado em versões posteriores à v1.11.7-r0
kubernetes.io/elb.lb-algorithm	String	Especifica o algoritmo de balanceamento de carga do grupo de servidores back-end. O valor padrão é **ROUND_ROBIN**. Opções: ● **ROUND_ROBIN**: algoritmo round robin ponderado ● **LEAST_CONNECTIONS**: algoritmo de menor conexão ponderada ● **SOURCE_IP**: algoritmo de hash do IP de origem **NOTA** Se esse parâmetro for definido como **SOURCE_IP**, a configuração de peso (campo **weight**) dos servidores de back-end vinculados ao grupo de servidores de back-end será inválida e a sessão fixa não poderá ser ativada.	v1.9 ou mais recente

A seguir, mostra como usar as anotações anteriores:

- Associar um balanceador de carga existente. Para mais detalhes, consulte **Usar o kubectl para criar um Serviço (usando um balanceador de carga existente)**.

```
apiVersion: v1
kind: Service
metadata:
  name: nginx
  annotations:
    kubernetes.io/elb.id: <your_elb_id>                          # ELB ID.
Replace it with the actual value.
    kubernetes.io/elb.class: performance                         # Load
balancer type
    kubernetes.io/elb.lb-algorithm: ROUND_ROBIN                    # Load
balancer algorithm
spec:
  selector:
     app: nginx
  ports:
  - name: service0
    port: 80
    protocol: TCP
    targetPort: 80
  type: LoadBalancer
```

- Criar automática de um balanceador de carga. Para mais detalhes, consulte **Usar o kubectl para criar um Serviço (criação automática de um balanceador de carga)**.

  Balanceador de carga compartilhado:

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    kubernetes.io/elb.class: union
    kubernetes.io/elb.autocreate: '{
      "type": "public",
      "bandwidth_name": "cce-bandwidth-1551163379627",
      "bandwidth_chargemode": "bandwidth",
      "bandwidth_size": 5,
      "bandwidth_sharetype": "PER",
      "eip_type": "5_bgp"
    }'
    kubernetes.io/elb.enterpriseID: 0                # ID of the enterprise
project to which the load balancer belongs
    kubernetes.io/elb.lb-algorithm: ROUND_ROBIN      # Load balancer algorithm
  labels:
    app: nginx
  name: nginx
spec:
  ports:
  - name: service0
    port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: nginx
  type: LoadBalancer
```

  Balanceador de carga dedicado:

```
apiVersion: v1
kind: Service
metadata:
  name: nginx
  labels:
    app: nginx
  namespace: default
  annotations:
    kubernetes.io/elb.class: performance
    kubernetes.io/elb.autocreate: '{
      "type": "public",
```



```
      "bandwidth_name": "cce-bandwidth-1626694478577",
      "bandwidth_chargemode": "bandwidth",
      "bandwidth_size": 5,
      "bandwidth_sharetype": "PER",
      "eip_type": "5_bgp",
      "available_zone": [
         ""
      ],
      "l4_flavor_name": "L4_flavor.elb.s1.small"
    }'
    kubernetes.io/elb.enterpriseID: 0                # ID of the enterprise
project to which the load balancer belongs
    kubernetes.io/elb.lb-algorithm: ROUND_ROBIN      # Load balancer algorithm
spec:
  selector:
    app: nginx
  ports:
  - name: cce-service-0
    targetPort: 80
    nodePort: 0
    port: 80
    protocol: TCP
  type: LoadBalancer
```

Sessão persistente

Tabela 7-11 Anotações para sessão persistente

Parâmetro	Tipo	Descrição	Versão do cluster suportada
kubernetes.io/elb.session-affinity-mode	String	A sessão persistente baseada em endereço IP de origem é suportada. Ou seja, as solicitações do mesmo endereço IP sejam encaminhadas para o mesmo servidor back-end. ● Disabling sticky session: não configure este parâmetro. ● Enabling sticky session: defina esse parâmetro como **SOURCE_IP**, indicando que a sessão adesiva é baseada no endereço IP de origem. **NOTA** Quando **kubernetes.io/elb.lb-algorithm** é definido como **SOURCE_IP** (source IP hash), sessão persistente não pode ser ativada.	v1.9 ou mais recente
kubernetes.io/elb.session-affinity-option	**Tabela 7-21**	Tempo limite de sessão persistente expira.	v1.9 ou mais recente

A seguir, mostra como usar as anotações anteriores:

```
apiVersion: v1
kind: Service
```



```
metadata:
  name: nginx
  annotations:
    kubernetes.io/elb.id: <your_elb_id>                          # ELB ID. Replace
it with the actual value.
    kubernetes.io/elb.class: performance                      # Load balancer type
    kubernetes.io/elb.session-affinity-mode: SOURCE_IP           # The sticky
session type is source IP address.
    kubernetes.io/elb.session-affinity-option: '{"persistence_timeout":
"30"}'     # Stickiness duration (min)
spec:
  selector:
     app: nginx
  ports:
  - name: service0
    port: 80
    protocol: TCP
    targetPort: 80
  type: LoadBalancer
```

Verificação de integridade

Tabela 7-12 Anotações para a verificação de integridade

Parâmetro	Tipo	Descrição	Versão do cluster suportada
kubernetes.io/elb.health-check-flag	String	Se deve ativar a verificação de integridade do ELB. ● Enabling health check: deixe este parâmetro em branco ou defina-o como **on**. ● Disabling health check: defina este parâmetro como **off**. Se esse parâmetro estiver ativado, o campo **kubernetes.io/elb.health-check-option** também deve ser especificado ao mesmo tempo.	v1.9 ou mais recente
kubernetes.io/elb.health-check-option	**Tabela 7-19**	Itens de configuração de verificação de integridade do ELB.	v1.9 ou mais recente
kubernetes.io/elb.health-check-options	**Tabela 7-20**	Item de configuração de verificação de integridade do ELB. Cada porta de serviço pode ser configurada separadamente e você pode configurar apenas algumas portas. **NOTA** **kubernetes.io/elb.health-check-option** e **kubernetes.io/elb.health-check-options** não podem ser configurados ao mesmo tempo.	v1.19.16-r5 ou mais recente v1.21.8-r0 ou mais recente v1.23.6-r0 ou mais recente v1.25.2-r0 ou mais recente

- O seguinte mostra como usar **kubernetes.io/elb.health-check-option**:

```
apiVersion: v1
kind: Service
metadata:
  name: nginx
  annotations:
    kubernetes.io/elb.id: <your_elb_id>                          # ELB ID.
Replace it with the actual value.
    kubernetes.io/elb.class: performance                      # Load balancer
type
    kubernetes.io/elb.health-check-flag: 'on'                    # Enable the
ELB health check function.
    kubernetes.io/elb.health-check-option: '{
      "protocol":"TCP",
      "delay":"5",
      "timeout":"10",
      "max_retries":"3"
    }'
spec:
  selector:
     app: nginx
  ports:
  - name: service0
    port: 80
    protocol: TCP
    targetPort: 80
  type: LoadBalancer
```

- Para obter detalhes sobre como usar **kubernetes.io/elb.health-check-options**, consulte **Configuração da verificação de integridade para várias portas**.

Protocolo HTTP

Tabela 7-13 Anotações para usar protocolos HTTP

Parâmetro	Tipo	Descrição	Versão do cluster suportada
kubernetes.io/elb.protocol-port	String	Porta de configuração de encaminhamento da Camada-7 usada pelo Serviço.	v1.19.16 ou mais recente
kubernetes.io/elb.cert-id	String	Certificado HTTP usado pelo Serviço para encaminhamento de Camada-7.	v1.19.16 ou mais recente

Para obter detalhes sobre os cenários de aplicação, consulte **Serviço usando HTTP**.

Ajuste dinâmico do peso do ECS de back-end

Tabela 7-14 Anotações para ajustar dinamicamente o peso do ECS de back-end

Parâmetro	Tipo	Descrição	Versão do cluster suportada
kubernetes.io/elb.adaptive-weight	String	Ajusta dinamicamente o peso do ECS de back-end do balanceador de carga com base em pods. As solicitações recebidas por cada pod são mais balanceadas. ● **true**: ativado ● **false**: desativado Esse parâmetro se aplica somente a clusters de v1.21 ou posterior e é inválido na rede de passagem.	v1.21 ou mais recente

A seguir, mostra como usar as anotações anteriores:

```
apiVersion: v1
kind: Service
metadata:
  name: nginx
  annotations:
    kubernetes.io/elb.id: <your_elb_id>                          # ELB ID. Replace
it with the actual value.
    kubernetes.io/elb.class: performance                       # Load balancer type
    kubernetes.io/elb.adaptive-weight: 'true'                  # Enable dynamic
adjustment of the weight of the backend ECS.
spec:
  selector:
    app: nginx
  ports:
  - name: service0
    port: 80
    protocol: TCP
    targetPort: 80
  type: LoadBalancer
```

Capacidade de passagem

Tabela 7-15 Anotações para capacidade de passagem

Parâmetro	Tipo	Descrição	Versão do cluster suportada
kubernetes.io/elb.pass-through	String	Se as solicitações de acesso de dentro do cluster para o Serviço passam pelo balanceador de carga do ELB.	v1.19 ou mais recente

Para obter detalhes sobre os cenários de aplicação, consulte **Configuração de rede de passagem para um Serviço de LoadBalancer**.

Lista branca

Tabela 7-16 Anotações para a lista de acesso do ELB

Parâmetro	Tipo	Descrição	Versão do cluster suportada
kubernetes.io/elb.acl-id	String	Esse parâmetro é obrigatório quando você define uma lista branca de endereços IP para um balanceador de carga. O valor desse parâmetro é o ID do grupo de endereços IP do balanceador de carga. Para obter detalhes, consulte **Grupo de endereços IP**. **Esse parâmetro tem efeito somente para balanceadores de carga dedicados e tem efeito somente quando um Serviço é criado ou uma nova porta de serviço (ouvinte) é especificada.** **Como obter**: Faça logon no console. Na **Service List**, escolha **Networking > Elastic Load Balance**. No Console de rede, escolha**Elastic Load Balance > IP Address Groups** e copie o **ID** do grupo de endereços IP de destino.	v1.19.16 v1.21.4
kubernetes.io/elb.acl-status	String	Esse parâmetro é obrigatório quando você define uma lista branca de endereços IP para um balanceador de carga. O valor está **on**, indicando que o controle de acesso está ativado. **Esse parâmetro tem efeito somente para balanceadores de carga dedicados e tem efeito somente quando um Serviço é criado ou uma nova porta de serviço (ouvinte) é especificada.**	v1.19.16 v1.21.4
kubernetes.io/elb.acl-type	String	Esse parâmetro é obrigatório quando você define uma lista branca de endereços IP para um balanceador de carga. ● black: indica a lista negra. O grupo de endereços IP selecionado não pode acessar o endereço do ELB. ● white: indica a lista branca. Somente o grupo de endereços IP selecionado pode acessar o endereço do ELB. **Esse parâmetro tem efeito somente para balanceadores de carga dedicados e tem efeito somente quando um Serviço é criado ou uma nova porta de serviço (ouvinte) é especificada.**	v1.19.16 v1.21.4

A seguir, mostra como usar as anotações anteriores:

```
apiVersion: v1
kind: Service
```



```
metadata:
  name: nginx
  annotations:
    kubernetes.io/elb.id: <your_elb_id>                    # ELB ID. Replace it
with the actual value.
    kubernetes.io/elb.class: performance                   # Load balancer type
    kubernetes.io/elb.acl-id: <your_acl_id>               # ELB IP address group
ID
    kubernetes.io/elb.acl-status: 'on'                    # Enable access control.
    kubernetes.io/elb.acl-type: 'white'                   # Whitelist control
spec:
  selector:
     app: nginx
  ports:
  - name: service0
    port: 80
    protocol: TCP
    targetPort: 80
  type: LoadBalancer
```

Rede host

Tabela 7-17 Anotações para rede host

Parâmetro	Tipo	Descrição	Versão do cluster suportada
kubernetes.io/hws-hostNetwork	String	Se o pod usar **hostNetwork**, o ELB encaminhará a solicitação para a rede host depois que essa anotação for usada. Opções: ● **true**: ativado ● **false** (padrão): desativado	v1.9 ou mais recente

The following shows how to use the preceding annotations:

```
apiVersion: v1
kind: Service
metadata:
  name: nginx
  annotations:
    kubernetes.io/elb.id: <your_elb_id>                          # ELB ID. Replace
it with the actual value.
    kubernetes.io/elb.class: performance                        # Load balancer type
    kubernetes.io/hws-hostNetwork: 'true'                          # The load balancer
forwards the request to the host network.
spec:
  selector:
     app: nginx
  ports:
  - name: service0
    port: 80
    protocol: TCP
    targetPort: 80
  type: LoadBalancer
```


Estrutura de dados

Tabela 7-18 Estrutura de dados de elb.autocreate

Parâmetro	Obrigatório	Tipo	Descrição
name	Não	String	Nome do balanceador de carga criado automaticamente. O valor pode conter de 1 a 64 caracteres. Somente letras, dígitos, sublinhados (_), hifens (-) e pontos (.) são permitidos. Padrão: **cce-lb+service.UID**
type	Não	String	Tipo de rede do balanceador de carga. ● **public**: balanceador de carga de rede pública ● **inner**: balanceador de carga de rede privada Padrão: **inner**
bandwidth_name	Sim para balanceadores de carga de rede pública	String	Nome da largura de banda. O valor padrão é **cce-bandwidth-******. O valor pode conter de 1 a 64 caracteres. Somente letras, dígitos, sublinhados (_), hifens (-) e pontos (.) são permitidos.
bandwidth_chargemode	Não	String	Modo de cobrança de largura de banda. ● **bandwidth**: cobrado pela largura de banda ● **traffic**: cobrado pelo tráfego Padrão: **bandwidth**
bandwidth_size	Sim para balanceadores de carga de rede pública	Integer	Tamanho da largura de banda. O valor padrão é de 1 a 2000 Mbit/s. Configure este parâmetro com base na faixa de largura de banda permitida em sua região. O incremento mínimo para ajuste de largura de banda varia dependendo do intervalo de largura de banda. ● O incremento mínimo é de 1 Mbit/s se a largura de banda permitida não exceder 300 Mbit/s. ● O incremento mínimo é de 50 Mbit/s se a largura de banda permitida varia de 300 Mbit/s a 1000 Mbit/s. ● O incremento mínimo é de 500 Mbit/s se a largura de banda permitida exceder 1000 Mbit/s.

Parâmetro	Obrigatório	Tipo	Descrição
bandwidth_sharetype	Sim para balanceadores de carga de rede pública	String	Modo de compartilhamento de largura de banda. ● **PER**: largura de banda dedicada
eip_type	Sim para balanceadores de carga de rede pública	String	Tipo de EIP. ● **5_telcom**: China Telecom ● **5_union**: China Unicom ● **5_bgp**: BGP dinâmico ● **5_sbgp**: BGP estático
vip_subnet_cidr_id	Não	String	Sub-rede onde o balanceador de carga está localizado. Este campo é suportado por clusters de v1.21 ou posterior. Se esse parâmetro não for especificado, o balanceador de carga do ELB e o cluster estarão na mesma sub-rede.
available_zone	Sim	Array of strings	AZ onde o balanceador de carga está localizado. Você pode obter todas as AZs suportadas **consultando a lista de AZ**. Esse parâmetro está disponível apenas para balanceadores de carga compartilhados.
l4_flavor_name	Sim	String	Nome do flavor do balanceador de carga da camada 4. Você pode obter todos os tipos suportados **consultando a lista de flavors**. Esse parâmetro está disponível apenas para balanceadores de carga compartilhados.
l7_flavor_name	Não	String	Nome do flavor do balanceador de carga da camada-7. Você pode obter todos os tipos suportados **consultando a lista de flavors**. Esse parâmetro está disponível apenas para balanceadores de carga compartilhados. O valor deste parâmetro deve ser o mesmo de **l4_flavor_name**, ou seja, ambos são especificações elásticas ou especificações fixas.

Parâmetro	Obrigatório	Tipo	Descrição
elb_virsubnet_ids	Não	Array of strings	Sub-rede onde o servidor back-end do balanceador de carga está localizado. Se esse parâmetro for deixado em branco, a sub-rede do cluster padrão será usada. Os balanceadores de carga ocupam um número diferente de endereços IP de sub-rede com base em suas especificações. Não use os blocos CIDR de sub-rede de outros recursos (como clusters e nós) como o bloco CIDR do balanceador de carga. Esse parâmetro está disponível apenas para balanceadores de carga compartilhados. Exemplo: `"elb_virsubnet_ids": [ "14567f27-8ae4-42b8-ae47-9f847a4690dd" ]`

Tabela 7-19 Estrutura de dados de elb.health-check-option

Parâmetro	Obrigatório	Tipo	Descrição
delay	Não	String	Tempo de espera inicial (em segundos) para iniciar a verificação de integridade. Intervalo de valores: 1 a 50. Valor padrão: **5**
timeout	Não	String	Tempo limite de verificação de integridade, em segundos. Intervalo de valores: 1 a 50. Valor padrão: **10**
max_retries	Não	String	Número máximo de tentativas de verificação de integridade. Intervalo de valores: 1 a 10. Valor padrão: **3**
protocol	Não	String	Protocolo de verificação de integridade. Opções de valor: TCP ou HTTP
path	Não	String	URL de verificação de integridade. Este parâmetro precisa ser configurado quando o protocolo é **HTTP**. Valor padrão: / O valor pode conter de 1 a 10.000 caracteres.

Tabela 7-20 Descrição da estrutura de dados do campo **elb.health-check-options**

Parâmetro	Obrigatório	Tipo	Descrição
target_service_port	Sim	String	Porta para verificação de integridade especificada por spec.ports. O valor consiste no protocolo e no número da porta, por exemplo, TCP:80.
monitor_port	Não	String	Porta re-especificada para verificação de integridade. Se esse parâmetro não for especificado, a porta de serviço será usada por padrão. **NOTA** Certifique-se de que a porta esteja no estado de escuta no nó onde o pod está localizado. Caso contrário, o resultado da verificação de integridade será afetado.
delay	Não	String	Tempo de espera inicial (em segundos) para iniciar a verificação de integridade. Intervalo de valores: 1 a 50. Valor padrão: **5**
timeout	Não	String	Tempo limite de verificação de integridade, em segundos. Intervalo de valores: 1 a 50. Valor padrão: **10**
max_retries	Não	String	Número máximo de tentativas de verificação de integridade. Intervalo de valores: 1 a 10. Valor padrão: **3**
protocol	Não	String	Protocolo de verificação de integridade. Valor padrão: protocolo do Serviço associado Opções de valor: TCP, UDP ou HTTP
path	Não	String	URL de verificação de saúde. Este parâmetro precisa ser configurado quando o protocolo é HTTP. Valor padrão: / O valor pode conter de 1 a 10.000 caracteres.

Tabela 7-21 Estrutura de dados de elb.session-affinity-option

Parâmetro	Obrigatório	Tipo	Descrição
persistence_timeout	Sim	String	Tempo limite de sessão persistente, em minutos. Esse parâmetro é válido somente quando **elb.session-affinity-mode** é definido como **SOURCE_IP**. Intervalo de valores: 1 a 60. Valor padrão: **60**

7.3.4.3 Serviço usando HTTP

Restrições

- Somente os clusters v1.19.16 ou posterior suportam HTTP.
- Não conecte o ingress e o Serviço que usa HTTP ao mesmo ouvinte do mesmo balanceador de carga. Caso contrário, ocorre um conflito de portas.
- O roteamento de camada-7 do ELB pode ser habilitado para Serviços. Os balanceadores de carga do ELB compartilhados e dedicados podem ser interconectados.

  As restrições em balanceadores de carga do ELB dedicados são as seguintes:

  – Para interconectar-se com um balanceador de carga dedicado existente, o flavor do balanceador de carga **deve suportar o roteamento de camada-4 e camada-7**. Caso contrário, o balanceador de carga não funcionará como esperado.

  – Se você usar um balanceador de carga criado automaticamente, não poderá usar o console do CCE para criar automaticamente um balanceador de carga dedicado à camada 7. Em vez disso, você pode usar o YAML para criar um balanceador de carga dedicado à camada 7, use os recursos da camada 4 e da camada 7 da instância exclusiva do ELB (ou seja, especifique os flavors de camada-4 e camada-7 na anotação de kubernetes.io/elb.autocreate).

Serviço usando HTTP

As seguintes anotações precisam ser adicionadas:

- **kubernetes.io/elb.protocol-port**: "https:443,http:80"

  O valor de **protocol-port** deve ser o mesmo que a porta no campo **spec.ports** do Serviço. O formato é *Protocol:Port*. A porta corresponde à do campo **service.spec.ports** e é liberada como o protocolo correspondente.

- **kubernetes.io/elb.cert-id**: "17e3b4f4bc40471c86741dc3aa211379"

  cert-id indica o ID do certificado no gerenciamento de certificados do ELB. Quando **https** é configurado para **protocol-port**, o certificado do ouvinte do ELB será ajustado ao certificado **cert-id**. Quando vários serviços HTTPS são liberados, o mesmo certificado é usado.

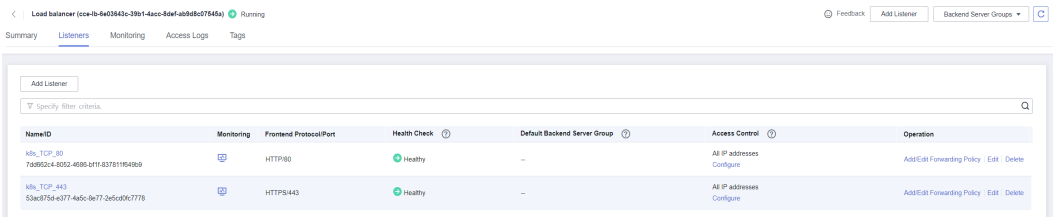
Segue-se um exemplo de configuração. Os dois ports em **spec.ports** correspondem aos de **kubernetes.io/elb.protocol-port**. As portas 443 e 80 estão habilitadas para solicitações HTTPS e HTTP, respectivamente.

```
apiVersion: v1
kind: Service
```



```
metadata:
  annotations:
    # When an ELB load balancer is automatically created, both layer-4 and
layer-7 flavors need to be specified.
    kubernetes.io/elb.autocreate: '
      {
          "type": "public",
          "bandwidth_name": "cce-bandwidth-1634816602057",
          "bandwidth_chargemode": "bandwidth",
          "bandwidth_size": 5,
          "bandwidth_sharetype": "PER",
          "eip_type": "5_bgp",
          "available_zone": [
              ""
          ],
          "l7_flavor_name": "L7_flavor.elb.s2.small"
      }'
    kubernetes.io/elb.class: performance
    kubernetes.io/elb.protocol-port: "https:443,http:80"
    kubernetes.io/elb.cert-id: "17e3b4f4bc40471c86741dc3aa211379"
  labels:
    app: nginx
    name: test
  name: test
  namespace: default
spec:
  ports:
  - name: cce-service-0
    port: 443
    protocol: TCP
    targetPort: 80
  - name: cce-service-1
    port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: nginx
    version: v1
  sessionAffinity: None
  type: LoadBalancer
```

Use as configurações de exemplo anteriores para criar um Serviço. No novo balanceador de carga do ELB, você pode ver que os ouvintes nas portas 443 e 80 são criados.



7.3.4.4 Configuração da verificação de integridade para várias portas

O campo de anotação relacionado à verificação de integridade do Serviço LoadBalancer é atualizado de **Kubernetes.io/elb.health-check-option** para **Kubernetes.io/elb.health-check-options**. Cada porta de Serviço pode ser configurada separadamente e você pode configurar apenas algumas portas. Se o protocolo de porta não precisar ser configurado separadamente, o campo de anotação original ainda estará disponível e não precisará ser modificado.

Restrições

- Este recurso entra em vigor apenas nas seguintes versões:

  - v1.19: v1.19.16-r5 ou mais recente
  - v1.21: v1.21.8-r0 ou mais recente
  - v1.23: v1.23.6-r0 ou mais recente
  - v1.25: v1.25.2-r0 ou mais recente
- **kubernetes.io/elb.health-check-option** e **kubernetes.io/elb.health-check-options** não podem ser configurados ao mesmo tempo.
- O campo **target_service_port** é obrigatório e deve ser exclusivo.
- Para uma porta TCP, o protocolo de verificação de integridade só pode ser TCP ou HTTP. Para uma porta UDP, o protocolo de verificação de integridade deve ser UDP.

Procedimento

Veja a seguir um exemplo de uso da anotação **kubernetes.io/elb.health-check-options**:

```
apiVersion: v1
kind: Service
metadata:
  name: nginx
  namespace: default
  labels:
    app: nginx
    version: v1
  annotations:
    kubernetes.io/elb.class: union         # Load balancer type
    kubernetes.io/elb.id: <your_elb_id>    # ELB ID. Replace it with the actual
value.
    kubernetes.io/elb.lb-algorithm: ROUND_ROBIN  # Load balancer algorithm
    kubernetes.io/elb.health-check-flag: 'on'    # Enable ELB health check.
    kubernetes.io/elb.health-check-options: '[
    {
        "protocol": "TCP",
        "delay": "5",
        "timeout": "10",
        "max_retries": "3",
        "target_service_port": "TCP:1", // (Mandatory) Port for health check
specified by spec.ports. The value consists of the protocol and port number, for
example, TCP:80.
        "monitor_port": "22" // (Optional) Re-specified port for health check. If
this parameter is not specified, the service port is used by default. Ensure that
the port is in the listening state on the node where the pod is located.
Otherwise, the health check result will be affected.
    },
    {
        "protocol": "HTTP",
        "delay": "5",
        "timeout": "10",
        "max_retries": "3",
        "path": "/",  // Health check URL. This parameter needs to be configured
when HTTP is used.
        "target_service_port": "TCP:2",
        "monitor_port": "22"
    }
    ]'
spec:
  selector:
    app: nginx
    version: v1
  externalTrafficPolicy: Cluster
  ports:
    - name: cce-service-0
      targetPort: 1
      nodePort: 0
      port: 1
      protocol: TCP
```



```
  - name: cce-service-1
    targetPort: 2
    nodePort: 0
    port: 2
    protocol: TCP
type: LoadBalancer
loadBalancerIP: **.**.**.**
```

Tabela 7-22 Descrição da estrutura de dados do campo **elb.health-check-options**

Parâmetro	Obrigatório	Tipo	Descrição
target_service_port	Sim	String	Porta para verificação de integridade especificada por spec.ports. O valor consiste no protocolo e no número da porta, por exemplo, TCP:80.
monitor_port	Não	String	Porta re-especificada para verificação de integridade. Se esse parâmetro não for especificado, a porta de serviço será usada por padrão. **NOTA** Certifique-se de que a porta esteja no estado de escuta no nó onde o pod está localizado. Caso contrário, o resultado da verificação de integridade será afetado.
delay	Não	String	Tempo de espera inicial (em segundos) para iniciar a verificação de integridade. Intervalo de valores: 1 a 50. Valor padrão: **5**
timeout	Não	String	Tempo limite de verificação de integridade, em segundos. Intervalo de valores: 1 a 50. Valor padrão: **10**
max_retries	Não	String	Número máximo de tentativas de verificação de integridade. Intervalo de valores: 1 a 10. Valor padrão: **3**
protocol	Não	String	Protocolo de verificação de integridade. Valor padrão: protocolo do Serviço associado Opções de valor: TCP, UDP ou HTTP
path	Não	String	URL de verificação de saúde. Este parâmetro precisa ser configurado quando o protocolo é HTTP. Valor padrão: / O valor pode conter de 1 a 10.000 caracteres.

7.3.4.5 Configuração do status pronto para o pod por meio da verificação de integridade do ELB

O status pronto do pod está associado à verificação de integridade do ELB. Depois que a verificação de integridade for bem-sucedida, o pod está pronto. Essa associação funciona com os parâmetros **strategy.rollingUpdate.maxSurge** e **strategy.rollingUpdate.maxUnavailable** do pod para implementar a atualização contínua graciosa.

Restrições

- Este recurso entra em vigor apenas nas seguintes versões:
  - v1.19: v1.19.16-r5 ou mais recente
  - v1.21: v1.21.8-r0 ou mais recente
  - v1.23: v1.23.6-r0 ou mais recente
  - v1.25: v1.25.2-r0 ou mais recente
- Essa função se aplica somente a cenários de passagem, ou seja, cenários em que balanceadores de carga dedicados são usados em clusters do CCE Turbo.
- Para usar essa função, configure o campo readinessGates no pod e especifique o rótulo **target-health.elb.k8s.cce/***{serviceName}*, onde *{serviceName}* indica o nome do serviço.
- O status pronto para pods só entra em vigor quando o back-end do ELB é conectado inicialmente. O status de verificação de integridade subsequente não afeta o status pronto para o pod.

Configuração do status pronto para o pod através da verificação de integridade do ELB

Para usar os Portões de prontidão do pod, execute as seguintes etapas:

Passo 1 Efetue logon no console do CCE e acesse o console do cluster.

Passo 2 No painel de navegação, escolha **Workloads**. No canto superior direito, clique em **Create from YAML**.

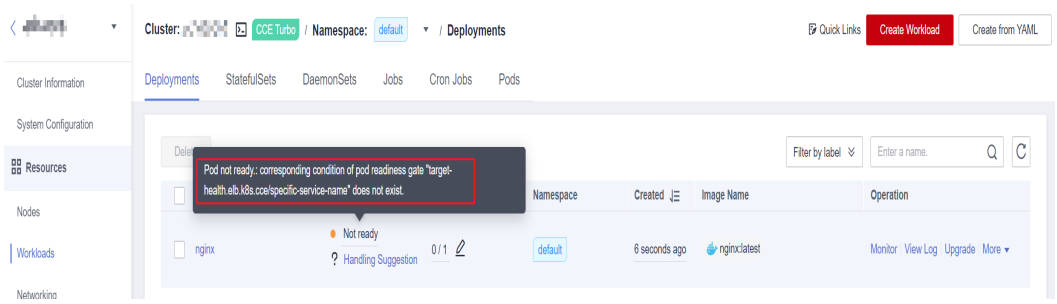
Conteúdo de YAML:

```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: nginx
  namespace: default
  labels:
    version: v1
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
      version: v1
  template:
    metadata:
      labels:
        app: nginx
        version: v1
    spec:
      containers:
        - name: container-1
          image: nginx:latest
```



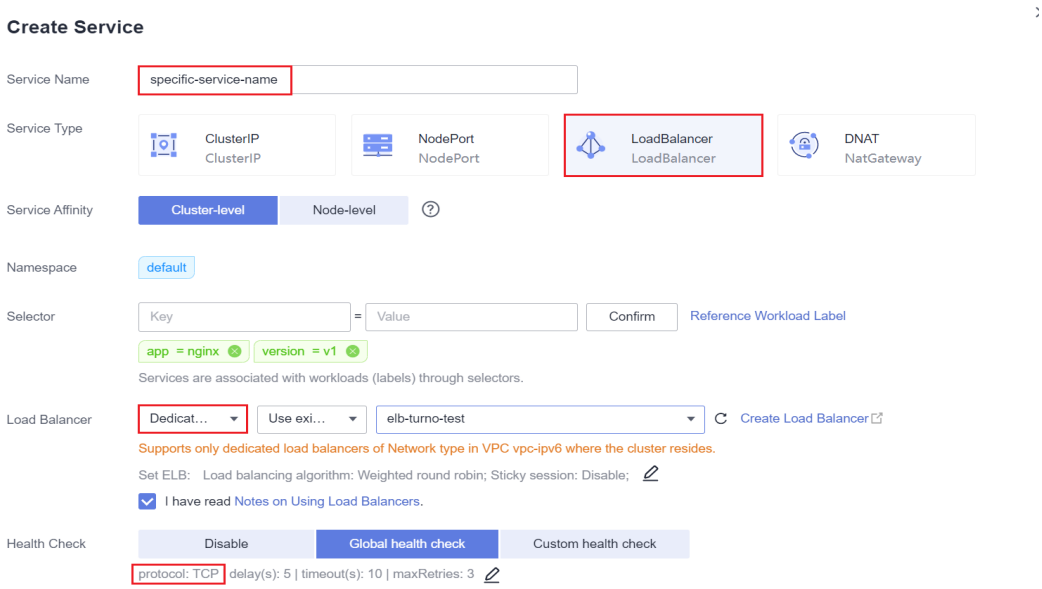
```
      imagePullSecrets:
        - name: default-secret
      readinessGates:
        - conditionType: target-health.elb.k8s.cce/specific-service-name  #
Specifies the ServiceName.
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxUnavailable: 25%           # Works with the following two parameters to
control the number of ELB backends and implement graceful rolling upgrade.
      maxSurge: 25%
```

Passo 3 Clique em **OK**. Na lista de cargas de trabalho, você pode verificar o status da carga de trabalho e descobrir que o pod não está pronto.



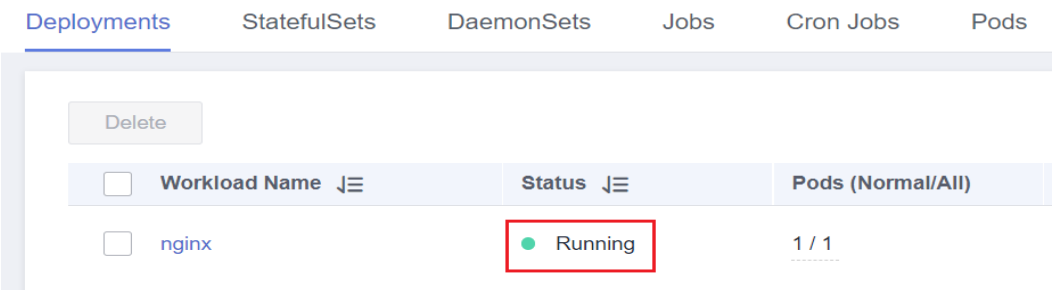
Passo 4 No painel de navegação, escolha **Networking**. No canto superior direito, clique em **Create Service** e defina os seguintes parâmetros:

- **Service Name**: o valor deve ser o mesmo que o valor de **readinessGates** no pod.
- **Service Type**: selecione **LoadBalancer**.
- **Selector**: clique em **Reference Workload Label**, selecione a carga de trabalho criada na etapa anterior e clique em **OK**.
- **Load Balancer**: devem ser usados balanceadores de carga dedicados. Você pode selecionar um balanceador de carga existente ou criar automaticamente um balanceador de carga.
- **Set ELB**: ative a verificação de integridade. (Caso contrário, a verificação de integridade é bem-sucedida por padrão.)



Passo 5 Vá para o console do ELB e verifique o grupo de servidores de back-end. O status de verificação de integridade é normal.

Passo 6 No console do CCE, a carga de trabalho está no status **Running**.



----Fim

7.3.4.6 Configuração do tempo limite para um Serviço LoadBalancer

Os Serviços LoadBalancer permitem que você configure o tempo limite, que é a duração máxima para manter uma conexão se nenhuma solicitação for recebida do cliente. Se não houver solicitações chegando ao balanceador de carga após o término do tempo limite, o balanceador de carga desconectará a conexão com o cliente e estabelecerá uma nova conexão quando houver uma nova solicitação.

Restrições

- Este recurso entra em vigor apenas nas seguintes versões:
  - v1.19: v1.19.16-r30 ou mais recente
  - v1.21: v1.21.10-r10 ou mais recente
  - v1.23: v1.23.8-r10 ou mais recente
  - v1.25: v1.25.3-r10 ou mais recente
- O tempo limite pode ser configurado apenas para os Serviços LoadBalancer usando balanceadores de carga dedicados.
- Se você deletar a configuração de tempo limite durante a atualização do Serviço, a configuração de tempo limite nos ouvintes existentes será mantida.

Procedimento

Use anotações para configurar o tempo limite. O seguinte mostra um exemplo:

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    kubernetes.io/elb.id: <your_elb_id>    # In this example, an existing
dedicated load balancer is used. Replace its ID with the ID of your dedicated
load balancer.
    kubernetes.io/elb.class: performance  # Load balancer type
    kubernetes.io/elb.keepalive_timeout: 300  # Timeout setting for client
connections
  name: nginx
spec:
  ports:
  - name: service0
    port: 80
    protocol: TCP
```



```
    targetPort: 80
  selector:
    app: nginx
  type: LoadBalancer
```

Tabela 7-23 Parâmetros de anotações principais

Parâmetro	Obrigatório	Tipo	Descrição
kubernetes.io/elb.keepalive_timeout	Não	Integer	Tempo limite para conexões de cliente. Se não houver solicitações chegando ao balanceador de carga após o término do tempo limite, o balanceador de carga desconectará a conexão com o cliente e estabelecerá uma nova conexão quando houver uma nova solicitação. Valor: ● Para ouvintes TCP, o valor varia de **10** a **4000**. O valor padrão é **300**. ● Para ouvintes HTTP, HTTPS e TERMINATED_HTTPS, o valor varia de **10** a **4000**. O valor padrão é **60**. ● Para ouvintes UDP, o valor varia de **10** a **4000**. O valor padrão é **300**.

7.3.4.7 Configuração de rede de passagem para um Serviço de LoadBalancer

Cenários de aplicações

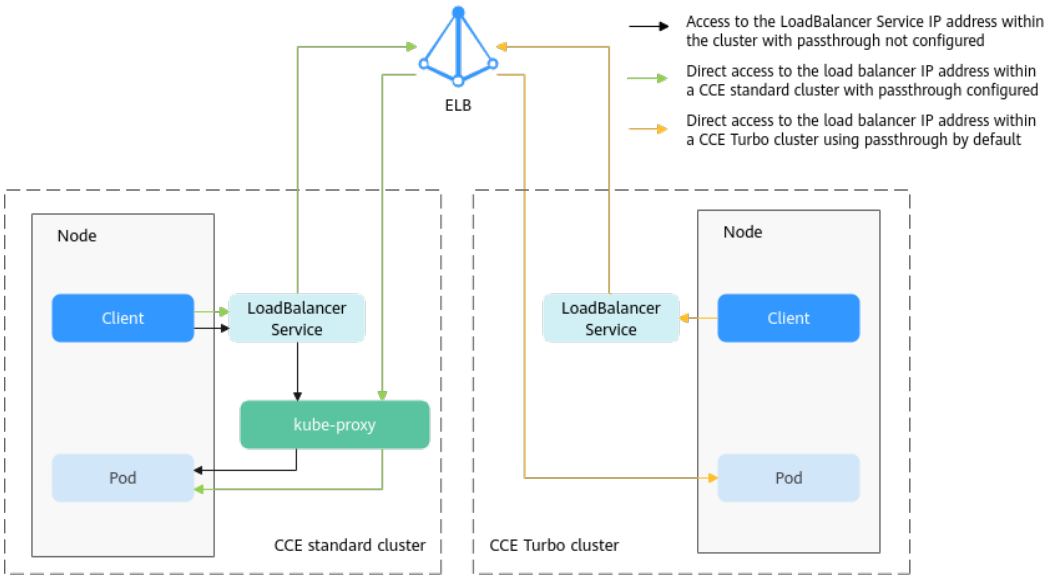
O kube-proxy, que é responsável pelo encaminhamento do tráfego intracluster, adiciona os endereços IP dos balanceadores de carga associados aos Serviços de LoadBalancer às regras de encaminhamento local dos nós por padrão. Quando um cliente de dentro de um cluster acessa o endereço IP de um balanceador de carga, o tráfego é encaminhado diretamente para o destino, em vez de ser encaminhado pelo balanceador de carga.

Se a afinidade em nível de nó estiver configurada para um Serviço (com **externalTrafficPolicy** definido como **Local**), o Serviço encaminhará o tráfego apenas para pods no nó que executa esses pods. Quando um nó ou pod acessa outro pod no mesmo cluster, se o nó em que o cliente é executado não tiver o pod de back-end correspondente, o acesso poderá falhar.

Solução

O CCE suporta redes de passagem. Você pode configurar a anotação **kubernetes.io/elb.pass-through** para o Serviço de LoadBalancer para que o balanceador de carga encaminhe o acesso intracluster para o endereço IP do balanceador de carga associado ao Serviço para pods de back-end.

Figura 7-25 Ilustração de rede de passagem



- Clusters do CCE

  Quando um Serviço de LoadBalancer é acessado dentro do cluster, o acesso é encaminhado para os pods de back-end usando iptables/IPVS por padrão.

  Quando um Serviço de LoadBalancer (configurado com elb.pass-through) é acessado dentro do cluster, o acesso é encaminhado primeiro para o balanceador de carga, depois para os nós e, finalmente, para os pods de back-end usando iptables/IPVS.

- Clusters do CCE Turbo

  Quando um cliente acessa um Serviço de LoadBalancer de dentro do cluster, a passagem é usada por padrão. Nesse caso, o cliente acessa diretamente o endereço IP da rede privada do balanceador de carga e, em seguida, acessa um contêiner por meio do balanceador de carga.

Restrições

- Em um cluster padrão do CCE, depois que a rede de passagem é configurada para um balanceador de carga dedicado, o endereço IP privado do balanceador de carga não pode ser acessado do nó onde o pod de carga de trabalho reside ou de outros contêineres no mesmo nó da carga de trabalho.
- A rede de passagem não é suportada para clusters de v1.15 ou anterior.
- No modo de rede IPVS, as configurações de passagem dos Serviços conectados ao mesmo balanceador de carga devem ser as mesmas.
- Se a afinidade de serviço em nível de nó (local) for usada, **kubernetes.io/elb.pass-through** será automaticamente definido como **onlyLocal** para ativar a passagem.

Procedimento

Esta seção descreve como criar um Deployment usando uma imagem do Nginx e criar um Serviço com a rede de passagem ativada.

Passo 1 Use a ferramenta de linha de comando kubectl para se conectar ao cluster. Para obter detalhes, consulte **Conexão a um cluster usando o kubectl**.

Passo 2 Use a imagem do Nginx para criar um Deployment.

Crie um arquivo **nginx-deployment.yaml**. O conteúdo do arquivo é o seguinte:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - image: nginx:latest
        name: container-0
        resources:
          limits:
            cpu: 100m
            memory: 200Mi
          requests:
            cpu: 100m
            memory: 200Mi
      imagePullSecrets:
      - name: default-secret
```

Execute o seguinte comando para implementar a carga de trabalho:

```
kubectl create -f nginx-deployment.yaml
```

Passo 3 Crie um Serviço de LoadBalancer e defina **kubernetes.io/elb.pass-through** como **true**. Para obter detalhes sobre como criar o Serviço de LoadBalancer, consulte **LoadBalancer**.

O conteúdo do arquivo **nginx-elb-svc.yaml** é o seguinte. (Neste exemplo, um balanceador de carga compartilhado chamado **james** é criado automaticamente.)

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    kubernetes.io/elb.pass-through: "true"
    kubernetes.io/elb.class: union
    kubernetes.io/elb.autocreate: '{"type":"public","bandwidth_name":"cce-
bandwidth","bandwidth_chargemode":"bandwidth","bandwidth_size":5,"bandwidth_sharet
ype":"PER","eip_type":"5_bgp","name":"james"}'
  labels:
    app: nginx
  name: nginx
spec:
  externalTrafficPolicy: Local
  ports:
  - name: service0
    port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: nginx
  type: LoadBalancer
```

Passo 4 Execute o seguinte comando para criar o Serviço:

```
kubectl create -f nginx-elb-svc.yaml
```

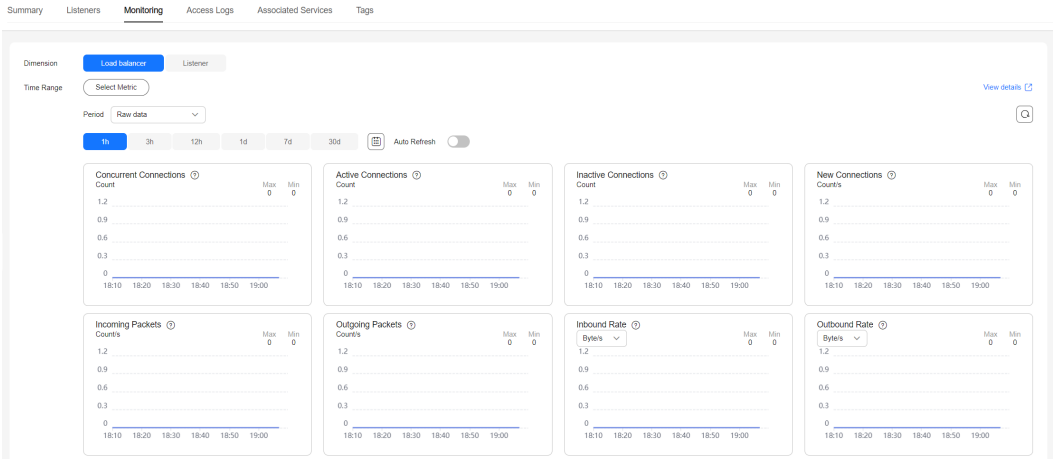
----Fim

Verificação

Passo 1 Faça logon no console do ELB e verifique o balanceador de carga (chamado **james** neste exemplo) associado ao Serviço.

Passo 2 Clique no nome do balanceador de carga e clique na guia **Monitoring**.

Há 0 conexões com o balanceador de carga.



Passo 3 Faça logon em um contêiner de Nginx no cluster usando kubectl e acesse o endereço IP do balanceador de carga.

1. Obtenha os contêineres de Nginx no cluster.
   ```
   kubectl get pod
   ```
   Informação semelhante à seguinte é exibida:
   ```
   NAME                     READY   STATUS    RESTARTS   AGE
   nginx-7c4c5cc6b5-vpncx   1/1     Running   0          9m47s
   nginx-7c4c5cc6b5-xj5wl   1/1     Running   0          9m47s
   ```
2. Efetue logon em um contêiner do Nginx.
   ```
   kubectl exec -it nginx-7c4c5cc6b5-vpncx -- /bin/sh
   ```
3. Acesse o endereço IP do balanceador de carga.
   ```
   curl **.**.**.**
   ```

Passo 4 Aguarde um pouco e verifique os dados de monitoramento no console do ELB.

Se uma nova conexão de acesso for exibida, o acesso será encaminhado pelo balanceador de carga conforme o esperado.

----Fim

7.3.4.8 Ativação de regras de grupo de segurança ICMP

Cenário

Se uma carga de trabalho usar UDP para balanceamento de carga e verificação de integridade, ative as regras do grupo de segurança ICMP para os servidores de back-end.

Procedimento

Passo 1 Faça logon no console do ECS, localize o ECS correspondente a qualquer nó em que a carga de trabalho seja executada e clique no nome do ECS. Na página de detalhes do ECS exibida, registre o nome do grupo de segurança.

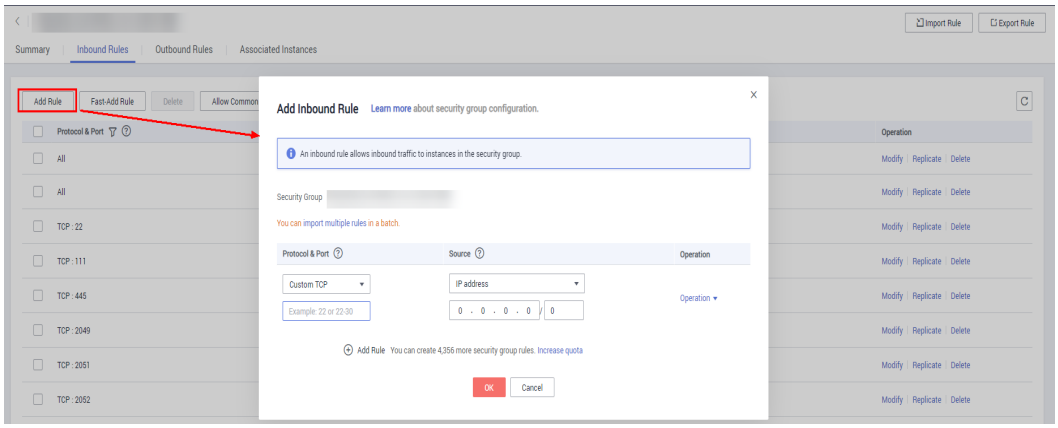
Passo 2 Efetue logon no console da VPC. No painel de navegação à esquerda, escolha **Access Control > Security Groups**. Na lista de grupo de segurança à direita, clique no nome do grupo de segurança obtido na etapa 1.

Passo 3 Na página exibida, clique na guia **Inbound Rules** e clique em **Add Rule** para adicionar uma regra de entrada para o ECS. Em seguida, clique em **OK**. Para obter detalhes sobre como adicionar uma regra de entrada para ECS, consulte **Figura 7-26**.

NOTA

- Você só precisa adicionar regras de grupo de segurança a qualquer nó onde a carga de trabalho é executada.
- O grupo de segurança deve ter regras para permitir o acesso do bloco CIDR 100.125.0.0/16.

Figura 7-26 Adição de uma regra de grupo de segurança



----Fim

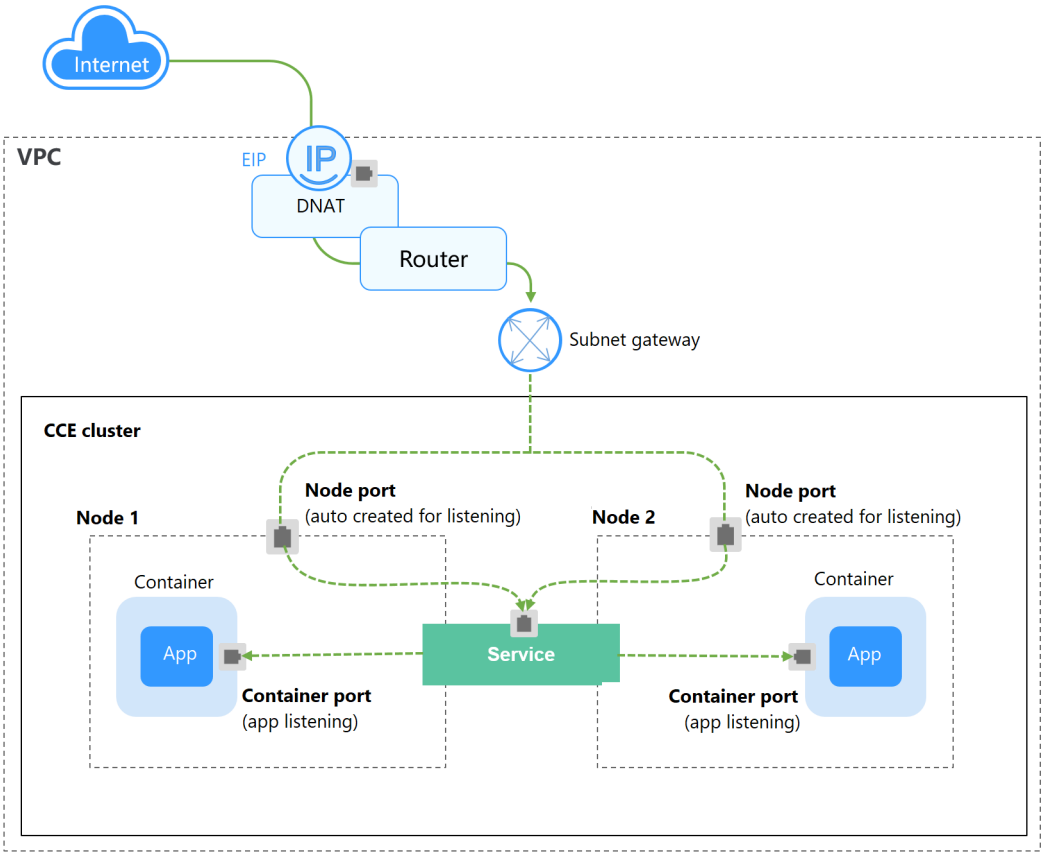
7.3.5 DNAT

Cenário

Um **gateway de conversão de endereços de rede de destino (DNAT)** está situado entre nós de cluster e redes públicas e é atribuído um EIP. Depois de receber solicitações de entrada de redes públicas, o gateway NAT traduz o EIP (endereço de destino nas solicitações de entrada) em um endereço interno de cluster. Parece aos usuários da carga de trabalho como se todos os nós que executam a carga de trabalho compartilhassem o mesmo EIP.

DNAT fornecem maior confiabilidade do que NodePort baseado em EIP, nos quais o EIP está vinculado a um único nó e, quando o nó estiver desativado, todas as solicitações de entrada para a carga de trabalho serão distribuídas. O endereço de acesso está no formato de <EIP>:<access port>, por exemplo, 10.117.117.117:80.

Figura 7-27 DNAT

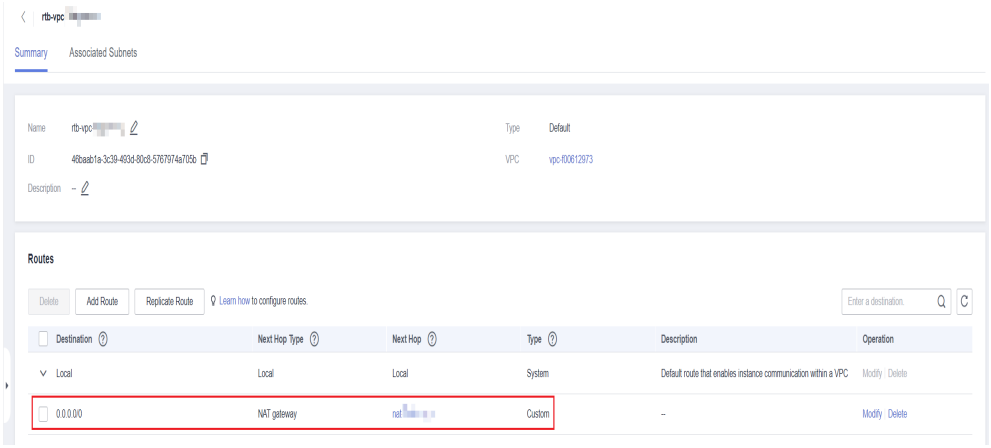


Restrições

Observe as seguintes restrições ao usar o serviço NAT Gateway:

- Os clusters que usam o modelo de rede VPC não permitem que os contêineres acessem os Serviços de DNAT cujo **externalTrafficPolicy** esteja definido como **local**.
- Várias regras para um gateway NAT podem usar o mesmo EIP, mas as regras para diferentes gateways NAT devem usar diferentes EIPs.
- Cada VPC pode ter apenas um gateway NAT.
- Os usuários não podem adicionar manualmente a rota padrão em uma VPC.
- Somente uma regra de SNAT pode ser adicionada a uma sub-rede em uma VPC.
- As regras de SNAT e de DNAT são projetadas para diferentes funções. Se as regras de SNAT e de DNAT usarem o mesmo EIP, a preempção de recursos ocorrerá. Uma regra de SNAT não pode compartilhar um EIP com uma regra de DNAT com **Port Type** definido como **All ports**.
- As regras de DNAT não suportam a vinculação de um EIP a um endereço IP virtual.
- Quando os serviços EIP e NAT Gateway estiverem configurados para um servidor, os dados serão encaminhados por meio do EIP.
- O bloco CIDR personalizado deve ser um subconjunto dos blocos CIDR da sub-rede de VPC.
- O bloco CIDR personalizado deve ser um bloco CIDR da Direct Connect e não pode entrar em conflito com os blocos CIDR de sub-rede existentes da VPC.

- Ao executar operações em recursos subjacentes de um ECS, por exemplo, alterar suas especificações, as regras de gateway NAT configuradas se tornarão inválidas. Exclua as regras e as reconfigure.
- Depois que um serviço é criado, se a configuração de afinidade for alternada do nível de cluster para o nível de nó, a tabela de rastreamento de conexão não será limpa. Recomendamos que você não modifique a configuração de afinidade de Serviço após a criação do Serviço. Para modificá-lo, crie um Serviço novamente.
- Se a sub-rede de nó estiver associada a uma tabela de rota personalizada, adicione a rota de NAT à tabela de rota personalizada ao usar o Serviço DNAT.



Criar um gateway NAT e um endereço IP elástico

Você criou um gateway NAT e um endereço IP elástico. O procedimento específico é o seguinte:

Passo 1 Faça logon no console de gerenciamento, escolha **Networking** > **NAT Gateway** na lista de serviços e clique em **Buy Public NAT Gateway** no canto superior direito. Configure parâmetros com base nos requisitos do site.

NOTA

Ao comprar um gateway NAT, verifique se o gateway NAT pertence à mesma VPC e sub-rede que o cluster do CCE em que a carga de trabalho está em execução.

Passo 2 Faça logon no console de gerenciamento, escolha **Networking** > **Elastic IP** na lista de serviços e clique em **Buy EIP** no canto superior direito. Configure parâmetros com base nos requisitos do site.

----Fim

Criar um Serviço de gateway DNAT

Passo 1 Efetue logon no console do CCE e acesse o console do cluster.

Passo 2 Escolha **Networking** no painel de navegação e clique em **Create Service** no canto superior direito.

Passo 3 Defina parâmetros relacionados.

- **Service Name**: especifique um nome do Serviço, que pode ser o mesmo que o nome da carga de trabalho.

- **Service Type**: selecione **DNAT**.
- **Namespace**: namespace ao qual a carga de trabalho pertence.
- **Service Affinity**: para mais detalhes, consulte **externalTrafficPolicy (afinidade de serviço)**.
  - **Cluster level**: os endereços IP e as portas de acesso de todos os nós em um cluster podem acessar a carga de trabalho associada ao Serviço. O acesso ao Serviço causará perda de desempenho devido ao redirecionamento de rota e o endereço IP de origem do cliente não pode ser obtido.
  - **Node level**: somente o endereço IP e a porta de acesso do nó onde a carga de trabalho está localizada podem acessar a carga de trabalho associada ao Serviço. O acesso ao Serviço não causará perda de desempenho devido ao redirecionamento de rota e o endereço IP de origem do cliente pode ser obtido.
- **Selector**: adicione um rótulo e clique em **Confirm**. Um Serviço seleciona um pod com base no rótulo adicionado. Você também pode clicar em **Reference Workload Label** para fazer referência ao rótulo de uma carga de trabalho existente. Na caixa de diálogo exibida, selecione uma carga de trabalho e clique em **OK**.
- **IPv6**: esta função está desativada por padrão. Depois que essa função é ativada, o endereço IP do cluster do Serviço muda para um endereço IPv6. Para obter detalhes, consulte **Criação de um cluster IPv4/IPv6 de pilha dupla no CCE**. **Este parâmetro está disponível apenas em clusters de v1.15 ou posterior com IPv6 habilitado (definido durante a criação do cluster).**
- **DNAT**: selecione o gateway DNAT e o EIP criados em **Criar um gateway NAT e um endereço IP elástico**.
- **Port**
  - **Protocol**: protocolo utilizado pelo Serviço.
  - **Container Port**: porta na qual a carga de trabalho escuta. A carga de trabalho do Nginx escuta na porta 80.
  - **Service Port**: uma porta mapeada para a porta do contêiner no endereço IP interno do cluster. A carga de trabalho pode ser acessada em <cluster-internal IP address>:<access port>. O intervalo de números de porta é de 1 a 65535.

Passo 4 Clique em **OK**.

----Fim

Configurar o tipo de acesso usando o kubectl

Você pode definir o Serviço ao criar uma carga de trabalho usando o kubectl. Esta seção usa uma carga de trabalho do Nginx como exemplo para descrever como implementar o acesso dentro do cluster usando kubectl.

Passo 1 Use o kubectl para se conectar ao cluster. Para mais detalhes, consulte **Conexão a um cluster usando o kubectl**.

Passo 2 Crie e edite os arquivos **nginx-deployment.yaml** e **nginx-nat-svc.yaml**.

Os nomes dos arquivos são definidos pelo usuário. **nginx-deployment.yaml** e **nginx-nat-svc.yaml** são apenas exemplo de nomes de arquivo.

vi nginx-deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
```



```
metadata:
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - image: nginx:latest
        name: nginx
      imagePullSecrets:
      - name: default-secret
```

Para obter descrições dos campos anteriores, consulte **Tabela 5-2**.

vi nginx-nat-svc.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: nginx
  annotations:
    kubernetes.io/elb.class: dnat
    kubernetes.io/natgateway.id: e4a1cfcf-29df-4ab8-a4ea-c05dc860f554
spec:
  loadBalancerIP: 10.78.42.242
  ports:
  - name: service0
    port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: nginx
  type: LoadBalancer
```

Tabela 7-24 Parâmetros principais

Parâmetro	Obrigatório	Tipo	Descrição
kubernetes.io/elb.class	Sim	String	Esse parâmetro é definido como **dnat** para que o CCE possa trabalhar com um gateway NAT e as regras de DNAT possam ser adicionadas.
kubernetes.io/natgateway.id	Sim	String	ID de um gateway NAT.
loadBalancerIP	Sim	String	ID do EIP.
port	Sim	Integer	Porta de acesso definida no console. O valor varia de 1 a 65535.
targetPort	Sim	String	Porta de contêiner definida no console. O valor varia de 1 a 65535.
type	Sim	String	O tipo de serviço de gateway NAT deve ser definido como **LoadBalancer**.

Passo 3 Crie uma carga de trabalho.

kubectl create -f nginx-deployment.yaml

Se forem exibidas informações semelhantes às seguintes, a carga de trabalho está sendo criada.

```
deployment "nginx" created
```

kubectl get po

Se informações semelhantes às seguintes forem exibidas, a carga de trabalho está em execução.

```
NAME                     READY     STATUS              RESTARTS   AGE
nginx-2601814895-sf71t   1/1       Running             0          8s
```

Passo 4 Crie um Serviço.

kubectl create -f nginx-nat-svc.yaml

Se forem apresentadas informações semelhantes às seguintes, o Serviço foi criado.

```
service "nginx-eip" created
```

kubectl get svc

Se as informações a seguir forem exibidas, o Serviço foi definido com êxito e a carga de trabalho está acessível.

```
NAME         TYPE         CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
kubernetes   ClusterIP    10.247.0.1       <none>        443/TCP        3d
nginx-nat    LoadBalancer 10.247.226.2  10.154.74.98   80:30589/TCP  5s
```

Passo 5 Na barra de endereços do seu navegador, digite **10.154.74.98:80** e pressione **Enter**.

Neste exemplo, **10.154.74.98** é o endereço IP elástico e **80** é o número da porta obtido na etapa anterior.

----Fim

7.3.6 Serviços headless

Os tipos de Serviços anteriores permitem acesso de pod interno e externo, mas não os seguintes cenários:

- Acessar todos os pods ao mesmo tempo
- Pods em um Serviço acessar uns aos outros

É aqui que o Serviço headless entra em serviço. Um Serviço headless não cria um endereço IP de cluster, e os registros DNS de todos os pods são retornados durante a consulta. Desta forma, os endereços IP de todos os pods podem ser consultados. **StatefulSets** usam serviços sem cabeça para dar suporte ao acesso mútuo entre os pods.

```
apiVersion: v1
kind: Service       # Object type (Service)
metadata:
  name: nginx-headless
  labels:
    app: nginx
spec:
  ports:
    - name: nginx      #     - name: nginx     # Name of the port for
communication between pods
```



```
    port: 80        # Port number for communication between pods
  selector:
    app: nginx      # Select the pod whose label is app:nginx.
  clusterIP: None   # Set this parameter to None, indicating that a headless
Service is to be created.
```

Execute o seguinte comando para criar um Serviço headless:

```
# kubectl create -f headless.yaml
service/nginx-headless created
```

Depois que o Serviço for criado, você poderá consultar o Serviço.

```
# kubectl get svc
NAME             TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
nginx-headless   ClusterIP   None         <none>        80/TCP    5s
```

Crie um pod para consultar o DNS. Você pode visualizar os registros de todos os pods. Desta forma, todos os pods podem ser acessados.

```
$ kubectl run -i --tty --image tutum/dnsutils dnsutils --restart=Never --
rm /bin/sh
If you do not see a command prompt, try pressing Enter.
/ # nslookup nginx-0.nginx
Server:         10.247.3.10
Address:        10.247.3.10#53
Name:   nginx-0.nginx.default.svc.cluster.local
Address: 172.16.0.31

/ # nslookup nginx-1.nginx
Server:         10.247.3.10
Address:        10.247.3.10#53
Name:   nginx-1.nginx.default.svc.cluster.local
Address: 172.16.0.18

/ # nslookup nginx-2.nginx
Server:         10.247.3.10
Address:        10.247.3.10#53
Name:   nginx-2.nginx.default.svc.cluster.local
Address: 172.16.0.19
```

7.4 Ingresses

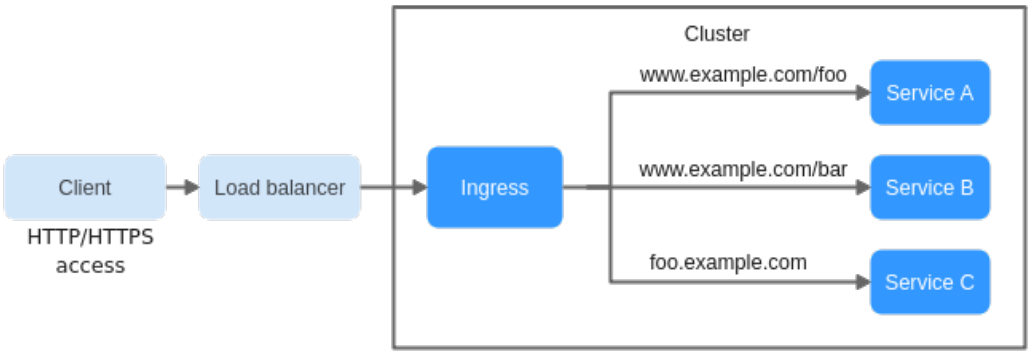
7.4.1 Visão geral

Por que precisamos de ingresses

Um Serviço é geralmente usado para encaminhar solicitações de acesso baseadas em TCP e UDP e fornecer balanceamento de carga de camada 4 para clusters. No entanto, em cenários reais, se houver um grande número de solicitações de acesso HTTP/HTTPS na camada de aplicações, o Serviço não poderá atender aos requisitos de encaminhamento. Portanto, o cluster do Kubernetes fornece um modo de acesso baseado em HTTP, ingress.

Um ingress é um recurso independente no cluster do Kubernetes e define regras para o encaminhamento de tráfego de acesso externo. Conforme mostrado em **Figura 7-28**, você pode personalizar regras de encaminhamento com base em nomes de domínio e URLs para implementar a distribuição refinada do tráfego de acesso.

Figura 7-28 Diagrama de ingress



A seguir, descrevem-se as definições relacionadas ao ingress:

- Ingress object: um conjunto de regras de acesso que encaminham solicitações para serviços específicos com base em nomes de domínio ou URLs. Ele pode ser adicionado, excluído, modificado e consultado chamando APIs.
- Ingress Controller: um executor para encaminhamento de solicitações. Ele monitora as mudanças de objetos de recursos, como ingresses, Serviços, pontos de extremidade, segredos (principalmente certificados e chaves TLS), nós e ConfigMaps em tempo real, analisa regras definidas por ingresses e encaminha solicitações para os Serviços de back-end correspondentes.

Ingress Controllers fornecidos por diferentes fornecedores são implementados de maneiras diferentes. Com base nos tipos de balanceadores de carga, Ingress Controllers são classificados em ELB Ingress Controller e Nginx Ingress Controller. Ambos são suportados no CCE. O ELB Ingress Controller encaminha o tráfego através do ELB. O Nginx Ingress Controller usa os modelos e imagens mantidos pela comunidade Kubernetes para encaminhar o tráfego através do componente do Nginx.

Comparação de recursos de ingress

Tabela 7-25 Comparação entre recursos de ingress

Recurso	ELB Ingress Controller	Nginx Ingress Controller
O&M	Sem O&M	Auto-instalação, atualização e manutenção
Desempenho	Um ingress suporta apenas um balanceador de carga.	Vários ingresses suportam um balanceador de carga.
	Balanceadores de carga de nível empresarial são usados para fornecer alto desempenho e alta disponibilidade. O encaminhamento de serviço não é afetado em cenários de atualização e falha.	O desempenho varia dependendo da configuração de recursos dos pods.

Recurso	ELB Ingress Controller	Nginx Ingress Controller
	O carregamento dinâmico é suportado.	• Os processos devem ser recarregados para alterações de ponto de extremidade não back-end, o que causa perda de conexões persistentes. • Lua suporta atualização a quente de alterações de ponto de extremidade. • Processos devem ser recarregados para uma modificação de Lua.
Implementação de componentes	Implementado no nó mestre	Implementado em nós de trabalho e custos operacionais necessários para o componente de Nginx
Redirecionamento de rota	Não compatível	Compatível
Configuração SSL	Compatível	Compatível
Usar ingress como um proxy para serviços de back-end	Compatível	Compatível, que pode ser implementado através de backend-protocol: anotações de HTTPS.

O ingress do ELB é essencialmente diferente do ingress do Nginx de código aberto. Portanto, seus tipos de Serviço suportados são diferentes. Para mais detalhes, consulte **Serviços suportados por ingresses**.

O ELB Ingress Controller é implementado em um nó principal. Todas as políticas e comportamentos de encaminhamento são configurados no lado do ELB. Balanceadores de carga fora do cluster podem se conectar a nós no cluster somente por meio do endereço IP da VPC em cenários de rede não passantes. Portanto, o Ingress do ELB oferece suporte apenas aos Serviços NodePort. No entanto, no cenário de rede de passagem (cluster do CCE Turbo + balanceador de carga dedicado), o ELB pode encaminhar diretamente o tráfego para pods no cluster. Neste caso, o ingress só pode interconectar com Serviços ClusterIP.

Nginx Ingress Controller é executado em um cluster e é exposto como um serviço através do NodePort. O tráfego é encaminhado para outros Serviços no cluster por meio do Nginx-ingress. O comportamento de encaminhamento de tráfego e o objeto de encaminhamento estão no cluster. Portanto, ambos os serviços ClusterIP e NodePort são suportados.

Em conclusão, o ELB Ingress usa balanceadores de carga de nível empresarial para encaminhar o tráfego e oferece alto desempenho e estabilidade. O Nginx Ingress Controller é implementado em nós de cluster, que consomem recursos de cluster, mas têm melhor capacidade de configuração.

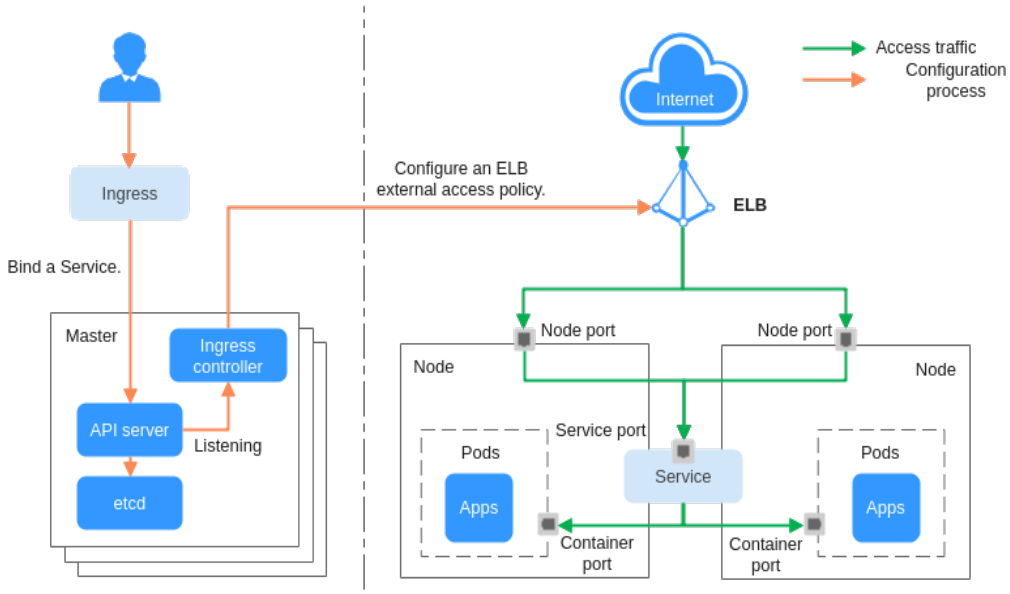
Princípio de funcionamento do ELB Ingress Controller

O ELB Ingress Controller desenvolvido pelo CCE implementa o acesso de rede de camada 7 para a Internet e intranet (na mesma VPC) com base no ELB e distribui o tráfego de acesso aos Serviços correspondentes usando URLs diferentes.

O ELB Ingress Controller é impllementado no nó principal e vinculado ao balanceador de carga na VPC onde o cluster reside. Diferentes nomes de domínio, portas e políticas de encaminhamento podem ser configurados para o mesmo balanceador de carga (com o mesmo endereço IP). **Figura 7-29** mostra o princípio de funcionamento do ELB Ingress Controller.

1. Um usuário cria um objeto de ingress e configura uma regra de acesso de tráfego no ingress, incluindo o balanceador de carga, o URL, o SSL e a porta de serviço de back-end.
2. Quando Ingress Controller detecta que o objeto de ingress muda, reconfigura a rota do servidor do ouvinte e do back-end no lado do ELB de acordo com a regra de acesso de tráfego.
3. Quando um usuário acessa uma carga de trabalho, o tráfego é encaminhado para a porta de serviço de back-end correspondente com base na política de encaminhamento configurada no ELB e, em seguida, encaminhado para cada carga de trabalho associada por meio do Serviço.

Figura 7-29 Princípio de funcionamento do ELB Ingress Controller



Princípio de funcionamento do Nginx Ingress Controller

Um Nginx ingress usa ELB como entrada de tráfego. O complemento **nginx-ingress** é implementado em um cluster para equilibrar o tráfego e controlar o acesso.

NOTA

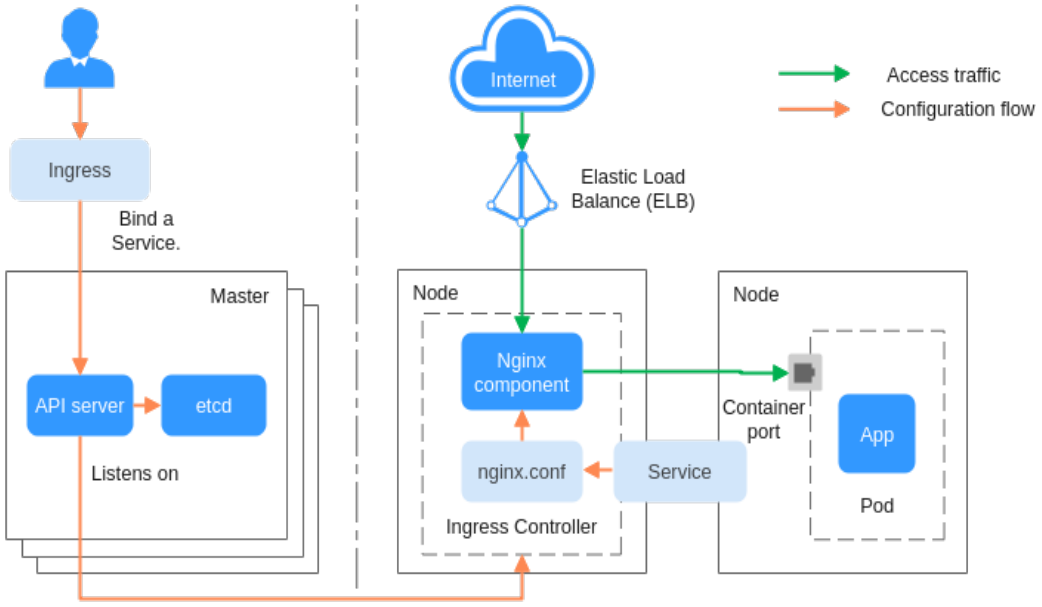
O complemento nginx-ingress no CCE é implementado usando o gráfico e a imagem da comunidade de código aberto. O CCE não mantém o complemento. Portanto, não é recomendável que o complemento nginx-ingress seja usado comercialmente.

Você pode visitar a **comunidade de código aberto** para obter mais informações.

O Nginx Ingress Controller é implementado em nós de trabalho por meio de pods, o que resultará em custos de O&M e sobrecargas de execução de componentes Nginx. **Figura 7-30** mostra os princípios de funcionamento do Nginx Ingress Controller.

1. Depois de atualizar os recursos de entrada, o Nginx Ingress Controller grava uma regra de encaminhamento definida nos recursos de entrada no arquivo de configuração **nginx.conf** do Nginx.
2. O componente interno do Nginx recarrega o arquivo de configuração atualizado para modificar e atualizar a regra de encaminhamento do Nginx.
3. Quando o tráfego acessa um cluster, o tráfego é primeiro encaminhado pelo balanceador de carga criado para o componente do Nginx no cluster. Em seguida, o componente de Nginx encaminha o tráfego para cada carga de trabalho com base na regra de encaminhamento.

Figura 7-30 Princípio de funcionamento do Nginx Ingress Controller



Serviços suportados por ingresses

Tabela 7-26 lista os serviços suportados pelos ingressos do ELB.

Tabela 7-26 Serviços suportados pelo ELB Ingresses

Tipo de cluster	Tipo do ELB	ClusterIP	NodePort
Cluster do CCE	Balanceador de carga compartilhado	Não compatível	Compatível
	Balanceador de carga dedicado	Não suportado (Falha ao acessar os balanceadores de carga dedicados porque nenhuma ENI está vinculada ao pod associado do Serviço ClusterIP.)	Compatível

Tipo de cluster	Tipo do ELB	ClusterIP	NodePort
Cluster do CCE Turbo	Balanceador de carga compartilhado	Não compatível	Compatível
	Balanceador de carga dedicado	Compatível	Não suportado (Falha ao acessar os balanceadores de carga dedicados porque um ENI foi vinculado ao pod associado do Serviço NodePort.)

Tabela 7-27 lista os serviços suportados pelo Nginx Ingresses.

Tabela 7-27 Serviços suportados pelo Nginx Ingresses

Tipo de cluster	Tipo do ELB	ClusterIP	NodePort
Cluster do CCE	Balanceador de carga compartilhado	Compatível	Compatível
	Balanceador de carga dedicado	Compatível	Compatível
Cluster do CCE Turbo	Balanceador de carga compartilhado	Compatível	Compatível
	Balanceador de carga dedicado	Compatível	Compatível

7.4.2 Ingresses do ELB

7.4.2.1 Criação de um ingress do ELB no console

Pré-requisitos

- Um ingress fornece acesso à rede para cargas de trabalho de back-end. Certifique-se de que uma carga de trabalho esteja disponível em um cluster. Se nenhuma carga de trabalho estiver disponível, implemente uma carga de trabalho referindo-se a **Criação de uma Implantação**, **Criação de um StatefulSet** ou **Criação de um DaemonSet**.
- **Serviços suportados por ingresses** lista os tipos de Serviços suportados pelos ingresses do ELB.

Precauções

- Recomenda-se que outros recursos não usem o balanceador de carga criado automaticamente por um ingress. Caso contrário, o balanceador de carga será ocupado quando o ingress for excluído, resultando em recursos residuais.
- Depois que um ingress é criado, atualize e mantenha a configuração dos balanceadores de carga selecionados no console do CCE. Não modifique a configuração no console do ELB. Caso contrário, o serviço de ingress pode ser anormal.

- O URL registrado em uma política de encaminhamento de entrada deve ser o mesmo que o URL usado para acessar o Serviço de back-end. Caso contrário, um erro 404 será retornado.
- Em um cluster usando o modo de proxy IPVS, se o ingress e o Serviço usarem o mesmo balanceador de carga do ELB, o ingress não pode ser acessado dos nós e contêineres no cluster porque o kube-proxy monta o endereço do Serviço LoadBalancer na ponte ipvs-0. Essa ponte intercepta o tráfego do balanceador de carga conectado ao ingress. Use balanceadores de carga do ELB diferentes para o ingress e o Serviço.
- Não conecte o ingress e o **Serviço usando HTTP** ao mesmo ouvinte do mesmo balanceador de carga. Caso contrário, ocorre um conflito de portas.
- Os balanceadores de carga dedicados devem ser do tipo de aplicação (HTTP/HTTPS) que suportam redes privadas (com um IP privado).
- Se vários ingresses forem usadas para se conectar à mesma porta do ELB no mesmo cluster, os itens de configuração do ouvinte (como o certificado associado ao ouvinte e o atributo HTTP2 do ouvinte) estão sujeitos à configuração da primeiro ingress.

Adicionar um ingress do ELB

Esta seção usa uma carga de trabalho de Nginx como um exemplo para descrever como adicionar um ingress do ELB.

Passo 1 Efetue logon no console do CCE e acesse o console do cluster.

Passo 2 Escolha **Networking** no painel de navegação, clique a guia **Ingresses** e clique em **Create Ingress** no canto direito superior.

Passo 3 Configure parâmetros de ingress.

- **Name**: especifica o nome de uma entrada, por exemplo, **ingress-demo**.
- **Interconnect with Nginx**: essa opção é exibida somente após a instalação do complemento **Nginx Ingress controller**. Se essa opção estiver disponível, o complemento nginx-ingress foi instalado. Ative essa opção criará um ingress de Nginx Desative-o se quiser criar um ingress do ELB. Para mais detalhes, consulte **Criação de ingresses de Nginx no console**.
- **Load Balancer**

  Selecione o balanceador de carga para interconexão. Somente balanceadores de carga na mesma VPC que o cluster são compatíveis. Se nenhum balanceador de carga estiver disponível, clique em **Create Load Balancer** para criar um no console do ELB.

  Os balanceadores de carga dedicados devem oferecer suporte a HTTP ou HTTPS e o tipo de rede deve oferecer suporte a redes privadas.

  O console do CCE suporta a criação automática de balanceadores de carga. Selecione **Auto create** na caixa de listagem suspensa e configure os seguintes parâmetros:

  – **Instance Name**: insira o nome de um balanceador de carga.

  – **Public Access**: se ativado, um EIP com largura de banda de 5 Mbit/s será criado. Por padrão, é cobrado pelo tráfego.

  – AZ, **Subnet** e **Specifications** (disponível apenas para balanceadores de carga dedicados): selecione a AZ, sub-rede e as especificações. Somente balanceadores de carga dedicados do tipo de aplicações (HTTP/HTTPS) podem ser criados automaticamente.
- **Listener**: o ingress configura um ouvinte para o balanceador de carga, que atende às solicitações do balanceador de carga e distribui o tráfego. Após a conclusão da

configuração, um ouvinte é criado no balanceador de carga. O nome padrão do ouvinte é *k8s__<Protocol type>_<Port number>*, por exemplo, *k8s_HTTP_80*.

- **Front-End Protocol**: **HTTP** e **HTTPS** estão disponíveis.
- **External Port**: número da porta que está aberta para o endereço de serviço do ELB. O número da porta pode ser especificado aleatoriamente.
- **Certificate Source**: o segredo do TLS e o certificado de servidor ELB são suportados.
- **Server Certificate**: quando um ouvinte HTTPS for criado para um balanceador de carga, vincule um certificado ao balanceador de carga para oferecer suporte à autenticação criptografada para transmissão de dados de HTTPS.
  - **TLS secret**: para obter detalhes sobre como criar um segredo, consulte **Criação de um segredo**.
  - **ELB server certificate**: use o certificado criado no serviço ELB.

  NOTA

  Se já houver um ingress HTTPS para a porta escolhida no balanceador de carga, o certificado do novo ingress HTTPS deve ser igual ao certificado do ingress existente. Isso significa que um ouvinte tem apenas um certificado. Se dois certificados, cada um com um ingress diferente, forem adicionados ao mesmo ouvinte do mesmo balanceador de carga, somente o certificado adicionado mais cedo terá efeito no balanceador de carga.
- **SNI**: o Server Name Indication (SNI) é um protocolo estendido do TLS. Ele permite que vários nomes de domínio de acesso baseados em TLS sejam fornecidos para sistemas externos usando o mesmo endereço IP e porta. Nomes de domínio diferentes podem usar certificados de segurança diferentes. Depois que o SNI é habilitado, o cliente tem permissão para enviar o nome de domínio solicitado ao iniciar uma solicitação de handshake TLS. Depois de receber a solicitação TLS, o balanceador de carga procura o certificado com base no nome de domínio na solicitação. Se o certificado correspondente ao nome de domínio for encontrado, o balanceador de carga retornará o certificado para autorização. Caso contrário, o certificado padrão (certificado de servidor) é retornado para autorização.

  NOTA

  - A opção **SNI** está disponível somente quando **HTTPS** é selecionado.
  - Esta função é suportada apenas em clusters de v1.15.11 e posteriores
  - Especifique o nome de domínio para o certificado de SNI. Apenas um nome de domínio pode ser especificado para cada certificado. Certificados de domínio curinga são suportados.

- **Forwarding Policies**: quando o endereço de acesso de uma solicitação corresponde à política de encaminhamento (uma política de encaminhamento consiste em um nome de domínio e URL, por exemplo, 10.117.117.117:80/helloworld), a solicitação é encaminhada ao Serviço de destino correspondente para processamento. Você pode clicar em + para adicionar várias políticas de encaminhamento.
  - **Domain Name**: nome de domínio atual. Certifique-se de que o nome de domínio tenha sido registrado e arquivado. Depois que uma regra de nome de domínio estiver configurada, você deve usar o nome de domínio para acesso.
  - URL Matching Rule
    - **Prefix match**: se o URL estiver definido como **/healthz**, o URL que atende ao prefixo poderá ser acessado, por exemplo, **/healthz/v1** e **/healthz/v2**.

- **Exact match**: o URL pode ser acessado somente quando é totalmente correspondido. Por exemplo, se o URL estiver definido como **/healthz**, apenas /healthz poderá ser acessado.
- **Regular expression**: o URL é correspondido com base na expressão regular. Por exemplo, se a expressão regular for **/[A-Za-z0-9_.-]+/test**, todos os URLs que estejam em conformidade com essa regra poderão ser acessadas, por exemplo, **/abcA9/test** e **/v1-Ab/test**. Dois padrões de expressão regular são suportados: POSIX e Perl.

– **URL**: caminho de acesso a ser registrado, por exemplo, **/healthz**.

NOTA

O caminho de acesso adicionado aqui deve existir na aplicação back-end. Caso contrário, o encaminhamento falha.

Por exemplo, o URL de acesso padrão da aplicação Nginx é **/usr/share/nginx/html**. Ao adicionar **/test** à política de encaminhamento de ingress, certifique-se de que o URL de acesso da sua aplicação Nginx contenha **/usr/share/nginx/html/test**. Caso contrário, o erro 404 será retornado.

– **Destination Service**: selecione um Serviço existente ou crie um Serviço. Os Serviços que não atendem aos critérios de pesquisa são automaticamente filtrados.

– **Destination Service Port**: selecione a porta de acesso do Serviço de destino.

– **Configure ELB**:

- **Algorithm**: três algoritmos estão disponíveis: weighted round robin, weighted least connections algorithm ou source IP hash.

  NOTA

  - **Weighted round robin**: as solicitações são encaminhadas para servidores diferentes com base em seus pesos, que indicam o desempenho do processamento do servidor. Servidores back-end com pesos mais altos recebem proporcionalmente mais solicitações, enquanto servidores com pesos iguais recebem o mesmo número de solicitações. Este algoritmo é normalmente usado para conexões curtas, como serviços HTTP.
  - **Weighted least connections**: além do peso atribuído a cada servidor, o número de conexões processadas por cada servidor back-end também é considerado. As solicitações são encaminhadas ao servidor com a menor relação entre conexões e peso. Desenvolvido com base no algoritmo de **menos conexões**, o algoritmo **weighted least connections** atribui um peso a cada servidor de acordo com sua capacidade de processamento. Este algoritmo é normalmente usado para conexões persistentes, como conexões de bancos de dados.
  - **Source IP hash**: o endereço IP de origem de cada solicitação é calculado usando-se o algoritmo de hash para obter uma chave de hash única, e todos os servidores back-end são numerados. A chave gerada aloca o cliente a um servidor particular. Isso permite que solicitações de clientes diferentes sejam distribuídas no modo de balanceamento de carga e garante que as solicitações do mesmo cliente sejam encaminhadas para o mesmo servidor. Este algoritmo aplica-se a conexões TCP sem cookies.

- **Sticky Session**: esta função está desativada por padrão. As opções são as seguintes:
  - **Load balancer cookie**: digite a **Stickiness Duration**, que varia de 1 a 1.440 minutos.
  - **Application cookie**: esse parâmetro está disponível apenas para balanceadores de carga compartilhados. Além disso, insira **Cookie Name**, que varia de 1 a 64 caracteres.

NOTA

Quando a **política de distribuição** utiliza o hash do IP de origem, a sessão adesiva não pode ser definida.

- **Health Check**: defina a configuração de verificação de integridade do balanceador de carga. Se esta função estiver ativada, as seguintes configurações são suportadas:

Parâmetro	Descrição
Protocol	Quando o protocolo de porta é definido como **TCP**, TCP e HTTP são suportados. Quando o protocolo é definido como **UDP**, apenas UDP é suportado. ○ **Check Path** (suportado apenas por HTTP para verificação de integridade): especifica o URL de verificação de integridade. O caminho de verificação deve começar com uma barra (/) e pode conter de 1 a 80 caracteres.
Port	Por padrão, a porta de serviço (Porta de nó e porta de contêiner do Serviço) é usada para verificação de integridade. Você também pode especificar uma porta para verificação de integridade. Depois que a porta for especificada, uma porta de serviço chamada **cce-healthz** será adicionada ao Serviço. ○ **Node Port**: se um balanceador de carga compartilhado for usado ou nenhuma instância de ENI estiver associada, a porta do nó será usada como a porta de verificação de integridade. Se este parâmetro não for especificado, uma porta aleatória será usada. O valor varia de 30000 a 32767. ○ **Container Port**: quando um balanceador de carga dedicado é associado a uma instância de ENI, a porta do contêiner é usada para verificação de integridade. O valor varia de 1 a 65535.
Check Period (s)	Especifica o intervalo máximo entre as verificações de integridade. O valor varia de 1 a 50.
Timeout (s)	Especifica a duração máxima do tempo limite para cada verificação de integridade. O valor varia de 1 a 50.
Max. Retries	Especifica o número máximo de tentativas de verificação de integridade. O valor varia de 1 a 10.

  - **Operation**: clique em **Delete** para excluir a configuração.

- **Annotation**: os ingresses fornecem algumas funções avançadas do CCE, que são implementadas por anotações. Quando você usa o kubectl para criar um contêiner, as anotações serão usadas. Para mais detalhes, veja **Criar um ingress - criar automaticamente um balanceador de carga** e **Criar um ingress - interconectarse com um balanceador de carga existente**.

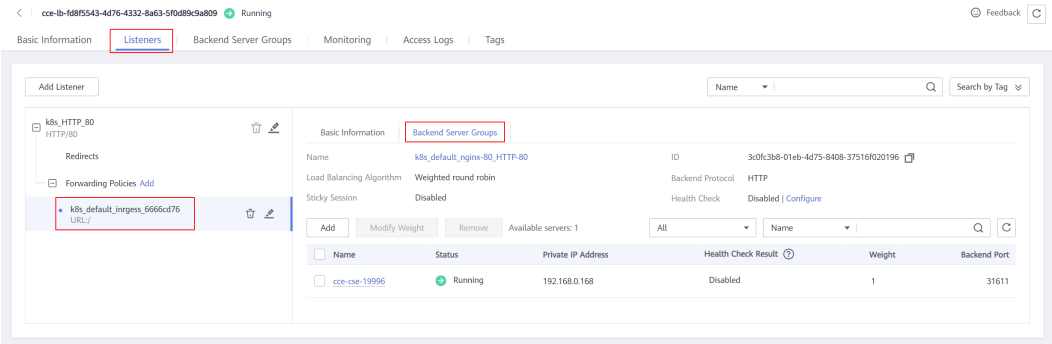
Passo 4 Após a conclusão da configuração, clique em **OK**. Depois que o ingress é criado, ele é exibido na lista de ingresses.

No console do ELB, você pode ver o ELB criado automaticamente por meio do CCE. O nome padrão é **cce-lb-ingress.UID**. Clique no nome do ELB para acessar sua página de detalhes. Na página de guia **Listeners**, exiba as definições de rota do ingresso, incluindo o URL, a porta do ouvinte e a porta do grupo de servidores de back-end.

AVISO

Depois que um ingress for criado, atualize e mantenha o balanceador de carga selecionado no console do CCE. Não modifique a configuração no console do ELB. Caso contrário, o serviço de ingress pode ser anormal.

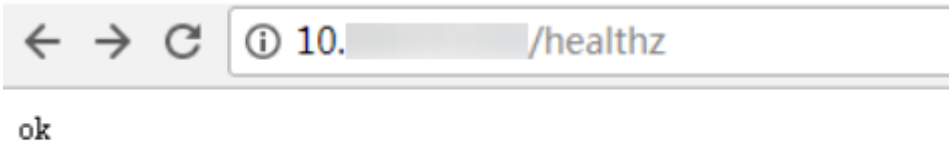
Figura 7-31 Configuração de roteamento do ELB



Passo 5 Acesse a interface /healthz da carga de trabalho, por exemplo, carga de trabalho **defaultbackend**.

1. Obtenha o endereço de acesso da interface **/healthz** da carga de trabalho. O endereço de acesso consiste no endereço IP do balanceador de carga, porta externa e URL de mapeamento, por exemplo, 10.**.**.**:80/healthz.
2. Digite o URL da interface /healthz, por exemplo, http://10.**.**.**:80/healthz, na caixa de endereço do navegador para acessar a carga de trabalho, conforme mostrado em **Figura 7-32**.

Figura 7-32 Acessar a interface /healthz de defaultbackend



----Fim

Operações relacionadas

A estrutura de ingress do Kubernetes não contém o campo **property**. Portanto, o ingress criado pela API chamada pelo client-go não contém o campo **property**. O CCE fornece uma solução para garantir a compatibilidade com o client-go do Kubernetes. Para obter detalhes sobre a solução, consulte **Como obter compatibilidade entre a propriedade do ingress e o cliente-go do Kubernetes?**

7.4.2.2 Uso do kubectl para criar um ingress do ELB

Cenário

Esta seção usa uma **carga de trabalho do Nginx** como exemplo para descrever como criar um ingress de ELB usando o kubectl.

- Se nenhum balanceador de carga estiver disponível na mesma VPC, o CCE poderá criar automaticamente um balanceador de carga ao criar um ingress. Para mais detalhes, consulte **Criar um ingress - criar automaticamente um balanceador de carga**.
- Se um balanceador de carga estiver disponível na mesma VPC, execute a operação fazendo referência a **Criar um ingress - interconectarse com um balanceador de carga existente**.

Pré-requisitos

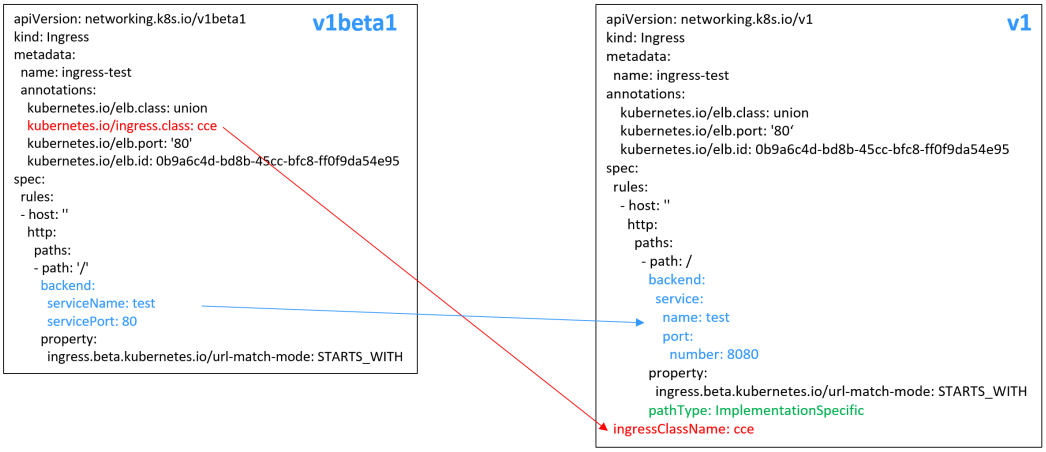
- Um ingress fornece acesso à rede para cargas de trabalho de back-end. Certifique-se de que uma carga de trabalho esteja disponível em um cluster. Se nenhuma carga de trabalho estiver disponível, implemente uma carga de trabalho de amostra do Nginx referindo-se a **Criação de uma Implantação**, **Criação de um StatefulSet** ou **Criação de um DaemonSet**.
- **Serviços suportados por ingresses** lista os Serviços suportados pelos ingresses do ELB.
- Os balanceadores de carga dedicados devem ser do tipo de aplicação (HTTP/HTTPS) que suportam redes privadas (com um IP privado).

Descrição de ingress de networking.k8s.io/v1

Em clusters do CCE de v1.23 ou posterior, a versão de ingress é comutada para **networking.k8s.io/v1**.

Comparada com v1beta1, v1 tem as seguintes diferenças nos parâmetros:

- O tipo de ingress é alterado de **kubernetes.io/ingress.class** em **annotations** para **spec.ingressClassName**.
- O formato do **backend** é alterado.
- O parâmetro **pathType** deve ser especificado para cada caminho. As opções são as seguintes:
  - **ImplementationSpecific**: o método de correspondência depende do Ingress Controller. O método de correspondência definido por **ingress.beta.kubernetes.io/url-match-mode** é usado no CCE, que é o mesmo que v1beta1.
  - **Exact**: correspondência exata do URL, que diferencia maiúsculas de minúsculas.
  - **Prefix**: correspondência com base no prefixo de URL separado por uma barra (/). A correspondência diferencia maiúsculas de minúsculas, e os elementos no caminho são correspondidos um a um. Um elemento path refere-se a uma lista de rótulos no caminho separados por uma barra (/).



Criar um ingress - criar automaticamente um balanceador de carga

A seguir, descrevemos como executar o comando kubectl para criar automaticamente um balanceador de carga ao criar um ingress.

Passo 1 Use o kubectl para se conectar ao cluster. Para mais detalhes, consulte **Conexão a um cluster usando o kubectl**.

Passo 2 Crie um arquivo YAML chamado **ingress-test.yaml**. O nome do arquivo pode ser personalizado.

vi ingress-test.yaml

NOTA

A partir do cluster v1.23, a versão de entrada é alternada de **networking.k8s.io/v1beta1** para **networking.k8s.io/v1**. Para obter detalhes sobre as diferenças entre v1 e v1beta1, consulte **Descrição de ingress de networking.k8s.io/v1**.

Exemplo de um balanceador de carga compartilhado (acesso de rede pública) para clusters da v1.23 ou posterior:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-test
  annotations:
    kubernetes.io/elb.class: union
    kubernetes.io/elb.port: '80'
    kubernetes.io/elb.autocreate:
      '{
          "type":"public",
          "bandwidth_name":"cce-bandwidth-******",
          "bandwidth_chargemode":"bandwidth",
          "bandwidth_size":5,
          "bandwidth_sharetype":"PER",
          "eip_type":"5_bgp"
        }'
 spec:
  rules:
  - host: ''
    http:
      paths:
      - path: '/'
        backend:
          service:
            name: <your_service_name>  # Replace it with the name of your target
Service.
```



```
            port:
              number: <your_service_port>  # Replace it with the port number of
your target Service.
        property:
          ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
        pathType: ImplementationSpecific
  ingressClassName: cce    # ELB ingress is used.
```

Exemplo de um balanceador de carga compartilhado (acesso de rede pública) para clusters da v1.21 ou anterior:

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ingress-test
  annotations:
    kubernetes.io/elb.class: union
    kubernetes.io/ingress.class: cce    # ELB ingress is used.
    kubernetes.io/elb.port: '80'
    kubernetes.io/elb.autocreate:
      '{
          "type":"public",
          "bandwidth_name":"cce-bandwidth-******",
          "bandwidth_chargemode":"bandwidth",
          "bandwidth_size":5,
          "bandwidth_sharetype":"PER",
          "eip_type":"5_bgp"
        }'
 spec:
  rules:
  - host: ''
    http:
      paths:
      - path: '/'
        backend:
          serviceName: <your_service_name>  # Replace it with the name of your
target Service.
          servicePort: <your_service_port>  # Replace it with the port number of
your target Service.
        property:
          ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
```

Exemplo de um balanceador de carga dedicado (acesso de rede pública) para clusters v1.23 ou posterior:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-test
  namespace: default
  annotations:
    kubernetes.io/elb.class: performance
    kubernetes.io/elb.port: '80'
    kubernetes.io/elb.autocreate:
      '{
          "type": "public",
          "bandwidth_name": "cce-bandwidth-******",
          "bandwidth_chargemode": "bandwidth",
          "bandwidth_size": 5,
          "bandwidth_sharetype": "PER",
          "eip_type": "5_bgp",
          "available_zone": [
              "ap-southeast-1a"
          ],
          "elb_virsubnet_ids":["b4bf8152-6c36-4c3b-9f74-2229f8e640c9"],
          "l7_flavor_name": "L7_flavor.elb.s1.small"
        }'
 spec:
  rules:
  - host: ''
    http:
```



```
      paths:
      - path: '/'
        backend:
          service:
            name: <your_service_name>  # Replace it with the name of your target
Service.
            port:
              number: <your_service_port>  # Replace it with the port number of
your target Service.
        property:
          ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
        pathType: ImplementationSpecific
  ingressClassName: cce
```

Exemplo de balanceador de carga dedicado (acesso de rede pública) para clusters da versão 1.21 ou anterior:

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ingress-test
  namespace: default
  annotations:
    kubernetes.io/elb.class: performance
    kubernetes.io/ingress.class: cce
    kubernetes.io/elb.port: '80'
    kubernetes.io/elb.autocreate:
      '{
          "type": "public",
          "bandwidth_name": "cce-bandwidth-******",
          "bandwidth_chargemode": "bandwidth",
          "bandwidth_size": 5,
          "bandwidth_sharetype": "PER",
          "eip_type": "5_bgp",
          "available_zone": [
              "ap-southeast-1a"
          ],
          "elb_virsubnet_ids":["b4bf8152-6c36-4c3b-9f74-2229f8e640c9"],
          "l7_flavor_name": "L7_flavor.elb.s1.small"
       }'
 spec:
  rules:
  - host: ''
    http:
      paths:
      - path: '/'
        backend:
          serviceName: <your_service_name>  # Replace it with the name of your
target Service.
          servicePort: <your_service_port>  # Replace it with the port number of
your target Service.
        property:
          ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
```


Tabela 7-28 Parâmetros principais

Parâmetro	Obrigatório	Tipo	Descrição
kubernetes.io/elb.class	Sim	String	Selecione um tipo adequado de balanceador de carga. O valor pode ser: ● **union**: balanceador de carga compartilhado ● **performance**: balanceador de carga dedicado. Para obter detalhes, consulte **Diferenças entre balanceadores de carga compartilhados e dedicados**. Padrão: **union**
kubernetes.io/ingress.class	Sim (somente para clusters da v1.21 ou anterior)	String	**cce**: o ingress auto-desenvolvido do ELB é usado. Esse parâmetro é obrigatório quando uma entrada é criada chamando a API.
ingressClassName	Sim (somente para clusters da v1.23 ou posterior)	String	**cce**: o ingress auto-desenvolvido do ELB é usado. Esse parâmetro é obrigatório quando um ingress é criado chamando a API.
kubernetes.io/elb.port	Sim	Integer	Este parâmetro indica a porta externa registrada com o endereço do Serviço LoadBalancer. Intervalo suportado: 1 a 65535 **NOTA** Algumas portas são portas de alto risco e são bloqueadas por padrão, por exemplo, a porta 21.
kubernetes.io/elb.subnet-id	Nenhum	String	ID da sub-rede onde o cluster está localizado. O valor pode conter de 1 a 100 caracteres. ● Obrigatório quando um cluster v1.11.7-r0 ou anterior deve ser criado automaticamente. ● Opcional para clusters posteriores à v1.11.7-r0. É deixado em branco por padrão. Para obter detalhes sobre como obter o valor, consulte**Qual é a diferença entre a API de sub-rede da VPC e a API de sub-rede do OpenStack Neutron**.

Parâmetro	Obrigatório	Tipo	Descrição
kubernetes.io/elb.enterpriseID	Não	String	**Os clusters do Kubernetes v1.15 e versões posteriores oferecem suporte a esse campo. Nos clusters do Kubernetes anteriores à v1.15, os balanceadores de carga são criados no projeto padrão por padrão.** ID do projeto empresarial no qual o balanceador de carga será criado. O valor contém de 1 a 100 caracteres. **Como obter**: Faça logon no console de gerenciamento e escolha **Enterprise** > **Project Management** na barra de menu superior. Na lista exibida, clique no nome do projeto empresarial de destino e copie o ID na página de detalhes do projeto empresarial.
kubernetes.io/elb.autocreate	Sim	elb.autocreate object	Se criar automaticamente um balanceador de carga associado a um ingress. Para obter detalhes sobre a descrição do campo, consulte **Tabela 7-29**. **Exemplo** ● Se um balanceador de carga de rede pública for criado automaticamente, defina esse parâmetro com o seguinte valor: {"type":"public","bandwidth_name":"cce-bandwidth-******","bandwidth_chargemode":"bandwidth","bandwidth_size":5,"bandwidth_sharetype":"PER","eip_type":"5_bgp","name":"james"} ● Se um balanceador de carga de rede privada for criado automaticamente, defina esse parâmetro com o seguinte valor: {"type":"inner","name":"A-location-d-test"}
host	Não	String	Nome de domínio para acessar o Serviço. Por padrão, esse parâmetro é deixado em branco e o nome de domínio precisa ser totalmente correspondido. Certifique-se de que o nome de domínio tenha sido registrado e arquivado. Depois que uma regra de nome de domínio estiver configurada, você deve usar o nome de domínio para acesso.

Parâmetro	Obrigatório	Tipo	Descrição
path	Sim	String	Caminho de rota definido pelo usuário. Todas as solicitações de acesso externo devem corresponder ao **host** e ao **path**. **NOTA** O caminho de acesso adicionado aqui deve existir na aplicação back-end. Caso contrário, o encaminhamento falha. Por exemplo, o URL de acesso padrão da aplicação Nginx é **/usr/share/nginx/html**. Ao adicionar **/test** à política de encaminhamento de ingress, certifique-se de que o URL de acesso da sua aplicação Nginx contenha **/usr/share/nginx/html/test**. Caso contrário, o erro 404 será retornado.
ingress.beta.kubernetes.io/url-match-mode	Não	String	Política de correspondência de rotas. Padrão: **STARTS_WITH** (correspondência de prefixo) Opções: ● **EQUAL_TO**: correspondência exata ● **STARTS_WITH**: correspondência de prefixo ● **REGEX**: correspondência de expressão regular

Parâmetro	Obrigatório	Tipo	Descrição
pathType	Sim	String	Tipo de caminho. Este campo é suportado apenas por clusters de v1.23 ou posterior. ● **ImplementationSpecific**: o método de correspondência depende do Ingress Controller. O método de correspondência definido por **ingress.beta.kubernetes.io/url-match-mode** é usado no CCE. ● **Exact**: correspondência exata do URL, que diferencia maiúsculas de minúsculas. ● **Prefix**: correspondência de prefixo, que diferencia maiúsculas de minúsculas. Com esse método, o caminho do URL é separado em vários elementos por barras (/) e os elementos são correspondidos um por um. Se cada elemento no URL corresponder ao caminho, os subcaminhos do URL poderão ser roteados normalmente. **NOTA** – Durante a correspondência de prefixo, cada elemento deve ser exatamente correspondido. Se o último elemento do URL for a subcadeia do último elemento no caminho da solicitação, nenhuma correspondência será executada. Por exemplo, **/foo/bar** corresponde a **/foo/bar/baz**, mas não corresponde a **/foo/barbaz**. – Quando elementos são separados por barras (/), se o URL ou o caminho da requisição termina com uma barra (/), a barra (/) no final é ignorada. Por exemplo, **/foo/bar** corresponde a **/foo/bar/**. Veja **exemplos** de correspondência de caminho de ingress.

Tabela 7-29 Estrutura de dados de elb.autocreate

Parâmetro	Obrigatório	Tipo	Descrição
name	Não	String	Nome do balanceador de carga criado automaticamente. O valor pode conter de 1 a 64 caracteres. Somente letras, dígitos, sublinhados (_), hifens (-) e pontos (.) são permitidos. Padrão: **cce-lb+service.UID**

Parâmetro	Obrigatório	Tipo	Descrição
type	Não	String	Tipo de rede do balanceador de carga. ● **public**: balanceador de carga de rede pública ● **inner**: balanceador de carga de rede privada Padrão: **inner**
bandwidth_name	Sim para balanceadores de carga de rede pública	String	Nome da largura de banda. O valor padrão é **cce-bandwidth-******. O valor pode conter de 1 a 64 caracteres. Somente letras, dígitos, sublinhados (_), hifens (-) e pontos (.) são permitidos.
bandwidth_chargemode	Não	String	Modo de cobrança de largura de banda. ● **bandwidth**: cobrado pela largura de banda ● **traffic**: cobrado pelo tráfego Padrão: **bandwidth**
bandwidth_size	Sim para balanceadores de carga de rede pública	Integer	Tamanho da largura de banda. O valor padrão é de 1 a 2000 Mbit/s. Configure este parâmetro com base na faixa de largura de banda permitida em sua região. O incremento mínimo para ajuste de largura de banda varia dependendo do intervalo de largura de banda. ● O incremento mínimo é de 1 Mbit/s se a largura de banda permitida não exceder 300 Mbit/s. ● O incremento mínimo é de 50 Mbit/s se a largura de banda permitida varia de 300 Mbit/s a 1000 Mbit/s. ● O incremento mínimo é de 500 Mbit/s se a largura de banda permitida exceder 1000 Mbit/s.
bandwidth_sharetype	Sim para balanceadores de carga de rede pública	String	Modo de compartilhamento de largura de banda. ● **PER**: largura de banda dedicada
eip_type	Sim para balanceadores de carga de rede pública	String	Tipo de EIP. ● **5_telcom**: China Telecom ● **5_union**: China Unicom ● **5_bgp**: BGP dinâmico ● **5_sbgp**: BGP estático

Parâmetro	Obrigatório	Tipo	Descrição
vip_subnet_cidr_id	Não	String	Sub-rede onde o balanceador de carga está localizado. Este campo é suportado por clusters de v1.21 ou posterior. Se esse parâmetro não for especificado, o balanceador de carga do ELB e o cluster estarão na mesma sub-rede.
available_zone	Sim	Array of strings	AZ onde o balanceador de carga está localizado. Você pode obter todas as AZs suportadas **consultando a lista de AZ**. Esse parâmetro está disponível apenas para balanceadores de carga compartilhados.
l4_flavor_name	Sim	String	Nome do flavor do balanceador de carga da camada 4. Você pode obter todos os tipos suportados **consultando a lista de flavors**. Esse parâmetro está disponível apenas para balanceadores de carga compartilhados.
l7_flavor_name	Não	String	Nome do flavor do balanceador de carga da camada-7. Você pode obter todos os tipos suportados **consultando a lista de flavors**. Esse parâmetro está disponível apenas para balanceadores de carga compartilhados. O valor deste parâmetro deve ser o mesmo de **l4_flavor_name**, ou seja, ambos são especificações elásticas ou especificações fixas.
elb_virsubnet_ids	Não	Array of strings	Sub-rede onde o servidor back-end do balanceador de carga está localizado. Se esse parâmetro for deixado em branco, a sub-rede do cluster padrão será usada. Os balanceadores de carga ocupam um número diferente de endereços IP de sub-rede com base em suas especificações. Não use os blocos CIDR de sub-rede de outros recursos (como clusters e nós) como o bloco CIDR do balanceador de carga. Esse parâmetro está disponível apenas para balanceadores de carga compartilhados. Exemplo: `"elb_virsubnet_ids": [ "14567f27-8ae4-42b8-ae47-9f847a4690dd" ]`

Passo 3 Crie um ingress.

kubectl create -f ingress-test.yaml

Se forem exibidas informações semelhantes às seguintes, o ingress foi criado.

```
ingress/ingress-test created
```

kubectl get ingress

Se informações semelhantes às seguintes forem exibidas, o ingress foi criado com êxito e a carga de trabalho está acessível.

```
NAME             HOSTS     ADDRESS          PORTS   AGE
ingress-test     *         121.**.**.**     80      10s
```

Passo 4 Digite **http://*121.**.**.***:80** na caixa de endereço do navegador para acessar a carga de trabalho (por exemplo, **carga de trabalho do Nginx**).

121.**.**.** indica o endereço IP do balanceador de carga unificado.

----Fim

Criar um ingress - interconectarse com um balanceador de carga existente

O CCE permite que você se conecte a um balanceador de carga existente ao criar um ingress.

NOTA

- Os balanceadores de carga dedicados existentes devem ser do tipo de aplicação (HTTP/HTTPS) que suportam redes privadas (com um IP privado).

Se a versão do cluster for 1.23 ou posterior, a configuração do arquivo YAML será a seguinte:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-test
  annotations:
    kubernetes.io/elb.id: <your_elb_id>  # Replace it with the ID of your
existing load balancer.
    kubernetes.io/elb.ip: <your_elb_ip>  # Replace it with the IP of your
existing load balancer.
    kubernetes.io/elb.class: performance  # Load balancer type
    kubernetes.io/elb.port: '80'
spec:
  rules:
  - host: ''
    http:
      paths:
      - path: '/'
        backend:
          service:
            name: <your_service_name>  # Replace it with the name of your target
Service.
            port:
              number: 8080             # Replace 8080 with the port number of
your target Service.
        property:
          ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
        pathType: ImplementationSpecific
  ingressClassName: cce
```

Se a versão do cluster for 1.21 ou anterior, a configuração do arquivo YAML será a seguinte:

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
```



```
metadata:
  name: ingress-test
  annotations:
    kubernetes.io/elb.id: <your_elb_id>  # Replace it with the ID of your
existing load balancer.
    kubernetes.io/elb.ip: <your_elb_ip>  # Replace it with the IP of your
existing load balancer.
    kubernetes.io/elb.class: performance  # Load balancer type
    kubernetes.io/elb.port: '80'
    kubernetes.io/ingress.class: cce
spec:
  rules:
  - host: ''
    http:
      paths:
      - path: '/'
        backend:
          serviceName: <your_service_name>  # Replace it with the name of your
target Service.
          servicePort: 80
        property:
          ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
```

Tabela 7-30 Parâmetros principais

Parâmetro	Obrigatório	Tipo	Descrição
kubernetes.io/elb.id	Sim	String	ID de um balanceador de carga. O valor pode conter de 1 a 100 caracteres. **Como obter**: No console de gerenciamento, clique em **Service List** e escolha **Networking** > **Elastic Load Balance**. Clique no nome do balanceador de carga de destino. Na página de guia **Summary**, localize e copie o ID.
kubernetes.io/elb.ip	Não	String	Endereço de serviço de um balanceador de carga. O valor pode ser o endereço IP público de um balanceador de carga de rede público ou o endereço IP privado de um balanceador de carga de rede privada.
kubernetes.io/elb.class	Sim	String	Tipo do balanceador de carga. ● **union**: balanceador de carga compartilhado ● **performance**: balanceador de carga dedicado, que pode ser usado apenas em clusters de v1.17 e posteriores. Para obter detalhes, consulte **Diferenças entre balanceadores de carga compartilhado e dedicado**. **NOTA** Se um Ingress do ELB acessar um balanceador de carga dedicado existente, o balanceador de carga dedicado deve ser do tipo de balanceamento de carga de aplicação (HTTP/HTTPS).

7.4.2.3 Configuração de ingresses do ELB usando anotações

Adicionando anotações a um arquivo YAML, você pode implementar funções de ingress mais avançadas. Esta seção descreve as anotações que podem ser usadas quando você cria um ingress do tipo do ELB.

- **Interconexão com o ELB**
- **Usar o HTTP/2**
- **Interconexão com serviços de back-end HTTPS**

Interconexão com o ELB

Tabela 7-31 Anotações para interconexão com o ELB

Parâmetro	Tipo	Descrição	Versão do cluster suportada
kubernetes.io/elb.class	String	Selecione um tipo adequado de balanceador de carga. O valor pode ser: ● **union**: balanceador de carga compartilhado ● **performance**: balanceador de carga dedicado, que pode ser usado apenas em clusters de v1.17 e posteriores. Para obter detalhes, consulte **Diferenças entre balanceadores de carga compartilhado e dedicado**.	v1.9 ou mais recente
kubernetes.io/ingress.class	String	● **cce**: o ingress autodesenvolvido do ELB é usado. ● **nginx**: ingress de Nginx é usado. Esse parâmetro é obrigatório quando um ingress é criado chamando a API. Para clusters v1.23 ou posterior, use o parâmetro **ingressClassName**. Para mais detalhes, consulte **Uso do kubectl para criar um ingress do ELB**.	Somente clusters da v1.21 ou anterior
kubernetes.io/elb.port	Integer	Este parâmetro indica a porta externa registrada com o endereço do Serviço LoadBalancer. Intervalo suportado: 1 a 65535 **NOTA** Algumas portas são portas de alto risco e são bloqueadas por padrão, por exemplo, a porta 21.	v1.9 ou mais recente

Parâmetro	Tipo	Descrição	Versão do cluster suportada
kubernetes.io/elb.id	String	Obrigatório **quando um balanceador de carga existente deve ser interconectado**. ID de um balanceador de carga. **Como obter**: No console de gerenciamento, clique em **Service List** e escolha **Networking** > **Elastic Load Balance**. Clique no nome do balanceador de carga de destino. Na página de guia **Summary**, localize e copie o ID.	v1.9 ou mais recente
kubernetes.io/elb.ip	String	Obrigatório **quando um balanceador de carga existente deve ser interconectado**. Este parâmetro indica o endereço de serviço de um balanceador de carga. O valor pode ser o endereço IP público de um balanceador de carga de rede público ou o endereço IP privado de um balanceador de carga de rede privada.	v1.9 ou mais recente
kubernetes.io/elb.autocreate	**Tabela 7-34** Object	Obrigatório **quando os balanceadores de carga são criados automaticamente**. **Exemplo** ● Se um balanceador de carga de rede pública for criado automaticamente, defina esse parâmetro com o seguinte valor: '{"type":"public","bandwidth_name":"cce-bandwidth-1551163379627","bandwidth_chargemode":"bandwidth","bandwidth_size":5,"bandwidth_sharetype":"PER","eip_type":"5_bgp","name":"james"}' ● Se um balanceador de carga de rede privada for criado automaticamente, defina esse parâmetro com o seguinte valor: {"type":"inner","name":"A-location-d-test"}	v1.9 ou mais recente

Parâmetro	Tipo	Descrição	Versão do cluster suportada
kubernetes.io/elb.enterpriseID	String	Opcional **quando balanceadores de carga são criados automaticamente**. **Os clusters da v1.15 e posterior oferecem suporte a esse campo. Nos clusters anteriores à v1.15, os balanceadores de carga são criados no projeto padrão por padrão.** Esse parâmetro indica o ID do projeto empresarial no qual o balanceador de carga do ELB será criado. Se esse parâmetro não for especificado ou for definido como **0**, os recursos serão vinculados ao projeto empresarial padrão. **Como obter**: Faça logon no console de gerenciamento e escolha **Enterprise** > **Project Management** na barra de menu superior. Na lista exibida, clique no nome do projeto empresarial de destino e copie o ID na página de detalhes do projeto empresarial.	v1.15 ou mais recente
kubernetes.io/elb.subnet-id	String	Opcional **quando balanceadores de carga são criados automaticamente**. ID da sub-rede onde o cluster está localizado. O valor pode conter de 1 a 100 caracteres. ● Obrigatório quando um cluster v1.11.7-r0 ou anterior deve ser criado automaticamente. ● Opcional para clusters posteriores à v1.11.7-r0.	Obrigatório para clusters anteriores à v1.11.7-r0 Descartado em clusters posteriores à v1.11.7-r0

Para usar as anotações anteriores, execute as seguintes etapas:

- Consulte **Criar um ingress - interconectarse com um balanceador de carga existente** para interconectar um balanceador de carga existente.
- Consulte **Criar um ingress - criar automaticamente um balanceador de carga** para criar automaticamente um balanceador de carga.

Usar o HTTP/2

Tabela 7-32 Anotações do uso do HTTP/2

Parâmetro	Tipo	Descrição	Versão do cluster suportada
kubernetes.io/elb.http2-enable	String	Se o HTTP/2 está ativado. O encaminhamento de solicitações usando HTTP/2 melhora o desempenho de acesso entre sua aplicação e o balanceador de carga. No entanto, o balanceador de carga ainda usa HTTP 1.X para encaminhar solicitações para o servidor back-end. **Este parâmetro é suportado em clusters de v1.19.16-r0, v1.21.3-r0 e posterior.** Opções: ● **true**: ativado ● **false**: desativado (valor padrão) Observação: **o HTTP/2 pode ser ativado ou desativado somente quando o ouvinte usa HTTPS.** Esse parâmetro é inválido e assume como padrão **false** quando o protocolo de ouvinte é HTTP.	v1.19.16-r0, v1.21.3-r0 ou mais recente

Para obter detalhes sobre os cenários de aplicação, consulte **Ingresses do ELB usando HTTP/2**.

Interconexão com serviços de back-end HTTPS

Tabela 7-33 Anotações para interconexão com serviços de back-end HTTPS

Parâmetro	Tipo	Descrição	Versão do cluster suportada
kubernetes.io/elb.pool-protocol	String	Para interconectar com serviços de back-end HTTPS, defina esse parâmetro como **https**.	v1.23.8, v1.25.3 ou mais recente

Para obter detalhes sobre os cenários de aplicação, consulte **Interconexão de ingresses do ELB com serviços de back-end HTTPS**.

Estrutura de dados

Tabela 7-34 Estrutura de dados de elb.autocreate

Parâmetro	Obrigatório	Tipo	Descrição
name	Não	String	Nome do balanceador de carga criado automaticamente. O valor pode conter de 1 a 64 caracteres. Somente letras, dígitos, sublinhados (_), hifens (-) e pontos (.) são permitidos. Padrão: **cce-lb+service.UID**
type	Não	String	Tipo de rede do balanceador de carga. ● **public**: balanceador de carga de rede pública ● **inner**: balanceador de carga de rede privada Padrão: **inner**
bandwidth_name	Sim para balanceadores de carga de rede pública	String	Nome da largura de banda. O valor padrão é **cce-bandwidth-******. O valor pode conter de 1 a 64 caracteres. Somente letras, dígitos, sublinhados (_), hifens (-) e pontos (.) são permitidos.
bandwidth_chargemode	Não	String	Modo de cobrança de largura de banda. ● **bandwidth**: cobrado pela largura de banda ● **traffic**: cobrado pelo tráfego Padrão: **bandwidth**
bandwidth_size	Sim para balanceadores de carga de rede pública	Integer	Tamanho da largura de banda. O valor padrão é de 1 a 2000 Mbit/s. Configure este parâmetro com base na faixa de largura de banda permitida em sua região. O incremento mínimo para ajuste de largura de banda varia dependendo do intervalo de largura de banda. ● O incremento mínimo é de 1 Mbit/s se a largura de banda permitida não exceder 300 Mbit/s. ● O incremento mínimo é de 50 Mbit/s se a largura de banda permitida varia de 300 Mbit/s a 1000 Mbit/s. ● O incremento mínimo é de 500 Mbit/s se a largura de banda permitida exceder 1000 Mbit/s.

Parâmetro	Obrigatório	Tipo	Descrição
bandwidth_sharetype	Sim para balanceadores de carga de rede pública	String	Modo de compartilhamento de largura de banda. ● **PER**: largura de banda dedicada
eip_type	Sim para balanceadores de carga de rede pública	String	Tipo de EIP. ● **5_telcom**: China Telecom ● **5_union**: China Unicom ● **5_bgp**: BGP dinâmico ● **5_sbgp**: BGP estático
vip_subnet_cidr_id	Não	String	Sub-rede onde o balanceador de carga está localizado. Este campo é suportado por clusters de v1.21 ou posterior. Se esse parâmetro não for especificado, o balanceador de carga do ELB e o cluster estarão na mesma sub-rede.
available_zone	Sim	Array of strings	AZ onde o balanceador de carga está localizado. Você pode obter todas as AZs suportadas **consultando a lista de AZ**. Esse parâmetro está disponível apenas para balanceadores de carga compartilhados.
l4_flavor_name	Sim	String	Nome do flavor do balanceador de carga da camada 4. Você pode obter todos os tipos suportados **consultando a lista de flavors**. Esse parâmetro está disponível apenas para balanceadores de carga compartilhados.
l7_flavor_name	Não	String	Nome do flavor do balanceador de carga da camada-7. Você pode obter todos os tipos suportados **consultando a lista de flavors**. Esse parâmetro está disponível apenas para balanceadores de carga compartilhados. O valor deste parâmetro deve ser o mesmo de **l4_flavor_name**, ou seja, ambos são especificações elásticas ou especificações fixas.

Parâmetro	Obrigatório	Tipo	Descrição
elb_virsubnet_ids	Não	Array of strings	Sub-rede onde o servidor back-end do balanceador de carga está localizado. Se esse parâmetro for deixado em branco, a sub-rede do cluster padrão será usada. Os balanceadores de carga ocupam um número diferente de endereços IP de sub-rede com base em suas especificações. Não use os blocos CIDR de sub-rede de outros recursos (como clusters e nós) como o bloco CIDR do balanceador de carga. Esse parâmetro está disponível apenas para balanceadores de carga compartilhados. Exemplo: `"elb_virsubnet_ids": [ "14567f27-8ae4-42b8-ae47-9f847a4690dd" ]`

7.4.2.4 Configuração de certificados HTTPS para ingresses do ELB

O ingress oferece suporte à configuração de certificados do TLS e protege seus Serviços com HTTPS.

Atualmente, você pode usar o certificado do segredo do TLS configurado no cluster e o certificado do ELB.

NOTA

Se o HTTPS estiver habilitado para a mesma porta do mesmo balanceador de carga de várias ingresses, você deverá selecionar o mesmo certificado.

Usar um certificado do segredo do TLS

Passo 1 Use o kubectl para se conectar ao cluster. Para mais detalhes, consulte **Conexão a um cluster usando o kubectl**.

Passo 2 Ingress oferece suporte a dois tipos de segredo do TLS: kubernetes.io/tls e IngressTLS. IngressTLS é usado como exemplo. Para mais detalhes, consulte **Criação de um segredo**. Para obter detalhes sobre exemplos do segredo kubernetes.io/tls e sua descrição, consulte **Segredos do TLS**.

Execute o seguinte comando para criar um arquivo YAML chamado **ingress-test-secret.yaml** (o nome do arquivo pode ser personalizado):

vi ingress-test-secret.yaml

O arquivo YAML é configurado da seguinte forma:

```
apiVersion: v1
data:
  tls.crt: LS0******tLS0tCg==
  tls.key: LS0tL******0tLS0K
kind: Secret
metadata:
  annotations:
```



```
    description: test for ingressTLS secrets
  name: ingress-test-secret
  namespace: default
type: IngressTLS
```

NOTA

Nas informações anteriores, **tls.crt** e **tls.key** são apenas exemplos. Substitua-os pelos arquivos reais. Os valores de **tls.crt** e **tls.key** são codificados em Base64.

Passo 3 Crie um segredo.

kubectl create -f ingress-test-secret.yaml

Se forem exibidas informações semelhantes às seguintes, o segredo foi criado.

```
secret/ingress-test-secret created
```

Visualize o segredo criado.

kubectl get secrets

Se forem exibidas informações semelhantes às seguintes, o segredo foi criado.

```
NAME                         TYPE                                  DATA      AGE
ingress-test-secret          IngressTLS                            2         13s
```

Passo 4 Crie um arquivo YAML chamado **ingress-test.yaml**. O nome do arquivo pode ser personalizado.

vi ingress-test.yaml

NOTA

A política de segurança padrão (kubernetes.io/elb.tls-ciphers-policy) é suportada apenas em clusters v1.17.17 ou posterior.

A política de segurança personalizada (kubernetes.io/elb.security_policy_id) é suportada apenas em clusters de v1.17.17 ou posterior.

A seguir, o balanceador de carga criado automaticamente é usado como exemplo. O arquivo YAML é configurado da seguinte forma:

Para clusters da v1.21 ou anterior:

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ingress-test
  annotations:
    kubernetes.io/elb.class: performance
    kubernetes.io/ingress.class: cce
    kubernetes.io/elb.port: '443'
    kubernetes.io/elb.autocreate:
      '{
          "type": "public",
          "bandwidth_name": "cce-bandwidth-******",
          "bandwidth_chargemode": "bandwidth",
          "bandwidth_size": 5,
          "bandwidth_sharetype": "PER",
          "eip_type": "5_bgp",
          "available_zone": [
              "ap-southeast-1a"
          ],
          "elb_virsubnet_ids":["b4bf8152-6c36-4c3b-9f74-2229f8e640c9"],
          "l7_flavor_name": "L7_flavor.elb.s1.small"
      }'
```



```
    kubernetes.io/elb.security_policy_id: 99bec42b-0dd4-4583-98e9-b05ce628d157  #
The priority of the custom security policy is higher than that of the default
security policy.
    kubernetes.io/elb.tls-ciphers-policy: tls-1-2
spec:
  tls:
  - secretName: ingress-test-secret
  rules:
  - host: foo.bar.com
    http:
      paths:
      - path: '/'
        backend:
          serviceName: <your_service_name>  # Replace it with the name of your
target Service.
          servicePort: 80
        property:
          ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
```

Para clusters da v1.23 ou posterior:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-test
  annotations:
    kubernetes.io/elb.class: performance
    kubernetes.io/elb.port: '443'
    kubernetes.io/elb.autocreate:
      '{
          "type": "public",
          "bandwidth_name": "cce-bandwidth-******",
          "bandwidth_chargemode": "bandwidth",
          "bandwidth_size": 5,
          "bandwidth_sharetype": "PER",
          "eip_type": "5_bgp",
          "available_zone": [
              "ap-southeast-1a"
          ],
          "elb_virsubnet_ids":["b4bf8152-6c36-4c3b-9f74-2229f8e640c9"],
          "l7_flavor_name": "L7_flavor.elb.s1.small"
       }'
    kubernetes.io/elb.security_policy_id: 99bec42b-0dd4-4583-98e9-b05ce628d157  #
The priority of the custom security policy is higher than that of the default
security policy.
    kubernetes.io/elb.tls-ciphers-policy: tls-1-2
spec:
  tls:
  - secretName: ingress-test-secret
  rules:
  - host: foo.bar.com
    http:
      paths:
      - path: '/'
        backend:
          service:
            name: <your_service_name>  # Replace it with the name of your target
Service.
            port:
              number: 8080             # Replace 8080 with the port number of
your target Service.
        property:
          ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
        pathType: ImplementationSpecific
  ingressClassName: cce
```


Tabela 7-35 Parâmetros principais

Parâmetro	Obrigatório	Tipo	Descrição
kubernetes.io/elb.security_policy_id	Não	String	O ID da política de grupo de segurança personalizada no ELB. Obtê-lo no console do ELB. Esse campo entra em vigor somente quando HTTPS é usado e tem uma prioridade mais alta do que a política de segurança padrão. Para obter detalhes sobre como criar e atualizar uma diretiva de segurança personalizada, consulte **Política de segurança do TLS**.
kubernetes.io/elb.tls-ciphers-policy	Não	String	O valor padrão é **tls-1-2**, que é a política de segurança padrão usada pelo ouvinte e só entra em vigor quando HTTPS é usado. Opções: ● tls-1-0 ● tls-1-1 ● tls-1-2 ● tls-1-2-strict Para obter detalhes sobre conjuntos de cifras para cada política de segurança, consulte **Tabela 7-36**.
tls	Não	Array of strings	Quando HTTPS é usado, esse parâmetro deve ser adicionado para especificar o certificado do segredo. Vários nomes de domínio independentes e certificados podem ser adicionados. Para mais detalhes, consulte **Configuração da Indicação de nome do servidor (SNI) para ingresses de ELB**.
secretName	Não	String	Esse parâmetro é obrigatório se o HTTPS for usado. Defina este parâmetro para o nome do segredo criado.

Tabela 7-36 Descrição do parâmetro **tls_ciphers_policy**

Política de segurança	Versão do TLS	Conjunto de criptografia
tls-1-0	TLS 1.2 TLS 1.1 TLS 1.0	ECDHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:AES128-GCM-SHA256:AES256-GCM-SHA384:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA256:AES128-SHA256:AES256-SHA256:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-ECDSA-AES128-SHA:ECDHE-RSA-AES128-SHA:ECDHE-RSA-AES256-SHA:ECDHE-ECDSA-AES256-SHA:AES128-SHA:AES256-SHA
tls-1-1	TLS 1.2 TLS 1.1	
tls-1-2	TLS 1.2	
tls-1-2-strict	TLS 1.2	ECDHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:AES128-GCM-SHA256:AES256-GCM-SHA384:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA256:AES128-SHA256:AES256-SHA256:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA384

Passo 5 Crie um ingress.

kubectl create -f ingress-test.yaml

Se forem exibidas informações semelhantes às seguintes, o ingress foi criado.

```
ingress/ingress-test created
```

Visualize o ingress criado.

kubectl get ingress

Se informações semelhantes às seguintes forem exibidas, o ingress foi criado e a carga de trabalho está acessível.

```
NAME             HOSTS     ADDRESS          PORTS   AGE
ingress-test     *         121.**.**.**     80      10s
```

Passo 6 Digite **https://*121.**.**.***:443** na caixa de endereço do navegador para acessar a carga de trabalho (por exemplo, **carga de trabalho do Nginx**).

121.**.**. indica o endereço IP do balanceador de carga unificado.

----Fim

Usar o certificado do ELB

Para usar o certificado do ELB, você pode especificar as anotações **kubernetes.io/elb.tls-certificate-ids**.

NOTA

1. Se você especificar o certificado IngressTLS e o certificado do ELB, o último será usado.
2. O CCE não verifica se o certificado do ELB é válido. Ele apenas verifica se o certificado existe.
3. Somente os clusters de v1.19.16-r2, v1.21.5-r0, v1.23.3-r0 ou posterior dão suporte ao certificado do ELB.

Para clusters da v1.21 ou anterior:

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ingress-test
  annotations:
    kubernetes.io/ingress.class: cce
    kubernetes.io/elb.port: '443'
    kubernetes.io/elb.id: 0b9a6c4d-bd8b-45cc-bfc8-ff0f9da54e95
    kubernetes.io/elb.class: union
    kubernetes.io/elb.tls-certificate-ids:
058cc023690d48a3867ad69dbe9cd6e5,b98382b1f01c473286653afd1ed9ab63
spec:
  rules:
  - host: ''
    http:
      paths:
      - path: '/'
        backend:
          serviceName: <your_service_name>  # Replace it with the name of your
target Service.
          servicePort: 80
        property:
          ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
```

Para clusters da v1.23 ou posterior:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-test
  namespace: default
  annotations:
    kubernetes.io/elb.port: '443'
    kubernetes.io/elb.id: 0b9a6c4d-bd8b-45cc-bfc8-ff0f9da54e95
    kubernetes.io/elb.class: union
    kubernetes.io/elb.tls-certificate-ids:
058cc023690d48a3867ad69dbe9cd6e5,b98382b1f01c473286653afd1ed9ab63
spec:
  rules:
    - host: ''
      http:
        paths:
          - path: '/'
            backend:
              service:
                name: <your_service_name>  # Replace it with the name of your
target Service.
                port:
                  number: 8080             # Replace 8080 with the port number of
your target Service.
            property:
              ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
            pathType: ImplementationSpecific
  ingressClassName: cce
```


7.4.2.5 Configuração da Indicação de nome do servidor (SNI) para ingresses de ELB

SNI permite que vários nomes de domínio de acesso baseados em TLS sejam fornecidos para sistemas externos usando o mesmo endereço IP e porta. Nomes de domínio diferentes podem usar certificados de segurança diferentes.

NOTA

- Esta função é suportada apenas em clusters de v1.15.11 e posteriores
- A opção **SNI** está disponível somente quando HTTPS é usado.
- Apenas um nome de domínio pode ser especificado para cada certificado de SNI. Certificados de domínio curinga são suportados.
- A política de segurança (kubernetes.io/elb.tls-ciphers-policy) é suportada apenas em clusters v1.17.11 ou posterior.

Você pode ativar a SNI quando as condições anteriores forem atendidas. O exemplo a seguir usa a criação automática de um balanceador de carga como exemplo. Neste exemplo, **sni-test-secret-1** e **sni-test-secret-2** são certificados de SNI. Os nomes de domínio especificados pelos certificados devem ser os mesmos que os dos certificados.

Para clusters da v1.21 ou anterior:

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ingress-test
  annotations:
    kubernetes.io/elb.class: performance
    kubernetes.io/ingress.class: cce
    kubernetes.io/elb.port: '443'
    kubernetes.io/elb.autocreate:
      '{
          "type": "public",
          "bandwidth_name": "cce-bandwidth-******",
          "bandwidth_chargemode": "bandwidth",
          "bandwidth_size": 5,
          "bandwidth_sharetype": "PER",
          "eip_type": "5_bgp",
          "available_zone": [
              "ap-southeast-1a"
          ],
          "elb_virsubnet_ids":["b4bf8152-6c36-4c3b-9f74-2229f8e640c9"],
          "l7_flavor_name": "L7_flavor.elb.s1.small"
      }'
    kubernetes.io/elb.tls-ciphers-policy: tls-1-2
spec:
  tls:
  - secretName: ingress-test-secret
  - hosts:
      - example.top  # Domain name specified when a certificate is issued
    secretName: sni-test-secret-1
  - hosts:
      - example.com  # Domain name specified when a certificate is issued
    secretName: sni-test-secret-2
  rules:
  - host: example.com
    http:
      paths:
      - path: '/'
        backend:
          serviceName: <your_service_name>  # Replace it with the name of your
target Service.
          servicePort: 80
        property:
          ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
```


Para clusters da v1.23 ou posterior:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-test
  annotations:
    kubernetes.io/elb.class: performance
    kubernetes.io/elb.port: '443'
    kubernetes.io/elb.autocreate:
      '{
          "type": "public",
          "bandwidth_name": "cce-bandwidth-******",
          "bandwidth_chargemode": "bandwidth",
          "bandwidth_size": 5,
          "bandwidth_sharetype": "PER",
          "eip_type": "5_bgp",
          "available_zone": [
              "ap-southeast-1a"
          ],
          "elb_virsubnet_ids":["b4bf8152-6c36-4c3b-9f74-2229f8e640c9"],
          "l7_flavor_name": "L7_flavor.elb.s1.small"
       }'
    kubernetes.io/elb.tls-ciphers-policy: tls-1-2
spec:
  tls:
  - secretName: ingress-test-secret
  - hosts:
      - example.top  # Domain name specified when a certificate is issued
    secretName: sni-test-secret-1
  - hosts:
      - example.com  # Domain name specified when a certificate is issued
    secretName: sni-test-secret-2
  rules:
  - host: example.com
    http:
      paths:
      - path: '/'
        backend:
          service:
            name: <your_service_name>  # Replace it with the name of your target
Service.
            port:
              number: 8080             # Replace 8080 with the port number of
your target Service.
        property:
          ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
        pathType: ImplementationSpecific
  ingressClassName: cce
```

7.4.2.6 Roteamento de ingresses do ELB para vários Serviços

Os ingresses podem encaminhar para vários Serviços de back-end com base em diferentes políticas correspondentes. O campo **spec** no arquivo YAML é definido como abaixo. Você pode acessar **www.example.com/foo**, **www.example.com/bar** e **foo.example.com/** para rotear para três Serviços de back-end diferentes.

AVISO

O URL registrado em uma política de encaminhamento de entrada deve ser o mesmo que o URL usado para acessar o Serviço de back-end. Caso contrário, um erro 404 será retornado.

```
...
spec:
  rules:
```



```
  - host: 'www.example.com'
    http:
      paths:
      - path: '/foo'
        backend:
          serviceName: <your_service_name>  # Replace it with the name of your
target Service.
          servicePort: 80
        property:
          ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
      - path: '/bar'
        backend:
          serviceName: <your_service_name>  # Replace it with the name of your
target Service.
          servicePort: 80
        property:
          ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
  - host: 'foo.example.com'
    http:
      paths:
      - path: '/'
        backend:
          serviceName: <your_service_name>  # Replace it with the name of your
target Service.
          servicePort: 80
        property:
          ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
```

7.4.2.7 Ingresses do ELB usando HTTP/2

Os ingresses podem usar HTTP/2 para expor Serviços. As conexões do balanceador de carga com sua aplicação usam HTTP/1.X por padrão. Se sua aplicação for capaz de receber solicitações HTTP2, você poderá adicionar o seguinte campo à anotação de entrada para habilitar o uso de HTTP/2:

```
kubernetes.io/elb.http2-enable: 'true'
```

A seguir, mostra o arquivo YAML para associação a um balanceador de carga existente:

Para clusters da v1.21 ou anterior:

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ingress-test
  annotations:
    kubernetes.io/elb.id: <your_elb_id>  # Replace it with the ID of your
existing load balancer.
    kubernetes.io/elb.ip: <your_elb_ip>  # Replace it with the IP of your
existing load balancer.
    kubernetes.io/elb.port: '443'
    kubernetes.io/ingress.class: cce
    kubernetes.io/elb.http2-enable: 'true' # Enable HTTP/2.
spec:
  tls:
  - secretName: ingress-test-secret
  rules:
  - host: ''
    http:
      paths:
      - path: '/'
        backend:
          serviceName: <your_service_name>  # Replace it with the name of your
target Service.
          servicePort: 80                   # Replace it with the port number of
your target Service.
        property:
          ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
```


Para clusters da v1.23 ou posterior:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-test
  annotations:
    kubernetes.io/elb.id: <your_elb_id>  # Replace it with the ID of your
existing load balancer.
    kubernetes.io/elb.ip: <your_elb_ip>  # Replace it with the IP of your
existing load balancer.
    kubernetes.io/elb.port: '443'
    kubernetes.io/elb.http2-enable: 'true' # Enable HTTP/2.
spec:
  tls:
  - secretName: ingress-test-secret
  rules:
  - host: ''
    http:
      paths:
      - path: '/'
        backend:
          service:
            name: <your_service_name>  # Replace it with the name of your target
Service.
            port:
              number: 8080             # Replace 8080 with the port number of
your target Service.
        property:
          ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
        pathType: ImplementationSpecific
  ingressClassName: cce
```

Tabela 6 Parâmetros de HTTP/2

Parâmetro	Obrigatório	Tipo	Descrição
kubernetes.io/elb.http2-enable	Não	Bool	Se o HTTP/2 está ativado. O encaminhamento de solicitações usando HTTP/2 melhora o desempenho de acesso entre a aplicação e o balanceador de carga. No entanto, o balanceador de carga ainda usa HTTP 1.X para encaminhar solicitações para o servidor back-end. **Este parâmetro é suportado em clusters de v1.19.16-r0, v1.21.3-r0 e versões posteriores.** Opções: ● **true**: ativado ● **false**: desativado (valor padrão) Observação: **o HTTP/2 pode ser ativado ou desativado somente quando o ouvinte usa HTTPS.** Este parâmetro é inválido quando o protocolo do ouvinte é HTTP e o padrão é **false**.

7.4.2.8 Interconexão de ingresses do ELB com serviços de back-end HTTPS

Ingress pode interconectar com serviços de back-end de diferentes protocolos. Por padrão, o canal de proxy de back-end de um ingress é um canal HTTP. Para criar um canal HTTPS, adicione a seguinte configuração ao campo **annotations**:

```
kubernetes.io/elb.pool-protocol: https
```

Restrições

- Esse recurso se aplica somente a clusters de v1.23.8, v1.25.3 e posteriores.
- O ingresso pode se interconectar com serviços de back-end HTTPS somente quando balanceadores de carga dedicados são usados.
- Ao interconectar-se com serviços de back-end HTTPS, defina **Client Protocol** de ingress como **HTTPS**.

Interconexão com serviços de back-end HTTPS

Um exemplo de configuração de ingress:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-test
  namespace: default
  annotations:
    kubernetes.io/elb.port: '443'
    kubernetes.io/elb.id: <your_elb_id>    # In this example, an existing
dedicated load balancer is used. Replace its ID with the ID of your dedicated
load balancer.
    kubernetes.io/elb.class: performance
    kubernetes.io/elb.pool-protocol: https  # Interconnected HTTPS backend service
    kubernetes.io/elb.tls-ciphers-policy: tls-1-2
spec:
  tls:
    - secretName: ingress-test-secret
  rules:
    - host: ''
      http:
        paths:
          - path: '/'
            backend:
              service:
                name: <your_service_name>  # Replace it with the name of your
target Service.
                port:
                  number: 80
            property:
              ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
            pathType: ImplementationSpecific
  ingressClassName: cce
```

7.4.2.9 Configuração do tempo limite para um ingress do ELB

ELB Ingress支持设置以下超时时间：

- 客户端连接空闲超时时间：没有收到客户端请求的情况下保持连接的最长时间。If no request is received during this period, the load balancer releases the connection and establishes a new one with the client when the next request arrives.
- 等待客户端请求超时时间：如果在规定的时间内客户端没有发送完请求头，或body体数据发送间隔超过一定时间，负载均衡会自动关闭连接。

- 等待后端服务器响应超时时间：向后端服务器发送请求后，如果在一定时间内没有收到响应，负载均衡将返回504错误码。

Restrições

- Este recurso entra em vigor apenas nas seguintes versões:
  - v1.19: v1.19.16-r30 ou mais recente
  - v1.21: v1.21.10-r10 ou mais recente
  - v1.23: v1.23.8-r10 ou mais recente
  - v1.25: v1.25.3-r10 ou mais recente
- 仅在使用独享型ELB时，Ingress支持设置超时时间。
- 更新Ingress时，如果删除超时时间配置，不会修改已有监听器的超时时间配置。

设置超时时间

Ingress配置示例如下：

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: test
  namespace: default
  annotations:
    kubernetes.io/elb.port: '80'
    kubernetes.io/elb.id: <your_elb_id>    # In this example, an existing
dedicated load balancer is used. Replace its ID with the ID of your dedicated
load balancer.
    kubernetes.io/elb.class: performance
    kubernetes.io/elb.keepalive_timeout: 300  # 客户端连接空闲超时时间
    kubernetes.io/elb.client_timeout: 60      # 等待客户端请求超时时间
    kubernetes.io/elb.member_timeout: 60      # 等待后端服务器响应超时时间
spec:
  rules:
    - host: ''
      http:
        paths:
          - path: /
            backend:
              service:
                name: test
                port:
                  number: 80
            property:
              ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
            pathType: ImplementationSpecific
  ingressClassName: cce
```


Tabela 7-37 Parâmetros de anotação principais

Parâmetro	Obrigatório	Tipo	Descrição
kubernetes.io/elb.keepalive_timeout	Não	Integer	Tempo limite para conexões do cliente. Se não houver solicitações chegando ao balanceador de carga após o término do tempo limite, o balanceador de carga desconectará a conexão com o cliente e estabelecerá uma nova conexão quando houver uma nova solicitação. 取值范围为0-4000s，默认值为60s。
kubernetes.io/elb.client_timeout	Não	Integer	等待客户端请求超时时间，包括两种情况： ● 读取整个客户端请求头的超时时长：如果客户端未在超时时长内发送完整个请求头，则请求将被中断。 ● 两个连续body体的数据包到达LB的时间间隔，超出client_timeout将会断开连接。 取值范围为1-300s，默认值为60s。
kubernetes.io/elb.member_timeout	Não	Integer	等待后端服务器响应超时时间。请求转发后端服务器后，等待超过member_timeout时长没有响应，负载均衡将终止等待，并返回 HTTP504错误码。 取值范围为1-300s，默认值为60s。

7.4.3 Ingresses de Nginx

7.4.3.1 Criação de ingresses de Nginx no console

Pré-requisitos

- Um ingresso fornece acesso à rede para cargas de trabalho de back-end. Certifique-se de que uma carga de trabalho esteja disponível em um cluster. Se nenhuma carga de trabalho estiver disponível, implemente uma carga de trabalho referindo-se a **Criação de uma Implantação**, **Criação de um StatefulSet** ou **Criação de um DaemonSet**.
- Um Serviço ClusterIP ou NodePort foi configurado para a carga de trabalho. Para obter detalhes sobre como configurar o Serviço, consulte **ClusterIP** ou **NodePort**.
- Para adicionar um Nginx ingress, verifique se o complemento **nginx-ingress** foi instalado no cluster. Para mais detalhes, consulte **Instalar o complemento**.

Precauções

- **Não é recomendável modificar nenhuma configuração de um balanceador de carga no console do ELB. Caso contrário, o Serviço será anormal.** Se você modificou a configuração, desinstale o complemento nginx-ingress e reinstale-o.
- O URL registrado em uma política de encaminhamento de entrada deve ser o mesmo que o URL usado para acessar o Serviço de back-end. Caso contrário, um erro 404 será retornado.
- O balanceador de carga selecionado ou criado deve estar na mesma VPC que o cluster atual e deve corresponder ao tipo de balanceador de carga (rede privada ou pública).
- O balanceador de carga tem pelo menos dois ouvintes, e as portas 80 e 443 não são ocupadas por ouvintes.

Criar um Nginx ingress

Esta seção usa uma carga de trabalho de Nginx como um exemplo para descrever como criar um Nginx ingress.

Passo 1 Efetue logon no console do CCE e acesse o console do cluster.

Passo 2 Escolha **Networking** no painel de navegação, clique a guia **Ingresses** e clique em **Create Ingress** no canto direito superior.

Passo 3 Configure parâmetros de ingress.

- **Name**: especifique um nome de um ingress, por exemplo, **nginx-ingress-demo**.
- **Namespace**: selecione o namespace ao qual o ingress será adicionado.
- **nginx-ingress**: essa opção é exibida somente quando o complemento **Nginx Ingress controller** foi instalado no cluster.

  Depois que você liga  , o nginx-ingress é interconectado para fornecer o acesso da camada-7. Você pode configurar os seguintes parâmetros:

  TLS: nginx-ingress suporta HTTP e HTTPS. A porta de escuta padrão reservada durante a instalação do nginx-ingress é **80** para solicitações HTTP e **443** para solicitações HTTPS. Para usar HTTPS, configure o certificado do servidor.

  – **Server Certificate**: ao criar um ouvinte HTTPS, vincule um certificado TLS para oferecer suporte à autenticação criptografada para transmissão de dados HTTPS. Para obter detalhes sobre como criar um segredo, consulte **Criação de um segredo**.

  – **SNI**: o Server Name Indication (SNI) é um protocolo estendido do TLS. Ele permite que vários nomes de domínio de acesso baseados em TLS sejam fornecidos para sistemas externos usando o mesmo endereço IP e porta. Nomes de domínio diferentes podem usar certificados de segurança diferentes. Depois que o SNI é habilitado, o cliente tem permissão para enviar o nome de domínio solicitado ao iniciar uma solicitação de handshake TLS. Depois de receber a solicitação TLS, o balanceador de carga procura o certificado com base no nome de domínio na solicitação. Se o certificado correspondente ao nome de domínio for encontrado, o balanceador de carga retornará o certificado para autorização. Caso contrário, o certificado padrão (certificado de servidor) é retornado para autorização.
- **Forwarding Policy**: quando o endereço de acesso de uma solicitação corresponde à política de encaminhamento (uma política de encaminhamento consiste em um nome de domínio e URL) a solicitação é encaminhada ao Serviço de destino correspondente para processamento. Clique em **Add Forwarding Policies** para adicionar várias políticas de encaminhamento.

- **Domain Name**: nome de domínio atual. Certifique-se de que o nome de domínio inserido tenha sido registrado e arquivado. Depois que o ingress for criado, vincule o nome de domínio ao endereço IP do balanceador de carga criado automaticamente (endereço IP do endereço de acesso de ingress). Se uma regra de nome de domínio estiver configurada, o nome de domínio deve sempre ser usado para acesso.
- **Regra de correspondência de URL**
  - **Default**: a correspondência de prefixo é usada por padrão.
  - **Prefix match**: se o URL estiver definido como **/healthz**, o URL que atende ao prefixo poderá ser acessado, por exemplo, **/healthz/v1** e **/healthz/v2**.
  - **Exact match**: o URL pode ser acessado somente quando é totalmente correspondido. Por exemplo, se o URL estiver definido como **/healthz**, apenas **/healthz** poderá ser acessado.
- **URL**: caminho de acesso a ser registrado, por exemplo, **/healthz**.

  NOTA

  - A regra de correspondência do caminho de acesso de Nginx ingress é baseada no prefixo do caminho separado pela barra (/) e diferencia maiúsculas de minúsculas. Se o subcaminho separado por uma barra (/) corresponder ao prefixo, o acesso será normal. No entanto, se o prefixo for apenas uma parte da cadeia de caracteres no subcaminho, o acesso não será correspondido. Por exemplo, se o URL estiver definido como /healthz, /healthz/v1 será correspondido, mas /healthzv1 não será correspondido.
  - O caminho de acesso adicionado aqui deve existir na aplicação back-end. Caso contrário, o encaminhamento falha.

    Por exemplo, o URL de acesso padrão da aplicação Nginx é **/usr/share/nginx/html**. Ao adicionar **/test** à política de encaminhamento de ingress, certifique-se de que o URL de acesso da seu aplicação Nginx contenha **/usr/share/nginx/html/test**. Caso contrário, o erro 404 será retornado.
- **Destination Service**: selecione um Serviço existente ou crie um Serviço. Os Serviços que não atendem aos critérios de pesquisa são automaticamente filtrados.
- **Destination Service Port**: selecione a porta de acesso do Serviço de destino.
- **Operation**: clique em **Delete** para excluir a configuração.

- **Annotation**: o valor está no formato de chave:valor. Você pode usar **anotações** para consultar as configurações suportadas pelo nginx-ingress.

Passo 4 Após a conclusão da configuração, clique em **OK**.

Depois que o ingress é criado, ele é exibido na lista de ingresses.

----Fim

7.4.3.2 Uso do kubectl para criar um ingress de Nginx

Cenário

Esta seção usa uma **carga de trabalho do Nginx** como exemplo para descrever como criar um ingress de Nginx usando o kubectl.

Pré-requisitos

- O complemento nginx-ingress foi instalado em um cluster. Para mais detalhes, consulte **Instalar o complemento**.

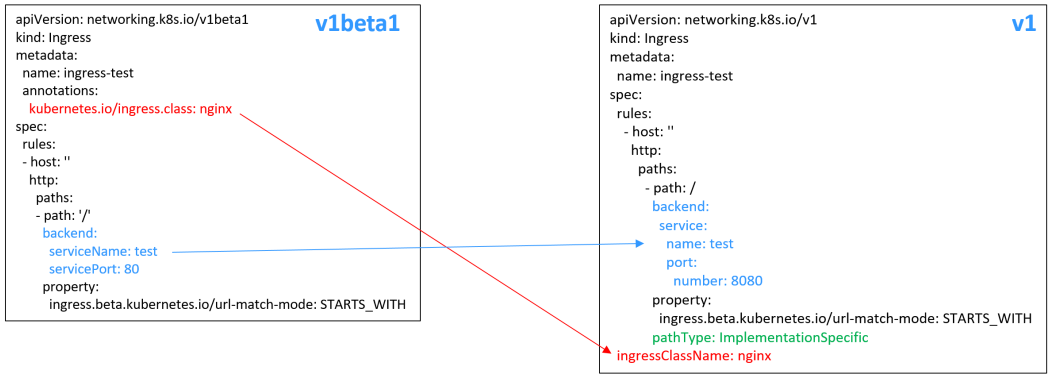
- Um ingress fornece acesso à rede para cargas de trabalho de back-end. Certifique-se de que uma carga de trabalho esteja disponível em um cluster. Se nenhuma carga de trabalho estiver disponível, implemente uma carga de trabalho referindo-se a **Criação de uma Implantação**, **Criação de um StatefulSet** ou **Criação de um DaemonSet**.
- Um Serviço ClusterIP ou NodePort foi configurado para a carga de trabalho. Para obter detalhes sobre como configurar o Serviço, consulte **ClusterIP** ou **NodePort**.

Descrição de ingress de networking.k8s.io/v1

Em clusters do CCE de v1.23 ou posterior, a versão de ingress é comutada para **networking.k8s.io/v1**.

Comparada com v1beta1, v1 tem as seguintes diferenças nos parâmetros:

- O tipo de ingress é alterado de **kubernetes.io/ingress.class** em **annotations** para **spec.ingressClassName**.
- O formato de **backend** é alterado.
- O parâmetro **pathType** deve ser especificado para cada caminho. As opções são as seguintes:
  - **ImplementationSpecific**: o método de correspondência depende do Ingress Controller. O método de correspondência definido por **ingress.beta.kubernetes.io/url-match-mode** é usado no CCE, que é o mesmo que v1beta1.
  - **Exact**: correspondência exata do URL, que diferencia maiúsculas de minúsculas.
  - **Prefix**: correspondência com base no prefixo de URL separado por uma barra (/). A correspondência diferencia maiúsculas de minúsculas, e os elementos no caminho são correspondidos um a um. Um elemento path refere-se a uma lista de rótulos no caminho separados por uma barra (/).



Criar um ingress de Nginx

Passo 1 Use o kubectl para se conectar ao cluster. Para mais detalhes, consulte **Conexão a um cluster usando o kubectl**.

Passo 2 Crie um arquivo YAML chamado **ingress-test.yaml**. O nome do arquivo pode ser personalizado.

vi ingress-test.yaml

NOTA

A partir do cluster v1.23, a versão de entrada é alternada de **networking.k8s.io/v1beta1** para **networking.k8s.io/v1**. Para obter detalhes sobre as diferenças entre v1 e v1beta1, consulte **Descrição de ingress de networking.k8s.io/v1**.

O seguinte usa HTTP como um exemplo para descrever como configurar o arquivo YAML:

Se o nó estiver em um cluster de v1.23 ou posterior:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-test
spec:
  rules:
    - host: ''
      http:
        paths:
          - path: /
            backend:
              service:
                name: <your_service_name>  # Replace it with the name of your
target Service.
                port:
                  number: <your_service_port>  # Replace it with the port number
of your target Service.
            property:
              ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
            pathType: ImplementationSpecific
  ingressClassName: nginx   # Nginx ingress is used.
```

Se o nó estiver em um cluster de v1.21 ou anterior:

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ingress-test
  namespace: default
  annotations:
    kubernetes.io/ingress.class: nginx   # Nginx ingress is used.
spec:
  rules:
    - host: ''
      http:
        paths:
          - path: '/'
            backend:
              serviceName: <your_service_name>  # Replace it with the name of
your target Service.
              servicePort: <your_service_port>  # Replace it with the port number
of your target Service.
```

Tabela 7-38 Parâmetros principais

Parâmetro	Obrigatório	Tipo	Descrição
kubernetes.io/ingress.class	Sim (somente para clusters da v1.21 ou anterior)	String	**nginx**: indica que ingress de Nginx é usado. Esta opção não pode ser usada se o complemento nginx-ingress não estiver instalado. Esse parâmetro é obrigatório quando um ingress é criado chamando a API.

Parâmetro	Obrigatório	Tipo	Descrição
ingressClassName	Sim (somente para clusters da v1.23 ou posterior)	String	**nginx**: indica que Nginx ingress é usado. Esta opção não pode ser usada se o complemento nginx-ingress não estiver instalado. Esse parâmetro é obrigatório quando um ingress é criado chamando a API.
host	Não	String	Nome de domínio para acessar o Serviço. Por padrão, esse parâmetro é deixado em branco e o nome de domínio precisa ser totalmente correspondido. Certifique-se de que o nome de domínio tenha sido registrado e arquivado. Depois que uma regra de nome de domínio estiver configurada, você deve usar o nome de domínio para acesso.
path	Sim	String	Caminho de rota definido pelo usuário. Todas as solicitações de acesso externo devem corresponder ao **host** e ao **path**. **NOTA** ● A regra de correspondência do caminho de acesso de Nginx ingress é baseada no prefixo do caminho separado pela barra (/) e diferencia maiúsculas de minúsculas. Se o subcaminho separado por uma barra (/) corresponder ao prefixo, o acesso será normal. No entanto, se o prefixo for apenas uma parte da cadeia de caracteres no subcaminho, o acesso não será correspondido. Por exemplo, se o URL estiver definido como /healthz, /healthz/v1 será correspondido, mas /healthzv1 não será correspondido. ● O caminho de acesso adicionado aqui deve existir na aplicação back-end. Caso contrário, o encaminhamento falha. Por exemplo, o URL de acesso padrão da aplicação Nginx é **/usr/share/nginx/html**. Ao adicionar **/test** à política de encaminhamento de ingress, certifique-se de que o URL de acesso da seu aplicação Nginx contenha **/usr/share/nginx/html/test**. Caso contrário, o erro 404 será retornado.

Parâmetro	Obrigatório	Tipo	Descrição
ingress.beta.kubernetes.io/url-match-mode	Não	String	Política de correspondência de rotas. Padrão: **STARTS_WITH** (correspondência de prefixo) Opções: ● **EQUAL_TO**: correspondência exata ● **STARTS_WITH**: correspondência de prefixo
pathType	Sim	String	Tipo de caminho. Este campo é suportado apenas por clusters de v1.23 ou posterior. ● **ImplementationSpecific**: o método de correspondência depende do Ingress Controller. O método de correspondência definido por **ingress.beta.kubernetes.io/url-match-mode** é usado no CCE. ● **Exact**: correspondência exata do URL, que diferencia maiúsculas de minúsculas. ● **Prefix**: correspondência de prefixo, que diferencia maiúsculas de minúsculas. Com esse método, o caminho do URL é separado em vários elementos por barras (/) e os elementos são correspondidos um por um. Se cada elemento no URL corresponder ao caminho, os subcaminhos do URL poderão ser roteados normalmente. **NOTA** – Durante a correspondência de prefixo, cada elemento deve ser exatamente correspondido. Se o último elemento do URL for a subcadeia do último elemento no caminho da solicitação, nenhuma correspondência será executada. Por exemplo, **/foo/bar** corresponde a **/foo/bar/baz**, mas não corresponde a **/foo/barbaz**. – Quando elementos são separados por barras (/), se o URL ou o caminho da requisição termina com uma barra (/), a barra (/) no final é ignorada. Por exemplo, **/foo/bar** corresponde a **/foo/bar/**. Veja **exemplos** de correspondência de caminho de ingress.

Passo 3 Crie um ingress.

kubectl create -f ingress-test.yaml

Se forem exibidas informações semelhantes às seguintes, o ingress foi criado.

```
ingress/ingress-test created
```

Visualize o ingress criado.

kubectl get ingress

Se informações semelhantes às seguintes forem exibidas, o ingress foi criado com êxito e a carga de trabalho está acessível.

```
NAME             HOSTS     ADDRESS          PORTS   AGE
ingress-test     *         121.**.**.**     80      10s
```

Passo 4 Digite **http://*121.**.**.***:80** na caixa de endereço do navegador para acessar a carga de trabalho (por exemplo, **carga de trabalho do Nginx**).

121.**.**.** indica o endereço IP do balanceador de carga unificado.

----Fim

7.4.3.3 Configuração de certificados HTTPS para ingresses do Nginx

Os certificados HTTPS podem ser configurados para ingress para fornecer serviços de segurança.

Passo 1 Use o kubectl para se conectar ao cluster. Para mais detalhes, consulte **Conexão a um cluster usando o kubectl**.

Passo 2 Ingress oferece suporte a dois tipos de chave do TLS: kubernetes.io/tls e IngressTLS. IngressTLS é usado como exemplo. Para mais detalhes, consulte **Criação de um segredo**. Para obter detalhes sobre exemplos do segredo kubernetes.io/tls e sua descrição, consulte **Segredos do TLS**.

Execute o seguinte comando para criar um arquivo YAML chamado **ingress-test-secret.yaml** (o nome do arquivo pode ser personalizado):

vi ingress-test-secret.yaml

O arquivo YAML é configurado da seguinte forma:

```
apiVersion: v1
data:
  tls.crt: LS0******tLS0tCg==
  tls.key: LS0tL******0tLS0K
kind: Secret
metadata:
  annotations:
    description: test for ingressTLS secrets
  name: ingress-test-secret
  namespace: default
type: IngressTLS
```

NOTA

Nas informações anteriores, **tls.crt** e **tls.key** são apenas exemplos. Substitua-os pelos arquivos reais. Os valores de **tls.crt** e **tls.key** são codificados em Base64.

Passo 3 Crie um segredo.

kubectl create -f ingress-test-secret.yaml

Se forem exibidas informações semelhantes às seguintes, o segredo foi criado.

```
secret/ingress-test-secret created
```

Visualize o segredo criado.

kubectl get secrets

Se forem exibidas informações semelhantes às seguintes, o segredo foi criado.

```
NAME                         TYPE                                  DATA      AGE
ingress-test-secret          IngressTLS                            2         13s
```

Passo 4 Crie um arquivo YAML chamado **ingress-test.yaml**. O nome do arquivo pode ser personalizado.

vi ingress-test.yaml

Para clusters da v1.23 ou posterior:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-test
  namespace: default
spec:
  tls:
  - hosts:
    - foo.bar.com
    secretName: ingress-test-secret  # Replace it with your TLS key certificate.
  rules:
    - host: foo.bar.com
      http:
        paths:
          - path: /
            backend:
              service:
                name: <your_service_name>  # Replace it with the name of your
target Service.
                port:
                  number: <your_service_port>  # Replace it with the port number
of your target Service.
            property:
              ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
            pathType: ImplementationSpecific
  ingressClassName: nginx
```

Para clusters da v1.21 ou anterior:

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ingress-test
  annotations:
    kubernetes.io/ingress.class: nginx
spec:
  tls:
  - hosts:
    - foo.bar.com
    secretName: ingress-test-secret   # Replace it with your TLS key certificate.
  rules:
  - host: foo.bar.com
    http:
      paths:
      - path: '/'
        backend:
          serviceName: <your_service_name>  # Replace it with the name of your
target Service.
          servicePort: <your_service_port>  # Replace 8080 with the port number
of your target Service.
  ingressClassName: nginx
```


Passo 5 Crie um ingress.

kubectl create -f ingress-test.yaml

Se forem exibidas informações semelhantes às seguintes, o ingress foi criado.

```
ingress/ingress-test created
```

Visualize o ingress criado.

kubectl get ingress

Se informações semelhantes às seguintes forem exibidas, o ingress foi criado e a carga de trabalho está acessível.

```
NAME             HOSTS     ADDRESS          PORTS   AGE
ingress-test     *         121.**.**.**     80      10s
```

Passo 6 Digite **https://*121.**.**.**:443*** na caixa de endereço do navegador para acessar a carga de trabalho (por exemplo, **carga de trabalho do Nginx**).

121.**.**.* indica o endereço IP do balanceador de carga unificado.

----Fim

7.4.3.4 Configuração de regras de reescrita de URL para ingresses de Nginx

Em alguns cenários de aplicação, o URL de acesso fornecido pelo serviço de back-end é diferente do caminho especificado na regra de ingress. O ingress encaminha diretamente o caminho de acesso para o mesmo caminho de back-end. Se a reescrita de URL não estiver configurada, 404 será retornado para todas as solicitações de acesso. Por exemplo, se o caminho de acesso na regra de ingress estiver definido como **/app/demo** e o caminho de acesso fornecido pelo serviço de back-end for **/demo**, as solicitações de acesso serão encaminhadas diretamente para o caminho **/app/demo** do serviço de back-end, que não corresponde ao caminho de acesso real (**/demo**) fornecido pelo serviço de back-end. Como resultado, 404 é retornado.

Nesse caso, você pode usar o método Rewrite para implementar a reescrita de URL. Ou seja, você pode usar a anotação **nginx.ingress.kubernetes.io/rewrite-target** para implementar regras de reescrita para caminhos diferentes.

Configurar regras de reescrita

Para clusters da v1.23 ou posterior:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-test
  namespace: default
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /$2
spec:
  rules:
    - host: 'rewrite.bar.com'
      http:
        paths:
          - path: '/something(/|$)(.*)'
            backend:
              service:
                name: <your_service_name>  # Replace it with the name of your
target Service.
                port:
```



```
                  number: <your_service_port>  # Replace 8080 with the port
number of your target Service.
            property:
              ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
            pathType: ImplementationSpecific
  ingressClassName: nginx
```

Para clusters da v1.21 ou anterior:

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ingress-test
  namespace: default
  annotations:
    kubernetes.io/ingress.class: nginx
    nginx.ingress.kubernetes.io/rewrite-target: /$2
spec:
  rules:
    - host: 'rewrite.bar.com'
      http:
        paths:
          - path: '/something(/|$)(.*)'
            backend:
              serviceName: <your_service_name>  # Replace it with the name of
your target Service.
              servicePort: <your_service_port>  # Replace 8080 with the port
number of your target Service.
```

NOTA

Enquanto **rewrite-target** é especificado para um ingress, todos os caminhos sob o mesmo host em todas as definições de ingress diferenciam maiúsculas de minúsculas, incluindo os ingresss que não têm **rewrite-target** especificado.

No exemplo anterior, o espaço reservado $2 indica que todos os caracteres correspondentes ao segundo parêntese (.*) são preenchidos na anotação **nginx.ingress.kubernetes.io/rewrite-target**.

Por exemplo, a definição de ingress anterior resultará nas seguintes reescritas:

- rewrite.bar.com/something reescreve para rewrite.bar.com/.
- rewrite.bar.com/something/ reescreve para rewrite.bar.com/.
- rewrite.bar.com/something/new reescreve para rewrite.bar.com/new.

No contêiner nginx-ingress-controller, você pode exibir todas as configurações de entrada no arquivo **nginx.conf** no diretório **/etc/nginx**. A regra de reescrita no exemplo anterior gera um comando Rewrite e o grava no campo **location** no arquivo **nginx.conf**.

```
## start server rewrite.bar.com
        server {
                server_name rewrite.bar.com ;
                ...
                location ~* "^/something(/|$)(.*)" {
                        set $namespace      "default";
                        set $ingress_name   "ingress-test";
                        set $service_name   "<your_service_name>";
                        set $service_port   "80";
                        ...
                        rewrite "(?i)/something(/|$)(.*)" /$2 break;
                        ...
                }
        }
        ## end server rewrite.bar.com
```

A sintaxe básica do comando Rewrite é a seguinte:

```
rewrite regex replacement [flag];
```


- **regex**: expressão regular para URIs correspondentes. No exemplo anterior, **(?i)/something(/|$)(.*)** é a expressão regular para URIs correspondentes, onde **(?i)** indica que não faz distinção entre maiúsculas e minúsculas.
- **replacement**: conteúdo a ser reescrito. No exemplo anterior, **/$2** indica que o caminho é reescrito para todos os caracteres correspondentes pelo segundo parêntese **(.*)**.
- **flag**: formato de reescrita.
  - **last**: continua a corresponder à próxima regra depois que a regra atual é correspondida.
  - **break**: pára a correspondência após a correspondência da regra atual.
  - **redirect**: retorna um redirecionamento temporário com o código 302.
  - **permanent**: retorna um redirecionamento permanente com o código 301.

Configuração de reescrita avançada

Alguns requisitos complexos e avançados de reescrita podem ser implementados modificando o arquivo de configuração do **nginx.conf**. No entanto, a função de anotação **nginx.ingress.kubernetes.io/rewrite-target** pode ser personalizada para atender a requisitos de reescrita mais complexos.

- nginx.ingress.kubernetes.io/server-snippet: adicione configurações personalizadas ao campo **server** no arquivo **nginx.conf**.
- nginx.ingress.kubernetes.io/configuration-snippet: adicione configurações personalizadas ao campo **location** no arquivo **nginx.conf**.

Você pode usar as duas anotações anteriores para inserir um comando Rewrite no campo **server** ou **location** no arquivo **nginx.conf** para reescrever o URL. O seguinte é um exemplo:

```
annotations:
     kubernetes.io/ingress.class: "nginx"
     nginx.ingress.kubernetes.io/configuration-snippet: |
        rewrite ^/stylesheets/(.*)$ /something/stylesheets/$1 redirect;  # Add
the /something prefix.
        rewrite ^/images/(.*)$ /something/images/$1 redirect;  # Add the /
something prefix.
```

Nas duas regras anteriores, o caminho **/something** é adicionado ao URL de acesso.

- Quando um usuário acessa **rewrite.bar.com/stylesheets/new.css**, ele reescreve para **rewrite.bar.com/something/stylesheets/new.css**.
- Quando um usuário acessa **rewrite.bar.com/images/new.jpg**, ele reescreve para **rewrite.bar.com/something/images/new.jpg**.

Redirecionar HTTP para HTTPS

Por padrão, se um ingress usar TLS, as solicitações serão redirecionadas (código de status 308) para HTTPS quando o HTTP for usado para acesso. Você também pode usar a seguinte anotação para redirecionar solicitações forçadamente para HTTPS.

Para clusters da v1.23 ou posterior:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-test
  namespace: default
  annotations:
    nginx.ingress.kubernetes.io/ssl-redirect: 'true'
spec:
```



```
  rules:
    - host: ''
      http:
        paths:
          - path: /
            backend:
              service:
                name: <your_service_name>  # Replace it with the name of your
target Service.
                port:
                  number: <your_service_port>  # Replace 8080 with the port
number of your target Service.
            property:
              ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
            pathType: ImplementationSpecific
  ingressClassName: nginx
```

Para clusters da v1.21 ou anterior:

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ingress-test
  namespace: default
  annotations:
    kubernetes.io/ingress.class: nginx
    nginx.ingress.kubernetes.io/ssl-redirect: 'true'
spec:
  rules:
    - host: ''
      http:
        paths:
          - path: /
            backend:
              serviceName: <your_service_name>  # Replace it with the name of
your target Service.
              servicePort: <your_service_port>  # Replace 8080 with the port
number of your target Service.
```

7.4.3.5 Interconexão de ingresses do Nginx com serviços de back-end HTTPS

Ingress pode funcionar como um proxy para serviços de back-end usando protocolos diferentes. Por padrão, o canal de proxy de back-end de um ingress é um canal HTTP. Para criar um canal HTTPS, adicione a seguinte configuração ao campo **annotations**:

```
nginx.ingress.kubernetes.io/backend-protocol: "HTTPS"
```

Um exemplo de configuração de ingress:

Para clusters da v1.23 ou posterior:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-test
  namespace: default
  annotations:
    nginx.ingress.kubernetes.io/backend-protocol: "HTTPS"
spec:
  tls:
    - secretName: ingress-test-secret  # Replace it with your TLS key certificate.
  rules:
    - host: ''
      http:
        paths:
          - path: '/'
            backend:
              service:
                name: <your_service_name>  # Replace it with the name of your
target Service.
```



```
                port:
                  number: <your_service_port>  # Replace 8080 with the port
number of your target Service.
            property:
              ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
            pathType: ImplementationSpecific
  ingressClassName: nginx
```

Para clusters da v1.21 ou anterior:

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ingress-test
  namespace: default
  annotations:
    kubernetes.io/ingress.class: nginx
    nginx.ingress.kubernetes.io/backend-protocol: "HTTPS"
spec:
  tls:
    - secretName: ingress-test-secret  # Replace it with your TLS key certificate.
  rules:
    - host: ''
      http:
        paths:
          - path: '/'
            backend:
              serviceName: <your_service_name>  # Replace it with the name of
your target Service.
              servicePort: <your_service_port>  # Replace 8080 with the port
number of your target Service.
```

7.4.3.6 Ingresses de Nginx usar hashing consistente para balanceamento de carga

O Nginx nativo suporta várias regras de balanceamento de carga, incluindo round robin ponderado e hash IP. Um ingress de Nginx suporta balanceamento de carga usando hash consistente com base nos recursos do Nginx nativo.

Por padrão, o método de hash IP suportado pelo Nginx usa o espaço de hash linear. O servidor back-end é selecionado com base no valor de hash do endereço IP. No entanto, quando esse método é usado para adicionar ou excluir um nó, todos os endereços IP precisam ser hash novamente e, em seguida, roteados novamente. Como resultado, um grande número de sessões são perdidas ou o cache se torna inválido. Portanto, o hashing consistente é introduzido no ingress do Nginx para resolver esse problema.

O hashing consistente é um algoritmo de hash especial, que constrói um espaço de hash em anel para substituir o espaço de hash linear comum. Quando um nó é adicionado ou excluído, apenas a rota de destino é migrada no sentido horário e outras rotas não precisam ser alteradas. Dessa forma, o reencaminhamento pode ser reduzido o máximo possível, resolvendo o problema de balanceamento de carga causado pela adição e exclusão de nó dinâmicos.

Se uma regra de hashing consistente estiver configurada, o servidor recém-adicionado compartilhará a carga de todos os outros servidores. Da mesma forma, quando um servidor é removido, todos os outros servidores podem compartilhar a carga do servidor removido. Isso equilibra a carga entre os nós no cluster e evita o efeito de avalanche causado pela quebra de um nó.

Configurar uma regra de hashing consistente

Um ingress de Nginx pode usar a anotação **nginx.ingress.kubernetes.io/upstream-hash-by** para configurar regras de hashing consistentes. O seguinte é um exemplo:

Para clusters da v1.23 ou posterior:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-test
  namespace: default
  annotations:
    nginx.ingress.kubernetes.io/upstream-hash-by: "$request_uri"  # Perform
hashing based on the request URI.
spec:
  rules:
    - host: ''
      http:
        paths:
          - path: '/'
            backend:
              service:
                name: <your_service_name>  # Replace it with the name of your
target Service.
                port:
                  number: <your_service_port>  # Replace 8080 with the port
number of your target Service.
            property:
              ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
            pathType: ImplementationSpecific
  ingressClassName: nginx
```

Para clusters da v1.21 ou anterior:

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ingress-test
  namespace: default
  annotations:
    kubernetes.io/ingress.class: nginx
    nginx.ingress.kubernetes.io/upstream-hash-by: "$request_uri"  # Perform
hashing based on the request URI.
spec:
  rules:
    - host: ''
      http:
        paths:
          - path: '/'
            backend:
              serviceName: <your_service_name>  # Replace it with the name of
your target Service.
              servicePort: <your_service_port>  # Replace 8080 with the port
number of your target Service.
```

O valor de **nginx.ingress.kubernetes.io/upstream-hash-by** pode ser uma variável de nginx, um valor de texto ou qualquer combinação:

- **nginx.ingress.kubernetes.io/upstream-hash-by: "$request_uri"** indica que as solicitações são transformadas em hash com base no URI da solicitação.
- **nginx.ingress.kubernetes.io/upstream-hash-by: "$request_uri$host"** indica que as solicitações são transformadas em hash com base no URI da solicitação e no nome do domínio.
- **nginx.ingress.kubernetes.io/upstream-hash-by: "${request_uri}-text-value"** indica que as solicitações são hash com base no URI da solicitação e no valor do texto.

Documentação

Hashing upstream de NGINX personalizado

7.4.3.7 Configuração de ingresses de Nginx usando anotações

O complemento nginx-ingress no CCE usa o gráfico de comunidade e a imagem. Se os parâmetros de complemento padrão não puderem atender às suas demandas, você poderá adicionar anotações para definir o que precisa, como o back-end padrão, o tempo limite e o tamanho do corpo da solicitação.

Esta seção descreve anotações comuns usadas para criar um ingress do tipo Nginx.

NOTA

- O valor da chave de uma anotação só pode ser uma cadeia. Outros tipos (como valores booleanos ou numéricos) devem ser colocados entre aspas (""), por exemplo, "true", "false" e "100".
- O ingress de Nginx suporta anotações nativas da comunidade. Para obter detalhes, consulte **Anotações**.

- **Tipo de ingress**
- **Configurar uma regra de reescrita de URL**
- **Interconexão com serviços de back-end HTTPS**
- **Criar uma regra de hash consistente para balanceamento de carga**
- **Intervalo de tempo limite personalizado**
- **Personalizar o tamanho do corpo**
- **Documentação**

Tipo de ingress

Tabela 7-39 Anotações do tipo de ingress

Parâmetro	Tipo	Descrição	Versão do cluster suportada
kubernetes.io/ingress.class	String	• **nginx**: ingress de Nginx é usado. • **cce**: o ingress autodesenvolvido do ELB é usado. Esse parâmetro é obrigatório quando um ingress é criado chamando a API. Para clusters v1.23 ou posterior, use o parâmetro **ingressClassName**. Para mais detalhes, consulte **Uso do kubectl para criar um ingress de Nginx**.	Somente clusters da v1.21 ou anterior

Para obter detalhes sobre como usar as anotações anteriores, consulte **Uso do kubectl para criar um ingress de Nginx**.

Configurar uma regra de reescrita de URL

Tabela 7-40 Anotações de regra de reescrever URL

Parâmetro	Tipo	Descrição
nginx.ingress.kubernetes.io/rewrite-target	String	URI de destino onde o tráfego deve ser redirecionado.
nginx.ingress.kubernetes.io/ssl-redirect	Bool	Indica se o acesso está disponível apenas por meio de SSL. O valor padrão é **true** quando o ingress contém um certificado.
nginx.ingress.kubernetes.io/force-ssl-redirect	Bool	Indica se deve redirecionar um pedido para HTTPS, mesmo que o TLS não esteja ativado para o ingress. Quando HTTP é usado para acesso, a solicitação é forçosamente redirecionada (código de status 308) para HTTPS.

Para obter detalhes sobre os cenários de aplicação, consulte **Configuração de regras de reescrita de URL para ingresses de Nginx**.

Interconexão com serviços de back-end HTTPS

Tabela 7-41 Anotações para interconexão com serviços de back-end HTTPS

Parâmetro	Tipo	Descrição
nginx.ingress.kubernetes.io/backend-protocol	String	Se esse parâmetro for definido como **HTTPS**, o HTTPS será usado para encaminhar solicitações para o contêiner de serviço de back-end.

Para obter detalhes sobre os cenários de aplicação, consulte **Interconexão de ingresses do Nginx com serviços de back-end HTTPS**.

Criar uma regra de hash consistente para balanceamento de carga

Tabela 7-42 Anotação de hashing consistente para balanceamento de carga

Parâmetro	Tipo	Descrição
nginx.ingress.kubernetes.io/upstream-hash-by	String	Habilitar hashing consistente para balanceamento de carga para servidores de back-end. O valor do parâmetro pode ser um parâmetro de Nginx, um valor de texto ou qualquer combinação. Por exemplo: ● **nginx.ingress.kubernetes.io/upstream-hash-by: "$request_uri"** indica que as solicitações são hash com base no URI da solicitação. ● **nginx.ingress.kubernetes.io/upstream-hash-by: "$request_uri$host"** indica que as solicitações são hash com base no URI da solicitação e no nome do domínio. ● **nginx.ingress.kubernetes.io/upstream-hash-by: "${request_uri}-text-value"** indica que as solicitações são hash com base no URI da solicitação e no valor do texto.

Para obter detalhes sobre os cenários de aplicação, consulte **Ingresses de Nginx usar hashing consistente para balanceamento de carga**.

Intervalo de tempo limite personalizado

Tabela 7-43 Anotações personalizadas do intervalo de tempo limite

Parâmetro	Tipo	Descrição
nginx.ingress.kubernetes.io/proxy-connect-timeout	String	Intervalo de tempo limite de conexão personalizado. Você não precisa ajustar a unidade ao definir o intervalo de tempo limite. A unidade padrão é a segunda. Exemplo: `nginx.ingress.kubernetes.io/proxy-connect-timeout: '120'`

Personalizar o tamanho do corpo

Tabela 7-44 Anotações de personalização do tamanho do corpo

Parâmetro	Tipo	Descrição
nginx.ingress.kubernetes.io/proxy-body-size	String	Quando o tamanho do corpo em uma solicitação excede o limite superior, o erro 413 é retornado ao cliente. Você pode usar este parâmetro para ajustar o limite superior do tamanho do corpo. Exemplo: `nginx.ingress.kubernetes.io/proxy-body-size: 8m`

Documentação

Para obter detalhes sobre os parâmetros de anotação suportados pelos ingresses do Nginx, consulte **Anotações**.

7.5 DNS

7.5.1 Visão geral

Introdução ao CoreDNS

Quando você cria um cluster, o **complemento CoreDNS** é instalado para resolver nomes de domínio no cluster.

Você pode ver o pod do complemento CoreDNS no namespace de kube-system.

```
$ kubectl get po --namespace=kube-system
NAME                                       READY   STATUS    RESTARTS   AGE
coredns-7689f8bdf-295rk                    1/1     Running   0          9m11s
coredns-7689f8bdf-h7n68                    1/1     Running   0          11m
```

Depois que o CoreDNS é instalado, ele se torna um DNS. Depois que o Serviço é criado, o CoreDNS registra o nome do Serviço e o endereço IP. Dessa forma, o pod pode obter o endereço IP do Serviço consultando o nome do Serviço do CoreDNS.

nginx.<namespace>.svc.cluster.local é usado para acessar o Serviço. **nginx** é o nome do Serviço, **<namespace>** é o namespace e **svc.cluster.local** é o sufixo do nome de domínio. Em uso real, você pode omitir **<namespace>.svc.cluster.local** no mesmo namespace e usar o ServiceName.

Uma vantagem de usar ServiceName é que você pode escrever ServiceName no programa ao desenvolver a aplicação. Desta forma, você não precisa saber o endereço IP de um Serviço específico.

Depois que o CoreDNS é instalado, há também um Serviço no namespace do kube-system, como mostrado abaixo.

```
$ kubectl get svc -n kube-system
NAME               TYPE           CLUSTER-IP      EXTERNAL-IP
```



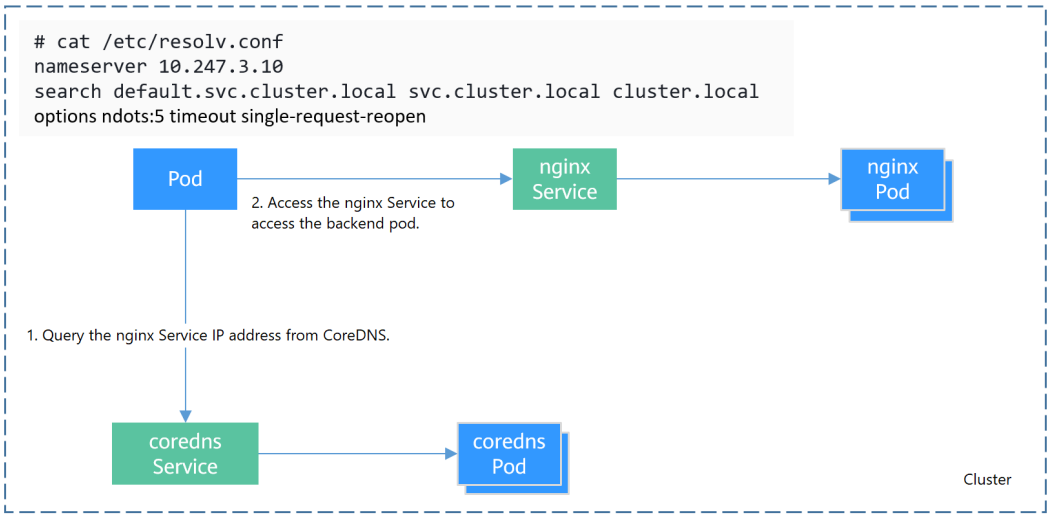
```
PORT(S)                       AGE
coredns            ClusterIP      10.247.3.10     <none>
53/UDP,53/TCP,8080/TCP        13d
```

Por padrão, depois que outros pods são criados, o endereço do Serviço CoreDNS é gravado como o endereço do servidor de resolução de nome de domínio no arquivo **/etc/resolv.conf** do pod. Crie um pod e visualize o arquivo **/etc/resolv.conf** da seguinte maneira:

```
$ kubectl exec test01-6cbbf97b78-krj6h -it -- /bin/sh
/ # cat /etc/resolv.conf
nameserver 10.247.3.10
search default.svc.cluster.local svc.cluster.local cluster.local
options ndots:5 timeout single-request-reopen
```

Quando um usuário acessa o *Service name:Port* do pod do Nginx, o endereço IP do Serviço do Nginx é resolvido a partir do CoreDNS e, em seguida, o endereço IP do Serviço do Nginx é acessado. Dessa forma, o usuário pode acessar o pod do Nginx de back-end.

Figura 7-33 Exemplo de resolução de nome de domínio em um cluster



Como funciona a resolução de nomes de domínio no Kubernetes?

As políticas de DNS podem ser configuradas para cada pod. O Kubernetes é compatível com as políticas de DNS **Default**, **ClusterFirst**, **ClusterFirstWithHostNet** e **None**. Para obter detalhes, consulte **DNS para Serviços e pods**. Essas políticas são especificadas no campo **dnsPolicy** no pod específico.

- **Default**: os pods herdam a configuração de resolução de nome do nó em que os pods são executados. O servidor DNS upstream personalizado e o domínio de stub não podem ser usados junto com esta política.
- **ClusterFirst**: qualquer consulta de DNS que não corresponda ao sufixo de domínio de cluster configurado, como **www.kubernetes.io**, é encaminhada para o servidor de nomes upstream herdado do nó. Os administradores de cluster podem ter domínios de stub extra e servidores DNS upstream configurados.
- **ClusterFirstWithHostNet**: para pods em execução com hostNetwork, defina a política de DNS **ClusterFirstWithHostNet**.
- **None**: ela permite que um pod ignore as configurações de DNS do ambiente do Kubernetes. Todas as configurações de DNS devem ser fornecidas usando o campo **dnsPolicy** no pod específico.

NOTA

- Os clusters do Kubernetes v1.10 e posteriores são compatíveis com **Default**, **ClusterFirst**, **ClusterFirstWithHostNet** e **None**. Os clusters anteriores ao Kubernetes v1.10 suportam apenas **Default**, **ClusterFirst** e **ClusterFirstWithHostNet**.
- **Default** não é a política de DNS padrão. Se **dnsPolicy** não for explicitamente especificado, será usado **ClusterFirst**.

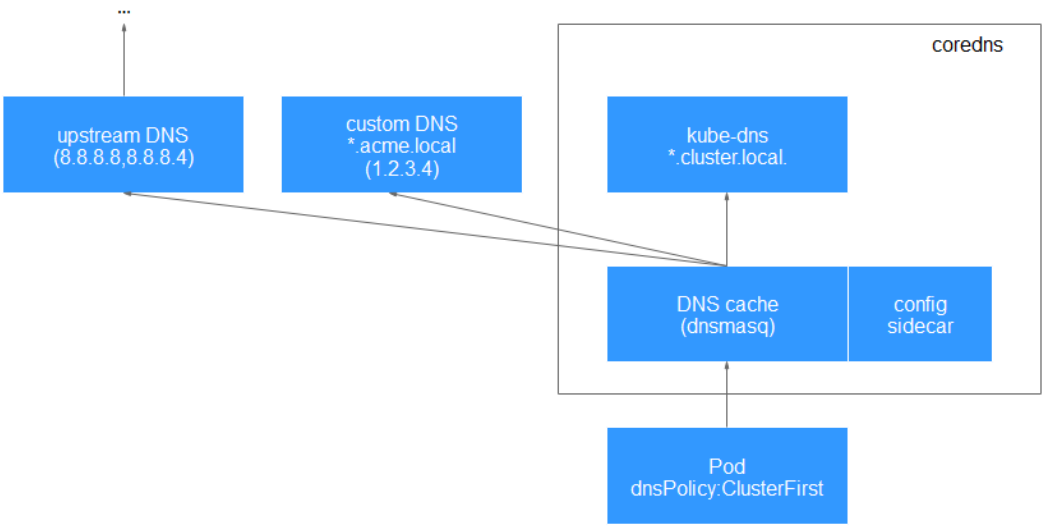
Roteamento

Without stub domain configurations: qualquer consulta que não corresponda ao sufixo de domínio de cluster configurado, como **www.kubernetes.io**, é encaminhada para o servidor DNS upstream herdado do nó.

With stub domain configurations: se os domínios de stub e servidores DNS upstream estiverem configurados, as consultas de DNS serão roteadas de acordo com o seguinte fluxo:

1. A consulta é enviada primeiro para a camada de cache do DNS no CoreDNS.
2. A partir da camada de cache, o sufixo da solicitação é examinado e, em seguida, a solicitação é encaminhada para o DNS correspondente:
   - Nomes com o sufixo de cluster, por exemplo, **.cluster.local**: a solicitação é enviada ao CoreDNS.
   - Nomes com o sufixo de domínio de stub, por exemplo, **.acme.local**: a solicitação é enviada para o resolvedor do DNS personalizado configurado que escuta, por exemplo, em 1.2.3.4.
   - Nomes que não correspondem ao sufixo (por exemplo, **widget.com**): a solicitação é encaminhada para o DNS upstream.

Figura 7-34 Roteamento



Operações relacionadas

Você também pode configurar o DNS em uma carga de trabalho. Para mais detalhes, consulte **Configuração de DNS**.

Você também pode usar o CoreDNS para implementar a resolução de nomes de domínio definida pelo usuário. Para mais detalhes, consulte **Uso de CoreDNS para resolução de nome de domínio personalizado**.

Você também pode usar o DNSCache para melhorar o desempenho da resolução do DNS. Para mais detalhes, consulte **Uso do DNSCache do NodeLocal para melhorar o desempenho do DNS**.

7.5.2 Configuração de DNS

Cada cluster do Kubernetes tem um complemento de DNS integrado (Kube-DNS ou CoreDNS) para fornecer resolução de nome de domínio para cargas de trabalho no cluster. Ao lidar com uma alta simultaneidade de consultas do DNS, Kube-DNS/CoreDNS pode encontrar um gargalo de desempenho, ou seja, pode falhar ocasionalmente para atender consultas do DNS. Há casos em que as cargas de trabalho do Kubernetes iniciam consultas do DNS desnecessárias. Isso torna o DNS sobrecarregado se houver muitas consultas de DNS simultâneas. Ajustar a configuração de DNS para cargas de trabalho reduzirá os riscos de falhas de consulta do DNS até certo ponto.

Para obter mais informações sobre o DNS, consulte **CoreDNS**.

Itens de configuração de DNS

Execute o comando **cat /etc/resolv.conf** em um nó ou contêiner do Linux para exibir o arquivo de configuração do resolvedor de DNS. Veja a seguir um exemplo de configuração de resolvedor do DNS de um contêiner em um cluster do Kubernetes:

```
nameserver 10.247.x.x
search default.svc.cluster.local svc.cluster.local cluster.local
options ndots:5
```

Opções de configuração

- **nameserver**: uma lista de endereços IP de um servidor de nomes que o resolvedor irá consultar. Se esse parâmetro for definido como 10.247.x.x, o resolvedor consultará o kube-dns/CoreDNS. Se este parâmetro for definido para outro endereço IP, o resolvedor irá consultar um servidor DNS na nuvem ou no local.
- **search**: uma lista de pesquisa de nome de host. Quando um nome de domínio não puder ser resolvido, as consultas de DNS serão tentadas combinando o nome de domínio com cada domínio na lista de pesquisa por vez até que uma correspondência seja encontrada ou todos os domínios na lista de pesquisa sejam tentados. Para clusters do CCE, a lista de pesquisa está atualmente limitada a três domínios por contêiner. Quando um nome de domínio inexistente está sendo resolvido, oito consultas de DNS serão iniciadas porque cada nome de domínio (incluindo aqueles na lista de pesquisa) será consultado duas vezes, uma para IPv4 e outra para IPv6.
- **options**: opções que permitem que certas variáveis internas do resolvedor sejam modificadas. As opções comuns incluem timeout e ndots.

  O valor **ndots:5** significa que, se um nome de domínio tiver menos de 5 pontos (.), as consultas de DNS serão tentadas combinando o nome de domínio com cada domínio na lista de pesquisa. Se nenhuma correspondência for encontrada depois que todos os domínios na lista de pesquisa forem tentados, o nome de domínio será usado para a consulta de DNS. Se o nome de domínio tiver 5 ou mais de 5 pontos, ele será tentado primeiro para consulta de DNS. Caso o nome de domínio não possa ser resolvido, as consultas de DNS serão tentadas combinando o nome de domínio com cada domínio na lista de pesquisa, por sua vez.

  Por exemplo, o nome de domínio **www.***.com** tem apenas dois pontos (menor que o valor de **ndots**) e, portanto, a sequência de consultas DNS é a seguinte: **www.***.default.svc.cluster.local**, **www.***.com.svc.cluster.local**, **www.***.com.cluster.local** e **www.***.com**. Isso significa que pelo menos sete

consultas de DNS serão iniciadas antes que o nome de domínio seja resolvido em um endereço IP. É claro que, quando muitas consultas desnecessárias de DNS serão iniciadas para acessar um nome de domínio externo. Há espaço para melhorias na configuração de DNS da carga de trabalho.

NOTA

Para obter mais informações sobre as opções de configuração no arquivo de configuração do resolvedor usado pelos sistemas operacionais Linux, visite **http://man7.org/linux/man-pages/man5/resolv.conf.5.html**.

Configurar o DNS para uma carga de trabalho usando o console

O Kubernetes fornece opções de configuração relacionadas ao DNS para aplicações. O uso da configuração de DNS da aplicação pode efetivamente reduzir consultas de DNS desnecessárias em determinados cenários e melhorar a simultaneidade do serviço. O procedimento a seguir usa uma aplicação Nginx como exemplo para descrever como adicionar configurações de DNS para uma carga de trabalho no console.

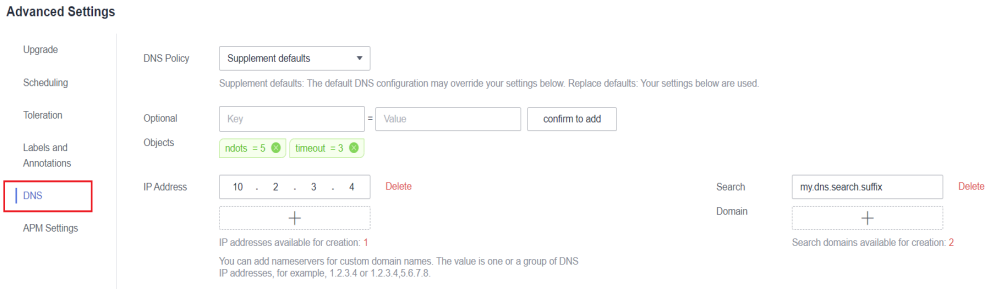
Passo 1 Efetue logon no console do CCE, acesse o console do cluster, selecione **Workloads** no painel de navegação e clique em **Create Workload** no canto superior direito.

Passo 2 Configure informações básicas sobre a carga de trabalho. Para mais detalhes, consulte **Criação de uma carga de trabalho**.

Passo 3 Na área **Advanced Settings**, clique na guia **DNS** e defina os seguintes parâmetros conforme necessário:

- **DNS Policy**: as políticas de DNS fornecidas no console correspondem ao campo **dnsPolicy** no arquivo YAML. Para mais detalhes, consulte **Tabela 7-45**.
  - **Supplement defaults**: corresponde a **dnsPolicy=ClusterFirst**. Os contêineres podem resolver os nomes de domínio internos do cluster registrados por um serviço e os nomes de domínio externos expostos a redes públicas.
  - **Replace defaults**: corresponde a **dnsPolicy=None**. Você deve configurar **IP Address** e **Search Domain**. Os contêineres usam apenas o endereço IP definido pelo usuário e as configurações de domínio de pesquisa para a resolução de nomes de domínio.
  - **Inherit defaults**: corresponde a **dnsPolicy=Default**. Os contêineres usam a configuração de resolução de nome de domínio do nó em que os pods são executados e não podem resolver os nomes de domínio internos do cluster.
- **Optional Objects**: os parâmetros de opções no **campo dnsConfig**. Cada objeto pode ter uma propriedade de nome (obrigatório) e uma propriedade de valor (opcional). Depois de definir as propriedades, clique em **confirm to add**.
  - **timeout**: intervalo de tempo limite, em segundos.
  - **ndots**: número de pontos (.) que devem estar presentes em um nome de domínio. Se um nome de domínio tiver pontos menores que esse valor, o sistema operacional procurará o nome no domínio de pesquisa. Caso contrário, o nome é um nome de domínio totalmente qualificado (FQDN) e será tentado primeiro como um nome absoluto.
- **IP Address**: **nameservers** no **dnsConfig**. Você pode configurar o servidor de nomes de domínio para o nome de domínio personalizado. O valor é um ou um grupo de endereços IP do DNS.
- **Search Domain**: **searches** no **dnsConfig**. Uma lista de domínios de pesquisa do DNS para pesquisa de nome de host no pod. Esta propriedade é opcional. Quando

especificada, a lista fornecida será mesclada nos nomes de domínio de pesquisa gerados a partir da política do DNS escolhida em **dnsPolicy**. Nomes de domínio duplicados são removidos.



Passo 4 Clique em **Create Workload**.

----Fim

Configurar o DNS usando a carga de trabalho de YAML

Ao criar uma carga de trabalho usando um arquivo YAML, você pode definir as definições de DNS no YAML. O seguinte é um exemplo para uma aplicação de Nginx:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: container-1
          image: nginx:latest
          imagePullPolicy: IfNotPresent
      imagePullSecrets:
        - name: default-secret
      dnsPolicy: None
      dnsConfig:
        options:
          - name: ndots
            value: '5'
          - name: timeout
            value: '3'
        nameservers:
          - 10.2.3.4
        searches:
          - my.dns.search.suffix
```

- **dnsPolicy**

  O campo **dnsPolicy** é usado para configurar uma política de DNS para uma aplicação. O valor padrão é **ClusterFirst**. A tabela a seguir lista as configurações de **dnsPolicy**.

Tabela 7-45 dnsPolicy

Parâmetro	Descrição
ClusterFirst (valor padrão)	Configuração de DNS personalizada adicionada à configuração de DNS padrão. Por padrão, a aplicação se conecta ao CoreDNS (CoreDNS do cluster do CCE conecta-se ao DNS na nuvem por padrão). O dnsConfig personalizado será adicionado aos parâmetros do DNS padrão. Os contêineres podem resolver os nomes de domínio internos do cluster registrados por um serviço e os nomes de domínio externos expostos a redes públicas. A lista de pesquisa (opção **search**) e **ndots: 5** estão presentes no arquivo de configuração do DNS. Portanto, ao acessar um nome de domínio externo e um nome de domínio interno de cluster longo (por exemplo, kubernetes.default.svc.cluster.local), a lista de pesquisa geralmente será percorrida primeiro, resultando em pelo menos seis consultas de DNS inválidas. O problema de consultas de DNS inválidas desaparece somente quando um nome de domínio curto interno do cluster (por exemplo, kubernetes) está sendo acessado.

ClusterFirstWithHostNet	Por padrão, os aplicativos configurados com a **rede do host** são interconectados com a configuração de DNS do nó onde o pod está localizado. A configuração de DNS é especificada no arquivo de DNS para o qual o parâmetro de kubelet **--resolv-conf** aponta. Nesse caso, o cluster do CCE usa o DNS na nuvem. Se as cargas de trabalho precisarem usar Kube-DNS/CoreDNS do cluster, defina **dnsPolicy** como **ClusterFirstWithHostNet** e o arquivo de configuração de DNS do contêiner seja o mesmo que ClusterFirst, no qual ainda existem consultas de DNS inválidas.
	`...`
	`spec:`
	`  containers:`
	`  - image: nginx:latest`
	`  imagePullPolicy: IfNotPresent`
	`  name: container-1`
	`  restartPolicy: Always`
	`  **hostNetwork: true**`
	`  dnsPolicy: **ClusterFirstWithHostNet**`

Default	A configuração de DNS do nó onde o pod está localizado é herdada e a configuração de DNS personalizada é adicionada à configuração herdada. O arquivo de configuração de DNS do contêiner é o arquivo de configuração de DNS para o qual o sinalizador do kubelet **--resolv-conf** aponta. Neste caso, um DNS da nuvem é usado para clusters do CCE. Ambos os campos **search** e **options** são deixados sem especificação. Essa configuração só pode resolver os nomes de domínio externos registrados na Internet e não nomes de domínio internos de cluster. Essa configuração está livre do problema de consultas de DNS inválidas.
None	A configuração de DNS padrão é substituída pela configuração de DNS personalizada e somente a configuração de DNS personalizada é usada. Se **dnsPolicy** estiver definido como **None**, o campo **dnsConfig** deve ser especificado porque todas as configurações de DNS devem ser fornecidas usando o campo **dnsConfig**.

NOTA

Se o campo **dnsPolicy** não for especificado, o valor padrão será **ClusterFirst** em vez de **Default**.

- **dnsConfig**

  O campo **dnsConfig** é usado para configurar parâmetros de DNS para cargas de trabalho. Os parâmetros configurados são mesclados ao arquivo de configuração de DNS gerado de acordo com **dnsPolicy**. Se **dnsPolicy** estiver definido como **None**, o arquivo de configuração de DNS da carga de trabalho será especificado pelo campo **dnsConfig**. Se **dnsPolicy** não estiver definido como **None**, os parâmetros de DNS configurados em **dnsConfig** serão adicionados ao arquivo de configuração de DNS gerado de acordo com **dnsPolicy**.

  Tabela 7-46 dnsConfig

Parâmetro	Descrição
options	Uma lista opcional de objetos onde cada objeto pode ter uma propriedade de nome (obrigatória) e uma propriedade de valor (opcional). O conteúdo desta propriedade será mesclado com as opções geradas a partir da política de DNS especificada no **dnsPolicy**. As entradas duplicadas são removidas.
nameservers	Uma lista de endereços IP que serão usados como servidores DNS. Se **dnsPolicy** da carga de trabalho estiver definido como **None**, a lista deverá conter pelo menos um endereço IP, caso contrário, essa propriedade será opcional. Os servidores listados serão combinados com os servidores de nomes gerados a partir da política de DNS especificada em **dnsPolicy** com endereços duplicados removidos. **NOTA** Um máximo de três endereços de DNS podem ser configurados para um servidor de nomes no arquivo de configuração de DNS do contêiner. ● Se **dnsPolicy** estiver definida como **ClusterFirst** e o cluster usar **CoreDNS**, você pode adicionar dois endereços de DNS personalizados além do endereço de CoreDNS. Endereços de DNS em excesso são inválidos. ● Se **dnsPolicy** estiver definida como **ClusterFirst** e o cluster usar **CoreDNS** e **NodeLocal DNSCache**, você poderá adicionar um endereço de DNS personalizado além dos endereços de CoreDNS e NodeLocal DNSCache. Endereços de DNS em excesso são inválidos.
searches	Uma lista de domínios de pesquisa de DNS para pesquisa de nome de host no Pod. Esta propriedade é opcional. Quando especificada, a lista fornecida será mesclada nos nomes de domínio de pesquisa gerados a partir da política do DNS escolhida em **dnsPolicy**. Nomes de domínio duplicados são removidos. O Kubernetes permite no máximo 6 domínios de pesquisa.

Exemplos de configuração

O exemplo a seguir descreve como configurar o DNS para cargas de trabalho.

- **Caso de uso 1: usar Kube-DNS/CoreDNS embutido em clusters do Kubernetes**

  Cenário

  Kube-DNS/CoreDNS do Kubernetes no cluster aplica-se a resolver apenas nomes de domínio internos ao cluster ou nomes de domínio internos ao cluster + nomes de domínio externos. Este é o DNS padrão para cargas de trabalho.

  Exemplo:

  ```
  apiVersion: v1
  kind: Pod
  metadata:
    namespace: default
    name: dns-example
  spec:
    containers:
    - name: test
      image: nginx:alpine
    dnsPolicy: ClusterFirst
    imagePullSecrets:
      - name: default-secret
  ```

  Arquivo de configuração de DNS do contêiner:

  ```
  nameserver 10.247.3.10
  search default.svc.cluster.local svc.cluster.local cluster.local
  options ndots:5
  ```

- **Caso de uso 2: usar um DNS em nuvem**

  Cenário

  Um DNS não pode resolver nomes de domínio internos de cluster e, portanto, aplica-se ao cenário em que as cargas de trabalho acessam apenas nomes de domínio externos registrados na Internet.

  Exemplo:

  ```
  apiVersion: v1
  kind: Pod
  metadata:
    namespace: default
    name: dns-example
  spec:
    containers:
    - name: test
      image: nginx:alpine
    dnsPolicy: Default  # The DNS configuration file that the kubelet --resolv-conf parameter points to is used. In this case, the CCE cluster uses the DNS on the cloud.
    imagePullSecrets:
      - name: default-secret
  ```

  Arquivo de configuração de DNS do contêiner:

  ```
  nameserver 100.125.x.x
  ```

- **Caso de uso 3: usar Kube-DNS/CoreDNS para cargas de trabalho em execução com hostNetwork**

  Cenário

  Por padrão, um DNS é usado para cargas de trabalho em execução com hostNetwork. Se as cargas de trabalho precisarem usar Kube-DNS/CoreDNS, defina **dnsPolicy** como **ClusterFirstWithHostNet**.

  Exemplo:

  ```
  apiVersion: v1
  kind: Pod
  metadata:
    name: nginx
  spec:
  ```



```
  hostNetwork: true
  dnsPolicy: ClusterFirstWithHostNet
  containers:
  - name: nginx
    image: nginx:alpine
    ports:
    - containerPort: 80
  imagePullSecrets:
    - name: default-secret
```

Arquivo de configuração de DNS do contêiner:

```
nameserver 10.247.3.10
search default.svc.cluster.local svc.cluster.local cluster.local
options ndots:5
```

- **Caso de uso 4: personalizar a configuração de DNS da aplicação**

  Cenário

  Você pode personalizar de forma flexível o arquivo de configuração de DNS para aplicações. Usar **dnsPolicy** e **dnsConfig** juntos pode resolver quase todos os cenários, incluindo os cenários em que um DNS local será usado, vários DNSs serão em cascata e as opções de configuração de DNS serão modificadas.

  Exemplo 1: usar seu DNS local

  *Defina **dnsPolicy** como **None** para que o arquivo de configuração de DNS do aplicação seja gerada com base no **dnsConfig**.*

```
apiVersion: v1
kind: Pod
metadata:
  namespace: default
  name: dns-example
spec:
  containers:
  - name: test
    image: nginx:alpine
  dnsPolicy: "None"
  dnsConfig:
    nameservers:
    - 10.2.3.4  # IP address of your on-premises DNS
    searches:
    - ns1.svc.cluster.local
    - my.dns.search.suffix
    options:
    - name: ndots
      value: "2"
    - name: timeout
      value: "3"
  imagePullSecrets:
    - name: default-secret
```

  Arquivo de configuração de DNS do contêiner:

```
nameserver 10.2.3.4
search ns1.svc.cluster.local my.dns.search.suffix
options timeout:3 ndots:2
```

  Exemplo 2: modificar a opção ndots no arquivo de configuração de DNS para reduzir consultas de DNS inválidas

  Defina **dnsPolicy** para um valor diferente de **None** para que os parâmetros de DNS configurados em **dnsConfig** sejam adicionados ao arquivo de configuração de DNS gerado com base em **dnsPolicy**.

```
apiVersion: v1
kind: Pod
metadata:
  namespace: default
  name: dns-example
spec:
```



```
  containers:
  - name: test
    image: nginx:alpine
  dnsPolicy: "ClusterFirst"
  dnsConfig:
    options:
    - name: ndots
      value: "2" # The ndots:5 option in the DNS configuration file generated based on the ClusterFirst policy is changed to ndots:2.
  imagePullSecrets:
    - name: default-secret
```

Arquivo de configuração de DNS do contêiner:

```
nameserver 10.247.3.10
search default.svc.cluster.local svc.cluster.local cluster.local
options ndots:2
```

Exemplo 3: usar vários DNSs em sequência serial

```
apiVersion: v1
kind: Pod
metadata:
  namespace: default
  name: dns-example
spec:
  containers:
  - name: test
    image: nginx:alpine
  dnsPolicy: ClusterFirst  # Added DNS configuration. The cluster connects to CoreDNS by default.
  dnsConfig:
    nameservers:
    - 10.2.3.4 # IP address of your on-premises DNS
  imagePullSecrets:
    - name: default-secret
```

NOTA

Um máximo de três endereços de DNS podem ser configurados para um servidor de nomes no arquivo de configuração de DNS do contêiner.

- Se **dnsPolicy** estiver definida como **ClusterFirst** e o cluster usar **CoreDNS**, você pode adicionar dois endereços de DNS personalizados além do endereço de CoreDNS. Endereços de DNS em excesso são inválidos.
- Se **dnsPolicy** estiver definida como **ClusterFirst** e o cluster usar **CoreDNS** e **NodeLocal DNSCache**, você poderá adicionar um endereço de DNS personalizado além dos endereços de CoreDNS e NodeLocal DNSCache. Endereços de DNS em excesso são inválidos.

Arquivo de configuração de DNS do contêiner:

```
nameserver 10.247.3.10 10.2.3.4
search default.svc.cluster.local svc.cluster.local cluster.local
options ndots:5
```

7.5.3 Uso de CoreDNS para resolução de nome de domínio personalizado

Desafios

Ao usar o CCE, talvez seja necessário resolver nomes de domínio internos personalizados nos seguintes cenários:

- No código legado, um nome de domínio fixo é configurado para chamar outros serviços internos. Se o sistema decidir usar os Serviços do Kubernetes, a carga de trabalho de refatoração de código pode ser pesada.

- Um serviço é criado fora do cluster. Os dados no cluster precisam ser enviados para o serviço por meio de um nome de domínio fixo.

Solução

Existem várias soluções baseadas em CoreDNS para resolução de nomes de domínio personalizados:

- **Configurar o domínio de stub para CoreDNS**: você pode adicioná-lo no console, que é fácil de operar.
- **Uso do plug-in de hosts do CoreDNS para configurar a resolução para qualquer nome de domínio**: você pode adicionar qualquer conjunto de registros, o que é semelhante a adicionar um conjunto de registros no arquivo **/etc/hosts** local.
- **Uso do plug-in de reescrita do CoreDNS para apontar um nome de domínio para um serviço no cluster**: um apelido é atribuído ao Serviço do Kubernetes. Você não precisa saber o endereço IP do registro de resolução com antecedência.
- **Uso do plug-in de encaminhamento do CoreDNS para definir o DNS autoconstruído como o DNS de upstream**: o DNS autoconstruído pode gerenciar um grande número de registros de resolução. Você não precisa modificar a configuração do CoreDNS ao adicionar ou excluir registros.

Precauções

A modificação incorreta na configuração do CoreDNS pode causar falhas de resolução de nome de domínio no cluster. Realize testes antes e depois da modificação.

Configurar o domínio de stub para CoreDNS

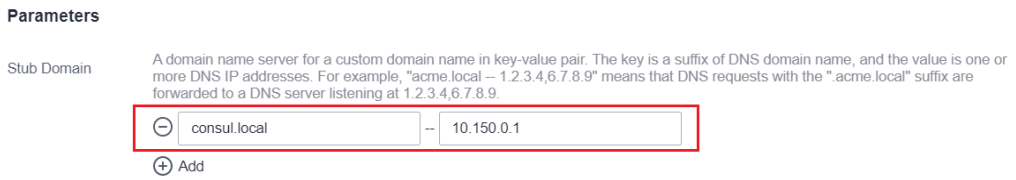
Os administradores de cluster podem modificar o ConfigMap do CoreDNS Corefile para alterar o funcionamento da descoberta de serviços.

Suponha que um administrador de cluster tem um servidor DNS Consul localizado em 10.150.0.1 e todos os nomes de domínio Consul têm o sufixo **.consul.local**.

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster.

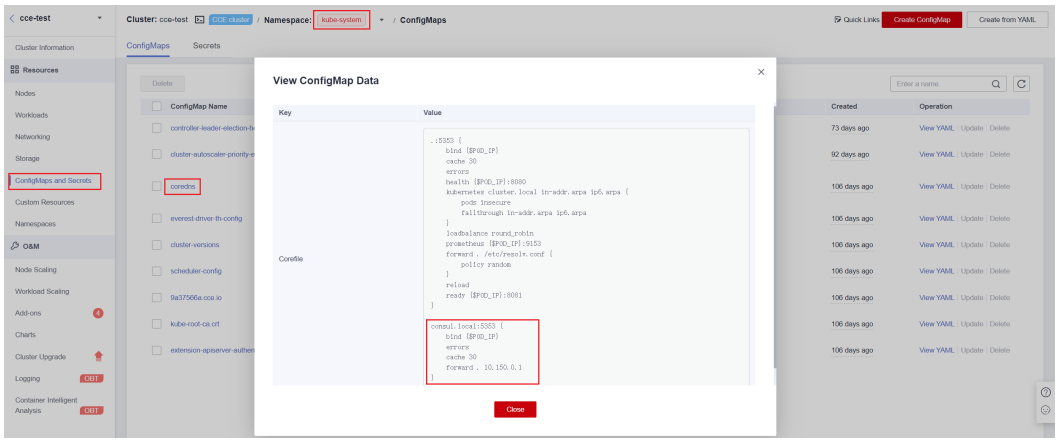
Passo 2 No painel de navegação, escolha **Add-ons**. Na página exibida, clique em **Edit** em **CoreDNS**.

Passo 3 Adicione um domínio de stub na área **Parameters**. O formato é um par chave-valor. A chave é um nome de domínio de sufixo do DNS e o valor é um endereço IP do DNS ou um grupo de endereços IP do DNS, por exemplo, **consul.local -- 10.150.0.1**.



Passo 4 Clique em **OK**.

Passo 5 Escolha **ConfigMaps and Secrets** no painel de navegação, selecione o namespace do **kube-system** e veja os dados do ConfigMap do CoreDNS para verificar se a atualização foi bem-sucedida.



O conteúdo do Corefile correspondente é o seguinte:

```
.:5353 {
    bind {$POD_IP}
    cache 30
    errors
    health {$POD_IP}:8080
    kubernetes cluster.local in-addr.arpa ip6.arpa {
        pods insecure
        fallthrough in-addr.arpa ip6.arpa
    }
    loadbalance round_robin
    prometheus {$POD_IP}:9153
    forward . /etc/resolv.conf {
        policy random
    }
    reload
    ready {$POD_IP}:8081
}
consul.local:5353 {
    bind {$POD_IP}
    errors
    cache 30
    forward . 10.150.0.1
}
```

----Fim

Modificar o arquivo de configuração de hosts do CoreDNS

Depois de modificar o arquivo hosts no CoreDNS, você não precisa configurar o arquivo hosts em cada pod para adicionar registros de resolução.

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster.

Passo 2 No painel de navegação, escolha **Add-ons**. Na página exibida, clique em **Edit** em **CoreDNS**.

Passo 3 Edite a configuração avançada em **Parameters** e adicione o seguinte conteúdo ao campo **plugins**:

```
{
  "configBlock": "192.168.1.1 www.example.com\nfallthrough",
  "name": "hosts"
}
```

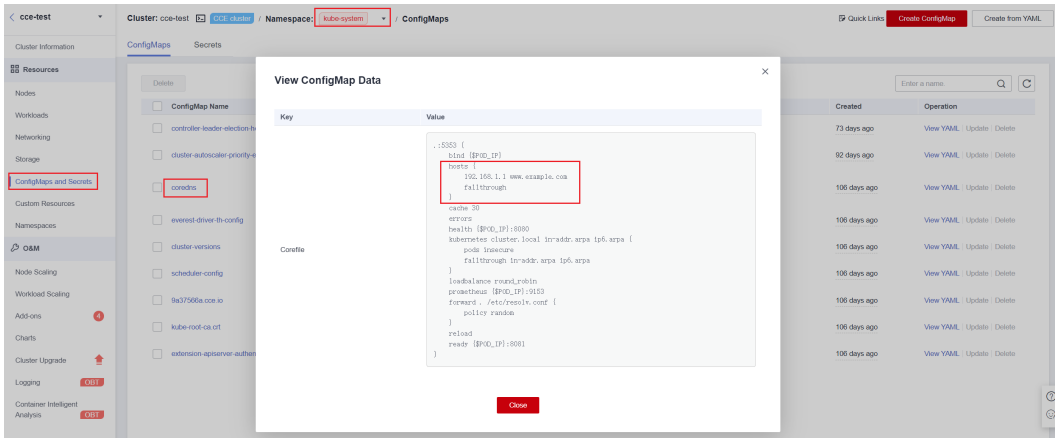

AVISO

O campo **fallthrough** deve ser configurado. **fallthrough** indica que quando o nome de domínio a ser resolvido não pode ser encontrado no arquivo hosts, a tarefa de resolução é transferida para o próximo plug-in de CoreDNS. Se **fallthrough** não for especificado, a tarefa termina e a resolução do nome de domínio pára. Como resultado, a resolução de nome de domínio no cluster falha.

Para obter detalhes sobre como configurar o arquivo hosts, visite **https://coredns.io/plugins/hosts/**.

Passo 4 Clique em **OK**.

Passo 5 Escolha **ConfigMaps and Secrets** no painel de navegação, selecione o namespace do **kube-system** e veja os dados do ConfigMap do CoreDNS para verificar se a atualização foi bem-sucedida.



O conteúdo do Corefile correspondente é o seguinte:

```
.:5353 {
    bind {$POD_IP}
    hosts {
      192.168.1.1 www.example.com
      fallthrough
    }
```



```
    cache 30
    errors
    health {$POD_IP}:8080
    kubernetes cluster.local in-addr.arpa ip6.arpa {
        pods insecure
        fallthrough in-addr.arpa ip6.arpa
    }
    loadbalance round_robin
    prometheus {$POD_IP}:9153
    forward . /etc/resolv.conf {
        policy random
    }
    reload
    ready {$POD_IP}:8081
}
```

----Fim

Adicionar a configuração de reescrita do CoreDNS para apontar o nome de domínio para Serviços no cluster

Use o plug-in de reescrita do CoreDNS para resolver um nome de domínio especificado para o nome de domínio de um Serviço. Por exemplo, a solicitação para acessar o nome de domínio example.com é redirecionada para o nome de domínio example.default.svc.cluster.local, ou seja, o serviço de exemplo no namespace padrão.

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster.

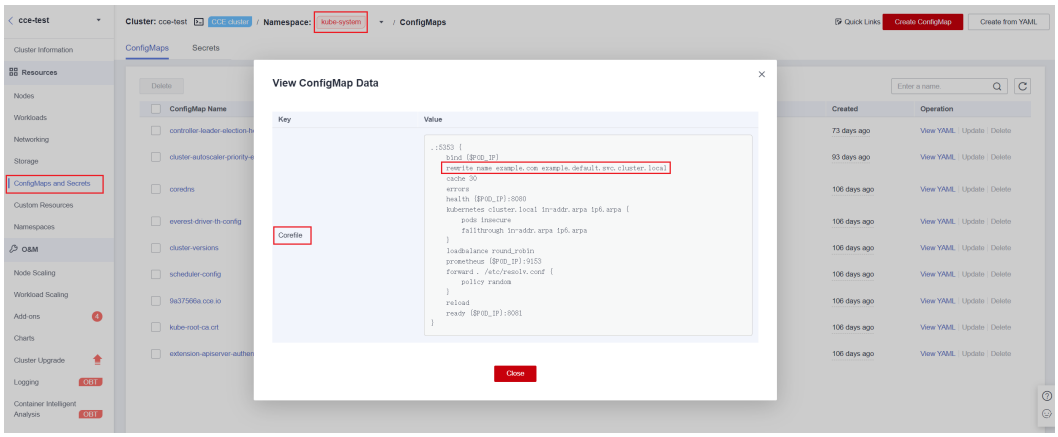
Passo 2 No painel de navegação, escolha **Add-ons**. Na página exibida, clique em **Edit** em **CoreDNS**.

Passo 3 Edite a configuração avançada em **Parameters** e adicione o seguinte conteúdo ao campo **plugins**:

```
{
   "name": "rewrite",
   "parameters": "name example.com example.default.svc.cluster.local"
}
```

Passo 4 Clique em **OK**.

Passo 5 Escolha **ConfigMaps and Secrets** no painel de navegação, selecione o namespace do **kube-system** e veja os dados do ConfigMap do CoreDNS para verificar se a atualização foi bem-sucedida.



Conteúdo do Corefile correspondente:

```
.:5353 {
    bind {$POD_IP}
    rewrite name example.com example.default.svc.cluster.local
    cache 30
    errors
    health {$POD_IP}:8080
    kubernetes cluster.local in-addr.arpa ip6.arpa {
        pods insecure
        fallthrough in-addr.arpa ip6.arpa
    }
    loadbalance round_robin
    prometheus {$POD_IP}:9153
    forward . /etc/resolv.conf {
        policy random
    }
    reload
    ready {$POD_IP}:8081
}
```

----Fim

Usar CoreDNS para DNS autoconstruído em cascata

Por padrão, o CoreDNS usa o arquivo **/etc/resolv.conf** do nó para resolução. Você também pode alterar o endereço de resolução para o do DNS externo.

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster.

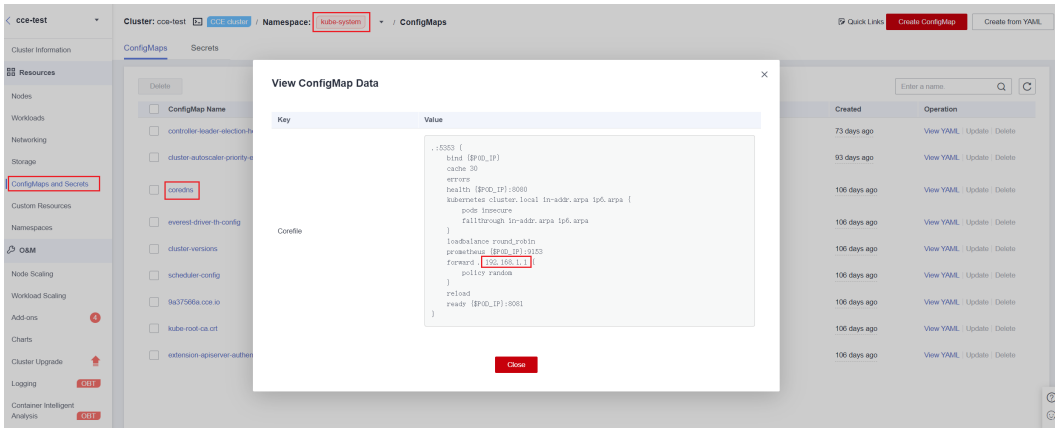
Passo 2 No painel de navegação, escolha **Add-ons**. Na página exibida, clique em **Edit** em **CoreDNS**.

Passo 3 Edite a configuração avançada em **Parameters** e modifique o seguinte conteúdo no campo **plugins**:

```
{
    "configBlock": "policy random",
    "name": "forward",
    "parameters": ". 192.168.1.1"
}
```


Passo 4 Clique em **OK**.

Passo 5 Escolha **ConfigMaps and Secrets** no painel de navegação, selecione o namespace do **kube-system** e veja os dados do ConfigMap do CoreDNS para verificar se a atualização foi bem-sucedida.



O conteúdo do Corefile correspondente é o seguinte:

```
.:5353 {
    bind {$POD_IP}
    cache 30
    errors
    health {$POD_IP}:8080
    kubernetes cluster.local in-addr.arpa ip6.arpa {
        pods insecure
        fallthrough in-addr.arpa ip6.arpa
    }
    loadbalance round_robin
    prometheus {$POD_IP}:9153
    forward . 192.168.1.1 {
        policy random
    }
    reload
    ready {$POD_IP}:8081
}
```

----Fim

7.5.4 Uso do DNSCache do NodeLocal para melhorar o desempenho do DNS

Desafios

Quando o número de solicitações de DNS em um cluster aumenta, a carga de CoreDNS aumenta e os seguintes problemas podem ocorrer:

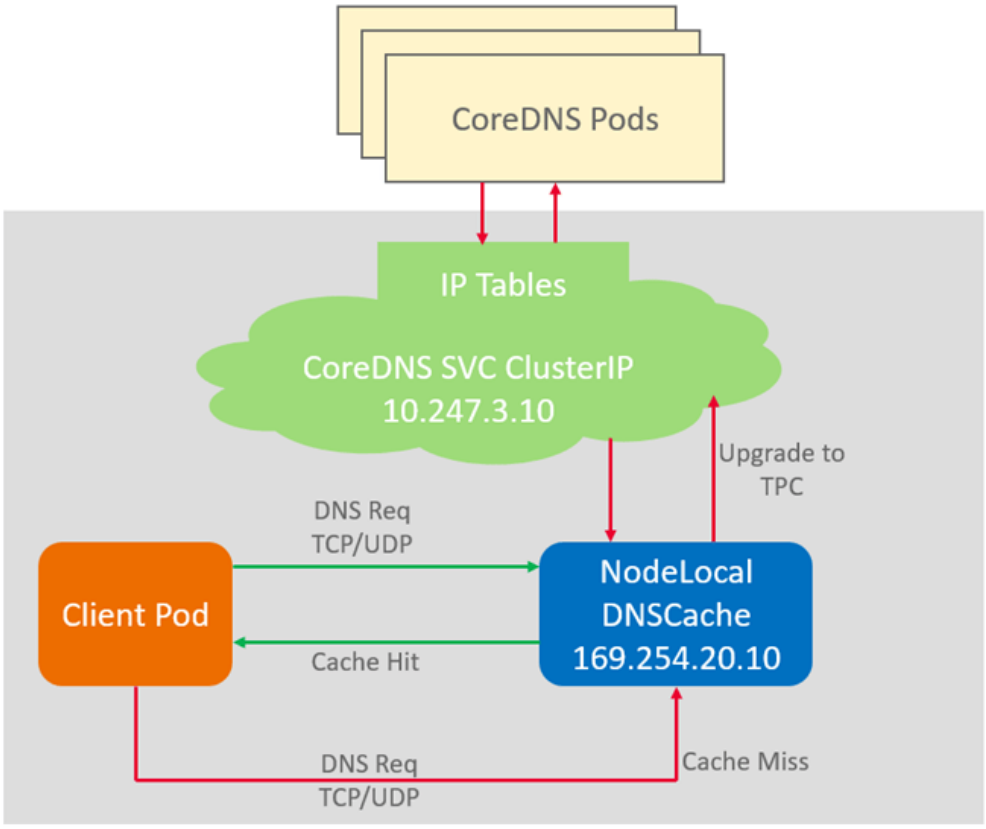
- Aumento do atraso: o CoreDNS precisa processar mais solicitações, o que pode retardar a consulta do DNS e afetar o desempenho do serviço.
- Aumento do uso de recursos: para garantir o desempenho do DNS, o CoreDNS exige especificações mais altas.

Solução

Para minimizar o impacto do atraso do DNS, implemente o DNSCache do NodeLocal no cluster para melhorar a estabilidade e o desempenho da rede. O DNSCache do NodeLocal executa um proxy de cache do DNS em nós de cluster. Todos os pods com configurações de DNS usam o proxy de cache do DNS em execução em nós em vez do serviço CoreDNS para resolução de nome de domínio. Isso reduz a carga do CoreDNS e melhora o desempenho do DNS do cluster.

Depois que o DNSCache do NodeLocal é habilitado, uma consulta de DNS percorre o caminho, conforme mostrado abaixo.

Figura 7-35 Caminho de consulta de DNSCache no NodeLocal



Restrições

node-local-dns-injection é um rótulo de sistema usado pelo DNSCache do NodeLocal. Use esse rótulo somente nos cenários descritos em **Impedir a injeção automática de DNSConfig**.

Instalar o complemento

O CCE fornece um complemento **NodeLocal DNSCache** para você instalar o DNSCache do NodeLocal.

NOTA

- O complemento node-local-dns suporta apenas clusters de v1.19 e posteriores.
- O DNSCache do NodeLocal serve como um proxy de cache transparente para o CoreDNS e não fornece plug-ins como hosts ou reescrita. Se você quiser ativar esses plug-ins, modifique as configurações do CoreDNS.
- Os pods não podem ser injetados automaticamente em namespaces do sistema, como o kube-system.

Passo 1 (Opcional) Modifique a configuração do CoreDNS para que o CoreDNS use preferencialmente o UDP para se comunicar com o servidor DNS upstream.

O DNSCache do NodeLocal usa o TCP para se comunicar com o CoreDNS. O CoreDNS se comunica com o servidor DNS upstream com base no protocolo usado pela fonte da solicitação. No entanto, o servidor de nuvem não suporta TCP. Para usar o DNSCache do NodeLocal, modifique a configuração do CoreDNS para que o UDP seja usado preferencialmente para se comunicar com o servidor DNS upstream, evitando exceções de resolução.

Realize as operações a seguir. No complemento avançado, especifique **prefer_udp** como o protocolo usado pelas solicitações. Após a modificação, o CoreDNS usa preferencialmente o UDP para se comunicar com o sistema upstream.

1. Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster.
2. No painel de navegação, escolha **Add-ons**. Na página exibida, clique em **Edit** em **CoreDNS**.
3. Edite a configuração avançada em **Parameters** e o seguinte conteúdo para o campo **plugins**:

```
{
    "configBlock": "prefer_udp",
    "name": "forward",
    "parameters": ". /etc/resolv.conf"
}
```

Passo 2 Efetue logon no console do CCE e acesse o console do cluster. Escolha **Add-ons** no painel de navegação, localize **node-local-dns** à direita e clique em **Install**.

Passo 3 Na página **Install Add-on**, selecione as especificações do complemento e defina os parâmetros relacionados.

- **enable_dnsconfig_admission**: depois que essa função for ativada, um controlador de injeção dinâmica de DNSConfig será criado. O controlador intercepta solicitações de criação de pods no namespace rotulado com **node-localdns-injection=enabled** com base em Admission Webhook e configura automaticamente **Pod dnsConfig** que usa o cache do DNS. Se essa função estiver desabilitada ou se o pod pertencer a um namespace que não seja de destino, será necessário configurar manualmente DNSConfig para o pod.

- **Target Namespace**: esse parâmetro está disponível depois que **DNSConfig Automatic Injection** está ativada. Somente NodeLocal DNSCache da v1.3.0 ou posterior suporta essa função.
  - **All Enabled**: o CCE adiciona o rótulo **node-local-dns-injection=enabled** a todos os namespaces criados, excluindo os internos (como o **kube-system**), identifica solicitações de criação de namespace e adiciona automaticamente o rótulo aos namespaces recém-criados.
  - **Manual configuration**: você deve adicionar manualmente o rótulo **node-local-dns-injection=enabled** aos namespaces que exigem a injeção de DNSConfig. Para mais detalhes, consulte **Gerenciar rótulos de namespace**.

Passo 4 Clique em **Install**.

----Fim

Usar o DNSCache do NodeLocal

Por padrão, as solicitações de aplicativos são enviadas por meio do proxy de CoreDNS. Para usar node-local-dns como proxy de cache de DNS, use um dos seguintes métodos:

- Injeção automática: configure automaticamente o campo **dnsConfig** do pod ao criar o pod. (Pods não podem ser injetados automaticamente em namespaces do sistema, como kube-system.)
- Configuração manual: configure manualmente o campo **dnsConfig** do pod.

Injeção automática

Devem ser satisfeitas as seguintes condições:

- **A injeção automática de DNSConfig** foi ativada durante a instalação do complemento.
- O rótulo **node-local-dns-injection=enabled** foi adicionado ao namespace. Por exemplo, execute o seguinte comando para adicionar o rótulo ao namespace **default**:

  kubectl label namespace *default* **node-local-dns-injection=enabled**
- O novo pod não é executado em namespaces do sistema, como namespace de kube-system e kube-public.
- O rótulo **node-local-dns-injection=disabled** para desabilitar a injeção de DNS não é adicionado ao novo pod.
- O novo pod usa a rede de host e **DNSPolicy** é **ClusterFirstWithHostNet**. Como alternativa, o pod não usa a rede host e **DNSPolicy** é **ClusterFirst**.

Depois que a injeção automática é ativada, as seguintes configurações do **dnsConfig** são adicionadas automaticamente ao pod criado. Além do endereço DNSCache do NodeLocal 169.254.20.10, o endereço de CoreDNS 10.247.3.10 é adicionado a **nameservers**, garantindo alta disponibilidade do servidor DNS do serviço.

```
...
  dnsConfig:
    nameservers:
      - 169.254.20.10
      - 10.247.3.10
    searches:
      - default.svc.cluster.local
      - svc.cluster.local
      - cluster.local
    options:
      - name: timeout
```



```
        value: ''
      - name: ndots
        value: '5'
      - name: single-request-reopen
...
```

Configuração manual

Adicione manualmente as configurações de **dnsConfig** ao pod.

Crie um pod e adicione o endereço IP do DNSCache do NodeLocal 169.254.20.10 à configuração de nameservers de DNSConfig.

NOTA

Os endereços de DNSCache do NodeLocal de diferentes tipos de cluster são os seguintes:

- Cluster do CCE: 169.254.20.10
- Cluster do CCE Turbo: 169.254.1.1

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - image: nginx:alpine
    name: container-0
  dnsConfig:
    nameservers:
    - 169.254.20.10
    - 10.247.3.10
    searches:
    - default.svc.cluster.local
    - svc.cluster.local
    - cluster.local
    options:
    - name: ndots
      value: '2'
  imagePullSecrets:
  - name: default-secret
```

Impedir a injeção automática de DNSConfig

Para impedir a injeção automática de DNSConfig para uma carga de trabalho, adicione **node-local-dns-injection: disabled** ao campo **labels** no modelo de pod. Exemplo:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: test
  namespace: default
spec:
  replicas: 2
  selector:
    matchLabels:
      app: test
  template:
    metadata:
      labels:
        app: test
        node-local-dns-injection: disabled   # Prevent automatic DNSConfig
injection.
    spec:
      containers:
        - name: container-1
          image: nginx:latest
          imagePullPolicy: IfNotPresent
```



```
      imagePullSecrets:
        - name: default-secret
```

7.6 Configurações de rede de contêiner

7.6.1 Rede host

Cenário

O Kubernetes permite que os pods usem diretamente a rede host/ de nó. Quando um pod é configurado com **hostNetwork: true**, as aplicações em execução no pod podem visualizar diretamente a interface da rede host onde o pod está localizado.

Configuração

Adicione **hostNetwork: true** à definição do pod.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      hostNetwork: true
      containers:
      - image: nginx:alpine
        name: nginx
      imagePullSecrets:
      - name: default-secret
```

A configuração é bem-sucedida se o IP do pod for o mesmo que o IP do nó.

```
$ kubectl get pod -owide
NAME                     READY   STATUS    RESTARTS   AGE     IP
NODE        NOMINATED NODE   READINESS GATES
nginx-6fdf99c8b-6wwft   1/1     Running   0          3m41s   10.1.0.55
10.1.0.55   <none>           <none>
```

Precauções

Se um pod usar a rede host, ele ocupará uma porta de host. O IP do pod é o IP do host. Para usar a rede host, você deve confirmar que os pods não entram em conflito uns com os outros em termos das portas de host que ocupam. Não use a rede host a menos que você saiba exatamente qual porta de host é usada por qual pod.

Ao usar a rede host, você acessa o nó para acessar um pod nele. Portanto, **permita o acesso da porta do grupo de segurança do nó**. Caso contrário, o acesso falha.

Além disso, o uso da rede host exige que você reserve portas de host para os pods. Ao usar uma Implementação para implementar pods do tipo hostNetwork, verifique se **o número de pods não excede o número de nós**. Caso contrário, vários pods serão agendados no nó e eles

não serão iniciados devido a conflitos de portas. Por exemplo, no exemplo anterior de nginx YAML, se dois pods (configurando **replicas** como **2**) forem implementados em um cluster com apenas um nó, um pod não poderá ser criado. Os logs do pod mostrarão que o Nginx não pode ser iniciado porque a porta está ocupada.

⚠ **CUIDADO**

Não agende vários pods que usam a rede host no mesmo nó. Caso contrário, quando um Serviço ClusterIP é criado para acessar um pod, o endereço IP do cluster não pode ser acessado.

```
$ kubectl get deploy
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
nginx   1/2     2            1           67m
$ kubectl get pod
NAME                       READY   STATUS             RESTARTS   AGE
nginx-6fdf99c8b-6wwft      1/1     Running            0          67m
nginx-6fdf99c8b-rglm7      0/1     CrashLoopBackOff   13         44m
$ kubectl logs nginx-6fdf99c8b-rglm7
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to
perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-
default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/
default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/
conf.d/default.conf
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2022/05/11 07:18:11 [emerg] 1#1: bind() to 0.0.0.0:80 failed (98: Address in use)
nginx: [emerg] bind() to 0.0.0.0:80 failed (98: Address in use)
2022/05/11 07:18:11 [emerg] 1#1: bind() to [::]:80 failed (98: Address in use)
nginx: [emerg] bind() to [::]:80 failed (98: Address in use)
2022/05/11 07:18:11 [emerg] 1#1: bind() to 0.0.0.0:80 failed (98: Address in use)
nginx: [emerg] bind() to 0.0.0.0:80 failed (98: Address in use)
2022/05/11 07:18:11 [emerg] 1#1: bind() to [::]:80 failed (98: Address in use)
nginx: [emerg] bind() to [::]:80 failed (98: Address in use)
2022/05/11 07:18:11 [emerg] 1#1: bind() to 0.0.0.0:80 failed (98: Address in use)
nginx: [emerg] bind() to 0.0.0.0:80 failed (98: Address in use)
2022/05/11 07:18:11 [emerg] 1#1: bind() to [::]:80 failed (98: Address in use)
nginx: [emerg] bind() to [::]:80 failed (98: Address in use)
2022/05/11 07:18:11 [emerg] 1#1: bind() to 0.0.0.0:80 failed (98: Address in use)
nginx: [emerg] bind() to 0.0.0.0:80 failed (98: Address in use)
2022/05/11 07:18:11 [emerg] 1#1: bind() to [::]:80 failed (98: Address in use)
nginx: [emerg] bind() to [::]:80 failed (98: Address in use)
2022/05/11 07:18:11 [emerg] 1#1: bind() to 0.0.0.0:80 failed (98: Address in use)
nginx: [emerg] bind() to 0.0.0.0:80 failed (98: Address in use)
2022/05/11 07:18:11 [emerg] 1#1: bind() to [::]:80 failed (98: Address in use)
nginx: [emerg] bind() to [::]:80 failed (98: Address in use)
2022/05/11 07:18:11 [emerg] 1#1: still could not bind()
nginx: [emerg] still could not bind()
```

7.6.2 Configuração da limitação da taxa de QoS para acesso entre pods

Cenário

A preempção de largura de banda ocorre entre diferentes contêineres implementados no mesmo nó, o que pode causar jitter de serviço. Você pode configurar a limitação da taxa de QoS para o acesso entre pods impedir este problema.

Restrições

A seguir, mostra as restrições na configuração do limite de taxa para o acesso entre pods:

Tipo de restrição	Modelo da rede de túnel	Modelo de rede da VPC	Modelo de rede de Cloud Native 2.0
Versões suportadas	Todas as versões	Clusters da v1.19.10 e posterior	Clusters da v1.19.10 e posterior
Tipos de tempo de execução suportados	Somente contêineres comuns (runC como o tempo de execução do contêiner) são suportados. Não há suporte para contêineres seguros.	Somente contêineres comuns (runC como o tempo de execução do contêiner) são suportados. Contêineres seguros (Kata como o runtime do contêiner) não são suportados.	Somente contêineres comuns (runC como o tempo de execução do contêiner) são suportados. Contêineres seguros (Kata como o runtime do contêiner) não são suportados.
Tipos de pod suportados	Apenas pods non-HostNetwork		
Cenários suportados	Acesso entre pods, pods que acessam nós e pods que acessam serviços		
Restrições	Nenhuma	Nenhuma	● Os pods acessam os blocos CIDR do serviço de nuvem externo 100.64.0.0/10 e 214.0.0.0/8. ● Taxa de tráfego limitante da verificação de integridade
Limite da taxa superior	Valor mínimo entre o limite superior de largura de banda e 34 Gbit/s	Valor mínimo entre o limite superior de largura de banda e 4,3 Gbit/s	Valor mínimo entre o limite superior de largura de banda e 4,3 Gbit/s

Tipo de restrição	Modelo da rede de túnel	Modelo de rede da VPC	Modelo de rede de Cloud Native 2.0
Limite da taxa inferior	Somente o limite de taxa de Kbit/s ou superior é suportado.	Atualmente, apenas o limite de taxa de Mbit/s ou superior é suportado.	

Usar o kubectl

Você pode adicionar anotações a uma carga de trabalho para especificar sua largura de banda de saída e entrada.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: test
  namespace: default
  labels:
    app: test
spec:
  replicas: 2
  selector:
    matchLabels:
      app: test
  template:
    metadata:
      labels:
        app: test
      annotations:
        kubernetes.io/ingress-bandwidth: 100M
        kubernetes.io/egress-bandwidth: 100M
    spec:
      containers:
        - name: container-1
          image: nginx:alpine
          imagePullPolicy: IfNotPresent
      imagePullSecrets:
        - name: default-secret
```

- **kubernetes.io/ingress-bandwidth**: largura de banda de entrada do pod
- **kubernetes.io/egress-bandwidth**: largura de banda de saída do pod

Se esses dois parâmetros não forem especificados, a largura de banda não será limitada.

NOTA

Depois de modificar o limite de largura de banda de entrada ou saída de um pod, reinicie o contêiner para que a modificação tenha efeito. Depois que as anotações forem modificadas em um pod não gerenciado por cargas de trabalho, o contêiner não será reiniciado, para que os limites de largura de banda não entrem em vigor. Você pode criar um pod novamente ou reiniciar manualmente o contêiner.

7.6.3 Configurações de rede de túnel de contêiner

7.6.3.1 Políticas de rede

As políticas de rede são projetadas pelo Kubernetes para restringir o acesso ao pod. É equivalente a um firewall na camada de aplicação para melhorar a segurança da rede. As capacidades suportadas pelas políticas de rede dependem das capacidades dos suplementos de rede do cluster.

Por padrão, se um namespace não tiver nenhuma política, os pods no namespace aceitarão tráfego de qualquer origem e enviarão tráfego para qualquer destino.

As políticas de rede são classificadas nos seguintes tipos:

- **namespaceSelector**: seleciona namespaces específicos para os quais todos os pods devem ser permitidos como fontes de ingress ou destinos de egress.
- **podSelector**: seleciona pods específicos no mesmo namespace da política de rede que deve ser permitida como fontes de ingress ou destinos de egress.
- **ipBlock**: seleciona blocos de IP específicos para permitir como fontes de ingress ou destinos de egress.

Restrições

- Apenas os clusters que usam o modelo de rede de túnel suportam políticas de rede. As políticas de rede são classificadas nos seguintes tipos:
  - Ingress: todas as versões suportam este tipo.
  - Egress: este tipo de regra não pode ser definido atualmente.
- O isolamento de rede não é suportado para endereços IPv6.

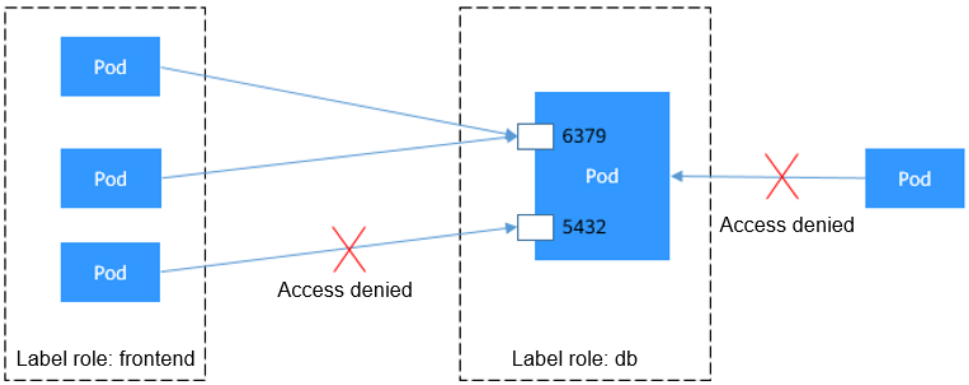
Usar regras de ingress

- **Usar o podSelector para especificar o escopo de acesso**

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: test-network-policy
  namespace: default
spec:
  podSelector:                  # The rule takes effect for pods with the
role=db label.
    matchLabels:
      role: db
  ingress:                      # This is an ingress rule.
  - from:
    - podSelector:              # Only traffic from the pods with the
"role=frontend" label is allowed.
        matchLabels:
          role: frontend
    ports:                      # Only TCP can be used to access port 6379.
    - protocol: TCP
      port: 6379
```

A figura a seguir mostra como o podSelector funciona.

Figura 7-36 podSelector

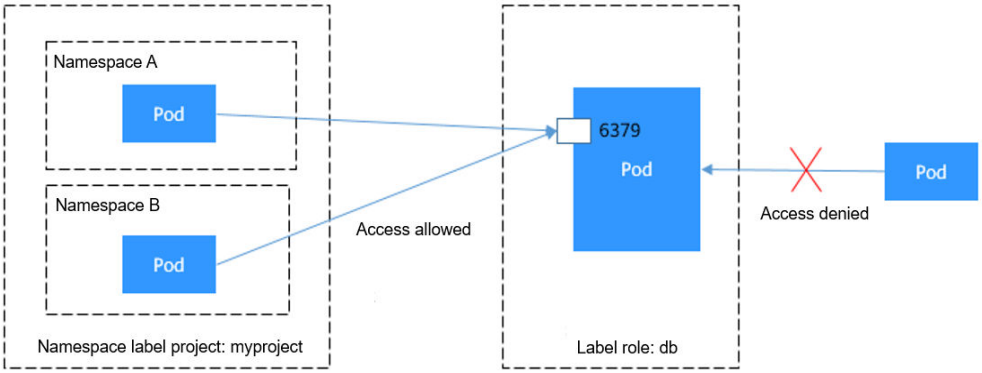


- **Usar namespaceSelector para especificar o escopo de acesso**

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: test-network-policy
spec:
  podSelector:                  # The rule takes effect for pods with the
role=db label.
    matchLabels:
      role: db
  ingress:                      # This is an ingress rule.
  - from:
    - namespaceSelector:        # Only traffic from the pods in the namespace
with the "project=myproject" label is allowed.
        matchLabels:
          project: myproject
    ports:                      # Only TCP can be used to access port 6379.
    - protocol: TCP
      port: 6379
```

A figura a seguir mostra como o namespaceSelector funciona.

Figura 7-37 namespaceSelector



Criar uma política de rede no console

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster.

Passo 2 Escolha **Networking** no painel de navegação, clique na guia **Network Policies** e o clique em **Create Network Policy** no canto direito superior.

- **Policy Name**: especifique um nome de política de rede.
- **Namespace**: selecione um namespace no qual a política de rede é aplicada.
- **Selector**: insira um rótulo, selecione o pod a ser associado e clique em **Add**. Você também pode clicar em **Reference Workload Label** para fazer referência ao rótulo de uma carga de trabalho existente.
- **Inbound Rule**: clique em + para adicionar uma regra de entrada. Para obter detalhes sobre as configurações de parâmetros, consulte **Tabela 7-47**.

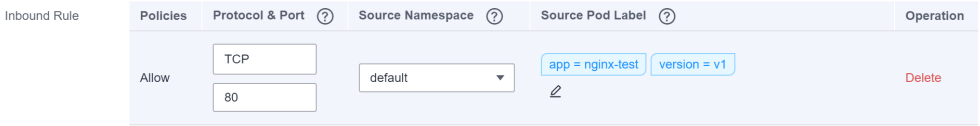


Tabela 7-47 Adicionar uma regra de entrada

Parâmetro	Descrição
Protocol & Port	Selecione o tipo de protocolo e a porta. Atualmente, TCP e UDP são suportados.
Source Namespace	Selecione um namespace cujos objetos possam ser acessados. Se esse parâmetro não for especificado, o objeto pertence ao mesmo namespace da diretiva atual.
Source Pod Label	Permita o acesso aos pods com este rótulo. Se esse parâmetro não for especificado, todos os pods no namespace poderão ser acessados.

Passo 3 Clique em **OK**.

----Fim

7.6.4 Configurações de Cloud Native Network 2.0

7.6.4.1 Políticas de grupo de segurança

No Cloud Native Network 2.0, os pods usam ENIs da VPC ou sub-ENIs para rede. Você pode vincular diretamente grupos de segurança e EIPs aos pods. Para vincular pods do CCE a grupos de segurança, o CCE fornece um objeto de recurso personalizado chamado **SecurityGroup**. Usando esse objeto de recurso, você pode personalizar o isolamento de segurança para cargas de trabalho.

Restrições

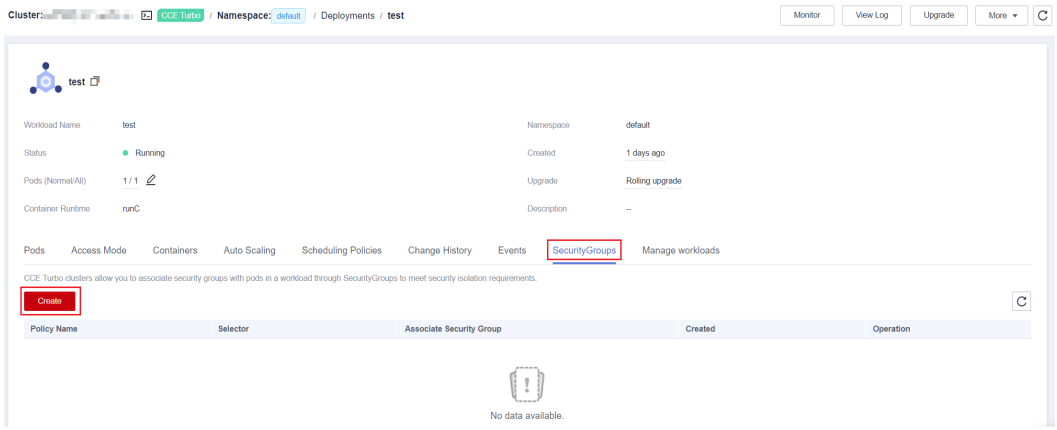
- Esta função é apoiada para os clusters do CCE Turbo de v1.19 e mais recente. Atualize seus clusters do CCE Turbo se suas versões forem anteriores à v1.19.
- Uma carga de trabalho pode ser vinculada a um máximo de cinco grupos de segurança.

Usar o console

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster.

Passo 2 No painel de navegação, escolha **Workloads**. Na página exibida, clique no nome da carga de trabalho desejada.

Passo 3 Alterne para a página de guia **SecurityGroups** e clique em **Create**.



Passo 4 Defina os parâmetros conforme descrito em **Tabela 7-48**.

Tabela 7-48 Parâmetros de configuração

Parâmetro	Descrição	Exemplo de valor
Security Group Policy Name	Insira um nome de política de segurança. Insira 1 a 63 caracteres. O valor deve começar com uma letra minúscula e não pode terminar com um hífen (-). Somente letras minúsculas, dígitos e hifens (-) são permitidos.	security-group
Associate Security Group	O grupo de segurança selecionado será vinculado à ENI ou à ENI suplementar da carga de trabalho selecionada. Um máximo de cinco grupos de segurança podem ser selecionados na lista suspensa. Você deve selecionar um ou vários grupos de segurança para criar um SecurityGroup. Se nenhum grupo de segurança não tiver sido criado, clique em **Create Security Group**. Depois que o grupo de segurança for criado, clique no botão de atualizar. **AVISO** ● Um máximo de cinco grupos de segurança podem ser selecionados. ● Passe o cursor em ⓘ ao lado do nome do grupo de segurança e você pode exibir detalhes sobre o grupo de segurança.	64566556-bd6f-48fb-b2c6-df8f44617953 5451f1b0-bd6f-48fb-b2c6-df8f44617953

Passo 5 Depois de definir os parâmetros, clique em **OK**.

Depois que o grupo de segurança é criado, o sistema retorna automaticamente para a página de lista de grupo de segurança onde você pode ver o novo grupo de segurança.

----Fim

Usar o kubectl

Passo 1 Use o kubectl para se conectar ao cluster. Para mais detalhes, consulte **Conexão a um cluster usando o kubectl**.

Passo 2 Crie um arquivo de descrição chamado **securitygroup-demo.yaml**.

vi securitygroup-demo.yaml

Por exemplo, crie o seguinte SecurityGroup para vincular todas as cargas de trabalho nginx com dois grupos de segurança 64566556-bd6f-48fb-b2c6-df8f44617953 e 5451f1b0-bd6f-48fb-b2c6-df8f44617953 que foram criados antecipadamente. Um exemplo é o seguinte:

```
apiVersion: crd.yangtse.cni/v1
kind: SecurityGroup
metadata:
  name: demo
  namespace: default
spec:
  podSelector:
    matchLabels:
      app: nginx
  securityGroups:
  - id: 64566556-bd6f-48fb-b2c6-df8f44617953
  - id: 5451f1b0-bd6f-48fb-b2c6-df8f44617953
```

Tabela 7-49 descreve os parâmetros no arquivo YAML.

Tabela 7-49 Descrição

Campo	Descrição	Obrigatório
apiVersion	Versão da API. O valor é **crd.yangtse.cni/v1**.	Sim
kind	Tipo do objeto a ser criado.	Sim
metadata	Definição de metadados do objeto de recurso.	Sim
name	Nome do SecurityGroup.	Sim
namespace	Nome do namespace.	Sim
spec	Descrição detalhada do SecurityGroup.	Sim
podSelector	Usado para definir a carga de trabalho a ser associada a grupos de segurança no SecurityGroup.	Sim
securityGroups	ID do grupo de segurança.	Sim

Passo 3 Execute o seguinte comando para criar o SecurityGroup:

kubectl create -f securitygroup-demo.yaml

Se as informações a seguir forem exibidas, o SecurityGroup está sendo criada.

```
securitygroup.crd.yangtse.cni/demo created
```

Passo 4 Execute o seguinte comando para exibir o SecurityGroup:

kubectl get sg

Se o nome do SecurityGroup criado for **demo** na saída do comando, o SecurityGroup será criado com sucesso.

```
NAME                          POD-SELECTOR                          AGE
all-no                        map[matchLabels:map[app:nginx]]       4h1m
s001test                      map[matchLabels:map[app:nginx]]       19m
demo                          map[matchLabels:map[app:nginx]]       2m9s
```

----Fim

7.6.4.2 NetworkAttachmentDefinition

Cenário

Em um cluster do CCE Turbo, você pode definir a sub-rede e o grupo de segurança de um contêiner por namespace usando NetworkAttachmentDefinition, um recurso de **CRD** no cluster. Depois que NetworkAttachmentDefinition é configurado para um namespace, os pods no namespace suportam as seguintes funções:

- Vincular um contêiner com uma sub-rede: o endereço IP do pod é restrito em um bloco CIDR específico. Namespaces diferentes podem ser isolados uns dos outros.
- Vincular um contêiner a um grupo de segurança: as regras de grupo de segurança podem ser definidas para pods no mesmo namespace para personalizar as políticas de acesso.

Restrições

- NetworkAttachmentDefinition está disponível somente em clusters do CCE Turbo de v1.23.8-r0, v1.25.3-r0 e posterior.
- Somente **default-network** oferece suporte ao pré-aquecimento da ENI. As sub-redes de contêiner definidas pelo usuário não oferecem suporte ao pré-aquecimento da ENI. Se o pré-aquecimento da ENI não estiver habilitado, a criação da instância de carga de trabalho ficará mais lenta. Portanto, essa função não é aplicável a cenários de criação de pods de alto desempenho.
- Para excluir um NetworkAttachmentDefinition exclua pods (com a anotação chamada **cni.yangtse.io/network-status**) criados usando a configuração no namespace correspondente primeiro. Para mais detalhes, consulte **Excluir uma configuração de rede**.

Usar o console do CCE

Passo 1 Efetue logon no console do CCE.

Passo 2 Clique no nome do cluster para acessar o console do cluster. Escolha **System Configuration** no painel de navegação e clique na guia **Network Configuration**.

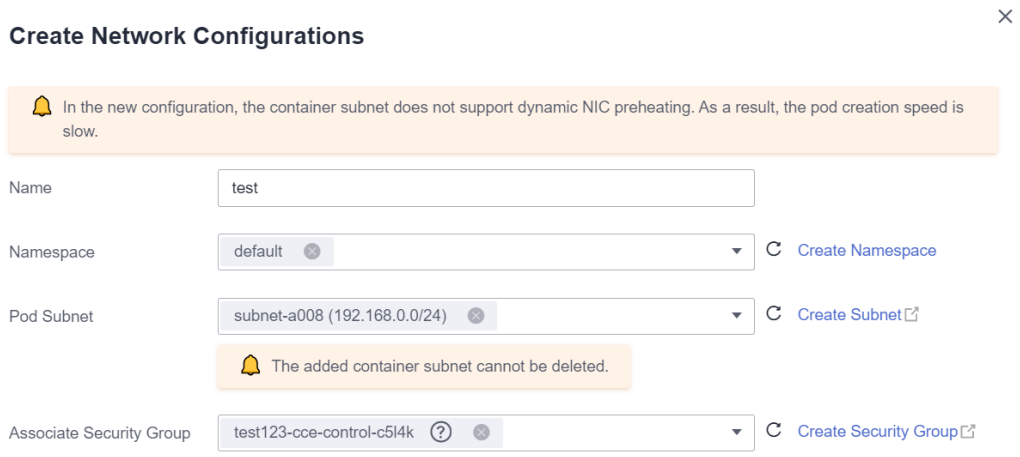
Figura 7-38 Configurações da rede

> **NOTA**
>
> Cada cluster tem uma **default-network** para namespaces sem sub-redes de contêiner. A sub-rede de contêiner padrão exibida nas informações de rede na área de configuração de rede é a sub-rede de contêiner em **default-network**. A **default-network** não pode ser excluída.

Passo 3 Clique em **Create Network Configurations** no canto superior direito. Configure os parâmetros básicos na caixa de diálogo exibida.

- **Name**: insira um nome que contenha no máximo 253 caracteres. Não use **default-network**, **default**, **mgnt0** e **mgnt1**.
- **Namespace**: selecione um namespace. Os namespaces de diferentes configurações devem ser exclusivos. Se nenhum namespace estiver disponível, clique em **Create Namespace** para criar um.
- **Pod Subnet**: selecione uma sub-rede. Se nenhuma sub-rede estiver disponível, clique em **Create Subnet** para criar uma sub-rede. Depois que a sub-rede for criada, clique no botão de atualizar. Um máximo de 20 sub-redes podem ser selecionadas.
- **Associate Security Group**: o valor padrão é o grupo de segurança da ENI do contêiner. Você também pode clicar em **Create Security Group** para criar um. Depois que o grupo de segurança for criado, clique no botão de atualizar. Um máximo de cinco grupos de segurança podem ser selecionados.

Figura 7-39 Criar uma configuração de rede



Passo 4 Clique em **Create**. Após a conclusão da criação, você será redirecionado para a lista de configuração de rede. Você pode ver que a sub-rede recém-adicionada está na lista.

----Fim

Usar o kubectl

Esta seção descreve como criar um NAD usando kubectl.

Passo 1 Use o kubectl para se conectar ao cluster. Para mais detalhes, consulte **Conexão a um cluster usando o kubectl**.

Passo 2 Modifique o arquivo **networkattachment-test.yaml**.

vi networkattachment-test.yaml

```
apiVersion: k8s.cni.cncf.io/v1
kind: NetworkAttachmentDefinition
```



```
metadata:
  annotations:
    yangtse.io/project-id: 05e38**
  name: example
  namespace: kube-system
spec:
  config:
  '{
    "type":"eni-neutron",
    "args":{
      "securityGroups":"41891**",
      "subnets":[
        {
          "subnetID":"27d95**"
        }
      ]
    },
    "selector":{
      "namespaceSelector":{
        "matchLabels":{
          "kubernetes.io/metadata.name":"default"
        }
      }
    }
  }'
```

Tabela 7-50 Parâmetros principais

Parâmetro	Obrigatório	Tipo	Descrição
apiVersion	Sim	String	Versão da API. O valor é fixado em **k8s.cni.cncf.io/v1**.
kind	Sim	String	Tipo do objeto a ser criado. O valor é fixado em **NetworkAttachmentDefinition**.
yangtse.io/project-id	Sim	String	ID do projeto.
name	Sim	String	Nome do item de configuração.
namespace	Sim	String	Namespace do recurso de configuração. O valor é fixado para **kube-system**.
config	Sim	Objeto de Tabela 7-51	Conteúdo de configuração, que é uma cadeia no formato JSON.

Tabela 7-51 Parâmetros de config

Parâmetro	Obrigatório	Tipo	Descrição
type	Sim	String	O valor é fixado em **eni-neutron**.
args	Não	Tabela 7-52 object	Parâmetros de configuração.

Parâmetro	Obrigatório	Tipo	Descrição
selector	Não	Tabela 7-53 object	Namespace no qual a configuração entra em vigor.

Tabela 7-52 Parâmetros de args

Parâmetro	Obrigatório	Tipo	Descrição
securityGroups	Não	String	ID do grupo de segurança. Se nenhum grupo de segurança estiver planejado, selecione o mesmo grupo de segurança que está em **default-network**. Obter o valor: Efetue logon no console da VPC. No painel de navegação à esquerda, escolha **Access Control** > **Security Groups**. Clique no nome do grupo de segurança de destino e copie o ID na página da guia **Summary**.
subnets	Sim	Array of subnetID Objects	Lista de IDs de sub-rede de contêiner. Pelo menos um ID de sub-rede deve ser inserido. O formato é o seguinte: `[{"subnetID":"27d95**"},{"subnetID":"827bb**"},{"subnetID":"bdd6b**"}]` ID de sub-rede não usado pelo cluster na mesma VPC. Obter o valor: Efetue logon no console da VPC. No painel de navegação, escolha **Virtual Private Cloud** > **Subnets**. Clique no nome da sub-rede de destino e copie o **Subnet ID** na página de guia **Summary**.

Tabela 7-53 Parâmetros de selector

Parâmetro	Obrigatório	Tipo	Descrição
namespaceSelector	Não	matchLabels Object	Um seletor padrão do Kubernetes. Insira o rótulo do namespace no seguinte formato: `"matchLabels":{ "kubernetes.io/metadata.name":"default" }` Os namespaces de diferentes configurações não podem se sobrepor.

Passo 3 Crie um NetworkAttachmentDefinition.

kubectl create -f networkattachment-test.yaml

Se informações semelhantes às seguintes forem exibidas, o NetworkAttachmentDefinition foi criado.

```
networkattachmentdefinition.k8s.cni.cncf.io/example created
```

----Fim

Excluir uma configuração de rede

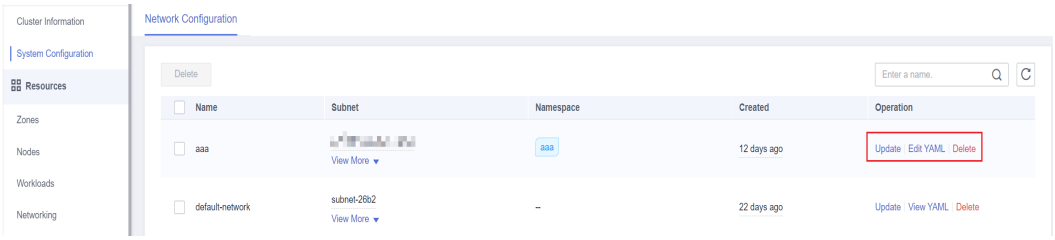
Você pode deletar a nova configuração de rede ou exibir seu arquivo YAML.

NOTA

Antes de excluir uma configuração de rede, exclua o contêiner correspondente à configuração. Caso contrário, a exclusão falha.

1. Execute o comando a seguir para filtrar o pod que usa a configuração no cluster (**example** é um nome de configuração de exemplo e você deve substituí-lo):
   ```
   kubectl get po -A -o=jsonpath="{.items[?(@.metadata.annotations.cni\.yangtse\.io/network-status=='[{\"name\":\"example\"}]')]['metadata.namespace', 'metadata.name']}"
   ```
   A saída do comando contém o nome do pod e o namespace associados à configuração.
2. Exclua o proprietário do pod. O proprietário pode ser uma Implementação, um StatefulSet ou uma Tarefa.

Figura 7-40 Gerenciar a configuração da rede



7.6.4.3 Configuração de um endereço IP estático para um pod

Cenários

No Cloud Native Network 2.0, cada pod é associado a uma ENI, fornecendo um endereço IP estático aos pods do StatefulSet (ENI de contêiner). Essa é uma prática comum no controle de acesso, registro de serviços, descoberta de serviços e auditoria de registros de endereços IP estáticos.

Por exemplo, se um serviço StatefulSet precisa controlar o acesso de um banco de dados em nuvem, você pode corrigir o endereço IP do pod do serviço e configurar o grupo de segurança do banco de dados em nuvem para permitir que apenas o endereço IP do serviço acesse o banco de dados.

Restrições

- Você pode configurar um endereço IP estático para um pod somente em clusters do CCE Turbo das seguintes versões:
  - v1.23: v1.23.7-r0 ou mais recente
  - v1.25: v1.25.3-r0 ou mais recente
  - v1.25 ou mais recente
- Atualmente, apenas pods de StatefulSet ou pods sem **ownerReferences** podem ser configuradas com endereços IP estáticos. As Implementações, os DaemonSets e outros tipos de cargas de trabalho não podem ser configurados com endereços IP estáticos. Além disso, pods com **HostNetwork** não podem ser configurados com endereços IP estáticos.
- Não configure endereços IP estáticos para serviços que não tenham requisitos específicos para endereços IP de pod. Caso contrário, a reconstrução do pod leva mais tempo e o uso do endereço IP diminui.
- As anotações do endereço IP estático do objeto pod não podem ser modificadas diretamente. Caso contrário, a modificação não terá efeito em segundo plano. Para modificar as anotações, modifique a configuração de **annotations** no campo **spec.template** da carga de trabalho de StatefulSet correspondente.
- Se não houver ENIs no nó em que o pod com um endereço IP estático é reconstruído e agendado (as ENIs pré-vinculadas também ocupam a cota de ENIs), as ENIs de endereço IP estático antecipam os ENIs pré-vinculadas. Neste caso, a pod começa um pouco lentamente. Se um nó usar um endereço IP estático, configure adequadamente a política de pré-vinculação dinâmica para o nó para garantir que nem todas as ENIs sejam pré-vinculadas.

Usar o kubectl

Você pode adicionar anotações a um StatefulSet para ativar ou desativar a função de endereço IP estático do pod.

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: nginx
spec:
  serviceName: nginx
  replicas: 3
  selector:
    matchLabels:
```



```
    app: nginx
template:
  metadata:
    labels:
      app: nginx
    annotations:
      pod.alpha.kubernetes.io/initialized: 'true'
      yangtse.io/static-ip: 'true'
      yangtse.io/static-ip-expire-no-cascading: 'false'
      yangtse.io/static-ip-expire-duration: 5m
  spec:
    containers:
      - name: container-0
        image: nginx:alpine
        resources:
          limits:
            cpu: 100m
            memory: 200Mi
          requests:
            cpu: 100m
            memory: 200Mi
    imagePullSecrets:
      - name: default-secret
```

Tabela 7-54 Anotações do endereço IP estático do pod

Anotação	Valor padrão	Descrição	Intervalo de valor
yangtse.io/static-ip	false	Especifica se o endereço IP estático de um pod deve ser ativado. Esta função é suportada apenas para pods de StatefulSet ou pods sem **ownerReferences**. Esta função está desativada por padrão.	**false** ou **true**
yangtse.io/static-ip-expire-duration	5m	Especifica o intervalo para recuperar a ENI expirada do endereço IP estático depois que o pod com um endereço IP estático for excluído.	O formato de tempo é Go time type, por exemplo, 1h30m e 5m. Para obter detalhes, consulte **Go time type**.
yangtse.io/static-ip-expire-no-cascading	false	Especifica se deve desativar a recuperação em cascata de cargas de trabalho do StatefulSet. O valor padrão é **false**, indicando que o endereço IP estático da ENI correspondente será excluído com a carga de trabalho StatefulSet. Se quiser manter o endereço IP estático de um novo StatefulSet com o mesmo nome durante o intervalo para recuperar a ENI expirada, defina o valor como **true**.	**false** ou **true**

7.6.4.4 Configuração de um EIP para um pod

Cenários

No Cloud Native Network 2.0, os pods usam ENIs da VPC ou sub-ENIs para rede. Você pode vincular diretamente EIPs a pods.

Para associar um EIP a um pod, basta definir o valor da anotação **yangtse.io/pod-with-eip** como **true** ao criar o pod. Em seguida, o EIP é automaticamente alocado e vinculado ao pod.

Restrições

- Para usar um EIP estático ou um EIP alocado automaticamente, **envie um tíquete de serviço** para solicitar a permissão para criar e excluir APIs do EIP v3.
- Você pode configurar um EIP para um pod somente em clusters do CCE Turbo das seguintes versões:
  - v1.19: v1.19.16-r20 ou mais recente
  - v1.21: v1.21.10-r0 ou mais recente
  - v1.23: v1.23.8-r0 ou mais recente
  - v1.25: v1.25.3-r0 ou mais recente
  - v1.25 ou mais recente
- Para acessar um pod vinculado a um EIP da Internet, adicione regras de grupo de segurança para permitir o tráfego de solicitação de destino.
- Apenas um EIP pode ser vinculado a um pod.
- Configure a anotação relacionada ao EIP ao criar um pod. Depois que o pod é criado, as anotações relacionadas ao EIP não podem ser modificadas.
- Não execute operações no EIP associado a um pod por meio do console ou da API do EIP. Caso contrário, o EIP pode funcionar mal. As operações incluem alterar o nome do EIP, excluir, desvincular ou vincular o EIP e alterar o modo de cobrança do EIP.
- Depois que um EIP alocado automaticamente é excluído manualmente, a rede funciona mal. Neste caso, reconstrua o pod.

Alocar um EIP com um Pod

Ao criar um pod, defina a anotação **pod-with-eip** como **true**. Um EIP será automaticamente alocado e vinculado ao pod.

O seguinte usa uma implantação chamada **nginx** como um exemplo. Para obter detalhes sobre anotações, consulte **Tabela 7-55**.

- Para um EIP alocado automaticamente com uma **largura de banda dedicada** ao criar uma Implementação, não é necessário especificar o ID da largura de banda. O seguinte mostra um exemplo:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 3
  selector:
```



```
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
      annotations:
        yangtse.io/pod-with-eip: "true"  # An EIP will be automatically
allocated when the pod is created.
        yangtse.io/eip-bandwidth-size: "5"  # EIP bandwidth
        yangtse.io/eip-network-type: 5_bgp  # EIP type
        yangtse.io/eip-charge-mode: bandwidth  # EIP billing mode
        yangtse.io/eip-bandwidth-name: <eip_bandwidth_name>  # EIP bandwidth
name
    spec:
      containers:
        - name: container-0
          image: nginx:alpine
          resources:
            limits:
              cpu: 100m
              memory: 200Mi
            requests:
              cpu: 100m
              memory: 200Mi
      imagePullSecrets:
        - name: default-secret
```

- Para um EIP alocado automaticamente com uma **largura de banda compartilhada** quando você cria uma Implementação, é necessário especificar o ID da largura de banda. O seguinte mostra um exemplo:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
      annotations:
        yangtse.io/pod-with-eip: "true"  # An EIP will be automatically
allocated when the pod is created.
        yangtse.io/eip-network-type: 5_bgp  # EIP type
        yangtse.io/eip-bandwidth-id: <eip_bandwidth_id>  # Shared bandwidth
ID of the EIP
    spec:
      containers:
        - name: container-0
          image: nginx:alpine
          resources:
            limits:
              cpu: 100m
              memory: 200Mi
            requests:
              cpu: 100m
              memory: 200Mi
      imagePullSecrets:
        - name: default-secret
```


Tabela 7-55 Anotações para alocação de um EIP com um pod

Anotação	Obrigatório	Valor padrão	Descrição	Value Range
yangtse.io/pod-with-eip	Sim	false	Se alocar um EIP com um pod e vincular o EIP ao pod	**false** ou **true**
yangtse.io/eip-bandwidth-size	Não	5	Largura de banda, em Mbit/s	1 Mbit/s a 2000 Mbit/s por padrão O intervalo específico varia de acordo com as regiões. Para obter detalhes, consulte o console do EIP.
yangtse.io/eip-network-type	Não	5_bgp	Tipo de EIP	● 5_bgp ● 5_union ● 5_sbgp O tipo específico varia de acordo com as regiões. Para obter detalhes, consulte o console do EIP.
yangtse.io/eip-charge-mode	Obrigatório quando é utilizada uma largura de banda dedicada	Nenhum	Faturado por tráfego ou largura de banda Se uma largura de banda compartilhada for usada, esse parâmetro poderá ser deixado em branco.	● **bandwidth**: cobrado pela largura de banda ● **traffic**: cobrado pelo tráfego
yangtse.io/eip-bandwidth-id	Obrigatório quando uma largura de banda compartilhada é usada	Nenhum	ID de uma largura de banda existente Se você especificar esse parâmetro, só poderá configurar um tipo do EIP, que é opcional. Se uma largura de banda dedicada for usada, esse parâmetro poderá ser deixado em branco.	Nenhuma

Anotação	Obrigatório	Valor padrão	Descrição	Value Range
yangtse.io/eip-bandwidth-name	Não	Nome do pod	Nome da largura de banda	• Insira de 1 a 64 caracteres. Apenas letras, dígitos, hifens (-), sublinhados (_) e pontos (.) são permitidos. • Comprimento mínimo: 1 caractere • Comprimento máximo: 64 caracteres

Verificar se o EIP vinculado ao pod está disponível

Depois que um EIP é alocado a um pod, o controlador de rede de contêiner vincula o EIP ao pod e grava o resultado da alocação de volta na anotação **yangtse.io/allocated-ipv4-eip** do pod. O tempo de inicialização dos contêineres de serviço do pod pode ser anterior ao momento em que o resultado de alocação do EIP é gravado de volta.

Você pode configurar um contêiner init para o pod, associar a anotação **yangtse.io/allocated-ipv4-eip** ao contêiner init por meio de um volume downwardAPI e verificar se o EIP foi alocado no contêiner init. Você pode configurar o contêiner init da seguinte maneira:

```
apiVersion: v1
kind: Pod
metadata:
  name: example
  annotations:
    yangtse.io/pod-with-eip: "true"
    yangtse.io/eip-bandwidth-size: "5"
    yangtse.io/eip-network-type: 5_bgp
    yangtse.io/eip-charge-mode: bandwidth
    yangtse.io/eip-bandwidth-name: "xxx"
spec:
  initContainers:
  - name: init
    image: busybox:latest
    command: ['timeout', '60', 'sh', '-c', "until grep -E '[0-9]+' /etc/eipinfo/
allocated-ipv4-eip; do echo waiting for allocated-ipv4-eip; sleep 2; done"]
    volumeMounts:
        - name: eipinfo
          mountPath: /etc/eipinfo
  volumes:
    - name: eipinfo
      downwardAPI:
        items:
          - path: "allocated-ipv4-eip"
            fieldRef:
              fieldPath: metadata.annotations['yangtse.io/allocated-ipv4-eip']
...
```


Excluir um EIP com um pod

Quando você excluir um pod, o EIP alocado automaticamente para o pod também será excluído.

7.6.4.5 Configuração de um EIP estático para um pod

Cenários

No Cloud Native Network 2.0, é possível atribuir endereços IP públicos (EIPs) estáticos a pods ou StatefulSets criados diretamente.

Restrições

- Para usar um EIP estático ou um EIP alocado automaticamente, **envie um tíquete de serviço** para solicitar a permissão para criar e excluir APIs do EIP v3.
- Você pode configurar um EIP estático para um pod somente em clusters do CCE Turbo das seguintes versões:
  - v1.19: v1.19.16-r20 ou mais recente
  - v1.21: v1.21.10-r0 ou mais recente
  - v1.23: v1.23.8-r0 ou mais recente
  - v1.25: v1.25.3-r0 ou mais recente
  - v1.25 ou mais recente
- A função de EIP estática deve ser ativada juntamente com a função de alocar automaticamente um EIP para um pod. Para mais detalhes, consulte **Configuração de um EIP para um pod**.
- Somente pods de StatefulSet ou pods criados diretamente podem ser vinculados a EIPs estáticos. Implementações, DaemonSets e outros tipos de cargas de trabalho não podem ser vinculados a EIPs estáticos.
- Depois que um EIP estático é alocado, os atributos do EIP não podem ser modificados pelo pod dentro do ciclo de vida do EIP (antes que o EIP expire ou ainda esteja sendo usado pelo pod).
- Não configure um EIP estático para serviços que não têm requisitos específicos em EIPs de pod. Caso contrário, a reconstrução do pod leva mais tempo.

Configuração de um EIP estático para um pod

Ao criar um pod para ser vinculado a um endereço IP estático, configure a anotação de EIP. Em seguida, um EIP será automaticamente alocado e vinculado ao pod.

O seguinte usa um StatefulSet chamado **nginx** como um exemplo. Para obter detalhes sobre anotações, consulte **Tabela 7-56**.

- Para um EIP alocado automaticamente com uma **largura de banda dedicada** ao criar um StatefulSet, não é necessário especificar o ID da largura de banda. O seguinte mostra um exemplo:

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: nginx
spec:
  serviceName: nginx
```



```
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
      annotations:
        yangtse.io/static-eip: 'true'   # Static EIP bound to the pod
        yangtse.io/static-eip-expire-no-cascading: 'false'   # Deleting the
EIP with the associated workload
        yangtse.io/static-eip-expire-duration: 5m   # Interval for reclaiming
expired EIPs
        yangtse.io/pod-with-eip: 'true'   # An EIP will be automatically
allocated when the pod is created.
        yangtse.io/eip-bandwidth-size: '5'    # EIP bandwidth
        yangtse.io/eip-network-type: 5_bgp    # EIP type
        yangtse.io/eip-charge-mode: bandwidth    # EIP billing mode
        yangtse.io/eip-ip-version: '4'     # EIP version, for example, IPv4
    spec:
      containers:
        - name: container-0
          image: nginx:alpine
          resources:
            limits:
              cpu: 100m
              memory: 200Mi
            requests:
              cpu: 100m
              memory: 200Mi
      imagePullSecrets:
        - name: default-secret
```

- Para um EIP alocado automaticamente com uma **largura de banda compartilhada** ao criar um StatefulSet, é necessário especificar o ID da largura de banda. O seguinte mostra um exemplo:

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: nginx
spec:
  serviceName: nginx
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
      annotations:
        yangtse.io/static-eip: 'true'   # Static EIP bound to the pod
        yangtse.io/pod-with-eip: 'true'   # An EIP will be automatically
allocated when the pod is created.
        yangtse.io/eip-network-type: 5_bgp    # EIP type
        yangtse.io/eip-bandwidth-id: <eip_bandwidth_id>    # Shared bandwidth
ID of the EIP
    spec:
      containers:
        - name: container-0
          image: nginx:alpine
          resources:
            limits:
              cpu: 100m
              memory: 200Mi
            requests:
              cpu: 100m
              memory: 200Mi
```



```
      imagePullSecrets:
        - name: default-secret
```

Tabela 7-56 Anotações do EIP estático do pod

Anotação	Obrigatório	Valor padrão	Descrição	Intervalo de valor
yangtse.io/static-eip	Sim	false	Especifica se o EIP estático de um pod deve ser ativado. Esta função é suportada apenas para pods de StatefulSet ou pods sem **ownerReferences**. Esta função está desativada por padrão.	**false** ou **true**
yangtse.io/static-eip-expire-duration	Não	5m	Especifica o intervalo para recuperar o EIP estático expirado após a exclusão do pod com um EIP estático.	O formato de tempo é Go time type, por exemplo, 1h30m e 5m. Para obter detalhes, consulte **Go time type**.
yangtse.io/static-eip-expire-no-cascading	Não	false	Especifica se deve desativar a recuperação em cascata de cargas de trabalho do StatefulSet. O valor padrão é **false**, indicando que o EIP estático correspondente será excluído com a carga de trabalho de StatefulSet. Se quiser manter o EIP estático de um novo StatefulSet com o mesmo nome durante o intervalo para recuperar o EIP expirado, defina o valor como **true**.	**false** ou **true**

Tabela 7-57 Anotações para alocação de um EIP com um pod

Anotação	Obrigatório	Valor padrão	Descrição	Value Range
yangtse.io/pod-with-eip	Sim	false	Se alocar um EIP com um pod e vincular o EIP ao pod	**false** ou **true**

Anotação	Obrigatório	Valor padrão	Descrição	Value Range
yangtse.io/eip-bandwidth-size	Não	5	Largura de banda, em Mbit/s	1 Mbit/s a 2000 Mbit/s por padrão O intervalo específico varia de acordo com as regiões. Para obter detalhes, consulte o console do EIP.
yangtse.io/eip-network-type	Não	5_bgp	Tipo de EIP	● 5_bgp ● 5_union ● 5_sbgp O tipo específico varia de acordo com as regiões. Para obter detalhes, consulte o console do EIP.
yangtse.io/eip-charge-mode	Obrigatório quando é utilizada uma largura de banda dedicada	Nenhum	Faturado por tráfego ou largura de banda Se uma largura de banda compartilhada for usada, esse parâmetro poderá ser deixado em branco.	● **bandwidth**: cobrado pela largura de banda ● **traffic**: cobrado pelo tráfego
yangtse.io/eip-bandwidth-id	Obrigatório quando uma largura de banda compartilhada é usada	Nenhum	ID de uma largura de banda existente Se você especificar esse parâmetro, só poderá configurar um tipo do EIP, que é opcional. Se uma largura de banda dedicada for usada, esse parâmetro poderá ser deixado em branco.	Nenhuma

Anotação	Obrigatório	Valor padrão	Descrição	Value Range
yangtse.io/eip-bandwidth-name	Não	Nome do pod	Nome da largura de banda	• Insira de 1 a 64 caracteres. Apenas letras, dígitos, hifens (-), sublinhados (_) e pontos (.) são permitidos. • Comprimento mínimo: 1 caractere • Comprimento máximo: 64 caracteres

Excluir um EIP estático

Depois que um pod for excluído, se outro pod com o mesmo nome for criado antes que o EIP estático expire, o EIP ainda poderá ser usado. O EIP estático é excluído somente se não houver um novo pod com o nome do pod excluído antes que o EIP expire ou se a função de excluir o EIP com o StatefulSet associado estiver ativada e o StatefulSet for excluído.

7.7 Configurações da rede do cluster

7.7.1 Adição de um bloco CIDR secundário de VPC a um cluster

Cenário

Ao criar um cluster, implemente-o em uma VPC. Se a VPC planejada for muito pequena e os endereços IP forem insuficientes, você poderá usar um bloco CIDR secundário da VPC para dar suporte ao escalonamento de serviços. Esta seção descreve como adicionar um bloco CIDR secundário da VPC ao cluster.

Restrições

Somente os clusters do CCE e os clusters do CCE Turbo de v1.21 e posteriores são suportados.

Planejar um bloco CIDR secundário

Antes de adicionar um bloco CIDR secundário, planeje-o adequadamente para evitar conflitos de CIDR. Observe os seguintes pontos:

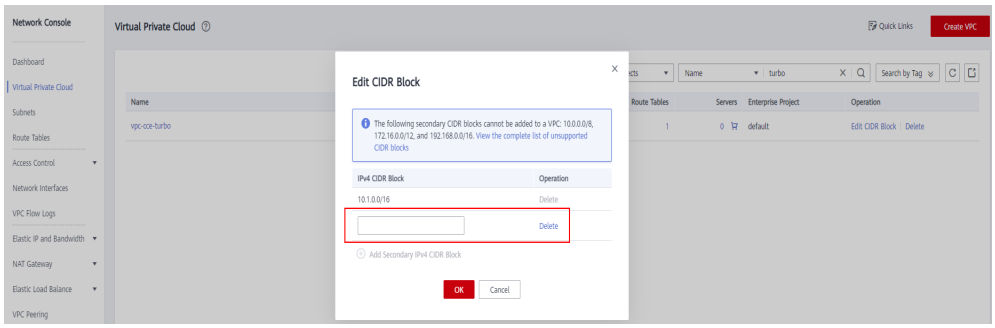
1. Todas as sub-redes (incluindo sub-redes estendidas) na VPC em que o cluster reside não podem entrar em conflito com o contêiner e os blocos CIDR de serviço.

2. Os blocos CIDR 10.0.0.0/8, 172.16.0.0/12 e 192.168.0.0/16 podem entrar em conflito com os endereços IP atribuídos aos nós principais do cluster. Não os selecione como blocos CIDR secundários.
3. Se um ECS que não estiver em um cluster na mesma VPC precisar acessar o cluster, a Secure Network Address Translation (SNAT) será executada. O endereço de origem do pod é o endereço IP do nó em vez do endereço IP do pod.
4. Os ECSs em um bloco CIDR secundário não podem acessar pods no cluster, a menos que esse bloco CIDR tenha sido usado para adicionar nós no cluster.

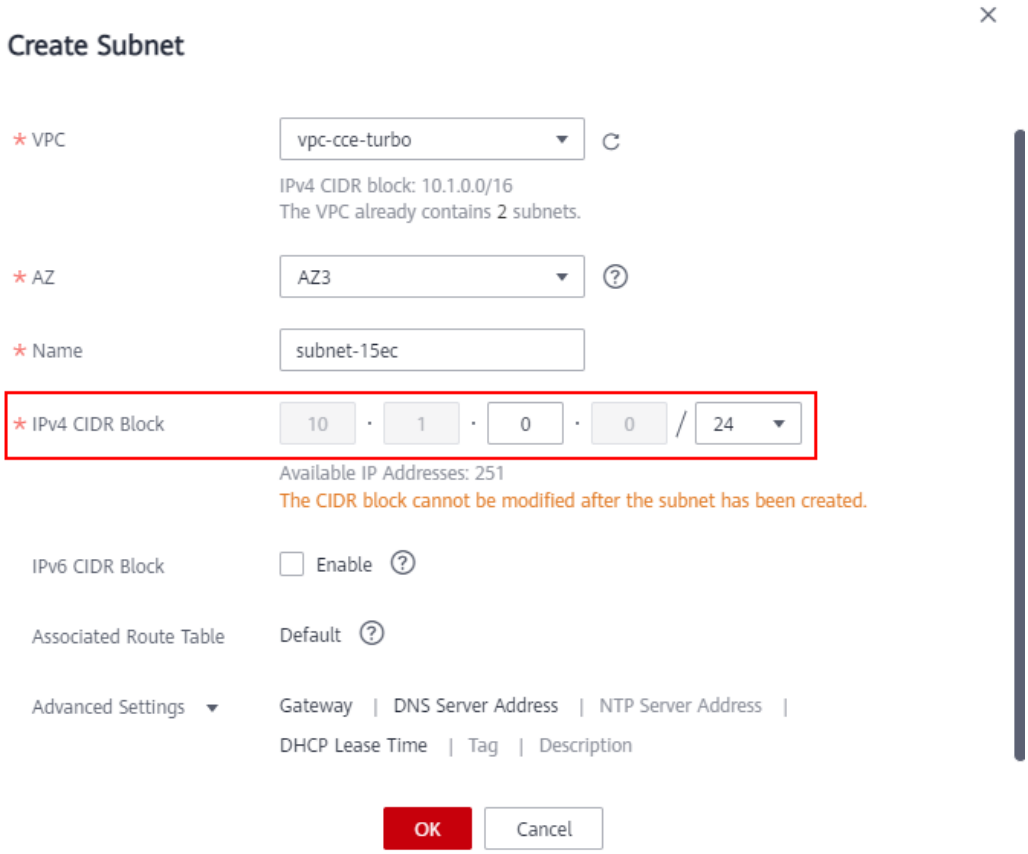
Procedimento

Passo 1 Efetue logon no console da VPC. No painel de navegação, escolha **Virtual Private Cloud > My VPCs**. Na coluna **Operation** da VPC à qual o cluster pertence, clique em **Edit CIDR Block** e em **Add Secondary IPv4 CIDR Block**.

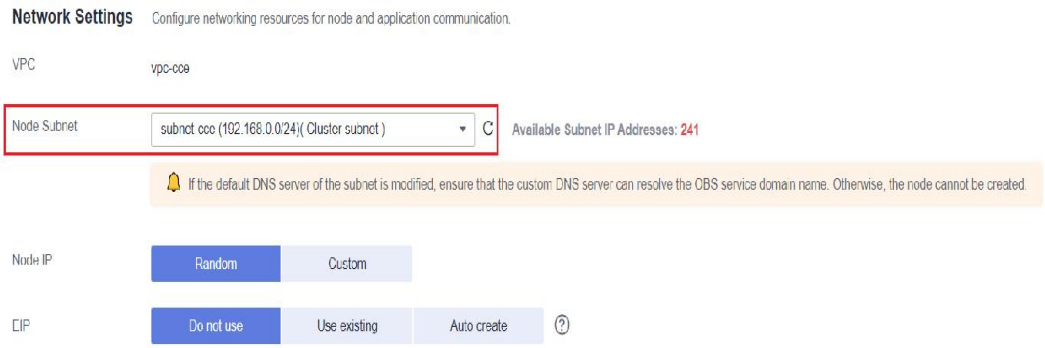
Figura 7-41 Adicionar um bloco CIDR IPv4 secundário



Passo 2 No painel de navegação, escolha **Virtual Private Cloud** > **Subnets**. Clique em **Create Subnet**. Em **IPv4 CIDR Block**, insira o bloco CIDR IPv4 secundário recém-adicionado. Configure outros parâmetros conforme planejado e clique em **OK**. Em seguida, você pode criar sub-redes no bloco CIDR IPv4 secundário para o cluster.



Passo 3 Depois que uma sub-rede é criada usando o bloco CIDR IPv4 secundário, você pode selecionar a sub-rede ao criar um nó ou pool de nós na página **Network Settings**.



----Fim

7.7.2 Alternação de uma sub-rede de nó

Cenário

Esta seção descreve como alternar sub-redes para nós em um cluster.

Restrições

- Somente sub-redes na mesma VPC que o cluster podem ser alternadas. O grupo de segurança do nó não pode ser alternado.

- Ao alternar a sub-rede de um nó, cumpra as restrições em **Alternação de uma VPC**.

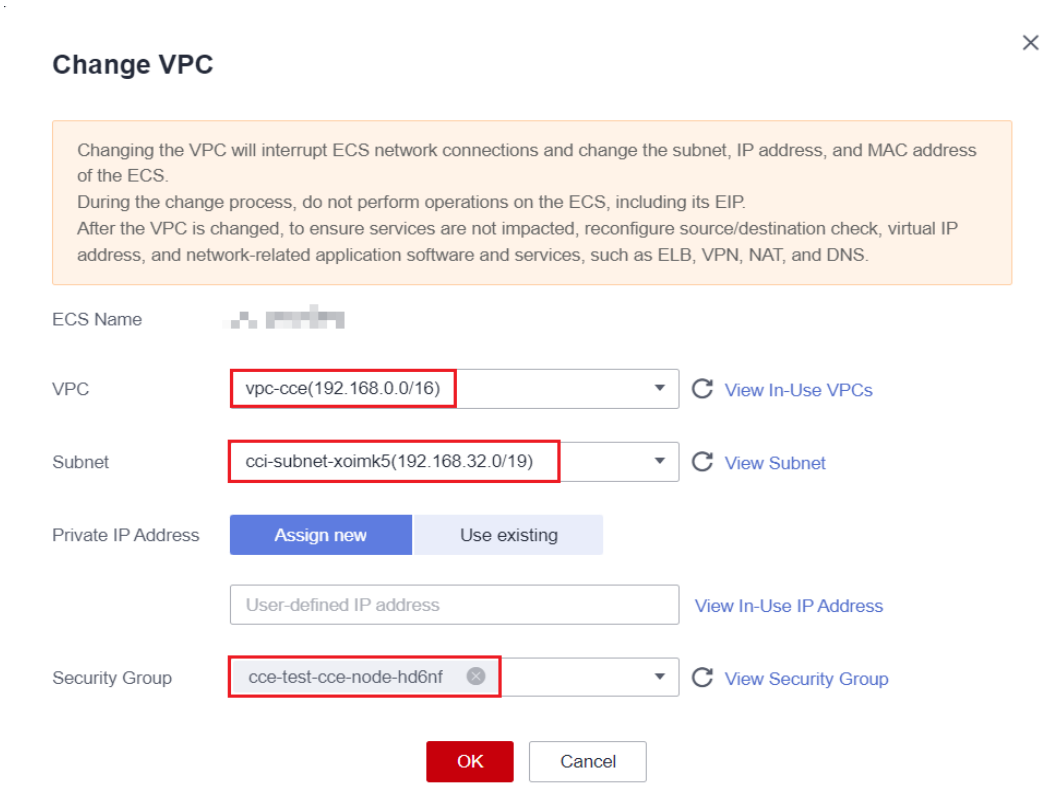
Procedimento

Passo 1 Efetue logon no console do ECS.

Passo 2 Clique em **More > Manage Network > Change VPC** na coluna **Operation** do ECS de destino.

Passo 3 Defina parâmetros para alterar a VPC.

- **VPC**: selecione a mesma VPC do cluster.
- **Subnet**: selecione a sub-rede de destino a ser alternada.
- **Private IP Address**: selecione **Assign new** ou **Use existing** conforme necessário.
- **Security Group**: selecione o grupo de segurança do nó do cluster. Caso contrário, o nó não está disponível.



Passo 4 Clique em **OK**.

Passo 5 Vá para o console do CCE e reinicie o nó. Você pode usar as configurações de parâmetro padrão. Para obter detalhes, consulte **Redefinição de um nó**.

----Fim

7.7.3 Adição de um bloco CIDR de contêiner para um cluster

Cenário

Se o bloco CIDR de contêiner (sub-rede de contêiner em um cluster do CCE Turbo) definido durante a criação do cluster do CCE for insuficiente, você poderá adicionar um bloco CIDR de contêiner para o cluster.

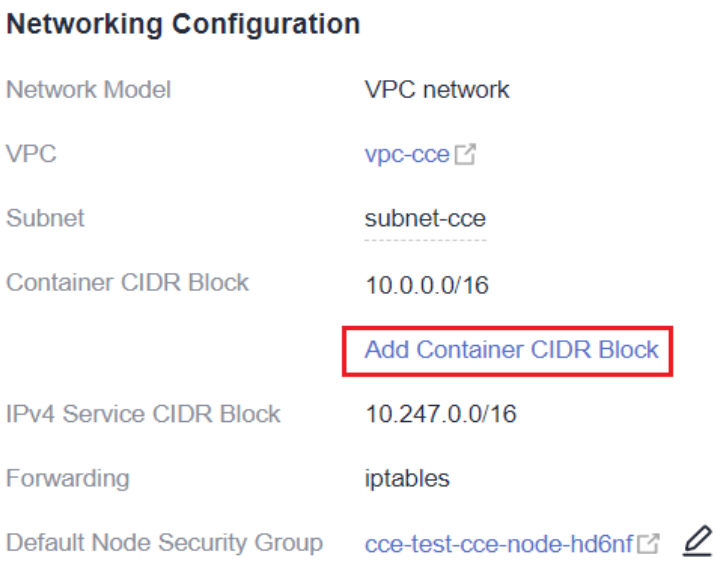
Restrições

- Essa função se aplica aos clusters do CCE e aos clusters do CCE Turbo de v1.19 ou posterior, mas não aos clusters que usam a rede de túnel de contêiner.
- O bloco CIDR do contêiner ou a sub-rede do contêiner não podem ser excluídos após serem adicionados. Tenha cuidado ao realizar esta operação.

Adicionar um bloco CIDR de contêiner para um cluster do CCE

Passo 1 Efetue logon no console do CCE e acesse o console do cluster do CCE.

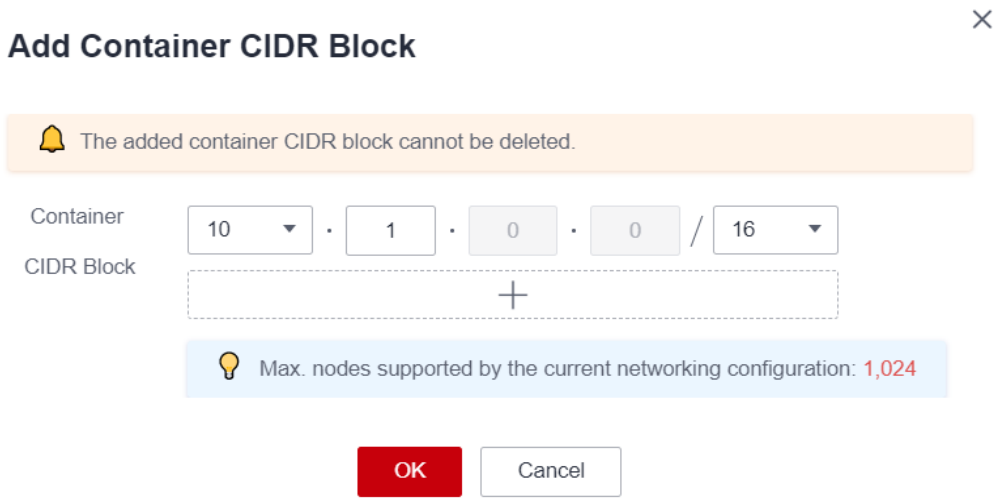
Passo 2 Na página **Cluster Information**, localize a área **Networking Configuration** e clique em **Add Pod Subnet**.



Passo 3 Configure o bloco CIDR do contêiner a ser adicionado. Você pode clicar em + para adicionar vários blocos CIDR de contêiner por vez.

NOTA

Os novos blocos CIDR de contêiner não podem entrar em conflito com blocos CIDR de serviço, blocos CIDR de VPC e blocos CIDR de contêiner existentes.



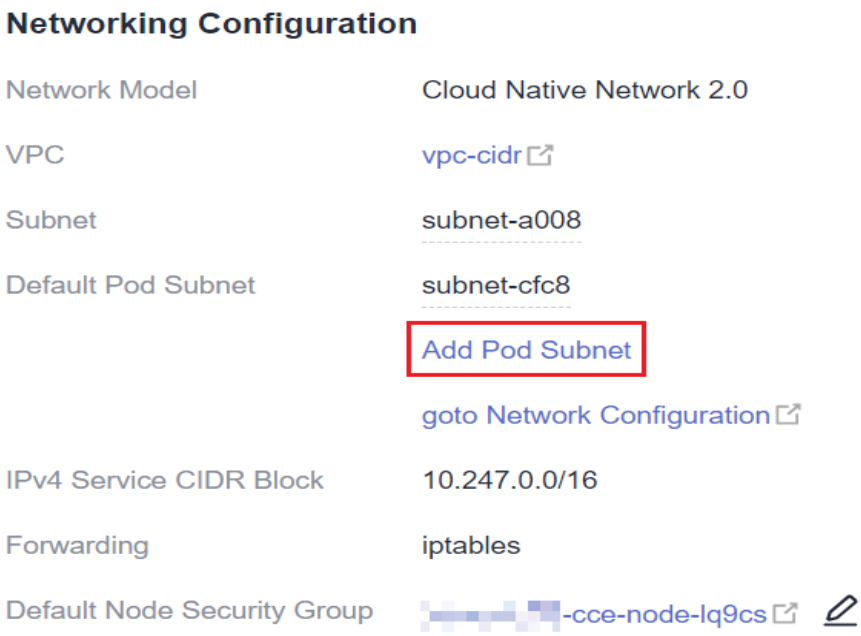
Passo 4 Clique em **OK**.

----Fim

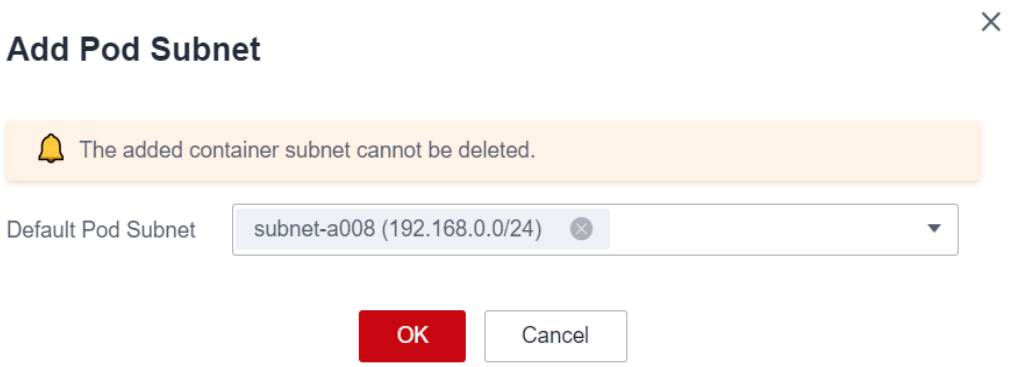
Adicionar uma sub-rede de contêiner para um cluster do CCE Turbo

Passo 1 Efetue logon no console do CCE e acesse o console do cluster do CCE Turbo.

Passo 2 Na página **Cluster Information**, localize a área **Networking Configuration** e clique em **Add Pod Subnet**.



Passo 3 Selecione uma sub-rede de contêiner na mesma VPC. Você pode adicionar várias sub-redes de contêiner por vez. Se nenhuma outra sub-rede de contêiner estiver disponível, acesse o console da VPC para criar uma.



Passo 4 Clique em **OK**.

----Fim

7.8 Configuração do acesso dentro da VPC

Esta seção descreve como acessar uma intranet de um contêiner (fora do cluster em uma VPC), incluindo acesso dentro da VPC e acesso entre VPCs.

Acesso dentro de VPC

O desempenho de acessar uma intranet a partir de um contêiner varia dependendo dos modelos de rede de contêineres de um cluster.

- **Rede de túneis de contêineres**

  A rede de túneis de contêiner encapsula pacotes de dados de rede através de túneis baseados na rede de nó. Um contêiner pode acessar outros recursos na mesma VPC, desde que o nó possa acessar os recursos. Se o acesso falhar, verifique se o grupo de segurança do recurso de par permite o acesso do nó onde o contêiner está localizado.

- **Cloud Native Network 2.0**

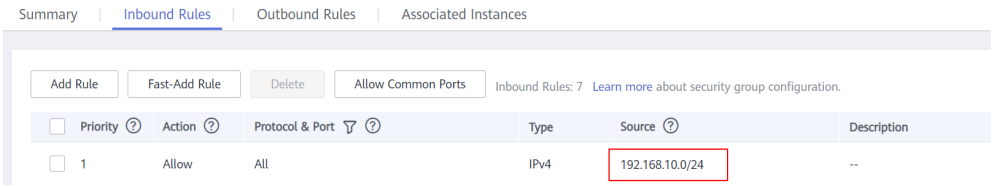
  No modelo Cloud Native Network 2.0, um contêiner recebe um endereço IP do bloco CIDR de uma VPC. O bloco CIDR do contêiner é a sub-rede da VPC onde o nó está localizado. O contêiner pode se comunicar naturalmente com outros endereços na VPC. Se o acesso falhar, verifique se o grupo de segurança de recursos peer permite o acesso do bloco CIDR do contêiner.

- **Rede da VPC**

  O modelo de rede da VPC usa rotas da VPC para encaminhar o tráfego de contêiner. O bloco CIDR do contêiner e o nó da VPC não estão no mesmo bloco CIDR. Quando um contêiner acessa outros recursos na mesma VPC, **o grupo de segurança do recurso correspondente deve permitir o acesso do bloco CIDR do contêiner**.

  Por exemplo, o bloco CIDR onde o nó do cluster reside é 192.168.10.0/24 e o bloco CIDR do contêiner é 172.16.0.0/16.

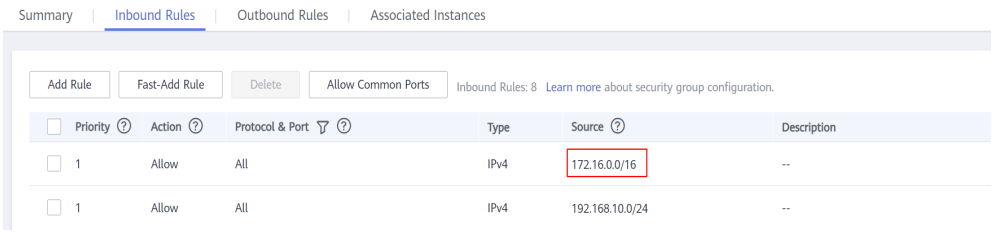
  Há um ECS cujo endereço IP é 192.168.10.52 na VPC (fora do cluster). O grupo de segurança do ECS permite o acesso apenas do bloco CIDR do nó do cluster.



Nesse caso, se você executar ping em 192.168.10.52 do contêiner, a operação de ping falhará.

```
kubectl exec test01-6cbbf97b78-krj6h  -it -- /bin/sh
/ # ping 192.168.10.25
PING 192.168.10.25 (192.168.10.25): 56 data bytes
^C
--- 192.168.10.25 ping statistics ---
104 packets transmitted, 0 packets received, 100% packet loss
```

Configure o grupo de segurança para permitir o acesso do bloco CIDR do contêiner 172.16.0.0/16.



Neste caso, 192.168.10.52 pode ser pingado a partir do contêiner.

```
$ kubectl exec test01-6cbbf97b78-krj6h  -it -- /bin/sh
/ # ping 192.168.10.25
PING 192.168.10.25 (192.168.10.25): 56 data bytes
64 bytes from 192.168.10.25: seq=0 ttl=64 time=1.412 ms
64 bytes from 192.168.10.25: seq=1 ttl=64 time=1.400 ms
64 bytes from 192.168.10.25: seq=2 ttl=64 time=1.299 ms
64 bytes from 192.168.10.25: seq=3 ttl=64 time=1.283 ms
^C
--- 192.168.10.25 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
```

Acesso entre VPCs

O acesso entre VPCs é implementado ao estabelecer uma conexão de emparelhamento entre VPCs.

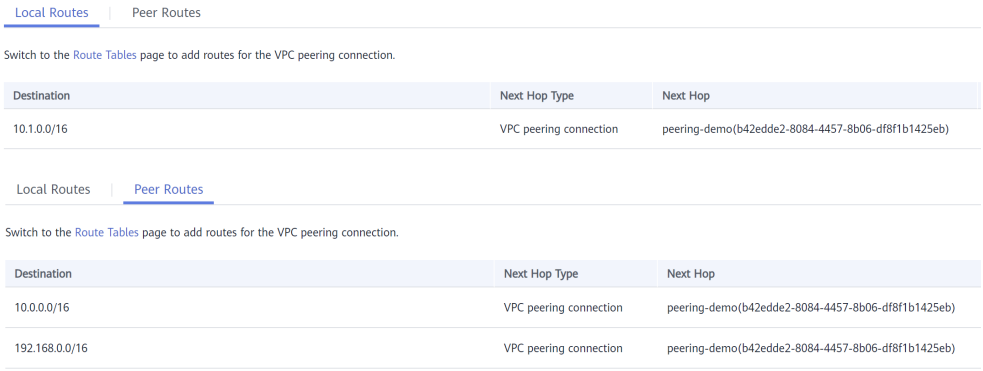
- No modelo de rede de túnel de contêiner, um contêiner pode acessar a VPC de par somente quando a comunicação é ativada entre a rede de nó e a VPC de par.
- O Cloud Native Network 2.0 é semelhante à rede de túnel de contêineres. Você só precisa habilitar a comunicação entre a sub-rede onde o contêiner está localizado e a VPC de mesmo nível.
- Cada rede da VPC tem um bloco CIDR de contêiner independente. Além do bloco CIDR da VPC, o bloco CIDR do contêiner também precisa ser conectado.

  Suponha que existam duas VPCs.

  – vpc-demo: seu bloco CIDR é 192.168.0.0/16, o cluster está em vpc-demo e o bloco CIDR do contêiner é 10.0.0.0/16.

  – vpc-demo2: seu bloco CIDR é 10.1.0.0/16.

  Crie uma conexão de emparelhamento chamada **peering-demo** (a VPC local é vpc-demo e a VPC de mesmo nível é vpc-demo2). Adicione o bloco CIDR do contêiner à rota da VPC de par.



Após essa configuração, você pode acessar o bloco CIDR de contêiner 10.0.0.0/16 em vpc-demo2. Durante o acesso, preste atenção à configuração do grupo de segurança e habilite a configuração da porta.

Acesso outros serviços de nuvem

Os serviços comuns que se comunicam com o CCE através de uma intranet incluem RDS, DCS, Kafka, RabbitMQ e ModelArts.

Além das configurações de rede descritas em **Acesso dentro de VPC** e **Acesso entre VPCs**, você também precisa verificar **se esses serviços em nuvem permitem acesso externo**. Por exemplo, a instância do DCS Redis pode ser acessada apenas pelos endereços IP em sua lista de permissões. Geralmente, esses serviços em nuvem podem ser acessados por endereços IP na mesma VPC. No entanto, o bloco CIDR do contêiner no modelo de rede da VPC é diferente do bloco CIDR da VPC. Portanto, você deve adicionar o bloco CIDR do contêiner à lista branca.

E se um contêiner não conseguir acessar uma intranet?

Se uma intranet não puder ser acessada de um contêiner, execute as seguintes operações:

1. Exiba a regra de grupo de segurança do servidor de par para verificar se o contêiner tem permissão para acessar o servidor de par.
   - O modelo de rede de túnel de contêiner precisa permitir o endereço IP do nó onde o contêiner está localizado.
   - O modelo de rede da VPC precisa permitir o bloco CIDR do contêiner.
   - O modelo Cloud Native Network 2.0 precisa permitir a sub-rede onde o contêiner está localizado.
2. Verifique se uma lista branca está configurada para o servidor de par. Por exemplo, a instância do DCS Redis pode ser acessada apenas pelos endereços IP em sua lista de permissões. Adicione os blocos CIDR do contêiner e do nó à lista branca.
3. Verifique se o mecanismo de contêiner está instalado no servidor de par e se ele entra em conflito com o bloco CIDR do contêiner no CCE. Se ocorrer um conflito de rede, o acesso falhará.

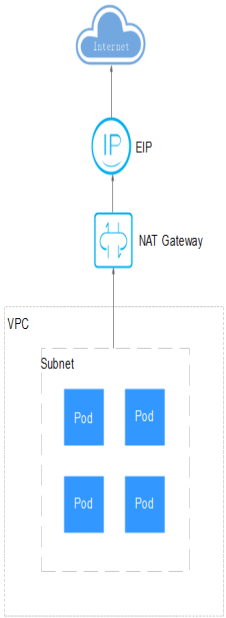
7.9 Acesso a redes públicas a partir de um contêiner

Os contêineres podem acessar redes públicas de uma das seguintes maneiras:

- Vincular um EIP ao nó onde o contêiner está localizado se o modelo de rede for VPC ou túnel.
- Vincular um EIP ao pod. (Essa função se aplica somente aos clusters de Cloud Native 2.0. Para fazer isso, vincule manualmente um EIP à ENI ou sub-ENI do pod no console da VPC. Esse método não é recomendado porque o endereço IP de um pod muda depois que o pod é reprogramado. Como resultado, o novo pod não pode acessar a rede pública.)
- Configurar regras SNAT através do NAT Gateway.

Você pode usar o NAT Gateway para habilitar pods de contêiner em uma VPC para acessar redes públicas. O NAT Gateway fornece tradução de endereço de rede de origem (SNAT), que converte endereços IP privados em um endereço IP público, vinculando um endereço IP elástico (EIP) ao gateway, fornecendo acesso seguro e eficiente à Internet. **Figura 7-42** mostra a arquitetura da SNAT. A função SNAT permite que os pods de contêiner em uma VPC acessem a Internet sem estarem vinculados a um EIP. A SNAT suporta um grande número de conexões simultâneas, o que o torna adequado para aplicações que envolvam um grande número de solicitações e conexões.

Figura 7-42 SNAT



Para habilitar um pod de contêiner para acessar a Internet, execute as seguintes etapas:

Passo 1 Atribua um EIP.

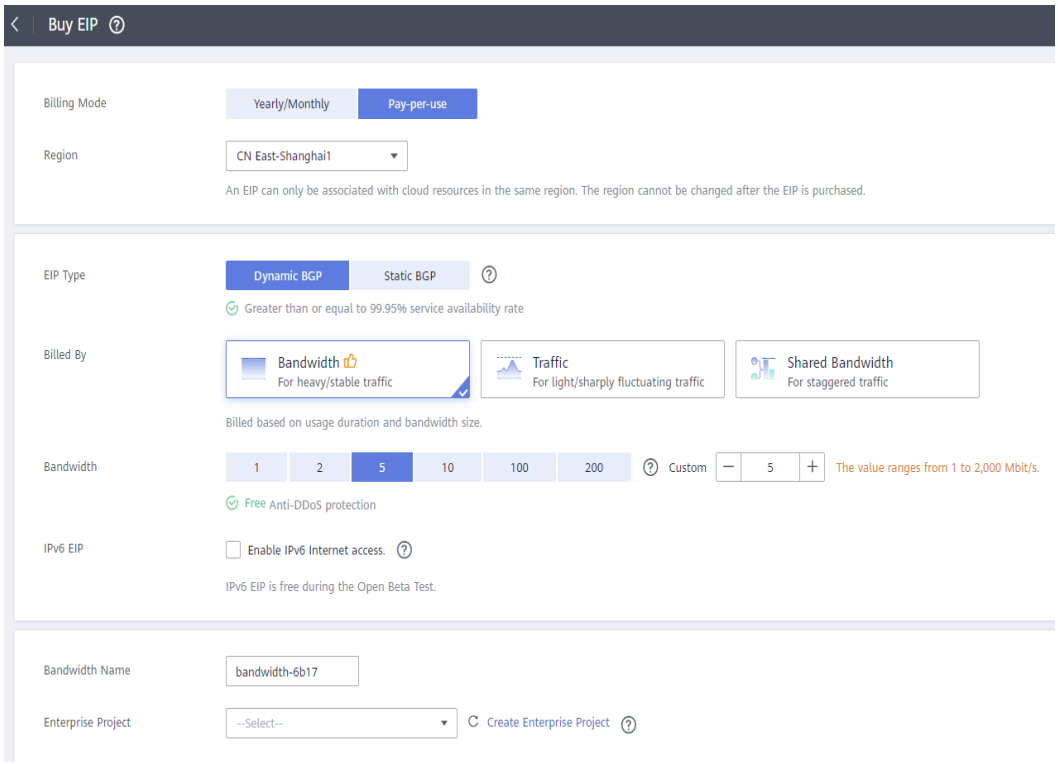
1. Faça logon no console de gerenciamento.
2. Clique em no canto superior esquerdo do console de gerenciamento e selecione uma região e um projeto.
3. Clique em no canto superior esquerdo e escolha **Networking** > **Elastic IP** na lista expandida.

4. Na página **EIPs**, clique em **Buy EIP**.
5. Configure os parâmetros conforme necessário.

   NOTA

   Defina **Region** como a região onde os pods de contêiner estão localizados.

Figura 7-43 Comprar um endereço IP elástico



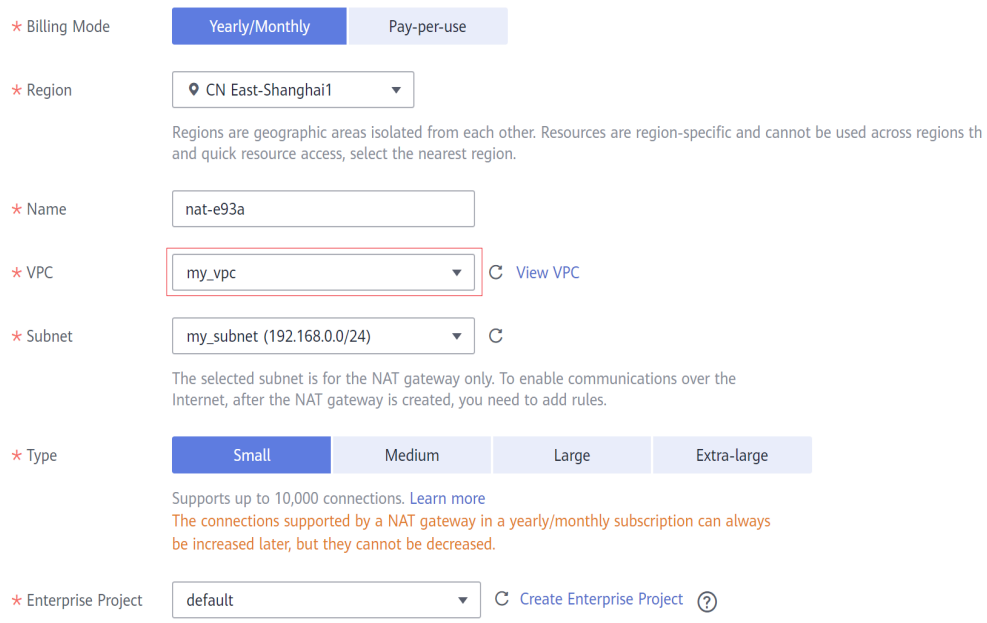
Passo 2 Crie um gateway NAT. Para obter detalhes, consulte **Compra de um gateway NAT**.

1. Faça logon no console de gerenciamento.
2. Clique em no canto superior esquerdo do console de gerenciamento e selecione uma região e um projeto.
3. Clique em no canto superior esquerdo e escolha **Networking** > **NAT Gateway** na lista expandida.
4. Na página exibida, clique em **Buy Public NAT Gateway** no canto superior direito.
5. Configure os parâmetros conforme necessário.

   NOTA

   Selecione a mesma VPC.

Figura 7-44 Comprar um gateway NAT



Passo 3 Configure uma regra SNAT e vincule o EIP à sub-rede. Para obter detalhes, consulte **Adição de uma regra SNAT**.

1. Faça logon no console de gerenciamento.
2. Clique em no canto superior esquerdo do console de gerenciamento e selecione uma região e um projeto.
3. Clique em no canto superior esquerdo e escolha **Networking** > **NAT Gateway** na lista expandida.
4. Na página exibida, clique no nome do gateway NAT para o qual você deseja adicionar a regra SNAT.
5. Na página de guia **SNAT Rules**, clique em **Add SNAT Rule**.
6. Defina os parâmetros conforme necessário.

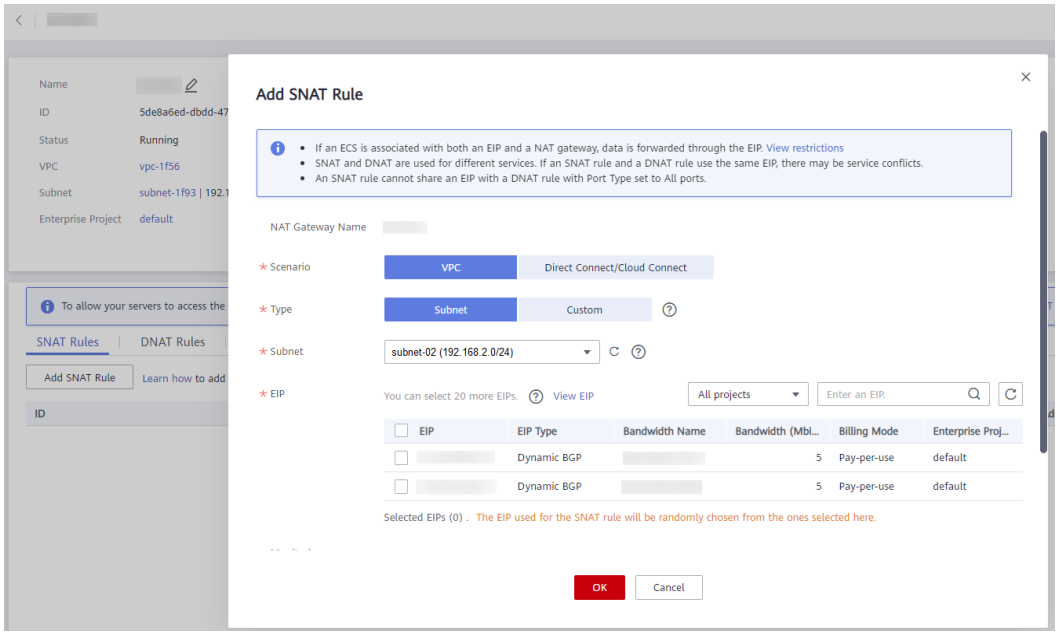
NOTA

As regras SNAT entram em vigor pelo bloco CIDR. Como diferentes modelos de rede de contêineres usam diferentes modos de comunicação, a sub-rede precisa ser selecionada de acordo com as seguintes regras:

- Rede de túneis e rede da VPC: selecione a sub-rede em que o nó está localizado, ou seja, a sub-rede selecionada durante a criação do nó.

Se houver vários blocos CIDR, você poderá criar várias regras SNAT ou personalizar um bloco CIDR, desde que o bloco CIDR contenha a sub-rede do nó.

Figura 7-45 Adição de uma regra SNAT



Depois que a regra SNAT é configurada, as cargas de trabalho podem acessar redes públicas do contêiner. Redes públicas podem ser pingadas a partir do contêiner.

----Fim

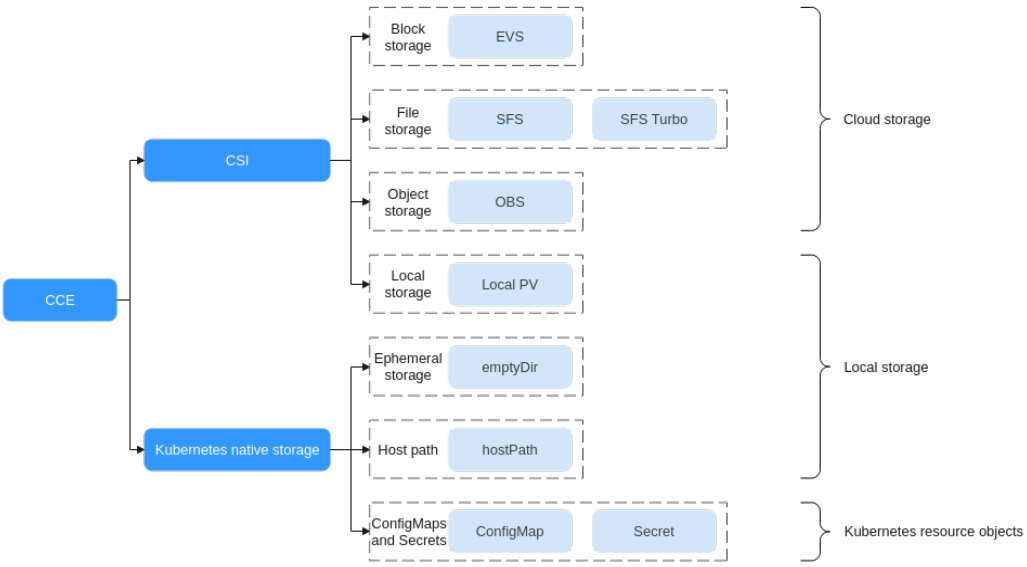
8 Armazenamento

8.1 Visão geral

Armazenamento do contêiner

O armazenamento de contêiner do CCE é implementado com base nas APIs de armazenamento de contêiner do Kubernetes (**CSI**). O CCE integra vários tipos de armazenamento em nuvem e abrange diferentes cenários de aplicações. O CCE é totalmente compatível com os serviços de armazenamento nativos do Kubernetes, como ConfigMap, hostPath, secret.

Figura 8-1 Tipo de armazenamento de contêiner



O CCE permite que você monte volumes de armazenamento em nuvem em seus pods. Suas características são descritas abaixo.

Tabela 8-1 Comparação de armazenamento em nuvem

Dimensão	EVS	SFS	SFS Turbo	OBS
Definição	O EVS oferece armazenamento em bloco escalável para servidores em nuvem. Com alta confiabilidade, alto desempenho e especificações avançadas, os discos EVS podem ser usados para sistemas de arquivos distribuídos, ambientes de desenvolvimento/teste, data warehouses e aplicações de computação de alto desempenho (HPC).	Expansível para petabytes, o SFS fornece armazenamento de arquivos compartilhados totalmente hospedado, altamente disponível e estável para lidar com aplicações com uso intensivo de dados e largura de banda em HPC, processamento de mídia, compartilhamento de arquivos, gerenciamento de conteúdo e serviços da Web.	Expansível até 320 TB, o SFS Turbo fornece um armazenamento de arquivos compartilhados totalmente hospedado, que é altamente disponível e estável, para suportar pequenos arquivos e aplicações que exigem baixa latência e alto IOPS. Você pode usar o SFS Turbo em sites de alto tráfego, armazenamento de logs, compactação/descompactação, DevOps, OA corporativo e aplicações em contêiner.	Object Storage Service (OBS) fornece armazenamento de dados maciço, seguro e econômico para que você armazene dados de qualquer tipo e tamanho. Você pode usá-lo em backup/arquivamento corporativo, vídeo sob demanda (VoD), vigilância por vídeo e muitos outros cenários.
Lógica de armazenamento de dados	Armazena dados binários e não pode armazenar arquivos diretamente. Para armazenar arquivos, formate o sistema de arquivos primeiro.	Armazena arquivos e classifica e exibe dados na hierarquia de arquivos e pastas.	Armazena arquivos e classifica e exibe dados na hierarquia de arquivos e pastas.	Armazena objetos. Os arquivos armazenados diretamente geram automaticamente os metadados do sistema, que também podem ser personalizados pelos usuários.

Dimensão	EVS	SFS	SFS Turbo	OBS
Modo de acesso	Acessível somente após ser montado em ECSs ou BMSs e inicializado.	Montado em ECSs ou BMSs usando protocolos de rede. Um endereço de rede deve ser especificado ou mapeado para um diretório local para acesso.	Suporta o protocolo NFS (Network File System) (apenas para NFSv3). Você pode integrar perfeitamente aplicações e ferramentas existentes com o SFS Turbo.	Acessível através da Internet ou Direct Connect (DC). Especifique o endereço do bucket e use protocolos de transmissão, como HTTP ou HTTPS.
Provisionamento estático	Compatível. Para mais detalhes, consulte **Uso de um disco EVS existente através de um PV estático**.	Compatível. Para mais detalhes, consulte **Uso de um sistema de arquivos do SFS existente por meio de um PV estático**.	Compatível. Para mais detalhes, consulte **Uso de um sistema de arquivos do SFS Turbo existente por meio de um PV estático**.	Compatível. Para mais detalhes, consulte **Uso de um bucket do OBS existente através de um PV estático**.
Provisionamento dinâmico	Compatível. Para mais detalhes, consulte **Uso de um disco EVS por meio de um PV dinâmico**.	Compatível. Para mais detalhes, consulte **Uso de um sistema de arquivos do SFS através de um PV dinâmico**.	Não compatível	Compatível. Para mais detalhes, consulte **Uso de um bucket do OBS através de um PV dinâmico**.
Recursos	Armazenamento não compartilhado. Cada volume pode ser montado em apenas um nó.	Armazenamento compartilhado com alto desempenho e taxa de transferência	Armazenamento compartilhado com alto desempenho e largura de banda	Sistema de arquivos compartilhado, em modo de usuário

Dimensão	EVS	SFS	SFS Turbo	OBS
Uso	HPC, aplicações de cluster de núcleo empresarial, sistemas de aplicações empresariais e desenvolvimento/teste **NOTA** As aplicações de HPC aqui exigem armazenamento de alta velocidade e alto IOPS, como design industrial e exploração de energia.	HPC, processamento de mídia, gerenciamento de conteúdo, serviços da Web, Big Data e aplicações de análise **NOTA** As aplicações de HPC aqui exigem alta largura de banda e armazenamento de arquivos compartilhados, como sequenciamento de genes e renderização de imagens.	Sites de alto tráfego, armazenamento de logs, DevOps e OA corporativo	Análise de Big Data, hospedagem estática de sites, vídeo on-line sob demanda (VoD), sequenciamento de genes, vigilância por vídeo inteligente, backup e arquivamento e caixas de nuvem corporativas (discos da Web)
Capacidade	TB	SFS 1.0: PB	Uso geral: TB	EB
Latência	1–2 ms	SFS 1.0: 3–20 ms	Uso geral: 1–5 ms	10 ms
IOPS/TPS	33.000 para um único disco	SFS 1.0: 2000	Uso geral: até 100.000	Dezenas de milhões
Largura de banda	MB/s	SFS 1.0: GB/s	Uso geral: até GB/s	TB/s

Suporte a projetos empresariais

NOTA

Para usar esta função, o complemento everest deve ser atualizado para v1.2.33 ou posterior.

- Criação automática de armazenamento:

  O CCE permite especificar um projeto empresarial ao criar discos EVS e PVCs do OBS. Os recursos de armazenamento criados (discos EVS e do OBS) pertencem ao projeto da empresa especificado. **O projeto corporativo pode ser o projeto empresarial ao qual o cluster pertence ou o projeto empresarial padrão.**

  Se nenhum projeto empresarial for especificado, o projeto da empresa especificado na StorageClass será usado por padrão para criar recursos de armazenamento.

  – Para uma StorageClass personalizado, você pode especificar um projeto empresarial na StorageClass. Para mais detalhes, consulte **Especificação de um projeto**

empresarial para classes de armazenamento. Se nenhum projeto empresarial for especificado na StorageClass, o projeto empresarial padrão será usado.

  - Para as classes de armazenamento csi-disk e csi-obs fornecidas pelo CCE, os recursos de armazenamento criados pertencem ao projeto empresarial padrão.

- Usar armazenamento existente:

  Ao criar uma PVC usando um PV, certifique-se de que **everest.io/enterprise-project-id** especificado na PVC e no PV seja o mesmo, pois um projeto empresarial foi especificado durante a criação do recurso de armazenamento. Caso contrário, a PVC e o PV não podem ser vinculados.

Documentação

- **Noções básicas de armazenamento**
- **Elastic Volume Service**
- **Scalable File Service**
- **SFS Turbo**
- **Object Storage Service**

8.2 Noções básicas de armazenamento

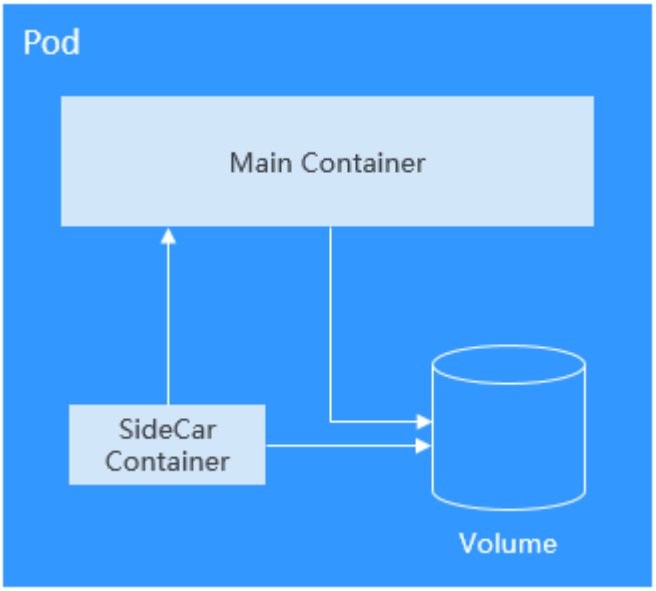
Volume

Os arquivos em disco em um contêiner são efêmeros, o que apresenta os seguintes problemas para aplicações importantes em execução no contêiner:

1. Quando um contêiner é reconstruído, os arquivos no contêiner serão perdidos.
2. Quando vários contêineres são executados em um pod ao mesmo tempo, os arquivos precisam ser compartilhados entre os contêineres.

Os volumes do Kubernetes resolvem esses dois problemas. Volumes, como parte de um pod, não podem ser criados independentemente e só podem ser definidos em pods. Todos os contêineres em um pod podem acessar seus volumes, mas os volumes devem ter sido montados em qualquer diretório em um contêiner.

A figura a seguir mostra como um volume de armazenamento é usado entre contêineres em um pod.



Os princípios básicos para o uso de volumes são os seguintes:

- Vários volumes podem ser montados em um pod. No entanto, não monte muitos volumes em um pod.
- Vários tipos de volumes podem ser montados em um pod.
- Cada volume montado em um pod pode ser compartilhado entre os contêineres no pod.
- Recomendamos que você use PVCs e PVs para montar volumes para o Kubernetes.

NOTA

O ciclo de vida de um volume é o mesmo do pod no qual o volume é montado. Quando o pod é excluído, o volume também é excluído. No entanto, os arquivos no volume podem superar o volume, dependendo do tipo de volume.

O Kubernetes fornece vários tipos de volume, que podem ser classificados como in-tree e out-of-tree.

Classificação do volume	Descrição
In-tree	Mantido por meio do repositório de código do Kubernetes e construído, editado e lançado com arquivos binários do Kubernetes. O Kubernetes não aceita mais esse tipo de volume. Volumes nativos do Kubernetes, como HostPath, EmptyDir, Secret e ConfigMap, são todos do tipo in-tree. As PVCs são um volume especial in-tree. O Kubernetes usa esse tipo de volume para converter de in-tree para out-of-tree. As PVCs permitem que você solicite PVs criados usando os recursos de armazenamento subjacentes fornecidos por diferentes fornecedores de armazenamento.

Classificação do volume	Descrição
Out-of-tree	Os volumes out-of-tree incluem interfaces de armazenamento de contêiner (CSIs) e FlexVolumes (obsoletos). Os fornecedores de armazenamento só precisam cumprir determinadas especificações para criar complementos de armazenamento personalizados e PVs que podem ser usados pelo Kubernetes, sem adicionar código-fonte do complemento ao repositório de código do Kubernetes. O armazenamento em nuvem, como SFS e OBS, é usado instalando drivers de armazenamento em um cluster. Você precisa criar PVs no cluster e montá-los em pods usando PVCs.

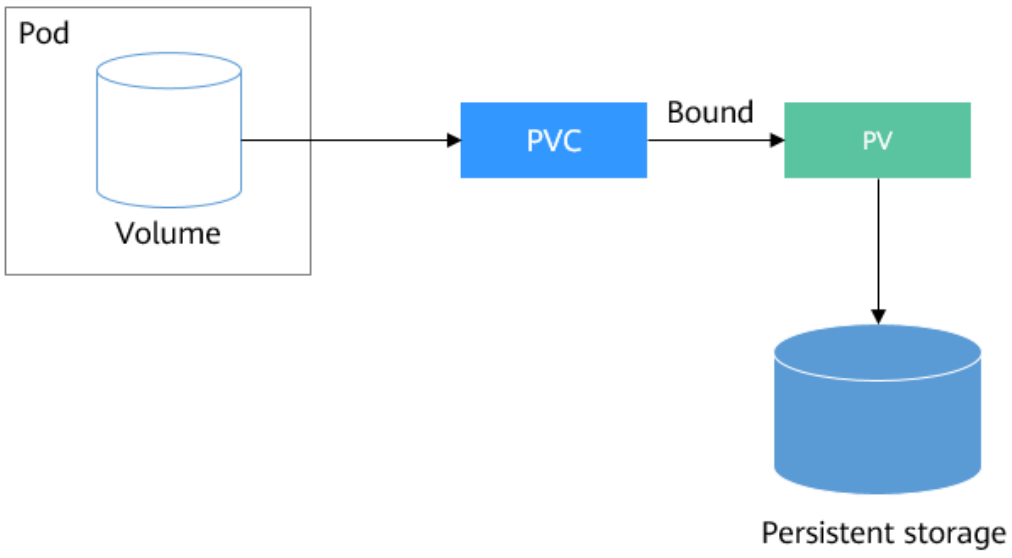
PV e PVC

O Kubernetes fornece PVs (PersistentVolumes) e PVCs (PersistentVolumeClaims) para abstrair detalhes de como o armazenamento é fornecido de como é consumido. Você pode solicitar um tamanho específico de armazenamento quando necessário, assim como os pods podem solicitar níveis específicos de recursos (CPU e memória).

- PV: descreve um volume de armazenamento persistente em um cluster. Um PV é um recurso de nível de cluster, assim como um nó. Ele se aplica a todo o cluster do Kubernetes. Um PV tem um ciclo de vida independente de qualquer Pod individual que use o PV.
- PVC: descreve uma solicitação de armazenamento por um usuário. Ao configurar o armazenamento para uma aplicação, declare uma solicitação de armazenamento (ou seja, PVC). O Kubernetes seleciona um PV que melhor atenda à solicitação e vincula o PV à PVC. Uma vinculação de PVC a PV é um mapeamento um-para-um. Ao criar uma PVC, descreva os atributos do armazenamento persistente solicitado, como o tamanho do armazenamento e a permissão de leitura/gravação.

Você pode vincular PVCs a PVs em um pod para que o pod possa usar recursos de armazenamento. A figura a seguir mostra a relação entre PVs e PVCs.

Figura 8-2 Vinculação de PVC para PV



CSI

A CSI é um padrão para interfaces de armazenamento de contêineres e uma solução de implementação de plug-in de armazenamento recomendada pela comunidade do Kubernetes. **everest** é um complemento de armazenamento desenvolvido pela Huawei baseado na CSI. Ela fornece diferentes tipos de armazenamento persistente para contêineres.

Modos de acesso a volume

Os volumes de armazenamento podem ser montados no sistema host somente no modo suportado pelos recursos de armazenamento subjacentes. Por exemplo, um sistema de armazenamento de arquivos pode ser lido e gravado por vários nós, mas um disco EVS pode ser lido e gravado por apenas um nó.

- **ReadWriteOnce**: um volume de armazenamento pode ser montado em um único nó no modo leitura-gravação.
- **ReadWriteMany**: um volume de armazenamento pode ser montado em vários nós no modo leitura-gravação.

Tabela 8-2 Modos de acesso suportados por volumes de armazenamento

Tipo de armazenamento	ReadWriteOnce	ReadWriteMany
EVS	√	×
SFS	×	√
OBS	×	√
SFS Turbo	×	√
PV local	√	×

Montagem de um volume de armazenamento

Você pode montar volumes das seguintes maneiras:

Use PVs para descrever os recursos de armazenamento existentes e, em seguida, crie PVCs para usar os recursos de armazenamento em pods. Você também pode usar o modo de criação dinâmica. Ou seja, especifique a **StorageClass** ao criar uma PVC e use o provisionador na StorageClass para criar automaticamente um PV e vincular o PV à PVC.

Tabela 8-3 Modos de montagem de volumes

Modo de montagem	Descrição	Tipo de volume suportado	Outras restrições
Criar estaticamente o volume de armazenamento (usando o armazenamento existente)	Use o armazenamento existente (como discos EVS e sistemas de arquivos do SFS) para criar PVs e montá-los na carga de trabalho por meio de PVCs. O Kubernetes vincula as PVCs aos PVs correspondentes para que as cargas de trabalho possam acessar os serviços de armazenamento.	Todos os volumes	Nenhuma
Criar dinâmica de volumes de armazenamento (criação automática de armazenamento)	Especifique uma **StorageClass** para uma PVC. O provedor de armazenamento cria mídia de armazenamento subjacente conforme necessário para criar automaticamente PVs e vincular diretamente o PV à PVC.	EVS, OBS, SFS e PV local	Nenhuma
Montagem dinâmica (VolumeClaimTemplate)	Obtida usando o campo **volumeClaimTemplates** e depende da capacidade de criação dinâmica de PV da StorageClass. Neste modo, cada pod está associado a uma única PVC e PV. Depois que um pod é reprogramado, os dados originais ainda podem ser montados nele com base no nome da PVC.	EVS e PV local	Suportado apenas pelo StatefulSets

Política de recuperação da PV

Uma política de recuperação de PV é usada para excluir ou recuperar volumes subjacentes quando uma PVC é excluída. O valor pode ser **Delete** ou **Retain**.

- **Delete**: a exclusão de uma PVC removerá o PV do Kubernetes, portanto, os ativos de armazenamento subjacentes associados da infraestrutura externa.

  NOTA

  Os recursos anuais/mensais não podem ser excluídos usando a política de recuperação **Delete**.

- **Retain**: quando uma PVC é excluída, o PV e os recursos de armazenamento subjacentes não são excluídos. Em vez disso, você deve excluir manualmente esses recursos. Depois disso, os recursos do PV estão no estado **Released** e não podem ser vinculados diretamente à PVC.

  Você pode excluir e recuperar volumes manualmente executando as seguintes operações:

  a. Exclua o PV.

b. Limpe os dados sobre os recursos de armazenamento subjacentes associados, conforme necessário.

c. Exclua os recursos de armazenamento subjacentes associados.

Para reutilizar os recursos de armazenamento subjacentes, crie um PV.

O CCE também permite excluir uma PVC sem excluir os recursos de armazenamento subjacentes. Esta função só pode ser obtida usando um arquivo YAML: Defina a política de recuperação de PV como **Delete** e adicione **everest.io/reclaim-policy: retain-volume-only** a **annotations**. Dessa forma, quando a PVC é excluída, o PV é excluído, mas os recursos de armazenamento subjacentes são retidos.

O seguinte arquivo YAML usa o EVS como exemplo:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: test
  namespace: default
  annotations:
    volume.beta.kubernetes.io/storage-provisioner: everest-csi-provisioner
    everest.io/disk-volume-type: SAS
  labels:
    failure-domain.beta.kubernetes.io/region: <your_region>   # Region of the
node where the application is to be deployed
    failure-domain.beta.kubernetes.io/zone: <your_zone>       # AZ of the node
where the application is to be deployed
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
  storageClassName: csi-disk
  volumeName: pv-evs-test

---
apiVersion: v1
kind: PersistentVolume
metadata:
  annotations:
    pv.kubernetes.io/provisioned-by: everest-csi-provisioner
    everest.io/reclaim-policy: retain-volume-only
  name: pv-evs-test
  labels:
    failure-domain.beta.kubernetes.io/region: <your_region>   # Region of the
node where the application is to be deployed
    failure-domain.beta.kubernetes.io/zone: <your_zone>       # AZ of the node
where the application is to be deployed
spec:
  accessModes:
    - ReadWriteOnce
  capacity:
    storage: 10Gi
  csi:
    driver: disk.csi.everest.io
    fsType: ext4
    volumeHandle: 2af98016-6082-4ad6-bedc-1a9c673aef20
    volumeAttributes:
      storage.kubernetes.io/csiProvisionerIdentity: everest-csi-provisioner
      everest.io/disk-mode: SCSI
      everest.io/disk-volume-type: SAS
  persistentVolumeReclaimPolicy: Delete
  storageClassName: csi-disk
```


Documentação

- Para obter mais informações sobre o armazenamento do Kubernetes, consulte **Armazenamento**.
- Para obter mais informações sobre o armazenamento de contêineres do CCE, consulte **Visão geral**.

8.3 Elastic Volume Service

8.3.1 Visão geral

Para obter armazenamento persistente, o CCE permite que você monte os volumes de armazenamento criados a partir de discos EVS (Elastic Volume Service) em um caminho de um contêiner. Quando o contêiner é migrado dentro uma AZ, os volumes do EVS montados também são migrados. Usando volumes do EVS, você pode montar o diretório de arquivos remotos de um sistema de armazenamento a um contêiner para que os dados no volume de dados sejam preservados permanentemente. Mesmo que o contêiner seja excluído, os dados no volume de dados ainda são armazenados no sistema de armazenamento.

Especificações de desempenho do disco EVS

As métricas de desempenho do EVS incluem:

- IOPS: número de operações de leitura/gravação realizadas por um disco do EVS por segundo
- Taxa de transferência: quantidade de dados lidos e gravados em um disco EVS por segundo
- Latência de I/O de leitura/gravação: intervalo mínimo entre duas operações consecutivas de leitura/escrita de um disco EVS

Tabela 8-4 Especificações de desempenho do disco EVS

Parâmetro	SSD extremo	SSD de uso geral	I/O ultra-alta	I/O alta
Capacidade máxima (GiB)	• Disco do sistema: 1.024 • Disco de dados: 32.768	• Disco do sistema: 1.024 • Disco de dados: 32.768	• Disco do sistema: 1.024 • Disco de dados: 32.768	• Disco do sistema: 1.024 • Disco de dados: 32.768
Max. IOPS	128.000	20.000	50.000	5.000
Max. throughput (MiB/s)	1.000	250	350	150
Burst IOPS limit	64.000	8.000	16.000	5.000

Parâmetro	SSD extremo	SSD de uso geral	I/O ultra-alta	I/O alta
Disk IOPS	Min. (128.000, 1.800 + 50 x capacidade)	Mín. (20.000, 1.800 + 12 x capacidade)	Mín. (50.000, 1.800 + 50 x capacidade)	Mín. (5.000, 1.800 + 8 x capacidade)
Disk throughput (MiB/s)	Mín. (1.000, 120 + 0.5 × capacidade)	Mín. (250, 100 + 0.5 × capacidade)	Mín. (350, 120 + 0.5 × capacidade)	Mín. (150, 100 + 0.15 × capacidade)
Single-queue access latency (ms)	Sub-milissegundo	1	1	1–3
API name	ESSD	GPSSD	SSD	SAS

Para obter detalhes sobre o desempenho do disco EVS, consulte **Tipos e desempenho de disco**.

Cenários de aplicações

Os discos EVS podem ser montados nos seguintes modos com base em cenários de aplicação:

- **Uso de um disco EVS existente através de um PV estático**: modo de criação estática, onde você usa um disco EVS existente para criar um PV e, em seguida, monta o armazenamento na carga de trabalho por meio de uma PVC. Esse modo se aplica a cenários em que o armazenamento subjacente está disponível ou é cobrado anualmente/mensalmente.
- **Uso de um disco EVS por meio de um PV dinâmico**: modo de criação dinâmica, onde você não precisa criar volumes do EVS antecipadamente. Em vez disso, especifique uma StorageClass durante a criação do PVC e um disco EVS e um PV serão criados automaticamente. Esse modo se aplica a cenários em que nenhum armazenamento subjacente está disponível.
- **Montagem dinâmica de um disco EVS para um StatefulSet**: somente os StatefulSets oferecem suporte a esse modo. Cada pod está associado a uma PVC e um PV únicos. Depois que um pod é reprogramado, os dados originais ainda podem ser montados nele com base no nome da PVC. Esse modo se aplica aos StatefulSets com vários pods.

Cobrança

- Para montar volumes de armazenamento do tipo do EVS, o modo de cobrança dos discos EVS **criado automaticamente** especificando a StorageClass é de pagamento por padrão e não pode ser alterada para anual/mensal. Se você quiser usar um disco EVS de faturamento anual/mensal, **use um existente**.
- Para obter detalhes sobre os preços de disco EVS, consulte **Cobrança de discos**.

8.3.2 Uso de um disco EVS existente através de um PV estático

O CCE permite que você crie um PV usando um disco EVS existente. Depois que o PV é criado, você pode criar uma PVC e ligá-la ao PV. Esse modo se aplica a cenários em que o armazenamento subjacente está disponível ou é cobrado anualmente/mensalmente..

Pré-requisitos

- Você criou um cluster e instalou o complemento **Armazenamento do contêiner do CCE (Everest)** no cluster.
- Você criou um disco EVS que atende aos seguintes requisitos:
  - O disco EVS existente não pode ser um disco do sistema, um disco DSS ou um disco partilhado.
  - O tipo de dispositivo do disco EVS deve ser **SCSI** (o tipo de dispositivo padrão é **VBD** quando você compra um disco EVS).
  - O disco EVS deve estar disponível e não ser usado por outros recursos.
  - A AZ do disco EVS deve ser a mesma do nó do cluster. Caso contrário, o disco EVS não pode ser montado e o pod não pode iniciar.
  - Se o disco EVS estiver encriptado, a chave tem de estar disponível.
  - Somente os discos EVS no projeto empresarial ao qual o cluster pertence e o projeto empresarial padrão são suportados.
  - Discos EVS com partições ou sistemas de arquivos não ext4 não podem ser importados.
- Se você quiser criar um cluster usando comandos, use kubectl para se conectar ao cluster. Para mais detalhes, consulte **Conexão a um cluster usando o kubectl**.

Restrições

- Os discos EVS não podem ser conectados em AZs e não podem ser usados por várias cargas de trabalho, vários pods da mesma carga de trabalho ou várias tarefas. O compartilhamento de dados de um disco compartilhado não é suportado entre nós em um cluster do CCE. Se um disco EVS for atacado a vários nós, podem ocorrer conflitos de I/O e conflitos de cache de dados. Portanto, crie apenas um pod ao criar uma Implementação que use discos EVS.
- Para clusters anteriores à v1.19.10, se uma política HPA for usada para expandir uma carga de trabalho com discos EVS anexados, os pods existentes não poderão ser lidos ou gravados quando um novo pod for agendado para outro nó.

  Para clusters de v1.19.10 e posterior, se uma política HPA for usada para expandir uma carga de trabalho com discos EVS anexados, um novo pod não poderá ser iniciado porque os discos EVS não podem ser anexados.

Usar um disco EVS existente no console

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster.

Passo 2 Crie estaticamente uma PVC e PV.

1. Escolha **Storage** no painel de navegação e clique na guia **PersistentVolumeClaims (PVCs)**. Clique em **Create PVC** no canto superior direito. Na caixa de diálogo exibida, configure os parâmetros de PVC.

Parâmetro	Descrição
PVC Type	Neste exemplo, selecione **EVS**.

Parâmetro	Descrição
PVC Name	Digite o nome da PVC, que deve ser exclusivo no mesmo namespace.
Creation Method	– Se o armazenamento subjacente estiver disponível, crie um volume de armazenamento ou use um volume de armazenamento existente para criar estaticamente uma PVC com base em se um PV foi criado. – Se nenhum armazenamento subjacente estiver disponível, selecione **Dynamically provision**. Para mais detalhes, consulte **Uso de um disco EVS por meio de um PV dinâmico**. Neste exemplo, selecione **Create new** para criar um PV e uma PVC ao mesmo tempo no console.
PV[a]	Selecione um PV existente no cluster. Crie um PV com antecedência. Para obter detalhes, consulte "Criação de um volume de armazenamento" em **Operações relacionadas**. Você não precisa especificar esse parâmetro neste exemplo.
EVS[b]	Clique em **Select EVS**. Na página exibida, selecione o disco EVS que atende aos seus requisitos e clique em **OK**.
PV Name[b]	Digite o nome do PV, que deve ser exclusivo no mesmo cluster.
Access Mode[b]	Discos EVS suportam apenas **ReadWriteOnce** indicando que um volume de armazenamento pode ser montado em um nó no modo leitura/gravação. Para mais detalhes, consulte **Modos de acesso a volume**.
Reclaim Policy[b]	Você pode selecionar **Delete** ou **Retain** para especificar a política de recuperação do armazenamento subjacente quando a PVC é excluída. Para mais detalhes, consulte **Política de recuperação da PV**.

NOTA

a: o parâmetro está disponível quando **Creation Method** está definido como **Use existing**.

b: o parâmetro está disponível quando **Creation Method** está definido como **Create new**.

2. Clique em **Create** para criar uma PVC e um PV.

   Você pode escolher **Storage** no painel de navegação e exibir a PVC e o PV criados nas páginas de guia **PersistentVolumeClaims (PVCs)** e **PersistentVolumes (PVs)**, respectivamente.

Passo 3 Crie uma aplicação.

1. No painel de navegação à esquerda, clique em **Workloads**. No painel direito, clique na guia **StatefulSets**.
2. Clique em **Create Workload** no canto superior direito. Na página exibida, clique em **Data Storage** na área **Container Settings** e clique em **Add Volume** para selecionar **PVC**.

Monte e use volumes de armazenamento, conforme mostrado na **Tabela 8-5**. Para obter detalhes sobre outros parâmetros, consulte **Cargas de trabalho**.

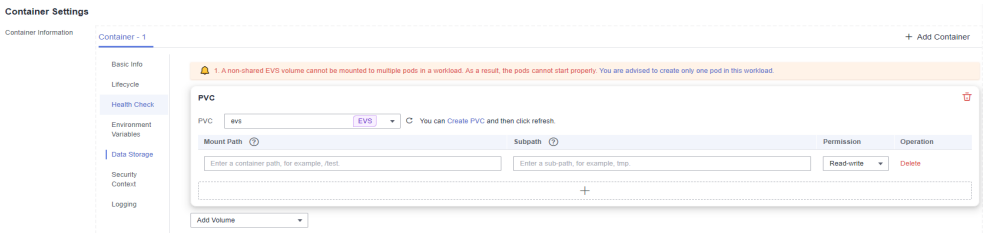
Tabela 8-5 Montagem de um volume de armazenamento

Parâmetro	Descrição
PVC	Selecione um volume do EVS existente. Um volume do EVS não pode ser montado repetidamente em várias cargas de trabalho.
Mount Path	Digite um caminho de montagem, por exemplo, **/tmp**. Este parâmetro indica o caminho do contêiner no qual um volume de dados será montado. Não monte o volume em um diretório do sistema como / ou **/var/run**. Caso contrário, os contêineres estarão com defeito. Monte o volume em um diretório vazio. Se o diretório não estiver vazio, verifique se não há arquivos que afetem a inicialização do contêiner. Caso contrário, os arquivos serão substituídos, causando falhas de inicialização do contêiner ou falhas de criação de carga de trabalho. **AVISO** Se um volume for montado em um diretório de alto risco, use uma conta com permissões mínimas para iniciar o contêiner. Caso contrário, os arquivos de alto risco na máquina host podem ser danificados.
Subpath	Digite um subcaminho, por exemplo, **tmp**, indicando que os dados no caminho de montagem do contêiner serão armazenados na pasta **tmp** do volume. Um subcaminho é usado para montar um volume local para que o mesmo volume de dados seja usado em um único pod. Se este parâmetro for deixado em branco, o caminho raiz é usado por padrão.
Permission	– **Read-only**: você só pode ler os dados nos volumes montados. – **Read/Write**: você pode modificar os volumes de dados montados no caminho. Os dados recém-gravados não serão migrados se o contêiner for migrado, o que pode causar perda de dados.

Neste exemplo, o disco é montado no caminho **/data** do contêiner. Os dados de contêiner gerados nesse caminho são armazenados no disco EVS.

NOTA

Um disco EVS não compartilhado não pode ser conectado a vários pods em uma carga de trabalho. Caso contrário, os pods não poderão ser iniciados corretamente. Certifique-se de que o número de pods de carga de trabalho seja 1 ao anexar um disco EVS.



3. Após a configuração, clique em **Create Workload**.

   Depois que a carga de trabalho for criada, os dados no diretório de montagem do contêiner serão armazenados persistentemente. Verifique o armazenamento referindo-se a **Verificar a persistência dos dados**.

----Fim

(kubectl) Usar um disco EVS existente

Passo 1 Use o kubectl para se conectar ao cluster.

Passo 2 Crie um PV. Se um PV tiver sido criado no cluster, ignore esta etapa.

1. Crie o arquivo **pv-evs.yaml**.

```
apiVersion: v1
kind: PersistentVolume
metadata:
  annotations:
    pv.kubernetes.io/provisioned-by: everest-csi-provisioner
    everest.io/reclaim-policy: retain-volume-only         # (Optional) The PV
is deleted while the underlying volume is retained.
  name: pv-evs    # PV name.
  labels:
    failure-domain.beta.kubernetes.io/region: <your_region>   # Region of the
node where the application is to be deployed.
    failure-domain.beta.kubernetes.io/zone: <your_zone>       # AZ of the
node where the application is to be deployed.
spec:
  accessModes:
    - ReadWriteOnce     # Access mode. The value must be ReadWriteOnce for
EVS disks.
  capacity:
    storage: 10Gi       # EVS disk capacity, in the unit of Gi. The value
ranges from 1 to 32768.
  csi:
    driver: disk.csi.everest.io     # Dependent storage driver for the
mounting.
    fsType: ext4
    volumeHandle: <your_volume_id>   # Volume ID of the EVS disk.
    volumeAttributes:
      everest.io/disk-mode: SCSI           # Device type of the EVS disk.
Only SCSI is supported.
      everest.io/disk-volume-type: SAS     # EVS disk type.
      storage.kubernetes.io/csiProvisionerIdentity: everest-csi-provisioner
      everest.io/crypt-key-id: <your_key_id>    # (Optional) Encryption key
ID. Mandatory for an encrypted disk.
      everest.io/enterprise-project-id: <your_project_id>  # (Optional)
Enterprise project ID. If an enterprise project is specified, use the same
enterprise project when creating a PVC. Otherwise, the PVC cannot be bound to
a PV.
  persistentVolumeReclaimPolicy: Delete    # Reclaim policy.
  storageClassName: csi-disk              # Storage class name. The value
must be csi-disk for EVS disks.
```


Tabela 8-6 Parâmetros principais

Parâmetro	Obrigatório	Descrição
everest.io/reclaim-policy: retain-volume-only	Não	Opcional. Atualmente, apenas **retain-volume-only** é suportado. Este campo é válido somente quando a versão do everest for 1.2.9 ou posterior e a política de recuperação for **Delete**. Se a política de recuperação for **Delete** e o valor atual for **retain-volume-only**, o PV associado será excluído enquanto o volume de armazenamento subjacente for retido, quando uma PVC for excluída.
failure-domain.beta.kubernetes.io/region	Sim	Região onde o cluster está localizado. Para obter detalhes sobre o valor da **region**, consulte **Regiões e pontos de extremidade**.
failure-domain.beta.kubernetes.io/zone	Sim	AZ onde o volume do EVS é criado. Deve ser a mesma que a AZ planejada para a carga de trabalho. Para obter detalhes sobre o valor da **zone**, consulte **Regiões e pontos de extremidade**.
volumeHandle	Sim	ID do volume do disco EVS. Para obter o ID do volume, faça logon no **Cloud Server Console**. No painel de navegação, escolha **Elastic Volume Service** > **Disks**. Clique no nome do disco EVS de destino para acessar sua página de detalhes. Na página de guia **Summary**, clique no botão copiar após **ID**.
everest.io/disk-volume-type	Sim	Tipos de disco EVS. Todas as letras estão em maiúsculas. – **SAS**: I/O alta – **SSD**: I/O ultra-alta – **GPSSD**: SSD de uso geral – **ESSD**: SSD extremo
everest.io/crypt-key-id	Não	Obrigatório quando o disco EVS é criptografado. Insira o ID da chave de criptografia selecionada durante a criação do disco EVS. Para obter o ID da chave de criptografia, faça logon no **Cloud Server Console**. No painel de navegação, escolha **Elastic Volume Service** > **Disks**. Clique no nome do disco EVS de destino para acessar sua página de detalhes. Na página de guia **Summary**, copie o valor de **KMS Key ID** na área **Configuration Information**.

Parâmetro	Obrigatório	Descrição
everest.io/enterprise-project-id	Não	Opcional. ID do projeto empresarial do disco EVS. Se um projeto empresarial for especificado, use o mesmo projeto empresarial ao criar uma PVC. Caso contrário, a PVC não pode ser vinculado a um PV. Para obter o código do projeto empresarial, efetue logon no **Cloud Server Console**. No painel de navegação, escolha **Elastic Volume Service** > **Disks**. Clique no nome do disco EVS de destino para acessar sua página de detalhes. Na página de guia **Summary**, clique no projeto empresarial em **Management Information** para acessar o console do projeto empresarial. Copie o ID correspondente para obter o ID do projeto corporativo ao qual o disco EVS pertence.
persistentVolumeReclaimPolicy	Sim	Uma política de recuperação é suportada quando a versão do cluster é ou posterior a 1.19.10 e a versão do everest é ou posterior a 1.2.9. As políticas de recuperação **Delete** e **Retain** são suportadas. Para mais detalhes, consulte **Política de recuperação da PV**. Se for necessária uma alta segurança de dados, selecione **Retain** para impedir que os dados sejam excluídos por engano. **Delete**: – Se **everest.io/reclaim-policy** não for especificada, o volume de PV e EVS serão excluídos quando uma PVC for excluída. – Se **everest.io/reclaim-policy** estiver definida para **retain-volume-only**, quando uma PVC for excluída, o PV será excluído, mas os recursos do EVS serão retidos. **Retain**: quando uma PVC é excluída, o PV e os recursos de armazenamento subjacentes não são excluídos. Em vez disso, você deve excluir manualmente esses recursos. Depois disso, o PV está no estado **Released** e não pode ser vinculado à PVC novamente.
storageClassName	Sim	O nome da classe de armazenamento para discos EVS é **csi-disk**.

2. Execute o seguinte comando para criar um PV:
```
kubectl apply -f pv-evs.yaml
```

Passo 3 Crie uma PVC.

1. Crie o arquivo **pvc-evs.yaml**.
```
apiVersion: v1
kind: PersistentVolumeClaim
```



```
metadata:
  name: pvc-evs
  namespace: default
  annotations:
      everest.io/disk-volume-type: SAS    # EVS disk type.
    everest.io/crypt-key-id: <your_key_id>    # (Optional) Encryption key ID.
Mandatory for an encrypted disk.
    everest.io/enterprise-project-id: <your_project_id>  # (Optional)
Enterprise project ID. If an enterprise project is specified, use the same
enterprise project when creating a PVC. Otherwise, the PVC cannot be bound to
a PV.
  labels:
    failure-domain.beta.kubernetes.io/region: <your_region>   # Region of the
node where the application is to be deployed.
    failure-domain.beta.kubernetes.io/zone: <your_zone>       # AZ of the
node where the application is to be deployed.
spec:
  accessModes:
  - ReadWriteOnce               # The value must be ReadWriteOnce for EVS
disks.
  resources:
    requests:
      storage: 10Gi             # EVS disk capacity, ranging from 1 to 32768.
The value must be the same as the storage size of the existing PV.
  storageClassName: csi-disk    # Storage class type for EVS disks.
  volumeName: pv-evs            # PV name.
```

Tabela 8-7 Parâmetros principais

Parâmetro	Obrigatório	Descrição
failure-domain.beta.kubernetes.io/region	Sim	Região onde o cluster está localizado. Para obter detalhes sobre o valor da **region**, consulte **Regiões e pontos de extremidade**.
failure-domain.beta.kubernetes.io/zone	Sim	AZ onde o volume do EVS é criado. Deve ser a mesma que a AZ planejada para a carga de trabalho. Para obter detalhes sobre o valor da **zone**, consulte **Regiões e pontos de extremidade**.
storage	Sim	Capacidade solicitada na PVC, em Gi. O valor deve ser o mesmo que o tamanho de armazenamento do PV existente.
volumeName	Sim	Nome do PV, que deve ser o mesmo que o nome do PV em **1**.
storageClassName	Sim	Nome da classe de armazenamento, que deve ser o mesmo que a classe de armazenamento do PV em **1**. O nome da classe de armazenamento dos volumes do EVS é **csi-disk**.

2. Execute o seguinte comando para criar uma PVC:
   ```
   kubectl apply -f pvc-evs.yaml
   ```

Passo 4 Crie uma aplicação.

1. Crie um arquivo chamado **web-evs.yaml**. Neste exemplo, o volume do EVS é montado no caminho /**data**.

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: web-evs
  namespace: default
spec:
 replicas: 1            # The number of workload replicas that use the EVS
volume must be 1.
  selector:
    matchLabels:
      app: web-evs
  serviceName: web-evs   # Headless Service name.
  template:
    metadata:
      labels:
        app: web-evs
    spec:
      containers:
      - name: container-1
        image: nginx:latest
        volumeMounts:
        - name: pvc-disk    # Volume name, which must be the same as the
volume name in the volumes field.
          mountPath: /data  # Location where the storage volume is mounted.
      imagePullSecrets:
        - name: default-secret
      volumes:
        - name: pvc-disk    # Volume name, which can be customized.
          persistentVolumeClaim:
            claimName: pvc-evs    # Name of the created PVC.
---
apiVersion: v1
kind: Service
metadata:
  name: web-evs   # Headless Service name.
  namespace: default
  labels:
    app: web-evs
spec:
  selector:
    app: web-evs
  clusterIP: None
  ports:
    - name: web-evs
      targetPort: 80
      nodePort: 0
      port: 80
      protocol: TCP
  type: ClusterIP
```

2. Execute o seguinte comando para criar uma carga de trabalho na qual o volume do EVS está montado:

```
kubectl apply -f web-evs.yaml
```

Depois que a carga de trabalho for criada, os dados no diretório de montagem do contêiner serão armazenados persistentemente. Verifique o armazenamento referindo-se a **Verificar a persistência dos dados**.

----Fim

Verificar a persistência dos dados

Passo 1 Visualize a aplicação implementado e os arquivos de volume do EVS.

1. Execute o seguinte comando para exibir o pod criado:


```
kubectl get pod | grep web-evs
```

Saída esperada:

```
web-evs-0                   1/1     Running   0               38s
```

2. Execute o seguinte comando para verificar se o volume do EVS foi montado no caminho **/data**:

```
kubectl exec web-evs-0 -- df | grep data
```

Saída esperada:

```
/dev/sdc              10255636     36888  10202364   0% /data
```

3. Execute o seguinte comando para exibir os arquivos no caminho **/data**:

```
kubectl exec web-evs-0 -- ls /data
```

Saída esperada:

```
lost+found
```

Passo 2 Execute o seguinte comando para criar um arquivo chamado **static** no caminho **/data**:

```
kubectl exec web-evs-0 --  touch /data/static
```

Passo 3 Execute o seguinte comando para exibir os arquivos no caminho **/data**:

```
kubectl exec web-evs-0 -- ls /data
```

Saída esperada:

```
lost+found
static
```

Passo 4 Execute o seguinte comando para excluir o pod chamado **web-evs-0**:

```
kubectl delete pod web-evs-0
```

Saída esperada:

```
pod "web-evs-0" deleted
```

Passo 5 Após a exclusão, o controlador de StatefulSet cria automaticamente uma réplica com o mesmo nome. Execute o seguinte comando para verificar se os arquivos no caminho **/data** foram modificados:

```
kubectl exec web-evs-0 -- ls /data
```

Saída esperada:

```
lost+found
static
```

Se o arquivo **static** ainda existir, os dados no volume de EVS podem ser armazenados persistentemente.

----Fim

Operações relacionadas

Você também pode executar as operações listadas em **Tabela 8-8**.

Tabela 8-8 Operações relacionadas

Operação	Descrição	Procedimento
Criar um volume de armazenamento (PV)	Crie um PV no console do CCE.	1. Escolha **Storage** no painel de navegação e clique na guia **PersistentVolumes (PVs)**. Clique em **Create Volume** no canto superior direito. Na caixa de diálogo exibida, configure os parâmetros. ● **Volume Type**: selecione **EVS**. ● **EVS**: clique em **Select EVS**. Na página exibida, selecione o disco EVS que atende aos seus requisitos e clique em **OK**. ● **PV Name**: digite o nome do PV, que deve ser exclusivo no mesmo cluster. ● **Access Mode**: discos EVS suportam apenas **ReadWriteOnce** indicando que um volume de armazenamento pode ser montado em um nó no modo leitura/gravação. Para mais detalhes, consulte **Modos de acesso a volume**. ● **Reclaim Policy**: **Delete** ou **Retain**. Para mais detalhes, consulte **Política de recuperação da PV**. 2. Clique em **Create**.
Expandir a capacidade de um disco EVS	Expanda rapidamente a capacidade de um disco EVS montado no console do CCE. Somente a capacidade de discos EVS de pagamento por uso pode ser expandida no console do CCE. Para expandir a capacidade de discos EVS anuais/mensais, clique no nome do volume para ir para o console do EVS.	1. Escolha **Storage** no painel de navegação e clique na guia **PersistentVolumeClaims (PVCs)**. Clique em **More** na coluna **Operation** do PVC de destino e selecione **Scale-out**. 2. Insira a capacidade a ser adicionada e clique em **OK**.

Operação	Descrição	Procedimento
Visualizar eventos	Você pode visualizar nomes de eventos, tipos de eventos, número de ocorrências, eventos do Kubernetes, horário da primeira ocorrência e horário da última ocorrência da PVC ou PV.	1. Escolha **Storage** no painel de navegação e clique na guia **PersistentVolumeClaims (PVCs)** ou **PersistentVolumes (PVs)**. 2. Clique em **View Events** na coluna **Operation** da PVC ou do PV de destino para exibir os eventos gerados em uma hora (os dados do evento são mantidos por uma hora).
Exibir um arquivo YAML	Você pode visualizar, copiar e fazer download dos arquivos YAML de uma PVC ou um PV.	1. Escolha **Storage** no painel de navegação e clique na guia **PersistentVolumeClaims (PVCs)** ou **PersistentVolumes (PVs)**. 2. Clique em **View YAML** na coluna **Operation** da PVC ou PV de destino para exibir ou fazer download do YAML.

8.3.3 Uso de um disco EVS por meio de um PV dinâmico

O CCE permite que você especifique uma StorageClass para criar automaticamente um disco EVS e o PV correspondente. Essa função é aplicável quando nenhum volume de armazenamento subjacente está disponível.

Pré-requisitos

- Você criou um cluster e instalou o complemento **Armazenamento do contêiner do CCE (Everest)** no cluster.
- Se você quiser criar um cluster usando comandos, use kubectl para se conectar ao cluster. Para mais detalhes, consulte **Conexão a um cluster usando o kubectl**.

Restrições

- Os discos EVS não podem ser conectados em AZs e não podem ser usados por várias cargas de trabalho, vários pods da mesma carga de trabalho ou várias tarefas. O compartilhamento de dados de um disco compartilhado não é suportado entre nós em um cluster do CCE. Se um disco EVS for atacado a vários nós, podem ocorrer conflitos de I/O e conflitos de cache de dados. Portanto, crie apenas um pod ao criar uma Implementação que use discos EVS.
- Para clusters anteriores à v1.19.10, se uma política HPA for usada para expandir uma carga de trabalho com discos EVS anexados, os pods existentes não poderão ser lidos ou gravados quando um novo pod for agendado para outro nó.

  Para clusters de v1.19.10 e posterior, se uma política HPA for usada para expandir uma carga de trabalho com discos EVS anexados, um novo pod não poderá ser iniciado porque os discos EVS não podem ser anexados.
- As tags de recurso podem ser adicionadas quando os discos EVS são criados dinamicamente. Depois que os discos EVS são criados, as tags de recurso não podem ser atualizadas no CCE. Para atualizá-los, acesse o console do EVS. Se você usar um disco

EVS existente para criar um PV, também precisará adicionar ou atualizar tags de recursos no console do EVS.

(Console) Criar automaticamente um disco EVS

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster.

Passo 2 Crie dinamicamente uma PVC e um PV.

1. Escolha **Storage** no painel de navegação e clique na guia **PersistentVolumeClaims (PVCs)**. Clique em **Create PVC** no canto superior direito. Na caixa de diálogo exibida, configure os parâmetros de PVC.

Parâmetro	Descrição
PVC Type	Neste exemplo, selecione **EVS**.
PVC Name	Digite o nome da PVC, que deve ser exclusivo no mesmo namespace.
Creation Method	– Se nenhum armazenamento subjacente estiver disponível, selecione **Dynamically provision** para criar uma PVC, um PV e armazenamento subjacente no console no modo de cascata. – Se o armazenamento subjacente estiver disponível, crie um PV ou use um PV existente para criar estaticamente uma PVC com base na disponibilidade de um PV. Para mais detalhes, consulte **Uso de um disco EVS existente através de um PV estático**. Neste exemplo, selecione **Dynamically provision**.
Storage Classes	A classe de armazenamento para discos EVS é **csi-disk**.
AZ	Selecione a AZ do disco EVS. A AZ deve ser a mesma do nó do cluster. **NOTA** Um disco EVS só pode ser montado em um nó na mesma AZ. Depois que um disco EVS é criado, sua AZ não pode ser alterada.
Disk Type	Selecione um tipo de disco EVS.
Access Mode	Discos EVS suportam apenas **ReadWriteOnce** indicando que um volume de armazenamento pode ser montado em um nó no modo leitura/gravação. Para mais detalhes, consulte **Modos de acesso a volume**.
Capacity (GiB)	Capacidade do volume de armazenamento solicitado.
Encryption	Você pode selecionar **Encryption** e uma chave de criptografia para criptografar o armazenamento subjacente. Antes de usar a função de criptografia, verifique se a região onde o disco EVS está localizado suporta criptografia de disco.
Enterprise Project	Projetos empresariais suportados: padrão, aquele ao qual o cluster pertence ou aquele especificado pela classe de armazenamento.

Parâmetro	Descrição
Resource Tag	Você pode adicionar tags de recurso para classificar recursos, que é suportado apenas quando a versão de everest no cluster é 2.1.39 ou posterior. Você pode criar **tags predefinidas** no console do TMS. As tags predefinidas estão disponíveis para todos os recursos que suportam tags. Você pode usar tags predefinidas para melhorar a criação de tags e a eficiência da migração de recursos. Para obter detalhes, consulte **Criação de tags predefinidas**. O CCE cria automaticamente as tags de sistema **CCE-Cluster-ID**=*{Cluster ID}*, **CCE-Cluster-Name**=*{Cluster name}* e **CCE-Namespace**=*{Namespace name}*. Essas tags não podem ser modificadas. **NOTA** Depois que um PV dinâmico do tipo do EVS é criado, as tags de recurso não podem ser atualizadas no console do CCE. Para atualizar essas tags de recurso, acesse o console do EVS.

2. Clique em **Create**.

   Você pode escolher **Storage** no painel de navegação e exibir a PVC e o PV criados nas páginas de guia **PersistentVolumeClaims (PVCs)** e **PersistentVolumes (PVs)**, respectivamente.

Passo 3 Crie uma aplicação.

1. No painel de navegação à esquerda, clique em **Workloads**. No painel direito, clique na guia **StatefulSets**.
2. Clique em **Create Workload** no canto superior direito. Na página exibida, clique em **Data Storage** na área **Container Settings** e clique em **Add Volume** para selecionar **PVC**.

   Monte e use volumes de armazenamento, conforme mostrado na **Tabela 8-9**. Para obter detalhes sobre outros parâmetros, consulte **Cargas de trabalho**.

Tabela 8-9 Montagem de um volume de armazenamento

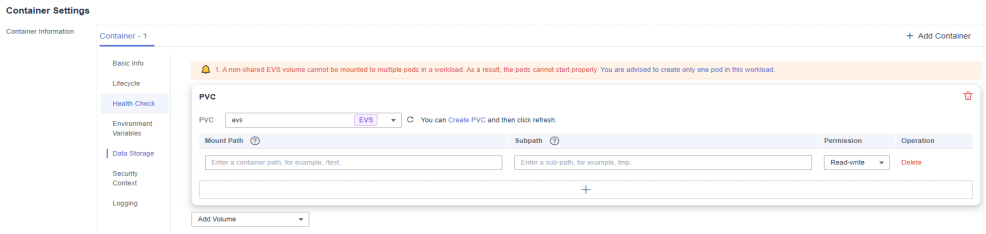
Parâmetro	Descrição
PVC	Selecione um volume do EVS existente. Um volume do EVS não pode ser montado repetidamente em várias cargas de trabalho.

Parâmetro	Descrição
Mount Path	Digite um caminho de montagem, por exemplo, **/tmp**. Este parâmetro indica o caminho do contêiner no qual um volume de dados será montado. Não monte o volume em um diretório do sistema como / ou **/var/run**. Caso contrário, os contêineres estarão com defeito. Monte o volume em um diretório vazio. Se o diretório não estiver vazio, verifique se não há arquivos que afetem a inicialização do contêiner. Caso contrário, os arquivos serão substituídos, causando falhas de inicialização do contêiner ou falhas de criação de carga de trabalho. **AVISO** Se um volume for montado em um diretório de alto risco, use uma conta com permissões mínimas para iniciar o contêiner. Caso contrário, os arquivos de alto risco na máquina host podem ser danificados.
Subpath	Digite um subcaminho, por exemplo, **tmp**, indicando que os dados no caminho de montagem do contêiner serão armazenados na pasta **tmp** do volume. Um subcaminho é usado para montar um volume local para que o mesmo volume de dados seja usado em um único pod. Se este parâmetro for deixado em branco, o caminho raiz é usado por padrão.
Permission	– **Read-only**: você só pode ler os dados nos volumes montados. – **Read/Write**: você pode modificar os volumes de dados montados no caminho. Os dados recém-gravados não serão migrados se o contêiner for migrado, o que pode causar perda de dados.

Neste exemplo, o disco é montado no caminho **/data** do contêiner. Os dados de contêiner gerados nesse caminho são armazenados no disco EVS.

NOTA

Um disco EVS não compartilhado não pode ser conectado a vários pods em uma carga de trabalho. Caso contrário, os pods não poderão ser iniciados corretamente. Certifique-se de que o número de pods de carga de trabalho seja 1 ao anexar um disco EVS.



3. Após a configuração, clique em **Create Workload**.

   Depois que a carga de trabalho for criada, os dados no diretório de montagem do contêiner serão armazenados persistentemente. Verifique o armazenamento referindo-se a **Verificar a persistência dos dados**.

----Fim

(kubectl) Criação automática de um disco EVS

Passo 1 Use o kubectl para se conectar ao cluster.

Passo 2 Use **StorageClass** para criar dinamicamente uma PVC e um PV.

1. Crie o arquivo **pvc-evs-auto.yaml**.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-evs-auto
  namespace: default
  annotations:
      everest.io/disk-volume-type: SAS    # EVS disk type.
    everest.io/crypt-key-id: <your_key_id>    # (Optional) Encryption key ID.
Mandatory for an encrypted disk.

    everest.io/enterprise-project-id: <your_project_id>  # (Optional)
Enterprise project ID. If an enterprise project is specified, use the same
enterprise project when creating a PVC. Otherwise, the PVC cannot be bound to
a PV.

  labels:
    failure-domain.beta.kubernetes.io/region: <your_region>   # Region of the
node where the application is to be deployed.
    failure-domain.beta.kubernetes.io/zone: <your_zone>       # AZ of the
node where the application is to be deployed.
spec:
  accessModes:
  - ReadWriteOnce               # The value must be ReadWriteOnce for EVS
disks.
  resources:
    requests:
      storage: 10Gi             # EVS disk capacity, ranging from 1 to 32768.
  storageClassName: csi-disk    # Storage class type for EVS disks.
```

Tabela 8-10 Parâmetros principais

Parâmetro	Obrigatório	Descrição
failure-domain.beta.kubernetes.io/region	Sim	Região onde o cluster está localizado. Para obter detalhes sobre o valor da **region**, consulte **Regiões e pontos de extremidade**.
failure-domain.beta.kubernetes.io/zone	Sim	AZ onde o volume do EVS é criado. Deve ser a mesma que a AZ planejada para a carga de trabalho. Para obter detalhes sobre o valor da **zone**, consulte **Regiões e pontos de extremidade**.
everest.io/disk-volume-type	Sim	Tipos de disco EVS. Todas as letras estão em maiúsculas. – **SAS**: I/O alta – **SSD**: I/O ultra-alta – **GPSSD**: SSD de uso geral – **ESSD**: SSD extremo

Parâmetro	Obrigatório	Descrição
everest.io/crypt-key-id	Não	Este parâmetro é obrigatório quando um disco EVS é criptografado. Insira o ID da chave de criptografia selecionada durante a criação do disco EVS. Você pode usar uma chave personalizada ou a chave padrão chamada **evs/default**. Para obter um ID da chave, faça logon no console do DEW, localize a chave a ser criptografada e copie o ID da chave.
everest.io/enterprise-project-id	Não	Opcional. ID do projeto empresarial do disco EVS. Se um projeto empresarial for especificado, use o mesmo projeto empresarial ao criar uma PVC. Caso contrário, a PVC não pode ser vinculada a um PV. Para obter um ID de projeto empresarial, efetue logon no console do EPS, clique no nome do projeto empresarial de destino e copie o ID do projeto empresarial.
	Sim	Capacidade de PVC solicitada, em Gi. O valor varia de **1** a **32768**.
storageClassName	Sim	O nome da classe de armazenamento dos volumes do EVS é **csi-disk**.

2. Execute o seguinte comando para criar uma PVC:
```
kubectl apply -f pvc-evs-auto.yaml
```

Passo 3 Crie uma aplicação.

1. Crie um arquivo chamado **web-evs-auto.yaml**. Neste exemplo, o volume do EVS é montado no caminho **/data**.
```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: web-evs-auto
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: web-evs-auto
  serviceName: web-evs-auto   # Headless Service name.
  template:
    metadata:
      labels:
        app: web-evs-auto
    spec:
      containers:
      - name: container-1
        image: nginx:latest
        volumeMounts:
        - name: pvc-disk    # Volume name, which must be the same as the volume name in the volumes field.
          mountPath: /data  # Location where the storage volume is mounted.
```



```
      imagePullSecrets:
        - name: default-secret
      volumes:
        - name: pvc-disk    # Volume name, which can be customized.
          persistentVolumeClaim:
            claimName: pvc-evs-auto    # Name of the created PVC.
---
apiVersion: v1
kind: Service
metadata:
  name: web-evs-auto   # Headless Service name.
  namespace: default
  labels:
    app: web-evs-auto
spec:
  selector:
    app: web-evs-auto
  clusterIP: None
  ports:
    - name: web-evs-auto
      targetPort: 80
      nodePort: 0
      port: 80
      protocol: TCP
  type: ClusterIP
```

2. Execute o seguinte comando para criar uma carga de trabalho na qual o volume do EVS está montado:

```
kubectl apply -f web-evs-auto.yaml
```

Depois que a carga de trabalho for criada, os dados no diretório de montagem do contêiner serão armazenados persistentemente. Verifique o armazenamento referindo-se a **Verificar a persistência dos dados**.

----Fim

Verificar a persistência dos dados

Passo 1 Visualize a aplicação implementado e os arquivos de volume do EVS.

1. Execute o seguinte comando para exibir o pod criado:

```
kubectl get pod | grep web-evs-auto
```

Saída esperada:

```
web-evs-auto-0                   1/1     Running   0               38s
```

2. Execute o seguinte comando para verificar se o volume do EVS foi montado no caminho **/data**:

```
kubectl exec web-evs-auto-0 -- df | grep data
```

Saída esperada:

```
/dev/sdc              10255636     36888  10202364   0% /data
```

3. Execute o seguinte comando para exibir os arquivos no caminho **/data**:

```
kubectl exec web-evs-auto-0 -- ls /data
```

Saída esperada:

```
lost+found
```

Passo 2 Execute o seguinte comando para criar um arquivo chamado **static** no caminho **/data**:

```
kubectl exec web-evs-auto-0 --  touch /data/static
```

Passo 3 Execute o seguinte comando para exibir os arquivos no caminho **/data**:

```
kubectl exec web-evs-auto-0 -- ls /data
```

Saída esperada:


```
lost+found
static
```

Passo 4 Execute o seguinte comando para excluir o pod chamado **web-evs-auto-0**:

```
kubectl delete pod web-evs-auto-0
```

Saída esperada:

```
pod "web-evs-auto-0" deleted
```

Passo 5 Após a exclusão, o controlador de StatefulSet cria automaticamente uma réplica com o mesmo nome. Execute o seguinte comando para verificar se os arquivos no caminho **/data** foram modificados:

```
kubectl exec web-evs-auto-0 -- ls /data
```

Saída esperada:

```
lost+found
static
```

Se o arquivo **static** ainda existir, os dados no volume de EVS podem ser armazenados persistentemente.

----Fim

Operações relacionadas

Você também pode executar as operações listadas em **Tabela 8-11**.

Tabela 8-11 Operações relacionadas

Operação	Descrição	Procedimento
Expandir a capacidade de um disco EVS	Expanda rapidamente a capacidade de um disco EVS montado no console do CCE. Somente a capacidade de discos EVS de pagamento por uso pode ser expandida no console do CCE. Para expandir a capacidade de discos EVS anuais/mensais, clique no nome do volume para ir para o console do EVS.	1. Escolha **Storage** no painel de navegação e clique na guia **PersistentVolumeClaims (PVCs)**. Clique em **More** na coluna **Operation** da PVC de destino e selecione **Scale-out**. 2. Insira a capacidade a ser adicionada e clique em **OK**.

Operação	Descrição	Procedimento
Visualizar eventos	Você pode visualizar nomes de eventos, tipos de eventos, número de ocorrências, eventos do Kubernetes, horário da primeira ocorrência e horário da última ocorrência da PVC ou PV.	1. Escolha **Storage** no painel de navegação e clique na guia **PersistentVolumeClaims (PVCs)** ou **PersistentVolumes (PVs)**. 2. Clique em **View Events** na coluna **Operation** da PVC ou do PV de destino para exibir os eventos gerados em uma hora (os dados do evento são mantidos por uma hora).
Exibir um arquivo YAML	Você pode visualizar, copiar e fazer download dos arquivos YAML de uma PVC ou um PV.	1. Escolha **Storage** no painel de navegação e clique na guia **PersistentVolumeClaims (PVCs)** ou **PersistentVolumes (PVs)**. 2. Clique em **View YAML** na coluna **Operation** da PVC ou PV de destino para exibir ou fazer download do YAML.

8.3.4 Montagem dinâmica de um disco EVS para um StatefulSet

Cenários de aplicações

A montagem dinâmica está disponível apenas para a criação de um **StatefulSet**. Ela é implementada por meio de um modelo de declaração de volume (campo **volumeClaimTemplates**) e depende da classe de armazenamento para provisionar PVs dinamicamente. Nesse modo, cada pod em um StatefulSet de vários pods está associado a uma PVC e um PV exclusivos. Depois que um pod é reprogramado, os dados originais ainda podem ser montados nele com base no nome da PVC. No modo de montagem comum de uma Implementação, se ReadWriteMany for suportado, vários pods da Implementação serão montados no mesmo armazenamento subjacente.

Pré-requisitos

- Você criou um cluster e instalou o complemento **Armazenamento do contêiner do CCE (Everest)** no cluster.
- Se você quiser criar um cluster usando comandos, use kubectl para se conectar ao cluster. Para mais detalhes, consulte **Conexão a um cluster usando o kubectl**.

(Console) Montagem dinâmica de um disco EVS

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster.

Passo 2 No painel de navegação à esquerda, clique em **Workloads**. No painel direito, clique na guia **StatefulSets**.

Passo 3 Clique em **Create Workload** no canto superior direito. Na página exibida, clique em **Data Storage** na área **Container Settings** e clique em **Add Volume** para selecionar **VolumeClaimTemplate (VTC)**.

Passo 4 Clique em **Create PVC**. Na caixa de diálogo exibida, configure os parâmetros de PVC.

Clique em **Create**.

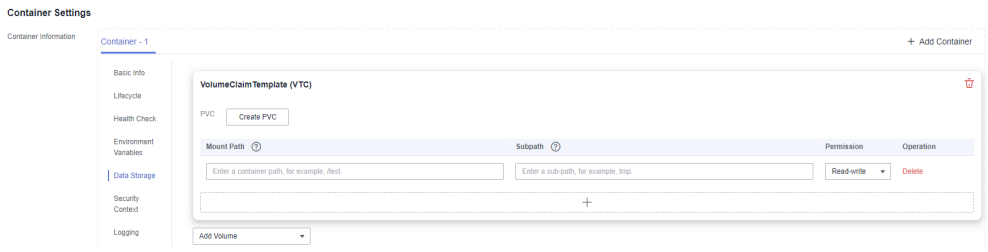
Parâmetro	Descrição
PVC Type	Neste exemplo, selecione **EVS**.
PVC Name	Digite o nome da PVC. Depois que uma PVC é criada, um sufixo é adicionado automaticamente com base no número de pods. O formato é *<Custom PVC name>-<Serial number>*, por exemplo, example-0.
Creation Method	Você só pode selecionar **Dynamically provision** para criar uma PVC, um PV e armazenamento subjacente no console no modo de cascata.
Storage Classes	A classe de armazenamento para discos EVS é **csi-disk**.
AZ	Selecione a AZ do disco EVS. A AZ deve ser a mesma do nó do cluster. **NOTA** Um disco EVS só pode ser montado em um nó na mesma AZ. Depois que um disco EVS é criado, sua AZ não pode ser alterada.
Disk Type	Selecione um tipo de disco EVS.
Access Mode	Discos EVS suportam apenas **ReadWriteOnce** indicando que um volume de armazenamento pode ser montado em um nó no modo leitura/gravação. Para mais detalhes, consulte **Modos de acesso a volume**.
Capacity (GiB)	Capacidade do volume de armazenamento solicitado.
Encryption	Você pode selecionar **Encryption** e uma chave de criptografia para criptografar o armazenamento subjacente. Somente discos EVS e sistemas de arquivos do SFS suportam criptografia.
Enterprise Project	Projetos empresariais suportados: padrão, aquele ao qual o cluster pertence ou aquele especificado pela classe de armazenamento.
Resource Tag	Você pode adicionar tags de recurso para classificar recursos, que é suportado apenas quando a versão de everest no cluster é 2.1.39 ou posterior. Você pode criar **tags predefinidas** no console do TMS. As tags predefinidas estão disponíveis para todos os recursos que suportam tags. Você pode usar tags predefinidas para melhorar a criação de tags e a eficiência da migração de recursos. Para obter detalhes, consulte **Criação de tags predefinidas**. O CCE cria automaticamente as tags de sistema **CCE-Cluster-ID=***{Cluster ID}*, **CCE-Cluster-Name=***{Cluster name}* e **CCE-Namespace=***{Namespace name}*. Essas tags não podem ser modificadas. **NOTA** Depois que um PV dinâmico do tipo de EVS é criado, as tags de recurso não podem ser atualizadas no console do CCE. Para atualizar essas tags de recurso, acesse o console do EVS.

Passo 5 Insira o caminho no qual o volume está montado.

Tabela 8-12 Montagem de um volume de armazenamento

Parâmetro	Descrição
Mount Path	Digite um caminho de montagem, por exemplo, **/tmp**. Este parâmetro indica o caminho do contêiner no qual um volume de dados será montado. Não monte o volume em um diretório do sistema como / ou **/var/run**. Caso contrário, os contêineres estarão com defeito. Monte o volume em um diretório vazio. Se o diretório não estiver vazio, verifique se não há arquivos que afetem a inicialização do contêiner. Caso contrário, os arquivos serão substituídos, causando falhas de inicialização do contêiner ou falhas de criação de carga de trabalho. **AVISO** Se um volume for montado em um diretório de alto risco, use uma conta com permissões mínimas para iniciar o contêiner. Caso contrário, arquivos de alto risco no host podem ser danificados.
Subpath	Digite um subcaminho, por exemplo, **tmp**, indicando que os dados no caminho de montagem do contêiner serão armazenados na pasta **tmp** do volume. Um subcaminho é usado para montar um volume local para que o mesmo volume de dados seja usado em um único pod. Se este parâmetro for deixado em branco, o caminho raiz é usado por padrão.
Permission	● **Read-only**: você só pode ler os dados nos volumes montados. ● **Read/Write**: você pode modificar os volumes de dados montados no caminho. Os dados recém-gravados não serão migrados se o contêiner for migrado, o que pode causar perda de dados.

Neste exemplo, o disco é montado no caminho **/data** do contêiner. Os dados de contêiner gerados nesse caminho são armazenados no disco EVS.



Passo 6 Monte e use dinamicamente volumes de armazenamento. Para obter detalhes sobre outros parâmetros, consulte **Criação de um StatefulSet**. Após a configuração, clique em **Create Workload**.

Depois que a carga de trabalho for criada, os dados no diretório de montagem do contêiner serão armazenados persistentemente. Verifique o armazenamento referindo-se a **Verificar a persistência dos dados**.

----Fim

Montagem dinâmica de um volume do EVS usando kubectl

Passo 1 Use o kubectl para se conectar ao cluster.

Passo 2 Crie um arquivo chamado **statefulset-evs.yaml**. Neste exemplo, o volume do EVS é montado no caminho **/data**.

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: statefulset-evs
  namespace: default
spec:
  selector:
    matchLabels:
      app: statefulset-evs
  template:
    metadata:
      labels:
        app: statefulset-evs
    spec:
      containers:
        - name: container-1
          image: nginx:latest
          volumeMounts:
            - name: pvc-disk           # The value must be the same as that in
the volumeClaimTemplates field.
              mountPath: /data         # Location where the storage volume is
mounted.
      imagePullSecrets:
        - name: default-secret
  serviceName: statefulset-evs         # Headless Service name.
  replicas: 2
  volumeClaimTemplates:
    - apiVersion: v1
      kind: PersistentVolumeClaim
      metadata:
        name: pvc-disk
        namespace: default
        annotations:
          everest.io/disk-volume-type: SAS    # EVS disk type.
          everest.io/crypt-key-id: <your_key_id>    # (Optional) Encryption key
ID. Mandatory for an encrypted disk.
          everest.io/enterprise-project-id: <your_project_id>  # (Optional)
Enterprise project ID. If an enterprise project is specified, use the same
enterprise project when creating a PVC. Otherwise, the PVC cannot be bound to a
PV.
        labels:
          failure-domain.beta.kubernetes.io/region: <your_region>   # Region of
the node where the application is to be deployed.
          failure-domain.beta.kubernetes.io/zone: <your_zone>       # AZ of the
node where the application is to be deployed.
      spec:
        accessModes:
          - ReadWriteOnce               # The value must be ReadWriteOnce for EVS
disks.
        resources:
          requests:
            storage: 10Gi             # EVS disk capacity, ranging from 1 to
32768.
        storageClassName: csi-disk    # Storage class type for EVS disks.
---
apiVersion: v1
kind: Service
metadata:
  name: statefulset-evs   # Headless Service name.
  namespace: default
  labels:
    app: statefulset-evs
```



```
spec:
  selector:
    app: statefulset-evs
  clusterIP: None
  ports:
    - name: statefulset-evs
      targetPort: 80
      nodePort: 0
      port: 80
      protocol: TCP
  type: ClusterIP
```

Tabela 8-13 Parâmetros principais

Parâmetro	Obrigatório	Descrição
failure-domain.beta.kubernetes.io/region	Sim	Região onde o cluster está localizado. Para obter detalhes sobre o valor da **region**, consulte **Regiões e pontos de extremidade**.
failure-domain.beta.kubernetes.io/zone	Sim	AZ onde o volume do EVS é criado. Deve ser a mesma que a AZ planejada para a carga de trabalho. Para obter detalhes sobre o valor da **zone**, consulte **Regiões e pontos de extremidade**.
everest.io/disk-volume-type	Sim	Tipos de disco EVS. Todas as letras estão em maiúsculas. ● **SAS**: I/O alta ● **SSD**: I/O ultra-alta ● **GPSSD**: SSD de uso geral ● **ESSD**: SSD extremo
everest.io/crypt-key-id	Não	Obrigatório quando o disco EVS é criptografado. Insira o ID da chave de criptografia selecionada durante a criação do disco EVS. Para obter o ID da chave de criptografia, faça logon no **Cloud Server Console**. No painel de navegação, escolha **Elastic Volume Service** > **Disks**. Clique no nome do disco EVS de destino para acessar sua página de detalhes. Na página de guia **Summary**, copie o valor de **KMS Key ID** na área **Configuration Information**.

Parâmetro	Obrigatório	Descrição
everest.io/enterprise-project-id	Não	Opcional. ID do projeto empresarial do disco EVS. Se um projeto empresarial for especificado, use o mesmo projeto empresarial ao criar uma PVC. Caso contrário, a PVC não pode ser vinculada a um PV. Para obter o código do projeto empresarial, efetue logon no **Cloud Server Console**. No painel de navegação, escolha **Elastic Volume Service** > **Disks**. Clique no nome do disco EVS de destino para acessar sua página de detalhes. Na página de guia **Summary**, clique no projeto empresarial em **Management Information** para acessar o console do projeto empresarial. Copie o ID correspondente para obter o ID do projeto corporativo ao qual o disco EVS pertence.
storage	Sim	Capacidade de PVC solicitada, em Gi. O valor varia de **1** a **32768**.
storageClassName	Sim	O nome da classe de armazenamento para discos EVS é **csi-disk**.

Passo 3 Execute o seguinte comando para criar uma carga de trabalho na qual o volume do EVS está montado:

```
kubectl apply -f statefulset-evs.yaml
```

Depois que a carga de trabalho for criada, os dados no diretório de montagem do contêiner serão armazenados persistentemente. Verifique o armazenamento referindo-se a **Verificar a persistência dos dados**.

----Fim

Verificar a persistência dos dados

Passo 1 Visualize a aplicação implementado e os arquivos de volume do EVS.

1. Execute o seguinte comando para exibir o pod criado:
   ```
   kubectl get pod | grep statefulset-evs
   ```
   Saída esperada:
   ```
   statefulset-evs-0          1/1     Running   0             45s
   statefulset-evs-1          1/1     Running   0             28s
   ```
2. Execute o seguinte comando para verificar se o volume do EVS foi montado no caminho **/data**:
   ```
   kubectl exec statefulset-evs-0 -- df | grep data
   ```
   Saída esperada:
   ```
   /dev/sdd              10255636     36888  10202364   0% /data
   ```
3. Execute o seguinte comando para exibir os arquivos no caminho **/data**:
   ```
   kubectl exec statefulset-evs-0 -- ls /data
   ```
   Saída esperada:
   ```
   lost+found
   ```


Passo 2 Execute o seguinte comando para criar um arquivo chamado **static** no caminho **/data**:

```
kubectl exec statefulset-evs-0 --  touch /data/static
```

Passo 3 Execute o seguinte comando para exibir os arquivos no caminho **/data**:

```
kubectl exec statefulset-evs-0 -- ls /data
```

Saída esperada:

```
lost+found
static
```

Passo 4 Execute o seguinte comando para excluir o pod chamado **web-evs-auto-0**:

```
kubectl delete pod statefulset-evs-0
```

Saída esperada:

```
pod "statefulset-evs-0" deleted
```

Passo 5 Após a exclusão, o controlador de StatefulSet cria automaticamente uma réplica com o mesmo nome. Execute o seguinte comando para verificar se os arquivos no caminho **/data** foram modificados:

```
kubectl exec statefulset-evs-0 -- ls /data
```

Saída esperada:

```
lost+found
static
```

Se o arquivo **static** ainda existir, os dados no volume de EVS podem ser armazenados persistentemente.

----Fim

Operações relacionadas

Você também pode executar as operações listadas em **Tabela 8-14**.

Tabela 8-14 Operações relacionadas

Operação	Descrição	Procedimento
Expandir a capacidade de um disco EVS	Expanda rapidamente a capacidade de um disco EVS montado no console do CCE. Somente a capacidade de discos EVS de pagamento por uso pode ser expandida no console do CCE. Para expandir a capacidade de discos EVS anuais/mensais, clique no nome do volume para ir para o console do EVS.	1. Escolha **Storage** no painel de navegação e clique na guia **PersistentVolumeClaims (PVCs)**. Clique em **More** na coluna **Operation** da PVC de destino e selecione **Scale-out**. 2. Insira a capacidade a ser adicionada e clique em **OK**.

Operação	Descrição	Procedimento
Visualizar eventos	Você pode visualizar nomes de eventos, tipos de eventos, número de ocorrências, eventos do Kubernetes, horário da primeira ocorrência e horário da última ocorrência da PVC ou PV.	1. Escolha **Storage** no painel de navegação e clique na guia **PersistentVolumeClaims (PVCs)** ou **PersistentVolumes (PVs)**. 2. Clique em **View Events** na coluna **Operation** da PVC ou do PV de destino para exibir os eventos gerados em uma hora (os dados do evento são mantidos por uma hora).
Exibir um arquivo YAML	Você pode visualizar, copiar e fazer download dos arquivos YAML de uma PVC ou um PV.	1. Escolha **Storage** no painel de navegação e clique na guia **PersistentVolumeClaims (PVCs)** ou **PersistentVolumes (PVs)**. 2. Clique em **View YAML** na coluna **Operation** da PVC ou PV de destino para exibir ou fazer download do YAML.

8.3.5 Snapshots e backups

O CCE trabalha com o EVS para suportar snapshots. Um snapshot é uma cópia ou imagem completa dos dados do disco EVS em um determinado ponto do tempo, o que é de grande ajuda para a DR de dados.

Você pode criar snapshots para salvar rapidamente os dados do disco em um determinado momento. Além disso, você pode usar snapshots para criar novos discos para que os discos criados contenham os dados do snapshot no início.

Precauções

- A função snapshot está disponível **apenas para clusters v1.15 ou posterior** e requer o complemento everest baseado em CSI.
- O subtipo (I/O comum, I/O alta ou I/O ultra-alta), modo de disco (SCSI ou VBD), criptografia de dados, estado de partilha, e a capacidade de um disco EVS criado a partir de um snapshot deve ser a mesma do disco associado ao snapshot. Esses atributos não podem ser modificados após serem consultados ou definidos.
- Snapshots podem ser criados apenas para discos EVS disponíveis ou em uso, e um máximo de sete snapshots podem ser criados para um único disco EVS.
- Snapshots só podem ser criados para PVCs criadas usando a classe de armazenamento (cujo nome começa com csi) fornecida pelo complemento everest. Não é possível criar instantâneos para PVCs criadas usando a classe de armazenamento Flexvolume cujo nome é ssd, sas ou sata.
- Os dados de snapshot de discos criptografados são armazenados criptografados, e os de discos não criptografados são armazenados não criptografados.

Cenários

O recurso de snapshot ajuda a atender às suas necessidades a seguir:

- **Backup de dados de rotina**

  Você pode criar snapshots para discos em tempo hábil e usar snapshots para recuperar seus dados no caso de perda de dados ou inconsistência de dados ocorrer devido a operações incorretas, vírus ou ataques.

- **Rápida restauração de dados**

  Você pode criar um snapshot ou vários snapshots antes de uma atualização de software de aplicação ou de uma migração de dados de serviço. Se ocorrer uma exceção durante a atualização ou migração, os dados do serviço poderão ser restaurados rapidamente para o ponto de tempo em que o instantâneo foi criado.

  Por exemplo, ocorreu uma falha no disco do sistema A do ECS A e, portanto, o ECS A não pode ser iniciado. Como o disco A do sistema já está com defeito, os dados no disco A do sistema não podem ser restaurados revertendo snapshots. Nesse caso, você pode usar um snapshot existente do disco do sistema A para criar o disco B do EVS e anexá-lo ao ECS B que esteja sendo executado corretamente. Em seguida, o ECS B pode ler dados do disco do sistema A usando o disco B do EVS.

  NOTA

  O recurso de snapshot fornecido pelo CCE é o mesmo que a função de snapshot da CSI fornecida pela comunidade do Kubernetes. Os discos EVS podem ser criados apenas com base em snapshots, e os snapshots não podem ser revertidos para discos EVS de origem.

- **Rápida implementação de vários serviços**

  Você pode usar um snapshot para criar vários discos EVS contendo os mesmos dados iniciais, e esses discos podem ser usados como recursos de dados para vários serviços, por exemplo, mineração de dados, consulta de relatórios e desenvolvimento e teste. Esse método protege os dados iniciais e cria discos rapidamente, atendendo aos requisitos de dados de serviço diversificados.

Criação de um snapshot

Usar o console do CCE

Passo 1 Efetue logon no console do CCE.

Passo 2 Clique no nome do cluster para acessar o console do cluster. Escolha **Storage** no painel de navegação e clique na guia **Snapshots and Backups**.

Passo 3 Clique em **Create Snapshot** no canto superior direito. Na caixa de diálogo exibida, defina os parâmetros relacionados.

- **Snapshot Name**: insira um nome de snapshot.
- **Storage**: selecione uma PVC do EVS.

Passo 4 Clique em **Create**.

----Fim

Usar YAML

```
kind: VolumeSnapshot
apiVersion: snapshot.storage.k8s.io/v1beta1
metadata:
  finalizers:
    - snapshot.storage.kubernetes.io/volumesnapshot-as-source-protection
    - snapshot.storage.kubernetes.io/volumesnapshot-bound-protection
  name: cce-disksnap-test   # Snapshot name
  namespace: default
```



```
spec:
  source:
    persistentVolumeClaimName: pvc-evs-test     # PVC name. Only an EVS PVC can
be selected.
  volumeSnapshotClassName: csi-disk-snapclass
```

Usar um snapshot para criar uma PVC

O tipo de disco, a configuração de criptografia e o modo de disco da PVC do EVS criado são consistentes com os do disco EVS de origem do snapshot.

Usar o console do CCE

Passo 1 Efetue logon no console do CCE.

Passo 2 Clique no nome do cluster para acessar o console do cluster. Escolha **Storage** no painel de navegação e clique na guia **Snapshots and Backups**.

Passo 3 Localize o snapshot que deseja usar para criar uma PVC, clique em **Create PVC** e configure os parâmetros de PVC na caixa de diálogo exibida.

- **PVC Name**: insira um nome da PVC.
- **Resource Tag**: as tags de recurso podem ser adicionadas para classificar recursos, o que é suportado apenas quando a versão de everest no cluster é 2.1.39 ou posterior.

  Você pode criar **tags predefinidas** no console do TMS. As tags predefinidas estão disponíveis para todos os recursos que suportam tags. Você pode usar tags predefinidas para melhorar a criação de tag e a eficiência da migração de recursos. Para obter detalhes, consulte **Criação de tags predefinidas**.

  O CCE cria automaticamente as tags de sistema **CCE-Cluster-ID**=*{Cluster ID}*, **CCE-Cluster-Name**=*{Cluster name}* e **CCE-Namespace**=*{Namespace name}*. Essas tags não podem ser modificadas.

Passo 4 Clique em **Create**.

----Fim

Usar YAML

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-test
  namespace: default
  annotations:
    everest.io/disk-volume-type: SSD     # EVS disk type, which must be the same
as that of the snapshot's source EVS disk.
  labels:
    failure-domain.beta.kubernetes.io/region: <your_region>   # Replace the
region with the one where the EVS disk is located.
    failure-domain.beta.kubernetes.io/zone: <your_zone>       # Replace the AZ
with the one where the EVS disk is located.
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
  storageClassName: csi-disk
  dataSource:
    name: cce-disksnap-test             # Snapshot name
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
```


8.4 Scalable File Service

8.4.1 Visão geral

Introdução

O CCE permite que você monte um volume criado a partir de um sistema de arquivos do Scalable File Service (SFS) em um contêiner para armazenar dados persistentemente. Os volumes do SFS são comumente usados em cenários de ReadWriteMany para expansão de grande capacidade e serviços de custo, como processamento de mídia, gerenciamento de conteúdo, análise de Big Data e análise de processos de carga de trabalho. Para serviços com grandes volumes de arquivos pequenos, os sistemas de arquivos do SFS Turbo são recomendados.

Expansível para petabytes, o SFS oferece armazenamento de arquivos compartilhados totalmente hospedado, altamente disponível e estável para lidar com aplicações com uso intenso de dados e largura de banda

- **Protocolos de arquivo padrão**: você pode montar sistemas de arquivos como volumes em servidores, o mesmo que usar diretórios locais.
- **Compartilhamento de dados**: o mesmo sistema de arquivos pode ser montado em vários servidores, para que os dados possam ser compartilhados.
- **Rede privada**: os usuários podem acessar dados apenas em redes privadas de data centers.
- **Capacidade e desempenho**: a capacidade de um único sistema de arquivos é alta (nível PB) e o desempenho é excelente (latência de I/O de leitura/gravação em nível ms).
- **Casos de uso**: as Implementações/StatefulSets no modo ReadWriteMany e tarefas criadas para computação de alto desempenho (HPC), processamento de mídia, gerenciamento de conteúdo, serviços Web, análise de Big Data e análise de processos de carga de trabalho

Desempenho

O CCE oferece suporte aos sistemas de arquivos do SFS Capacity-Oriented e SFS 3.0 Capacity-Oriented. Para obter detalhes sobre os tipos de sistema de arquivos, consulte **Tipo de sistema de arquivos**.

NOTA

- Os sistemas de arquivos do SFS Capacity-Oriented estão esgotados e não podem ser criados no console do CCE. Você ainda pode criar PVs para sistemas de arquivos do SFS Capacity-Oriented existentes usando **kubectl**.

Tabela 8-15 Desempenho

Parâmetro	SFS Capacity-Oriented
Largura de banda máxima	2 GB/s
IOPS máximo	2.000

Parâmetro	SFS Capacity-Oriented
Latência	3–20 ms
Capacidade máxima	4 PB

Cenários de aplicações

O SFS oferece suporte aos seguintes modos de montagem com base em cenários de aplicações:

- **Uso de um sistema de arquivos do SFS existente por meio de um PV estático**: modo de criação estática, no qual você usa um volume do SFS existente para criar um PV e, em seguida, monta o armazenamento na carga de trabalho por meio de uma PVC. Esse modo se aplica a cenários em que o armazenamento subjacente está disponível ou é cobrado anualmente/mensalmente.
- **Uso de um sistema de arquivos do SFS através de um PV dinâmico**: modo de criação dinâmica, onde você não precisa criar volumes do SFS antecipadamente. Em vez disso, especifique uma StorageClass durante a criação da PVC e um volume do SFS e um PV serão criados automaticamente. Esse modo se aplica a cenários em que nenhum armazenamento subjacente está disponível.

Cobrança

- Para volumes do SFS **criados automaticamente** usando StorageClass, o modo de faturamento padrão é **Pay-per-use**, indicando que é cobrado com base na capacidade de armazenamento usada e na duração. Para obter detalhes sobre a definição de preço do SFS, consulte **Cobrança**.
- Se você quiser ser cobrado no modo anual/mensal, **use os volumes de arquivos do SFS existentes**.

8.4.2 Uso de um sistema de arquivos do SFS existente por meio de um PV estático

O SFS é um armazenamento conectado à rede (NAS) que fornece armazenamento de arquivos compartilhado, escalável e de alto desempenho. Aplica-se à expansão de grande capacidade e a serviços sensíveis aos custos. Esta seção descreve como usar um sistema de arquivos do SFS existente para criar estaticamente PVs e PVCs e implementar a persistência e o compartilhamento de dados em cargas de trabalho.

Pré-requisitos

- Você criou um cluster e instalou o complemento **Armazenamento do contêiner do CCE (Everest)** no cluster.
- Se você quiser criar um cluster usando comandos, use kubectl para se conectar ao cluster. Para mais detalhes, consulte **Conexão a um cluster usando o kubectl**.
- Você criou um sistema de arquivos do SFS que está na mesma VPC que o cluster.

Restrições

- Vários PVs podem usar o mesmo sistema de arquivos do SFS ou SFS Turbo com as seguintes restrições:
  - Não monte todos as PVCs/PVs que usam o mesmo sistema de arquivos do SFS ou SFS Turbo subjacente em um pod. Isso leva a uma falha de inicialização do pod porque nem todos as PVCs podem ser montados no pod devido aos mesmos valores de **volumeHandle** desses PVs.
  - Sugere-se que o parâmetro **persistentVolumeReclaimPolicy** nos PVs seja definido como **Retain**. Caso contrário, quando um PV for excluído, o volume subjacente associado poderá ser excluído. Neste caso, outros PVs associados ao mau funcionamento do volume subjacente.
  - Quando o volume subjacente é usado repetidamente, ative o isolamento e a proteção de ReadWriteMany na camada da aplicação para evitar a substituição e a perda de dados.

Usar um sistema de arquivos do SFS existente no console

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster.

Passo 2 Crie estaticamente uma PVC e PV.

1. Escolha **Storage** no painel de navegação e clique na guia **PersistentVolumeClaims (PVCs)**. Clique em **Create PVC** no canto superior direito. Na caixa de diálogo exibida, configure os parâmetros de PVC.

Parâmetro	Descrição
PVC Type	Neste exemplo, selecione **SFS**.
PVC Name	Digite o nome da PVC, que deve ser exclusivo no mesmo namespace.
Creation Method	– Se o armazenamento subjacente estiver disponível, crie um volume de armazenamento ou use um volume de armazenamento existente para criar estaticamente uma PVC com base em se um PV foi criado. – Se nenhum armazenamento subjacente estiver disponível, selecione **Dynamically provision**. Para mais detalhes, consulte **Uso de um sistema de arquivos do SFS através de um PV dinâmico**. Neste exemplo, selecione **Create new** para criar um PV e uma PVC ao mesmo tempo no console.
PV[a]	Selecione um PV existente no cluster. Crie um PV com antecedência. Para obter detalhes, consulte "Criação de um volume de armazenamento" em **Operações relacionadas**. Você não precisa especificar esse parâmetro neste exemplo.
SFS[b]	Clique em **Select SFS**. Na página exibida, selecione o sistema de arquivos do SFS que atende aos seus requisitos e clique em **OK**. **NOTA** Atualmente, somente SFS 3.0 Capacity-Oriented é suportado.

Parâmetro	Descrição
PV Name[b]	Digite o nome do PV, que deve ser exclusivo no mesmo cluster.
Access Mode[b]	Os volumes do SFS suportam apenas **ReadWriteMany** indicando que um volume de armazenamento pode ser montado em vários nós no modo de leitura/gravação. Para mais detalhes, consulte **Modos de acesso a volume**.
Reclaim Policy[b]	Você pode selecionar **Delete** ou **Retain** para especificar a política de recuperação do armazenamento subjacente quando a PVC é excluída. Para mais detalhes, consulte **Política de recuperação da PV**. **NOTA** Se vários PVs usarem o mesmo volume de armazenamento subjacente, use **Retain** para evitar a exclusão em cascata de volumes subjacentes.
Mount Options[b]	Insira os pares chave-valor do parâmetro de montagem. Para mais detalhes, consulte **Configuração de opções de montagem de volume do SFS**.

NOTA

a: o parâmetro está disponível quando **Creation Method** está definido como **Use existing**.

b: o parâmetro está disponível quando **Creation Method** está definido como **Create new**.

2. Clique em **Create** para criar uma PVC e um PV.

   Você pode escolher **Storage** no painel de navegação e exibir a PVC e o PV criados nas páginas de guia **PersistentVolumeClaims (PVCs)** e **PersistentVolumes (PVs)**, respectivamente.

Passo 3 Crie uma aplicação.

1. No painel de navegação à esquerda, clique em **Workloads**. No painel direito, clique na guia **Deployments**.
2. Clique em **Create Workload** no canto superior direito. Na página exibida, clique em **Data Storage** na área **Container Settings** e clique em **Add Volume** para selecionar **PVC**.

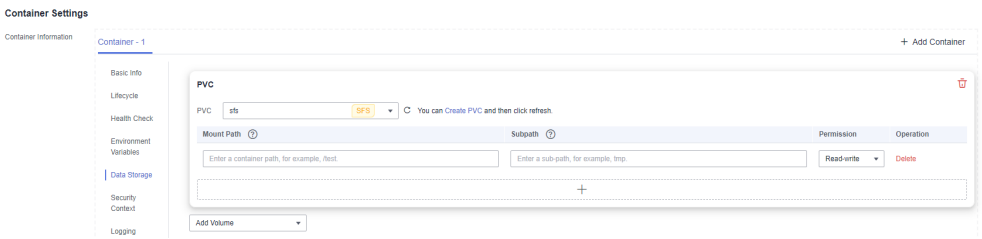
   Monte e use volumes de armazenamento, conforme mostrado na **Tabela 8-16**. Para obter detalhes sobre outros parâmetros, consulte **Cargas de trabalho**.

Tabela 8-16 Montagem de um volume de armazenamento

Parâmetro	Descrição
PVC	Selecione um volume do SFS existente.

Parâmetro	Descrição
Mount Path	Digite um caminho de montagem, por exemplo, **/tmp**. Este parâmetro indica o caminho do contêiner no qual um volume de dados será montado. Não monte o volume em um diretório do sistema como / ou **/var/run**. Caso contrário, os contêineres estarão com defeito. Monte o volume em um diretório vazio. Se o diretório não estiver vazio, verifique se não há arquivos que afetem a inicialização do contêiner. Caso contrário, os arquivos serão substituídos, causando falhas de inicialização do contêiner ou falhas de criação de carga de trabalho. **AVISO** Se um volume for montado em um diretório de alto risco, use uma conta com permissões mínimas para iniciar o contêiner. Caso contrário, arquivos de alto risco no host podem ser danificados.
Subpath	Digite um subcaminho, por exemplo, **tmp**, indicando que os dados no caminho de montagem do contêiner serão armazenados na pasta **tmp** do volume. Um subcaminho é usado para montar um volume local para que o mesmo volume de dados seja usado em um único pod. Se este parâmetro for deixado em branco, o caminho raiz é usado por padrão.
Permission	– **Read-only**: você só pode ler os dados nos volumes montados. – **Read/Write**: você pode modificar os volumes de dados montados no caminho. Os dados recém-gravados não serão migrados se o contêiner for migrado, o que pode causar perda de dados.

Neste exemplo, o disco é montado no caminho **/data** do contêiner. Os dados de contêiner gerados nesse caminho são armazenados no sistema de arquivos do SFS.



3. Após a configuração, clique em **Create Workload**.

   Depois que a carga de trabalho for criada, os dados no diretório de montagem do contêiner serão armazenados persistentemente. Verifique o armazenamento referindo-se a **Verificar a persistência e o compartilhamento de dados**.

----Fim

(kubectl) Usar um sistema de arquivos do SFS existente

Passo 1 Use o kubectl para se conectar ao cluster.

Passo 2 Crie um PV.

1. Crie o arquivo **pv-sfs.yaml**.

   SFS Capacity-Oriented:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  annotations:
    pv.kubernetes.io/provisioned-by: everest-csi-provisioner
    everest.io/reclaim-policy: retain-volume-only      # (Optional) The PV is
deleted while the underlying volume is retained.
  name: pv-sfs    # PV name.
spec:
  accessModes:
  - ReadWriteMany      # Access mode. The value must be ReadWriteMany for SFS.
  capacity:
    storage: 1Gi     # SFS volume capacity.
  csi:
    driver: disk.csi.everest.io   # Dependent storage driver for the mounting.
    fsType: nfs
    volumeHandle: <your_volume_id>   # SFS Capacity-Oriented volume ID.
    volumeAttributes:
      everest.io/share-export-location: <your_location>  # Shared path of the
SFS volume.
      storage.kubernetes.io/csiProvisionerIdentity: everest-csi-provisioner
  persistentVolumeReclaimPolicy: Retain    # Reclaim policy.
  storageClassName: csi-nas              # Storage class name. csi-nas
indicates that SFS Capacity-Oriented is used.
  mountOptions: []                        # Mount options.
```

Tabela 8-17 Parâmetros principais

Parâmetro	Obrigatório	Descrição
everest.io/reclaim-policy: retain-volume-only	Não	Opcional. Atualmente, apenas **retain-volume-only** é suportado. Este campo é válido somente quando a versão do everest for 1.2.9 ou posterior e a política de recuperação for **Delete**. Se a política de recuperação for **Delete** e o valor atual for **retain-volume-only**, o PV associado será excluído enquanto o volume de armazenamento subjacente for retido, quando uma PVC for excluída.
volumeHandle	Sim	– Se um volume orientado do SFS Capacity-Oriented for usado, insira o ID do volume. Faça logon no console, escolha **Service List** > **Storage** > **Scalable File Service** e selecione **SFS Turbo**. Na lista, clique no nome do sistema de arquivos do SFS de destino. Na página de detalhes, copie o conteúdo seguinte do **ID**.

Parâmetro	Obrigatório	Descrição
everest.io/share-export-location	Sim	Caminho compartilhado do sistema de arquivos. – Para um sistema de arquivos do SFS Capacity-Oriented, faça logon no console, escolha **Service List** > **Storage** > **Scalable File Service** e obtenha o caminho compartilhado da coluna **Mount Address**.

mountOptions	Sim	Opções de montagem.
		Se não for especificado, as seguintes configurações são usadas por padrão. Para mais detalhes, consulte **Configuração de opções de montagem de volume do SFS**.
		`mountOptions:`
		`- vers=3`
		`- timeo=600`
		`- nolock`
		`- hard`

persistentVolumeReclaimPolicy	Sim	Uma política de recuperação é suportada quando a versão do cluster é ou posterior a 1.19.10 e a versão do everest é ou posterior a 1.2.9. As políticas de recuperação **Delete** e **Retain** são suportadas. Para mais detalhes, consulte **Política de recuperação da PV**. Se vários PVs usarem o mesmo volume do SFS, use **Retain** para impedir que o volume subjacente seja excluído com um PV. **Delete**: – Se **everest.io/reclaim-policy** não for especificada, ambos os volumes de PV e SFS serão excluídos quando uma PVC for excluída. – Se **everest.io/reclaim-policy** estiver definida para **retain-volume-only**, quando uma PVC for excluída, o PV será excluído, mas os recursos de volumes do SFS serão retidos. **Retain**: quando uma PVC é excluída, o PV e os recursos de armazenamento subjacentes não são excluídos. Em vez disso, você deve excluir manualmente esses recursos. Depois disso, o PV está no estado **Released** e não pode ser vinculado à PVC novamente.
storage	Sim	Capacidade solicitada na PVC, em Gi. Para SFS, esse campo é usado apenas para verificação (não pode estar vazio ou **0**). Seu valor é fixado em **1**, e qualquer valor que você definir não terá efeito para sistemas de arquivos do SFS.

2. Execute o seguinte comando para criar um PV:
```
kubectl apply -f pv-sfs.yaml
```

Passo 3 Crie uma PVC.

1. Crie o arquivo **pvc-sfs.yaml**.
```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-sfs
  namespace: default
  annotations:
    volume.beta.kubernetes.io/storage-provisioner: everest-csi-provisioner
spec:
  accessModes:
  - ReadWriteMany               # The value must be ReadWriteMany for SFS.
  resources:
    requests:
      storage: 1Gi               # SFS volume capacity.
storageClassName: csi-nas # Storage class name, which must be the same as the
PV's storage class.
  volumeName: pv-sfs    # PV name.
```

Tabela 8-18 Parâmetros principais

Parâmetro	Obrigatório	Descrição
storage	Sim	Capacidade solicitada na PVC, em Gi. O valor deve ser o mesmo que o tamanho de armazenamento do PV existente.
volumeName	Sim	Nome do PV, que deve ser o mesmo que o nome do PV em **1**.

2. Execute o seguinte comando para criar uma PVC:
```
kubectl apply -f pvc-sfs.yaml
```

Passo 4 Crie uma aplicação.

1. Crie um arquivo chamado **web-demo.yaml**. Neste exemplo, o volume do SFS é montado no caminho **/data**.
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web-demo
  namespace: default
spec:
  replicas: 2
  selector:
    matchLabels:
      app: web-demo
  template:
    metadata:
      labels:
        app: web-demo
    spec:
      containers:
      - name: container-1
        image: nginx:latest
        volumeMounts:
        - name: pvc-sfs-volume    # Volume name, which must be the same as
the volume name in the volumes field.
```



```
          mountPath: /data  # Location where the storage volume is mounted.
      imagePullSecrets:
        - name: default-secret
      volumes:
        - name: pvc-sfs-volume    # Volume name, which can be customized.
          persistentVolumeClaim:
            claimName: pvc-sfs    # Name of the created PVC.
```

2. Execute o seguinte comando para criar uma carga de trabalho na qual o volume do SFS está montado:
   ```
   kubectl apply -f web-demo.yaml
   ```
   Depois que a carga de trabalho for criada, os dados no diretório de montagem do contêiner serão armazenados persistentemente. Verifique o armazenamento referindo-se a **Verificar a persistência e o compartilhamento de dados**.

----Fim

Verificar a persistência e o compartilhamento de dados

Passo 1 Exiba a aplicação implementada e os arquivos.

1. Execute o seguinte comando para exibir o pod criado:
   ```
   kubectl get pod | grep web-demo
   ```
   Saída esperada:
   ```
   web-demo-846b489584-mjhm9   1/1     Running   0             46s
   web-demo-846b489584-wvv5s   1/1     Running   0             46s
   ```
2. Execute os seguintes comandos em sequência para visualizar os arquivos no caminho **/data** dos pods:
   ```
   kubectl exec web-demo-846b489584-mjhm9 -- ls /data
   kubectl exec web-demo-846b489584-wvv5s -- ls /data
   ```
   Se nenhum resultado for retornado para ambos os pods, nenhum arquivo existirá no caminho **/data**.

Passo 2 Execute o seguinte comando para criar um arquivo chamado **static** no caminho **/data**:
```
kubectl exec web-demo-846b489584-mjhm9 --  touch /data/static
```

Passo 3 Execute o seguinte comando para exibir os arquivos no caminho **/data**:
```
kubectl exec web-demo-846b489584-mjhm9 -- ls /data
```
Saída esperada:
```
static
```

Passo 4 **Verifique a persistência dos dados.**

1. Execute o seguinte comando para excluir o pod chamado **web-demo-846b489584-mjhm9**:
   ```
   kubectl delete pod web-demo-846b489584-mjhm9
   ```
   Saída esperada:
   ```
   pod "web-demo-846b489584-mjhm9" deleted
   ```
   Após a eliminação, o controlador de Implementação cria automaticamente uma réplica.
2. Execute o seguinte comando para exibir o pod criado:
   ```
   kubectl get pod | grep web-demo
   ```
   A saída esperada é a seguinte, na qual **web-demo-846b489584-d4d4j** é o pod recém-criado:
   ```
   web-demo-846b489584-d4d4j   1/1     Running   0             110s
   web-demo-846b489584-wvv5s    1/1     Running   0             7m50s
   ```
3. Execute o seguinte comando para verificar se os arquivos no caminho **/data** do pod novo foram modificados:


```
kubectl exec web-demo-846b489584-d4d4j -- ls /data
```

Saída esperada:

```
static
```

Se o arquivo **static** ainda existir, os dados podem ser armazenados persistentemente.

Passo 5 Verifique o compartilhamento de dados.

1. Execute o seguinte comando para exibir o pod criado:
   ```
   kubectl get pod | grep web-demo
   ```
   Saída esperada:
   ```
   web-demo-846b489584-d4d4j   1/1     Running   0             7m
   web-demo-846b489584-wvv5s   1/1     Running   0             13m
   ```
2. Execute o seguinte comando para criar um arquivo chamado **share** no caminho **/data** de qualquer pod: Neste exemplo, selecione o pod chamado **web-demo-846b489584-d4d4j**.
   ```
   kubectl exec web-demo-846b489584-d4d4j --  touch /data/share
   ```
   Verifique os arquivos no caminho **/data** do pod.
   ```
   kubectl exec web-demo-846b489584-d4d4j -- ls /data
   ```
   Saída esperada:
   ```
   share
   static
   ```
3. Verifique se o arquivo **share** existe no caminho **/data** de outro pod (**web-demo-846b489584-wvv5s**) também para verificar o compartilhamento de dados.
   ```
   kubectl exec web-demo-846b489584-wvv5s -- ls /data
   ```
   Saída esperada:
   ```
   share
   static
   ```
   Depois de criar um arquivo no caminho **/data** de um pod, se o arquivo também for criado no caminho **/data** do outro pod, os dois pods compartilharão o mesmo volume.

----Fim

Operações relacionadas

Você também pode executar as operações listadas em **Tabela 8-19**.

Tabela 8-19 Operações relacionadas

Operação	Descrição	Procedimento
Criar um volume de armazenamento (PV)	Crie um PV no console do CCE.	1. Escolha **Storage** no painel de navegação e clique na guia **PersistentVolumes (PVs)**. Clique em **Create Volume** no canto superior direito. Na caixa de diálogo exibida, configure os parâmetros. ● **Volume Type**: selecione **SFS**. ● **SFS**: clique em **Select SFS**. Na página exibida, selecione o sistema de arquivos do SFS que atende aos seus requisitos e clique em **OK**. ● PV Name: insira o nome do PV. O nome do PV deve ser exclusivo no mesmo cluster. ● **Access Mode**: os volumes do SFS suportam apenas **ReadWriteMany**, indicando que um volume de armazenamento pode ser montado em vários nós no modo de leitura/gravação. Para mais detalhes, consulte **Modos de acesso a volume**. ● **Reclaim Policy**: **Delete** ou **Retain**. Para mais detalhes, consulte **Política de recuperação da PV**. **NOTA** Se vários PVs usarem o mesmo volume de armazenamento subjacente, use **Retain** para evitar a exclusão em cascata de volumes subjacentes. ● **Mount Options**: insira os pares chave-valor do parâmetro de montagem. Para mais detalhes, consulte **Configuração de opções de montagem de volume do SFS**. 2. Clique em **Create**.
Visualizar eventos	Você pode visualizar nomes de eventos, tipos de eventos, número de ocorrências, eventos do Kubernetes, horário da primeira ocorrência e horário da última ocorrência da PVC ou PV.	1. Escolha **Storage** no painel de navegação e clique na guia **PersistentVolumeClaims (PVCs)** ou **PersistentVolumes (PVs)**. 2. Clique em **View Events** na coluna **Operation** da PVC ou do PV de destino para exibir os eventos gerados em uma hora (os dados do evento são mantidos por uma hora).

Operação	Descrição	Procedimento
Exibir um arquivo YAML	Você pode visualizar, copiar e fazer download dos arquivos YAML de uma PVC ou um PV.	1. Escolha **Storage** no painel de navegação e clique na guia **PersistentVolumeClaims (PVCs)** ou **PersistentVolumes (PVs)**. 2. Clique em **View YAML** na coluna **Operation** da PVC ou do PV de destino para exibir ou fazer download do YAML.

8.4.3 Uso de um sistema de arquivos do SFS através de um PV dinâmico

Esta seção descreve como usar classes de armazenamento para criar PVs e PVCs dinamicamente e implementar a persistência e o compartilhamento de dados em cargas de trabalho.

Automatically Creating an SFS File System on the Console

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster.

Passo 2 Crie dinamicamente uma PVC e um PV.

1. Escolha **Storage** no painel de navegação e clique na guia **PersistentVolumeClaims (PVCs)**. Clique em **Create PVC** no canto superior direito. Na caixa de diálogo exibida, configure os parâmetros de PVC.

Parâmetro	Descrição
PVC Type	Neste exemplo, selecione **SFS**.
PVC Name	Digite o nome da PVC, que deve ser exclusivo no mesmo namespace.
Creation Method	– Se nenhum armazenamento subjacente estiver disponível, selecione **Dynamically provision** para criar uma PVC, um PV e armazenamento subjacente no console no modo de cascata. – Se o armazenamento subjacente estiver disponível, crie um volume de armazenamento ou use um volume de armazenamento existente para criar estaticamente uma PVC com base em se um PV foi criado. Para mais detalhes, consulte **Uso de um sistema de arquivos do SFS existente por meio de um PV estático**. Neste exemplo, selecione **Dynamically provision**.
Storage Classes	A classe de armazenamento para volumes do SFS é **csi-sfs**.

Parâmetro	Descrição
Access Mode	Os volumes do SFS suportam apenas **ReadWriteMany** indicando que um volume de armazenamento pode ser montado em vários nós no modo leitura/gravação. Para mais detalhes, consulte **Modos de acesso a volume**.

2. Clique em **Create** para criar uma PVC e um PV.

   Você pode escolher **Storage** no painel de navegação e exibir a PVC e o PV criados nas páginas de guia **PersistentVolumeClaims (PVCs)** e **PersistentVolumes (PVs)**, respectivamente.

Passo 3 Crie uma aplicação.

1. No painel de navegação à esquerda, clique em **Workloads**. No painel direito, clique na guia **Deployments**.
2. Clique em **Create Workload** no canto superior direito. Na página exibida, clique em **Data Storage** na área **Container Settings** e clique em **Add Volume** para selecionar **PVC**.

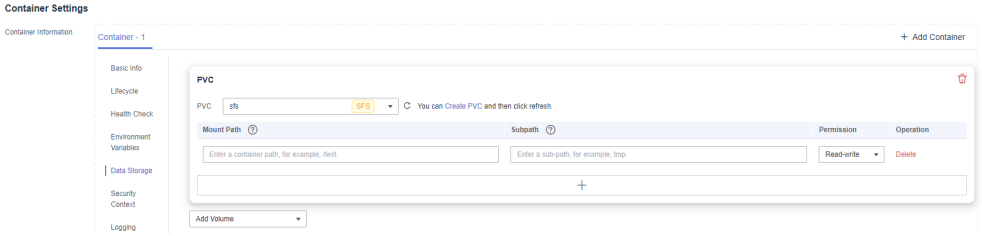
   Monte e use volumes de armazenamento, conforme mostrado na **Tabela 8-20**. Para obter detalhes sobre outros parâmetros, consulte **Cargas de trabalho**.

Tabela 8-20 Montagem de um volume de armazenamento

Parâmetro	Descrição
PVC	Selecione um volume do SFS existente.
Mount Path	Digite um caminho de montagem, por exemplo, **/tmp**. Este parâmetro indica o caminho do contêiner no qual um volume de dados será montado. Não monte o volume em um diretório do sistema como / ou **/var/run**. Caso contrário, os contêineres estarão com defeito. Monte o volume em um diretório vazio. Se o diretório não estiver vazio, verifique se não há arquivos que afetem a inicialização do contêiner. Caso contrário, os arquivos serão substituídos, causando falhas de inicialização do contêiner ou falhas de criação de carga de trabalho. **AVISO** Se um volume for montado em um diretório de alto risco, use uma conta com permissões mínimas para iniciar o contêiner. Caso contrário, arquivos de alto risco no host podem ser danificados.
Subpath	Digite um subcaminho, por exemplo, **tmp**, indicando que os dados no caminho de montagem do contêiner serão armazenados na pasta **tmp** do volume. Um subcaminho é usado para montar um volume local para que o mesmo volume de dados seja usado em um único pod. Se este parâmetro for deixado em branco, o caminho raiz é usado por padrão.

Parâmetro	Descrição
Permission	– **Read-only**: você só pode ler os dados nos volumes montados. – **Read/Write**: você pode modificar os volumes de dados montados no caminho. Os dados recém-gravados não serão migrados se o contêiner for migrado, o que pode causar perda de dados.

Neste exemplo, o disco é montado no caminho **/data** do contêiner. Os dados de contêiner gerados nesse caminho são armazenados no sistema de arquivos do SFS.



3. Após a configuração, clique em **Create Workload**.

   Depois que a carga de trabalho for criada, os dados no diretório de montagem do contêiner serão armazenados persistentemente. Verifique o armazenamento referindo-se a **Verificar a persistência e o compartilhamento de dados**.

----Fim

(kubectl) Criar automaticamente um sistema de arquivos do SFS

Passo 1 Use o kubectl para se conectar ao cluster.

Passo 2 Use **StorageClass** para criar dinamicamente uma PVC e um PV.

1. Crie o arquivo **pvc-sfs-auto.yaml**.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-sfs-auto
  namespace: default
  annotations:
    everest.io/crypt-key-id: <your_key_id>      # (Optional) ID of the key
for encrypting file systems
    everest.io/crypt-alias: sfs/default         # (Optional) Key name.
Mandatory for encrypting volumes.
    everest.io/crypt-domain-id: <your_domain_id>   # (Optional) ID of the
tenant to which an encrypted volume belongs. Mandatory for encrypting volumes.
spec:
  accessModes:
    - ReadWriteMany             # The value must be ReadWriteMany for SFS.
  resources:
    requests:
      storage: 1Gi              # SFS volume capacity.
  storageClassName: csi-nas     # The storage class type is SFS.
```


Tabela 8-21 Parâmetros principais

Parâmetro	Obrigatório	Descrição
storage	Sim	Capacidade solicitada na PVC, em Gi. Para buckets do SFS, esse campo é usado apenas para verificação (não pode estar vazio ou **0**). Seu valor é fixado em **1**, e qualquer valor que você definir não terá efeito para sistemas de arquivos do SFS.
everest.io/crypt-key-id	Não	Este parâmetro é obrigatório quando um sistema do SFS é criptografado. Digite o ID da chave de criptografia selecionada durante a criação do sistema do SFS. Você pode usar uma chave personalizada ou a chave padrão chamada **sfs/default**. Para obter um ID da chave, faça logon no console do DEW, localize a chave a ser criptografada e copie o ID da chave.
everest.io/crypt-alias	Não	Nome da chave, que é obrigatório quando você cria um volume criptografado. Para obter um nome de chave, faça logon no console do DEW, localize a chave a ser criptografada e copie o nome da chave.
everest.io/crypt-domain-id	Não	ID do locatário ao qual o volume criptografado pertence. Este parâmetro é obrigatório para criar um volume encriptado. Para obter um ID de locatário, passe o cursor sobre o nome de usuário no canto superior direito do console do ECS, escolha **My Credentials** e copie o ID de conta.

2. Execute o seguinte comando para criar uma PVC:
```
kubectl apply -f pvc-sfs-auto.yaml
```

Passo 3 Crie uma aplicação.

1. Crie um arquivo chamado **web-demo.yaml**. Neste exemplo, o volume do SFS é montado no caminho **/data**.
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web-demo
  namespace: default
spec:
  replicas: 2
  selector:
    matchLabels:
      app: web-demo
  template:
    metadata:
      labels:
```



```
        app: web-demo
    spec:
      containers:
      - name: container-1
        image: nginx:latest
        volumeMounts:
        - name: pvc-sfs-volume    # Volume name, which must be the same as
the volume name in the volumes field.
          mountPath: /data  # Location where the storage volume is mounted.
      imagePullSecrets:
        - name: default-secret
      volumes:
        - name: pvc-sfs-volume    # Volume name, which can be customized.
          persistentVolumeClaim:
            claimName: pvc-sfs-auto    # Name of the created PVC.
```

2. Execute o seguinte comando para criar uma carga de trabalho na qual o volume do SFS está montado:
   ```
   kubectl apply -f web-demo.yaml
   ```
   Depois que a carga de trabalho for criada, os dados no diretório de montagem do contêiner serão armazenados persistentemente. Verifique o armazenamento referindo-se a **Verificar a persistência e o compartilhamento de dados**.

----Fim

Verificar a persistência e o compartilhamento de dados

Passo 1 Exiba a aplicação implementada e os arquivos.

1. Execute o seguinte comando para exibir o pod criado:
   ```
   kubectl get pod | grep web-demo
   ```
   Saída esperada:
   ```
   web-demo-846b489584-mjhm9   1/1     Running   0             46s
   web-demo-846b489584-wvv5s   1/1     Running   0             46s
   ```
2. Execute os seguintes comandos em sequência para visualizar os arquivos no caminho **/data** dos pods:
   ```
   kubectl exec web-demo-846b489584-mjhm9 -- ls /data
   kubectl exec web-demo-846b489584-wvv5s -- ls /data
   ```
   Se nenhum resultado for retornado para ambos os pods, nenhum arquivo existirá no caminho **/data**.

Passo 2 Execute o seguinte comando para criar um arquivo chamado **static** no caminho **/data**:
```
kubectl exec web-demo-846b489584-mjhm9 --  touch /data/static
```

Passo 3 Execute o seguinte comando para exibir os arquivos no caminho **/data**:
```
kubectl exec web-demo-846b489584-mjhm9 -- ls /data
```
Saída esperada:
```
static
```

Passo 4 **Verifique a persistência dos dados.**

1. Execute o seguinte comando para excluir o pod chamado **web-demo-846b489584-mjhm9**:
   ```
   kubectl delete pod web-demo-846b489584-mjhm9
   ```
   Saída esperada:
   ```
   pod "web-demo-846b489584-mjhm9" deleted
   ```
   Após a eliminação, o controlador de Implementação cria automaticamente uma réplica.
2. Execute o seguinte comando para exibir o pod criado:


```
kubectl get pod | grep web-demo
```

A saída esperada é a seguinte, na qual **web-demo-846b489584-d4d4j** é o pod recém-criado:

```
web-demo-846b489584-d4d4j   1/1     Running   0             110s
web-demo-846b489584-wvv5s    1/1     Running   0             7m50s
```

3. Execute o seguinte comando para verificar se os arquivos no caminho **/data** do pod novo foram modificados:

```
kubectl exec web-demo-846b489584-d4d4j -- ls /data
```

Saída esperada:

```
static
```

Se o arquivo **static** ainda existir, os dados podem ser armazenados persistentemente.

Passo 5 Verifique o compartilhamento de dados.

1. Execute o seguinte comando para exibir o pod criado:

```
kubectl get pod | grep web-demo
```

Saída esperada:

```
web-demo-846b489584-d4d4j   1/1     Running   0             7m
web-demo-846b489584-wvv5s   1/1     Running   0             13m
```

2. Execute o seguinte comando para criar um arquivo chamado **share** no caminho **/data** de qualquer pod: Neste exemplo, selecione o pod chamado **web-demo-846b489584-d4d4j**.

```
kubectl exec web-demo-846b489584-d4d4j --  touch /data/share
```

Verifique os arquivos no caminho **/data** do pod.

```
kubectl exec web-demo-846b489584-d4d4j -- ls /data
```

Saída esperada:

```
share
static
```

3. Verifique se o arquivo **share** existe no caminho **/data** de outro pod (**web-demo-846b489584-wvv5s**) também para verificar o compartilhamento de dados.

```
kubectl exec web-demo-846b489584-wvv5s -- ls /data
```

Saída esperada:

```
share
static
```

Depois de criar um arquivo no caminho **/data** de um pod, se o arquivo também for criado no caminho **/data** do outro pod, os dois pods compartilharão o mesmo volume.

----Fim

Operações relacionadas

Você também pode executar as operações listadas em **Tabela 8-22**.

Tabela 8-22 Operações relacionadas

Operação	Descrição	Procedimento
Visualização de eventos	Você pode visualizar nomes de eventos, tipos de eventos, número de ocorrências, eventos do Kubernetes, horário da primeira ocorrência e horário da última ocorrência da PVC ou do PV.	1. Escolha **Storage** no painel de navegação e clique na guia **PersistentVolumeClaims (PVCs)** ou **PersistentVolumes (PVs)**. 2. Clique em **View Events** na coluna **Operation** da PVC ou do PV de destino para exibir os eventos gerados em uma hora (os dados do evento são mantidos por uma hora).
Exibir um arquivo YAML	Você pode visualizar, copiar e fazer download dos arquivos YAML de uma PVC ou um PV.	1. Escolha **Storage** no painel de navegação e clique na guia **PersistentVolumeClaims (PVCs)** ou **PersistentVolumes (PVs)**. 2. Clique em **View YAML** na coluna **Operation** da PVC ou do PV de destino para exibir ou fazer download do YAML.

8.4.4 Configuração de opções de montagem de volume do SFS

Esta seção descreve como configurar as opções de montagem de volume do SFS. Você pode configurar opções de montagem em um PV e vincular o PV a uma PVC. Como alternativa, configure as opções de montagem em uma StorageClass e use a StorageClass para criar uma PVC. Dessa forma, os PVs podem ser criados dinamicamente e herdar opções de montagem configuradas na StorageClass por padrão.

Pré-requisitos

A versão do complemento **Armazenamento do contêiner do CCE (Everest)** deve ser **1.2.8 ou posterior**. Esse complemento identifica as opções de montagem e as transfere para os recursos de armazenamento subjacentes. As configurações de parâmetros terão efeito somente se os recursos de armazenamento subjacentes suportarem as opções especificadas.

Restrições

- As opções de montagem não podem ser configuradas para contêineres seguros.
- Devido às restrições do protocolo NFS, se um volume do SFS for montado em um nó por várias vezes, os parâmetros de montagem relacionados ao link (como **timeo**) terão efeito somente quando o volume do SFS for montado pela primeira vez. Por exemplo, se um volume do SFS for montado em vários pods em execução em um nó, os valores dos parâmetros de montagem configurados posteriormente não substituirão os valores de parâmetros existentes.

Opções de montagem de volume do SFS

O complemento everest no CCE predefine as opções descritas em **Tabela 8-23** para montar volumes do SFS.

Tabela 8-23 Opções de montagem de volume do SFS

Parâmetro	Valor	Descrição
keep-original-ownership	Deixe em branco.	Se deve reter a propriedade do ponto de montagem do arquivo. Se esta opção for usada, o complemento everest deve ser v1.2.63 ou v2.1.2 ou posterior. ● Por predefinição, esta opção não é adicionada e a propriedade do ponto de montagem é **root:root** quando o SFS é montado. ● Se essa opção for adicionada, a propriedade original do sistema de arquivos será mantida quando o SFS for montado.
vers	3	Versão do sistema de arquivos. No momento, apenas NFSv3 é suportada. Valor: **3**
nolock	Deixe em branco.	Se deseja bloquear arquivos no servidor usando o protocolo NLM. Se **nolock** for selecionado, o bloqueio é válido para aplicações em um host. Para aplicações em outro host, o bloqueio é inválido.
timeo	600	Tempo de espera antes do cliente de NFS retransmitir uma solicitação. A unidade é 0,1 segundo. Valor recomendado: **600**
hard/soft	Deixe em branco.	Modos de montagem. ● **hard**: se a solicitação NFS expirar, o cliente continuará reenviando a solicitação até que a solicitação seja bem-sucedida. ● **soft**: se uma solicitação NFS expirar, o cliente retornará um erro para o programa invocando. O valor padrão é **hard**.

Você pode definir outras opções de montagem, se necessário. Para obter detalhes, consulte **MMontagem de um sistema de arquivos NFS para ECSs (Linux)**.

Definir opções de montagem em um PV

Você pode usar o campo **mountOptions** para definir opções de montagem em um PV. As opções que você pode configurar em **mountOptions** estão listadas em **Opções de montagem de volume do SFS**.

Passo 1 Use o kubectl para se conectar ao cluster. Para mais detalhes, consulte **Conexão a um cluster usando o kubectl**.

Passo 2 Defina as opções de montagem em um PV. Exemplo:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  annotations:
    pv.kubernetes.io/provisioned-by: everest-csi-provisioner
    everest.io/reclaim-policy: retain-volume-only      # (Optional) The PV is
```



```
deleted while the underlying volume is retained.
  name: pv-sfs
spec:
  accessModes:
  - ReadWriteMany      # Access mode. The value must be ReadWriteMany for SFS.
  capacity:
    storage: 1Gi     # SFS volume capacity.
  csi:
    driver: disk.csi.everest.io   # Dependent storage driver for the mounting.
    fsType: nfs
    volumeHandle: <your_volume_id>   # ID of the SFS Capacity-Oriented volume.
    volumeAttributes:
      everest.io/share-export-location: <your_location>  # Shared path of the SFS
volume.
      storage.kubernetes.io/csiProvisionerIdentity: everest-csi-provisioner
  persistentVolumeReclaimPolicy: Retain    # Reclaim policy.
  storageClassName: csi-nas                # Storage class name.
  mountOptions:                            # Mount options.
  - vers=3
  - nolock
  - timeo=600
  - hard
```

Passo 3 Depois que um PV é criado, você pode criar uma PVC e vinculá-la ao PV e, em seguida, montar o PV ao contêiner na carga de trabalho. Para mais detalhes, consulte **Uso de um sistema de arquivos do SFS existente por meio de um PV estático**.

Passo 4 Verifique se as opções de montagem têm efeito.

Neste exemplo, a PVC é montada na carga de trabalho que usa a imagem **nginx:latest**. Você pode executar o comando **mount -l** para verificar se as opções de montagem têm efeito.

1. Visualize o pod no qual o volume do SFS foi montado. Neste exemplo, o nome da carga de trabalho é **web-sfs**.

   ```
   kubectl get pod | grep web-sfs
   ```

   Saída do comando:

   ```
   web-sfs-***   1/1     Running   0             23m
   ```

2. Execute o seguinte comando para verificar as opções de montagem (**web-sfs-*** é um pod de exemplo):

   ```
   kubectl exec -it web-sfs-*** -- mount -l | grep nfs
   ```

   Se as informações de montagem na saída do comando forem consistentes com as opções de montagem configuradas, as opções de montagem serão definidas com sucesso.

   ```
   <Your shared path> on /data type nfs
   (rw,relatime,vers=3,rsize=1048576,wsize=1048576,namlen=255,hard,nolock,noresvp
   ort,proto=tcp,timeo=600,retrans=2,sec=sys,mountaddr=**.**.**.**,mountvers=3,mo
   untport=2050,mountproto=tcp,local_lock=all,addr=**.**.**.**)
   ```

----Fim

Configurar opções de montagem em uma StorageClass

Você pode usar o campo **mountOptions** para definir opções de montagem em uma StorageClass. As opções que você pode configurar em **mountOptions** estão listadas em **Opções de montagem de volume do SFS**.

Passo 1 Use o kubectl para se conectar ao cluster. Para mais detalhes, consulte **Conexão a um cluster usando o kubectl**.

Passo 2 Crie uma StorageClass personalizada. Exemplo:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
```



```
metadata:
  name: csi-sfs-mount-option
provisioner: everest-csi-provisioner
parameters:
  csi.storage.k8s.io/csi-driver-name: nas.csi.everest.io
  csi.storage.k8s.io/fstype: nfs
everest.io/share-access-to: <your_vpc_id> # VPC ID of the cluster.
reclaimPolicy: Delete
volumeBindingMode: Immediate
mountOptions:                             # Mount options
- vers=3
- nolock
- timeo=600
- hard
```

Passo 3 Depois que a StorageClass é configurada, você pode usá-la para criar uma PVC. Por padrão, os PVs criados dinamicamente herdam as opções de montagem configuradas na StorageClass. Para mais detalhes, consulte **Uso de um sistema de arquivos do SFS através de um PV dinâmico**.

Passo 4 Verifique se as opções de montagem têm efeito.

Neste exemplo, a PVC é montada na carga de trabalho que usa a imagem **nginx:latest**. Você pode executar o comando **mount -l** para verificar se as opções de montagem têm efeito.

1. Visualize o pod no qual o volume do SFS foi montado. Neste exemplo, o nome da carga de trabalho é **web-sfs**.
   ```
   kubectl get pod | grep web-sfs
   ```
   Saída do comando:
   ```
   web-sfs-***   1/1     Running   0             23m
   ```
2. Execute o seguinte comando para verificar as opções de montagem (**web-sfs-*** é um pod de exemplo):
   ```
   kubectl exec -it web-sfs-*** -- mount -l | grep nfs
   ```
   Se as informações de montagem na saída do comando forem consistentes com as opções de montagem configuradas, as opções de montagem serão definidas com sucesso.
   ```
   <Your shared path> on /data type nfs
   (rw,relatime,vers=3,rsize=1048576,wsize=1048576,namlen=255,hard,nolock,noresvp
   ort,proto=tcp,timeo=600,retrans=2,sec=sys,mountaddr=**.**.**.**,mountvers=3,mo
   untport=2050,mountproto=tcp,local_lock=all,addr=**.**.**.**)
   ```

----Fim

8.5 SFS Turbo

8.5.1 Visão geral

Introdução

O CCE permite que você monte volumes de armazenamento criados por sistemas de arquivos do SFS Turbo em um caminho de um contêiner para atender aos requisitos de persistência de dados. Os sistemas de arquivos do SFS Turbo são rápidos, escaláveis e sob demanda, adequados para cenários com um grande número de arquivos pequenos, como DevOps e aplicativos corporativos.

Expansível até 320 TB, o SFS Turbo fornece um armazenamento de arquivos compartilhados totalmente hospedado, que é altamente disponível e estável, para suportar pequenos arquivos e aplicativos que exigem baixa latência e alto IOPS.

- **Protocolos de arquivo padrão**: você pode montar sistemas de arquivos como volumes em servidores, o mesmo que usar diretórios locais.
- **Compartilhamento de dados**: o mesmo sistema de arquivos pode ser montado em vários servidores, para que os dados possam ser compartilhados.
- **Rede privada**: os usuários podem acessar dados apenas em redes privadas de data centers.
- **Isolamento de dados**: o serviço de armazenamento na nuvem fornece armazenamento exclusivo de arquivos na nuvem, que oferece isolamento de dados e garante o desempenho de IOPS.
- **Casos de uso**: Implementações/StatefulSets no modo ReadWriteMany, DaemonSets e tarefas criadas para sites de alto tráfego, armazenamento de logs, DevOps e aplicações de OA corporativa

Desempenho do SFS Turbo

Para obter detalhes sobre os parâmetros de desempenho do SFS Turbo, consulte **Tipos de sistema de arquivos**.

Cenários

O SFS Turbo suporta os seguintes modos de montagem:

- **Uso de um sistema de arquivos do SFS Turbo existente por meio de um PV estático**: modo de criação estática, no qual você usa um volume do SFS existente para criar um PV e, em seguida, monta o armazenamento na carga de trabalho por meio de uma PVC.
- **Criação dinâmica de um subdiretório do SFS Turbo usando StorageClass**: o SFS Turbo permite que você crie dinamicamente subdiretórios e os monte em contêineres para que o SFS Turbo possa ser compartilhado e a capacidade de armazenamento do SFS Turbo possa ser usada de forma mais econômica e adequada.

Cobrança

O SFS Turbo não oferece suporte à criação dinâmica. Somente os volumes do SFS Turbo criados podem ser montados. Você pode selecionar o modo de faturamento pagamento por uso ou pacote anual/mensal, conforme necessário. Para obter detalhes sobre os preços do SFS Turbo, consulte **Cobrança**.

8.5.2 Uso de um sistema de arquivos do SFS Turbo existente por meio de um PV estático

O SFS Turbo é um sistema de arquivos compartilhado com alta disponibilidade e durabilidade. É adequado para aplicações que contêm grandes arquivos pequenos e exigem baixa latência e alto IOPS. Esta seção descreve como usar um sistema de arquivos do SFS Turbo existente para criar estaticamente PVs e PVCs e implementar a persistência e o compartilhamento de dados em cargas de trabalho.

Pré-requisitos

- Você criou um cluster e instalou o complemento **Armazenamento do contêiner do CCE (Everest)** no cluster.
- Se você quiser criar um cluster usando comandos, use kubectl para se conectar ao cluster. Para mais detalhes, consulte **Conexão a um cluster usando o kubectl**.

- Você criou um sistema de arquivos do SFS Turbo disponível, e o sistema de arquivos do SFS Turbo e o cluster estão na mesma VPC.

Restrições

- Vários PVs podem usar o mesmo sistema de arquivos do SFS ou SFS Turbo com as seguintes restrições:
  - Não monte todos as PVCs/PVs que usam o mesmo sistema de arquivos do SFS ou SFS Turbo subjacente em um pod. Isso leva a uma falha de inicialização do pod porque nem todos as PVCs podem ser montados no pod devido aos mesmos valores de **volumeHandle** desses PVs.
  - Sugere-se que o parâmetro **persistentVolumeReclaimPolicy** nos PVs seja definido como **Retain**. Caso contrário, quando um PV for excluído, o volume subjacente associado poderá ser excluído. Neste caso, outros PVs associados ao mau funcionamento do volume subjacente.
  - Quando o volume subjacente é usado repetidamente, ative o isolamento e a proteção de ReadWriteMany na camada da aplicação para evitar a substituição e a perda de dados.
- Para o armazenamento do SFS Turbo, os recursos anuais/mensais do SFS Turbo não serão recuperados quando o cluster ou a PVC forem excluídos. Recupere os recursos no console do SFS Turbo.

Usar um sistema de arquivos do SFS Turbo existente no console

Passo 1 Efetue logon no console do CCE e acesse o console do cluster.

Passo 2 Crie estaticamente uma PVC e PV.

1. Escolha **Storage** no painel de navegação e clique na guia **PersistentVolumeClaims (PVCs)**. Clique em **Create PVC** no canto superior direito. Na caixa de diálogo exibida, configure os parâmetros de PVC.

Parâmetro	Descrição
PVC Type	Neste exemplo, selecione **SFS Turbo**.
PVC Name	Digite o nome da PVC, que deve ser exclusivo no mesmo namespace.
Creation Method	Você pode criar um volume de armazenamento ou usar um volume de armazenamento existente para criar estaticamente uma PVC com base em se um PV foi criado. Neste exemplo, selecione **Create new** para criar um PV e uma PVC ao mesmo tempo no console.
PV[a]	Selecione um volume de PV existente no cluster. Crie um PV com antecedência. Para obter detalhes, consulte "Criação de um volume de armazenamento" em **Operações relacionadas**. Você não precisa especificar esse parâmetro neste exemplo.
SFS Turbo[b]	Clique em **Select SFS Turbo**. Na página exibida, selecione o sistema de arquivos do SFS Turbo que atende aos seus requisitos e clique em **OK**.

Parâmetro	Descrição
PV Name[b]	Digite o nome do PV, que deve ser exclusivo no mesmo cluster.
Access Mode[b]	Os volumes do SFS Turbo suportam apenas **ReadWriteMany**, indicando que um volume de armazenamento pode ser montado em vários nós no modo leitura/gravação. Para mais detalhes, consulte **Modos de acesso a volume**.
Reclaim Policy[b]	Somente **Retain** é suportada, indicando que o PV não é excluído quando a PVC é excluída. Para mais detalhes, consulte **Política de recuperação da PV**.
Mount Options[b]	Insira os pares chave-valor do parâmetro de montagem. Para mais detalhes, consulte **Configuração de opções de montagem do SFS Turbo**.

NOTA

a: o parâmetro está disponível quando **Creation Method** está definido como **Use existing**.

b: o parâmetro está disponível quando **Creation Method** está definido como **Create new**.

2. Clique em **Create** para criar uma PVC e um PV.

   Você pode escolher **Storage** no painel de navegação e exibir a PVC e o PV criados nas páginas de guia **PersistentVolumeClaims (PVCs)** e **PersistentVolumes (PVs)**, respectivamente.

Passo 3 Crie uma aplicação.

1. No painel de navegação à esquerda, clique em **Workloads**. No painel direito, clique na guia **Deployments**.
2. Clique em **Create Workload** no canto superior direito. Na página exibida, clique em **Data Storage** na área **Container Settings** e clique em **Add Volume** para selecionar **PVC**.

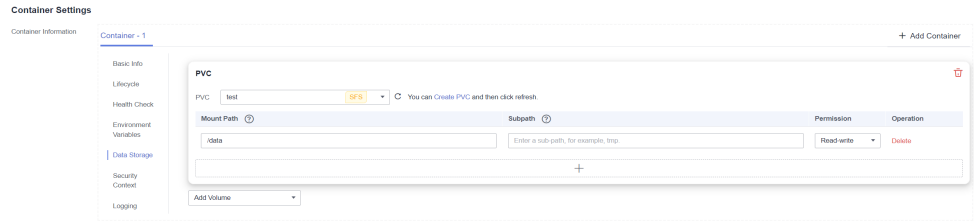
   Monte e use volumes de armazenamento, conforme mostrado na **Tabela 8-24**. Para obter detalhes sobre outros parâmetros, consulte **Cargas de trabalho**.

Tabela 8-24 Montagem de um volume de armazenamento

Parâmetro	Descrição
PVC	Selecione um volume do SFS Turbo existente.

Parâmetro	Descrição
Mount Path	Digite um caminho de montagem, por exemplo, **/tmp**. Este parâmetro indica o caminho do contêiner no qual um volume de dados será montado. Não monte o volume em um diretório do sistema como / ou **/var/run**. Caso contrário, os contêineres estarão com defeito. Monte o volume em um diretório vazio. Se o diretório não estiver vazio, verifique se não há arquivos que afetem a inicialização do contêiner. Caso contrário, os arquivos serão substituídos, causando falhas de inicialização do contêiner ou falhas de criação de carga de trabalho. **AVISO** Se um volume for montado em um diretório de alto risco, use uma conta com permissões mínimas para iniciar o contêiner. Caso contrário, arquivos de alto risco no host podem ser danificados.
Subpath	Digite um subcaminho, por exemplo, **tmp**, indicando que os dados no caminho de montagem do contêiner serão armazenados na pasta **tmp** do volume. Um subcaminho é usado para montar um volume local para que o mesmo volume de dados seja usado em um único pod. Se este parâmetro for deixado em branco, o caminho raiz é usado por padrão.
Permission	– **Read-only**: você só pode ler os dados nos volumes montados. – **Read/Write**: você pode modificar os volumes de dados montados no caminho. Os dados recém-gravados não serão migrados se o contêiner for migrado, o que pode causar perda de dados.

Neste exemplo, o disco é montado no caminho **/data** do contêiner. Os dados de contêiner gerados nesse caminho são armazenados no sistema de arquivos do SFS Turbo.



3. Após a configuração, clique em **Create Workload**.

   Depois que a carga de trabalho for criada, os dados no diretório de montagem do contêiner serão armazenados persistentemente. Verifique o armazenamento referindo-se a **Verificar a persistência e o compartilhamento de dados**.

----Fim

(kubectl) Usar um sistema de arquivos do SFS existente

Passo 1 Use o kubectl para se conectar ao cluster.

Passo 2 Crie um PV.

1. Crie o arquivo **pv-sfsturbo.yaml**.

```
apiVersion: v1
kind: PersistentVolume
metadata:
  annotations:
    pv.kubernetes.io/provisioned-by: everest-csi-provisioner
  name: pv-sfsturbo    # PV name.
spec:
  accessModes:
  - ReadWriteMany      # Access mode. The value must be ReadWriteMany for SFS
Turbo.
  capacity:
    storage: 500Gi       # SFS Turbo volume capacity.
  csi:
    driver: sfsturbo.csi.everest.io    # Dependent storage driver for the
mounting.
    fsType: nfs
    volumeHandle: <your_volume_id>   # SFS Turbo volume ID.
    volumeAttributes:
      everest.io/share-export-location: <your_location>   # Shared path of
the SFS Turbo volume.
      everest.io/enterprise-project-id: <your_project_id>  # Project ID of
the SFS Turbo volume.
      storage.kubernetes.io/csiProvisionerIdentity: everest-csi-provisioner
  persistentVolumeReclaimPolicy: Retain    # Reclaim policy.
  storageClassName: csi-sfsturbo          # Storage class name of the SFS
Turbo file system.
  mountOptions: []                         # Mount options.
```

Tabela 8-25 Parâmetros principais

Parâmetro	Obrigatório	Descrição
volumeHandle	Sim	ID do volume do SFS Turbo. Como obter: faça logon no console, escolha **Service List** > **Storage** > **Scalable File Service** e selecione **SFS Turbo**. Na lista, clique no nome do volume do SFS Turbo de destino. Na página de detalhes, copie o conteúdo seguinte **ID**.
everest.io/share-export-location	Sim	Caminho compartilhado do volume do SFS Turbo. Faça logon no console, escolha **Service List** > **Storage** > **Scalable File Service** e selecione **SFS Turbo**. Você pode obter o caminho compartilhado do sistema de arquivos na coluna **Mount Address**.

Parâmetro	Obrigatório	Descrição
everest.io/enterprise-project-id	Não	ID do projeto do volume do SFS Turbo. Como obter: no console do SFS, clique em **SFS Turbo** no painel de navegação esquerdo. Clique no nome do sistema de arquivos do SFS Turbo para interconectar. Na guia **Basic Info**, localize e clique no projeto empresarial para acessar o console e copiar o ID.

mountOptions	Não	Opções de montagem.
		Se não for especificada, as seguintes configurações são usadas por padrão. Para mais detalhes, consulte **Configuração de opções de montagem do SFS Turbo**.
		`mountOptions:`
		`- vers=3`
		`- timeo=600`
		`- nolock`
		`- hard`

persistentVolumeReclaimPolicy	Sim	Uma política de recuperação é suportada quando a versão do cluster é ou posterior a 1.19.10 e a versão do everest é ou posterior a 1.2.9. Apenas a política de recuperação **Retain** é suportada. Para mais detalhes, consulte **Política de recuperação da PV**. **Retain**: quando uma PVC é excluída, o PV e os recursos de armazenamento subjacentes não são excluídos. Em vez disso, você deve excluir manualmente esses recursos. Depois disso, o PV está no estado **Released** e não pode ser vinculado à PVC novamente.
storage	Sim	Capacidade solicitada na PVC, em Gi.
storageClassName	Sim	O nome da classe de armazenamento dos volumes do SFS Turbo é **csi-sfsturbo**.

2. Execute o seguinte comando para criar um PV:
   ```
   kubectl apply -f pv-sfsturbo.yaml
   ```

Passo 3 Crie uma PVC.

1. Crie o arquivo **pvc-sfsturbo.yaml**.
   ```
   apiVersion: v1
   kind: PersistentVolumeClaim
   metadata:
     name: pvc-sfsturbo
     namespace: default
     annotations:
       volume.beta.kubernetes.io/storage-provisioner: everest-csi-provisioner
       everest.io/enterprise-project-id: <your_project_id>  # Project ID of the
   SFS Turbo volume.
   spec:
   ```



```
  accessModes:
  - ReadWriteMany                 # The value must be ReadWriteMany for SFS
Turbo.
  resources:
    requests:
      storage: 500Gi              # SFS Turbo volume capacity.
  storageClassName: csi-sfsturbo      # Storage class of the SFS Turbo
volume, which must be the same as that of the PV.
  volumeName: pv-sfsturbo    # PV name.
```

Tabela 8-26 Parâmetros principais

Parâmetro	Obrigatório	Descrição
everest.io/enterprise-project-id	Não	ID do projeto do volume do SFS Turbo. Como obter: no console do SFS, clique em **SFS Turbo** no painel de navegação esquerdo. Clique no nome do sistema de arquivos do SFS Turbo para interconectar. Na guia **Basic Info**, localize e clique no projeto empresarial para acessar o console e copiar o ID.
Armazenamento	Sim	Capacidade solicitada na PVC, em Gi. O valor deve ser o mesmo que o tamanho de armazenamento do PV existente.
storageClassName	Sim	Nome da classe de armazenamento, que deve ser o mesmo que a classe de armazenamento do PV em **1**. O nome da classe de armazenamento dos volumes do SFS Turbo é **csi-sfsturbo**.
volumeName	Sim	Nome do PV, que deve ser o mesmo que o nome do PV em **1**.

2. Execute o seguinte comando para criar uma PVC:
   ```
   kubectl apply -f pvc-sfsturbo.yaml
   ```

Passo 4 Crie uma aplicação.

1. Crie um arquivo chamado **web-demo.yaml**. Neste exemplo, o volume do SFS Turbo é montado no caminho **/data**.
   ```
   apiVersion: apps/v1
   kind: Deployment
   metadata:
     name: web-demo
     namespace: default
   spec:
     replicas: 2
     selector:
       matchLabels:
         app: web-demo
     template:
       metadata:
         labels:
           app: web-demo
       spec:
         containers:
   ```



```
      - name: container-1
        image: nginx:latest
        volumeMounts:
        - name: pvc-sfsturbo-volume    #Volume name, which must be the same
as the volume name in the volumes field.
          mountPath: /data  #Location where the storage volume is mounted.
      imagePullSecrets:
        - name: default-secret
      volumes:
        - name: pvc-sfsturbo-volume    #Volume name, which can be customized.
          persistentVolumeClaim:
            claimName: pvc-sfsturbo    #Name of the created PVC.
```

2. Execute o seguinte comando para criar uma carga de trabalho para qual o volume do SFS Turbo está montado:
   ```
   kubectl apply -f web-demo.yaml
   ```
   Depois que a carga de trabalho é criada, você pode tentar **Verificar a persistência e o compartilhamento de dados**.

----Fim

Verificar a persistência e o compartilhamento de dados

Passo 1 Exiba a aplicação implementada e os arquivos.

1. Execute o seguinte comando para exibir o pod criado:
   ```
   kubectl get pod | grep web-demo
   ```
   Saída esperada:
   ```
   web-demo-846b489584-mjhm9   1/1     Running   0             46s
   web-demo-846b489584-wvv5s   1/1     Running   0             46s
   ```
2. Execute os seguintes comandos em sequência para visualizar os arquivos no caminho **/data** dos pods:
   ```
   kubectl exec web-demo-846b489584-mjhm9 -- ls /data
   kubectl exec web-demo-846b489584-wvv5s -- ls /data
   ```
   Se nenhum resultado for retornado para ambos os pods, nenhum arquivo existirá no caminho **/data**.

Passo 2 Execute o seguinte comando para criar um arquivo chamado **static** no caminho **/data**:
```
kubectl exec web-demo-846b489584-mjhm9 --  touch /data/static
```

Passo 3 Execute o seguinte comando para exibir os arquivos no caminho **/data**:
```
kubectl exec web-demo-846b489584-mjhm9 -- ls /data
```

Saída esperada:

```
static
```

Passo 4 **Verifique a persistência dos dados.**

1. Execute o seguinte comando para excluir o pod chamado **web-demo-846b489584-mjhm9**:
   ```
   kubectl delete pod web-demo-846b489584-mjhm9
   ```
   Saída esperada:
   ```
   pod "web-demo-846b489584-mjhm9" deleted
   ```
   Após a eliminação, o controlador de Implementação cria automaticamente uma réplica.
2. Execute o seguinte comando para exibir o pod criado:
   ```
   kubectl get pod | grep web-demo
   ```
   A saída esperada é a seguinte, na qual **web-demo-846b489584-d4d4j** é o pod recém-criado:


```
web-demo-846b489584-d4d4j   1/1     Running   0             110s
web-demo-846b489584-wvv5s    1/1     Running   0             7m50s
```

3. Execute o seguinte comando para verificar se os arquivos no caminho **/data** do pod novo foram modificados:
   ```
   kubectl exec web-demo-846b489584-d4d4j -- ls /data
   ```
   Saída esperada:
   ```
   static
   ```
   Se o arquivo **static** ainda existir, os dados podem ser armazenados persistentemente.

Passo 5 Verifique o compartilhamento de dados.

1. Execute o seguinte comando para exibir o pod criado:
   ```
   kubectl get pod | grep web-demo
   ```
   Saída esperada:
   ```
   web-demo-846b489584-d4d4j   1/1     Running   0             7m
   web-demo-846b489584-wvv5s   1/1     Running   0             13m
   ```
2. Execute o seguinte comando para criar um arquivo chamado **share** no caminho **/data** de qualquer pod: Neste exemplo, selecione o pod chamado **web-demo-846b489584-d4d4j**.
   ```
   kubectl exec web-demo-846b489584-d4d4j --  touch /data/share
   ```
   Verifique os arquivos no caminho **/data** do pod.
   ```
   kubectl exec web-demo-846b489584-d4d4j -- ls /data
   ```
   Saída esperada:
   ```
   share
   static
   ```
3. Verifique se o arquivo **share** existe no caminho **/data** de outro pod (**web-demo-846b489584-wvv5s**) também para verificar o compartilhamento de dados.
   ```
   kubectl exec web-demo-846b489584-wvv5s -- ls /data
   ```
   Saída esperada:
   ```
   share
   static
   ```
   Depois de criar um arquivo no caminho **/data** de um pod, se o arquivo também for criado no caminho **/data** do outro pod, os dois pods compartilharão o mesmo volume.

----Fim

Operações relacionadas

Você também pode executar as operações listadas em **Tabela 8-27**.

Tabela 8-27 Operações relacionadas

Operação	Descrição	Procedimento
Criar um volume de armazenamento (PV)	Crie um PV no console do CCE.	1. Escolha **Storage** no painel de navegação e clique na guia **PersistentVolumes (PVs)**. Clique em **Create Volume** no canto superior direito. Na caixa de diálogo exibida, configure os parâmetros. ● **Volume Type**: selecione **SFS Turbo**. ● **SFS Turbo**: clique em **Select SFS Turbo**. Na página exibida, selecione o volume do SFS Turbo que atende aos requisitos e clique em **OK**. ● **PV Name**: digite o nome do PV, que deve ser exclusivo no mesmo cluster. ● **Access Mode**: os volumes do SFS suportam apenas **ReadWriteMany** indicando que um volume de armazenamento pode ser montado em vários nós no modo leitura/gravação. Para mais detalhes, consulte **Modos de acesso a volume**. ● **Reclaim Policy**: somente **Retain** é suportada. Para mais detalhes, consulte **Política de recuperação da PV**. ● **Mount Options**: insira os pares chave-valor do parâmetro de montagem. Para mais detalhes, consulte **Configuração de opções de montagem do SFS Turbo**. 2. Clique em **Create**.
Expandir a capacidade de um volume do SFS Turbo	Expanda rapidamente a capacidade de um volume do SFS Turbo montado no console do CCE.	1. Escolha **Storage** no painel de navegação e clique na guia **PersistentVolumeClaims (PVCs)**. Clique em **More** na coluna **Operation** da PVC de destino e selecione **Scale-out**. 2. Insira a capacidade a ser adicionada e clique em **OK**.
Visualizar eventos	Você pode visualizar nomes de eventos, tipos de eventos, número de ocorrências, eventos do Kubernetes, horário da primeira ocorrência e horário da última ocorrência da PVC ou PV.	1. Escolha **Storage** no painel de navegação e clique na guia **PersistentVolumeClaims (PVCs)** ou **PersistentVolumes (PVs)**. 2. Clique em **View Events** na coluna **Operation** da PVC ou do PV de destino para exibir os eventos gerados em uma hora (os dados do evento são mantidos por uma hora).

Operação	Descrição	Procedimento
Exibir um arquivo YAML	Você pode visualizar, copiar e fazer download dos arquivos YAML de uma PVC ou um PV.	1. Escolha **Storage** no painel de navegação e clique na guia **PersistentVolumeClaims (PVCs)** ou **PersistentVolumes (PVs)**. 2. Clique em **View YAML** na coluna **Operation** da PVC ou PV de destino para exibir ou fazer download do YAML.

8.5.3 Configuração de opções de montagem do SFS Turbo

Esta seção descreve como configurar as opções de montagem do SFS Turbo. Para o SFS Turbo, você só pode definir opções de montagem em um PV e vincular o PV criando uma PVC.

Pré-requisitos

A versão do complemento **Armazenamento do contêiner do CCE (Everest)** deve ser **1.2.8 ou posterior**. Esse complemento identifica as opções de montagem e as transfere para os recursos de armazenamento subjacentes. As configurações de parâmetros terão efeito somente se os recursos de armazenamento subjacentes suportarem as opções especificadas.

Restrições

- As opções de montagem não podem ser configuradas para contêineres seguros.
- Devido às restrições do protocolo NFS, se um volume do SFS for montado em um nó por várias vezes, os parâmetros de montagem relacionados ao link (como **timeo**) terão efeito somente quando o volume do SFS for montado pela primeira vez. Por exemplo, se um volume do SFS for montado em vários pods em execução em um nó, os valores dos parâmetros de montagem configurados posteriormente não substituirão os valores de parâmetros existentes.

Opções de montagem do SFS Turbo

O complemento everest no CCE pré-ajusta as opções descritas em **Tabela 8-28** para a montagem de volumes do SFS Turbo.

Tabela 8-28 Opções de montagem do SFS Turbo

Parâmetro	Valor	Descrição
vers	3	Versão do sistema de arquivos. No momento, apenas NFSv3 é suportada. Valor: **3**
nolock	Deixe em branco.	Se deseja bloquear arquivos no servidor usando o protocolo NLM. Se **nolock** for selecionado, o bloqueio é válido para aplicações em um host. Para aplicações em outro host, o bloqueio é inválido.

Parâmetro	Valor	Descrição
timeo	600	Tempo de espera antes do cliente de NFS retransmitir uma solicitação. A unidade é de 0,1 segundo. Valor recomendado: **600**
hard/soft	Deixe em branco.	Modo de montagem. ● **hard**: se a solicitação NFS expirar, o cliente continuará reenviando a solicitação até que a solicitação seja bem-sucedida. ● **soft**: se a solicitação NFS expirar, o cliente retornará um erro para o programa invocando. O valor padrão é **hard**.

Você pode definir outras opções de montagem, se necessário. Para obter detalhes, consulte **Montagem de um sistema de arquivos NFS para ECSs (Linux)**.

Configurar opções de montagem em um PV

Você pode usar o campo **mountOptions** para configurar opções de montagem em um PV. As opções que você pode configurar em **mountOptions** estão listadas em **Opções de montagem do SFS Turbo**.

Passo 1 Use o kubectl para se conectar ao cluster. Para mais detalhes, consulte **Conexão a um cluster usando o kubectl**.

Passo 2 Defina as opções de montagem em um PV. Exemplo:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  annotations:
    pv.kubernetes.io/provisioned-by: everest-csi-provisioner
  name: pv-sfsturbo    # PV name.
spec:
  accessModes:
  - ReadWriteMany      # Access mode. The value must be ReadWriteMany for SFS
Turbo.
  capacity:
    storage: 500Gi       # SFS Turbo volume capacity.
  csi:
    driver: sfsturbo.csi.everest.io    # Dependent storage driver for the
mounting.
    fsType: nfs
    volumeHandle: {your_volume_id}   # SFS Turbo volume ID
    volumeAttributes:
      everest.io/share-export-location: {your_location}   # Shared path of the
SFS Turbo volume.
      everest.io/enterprise-project-id: {your_project_id}  # Project ID of the
SFS Turbo volume.
      storage.kubernetes.io/csiProvisionerIdentity: everest-csi-provisioner
  persistentVolumeReclaimPolicy: Retain    # Reclaim policy.
  storageClassName: csi-sfsturbo           # SFS Turbo storage class name.
  mountOptions:                            # Mount options.
  - vers=3
  - nolock
  - timeo=600
  - hard
```


Passo 3 Depois que um PV é criado, você pode criar uma PVC e vinculá-la ao PV e, em seguida, montar o PV ao contêiner na carga de trabalho. Para mais detalhes, consulte **Uso de um sistema de arquivos do SFS Turbo existente por meio de um PV estático**.

Passo 4 Verifique se as opções de montagem têm efeito.

Neste exemplo, a PVC é montada na carga de trabalho que usa a imagem **nginx:latest**. Você pode executar o comando **mount -l** para verificar se as opções de montagem têm efeito.

1. Visualize o pod no qual o volume do SFS Turbo foi montado. Neste exemplo, o nome da carga de trabalho é **web-sfsturbo**.
   ```
   kubectl get pod | grep web-sfsturbo
   ```
   Saída do comando:
   ```
   web-sfsturbo-***   1/1     Running   0             23m
   ```
2. Execute o seguinte comando para verificar as opções de montagem (**web-sfsturbo-*** ** é um pod de exemplo):
   ```
   kubectl exec -it web-sfsturbo-*** -- mount -l | grep nfs
   ```
   Se as informações de montagem na saída do comando forem consistentes com as opções de montagem configuradas, as opções de montagem foram configuradas.
   ```
   <Your mount path> on /data type nfs
   (rw,relatime,vers=3,rsize=1048576,wsize=1048576,namlen=255,hard,nolock,noresvp
   ort,proto=tcp,timeo=600,retrans=2,sec=sys,mountaddr=**.**.**.**,mountvers=3,mo
   untport=20048,mountproto=tcp,local_lock=all,addr=**.**.**.**)
   ```

----Fim

8.5.4 Criação dinâmica de um subdiretório do SFS Turbo usando StorageClass

Plano de fundo

A capacidade mínima de um sistema de arquivos do SFS Turbo é de 500 GiB e o sistema de arquivos do SFS Turbo não pode ser cobrado pelo uso. Por padrão, o diretório raiz de um sistema de arquivos do SFS Turbo é montado em um contêiner que, na maioria dos casos, não requer uma capacidade tão grande.

O complemento everest permite criar dinamicamente subdiretórios em um sistema de arquivos SFS Turbo e montar esses subdiretórios em contêineres. Dessa forma, um sistema de arquivos do SFS Turbo pode ser compartilhado por vários contêineres para aumentar a eficiência do armazenamento.

Restrições

- Somente clusters da v1.15 ou posterior são suportados.
- O cluster deve usar o complemento everest da versão 1.1.13 ou posterior.
- Não há suporte para contêineres de Kata.
- Quando o complemento everest anterior a 1.2.69 ou 2.1.11 é usado, um máximo de 10 PVCs podem ser criados simultaneamente por vez usando a função de subdiretório. é recomendado o everest de 1.2.69 ou posterior ou de 2.1.11 ou posterior.
- Um volume subPath é um subdiretório de um sistema de arquivos de SFS Turbo. Aumentar a capacidade de um PVC desse tipo altera apenas o intervalo de recursos especificado pelo PVC, mas não altera a capacidade total do sistema de arquivos de SFS Turbo. Se a capacidade total de recursos do sistema de arquivos de SFS Turbo não for

suficiente, a capacidade disponível do volume subPath será restringida. Para corrigir isso, você deve aumentar a capacidade de recursos do sistema de arquivos de SFS Turbo no console do SFS Turbo.

A exclusão do volume subPath não resulta na exclusão dos recursos do sistema de arquivos de SFS Turbo.

Criação de um volume de SFS Turbo do tipo subPath

Passo 1 Crie um sistema de arquivos do SFS Turbo na mesma VPC e sub-rede que o cluster.

Passo 2 Crie um arquivo YAML de StorageClass, por exemplo, **sfsturbo-subpath-sc.yaml**.

O seguinte é um exemplo:

```
apiVersion: storage.k8s.io/v1
allowVolumeExpansion: true
kind: StorageClass
metadata:
  name: sfsturbo-subpath-sc
mountOptions:
- lock
parameters:
  csi.storage.k8s.io/csi-driver-name: sfsturbo.csi.everest.io
  csi.storage.k8s.io/fstype: nfs
  everest.io/archive-on-delete: "true"
  everest.io/share-access-to: 7ca2dba2-1234-1234-1234-626371a8fb3a
  everest.io/share-expand-type: bandwidth
  everest.io/share-export-location: 192.168.1.1:/sfsturbo/
  everest.io/share-source: sfs-turbo
  everest.io/share-volume-type: STANDARD
  everest.io/volume-as: subpath
  everest.io/volume-id: 0d773f2e-1234-1234-1234-de6a35074696
provisioner: everest-csi-provisioner
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

Neste exemplo:

- **name**: indica o nome do StorageClass.
- **mountOptions**: indica as opções de montagem. Este campo é opcional.
  - Em versões posteriores ao everest 1.1.13 e anteriores ao everest 1.2.8, somente o parâmetro **nolock** pode ser configurado. Por padrão, o parâmetro **nolock** é usado para a operação de montagem e não precisa ser configurado. Se **nolock** for definido como **false**, o campo **lock** será usado.
  - A partir do everest 1.2.8, mais opções de montagem são suportadas. Para obter detalhes, consulte **Configuração das opções de montagem do SFS Turbo**. **Não defina nolock como true. Caso contrário, a operação de montagem falhará.**

    ```
    mountOptions:
    - vers=3
    - timeo=600
    - nolock
    - hard
    ```
- **everest.io/volume-as**: este parâmetro é definido como **subpath** para usar o volume subPath.
- **everest.io/share-access-to**: este parâmetro é opcional. Em um volume de subPath, defina esse parâmetro como o ID da VPC onde o sistema de arquivos do SFS Turbo está localizado.

- **everest.io/share-expand-type**: este parâmetro é opcional. Se o tipo do sistema de arquivos de SFS Turbo for SFS Turbo Standard – Enhanced ou SFS Turbo Performance – Enhanced, defina esse parâmetro como **bandwidth**.
- **everest.io/share-export-location**: este parâmetro indica o diretório de montagem. Consiste no caminho compartilhado e no subdiretório do SFS Turbo. O caminho compartilhado pode ser obtido no console do SFS Turbo. O subdiretório é definido pelo usuário. Os PVCs criados usando o StorageClass estão localizados nesse subdiretório.
- **everest.io/share-volume-type**: este parâmetro é opcional. Especifica o tipo de sistema de arquivos de SFS Turbo. O valor pode ser **STANDARD** ou **PERFORMANCE**. Para tipos avançados, este parâmetro deve ser usado em conjunto com **everest.io/share-expand-type** (cujo valor deve ser **bandwidth**).
- **everest.io/zone**: este parâmetro é opcional. Defina-o como a AZ em que o sistema de arquivos de SFS Turbo está localizado.
- **everest.io/volume-id**: este parâmetro indica o ID do volume do SFS Turbo. Você pode obter o ID do volume na página de SFS Turbo.
- **everest.io/archive-on-delete**: se esse parâmetro for definido como **true** e **Delete** for selecionado para **Reclaim Policy**, os documentos originais do PV serão arquivados no diretório chamado **archived-***{$PV name.timestamp}* antes que o PVC seja excluído. Se esse parâmetro for definido como **false**, o subdiretório de SFS Turbo do PV correspondente será excluído. O valor padrão é **true**, indicando que os documentos originais do PV serão arquivados no diretório denominado **archived-***{$PV name.timestamp}* antes que o PVC seja excluído.

Passo 3 Execute **kubectl create -f sfsturbo-subpath-sc.yaml**.

Passo 4 Crie um arquivo YAML do PVC chamado **sfs-turbo-test.yaml**.

O seguinte é um exemplo:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: sfs-turbo-test
  namespace: default
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 50Gi
  storageClassName: sfsturbo-subpath-sc
  volumeMode: Filesystem
```

Neste exemplo:

- **name**: indica o nome do PVC.
- **storageClassName**: especifica o nome de StorageClass.
- **storage**: em um volume de subPath, modificar o valor deste parâmetro não afeta a capacidade do recurso do sistema de arquivos de SFS Turbo. Um volume de subPath é essencialmente um caminho de arquivo dentro de um sistema de arquivos de SFS Turbo. Como resultado, o aumento da capacidade do volume subPath em um PVC não leva a um aumento nos recursos do sistema de arquivos de SFS Turbo.

  NOTA

  A capacidade de um volume de subPath é restrita pela capacidade geral de recursos do sistema de arquivos de SFS Turbo correspondente. Se os recursos do sistema de arquivos do SFS Turbo forem inadequados, você poderá ajustar a capacidade do recurso por meio do console do SFS Turbo.

Passo 5 Execute **kubectl create -f sfs-turbo-test.yaml**.

----Fim

Criação de um Deployment e montagem de um volume existente

Passo 1 Crie um arquivo YAML para o Deployment, por exemplo, **deployment-test.yaml**.

O seguinte é um exemplo:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: test-turbo-subpath-example
  namespace: default
  generation: 1
  labels:
    appgroup: ''
spec:
  replicas: 1
  selector:
    matchLabels:
      app: test-turbo-subpath-example
  template:
    metadata:
      labels:
        app: test-turbo-subpath-example
    spec:
      containers:
      - image: nginx:latest
        name: container-0
        volumeMounts:
        - mountPath: /tmp
          name: pvc-sfs-turbo-example
      restartPolicy: Always
      imagePullSecrets:
      - name: default-secret
      volumes:
      - name: pvc-sfs-turbo-example
        persistentVolumeClaim:
          claimName: sfs-turbo-test
```

Neste exemplo:

- **name**: indica o nome da carga de trabalho criada.
- **image**: especifica a imagem usada pela carga de trabalho.
- **mountPath**: indica o caminho de montagem do contêiner. Neste exemplo, o volume é montado no diretório **/tmp**.
- **claimName**: indica o nome de um PVC existente.

Passo 2 Crie o Deployment.

kubectl create -f deployment-test.yaml

----Fim

Criação dinâmica de um volume de subPath para um StatefulSet

Passo 1 Crie um arquivo YAML para um StatefulSet, por exemplo, **statefulset-test.yaml**.

O seguinte é um exemplo:

```
apiVersion: apps/v1
kind: StatefulSet
```



```
metadata:
  name: test-turbo-subpath
  namespace: default
  generation: 1
  labels:
    appgroup: ''
spec:
  replicas: 2
  selector:
    matchLabels:
      app: test-turbo-subpath
  template:
    metadata:
      labels:
        app: test-turbo-subpath
      annotations:
        metrics.alpha.kubernetes.io/custom-endpoints:
'[{"api":"","path":"","port":"","names":""}]'
        pod.alpha.kubernetes.io/initialized: 'true'
    spec:
      containers:
        - name: container-0
          image: 'nginx:latest'
          resources: {}
          volumeMounts:
            - name: sfs-turbo-160024548582479676
              mountPath: /tmp
          terminationMessagePath: /dev/termination-log
          terminationMessagePolicy: File
          imagePullPolicy: IfNotPresent
      restartPolicy: Always
      terminationGracePeriodSeconds: 30
      dnsPolicy: ClusterFirst
      securityContext: {}
      imagePullSecrets:
        - name: default-secret
      affinity: {}
      schedulerName: default-scheduler
  volumeClaimTemplates:
    - metadata:
        name: sfs-turbo-160024548582479676
        namespace: default
        annotations: {}
      spec:
        accessModes:
          - ReadWriteOnce
        resources:
          requests:
            storage: 10Gi
        storageClassName: sfsturbo-subpath-sc
  serviceName: wwww
  podManagementPolicy: OrderedReady
  updateStrategy:
    type: RollingUpdate
  revisionHistoryLimit: 10
```

Neste exemplo:

- **name**: indica o nome da carga de trabalho criada.
- **image**: especifica a imagem usada pela carga de trabalho.
- **mountPath**: indica o caminho de montagem do contêiner. Neste exemplo, o volume é montado no diretório **/tmp**.
- **spec.template.spec.containers.volumeMounts.name** e **spec.volumeClaimTemplates.metadata.name**: devem ser consistentes porque têm um relacionamento de mapeamento.

- **storageClassName**: especifica o nome de um StorageClass local.

Passo 2 Crie o StatefulSet.

kubectl create -f statefulset-test.yaml

----Fim

8.6 Object Storage Service

8.6.1 Visão geral

Introdução

O Object Storage Service (OBS) fornece recursos de armazenamento de dados maciços, seguros e econômicos para que você armazene dados de qualquer tipo e tamanho. Você pode usá-lo em backup/arquivamento corporativo, vídeo sob demanda (VoD), vigilância por vídeo e muitos outros cenários.

- **APIs padrão**: com as APIs RESTful HTTP, o OBS permite que você use ferramentas de cliente ou ferramentas de terceiros para acessar o armazenamento de objetos.
- **Compartilhamento de dados**: servidores, dispositivos incorporados e dispositivos IoT podem usar o mesmo caminho para acessar dados de objetos compartilhados no OBS.
- **Redes públicas/privadas**: o OBS permite que os dados sejam acessados a partir de redes públicas para atender aos requisitos de aplicações da Internet.
- **Capacidade e desempenho**: sem limite de capacidade; alto desempenho (latência de I/O de leitura/gravação em até 10 ms).
- **Casos de uso**: as Implementações/StatefulSets no modo **ReadOnlyMany** e tarefas criadas para análise de Big Data, hospedagem de sites estáticos, vídeo on-line sob demanda (VoD), sequenciamento de genes, vigilância por vídeo inteligente, backup e arquivamento e caixas de nuvem empresariais (discos da Web). Você pode criar armazenamento de objetos usando o console do OBS, as ferramentas e os SDKs.

Especificações do OBS

O OBS fornece várias classes de armazenamento para atender aos requisitos dos clientes sobre desempenho e custos de armazenamento.

- Sistema de arquivos paralelos (PFS, **recomendado**): é um sistema de arquivos otimizado de alto desempenho fornecido pelo OBS. Ele fornece latência de acesso de milissegundos, largura de banda de TB/s e IOPS de milhões de níveis e pode processar rapidamente cargas de trabalho de HPC. O PFS supera o desempenho dos buckets do OBS. Para obter detalhes, consulte **Sobre o sistema de arquivos paralelos**.
- Bucket de objetos (**não recomendado**):
  - Standard: apresenta baixa latência e alta taxa de transferência. Portanto, é bom para armazenar arquivos acessados com frequência (várias vezes por mês) ou arquivos pequenos (menos de 1 MB). Seus cenários de aplicações incluem análise de big data, aplicativos móveis, vídeos quentes e aplicações sociais.
  - OBS Infrequent Access: aplicável ao armazenamento de dados acessados semi-frequentemente (menos de 12 vezes por ano) que exigem resposta rápida. Seus

cenários de aplicações incluem sincronização ou compartilhamento de arquivos e backup em nível empresarial. Essa classe de armazenamento tem a mesma durabilidade, baixa latência e alta taxa de transferência que a classe de armazenamento Standard, com um custo menor, mas sua disponibilidade é um pouco menor do que a classe de armazenamento Standard.

Para obter detalhes sobre as classes de armazenamento do OBS, consulte **Classes do armazenamento**.

Cenários

O OBS suporta os seguintes modos de montagem com base em cenários de aplicação:

- **Uso de um bucket do OBS existente através de um PV estático**: modo de criação estática, no qual você usa um volume do OBS existente para criar um PV e, em seguida, monta o armazenamento na carga de trabalho por meio de uma PVC. Esse modo se aplica a cenários em que o armazenamento subjacente está disponível ou é cobrado anualmente/mensalmente.
- **Uso de um bucket do OBS através de um PV dinâmico**: modo de criação dinâmica, onde você não precisa criar volumes do OBS antecipadamente. Em vez disso, especifique uma StorageClass durante a criação de PVC e um volume do OBS e um PV serão criados automaticamente. Esse modo se aplica a cenários em que nenhum armazenamento subjacente está disponível.

Cobrança

- Quando um volume OBS é montado, o modo de faturamento do OBS **criado automaticamente** usando StorageClass é **pagamento por uso** por padrão. Para obter detalhes sobre preços do OBS, consulte **Detalhes de preços do OBS**.
- Se você quiser ser cobrado no modo anual/mensal, **use os volumes existentes do OBS**.

8.6.2 Uso de um bucket do OBS existente através de um PV estático

Esta seção descreve como usar um bucket do Object Storage Service (OBS) existente para criar estaticamente PVs e PVCs e implementar a persistência e o compartilhamento de dados em cargas de trabalho.

Pré-requisitos

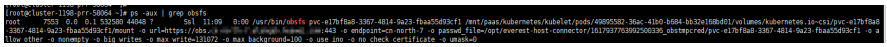
- Você criou um cluster e instalou o complemento **Armazenamento do contêiner do CCE (Everest)** no cluster.
- Se você quiser criar um cluster usando comandos, use kubectl para se conectar ao cluster. Para mais detalhes, consulte **Conexão a um cluster usando o kubectl**.
- Você criou um bucket do OBS. Um bucket do OBS do tipo de sistema de arquivos paralelo pode ser selecionado somente quando estiver na mesma região do cluster.

Restrições

- Se forem utilizados volumes do OBS, o grupo proprietário e a permissão do ponto de montagem não podem ser modificados.
- O CCE permite que sistemas de arquivos paralelos sejam montados usando SDKs ou PVCs do OBS. Se for usada montagem em PVC, a ferramenta obsfs fornecida pelo OBS

deve ser usada. Um processo residente de obsfs é gerado cada vez que um volume de armazenamento de objetos gerado a partir de um sistema de arquivos paralelo é montado em um nó.

Figura 8-3 Processo residente do obsfs



Reserve 1 GiB de memória para cada processo de obsfs. Por exemplo, para um nó com 4 vCPUs e 8 GiB de memória, um sistema de arquivos paralelo obsfs deve ser montado em **no máximo** oito pods.

NOTA

- Um processo residente do obsfs é executado em um nó. Se a memória consumida exceder o limite superior do nó, o nó funciona mal. Em um nó com 4 vCPUs e 8 GiB de memória, se mais de 100 pods forem montados em um sistema de arquivos paralelo, o nó não estará disponível. Controle o número de pods montados em um sistema de arquivos paralelo em um único nó.
- Ao usar obsfs, cumpra com as **Restrições de obsfs**.

- Os contêineres de Kata não suportam volumes do OBS.
- Vários PVs podem usar o mesmo volume de armazenamento do OBS com as seguintes restrições:
  - Não monte todos os PVCs/PVs que usam o mesmo volume de armazenamento de objetos subjacente em um pod. Isso leva a uma falha de inicialização do pod porque nem todas as PVCs podem ser montados no pod devido aos mesmos valores de **volumeHandle** desses PVs.
  - O parâmetro **persistentVolumeReclaimPolicy** nos PVs deve ser definido como **Retain**. Caso contrário, quando um PV for excluído, o volume subjacente associado poderá ser excluído. Neste caso, outros PVs associados ao mau funcionamento do volume subjacente.
  - Se o armazenamento subjacente for usado repetidamente, será necessário manter a consistência dos dados. Ative o isolamento e a proteção do ReadWriteMany na camada de aplicação e evite que vários clientes gravem o mesmo arquivo para evitar a substituição e a perda de dados.

Usar um bucket do OBS existente no console

Passo 1 Efetue logon no console do CCE e acesse o console do cluster.

Passo 2 Crie estaticamente uma PVC e PV.

1. Escolha **Storage** no painel de navegação e clique na guia **PersistentVolumeClaims (PVCs)**. Clique em **Create PVC** no canto superior direito. Na caixa de diálogo exibida, configure os parâmetros de PVC.

Parâmetro	Descrição
PVC Type	Neste exemplo, selecione **OBS**.
PVC Name	Digite o nome da PVC, que deve ser exclusivo no mesmo namespace.

Parâmetro	Descrição
Creation Method	– Se o armazenamento subjacente estiver disponível, crie um volume de armazenamento ou use um volume de armazenamento existente para criar estaticamente uma PVC com base em se um PV foi criado. – Se nenhum armazenamento subjacente estiver disponível, selecione **Dynamically provision**. Para mais detalhes, consulte **Uso de um bucket do OBS através de um PV dinâmico**. Neste exemplo, selecione **Create new** para criar um PV e uma PVC ao mesmo tempo no console.
PV[a]	Selecione um volume de PV existente no cluster. Crie um PV com antecedência. Para obter detalhes, consulte "Criação de um volume de armazenamento" em **Operações relacionadas**. Você não precisa especificar esse parâmetro neste exemplo.
OBS[b]	Clique em **Select OBS**. Na página exibida, selecione o bucket do OBS que atende aos seus requisitos e clique em **OK**. **NOTA** Atualmente, apenas sistemas de arquivos paralelos são suportados.
PV Name[b]	Digite o nome do PV, que deve ser exclusivo no mesmo cluster.
Access Mode[b]	Os volumes do OBS suportam apenas **ReadWriteMany**, indicando que um volume de armazenamento pode ser montado em vários nós no modo de leitura/gravação. Para mais detalhes, consulte **Modos de acesso a volume**.
Reclaim Policy[b]	Você pode selecionar **Delete** ou **Retain** para especificar a política de recuperação do armazenamento subjacente quando a PVC é excluída. Para mais detalhes, consulte **Política de recuperação da PV**. **NOTA** Se vários PVs usarem o mesmo volume do OBS, use **Retain** para evitar a exclusão em cascata de volumes subjacentes.
Secret[b]	**Custom**: personalize um segredo se você quiser atribuir permissões de usuário diferentes a diferentes dispositivos de armazenamento do OBS. Para mais detalhes, consulte **Uso de uma chave de acesso (AK/SK) personalizada para montar um volume do OBS**. Somente segredos com o rótulo **secret.kubernetes.io/used-by = csi** podem ser selecionados. O tipo de segredo é cfe/secure-opaque. Se nenhum segredo estiver disponível, clique em **Create Secret** para criar uma. – **Name**: insira um nome do segredo. – **Namespace**: selecione o namespace onde o segredo está. – **Access Key (AK/SK)**: carregue um arquivo de chave no formato .csv. Para mais detalhes, consulte **Obter uma chave de acesso**.

Parâmetro	Descrição
Mount Options[b]	Insira os pares chave-valor do parâmetro de montagem. Para mais detalhes, consulte **Configuração das opções de montagem do OBS**.

NOTA

a: o parâmetro está disponível quando **Creation Method** está definido como **Use existing**.

b: o parâmetro está disponível quando **Creation Method** está definido como **Create new**.

2. Clique em **Create** para criar uma PVC e um PV.

   Você pode escolher **Storage** no painel de navegação e exibir a PVC e o PV criados nas páginas de guia **PersistentVolumeClaims (PVCs)** e **PersistentVolumes (PVs)**, respectivamente.

Passo 3 Crie uma aplicação.

1. No painel de navegação à esquerda, clique em **Workloads**. No painel direito, clique na guia **Deployments**.
2. Clique em **Create Workload** no canto superior direito. Na página exibida, clique em **Data Storage** na área **Container Settings** e clique em **Add Volume** para selecionar **PVC**.

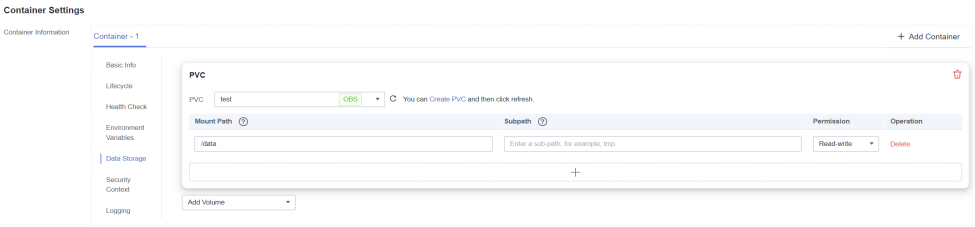
   Monte e use volumes de armazenamento, conforme mostrado na **Tabela 8-29**. Para obter detalhes sobre outros parâmetros, consulte **Cargas de trabalho**.

Tabela 8-29 Montagem de um volume de armazenamento

Parâmetro	Descrição
PVC	Selecione um volume de armazenamento de objetos existente.
Mount Path	Digite um caminho de montagem, por exemplo, **/tmp**. Este parâmetro indica o caminho do contêiner no qual um volume de dados será montado. Não monte o volume em um diretório do sistema como / ou **/var/run**. Caso contrário, os contêineres estarão com defeito. Monte o volume em um diretório vazio. Se o diretório não estiver vazio, verifique se não há arquivos que afetem a inicialização do contêiner. Caso contrário, os arquivos serão substituídos, causando falhas de inicialização do contêiner ou falhas de criação de carga de trabalho. **AVISO** Se um volume for montado em um diretório de alto risco, use uma conta com permissões mínimas para iniciar o contêiner. Caso contrário, arquivos de alto risco no host podem ser danificados.

Parâmetro	Descrição
Subpath	Digite um subcaminho, por exemplo, **tmp**, indicando que os dados no caminho de montagem do contêiner serão armazenados na pasta **tmp** do volume. Um subcaminho é usado para montar um volume local para que o mesmo volume de dados seja usado em um único pod. Se este parâmetro for deixado em branco, o caminho raiz é usado por padrão.
Permission	– **Read-only**: você só pode ler os dados nos volumes montados. – **Read/Write**: você pode modificar os volumes de dados montados no caminho. Os dados recém-gravados não serão migrados se o contêiner for migrado, o que pode causar perda de dados.

Neste exemplo, o disco é montado no caminho **/data** do contêiner. Os dados de contêiner gerados nesse caminho são armazenados no volume do OBS.



3. Após a configuração, clique em **Create Workload**.

   Depois que a carga de trabalho for criada, os dados no diretório de montagem do contêiner serão armazenados persistentemente. Verifique o armazenamento referindo-se a **Política de recuperação da PV**.

----Fim

(kubectl) Usar um bucket do OBS existente

Passo 1 Use o kubectl para se conectar ao cluster.

Passo 2 Crie um PV.

1. Crie o arquivo **pv-obs.yaml**.

```
apiVersion: v1
kind: PersistentVolume
metadata:
  annotations:
    pv.kubernetes.io/provisioned-by: everest-csi-provisioner
    everest.io/reclaim-policy: retain-volume-only      # (Optional) The PV is deleted while the underlying volume is retained.
  name: pv-obs       # PV name.
spec:
  accessModes:
  - ReadWriteMany    # Access mode. The value must be ReadWriteMany for OBS.
  capacity:
    storage: 1Gi     # OBS volume capacity.
  csi:
```



```
    driver: obs.csi.everest.io        # Dependent storage driver for the
mounting.
    driver: obs.csi.everest.io        # Instance type.
    volumeHandle: <your_volume_id>    # Name of the OBS volume.
    volumeAttributes:
      storage.kubernetes.io/csiProvisionerIdentity: everest-csi-provisioner
      everest.io/obs-volume-type: STANDARD
      everest.io/region: <your_region>                        # Region where
the OBS volume is.
      everest.io/enterprise-project-id: <your_project_id>     # (Optional)
Enterprise project ID. If an enterprise project is specified, use the same
enterprise project when creating a PVC. Otherwise, the PVC cannot be bound to
a PV.
    nodePublishSecretRef:             # Custom secret of the OBS volume.
      name: <your_secret_name>        # Custom secret name.
      namespace: <your_namespace>     # Namespace of the custom secret.
  persistentVolumeReclaimPolicy: Retain    # Reclaim policy.
  storageClassName: csi-obs                # Storage class name.
  mountOptions: []                         # Mount options.
```

Tabela 8-30 Parâmetros principais

Parâmetro	Obrigatório	Descrição
everest.io/reclaim-policy: retain-volume-only	Não	Opcional. Atualmente, apenas **retain-volume-only** é suportado. Este campo é válido somente quando a versão do everest for 1.2.9 ou posterior e a política de recuperação for **Delete**. Se a política de recuperação for **Delete** e o valor atual for **retain-volume-only**, o PV associado será excluído enquanto o volume de armazenamento subjacente for retido, quando uma PVC for excluída.
fsType	Sim	Tipos de instância. O valor pode ser **obsfs** ou **s3fs**. – **obsfs**: sistema de arquivos paralelo, que é montado usando obsfs (recomendado). – **s3fs**: bucket de objetos, que é montado usando s3fs.
volumeHandle	Sim	Nome do volume do OBS.
everest.io/obs-volume-type	Sim	Classe de armazenamento do OBS. – Se **fsType** for definido como **s3fs**, **STANDARD** (bucket padrão) e **WARM** (bucket de acesso infrequente) serão suportados. – Este parâmetro é inválido quando **fsType** é definido como **obsfs**.
everest.io/region	Sim	Região onde o bucket do OBS é implementado. Para obter detalhes sobre o valor da **region**, consulte **Regiões e pontos de extremidade**.

Parâmetro	Obrigatório	Descrição
everest.io/enterprise-project-id	Não	Opcional. ID do projeto empresarial do OBS. Se um projeto empresarial for especificado, use o mesmo projeto empresarial ao criar uma PVC. Caso contrário, a PVC não pode ser vinculado a um PV. **Como obter**: no console do OBS, escolha **Buckets** ou **Parallel File Systems** no painel de navegação à esquerda. Clique no nome do bucket do OBS para acessar sua página de detalhes. Na área **Basic Information**, localize o projeto empresarial e clique nele para acessar o console do projeto empresarial. Copie o ID correspondente para obter o ID do projeto empresarial ao qual o armazenamento de objetos pertence.

nodePublishSecretRef	Não	Chave de acesso (AK/SK) usada para montar o volume de armazenamento do objeto. Você pode usar a AK/SK para criar um segredo e montá-lo no PV. Para mais detalhes, consulte **Uso de uma chave de acesso (AK/SK) personalizada para montar um volume do OBS**.
		Um exemplo é o seguinte:
		`nodePublishSecretRef:`
		`  name: secret-demo`
		`  namespace: default`

mountOptions	Não	Opções de montagem. Para mais detalhes, consulte **Configuração das opções de montagem do OBS**.

Parâmetro	Obrigatório	Descrição
persistentVolumeReclaimPolicy	Sim	Uma política de recuperação é suportada quando a versão do cluster é ou posterior a 1.19.10 e a versão do everest é ou posterior a 1.2.9. As políticas de recuperação **Delete** e **Retain** são suportadas. Para mais detalhes, consulte **Política de recuperação da PV**. Se vários PVs usarem o mesmo volume do OBS, use **Retain** para evitar a exclusão em cascata de volumes subjacentes. **Delete**: – Se **everest.io/reclaim-policy** não for especificada, ambos os recursos de PV e de armazenamento serão excluídos quando uma PVC for excluída. – Se **everest.io/reclaim-policy** estiver definida para **retain-volume-only**, quando uma PVC for excluída, o PV será excluído, mas os recursos do armazenamento serão retidos. **Retain**: quando uma PVC é excluída, o PV e os recursos de armazenamento subjacentes não são excluídos. Em vez disso, você deve excluir manualmente esses recursos. Depois disso, o PV está no estado **Released** e não pode ser vinculado à PVC novamente.
storage	Sim	Capacidade de armazenamento, em Gi. Para buckets do OBS, esse campo é usado apenas para verificação (não pode estar vazio ou 0). Seu valor é fixado em **1**, e qualquer valor que você definir não terá efeito para buckets do OBS.
storageClassName	Sim	O nome da classe de armazenamento dos volumes do OBS é **csi-obs**.

2. Execute o seguinte comando para criar um PV:
```
kubectl apply -f pv-obs.yaml
```

Passo 3 Crie uma PVC.

1. Crie o arquivo **pvc-obs.yaml**.
```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-obs
  namespace: default
  annotations:
    volume.beta.kubernetes.io/storage-provisioner: everest-csi-provisioner
    everest.io/obs-volume-type: STANDARD
    csi.storage.k8s.io/fstype: obsfs
    csi.storage.k8s.io/node-publish-secret-name: <your_secret_name>  # Custom
secret name.
    csi.storage.k8s.io/node-publish-secret-namespace: <your_namespace>
```



```
# Namespace of the custom secret.
    everest.io/enterprise-project-id: <your_project_id>     # (Optional)
Enterprise project ID. If an enterprise project is specified, use the same
enterprise project when creating a PVC. Otherwise, the PVC cannot be bound to
a PV.
spec:
  accessModes:
  - ReadWriteMany                  # The value must be ReadWriteMany for OBS.
  resources:
    requests:
      storage: 1Gi
  storageClassName: csi-obs       # Storage class name, which must be the
same as that of the PV.
  volumeName: pv-obs    # PV name.
```

Tabela 8-31 Parâmetros principais

Parâmetro	Obrigatório	Descrição
csi.storage.k8s.io/node-publish-secret-name	Não	Nome do segredo personalizado especificado no PV.
csi.storage.k8s.io/node-publish-secret-namespace	Não	Namespace do segredo personalizado especificado no PV.
everest.io/enterprise-project-id	Não	ID do projeto do OBS. **Como obter**: no console do OBS, escolha **Buckets** ou **Parallel File Systems** no painel de navegação à esquerda. Clique no nome do bucket do OBS para acessar sua página de detalhes. Na área **Basic Information**, localize o projeto empresarial e clique nele para acessar o console do projeto empresarial. Copie o ID correspondente para obter o ID do projeto empresarial ao qual o armazenamento de objetos pertence.
storage	Sim	Capacidade solicitada na PVC, em Gi. Para OBS, esse campo é usado apenas para verificação (não pode estar vazio ou 0). Seu valor é fixado em **1**, e qualquer valor que você definir não terá efeito para o OBS.
storageClassName	Sim	Nome da classe de armazenamento, que deve ser o mesmo que a classe de armazenamento do PV em **1**. O nome da classe de armazenamento dos volumes do OBS é **csi-obs**.
volumeName	Sim	Nome do PV, que deve ser o mesmo que o nome do PV em **1**.

2. Execute o seguinte comando para criar uma PVC:
   ```
   kubectl apply -f pvc-obs.yaml
   ```


Passo 4 Crie uma aplicação.

1. Crie um arquivo chamado **web-demo.yaml**. Neste exemplo, o volume do OBS é montado no caminho **/data**.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web-demo
  namespace: default
spec:
  replicas: 2
  selector:
    matchLabels:
      app: web-demo
  template:
    metadata:
      labels:
        app: web-demo
    spec:
      containers:
      - name: container-1
        image: nginx:latest
        volumeMounts:
        - name: pvc-obs-volume    #Volume name, which must be the same as the volume name in the volumes field.
          mountPath: /data  #Location where the storage volume is mounted.
      imagePullSecrets:
        - name: default-secret
      volumes:
        - name: pvc-obs-volume    #Volume name, which can be customized.
          persistentVolumeClaim:
            claimName: pvc-obs    #Name of the created PVC.
```

2. Execute o seguinte comando para criar uma carga de trabalho na qual o volume do OBS está montado:

```
kubectl apply -f web-demo.yaml
```

Depois que a carga de trabalho é criada, você pode tentar **Verificar a persistência e o compartilhamento de dados**.

----Fim

Verificar a persistência e o compartilhamento de dados

Passo 1 Exiba a aplicação implementada e os arquivos.

1. Execute o seguinte comando para exibir o pod criado:

```
kubectl get pod | grep web-demo
```

Saída esperada:

```
web-demo-846b489584-mjhm9   1/1     Running   0             46s
web-demo-846b489584-wvv5s   1/1     Running   0             46s
```

2. Execute os seguintes comandos em sequência para visualizar os arquivos no caminho **/data** dos pods:

```
kubectl exec web-demo-846b489584-mjhm9 -- ls /data
kubectl exec web-demo-846b489584-wvv5s -- ls /data
```

Se nenhum resultado for retornado para ambos os pods, nenhum arquivo existirá no caminho **/data**.

Passo 2 Execute o seguinte comando para criar um arquivo chamado **static** no caminho **/data**:

```
kubectl exec web-demo-846b489584-mjhm9 --  touch /data/static
```

Passo 3 Execute o seguinte comando para exibir os arquivos no caminho **/data**:

```
kubectl exec web-demo-846b489584-mjhm9 -- ls /data
```


Saída esperada:

```
static
```

Passo 4 Verifique a persistência dos dados.

1. Execute o seguinte comando para excluir o pod chamado **web-demo-846b489584-mjhm9**:
   ```
   kubectl delete pod web-demo-846b489584-mjhm9
   ```
   Saída esperada:
   ```
   pod "web-demo-846b489584-mjhm9" deleted
   ```
   Após a eliminação, o controlador de Implementação cria automaticamente uma réplica.
2. Execute o seguinte comando para exibir o pod criado:
   ```
   kubectl get pod | grep web-demo
   ```
   A saída esperada é a seguinte, na qual **web-demo-846b489584-d4d4j** é o pod recém-criado:
   ```
   web-demo-846b489584-d4d4j   1/1     Running   0             110s
   web-demo-846b489584-wvv5s    1/1     Running   0             7m50s
   ```
3. Execute o seguinte comando para verificar se os arquivos no caminho **/data** do pod novo foram modificados:
   ```
   kubectl exec web-demo-846b489584-d4d4j -- ls /data
   ```
   Saída esperada:
   ```
   static
   ```
   Se o arquivo **static** ainda existir, os dados podem ser armazenados persistentemente.

Passo 5 Verifique o compartilhamento de dados.

1. Execute o seguinte comando para exibir o pod criado:
   ```
   kubectl get pod | grep web-demo
   ```
   Saída esperada:
   ```
   web-demo-846b489584-d4d4j   1/1     Running   0             7m
   web-demo-846b489584-wvv5s   1/1     Running   0             13m
   ```
2. Execute o seguinte comando para criar um arquivo chamado **share** no caminho **/data** de qualquer pod: Neste exemplo, selecione o pod chamado **web-demo-846b489584-d4d4j**.
   ```
   kubectl exec web-demo-846b489584-d4d4j --  touch /data/share
   ```
   Verifique os arquivos no caminho **/data** do pod.
   ```
   kubectl exec web-demo-846b489584-d4d4j -- ls /data
   ```
   Saída esperada:
   ```
   share
   static
   ```
3. Verifique se o arquivo **share** existe no caminho **/data** de outro pod (**web-demo-846b489584-wvv5s**) também para verificar o compartilhamento de dados.
   ```
   kubectl exec web-demo-846b489584-wvv5s -- ls /data
   ```
   Saída esperada:
   ```
   share
   static
   ```
   Depois de criar um arquivo no caminho **/data** de um pod, se o arquivo também for criado no caminho **/data** do outro pod, os dois pods compartilharão o mesmo volume.

----Fim

Operações relacionadas

Você também pode executar as operações listadas em **Tabela 8-32**.

Tabela 8-32 Operações relacionadas

Operação	Descrição	Procedimento
Criar um volume de armazenamento (PV)	Crie um PV no console do CCE.	1. Escolha **Storage** no painel de navegação e clique na guia **PersistentVolumes (PVs)**. Clique em **Create Volume** no canto superior direito. Na caixa de diálogo exibida, configure os parâmetros. ● **Volume Type**: selecione **OBS**. ● **OBS**: clique em **Select OBS**. Na página exibida, selecione o armazenamento do OBS que atende aos seus requisitos e clique em **OK**. ● **PV Name**: digite o nome do PV, que deve ser exclusivo no mesmo cluster. ● **Access Mode**: os volumes do SFS suportam apenas **ReadWriteMany**, indicando que um volume de armazenamento pode ser montado em vários nós no modo de leitura/gravação. Para mais detalhes, consulte **Modos de acesso a volume**. ● **Reclaim Policy**: **Delete** ou **Retain**. Para mais detalhes, consulte **Política de recuperação da PV**. **NOTA** Se vários PVs usarem o mesmo volume de armazenamento subjacente, use **Retain** para evitar a exclusão em cascata de volumes subjacentes. ● **Custom**: personalize um segredo se você quiser atribuir permissões de usuário diferentes a diferentes dispositivos de armazenamento do OBS. Para mais detalhes, consulte **Uso de uma chave de acesso (AK/SK) personalizada para montar um volume do OBS**. Somente segredos com o rótulo **secret.kubernetes.io/used-by = csi** podem ser selecionados. O tipo de segredo é cfe/secure-opaque. Se nenhum segredo estiver disponível, clique em **Create Secret** para criar uma. ● **Mount Options**: insira os pares chave-valor do parâmetro de montagem. Para mais detalhes, consulte **Configuração das opções de montagem do OBS**. 2. Clique em **Create**.

Operação	Descrição	Procedimento
Atualizar uma chave de acesso	Atualize a chave de acesso do armazenamento de objetos no console do CCE.	1. Escolha **Storage** no painel de navegação e clique na guia **PersistentVolumeClaims (PVCs)**. Clique em **More** na coluna **Operation** da PVC de destino e selecione **Update Access Key**. 2. Carregue um arquivo de chave no formato .csv. Para mais detalhes, consulte **Obter uma chave de acesso**. Clique em **OK**. **NOTA** Depois que uma chave de acesso global é atualizada, todos os pods montados com o armazenamento de objetos que usam essa chave de acesso podem ser acessados somente após serem reiniciados.
Visualizar eventos	Você pode visualizar nomes de eventos, tipos de eventos, número de ocorrências, eventos do Kubernetes, horário da primeira ocorrência e horário da última ocorrência da PVC ou PV.	1. Escolha **Storage** no painel de navegação e clique na guia **PersistentVolumeClaims (PVCs)** ou **PersistentVolumes (PVs)**. 2. Clique em **View Events** na coluna **Operation** da PVC ou do PV de destino para exibir os eventos gerados em uma hora (os dados do evento são mantidos por uma hora).
Exibir um arquivo YAML	Você pode visualizar, copiar e fazer download dos arquivos YAML de uma PVC ou um PV.	1. Escolha **Storage** no painel de navegação e clique na guia **PersistentVolumeClaims (PVCs)** ou **PersistentVolumes (PVs)**. 2. Clique em **View YAML** na coluna **Operation** da PVC ou PV de destino para exibir ou fazer download do YAML.

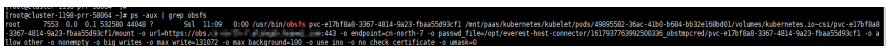
8.6.3 Uso de um bucket do OBS através de um PV dinâmico

Esta seção descreve como criar automaticamente um bucket do OBS. É aplicável quando nenhum volume de armazenamento subjacente estiver disponível.

Restrições

- Se forem utilizados volumes do OBS, o grupo proprietário e a permissão do ponto de montagem não podem ser modificados.
- O CCE permite que sistemas de arquivos paralelos sejam montados usando SDKs ou PVCs do OBS. Se for usada montagem em PVC, a ferramenta obsfs fornecida pelo OBS deve ser usada. Um processo residente de obsfs é gerado cada vez que um volume de armazenamento de objetos gerado a partir de um sistema de arquivos paralelo é montado em um nó.

Figura 8-4 Processo residente do obsfs



Reserve 1 GiB de memória para cada processo de obsfs. Por exemplo, para um nó com 4 vCPUs e 8 GiB de memória, um sistema de arquivos paralelo obsfs deve ser montado em **no máximo** oito pods.

NOTA

- Um processo residente do obsfs é executado em um nó. Se a memória consumida exceder o limite superior do nó, o nó funciona mal. Em um nó com 4 vCPUs e 8 GiB de memória, se mais de 100 pods forem montados em um sistema de arquivos paralelo, o nó não estará disponível. Controle o número de pods montados em um sistema de arquivos paralelo em um único nó.
- Ao usar obsfs, cumpra com as **Restrições de obsfs**.

- Os contêineres de Kata não suportam volumes do OBS.
- O OBS permite que um único usuário crie no máximo 100 buckets. Se um grande número de PVCs dinâmicas for criado, o número de buckets poderá exceder o limite superior e nenhum outro bucket do OBS poderá ser criado. Nesse caso, use o OBS chamando sua API ou SDK e não monte buckets do OBS em cargas de trabalho.

Criar automaticamente um volume do OBS no console

Passo 1 Efetue logon no console do CCE e acesse o console do cluster.

Passo 2 Crie dinamicamente uma PVC e um PV.

1. Escolha **Storage** no painel de navegação e clique na guia **PersistentVolumeClaims (PVCs)**. Clique em **Create PVC** no canto superior direito. Na caixa de diálogo exibida, configure os parâmetros de PVC.

Parâmetro	Descrição
PVC Type	Neste exemplo, selecione **OBS**.
PVC Name	Digite o nome da PVC, que deve ser exclusivo no mesmo namespace.
Creation Method	– Se nenhum armazenamento subjacente estiver disponível, selecione **Dynamically provision** para criar uma PVC, um PV e armazenamento subjacente no console no modo de cascata. – Se o armazenamento subjacente estiver disponível, crie um volume de armazenamento ou use um volume de armazenamento existente para criar estaticamente uma PVC com base em se um PV foi criado. Para mais detalhes, consulte **Uso de um bucket do OBS existente através de um PV estático**. Neste exemplo, selecione **Dynamically provision**.
Storage Classes	A classe de armazenamento dos volumes do OBS é **csi-obs**.
Instance Type	– **Parallel file system**: um sistema de arquivos de alto desempenho fornecido pelo OBS. Ele fornece latência de acesso de milissegundos, largura de banda no nível de TB/s e IOPS no nível de milhões. **Sistemas de arquivos paralelos são recomendados.** – **Object bucket**: um contêiner que armazena objetos no OBS. Todos os objetos em um bucket estão no mesmo nível lógico.

Parâmetro	Descrição
OBS Class	Você pode selecionar os seguintes tipos de bucket de objetos: – **Standard**: aplicável quando um grande número de arquivos de hotspot ou arquivos de pequeno porte precisam ser acessados com frequência (várias vezes por mês, em média) e exigem resposta de acesso rápida. – **Infrequent access**: aplicável quando os dados não são acessados com frequência (menos de 12 vezes por ano, em média), mas exigem resposta de acesso rápido.
Access Mode	Os volumes do OBS suportam apenas **ReadWriteMany**, indicando que um volume de armazenamento pode ser montado em vários nós no modo de leitura/gravação. Para mais detalhes, consulte **Modos de acesso a volume**.
Secret	**Custom**: personalize um segredo se você quiser atribuir permissões de usuário diferentes a diferentes dispositivos de armazenamento do OBS. Para mais detalhes, consulte **Uso de uma chave de acesso (AK/SK) personalizada para montar um volume do OBS**. Somente segredos com o rótulo **secret.kubernetes.io/used-by = csi** podem ser selecionados. O tipo de segredo é cfe/secure-opaque. Se nenhum segredo estiver disponível, clique em **Create Secret** para criar um. – **Name**: insira um nome de segredo. – **Namespace**: selecione o namespace onde o segredo está. – **Access Key (AK/SK)**: carregue um arquivo de chave no formato .csv. Para mais detalhes, consulte **Obter uma chave de acesso**.
Enterprise Project	Projetos empresariais suportados: padrão, aquele ao qual o cluster pertence ou aquele especificado pela classe de armazenamento.

2. Clique em **Create** para criar uma PVC e um PV.

   Você pode escolher **Storage** no painel de navegação e exibir a PVC e o PV criados nas páginas de guia **PersistentVolumeClaims (PVCs)** e **PersistentVolumes (PVs)**, respectivamente.

Passo 3 Crie uma aplicação.

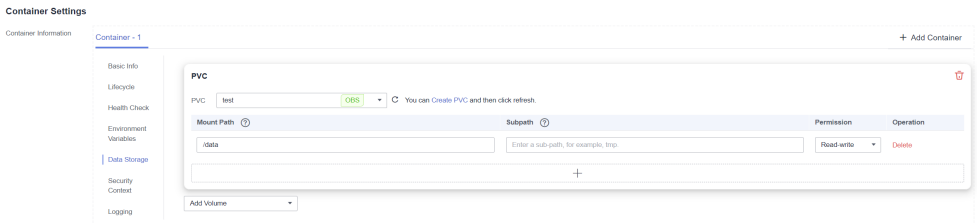
1. No painel de navegação à esquerda, clique em **Workloads**. No painel direito, clique na guia **Deployments**.
2. Clique em **Create Workload** no canto superior direito. Na página exibida, clique em **Data Storage** na área **Container Settings** e clique em **Add Volume** para selecionar **PVC**.

   Monte e use volumes de armazenamento, conforme mostrado na **Tabela 8-33**. Para obter detalhes sobre outros parâmetros, consulte **Cargas de trabalho**.

Tabela 8-33 Montagem de um volume de armazenamento

Parâmetro	Descrição
PVC	Selecione um volume de armazenamento de objetos existente.
Mount Path	Digite um caminho de montagem, por exemplo, **/tmp**. Este parâmetro indica o caminho do contêiner no qual um volume de dados será montado. Não monte o volume em um diretório do sistema como / ou **/var/run**. Caso contrário, os contêineres estarão com defeito. Monte o volume em um diretório vazio. Se o diretório não estiver vazio, verifique se não há arquivos que afetem a inicialização do contêiner. Caso contrário, os arquivos serão substituídos, causando falhas de inicialização do contêiner ou falhas de criação de carga de trabalho. **AVISO** Se um volume for montado em um diretório de alto risco, use uma conta com permissões mínimas para iniciar o contêiner. Caso contrário, arquivos de alto risco no host podem ser danificados.
Subpath	Digite um subcaminho, por exemplo, **tmp**, indicando que os dados no caminho de montagem do contêiner serão armazenados na pasta **tmp** do volume. Um subcaminho é usado para montar um volume local para que o mesmo volume de dados seja usado em um único pod. Se este parâmetro for deixado em branco, o caminho raiz é usado por padrão.
Permission	– **Read-only**: você só pode ler os dados nos volumes montados. – **Read/Write**: você pode modificar os volumes de dados montados no caminho. Os dados recém-gravados não serão migrados se o contêiner for migrado, o que pode causar perda de dados.

Neste exemplo, o disco é montado no caminho **/data** do contêiner. Os dados de contêiner gerados nesse caminho são armazenados no volume do OBS.



3. Após a configuração, clique em **Create Workload**.

   Depois que a carga de trabalho for criada, os dados no diretório de montagem do contêiner serão armazenados persistentemente. Verifique o armazenamento referindo-se a **Política de recuperação da PV**.

----Fim

(kubectl) Criação automática de um volume do OBS

Passo 1 Use o kubectl para se conectar ao cluster.

Passo 2 Use **StorageClass** para criar dinamicamente uma PVC e um PV.

1. Crie o arquivo **pvc-obs-auto.yaml**.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-obs-auto
  namespace: default
  annotations:
    everest.io/obs-volume-type: STANDARD    # Object storage type.
    csi.storage.k8s.io/fstype: obsfs        # Instance type.
    csi.storage.k8s.io/node-publish-secret-name: <your_secret_name>       #
Custom secret name.
    csi.storage.k8s.io/node-publish-secret-namespace: <your_namespace>    #
Namespace of the custom secret.
    everest.io/enterprise-project-id: <your_project_id>     # (Optional)
Enterprise project ID. If an enterprise project is specified, use the same
enterprise project when creating a PVC. Otherwise, the PVC cannot be bound to
a PV.
spec:
  accessModes:
    - ReadWriteMany             # The value must be ReadWriteMany for object
storage.
  resources:
    requests:
      storage: 1Gi               # OBS volume capacity.
  storageClassName: csi-obs    # The storage class type is OBS.
```

Tabela 8-34 Parâmetros principais

Parâmetro	Obrigatório	Descrição
everest.io/obs-volume-type	Sim	Classe de armazenamento do OBS. – Se **fsType** for definido como **s3fs**, **STANDARD** (bucket padrão) e **WARM** (bucket de acesso infrequente) serão suportados. – Este parâmetro é inválido quando **fsType** é definido como **obsfs**.
csi.storage.k8s.io/fstype	Sim	Tipos de instância. O valor pode ser **obsfs** ou **s3fs**. – **obsfs**: sistema de arquivos paralelo, que é montado usando obsfs (recomendado). – **s3fs**: bucket de objetos, que é montado usando s3fs.

Parâmetro	Obrigatório	Descrição
csi.storage.k8s.io/node-publish-secret-name	Não	Nome de segredo personalizado. (Recomendado) Selecione esta opção se quiser atribuir permissões de usuário diferentes a diferentes dispositivos de armazenamento do OBS. Para mais detalhes, consulte **Uso de uma chave de acesso (AK/SK) personalizada para montar um volume do OBS**.
csi.storage.k8s.io/node-publish-secret-namespace	Não	Namespace de um segredo personalizado.
everest.io/enterprise-project-id	Não	ID do projeto do OBS. Para obter um ID de projeto empresarial, efetue logon no console do EPS, clique no nome do projeto empresarial de destino e copie o ID do projeto empresarial.
storage	Sim	Capacidade solicitada na PVC, em Gi. Para buckets do OBS, esse campo é usado apenas para verificação (não pode estar vazio ou 0). Seu valor é fixado em **1**, e qualquer valor que você definir não terá efeito para buckets do OBS.
storageClassName	Sim	Nome da classe de armazenamento. O nome da classe de armazenamento dos volumes do OBS é **csi-obs**.

2. Execute o seguinte comando para criar uma PVC:
```
kubectl apply -f pvc-obs-auto.yaml
```

Passo 3 Crie uma aplicação.

1. Crie um arquivo chamado **web-demo.yaml**. Neste exemplo, o volume do OBS é montado no caminho **/data**.
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web-demo
  namespace: default
spec:
  replicas: 2
  selector:
    matchLabels:
      app: web-demo
  template:
    metadata:
      labels:
        app: web-demo
    spec:
      containers:
      - name: container-1
        image: nginx:latest
        volumeMounts:
```



```
        - name: pvc-obs-volume    #Volume name, which must be the same as the
volume name in the volumes field.
          mountPath: /data  #Location where the storage volume is mounted.
      imagePullSecrets:
        - name: default-secret
      volumes:
        - name: pvc-obs-volume    #Volume name, which can be customized.
          persistentVolumeClaim:
            claimName: pvc-obs-auto    #Name of the created PVC.
```

2. Execute o seguinte comando para criar uma carga de trabalho na qual o volume do OBS está montado:
   ```
   kubectl apply -f web-demo.yaml
   ```
   Depois que a carga de trabalho é criada, você pode tentar **Verificar a persistência e o compartilhamento de dados**.

----Fim

Verificar a persistência e o compartilhamento de dados

Passo 1 Exiba a aplicação implementada e os arquivos.

1. Execute o seguinte comando para exibir o pod criado:
   ```
   kubectl get pod | grep web-demo
   ```
   Saída esperada:
   ```
   web-demo-846b489584-mjhm9   1/1     Running   0             46s
   web-demo-846b489584-wvv5s   1/1     Running   0             46s
   ```
2. Execute os seguintes comandos em sequência para visualizar os arquivos no caminho **/data** dos pods:
   ```
   kubectl exec web-demo-846b489584-mjhm9 -- ls /data
   kubectl exec web-demo-846b489584-wvv5s -- ls /data
   ```
   Se nenhum resultado for retornado para ambos os pods, nenhum arquivo existirá no caminho **/data**.

Passo 2 Execute o seguinte comando para criar um arquivo chamado **static** no caminho **/data**:
```
kubectl exec web-demo-846b489584-mjhm9 --  touch /data/static
```

Passo 3 Execute o seguinte comando para exibir os arquivos no caminho **/data**:
```
kubectl exec web-demo-846b489584-mjhm9 -- ls /data
```
Saída esperada:
```
static
```

Passo 4 Verifique a persistência dos dados.

1. Execute o seguinte comando para excluir o pod chamado **web-demo-846b489584-mjhm9**:
   ```
   kubectl delete pod web-demo-846b489584-mjhm9
   ```
   Saída esperada:
   ```
   pod "web-demo-846b489584-mjhm9" deleted
   ```
   Após a eliminação, o controlador de Implementação cria automaticamente uma réplica.
2. Execute o seguinte comando para exibir o pod criado:
   ```
   kubectl get pod | grep web-demo
   ```
   A saída esperada é a seguinte, na qual **web-demo-846b489584-d4d4j** é o pod recém-criado:
   ```
   web-demo-846b489584-d4d4j   1/1     Running   0             110s
   web-demo-846b489584-wvv5s    1/1     Running   0             7m50s
   ```


3. Execute o seguinte comando para verificar se os arquivos no caminho **/data** do pod novo foram modificados:
   ```
   kubectl exec web-demo-846b489584-d4d4j -- ls /data
   ```
   Saída esperada:
   ```
   static
   ```
   Se o arquivo **static** ainda existir, os dados podem ser armazenados persistentemente.

Passo 5 Verifique o compartilhamento de dados.

1. Execute o seguinte comando para exibir o pod criado:
   ```
   kubectl get pod | grep web-demo
   ```
   Saída esperada:
   ```
   web-demo-846b489584-d4d4j   1/1     Running   0             7m
   web-demo-846b489584-wvv5s   1/1     Running   0             13m
   ```
2. Execute o seguinte comando para criar um arquivo chamado **share** no caminho **/data** de qualquer pod: Neste exemplo, selecione o pod chamado **web-demo-846b489584-d4d4j**.
   ```
   kubectl exec web-demo-846b489584-d4d4j --  touch /data/share
   ```
   Verifique os arquivos no caminho **/data** do pod.
   ```
   kubectl exec web-demo-846b489584-d4d4j -- ls /data
   ```
   Saída esperada:
   ```
   share
   static
   ```
3. Verifique se o arquivo **share** existe no caminho **/data** de outro pod (**web-demo-846b489584-wvv5s**) também para verificar o compartilhamento de dados.
   ```
   kubectl exec web-demo-846b489584-wvv5s -- ls /data
   ```
   Saída esperada:
   ```
   share
   static
   ```
   Depois de criar um arquivo no caminho **/data** de um pod, se o arquivo também for criado no caminho **/data** do outro pod, os dois pods compartilharão o mesmo volume.

----Fim

Operações relacionadas

Você também pode executar as operações listadas em **Tabela 8-35**.

Tabela 8-35 Operações relacionadas

Operação	Descrição	Procedimento
Atualizar uma chave de acesso	Atualize a chave de acesso do armazenamento de objetos no console do CCE.	1. Escolha **Storage** no painel de navegação e clique na guia **PersistentVolumeClaims (PVCs)**. Clique em **More** na coluna **Operation** da PVC de destino e selecione **Update Access Key**. 2. Carregue um arquivo de chave no formato .csv. Para mais detalhes, consulte **Obter uma chave de acesso**. Clique em **OK**. **NOTA** Depois que uma chave de acesso global é atualizada, todos os pods montados com o armazenamento de objetos que usam essa chave de acesso podem ser acessados somente após serem reiniciados.
Visualizar eventos	Você pode visualizar nomes de eventos, tipos de eventos, número de ocorrências, eventos do Kubernetes, horário da primeira ocorrência e horário da última ocorrência da PVC ou PV.	1. Escolha **Storage** no painel de navegação e clique na guia **PersistentVolumeClaims (PVCs)** ou **PersistentVolumes (PVs)**. 2. Clique em **View Events** na coluna **Operation** da PVC ou do PV de destino para exibir os eventos gerados em uma hora (os dados do evento são mantidos por uma hora).
Exibir um arquivo YAML	Você pode visualizar, copiar e fazer download dos arquivos YAML de uma PVC ou um PV.	1. Escolha **Storage** no painel de navegação e clique na guia **PersistentVolumeClaims (PVCs)** ou **PersistentVolumes (PVs)**. 2. Clique em **View YAML** na coluna **Operation** da PVC ou PV de destino para exibir ou fazer download do YAML.

8.6.4 Configuração das opções de montagem do OBS

Esta seção descreve como configurar as opções de montagem de volume do OBS. Você pode configurar opções de montagem em um PV e vincular o PV a uma PVC. Como alternativa, configure as opções de montagem em uma StorageClass e use a StorageClass para criar uma PVC. Dessa forma, os PVs podem ser criados dinamicamente e herdar opções de montagem configuradas na StorageClass por padrão.

Pré-requisitos

A versão do complemento **Armazenamento do contêiner do CCE (Everest)** deve ser **1.2.8 ou posterior**. Esse complemento identifica as opções de montagem e as transfere para os recursos de armazenamento subjacentes. As configurações de parâmetros terão efeito somente se os recursos de armazenamento subjacentes suportarem as opções especificadas.

Restrições

As opções de montagem não podem ser configuradas para contêineres do Kata.

Opções de montagem do OBS

Ao montar um volume do OBS, o complemento everest pré-ajusta as opções descritas em **Tabela 8-36** e **Tabela 8-37** por padrão. As opções em **Tabela 8-36** são obrigatórias. Você pode definir outras opções de montagem, se necessário. Para obter detalhes, consulte **Montagem de um sistema de arquivos paralelo**.

Tabela 8-36 Opções de montagem obrigatórias configuradas por padrão

Parâmetro	Valor	Descrição
use_ino	Deixe em branco.	Se ativado, obsfs aloca o número do **inode**. Ativado por padrão no modo de leitura/gravação.
big_writes	Deixe em branco.	Se configurado, o tamanho máximo do cache pode ser modificado.
nonempty	Deixe em branco.	Permite caminhos de montagem não vazios.
allow_other	Deixe em branco.	Permite que outros usuários acessem o sistema de arquivos paralelo.
no_check_certificate	Deixe em branco.	Desativa a verificação do certificado do servidor.
enable_noobj_cache	Deixe em branco.	Ativa entradas de cache para objetos que não existem, o que pode melhorar o desempenho. Ativado por padrão no modo de leitura/gravação do intervalo de objetos. **Esta opção não está mais configurada por padrão desde o everest 1.2.40.**
sigv2	Deixe em branco.	Especifica a versão de assinatura. Usado por padrão em buckets de objetos.

Tabela 8-37 Opções de montagem opcionais configuradas por padrão

Parâmetro	Valor	Descrição
max_write	131072	Este parâmetro é válido somente quando **big_writes** é configurado. O valor recomendado é de **128 KB**.
ssl_verify_hostname	0	Desativa a verificação do certificado SSL com base no nome do host.
max_background	100	Permite definir o número máximo de solicitações em espera em segundo plano. Usado por padrão em sistemas de arquivos paralelos.
public_bucket	1	Se definido como **1**, os buckets públicos são montados anonimamente. Ativado por padrão no modo de leitura/gravação do intervalo de objetos.
umask	Deixe em branco.	Máscara da permissão do arquivo de configuração.

Configurar opções de montagem em um PV

Você pode usar o campo **mountOptions** para configurar opções de montagem em um PV. As opções que você pode configurar em **mountOptions** estão listadas em **Opções de montagem do OBS**.

Passo 1 Use o kubectl para se conectar ao cluster. Para mais detalhes, consulte **Conexão a um cluster usando o kubectl**.

Passo 2 Configurar opções de montagem em um PV. Exemplo:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  annotations:
    pv.kubernetes.io/provisioned-by: everest-csi-provisioner
    everest.io/reclaim-policy: retain-volume-only      # (Optional) The PV is
deleted while the underlying volume is retained.
  name: pv-obs       # PV name.
spec:
  accessModes:
  - ReadWriteMany    # Access mode. The value must be ReadWriteMany for OBS.
  capacity:
    storage: 1Gi     # OBS volume capacity.
  csi:
    driver: obs.csi.everest.io        # Dependent storage driver for the mounting.
    fsType: obsfs                     # Instance type.
    volumeHandle: <your_volume_id>    # Name of the OBS volume.
    volumeAttributes:
      storage.kubernetes.io/csiProvisionerIdentity: everest-csi-provisioner
      everest.io/obs-volume-type: STANDARD
      everest.io/region: <your_region>                        # Region where the
OBS volume is.
      everest.io/enterprise-project-id: <your_project_id>     # (Optional)
Enterprise project ID. If an enterprise project is specified, use the same
enterprise project when creating a PVC. Otherwise, the PVC cannot be bound to a
PV.
    nodePublishSecretRef:            # Custom secret of the OBS volume.
```



```
    name: <your_secret_name>       # Custom secret name.
    namespace: <your_namespace>    # Namespace of the custom secret.
persistentVolumeReclaimPolicy: Retain    # Reclaim policy.
storageClassName: csi-obs                # Storage class name.
mountOptions:                             # Mount options.
- umask=0027
```

Passo 3 Depois que um PV é criado, você pode criar uma PVC e vinculá-la ao PV e, em seguida, montar o PV ao contêiner na carga de trabalho. Para mais detalhes, consulte **Uso de um bucket do OBS existente através de um PV estático**.

Passo 4 Verifique se as opções de montagem têm efeito.

Neste exemplo, a PVC é montada na carga de trabalho que usa a imagem **nginx:latest**. Você pode fazer logon no nó onde reside o pod no qual o volume do OBS é montado e exibir os detalhes do andamento.

Execute o seguinte comando:

- Bucket do objeto: **ps -ef | grep s3fs**

```
root     22142     1  0 Jun03 ?        00:00:00 /usr/bin/s3fs
{your_obs_name} /mnt/paas/kubernetes/kubelet/pods/{pod_uid}/volumes/
kubernetes.io~csi/{your_pv_name}/mount -o url=https://{endpoint}:443 -o
endpoint={region} -o passwd_file=/opt/everest-host-connector/***_obstmpcred/
{your_obs_name} -o nonempty -o big_writes -o sigv2 -o allow_other -o
no_check_certificate -o ssl_verify_hostname=0 -o umask=0027 -o
max_write=131072 -o multipart_size=20
```

- Sistema de arquivos paralelo: **ps -ef | grep obsfs**

```
root      1355     1  0 Jun03 ?        00:03:16 /usr/bin/obsfs
{your_obs_name} /mnt/paas/kubernetes/kubelet/pods/{pod_uid}/volumes/
kubernetes.io~csi/{your_pv_name}/mount -o url=https://{endpoint}:443 -o
endpoint={region} -o passwd_file=/opt/everest-host-connector/***_obstmpcred/
{your_obs_name} -o allow_other -o nonempty -o big_writes -o use_ino -o
no_check_certificate -o ssl_verify_hostname=0 -o max_background=100 -o
umask=0027 -o max_write=131072
```

----Fim

Configurar opções de montagem em uma StorageClass

Você pode usar o campo **mountOptions** para configurar opções de montagem em uma StorageClass. As opções que você pode configurar em **mountOptions** estão listadas em **Opções de montagem do OBS**.

Passo 1 Use o kubectl para se conectar ao cluster. Para mais detalhes, consulte **Conexão a um cluster usando o kubectl**.

Passo 2 Crie uma StorageClass personalizada. Exemplo:

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: csi-obs-mount-option
provisioner: everest-csi-provisioner
parameters:
  csi.storage.k8s.io/csi-driver-name: obs.csi.everest.io
  csi.storage.k8s.io/fstype: s3fs
  everest.io/obs-volume-type: STANDARD
reclaimPolicy: Delete
volumeBindingMode: Immediate
mountOptions:                            # Mount options.
- umask=0027
```


Passo 3 Depois que a StorageClass é configurada, você pode usá-la para criar uma PVC. Por padrão, os PVs criados dinamicamente herdam as opções de montagem configuradas no StorageClass. Para mais detalhes, consulte **Uso de um bucket do OBS através de um PV dinâmico**.

Passo 4 Verifique se as opções de montagem têm efeito.

Neste exemplo, a PVC é montada na carga de trabalho que usa a imagem **nginx:latest**. Você pode fazer logon no nó onde reside o pod no qual o volume do OBS é montado e exibir os detalhes do andamento.

Execute o seguinte comando:

- Bucket do objeto: **ps -ef | grep s3fs**

```
root     22142     1  0 Jun03 ?        00:00:00 /usr/bin/s3fs
{your_obs_name} /mnt/paas/kubernetes/kubelet/pods/{pod_uid}/volumes/
kubernetes.io~csi/{your_pv_name}/mount -o url=https://{endpoint}:443 -o
endpoint={region} -o passwd_file=/opt/everest-host-connector/***_obstmpcred/
{your_obs_name} -o nonempty -o big_writes -o sigv2 -o allow_other -o
no_check_certificate -o ssl_verify_hostname=0 -o umask=0027 -o
max_write=131072 -o multipart_size=20
```

- Sistema de arquivos paralelo: **ps -ef | grep obsfs**

```
root      1355     1  0 Jun03 ?        00:03:16 /usr/bin/obsfs
{your_obs_name} /mnt/paas/kubernetes/kubelet/pods/{pod_uid}/volumes/
kubernetes.io~csi/{your_pv_name}/mount -o url=https://{endpoint}:443 -o
endpoint={region} -o passwd_file=/opt/everest-host-connector/***_obstmpcred/
{your_obs_name} -o allow_other -o nonempty -o big_writes -o use_ino -o
no_check_certificate -o ssl_verify_hostname=0 -o max_background=100 -o
umask=0027 -o max_write=131072
```

----Fim

8.6.5 Uso de uma chave de acesso (AK/SK) personalizada para montar um volume do OBS

Cenário

Armazenamento do contêiner do CCE (Everest) da versão 1.2.8 ou posterior suporta chaves de acesso personalizadas. Dessa forma, os usuários do IAM podem usar suas próprias chaves de acesso personalizadas para montar um volume do OBS. Para obter detalhes, consulte **Como controlar o acesso ao OBS?**

Pré-requisitos

- A versão do complemento **Armazenamento do contêiner do CCE (Everest)** deve ser 1.2.8 ou posterior.
- A versão do cluster deve ser 1.15.11 ou posterior.

Restrições

- Quando um volume do OBS é montado usando uma chave de acesso (AK/SK) personalizada, a chave de acesso não pode ser excluída ou desabilitada. Caso contrário, o contêiner de serviço não pode acessar o volume do OBS montado.
- Teclas de acesso personalizadas não podem ser configuradas para contêineres de Kata.

Desativar a montagem automática da chave

A chave que você carregou é usada por padrão ao montar um volume do OBS. Ou seja, todos os usuários do IAM em sua conta usarão a mesma chave para montar buckets do OBS e terão

as mesmas permissões em buckets. Essa configuração não permite que você configure permissões diferenciadas para diferentes usuários do IAM.

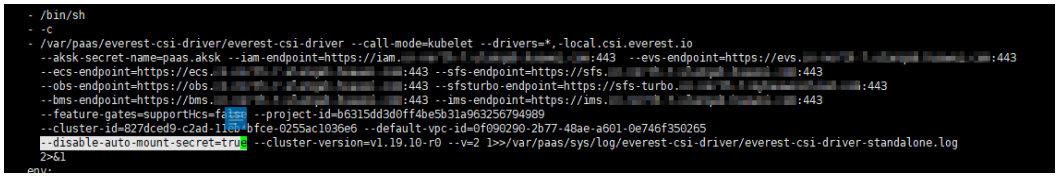
Se você carregou a AK/SK, desative a montagem automática de chaves de acesso ativando o parâmetro **disable_auto_mount_secret** no complemento everest para impedir que os usuários do IAM executem operações não autorizadas. Desta forma, as teclas de acesso carregadas no console não serão usadas ao criar volumes do OBS.

NOTA

- Ao ativar **disable-auto-mount-secret**, certifique-se de que não exista nenhum volume do OBS no cluster. Uma carga de trabalho montada com um volume do OBS, quando dimensionada ou reiniciada, não conseguirá remontar o volume do OBS porque precisa especificar a chave de acesso, mas é proibida por **disable-auto-mount-secret**.
- Se **disable-auto-mount-secret** estiver definido como **true**, uma chave de acesso deve ser especificada quando um PV ou PVC for criado. Caso contrário, o volume do OBS não será montado.

kubectl edit ds everest-csi-driver -nkube-system

Pesquise **disable-auto-mount-secret** e defina-o como **true**.



Execute **:wq** para salvar as configurações e sair. Aguarde até que o pod seja reiniciado.

Obter uma chave de acesso

Passo 1 Faça logon no console.

Passo 2 Passe o cursor sobre o nome de usuário no canto superior direito e escolha **My Credentials** na lista suspensa.

Passo 3 No painel de navegação, escolha **Access Keys**.

Passo 4 Clique em **Create Access Key**. A caixa de diálogo **Create Access Key** é exibida.

Passo 5 Clique em **OK** para baixar a chave de acesso.

----Fim

Criar um segredo usando uma chave de acesso

Passo 1 Obtenha uma chave de acesso.

Passo 2 Codifique as chaves usando Base64. (Suponha que o AK é xxx e a SK é yyy.)

echo -n xxx|base64

echo -n yyy|base64

Grave o AK e a SK codificados.

Passo 3 Crie um arquivo YAML para o segredo, por exemplo, **test-user.yaml**.

```
apiVersion: v1
data:
```



```
  access.key: WE5WWVhVNU*****
  secret.key: Nnk4emJyZ0*****
kind: Secret
metadata:
  name: test-user
  namespace: default
  labels:
    secret.kubernetes.io/used-by: csi
type: cfe/secure-opaque
```

Especificamente:

Parâmetro	Descrição
access.key	AK codificado em Base64.
secret.key	SK codificada em Base64.
name	Nome de segredo.
namespace	Namespace do segredo.
secret.kubernetes.io/used-by: csi	Adicione este rótulo no arquivo YAML se quiser disponibilizá-lo no console do CCE ao criar um PV/PVC do OBS.
type	Tipo de segredo. O valor deve ser **cfe/secure-opaque**. Quando este tipo é usado, os dados inseridos pelos usuários são automaticamente criptografados.

Passo 4 Crie o segredo.

kubectl create -f test-user.yaml

----Fim

Montar um segredo ao criar estaticamente um volume do OBS

Depois que um segredo é criado usando a AK/SK, você pode associar o segredo com o PV a ser criado e, em seguida, usar a AK/SK no segredo para montar um volume do OBS.

Passo 1 Faça logon no console do OBS, crie um bucket do OBS e registre o nome do bucket e a classe de armazenamento. O sistema de arquivos paralelo é usado como exemplo.

Passo 2 Crie um arquivo YAML para o PV, por exemplo, **pv-example.yaml**.

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-obs-example
  annotations:
    pv.kubernetes.io/provisioned-by: everest-csi-provisioner
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 1Gi
  csi:
    nodePublishSecretRef:
      name: test-user
      namespace: default
    driver: obs.csi.everest.io
```



```
    fsType: obsfs
    volumeAttributes:
      everest.io/obs-volume-type: STANDARD
      everest.io/region: ap-southeast-1
      storage.kubernetes.io/csiProvisionerIdentity: everest-csi-provisioner
    volumeHandle: obs-normal-static-pv
  persistentVolumeReclaimPolicy: Delete
  storageClassName: csi-obs
```

Parâmetro	Descrição
nodePublishSecretRef	Segredo especificado durante a montagem. ● **name**: nome do segredo ● **namespace**: namespace do segredo
fsType	Tipo de arquivo. O valor pode ser **obsfs** ou **s3fs**. Se o valor for **s3fs**, um bucket do OBS será criado e montado usando s3fs. Se o valor for **obsfs**, um sistema de arquivos paralelo do OBS será criado e montado usando obsfs. É aconselhável definir este campo para **obsfs**.
volumeHandle	Nome do bucket do OBS.

Passo 3 Crie um PV.

kubectl create -f pv-example.yaml

Depois que um PV é criado, você pode criar uma PVC e associá-lo ao PV.

Passo 4 Crie um arquivo YAML para o PVC, por exemplo, **pvc-example.yaml**.

Exemplo de arquivo YAML para a PVC:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    csi.storage.k8s.io/node-publish-secret-name: test-user
    csi.storage.k8s.io/node-publish-secret-namespace: default
    volume.beta.kubernetes.io/storage-provisioner: everest-csi-provisioner
    everest.io/obs-volume-type: STANDARD
    csi.storage.k8s.io/fstype: obsfs
  name: obs-secret
  namespace: default
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: csi-obs
  volumeName: pv-obs-example
```

Parâmetro	Descrição
csi.storage.k8s.io/node-publish-secret-name	Nome do segredo
csi.storage.k8s.io/node-publish-secret-namespace	Namespace do segredo

Passo 5 Crie uma PVC.

kubectl create -f pvc-example.yaml

Depois que a PVC é criada, você pode criar uma carga de trabalho e associá-la à PVC para criar volumes.

----Fim

Montar um segredo ao criar dinamicamente um volume do OBS

Ao criar dinamicamente um volume do OBS, você pode usar o seguinte método para especificar um segredo:

Passo 1 Crie um arquivo YAML para a PVC, por exemplo, **pvc-example.yaml**.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    csi.storage.k8s.io/node-publish-secret-name: test-user
    csi.storage.k8s.io/node-publish-secret-namespace: default
    everest.io/obs-volume-type: STANDARD
    csi.storage.k8s.io/fstype: obsfs
  name: obs-secret
  namespace: default
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: csi-obs
```

Parâmetro	Descrição
csi.storage.k8s.io/node-publish-secret-name	Nome do segredo
csi.storage.k8s.io/node-publish-secret-namespace	Namespace do segredo

Passo 2 Crie uma PVC.

kubectl create -f pvc-example.yaml

Depois que a PVC é criada, você pode criar uma carga de trabalho e associá-la à PVC para criar volumes.

----Fim

Verificação

Você pode usar um segredo de um usuário do IAM para montar um volume do OBS. Suponha que uma carga de trabalho denominada **obs-secret** seja criada, o caminho de montagem no contêiner seja **/temp** e o usuário do IAM tenha as permissões **ReadOnlyAccess** e **Tenant Guest**.

1. Consulte o nome do pod da carga de trabalho.

   kubectl get po | grep obs-secret

Saídas esperadas:

```
obs-secret-5cd558f76f-vxslv          1/1     Running   0          3m22s
```

2. Consulte os objetos no caminho de montagem. Neste exemplo, a consulta foi bem-sucedida.

   kubectl exec obs-secret-5cd558f76f-vxslv -- ls -l /temp/

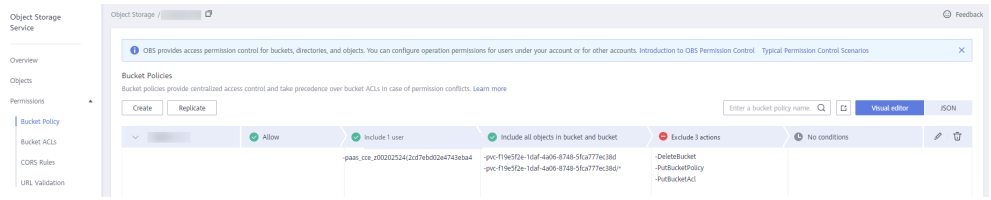
3. Grave dados no caminho de montagem. Neste exemplo, a operação de gravação falhou.

   kubectl exec obs-secret-5cd558f76f-vxslv -- touch /temp/test

   Saídas esperadas:

   ```
   touch: setting times of '/temp/test': No such file or directory
   command terminated with exit code 1
   ```

4. Defina as permissões de leitura/gravação para o usuário do IAM que montou o volume do OBS referindo-se à configuração da política de bucket.



5. Grave dados no caminho de montagem novamente. Neste exemplo, a operação de gravação foi bem-sucedida.

   kubectl exec obs-secret-5cd558f76f-vxslv -- touch /temp/test

6. Verifique o caminho de montagem no contêiner para ver se os dados foram gravados com êxito.

   kubectl exec obs-secret-5cd558f76f-vxslv -- ls -l /temp/

   Saídas esperadas:

   ```
   -rwxrwxrwx 1 root root 0 Jun  7 01:52 test
   ```

8.6.6 Uso de buckets do OBS entre regiões

Por padrão, um pod pode usar buckets do OBS somente na mesma região. O CCE permite que uma carga de trabalho use buckets do OBS entre regiões, o que pode melhorar a utilização de recursos em alguns cenários, mas também pode resultar em uma latência mais alta.

Restrições

- O complemento **Armazenamento do contêiner do CCE (Everest)** deve ser de **1.2.42 ou posterior**.
- O nó no qual o armazenamento é montado deve ser capaz de acessar os buckets do OBS. Geralmente, a Internet ou Direct Connect são usadas para acessar os buckets do OBS em todas as regiões. Você pode executar ping no ponto de extremidade do OBS no nó onde o OBS está localizado para verificar se o OBS está acessível.
- Somente os PVs podem usar buckets do OBS em regiões e, em seguida, são vinculadas às PVCs. A política de recuperação de PV deve ser **Retain**. As classes de armazenamento não podem ser usadas para criar PVCs dinamicamente para usar buckets do OBS entre regiões.

Procedimento

Passo 1 Crie o ConfigMap paas-obs-endpoint e configure a região e o ponto de extremidade do OBS.

O nome do ConfigMap é fixado em **paas-obs-endpoint** e o namespace é fixado para **kube-system**.

Os nomes de região e os pontos de extremidade estão em pares chave-valor. Substitua *<region_name>* e *<endpoint_address>* por valores específicos. Use vírgulas (,) para separar vários valores.

Para obter detalhes sobre o valor da **region**, consulte **Regiões e pontos de extremidade**.

Exemplo: {"ap-southeast-1": "https://obs.ap-southeast-1.myhuaweicloud.com:443", "ap-southeast-3": "https://obs.ap-southeast-3.myhuaweicloud.com:443"}

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: paas-obs-endpoint   # The value must be paas-obs-endpoint.
  namespace: kube-system    # The value must be kube-system.
data:
  obs-endpoint: |
    {"<region_name>": "<endpoint_address>"}
```

Passo 2 Crie um PV.

Defina **everest.io/region** como a região onde o OBS está localizado.

```
kind: PersistentVolume
apiVersion: v1
metadata:
  name: testing-abc
  annotations:
    pv.kubernetes.io/bound-by-controller: 'yes'
    pv.kubernetes.io/provisioned-by: everest-csi-provisioner
spec:
  capacity:
    storage: 1Gi
  csi:
    driver: obs.csi.everest.io
    volumeHandle: testing-abc             # OBS bucket name
    fsType: s3fs                          # obsfs indicates a parallel file
system (recommended), and s3fs indicates an object bucket.
    volumeAttributes:
      everest.io/obs-volume-type: STANDARD
      everest.io/region: <region_name>       # Region where the OBS bucket
resides. Replace it with a specific value.
      storage.kubernetes.io/csiProvisionerIdentity: everest-csi-provisioner
    nodePublishSecretRef:                 # AK/SK used for mounting an OBS bucket
      name: test-user
      namespace: default
  accessModes:
    - ReadWriteMany
  persistentVolumeReclaimPolicy: Retain   # The value must be Retain.
  storageClassName: csi-obs
  volumeMode: Filesystem
```

nodePublishSecretRef é a chave de acesso (AK/SK) usada para montar o volume de armazenamento do objeto. Use a AK/SK para criar um segredo, que será usado ao criar um PV. Para mais detalhes, consulte **Uso de uma chave de acesso (AK/SK) personalizada para montar um volume do OBS**.

Passo 3 Crie uma PVC.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
```



```
  name: pvc-test-abc
  namespace: default
  annotations:
    everest.io/obs-volume-type: STANDARD                          # OBS bucket
type. Currently, standard (STANDARD) and infrequent access (WARM) are supported.
    csi.storage.k8s.io/fstype: s3fs                               # File type.
obsfs indicates a parallel file system (recommended), and s3fs indicates an OBS
bucket.
    volume.beta.kubernetes.io/storage-provisioner: everest-csi-provisioner
spec:
  accessModes:
  - ReadWriteMany             # The value must be ReadWriteMany for object
storage.
  resources:
    requests:
      storage: 1Gi            # Storage capacity of a PVC. This field is valid
only for verification (fixed to 1, cannot be empty or 0). The value setting does
not take effect for OBS buckets.
  storageClassName: csi-obs   # Storage class name. For object storage, the value
is fixed to csi-obs.
  volumeName: testing-abc     # PV name
```

Passo 4 Crie uma carga de trabalho, selecione a PVC na opção de armazenamento de dados das configurações do contêiner e adicione a PVC criada. Se a carga de trabalho for criada com êxito, o bucket do OBS poderá ser usado entre regiões.

```
apiVersion: apps/v1
kind: Deployment
metadata:
   name: obs-deployment-example                       # Workload name
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: obs-deployment-example
  template:
    metadata:
      labels:
        app: obs-deployment-example
    spec:
      containers:
      - image: nginx
        name: container-0
        volumeMounts:
        - mountPath: /tmp                        # Mount path
          name: pvc-obs-example
      restartPolicy: Always
      imagePullSecrets:
        - name: default-secret
      volumes:
      - name: pvc-obs-example
        persistentVolumeClaim:
          claimName: pvc-test-abc                # PVC name
```

----Fim

8.7 Volumes persistentes locais

8.7.1 Visão geral

Introdução

O CCE permite usar o LVM para combinar volumes de dados em nós em um pool de armazenamento (VolumeGroup) e criar LVs para montagem de contêineres. Um PV que utiliza um volume persistente local como meio é considerado PV local.

Comparado com o volume de HostPath, o PV local pode ser usado de maneira persistente e portátil. Além disso, o PV do PV local tem a configuração de afinidade de nó. O pod montado no PV local é agendado automaticamente com base na configuração de afinidade. Não é necessário programar manualmente o pod para um nó específico.

Modos de montagem

Os PVs locais só podem ser montados nos seguintes modos:

- **Uso de um PV local através de um PV dinâmico**: modo de criação dinâmica, onde você especifica uma StorageClass durante a criação de PVC e um volume do OBS e um PV serão criados automaticamente.
- **Montagem dinâmica de um PV local para um StatefulSet**: somente os StatefulSets oferecem suporte a esse modo. Cada pod está associado a uma única PVC e PV. Depois que um pod é reprogramado, os dados originais ainda podem ser montados nele com base no nome da PVC. Esse modo se aplica aos StatefulSets com vários pods.

NOTA

PVs locais não podem ser usados através de PVs estáticos. Ou seja, PVs locais não podem ser criados manualmente e, em seguida, montados em cargas de trabalho por meio de PVCs.

Restrições

- Os PVs locais são suportados apenas quando a versão do cluster é v1.21.2-r0 ou posterior e a versão do complemento everest é 2.1.23 ou posterior. Recomenda-se a versão 2.1.23 ou posterior.
- Excluir, remover, redefinir ou dimensionar um nó fará com que os dados de PVC/PV do PV local associado ao nó sejam perdidos, que não podem ser restaurados ou usados novamente. Para obter detalhes, consulte **Remoção de um nó**, **Exclusão de um nó**, **Redefinição de um nó** e **Redução de um nó**. Nesses cenários, o pod que usa o PV local é despejado do nó. Um novo pod será criado e permanece no estado pendente. Isso ocorre porque a PVC usada pelo pod tem um rótulo de nó, devido ao qual o pod não pode ser programado. Depois que o nó é redefinido, o pod pode ser agendado para o nó de redefinição. Nesse caso, o pod está sempre no estado de criação porque o volume lógico subjacente correspondente à PVC não existe.
- Não exclua manualmente o pool de armazenamento correspondente nem desanexe discos de dados do nó. Caso contrário, exceções como perda de dados podem ocorrer.
- Um PV local não pode ser montado em várias cargas de trabalho ou trabalhos ao mesmo tempo.

8.7.2 Importação de um PV para um pool de armazenamento

O CCE permite usar o LVM para combinar volumes de dados em nós em um pool de armazenamento (VolumeGroup) e criar LVs para montagem de contêineres. Antes de criar um PV local, importe o disco de dados do nó para o pool de armazenamento.

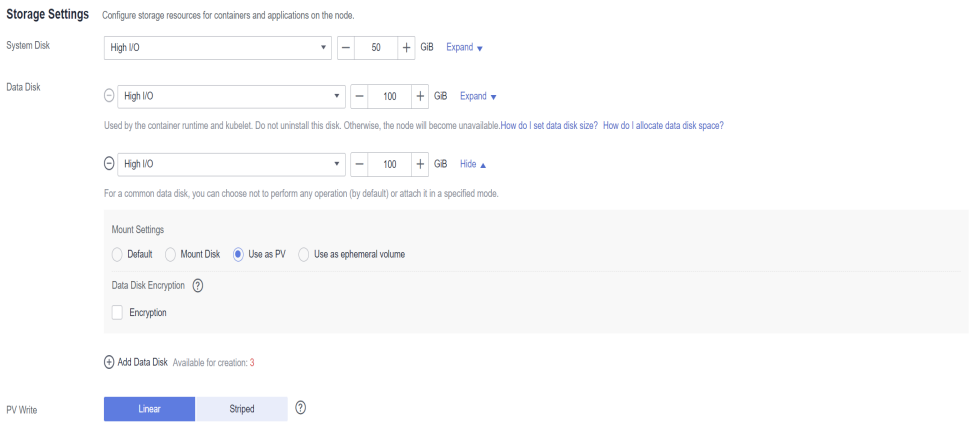
Restrições

- Os PVs locais são suportados apenas quando a versão do cluster é v1.21.2-r0 ou posterior e a versão do complemento everest é 2.1.23 ou posterior. Recomenda-se a versão 2.1.23 ou posterior.
- O primeiro disco de dados (usado pelo tempo de execução do contêiner e pelo componente kubelet) em um nó não pode ser importado como um pool de armazenamento.
- Os pools de armazenamento no modo distribuído não oferecem suporte a expansão. Após a expansão, o espaço fragmentado pode ser gerado e o pool de armazenamento não pode ser usado.
- Os pools de armazenamento não podem ser reduzidos ou excluídos.
- Se os discos em um pool de armazenamento em um nó forem excluídos, o pool de armazenamento funcionará mal.

Importar um pool de armazenamento

Importado durante a criação do nó

Ao criar um nó, você pode adicionar um disco de dados ao nó em **Storage Settings** e importar o disco de dados para o pool de armazenamento como um PV. Para mais detalhes, consulte **Criação de um nó**.



Importado manualmente

Se nenhum PV for importado durante a criação do nó ou se a capacidade do volume de armazenamento atual for insuficiente, você poderá importar manualmente um pool de armazenamento.

Passo 1 Vá para o console do ECS e adicione um disco SCSI ao nó. Para obter detalhes, consulte **Adição de um disco**.

Passo 2 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster.

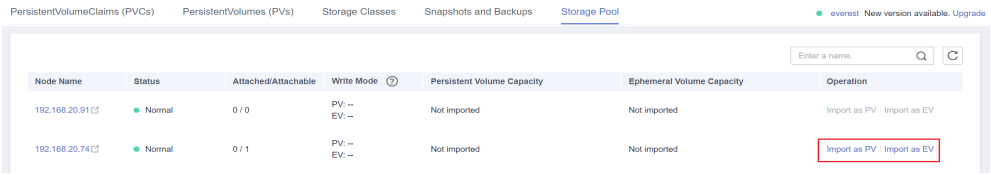
Passo 3 Escolha **Storage** no painel de navegação e clique na guia **Storage Pool**.

Passo 4 Visualize o nó ao qual o disco foi adicionado e selecione **Import as PV**. Você pode selecionar um modo de gravação durante a importação.

NOTA

Se o disco conectado manualmente não for exibido no pool de armazenamento, aguarde 1 minuto e atualize a lista.

- **Linear**: um volume lógico linear integra um ou mais volumes físicos. Os dados são gravados no próximo volume físico quando o anterior é usado.
- **Striped**: um volume lógico distribuído distribui dados em blocos do mesmo tamanho e os armazena em vários volumes físicos em sequência, permitindo que os dados sejam lidos e gravados simultaneamente. Selecione esta opção somente quando houver vários volumes.



----Fim

8.7.3 Uso de um PV local através de um PV dinâmico

Pré-requisitos

- Você criou um cluster e instalou o complemento de CSI (**everest**) no cluster.
- Se você quiser criar um cluster usando comandos, use kubectl para se conectar ao cluster. Para mais detalhes, consulte **Conexão a um cluster usando o kubectl**.
- Você importou um disco de dados de um nó para o pool de armazenamento de PV local. Para mais detalhes, consulte **Importação de um PV para um pool de armazenamento**.

Restrições

- Os PVs locais são suportados apenas quando a versão do cluster é v1.21.2-r0 ou posterior e a versão do complemento everest é 2.1.23 ou posterior. Recomenda-se a versão 2.1.23 ou posterior.
- Excluir, remover, redefinir ou dimensionar um nó fará com que os dados de PVC/PV do PV local associado ao nó sejam perdidos, que não podem ser restaurados ou usados novamente. Para obter detalhes, consulte **Remoção de um nó**, **Exclusão de um nó**, **Redefinição de um nó** e **Redução de um nó**. Nesses cenários, o pod que usa o PV local é despejado do nó. Um novo pod será criado e permanece no estado pendente. Isso ocorre porque a PVC usada pelo pod tem um rótulo de nó, devido ao qual o pod não pode ser programado. Depois que o nó é redefinido, o pod pode ser agendado para o nó de redefinição. Nesse caso, o pod está sempre no estado de criação porque o volume lógico subjacente correspondente à PVC não existe.
- Não exclua manualmente o pool de armazenamento correspondente nem desanexe discos de dados do nó. Caso contrário, exceções como perda de dados podem ocorrer.
- Um PV local não pode ser montado em várias cargas de trabalho ou trabalhos ao mesmo tempo.

Criar automaticamente um PV local no console

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster.

Passo 2 Crie dinamicamente uma PVC e um PV.

1. Escolha **Storage** no painel de navegação e clique na guia **PersistentVolumeClaims (PVCs)**. Clique em **Create PVC** no canto superior direito. Na caixa de diálogo exibida, configure os parâmetros de PVC.

Parâmetro	Descrição
PVC Type	Nesta seção, selecione **Local PV**.
PVC Name	Digite o nome da PVC, que deve ser exclusivo no mesmo namespace.
Creation Method	Você só pode selecionar **Dynamically provision** para criar uma PVC, PV e armazenamento subjacente no console no modo de cascata.
Storage Classes	A classe de armazenamento de PVs locais é **csi-local-topology**.
Access Mode	PVs locais suportam apenas **ReadWriteOnce**, indicando que um volume de armazenamento pode ser montado em um nó no modo leitura/gravação. Para mais detalhes, consulte **Modos de acesso a volume**.
Storage Pool	Exibir o pool de armazenamento importado. Para obter detalhes sobre como importar um novo volume de dados para o pool de armazenamento, consulte **Importação de um PV para um pool de armazenamento**.
Capacity (GiB)	Capacidade do volume de armazenamento solicitado.

2. Clique em **Create** para criar uma PVC e um PV.

   Você pode escolher **Storage** no painel de navegação e exibir a PVC e o PV criados nas páginas de guia **PersistentVolumeClaims (PVCs)** e **PersistentVolumes (PVs)**, respectivamente.

   NOTA

   O modo de vinculação de volume da classe de armazenamento local (chamada **csi-local-topology**) é vinculação tardia (ou seja, o valor de **volumeBindingMode** é **WaitForFirstConsumer**). Nesse modo, a criação e a vinculação do PV são atrasadas. O PV correspondente é criado e vinculado somente quando a PVC é usada durante a criação da carga de trabalho.

Passo 3 Crie uma aplicação.

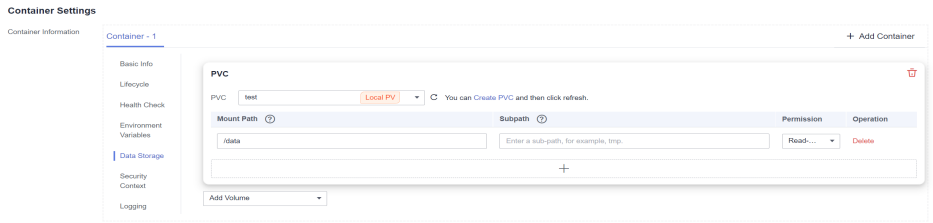
1. No painel de navegação à esquerda, clique em **Workloads**. No painel direito, clique na guia **Deployments**.
2. Clique em **Create Workload** no canto superior direito. Na página exibida, clique em **Data Storage** na área **Container Settings** e clique em **Add Volume** para selecionar **PVC**.

   Monte e use volumes de armazenamento, conforme mostrado em **Tabela 8-38**. Para obter detalhes sobre outros parâmetros, consulte **Cargas de trabalho**.

Tabela 8-38 Montagem de um volume de armazenamento

Parâmetro	Descrição
PVC	Selecione um PV local existente. Um PV local não pode ser montado repetidamente em várias cargas de trabalho.
Mount Path	Digite um caminho de montagem, por exemplo, **/tmp**. Este parâmetro indica o caminho do contêiner no qual um volume de dados será montado. Não monte o volume em um diretório do sistema como / ou **/var/run**. Caso contrário, os contêineres estarão com defeito. Monte o volume em um diretório vazio. Se o diretório não estiver vazio, verifique se não há arquivos que afetem a inicialização do contêiner. Caso contrário, os arquivos serão substituídos, causando falhas de inicialização do contêiner ou falhas de criação de carga de trabalho. **AVISO** Se um volume for montado em um diretório de alto risco, use uma conta com permissões mínimas para iniciar o contêiner. Caso contrário, arquivos de alto risco no host podem ser danificados.
Subpath	Digite um subcaminho, por exemplo, **tmp**, indicando que os dados no caminho de montagem do contêiner serão armazenados na pasta **tmp** do volume. Um subcaminho é usado para montar um volume local para que o mesmo volume de dados seja usado em um único pod. Se este parâmetro for deixado em branco, o caminho raiz é usado por padrão.
Permission	– **Read-only**: você só pode ler os dados nos volumes montados. – **Read/Write**: você pode modificar os volumes de dados montados no caminho. Os dados recém-gravados não serão migrados se o contêiner for migrado, o que pode causar perda de dados.

Neste exemplo, o disco é montado no caminho **/data** do contêiner. Os dados de contêiner gerados nesse caminho são armazenados no PV local.



3. Após a configuração, clique em **Create Workload**.

Depois que a carga de trabalho for criada, os dados no diretório de montagem do contêiner serão armazenados persistentemente. Verifique o armazenamento referindo-se a **Verificar a persistência dos dados**.

----Fim

(kubectl) Criar automaticamente um PV local

Passo 1 Use o kubectl para se conectar ao cluster.

Passo 2 Use **StorageClass** para criar dinamicamente uma PVC e um PV.

1. Crie o arquivo **pvc-local.yaml**.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-local
  namespace: default
spec:
  accessModes:
    - ReadWriteOnce             # The local PV must adopt ReadWriteOnce.
  resources:
    requests:
      storage: 10Gi             # Size of the local PV.
  storageClassName: csi-local-topology    # StorageClass is local PV.
```

Tabela 8-39 Parâmetros principais

Parâmetro	Obrigatório	Descrição
storage	Sim	Capacidade solicitada na PVC, em Gi.
storageClassName	Sim	Nome da classe de armazenamento. O nome da classe de armazenamento do PV local é **csi-local-topology**.

2. Execute o seguinte comando para criar uma PVC:

```
kubectl apply -f pvc-local.yaml
```

Passo 3 Crie uma aplicação.

1. Crie um arquivo chamado **web-demo.yaml**. Neste exemplo, o PV local é montado no caminho **/data**.

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: web-local
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: web-local
  serviceName: web-local   # Headless Service name.
  template:
    metadata:
      labels:
        app: web-local
    spec:
      containers:
```



```
      - name: container-1
        image: nginx:latest
        volumeMounts:
        - name: pvc-disk    #Volume name, which must be the same as the
volume name in the volumes field.
          mountPath: /data  #Location where the storage volume is mounted.
      imagePullSecrets:
        - name: default-secret
      volumes:
        - name: pvc-disk    #Volume name, which can be customized.
          persistentVolumeClaim:
            claimName: pvc-local    #Name of the created PVC.
---
apiVersion: v1
kind: Service
metadata:
  name: web-local   # Headless Service name.
  namespace: default
  labels:
    app: web-local
spec:
  selector:
    app: web-local
  clusterIP: None
  ports:
    - name: web-local
      targetPort: 80
      nodePort: 0
      port: 80
      protocol: TCP
  type: ClusterIP
```

2. Execute o seguinte comando para criar uma carga de trabalho na qual o PV local é montado:

```
kubectl apply -f web-local.yaml
```

Depois que a carga de trabalho for criada, os dados no diretório de montagem do contêiner serão armazenados persistentemente. Verifique o armazenamento referindo-se a **Verificar a persistência dos dados**.

----Fim

Verificar a persistência dos dados

Passo 1 Exibir a aplicação implementada e os arquivos locais.

1. Execute o seguinte comando para exibir o pod criado:

```
kubectl get pod | grep web-local
```

Saída esperada:

```
web-local-0                  1/1     Running   0               38s
```

2. Execute o seguinte comando para verificar se o PV local foi montado no caminho **/data**:

```
kubectl exec web-local-0 -- df | grep data
```

Saída esperada:

```
/dev/mapper/vg--everest--localvolume--persistent-pvc-local
10255636     36888  10202364   0% /data
```

3. Execute o seguinte comando para exibir os arquivos no caminho **/data**:

```
kubectl exec web-local-0 -- ls /data
```

Saída esperada:

```
lost+found
```

Passo 2 Execute o seguinte comando para criar um arquivo chamado **static** no caminho **/data**:

```
kubectl exec web-local-0 --  touch /data/static
```


Passo 3 Execute o seguinte comando para exibir os arquivos no caminho **/data**:

```
kubectl exec web-local-0 -- ls /data
```

Saída esperada:

```
lost+found
static
```

Passo 4 Execute o seguinte comando para excluir o pod chamado **web-local-0**:

```
kubectl delete pod web-local-0
```

Saída esperada:

```
pod "web-local-0" deleted
```

Passo 5 Após a exclusão, o controlador de StatefulSet cria automaticamente uma réplica com o mesmo nome. Execute o seguinte comando para verificar se os arquivos no caminho **/data** foram modificados:

```
kubectl exec web-local-0 -- ls /data
```

Saída esperada:

```
lost+found
static
```

Se o arquivo **static** ainda existir, os dados no PV local podem ser armazenados persistentemente.

----Fim

Operações relacionadas

Você também pode executar as operações listadas em **Tabela 8-40**.

Tabela 8-40 Operações relacionadas

Operação	Descrição	Procedimento
Visualização de eventos	Você pode visualizar nomes de eventos, tipos de eventos, número de ocorrências, eventos do Kubernetes, horário da primeira ocorrência e horário da última ocorrência da PVC ou PV.	1. Escolha **Storage** no painel de navegação e clique na guia **PersistentVolumeClaims (PVCs)** ou **PersistentVolumes (PVs)**. 2. Clique em **View Events** na coluna **Operation** da PVC ou PV de destino para exibir os eventos gerados dentro de uma hora (os dados do evento são mantidos por uma hora).
Exibição de um arquivo YAML	Você pode visualizar, copiar e fazer download dos arquivos YAML de uma PVC ou um PV.	1. Escolha **Storage** no painel de navegação e clique na guia **PersistentVolumeClaims (PVCs)** ou **PersistentVolumes (PVs)**. 2. Clique em **View YAML** na coluna **Operation** da PVC ou PV de destino para exibir ou fazer download do YAML.

8.7.4 Montagem dinâmica de um PV local para um StatefulSet

Cenários de aplicações

A montagem dinâmica está disponível apenas para a criação de um **StatefulSet**. Ela é implementada por meio de um modelo de declaração de volume (campo **volumeClaimTemplates**) e depende da classe de armazenamento para provisionar PVs dinamicamente. Nesse modo, cada pod em um StatefulSet de vários pods está associado a uma PVC e um PV exclusivos. Depois que um pod é reprogramado, os dados originais ainda podem ser montados nele com base no nome da PVC. No modo de montagem comum de uma Implementação, se ReadWriteMany for suportado, vários pods da Implementação serão montados no mesmo armazenamento subjacente.

Pré-requisitos

- Você criou um cluster e instalou o complemento de CSI (**everest**) no cluster.
- Se você quiser criar um cluster usando comandos, use kubectl para se conectar ao cluster. Para mais detalhes, consulte **Conexão a um cluster usando o kubectl**.
- Você importou um disco de dados de um nó para o pool de armazenamento de PV local. Para mais detalhes, consulte **Importação de um PV para um pool de armazenamento**.

Montagem dinâmica de um PV local no console

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster.

Passo 2 No painel de navegação à esquerda, clique em **Workloads**. No painel direito, clique na guia **StatefulSets**.

Passo 3 Clique em **Create Workload** no canto superior direito. Na página exibida, clique em **Data Storage** na área **Container Settings** e clique em **Add Volume** para selecionar **VolumeClaimTemplate (VTC)**.

Passo 4 Clique em **Create PVC**. Na caixa de diálogo exibida, configure os parâmetros do modelo de declaração de volume.

Clique em **Create**.

Parâmetro	Descrição
PVC Type	Nesta seção, selecione **Local PV**.
PVC Name	Digite o nome da PVC. Depois que uma PVC é criada, um sufixo é adicionado automaticamente com base no número de pods. O formato é *<Custom PVC name>-<Serial number>*, por exemplo, example-0.
Creation Method	Você só pode selecionar **Dynamically provision** para criar uma PVC, PV e armazenamento subjacente no console no modo de cascata.
Storage Classes	A classe de armazenamento de PVs locais é **csi-local-topology**.
Access Mode	PVs locais suportam apenas **ReadWriteOnce**, indicando que um volume de armazenamento pode ser montado em um nó no modo leitura/gravação. Para mais detalhes, consulte **Modos de acesso a volume**.

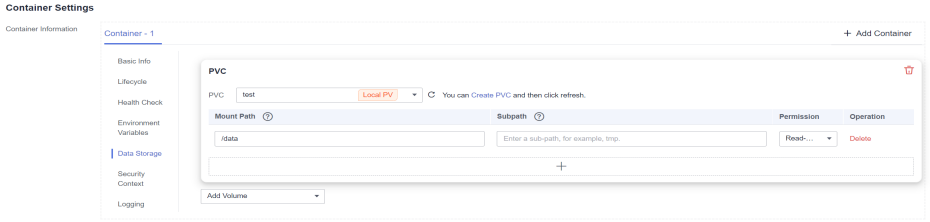
Parâmetro	Descrição
Storage Pool	Exiba o pool de armazenamento importado. Para obter detalhes sobre como importar um novo volume de dados para o pool de armazenamento, consulte **Importação de um PV para um pool de armazenamento**.
Capacity (GiB)	Capacidade do volume de armazenamento solicitado.

Passo 5 Insira o caminho no qual o volume está montado.

Tabela 8-41 Montagem de um volume de armazenamento

Parâmetro	Descrição
Mount Path	Digite um caminho de montagem, por exemplo, **/tmp**. Este parâmetro indica o caminho do contêiner no qual um volume de dados será montado. Não monte o volume em um diretório do sistema como / ou **/var/run**. Caso contrário, os contêineres estarão com defeito. Monte o volume em um diretório vazio. Se o diretório não estiver vazio, verifique se não há arquivos que afetem a inicialização do contêiner. Caso contrário, os arquivos serão substituídos, causando falhas de inicialização do contêiner ou falhas de criação de carga de trabalho. **AVISO** Se um volume for montado em um diretório de alto risco, use uma conta com permissões mínimas para iniciar o contêiner. Caso contrário, arquivos de alto risco no host podem ser danificados.
Subpath	Digite um subcaminho, por exemplo, **tmp**, indicando que os dados no caminho de montagem do contêiner serão armazenados na pasta **tmp** do volume. Um subcaminho é usado para montar um volume local para que o mesmo volume de dados seja usado em um único pod. Se este parâmetro for deixado em branco, o caminho raiz é usado por padrão.
Permission	● **Read-only**: você só pode ler os dados nos volumes montados. ● **Read/Write**: você pode modificar os volumes de dados montados no caminho. Os dados recém-gravados não serão migrados se o contêiner for migrado, o que pode causar perda de dados.

Neste exemplo, o disco é montado no caminho **/data** do contêiner. Os dados de contêiner gerados nesse caminho são armazenados no PV local.



Passo 6 Monte e use dinamicamente volumes de armazenamento. Para obter detalhes sobre outros parâmetros, consulte **Criação de um StatefulSet**. Após a configuração, clique em **Create Workload**.

Depois que a carga de trabalho for criada, os dados no diretório de montagem do contêiner serão armazenados persistentemente. Verifique o armazenamento referindo-se a **Verificar a persistência dos dados**.

----Fim

Montagem dinâmica de um PV local usando kubectl

Passo 1 Use o kubectl para se conectar ao cluster.

Passo 2 Crie um arquivo chamado **statefulset-local.yaml**. Neste exemplo, o PV local é montado no caminho **/data**.

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: statefulset-local
  namespace: default
spec:
  selector:
    matchLabels:
      app: statefulset-local
  template:
    metadata:
      labels:
        app: statefulset-local
    spec:
      containers:
        - name: container-1
          image: nginx:latest
          volumeMounts:
            - name: pvc-local          # The value must be the same as that in
the volumeClaimTemplates field.
              mountPath: /data         # Location where the storage volume is
mounted.
      imagePullSecrets:
        - name: default-secret
  serviceName: statefulset-local       # Headless Service name.
  replicas: 2
  volumeClaimTemplates:
    - apiVersion: v1
      kind: PersistentVolumeClaim
      metadata:
        name: pvc-local
        namespace: default
      spec:
        accessModes:
          - ReadWriteOnce               # The local PV must adopt ReadWriteOnce.
        resources:
          requests:
            storage: 10Gi               # Storage volume capacity.
        storageClassName: csi-local-topology      # StorageClass is local PV.
---
apiVersion: v1
kind: Service
metadata:
  name: statefulset-local   # Headless Service name.
  namespace: default
  labels:
    app: statefulset-local
spec:
  selector:
    app: statefulset-local
```



```
  clusterIP: None
  ports:
    - name: statefulset-local
      targetPort: 80
      nodePort: 0
      port: 80
      protocol: TCP
  type: ClusterIP
```

Tabela 8-42 Parâmetros principais

Parâmetro	Obrigatório	Descrição
storage	Sim	Capacidade solicitada na PVC, em Gi.
storageClassName	Sim	A classe de armazenamento de PVs locais é **csi-local-topology**.

Passo 3 Execute o seguinte comando para criar uma carga de trabalho na qual o PV local é montado:

```
kubectl apply -f statefulset-local.yaml
```

Depois que a carga de trabalho é criada, você pode tentar **Verificar a persistência dos dados**.

----Fim

Verificar a persistência dos dados

Passo 1 Exiba a aplicação implementada e os arquivos.

1. Execute o seguinte comando para exibir o pod criado:

   ```
   kubectl get pod | grep statefulset-local
   ```

   Saída esperada:

   ```
   statefulset-local-0          1/1     Running   0             45s
   statefulset-local-1          1/1     Running   0             28s
   ```

2. Execute o seguinte comando para verificar se o PV local foi montado no caminho **/data**:

   ```
   kubectl exec statefulset-local-0 -- df | grep data
   ```

   Saída esperada:

   ```
   /dev/mapper/vg--everest--localvolume--persistent-pvc-local
   10255636     36888  10202364   0% /data
   ```

3. Execute o seguinte comando para exibir os arquivos no caminho **/data**:

   ```
   kubectl exec statefulset-local-0 -- ls /data
   ```

   Saída esperada:

   ```
   lost+found
   ```

Passo 2 Execute o seguinte comando para criar um arquivo chamado **static** no caminho **/data**:

```
kubectl exec statefulset-local-0 --  touch /data/static
```

Passo 3 Execute o seguinte comando para exibir os arquivos no caminho **/data**:

```
kubectl exec statefulset-local-0 -- ls /data
```

Saída esperada:

```
lost+found
static
```

Passo 4 Execute o seguinte comando para excluir o pod chamado **web-local-auto-0**:

```
kubectl delete pod statefulset-local-0
```


Saída esperada:

```
pod "statefulset-local-0" deleted
```

Passo 5 Após a exclusão, o controlador de StatefulSet cria automaticamente uma réplica com o mesmo nome. Execute o seguinte comando para verificar se os arquivos no caminho **/data** foram modificados:

```
kubectl exec statefulset-local-0 -- ls /data
```

Saída esperada:

```
lost+found
static
```

Se o arquivo **static** ainda existir, os dados no PV local podem ser armazenados persistentemente.

----Fim

Operações relacionadas

Você também pode executar as operações listadas em **Tabela 8-43**.

Tabela 8-43 Operações relacionadas

Operação	Descrição	Procedimento
Visualização de eventos	Você pode visualizar nomes de eventos, tipos de eventos, número de ocorrências, eventos do Kubernetes, horário da primeira ocorrência e horário da última ocorrência da PVC ou PV.	1. Escolha **Storage** no painel de navegação e clique na guia **PersistentVolumeClaims (PVCs)** ou **PersistentVolumes (PVs)**. 2. Clique em **View Events** na coluna **Operation** da PVC ou PV de destino para exibir os eventos gerados dentro de uma hora (os dados do evento são mantidos por uma hora).
Exibição de um arquivo YAML	Você pode visualizar, copiar e fazer download dos arquivos YAML de uma PVC ou um PV.	1. Escolha **Storage** no painel de navegação e clique na guia **PersistentVolumeClaims (PVCs)** ou **PersistentVolumes (PVs)**. 2. Clique em **View YAML** na coluna **Operation** da PVC ou PV de destino para exibir ou fazer download do YAML.

8.8 Volumes efêmeros

8.8.1 Visão geral

Introdução

Algumas aplicações exigem armazenamento adicional, mas se os dados ainda estão disponíveis após uma reinicialização não é importante. Por exemplo, embora os serviços de

cache sejam limitados pelo tamanho da memória, os serviços de cache podem mover dados usados com pouca frequência para um armazenamento mais lento que a memória. Como resultado, o desempenho geral não é afetado significativamente. Outras aplicações exigem dados de somente leitura injetados como arquivos, como dados de configuração ou segredos.

Os **volumes efêmeros** (EVs) no Kubernetes são projetados para o cenário acima. Os EVs são criados e excluídos juntamente com os pods após o ciclo de vida do pod.

EVs comuns em Kubernetes:

- **emptyDir**: vazio na inicialização do pod, com armazenamento vindo localmente do diretório base do kubelet (geralmente o disco raiz) ou da memória. emptyDir é alocado a partir do **EV do nó**. Se dados de outras fontes (como arquivos de log ou dados de camadas de imagem) ocuparem o armazenamento temporário, a capacidade de armazenamento pode ser insuficiente.
- **ConfigMap**: os dados do tipo ConfigMap do Kubernetes são montados em pods como volumes de dados.
- **Secret**: os dados do Kubernetes do tipo Secret são montados em pods como volumes de dados.

Tipos de emptyDir

O CCE fornece os seguintes tipos de emptyDir:

- **Uso de um caminho temporário**: tipo emptyDir de Kubernetes nativo. Seu ciclo de vida é o mesmo que o de um pod. A memória pode ser especificada como o meio de armazenamento. Quando o pod é excluído, o volume de emptyDir é excluído e seus dados são perdidos.
- **Uso de um EV local**: os discos de dados locais em um nó formam um **pool de armazenamento** (VolumeGroup) por meio do LVM. Os LVs são criados como o meio de armazenamento de emptyDir e montados em contêineres. Os LVs oferecem um desempenho melhor do que o meio de armazenamento padrão do emptyDir.

Restrições

- Os EVs locais são suportados apenas quando a versão do cluster é v1.21.2-r0 ou posterior e a versão do complemento everest é 2.1.23 ou posterior.
- Não exclua manualmente o pool de armazenamento correspondente nem desanexe discos de dados do nó. Caso contrário, exceções como perda de dados podem ocorrer.
- Certifique-se de que o diretório **/var/lib/kubelet/pods/** não esteja montado no pod no nó. Caso contrário, o pod, montado com tais volumes, pode falhar em ser excluído.

8.8.2 Importação de um EV para um pool de armazenamento

O CCE permite usar o LVM para combinar volumes de dados em nós em um pool de armazenamento (VolumeGroup) e criar LVs para montagem de contêineres. Antes de criar um EV local, importe o disco de dados do nó para o pool de armazenamento.

Restrições

- Os EVs locais são suportados apenas quando a versão do cluster é v1.21.2-r0 ou posterior e a versão do complemento everest é 2.1.23 ou posterior.

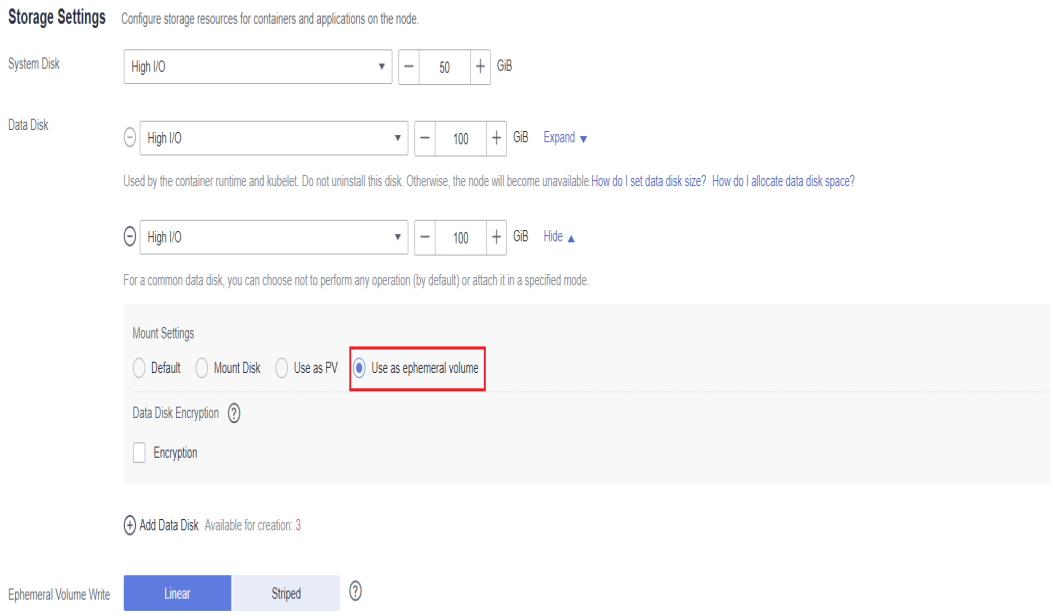
- O primeiro disco de dados (usado pelo tempo de execução do contêiner e pelo componente kubelet) em um nó não pode ser importado como um pool de armazenamento.
- Os pools de armazenamento no modo distribuído não oferecem suporte a expansão. Após a expansão, o espaço fragmentado pode ser gerado e o pool de armazenamento não pode ser usado.
- Os pools de armazenamento não podem ser reduzidos ou excluídos.
- Se os discos em um pool de armazenamento em um nó forem excluídos, o pool de armazenamento funcionará mal.

Importar um pool de armazenamento

Importado durante a criação do nó

Ao criar um nó, você pode adicionar um disco de dados ao nó em **Storage Settings** e importar o disco de dados para o pool de armazenamento como um EV. Para mais detalhes, consulte **Criação de um nó**.

Figura 8-5 Importar como um EV



Importado manualmente

Se nenhum EV for importado durante a criação do nó ou se a capacidade do volume de armazenamento atual for insuficiente, você poderá importar manualmente um pool de armazenamento.

Passo 1 Vá para o console do ECS e adicione um disco SCSI ao nó. Para obter detalhes, consulte **Adição de um disco**.

Passo 2 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster.

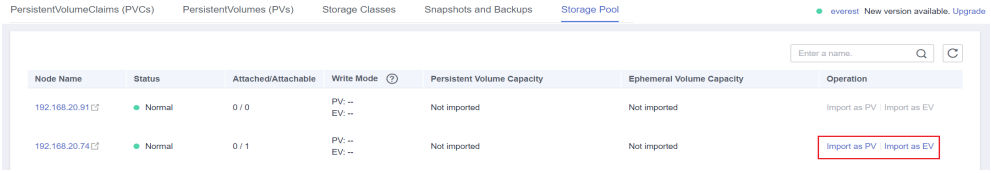
Passo 3 Escolha **Storage** no painel de navegação e clique na guia **Storage Pool**.

Passo 4 Visualize o nó ao qual o disco foi adicionado e selecione **Import as EV**. Você pode selecionar um modo de gravação durante a importação.

NOTA

Se o disco conectado manualmente não for exibido no pool de armazenamento, aguarde 1 minuto e atualize a lista.

- **Linear**: um volume lógico linear integra um ou mais volumes físicos. Os dados são gravados no próximo volume físico quando o anterior é usado.
- **Striped**: um volume lógico distribuído distribui dados em blocos do mesmo tamanho e os armazena em vários volumes físicos em sequência, permitindo que os dados sejam lidos e gravados simultaneamente. Selecione esta opção somente quando houver vários volumes.



----Fim

8.8.3 Uso de um EV local

Os Volumes efêmeros locais (EVs) são armazenados em **pools de armazenamento** de EV. Os EVs locais oferecem um desempenho melhor do que o meio de armazenamento padrão do emptyDir nativo e suportam expansão.

Pré-requisitos

- Você criou um cluster e instalou o complemento de CSI (**everest**) no cluster.
- Se você quiser criar um cluster usando comandos, use kubectl para se conectar ao cluster. Para mais detalhes, consulte **Conexão a um cluster usando o kubectl**.
- Para usar um EV local, importe um disco de dados de um nó para o pool de armazenamento de EV local. Para mais detalhes, consulte **Importação de um EV para um pool de armazenamento**.

Restrições

- Os EVs locais são suportados apenas quando a versão do cluster é v1.21.2-r0 ou posterior e a versão do complemento everest é 2.1.23 ou posterior.
- Não exclua manualmente o pool de armazenamento correspondente nem desanexe discos de dados do nó. Caso contrário, exceções como perda de dados podem ocorrer.
- Certifique-se de que o diretório **/var/lib/kubelet/pods/** não esteja montado no pod no nó. Caso contrário, o pod, montado com tais volumes, pode falhar em ser excluído.

Usar o console para montar um EV local

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster.

Passo 2 No painel de navegação à esquerda, clique em **Workloads**. No painel direito, clique na guia **Deployments**.

Passo 3 Clique em **Create Workload** no canto superior direito da página. Na área **Container Settings**, clique na guia **Data Storage** e clique em **Add Volume** > **Local Ephemeral Volume (emptyDir)**.

Passo 4 Monte e use volumes de armazenamento, conforme mostrado na **Tabela 8-44**. Para obter detalhes sobre outros parâmetros, consulte **Cargas de trabalho**.

Tabela 8-44 Montagem de um EV local

Parâmetro	Descrição
Capacity	Capacidade do volume de armazenamento solicitado.
Mount Path	Digite um caminho de montagem, por exemplo, **/tmp**. Este parâmetro indica o caminho do contêiner no qual um volume de dados será montado. Não monte o volume em um diretório do sistema como / ou **/var/run**. Caso contrário, os contêineres estarão com defeito. Monte o volume em um diretório vazio. Se o diretório não estiver vazio, verifique se não há arquivos que afetem a inicialização do contêiner. Caso contrário, os arquivos serão substituídos, causando falhas de inicialização do contêiner ou falhas de criação de carga de trabalho. **AVISO** Se um volume for montado em um diretório de alto risco, use uma conta com permissões mínimas para iniciar o contêiner. Caso contrário, arquivos de alto risco no host podem ser danificados.
Subpath	Digite um subcaminho, por exemplo, **tmp**, indicando que os dados no caminho de montagem do contêiner serão armazenados na pasta **tmp** do volume. Um subcaminho é usado para montar um volume local para que o mesmo volume de dados seja usado em um único pod. Se este parâmetro for deixado em branco, o caminho raiz é usado por padrão.
Permission	● **Read-only**: você só pode ler os dados nos volumes montados. ● **Read/Write**: você pode modificar os volumes de dados montados no caminho. Os dados recém-gravados não serão migrados se o contêiner for migrado, o que pode causar perda de dados.

Passo 5 Após a configuração, clique em **Create Workload**.

----Fim

Usar o kubectl para montar um EV local

Passo 1 Use o kubectl para se conectar ao cluster. Para mais detalhes, consulte **Conexão a um cluster usando o kubectl**.

Passo 2 Crie um arquivo chamado **nginx-emptydir.yaml** e edite-o.

vi nginx-emptydir.yaml

Conteúdo do arquivo YAML:


```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-emptydir
  namespace: default
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx-emptydir
  template:
    metadata:
      labels:
        app: nginx-emptydir
    spec:
      containers:
        - name: container-1
          image: nginx:latest
          volumeMounts:
            - name: vol-emptydir         # Volume name, which must be the same as
the volume name in the volumes field.
              mountPath: /tmp            # Path to which an EV is mounted.
      imagePullSecrets:
        - name: default-secret
      volumes:
        - name: vol-emptydir             # Volume name, which can be customized.
          emptyDir:
            medium: LocalVolume          # If the disk medium of emptyDir is set
to LocalVolume, the local EV is used.
            sizeLimit: 1Gi               # Volume capacity.
```

Passo 3 Crie uma carga de trabalho.

kubectl apply -f nginx-emptydir.yaml

----Fim

Manipulação de exceções de EV local

Se um usuário desanexar manualmente um disco do ECS ou executar manualmente o comando **vgremove**, o pool de armazenamento do EV poderá funcionar mal. Para resolver esse problema, defina o nó como não agendado seguindo o procedimento descrito em **Configurações de agendamento de nó** e redefina o nó.

8.8.4 Uso de um caminho temporário

Um caminho temporário é do tipo emptyDir do Kubernetes nativo. Seu ciclo de vida é o mesmo que o de um pod. A memória pode ser especificada como o meio de armazenamento. Quando o pod é excluído, o volume de emptyDir é excluído e seus dados são perdidos.

Usar o console para usar um caminho temporário

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster.

Passo 2 No painel de navegação à esquerda, clique em **Workloads**. No painel direito, clique na guia **Deployments**.

Passo 3 Clique em **Create Workload** no canto superior direito da página. Na área **Container Settings**, clique na guia **Data Storage** e clique em **Add Volume** > **emptyDir**.

Passo 4 Monte e use volumes de armazenamento, conforme mostrado na **Tabela 8-45**. Para obter detalhes sobre outros parâmetros, consulte **Cargas de trabalho**.

Tabela 8-45 Montagem de um EV

Parâmetro	Descrição
Storage Medium	**Memory**: ● você pode selecionar essa opção para melhorar a velocidade de execução, mas a capacidade de armazenamento está sujeita ao tamanho da memória. Esse modo é aplicável quando o volume de dados é pequeno e a leitura e gravação eficientes são necessárias. ● Se esta função estiver desativada, os dados são armazenados em discos rígidos, o que se aplica a uma grande quantidade de dados com baixos requisitos de eficiência de leitura e escrita. **NOTA** ● Se **Memory** estiver selecionada, preste atenção ao tamanho da memória. Se a capacidade de armazenamento exceder o tamanho da memória, ocorrerá um evento OOM. ● Se **Memory** for selecionada, o tamanho de um EV é o mesmo que as especificações do pod. ● Se **Memory** não estiver selecionada, os EVs não ocuparão a memória do sistema.
Mount Path	Digite um caminho de montagem, por exemplo, **/tmp**. Este parâmetro indica o caminho do contêiner no qual um volume de dados será montado. Não monte o volume em um diretório do sistema como **/** ou **/var/run**. Caso contrário, os contêineres estarão com defeito. Monte o volume em um diretório vazio. Se o diretório não estiver vazio, verifique se não há arquivos que afetem a inicialização do contêiner. Caso contrário, os arquivos serão substituídos, causando falhas de inicialização do contêiner ou falhas de criação de carga de trabalho. **AVISO** Se um volume for montado em um diretório de alto risco, use uma conta com permissões mínimas para iniciar o contêiner. Caso contrário, arquivos de alto risco no host podem ser danificados.
Subpath	Digite um subcaminho, por exemplo, **tmp**, indicando que os dados no caminho de montagem do contêiner serão armazenados na pasta **tmp** do volume. Um subcaminho é usado para montar um volume local para que o mesmo volume de dados seja usado em um único pod. Se este parâmetro for deixado em branco, o caminho raiz é usado por padrão.
Permission	● **Read-only**: você só pode ler os dados nos volumes montados. ● **Read/Write**: você pode modificar os volumes de dados montados no caminho. Os dados recém-gravados não serão migrados se o contêiner for migrado, o que pode causar perda de dados.

Passo 5 Após a configuração, clique em **Create Workload**.

----Fim

Usar o kubectl para usar um caminho temporário

Passo 1 Use o kubectl para se conectar ao cluster. Para mais detalhes, consulte **Conexão a um cluster usando o kubectl**.

Passo 2 Crie um arquivo chamado **nginx-emptydir.yaml** e edite-o.

vi nginx-emptydir.yaml

Conteúdo do arquivo YAML:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-emptydir
  namespace: default
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx-emptydir
  template:
    metadata:
      labels:
        app: nginx-emptydir
    spec:
      containers:
        - name: container-1
          image: nginx:latest
          volumeMounts:
            - name: vol-emptydir     # Volume name, which must be the same as the volume name in the volumes field.
              mountPath: /tmp        # Path to which an EV is mounted.
      imagePullSecrets:
        - name: default-secret
      volumes:
        - name: vol-emptydir         # Volume name, which can be customized.
          emptyDir:
            medium: Memory           # EV disk medium: If this parameter is set to Memory, the memory is enabled. If this parameter is left blank, the native default storage medium is used.
            sizeLimit: 1Gi           # Volume capacity.
```

Passo 3 Crie uma carga de trabalho.

kubectl apply -f nginx-emptydir.yaml

----Fim

8.9 hostPath

hostPath é usado para montar o diretório de arquivos do host onde o contêiner está localizado no ponto de montagem especificado do contêiner. Se o contêiner precisar acessar **/etc/hosts**, use hostPath para mapear **/etc/hosts**.

AVISO

- Evite usar volumes de hostPath o máximo possível, pois eles são propensos a riscos de segurança. Se os volumes de hostPath precisarem ser usados, eles só poderão ser aplicados a arquivos ou caminhos e montados no modo somente leitura.
- Depois que o pod no qual um volume de hostPath é montado é excluído, os dados no volume de hostPath são retidos.

Montar um volume de hostPath no console

Você pode montar um caminho no host para um caminho de contêiner especificado. Um volume hostPath geralmente é usado para **armazenar logs de carga de trabalho permanentemente** ou usado por cargas de trabalho que precisam **acessar a estrutura de dados interna do mecanismo Docker no host**.

Passo 1 Efetue logon no console do CCE.

Passo 2 Ao criar uma carga de trabalho, clique em **Data Storage** em **Container Settings**. Clique em **Add Volume** e escolha **hostPath** na lista suspensa.

Passo 3 Defina os parâmetros para adicionar um volume local, conforme listado em **Tabela 8-46**.

Tabela 8-46 Configurar parâmetros para montar um volume de hostPath

Parâmetro	Descrição
Storage Type	Selecione **HostPath**.
Host Path	Caminho do host no qual o volume local deve ser montado, por exemplo, **/etc/hosts**. **NOTA** **Host Path** não pode ser definido para o diretório raiz /. Caso contrário, a montagem falhará. Os caminhos de montagem podem ser os seguintes: ● **/opt/*xxxx*** (excluindo **/opt/cloud**) ● **/mnt/*xxxx*** (excluindo **/mnt/paas**) ● **/tmp/*xxx*** ● **/var/*xxx*** (excluindo diretórios-chave como **/var/lib**, **/var/script** e **/var/paas**) ● **/*xxxx*** (Ele não pode entrar em conflito com o diretório do sistema, como **bin**, **lib**, **home**, **root**, **boot**, **dev**, **etc**, **lost+found**, **mnt**, **proc**, **sbin**, **srv**, **tmp**, **var**, **media**, **opt**, **selinux**, **sys** e **usr**.) Não defina esse parâmetro para **/home/paas**, **/var/paas**, **/var/lib**, **/var/script**, **/mnt/paas** ou **/opt/cloud**. Caso contrário, a instalação do sistema ou nó falhará.

Parâmetro	Descrição
Mount Path	Digite um caminho de montagem, por exemplo, **/tmp**. Este parâmetro indica o caminho do contêiner no qual um volume de dados será montado. Não monte o volume em um diretório do sistema como / ou **/var/run**. Caso contrário, os contêineres estarão com defeito. Monte o volume em um diretório vazio. Se o diretório não estiver vazio, verifique se não há arquivos que afetem a inicialização do contêiner. Caso contrário, os arquivos serão substituídos, causando falhas de inicialização do contêiner ou falhas de criação de carga de trabalho. **AVISO** Se um volume for montado em um diretório de alto risco, use uma conta com permissões mínimas para iniciar o contêiner. Caso contrário, arquivos de alto risco no host podem ser danificados.
Subpath	Digite um subcaminho, por exemplo, **tmp**, indicando que os dados no caminho de montagem do contêiner serão armazenados na pasta **tmp** do volume. Um subcaminho é usado para montar um volume local para que o mesmo volume de dados seja usado em um único pod. Se este parâmetro for deixado em branco, o caminho raiz é usado por padrão.
Permission	● **Read-only**: você só pode ler os dados nos volumes montados. ● **Read/Write**: você pode modificar os volumes de dados montados no caminho. Os dados recém-gravados não serão migrados se o contêiner for migrado, o que pode causar perda de dados.

Passo 4 Após a configuração, clique em **Create Workload**.

----Fim

Montar um volume de hostPath usando kubectl

Passo 1 Use o kubectl para se conectar ao cluster.

Passo 2 Crie um arquivo chamado **nginx-hostpath.yaml** e edite-o.

vi nginx-hostpath.yaml

O conteúdo do arquivo YAML é o seguinte. Monte o diretório **/data** no nó no diretório **/data** no contêiner.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-hostpath
  namespace: default
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx-hostpath
  template:
    metadata:
      labels:
        app: nginx-hostpath
```



```
    spec:
      containers:
        - name: container-1
          image: nginx:latest
          volumeMounts:
            - name: vol-hostpath          # Volume name, which must be the same as
the volume name in the volumes field.
              mountPath: /data            # Mount path in the container.
      imagePullSecrets:
        - name: default-secret
      volumes:
        - name: vol-hostpath              # Volume name, which can be customized.
          hostPath:
            path: /data                   # Directory location on the host node.
```

Passo 3 Crie uma carga de trabalho.

kubectl apply -f nginx-hostpath.yaml

----Fim

8.10 StorageClass

Introdução

StorageClass descreve a classificação dos tipos de armazenamento em um cluster e pode ser representado como um modelo de configuração para criar PVs. Ao criar uma PVC ou PV, especifique StorageClass.

Como usuário, você só precisa especificar **storageClassName** ao definir uma PVC para criar automaticamente um PV e armazenamento subjacente, reduzindo significativamente a carga de trabalho de criação e manutenção de um PV.

Além das **classes de armazenamento padrão** fornecidas pelo CCE, você também pode personalizar classes de armazenamento.

- **Cenários de aplicações de armazenamento personalizado**
- **Classe de armazenamento personalizada**
- **Especificação de uma classe de armazenamento padrão**
- **Especificação de um projeto empresarial para classes de armazenamento**

Classes de armazenamento padrão do CCE

A partir de agora, o CCE fornece classes de armazenamento como csi-disk, csi-nas e csi-obs por padrão. Ao definir uma PVC, você pode usar um **storageClassName** para criar automaticamente um PV do tipo correspondente e criar automaticamente recursos de armazenamento subjacentes.

Execute o seguinte comando kubectl para obter as classes de armazenamento que o CCE suporta. Use o complemento CSI fornecido pelo CCE para criar uma classe de armazenamento.

```
# kubectl get sc
NAME                PROVISIONER                     AGE
csi-disk            everest-csi-provisioner         17d          # EVS disk
csi-disk-topology   everest-csi-provisioner         17d          # EVS disks
created with delayed
csi-nas             everest-csi-provisioner         17d          # SFS 1.0
csi-obs             everest-csi-provisioner         17d          # OBS
```



```
csi-sfsturbo        everest-csi-provisioner         17d          # SFS Turbo
csi-local           everest-csi-provisioner         17d          # Local PV
csi-local-topology  everest-csi-provisioner         17d          # Local PV
created with delay
```

Cada classe de armazenamento contém os parâmetros padrão usados para criar dinamicamente um PV. Veja a seguir um exemplo de classe de armazenamento para discos EVS:

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: csi-disk
provisioner: everest-csi-provisioner
parameters:
  csi.storage.k8s.io/csi-driver-name: disk.csi.everest.io
  csi.storage.k8s.io/fstype: ext4
  everest.io/disk-volume-type: SAS
  everest.io/passthrough: 'true'
reclaimPolicy: Delete
allowVolumeExpansion: true
volumeBindingMode: Immediate
```

Parâmetro	Descrição
provisioner	Especifica o provedor de recursos de armazenamento, que é o complemento everest para CCE. Defina este parâmetro como **everest-csi-provisioner**.
parameters	Especifica os parâmetros de armazenamento, que variam de acordo com os tipos de armazenamento.
reclaimPolicy	Especifica o valor de **persistentVolumeReclaimPolicy** para criar um PV. O valor pode ser **Delete** ou **Retain**. Se **reclaimPolicy** não for especificada quando um objeto de StorageClass for criado, o valor padrão será **Delete**. ● **Delete**: indica que um PV criado dinamicamente será destruído automaticamente. ● **Retain**: indica que um PV criado dinamicamente não será destruído automaticamente.
allowVolumeExpansion	Especifica se o PV desta classe de armazenamento suporta expansão de capacidade dinâmica. O valor padrão é **false**. A expansão de capacidade dinâmica é implementada pelo complemento de armazenamento subjacente. Este é apenas um interruptor.
volumeBindingMode	Especifica o modo de vinculação de volume, ou seja, o momento em que um PV é criado dinamicamente. O valor pode ser **Immediate** ou **WaitForFirstConsumer**. ● **Immediate**: a vinculação e a criação dinâmica de PV são concluídas quando uma PVC é criada. ● **WaitForFirstConsumer**: a vinculação e a criação do PV são atrasadas. Os processos de criação e encadernação de PV são executados somente quando a PVC é usada na carga de trabalho.

Parâmetro	Descrição
mountOptions	Este campo deve ser suportado pelo armazenamento subjacente. Se este campo não for suportado, mas for especificado, a criação do PV falhará.

Cenários de aplicações de armazenamento personalizado

Ao usar recursos de armazenamento no CCE, o método mais comum é especificar **storageClassName** para definir o tipo de recursos de armazenamento a serem criados ao criar uma PVC. A configuração a seguir mostra como usar uma PVC para solicitar um disco EVS (bloco de armazenamento) SAS (I/O alta).

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-evs-example
  namespace: default
  annotations:
    everest.io/disk-volume-type: SAS
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
  storageClassName: csi-disk
```

Para especificar o tipo de disco EVS no CCE, use o campo **everest.io/disk-volume-type**. SAS indica o tipo de disco EVS.

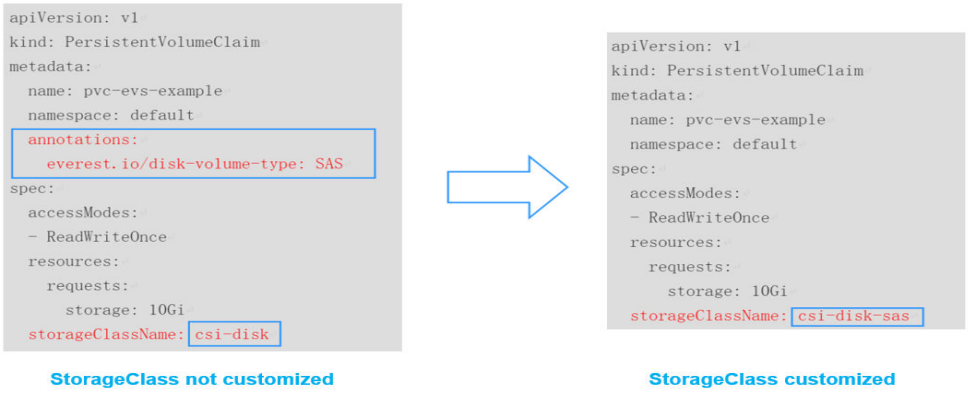
O anterior é um método básico de usar StorageClass. Em cenários do mundo real, você pode usar a StorageClass para executar outras operações.

Cenário de aplicação	Solução	Procedimento
Quando **anotações** são usadas para especificar a configuração de armazenamento, a configuração é complexa. Por exemplo, o campo **everest.io/disk-volume-type** é usado para especificar o tipo de disco EVS.	Defina anotações de PVC no campo de **parameters** de StorageClass. Ao compilar um arquivo YAML, você só precisa especificar **storageClassName**. Por exemplo, você pode definir o disco EVS SAS e o disco EVS SSD como uma classe de armazenamento, respectivamente. Se uma classe de armazenamento chamada **csi-disk-sas** for definida, ela será usada para criar armazenamento SAS.	**Classe de armazenamento personalizada**

Cenário de aplicação	Solução	Procedimento
Quando um usuário migra serviços de um cluster do Kubernetes criado automaticamente ou de outros serviços do Kubernetes para o CCE, a classe de armazenamento usada no arquivo YAML da aplicação original é diferente daquela usada no CCE. Como resultado, um grande número de arquivos YAML ou pacotes de gráficos Helm precisam ser modificados quando o armazenamento é usado, o que é complexo e propenso a erros.	Crie uma classe de armazenamento com o mesmo nome do arquivo YAML da aplicação original na centralização do CCE. Após a migração, não é necessário modificar o **storageClassName** no arquivo YAML da aplicação. Por exemplo, a classe de armazenamento em disco EVS usada antes da migração é **disk-standard**. Depois de migrar serviços para um cluster do CCE, você pode copiar o arquivo YAML da classe de armazenamento **csi-disk** no cluster do CCE, alterar seu nome para **disk-standard** e criar outra classe de armazenamento.	
storageClassName deve ser especificado no arquivo YAML para usar o armazenamento. Caso contrário, o armazenamento não pode ser criado.	Se você definir a StorageClass padrão no cluster, poderá criar armazenamento sem especificar **storageClassName** no arquivo YAML.	**Especificação de uma classe de armazenamento padrão**

Classe de armazenamento personalizada

Esta seção usa a classe de armazenamento personalizada de discos EVS como um exemplo para descrever como definir o disco EVS SAS e o disco EVS SSD como uma classe de armazenamento, respectivamente. Por exemplo, se você definir uma classe de armazenamento chamada **csi-disk-sas**, que é usada para criar armazenamento SAS, as diferenças serão mostradas na figura a seguir. Ao compilar um arquivo YAML, você só precisa especificar **storageClassName**.



- Você pode personalizar uma classe de armazenamento de alta I/O em um arquivo YAML. Por exemplo, o nome **csi-disk-sas** indica que o tipo de disco é SAS (alta I/O).

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: csi-disk-sas                            # Name of the high I/O storage
class, which can be customized.
parameters:
  csi.storage.k8s.io/csi-driver-name: disk.csi.everest.io
  csi.storage.k8s.io/fstype: ext4
   everest.io/disk-volume-type: SAS             # High I/O EVS disk type,
which cannot be customized.
  everest.io/passthrough: "true"
provisioner: everest-csi-provisioner
reclaimPolicy: Delete
volumeBindingMode: Immediate
allowVolumeExpansion: true                      # true indicates that capacity
expansion is allowed.
```

- Para uma classe de armazenamento de I/O ultra-alta, você pode definir o nome da classe como **csi-disk-ssd** para criar um disco EVS SSD (I/O ultra-alta).

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: csi-disk-ssd                          # Name of the ultra-high I/O
storage class, which can be customized.
parameters:
  csi.storage.k8s.io/csi-driver-name: disk.csi.everest.io
  csi.storage.k8s.io/fstype: ext4
  everest.io/disk-volume-type: SSD          # Ultra-high I/O EVS disk type,
which cannot be customized.
  everest.io/passthrough: "true"
provisioner: everest-csi-provisioner
reclaimPolicy: Delete
volumeBindingMode: Immediate
allowVolumeExpansion: true
```

reclaimPolicy: indica as políticas de recuperação do armazenamento em nuvem subjacente. O valor pode ser **Delete** ou **Retain**.

- **Delete**: quando uma PVC é excluída, ambos o PV e o disco EVS são excluídos.
- **Retain**: quando uma PVC é excluída, o PV e os recursos de armazenamento subjacentes não são excluídos. Em vez disso, você deve excluir manualmente esses recursos. Depois disso, o PV está no estado **Released** e não pode ser vinculado à PVC novamente.

Se for necessária uma alta segurança de dados, selecione **Retain** para impedir que os dados sejam excluídos por engano.

Após a conclusão da definição, execute os comandos **kubectl create** para criar recursos de armazenamento.


```
# kubectl create -f sas.yaml
storageclass.storage.k8s.io/csi-disk-sas created
# kubectl create -f ssd.yaml
storageclass.storage.k8s.io/csi-disk-ssd created
```

Consulte **StorageClass** novamente. A saída do comando é a seguinte:

```
# kubectl get sc
NAME                PROVISIONER                     AGE
csi-disk            everest-csi-provisioner         17d
csi-disk-sas        everest-csi-provisioner         2m28s
csi-disk-ssd        everest-csi-provisioner         16s
csi-disk-topology   everest-csi-provisioner         17d
csi-nas             everest-csi-provisioner         17d
csi-obs             everest-csi-provisioner         17d
csi-sfsturbo        everest-csi-provisioner         17d
```

Especificação de uma classe de armazenamento padrão

Você pode especificar uma classe de armazenamento como a classe padrão. Dessa forma, se você não especificar **storageClassName** ao criar um PVC, o PVC será criado usando a classe de armazenamento padrão.

Por exemplo, para especificar **csi-disk-ssd** como a classe de armazenamento padrão, edite seu arquivo YAML da seguinte forma:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: csi-disk-ssd
  annotations:
    storageclass.kubernetes.io/is-default-class: "true"   # Specifies the default
storage class in a cluster. A cluster can have only one default storage class.
parameters:
  csi.storage.k8s.io/csi-driver-name: disk.csi.everest.io
  csi.storage.k8s.io/fstype: ext4
  everest.io/disk-volume-type: SSD
  everest.io/passthrough: "true"
provisioner: everest-csi-provisioner
reclaimPolicy: Delete
volumeBindingMode: Immediate
allowVolumeExpansion: true
```

Exclua o disco csi-disk-ssd criado, execute o comando **kubectl create** para criar um disco csi-disk-ssd novamente e, em seguida, consulte a classe de armazenamento. As seguintes informações são exibidas.

```
# kubectl delete sc csi-disk-ssd
storageclass.storage.k8s.io "csi-disk-ssd" deleted
# kubectl create -f ssd.yaml
storageclass.storage.k8s.io/csi-disk-ssd created
# kubectl get sc
NAME                    PROVISIONER                     AGE
csi-disk                everest-csi-provisioner         17d
csi-disk-sas            everest-csi-provisioner         114m
csi-disk-ssd (default)  everest-csi-provisioner         9s
csi-disk-topology       everest-csi-provisioner         17d
csi-nas                 everest-csi-provisioner         17d
csi-obs                 everest-csi-provisioner         17d
csi-sfsturbo            everest-csi-provisioner         17d
```

Especificação de um projeto empresarial para classes de armazenamento

O CCE permite especificar um projeto empresarial ao criar discos EVS e PVCs do OBS. Os recursos de armazenamento criados (discos do EVS e do OBS) pertencem ao projeto

empresarial especificado. **O projeto empresarial pode ser o projeto empresarial ao qual o cluster pertence ou o projeto empresarial padrão.**

Se você não especificar nenhum projeto empresarial, o projeto empresarial em StorageClass será usado por padrão. Os recursos de armazenamento criados usando as classes de armazenamento csi-disk e csi-obs do CCE pertencem ao projeto empresarial padrão.

Se desejar que os recursos de armazenamento criados a partir das classes de armazenamento estejam no mesmo projeto empresarial que o cluster, você poderá personalizar uma classe de armazenamento e especificar o ID do projeto empresarial, conforme mostrado abaixo.

NOTA

Para usar essa função, o complemento everest deve ser atualizado para a versão 1.2.33 ou posterior.

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: csi-disk-epid      #Customize a storage class name.
provisioner: everest-csi-provisioner
parameters:
  csi.storage.k8s.io/csi-driver-name: disk.csi.everest.io
  csi.storage.k8s.io/fstype: ext4
  everest.io/disk-volume-type: SAS
  everest.io/enterprise-project-id: 86bfc701-9d9e-4871-a318-6385aa368183
#Specify the enterprise project ID.
  everest.io/passthrough: 'true'
reclaimPolicy: Delete
allowVolumeExpansion: true
volumeBindingMode: Immediate
```

Verificação

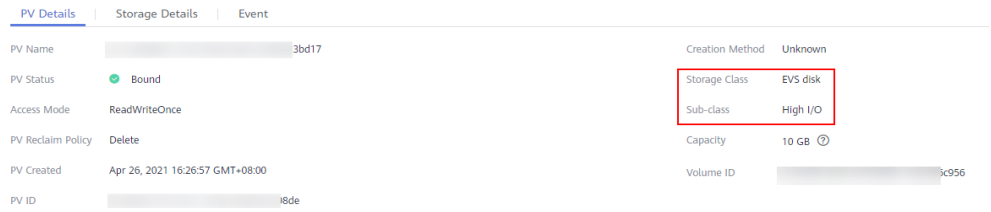
- Use **csi-disk-sas** para criar um PVC.

  ```
  apiVersion: v1
  kind: PersistentVolumeClaim
  metadata:
    name:  sas-disk
  spec:
    accessModes:
    - ReadWriteOnce
    resources:
      requests:
        storage: 10Gi
    storageClassName: csi-disk-sas
  ```

  Crie uma classe de armazenamento e exiba seus detalhes. Como mostrado abaixo, o objeto pode ser criado e o valor de **STORAGECLASS** é **csi-disk-sas**.

  ```
  # kubectl create -f sas-disk.yaml
  persistentvolumeclaim/sas-disk created
  # kubectl get pvc
  NAME       STATUS   VOLUME                                     CAPACITY
  ACCESS MODES   STORAGECLASS   AGE
  sas-disk   Bound    pvc-6e2f37f9-7346-4419-82f7-b42e79f7964c   10Gi
  RWO            csi-disk-sas   24s
  # kubectl get pv
  NAME                                       CAPACITY   ACCESS MODES   RECLAIM
  POLICY   STATUS      CLAIM                    STORAGECLASS   REASON   AGE
  pvc-6e2f37f9-7346-4419-82f7-b42e79f7964c   10Gi       RWO
  Delete           Bound       default/sas-disk          csi-disk-
  sas            30s
  ```

  Veja os detalhes do PVC no console do CCE. Na página de detalhes do PV, você pode ver que o tipo de disco é de I/O alta.



- Se **storageClassName** não for especificado, a configuração padrão será usada, conforme mostrado abaixo.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name:  ssd-disk
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
```

Crie e visualize o recurso de armazenamento. Você pode ver que a classe de armazenamento do PVC ssd-disk é csi-disk-ssd, indicando que o csi-disk-ssd é usado por padrão.

```
# kubectl create -f ssd-disk.yaml
persistentvolumeclaim/ssd-disk created
# kubectl get pvc
NAME       STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS   AGE
sas-disk   Bound    pvc-6e2f37f9-7346-4419-82f7-b42e79f7964c   10Gi
RWO            csi-disk-sas   16m
ssd-disk   Bound    pvc-4d2b059c-0d6c-44af-9994-f74d01c78731   10Gi
RWO            csi-disk-ssd   10s
# kubectl get pv
NAME                                       CAPACITY   ACCESS MODES   RECLAIM
POLICY   STATUS      CLAIM                   STORAGECLASS   REASON   AGE
pvc-4d2b059c-0d6c-44af-9994-f74d01c78731   10Gi       RWO
Delete           Bound       default/ssd-disk          csi-disk-
ssd            15s
pvc-6e2f37f9-7346-4419-82f7-b42e79f7964c   10Gi       RWO
Delete           Bound       default/sas-disk          csi-disk-
sas            17m
```

Veja os detalhes do PVC no console do CCE. Na página de detalhes do PV, você pode ver que o tipo de disco é de I/O ultra-alta.

9 Observabilidade

9.1 Registro em logs

9.1.1 Visão geral

O CCE permite que você configure políticas para coletar, gerenciar e analisar logs de carga de trabalho periodicamente para evitar que os logs sejam extra grandes.

- Usar o ICAgent:

  Por padrão, o ICAgent coleta saídas padrão do contêiner (logs stdout). Não há configuração necessária.

  Você também pode configurar o caminho para armazenar logs de contêiner ao criar uma carga de trabalho para que o ICAgent colete logs desse caminho.

  Você pode selecionar um dos seguintes modos para logs de contêiner:

  – hostPath: um caminho de host é montado para o caminho de contêiner especificado (caminho de montagem). No caminho do host do nó, você pode visualizar a saída dos logs do contêiner no caminho de montagem.

  – emptyDir: um caminho temporário do nó é montado no caminho especificado (caminho de montagem). Os dados de log que existem no caminho temporário, mas não são relatados pelo coletor ao AOM, desaparecerão depois que o pod for excluído.

- Usar o log-agent:

  O CCE fornece **log-agent** para relatar logs ao coletor de log do LTS. Depois de criar um cluster, você pode consultar e gerenciar regras de coleta de log na página **Logging** do console de cluster do CCE.

  Por padrão, os logs de stdout e os eventos do Kubernetes são coletados.

  Você pode entrar ao console do CCE, alcançar um conjunto, e escolher **Logging** no painel de navegação para ver se ICAgent ou log-agent foi instalado. Caso contrário, o CCE solicita que você instale ICAgent ou log-agent.

ICAgent vs log-agent

Tabela 9-1 ICAgent vs log-agent

Ferramenta de coleção	Localização de armazenamento de log	Conteúdo a ser coletado	Vantagens e desvantagens	Método de configuração
ICAgent	LTS	Saída padrão do contêiner Arquivo de contêiner Arquivo de nó	As políticas de coleta de logs e as cargas de trabalho são configuradas separadamente. Modificar políticas não afeta a execução do pod. Você pode especificar um contêiner cujos logs serão coletados. Suporta apenas nós do Docker. Atualmente, a coleção de log de arquivos de contêiner suporta apenas o driver de armazenamento Overlay2, não Device Mapper.	Crie uma política de coleta no LTS.
	AOM	Saída padrão do contêiner Arquivo de contêiner	Cada carga de trabalho precisa ser configurada separadamente. As políticas de coleta de logs são acopladas a pods. Modificar a política reiniciará o pod.	Crie uma política de coleta na carga de trabalho. Para mais detalhes, consulte **Uso do ICAgent para coletar logs de contêiner**.

Ferramenta de coleção	Localização de armazenamento de log	Conteúdo a ser coletado	Vantagens e desvantagens	Método de configuração
log-agent	LTS	Saída padrão do contêiner Arquivo de contêiner Arquivo de nó Evento do Kubernetes	As políticas de coleta de logs e as cargas de trabalho são configuradas separadamente. Modificar políticas não afeta a execução do pod. Você pode especificar um contêiner cujos logs serão coletados. Se o driver de armazenamento do nó for Device Mapper, o caminho deve ser o caminho de montagem do disco de dados do nó.	Crie uma política na página **Logging**. Para mais detalhes, consulte **Uso do log-agent para coletar logs de contêiner**.

9.1.2 Uso do ICAgent para coletar logs de contêiner

O CCE trabalha com o AOM para coletar logs de carga de trabalho. Ao criar um nó, o CCE instala o ICAgent para você (o DaemonSet chamado **icagent** no namespace do kube-system do cluster). Depois que o ICAgent coletar logs de carga de trabalho e os reportar ao AOM, você poderá exibir logs de carga de trabalho no console do CCE ou AOM.

Restrições

O ICAgent coleta apenas arquivos de log de texto ***.log**, ***.trace** e ***.out**.

Cobrança

O AOM oferece uma cota de coleta de registros gratuita de 500 MB para cada conta todos os meses. Se a cota for excedida, você será cobrado. Para detalhes, veja **Cobrança**. Você pode clicar **aqui** para visualizar os logs no console do AOM.

Usar o ICAgent para coletar logs

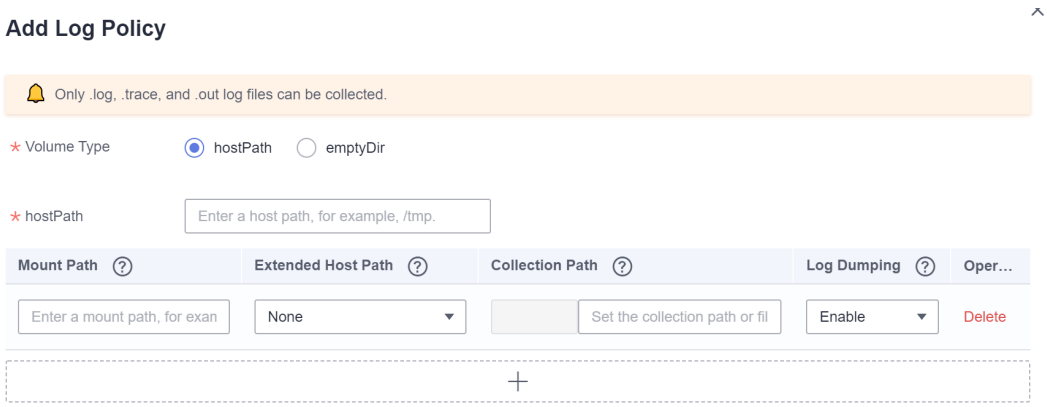
Você pode adicionar uma política para coletar logs usando o ICAgent para uma carga de trabalho.

Passo 1 Ao **criar uma carga de trabalho**, defina o registro em logs para o contêiner.

Passo 2 Clique em + para adicionar uma política de log.

O seguinte usa Nginx como um exemplo.As políticas de log variam dependendo das cargas de trabalho.

Figura 9-1 Adicionar uma política de log



Passo 3 Defina **Storage Type** como **Host Path** ou **Container Path**.

Tabela 9-2 Configurar políticas de log

Parâmetro	Descrição
Storage Type	● **Host Path** (hostPath): um caminho de host é montado para o caminho de contêiner especificado (caminho de montagem). No caminho do host do nó, você pode visualizar a saída dos logs do contêiner no caminho de montagem. ● **Container Path** (emptyDir): um caminho temporário do nó é montado no caminho especificado (caminho de montagem). Os dados de log que existem no caminho temporário, mas não são relatados pelo coletor ao AOM, desaparecerão depois que o pod for excluído.
Host Path	Insira um caminho de host, por exemplo, **/var/paas/sys/log/nginx**.
Container Path	Caminho do contêiner (por exemplo, **/tmp**) no qual os recursos de armazenamento serão montados. **AVISO** ● Não monte o armazenamento em um diretório do sistema, como / ou **/var/run**; essa ação pode causar um erro de contêiner. É aconselhável montar o contêiner em um diretório vazio. Se o diretório não estiver vazio, verifique se não há arquivos afetando a inicialização do contêiner no diretório. Caso contrário, esses arquivos serão substituídos, resultando em falhas para iniciar o contêiner e criar a carga de trabalho. ● Quando o contêiner é montado em um diretório de alto risco, é aconselhável usar uma conta com permissões mínimas para iniciar o contêiner; caso contrário, os arquivos de alto risco no computador host podem ser danificados. ● O AOM coleta apenas os primeiros 20 arquivos de log que foram modificados recentemente. Ele coleta arquivos de 2 níveis de subdiretórios por padrão. ● O AOM apenas coleta arquivos de log de texto **.log**, **.trace** e **.out** nos caminhos de montagem. ● Para obter detalhes sobre como definir permissões para pontos de montagem em um contêiner, consulte **Configuração de um contexto de segurança para um pod ou contêiner**.

Parâmetro	Descrição
Extended Host Path	Esse parâmetro é obrigatório somente quando **Storage Type** estiver definido como **Host Path**. Caminhos de host estendidos contêm IDs de pod ou nomes de contêiner para distinguir diferentes contêineres nos quais o caminho do host é montado. Um diretório de nível 3 é adicionado ao diretório/subdiretório do volume original. Você pode facilmente obter os arquivos de saída por um único Pod. ● **None**: nenhum caminho estendido está configurado. ● **PodUID**: ID de um pod. ● **PodName**: nome de um casulo. ● **PodUID/ContainerName**: ID de um pod ou nome de um contêiner. ● **PodName/ContainerName**: nome de um pod ou contêiner
Collection Path	Um caminho de coleta restringe o escopo da coleta para logs especificados. ● Se nenhum caminho de coleta for especificado, os arquivos de log nos formatos **.log**, **.trace** e **.out** serão coletados do caminho especificado. ● **/Path/**/*/** indica que todos os arquivos de log nos formatos **.log**, **.trace** e **.out** serão coletados recursivamente do caminho especificado e de todos os subdiretórios em 5 níveis de profundidade. ● * em nomes de arquivos de log indica uma correspondência difusa. Exemplo: o caminho de coleta **/tmp/**/test*.log** indica que todos os arquivos **.log** prefixados com **test** serão coletados de **/tmp** e subdiretórios de profundidade em 5 níveis. **CUIDADO** Certifique-se de que a versão do **ICAgent** seja 5.12.22 ou posterior.

Parâmetro	Descrição
Despejo de log	O despejo de log refere-se à rotação de arquivos de log em um host local. ● **Enabled**: o AOM verifica os arquivos de log a cada minuto. Quando um arquivo de log excede 50 MB, ele é despejado. Um novo arquivo **.zip** é gerado no diretório em que o arquivo de log está localizado. Para um arquivo de log, o AOM armazena somente os 20 arquivos **.zip** mais recentes. Quando o número de arquivos **.zip** exceder 20, os arquivos **.zip** anteriores serão excluídos. ● **Disabled**: o AOM não despeja arquivos de log. **NOTA** ● O AOM rotaciona arquivos de log usando copytruncate. Antes de ativar o despejo de log, certifique-se de que os arquivos de log sejam gravados no modo de acréscimo. Caso contrário, podem ocorrer furos de arquivo. ● Atualmente, os principais componentes de log, como Log4j e Logback, suportam a rotação de arquivos de log. Se você já definiu a rotação para arquivos de log, ignore a configuração. Caso contrário, conflitos podem ocorrer. ● É aconselhável configurar a rotação de arquivos de log para seus próprios serviços para controlar de forma flexível o tamanho e o número de arquivos rolados.

Passo 4 Clique em **OK**.

----Fim

Exemplo YAML (ICAgent)

Você pode definir o caminho de armazenamento do log do contêiner definindo um arquivo YAML.

Como mostrado na figura a seguir, um volume emptyDir é montado um caminho temporário para **/var/log/nginx**. Dessa forma, o ICAgent coleta logs em **/var/log/nginx**. O campo **policy** é personalizado pelo CCE e permite que o ICAgent identifique e colete logs.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: testlog
  namespace: default
spec:
  selector:
    matchLabels:
      app: testlog
  template:
    replicas: 1
    metadata:
      labels:
        app: testlog
    spec:
      containers:
        - image: 'nginx:alpine'
          name: container-0
          resources:
            requests:
```



```
              cpu: 250m
              memory: 512Mi
            limits:
              cpu: 250m
              memory: 512Mi
          volumeMounts:
            - name: vol-log
              mountPath: /var/log/nginx
              policy:
                logs:
                  rotate: ''
      volumes:
        - emptyDir: {}
          name: vol-log
      imagePullSecrets:
        - name: default-secret
```

A seguir mostra como usar um volume de hostPath. Comparado com emptyDir, o tipo de **volumes** é alterado para **hostPath** e o caminho no host precisa ser configurado para esse volume de hostPath. No exemplo a seguir, **/tmp/log** no host é montado em **/var/log/nginx**. Desta forma, o ICAgent pode coletar logs em **/var/log/nginx**, sem excluir os logs de **/tmp/log**.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: testlog
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: testlog
  template:
    metadata:
      labels:
        app: testlog
    spec:
      containers:
        - image: 'nginx:alpine'
          name: container-0
          resources:
            requests:
              cpu: 250m
              memory: 512Mi
            limits:
              cpu: 250m
              memory: 512Mi
          volumeMounts:
            - name: vol-log
              mountPath: /var/log/nginx
              readOnly: false
              extendPathMode: PodUID
              policy:
                logs:
                  rotate: Hourly
                  annotations:
                    pathPattern: '**'
                    format: ''
      volumes:
        - hostPath:
            path: /tmp/log
          name: vol-log
      imagePullSecrets:
        - name: default-secret
```


Tabela 9-3 Descrição do parâmetro

Parâmetro	Descrição	Descrição
extendPathMode	Caminho do host estendido	Caminhos de host estendidos contêm IDs de pod ou nomes de contêiner para distinguir diferentes contêineres nos quais o caminho do host é montado. Um diretório de nível 3 é adicionado ao diretório/subdiretório do volume original. Você pode facilmente obter os arquivos de saída por um único Pod. ● **None**: nenhum caminho estendido está configurado. ● **PodUID**: ID de um pod. ● **PodName**: nome de um casulo. ● **PodUID/ContainerName**: ID de um pod ou nome de um contêiner. ● **PodName/ContainerName**: nome de um pod ou contêiner
policy.logs.rotate	Despejo de log	O despejo de log refere-se à rotação de arquivos de log em um host local. ● **Enabled**: o AOM verifica os arquivos de log a cada minuto. Quando um arquivo de log excede 50 MB, ele é despejado imediatamente. Um novo arquivo **.zip** é gerado no diretório em que o arquivo de log está localizado. Para um arquivo de log, o AOM armazena somente os 20 arquivos **.zip** mais recentes. Quando o número de arquivos **.zip** exceder 20, os arquivos **.zip** anteriores serão excluídos. Depois que o despejo for concluído, o arquivo de log no AOM será limpo. ● **Disabled**: o AOM não despeja arquivos de log. **NOTA** ● O AOM rotaciona arquivos de log usando copytruncate. Antes de ativar o despejo de log, certifique-se de que os arquivos de log sejam gravados no modo de acréscimo. Caso contrário, podem ocorrer furos de arquivo. ● Atualmente, os principais componentes de log, como Log4j e Logback, suportam a rotação de arquivos de log. Se você definiu a rotação para os arquivos de log, ignore a configuração. Caso contrário, conflitos podem ocorrer. ● É aconselhável configurar a rotação de arquivos de log para seus próprios serviços para controlar de forma flexível o tamanho e o número de arquivos rolados.

Parâmetro	Descrição	Descrição
policy.logs.annotations.pathPattern	Caminho da coleção	Um caminho de coleta restringe o escopo da coleta para logs especificados. ● Se nenhum caminho de coleta for especificado, os arquivos de log nos formatos **.log**, **.trace** e **.out** serão coletados do caminho especificado. ● **/Path/**/** indica que todos os arquivos de log nos formatos **.log**, **.trace** e **.out** serão coletados recursivamente do caminho especificado e de todos os subdiretórios em profundidade de 5 níveis. ● * em nomes de arquivos de log indica uma correspondência difusa. Exemplo: o caminho de coleta **/tmp/**/test*.log** indica que todos os arquivos **.log** prefixados com **test** serão coletados de **/tmp** e subdiretórios em profundidade de 5 níveis. **CUIDADO** Certifique-se de que a versão do **ICAgent** seja 5.12.22 ou posterior.
policy.logs.annotations.format	Correspondência de log de várias linhas	Alguns logs de programas (por exemplo, logs de programas Java) contêm um log que ocupa várias linhas. Por padrão, o sistema de coleta de logs coleta logs por linha. Se você quiser exibir logs como uma única mensagem de log no sistema de coleta de log, você pode ativar a função de log de várias linhas e usar o modo de tempo de log ou padrão regular. Quando uma linha de mensagem de log corresponde ao formato de hora predefinido ou expressão regular, ela é considerada como o início de uma mensagem de log e a próxima linha começa com essa linha de mensagem de log é considerada como o identificador final da mensagem de log. O formato é o seguinte: `{ "multi": { "mode": "time", "value": "YYYY-MM-DD hh:mm:ss" } }` **multi** indica o modo multi-linha. ● **time**: tempo de log. Insira um curinga de tempo. Por exemplo, se a hora no registro for 2017-01-01 23:59:59, o curinga será AAAA-MM-DD hh:mm:ss. ● **regular**: padrão regular. Insira uma expressão regular.

Visualização de logs

Depois que um caminho de coleta de log é configurado e a carga de trabalho é criada, o ICAgent coleta arquivos de log do caminho configurado. A coleta leva cerca de 1 minuto.

Após a conclusão da coleta de logs, vá para a página de detalhes da carga de trabalho e clique em **Logs** no canto superior direito para exibir logs.

Você também pode exibir logs no console do AOM.

Você também pode executar o comando **kubectl logs** para exibir a saída padrão de um contêiner.

```
# View logs of a specified pod.
kubectl logs <pod_name>
kubectl logs -f <pod_name> # Similar to tail -f

# View logs of a specified container in a specified pod.
kubectl logs <pod_name> -c <container_name>

kubectl logs pod_name -c container_name -n namespace (one-off query)
kubectl logs -f <pod_name> -n namespace (real-time query in tail -f mode)
```

9.1.3 Uso do log-agent para coletar logs de contêiner

Restrições

As restrições sobre o uso do complemento log-agent são as seguintes:

- O log-agent está disponível somente em clusters da v1.17 ou posterior.
- Um máximo de 50 regras de log podem ser configuradas para cada cluster.
- O log-agent não pode coletar arquivos de log .gz, .tar ou .zip.
- Durante a coleta de log de arquivo de contêiner, se o driver de armazenamento Device Mapper for usado em nós, o caminho deverá ser o caminho de montagem do disco de dados do nó.
- Se o tempo de execução do contêiner for containerd, os logs de stdout não podem ser multilinhas.
- Em cada cluster, a taxa de coleta de uma única linha de logs não pode exceder 10.000 de registros por segundo e a taxa de coleta de várias linhas de logs não pode exceder 2.000 registros por segundo.

Cobrança

O Log Tank Service (LTS) não cobra pela criação de grupos de logs e oferece uma cota gratuita de um determinado valor para a coleta de logs todos os meses. Você paga apenas pelo volume de log que excede a cota (verifique **Calculadora de preços**).

Usar log-agent para coletar logs

O CCE integra o LTS para coletar logs de recursos, como contêineres e nós, e relatar os logs ao LTS de acordo com as políticas de log definidas por você.

Passo 1 Entre ao console do CCE, alcance um conjunto, e escolha **Logging** no painel de navegação.

Se **log-agent** não tiver sido instalado, instale-o conforme solicitado ou vá para a página **Add-ons** para instalar o log-agent.

Passo 2 Crie um grupo de log do LTS e um fluxo de log.

- Um grupo de logs é a unidade básica para o LTS gerenciar logs. Se você não tiver um grupo de logs, o CCE solicitará que você crie um. O nome padrão é **k8s-log-{*Cluster ID*}**, por exemplo, k8s-log-bb7eaa87-07dd-11ed-ab6c-0255ac1001b3.

- Um fluxo de log é a unidade básica para ler e gravar logs. Você pode colocar diferentes tipos de logs em diferentes fluxos para facilitar o gerenciamento. Na página **Logging** do console do CCE, você é solicitado a criar um fluxo de log. Os logs de serviço coletados pelo CCE são classificados nos seguintes tipos:
  - **Container standard output**: o nome padrão é **stdout-*{Cluster ID}***, por exemplo, **stdout-bb7eaa87-07dd-11ed-ab6c-0255ac1001b3**.
  - **Container file log**: o nome padrão é **containerfile-*{Cluster ID}***, por exemplo, **containerfile-bb7eaa87-07dd-11ed-ab6c-0255ac1001b3**.
  - **Node log**: o nome padrão é **hostfile-*{Cluster ID}***, por exemplo, **hostfile-bb7eaa87-07dd-11ed-ab6c-0255ac1001b3**.
  - **K8s event**: o nome padrão é **event-*{Cluster ID}***, por exemplo, **event-bb7eaa87-07dd-11ed-ab6c-0255ac1001b3**.

Passo 3 Crie uma política de coleta de log.

Depois que o complemento log-agent é instalado, a política de saída de contêiner padrão **default-stdout** e a política de eventos do Kubernetes **default-event** são geradas por padrão.

Na página **Logging**, clique em **View Log Policy** no canto superior direito e edite-a, ou clique em **Create Log Policy** no canto superior direito e defina os parâmetros relacionados.

Figura 9-2 Criar uma política de WAF

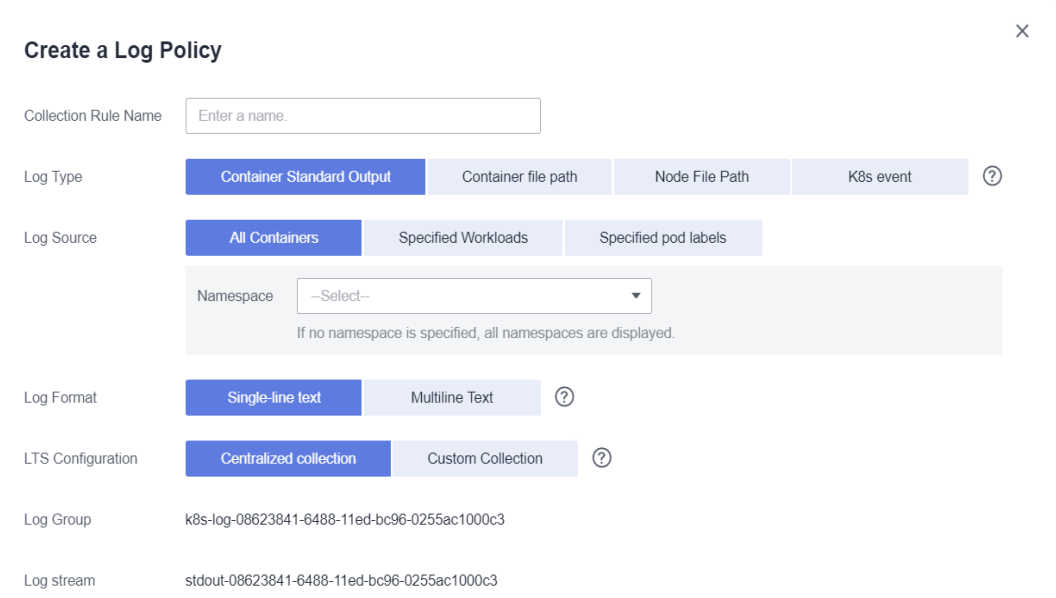


Tabela 9-4 Parâmetros de política de HPA

Parâmetro	Descrição
Log Type	Tipo de logs a serem coletados. ● **Container standard output** ● **Container file** ● **Node file** ● Kubernetes event

Parâmetro	Descrição
Log Source	Contêineres cujos logs devem ser coletados. ● **All containers**: você pode especificar um namespace e coletar seus logs de contêiner. Se esse parâmetro não for especificado, os contêineres em todos os namespaces serão coletados. ● **Specified workloads**: você pode especificar um contêiner e coletar seus logs ou pode especificar logs a serem coletados. Se este parâmetro não for especificado, os logs de todos os contêineres serão coletados. ● **Specified pod labels**: você pode especificar um contêiner e coletar seus logs com base em um rótulo ou pode especificar os logs a serem coletados. Se esse parâmetro não for especificado, os logs de todos os contêineres serão coletados.
Log Format	● **Single-line text** Cada log contém apenas uma linha de texto. O caractere de nova linha **\n** é usado como limite de cada log. ● **Multi-line text** Alguns logs de programas (por exemplo, logs de programas Java) contêm um log que ocupa várias linhas. Por padrão, o sistema de coleta de logs coleta logs por linha. Se você quiser exibir logs como uma única mensagem de log no sistema de coleta de log, você pode ativar a função de texto com várias linhas e usar o padrão regular. Se você selecionar o texto de várias linhas, insira o formato de correspondência de log. Exemplo: Se o formato dos logs a serem coletados for o seguinte, digite **\d{4}-\d{2}-\d{2} \d{2}\:\d{2}\:\d{2}.***. As seguintes três linhas que começam com a data são consideradas um log completo. 2022-01-01 00:00:00 Exception in thread "main" java.lang.RuntimeException: Something has gone wrong, aborting! at com.myproject.module.MyProject.badMethod(MyProject.java:22) at com.myproject.module.MyProject.oneMoreMethod(MyProject.java:18) 2022-01-01 00:00:10 Normal
LTS Configuration	● **Centralized collection**: todos os logs são armazenados no fluxo de log criado em**Passo 2**. ● **Custom collection**: especifique um fluxo de log para armazenar logs.

Passo 4 Depois que a política de registro for criada, você poderá clicar em **View Log Policy** no canto superior direito para exibir ou editar a política de log.

----Fim

Visualização de logs

Depois que os logs são coletados, faça logon no console do CCE, acesse um cluster e escolha **Logging** no painel de navegação. Na página de guia **Service Logs**, visualize logs por tipo de log. Você pode ver logs no console do LTS como você faz no console do CCE. Para obter detalhes, consulte **Guia de usuário do LTS**.

Você também pode acessar o console do LTS e encontrar o grupo de logs (chamado **k8s-log-*{Cluster ID}***) correspondente ao cluster para exibir os logs.

9.1.4 Exibição de logs do plano de controle de cluster

O CCE agora suporta o registro em logs de nós principais. Na página **Logging**, você pode selecionar o tópico de registro a ser relatado. Atualmente, **kube-controller-manager**, **kube-apiserver**, **kube-scheduler** e **audit** são suportados.

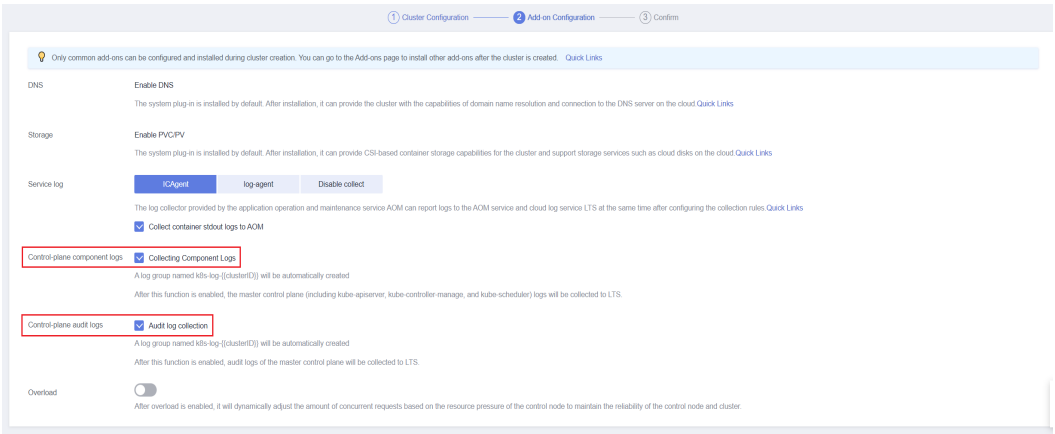
Restrições

- Para exibir os logs do plano de controle do cluster, a versão secundária do cluster deve ser v1.21.7-r0, v1.23.5-r0 ou 1.25.
- Certifique-se de que você tenha uma cota de recursos do LTS suficiente. Para obter detalhes sobre a cota do LTS padrão, consulte **Recursos básicos**.

Ativar o registro em logs do plano de controle

Método 1: habilitá-lo ao criar um cluster

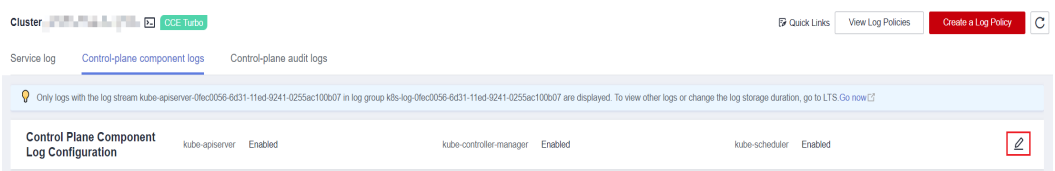
Ao criar um cluster de v1.21 ou posterior, ative **Control-plane component logs** e **Control-plane audit logs** na **Add-on Configuration**.



Método 2: habilitá-lo em um cluster existente

Passo 1 Entre ao console do CCE, alcance um conjunto e escolha **Logging** no painel de navegação.

Passo 2 Clique na guia **Control-plane component logs** ou **Control-plane audit logs** e modifique a configuração do registro.



Passo 3 Especifique se deseja ativar o log para cada componente e clique em ✔.

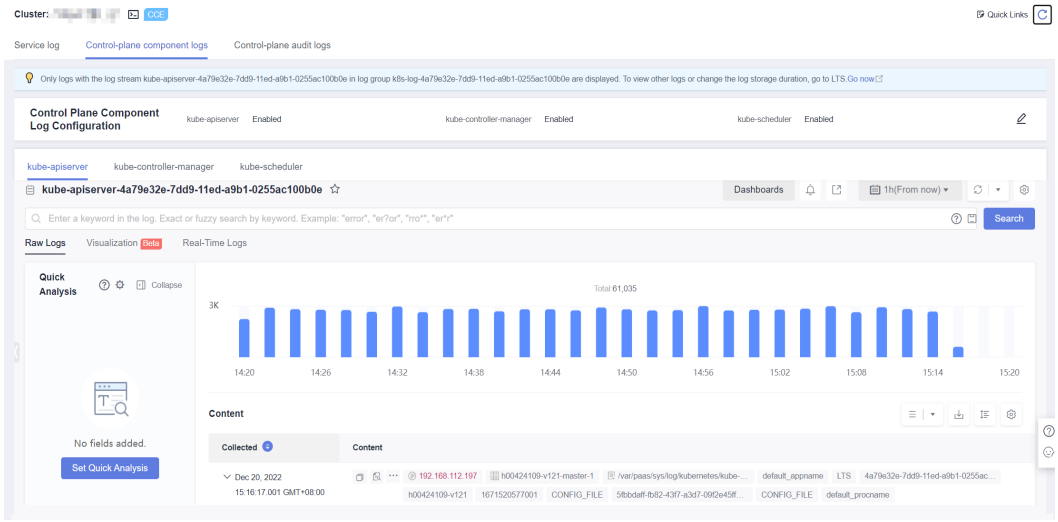
----Fim

Exibição de logs do plano de controle de cluster

Método 1: visualizá-los no console do CCE

Passo 1 Entre ao console do CCE, alcance um conjunto, e escolha **Logging** no painel de navegação.

Passo 2 Clique na guia **Control-plane component logs** ou **Control-plane audit logs** e selecione o tópico de log a ser exibido. Para obter detalhes sobre o log de componentes do plano de controle suportado, consulte **Componentes no plano de controle de cluster**. Para obter detalhes, consulte ***Guia de usuário do Log Tank Service***.

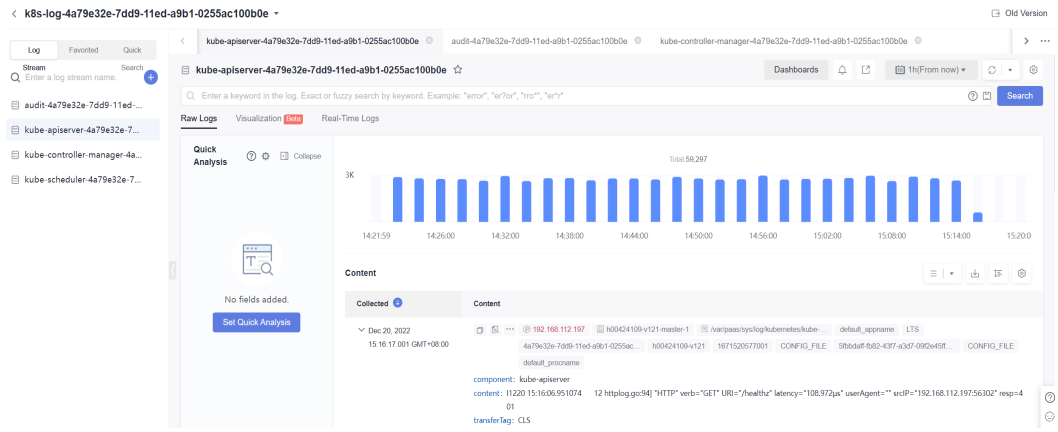


----Fim

Método 2: visualizá-los no console do LTS

Passo 1 Efetue logon no console LTS e escolha **Log Management**.

Passo 2 Consulte o grupo de logs com base no ID do cluster e clique no nome do grupo de logs para exibir o fluxo de logs. Para obter detalhes, consulte ***Guia de usuário do Log Tank Service***.



----Fim

Componentes no plano de controle de cluster

O CCE pode coletar os seguintes tipos de logs de plano de controle. Cada fluxo de log corresponde a um componente de plano de controle do Kubernetes. Para obter detalhes, consulte **Componentes do Kubernetes**.

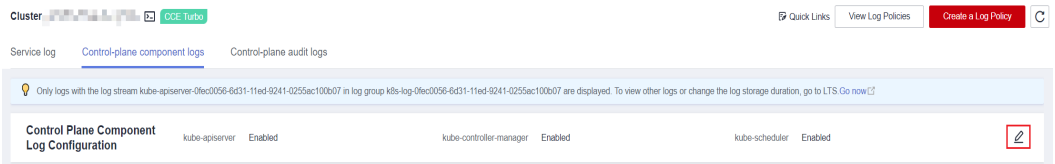
Tabela 9-5 Componentes no plano de controle do cluster

Tipo	Componente	Fluxo de log	Descrição
Logs de componentes do plano de controle	kube-apiserver	kube-apiserver-{{clusterID}}	Expõe as APIs do Kubernetes. Para obter mais informações, consulte **kube-apiserver**.
	kube-controller-manager	kube-controller-manager-{{clusterID}}	Executa processos do controlador e incorpora lógica de controle específica da nuvem. Para obter mais informações, consulte **kube-controller-manager**.
	kube-scheduler	kube-scheduler-{{clusterID}}	O agendador padrão de um cluster do Kubernetes. Para obter mais informações, consulte **kube-scheduler**.
Logs de auditoria do plano de controle	audit	audit-{{clusterID}}	Os logs de auditoria fornecem registros relacionados à segurança em ordem cronológica. Eles registram as operações do usuário e controlam as atividades do avião.

Desativar o log do plano de controle de cluster

Passo 1 Entre ao console do CCE, alcance um conjunto, e escolha **Logging** no painel de navegação.

Passo 2 Clique na guia **Control-plane component logs** ou **Control-plane audit logs** e modifique a configuração do registro.



Passo 3 Especifique se deseja ativar o log para cada componente e clique em ✔.

NOTA

Depois de desabilitar o log de plano de controle de cluster, o fluxo de log original não atualizará logs, mas os logs existentes não serão excluídos e taxas podem ser incorridas para isso. Para obter detalhes, consulte **Calculadora de preço**.

----Fim

Excluir logs do plano de controle de cluster

Os logs do plano de controle de cluster são armazenados nos fluxos de log do LTS. Depois que o log de plano de controle de cluster estiver desabilitado, os fluxos de log originais não atualizarão os logs, mas os logs existentes não serão excluídos. Se você quiser excluir logs existentes, vá para o console do LTS, procure pelo grupo de logs chamado **k8s-log-*{Cluster ID}*** e exclua o fluxo de log correspondente.

9.2 Monitoramento

9.2.1 Visão geral do monitoramento

O CCE trabalha com o AOM para monitorar de forma abrangente os clusters. Quando um nó é criado, o ICAgent (o DaemonSet chamado **icagent** no namespace kube-system do cluster) do AOM é instalado por padrão. O ICAgent coleta dados de monitoramento de recursos subjacentes e cargas de trabalho em execução no cluster. Ele também coleta dados de monitoramento de métricas personalizadas da carga de trabalho.

- Métricas de recursos

  O monitoramento básico de recursos inclui monitoramento de CPU, memória e disco. Para mais detalhes, consulte **Métricas de recursos**. Você pode exibir essas métricas de clusters, nós e cargas de trabalho no console do CCE ou AOM.

- Métricas personalizadas

  O ICAgent coleta métricas personalizadas de aplicações e as carrega no AOM. Para mais detalhes, consulte **Monitoramento de métricas personalizadas no AOM**.

- Monitoramento de NPD

  node-problem-detector (npd para breve) é um complemento que monitora e relata a integridade de um nó. Ele pode se conectar a uma plataforma de monitoramento de terceiros. É um daemon em execução em cada nó. Ele coleta problemas de nó de diferentes daemons e os reporta ao servidor da API. O complemento NPD pode ser executado como um daemon ou DaemonSet.

  O CCE aprimora o npd na versão 1.16.0, que agora suporta verificações em recursos de nó, componentes e eventos, bem como isolamento de falhas. Para mais detalhes, consulte **Detector de problema de nó do CCE**.

Além disso, você pode instalar o complemento Prometheus em um cluster e usar o Prometheus para coletar e exibir dados de monitoramento. Para mais detalhes, consulte **Monitoramento de métricas personalizadas usando o Prometheus**.

Métricas de recursos

No console do CCE, você pode exibir as seguintes métricas.

- **Exibir dados de monitoramento de cluster**
- **Visualizar dados de monitoramento de nós de trabalho**
- **Exibir dados de monitoramento da carga de trabalho**
- **Visualizar dados de monitoramento de pods**

No console do AOM, você pode exibir métricas do host e métricas do contêiner. Para obter detalhes, consulte **Visão geral de métrica**.

Exibir dados de monitoramento de cluster

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster.

Passo 2 O CCE permite que você visualize os dados de monitoramento de todos os nós. Escolha **Clusters** no painel de navegação. Clique no nome do cluster e informações como **CPU Metrics** e **Memory** de todos os nós (excluindo nós principais) na última hora, o **Status**, a **AZ** são exibidos.

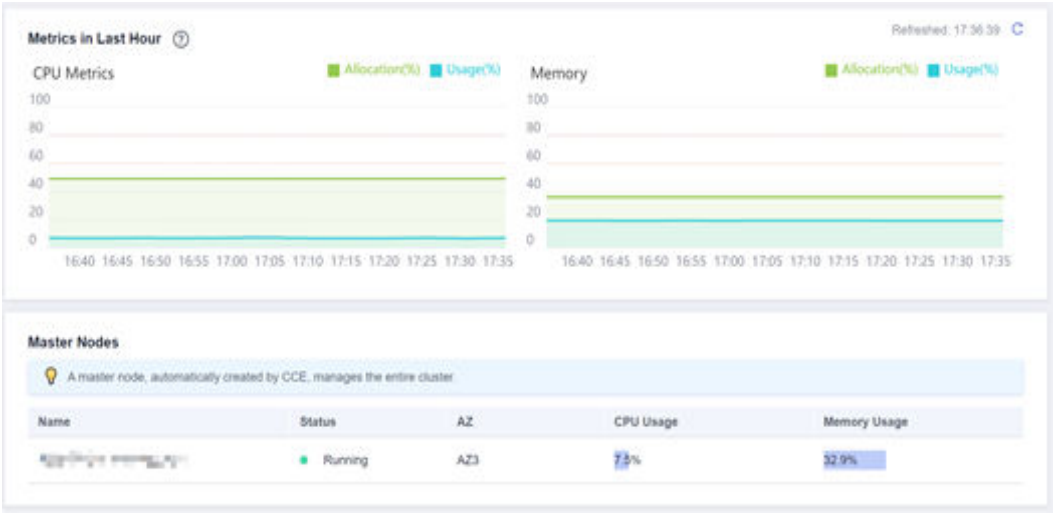
Tabela 9-6 Métricas de monitoramento de cluster

Métrica	Descrição
CPU Allocation (%)	Uma métrica indica a porcentagem de CPUs alocadas às cargas de trabalho **CPU Allocation (%)** = soma das cotas de CPU solicitadas pela execução de pods no cluster/soma das cotas de CPU que podem ser alocadas de todos os nós (excluindo os nós principais) para cargas de trabalho
Memory Allocation (%)	Uma métrica indica a porcentagem de memória alocada às cargas de trabalho **Memory Allocation (%)** = soma das cotas de memória solicitadas pela execução de pods no cluster/soma das cotas de memória que podem ser alocadas de todos os nós (excluindo os nós principais) para cargas de trabalho
CPU Usage (%)	Uma métrica indica o uso da CPU do cluster Essa métrica é o uso médio da CPU de todos os nós (excluindo os nós mestres) em um cluster.
Memory Usage (%)	Uma métrica indica o uso de memória do cluster Essa métrica é o uso médio de memória de todos os nós (excluindo os nós mestres) em um cluster.

NOTA

Recursos de nó alocáveis (CPU ou memória) = valor total – valor reservado – limites de despejo. Para mais detalhes, consulte **Política de reserva de recursos de nó**.

Figura 9-3 Visualizar dados de monitoramento de cluster



----Fim

Visualizar dados de monitoramento de nós de trabalho

O CCE também permite que você visualize os dados de monitoramento de um único nó.

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster.

Passo 2 Escolha **Nodes** no painel de navegação. À direita da página, clique em **Monitor** do nó de destino para exibir os dados de monitoramento.

Passo 3 Você pode selecionar **Dimension** estatística e escolher o intervalo de tempo para exibir os dados de monitoramento. Os dados são fornecidos pelo AOM. Você pode visualizar os dados de monitoramento de um nó, incluindo a CPU, a memória, o disco, a rede e a GPU.

Figura 9-4 Exibição de dados de monitoramento de nós de trabalho

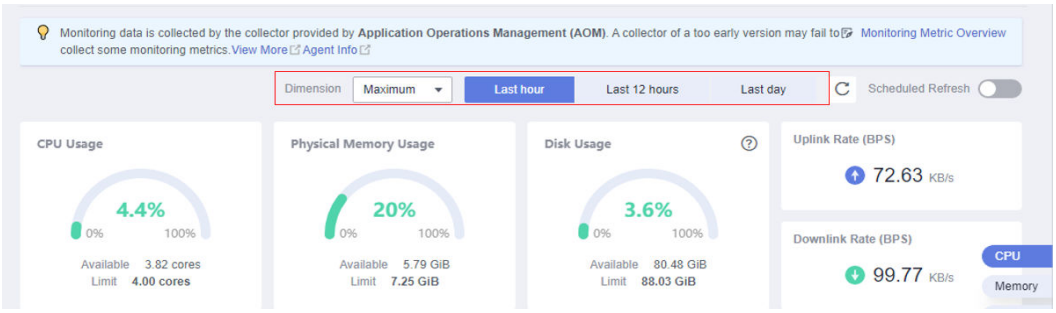


Tabela 9-7 Métricas de monitoramento do nó

Métrica	Descrição
CPU Usage (%)	Uma métrica indica o uso da CPU do nó **CPU Usage (%)** = núcleos de CPU usados/número total de núcleos de CPU

Métrica	Descrição
Used CPU Cores (cores)	Uma métrica indica o número de núcleos de CPU usados
Physical Memory Usage (%)	Uma métrica indica o uso de memória física do nó **Physical Memory Usage (%)** = (capacidade da memória física – memória física disponível)/capacidade da memória física
Available Physical Memory (GiB)	Uma métrica indica a memória física não utilizada do nó
Disk Usage (%)	Uma métrica indica o uso do disco do sistema de arquivos no disco de dados do nó. É calculada com base na partição do arquivo. Para mais detalhes, consulte **Alocação de espaço em disco de dados**. **Disk Usage (%)** = (capacidade do disco – espaço em disco disponível)/ capacidade do disco
Available Disk Space (GiB)	Uma métrica indica o espaço em disco não utilizado
Downlink Rate (BPS) (KB/s)	Uma métrica indica a velocidade na qual os dados são baixados da Internet para o nó
Uplink Rate (BPS) (KB/s)	Uma métrica indica a velocidade na qual os dados são carregados do nó para a Internet
GPU Usage (%)	Uma métrica indica o uso da GPU do nó
GPU Memory Usage (%)	Uma métrica indica a porcentagem da memória da GPU usada em relação à capacidade de memória da GPU **GPU Memory Usage (%)** = memória usada da GPU/capacidade da memória da GPU
Used GPU Memory (GiB)	Uma métrica indica a memória da GPU usada

----Fim

Exibir dados de monitoramento da carga de trabalho

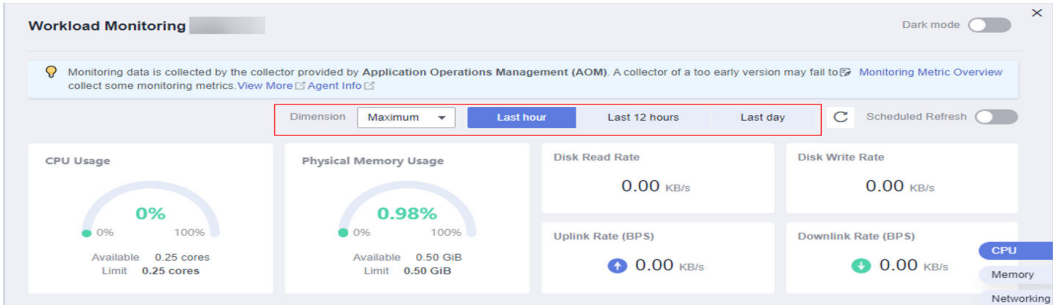
O CCE permite que você visualize os dados de monitoramento de uma única carga de trabalho.

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster.

Passo 2 Escolha **Workloads** no painel de navegação. À direita da página, clique em **Monitor** da carga de trabalho de destino. Na janela que desliza para fora da direita, os dados de monitoramento da carga de trabalho são exibidos.

Passo 3 Você pode selecionar **Dimension** estatística e escolher o intervalo de tempo para exibir os dados de monitoramento. Os dados são fornecidos pelo AOM. Você pode visualizar os dados de monitoramento de uma carga de trabalho, incluindo CPU, memória, rede e GPU.

Figura 9-5 Exibição de dados de monitoramento de carga de trabalho



NOTA

Se existirem vários pods na carga de trabalho, os dados de monitoramento podem variar de acordo com a **Dimension** estatística. Por exemplo, se você selecionar **Maximum** ou **Minimum** para **Dimension**, o valor de cada dado de monitoramento será o valor máximo ou mínimo de todos os pods sob a carga de trabalho. Se **Average** for selecionada, o valor de cada dado de monitoramento é o valor médio de todos os pods sob a carga de trabalho.

Tabela 9-8 Métricas de monitoramento da carga de trabalho

Métrica	Descrição
CPU Usage (%)	Uma métrica indica o uso da CPU da carga de trabalho **CPU Usage (%)** = núcleos de CPU usados/número total de núcleos de CPU de todos os pods em execução (Se nenhum limite for configurado, o número total de núcleos de CPU do nó será usado.)
Used CPU Cores (cores)	Uma métrica indica o número de núcleos de CPU usados
Physical Memory Usage (%)	Uma métrica indica o uso de memória física da carga de trabalho **Physical Memory Usage (%)** = uso de memória física/número total de núcleos de CPU de todos os pods em execução (Se nenhum limite for configurado, o número total de núcleos de CPU do nó será usado.)
Used Physical Memory (GiB)	Uma métrica indica a quantidade de memória física usada
Disk Read Rate	Uma métrica indica o volume de dados lido de um disco por segundo. A unidade é KB/s.
Disk Write Rate	Uma métrica indica o volume de dados gravado em um disco por segundo. A unidade é KB/s.
Downlink Rate (BPS) (KB/s)	Uma métrica indica a velocidade na qual os dados são baixados da Internet
Uplink Rate (BPS) (KB/s)	Uma métrica indica a velocidade na qual os dados são carregados do nó para a Internet

Métrica	Descrição
GPU Usage (%)	Uma métrica indica o uso da GPU da carga de trabalho
GPU Memory Usage (%)	Uma métrica indica a porcentagem da memória da GPU usada em relação à capacidade de memória da GPU **GPU Memory Usage (%)** = memória usada da GPU/capacidade da memória da GPU
Used GPU Memory (GiB)	Uma métrica indica a memória da GPU usada

----Fim

Visualizar dados de monitoramento de pods

O CCE permite que você visualize a data de monitoramento de seus pods.

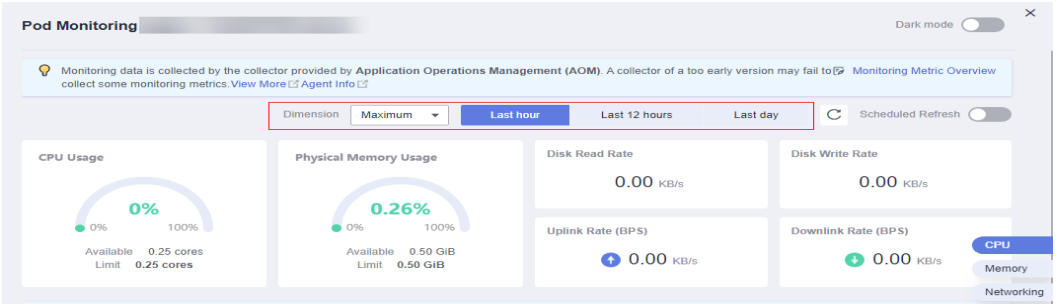
Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster.

Passo 2 Escolha **Workloads** no painel de navegação. Em seguida, clique no nome da carga de trabalho da carga de trabalho de destino para listar os pods.

Passo 3 Clique em **Monitor** do pod de destino para exibir os dados de monitoramento.

Passo 4 Você pode selecionar **Dimension** estatística e escolher o intervalo de tempo para exibir os dados de monitoramento. Os dados são fornecidos pelo AOM. Você pode visualizar os dados de monitoramento de um pod, incluindo CPU, memória, disco, rede e GPU.

Figura 9-6 Visualizar dados de monitoramento do pod



NOTA

Se existirem vários contêineres em um único pod, os dados de monitoramento podem variar de acordo com a **Dimension** estatística. Por exemplo, se você selecionar **Maximum** ou **Minimum** para **Dimension**, o valor de cada dado de monitoramento será o valor máximo ou mínimo de todos os contêineres sob o pod. Se **Average** estiver selecionada, o valor de cada dado de monitoramento será o valor médio de todos os contêineres no pod.

Tabela 9-9 Métricas de monitoramento do pod

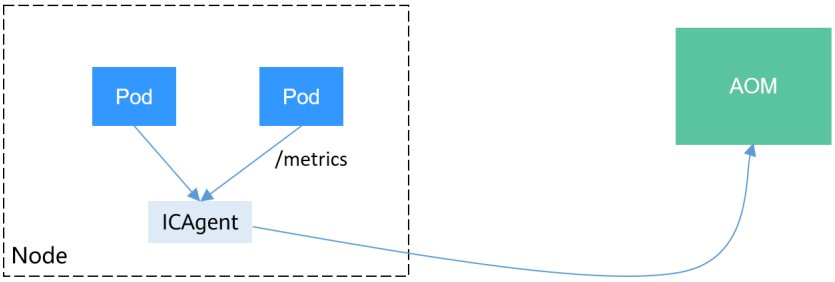
Métrica	Descrição
CPU Usage (%)	Uma métrica indica o uso da CPU do pod **CPU Usage (%)** = núcleos de CPU usados/número total de núcleos de CPU limitados de todos os contêineres em execução no pod (Se os núcleos de CPU limitados de todos os contêineres em execução não forem especificados, o número de núcleos de CPU do nó será usado.)
Used CPU Cores (cores)	Uma métrica indica o número de núcleos de CPU usados
Physical Memory Usage (%)	Uma métrica indica o uso de memória física do pod **Physical Memory Usage (%)** = memória física usada/soma dos limites de memória física de todos os contêineres em execução no pod (Se não especificado, o valor da memória física do nó é usado.)
Used Physical Memory (GiB)	Uma métrica indica a quantidade de memória física usada
Disk Read Rate	Uma métrica indica o volume de dados lido de um disco por segundo. A unidade é KB/s.
Disk Write Rate	Uma métrica indica o volume de dados gravado em um disco por segundo. A unidade é KB/s.
Downlink Rate (BPS) (KB/s)	Uma métrica indica a velocidade na qual os dados são baixados da Internet
Uplink Rate (BPS) (KB/s)	Uma métrica indica a velocidade na qual os dados são carregados do nó para a Internet
GPU Usage (%)	Uma métrica indica o uso da GPU do pod
GPU Memory Usage (%)	Uma métrica indica a porcentagem da memória da GPU usada em relação à capacidade de memória da GPU **GPU Memory Usage (%)** = memória usada da GPU/capacidade da memória da GPU
Used GPU Memory (GiB)	Uma métrica indica a memória da GPU usada do pod

----Fim

9.2.2 Monitoramento de métricas personalizadas no AOM

O CCE permite que você carregue métricas personalizadas no AOM. O ICAgent em um nó chama periodicamente a API de monitoramento de métrica configurada em uma carga de trabalho para ler os dados de monitoramento e, em seguida, carrega os dados para o AOM.

Figura 9-7 Usar o ICAgent para coletar métricas de monitoramento



A API de métrica personalizada de uma carga de trabalho pode ser configurada quando a carga de trabalho é criada. O procedimento a seguir usa uma aplicação Nginx como exemplo para descrever como relatar métricas personalizadas para o AOM.

1. **Preparar uma aplicação**

   Preparar uma imagem de aplicação. A aplicação deve fornecer uma API de monitoramento de métricas para o ICAgent coletar dados, e os dados de monitoramento devem **estar em conformidade com as especificações do prometheus**.

2. **Implementar aplicações e converter métricas de Nginx**

   Use a imagem de aplicação para implementar uma carga de trabalho em um cluster. Métricas de monitoramento personalizadas são relatadas automaticamente.

3. **Verificação**

   Vá para AOM para verificar se as métricas personalizadas foram coletadas com êxito.

Restrições

- O ICAgent é compatível com as especificações de dados de monitorização de **Prometheus**. As métricas personalizadas fornecidas pelos pods podem ser coletadas pelo ICAgent somente quando atendem às especificações de dados de monitoramento do Prometheus. Para mais detalhes, consulte **Coleta de dados de monitoramento de Prometheus**.
- O ICAgent suporta apenas métricas **Gauge**.
- O intervalo para o ICAgent chamar a API de métrica personalizada é de 1 minuto, o que não pode ser alterado.

Coleta de dados de monitoramento de Prometheus

Prometheus chama periodicamente a API de monitoramento de métricas (**/metrics** por padrão) de uma aplicação para obter dados de monitoramento. A aplicação precisa fornecer a API de monitoramento de métricas para a chamada do Prometheus, e os dados de monitoramento devem atender às seguintes especificações do Prometheus:

```
# TYPE nginx_connections_active gauge
nginx_connections_active 2
# TYPE nginx_connections_reading gauge
nginx_connections_reading 0
```

Prometheus fornece clientes em vários idiomas. Para obter detalhes sobre os clientes, consulte **Prometheus CLIENT LIBRARIES**. Para obter detalhes sobre como desenvolver um exportador, consulte **WRITING EXPORTERS**. A comunidade de Prometheus fornece vários exportadores de terceiros que podem ser usados diretamente. Para obter detalhes, consulte **EXPORTERS AND INTEGRATIONS**.

Preparar uma aplicação

A aplicação deve fornecer uma API de monitoramento de métricas para o ICAgent coletar dados, e os dados de monitoramento devem estar em conformidade com as especificações do Prometheus. Para mais detalhes, consulte **Coleta de dados de monitoramento de Prometheus**.

Este documento usa o Nginx como um exemplo para descrever como coletar dados de monitoramento. Existe um módulo chamado **ngx_http_stub_status_module** no Nginx, que fornece funções básicas de monitoramento. Você pode configurar o arquivo **nginx.conf** para fornecer uma interface para que sistemas externos acessem dados de monitoramento do Nginx.

Passo 1 Faça logon em uma VM Linux que possa acessar a Internet e executar comandos do Docker.

Passo 2 Crie um arquivo **nginx.conf**. Adicione a configuração do servidor em **http** para permitir que o Nginx forneça uma interface para os sistemas externos acessarem os dados de monitoramento.

```
user  nginx;
worker_processes  auto;

error_log  /var/log/nginx/error.log warn;
pid        /var/run/nginx.pid;

events {
    worker_connections  1024;
}

http {
    include       /etc/nginx/mime.types;
    default_type  application/octet-stream;
    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';

    access_log  /var/log/nginx/access.log  main;
    sendfile        on;
    #tcp_nopush     on;
    keepalive_timeout  65;
    #gzip  on;
    include /etc/nginx/conf.d/*.conf;

    server {
      listen 8080;
      server_name  localhost;
      location /stub_status {
         stub_status on;
         access_log off;
      }
    }
}
```

Passo 3 Use essa configuração para criar uma imagem e um arquivo Dockerfile.

```
vi Dockerfile
```

O conteúdo do Dockerfile é o seguinte:

```
FROM nginx:1.21.5-alpine
ADD nginx.conf /etc/nginx/nginx.conf
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

Passo 4 Use este Dockerfile para criar uma imagem e enviá-la para o SWR. O nome da imagem é **nginx:exporter**. Para detalhes sobre como fazer upload de uma imagem, consulte **Carregamento de uma imagem por meio de um cliente do Container Engine**.

1. No painel de navegação, escolha **My Images** e clique em **Upload Through Client** no canto superior direito. Na página exibida, clique em **Generate a temporary login command** e clique em  para copiar o comando.
2. Execute o comando de logon copiado na etapa anterior no nó. Se o logon for bem-sucedido, a mensagem "Login Succeeded" será exibida.
3. Execute o seguinte comando para criar uma imagem chamada nginx. A versão da imagem é exportadora.
   ```
   docker build -t nginx:exporter .
   ```
4. Marque a imagem e envie-a para o repositório de imagens. Altere o endereço do repositório de imagens e o nome da organização com base nos seus requisitos.
   ```
   docker tag nginx:exporter swr.ap-southeast-1.myhuaweicloud.com/dev-container/nginx:exporter
   docker push swr.ap-southeast-1.myhuaweicloud.com/dev-container/nginx:exporter
   ```

Passo 5 Veja as métricas da aplicação.

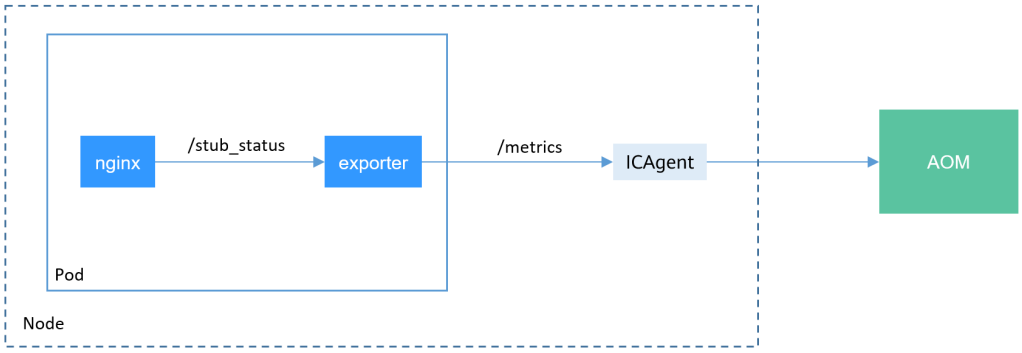
1. Use **nginx:exporter** para criar uma carga de trabalho.
2. **Acesse o contêiner** e use http://<ip_address>:8080/stub_status para obter dados de monitoramento do nginx. **<ip_address>** indica o endereço IP do contêiner. Informação semelhante à seguinte é exibida.
   ```
   # curl http://127.0.0.1:8080/stub_status
   Active connections: 3
   server accepts handled requests
    146269 146269 212
   Reading: 0 Writing: 1 Waiting: 2
   ```

----Fim

Implementar aplicações e converter métricas de Nginx

O formato dos dados de monitorização fornecidos pelo **nginx:exporter** não cumpre os requisitos do Prometheus. Converta o formato de dados para o formato exigido pelo Prometheus. Para converter o formato das métricas do Nginx, use **nginx-prometheus-exporter**, como mostrado na figura a seguir.

Figura 9-8 Usar exportador para converter o formato de dados



Implemente **nginx:exporter** e **nginx-prometheus-exporter** no mesmo pod.

```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: nginx-exporter
  namespace: default
spec:
```



```
  replicas: 1
  selector:
    matchLabels:
      app: nginx-exporter
  template:
    metadata:
      labels:
        app: nginx-exporter
      annotations:
        metrics.alpha.kubernetes.io/custom-endpoints:
'[{"api":"prometheus","path":"/metrics","port":"9113","names":""}]'
    spec:
      containers:
        - name: container-0
          image: 'nginx:exporter'  # Replace it with the address of the image you
uploaded to SWR.
          resources:
            limits:
              cpu: 250m
              memory: 512Mi
            requests:
              cpu: 250m
              memory: 512Mi
        - name: container-1
          image: 'nginx/nginx-prometheus-exporter:0.9.0'
          command:
            - nginx-prometheus-exporter
          args:
            - '-nginx.scrape-uri=http://127.0.0.1:8080/stub_status'
      imagePullSecrets:
        - name: default-secret
```

NOTA

A imagem **nginx/nginx-prometheus-exporter:0.9.0** precisa ser extraída da rede pública. Portanto, um endereço IP público precisa ser vinculado a cada nó no cluster.

nginx-prometheus-exporter requer um comando de inicialização. **nginx-prometheus-exporter -nginx.scrape-uri=http://127.0.0.1:8080/stub_status** é usado para obter dados de monitoramento do Nginx.

Além disso, adicione uma anotação **metrics.alpha.kubernetes.io/custom-endpoints: '[{"api":"prometheus","path":"/metrics","port":"9113","names":""}]'** ao pod.

Verificação

Depois que uma aplicação é implementada, você pode acessar o Nginx para criar alguns dados de acesso e verificar se os dados de monitoramento correspondentes podem ser obtidos no AOM.

Passo 1 Obtenha o nome do pod do Nginx.

```
$ kubectl get pod
NAME                               READY   STATUS    RESTARTS   AGE
nginx-exporter-78859765db-6j8sw   2/2     Running   0          4m
```

Passo 2 Faça logon no contêiner e execute comandos para acessar o Nginx.

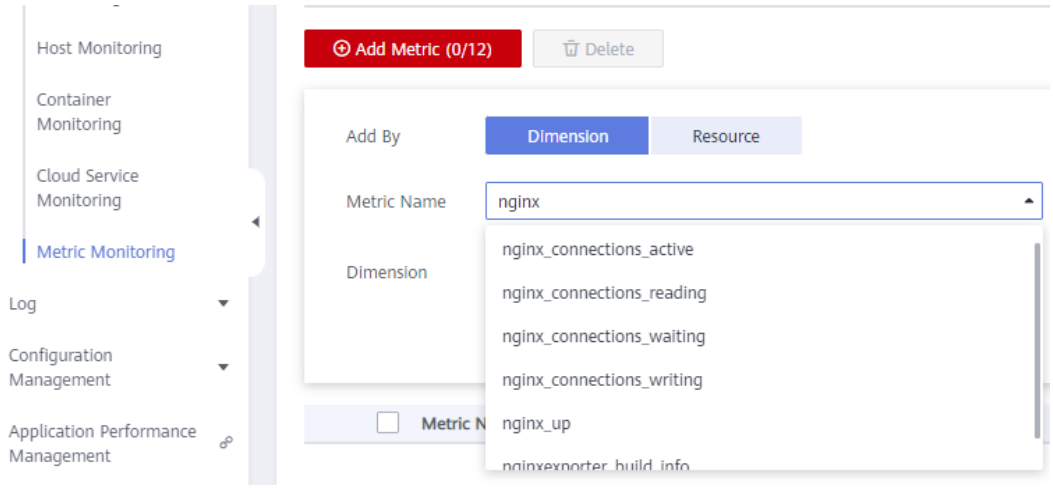
```
$ kubectl exec -it nginx-exporter-78859765db-6j8sw -- /bin/sh
Defaulting container name to container-0.
Use 'kubectl describe pod/nginx-exporter-78859765db-6j8sw -n default' to see all
of the containers in this pod.
/ # curl http://localhost
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
```



```
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
/ #
```

Passo 3 Faça logon no AOM. No painel de navegação, escolha **Monitoring** > **Metric Monitoring** para exibir métricas relacionadas ao Nginx, por exemplo, **nginx_connections_active**.

Figura 9-9 Visualização de métricas de monitoramento



----Fim

9.2.3 Monitoramento de métricas personalizadas usando o Prometheus

Você pode usar o ICAgent do AOM para obter dados de métrica personalizados de cargas de trabalho, conforme descrito em **Monitoramento de métricas personalizadas no AOM**. Você também pode instalar o complemento prometheus em um cluster e usar o Prometheus como plataforma de monitoramento.

O procedimento a seguir usa uma aplicação Nginx como exemplo para descrever como usar o Prometheus para monitorar métricas personalizadas:

1. **Instalar o complemento**

   CCE fornece um complemento que integra funções prometheus. Você pode instalá-lo com vários cliques.

2. **Acessar o Prometheus**

(Opcional) Vincule um Serviço LoadBalancer ao Prometheus para que o Prometheus possa ser acessado a partir de redes externas.

3. **Preparar uma aplicação**

   Preparar uma imagem de aplicação. A aplicação deve fornecer uma API de monitoramento de métricas para o ICAgent coletar dados, e os dados de monitoramento devem **estar em conformidade com as especificações do Prometheus**.

4. **Monitorar métricas personalizadas**

   Use a imagem da aplicação para implementar uma carga de trabalho em um cluster. As métricas de monitorização personalizadas são automaticamente comunicadas ao Prometheus.

5. **Configurar regras de coleta para métricas personalizadas**

   Depois que as regras de coleta são configuradas, as métricas personalizadas são relatadas para o servidor de métricas, que pode ser usado em cenários como o dimensionamento automático da carga de trabalho.

6. **Acessar ao Grafana**

   Veja os dados de monitoramento do Prometheus no Grafana, um painel de visualização.

Restrições

Para usar o prometheus para monitorar métricas personalizadas, o aplicativo precisa fornecer uma API de monitoramento de métricas. Para mais detalhes, consulte **Coleta de dados de monitoramento de Prometheus**.

Coleta de dados de monitoramento de Prometheus

Prometheus chama periodicamente a API de monitoramento de métricas (/**metrics** por padrão) de uma aplicação para obter dados de monitoramento. A aplicação precisa fornecer a API de monitoramento de métricas para a chamada do Prometheus, e os dados de monitoramento devem atender às seguintes especificações do Prometheus:

```
# TYPE nginx_connections_active gauge
nginx_connections_active 2
# TYPE nginx_connections_reading gauge
nginx_connections_reading 0
```

Prometheus fornece clientes em vários idiomas. Para obter detalhes sobre os clientes, consulte **Prometheus CLIENT LIBRARIES**. Para obter detalhes sobre como desenvolver um exportador, consulte **WRITING EXPORTERS**. A comunidade de Prometheus fornece vários exportadores de terceiros que podem ser usados diretamente. Para obter detalhes, consulte **EXPORTERS AND INTEGRATIONS**.

Instalar o complemento

Instale o complemento com base na versão do cluster e nos requisitos reais.

- **Monitoramento de cluster da nuvem nativa**: suporta clusters de v1.17 ou posterior. Além dos recursos de monitoramento, esse complemento fornece interconexão entre os dados de monitoramento e o Container Intelligent Analysis (CIA).
- **Prometheus (EOM)**: suporta apenas clusters da v1.21 ou anterior.

Acessar o Prometheus

Depois que o complemento for instalado, você poderá implementar cargas de trabalho e Serviços. O StatefulSet chamado **prometheus** refere-se ao Servidor Prometheus.

Pode criar um **Serviço LoadBalancer** de rede pública para que o Prometheus possa ser acedido a partir de uma rede externa.

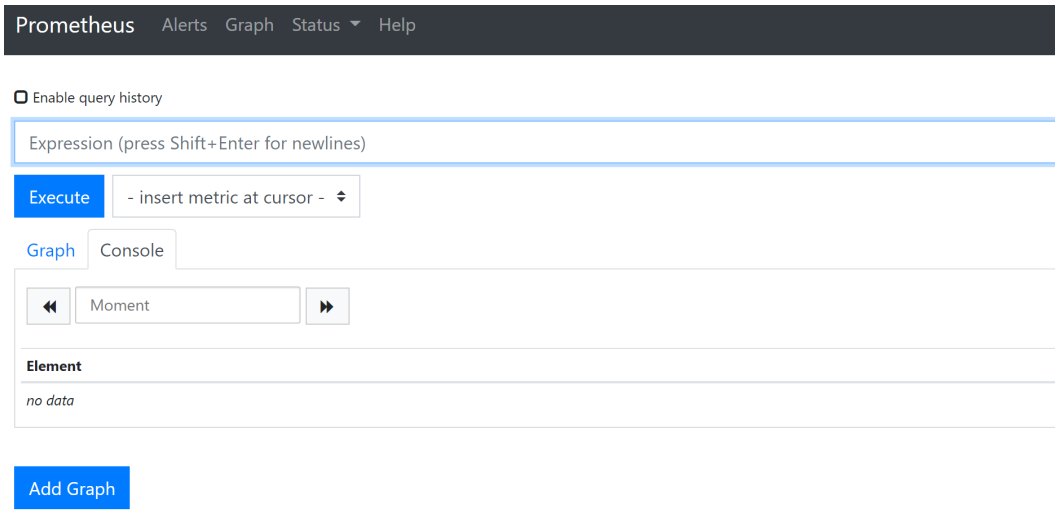
Passo 1 Faça logon no console do CCE e clique no nome do cluster com o complemento prometheus instalado para acessar o console do cluster. Na página exibida, escolha **Networking** no painel de navegação.

Passo 2 Clique em **Create from YAML** no canto superior direito para criar um Serviço LoadBalancer de rede pública.

```
apiVersion: v1
kind: Service
metadata:
  name: prom-lb     #Service name, which can be customized.
  namespace: monitoring
  labels:
    app: prometheus
    component: server
  annotations:
    kubernetes.io/elb.id: 038ff***     #Replace it with the ID of the public
networkload balancer in the VPC to which the cluster belongs.
spec:
  ports:
    - name: cce-service-0
      protocol: TCP
      port: 88     #Service port, which can be customized.
      targetPort: 9090     #Default port of Prometheus. Retain the default value.
  selector:
    app: prometheus
    component: server
    release: cceaddon-prometheus
  type: LoadBalancer
```

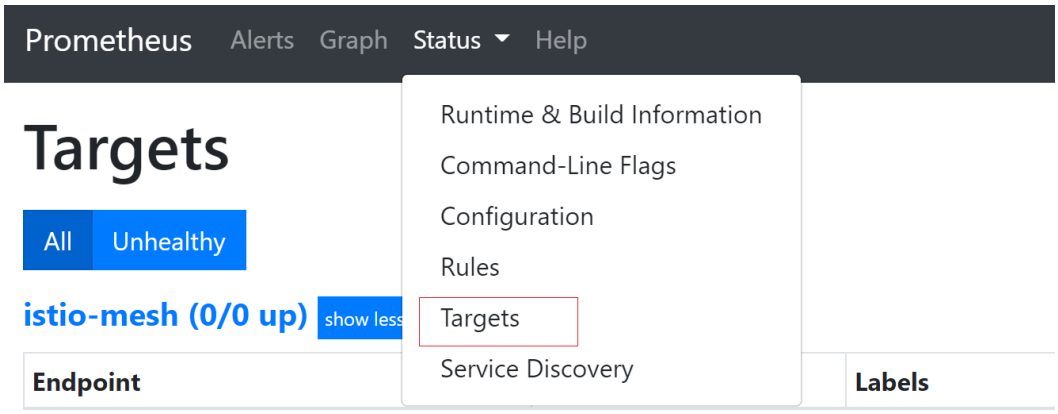
Passo 3 Após a criação, visite ***load balancer public IP:Service port*** para acessar o Prometheus.

Figura 9-10 Acessar o Prometheus



Passo 4 Escolha **Status** > **Targets** para exibir os alvos monitorados pelo prometheus.

Figura 9-11 Exibir alvos monitorados



----Fim

Preparar uma aplicação

A aplicação deve fornecer uma API de monitoramento de métricas para o ICAgent coletar dados, e os dados de monitoramento devem estar em conformidade com as especificações do Prometheus. Para mais detalhes, consulte **Coleta de dados de monitoramento de Prometheus**.

Este documento usa o Nginx como um exemplo para descrever como coletar dados de monitoramento. Existe um módulo chamado **ngx_http_stub_status_module** no Nginx, que fornece funções básicas de monitoramento. Você pode configurar o arquivo **nginx.conf** para fornecer uma interface para que sistemas externos acessem dados de monitoramento do Nginx.

Passo 1 Faça logon em uma VM Linux que possa acessar a Internet e executar comandos do Docker.

Passo 2 Crie um arquivo **nginx.conf**. Adicione a configuração do servidor em **http** para permitir que o Nginx forneça uma interface para os sistemas externos acessarem os dados de monitoramento.

```
user  nginx;
worker_processes  auto;

error_log  /var/log/nginx/error.log warn;
pid        /var/run/nginx.pid;

events {
    worker_connections  1024;
}

http {
    include       /etc/nginx/mime.types;
    default_type  application/octet-stream;
    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';

    access_log  /var/log/nginx/access.log  main;
    sendfile        on;
    #tcp_nopush     on;
    keepalive_timeout  65;
    #gzip  on;
    include /etc/nginx/conf.d/*.conf;

    server {
      listen 8080;
```



```
      server_name  localhost;
      location /stub_status {
         stub_status on;
         access_log off;
      }
    }
}
```

Passo 3 Use essa configuração para criar uma imagem e um arquivo Dockerfile.

```
vi Dockerfile
```

O conteúdo do Dockerfile é o seguinte:

```
FROM nginx:1.21.5-alpine
ADD nginx.conf /etc/nginx/nginx.conf
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

Passo 4 Use este Dockerfile para criar uma imagem e enviá-la para o SWR. O nome da imagem é **nginx:exporter**. Para detalhes sobre como fazer upload de uma imagem, consulte **Carregamento de uma imagem por meio de um cliente do Container Engine**.

1. No painel de navegação, escolha **My Images** e clique em **Upload Through Client** no canto superior direito. Na página exibida, clique em **Generate a temporary login command** e clique em  para copiar o comando.
2. Execute o comando de logon copiado na etapa anterior no nó. Se o logon for bem-sucedido, a mensagem "Login Succeeded" será exibida.
3. Execute o seguinte comando para criar uma imagem chamada nginx. A versão da imagem é exportadora.
   ```
   docker build -t nginx:exporter .
   ```
4. Marque a imagem e envie-a para o repositório de imagens. Altere o endereço do repositório de imagens e o nome da organização com base nos seus requisitos.
   ```
   docker tag nginx:exporter swr.ap-southeast-1.myhuaweicloud.com/dev-container/
   nginx:exporter
   docker push swr.ap-southeast-1.myhuaweicloud.com/dev-container/nginx:exporter
   ```

Passo 5 Veja as métricas da aplicação.

1. Use **nginx:exporter** para criar uma carga de trabalho.
2. **Acesse o contêiner** e use http://<ip_address>:8080/stub_status para obter dados de monitoramento do nginx. **<ip_address>** indica o endereço IP do contêiner. Informação semelhante à seguinte é exibida.
   ```
   # curl http://127.0.0.1:8080/stub_status
   Active connections: 3
   server accepts handled requests
    146269 146269 212
   Reading: 0 Writing: 1 Waiting: 2
   ```

----Fim

Monitorar métricas personalizadas

O formato dos dados de monitorização fornecidos pelo **nginx:exporter** não cumpre os requisitos do Prometheus. Converta o formato de dados para o formato exigido pelo Prometheus. Para converter o formato das métricas do Nginx, use **nginx-prometheus-exporter**. Implemente **nginx:exporter** e **nginx-prometheus-exporter** no mesmo pod e adicione as seguintes anotações durante a implementação. Em seguida, o Prometheus pode coletar métricas automaticamente.

```
kind: Deployment
apiVersion: apps/v1
```



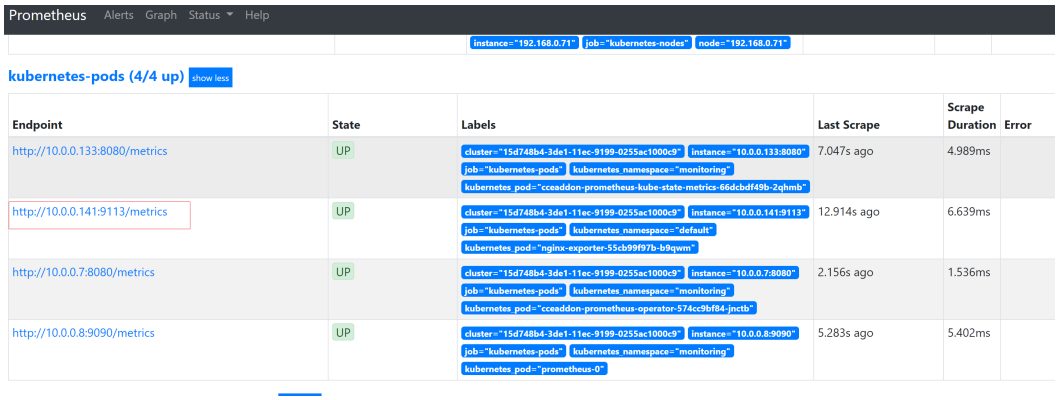
```
metadata:
  name: nginx-exporter
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx-exporter
  template:
    metadata:
      labels:
        app: nginx-exporter
      annotations:
        prometheus.io/scrape: "true"
        prometheus.io/port: "9113"
        prometheus.io/path: "/metrics"
        prometheus.io/scheme: "http"
    spec:
      containers:
        - name: container-0
          image: 'nginx:exporter'      # Replace it with the address of the image
you uploaded to SWR.
          resources:
            limits:
              cpu: 250m
              memory: 512Mi
            requests:
              cpu: 250m
              memory: 512Mi
        - name: container-1
          image: 'nginx/nginx-prometheus-exporter:0.9.0'
          command:
            - nginx-prometheus-exporter
          args:
            - '-nginx.scrape-uri=http://127.0.0.1:8080/stub_status'
      imagePullSecrets:
        - name: default-secret
```

Na descrição anterior:

- **prometheus.io/scrape** indica se deve ativar o Prometheus para coletar dados de monitoramento do pod. O valor é **true**.
- **prometheus.io/port** indica a porta para coletar dados de monitoramento.
- **prometheus.io/path** indica o URL da API para coletar dados de monitoramento. Se este parâmetro não for definido, o valor padrão **/metrics** é usado.
- **prometheus.io/scheme**: protocolo usado para coleta de dados. O valor pode ser **http** ou **https**.

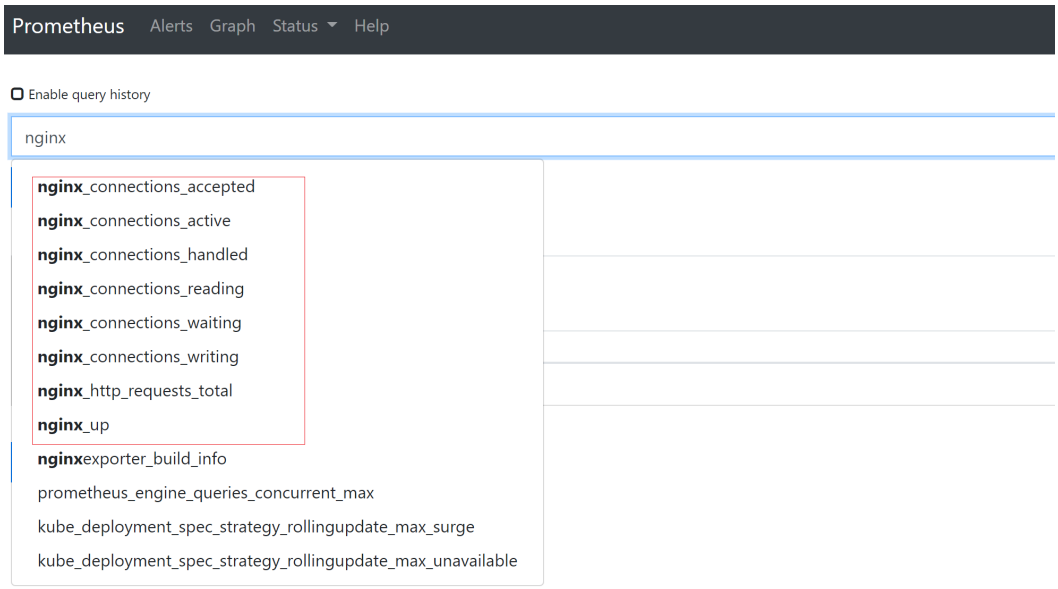
Depois que a aplicação é implementada, um pod com um caminho de coleção da porta 9113 pode ser encontrado em **Status** > **Targets**.

Figura 9-12 Visualizar o caminho da coleção do pod



Na guia **Graph**, insira **nginx**. As métricas relacionadas são exibidas.

Figura 9-13 Métricas relacionadas ao Nginx



Configurar regras de coleta para métricas personalizadas

O complemento kube-prometheus-stack da nova versão não fornece métricas personalizadas. Ou seja, as regras de coleta de métricas não são mais configuradas na ConfigMap **user-adapter-config** (**adapter-config** em versões anteriores). Adicione regras de coleta de métricas. Para obter detalhes, consulte **Detecção de métricas e configuração de apresentação**. Se você atualizou o complemento, as configurações originais são herdadas e usadas.

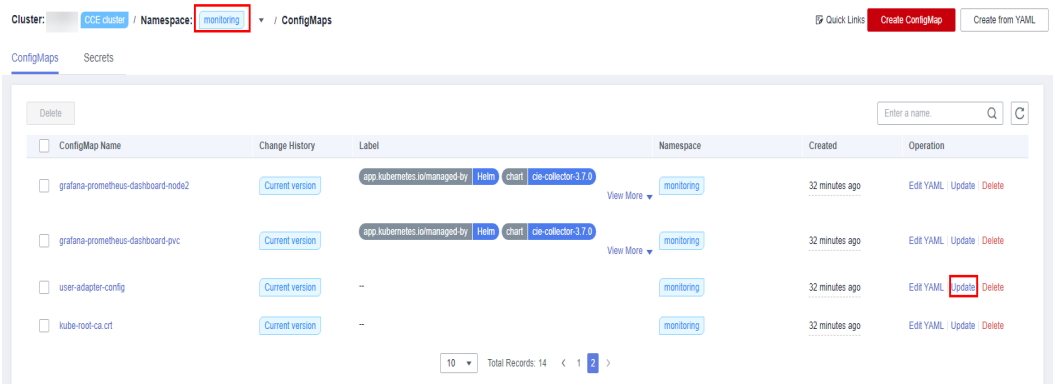
AVISO

Para usar o prometheus para monitorar métricas personalizadas, o aplicativo precisa fornecer uma API de monitoramento de métricas. Para mais detalhes, consulte **Coleta de dados de monitoramento de Prometheus**.

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster. No painel de navegação, escolha **ConfigMaps and Secrets**.

Passo 2 Alterne para o namespace de **monitoring**, localize a ConfigMap **user-adapter-config** (**adapter-config** em versões anteriores) na guia **ConfigMaps** e clique em **Update**.

Figura 9-14 Atualizar um ConfigMap



Passo 3 Na janela que desliza para fora da direita, clique em **Edit** na coluna de operação de **Data** para o arquivo **config.yaml**. Em seguida, adicione uma regra de coleta de métricas personalizadas no campo **rules**. Clique em **OK**.

Você pode adicionar várias regras de coleta adicionando várias configurações no campo **rules**. Para obter detalhes, consulte **Detecção de métricas e configuração de apresentação**.

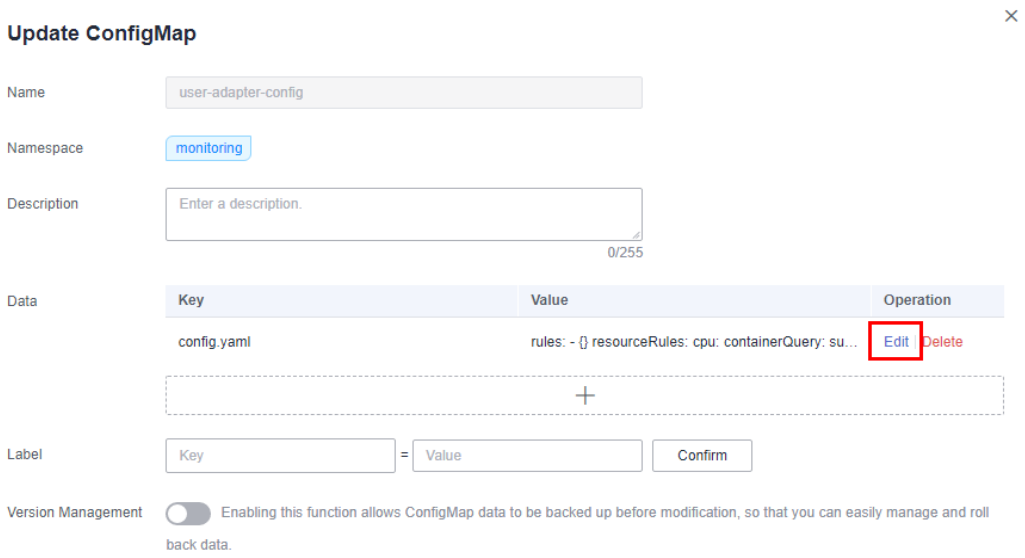
Este é um exemplo de personalização de uma regra de coleta para o nginx:export:

```
rules:
- seriesQuery: '{__name__=~"^nginx_.*",container!="POD",namespace!="",pod!=""}'
  resources:
    overrides:
      namespace:
        resource: namespace
      pod:
        resource: pod
  name:
    matches: (.*)
  metricsQuery: 'sum(<<.Series>>{<<.LabelMatchers>>,container!="POD"}) by
(<<.GroupBy>>)'
```

NOTA

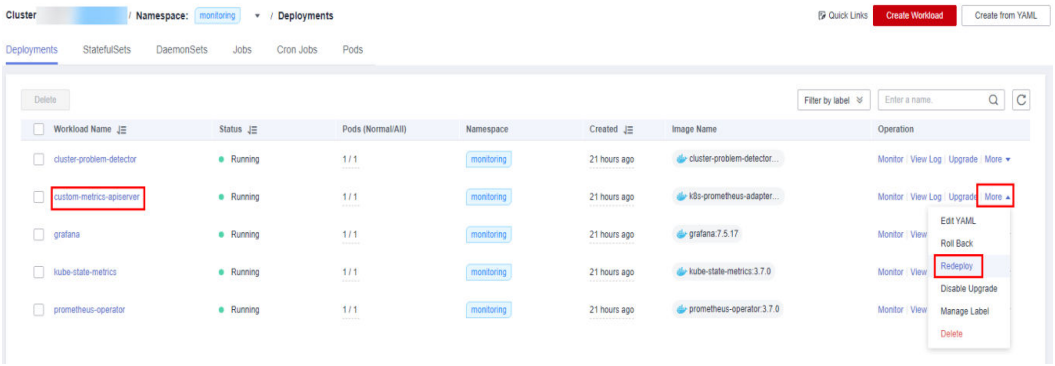
O exemplo anterior aplica-se somente à aplicação nginx:export neste exemplo. Se você precisar coletar métricas personalizadas, adicione ou altere regras de acordo com o **guia oficial**.

Figura 9-15 Editar dados do ConfigMap



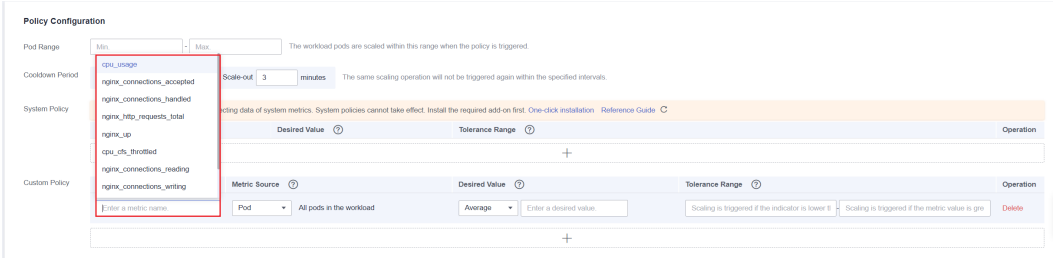
Passo 4 Reimplante o **custom-metrics-apiserver** no namespace de **monitoring**.

Figura 9-16 Reimplantação de custom-metrics-apiserver



Passo 5 Depois que **custom-metrics-apiserver** é executado com êxito, você pode selecionar as métricas personalizadas relatadas pela aplicação nginx:export ao criar uma política HPA. Para mais detalhes, consulte **Criação de uma política HPA para dimensionamento automático da carga de trabalho**.

Figura 9-17 Criar uma política HPA usando métricas personalizadas



----Fim

Acessar ao Grafana

O complemento prometheus tem **Grafana** (uma ferramenta de visualização de código aberto) instalado e interconectado com o Prometheus. Você pode criar um **Serviço LoadBalancer** de rede pública para acessar o Grafana da rede pública e visualizar os dados de monitoramento do Prometheus no Grafana.

Clique no endereço de acesso para acessar o Grafana e selecione um painel adequado para visualizar o conteúdo agregado.

Passo 1 Faça logon no console do CCE e clique no nome do cluster com o complemento prometheus instalado para acessar o console do cluster. Na página exibida, escolha **Networking** no painel de navegação.

Passo 2 Clique em **Create from YAML** no canto superior direito para criar um Serviço LoadBalancer de rede pública para o Grafana.

```
apiVersion: v1
kind: Service
metadata:
  name: grafana-lb     #Service name, which can be customized.
  namespace: monitoring
  labels:
    app: grafana
  annotations:
    kubernetes.io/elb.id: 038ff***     #Replace it with the ID of the public
networkload balancer in the VPC to which the cluster belongs.
spec:
  ports:
    - name: cce-service-0
      protocol: TCP
      port: 80     #Service port, which can be customized.
      targetPort: 3000     #Default port of Grafana. Retain the default value.
  selector:
    app: grafana
  type: LoadBalancer
```

Passo 3 Após a criação, visite ***load balancer public IP*:*Service port*** para acessar o Grafana e selecione um painel adequado para visualizar os dados agregados.

Figura 9-18 Painel de Grafana

----Fim

Apêndice: persistência dos dados de Grafana

Se os dados do Grafana não forem persistentes, os dados podem ser perdidos quando o contêiner do Grafana for reiniciado. Você pode montar o armazenamento em nuvem no contêiner do Grafana para obter persistência de dados do Grafana.

NOTA

Por padrão, o complemento **kube-prometheus-stack** cria um volume de armazenamento de 5 GiB para o Grafana. Desinstalar o complemento não excluirá esse volume. Você não precisa montar manualmente o armazenamento em nuvem.

Passo 1 Use o kubectl para se conectar ao cluster onde o cluster Grafana reside. Para mais detalhes, consulte **Conexão a um cluster usando o kubectl**.

Passo 2 Crie a PVC de um disco EVS.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: grafana-pvc
  namespace: monitoring
  annotations:
    everest.io/disk-volume-type: SSD
  labels:
    failure-domain.beta.kubernetes.io/region: ap-southeast-3
    failure-domain.beta.kubernetes.io/zone: ap-southeast-3a
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
  storageClassName: csi-disk
```

O disco EVS e o nó onde o Grafana reside devem estar na mesma AZ. Caso contrário, o disco EVS não pode ser conectado.

- **failure-domain.beta.kubernetes.io/region**: região onde reside o disco EVS.
- **failure-domain.beta.kubernetes.io/zone**: a AZ onde o disco EVS reside.
- **storage**: tamanho do disco EVS. Defina este parâmetro conforme necessário.

Você também pode criar discos EVS no console do CCE. Para mais detalhes, consulte **(Console) Criar automaticamente um disco EVS**.

Passo 3 Modifique a configuração da carga de trabalho do Grafana e monte o disco EVS.

kubectl edit deploy grafana -n monitoring

Adicione o disco EVS ao contêiner no arquivo YAML, conforme mostrado na figura a seguir. O nome da PVC deve ser o mesmo em **Passo 2**, e o caminho de montagem deve ser **/var/lib/grafana**.

Além disso, a política de atualização deve ser modificada para a carga de trabalho do Grafana. O número máximo de pods é 1.

```
...
  template:
    spec:
      volumes:
        - name: cce-pvc-grafana
          persistentVolumeClaim:
            claimName: grafana-pvc
...
```



```
      containers:
        - volumeMounts:
            - name: cce-pvc-grafana
              mountPath: /var/lib/grafana
...
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxUnavailable: 1
      maxSurge: 1
```

Salve a configuração. A carga de trabalho do Grafana será atualizada e o disco EVS será montado.

----Fim

9.2.4 Monitoramento de métricas dos componentes do nó principais

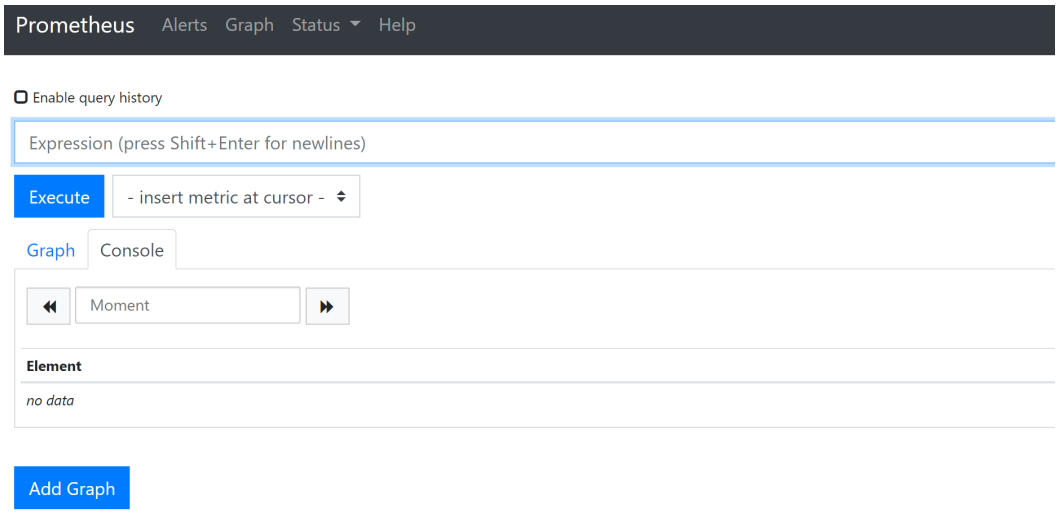
Exibir métricas dos componentes do nó principal

O complemento kube-prometheus-stack (**Monitoramento de cluster da nuvem nativa**) da versão 3.5.0 ou posterior pode monitorar e coletar métricas de kube-apiserver, kube-controller, kube-schedule e etcd-server nos nós principais.

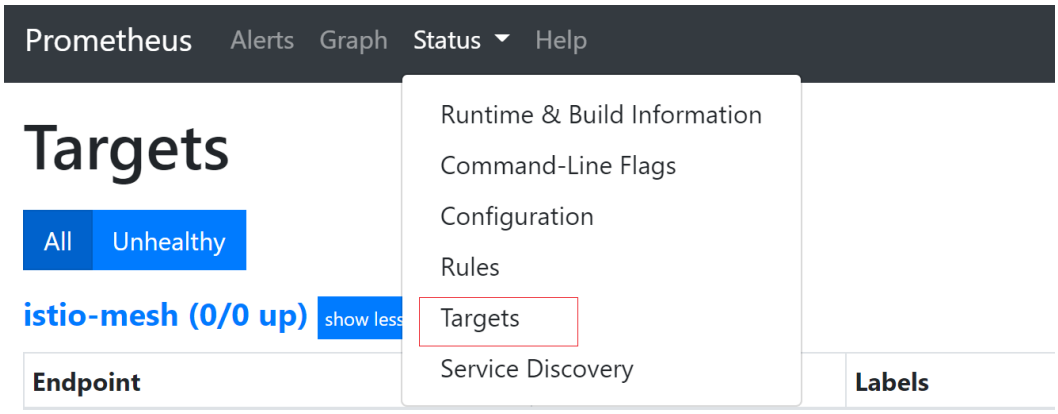
Passo 1 Instale o kube-prometheus-stack da versão 3.5.0 ou posterior em seu cluster. Para mais detalhes, consulte **Monitoramento de cluster da nuvem nativa**.

Passo 2 Após a instalação, aguarde até que todas as instâncias complementares estejam em execução.

Passo 3 Crie um serviço de rede pública do Serviço LoadBalancer para o Prometheus. Digite **load balancer public IP: Service port** na caixa de endereço do navegador para acessar o Prometheus. Para obter detalhes, consulte **Acesso do Prometheus**.



Passo 4 Escolha **Status > Targets**. Os componentes de nó principal anteriores são exibidos.



----Fim

Coletar métricas dos componentes do nó principal usando Prometheus autoconstruído

Esta seção descreve como coletar as métricas dos componentes do nó principal usando o prometheus autoconstruído.

AVISO

- A versão do cluster deve ser 1.19 ou posterior.
- O prometheus-operator deve ser instalado no cluster. Para obter detalhes, consulte **Operador de Prometheus**.

Passo 1 Prometheus autoconstruído deve ser instalado no cluster. Para obter detalhes, consulte **Gráficos do Helm da comunidade de Prometheus**. O complemento de prometheus (**Prometheus (EOM)**) está em fim de manutenção e não suporta esta função. Portanto, evite usar esse complemento.

Passo 2 Use **kubectl** para se conectar ao cluster.

Passo 3 Modifique o ClusterRole de Prometheus.

```
kubectl edit ClusterRole prometheus -n {namespace}
```

Adicione o seguinte conteúdo sob o campo **rules**:

```
rules:
...
- apiGroups:
  - proxy.exporter.k8s.io
  resources:
  - "*"
  verbs: ["get", "list", "watch"]
```

Passo 4 Crie um arquivo chamado **kube-apiserver.yaml** e edite-o.

```
vi kube-apiserver.yaml
```

Exemplo de conteúdo do arquivo:

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  labels:
    app.kubernetes.io/name: apiserver
```



```
  name: kube-apiserver
  namespace: monitoring    # Change it to the namespace where Prometheus will be
installed.
spec:
  endpoints:
  - bearerTokenFile: /var/run/secrets/kubernetes.io/serviceaccount/token
    interval: 30s
    metricRelabelings:
    - action: keep
      regex: (aggregator_unavailable_apiservice|
apiserver_admission_controller_admission_duration_seconds_bucket|
apiserver_admission_webhook_admission_duration_seconds_bucket|
apiserver_admission_webhook_admission_duration_seconds_count|
apiserver_client_certificate_expiration_seconds_bucket|
apiserver_client_certificate_expiration_seconds_count|
apiserver_current_inflight_requests|apiserver_request_duration_seconds_bucket|
apiserver_request_total|go_goroutines|kubernetes_build_info|
process_cpu_seconds_total|process_resident_memory_bytes|
rest_client_requests_total|workqueue_adds_total|workqueue_depth|
workqueue_queue_duration_seconds_bucket|aggregator_unavailable_apiservice_total|
rest_client_request_duration_seconds_bucket)
      sourceLabels:
      - __name__
    - action: drop
      regex: apiserver_request_duration_seconds_bucket;(0.15|0.25|0.3|0.35|0.4|
0.45|0.6|0.7|0.8|0.9|1.25|1.5|1.75|2.5|3|3.5|4.5|6|7|8|9|15|25|30|50)
      sourceLabels:
      - __name__
      - le
    port: https
    scheme: https
    tlsConfig:
      caFile: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
      serverName: kubernetes
  jobLabel: component
  namespaceSelector:
    matchNames:
    - default
  selector:
    matchLabels:
      component: apiserver
      provider: kubernetes
```

Crie um ServiceMonitor:

```
kubectl apply -f kube-apiserver.yaml
```

Passo 5 Crie um arquivo chamado **kube-controller.yaml** e edite-o.

```
vi kube-controller.yaml
```

Exemplo de conteúdo do arquivo:

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  labels:
    app.kubernetes.io/name: kube-controller
  name: kube-controller-manager
  namespace: monitoring    # Change it to the namespace where Prometheus will be
installed.
spec:
  endpoints:
    - bearerTokenFile: /var/run/secrets/kubernetes.io/serviceaccount/token
      interval: 15s
      honorLabels: true
      port: https
      relabelings:
        - regex: (.+)
          replacement: /apis/proxy.exporter.k8s.io/v1beta1/kube-controller-proxy/
${1}/metrics
```



```
          sourceLabels:
            - __address__
          targetLabel: __metrics_path__
        - regex: (.+)
          replacement: ${1}
          sourceLabels:
            - __address__
          targetLabel: instance
        - replacement: kubernetes.default.svc.cluster.local:443
          targetLabel: __address__
      scheme: https
      tlsConfig:
        caFile: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
  jobLabel: app
  namespaceSelector:
    matchNames:
      - kube-system
  selector:
    matchLabels:
      app: kube-controller-proxy
      version: v1
```

Crie um ServiceMonitor:

```
kubectl apply -f kube-controller.yaml
```

Passo 6 Crie um arquivo chamado **kube-scheduler.yaml** e edite-o.

```
vi kube-scheduler.yaml
```

Exemplo de conteúdo do arquivo:

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  labels:
    app.kubernetes.io/name: kube-scheduler
  name: kube-scheduler
  namespace: monitoring    # Change it to the namespace where Prometheus will be
installed.
spec:
  endpoints:
    - bearerTokenFile: /var/run/secrets/kubernetes.io/serviceaccount/token
      interval: 15s
      honorLabels: true
      port: https
      relabelings:
        - regex: (.+)
          replacement: /apis/proxy.exporter.k8s.io/v1beta1/kube-scheduler-proxy/
${1}/metrics
          sourceLabels:
            - __address__
          targetLabel: __metrics_path__
        - regex: (.+)
          replacement: ${1}
          sourceLabels:
            - __address__
          targetLabel: instance
        - replacement: kubernetes.default.svc.cluster.local:443
          targetLabel: __address__
      scheme: https
      tlsConfig:
        caFile: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
  jobLabel: app
  namespaceSelector:
    matchNames:
      - kube-system
  selector:
    matchLabels:
      app: kube-scheduler-proxy
      version: v1
```


Crie um ServiceMonitor:

```
kubectl apply -f kube-scheduler.yaml
```

Passo 7 Crie um arquivo chamado **etcd-server.yaml** e edite-o.

```
vi etcd-server.yaml
```

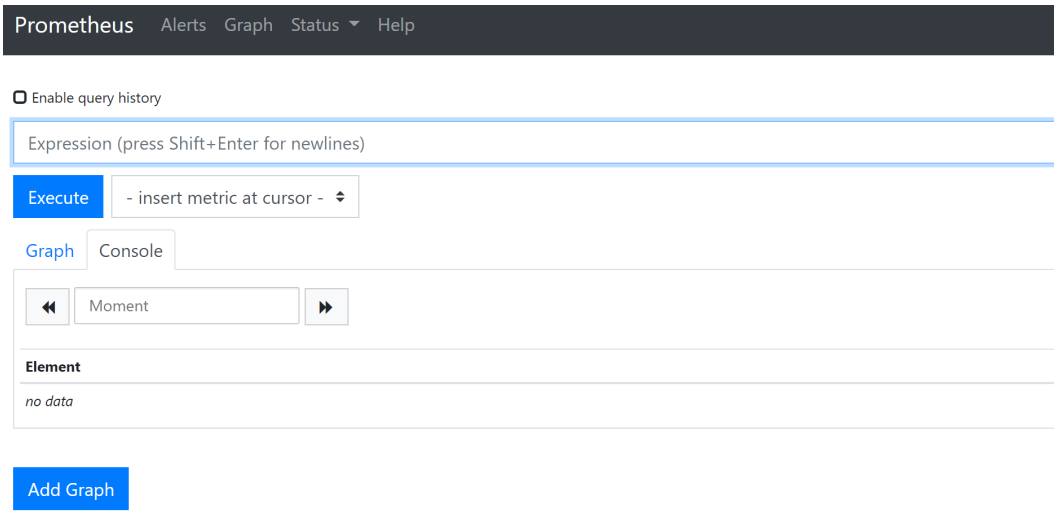
Exemplo de conteúdo do arquivo:

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  labels:
    app.kubernetes.io/name: etcd-server
  name: etcd-server
  namespace: monitoring    # Change it to the namespace where Prometheus will be
installed.
spec:
  endpoints:
    - bearerTokenFile: /var/run/secrets/kubernetes.io/serviceaccount/token
      interval: 15s
      honorLabels: true
      port: https
      relabelings:
        - regex: (.+)
          replacement: /apis/proxy.exporter.k8s.io/v1beta1/etcd-server-proxy/${1}/
metrics
          sourceLabels:
            - __address__
          targetLabel: __metrics_path__
        - regex: (.+)
          replacement: ${1}
          sourceLabels:
            - __address__
          targetLabel: instance
        - replacement: kubernetes.default.svc.cluster.local:443
          targetLabel: __address__
      scheme: https
      tlsConfig:
        caFile: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
  jobLabel: app
  namespaceSelector:
    matchNames:
      - kube-system
  selector:
    matchLabels:
      app: etcd-server-proxy
      version: v1
```

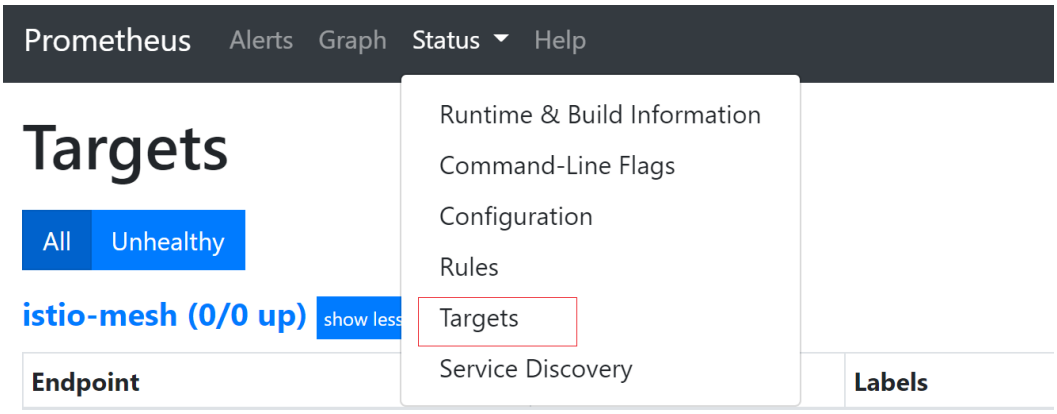
Crie um ServiceMonitor:

```
etcd-server.yaml
```

Passo 8 Após a criação, se você tiver criado um **Serviço LoadBalancer** de rede pública para Prometheus, poderá acessar **load balancer public IP: Service port** para acessar o Prometheus.



Passo 9 Escolha **Status > Targets**. Os componentes de nó principal anteriores são exibidos.



----Fim

9.3 Centro de monitoramento

9.3.1 Visão geral

O Centro de monitoramento é uma plataforma de O&M de última geração para contêineres da nuvem nativa. Ele monitora aplicações e recursos em tempo real, coleta métricas e eventos para analisar o status de integridade da aplicação e visualiza dados multidimensionais. Compatível com os principais componentes de código aberto, o Centro de monitoramento suporta a localização rápida de falhas.

Funções

- **Insight de contêineres**: monitora de forma abrangente os contêineres do Kubernetes nativo e exibe métricas de clusters, nós, cargas de trabalho, pods e eventos para o status de integridade e carga de clusters.
- **Diagnóstico de integridade**: verifica periodicamente os status de integridade dos clusters, incluindo o uso de recursos e os status de execução dos nós principais, nós de trabalho, clusters, cargas de trabalho e dependências externas.

- **Dashboard**: mostra vários gráficos, como gráficos de linha e gráficos de dígitos na mesma tela para exibição abrangente de dados.

Vantagens

- O Centro de monitoramento está profundamente integrado ao Prometheus, um projeto de monitoramento maduro da CNCF, e está em conformidade com as especificações OpenTracing e OpenTelemetry. Ele traz observabilidade para suas aplicações da nuvem nativa, coletando, armazenando e apresentando visualmente dados de O&M, como métricas e eventos importantes.
- CIA fornece monitoramento de pilha completa de recursos de infraestrutura da nuvem nativa para cargas de trabalho de aplicações, permitindo que você perceba claramente o status de carga de infraestrutura e aplicações a qualquer hora e em qualquer lugar.
- CIA monitora clusters, nós e pods do Kubernetes, permite rastreamento e visualização de ponta a ponta para serviços e fornece diagnóstico de integridade do cluster, acelerando muito a análise e a localização de falhas.
- A CIA fornece complementos prontos para uso, coleta de dados e monitoramento baseado em painel. Em comparação com os produtos de monitoramento desenvolvidos com base em tecnologias de código aberto, é mais competitivo em confiabilidade, disponibilidade e implementação.

Restrições

- A versão do cluster deve ser v1.17 ou posterior.
- Somente contas da Huawei Cloud, HUAWEI IDs ou usuários do IAM com permissões CCE administrator ou FullAccess podem executar todas as operações usando o Centro de monitoramento. Os usuários do IAM com a permissão CCE ReadOnlyAccess podem visualizar todos os recursos, mas não podem executar nenhuma operação.

Obter permissões de recursos

O Centro de monitoramento trabalha em estreita colaboração com os serviços em nuvem para monitoramento de clusters, relatórios de alarmes e notificações. Quando você acessa o Centro de monitoramento pela primeira vez, o Centro de monitoramento e o AOM solicitam automaticamente permissões para acessar esses serviços de nuvem na região onde você executa suas aplicações.

O Centro de monitoramento obtém as seguintes permissões de serviço:

- Permissões do CCE

  O Centro de monitoramento precisa acessar o CCE para obter informações sobre clusters, nós e cargas de trabalho para verificações de integridade.
- Permissões do SWR

  O Centro de monitoramento precisa acessar o SWR para obter informações de imagem.
- Permissões somente leitura para serviços de nuvem

  O Centro de monitoramento precisa acessar o CCE para obter informações sobre recursos de computação, rede e armazenamento para verificações de integridade.
- Permissões somente leitura para IAM

  O Centro de monitoramento precisa acessar o IAM para obter informações da agência.

Para obter detalhes sobre as permissões de serviço de nuvem solicitadas pelo AOM, consulte **Autorização de serviço de nuvem do AOM**.

Depois de concordar com a autorização, as agências são criadas automaticamente no IAM para delegar as permissões de operação de recursos necessárias em sua conta para o CCE e AOM da Huawei Cloud. Para obter detalhes sobre agências, consulte **Delegação do serviço de muvem**. As agências criadas automaticamente no IAM são:

- cia_admin_trust
- aom_admin_trust

A agência**cia_admin_trust** tem as permissões Tenant Guest e IAM ReadOnlyAccess em projetos globais, bem como as permissões Tenant Guest, CCE Administrator e SWR Administrator em projetos regionais. Essas permissões são exigidas pelo Centro de monitoramento para acessar outros serviços em nuvem.

Para usar o Centro de monitoramento em várias regiões, solicite as permissões Tenant Guest, CCE Administrator e SWR Administrator em cada região. Você pode acessar o console do IAM, escolher **Agencies** e clicar em **cia_admin_trust** para exibir os registros de autorização de cada região.

NOTA

O Centro de monitoramento pode falhar a execução conforme o esperado se as permissões necessárias não forem atribuídas. Ao usar o Centro de monitoramento, não exclua nem modifique a agência **cia_admin_trust**.

Para obter detalhes sobre a agência **aom_admin_trust**, consulte **Autorização de serviço de nuvem do AOM**.

Ativar o Centro de monitoramento

Antes de usar o Centro de monitoramento, ative-o. Ativar a CIA requer a instalação do kube-prometheus-stack, que reporta dados para instâncias de Prometheus do AOM. Portanto, você também precisa selecionar a instância do Prometheus de destino. Para fazer isso, execute as seguintes operações:

1. Clique no nome do cluster de destino e escolha **Monitoring Center** no painel de navegação.
2. Clique em **Enable**. Na página exibida, selecione uma instância de Prometheus existente do AOM ou crie uma nova e clique em **OK**.

Figura 9-19 Selecionar uma instância existente

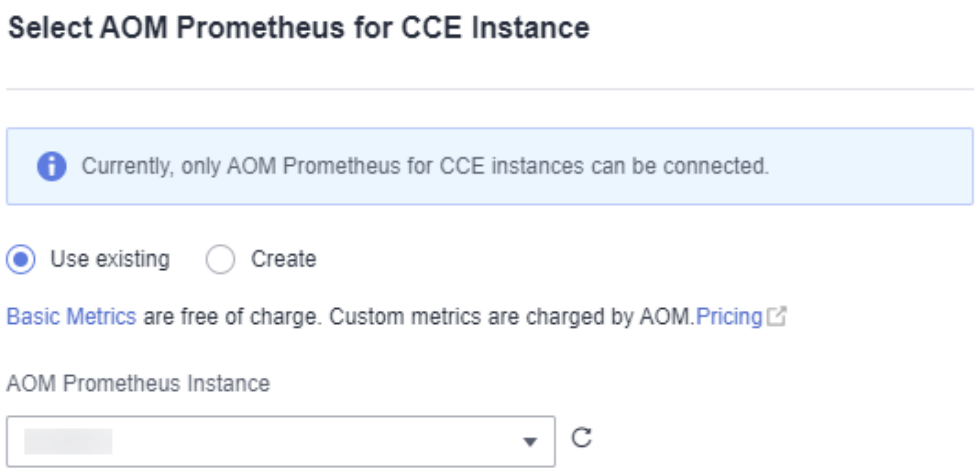
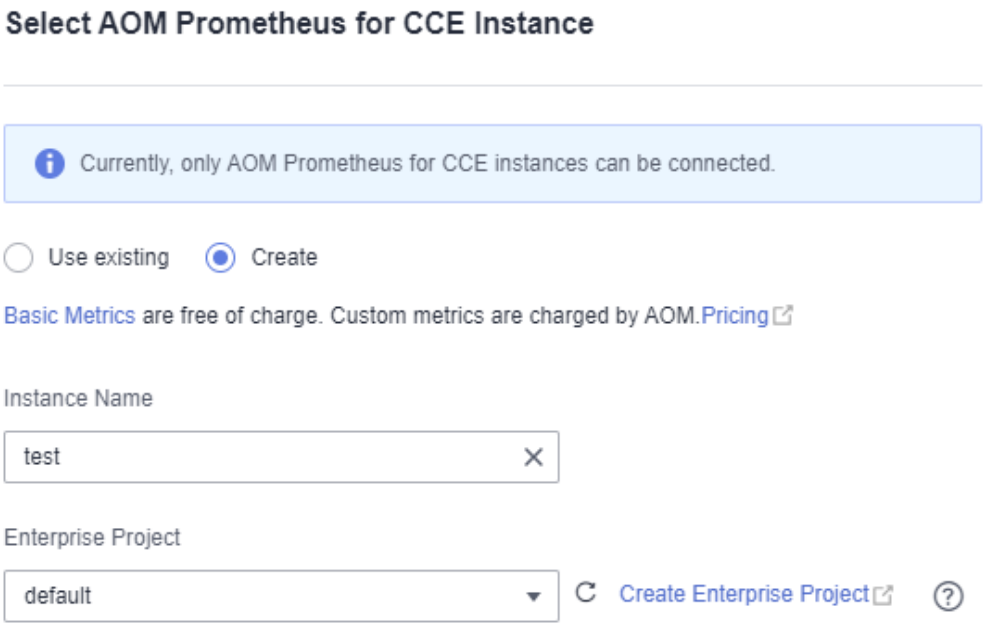


Figura 9-20 Criação de uma instância



9.3.2 Insight de contêiner

9.3.2.1 Cluster

Para observar o uso de recursos e o status de integridade de todo o cluster, clique em **Container Insight** > **Clusters**. A página exibida fornece as informações de monitoramento de um único cluster, incluindo **Integridade do recurso**, **Visão geral dos recursos**, **Estatísticas principais de consumo de recursos** e **Estatísticas de uso**.

Caminho de navegação

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster.

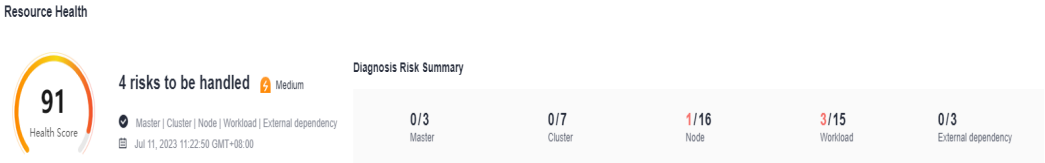
Passo 2 Escolha **Monitoring Center** no painel de navegação e clique em **Container Insight** > **Clusters**.

----Fim

Integridade do recurso

Se um complemento kube-prometheus-stack instalado em um cluster for implementado no modo **Server**, você poderá exibir o status de integridade do recurso do cluster.

Figura 9-21 Integridade do recurso



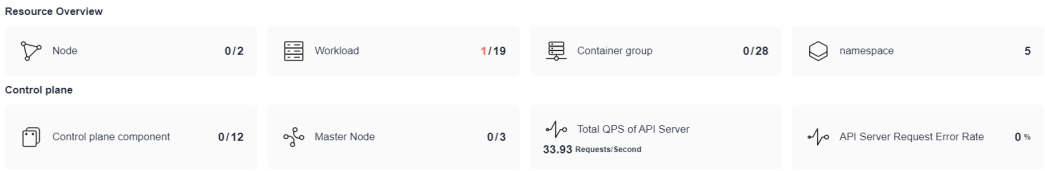
A avaliação de integridade do recurso inclui várias dimensões, como pontuação de integridade, número de itens de risco a serem processados, nível de risco e proporção de itens de risco diagnosticados em nós principais, clusters, nós de trabalho, cargas de trabalho e dependências externas. Os dados anormais são exibidos em vermelho. Para obter mais resultados de diagnóstico, vá para a página da guia **Diagnóstico de integridade**.

Visão geral dos recursos

Resource Overview exibe a proporção de recursos anormais em nós, cargas de trabalho e pods e o número total de namespaces. Além disso, a proporção de exceção de componentes do plano de controle e nós principais, o QPS total do servidor da API e a taxa de erro de solicitação do servidor da API também estão incluídos.

Como o provedor de serviços da API do cluster, se o servidor da API no plano de controle for anormal, todo o cluster pode falhar ao ser acessado e as cargas de trabalho que dependem do servidor da API podem falhar ao ser executadas corretamente. Para ajudá-lo a identificar e corrigir problemas rapidamente, a visão geral dos recursos fornece o QPS total e as métricas de taxa de erro de solicitação do servidor da API.

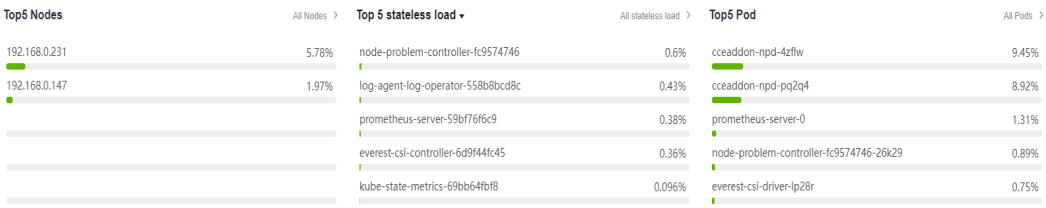
Figura 9-22 Visão geral dos recursos



Estatísticas principais de consumo de recursos

O CCE coleta estatísticas sobre nós, Implementações, StatefulSets e pods com 5 principais usos de CPU e memória, ajudando a identificar usuários com alto consumo de recursos. Para exibir todos os dados, vá para a página da guia **Nodes**, **Workloads** ou **Pods**.

Figura 9-23 Estatísticas principais de consumo de recursos



Explicação das métricas de monitoramento

- CPU Usage

  Uso da CPU da carga de trabalho = uso médio da CPU em cada pod da carga de trabalho

  Uso de CPU do pod = os núcleos de CPU usados/a soma de todos os limites de CPU dos pods (se não especificado, todos os núcleos de CPU do nó são usados.)

- Memory Usage

  Uso da memória da carga de trabalho = uso médio da memória em cada pod da carga de trabalho

Uso de memória do pod = a memória física usada/a soma de todos os limites de memória dos pods (se não for especificado, toda a memória do nó será usada.)

Estatísticas de uso

Por padrão, o uso de recursos de cada dimensão na última hora, nas últimas 8 horas e nas últimas 24 horas é coletado. Para exibir mais informações sobre monitoramento, clique em **View All Metrics** para acessar a página **Dashboard**. Para mais detalhes, consulte **Painel**.

NOTA

Você pode passar o mouse sobre um gráfico para visualizar os dados de monitoramento de cada minuto.

- **CPU**

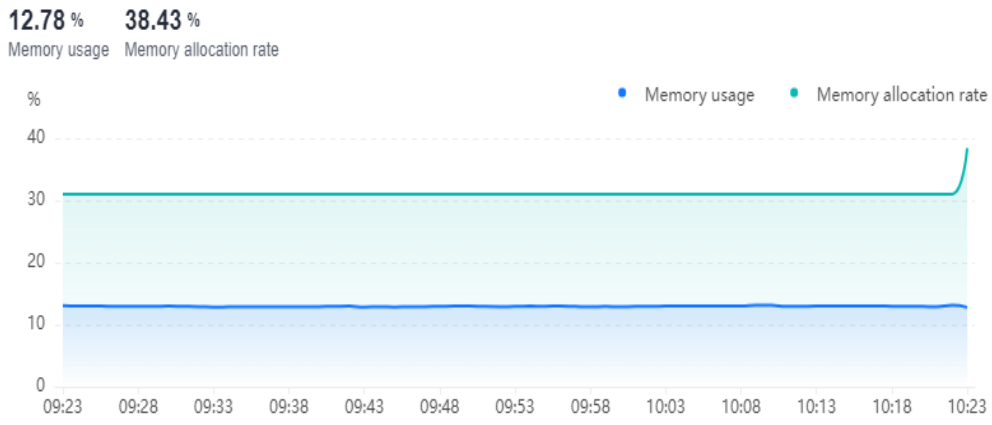
  Estatísticas sobre o uso da CPU do cluster em um período especificado.

  Figura 9-24 CPU

- **Memory**

  Estatísticas sobre o uso de memória do cluster em um período especificado.

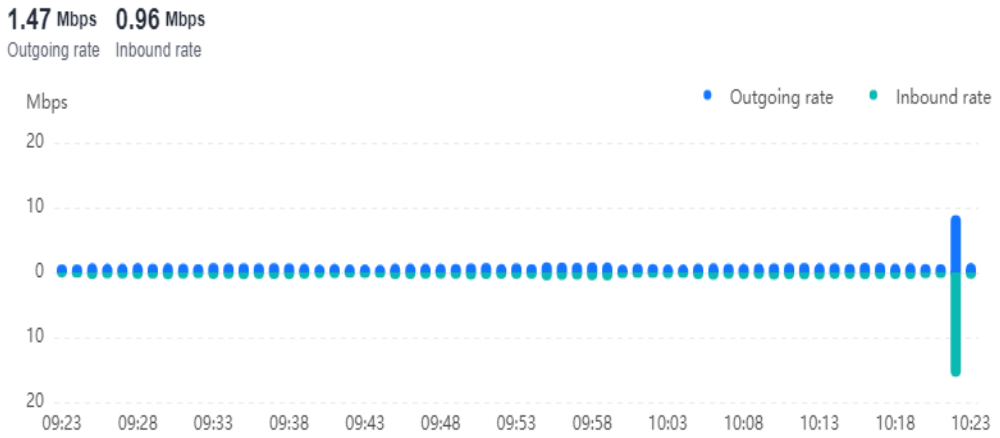
  Figura 9-25 Memória



- **Network**

  Tráfego de rede de contêineres, indicando a carga de rede do contêiner e a comunicação de rede entre contêineres.

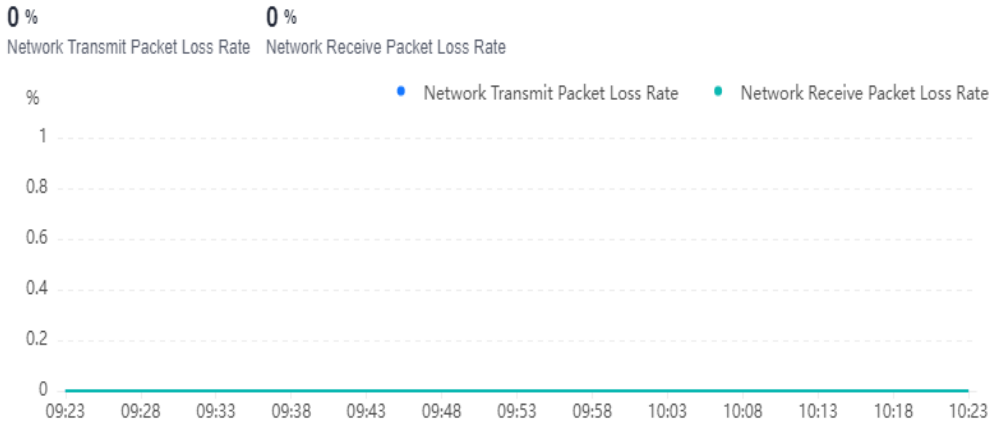
Figura 9-26 Rede



- **Network Packet Loss Rate**

  Relação entre o número de pacotes perdidos e o número total de pacotes transmitidos.

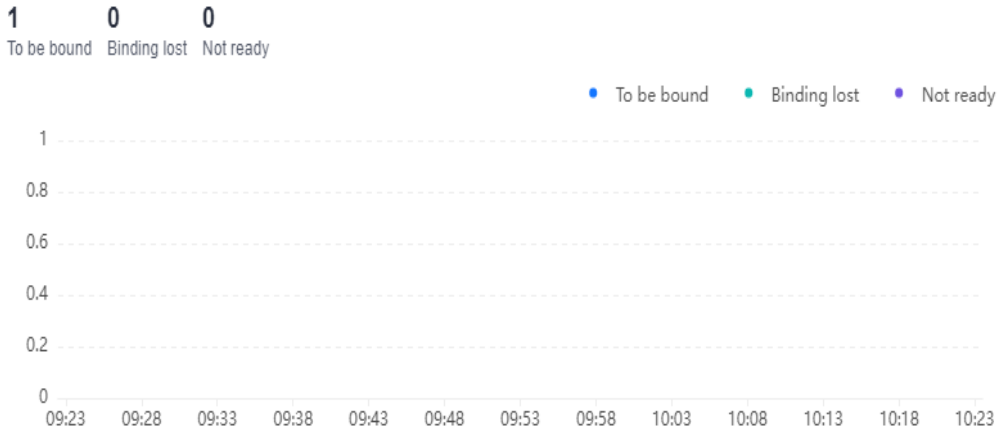
Figura 9-27 Taxa de perda de pacotes de rede



- **PVC Storage Status**

  Estado de vinculação entre a PVC e o PV.

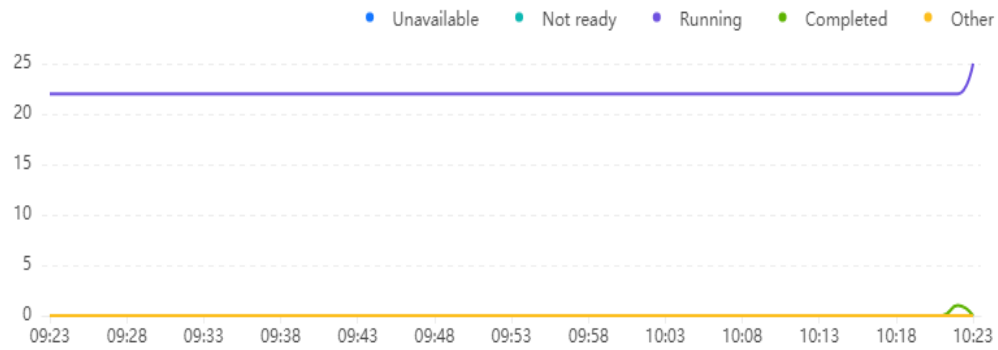
Figura 9-28 Status de armazenamento de PVC



- **Pod Status Trend**

  Monitora o status do pod no cluster em tempo real.

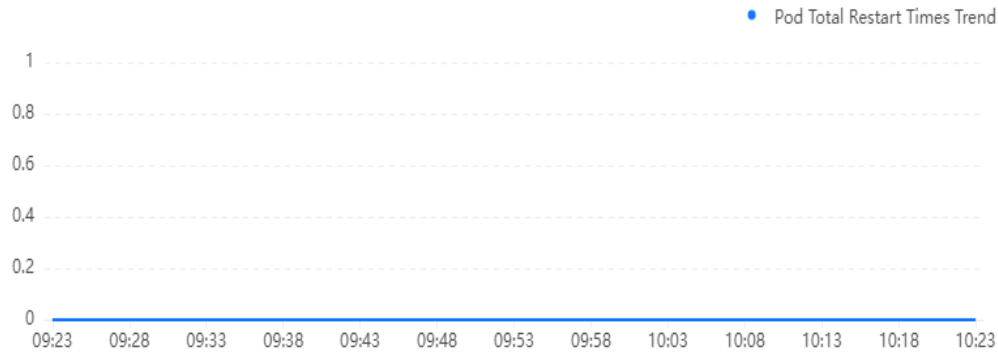
  Figura 9-29 Tendência de status do pod



- **Trend of Total Pod Restarts**

  Número total de reinicializações do pod no cluster nos últimos 5 minutos.

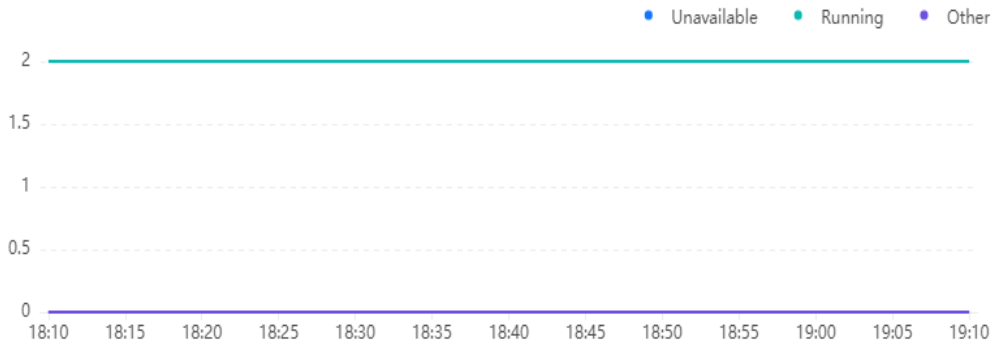
  Figura 9-30 Tendência de reinicializações totais do pod



- **Node Status Trend**

  Monitora o status do nó no cluster em tempo real.

  Figura 9-31 Tendência de status do nó



9.3.2.2 Nós

Para monitorar o uso de recursos dos nós, clique em **Container Insight** > **Nodes**. Esta página fornece informações abrangentes sobre todos os nós em um cluster especificado e dados de

monitoramento detalhados de um único nó, incluindo o uso de CPU/memória, a taxa de entrada/saída da rede e a taxa de leitura ou gravação de I/O de disco.

Caminho de navegação

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster.

Passo 2 Escolha **Monitoring Center** no painel de navegação e clique em **Container Insight** > **Nodes**.

A página exibe informações abrangentes sobre todos os nós. Para exibir as informações de monitoramento de um nó, clique no nome do nó para acessar sua página **Overview** e alterne para a página de guia **Pods** ou **Monitoring**.

----Fim

Lista de nós

Esta página de guia lista o nome, o status, o endereço IP, o número de pods (alocados/total), a solicitação/limite/uso de CPU e a solicitação/limite/uso de memória de cada nó.

Figura 9-32 Lista de nós

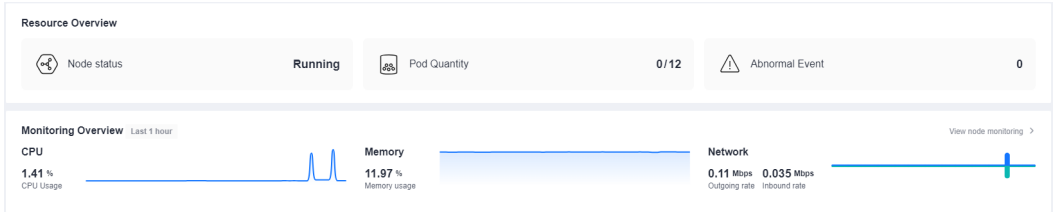
Você pode filtrar os nós por nome, status, endereço IP privado ou endereço IP público para encontrar rapidamente os nós desejados. Você pode clicar em  no canto superior direito da lista para exportar dados de todos os nós ou nós selecionados. O arquivo exportado está no formato .xlsx e o nome do arquivo contém o carimbo de data/hora.

Se a taxa de limite de CPU ou a taxa de limite de memória de um nó exceder 100%, os recursos do nó serão comprometidos em excesso e a soma dos limites de carga de trabalho (valores máximos disponíveis) do nó excederá as especificações do nó. Se uma carga de trabalho ocupa muitos recursos, o nó pode ser anormal.

Visão geral

Esta página de guia mostra a visão geral do recurso, incluindo o status do nó, o número de pods e eventos anormais. Você também pode ver a visão geral do monitoramento da última hora, incluindo o uso da CPU, o uso da memória e a taxa de entrada/saída da rede.

Figura 9-33 Visão geral de recursos e visão geral de monitoramento



A página de guia **Overview** também mostra a tendência de uso do pod. Você pode alternar as métricas no canto superior esquerdo do gráfico para visualizar os pods do nó em termos de uso da CPU, CPUs usadas, uso de memória ou memória usada. Você também pode clicar em **Top 5 (Descending)** ou **Top 5 (Ascending)** no canto superior direito para visualizar os 5 principais dados em ordem decrescente ou crescente.

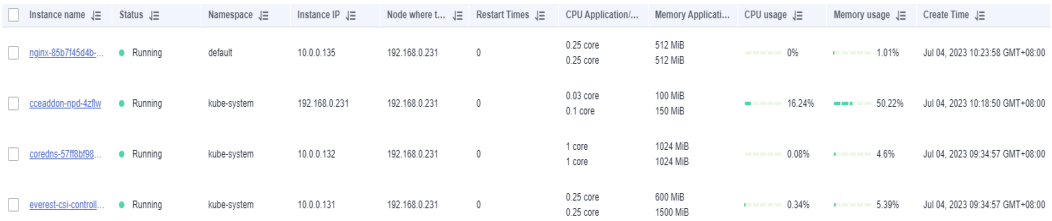
Figura 9-34 Tendência de uso de pods

Para obter mais métricas, vá para a página de guia **Monitoramento**.

Pods

Esta página de guia lista o nome, status, namespace, endereço IP, nó, número de reinicializações, solicitação/limite de CPU, solicitação/limite de memória e uso de CPU e memória de cada pod.

Figura 9-35 Pods



Você pode filtrar pods por nome, status, namespace, endereço IP ou nó para localizar rapidamente o pod desejado. Você pode clicar em  no canto superior direito da lista para exportar dados de todos os pods ou pods selecionados. O arquivo exportado está no formato .xlsx e o nome do arquivo contém o carimbo de data/hora.

Você pode clicar no nome de um pod para visualizar seus dados de monitoramento detalhados. Para obter mais informações, consulte **Pods**.

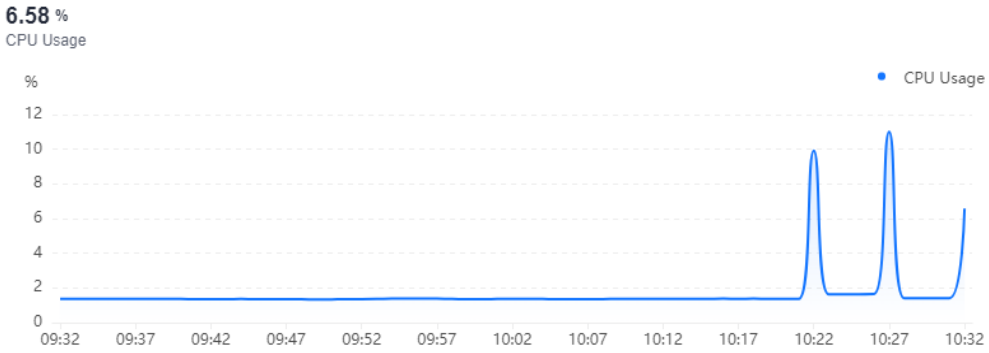
Monitoramento

Esta página de guia mostra o uso de recursos do nó em cada dimensão nas últimas 1 hora, últimas 8 horas, últimas 24 horas ou um período personalizado. Para exibir mais informações de monitoramento, clique em **View Dashboard** para acessar a página **Dashboard**. Para mais detalhes, consulte **Painel**.

- **CPU**

  Uso da CPU do nó em um período especificado.

  Figura 9-36 CPU

- **Memory**

  Uso da memória do nó em um período especificado.

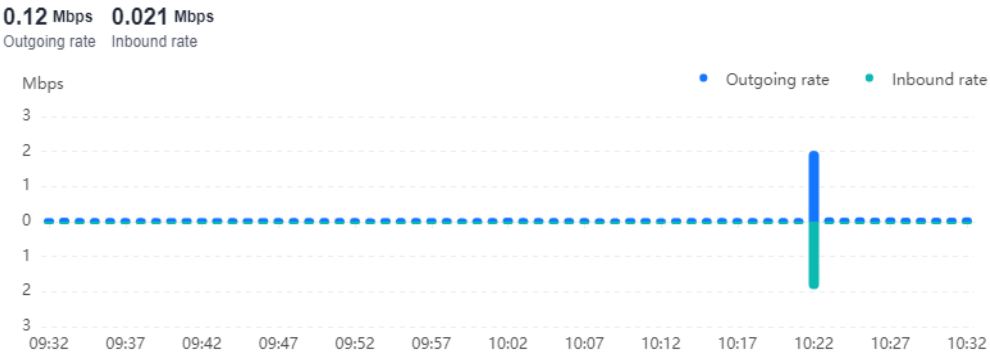
  Figura 9-37 Memória

- **Network**

  O tráfego de rede de contêineres em um nó pode ser usado para monitorar o desempenho da rede de contêineres no nó e a comunicação de rede entre contêineres.

  Figura 9-38 Rede

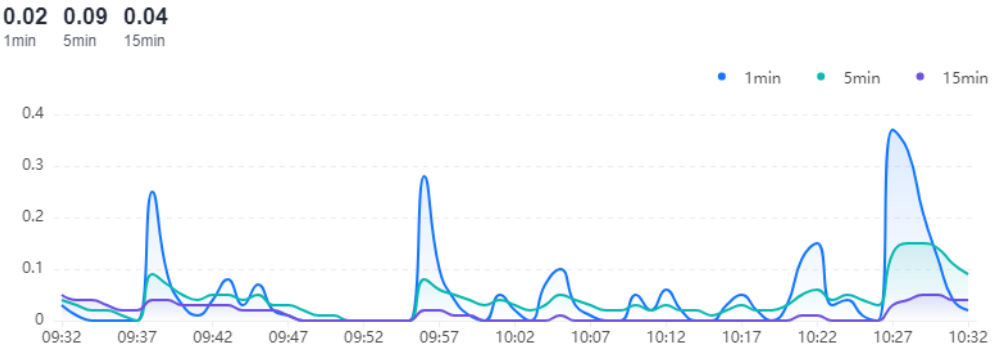
  

- **Average Node Loads**

  Número médio de processos em execução no nó em um período especificado. Essa métrica é usada para verificar se o número de processos em execução no nó excede sua

capacidade de processamento. Geralmente, ela deve ser mantida dentro de uma faixa razoável para garantir estabilidade e confiabilidade para o nó. Ela mostra dados nos últimos 1 minuto, 5 minutos e 15 minutos.

Figura 9-39 Cargas médias do nó



- **Single-Core Usage of Node CPU**

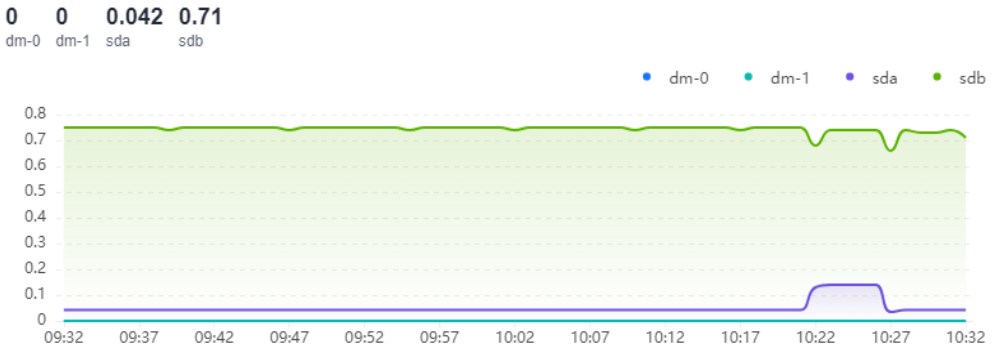
  Uso da CPU de núcleo único do nó em um período especificado.

Figura 9-40 Uso de núcleo único da CPU do nó

- **Disk Read I/O**

  Quantidade de dados lidos do disco de nó nos últimos 5 minutos.

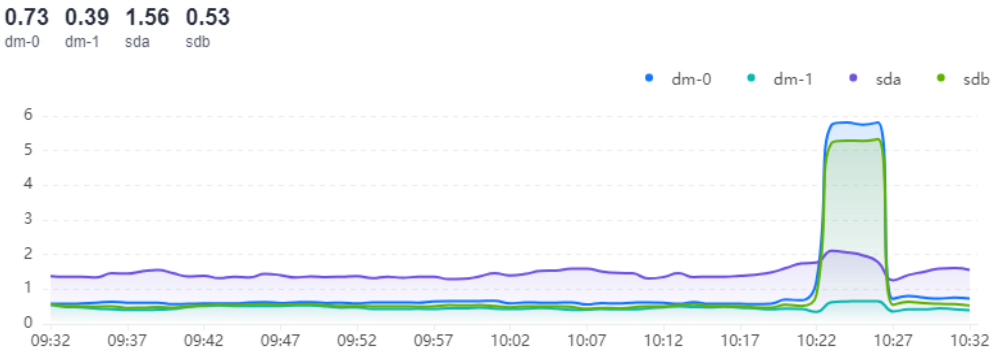
Figura 9-41 I/O de leitura de disco



- **Disk Write I/O**

  Quantidade de dados gravados no disco de nó nos últimos 5 minutos.

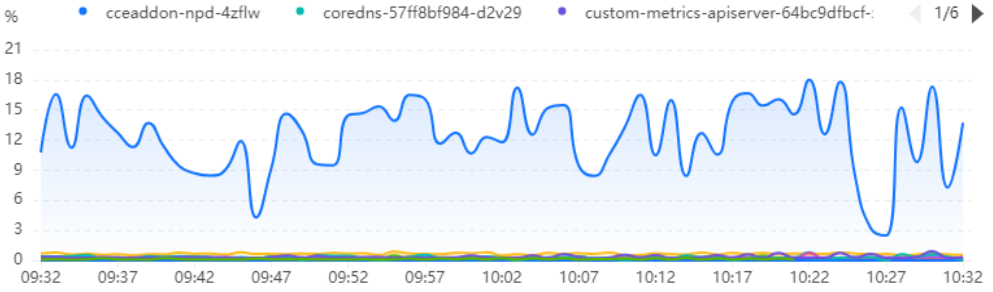
Figura 9-42 I/O de gravação de disco



- **Pod CPU Usage**

  Pod uso da CPU em um período especificado.

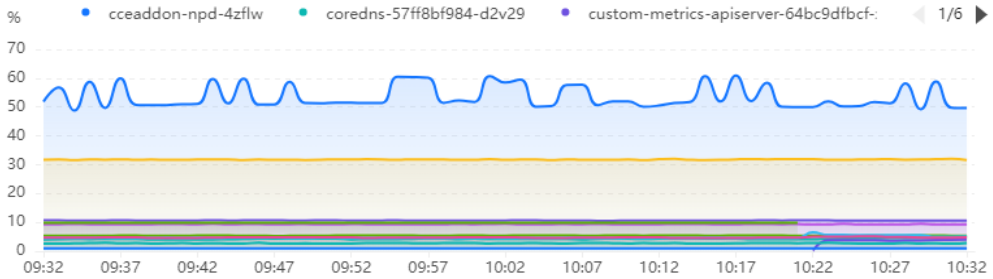
Figura 9-43 Uso da CPU do pod



- **Pod Memory Usage**

  Uso de memória do pod em um período especificado.

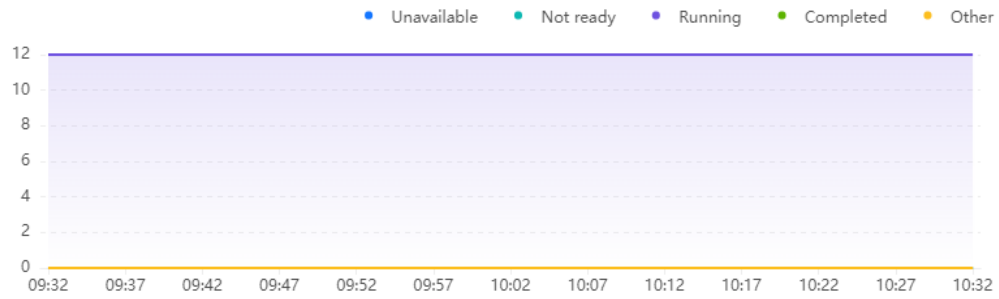
Figura 9-44 Uso da memória do pod



- **Pod Status Trend**

  Monitora o status do pod no nó em tempo real.

Figura 9-45 Tendência de status do pod



- **Pod Quantity Trend**

  Número de pods alocados para o nó.

Figura 9-46 Tendência de quantidade de pods

9.3.2.3 Cargas de trabalho

Para monitorar o uso de recursos das cargas de trabalho, clique em **Container Insight** > **Workloads**. Esta página fornece informações abrangentes sobre todas as cargas de trabalho em um cluster especificado e dados de monitoramento detalhados de uma única carga de trabalho, incluindo o uso de CPU/memória, a taxa de entrada/saída da rede e o uso do disco.

Caminho de navegação

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster.

Passo 2 Escolha **Monitoring Center** no painel de navegação e clique em **Container Insight** > **Workloads**.

A página exibe informações abrangentes sobre todas as cargas de trabalho. Para exibir as informações de monitoramento de uma carga de trabalho, clique no nome da carga de trabalho para acessar sua página **Overview** e alterne para a página de guia **Pods** ou **Monitoring**.

----Fim

Lista de cargas de trabalho

A lista de cargas de trabalho mostra o nome, o status, o número de pods (normais ou todos), o namespace, o nome da imagem, o uso da CPU e o uso da memória de cada carga de trabalho.

Figura 9-47 Lista de cargas de trabalho

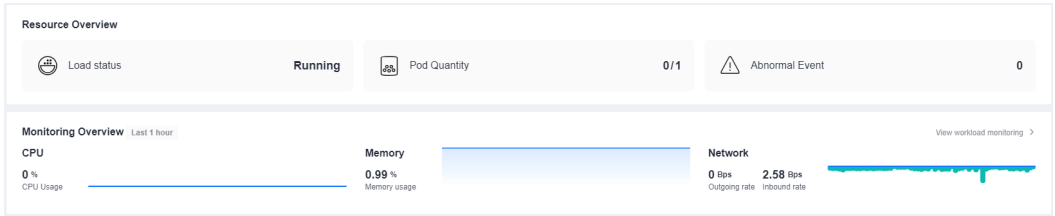
Você pode selecionar um namespace ou tipo de carga de trabalho no canto superior direito ou selecionar **Workload name**, **Status** e **Namespace** acima da lista para localizar rapidamente a carga de trabalho necessária.

Você pode clicar em  no canto superior direito da lista para exportar dados de todas as cargas de trabalho ou cargas de trabalho selecionadas. O arquivo exportado está no formato .xlsx e o nome do arquivo contém o carimbo de data/hora.

Visão geral

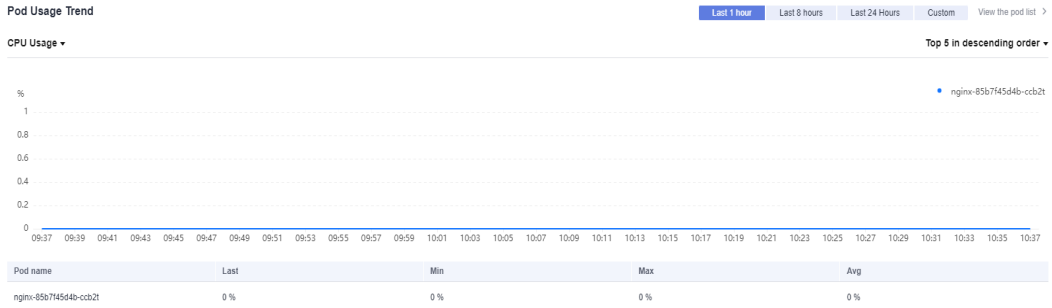
Esta página de guia mostra a visão geral do recurso, incluindo o status da carga de trabalho, número de pods (anormal/total) e eventos anormais. Você também pode ver a visão geral do monitoramento da última hora, incluindo o uso da CPU, o uso da memória e a taxa de entrada/saída da rede.

Figura 9-48 Visão geral de recursos e visão geral de monitoramento



A página de guia **Overview** também mostra a tendência de uso do pod. Você pode alternar as métricas no canto superior esquerdo do gráfico para ver o uso da CPU, as CPUs usadas, o uso da memória e a memória usada de cada pod da carga de trabalho. Você também pode clicar em **Top 5 (Descending)** ou **Top 5 (Ascending)** no canto superior direito para visualizar os 5 principais dados em ordem decrescente ou crescente.

Figura 9-49 Tendência de uso de pods

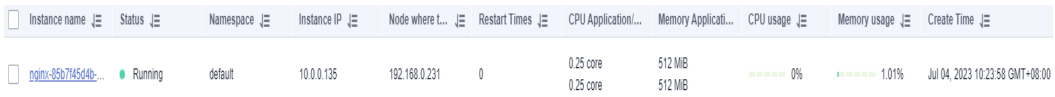


Para mais métricas, vá para a página de guia **Monitoramento**.

Pods

Esta página de guia lista o nome, status, namespace, endereço IP, nó, número de reinicializações, solicitação/limite de CPU, solicitação/limite de memória e uso de CPU e memória de cada pod.

Figura 9-50 Pods



Você pode filtrar pods por nome, status, namespace, endereço IP ou nó para localizar rapidamente o pod desejado. Você pode clicar em  no canto superior direito da lista para exportar dados de todos os pods ou pods selecionados. O arquivo exportado está no formato .xlsx e o nome do arquivo contém o carimbo de data/hora.

Você pode clicar no nome de um pod para visualizar seus dados de monitoramento detalhados. Para obter mais informações, consulte **Pods**.

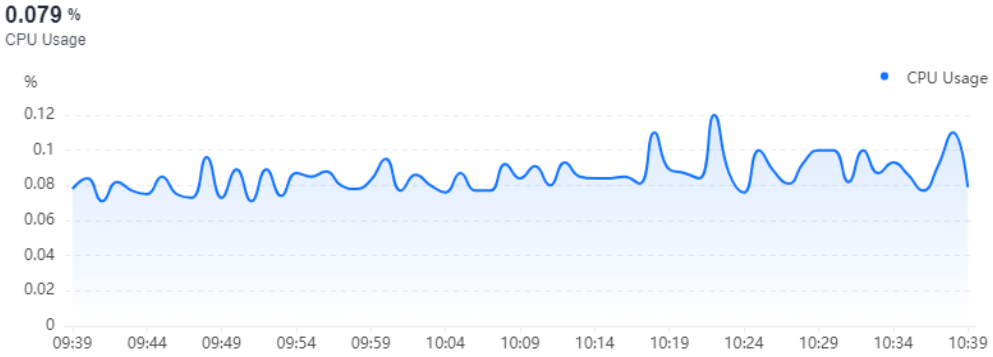
Monitoramento

Esta página de guia mostra o uso de recursos da carga de trabalho em cada dimensão nas últimas 1 hora, últimas 8 horas, últimas 24 horas ou um período personalizado. Para exibir mais informações de monitoramento, clique em **View Dashboard** para acessar a página **Dashboard**. Para mais detalhes, consulte **Painel**.

- **CPU**

  Uso médio da CPU da carga de trabalho em um período especificado.

Figura 9-51 CPU



- **Memory**

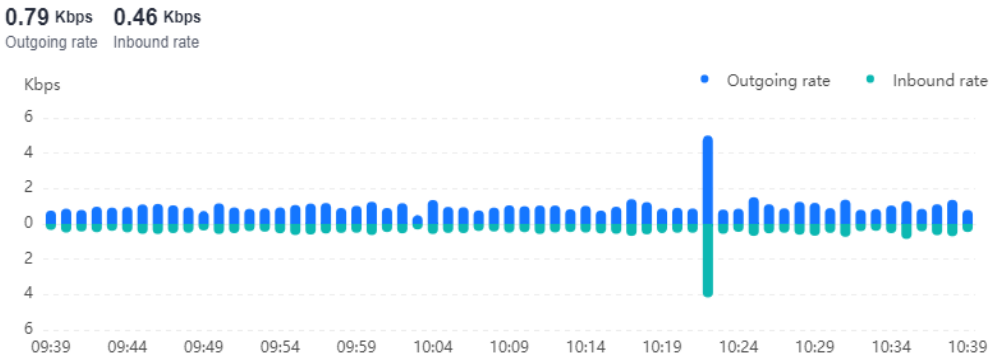
  Uso médio da memória da carga de trabalho em um período especificado.

Figura 9-52 Memória

- **Network**

  Tráfego de rede de contêineres, indicando a carga de rede do contêiner e a comunicação de rede entre contêineres.

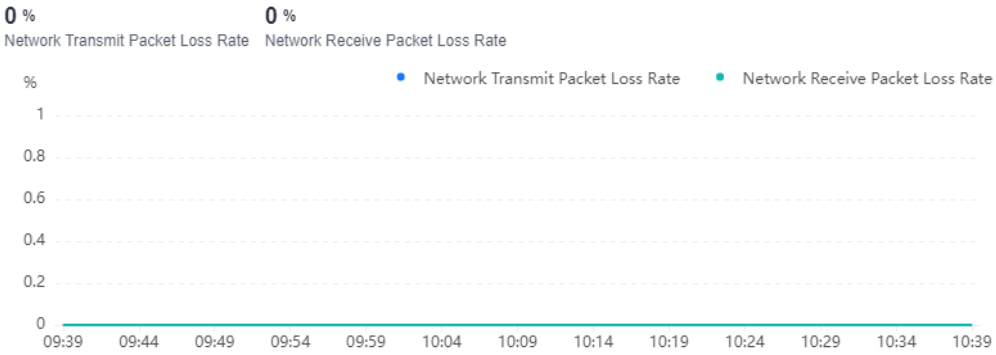
Figura 9-53 Rede



- **Network Packet Loss Rate**

  Proporção entre o número de pacotes perdidos e o número total de pacotes transmitidos.

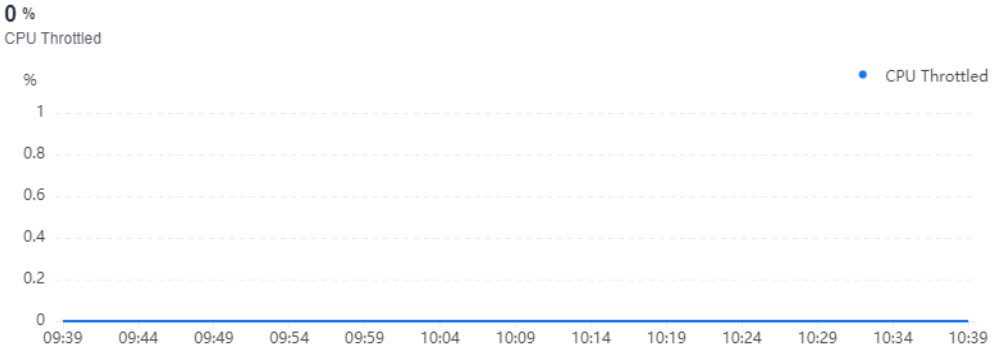
Figura 9-54 Taxa de perda de pacotes de rede



- **CPU Throttled**

  Proporção entre o número de ciclos de CPU restritos e o número total de ciclos no ciclo de vida do contêiner da carga de trabalho.

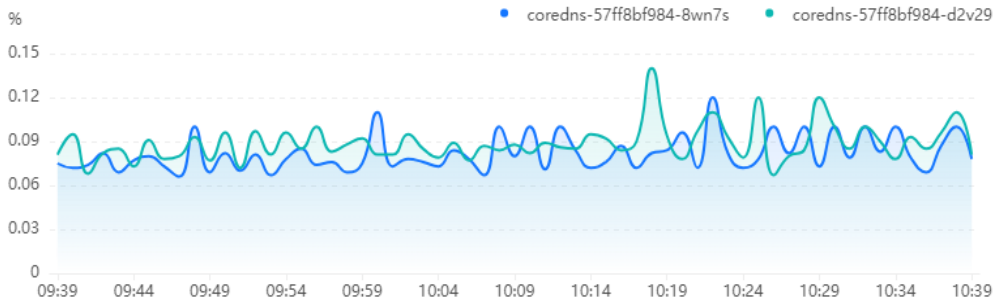
Figura 9-55 CPU acelerada



- **Pod CPU Usage**

  Uso da CPU do pod em um período especificado.

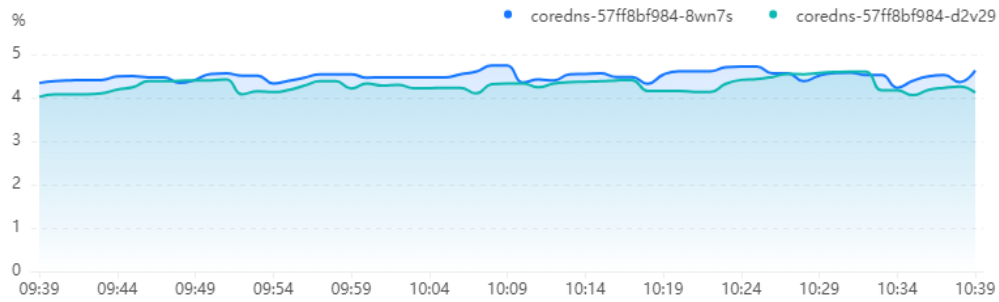
Figura 9-56 Uso da CPU do pod



- **Pod Memory Usage**

  Uso de memória do pod em um período especificado.

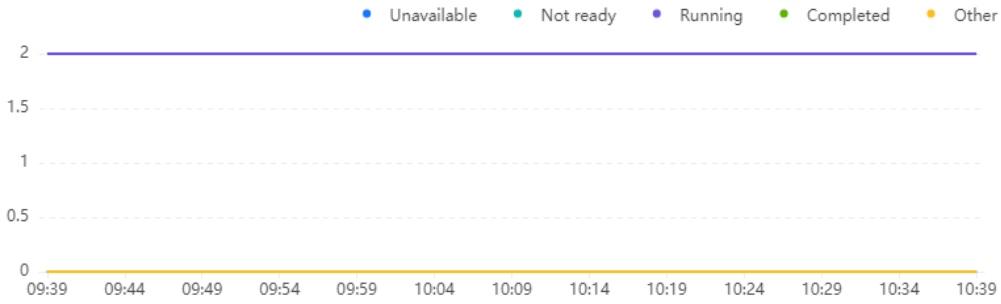
Figura 9-57 Uso da memória do pod



- **Pod Status Trend**

  Monitora o status do pod na carga de trabalho em tempo real.

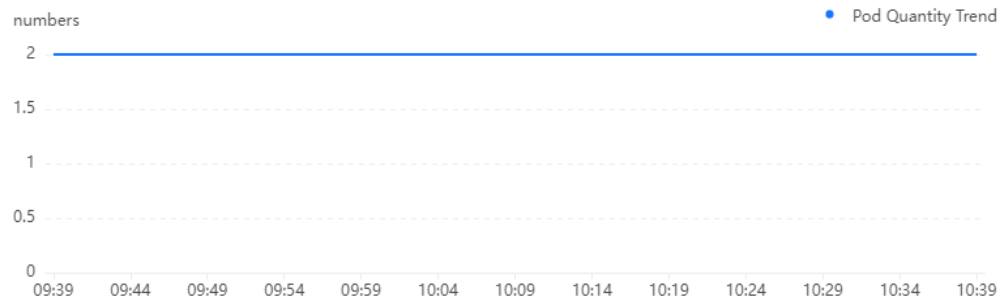
Figura 9-58 Tendência de status do pod



- **Pod Quantity Trend**

  Número de pods alocados para a carga de trabalho.

Figura 9-59 Tendência de quantidade de pods



9.3.2.4 Pods

Para monitorar o uso de recursos dos pods, clique em **Container Insight** > **Pods**. Esta página fornece informações abrangentes sobre todos os pods em um cluster especificado e dados de monitoramento detalhados de um único pod, incluindo o uso de CPU/memória, a taxa de entrada/saída da rede e o uso do disco.

NOTA

Pods e instâncias são o mesmo conceito.

Caminho de navegação

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster.

Passo 2 Escolha **Monitoring Center** no painel de navegação e clique em **Container Insight** > **Pods**.

A página exibe informações abrangentes sobre todos os pods. Para exibir as informações de monitoramento de um pod, clique no nome do pod para acessar sua página **Overview** e alterne para a página de guia **Containers** ou **Monitoring**.

----Fim

Pods

Esta lista mostra o nome, o status, o namespace, o endereço IP, o nó, o número de reinicializações, a solicitação/limite da CPU, a solicitação/limite de memória, o uso da CPU e o uso da memória de cada pod.

Figura 9-60 Pods

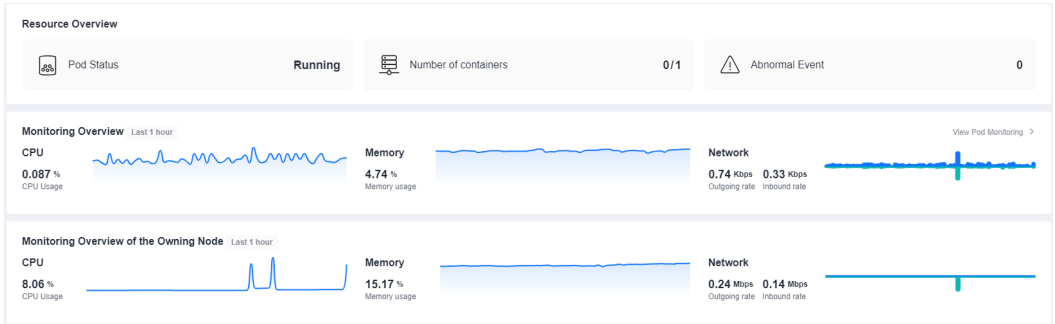
Você pode selecionar um namespace no canto superior direito ou selecionar **Pod**, **Status**, **Namespace**, **Pod IP** e **Node** acima da lista para localizar rapidamente o pod necessário.

Você pode clicar em  no canto superior direito da lista para exportar dados de todos os pods ou pods selecionados. O arquivo exportado está no formato .xlsx e o nome do arquivo contém o carimbo de data/hora.

Visão geral

Esta página de guia mostra a visão geral do recurso, incluindo o status do pod, número de contêineres (anormal/total) e eventos anormais. Você também pode ver a visão geral do monitoramento do pod e do nó na última hora, incluindo o uso da CPU, o uso da memória e a taxa de entrada/saída da rede.

Figura 9-61 Visão geral de recursos e visão geral de monitoramento



A página de guia **Overview** também mostra a tendência de uso do contêiner. Você pode alternar as métricas no canto superior esquerdo do gráfico para ver o uso da CPU, as CPUs usadas, o uso da memória e a memória usada de cada contêiner no pod. Você também pode clicar em **Top 5 (Descending)** ou **Top 5 (Ascending)** no canto superior direito para visualizar os 5 principais dados em ordem decrescente ou crescente.

Figura 9-62 Tendência de uso de contêineres

Para obter mais métricas, vá para a página de guia **Monitoramento**.

Contêineres

Essa página de guia contém detalhes como nome, status, namespace, número de reinicializações e imagem de cada contêiner.

Figura 9-63 Contêineres

Você pode filtrar contêineres por nome, status ou namespace para encontrar rapidamente o contêiner desejado. Você pode clicar em  no canto superior direito da lista para exportar dados de todos os contêineres ou contêineres selecionados. O arquivo exportado está no formato .xlsx e o nome do arquivo contém o carimbo de data/hora.

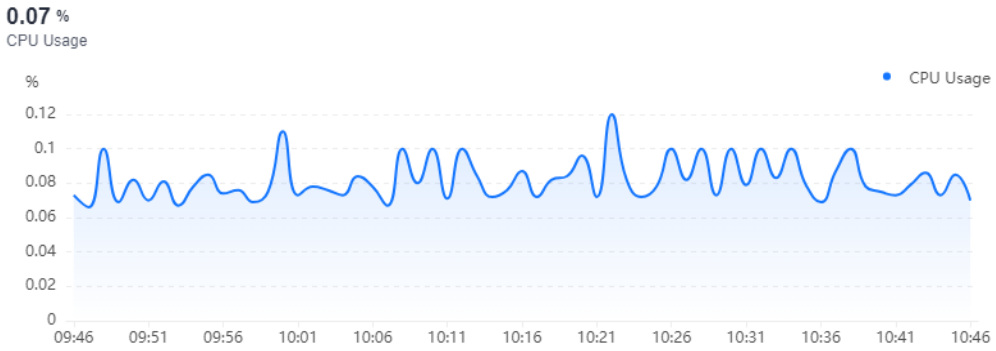
Monitoramento

Esta página de guia mostra o uso de recursos do pod em cada dimensão nas últimas 1 hora, últimas 8 horas, últimas 24 horas ou um período personalizado. Para exibir mais informações de monitoramento, clique em **View Dashboard** para acessar a página **Dashboard**. Para mais detalhes, consulte **Painel**.

- **CPU**

  Uso médio da CPU do pod em um período especificado.

Figura 9-64 CPU



- **Memory**

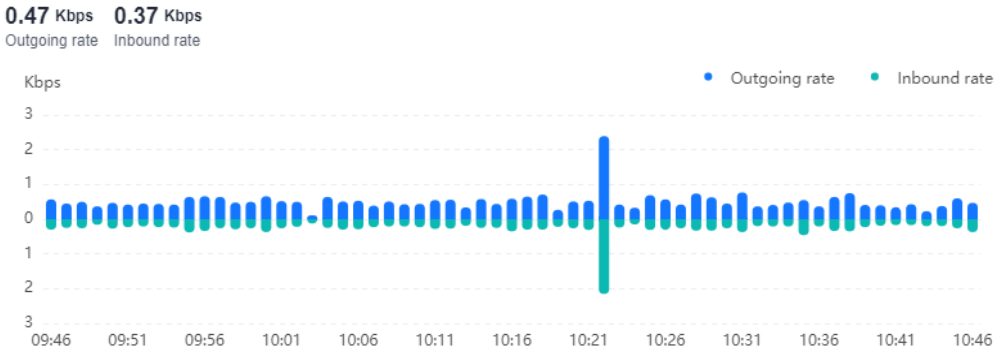
  Uso médio de memória do pod em um período especificado.

Figura 9-65 Memória

- **Network Rate**

  Tráfego de rede de contêiner, indicando a carga de rede do contêiner e a comunicação de rede entre contêineres.

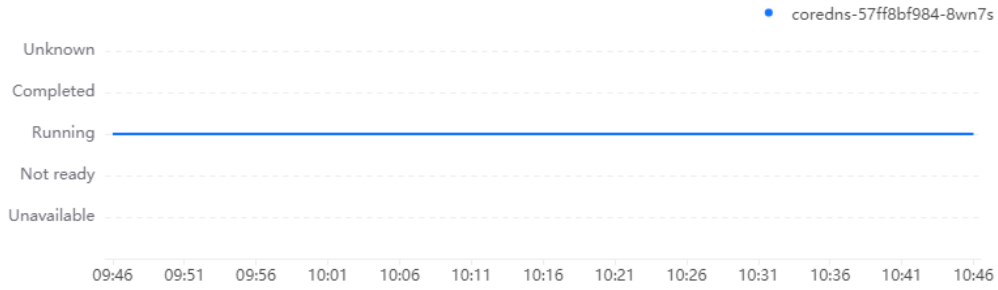
Figura 9-66 Taxa de rede



- **Historical Pod Status**

  Ciclo de vida do pod.

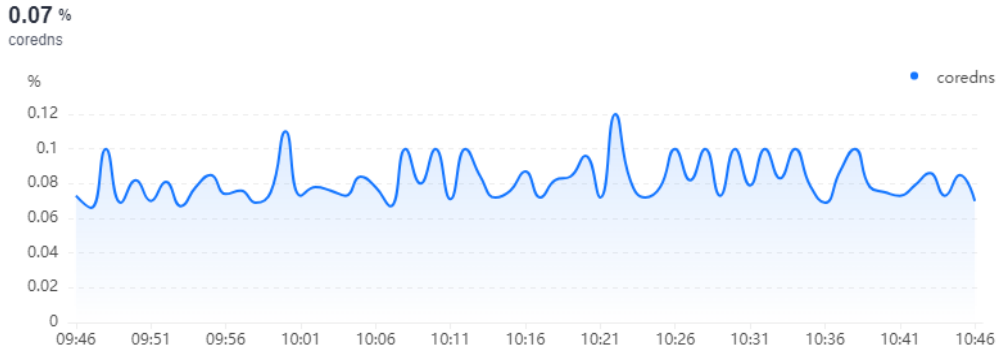
Figura 9-67 Status histórico do pod



- **Container CPU**

  Uso médio da CPU do contêiner em um período especificado.

Figura 9-68 CPU do contêiner



- **Container Memory**

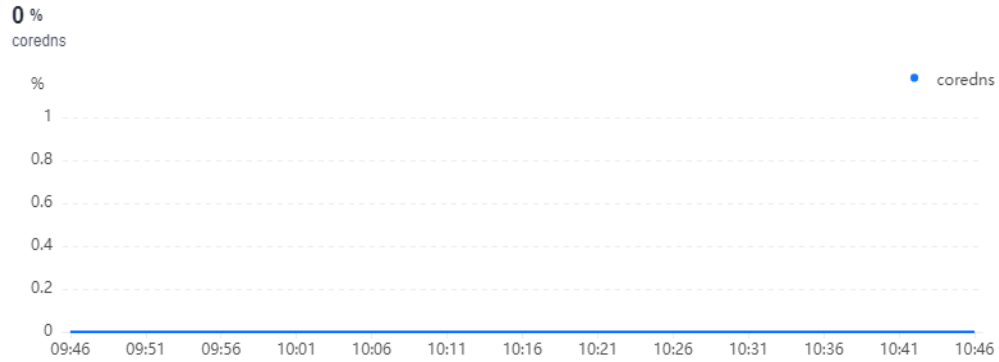
  Uso médio de memória do contêiner em um período especificado.

Figura 9-69 Memória do contêiner

- **Container CPU Throttled**

  Proporção entre o número de ciclos de CPU restritos e o número total de ciclos no ciclo de vida do contêiner.

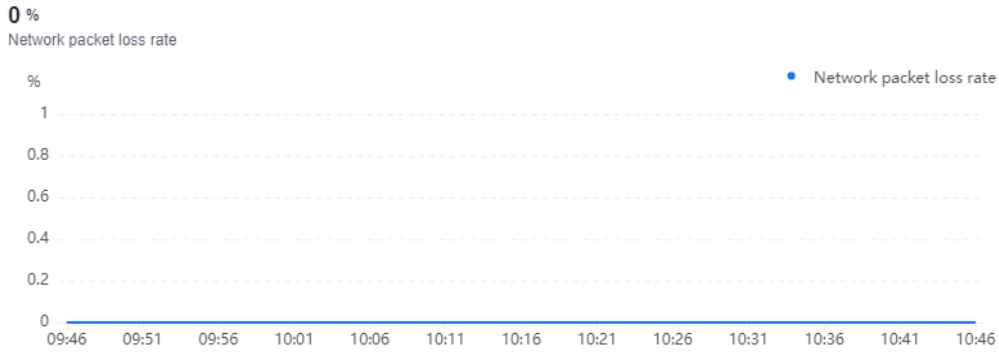
Figura 9-70 CPU do contêiner limitada



- **Container Network Packet Loss Rate**

  Proporção do número de pacotes de contêiner perdidos para o número total de pacotes transmitidos.

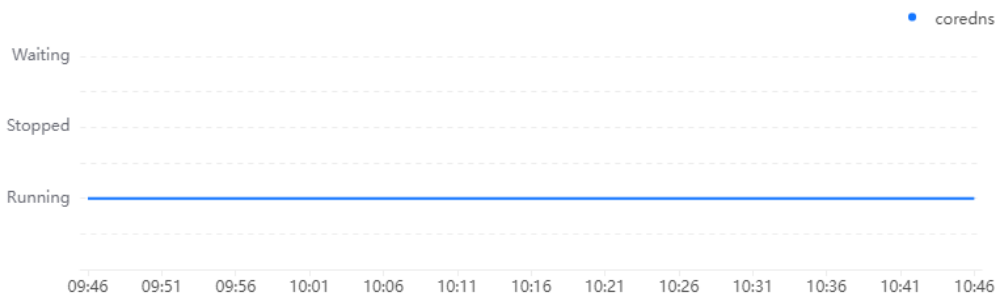
Figura 9-71 Taxa de perda de pacotes de rede do contêiner



- **Historical Container Status**

  Ciclo de vida do contêiner.

Figura 9-72 Status do contêiner histórico



9.3.2.5 Eventos

Os eventos do Kubernetes mostram o status de execução do cluster e o status de agendamento de recursos, ajudando a equipe de O&M a observar alterações de recursos e localizar falhas. Para ativar essa função, você precisa instalar o complemento log-agent no cluster. O log-agent pode coletar eventos do Kubernetes e exibi-los na página **Container Insight** > **Events**.

Caminho de navegação

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster.

Passo 2 Escolha **Monitoring Center** no painel de navegação e clique em **Container Insight** > **Events**.

A página **Events** tem duas páginas de guia: **Overview** e **Events**. Na página de guia **Overview**, você pode exibir o número total, a tendência e a classificação de eventos no cluster. Na página de guia **Events**, você pode exibir os detalhes do evento, incluindo o nome, o tipo, o conteúdo e as informações sobre o recurso que aciona o evento.

----Fim

Visão geral

Por default, a página de guia **Overview** exibe as estatísticas de eventos de todos os namespaces no cluster. Você também pode selecionar um namespace especificado na lista suspensa no canto superior direito para exibir seus dados de evento.

Figura 9-73 exibe o gráfico de rosca **Total Events** que mostra a distribuição de eventos normais e de alerta, o gráfico **Top 5 Warning Events by Resource** que mostra as informações de recursos correspondentes aos 5 principais eventos de alerta, e o gráfico **Warning Events by Resource Type** que mostra a comparação entre o número de eventos de alerta e o número de eventos de aviso nas últimas 24 horas.

Figura 9-73 Estatísticas de eventos

Os gráficos de barras em Figura 9-74 exibem a tendência do número de eventos normais e eventos de alerta nas últimas 24 horas.

Figura 9-74 Tendência de eventos de alerta/normal

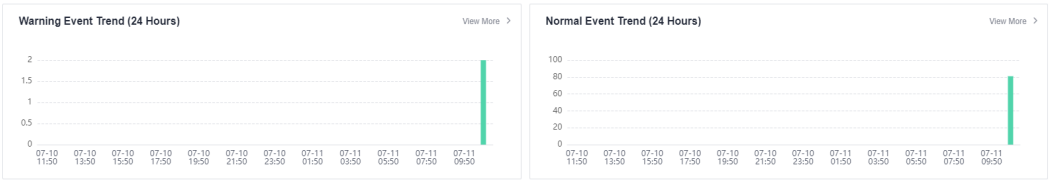
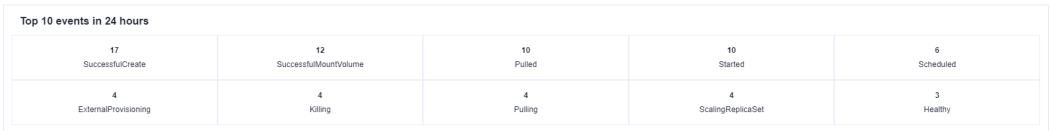


Figura 9-75 exibe os nomes dos 10 principais eventos nas últimas 24 horas.

Figura 9-75 Top 10 eventos em 24 horas



Eventos

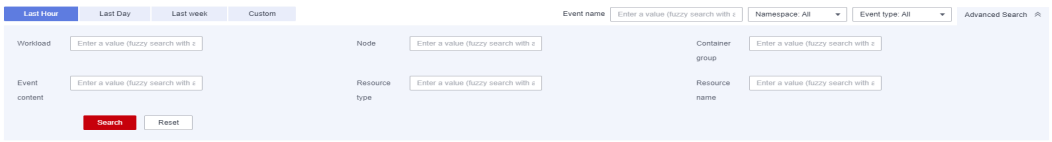
Procurar eventos

Na página de guia **Events**, você pode pesquisar um recurso especificado com base em determinados critérios para exibir convenientemente suas informações de evento, incluindo a tendência e os detalhes de eventos normais e de alerta.

Pesquise eventos de uma das seguintes maneiras:

- Digite o nome do evento a ser pesquisado na caixa de texto, selecione um namespace ou tipo de evento e clique em **Search**.
- Clique em **Advanced Search** e insira a carga de trabalho, o nó, o pod, o conteúdo do evento, o tipo de recurso ou o nome do recurso desejado.
- Selecione um intervalo de tempo no canto superior esquerdo para exibir os eventos gerados nesse período, incluindo última hora, último dia, última semana e um intervalo personalizado.

Figura 9-76 Procurar eventos



Lista de eventos

Você pode exibir detalhes sobre eventos que atendem aos critérios de pesquisa na lista. Os detalhes incluem a hora da última ocorrência, o nome do evento, o tipo de recurso, o nome do recurso, o conteúdo do evento, o tipo de evento e as horas de ocorrência. Clique em **Historical Events** na coluna **Operation**. Uma caixa de diálogo é exibida para mostrar todos os eventos do tipo de recurso atual e do recurso.

Figura 9-77 Lista de eventos

Last occurrence time	Event name	Resource type	Resource name	Event content	Event type	Number of oc...	Operation
Jul 11, 2023 10:22:16 GMT+08:00	Healthy	Pod	prometheus-server-0	container containerd://923e6e4c45ddfcbe25d94b99d6445de1df68...	Normal	2	Historical Event
Jul 11, 2023 10:22:15 GMT+08:00	Healthy	Pod	prometheus-server-0	container containerd://923e6e4c45ddfcbe25d94b99d6445de1df68...	Normal	1	Historical Event
Jul 11, 2023 10:22:12 GMT+08:00	SuccessfulMountVolume	Pod	prometheus-server-0	Successfully mounted volumes for pod "prometheus-server-0_mon...	Normal	2	Historical Event
Jul 11, 2023 10:22:12 GMT+08:00	Started	Pod	prometheus-server-0	Started container config-reloader	Normal	1	Historical Event
Jul 11, 2023 10:22:12 GMT+08:00	SuccessfulCreate	Pod	prometheus-server-0	Created container config-reloader	Normal	1	Historical Event
Jul 11, 2023 10:22:12 GMT+08:00	Pulled	Pod	prometheus-server-0	Container image "swr.cn-south-1.myhuaweicloud.com/hwofficial/pr...	Normal	1	Historical Event
Jul 11, 2023 10:22:12 GMT+08:00	Started	Pod	prometheus-server-0	Started container prometheus	Normal	1	Historical Event
Jul 11, 2023 10:22:12 GMT+08:00	SuccessfulCreate	Pod	prometheus-server-0	Created container prometheus	Normal	1	Historical Event
Jul 11, 2023 10:22:12 GMT+08:00	Pulled	Pod	prometheus-server-0	Container image "swr.cn-south-1.myhuaweicloud.com/hwofficial/pr...	Normal	1	Historical Event
Jul 11, 2023 10:22:12 GMT+08:00	Started	Pod	prometheus-server-0	Started container init-config-reloader	Normal	1	Historical Event

10 Total Records: 83 1 2 3 4 5 ... 9

9.3.3 Diagnóstico de integridade

O CCE fornece diagnóstico de integridade com um clique em clusters, nós, cargas de trabalho, complementos principais e dependências externas para ajudá-lo a localizar rapidamente falhas de cluster (se houver). Esta seção descreve como realizar o diagnóstico de integridade em um cluster.

Pré-requisitos

- Você obteve **permissões de recurso**.
- A versão do cluster é posterior à v1.17.
- O cluster está em execução.

Caminho de navegação

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster.

Passo 2 Escolha **Monitoring Center** no painel de navegação e clique na guia **Health Diagnosis**.

Você pode realizar diagnósticos básicos sem ativar o Centro de monitoramento. Para experimentar mais serviços de diagnóstico, ative o Centro de monitoramento consultando **Ativar o Centro de monitoramento**.

----Fim

Configurar uma inspeção agendada

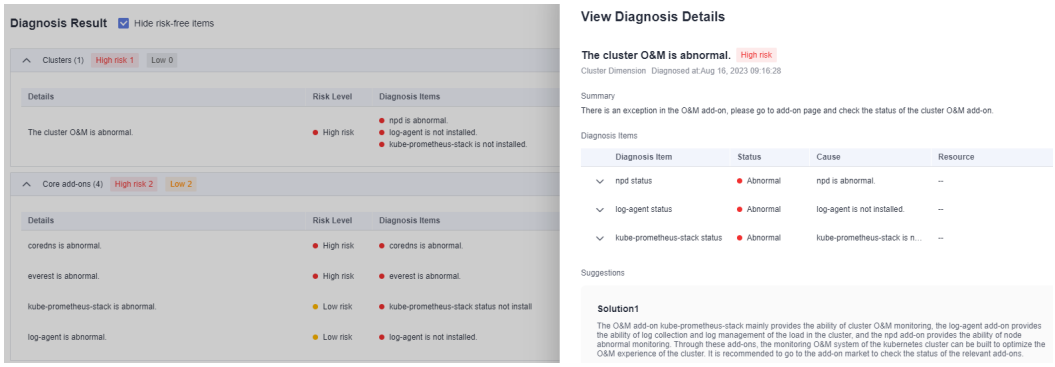
No canto superior direito da guia **Health Diagnosis**, ative **Scheduled Inspection** e configure a hora de início da inspeção. A tarefa de inspeção será iniciada automaticamente no horário especificado. Um cluster pode ser agendado para ser inspecionado apenas uma vez por dia.

Diagnóstico de integridade manual

Ao usar o diagnóstico de saúde pela primeira vez, clique em **Diagnose Now** para iniciar o diagnóstico. Depois que o diagnóstico for concluído, a página será atualizada automaticamente para exibir o resultado do diagnóstico. Os itens normais são ocultos por padrão no resultado.

Os problemas do Kubernetes serão resumidos a partir dos itens anormais. Sugestões de solução de problemas também serão fornecidas. Você pode clicar em **View Diagnosis Details** para exibir detalhes sobre um item de diagnóstico específico e recursos anormais relacionados. Em alguns casos, também há documentos de solução de problemas na página de detalhes do diagnóstico para sua referência.

Figura 9-78 Resultado do diagnóstico



Itens de inspeção

Dimensão	Cenário	Item de inspeção
Cluster	Planejamento de recursos do cluster	Se o HA está ativado para os nós principais
		Se as solicitações de CPU dos pods no cluster excederam 80% da CPU do cluster
		Se os limites de CPU dos pods no cluster excederam 150% da CPU do cluster
		Se as solicitações de memória dos pods no cluster excederam 80% da memória do cluster
		Se os limites de memória dos pods no cluster excederam 150% da memória do cluster
		Se a versão do cluster expirou
	O&M de cluster	Se kube-prometheus-stack é normal
		Se o log-agent é normal
		Se o npd é normal
	Configuração do cluster	Se os grupos de segurança estão configurados corretamente
Complementos principais	Status do coredns	Se o uso da CPU de coredns excedeu 80% nas últimas 24 horas
		Se o uso de memória de coredns excedeu 80% nas últimas 24 horas
		Se o coredns não conseguir resolver nomes de domínio nas últimas 24 horas
		Se a latência P99 de coredns excedeu 5s nas últimas 24 horas
		Se o coredns é normal
	Status do everest	Se o everest é normal
		Se o uso da CPU do everest excedeu 80% nas últimas 24 horas
		Se o uso de memória do everest excedeu 80% nas últimas 24 horas
	Estado de kube-prometheus-stack	Se o uso da CPU do kube-prometheus-stack excedeu 80% nas últimas 24 horas
		Se o uso de memória do kube-prometheus-stack excedeu 80% nas últimas 24 horas
		Se kube-prometheus-stack é normal

Dimensão	Cenário	Item de inspeção
		Se ocorreu falta de memória (OOM) no kube-prometheus-stack nas últimas 24 horas
		Se o uso de PVC do prometheus-server excedeu 80% quando kube-prometheus-stack é implementado no modo de servidor
	Status do log-agent	Se o log-agent é normal
		Se os grupos de log do LTS e o fluxo de log são criados com êxito
		Se a estruturação de log está ativada para grupos de log do LTS
	Status do autoscaler	Se o autoscaler está disponível quando o dimensionamento automático está ativado para pools de nós
Nó	Status do nó	Se os nós estão prontos
		Se os nós podem ser agendados
		Se o kubelet é normal
	Configuração do nó	Se as solicitações de memória de pods em um nó excederam 80% da memória do nó
		Se as solicitações de CPU de pods em um nó excederam 80% da CPU do nó
		Se os limites de memória dos pods em um nó excederam 150% da memória do nó
		Se os limites de CPU dos pods em um nó excederam 150% da CPU do nó
	Marcas d'água de recursos de nós	Se o uso da CPU de um nó excedeu 80% nas últimas 24 horas
		Se o uso de memória de um nó excedeu 80% nas últimas 24 horas
		Se o uso do disco de um nó excedeu 80%
		Se o número de PIDs para um nó excede o limite
		Se a OOM ocorreu em um nó nas últimas 24 horas
Carga de trabalho	Status do pod	Se os pods são normais
	Carga de trabalho do pod	Se a OOM ocorreu em um pod nas últimas 24 horas
		Se o uso da CPU de um pod excedeu 80% nas últimas 24 horas

Dimensão	Cenário	Item de inspeção
		Se o uso de memória de um pod excedeu 80% nas últimas 24 horas
	Configuração do pod	Se as solicitações são configuradas para contêineres em um pod
		Se os limites são configurados para contêineres em um pod
	Configuração da sonda de pod	Se as sondas de vivacidade estão configuradas para contêineres em um pod
		Se as sondas de prontidão estão configuradas para contêineres em um pod
Dependência externa	Cotas de recursos de um nó	Se 90% ou mais da cota de disco EVS foi utilizada
		Se 90% ou mais da cota do ECS foi utilizada

9.3.4 Painel

O painel exibe vários gráficos, como gráficos de linhas e dígitos, na mesma tela para apresentar dados de recursos, ajudando você a entender de forma abrangente os dados de monitoramento.

Verificar e alternar visualizações

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster.

Passo 2 Escolha **Monitoring Center** no painel de navegação e clique na guia **Dashboard**. A visualização de cluster é exibida por padrão.

Passo 3 Configurar parâmetros relacionados para verificação de visualizações. Os parâmetros disponíveis para configuração variam de acordo com as vistas. Veja **Tabela 9-10** para mais detalhes.

Passo 4 Especifique a janela de exibição.

Selecione ou personalize segmentos de tempo no canto superior direito da página e clique em  para atualizar a página.

Passo 5 O painel fornece visualizações predefinidas. Você pode clicar no botão **Switch View** ao lado do nome da exibição para selecionar os dados de monitoramento a serem exibidos. **Tabela 9-10** descreve as visualizações predefinidas.

Tabela 9-10 Visualizações predefinidas

Nome da visualização	Parâmetro	Métrica de nonitoramento incluída
Visualização de cluster (padrão)	Cluster	● Nodes ● Nodes with Unavailable Disks ● Unavailable Nodes ● CPU Usage ● CPU Request ● CPU Limit ● Memory Usage ● Memory Request ● Memory Limit ● Pods ● Containers ● Used CPUs ● Used Memory ● Network Receive Rate ● Network Transmit Rate ● Average Network Receive Rate ● Average Network Transmit Rate ● Packet Receive Rate ● Packet Transmit Rate ● Packet Loss Rate (Receive) ● Packet Loss Rate (Transmit) ● Disk I/O Rate (Read + Write) ● Disk Throughput (Read + Write)

Nome da visualização	Parâmetro	Métrica de nonitoramento incluída
Visualização do servidor da API	● Cluster ● Pod	● Alived ● QPS ● Request Success Rate (Read) ● Requests Being Processed ● Request Rate (Read) ● Request Error Rate (Read) ● P99 Request Latency (Read) ● Request Rate (Write) ● Request Error Rate (Write) ● P99 Request Latency (Write) ● Work Queue Growth Rate ● Work Queue Depth ● Work Queue Latency (P99) ● Used Memory ● Used CPUs ● Goroutines
Visualização do Pod	● Cluster ● Namespace ● Pod	● Containers ● Running Containers ● Pod Status ● Container Restarts ● Used CPUs ● CPU Throttling ● Used Memory ● Network Receive Rate ● Network Transmit Rate ● Packet Receive Rate ● Packet Transmit Rate ● Packet Loss Rate (Receive) ● Packet Loss Rate (Transmit) ● Pod Disk I/O Rate (Read + Write) ● Pod Disk Throughput (Read + Write) ● Container Disk I/O Rate (Read + Write) ● Container Disk Throughput (Read + Write) ● File System Usage ● Used File System Space

Nome da visualização	Parâmetro	Métrica de nonitoramento incluída
Visualização de host	● Cluster ● Node	● CPU Usage ● Load Average ● Used Memory ● Memory Usage ● Disk Write Rate ● Disk Read Rate ● Disk Space Usage ● Disk I/O ● TCP Connection ● UDP Usage ● Max. File Descriptor ● Used File Descriptors ● Socket Usage ● Abnormal File System ● Disk Rate ● I/O Latency ● I/O Queues ● Process Status
k8s-node	● Cluster ● Nó	● CPU Usage ● CPU Request ● CPU Limit ● Memory Usage ● Memory Request ● Memory Limit ● Used Memory ● Network Receive Rate ● Network Transmit Rate ● Network Receive Rate (Pod) ● Network Transmit Rate (Pod) ● Packet Receive Rate ● Packet Transmit Rate ● Packet Loss Rate (Receive) ● Packet Loss Rate (Transmit) ● Node Disk I/O Rate (Read + Write) ● Node Disk Throughput (Read + Write)

Nome da visualização	Parâmetro	Métrica de nonitoramento incluída
CoreDNS	● Cluster ● Pod	● Request Rate ● Request Rate (Record Type) ● Request Rate (Region) ● Request Rate (DO Flag) ● Request Packet (UDP) ● Request Packet (TCP) ● Response Rate (by rcode) ● Response Rate (duration) ● Response Packet (UDP) ● Response Packet (TCP) ● Cached Records ● Cache Hit Ratio
Visualização de PVC (somente clusters do CCE)	● Cluster ● Namespace ● PV ● PVC	● PV Status ● PVC Status ● Used PVC Capacity ● PVC Usage ● PVC Inodes ● PVC Inodes Usage ● PVC Capacity Used per Hour ● PVC Capacity Used per Day ● PVC Capacity Used per Week ● Used PVC Capacity After One Week

Nome da visualização	Parâmetro	Métrica de nonitoramento incluída
kubelet	● Cluster ● Pod	● Running kubelets ● Running Pods ● Running Containers ● Actual Volumes ● Expected Volumes ● Configuration Errors ● Operation Rate ● Operation Error Rate ● Operation Latency ● Pod Startup Rate ● Pod Startup Latency (P99) ● Storage Operation Rate ● Storage Operation Error Rate ● Storage Operation Latency (P99) ● Cgroup Manager Operation Rate ● Cgroup Manager Operation Latency (P99) ● PLEG Relist Rate ● PLEG Relist Interval ● PLEG Relist Latency (P99) ● RPC Rate ● Request Latency (P99) ● Used Memory ● Used CPUs ● Goroutines
Prometheus	● Cluster ● Tarefa ● Pod	● Target Synchronization ● Targets ● Average Scrape Interval ● Scrape Failures ● Sample Adding Rate ● Series in the Head ● Head Chunks ● Query Rate ● P90 Query Duration

Nome da visualização	Parâmetro	Métrica de nonitoramento incluída
Gravação remota de Prometheus	● Cluster ● Pod ● url	● Remote Sample Lag Ratio ● Remote Write Traffic ● Current Shards ● Max Shards ● Min Shards ● Desired Shards ● Shard Capacity ● Pending Samples ● Current TSDB Segment ● Current Segment of Remote Write ● Sample Discard Rate ● Sample Failure Rate ● Sample Retry Rate ● Retry Rate of Enqueuing
Visualização do pool de nós	● Cluster ● Pool de nós	● Node Pool CPU Allocation Rate ● Node CPU Allocation Rate ● Node Pool Memory Allocation Rate ● Node Memory Allocation Rate ● Node Count Trend

Nome da visualização	Parâmetro	Métrica de nonitoramento incluída
Visualização de XGPU	Cluster	• Cluster - XGPU Device GPU Memory Usage • Cluster - XGPU Device Computing Usage • Node - XGPU Device GPU Memory Usage • Node - XGPU Device Computing Usage • Node - Number of XGPU Devices • Node - Allocated XGPU Device GPU Memory • GPU - XGPU Device GPU Memory Usage • GPU - Allocated XGPU Device GPU Memory • GPU - XGPU Device GPU Memory Allocation Rate • GPU - XGPU Device Computing Usage • GPU - Number of XGPU Devices • GPU - Scheduling Policy • GPU - Number of Unhealthy XGPU Devices • Container - Allocated GPU Memory • Container - Computing Usage • Container - Used GPU Memory • Container - GPU Memory Usage

----Fim

9.4 Gerenciamento de alarmes

9.4.1 Alarm Assistant

Com base no Application Operations Management (AOM), o Alarm Assistant pode detectar prontamente falhas de cluster e gerar alarmes para ajudá-lo a manter a estabilidade do serviço. Alarm Assistant fornece regras de alarme incorporadas, que podem liberá-lo da configuração manual de regras de alarme no AOM. Essas regras são estabelecidas com base na extensa experiência de O&M de cluster de nossa equipe de contêineres da Huawei Cloud e podem cobrir exceções de serviços de contêineres, alarmes de métricas importantes de recursos básicos de cluster e alarmes de métrica de aplicações em um cluster para atender aos seus requisitos de O&M de rotina.

Restrições

- A versão do cluster deve ser v1.17 ou posterior.
- Somente contas da Huawei Cloud, HUAWEI IDs ou usuários do IAM com permissões CCE administrator ou FullAccess podem executar todas as operações usando Alarm Assistant. Os usuários do IAM com a permissão CCE ReadOnlyAccess só podem visualizar todos os recursos.

Obter permissões de recursos

Alarm Assistant trabalha em estreita colaboração com os serviços em nuvem para monitoramento de clusters, relatórios de alarmes e notificações. Quando você acessa o Alarm Assistant pela primeira vez, o Alarm Assistant e o AOM solicitam automaticamente permissões para acessar esses serviços de nuvem na região onde você executa suas aplicações.

Alarm Assistant obtém as seguintes permissões de serviço:

- Permissões do CCE

  Alarm Assistant precisa acessar o CCE para obter informações sobre clusters, nós e cargas de trabalho para verificações de integridade.
- Permissões do SWR

  Alarm Assistant precisa acessar o SWR para obter informações de imagem.
- Permissões somente leitura rm serviços de nuvem

  Alarm Assistant precisa acessar o CCE para obter informações sobre recursos de computação, rede e armazenamento para verificações de integridade.
- Permissões somente leitura no IAM

  Alarm Assistant precisa acessar o IAM para obter informações da agência.

Para obter detalhes sobre as permissões de serviço de nuvem solicitadas pelo AOM, consulte **Autorização de serviço de nuvem do AOM**.

Depois de concordar com a autorização, as agências são criadas automaticamente no IAM para delegar outras permissões de operação de recursos na sua conta para o CCE e AOM. Para obter detalhes sobre a agência, consulte **Delegação do serviço de muvem**. As agências criadas automaticamente no IAM são as seguintes:

- cia_admin_trust
- aom_admin_trust

A agência **cia_admin_trust** tem as permissões Tenant Guest e IAM ReadOnlyAccess em projetos globais, bem como as permissões Tenant Guest, CCE Administrator e SWR Administrator em projetos regionais. Essas permissões são exigidas pelo Alarm Assistant para acessar outros serviços em nuvem.

Para usar o Alarm Assistant em várias regiões, você precisa solicitar as permissões Tenant Guest, CCE Administrator e SWR Administrator em cada região. Você pode acessar o console do IAM, escolher **Agencies** e clicar em **cia_admin_trust** para exibir os registros de delegação de cada região.

NOTA

O Alarm Assistant pode não funcionar normalmente se as permissões necessárias não forem atribuídas. Ao usar o Alarm Assistant, não exclua ou modifique a agência **cia_admin_trust**.

Para obter detalhes sobre a agência **aom_admin_trust**, consulte **Autorização de serviço de nuvem do AOM**.

Habilitar o Alarm Assistant

O Alarm Assistant pode ser habilitado para clusters do CCE e do CCE Turbo.

Passo 1 Clique no nome do cluster de destino e escolha **Alarm assistant** no painel de navegação.

Passo 2 Na guia **Alarm Rules**, clique em **Enable**. Na janela que desliza para fora da direita, selecione um ou mais grupos de contatos para gerenciar pontos de extremidade de assinatura e receber mensagens de alarme por grupo. Se nenhum grupo de contatos estiver disponível, crie um fazendo referência a **Criar um grupo de contatos**.

Passo 3 Clique em **OK**.

NOTA

As regras de alarme de métrica no Alarm Assistant dependem do complemento kube-prometheus-stack para relatar métricas para uma instância de Prometheus do AOM. Se você não tiver instalado este complemento no cluster ou conectado o complemento instalado a uma instância de Prometheus do AOM, não será possível criar regras de alarme de métrica usando o Alarm Assistant. Para obter detalhes sobre o complemento, consulte **Monitoramento de cluster da nuvem nativa**.

----Fim

Configuração de regras de alarme

Depois que o Alarm Assistant estiver habilitado para clusters do CCE e do CCE Turbo, você poderá configurar e gerenciar regras de alarme.

Passo 1 Efetue logon no console do CCE.

Passo 2 Na página da lista de clusters, clique no nome do cluster de destino para ir para a página de detalhes.

Passo 3 No painel de navegação, escolha **Alarm assistant**. Na página exibida, clique na guia **Alarm Rules**, na qual você pode configurar e gerenciar regras de alarme.

Por padrão, o assistente de alarme gera regras de alarme para contêineres. As regras são destinadas a alarmes, incluindo alarmes de eventos e alarmes métricos para exceções. As regras de alarme são classificadas em vários conjuntos. Você pode associar um conjunto de regras de alarme a vários grupos de contatos e ativar ou desativar itens de alarme. Um conjunto de regras de alarme consiste em várias regras de alarme. Uma regra de alarme corresponde aos itens de verificação para uma única exceção. **Tabela 9-11** lista as regras de alarme padrão.

----Fim

Tabela 9-11 Regras de alarme padrão

Tipo de regra	Item de alarme	Descrição	Tipo de alarme	PromQL
Carregar conjunto de regras	Pod anormal	Verifique se o pod está funcionando corretamente.	Métrica	sum(min_over_time(kube_pod_status_phase{phase=~"Pending|Unknown|Failed"}[10m]) and count_over_time(kube_pod_status_phase{phase=~"Pending|Unknown|Failed"}[10m]) > 18)by (namespace,pod, phase, cluster_name, cluster) > 0
	Reinicializações frequentes de pods	Verifique se o pod é reiniciado com frequência.	Métrica	increase(kube_pod_container_status_restarts_total[5m]) > 3
	Número inesperado de réplicas de Implementação	Verifique se o número de réplicas de Implementação é o mesmo que o valor esperado.	Métrica	(kube_deployment_spec_replicas != kube_deployment_status_replicas_available) and (changes(kube_deployment_status_replicas_updated[5m]) == 0)
	Número inesperado de réplicas de StatefulSet	Verifique se o número de réplicas de StatefulSet é igual ao valor esperado.	Métrica	(kube_statefulset_status_replicas_ready != kube_statefulset_status_replicas) and (changes(kube_statefulset_status_replicas_updated[5m]) == 0)
	Uso de CPU de contêiner superior a 80%	Verifique se o uso da CPU do contêiner é maior que 80%.	Métrica	100 * (sum(rate(container_cpu_usage_seconds_total{image!="", container!="POD"}[1m])) by (cluster_name,pod,node,namespace,container, cluster) / sum(kube_pod_container_resource_limits{resource="cpu"}) by (cluster_name,pod,node,namespace,container, cluster)) > 80

Tipo de regra	Item de alarme	Descrição	Tipo de alarme	PromQL
	Uso de memória de contêiner superior a 80%	Verifique se o uso de memória do contêiner é maior que 80%.	Métrica	(sum(container_memory_working_set_bytes{image!="", container!="POD"}) BY (cluster_name, node,container, pod , namespace, cluster) / sum(container_spec_memory_limit_bytes > 0) BY (cluster_name, node, container, pod , namespace, cluster) * 100) > 80
	Contêiner anormal	Verifique se o contêiner está funcionando corretamente.	Métrica	sum by (namespace, pod, container, cluster_name, cluster) (kube_pod_container_status_waiting_reason) > 0
	Falhou ao atualizar um balanceador de carga	Verifique se um balanceador de carga está atualizado.	Evento	N/D
Conjunto de regras de recursos de nó	Alta utilização do PV do Kubernetes	Verifique se o uso de PV em um nó é muito alto.	Métrica	(kubelet_volume_stats_available_bytes{job="kubelet"} / kubelet_volume_stats_capacity_bytes{job="kubelet"}) < 0.03 and kubelet_volume_stats_used_bytes{job="kubelet"} > 0
	PVC de Kubernetes anormal	Verifique se a PVC está funcionando corretamente.	Métrica	kube_persistentvolume-claim_status_phase{phase=~"Failed|Pending|Lost"} > 0
	PV de Kubernetes anormal	Verifique se o PV está funcionando corretamente.	Métrica	kube_persistentvolume_status_phase{phase=~"Failed|Pending"} > 0
	Uso da CPU do nó superior a 80%	Verifique se o uso da CPU do nó é superior a 80%.	Métrica	100 - (avg by(node, cluster_name, cluster) (rate(node_cpu_seconds_total{mode="idle"}[2m])) * 100) > 80
	Memória de nó disponível inferior a 10%	Verifique se a memória de nó disponível é inferior a 10%.	Métrica	node_memory_MemAvailable_bytes / node_memory_MemTotal_bytes * 100 < 10

Tipo de regra	Item de alarme	Descrição	Tipo de alarme	PromQL
	Espaço em disco disponível no nó inferior a 10%	Verifique se o espaço em disco disponível do nó é inferior a 10%.	Métrica	avg((node_filesystem_avail_bytes * 100) / node_filesystem_size_bytes) by (device, node, cluster_name, cluster) < 10
	Espaço em disco de nó insuficiente	Verifique se o espaço em disco do nó é suficiente.	Evento	N/D
	Erro do pool de armazenamento emptyDir	Verifique se o pool de armazenamento de EV do nó está funcional.	Métrica	problem_gauge{type="EmptyDirVolumeGroupStatusError"} >= 1
	Memória de nó insuficiente	Verifique se a memória geral do nó é suficiente.	Métrica	problem_gauge{type="MemoryProblem"} >= 1
	Erro do pool de armazenamento de PV	Verifique se o pool de armazenamento de PV do nó está funcional.	Métrica	problem_gauge{type="LocalPvVolumeGroupStatusError"} >= 1
	Ponto de montagem do nó anormal	Verifique se o ponto de montagem do nó está disponível.	Métrica	problem_gauge{type="MountPointProblem"} >= 1
	Identificadores de arquivo de nó insuficientes	Verifique se os identificadores do arquivo FD são suficientes.	Métrica	problem_gauge{type="FDProblem"} >= 1
	Suspensão de I/O de nó de disco	Verifique se a suspensão de I/O ocorre no disco do nó.	Métrica	problem_gauge{type="DiskHung"} >= 1
	Disco de nó de somente leitura	Verifique se o disco do nó é de somente leitura.	Métrica	problem_gauge{type="DiskReadonly"} >= 1

Tipo de regra	Item de alarme	Descrição	Tipo de alarme	PromQL
	Disco de nó anormal	Verifique o uso do disco do sistema do nó e dos discos de dados do CCE (incluindo discos lógicos de Docker e kubelet).	Métrica	problem_gauge{type="DiskProblem"} >= 1
	I/O de disco de nó lenta	Verifique se ocorre I/O lenta no disco de nó.	Métrica	problem_gauge{type="DiskSlow"} >= 1
	PIDs de nó insuficientes	Verifique se os PIDs são suficientes.	Métrica	problem_gauge{type="PIDProblem"} >= 1
	Tabela de nó conntrack cheia	Verifique se o espaço da tabela conntrack do nó é suficiente.	Métrica	problem_gauge{type="ConntrackFullProblem"} >= 1
Conjunto de regras de status do nó	Erro de ResolvConf	Verifique se o arquivo de configuração ResolvConf está disponível.	Métrica	problem_gauge{type="ResolvConfFileProblem"} >= 1
	Componente CNI de nó anormal	Verifique se o componente CNI do nó está sendo executado corretamente.	Métrica	problem_gauge{type="CNIProblem"} >= 1
	Componente CRI de nó anormal	Verifique a execução do componente importante CRI (Docker ou containerd).	Métrica	problem_gauge{type="CRIProblem"} >= 1
	Erro do nó kube-proxy	Verifique se o kube-proxy está funcionando corretamente.	Métrica	problem_gauge{type="KUBEPROXYProblem"} >= 1
	Nó de kubelet anormal	Verifique se o kubelet está funcionando corretamente.	Métrica	problem_gauge{type="KUBELETProblem"} >= 1

Tipo de regra	Item de alarme	Descrição	Tipo de alarme	PromQL
	Evento programado no nó	Verifique se há um evento agendado do host no nó.	Métrica	problem_gauge{type="ScheduledEvent"} >= 1
	Status do nó instável	Verifique se o status do nó alterna entre normal e anormal.	Métrica	sum(changes(kube_node_status_condition{status="true",condition="Ready"}[15m])) by (cluster_name, node, cluster) > 2
	Reinicializações frequentes do nó containerd	Verifique se o containerd é reiniciado com frequência.	Métrica	problem_gauge{type="FrequentContainerdRestart"} >= 1
	Tarefa do nó suspensa	Verifique se uma tarefa está suspensa no nó.	Evento	N/D
	Configuração incorreta do pool de armazenamento de nó	Verifique se os pools de armazenamento de EV e PV do nó estão configurados corretamente.	Evento	N/D
	Nó anormal	Verifique se o nó está funcionando corretamente.	Evento	N/D
	Processo de nó D anormal	Verifique se há um processo de estado D no nó.	Métrica	problem_gauge{type="ProcessD"} >= 1
	Processo do nó Z anormal	Verifique se há um processo de estado Z no nó.	Métrica	problem_gauge{type="ProcessZ"} >= 1
	Reinicializações frequentes da CRI do nó	Verifique se a CRI é reiniciada com frequência.	Métrica	problem_gauge{type="FrequentCRIRestart"} >= 1
	Reinicializações frequentes do nó Docker	Verifique se o Docker é reiniciado com frequência.	Métrica	problem_gauge{type="FrequentDockerRestart"} >= 1

Tipo de regra	Item de alarme	Descrição	Tipo de alarme	PromQL
	Reinicializações frequentes do nó kubelet	Verifique se o kubelet é reiniciado com frequência.	Métrica	problem_gauge{type="FrequentKubeletRestart"} >= 1
	Erro do serviço NTP do nó	Verifique se o serviço de sincronização de relógio de nó ntpd ou chronyd está sendo executado corretamente.	Métrica	problem_gauge{type="NTPProblem"} >= 1
	Processos interrompidos forçadamente devido à OOM do nó	Verifique se ocorreu um evento OOM no nó.	Evento	N/D
Conjunto de regras de dimensionamento de nó	Pool de nó esgotado	Verifique se os recursos do pool de nó são suficientes.	Evento	N/D
	Expansão expirou o tempo limite	Verifique se a expansão de nós ao pool de nós expirou o tempo limite.	Evento	N/D
	Falhou na expansão do pool de nós	Verifique se ocorreu um erro durante uma expansão do pool de nós.	Evento	N/D
	Falhou na redução do pool de nós	Verifique se ocorreu um erro durante uma redução do pool de nós.	Evento	N/D
Conjunto de regras de status do cluster	Cluster indisponível	Verifique se o cluster está disponível.	Evento	N/D

Criar um grupo de contatos

Um grupo de contatos, apoiado em **Simple Message Notification**, permite que editores de mensagens e assinantes entrem em contato. Um grupo de contatos contém um ou mais pontos de extremidade. Você pode configurar grupos de contatos para gerenciar pontos finais que assinaram mensagens de alarme. Depois de criar um grupo de contatos, associe o conjunto de regras de alarme ao grupo. Quando um alarme é acionado, os pontos de extremidade de assinatura no grupo de contatos podem receber as mensagens de alarme.

Passo 1 Efetue logon no console do CCE.

Passo 2 Na página da lista de clusters, clique no nome do cluster de destino para ir para a página de detalhes.

Passo 3 Escolha **Alarm assistant** no painel de navegação e clique na guia **Contact Group**.

Passo 4 Clique em **Create Contact Group** e configure parâmetros.

- **Contact Group Name**: digite o nome do grupo de contatos, que não pode ser alterado após a criação do grupo de contatos. O nome do modelo pode conter de 1 a 255 caracteres e deve começar com uma letra ou dígito. Somente letras, dígitos, hifens (-) e sublinhados (_) são permitidos.
- **Alarm message display name**: digite o título da mensagem recebida pelo ponto de extremidade de assinatura especificado. Por exemplo, se você definir **Terminal Type** para **Email** e especificar um nome de exibição, o nome que você especificou será exibido como o remetente da mensagem de alarme. Se você não especificar **Alarm message display name**, o remetente será **username@example.com**. O nome de exibição de uma mensagem de alarme pode ser alterado após a criação do grupo de contatos.
- **Add Subscription Terminal**: adicione um ou mais pontos de extremidade para receber mensagens de alarme. O tipo de ponto de extremidade pode ser **SMS** ou **Email**. Se você selecionar **SMS**, insira um número de celular válido. Se selecionar **Email**, introduza um endereço de e-mail válido.

Passo 5 Clique em **OK**.

Você será redirecionado para a lista de grupos de contatos. O ponto de extremidade da assinatura está no estado **Unconfirmed**. Envie uma solicitação de assinatura ao ponto de extremidade para verificar a validade do ponto de extremidade.

Passo 6 Clique em **Request Confirmation** na coluna **Operation** para enviar uma solicitação de assinatura ao ponto de extremidade. Se o ponto de extremidade receber a solicitação, confirme a solicitação conforme solicitado. Após a conclusão da confirmação, o ponto de extremidade da assinatura muda para **Confirmed**.

Passo 7 Clique em  para habilitar o grupo de contatos para que o grupo de contatos seja vinculado ao conjunto de regras de alarme.

NOTA

Um conjunto de regras de alarme pode ser vinculado a um máximo de cinco grupos de contatos.

----Fim

Visualizar alarmes

Você pode ver os últimos alarmes históricos na guia **Alarm list**.

Passo 1 Efetue logon no console do CCE.

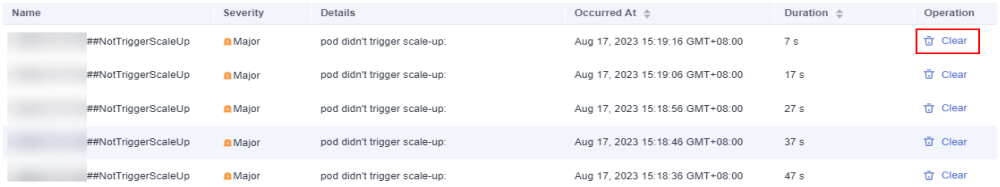
Passo 2 Na página da lista de clusters, clique no nome do cluster de destino para ir para a página de detalhes.

Passo 3 No painel de navegação, escolha **Alarm assistant**. Na página exibida, clique na guia **Alarm list**.

Por padrão, todos os alarmes a serem apagados são exibidos na lista. Você pode consultar alarmes por palavra-chave do alarme, gravidade do alarme ou hora do alarme. Além disso, você pode visualizar a distribuição de alarmes que atendem aos critérios especificados em diferentes períodos.

Se você confirmar que um alarme foi manipulado, clique em **Clear** na coluna **Operation**. Depois que o alarme for apagado, você poderá visualizá-lo na lista de alarmes históricos.

Figura 9-79 Consultar alarmes



----Fim

9.4.2 Configurações de alarme personalizadas

O CCE interage com o Application Operations Management (AOM) para relatar alarmes e eventos. Ao definir regras de alarme no AOM, você pode verificar se os recursos nos clusters estão normais em tempo hábil.

Processo

1. **Criar um tópico no SMN**
2. **Criar uma política de ação**
3. Adicionar uma regra de alarme
   a. Alarmes de evento: gerar alarmes com base nos eventos relatados pelos clusters para o AOM. Para obter detalhes sobre os eventos e configurações, consulte **Adicionar alarmes de evento**.
   b. Alarmes de limite: gerar alarmes com base nos limites de métricas de monitoramento, como utilização de recursos de servidores e componentes. Para obter detalhes sobre os limites e configurações de métricas, consulte **Adicionar alarmes de limite**.

Criar um tópico no SMN

Simple Message Notification (SMN) envia mensagens para assinantes através de e-mails, mensagens SMS e solicitações HTTP/HTTPS.

Um tópico é usado para publicar mensagens e assinar notificações. Ele serve como um canal de transmissão de mensagens entre editores e assinantes.

Você precisa criar um tópico e assiná-lo. Para obter detalhes, consulte **Criação de um tópico** e **Assinatura de um tópico**.

NOTA

Depois de se inscrever em um tópico, confirme a assinatura no e-mail ou mensagem SMS para que a notificação entre em vigor.

Criar uma política de ação

O AOM permite que você personalize políticas de ação de alarme. Você pode criar uma política de ação de alarme para associar um tópico SMN e um modelo de mensagem. Você também pode personalizar o conteúdo da notificação usando um modelo de mensagem.

Para obter detalhes, consulte **Criação de políticas de ação de alarme**. Ao criar uma política de ação, selecione o tópico criado e inscrito em **Criar um tópico no SMN**.

Adicionar alarmes de evento

O seguinte usa o alarme **NodeNotReady** como um exemplo para descrever como adicionar um alarme de evento.

Esta função é fornecida pelo AOM. Para obter detalhes sobre os parâmetros, consulte **Criação de regras de alarme de evento**.

Tabela 9-12 Alarmes baseados em eventos

Nome do evento	Fonte	Descrição	Solução
NodeNotReady	CCE	Um alarme é acionado imediatamente quando um nó é anormal.	Efetue logon no cluster e verifique o status do nó para o qual o alarme é gerado. Defina o nó como não programável e programe os pods de serviço para outro nó.
Rebooted	CCE	Um alarme é disparado imediatamente quando um nó é reiniciado.	Efetue logon no cluster para verificar o status do nó para o qual o alarme é gerado, verificar se o nó pode ser iniciado corretamente e localizar a causa da reinicialização.
KUBELETIsDown	CCE	Um alarme é acionado imediatamente quando um nó é anormal.	Efetue logon no cluster e verifique o status do nó para o qual o alarme é gerado. Defina o nó como não programável e programe os pods de serviço para outro nó. Em seguida, reinicie o kubelet.
DOCKERIsDown	CCE	Um alarme é acionado imediatamente quando um nó é anormal.	Efetue logon no cluster e verifique o status do nó para o qual o alarme é gerado. Defina o nó como não programável e programe os pods de serviço para outro nó. Em seguida, reinicie o Docker.

Nome do evento	Fonte	Descrição	Solução
KUBEPROXYIsDown	CCE	Um alarme é acionado imediatamente quando um nó é anormal.	Efetue logon no cluster e verifique o status do nó para o qual o alarme é gerado. Defina o nó como não programável e programe os pods de serviço para outro nó.
KernelOops	CCE	Um alarme é acionado imediatamente quando um nó é anormal.	Efetue logon no cluster e verifique o status do nó para o qual o alarme é gerado. Defina o nó como não programável e programe os pods de serviço para outro nó.
ConntrackFull	CCE	Um alarme é acionado imediatamente quando um nó é anormal.	Efetue logon no cluster e verifique o status do nó para o qual o alarme é gerado. Defina o nó como não programável e programe os pods de serviço para outro nó.
NodePoolSoldOut	CCE	Um alarme é acionado imediatamente quando os recursos do pool de nós estão esgotados.	Defina a alternância automática do pool de nós ou altere as especificações do pool de nós.
NodeCreateFailed	CCE	Um alarme é disparado imediatamente após uma falha de criação de nó.	Corrija a falha e crie o nó novamente.
ScaleUpTimedOut	CCE	Um alarme é disparado imediatamente após o tempo limite da expansão do nó.	Corrija a falha e tente expandir novamente.
ScaleDownFailed	CCE	Um alarme é disparado imediatamente após o tempo limite da redução do nó.	Corrija a falha e tente reduzir novamente.
BackOffPullImage	CCE	Falha na tentativa de extração da imagem.	Efetue logon no cluster, localize a causa da falha e implemente a carga de trabalho do serviço novamente.

Passo 1 Efetue logon no console do AOM.

Passo 2 No painel de navegação, escolha **Alarm Center** > **Alarm Rules** e clique em **Add Alarm**.

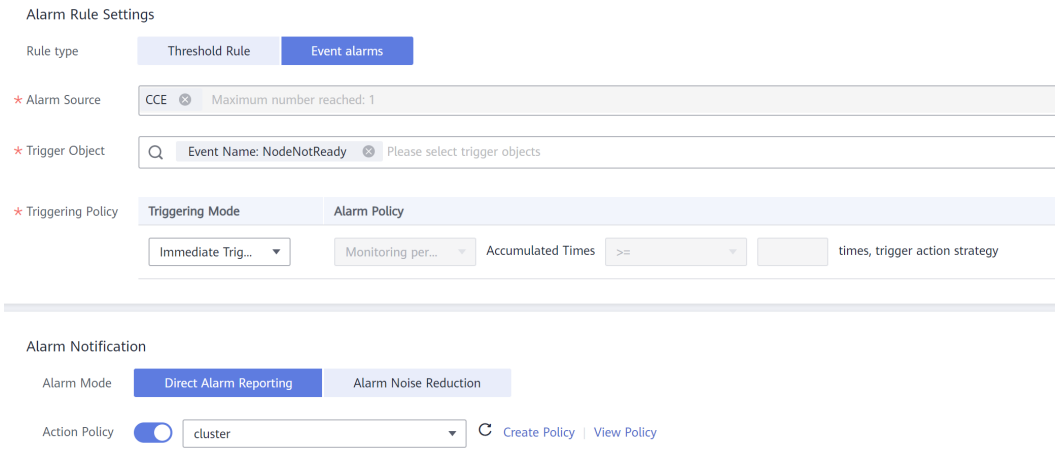
Passo 3 Defina uma regra de alarme.

- **Rule Type**: selecione **Event alarm**.
- **Alarm Source**: selecione **CCE**.
- **Select Object**: selecione **Event Name** e, em seguida, **NodeNotReady**. Você pode filtrar objetos de gatilho por tipo de notificação, nome do evento, gravidade do alarme, atributo personalizado, namespace e nome do cluster.
- **Triggering Policy**: selecione **Immediate Triggering**.
- **Alarm Mode**: selecione **Direct Alarm Reporting**.
- **Action Policy**: selecione a política de ação criada em**Criar uma política de ação**.

Esta regra de alarme funciona da seguinte forma:

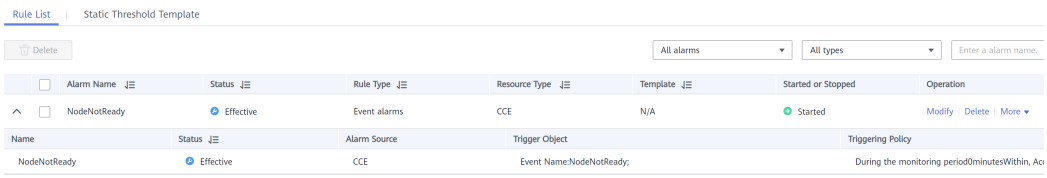
Se um nó no cluster se tornar anormal, o CCE reporta o evento **NodeNotReady** para o AOM. O AOM imediatamente o notifica através do SMN com base na política de ação.

Figura 9-80 Criar um alarme de evento



Passo 4 Clique em **Create Now**.

Se as seguintes informações forem exibidas na lista de regras, a regra será criada com êxito.

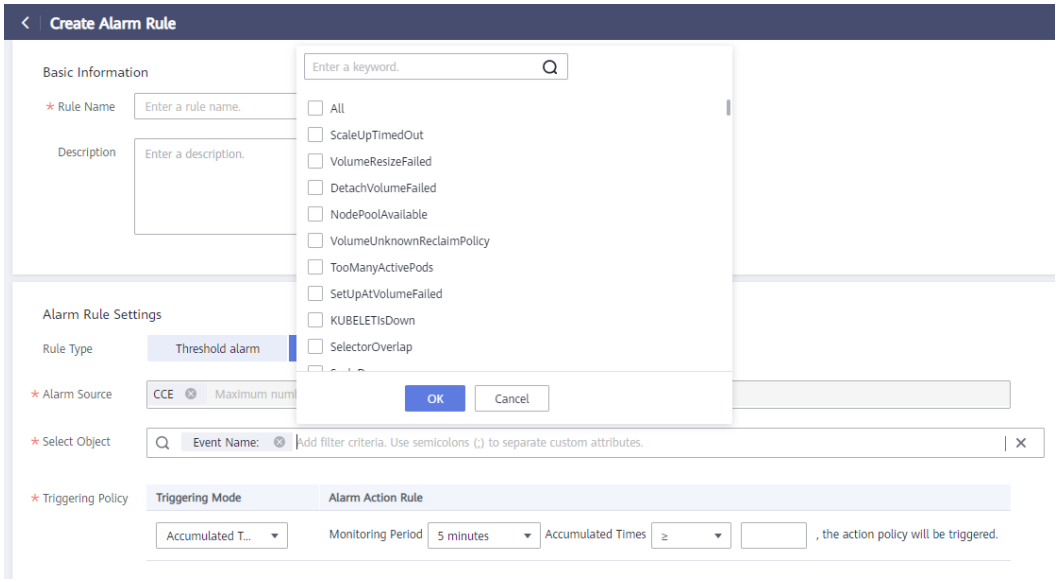


----Fim

Eventos do CCE

Os alarmes de eventos são gerados com base nos eventos relatados pelo CCE para o AOM. CCE relata uma série de eventos para AOM. Você pode exibir eventos específicos nas áreas **Alarm Rule Settings** e adicionar alarmes de evento conforme necessário.

Figura 9-81 Eventos reportados pelo CCE



A tabela a seguir lista os eventos do CCE.

Nome do evento	ID do evento	Gravidade	Descrição
ScaleUpTimedOut	ScaleUpTimedOut	Importante	Verifique se a adição de nós ao pool de nós expirou o tempo limite.
VolumeResizeFailed	VolumeResizeFailed	Secundária	Verifique se a capacidade do volume de dados é expandida.
DetachVolumeFailed	DetachVolumeFailed	Secundária	Verifique se o armazenamento em bloco está desanexado.
NodePoolAvailable	NodePoolAvailable	Importante	Verifique se os recursos do pool de nós são suficientes.
VolumeUnknownReclaimPolicy	VolumeUnknownReclaimPolicy	Secundária	Verifique se uma política de recuperação de volume foi especificada.
TooManyActivePods	TooManyActivePods	Secundária	Verifique se ainda existem pods ativos após o número de pods em uma tarefa atingir o valor predefinido.
SetUpAtVolumeFailed	SetUpAtVolumeFailed	Secundária	Verifique se o volume de dados está montado.
KUBELETIsDown	KUBELETIsDown	Secundária	Verifique o status do kubelet no nó.
SelectorOverlap	SelectorOverlap	Secundária	Verifique se seletores de rótulo estão no conflito de cluster.

Nome do evento	ID do evento	Gravidade	Descrição
ScaleDown	ScaleDown	Importante	Os nós estão sendo excluídos do cluster.
NodeHasInsufficientMemory	NodeHasInsufficientMemory	Secundária	Verifique se a memória disponível do nó é suficiente.
ClaimLost	ClaimLost	Secundária	Verifique se o volume de PVC está funcionando corretamente.
UnregisterNetDevice	UnregisterNetDevice	Secundária	Verifique se o nó está associado a qualquer dispositivo de rede não registrado.
VolumeFailedRecycle	VolumeFailedRecycle	Secundária	Verifique se o volume de dados é recuperado.
NotTriggerScale-Up	NotTriggerScale-Up	Importante	Verifique se uma expansão de nó é acionada.
DeleteUnregistered	DeleteUnregistered	Importante	Verifique se os nós não registrados são excluídos.
Unhealthy	Unhealthy	Secundária	Verifique se o pod está funcionando corretamente.
FailedDelete	FailedDelete	Secundária	Verifique se a carga de trabalho foi excluída.
NetworkCardNotFound	NetworkCardNotFound	Secundária	Verifique o status da ENI do nó.
TooManySucceedPods	TooManySucceedPods	Secundária	Verifique se há pods extras em execução após o número de pods em uma tarefa atingir o valor predefinido.
ScaleDownEmpty	ScaleDownEmpty	Importante	Verifique se os nós ociosos são reduzidos.
ErrImageNeverPull	ErrImageNeverPull	Secundária	Verifique se a carga de trabalho extraiu uma imagem.
Rebooted	Rebooted	Importante	Verifique se o nó é reiniciado.
KUBEPROXYIsDown	KUBEPROXYIsDown	Secundária	Verifique se o kube-proxy está sendo executado corretamente no nó.
FailedScaleOut	FailedScaleOut	Secundária	Verifique se os nós estão expandidos corretamente.
NodeOutOfDisk	NodeOutOfDisk	Secundária	Verifique se o espaço em disco do nó é suficiente.

Nome do evento	ID do evento	Gravidade	Descrição
TaskHung	TaskHung	Secundária	Verifique se há alguma tarefa suspensa no nó.
WaitForAttachVolumeFailed	WaitForAttachVolumeFailed	Secundária	Verifique se o armazenamento em bloco está anexado ao nó.
FailedStart	FailedStart	Importante	Verifique se o pod foi iniciado.
FailedPullImage	FailedPullImage	Importante	Verifique se o pod extraiu uma imagem.
DeleteNodeWithNoServer	DeleteNodeWithNoServer	Secundária	Verifique se os nós descartados estão limpos.
ReplicaSetCreateError	ReplicaSetCreateError	Secundária	Verifique se um ReplicaSet de carga de trabalho pode ser criado.
CIDRNotAvailable	CIDRNotAvailable	Secundária	Verifique se o bloco CIDR do nó está disponível.
ConntrackFull	ConntrackFull	Secundária	Verifique se a tabela conntrack do nó está cheia.
NodeHasDiskPressure	NodeHasDiskPressure	Secundária	Verifique se o espaço em disco do nó é suficiente.
FailedStandBy	FailedStandBy	Secundária	Verifique se o pod entra no estado de espera.
ScaleDownFailed	ScaleDownFailed	Importante	Verifique se os nós são reduzidos.
NodeNotSchedulable	NodeNotSchedulable	Importante	Verifique se o nó é agendável.
FailedToScaleUpGroup	FailedToScaleUpGroup	Importante	Verifique se ocorreu um erro durante uma expansão do pool de nós.
FailedReconfig	FailedReconfig	Secundária	Verifique se a configuração do pod está atualizada.
ScaledUpGroup	ScaledUpGroup	Importante	Verifique se o pool de nós foi expandido.
NodeInstallFailed	NodeInstallFailed	Secundária	Verifique se os nós são gerenciados no cluster.
CreatingLoadBalancerFailed	CreatingLoadBalancerFailed	Secundária	Verifique se um balanceador de carga foi criado.
FailedGet	FailedGet	Secundária	Verifique se CronJobs pode ser obtidas.

Nome do evento	ID do evento	Gravidade	Descrição
VolumeFailedDelete	VolumeFailedDelete	Secundária	Verifique se o volume de dados foi excluído.
KernelOops	KernelOops	Secundária	Verifique se o kernel do sistema operacional do nó está com defeito.
ScaleUpFailed	ScaleUpFailed	Importante	Verifique se o nó está expandido.
MountDeviceFailed	MountDeviceFailed	Secundária	Verifique se o volume de dados está montado.
DeletingLoadBalancerFailed	DeletingLoadBalancerFailed	Secundária	Verifique se o balanceador de carga foi excluído.
FixNodeGroupSizeDone	FixNodeGroupSizeDone	Importante	Verifique se o número de nós no pool de nós é restaurado.
TearDownAtVolumeFailed	TearDownAtVolumeFailed	Secundária	Verifique se o volume de dados está desmontado.
FailedActive	FailedActive	Secundária	Verifique se o pod está ativado.
OOMKilling	OOMKilling	Secundária	Verifique se a OOM ocorre no nó.
UnmountDeviceFailed	UnmountDeviceFailed	Secundária	Verifique se a letra da unidade do volume de dados está desmontada.
DOCKERIsDown	DOCKERIsDown	Secundária	Verifique se o mecanismo de contêiner do nó está funcionando corretamente.
FailedRollback	FailedRollback	Secundária	Verifique se o pod está revertido.
CIDRAssignmentFailed	CIDRAssignmentFailed	Secundária	Verifique se um bloco CIDR é alocado para o nó.
DockerHung	DockerHung	Secundária	Verifique se o processo Docker no nó está suspenso.
SelectingAll	SelectingAll	Secundária	Verifique se o seletor de rótulo da carga de trabalho está configurado corretamente.
NodeNotReady	NodeNotReady	Importante	Verifique se o nó está funcionando corretamente.
ProvisioningFailed	ProvisioningFailed	Secundária	Verifique se o volume de dados foi criado.
ProvisioningCleanupFailed	ProvisioningCleanupFailed	Secundária	Verifique se o volume de dados foi limpo.

Nome do evento	ID do evento	Gravidade	Descrição
NodeGroupInBackOff	NodeGroupInBackOff	Importante	Verifique se há tentativas de reversão durante o dimensionamento do pool de nós.
BackOffStart	BackOffStart	Importante	Verifique se o pod falha ao ser reiniciado.
DeploymentRollbackRevisionNotFound	DeploymentRollbackRevisionNotFound	Secundária	Verifique se a versão de reversão de Implementação está disponível.
FailedScheduling	FailedScheduling	Importante	Verifique se o pod está agendado.
FixNodeGroupSizeError	FixNodeGroupSizeError	Importante	Verifique se o número de nós no pool de nós é restaurado.
FilesystemIsReadOnly	FilesystemIsReadOnly	Secundária	Verifique se o sistema de arquivos do nó é somente leitura.
FailedUpdate	FailedUpdate	Secundária	Verifique se o pod está atualizado.
NTPIsDown	NTPIsDown	Secundária	Verifique se NTP está sendo executado corretamente no nó.
NodeCreateFailed	NodeCreateFailed	Importante	Verifique se o nó foi criado.
BackOffPullImage	BackOffPullImage	Importante	Verifique se o pod extraiu uma imagem após uma nova tentativa.
NodeUninstallFailed	NodeUninstallFailed	Secundária	Verifique se o nó está desinstalado.
ClaimMisbound	ClaimMisbound	Secundária	Verifique se a PVC está vinculada a um volume incorreto.
FailedList	FailedList	Secundária	Verifique se os pods podem ser obtidos.
NodePoolSoldOut	NodePoolSoldOut	Importante	Verifique se os recursos do pool de nós são suficientes.
AUFSUmountHung	AUFSUmountHung	Secundária	Verifique se a desanexação do disco de nó está suspensa.
FailedCreate	FailedCreate	Importante	Verifique se um pod foi criado.
UpdateLoadBalancerFailed	UpdateLoadBalancerFailed	Secundária	Verifique se o balanceador de carga está atualizado.
UnexpectedJob	UnexpectedJob	Secundária	Verifique se há alguma CronJob desconhecida.

Nome do evento	ID do evento	Gravidade	Descrição
FailedScaleIn	FailedScaleIn	Secundária	Verifique se uma redução de pod falhou.
TriggeredScaleUp	TriggeredScaleUp	Importante	Verifique se uma expansão de nó é acionada.
AttachVolumeFailed	AttachVolumeFailed	Secundária	Verifique se o armazenamento em bloco está separado do nó.
FailedRestart	FailedRestart	Secundária	Verifique se o pod foi reiniciado.
CNIIsDown	CNIIsDown	Secundária	Verifique se o complemento CNI no nó está com defeito.
StartScaledUpGroup	StartScaledUpGroup	Importante	Verifique se um pool de nós expandido foi iniciado.
DeleteUnregisteredFailed	DeleteUnregisteredFailed	Importante	Verifique se os nós não registrados são excluídos.
Internal error	Internal error	Importante	Verifique se ocorre um erro interno no cluster.
External dependency error	External dependency error	Importante	Verifique se ocorre um erro nas dependências externas do cluster.
Failed to initialize process thread	Failed to initialize process thread	Importante	Verifique se um thread de inicialização de cluster é executado.
Failed to update database	Failed to update database	Importante	Verifique se o banco de dados do cluster está atualizado.
Failed to create node by node pool	Failed to create node by nodepool	Importante	Verifique se os nós são criados no pool de nós.
Failed to delete node by node pool	Failed to delete node by nodepool	Importante	Verifique se os nós são excluídos do pool de nós.
Failed to create yearly/monthly subscription node	Failed to create yearly/monthly subscription node	Importante	Verifique se o nó anual/mensal é criado no cluster.
Failed to cancel the authorization of accessing the image of the master	Failed to cancel the authorization of accessing the image of the master.	Importante	Ao criar um cluster, verifique se a autorização para o locatário do recurso acessar a imagem do nó principal é cancelada.

Nome do evento	ID do evento	Gravidade	Descrição
Failed to create the virtual IP for the master	Failed to create the virtual IP for the master	Important e	Ao criar um cluster, verifique se o endereço IP virtual está alocado.
Failed to delete the node VM	Failed to delete the node VM	Important e	Verifique se o nó (VM) é excluído do cluster.
Failed to delete the security group of node	Failed to delete the security group of node	Important e	Verifique se o grupo de segurança do nó é excluído do cluster.
Failed to delete the security group of master	Failed to delete the security group of master	Important e	Verifique se o grupo de segurança do nó principal é excluído do cluster.
Failed to delete the security group of port	Failed to delete the security group of port	Important e	Verifique se o grupo de segurança da ENI do nó principal é excluído do cluster.
Failed to delete the security group of eni or subeni	Failed to delete the security group of eni or subeni	Important e	Verifique se o grupo de segurança da ENI ou sub-ENI é excluído do cluster.
Failed to detach the port of master	Failed to detach the port of master	Important e	Verifique se a ENI do nó principal está desacoplada do cluster.
Failed to delete the port of master	Failed to delete the port of master	Important e	Verifique se a ENI do nó principal é excluída do cluster.
Failed to delete the master VM	Failed to delete the master VM	Important e	Verifique se o nó principal (VM) é excluído do cluster.
Failed to delete the key pair of master	Failed to delete the key pair of master	Important e	Verifique se o par de chaves do nó principal é excluído do cluster.
Failed to delete the subnet of master	Failed to delete the subnet of master	Important e	Verifique se a sub-rede do nó principal é excluída do cluster.
Failed to delete the VPC of master	Failed to delete the VPC of master	Important e	Verifique se a VPC do nó principal é excluída do cluster.
Failed to delete certificate of cluster	Failed to delete certificate of cluster	Important e	Verifique se o certificado foi excluído do cluster.

Nome do evento	ID do evento	Gravidade	Descrição
Failed to delete the server group of master	Failed to delete the server group of master	Important e	Verifique se o nó principal (ECS) é excluído do cluster.
Failed to delete the virtual IP for the master	Failed to delete the virtual IP for the master	Important e	Verifique se o endereço IP virtual é excluído do cluster.
Failed to get floating IP of the master	Failed to get floating IP of the master	Important e	Verifique se o endereço IP flutuante do nó principal é obtido.
Failed to get cluster flavor	Failed to get cluster flavor	Important e	Verifique se o flavor de cluster é obtido.
Failed to get cluster endpoint	Failed to get cluster endpoint	Important e	Verifique se o ponto de extremidade do cluster foi obtido.
Failed to get Kubernetes connection	Failed to get Kubernetes connection	Important e	Verifique se as conexões de cluster do Kubernetes foram obtidas.
Failed to update secret	Failed to update secret	Important e	Verifique se o segredo do cluster está atualizado.
Operation timed out	Operation timed out	Important e	Verifique se a operação do usuário expirou.
Connecting to Kubernetes cluster timed out	Connecting to Kubernetes cluster timed out	Important e	Verifique se o tempo limite de acesso ao cluster do Kubernetes expirou.
Failed to check component status or components are abnormal	Failed to check component status or components are abnormal	Important e	Verifique se os status dos componentes do cluster podem ser obtidos ou se os componentes funcionam mal.
The node is not found in kubernetes cluster	The node is not found in kubernetes cluster	Important e	Verifique se o nó pode ser encontrado no cluster do Kubernetes.
The status of node is not ready in kubernetes cluster	The status of node is not ready in kubernetes cluster	Important e	Verifique se o nó está sendo executado corretamente no cluster do Kubernetes.
Can't find corresponding vm of this node in ECS	Can't find corresponding vm of this node in ECS	Important e	Verifique se o nó pode ser encontrado no console do ECS.

Nome do evento	ID do evento	Gravidade	Descrição
Failed to upgrade the master	Failed to upgrade the master	Important e	Verifique se o nó principal foi atualizado.
Failed to upgrade the node	Failed to upgrade the node	Important e	Verifique se o nó foi atualizado.
Failed to change flavor of the master	Failed to change flavor of the master	Important e	Verifique se o flavor do nó principal foi alterado.
Change flavor of the master timeout	Change flavor of the master timeout	Important e	Verifique se a alteração do flavor do nó principal expirou.
Failed to pass verification while creating yearly/monthly subscription node	Failed to pass verification while creating yearly/monthly subscription node	Important e	Verifique se a criação de um nó anual/mensal foi verificada.
Failed to install the node	Failed to install the node	Important e	Verifique se o nó está instalado no cluster.
Failed to clean routes of cluster container network in VPC	Failed to clean routes of cluster container network in VPC	Important e	Verifique se as rotas das VPCs de contêiner de cluster foram limpas.
Cluster status is Unavailable	Cluster status is Unavailable	Important e	Verifique se o cluster está disponível.
Cluster status is Error	Cluster status is Error	Important e	Verifique se o cluster está com defeito.
Cluster status is not updated for a long time	Cluster status is not updated for a long time	Important e	Verifique se o cluster mantém em um estado por um longo período de tempo.
Failed to update master status after upgrading cluster timeout	Failed to update master status after upgrading cluster timeout	Important e	Verifique se o status do nó mestre é atualizado após o tempo limite da atualização do cluster.
Failed to update running jobs after upgrading cluster timeout	Failed to update running jobs after upgrading cluster timeout	Important e	Verifique se as tarefas em execução são atualizadas após o término do tempo de atualização do cluster.
Failed to update cluster status	Failed to update cluster status	Important e	Verifique se o status do cluster está atualizado.

Nome do evento	ID do evento	Gravidade	Descrição
Failed to update node status	Failed to update node status	Importante	Verifique se o status do nó está atualizado.
Failed to remove the static node from database	Failed to remove the static node from database	Importante	Verifique se os nós são removidos do banco de dados após o gerenciamento de nós expirar.
Failed to update node status to abnormal after node processing timeout	Failed to update node status to abnormal after node processing timeout	Importante	Verifique se o status do nó está atualizado para anormal após o processamento do nó expirado.
Failed to update the cluster endpoint	Failed to update the cluster endpoint	Importante	Verifique se o ponto de extremidade do cluster está atualizado.
Failed to delete the unavailable connection of the Kubernetes cluster	Failed to delete the unavailable connection of the Kubernetes cluster	Importante	Verifique se as conexões indisponíveis do Kubernetes são excluídas.
Failed to sync the cluster cert	Failed to sync the cluster cert	Importante	Verifique se o certificado de cluster está sincronizado.

Adicionar alarmes de limite

A seguir, o alarme **Workload CPU Usage** é usado como exemplo para descrever como adicionar um alarme baseado em limite. Você também pode usar esse método para adicionar outros alarmes de limite.

Esta função é fornecida pelo AOM. Para obter detalhes, consulte **Personalização de regras de limite estático**.

Você pode configurar os alarmes de limite de acordo com **Tabela 9-13**.

AVISO

O uso da CPU do pod, o uso da memória física e os alarmes de uso do sistema de arquivos devem ser configurados para os componentes everest-csi-controller, everest-csi-driver, coredns, autoscaler e Yangtse. Atualize as especificações no caso de alto uso de recursos para evitar falhas do sistema.

Tabela 9-13 Configurações de alarme de limite

Recurso	Item de monitoramento	Descrição	Gatilho recomendado
Cluster	CPU Usage	Essa métrica é usada para calcular o uso da CPU do objeto medido.	Condição de limite: > 85%; período estatístico (minutos): 1; períodos consecutivos: 3
	Disk Usage	Essa métrica é usada para calcular a porcentagem do espaço em disco em uso para o espaço total em disco.	Condição de limite: > 85%; período estatístico (minutos): 1; períodos consecutivos: 3
	Physical Memory Usage	Essa métrica é usada para calcular a porcentagem da memória física usada pelo objeto medido em relação à memória física total.	Condição de limite: > 85%; período estatístico (minutos): 1; períodos consecutivos: 3
	Virtual Memory Usage	Essa métrica é usada para calcular a porcentagem da memória virtual usada pelo objeto medido em relação à memória virtual total.	Condição de limite: > 85%; período estatístico (minutos): 1; períodos consecutivos: 3
Host	CPU Usage	Essa métrica é usada para calcular o uso da CPU do objeto medido.	Condição de limite: > 85%; período estatístico (minutos): 1; períodos consecutivos: 3
	Physical Memory Usage	Essa métrica é usada para calcular a porcentagem da memória física usada pelo objeto medido em relação à memória física total.	Condição de limite: > 85%; período estatístico (minutos): 1; períodos consecutivos: 3
	Virtual Memory Usage	Essa métrica é usada para calcular a porcentagem da memória virtual usada pelo objeto medido em relação à memória virtual total.	Condição de limite: > 85%; período estatístico (minutos): 1; períodos consecutivos: 3
Host–network	Received Error Packet Rate	Essa métrica é usada para calcular o número de pacotes de erro recebidos por uma NIC por segundo.	Condição de limite: > 0; período estatístico (minutos): 1; períodos consecutivos: 3
	Send Error Packet Rate	Essa métrica é usada para calcular o número de pacotes de erro enviados por uma NIC por segundo.	Condição de limite: > 0; período estatístico (minutos): 1; períodos consecutivos: 3

Recurso	Item de monitoramento	Descrição	Gatilho recomendado
Host–file system	Disk Usage	Essa métrica é usada para calcular a porcentagem do espaço em disco em uso para o espaço total em disco.	Condição de limite: > 85%; período estatístico (minutos): 1; períodos consecutivos: 3
	Disk read/write status	Essa métrica é usada para coletar estatísticas sobre o status de leitura e gravação de discos em um host.	Condição de limite: >= 1; período estatístico (minutos): 1; períodos consecutivos: 1
Workload	Workload Status	Essa métrica é usada para verificar o status da carga de trabalho anormal.	Condição de limite: >= 1; período estatístico (minutos): 1; períodos consecutivos: 1
	CPU Usage	Essa métrica é usada para calcular o uso da CPU do objeto medido, ou seja, a relação entre os núcleos da CPU realmente usados pelo objeto medido e o total de núcleos da CPU para os quais o objeto medido solicitou.	Condição de limite: > 85%; período estatístico (minutos): 1; períodos consecutivos: 3
	Physical Memory Usage	Essa métrica é usada para calcular a porcentagem da memória física usada pelo objeto medido em relação à memória física total.	Condição de limite: > 85%; período estatístico (minutos): 1; períodos consecutivos: 3
	File System Usage	Essa métrica é usada para calcular o uso do sistema de arquivos de um objeto medido, ou seja, a porcentagem do sistema de arquivos usado em relação ao sistema de arquivos total. Essa métrica é suportada apenas para os contêineres que usam o Device Mapper no cluster do Kubernetes da versão 1.11 ou posterior.	Condição de limite: > 85%; período estatístico (minutos): 1; períodos consecutivos: 3
Pod	CPU Usage	Essa métrica é usada para calcular o uso da CPU do objeto medido, ou seja, a relação entre os núcleos da CPU realmente usados pelo objeto medido e o total de núcleos da CPU para os quais o objeto medido solicitou.	Condição de limite: > 85%; período estatístico (minutos): 1; períodos consecutivos: 3

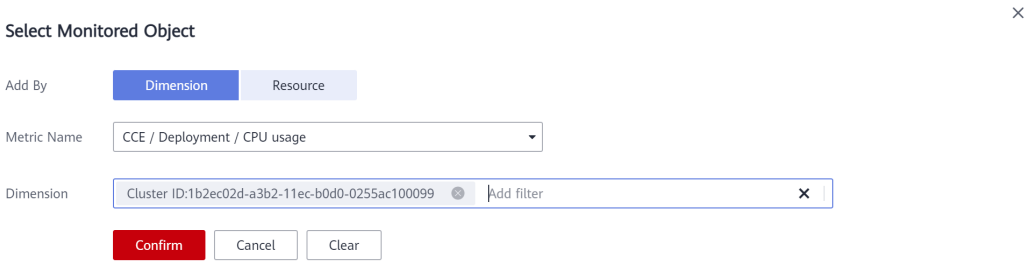
Recurso	Item de monitoramento	Descrição	Gatilho recomendado
	File System Usage	Essa métrica é usada para calcular o uso do sistema de arquivos de um objeto medido, ou seja, a porcentagem do sistema de arquivos usado em relação ao sistema de arquivos total. Essa métrica é suportada apenas para os contêineres que usam o Device Mapper no cluster do Kubernetes da versão 1.11 ou posterior.	Condição de limite: > 85%; período estatístico (minutos): 1; períodos consecutivos: 3
	Physical Memory Usage	Essa métrica é usada para calcular a porcentagem da memória física usada pelo objeto medido em relação à memória física total.	Condição de limite: > 85%; período estatístico (minutos): 1; períodos consecutivos: 3
	Container Status	Essa métrica é usada para verificar se o status do contêiner do Docker é normal.	Condição de limite: >= 1; período estatístico (minutos): 1; períodos consecutivos: 1
	Received Error Packet Rate	Essa métrica é usada para calcular o número de pacotes de erro recebidos por uma NIC por segundo.	Condição de limite: > 0; período estatístico (minutos): 1; períodos consecutivos: 3
	Error Packets Received	Essa métrica é usada para calcular o número de pacotes de erro recebidos por um objeto medido	Condição de limite: > 0; período estatístico (minutos): 1; períodos consecutivos: 3
	Send Error Packet Rate	Essa métrica é usada para calcular o número de pacotes de erro enviados por uma NIC por segundo.	Condição de limite: > 0; período estatístico (minutos): 1; períodos consecutivos: 3

Passo 1 Efetue logon no console do AOM.

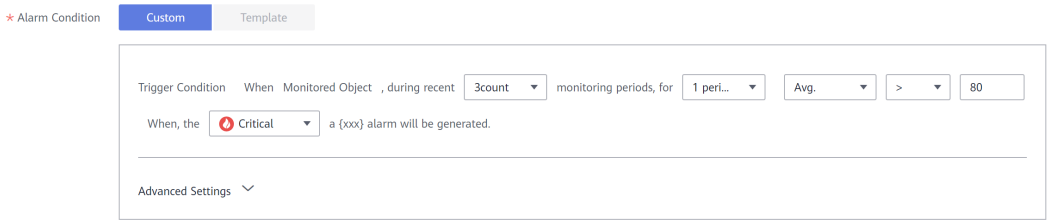
Passo 2 No painel de navegação, escolha **Alarm Center** > **Alarm Rules** e clique em **Add Alarm**.

Passo 3 Defina uma regra de alarme.

- **Rule Type**: selecione **Threshold Rule**.
- **Monitored Object**: clique em **Select resource objects**, defina **Add By** para **Dimension** e selecione **CCE/Deployment/CPU Usage** para **Metric Name**. Você pode filtrar recursos por várias dimensões, conforme necessário.



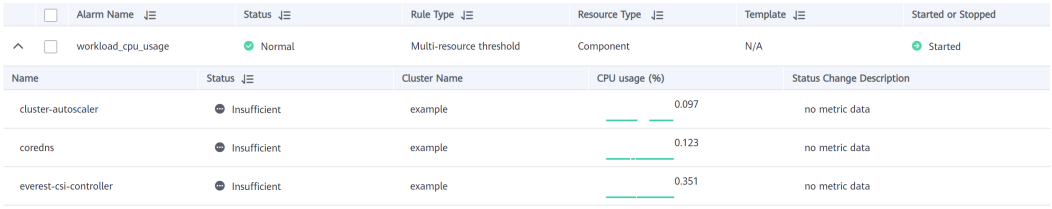
- **Alarm Condition**: defina parâmetros como o período estatístico, tempos consecutivos e condições de limite, conforme necessário.



- **Triggering Mode**: selecione **Immediate Triggering**.
- **Alarm Mode**: selecione **Direct Alarm Reporting**.
- **Action Policy**: selecione a política de ação criada em**Criar uma política de ação**.

Passo 4 Clique em **Create Now**.

Se as seguintes informações forem exibidas na lista de regras, a regra será criada com êxito. Neste exemplo, há várias cargas de trabalho porque nenhuma carga de trabalho é especificada nos critérios de filtro. Portanto, todas as cargas de trabalho no cluster são exibidas.



----Fim

9.5 Logs do CTS

9.5.1 Operações do CCE suportadas pelo CTS

O Cloud Trace Service (CTS) registra as operações nos recursos do serviço de nuvem, permitindo que os usuários consultem, auditem e retrocedam as solicitações de operação de recursos iniciadas a partir do console de gerenciamento ou de APIs abertas, bem como as respostas às solicitações.

Tabela 9-14 Operações do CCE suportadas pelo CTS

Operação	Tipo de recurso	Nome do evento
Criar uma agência	Cluster	createUserAgencies

Operação	Tipo de recurso	Nome do evento
Criar um cluster	Cluster	createCluster
Criar um cluster de faturamento anual/mensal	Cluster	createCluster/createPeriodicCluster
Atualizar a descrição de um cluster	Cluster	updateCluster
Atualizar um cluster	Cluster	clusterUpgrade
Excluir um cluster	Cluster	claimCluster/deleteCluster
Baixar um certificado de cluster	Cluster	getClusterCertByUID
Vincular e desvincular um EIP	Cluster	operateMasterEIP
Despertar um cluster e redefinir o gerenciamento de nó (V2)	Cluster	OperaCluster
Hibernar um cluster (V3)	Cluster	hibernateCluster
Despertar um cluster (V3)	Cluster	awakeCluster
Alterar as especificações de um cluster de pagamento por uso	Cluster	resizeCluster
Alterar as especificações de um cluster de cobrança anual/mensal	Cluster	resizePeriodCluster
Modificar configurações de um cluster	Cluster	updateConfiguration
Criar um pool de nós	Pool de nós	createNodePool
Atualizar um pool de nós	Pool de nós	updateNodePool
Excluir um pool de nós	Pool de nós	claimNodePool
Migrar um pool de nós	Pool de nós	migrateNodepool
Modificar configurações de pool de nós	Pool de nós	updateConfiguration

Operação	Tipo de recurso	Nome do evento
Criar um nó	Nó	createNode
Criar um nó de cobrança anual/mensal	Nó	createPeriodNode
Excluir todos os nós de um cluster especificado	Nó	deleteAllHosts
Excluir um único nó	Nó	deleteOneHost/claimOneHost
Atualizar a descrição de um nó	Nó	updateNode
Criar uma instância de complemento	Instância de complemento	createAddonInstance
Excluir uma instância de complemento	Instância de complemento	deleteAddonInstance
Carregar um gráfico	Gráfico	uploadChart
Atualizar um gráfico	Gráfico	updateChart
Excluir um gráfico	Gráfico	deleteChart
Criar uma release	Release	createRelease
Atualizar uma release	Release	updateRelease
Excluir uma release	Release	deleteRelease
Criar um ConfigMap	Recurso do Kubernetes	createConfigmaps
Criar um DaemonSet	Recurso do Kubernetes	createDaemonsets
Criar uma Implementação	Recurso do Kubernetes	createDeployments
Criar um evento	Recurso do Kubernetes	createEvents
Criar um Ingress	Recurso do Kubernetes	createIngresses
Criar uma tarefa	Recurso do Kubernetes	createJobs
Criar um namespace	Recurso do Kubernetes	createNamespaces
Criar um nó	Recurso do Kubernetes	createNodes
Criar uma PersistentVolumeClaim	Recurso do Kubernetes	createPersistentvolumeclaims

Operação	Tipo de recurso	Nome do evento
Criar um pod	Recurso do Kubernetes	createPods
Criar um conjunto de réplicas	Recurso do Kubernetes	createReplicasets
Criar uma cota de recursos	Recurso do Kubernetes	createResourcequotas
Criar um segredo	Recurso do Kubernetes	createSecrets
Criar um Serviço	Recurso do Kubernetes	createServices
Criar um StatefulSet	Recurso do Kubernetes	createStatefulsets
Criar um volume	Recurso do Kubernetes	createVolumes
Excluir um ConfigMap	Recurso do Kubernetes	deleteConfigmaps
Excluir um DaemonSet	Recurso do Kubernetes	deleteDaemonsets
Excluir uma Implementação	Recurso do Kubernetes	deleteDeployments
Excluir um evento	Recurso do Kubernetes	deleteEvents
Excluir um Ingress	Recurso do Kubernetes	deleteIngresses
Excluir uma tarefa	Recurso do Kubernetes	deleteJobs
Excluir um namespace	Recurso do Kubernetes	deleteNamespaces
Excluir um nó	Recurso do Kubernetes	deleteNodes
Excluir um pod	Recurso do Kubernetes	deletePods
Excluir um conjunto de réplicas	Recurso do Kubernetes	deleteReplicasets
Excluir uma cota de recurso	Recurso do Kubernetes	deleteResourcequotas
Excluir um segredo	Recurso do Kubernetes	deleteSecrets

Operação	Tipo de recurso	Nome do evento
Excluir um serviço	Recurso do Kubernetes	deleteServices
Excluir um StatefulSet	Recurso do Kubernetes	deleteStatefulsets
Excluir volumes	Recurso do Kubernetes	deleteVolumes
Substituir um ConfigMap especificado	Recurso do Kubernetes	updateConfigmaps
Substituir um DaemonSet especificado	Recurso do Kubernetes	updateDaemonsets
Substituir uma Implementação especificada	Recurso do Kubernetes	updateDeployments
Substituir um evento especificado	Recurso do Kubernetes	updateEvents
Substituir um ingress especificada	Recurso do Kubernetes	updateIngresses
Substituir uma tarefa especificada	Recurso do Kubernetes	updateJobs
Substituir um namespace especificado	Recurso do Kubernetes	updateNamespaces
Substituir um nó especificado	Recurso do Kubernetes	updateNodes
Substituir uma PersistentVolumeClaim especificada	Recurso do Kubernetes	updatePersistentvolumeclaims
Substituir um pod especificado	Recurso do Kubernetes	updatePods
Substituir um conjunto de réplicas especificado	Recurso do Kubernetes	updateReplicasets
Substituir uma cota de recurso especificada	Recurso do Kubernetes	updateResourcequotas
Substituir um segredo especificado	Recurso do Kubernetes	updateSecrets
Substituir um serviço especificado	Recurso do Kubernetes	updateServices

Operação	Tipo de recurso	Nome do evento
Substituir um StatefulSet especificado	Recurso do Kubernetes	updateStatefulsets
Substituir o status especificado	Recurso do Kubernetes	updateStatus
Carregar um gráfico	Recurso do Kubernetes	uploadChart
Atualizar um modelo de componente	Recurso do Kubernetes	updateChart
Excluir um gráfico	Recurso do Kubernetes	deleteChart
Criar uma aplicação de modelo	Recurso do Kubernetes	createRelease
Atualizar uma aplicação de modelo	Recurso do Kubernetes	updateRelease
Excluir uma aplicação de modelo	Recurso do Kubernetes	deleteRelease

9.5.2 Consulta de logs do CTS

Cenário

Depois que você habilita o CTS, o sistema começa a gravar operações em recursos do CCE. Os registros da operação dos últimos 7 dias podem ser vistos no console de gerenciamento do CTS.

Procedimento

Passo 1 Faça logon no console de gerenciamento.

Passo 2 Clique em no canto superior esquerdo e selecione uma região.

Passo 3 Escolha **Service List** no menu principal. Escolha **Management & Deployment** > **Cloud Trace Service**.

Passo 4 No painel de navegação do console do CTS, escolha **Cloud Trace Service** > **Trace List**.

Passo 5 Na página **Trace List**, os registros de operação de consulta com base nos critérios de pesquisa. Atualmente, a lista de rastreamento oferece suporte à consulta de rastreamento com base na combinação dos seguintes critérios de pesquisa:

- **Trace Source**, **Resource Type** e **Search By**

  Selecione os critérios de pesquisa nas listas suspensas. Selecione **CCE** na lista suspensa **Trace Source**.

  Se você selecionar **Trace name** na lista suspensa **Search By**, especifique o nome do rastreamento.

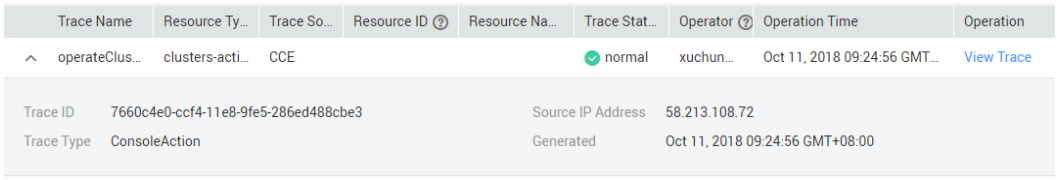
Se você selecionar **Resource ID** na lista suspensa **Search By**, selecione ou insira um ID de recurso específico.

Se você selecionar **Resource name** na lista suspensa **Search By**, selecione ou insira um nome de recurso específico.

- **Operator**: selecione um operador específico (no nível do usuário em vez de no nível do locatário).
- **Trace Status**: defina este parâmetro para qualquer um dos seguintes valores: **All trace statuses**, **normal**, **warning** e **incident**.
- Time range: você pode consultar rastreamentos gerados durante qualquer intervalo de tempo dos últimos sete dias.

Passo 6 Clique em à esquerda de um rastreamento para expandir seus detalhes, conforme mostrado abaixo .

Figura 9-82 Expansão de detalhes do rastreamento



Passo 7 Clique em **View Trace** na coluna **Operation**. Os detalhes do rastreamento são exibidos.

Figura 9-83 Exibir detalhes do evento

----Fim

10 Namespaces

10.1 Criação de um namespace

Quando usar namespaces

Um namespace é uma coleção de recursos e objetos. Vários namespaces podem ser criados dentro de um cluster e isolados uns dos outros. Isso permite que os namespaces compartilhem os mesmos serviços de cluster sem afetar uns aos outros.

Por exemplo, você pode implementar cargas de trabalho em um ambiente de desenvolvimento em um namespace e implementar cargas de trabalho em um ambiente de teste em outro namespace.

Pré-requisitos

Pelo menos um cluster foi criado.

Restrições

Um máximo de 6.000 Serviços podem ser criados em cada namespace. Os Serviços mencionados aqui indicam os recursos do Kubernetes Service adicionados para cargas de trabalho.

Tipos de namespace

Namespaces podem ser criados de uma das seguintes maneiras:

- Criado automaticamente: quando um cluster está ativo, os namespaces **default**, **kube-public**, **kube-system** e **kube-node-lease** são criados por padrão.
  - **default**: todos os objetos para os quais nenhum namespace é especificado são alocados para este namespace.
  - **kube-public**: os recursos neste namespace podem ser acessados por todos os usuários (incluindo usuários não autenticados), como complementos públicos e gráficos de contêineres.
  - **kube-system**: todos os recursos criados pelo Kubernetes estão neste namespace.

  - **kube-node-lease**: cada nó tem um objeto Lease associado nesse namespace. O objeto é atualizado periodicamente pelo nó. Tanto o NodeStatus quanto o NodeLease são considerados pulsações de um nó. Nas versões anteriores à v1.13, apenas o NodeStatus está disponível. O recurso NodeLease é introduzido na v1.13. O NodeLease é mais leve que o NodeStatus. Esse recurso melhora significativamente a escalabilidade e o desempenho do cluster.
- Criado manualmente: você pode criar namespaces para servir a propósitos separados. Por exemplo, você pode criar três namespaces, um para um ambiente de desenvolvimento, um para um ambiente de depuração conjunto e um para um ambiente de teste. Você também pode criar um namespace para serviços de logon e outro para serviços de jogos.

Criação de um namespace

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster.

Passo 2 Escolha **Namespaces** no painel de navegação e clique em **Create Namespace** no canto superior direito.

Passo 3 Defina parâmetros de namespace com base em **Tabela 10-1**.

Tabela 10-1 Parâmetros para criar um namespace

Parâmetro	Descrição
Name	Nome exclusivo do namespace criado.
Description	Descrição sobre o namespace.
Quota Management	As cotas de recursos podem limitar a quantidade de recursos disponíveis em namespaces, obtendo alocação de recursos por namespace. **AVISO** **É aconselhável definir cotas de recursos no namespace conforme necessário para evitar exceções de cluster ou nó causadas por sobrecarga de recursos.** Por exemplo, o número padrão de pods que podem ser criados em cada nó em um cluster é 110. Se você criar um cluster com 50 nós, poderá criar um máximo de 5.500 pods. Portanto, você pode definir uma cota de recursos para garantir que o número total de pods não exceda 5.500 em todos os namespaces. Insira um número inteiro. Se a cota de um recurso não for especificada, nenhum limite será colocado no recurso. Se desejar limitar a CPU ou a cota de memória, especifique o valor da solicitação de CPU ou memória ao criar uma carga de trabalho.

Passo 4 Após a conclusão da configuração, clique em **OK**.

----Fim

Usar o kubectl para criar um namespace

Defina um namespace.


```
apiVersion: v1
kind: Namespace
metadata:
  name: custom-namespace
```

Execute o comando **kubectl** para criá-lo.

```
$ kubectl create -f custom-namespace.yaml
namespace/custom-namespace created
```

Você também pode executar o comando **kubectl create namespace** para criar um namespace.

```
$ kubectl create namespace custom-namespace
namespace/custom-namespace created
```

10.2 Gerenciamento de namespaces

Usar namespaces

- Ao criar uma carga de trabalho, você pode selecionar um namespace para isolar recursos ou usuários.
- Ao consultar cargas de trabalho, você pode selecionar um namespace para exibir todas as cargas de trabalho no namespace.

Isolar namespaces

- **Isolar namespaces por ambiente**

  Uma aplicação geralmente passa pelos estágios de desenvolvimento, depuração conjunta e teste antes de ser lançado. Nesse processo, as cargas de trabalho implementadas em cada ambiente (estágio) são as mesmas, mas são definidas logicamente. Existem duas maneiras de defini-los:

  – Agrupe-os em diferentes clusters para diferentes ambientes.

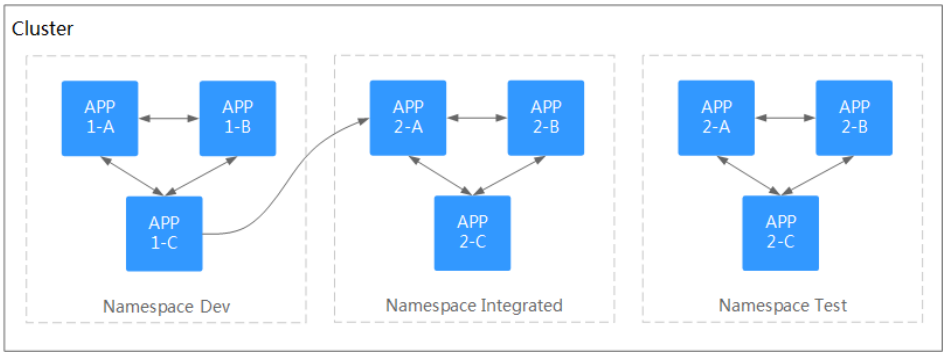
    Os recursos não podem ser compartilhados entre diferentes clusters. Além disso, os serviços em diferentes ambientes podem acessar uns aos outros apenas por meio do balanceamento de carga.

  – Agrupe-os em diferentes namespaces para diferentes ambientes.

    Cargas de trabalho no mesmo namespace podem ser acessadas mutuamente usando o nome do Serviço. O acesso entre namespaces pode ser implementado usando o nome do Serviço ou o nome do namespace.

    A figura a seguir mostra os namespaces criados para os ambientes de desenvolvimento, depuração conjunta e teste, respectivamente.

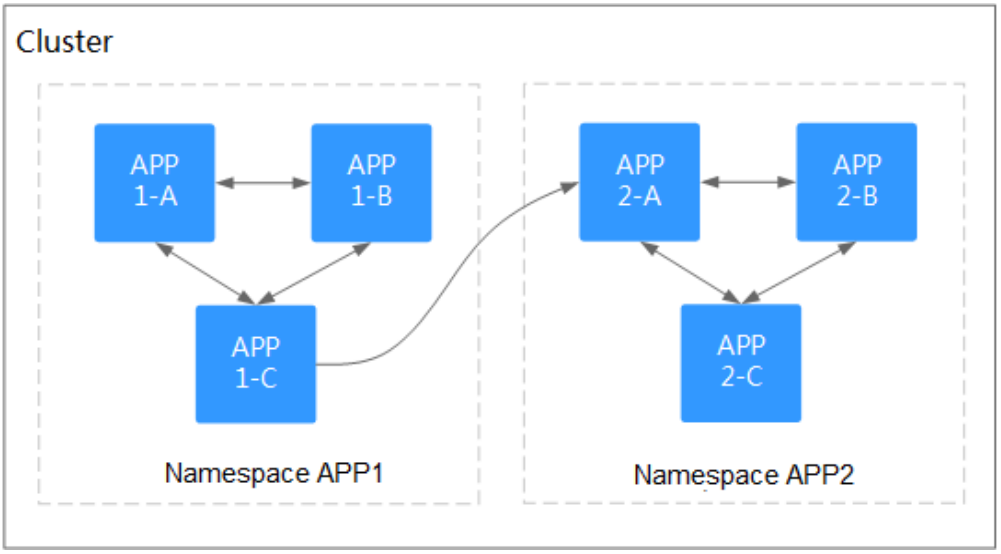
    Figura 10-1 Um namespace para um ambiente



- **Isolar namespaces por aplicação**

  É aconselhável usar esse método se um grande número de cargas de trabalho for implementado no mesmo ambiente. Por exemplo, na figura a seguir, diferentes namespaces (APP1 e APP2) são criados para gerenciar logicamente cargas de trabalho como grupos diferentes. As cargas de trabalho no mesmo namespace acessam umas às outras usando o nome do Serviço, e as cargas de trabalho em namespaces diferentes acessam umas às outras usando o nome do Serviço ou o nome do namespace.

  Figura 10-2 Agrupamento de cargas de trabalho em diferentes namespaces

Gerenciar rótulos de namespace

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster. No painel de navegação, escolha **Namespaces**.

Passo 2 Localize a linha que contém o namespace de destino e escolha **More** > **Manage Label** na coluna **Operation**.

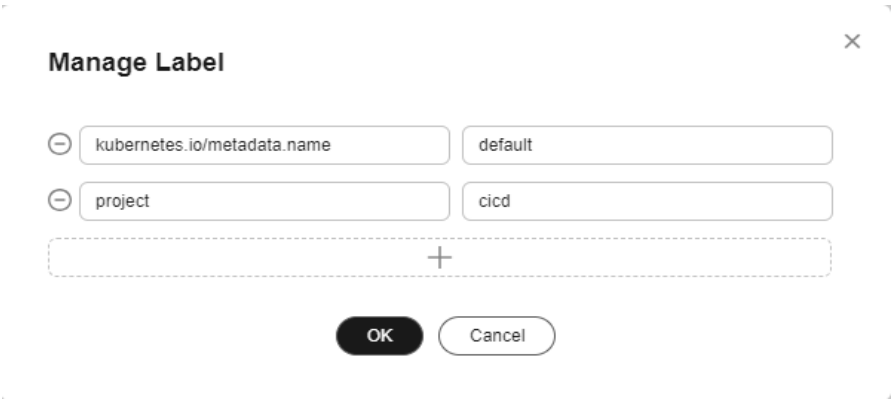
Passo 3 Na caixa de diálogo exibida, os rótulos existentes do namespace são exibidos. Modifique os rótulos conforme necessário.

- Adicionar um rótulo: clique no ícone de adicionar, insira a chave e o valor do rótulo a ser adicionado e clique em **OK**.

  Por exemplo, a chave é **project** e o valor é **cicd**, indicando que o namespace é usado para implementar o CICD.

- Excluir um rótulo: clique ⊖ ao lado do rótulo a ser excluído e, em seguida, em **OK**.

Figura 10-3 Adicionar ou excluir um rótulo de namespace



Passo 4 Alterne para a caixa de diálogo **Manage Label** novamente e verifique os rótulos modificados.

----Fim

Excluir um namespace

Se um namespace for excluído, todos os recursos (como cargas de trabalho, tarefas e ConfigMaps) nesse namespace também serão excluídos. Tenha cuidado ao excluir um namespace.

Passo 1 Efetue logon no console do CCE e acesse o console do cluster.

Passo 2 Escolha **Namespaces** no painel de navegação. Na página exibida, clique em **More** na linha do namespace de destino e escolha **Delete**.

Siga os prompts para excluir o namespace. Os namespaces padrão não podem ser excluídos.

----Fim

10.3 Configuração de uma cota de recurso

As cotas de recursos em nível de namespace limitam a quantidade de recursos disponíveis para equipes ou usuários quando essas equipes ou usuários usam o mesmo cluster. As cotas incluem o número total de um tipo de objeto e a quantidade total de recursos de computação (CPU e memória) consumidos pelos objetos.

Utilização

Por padrão, os pods em execução podem usar as CPUs e a memória de um nó sem restrições. Isso significa que os pods em um namespace podem esgotar todos os recursos do cluster.

O Kubernetes fornece namespaces para você agrupar cargas de trabalho em um cluster. Ao definir cotas de recursos para cada namespace, você pode evitar o esgotamento de recursos e garantir a confiabilidade do cluster.

Você pode configurar cotas para recursos como CPU, memória e o número de pods em um namespace. Para obter mais informações, consulte **Cotas de recursos**.

A tabela a seguir recomenda quantos pods você pode configurar para seus clusters de tamanhos diferentes.

Escala do cluster	Número recomendado de pods
50 nós	2.500 pods
200 nós	10.000 pods
1.000 nós	30.000 pods
2.000 nós	50.000 pods

A partir de clusters da v1.21 e posterior, as **Cotas de recursos** padrão são criadas quando um namespace é criado se você tiver habilitado **enable-resource-quota** no **Gerenciamento de configuração de cluster**. **Tabela 10-2** lista as cotas de recursos com base nas especificações do cluster. Você pode modificá-las de acordo com suas necessidades de serviço.

Tabela 10-2 Cotas de recurso padrão

Escala do cluster	Pod	Implementação	Segredo	ConfigMap	Serviço
50 nós	2000	1000	1000	1000	1000
200 nós	2000	1000	1000	1000	1000
1.000 nós	5000	2000	2000	2000	2000
2.000 nós	5000	2000	2000	2000	2000

Restrições

O Kubernetes fornece controle de simultaneidade otimista (OCC), também conhecido como bloqueio otimista, para atualizações frequentes de dados. Você pode usar o bloqueio otimista definindo o campo **resourceVersion**. Este campo está nos metadados do objeto. Este campo identifica o número da versão interna do objeto. Quando o objeto é modificado, este campo é modificado de acordo. Você pode usar kube-apiserver para verificar se um objeto foi modificado. Quando o servidor de API recebe uma solicitação de atualização contendo o campo **resourceVersion**, o servidor compara os dados solicitados com o número da versão do recurso do servidor. Se forem diferentes, o objeto no servidor foi modificado quando a atualização é enviada. Nesse caso, o servidor da API retorna um erro de conflito (409). Obtenha os dados do servidor, modifique os dados e envie os dados para o servidor novamente. A cota de recursos limita o consumo total de recursos de cada namespace e registra as informações de recursos no cluster. Portanto, depois que a opção **enable-resource-quota** é ativada, a probabilidade de conflitos de criação de recursos aumenta em cenários de simultaneidade em larga escala, afetando o desempenho da criação de recursos em lote.

Procedimento

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster.

Passo 2 No painel de navegação, clique em **Namespaces**.

Passo 3 Clique em **Quota Management** ao lado do namespace de destino.

Esta operação não pode ser executada nos namespaces do sistema **kube-system** e **kube-public**.

Passo 4 Defina as cotas de recursos e clique em **OK**.

AVISO

- Depois de definir as cotas de CPU e memória para um namespace, você deve especificar os valores de solicitação e limite de recursos de CPU e memória ao criar uma carga de trabalho. Caso contrário, a carga de trabalho não poderá ser criada. Se a cota de um recurso for definida como **0**, o uso do recurso não será limitado.
- O uso da cota acumulada inclui os recursos usados pelo CCE para criar componentes padrão, como os Serviços do Kubernetes (que podem ser visualizados usando o kubectl) criados no namespace **default**. Portanto, é aconselhável definir uma cota de recursos maior do que o esperado para reservar recursos para criar componentes padrão.

----Fim

11 ConfigMaps e segredos

11.1 Criação de um ConfigMap

Cenário

Um ConfigMap é um tipo de recurso que armazena informações de configuração exigidas por uma carga de trabalho. Seu conteúdo é definido pelo usuário. Depois de criar ConfigMaps, você pode usá-los como arquivos ou variáveis de ambiente em uma carga de trabalho em contêiner.

ConfigMaps permitem desacoplar arquivos de configuração de imagens de contêiner para melhorar a portabilidade de cargas de trabalho.

Benefícios dos ConfigMaps:

- Gerenciar configurações de diferentes ambientes e serviços.
- Implementar cargas de trabalho em diferentes ambientes. Várias versões são suportadas para arquivos de configuração para que você possa atualizar e reverter cargas de trabalho facilmente.
- Importar rapidamente configurações na forma de arquivos para contêineres.

Restrições

- O tamanho de um arquivo de recurso ConfigMap não pode exceder 2 MB.
- ConfigMaps não podem ser usados em **pods estáticos**.

Procedimento

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster.

Passo 2 Escolha **ConfigMaps and Secrets** no painel de navegação e clique em **Create ConfigMap** no canto superior direito.

Passo 3 Defina parâmetros.

Tabela 11-1 Parâmetros para criar um ConfigMap

Parâmetro	Descrição
Name	Nome de um ConfigMap que deve ser exclusivo em um namespace.
Namespace	Namespace ao qual o ConfigMap pertence. Se você não especificar esse parâmetro, o valor **default** será usado por padrão.
Description	Descrição do ConfigMap.
Data	Dados de um ConfigMap no formato de par chave-valor. Clique em + para adicionar dados. O valor pode estar no formato cadeia, JSON ou YAML.
Label	Rótulo do ConfigMap. Insira um par chave-valor e clique em **Add**.

Passo 4 Após a conclusão da configuração, clique em **OK**.

O novo ConfigMap é exibido na lista de ConfigMap.

----Fim

Criar um ConfigMap usando kubectl

Passo 1 Configure o comando **kubectl** para conectar um ECS ao cluster. Para mais detalhes, consulte **Conexão a um cluster usando o kubectl**.

Passo 2 Crie um arquivo chamado **cce-configmap.yaml** e edite-o.

vi cce-configmap.yaml

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cce-configmap
data:
  SPECIAL_LEVEL: Hello
  SPECIAL_TYPE: CCE
```

Tabela 11-2 Parâmetros principais

Parâmetro	Descrição
apiVersion	O valor é fixado em **v1**.
kind	O valor é fixado em **ConfigMap**.
metadata.name	Nome do ConfigMap, que pode ser personalizado.
data	Dados do ConfigMap. O valor deve ser pares chave-valor.

Passo 3 Execute os seguintes comandos para criar um ConfigMap.

kubectl create -f cce-configmap.yaml

Execute os seguintes comandos para exibir o ConfigMap criado:

kubectl get cm

```
NAME                DATA            AGE
cce-configmap       3               7m
```

----Fim

Operações relacionadas

Depois de criar um item de configuração, você pode atualizá-lo ou excluí-lo conforme descrito em **Tabela 11-3**.

Tabela 11-3 Operações relacionadas

Operação	Descrição
Editar um arquivo YAML	Clique em **Edit YAML** na linha em que o ConfigMap de destino reside para editar seu arquivo YAML.
Atualizar um ConfigMap	1. Selecione o nome do ConfigMap a ser atualizado e clique em **Update**. 2. Modifique os dados do segredo. Para obter mais informações, consulte **Tabela 11-1**. 3. Clique em **OK**.
Excluir um ConfigMap	Selecione a configuração que deseja excluir e clique em **Delete**. Siga as instruções para excluir o ConfigMap.

11.2 Uso de um ConfigMap

Depois que um ConfigMap é criado, ele pode ser usado em três cenários de carga de trabalho: variáveis de ambiente, parâmetros de linha de comando e volumes de dados.

- **Configurar variáveis de ambiente de carga de trabalho**
- **Definição de parâmetros de linha de comando**
- **Anexação de um ConfigMap ao volume de dados da carga de trabalho**

O exemplo a seguir mostra como usar um ConfigMap.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cce-configmap
data:
  SPECIAL_LEVEL: Hello
  SPECIAL_TYPE: CCE
```


AVISO

- Quando um ConfigMap é usado em uma carga de trabalho, a carga de trabalho e o ConfigMap devem estar no mesmo cluster e namespace.
- Quando um ConfigMap é montado como um volume de dados e o ConfigMap é atualizado, o Kubernetes atualiza os dados no volume de dados ao mesmo tempo.

  Para um volume de dados do ConfigMap montado no modo **subPath**, o Kubernetes não pode atualizar automaticamente os dados no volume de dados quando o ConfigMap é atualizado.
- Quando um ConfigMap é usado como uma variável de ambiente, os dados não são atualizados automaticamente quando o ConfigMap é atualizado. Para atualizar os dados, reinicie o pod.

Configurar variáveis de ambiente de carga de trabalho

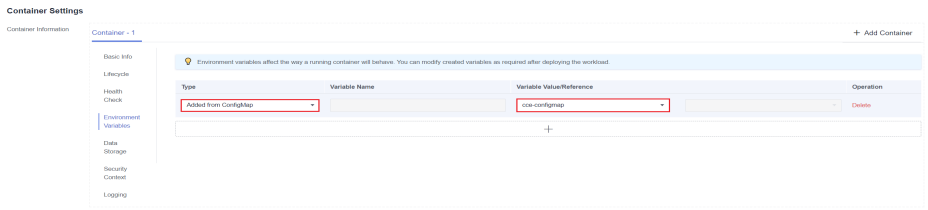
Usar o console

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster.

Passo 2 No painel de navegação, escolha **Workloads**. Em seguida, clique em **Create Workload**.

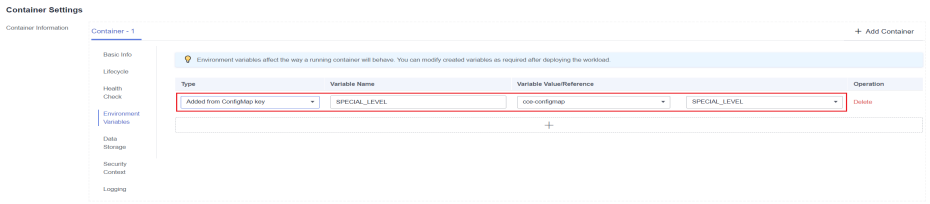
Ao criar uma carga de trabalho, clique em **Environment Variables** na área **Container Settings** e clique em .

- **Added from ConfigMap**: selecione um ConfigMap para importar todas as suas chaves como variáveis de ambiente.



- **Added from ConfigMap key**: importe uma chave em um ConfigMap como o valor de uma variável de ambiente.
  - **Variable Name**: nome de uma variável de ambiente na carga de trabalho. O nome pode ser personalizado e é definido como o nome da chave selecionado no ConfigMap por padrão.
  - **Variable Value/Reference**: selecione um ConfigMap e a chave a serem importados. O valor correspondente é importado como uma variável de ambiente de carga de trabalho.

  Por exemplo, depois de importar o valor **Hello** de **SPECIAL_LEVEL** no ConfigMap **cce-configmap** como o valor da variável de ambiente de carga de trabalho **SPECIAL_LEVEL**, uma variável de ambiente denominada **SPECIAL_LEVEL** com seu valor **Hello** existe no contêiner.



Passo 3 Configure outros parâmetros de carga de trabalho e clique em **Create Workload**.

Depois que a carga de trabalho for executada corretamente, **efetue logon no contêiner** e execute a seguinte instrução para verificar se o ConfigMap foi definido como uma variável de ambiente da carga de trabalho:

```
printenv SPECIAL_LEVEL
```

A saída de exemplo é a seguinte:

```
Hello
```

----Fim

Usar kubectl

Passo 1 Use o kubectl para se conectar ao cluster. Para mais detalhes, consulte **Conexão a um cluster usando o kubectl**.

Passo 2 Crie um arquivo chamado **nginx-configmap.yaml** e edite-o.

vi nginx-configmap.yaml

Conteúdo do arquivo YAML:

- **Added from a ConfigMap**: para adicionar todos os dados em um ConfigMap às variáveis de ambiente, use o parâmetro **envFrom**. As chaves no ConfigMap se tornarão nomes de variáveis de ambiente em uma carga de trabalho.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-configmap
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx-configmap
  template:
    metadata:
      labels:
        app: nginx-configmap
    spec:
      containers:
      - name: container-1
        image: nginx:latest
        envFrom:                        # Use envFrom to specify a ConfigMap to be referenced by environment variables.
        - configMapRef:
            name: cce-configmap         # Name of the referenced ConfigMap.
      imagePullSecrets:
      - name: default-secret
```

- **Added from a ConfigMap key**: ao criar uma carga de trabalho, você pode usar um segredo para definir variáveis de ambiente e usar o parâmetro **valueFrom** para fazer referência ao par chave-valor no ConfigMap separadamente.

```
apiVersion: apps/v1
kind: Deployment
```



```
metadata:
  name: nginx-configmap
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx-configmap
  template:
    metadata:
      labels:
        app: nginx-configmap
    spec:
      containers:
      - name: container-1
        image: nginx:latest
        env:                                # Set the environment variable in
the workload.
        - name: SPECIAL_LEVEL               # Name of the environment variable in
the workload.
          valueFrom:                        # Specify a ConfigMap to be
referenced by the environment variable.
            configMapKeyRef:
              name: cce-configmap           # Name of the referenced ConfigMap.
              key: SPECIAL_LEVEL            # Key in the referenced ConfigMap.
        - name: SPECIAL_TYPE                # Add multiple environment variables
to import them at the same time.
          valueFrom:
            configMapKeyRef:
              name: cce-configmap
              key: SPECIAL_TYPE
      imagePullSecrets:
      - name: default-secret
```

Passo 3 Crie uma carga de trabalho.

kubectl apply -f nginx-configmap.yaml

Passo 4 Visualize a variável de ambiente no pod.

1. Execute o seguinte comando para exibir o pod criado:
   ```
   kubectl get pod | grep nginx-configmap
   ```
   Saída esperada:
   ```
   nginx-configmap-***   1/1     Running   0             2m18s
   ```
2. Execute o seguinte comando para exibir as variáveis de ambiente no pod:
   ```
   kubectl exec nginx-configmap-*** -- printenv SPECIAL_LEVEL SPECIAL_TYPE
   ```
   Saída esperada:
   ```
   Hello
   CCE
   ```
   O ConfigMap foi definido como variáveis de ambiente da carga de trabalho.

----Fim

Definição de parâmetros de linha de comando

Você pode usar um ConfigMap como uma variável de ambiente para definir comandos ou valores de parâmetro para um contêiner usando a sintaxe de substituição de variável de ambiente $(VAR_NAME).

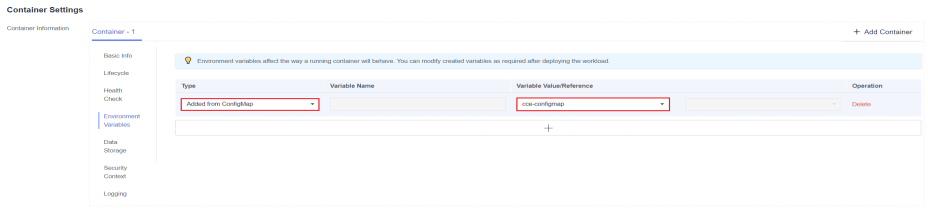
Usar o console

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster.

Passo 2 No painel de navegação, escolha **Workloads**. Em seguida, clique em **Create Workload**.

Ao criar uma carga de trabalho, clique em **Environment Variables** na área **Container Settings** e clique em +. Neste exemplo, selecione **Added from ConfigMap**.

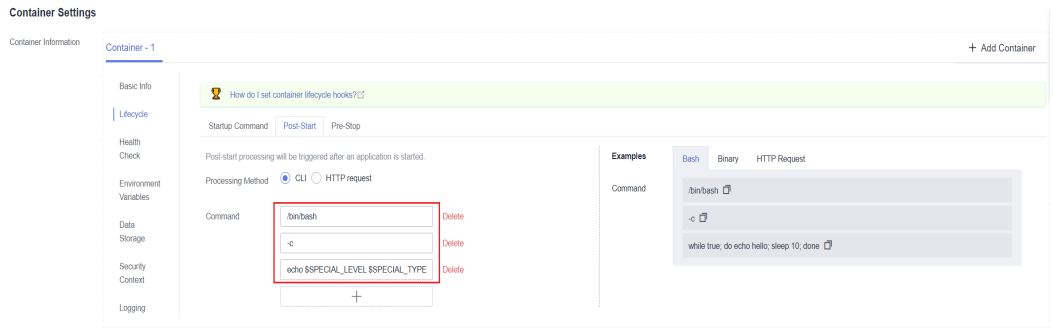
- **Added from ConfigMap**: selecione um ConfigMap para importar todas as suas chaves como variáveis de ambiente.



Passo 3 Clique em **Lifecycle** na área **Container Settings**, clique na guia **Post-Start** à direita e defina os seguintes parâmetros:

- **Processing Method**: **CLI**
- **Command**: insira as três linhas de comando a seguir. *SPECIAL_LEVEL* e *SPECIAL_TYPE* são os nomes das variáveis de ambiente na carga de trabalho, ou seja, os nomes das chaves no ConfigMap **cce-configmap**.

```
/bin/bash
-c
echo $SPECIAL_LEVEL $SPECIAL_TYPE > /usr/share/nginx/html/index.html
```



Passo 4 Defina outros parâmetros de carga de trabalho e clique em **Create Workload**.

Depois que a carga de trabalho for executada corretamente, **efetue logon no contêiner** e execute a seguinte instrução para verificar se o ConfigMap foi definido como uma variável de ambiente da carga de trabalho:

```
cat /usr/share/nginx/html/index.html
```

A saída de exemplo é a seguinte:

```
Hello CCE
```

----Fim

Usar kubectl

Passo 1 Use o kubectl para se conectar ao cluster. Para mais detalhes, consulte **Conexão a um cluster usando o kubectl**.

Passo 2 Crie um arquivo chamado **nginx-configmap.yaml** e edite-o.

vi nginx-configmap.yaml

Conforme mostrado no exemplo a seguir, o ConfigMap **cce-configmap** é importado para a carga de trabalho. *SPECIAL_LEVEL* e *SPECIAL_TYPE* são os nomes das variáveis de ambiente na carga de trabalho, ou seja, os nomes das chaves no ConfigMap **cce-configmap**.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-configmap
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx-configmap
  template:
    metadata:
      labels:
        app: nginx-configmap
    spec:
      containers:
      - name: container-1
        image: nginx:latest
        lifecycle:
          postStart:
            exec:
              command: [ "/bin/sh", "-c", "echo $SPECIAL_LEVEL $SPECIAL_TYPE
> /usr/share/nginx/html/index.html" ]
        envFrom:                        # Use envFrom to specify a ConfigMap to be
referenced by environment variables.
        - configMapRef:
            name: cce-configmap         # Name of the referenced ConfigMap.
      imagePullSecrets:
        - name: default-secret
```

Passo 3 Crie uma carga de trabalho.

kubectl apply -f nginx-configmap.yaml

Passo 4 Depois que a carga de trabalho é executada corretamente, o seguinte conteúdo é inserido no arquivo **/usr/share/nginx/html/index.html** no contêiner:

1. Execute o seguinte comando para exibir o pod criado:
   ```
   kubectl get pod | grep nginx-configmap
   ```
   Saída esperada:
   ```
   nginx-configmap-***   1/1     Running   0             2m18s
   ```
2. Execute o seguinte comando para exibir as variáveis de ambiente no pod:
   ```
   kubectl exec nginx-configmap-*** -- cat /usr/share/nginx/html/index.html
   ```
   Saída esperada:
   ```
   Hello CCE
   ```

----Fim

Anexação de um ConfigMap ao volume de dados da carga de trabalho

Os dados armazenados em um ConfigMap podem ser referenciados em um volume do tipo de ConfigMap. Você pode montar esse volume em um caminho de contêiner especificado. A plataforma suporta a separação de códigos de carga de trabalho e arquivos de configuração. Os volumes de ConfigMap são usados para armazenar parâmetros de configuração da carga de trabalho. Antes disso, crie ConfigMaps com antecedência. Para mais detalhes, consulte **Criação de um ConfigMap**.

Usar o console

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster.

Passo 2 No painel de navegação, escolha **Workloads**. Em seguida, clique em **Create Workload**.

Ao criar uma carga de trabalho, clique em **Data Storage** na área **Container Settings**. Clique em **Add Volume** e selecione **ConfigMap** na lista suspensa.

Passo 3 Defina o tipo de volume local como **ConfigMap** e defina os parâmetros para adicionar um volume local, conforme mostrado na **Tabela 11-4**.

Tabela 11-4 Montagem de um volume de ConfigMap

Parâmetro	Descrição
ConfigMap	Selecione o ConfigMap desejado. Um ConfigMap deve ser criado com antecedência. Para mais detalhes, consulte **Criação de um ConfigMap**.
Add Container Path	Configure os seguintes parâmetros: 1. **Mount Path**: insira um caminho do contêiner, por exemplo, **/tmp**. Este parâmetro indica o caminho do contêiner no qual um volume de dados será montado. Não monte o volume para um diretório do sistema, como / ou **/var/run**; essa ação pode causar erros de contêiner. É aconselhável montar o volume para um diretório vazio. Se o diretório não estiver vazio, verifique se não há arquivos que afetem a inicialização do contêiner. Caso contrário, os arquivos serão substituídos, causando falhas de inicialização do contêiner ou falhas de criação de carga de trabalho. **AVISO** Quando o contêiner é montado em um diretório de alto risco, é aconselhável usar uma conta com permissões mínimas para iniciar o contêiner; caso contrário, os arquivos de alto risco no computador host podem ser danificados. 2. **Subpath**: digite um subcaminho, por exemplo, **tmp**. – Um subcaminho é usado para montar um volume local para que o mesmo volume de dados seja usado em um único pod. Se este parâmetro for deixado em branco, o caminho raiz é usado por padrão. – O subcaminho pode ser a chave e o valor de um ConfigMap ou segredo. Se o subcaminho for um par chave-valor que não existe, a importação de dados não terá efeito. – Os dados importados especificando um subcaminho não serão atualizados junto com as atualizações do ConfigMap/segredo. 3. Defina a permissão para **Read-only**. Os volumes de dados no caminho são somente leitura. Você pode clicar em + para adicionar vários caminhos e subcaminhos.

----Fim

Usar kubectl

Passo 1 Use o kubectl para se conectar ao cluster. Para mais detalhes, consulte **Conexão a um cluster usando o kubectl**.

Passo 2 Crie um arquivo chamado **nginx-configmap.yaml** e edite-o.

vi nginx-configmap.yaml

Como mostrado no exemplo a seguir, depois que o volume de ConfigMap é montado, um arquivo de configuração com a chave como o nome do arquivo e o valor como o conteúdo do arquivo é gerado no diretório **/etc/config** do contêiner.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-configmap
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx-configmap
  template:
    metadata:
      labels:
        app: nginx-configmap
    spec:
      containers:
      - name: container-1
        image: nginx:latest
        volumeMounts:
        - name: config-volume
          mountPath: /etc/config            # Mount to the /etc/config directory.
          readOnly: true
    volumes:
    - name: config-volume
      configMap:
        name: cce-configmap                 # Name of the referenced ConfigMap.
```

Passo 3 Crie uma carga de trabalho.

kubectl apply -f nginx-configmap.yaml

Passo 4 Depois que a carga de trabalho é executada corretamente, os arquivos **SPECIAL_LEVEL** e **SPECIAL_TYPE** são gerados no diretório **/etc/config**. O conteúdo dos arquivos são **Hello** e **CCE**, respectivamente.

1. Execute o seguinte comando para exibir o pod criado:
   ```
   kubectl get pod | grep nginx-configmap
   ```
   Saída esperada:
   ```
   nginx-configmap-***   1/1     Running   0             2m18s
   ```
2. Execute o seguinte comando para exibir o arquivo **SPECIAL_LEVEL** ou **SPECIAL_TYPE** no pod:
   ```
   kubectl exec nginx-configmap-*** -- /etc/config/SPECIAL_LEVEL
   ```
   Saída esperada:
   ```
   Hello
   ```

----Fim

11.3 Criação de um segredo

Cenário

Um segredo é um tipo de recurso que contém dados confidenciais, como autenticação e informações da chave. Seu conteúdo é definido pelo usuário. Depois de criar segredos, você pode usá-los como arquivos ou variáveis de ambiente em uma carga de trabalho em contêiner.

Restrições

Segredos não podem ser usados em **pods estáticos**.

Procedimento

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster.

Passo 2 Escolha **ConfigMaps and Secrets** no painel de navegação, clique na guia **Secrets** e clique em **Create Secret** no canto superior direito.

Passo 3 Defina parâmetros.

Tabela 11-5 Parâmetros para criar um segredo

Parâmetro	Descrição
Name	Nome do segredo que você cria, que deve ser único.
Namespace	Namespace ao qual o segredo pertence. Se você não especificar esse parâmetro, o valor **default** será usado por padrão.
Description	Descrição de um segredo.
Type	Tipo do segredo que você cria. ● Opaque: segredo comum. ● kubernetes.io/dockerconfigjson: um segredo que armazena as informações de autenticação necessárias para extrair imagens de um repositório privado. ● **kubernetes.io/tls**: segredo TLS do Kubernetes, que é usado para armazenar o certificado exigido pelos serviços de balanceamento de carga da camada 7. Para obter detalhes sobre exemplos do segredo kubernetes.io/tls e sua descrição, consulte **Segredos TLS**. ● **IngressTLS**: segredo TLS fornecido pelo CCE para armazenar o certificado exigido pelos Serviços de balanceamento de carga da camada-7. ● **Outros:** outro tipo de segredo, que é especificado manualmente.

Parâmetro	Descrição
Secret Data	Os dados secretos da carga de trabalho podem ser usados em contêineres. ● Se **Secret Type** for **Opaque**, clique em +. Na caixa de diálogo exibida, insira um par chave-valor e selecione **Auto Base64 Encoding**. ● Se **Secret Type** for **kubernetes.io/dockerconfigjson**, digite a conta e a senha do repositório de imagens privadas. ● Se **Secret Type** for **kubernetes.io/tls** ou **IngressTLS**, carregue o arquivo de certificado e o arquivo de chave privada. **NOTA** – Um certificado é uma credencial autoassinada ou assinada pela AC usada para autenticação de identidade. – Uma solicitação de certificado é uma solicitação de assinatura com uma chave privada.
Secret Label	Rótulo do segredo. Insira um par chave-valor e clique em **Add**.

Passo 4 Após a conclusão da configuração, clique em **OK**.

O novo segredo é exibido na lista de chaves.

----Fim

Exemplo de configuração do arquivo de recursos de segredos

Esta seção descreve exemplos de configuração de arquivos de descrição de recursos de segredos.

- Tipo Opaque

  O arquivo **secret.yaml** é definido como mostrado abaixo. O campo **data** é preenchido como um par chave-valor e o campo **value** deve ser codificado usando Base64. Para obter detalhes sobre o método de codificação Base64, consulte **Codificação Base64**.

```
apiVersion: v1
kind: Secret
metadata:
  name: mysecret           #Secret name
  namespace: default       #Namespace. The default value is default.
data:
  <your_key>: <your_value>  # Enter a key-value pair. The value must be
encoded using Base64.
type: Opaque
```

- Tipo kubernetes.io/dockerconfigjson

  O arquivo **secret.yaml** é definido como mostrado abaixo. O valor de **.dockerconfigjson** deve ser codificado usando Base64. Para mais detalhes, consulte **Codificação Base64**.

```
apiVersion: v1
kind: Secret
metadata:
  name: mysecret           #Secret name
  namespace: default       #Namespace. The default value is default.
data:
```



```
  .dockerconfigjson: eyJh*****    # Content encoded using Base64.
type: kubernetes.io/dockerconfigjson
```

Para obter o conteúdo de **.dockerconfigjson**, execute as seguintes etapas:

a. Obtenha as seguintes informações de logon do repositório de imagens.
   - Endereço do repositório de imagens: a seção usa *address* como exemplo. Substitua-o pelo endereço real.
   - Nome de usuário: a seção usa *username* como um exemplo. Substitua-o pelo nome de usuário real.
   - Senha: a seção usa *password* como exemplo. Substitua-a pela senha atual.

b. Use Base64 para codificar o par chave-valor *username:password* e preencher o conteúdo codificado em **3**.

```
echo -n "username:password" | base64
```

Saída do comando:

```
dXNlcm5hbWU6cGFzc3dvcmQ=
```

c. Use Base64 para codificar o seguinte conteúdo JSON:

```
echo -n '{"auths":{"address":
{"username":"username","password":"password","auth":"dXNlcm5hbWU6cGFzc3dv
cmQ="}}}' | base64
```

Saída do comando:

```
eyJhdXRocyI6eyJhZGRyZXNzIjp7InVzZXJuYW1lIjoidXNlcm5hbWUiLCJwYXNzd29yZCI6I
nBhc3N3b3JkIiwiYXV0aCI6ImRYTmxjbTVoYldVNmNHRnpjM2R2Y21RPSJ9fX0=
```

O conteúdo codificado é o conteúdo de **.dockerconfigjson**.

- Tipo kubernetes.io/tls

  O valor de **tls.crt** e **tls.key** deve ser codificado usando Base64. Para mais detalhes, consulte **Codificação Base64**.

```
kind: Secret
apiVersion: v1
metadata:
  name: mysecret           #Secret name
  namespace: default       #Namespace. The default value is default.
data:
  tls.crt: LS0tLS1CRU*****FURS0tLS0t  # Certificate content, which must be
encoded using Base64.
  tls.key: LS0tLS1CRU*****VZLS0tLS0=  # Private key content, which must be
encoded using Base64.
type: kubernetes.io/tls
```

- Tipo IngressTLS

  O valor de **tls.crt** e **tls.key** deve ser codificado usando Base64. Para mais detalhes, consulte **Codificação Base64**.

```
kind: Secret
apiVersion: v1
metadata:
  name: mysecret           #Secret name
  namespace: default       #Namespace. The default value is default.
data:
  tls.crt: LS0tLS1CRU*****FURS0tLS0t  # Certificate content, which must be
encoded using Base64.
  tls.key: LS0tLS1CRU*****VZLS0tLS0=  # Private key content, which must be
encoded using Base64.
type: IngressTLS
```

Criar um segredo usando o kubectl

Passo 1 De acordo com **Conexão a um cluster usando o kubectl**, configure o comando **kubectl** para conectar um ECS ao cluster.

Passo 2 Crie e edite o arquivo **cce-secret.yaml** codificado em Base64.

```
# echo -n "content to be encoded" | base64
******
```

vi cce-secret.yaml

O arquivo YAML a seguir usa o tipo Opaque como exemplo. Para obter detalhes sobre os tipos, consulte **Exemplo de configuração do arquivo de recursos de segredos**.

```
apiVersion: v1
kind: Secret
metadata:
  name: mysecret
type: Opaque
data:
  <your_key>: <your_value>  # Enter a key-value pair. The value must be encoded
using Base64.
```

Passo 3 Crie um segredo.

kubectl create -f cce-secret.yaml

Você pode consultar o segredo após a criação.

kubectl get secret -n default

----Fim

Operações relacionadas

Depois de criar um segredo, você pode atualizá-lo ou excluí-lo conforme descrito em **Tabela 11-6**.

NOTA

A lista de segredos contém recursos de segredos do sistema que só podem ser consultados. Os recursos de segredos do sistema não podem ser atualizados ou excluídos.

Tabela 11-6 Operações relacionadas

Operação	Descrição
Editar um arquivo YAML	Clique em **Edit YAML** na linha em que o segredo de destino reside para editar seu arquivo YAML.
Atualizar um segredo	1. Selecione o nome do segredo a ser atualizado e clique em **Update**. 2. Modifique os dados do segredo. Para obter mais informações, consulte **Tabela 11-5**. 3. Clique em **OK**.
Excluir um segredo	Selecione o segredo que deseja excluir e clique em **Delete**. Siga as instruções para excluir o segredo.
Excluir segredos em lotes	1. Selecione os segredos a serem excluídos. 2. Clique em **Delete** acima da lista de segredos. 3. Siga as instruções para excluir os segredos.

Codificação Base64

Para codificar uma cadeia em Base64, execute o comando **echo -n *content to be encoded* | base64**. O seguinte é um exemplo:

```
root@ubuntu:~# echo -n "content to be encoded" | base64
******
```

11.4 Uso de um segredo

Depois que os segredos são criados, eles podem ser montados como volumes de dados ou expostos como variáveis de ambiente a serem usadas por um contêiner em um pod.

AVISO

Não execute nenhuma operação nos seguintes segredos. Para mais detalhes, consulte **Segredos do cluster**.

- Não opere segredos sob kube-system.
- Não opere default-secret e paas.elb em nenhum dos namespaces. O segredo padrão é usado para puxar a imagem privada do SWR e o paas.elb é usado para conectar o serviço no namespace ao serviço ELB.

- **Configurar variáveis de ambiente de uma carga de trabalho**
- **Configurar o volume de dados de uma carga de trabalho**

O exemplo a seguir mostra como usar um segredo.

```
apiVersion: v1
kind: Secret
metadata:
  name: mysecret
type: Opaque
data:
  username: ******  #The value must be Base64-encoded.
  password: ******  #The value must be encoded using Base64.
```

AVISO

- Quando um segredo é usado em um pod, o pod e o secret devem estar no mesmo cluster e namespace.
- Quando um segredo é atualizado, o Kubernetes atualiza os dados no volume de dados ao mesmo tempo.

  No entanto, quando um volume de dados de segredos montado no modo **subPath** é atualizado, o Kubernetes não pode atualizar automaticamente os dados no volume de dados.

Configurar variáveis de ambiente de uma carga de trabalho

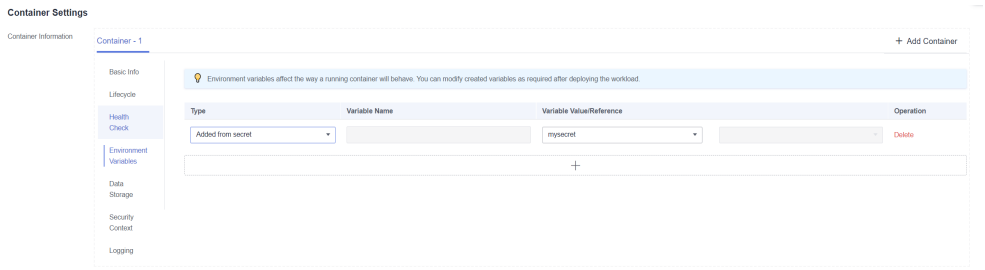
Usar o console

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster.

Passo 2 No painel de navegação, escolha **Workloads**. Em seguida, clique em **Create Workload**.

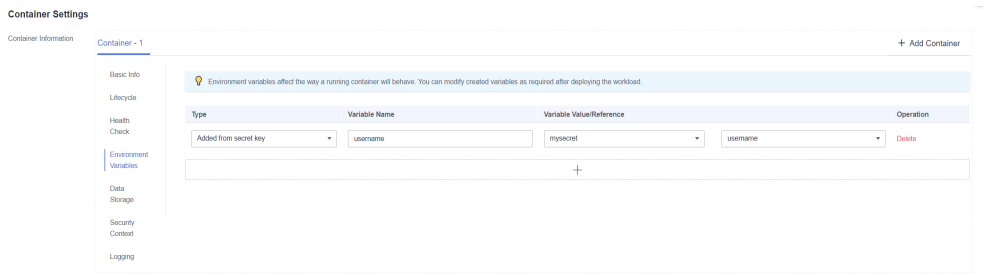
Ao criar uma carga de trabalho, clique em **Environment Variables** na área **Container Settings** e clique em +.

- **Added from secret**: selecione um segredo e importe todas as chaves no segredo como variáveis de ambiente.



- **Added from secret key**: importe o valor de uma chave em um segredo como o valor de uma variável de ambiente.
  - **Variable Name**: nome de uma variável de ambiente na carga de trabalho. O nome pode ser personalizado e é definido como o nome da chave selecionado no segredo por padrão.
  - **Variable Value/Reference**: selecione um segredo e a chave a ser importada. O valor correspondente é importado como uma variável de ambiente de carga de trabalho.

  Por exemplo, depois de importar o valor de **username** em segredo **mysecret** como o valor da variável de ambiente de carga de trabalho **username**, uma variável de ambiente chamada **username** existe no contêiner.



Passo 3 Defina outros parâmetros de carga de trabalho e clique em **Create Workload**.

Depois que a carga de trabalho for executada corretamente, **efetue logon no contêiner** e execute a seguinte instrução para verificar se o segredo foi definido como uma variável de ambiente da carga de trabalho:

```
printenv username
```

Se a saída for a mesma que o conteúdo no segredo, o segredo foi definido como uma variável de ambiente da carga de trabalho.

----Fim

Usar kubectl

Passo 1 Use o kubectl para se conectar ao cluster. Para mais detalhes, consulte **Conexão a um cluster usando o kubectl**.

Passo 2 Crie um arquivo chamado **nginx-secret.yaml** e edite-o.

vi nginx-secret.yaml

Conteúdo do arquivo YAML:

- **Adicionado do segredo**: para adicionar todos os dados em um segredo às variáveis de ambiente, use o parâmetro **envFrom**. As chaves no segredo se tornarão nomes de variáveis de ambiente em uma carga de trabalho.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-secret
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx-secret
  template:
    metadata:
      labels:
        app: nginx-secret
    spec:
      containers:
      - name: container-1
        image: nginx:latest
        envFrom:                 # Use envFrom to specify a secret to be
referenced by environment variables.
        - secretRef:
            name: mysecret       # Name of the referenced secret.
      imagePullSecrets:
      - name: default-secret
```

- **Adicionado de uma chave do segredo**: ao criar uma carga de trabalho, você pode usar um segredo para definir variáveis de ambiente e usar o parâmetro **valueFrom** para fazer referência ao par chave-valor no segredo separadamente.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-secret
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx-secret
  template:
    metadata:
      labels:
        app: nginx-secret
    spec:
      containers:
      - name: container-1
        image: nginx:latest
        env:                                # Set the environment variable in
the workload.
        - name: SECRET_USERNAME             # Name of the environment variable
in the workload.
          valueFrom:                       # Use valueFrom to specify a secret
to be referenced by environment variables.
            secretKeyRef:
              name: mysecret       # Name of the referenced secret.
              key: username        # Key in the referenced secret.
        - name: SECRET_PASSWORD            # Add multiple environment
variables to import them at the same time.
          valueFrom:
            secretKeyRef:
              name: mysecret
              key: password
```



```
      imagePullSecrets:
      - name: default-secret
```

Passo 3 Crie uma carga de trabalho.

kubectl apply -f nginx-secret.yaml

Passo 4 Visualize as variáveis de ambiente no pod.

1. Execute o seguinte comando para exibir o pod criado:
   ```
   kubectl get pod | grep nginx-secret
   ```
   Saída esperada:
   ```
   nginx-secret-***   1/1     Running   0              2m18s
   ```
2. Execute o seguinte comando para exibir as variáveis de ambiente no pod:
   ```
   kubectl exec nginx-secret-*** -- printenv SPECIAL_USERNAME SPECIAL_PASSWORD
   ```
   Se a saída for a mesma que o conteúdo no segredo, o segredo foi definido como uma variável de ambiente da carga de trabalho.

----Fim

Configurar o volume de dados de uma carga de trabalho

Você pode montar um segredo como um volume no caminho do contêiner especificado. O conteúdo em um segredo é definido pelo usuário. Antes disso, crie um segredo. Para mais detalhes, consulte **Criação de um segredo**.

Usar o console

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster.

Passo 2 No painel de navegação à esquerda, clique em **Workloads**. No painel direito, clique na guia **Deployments**. Clique em **Create Workload** no canto superior direito.

Ao criar uma carga de trabalho, clique em **Data Storage** na área **Container Settings**. Clique em **Add Volume** e selecione **Secret** na lista suspensa.

Passo 3 Defina o tipo de volume local como **Secret** e defina os parâmetros para adicionar um volume local, conforme mostrado na **Tabela 11-7**.

Tabela 11-7 Montar um volume de segredo

Parâmetro	Descrição
Secret	Selecione o segredo desejado. Um segredo deve ser criado antecipadamente. Para mais detalhes, consulte **Criação de um segredo**.

Parâmetro	Descrição
Add Container Path	Configure os seguintes parâmetros: 1. **Mount Path**: insira um caminho do contêiner, por exemplo, **/tmp**. Este parâmetro indica o caminho do contêiner no qual um volume de dados será montado. Não monte o volume para um diretório do sistema, como / ou **/var/run**; essa ação pode causar erros de contêiner. É aconselhável montar o volume para um diretório vazio. Se o diretório não estiver vazio, verifique se não há arquivos que afetem a inicialização do contêiner. Caso contrário, os arquivos serão substituídos, causando falhas de inicialização do contêiner ou falhas de criação de carga de trabalho. **AVISO** Quando o contêiner é montado em um diretório de alto risco, é aconselhável usar uma conta com permissões mínimas para iniciar o contêiner; caso contrário, os arquivos de alto risco no computador host podem ser danificados. 2. **Subpath**: digite um subcaminho, por exemplo, **tmp**. – Um subcaminho é usado para montar um volume local para que o mesmo volume de dados seja usado em um único pod. Se este parâmetro for deixado em branco, o caminho raiz é usado por padrão. – O subcaminho pode ser a chave e o valor de um ConfigMap ou segredo. Se o subcaminho for um par chave-valor que não existe, a importação de dados não terá efeito. – Os dados importados especificando um subcaminho não serão atualizados junto com as atualizações do ConfigMap/segredo. 3. Defina a permissão para **Read-only**. Os volumes de dados no caminho são somente leitura. Você pode clicar em + para adicionar vários caminhos e subcaminhos.

----Fim

Usar kubectl

Passo 1 Use o kubectl para se conectar ao cluster. Para mais detalhes, consulte **Conexão a um cluster usando o kubectl**.

Passo 2 Crie um arquivo chamado **nginx-secret.yaml** e edite-o.

vi nginx-secret.yaml

No exemplo a seguir, o nome de usuário e a senha no segredo **mysecret** são salvos no diretório **/etc/foo** como arquivos.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-secret
spec:
  replicas: 1
```



```
  selector:
    matchLabels:
      app: nginx-secret
  template:
    metadata:
      labels:
        app: nginx-secret
    spec:
      containers:
      - name: container-1
        image: nginx:latest
        volumeMounts:
       - name: foo
         mountPath: /etc/foo          # Mount to the /etc/foo directory.
         readOnly: true
    volumes:
    - name: foo
      secret:
        secretName: mysecret      # Name of the referenced secret.
```

Você também pode usar o campo **items** para controlar o caminho de mapeamento de chaves de segredo. Por exemplo, armazene o nome de usuário no diretório **/etc/foo/my-group/my-username** no contêiner.

NOTA

- Se você usar o campo **items** para especificar o caminho de mapeamento das chaves de segredo, as chaves que não forem especificadas não serão criadas como arquivos. Por exemplo, se a chave de **password** no exemplo a seguir não for especificada, o arquivo não será criado.
- Se você quiser usar todas as chaves em um segredo, você deve listar todas as chaves no campo **items**.
- Todas as chaves listadas no campo **items** devem existir no segredo correspondente. Caso contrário, o volume não será criado.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-secret
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx-secret
  template:
    metadata:
      labels:
        app: nginx-secret
    spec:
      containers:
      - name: container-1
        image: nginx:latest
        volumeMounts:
       - name: foo
         mountPath: /etc/foo          # Mount to the /etc/foo directory.
         readOnly: true
    volumes:
    - name: foo
      secret:
        secretName: mysecret      # Name of the referenced secret.
        items:
        - key: username      # Name of the referenced key.
          path: my-group/my-username    # Mapping path of the secret key
```

Passo 3 Crie uma carga de trabalho.

kubectl apply -f nginx-secret.yaml

Passo 4 Depois que a carga de trabalho é executada corretamente, os arquivos **username** e **password** são gerados no diretório **/etc/foo**.

1. Execute o seguinte comando para exibir o pod criado:
   ```
   kubectl get pod | grep nginx-secret
   ```
   Saída esperada:
   ```
   nginx-secret-***   1/1     Running   0             2m18s
   ```
2. Execute o seguinte comando para visualizar o arquivo **username** ou **password** no pod:
   ```
   kubectl exec nginx-secret-*** -- /etc/foo/username
   ```
   A saída esperada é a mesma que o conteúdo no segredo.

----Fim

11.5 Segredos do cluster

Por padrão, o CCE cria os seguintes segredos em cada namespace:

- default-secret
- paas.elb
- default-token-*xxxxx* (*xxxxx* é um número aleatório.)

As funções desses segredos são descritas a seguir.

default-secret

O tipo de **default-secret** é **kubernetes.io/dockerconfigjson**. Os dados são a credencial para fazer logon no repositório de imagens do SWR e são usados para extrair imagens do SWR. Para extrair uma imagem do SWR ao criar uma carga de trabalho no CCE, defina **imagePullSecrets** como **default-secret**.

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - image: nginx:alpine
    name: container-0
    resources:
      limits:
        cpu: 100m
        memory: 200Mi
      requests:
        cpu: 100m
        memory: 200Mi
  imagePullSecrets:
  - name: default-secret
```

Os dados do **default-secret** são atualizados periodicamente e os dados atuais expiram após um determinado período de tempo. Você pode executar o comando **describe** para exibir o tempo de expiração de default-secret.

AVISO

Use default-secret diretamente em vez de copiar o conteúdo secreto para criar um novo. A credencial no segredo copiado expirará e a imagem não poderá ser extraída.


```
$ kubectl describe secret default-secret
Name:         default-secret
Namespace:    default
Labels:       secret-generated-by=cce
Annotations:  temporary-ak-sk-expires-at: 2021-11-26 20:55:31.380909 +0000 UTC

Type:  kubernetes.io/dockerconfigjson

Data
====
.dockerconfigjson:  347 bytes
```

paas.elb

Os dados de **paas.elb** são os dados temporários de AK/SK, que são usados para criar balanceadores de carga do ELB durante a criação do Serviço e da entrada. Os dados de paas.elb são atualizados periodicamente e expiram após um determinado período de tempo.

Na prática, você não usará diretamente paas.elb. No entanto, não o elimine. Caso contrário, os balanceadores de carga do ELB não serão criados.

default-token-xxxxx

Por padrão, o Kubernetes cria uma conta de serviço chamada **default** para cada **default-token-*xxxxx*** é a chave da conta de serviço e ***xxxxx*** é um número aleatório.

```
$ kubectl get sa
NAME     SECRETS   AGE
default  1         30d
$ kubectl describe sa default
Name:                default
Namespace:           default
Labels:              <none>
Annotations:         <none>
Image pull secrets:  <none>
Mountable secrets:   default-token-xxxxx
Tokens:              default-token-xxxxx
Events:              <none>
```


12 Auto Scaling

12.1 Visão geral

Auto scaling é um serviço que ajusta automaticamente e economicamente os recursos de serviço com base em seus requisitos de serviço e políticas configuradas.

Contexto

Cada vez mais aplicações são desenvolvidas com base no Kubernetes. Torna-se cada vez mais importante expandir rapidamente aplicações no Kubernetes para lidar com picos de serviço e reduzir aplicações fora do horário de pico para economizar recursos e reduzir custos.

Em um cluster do Kubernetes, o dimensionamento automático envolve pods e nós. Um pod é uma instância de aplicação. Cada pod contém um ou mais contêineres e é executado em um nó (servidor VM ou bare-metal). Se um cluster não tiver nós suficientes para executar novos pods, adicione nós ao cluster para garantir a execução do serviço.

No CCE, o dimensionamento automático é usado para serviços on-line, computação e treinamento em larga escala, treinamento e inferência de GPU de aprendizado profundo ou GPU compartilhada, mudanças periódicas de carga e muitos outros cenários.

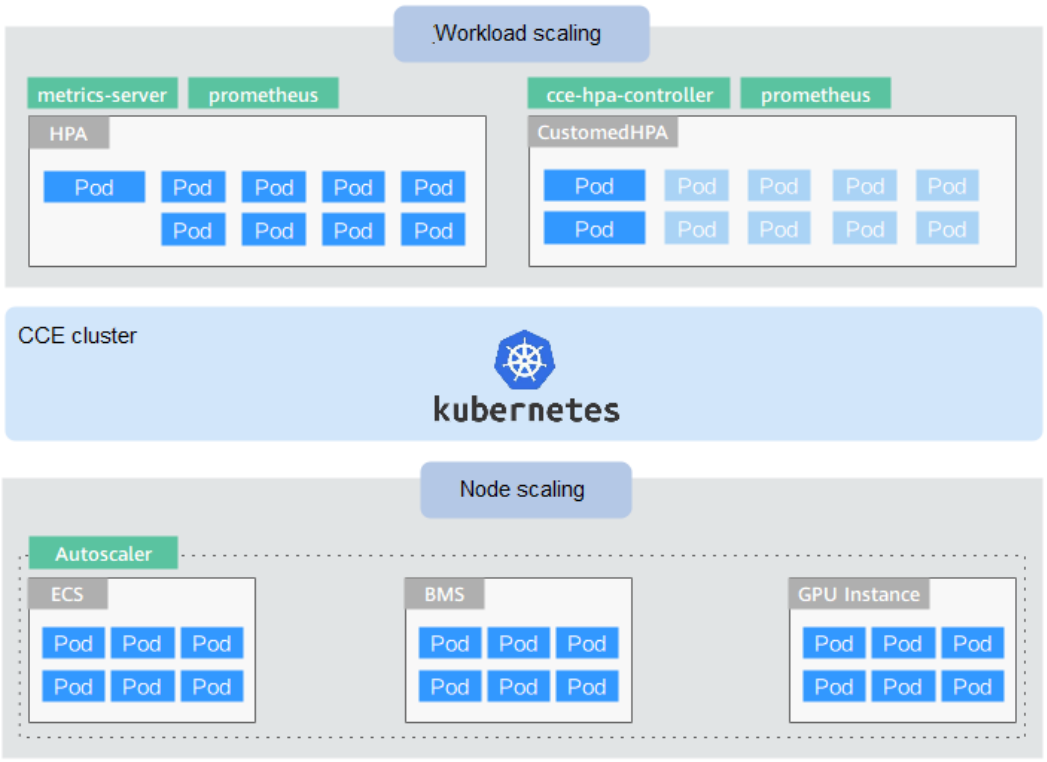
Dimensionamento automático em CCE

O CCE oferece suporte ao dimensionamento automático para cargas de trabalho e nós.

- **Workload scaling:** dimensionamento automático na camada de agendamento para alterar a capacidade de agendamento das cargas de trabalho. Por exemplo, você pode usar o HPA, um componente de escala na camada de agendamento, para ajustar o número de réplicas de uma aplicação. Ajustar o número de réplicas altera a capacidade de agendamento ocupada pela carga de trabalho atual, permitindo assim o dimensionamento na camada de agendamento.
- **Node scaling:** dimensionamento automático na camada de recurso. Quando os nós de cluster planejados não podem permitir o agendamento da carga de trabalho, os recursos ECS são fornecidos para dar suporte ao agendamento.

O dimensionamento de carga de trabalho e o dimensionamento de nós podem funcionar separadamente ou em conjunto. Para mais detalhes, consulte **Uso de HPA e CA para dimensionamento automático de cargas de trabalho e nós**.

Componentes



Os componentes de dimensionamento da carga de trabalho são descritos a seguir:

Tabela 12-1 Componentes de dimensionamento da carga de trabalho

Tipo	Nome do componente	Descrição do componente	Referência
HPA	**Kubernetes Metrics Server**	Um componente interno do Kubernetes, que permite o dimensionamento horizontal de pods. Ele adiciona a janela de tempo de resfriamento em nível de aplicação e as funções de limite de dimensionamento com base no HPA.	**Criação de uma política HPA para dimensionamento automático da carga de trabalho**
CustomedHPA	**HPA de CCE avançado**	Um recurso de escalonamento automático aprimorado, usado para escalonamento automático de implantações com base em métricas (uso da CPU e uso da memória) ou em intervalos periódicos (um ponto de tempo específico todos os dias, todas as semanas, todos os meses ou todos os anos).	**Criação de uma política CustomedHPA para o dimensionamento automático da carga de trabalho**

Tipo	Nome do componente	Descrição do componente	Referência
	Prometheus (EOM) **Monitoramento de cluster da nuvem nativa**	Uma estrutura de monitoramento e alarme de sistema de código aberto, que coleta métricas públicas (uso de CPU e uso de memória) do kubelet no cluster do Kubernetes.	
CronHPA	**HPA de CCE avançado**	CronHPA pode escalar dentro ou fora de um cluster em um tempo fixo. Ele pode trabalhar com políticas HPA para ajustar periodicamente o escopo de dimensionamento HPA, implementando o dimensionamento de carga de trabalho em cenários complexos.	**Políticas CronHPA**

Os componentes de dimensionamento de nó são descritos a seguir:

Tabela 12-2 Componentes de dimensionamento de nó

Nome do componente	Descrição do componente	Cenário de aplicação	Referência
Autoscaler de cluster do CCE	Um componente Kubernetes de código aberto para dimensionamento horizontal de nós, que é otimizado em termos de recursos de agendamento e dimensionamento automático.	Serviços on-line, aprendizagem profunda e computação em larga escala com orçamentos de recursos limitados	**Criação de uma política de dimensionamento de nós**

12.2 Dimensionamento de uma carga de trabalho

12.2.1 Mecanismos de dimensionamento da carga de trabalho

O CCE oferece suporte a vários modos de dimensionamento de carga de trabalho. As comparações entre as políticas de dimensionamento são listadas na tabela a seguir.

Tabela 12-3 Comparações entre políticas de dimensionamento automático

Item	HPA	CronHPA	CustomedHPA
Introdução	**Dimensionamento automático horizontal do pod**	Aprimorado com base em HPA, CronHPA é usado principalmente se o uso de recursos de aplicações muda periodicamente.	Dimensionamento automático do CCE aprimorado que é acionado com base em métricas ou em um horário programado.
Regras	Dimensiona Implementações com base em **métricas** (uso de CPU e uso de memória).	Escalas Implantações **periodicamente** (diária, semanal, mensal ou anual em um horário específico).	Dimensiona Implementações com base em **métricas** (uso de CPU e uso de memória) ou em um intervalo **periódico** (um ponto de tempo específico todos os dias, todas as semanas, todos os meses ou todos os anos).
Aprimoramento	Adiciona a janela de tempo de resfriamento no nível da aplicação e as funções de limite de escala com base no Kubernetes HPA.	Compatível com objetos de HPA, o que permite que você use CronHPA e HPA. ● Se ambos CronHPA e HPA são usados, CronHPA é executado com base em HPA e ajusta periodicamente o número de vagens para HPA. ● Se CronHPA for utilizado separadamente: o CronHPA ajusta periodicamente o número de pods para cargas de trabalho.	**Metric-based**: ● o dimensionamento pode ser realizado com base na porcentagem do número atual de pods. ● A etapa mínima de dimensionamento pode ser definida. O dimensionamento pode ser realizado passo a passo. ● Diferentes operações de escala podem ser executadas com base nos valores métricos reais. **Periodic**: você pode selecionar um ponto de tempo específico todos os dias, todas as semanas, todos os meses ou todos os anos, ou um período como o tempo de disparo.

Item	HPA	CronHPA	CustomedHPA
Utilização	**Criação de uma política HPA para dimensionamento automático da carga de trabalho**	**Políticas CronHPA**	**Criação de uma política CustomedHPA para o dimensionamento automático da carga de trabalho**

Como funciona o HPA

HPA é um controlador que controla o dimensionamento horizontal do pod. O HPA verifica periodicamente as métricas do pod, calcula o número de réplicas necessárias para atender aos valores de destino configurados para recursos HPA e, em seguida, ajusta o valor do campo **replicas** no objeto de recurso de destino (como uma Implementação).

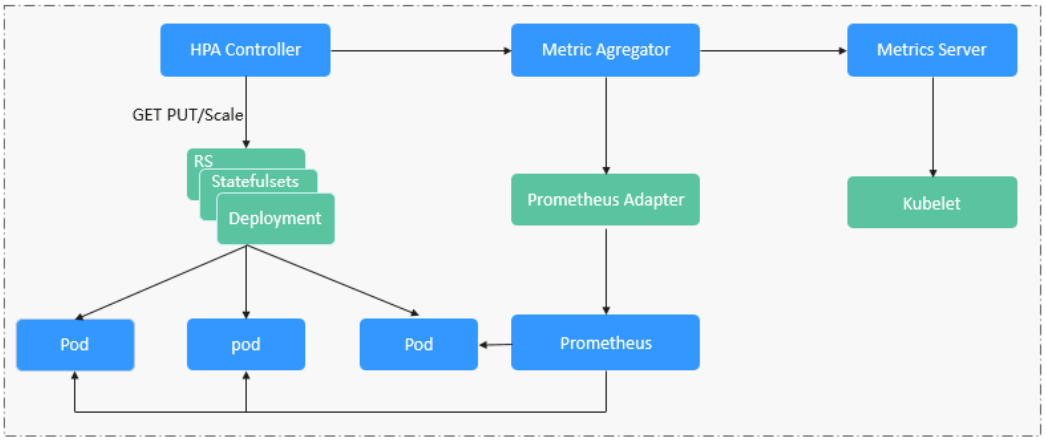
Um pré-requisito para o dimensionamento automático é que os dados em execução do contêiner possam ser coletados, como o número de nós/pods de cluster e o uso de CPU e memória de contêineres. O Kubernetes não fornece esses recursos de monitoramento em si. Você pode usar extensões para monitorar e coletar seus dados. O CCE integra **Prometheus** e **Metrics Server** para realizar esses recursos:

- **Prometheus** é uma estrutura de monitoramento e alarme de código aberto que pode coletar vários tipos de métricas. O Prometheus tem sido uma solução de monitoramento padrão do Kubernetes.
- **Metrics Server** é um agregador de dados de utilização de recursos em todo o cluster. O Metrics Server coleta métricas da API Summary exposta pelo kubelet. Essas métricas são definidas para os principais recursos do Kubernetes, como pods, nodes, contêineres e serviços. O Metrics Server fornece um conjunto de APIs padrão para sistemas externos coletarem essas métricas.

O HPA pode trabalhar com o Metrics Server para implementar o escalonamento automático com base no uso da CPU e da memória. Também pode funcionar com o Prometheus para escalonamento automático com base em métricas de monitoramento personalizadas.

Figura 12-1 mostra como funciona o HPA.

Figura 12-1 Processo de trabalho do HPA



Dois módulos principais do HPA:

- Monitoramento da fonte de dados

  A comunidade forneceu apenas HPA baseado em CPU e memória no estágio inicial. Com a população de Kubernetes e Prometheus, os desenvolvedores precisam de mais métricas personalizadas ou informações de monitoramento na camada de acesso para suas próprias aplicações, por exemplo, o QPS do balanceador de carga e o número de usuários on-line do site. Em resposta, a comunidade define um conjunto de APIs de métrica padrão para fornecer serviços externamente por meio dessas APIs agregadas.

  – **metrics.k8s.io** fornece métricas de monitoramento relacionadas à CPU e à memória de pods e nodes.

  – **custom.metrics.k8s.io** fornece métricas de monitoramento personalizadas relacionadas a objetos do Kubernetes.

  – **external.metrics.k8s.io** fornece métricas que vêm de sistemas externos e são irrelevantes para qualquer métrica de recurso do Kubernetes.

- Dimensionar algoritmos de tomada de decisão

  O controlador HPA calcula a taxa de escala com base nos valores métricos atuais e nos valores métricos desejados usando a seguinte fórmula:

  desiredReplicas = ceil[currentReplicas x (currentMetricValue/desiredMetricValue)]

  Por exemplo, se o valor métrico atual for 200m e o valor alvo for 100m, o número desejado de pods será duplicado de acordo com a fórmula. Na prática, as vagens podem ser constantemente adicionadas ou reduzidas. Para garantir a estabilidade, o controlador HPA é otimizado a partir dos seguintes aspectos:

  – Intervalo de resfriamento: na v1.11 e versões anteriores, o Kubernetes introduziu os parâmetros **horizontal-pod-autoscaler-downscale-stabilization-window** e **horizontal-pod-autoScaler-upscale-stabilization-window** para indicar os intervalos de resfriamento após uma redução e expansão, respectivamente, no qual nenhuma operação de dimensionamento não será executada. Nas versões posteriores à v1.14, a fila de agendamento é introduzida para armazenar todas as sugestões de tomada de decisão detectadas dentro de um período de tempo. Em seguida, o sistema toma decisões com base em todas as sugestões válidas de tomada de decisão para minimizar as alterações do número desejado de réplicas para garantir a estabilidade.

  – Tolerâncias: pode ser considerada como uma zona-tampão. Se as alterações do número de vagens puderem ser toleradas, o número de vagens permanece inalterado.

    Use a fórmula: ratio = currentMetricValue/desiredMetricValue

    Quando |ratio – 1,0| ≤ tolerância, o dimensionamento não será realizado.

    Quando |rácio – 1,0| > tolerância, o valor desejado é calculado usando a fórmula mencionada acima.

    O valor padrão é 0,1 na versão atual da comunidade.

O HPA executa dimensionamento com base em limiares métricos. As métricas comuns incluem o uso da CPU e da memória. Você também pode definir métricas personalizadas, como o QPS e o número de conexões, para acionar o escalonamento. No entanto, o escalonamento baseado em métricas traz latência de minutos gerados durante as fases de coleta, determinação e dimensionamento de dados. Essa latência pode causar alto uso da CPU e resposta lenta. Para resolver esse problema, o CCE permite que você configure políticas agendadas para dimensionar recursos regularmente para aplicações com alterações periódicas.

12.2.2 Criação de uma política HPA para dimensionamento automático da carga de trabalho

Horizontal Pod Autoscaling (HPA) no Kubernetes implementa o dimensionamento horizontal de pods. Em uma política HPA do CCE, você pode configurar diferentes janelas de tempo de resfriamento e limites de escala para diferentes aplicações com base no HPA do Kubernetes.

Pré-requisitos

Para usar HPA, instale um complemento que forneça APIs de métricas. Selecione um dos complementos a seguir com base na versão do cluster e nos requisitos reais.

- **Kubernetes Metrics Server**: fornece métricas básicas de uso de recursos, como uso de CPU e memória do contêiner. É suportado por todas as versões de cluster.
- **Monitoramento de cluster da nuvem nativa**: fornece métricas personalizadas além das métricas básicas de recursos. Registre o Prometheus como o serviço que fornece a API de métricas. Para mais detalhes, consulte **Fornecer métricas de recursos por meio de Metrics API**. Este suplemento suporta clusters v1.17 ou posterior.
- **Prometheus (EOM)**: fornece métricas personalizadas além das métricas básicas de recursos. Registre o Prometheus como o serviço que fornece a API de métricas. Para mais detalhes, consulte **Fornecer métricas de recursos por meio de Metrics API**. Este suplemento suporta apenas clusters de v1.21 ou anterior.

Restrições

- As políticas HPA podem ser criadas apenas para clusters de v1.13 ou posterior.
- Para clusters anteriores à v1.19.10, se uma política HPA for usada para expandir uma carga de trabalho com volumes do EVS montados, os pods existentes não poderão ser lidos ou gravados quando um novo pod for agendado para outro nó.

  Para clusters de v1.19.10 e posterior, se uma política HPA for usada para expandir uma carga de trabalho com o volume do EVS montado, um novo pod não poderá ser iniciado porque os discos EVS não podem ser anexados.

Procedimento

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster.

Passo 2 No painel de navegação, escolha **Workload Scaling**. Em seguida, clique em **Create HPA Policy** no canto superior direito.

Passo 3 Defina parâmetros de política.

Tabela 12-4 Parâmetros de política HPA

Parâmetro	Descrição
Nome da política	Name of the policy to be created. Defina este parâmetro conforme necessário.
Namespace	Namespace ao qual a carga de trabalho pertence.

Parâmetro	Descrição
Associated Workload	Carga de trabalho com a qual a política HPA está associada.
Pod Range	Número mínimo e máximo de vagens. Quando uma política é acionada, os pods de carga de trabalho são escalados dentro desse intervalo.
Cooldown Period	Intervalo entre uma redução e uma expansão. A unidade é minuto. **O intervalo não pode ser inferior a 1 minuto.** **Este parâmetro é suportado apenas em clusters de v1.15 a v1.23.** Este parâmetro indica o intervalo entre operações consecutivas de dimensionamento. O período de resfriamento garante que uma operação de dimensionamento seja iniciada somente quando a anterior for concluída e o sistema estiver funcionando de forma estável.
Scaling Behavior	**Este parâmetro é suportado apenas em clusters de v1.25 ou posterior.** ● **Default**: dimensiona cargas de trabalho usando o comportamento padrão do Kubernetes. Para obter detalhes, consulte **Comportamento padrão**. ● **Custom**: dimensiona cargas de trabalho usando políticas personalizadas, como stabilization window, steps e priorities. Parâmetros não especificados usam os valores recomendados pelo Kubernetes. – **Disable scale-out/scale-in**: selecione se deseja desativar a redução ou expansão. – **Stabilization Window**: um período durante o qual o CCE verifica continuamente se as métricas usadas para o dimensionamento continuam flutuando. O CCE dispara a escala se o estado desejado não é mantido para a janela inteira. Esta janela restringe o oscilação indesejado da contagem de pods devido a alterações métricas. – **Step**: especifica a etapa de dimensionamento. Você pode definir o número ou a porcentagem de pods a serem reduzidos ou expandidos de um período especificado. Se houver várias políticas, você poderá selecionar a política que maximiza ou minimiza o número de pods.

Parâmetro	Descrição
System Policy	• **Metric**: você pode selecionar **CPU usage** ou **Memory usage**. **NOTA** Uso = CPUs ou memória usadas por pods/CPUs ou memória solicitadas. • **Desired Value**: insira o uso médio de recursos desejado. Este parâmetro indica o valor desejado da métrica selecionada. Número de pods a serem escalados (arredondado) = (valor métrico atual/valor desejado) x número de pods atuais **NOTA** Ao calcular o número de pods a serem adicionados ou reduzidos, a política HPA usa o número máximo de pods nos últimos 5 minutos. • **Tolerance Range**: o dimensionamento não é disparado quando o valor da métrica está dentro da faixa de tolerância. O valor desejado deve estar dentro da faixa de tolerância. Se o valor da métrica for maior que o limite de redução e menor que o limite de expansão, nenhum dimensionamento será acionado. **Este parâmetro é suportado apenas em clusters de v1.15 ou posterior.**
Política personalizada (suportada apenas em clusters da v1.15 ou posterior)	**NOTA** Antes de definir uma política personalizada, instale um complemento que suporte a coleta de métricas personalizadas no cluster, por exemplo, o complemento prometheus. • **Metric Name**: nome da métrica personalizada. Você pode selecionar um nome conforme solicitado. Para mais detalhes, consulte **Monitoramento de métricas personalizadas usando o Prometheus**. • **Metric Source**: selecione um tipo de objeto na lista suspensa. Você pode selecionar **Pod**. • **Desired Value**: o valor médio da métrica de todos os pods. Número de pods a serem escalados (arredondado) = (valor métrico atual/valor desejado) x número de pods atuais **NOTA** Ao calcular o número de pods a serem adicionados ou reduzidos, a política HPA usa o número máximo de pods nos últimos 5 minutos. • **Tolerance Range**: o dimensionamento não é disparado quando o valor da métrica está dentro da faixa de tolerância. O valor desejado deve estar dentro da faixa de tolerância.

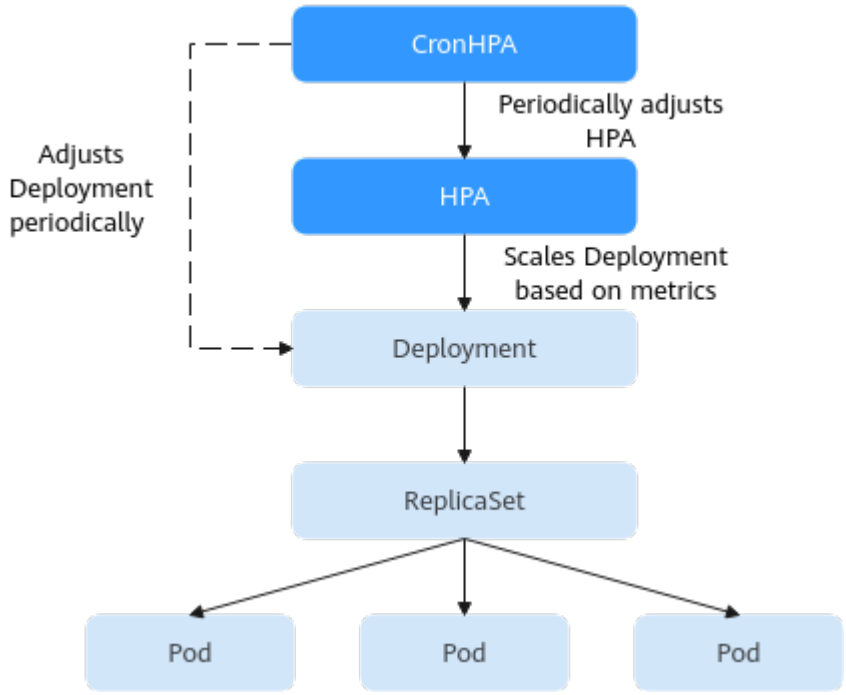
Passo 4 Clique em **Create**.

----Fim

12.2.3 Políticas CronHPA

Visão geral

Há picos de tráfego previsíveis e imprevisíveis para alguns serviços. Para esses serviços, CronHPA do CCE permite que você dimensione recursos em períodos fixos. Ele pode trabalhar com políticas HPA para ajustar periodicamente o escopo de dimensionamento HPA, implementando o dimensionamento de carga de trabalho.



CronHPA pode ajustar periodicamente os números máximo e mínimo de pods na política HPA ou ajustar diretamente o número de pods de uma Implementação.

O seguinte é um exemplo YAML de CronHPA:

```
apiVersion: autoscaling.cce.io/v2alpha1
kind: CronHorizontalPodAutoscaler
metadata:
  name: ccetest
  namespace: default
spec:
   scaleTargetRef:                  # Associate an HPA policy or Deployment.
      apiVersion: autoscaling/v1
      kind: HorizontalPodAutoscaler
      name: hpa-test
   rules:
   - ruleName: "scale-down"
     schedule: "15 * * * *"         # takes a Cron format string, for example, 0
* * * * or @hourly.
     targetReplicas: 1              # Number of target pods
     disable: false
   - ruleName: "scale-up"
     schedule: "13 * * * *"
     targetReplicas: 6
     disable: false
```

Tabela 12-5 Principais campos de CronHPA

Campo	Descrição
apiVersion	Versão da API. O valor é fixado em **autoscaling.cce.io/v2alpha1**.
kind	Tipo de API. O valor é fixado em **CronHorizontalPodAutoscaler**.
metadata.name	Nome de uma política CronHPA.
metadata.namespace	Namespace ao qual a política CronHPA pertence.

Campo	Descrição
spec.scaleTargetRef	Especifica o objeto de dimensionamento de CronHPA. Os seguintes campos podem ser configurados: ● **apiVersion**: versão da API do objeto de dimensionamento de CronHPA. ● **kind**: tipo de API do objeto de dimensionamento de CronHPA. ● **name**: nome do objeto de dimensionamento de CronHPA. CronHPA suporta políticas HPA ou Implementações. Para mais detalhes, veja **Usar CronHPA para ajustar o escopo de dimensionamento de HPA** e **Usar CronHPA para ajustar diretamente o número de pods de Implementação**.
spec.rules	Regra da política de CronHPA. Várias regras podem ser adicionadas. Os seguintes campos podem ser configurados para cada regra: ● **ruleName**: nome da regra de CronHPA, que deve ser exclusivo. ● **schedule**: tempo de execução e período de um trabalho. Para obter detalhes, consulte **Cron**, por exemplo, 0 * * * * or @hourly. ● **targetReplicas**: indica o número de pods a serem dimensionados para dentro ou para fora. ● **disable**: o valor pode ser **true** ou **false**. **false** indica que a regra entra em vigor e **true** indica que a regra não entra em vigor.

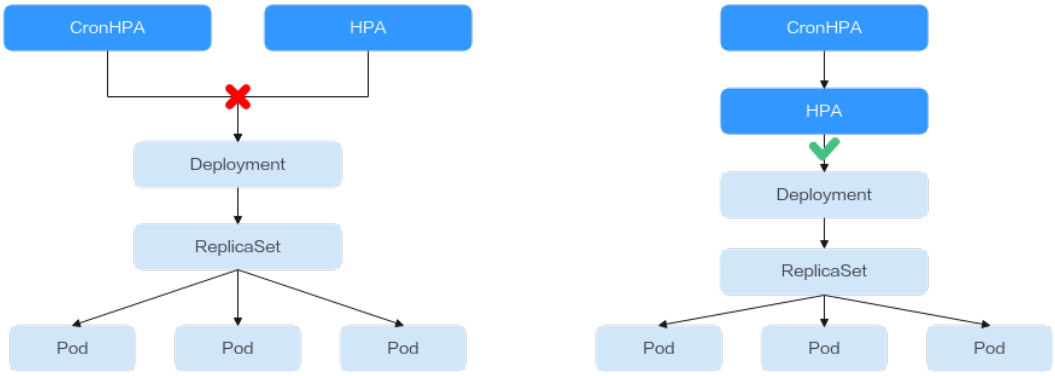
Pré-requisitos

O complemento **HPA de CCE avançado** da v1.2.13 ou posterior foi instalado.

Usar CronHPA para ajustar o escopo de dimensionamento de HPA

CronHPA pode periodicamente expandir/reduzir pods em políticas HPA para satisfazer serviços complexos.

HPA e CronHPA associam objetos de dimensionamento usando o campo **scaleTargetRef**. Se uma Implementação for o objeto de dimensionamento para CronHPA e HPA, as duas políticas de dimensionamento serão independentes uma da outra. A operação realizada posteriormente sobrescreve a operação realizada anteriormente. Como resultado, o efeito de escala não atende à expectativa.



Quando o CronHPA é compatível com a política HPA, o campo **scaleTargetRef** em CronHPA deve ser definido para a política HPA, e o campo **scaleTargetRef** na política HPA deve ser definido para Implementação. Desta forma, o CronHPA ajusta os números máximo e mínimo de pods na política HPA em um horário fixo e o dimensionamento agendado é compatível com o dimensionamento automático.

Passo 1 Crie uma política HPA para a Implementação.

```
apiVersion: autoscaling/v1
kind: HorizontalPodAutoscaler
metadata:
  name: hpa-test
  namespace: default
spec:
   maxReplicas: 10               #  Maximum number of pods
  minReplicas: 5                #  Minimum number of pods
  scaleTargetRef:               #  Associate a Deployment.
    apiVersion: apps/v1
    kind: Deployment
    name: nginx
  targetCPUUtilizationPercentage: 50
```

Passo 2 Crie uma política CronHPA e associe-a à política HPA criada em **Passo 1**.

```
apiVersion: autoscaling.cce.io/v2alpha1
kind: CronHorizontalPodAutoscaler
metadata:
  name: ccetest
  namespace: default
spec:
   scaleTargetRef:                  # Associate the HPA policy
      apiVersion: autoscaling/v1
      kind: HorizontalPodAutoscaler
      name: hpa-test
   rules:
   - ruleName: "scale-down"
     schedule: "15 * * * *"         #  Running time and period of a job. For
details, see Cron, for example, 0 * * * * or @hourly.
     targetReplicas: 1              # Number of target pods
     disable: false
   - ruleName: "scale-up"
     schedule: "13 * * * *"
     targetReplicas: 11
     disable: false
```

Quando CronHPA e HPA são usados juntos, as regras de CronHPA entram em vigor com base na política HPA. CronHPA usa HPA para realizar operações na Implementação. Compreender os seguintes parâmetros pode entender melhor o princípio de funcionamento do CronHPA.

- **targetReplicas**: número de pods definido para CronHPA. Quando o CronHPA entra em vigor, esse parâmetro ajusta o número máximo ou mínimo de pods nas políticas HPA para ajustar o número de pods de Implementação.

- **minReplicas**: número mínimo de pods de Implementação.
- **maxReplicas**: número máximo de pods de Implementação.
- **replicas**: número de pods em uma Implementação antes da política CronHPA entrar em vigor.

Quando a regra de CronHPA entra em vigor, o número máximo ou mínimo de pods é ajustado comparando o número de **targetReplicas** com o número real de pods e combinando o número mínimo ou máximo de pods na política HPA.

Figura 12-2 Cenários de dimensionamento CronHPA

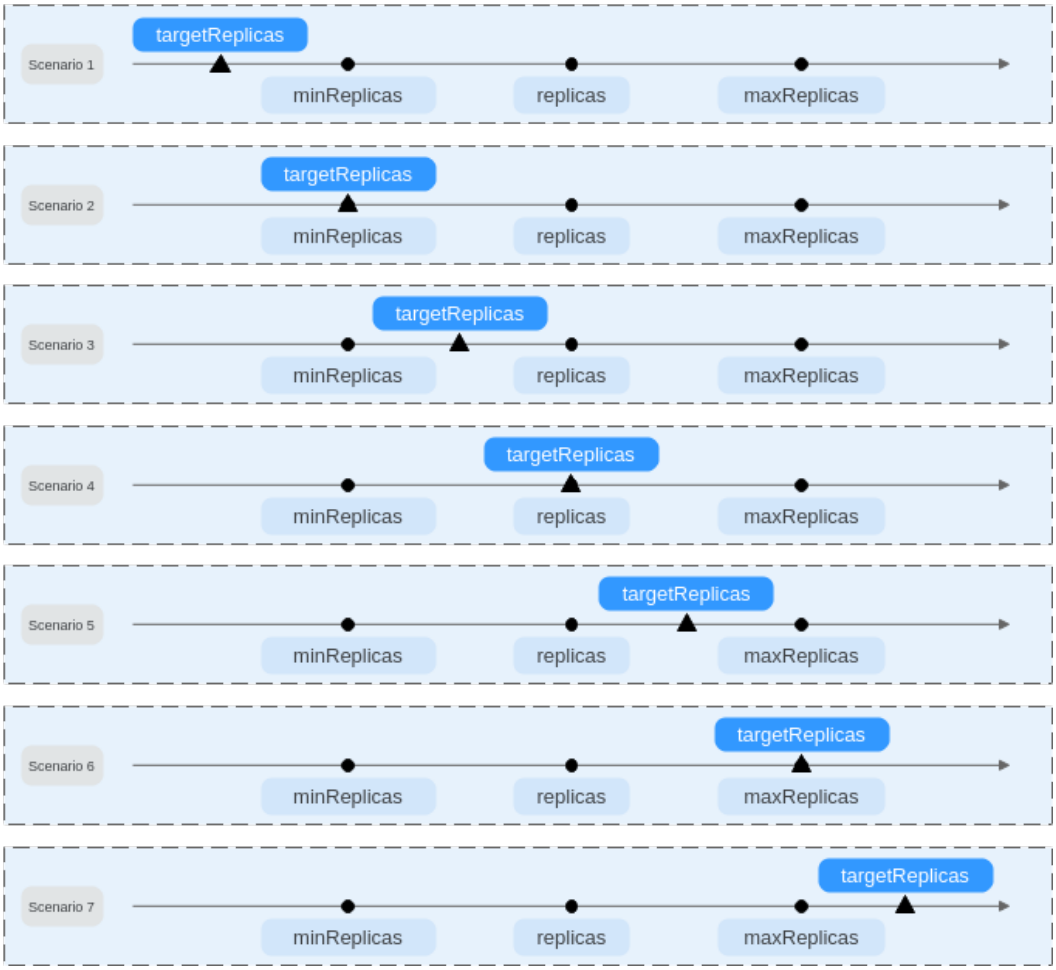


Figura 12-2 mostra possíveis cenários de dimensionamento. Os exemplos a seguir detalham como CronHPA modifica o número de pods em HPAs.

Cenário	Descrição do cenário	Cronhpa (targetReplicas)	Implementação (replicas)	HPA (minReplicas / maxReplicas)	Resultados	Descrição da operação
Cenário 1	**targetReplicas** < minReplicas ≤ replicas ≤ maxReplicas	4	5	5/10	HPA: 4/10 Implementação: 5	Quando o valor de**targetReplicas** for menor que o de **minReplicas**: ● Altere o valor de **minReplicas**. ● O valor de **replicas** não requer alteração.
Cenário 2	**targetReplicas** = minReplicas ≤ replicas ≤ maxReplicas	5	6	5/10	HPA: 5/10 Implementação: 6	Quando o valor de **targetReplicas** é igual ao de **minReplicas**: ● O valor de **minReplicas** não requer alteração. ● O valor de **replicas** não requer alteração.
Cenário 3	minReplicas < **targetReplicas** < replicas ≤ maxReplicas	4	5	1/10	HPA: 4/10 Implementação: 5	Quando o valor de **targetReplicas** for maior que o de **minReplicas** e menor que o de **replicates**: ● Altere o valor de **minReplicas**. ● O valor de **replicas** não requer alteração.

Cenário	Descrição do cenário	Cronhpa (targetReplicas)	Implementação (replicas)	HPA (minReplicas / maxReplicas)	Resultados	Descrição da operação
Cenário 4	minReplicas < **targetReplicas** = replicas < maxReplicas	5	5	1/10	HPA: 5/10 Implementação: 5	Quando o valor de **targetReplicas** for maior que o de **minReplicas** e igual ao de **replicates**: ● Altere o valor de **minReplicas**. ● O valor de **replicas** não requer alteração.
Cenário 5	minReplicas ≤ replicas < **targetReplicas** < maxReplicas	6	5	1/10	HPA: 6/10 Implementação: 6	Quando o valor de **targetReplicas** for maior que o de **replicates** e menor que o de **maxReplicas**: ● Altere o valor de **minReplicas**. ● Altere o valor de **replicas**.
Cenário 6	minReplicas ≤ replicas < **targetReplicas** = maxReplicas	10	5	1/10	HPA: 10/10 Implementação: 10	Quando o valor de **targetReplicas** for maior que o de **replicates** e igual ao de **maxReplicas**: ● Altere o valor de **minReplicas**. ● Altere o valor de **replicas**.
Cenário 7	minReplicas ≤ replicas ≤ maxReplicas < **targetReplicas**	11	5	5/10	HPA: 11/11 Implementação: 11	Quando o valor de **targetReplicas** é maior que o de **maxReplicas**: ● Altere o valor de **minReplicas**. ● Altere o valor de **maxReplicas**. ● Altere o valor de **replicas**.

----Fim

Usar CronHPA para ajustar diretamente o número de pods de Implementação

CronHPA ajusta as Implementações associadas separadamente para ajustar periodicamente o número de pods de Implementação. O método é o seguinte:

```
apiVersion: autoscaling.cce.io/v2alpha1
kind: CronHorizontalPodAutoscaler
metadata:
  name: ccetest
  namespace: default
spec:
   scaleTargetRef:             # Associate a Deployment.
      apiVersion: apps/v1
      kind: Deployment
      name: nginx
   rules:
   - ruleName: "scale-down"
     schedule: "08 * * * *"    # Running time and period of a job. For details,
see Cron, for example, 0 * * * * or @hourly.
     targetReplicas: 1
     disable: false
   - ruleName: "scale-up"
     schedule: "05 * * * *"
     targetReplicas: 3
     disable: false
```

12.2.4 Criação de uma política CustomedHPA para o dimensionamento automático da carga de trabalho

Uma política CustomedHPA dimensiona as Implementações com base em métricas (como uso da CPU e uso da memória) ou em um intervalo periódico (um ponto de tempo específico todos os dias, todas as semanas, todos os meses ou todos os anos). Esse tipo de política é um recurso de dimensionamento automático aprimorado do CCE.

Funções suportadas:

- O dimensionamento pode ser realizado com base na porcentagem do número atual de pods.
- A etapa mínima de dimensionamento pode ser definida.
- Diferentes operações de escala podem ser executadas com base nos valores métricos reais.

Pré-requisitos

Para usar uma política CustomedHPA, você deve instalar o complemento **cce-hpa-controller**. Se a versão do cce-hpa-controller for anterior a 1.2.11, o complemento **Prometheus** deve ser instalado. Se a versão do cce-hpa-controller for 1.2.11 ou posterior, os complementos que podem fornecer a API de métricas devem ser instalados. Selecione um dos seguintes complementos com base na versão do cluster e nos requisitos reais.

- **Kubernetes Metrics Server**: fornece métricas básicas de uso de recursos, como uso de CPU e memória do contêiner. É suportado por todas as versões de cluster.
- **Monitoramento de cluster da nuvem nativa**: fornece métricas personalizadas além das métricas básicas de recursos. Registre o Prometheus como o serviço que fornece a API

de métricas. Para mais detalhes, consulte **Fornecer métricas de recursos por meio de Metrics API**. Este suplemento suporta clusters v1.17 ou posterior.

- **Prometheus (EOM)**: fornece métricas personalizadas além das métricas básicas de recursos. Registre o Prometheus como o serviço que fornece a API de métricas. Para mais detalhes, consulte **Fornecer métricas de recursos por meio de Metrics API**. Este suplemento suporta apenas clusters de v1.21 ou anterior.

Restrições

- As políticas CustomedHPA podem ser criadas apenas para clusters de v1.15 ou posterior.
- Para clusters anteriores à v1.19.10, se uma política HPA for usada para expandir uma carga de trabalho com volumes do EVS montados, os pods existentes não poderão ser lidos ou gravados quando um novo pod for agendado para outro nó.

  Para clusters de v1.19.10 e posterior, se uma política HPA for usada para expandir uma carga de trabalho com o volume do EVS montado, um novo pod não poderá ser iniciado porque os discos EVS não podem ser anexados.
- As especificações do cce-hpa-controller são decididas pelo número total de contêineres no cluster e pelo número de políticas de escala. É aconselhável configurar 500m de CPU e 1.000 MiB de memória para cada 5.000 contêineres e 100m de CPU e 500 MiB de memória para cada 1.000 políticas de dimensionamento.

Procedimento

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster.

Passo 2 Escolha **Auto Scaling** no painel de navegação e clique na guia **CustomedHPA Policy**.

- Se **Uninstalled** for exibido ao lado do nome do complemento, clique em **Install**, defina os parâmetros do complemento conforme necessário e clique em **Install** para instalar o complemento.
- Se **Installed** for exibido ao lado do nome do complemento, o complemento foi instalado.

Passo 3 Depois que o complemento for instalado, clique em **Create CustomedHPA Policy** no canto superior direito.

Passo 4 Defina parâmetros de política.

Tabela 12-6 Parâmetros de política CustomedHPA

Parâmetro	Descrição
Policy Name	Nome da política a ser criada. Defina este parâmetro conforme necessário.
Namespace	Namespace ao qual a carga de trabalho pertence.
Associated Workload	Carga de trabalho com a qual a política CustomedHPA está associada.
Pod Range	Número mínimo e máximo de pods. Quando uma política é acionada, os pods de carga de trabalho são escalados dentro desse intervalo.

Parâmetro	Descrição
Cooldown Period	Introduza um intervalo, em minutos. Este parâmetro indica o intervalo entre operações consecutivas de dimensionamento. O período de resfriamento garante que uma operação de dimensionamento seja iniciada somente quando a anterior for concluída e o sistema estiver funcionando de forma estável.
Rules	Clique em +. Na caixa de diálogo exibida, defina os seguintes parâmetros: ● **Name**: insira um nome de regra personalizado. ● **Type**: você pode selecionar **Metric-based** (Tabela 12-7) ou **Periodic** (Tabela 12-8). Em seguida, configure as condições e ações do gatilho. ● **Enable**: habilitar ou desabilitar a regra de política. Depois de configurar os parâmetros anteriores, clique em **OK**. Em seguida, a regra de política adicionada é apresentada na lista de regras.

Tabela 12-7 Regras baseadas em métricas

Parâmetro	Descrição
Trigger	Selecione **CPU usage** ou **Memory usage**, escolha **>** ou **<** e insira uma porcentagem. **NOTA** Uso = CPUs ou memória usadas por pods/CPUs ou memória solicitadas.

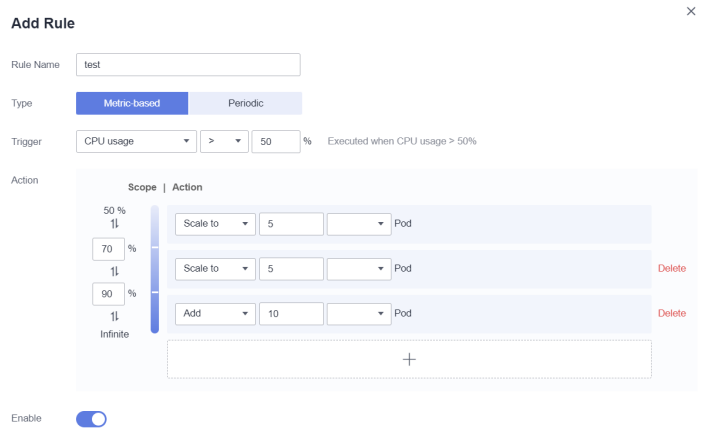
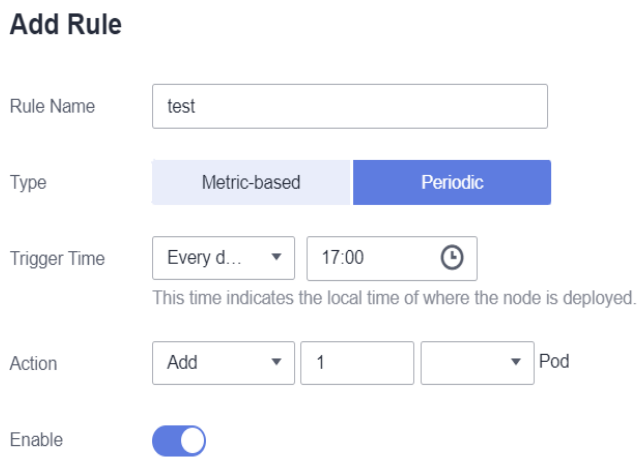
Parâmetro	Descrição
Action	Defina uma ação a ser executada quando a condição de gatilho for atendida. Várias ações podem ser adicionadas. ● **Scale To**: ajuste o número de pods para o valor especificado. Tanto um número quanto uma porcentagem servirão. Esta ação pode ser usada para reduzir ou expandir pods. Se o número atual de pods for menor que o valor alvo ou a porcentagem alvo for maior que 100%, o número de pods será expandido para o valor alvo. Se o número atual de pods for maior que o valor alvo ou se a porcentagem alvo for menor que 100%, o número de pods será reduzido para o valor alvo. ● **Add**: configure este parâmetro quando **Trigger** estiver definido como >. Adicione o número de pods. Você pode especificar um número ou uma porcentagem. Esta ação só pode ser usada para expandir pods. ● **Reduce**: configure este parâmetro quando **Trigger** estiver definido como <. Reduza o número de pods. Você pode especificar um número ou uma porcentagem. Esta ação só pode ser usada para reduzir pods. **NOTA** Você pode inserir um número ou uma porcentagem para as ações anteriores. Ao inserir uma porcentagem, você deve especificar o número mínimo de pods disponíveis. Número final de pods = número de pods atuais x porcentagem. O resultado é arredondado. Se o resultado for menor que o número mínimo de pods disponíveis, o valor predefinido será usado. Caso contrário, o resultado do cálculo é usado. Conforme mostrado abaixo, quando o uso da CPU excede 50%, o número de pods é dimensionado para 5. Quando o uso da CPU excede 70%, o número de pods é ampliado para 8. Quando o uso da CPU excede 90%, o número de pods é ampliado para 18 (adicionando mais 10 pods). Essas regras também funcionam para operações de escala. **Figura 12-3** Definir uma condição de gatilho 

Tabela 12-8 Regras baseadas em periódicos

Parâmetro	Descrição
Trigger Time	Você pode selecionar um horário específico todos os dias, todas as semanas, todos os meses ou todos os anos.
Action	Defina uma ação a ser executada no **Triggered Time**. Como mostrado abaixo, um pod será adicionado às 17:00 todos os dias. ● **Scale To**: ajuste o número de pods para o valor especificado. Tanto um número quanto uma porcentagem servirão. Esta ação pode ser usada para reduzir ou expandir pods. Se o número atual de pods for menor que o valor alvo ou a porcentagem alvo for maior que 100%, o número de pods será expandido para o valor alvo. Se o número atual de pods for maior que o valor alvo ou se a porcentagem alvo for menor que 100%, o número de pods será reduzido para o valor alvo. ● **Add**: adicione o número de pods. Você pode especificar um número ou uma porcentagem. Esta ação só pode ser usada para expandir pods. ● **Reduce**: reduza o número de pods. Você pode especificar um número ou uma porcentagem. Esta ação só pode ser usada para reduzir pods. **NOTA** Você pode inserir um número ou uma porcentagem para as ações anteriores. Ao inserir uma porcentagem, você deve especificar o número mínimo de pods disponíveis. Número final de pods = número de pods atuais x porcentagem. O resultado é arredondado. Se o resultado for menor que o número mínimo de pods disponíveis, o valor predefinido será usado. Caso contrário, o resultado do cálculo é usado. **Figura 12-4** Disparo periódico (diário) 

Passo 5 Clique em **Create**.

----Fim

Usar o kubectl

Uma política CustomHPA é uma CustomResourceDefinition (CRD) e pode ser definida da seguinte forma no YAML:

```
apiVersion: autoscaling.cce.io/v1alpha1
kind: CustomedHorizontalPodAutoscaler
metadata:
  name: customhpa-example
  namespace: default
spec:
  coolDownTime: 3m             #Cooldown period
  maxReplicas: 10              # Maximum number of pods
  minReplicas: 1               # Minimum number of pods
  rules:
    - actions:                           #Policy rules
        - metricRange: 0,0.1             # Metric range, from 0 to 10%
          operationType: ScaleDown       # Scaling type. ScaleDown indicates
downsizing.
          operationUnit: Task            #Operation unit. Task indicates the
number of tasks.
          operationValue: 1              # Resource quantity in each scaling
        - metricRange: 0.1,0.3           # Metric range, from 10% to 30%
          operationType: ScaleDown
          operationUnit: Task
          operationValue: 2
      disable: false
      metricTrigger:
        hitThreshold: 1
        metricName: CPURatioToRequest    # Metric name. CPURatioToRequest
indicates the CPU usage.
        metricOperation: <               # Metric expression operator
        metricValue: 0.3                 # Value on the right of the metric
expression
        periodSeconds: 60                #
        statistic: instantaneous         #
      ruleName: low
      ruleType: Metric
    - actions:
        - metricRange: 0.7,0.9
          operationType: ScaleUp
          operationUnit: Task
          operationValue: 1
        - metricRange: 0.9,+Infinity
          operationType: ScaleUp
          operationUnit: Task
          operationValue: 2
      disable: false
      metricTrigger:
        hitThreshold: 1
        metricName: CPURatioToRequest
        metricOperation: '>'
        metricValue: 0.7
        periodSeconds: 60
        statistic: instantaneous
      ruleName: high
      ruleType: Metric
  scaleTargetRef:                  # Associated workload
    apiVersion: apps/v1
    kind: Deployment
    name: nginx
```


12.2.5 Gerenciamento de políticas de dimensionamento de carga de trabalho

Cenário

Depois que uma política HPA ou CustomedHPA é criada, você pode atualizar, clonar, editar e excluir a política, bem como editar o arquivo YAML.

Verificar uma política HPA

Você pode exibir as regras, o status e os eventos de uma política HPA e manipular exceções com base nas informações de erro exibidas.

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster.

Passo 2 No painel de navegação, escolha **Policies**. Na página de guia **HPA Policies**, clique em ao lado da política HPA de destino.

Passo 3 Na área expandida, escolha **View Events** na coluna **Operation**. Se a política funcionar mal, localize e corrija a falha com base na mensagem de erro exibida na página.

NOTA

Você também pode exibir a política HPA criada na página de detalhes da carga de trabalho.

1. Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster.
2. No painel de navegação, escolha **Workloads**. Clique no nome da carga de trabalho para exibir seus detalhes.
3. Na página de detalhes da carga de trabalho, alterne para a página de guia **Auto Scaling** para exibir as políticas HPA ou CustomedHPA. Você também pode visualizar as políticas de dimensionamento configuradas em **Workload Scaling**.

Tabela 12-9 Tipos e nomes de eventos

Tipo de evento	Nome do evento	Descrição
Normal	SuccessfulRescale	O dimensionamento é realizado com sucesso.
Anormal	InvalidTargetRange	Intervalo do alvo inválido.
	InvalidSelector	Seletor inválido.
	FailedGetObjectMetric	Os objetos não conseguem ser obtidos.
	FailedGetPodsMetric	Pods não conseguem ser obtidos.
	FailedGetResourceMetric	Os recursos não conseguem ser obtidos.
	FailedGetExternalMetric	Métricas externas não conseguem ser obtidas.

Tipo de evento	Nome do evento	Descrição
	InvalidMetricSourceType	Tipo de origem da métrica inválido.
	FailedConvertHPA	Falhou na conversão de HPA.
	FailedGetScale	Não é possível obter a escala.
	FailedComputeMetricsReplicas	Falhou ao calcular réplicas definidas por métricas.
	FailedGetScaleWindow	Falhou ao obter ScaleWindow.
	FailedRescale	Falhou ao dimensionar o serviço.

----Fim

Exibição de uma política CustomedHPA

Você pode exibir as regras e o status mais recente de uma política CustomedHPA e corrigir falhas com base nas informações de erro exibidas.

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster.

Passo 2 No painel de navegação, escolha **Workload Scaling**. Na página de guia **CustomHPA Policies**, clique em ⌄ ao lado da política CustomHPA de destino.

Passo 3 Na área expandida, se a política for anormal na página de guia **Rules**, clique em **Details** em **Latest Status** e localize a falha com base nas informações exibidas.

NOTA

Você também pode exibir a política HPA criada na página de detalhes da carga de trabalho.

1. Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster.
2. No painel de navegação, escolha **Workloads**. Clique no nome da carga de trabalho para exibir seus detalhes.
3. Na página de detalhes da carga de trabalho, alterne para a página de guia **Auto Scaling** para exibir as políticas HPA ou CustomedHPA. Você também pode visualizar as políticas de dimensionamento configuradas em **Workload Scaling**.

----Fim

Atualização de uma política HPA ou CustomedHPA

Uma política HPA é usada como exemplo.

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster.

Passo 2 No painel de navegação, escolha **Workload Scaling**. Clique em **Update** na coluna **Operation** da política de destino.

Passo 3 Na página **Update HPA Policy** exibida, defina os parâmetros de política listados em **Tabela 12-4**.

Passo 4 Clique em **Update**.

----Fim

Editação do arquivo YAML (Política HPA)

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster.

Passo 2 No painel de navegação, escolha **Policies**. Escolha **Edit YAML** na coluna **Operation** da política HPA de destino.

Passo 3 Na caixa de diálogo **Edit YAML** exibida, edite ou faça download do arquivo YAML.

Passo 4 Clique no botão de fechar no canto superior direito.

----Fim

Exibição do arquivo YAML (Política CustomedHPA)

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster.

Passo 2 No painel de navegação, escolha **Workload Scaling**. Clique em **View YAML** na coluna **Operation** da política CustomedHPA de destino.

Passo 3 Na caixa de diálogo exibida, você pode copiar e fazer download do arquivo YAML, mas não pode modificá-lo.

Passo 4 Clique no botão de fechar no canto superior direito.

----Fim

Exclusão de uma política HPA ou CustomedHPA

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster.

Passo 2 No painel de navegação, escolha **Policies**. Escolha **Delete** > **Delete** na coluna **Operation** da política de destino.

Passo 3 Na caixa de diálogo exibida, clique em **Yes**.

----Fim

12.3 Dimensionamento de um nó

12.3.1 Mecanismos de dimensionamento de nós

O HPA foi projetado para dimensionamento em nível de pod e pode ajustar dinamicamente o número de réplicas com base em métricas de carga de trabalho. No entanto, se os recursos do cluster forem insuficientes e as novas réplicas não puderem ser executadas, você só poderá expandir o cluster.

autoscaler é um componente de dimensionamento automático fornecido pelo Kubernetes. Ele expanda ou reduze automaticamente nós em um cluster com base no status de agendamento

do pod e no uso de recursos. Ele suporta vários modos de dimensionamento, como multi-AZ, multi-pod-especificações, disparo de métricas e disparo periódico, para atender aos requisitos de diferentes cenários de dimensionamento de nós.

Pré-requisitos

Antes de usar a função de dimensionamento de nó, deve instalar o complemento **autoscaler** da v1.13.8 ou posterior.

Como funciona o autoscaler

O autoscaler passa por dois processos.

- Expansão: o autoscaler verifica todos os pods não agendados a cada 10 segundos e seleciona um pool de nós que atenda aos requisitos de expansão com base na política definida.

  NOTA

  Quando o autoscaler verifica pods não agendados em busca de expansões, ele usa o algoritmo de agendamento consistente com a versão da comunidade do Kubernetes para cálculo de agendamento simulado. Se kube-schedulers não embutidas ou outras políticas de agendamento da comunidade não de Kubernetes forem usadas para agendamento de aplicações, quando o autoscaler for usado para expandir a capacidade de tais aplicações, a capacidade pode deixar de ser expandida ou pode ser expandida mais do que o esperado devido a algoritmos de agendamento inconsistentes.

- Redução: o autoscaler verifica todos os nós a cada 10 segundos. Se o número de solicitações de pods em um nó for menor que a porcentagem definida pelo usuário para redução, o autoscaler simula se os pods no nó podem ser migrados para outros nós. Se sim, o nó será removido após uma janela de tempo ocioso.

  Quando um nó de cluster fica ocioso por um período (10 minutos por padrão), o dimensionamento de cluster é acionado e o nó é excluído automaticamente. No entanto, um nó não pode ser excluído de um cluster se os seguintes pods existirem:

  – Pods que não atendem aos requisitos específicos definidos nos Orçamentos de interrupção de pods (**PodDisruptionBudget**)
  – Pods que não podem ser programados para outros nós devido a restrições, como políticas de afinidade e antiafinidade
  – Pods que têm a anotação **cluster-autoscaler.kubernetes.io/safe-to-evict: 'false'**
  – Pods (exceto aqueles criados por DaemonSets no namespace do kube-system) que existem no namespace do kube-system no nó
  – Pods que não são criados pelo controlador (Implementação/ReplicaSet/tarefa/StatefulSet)

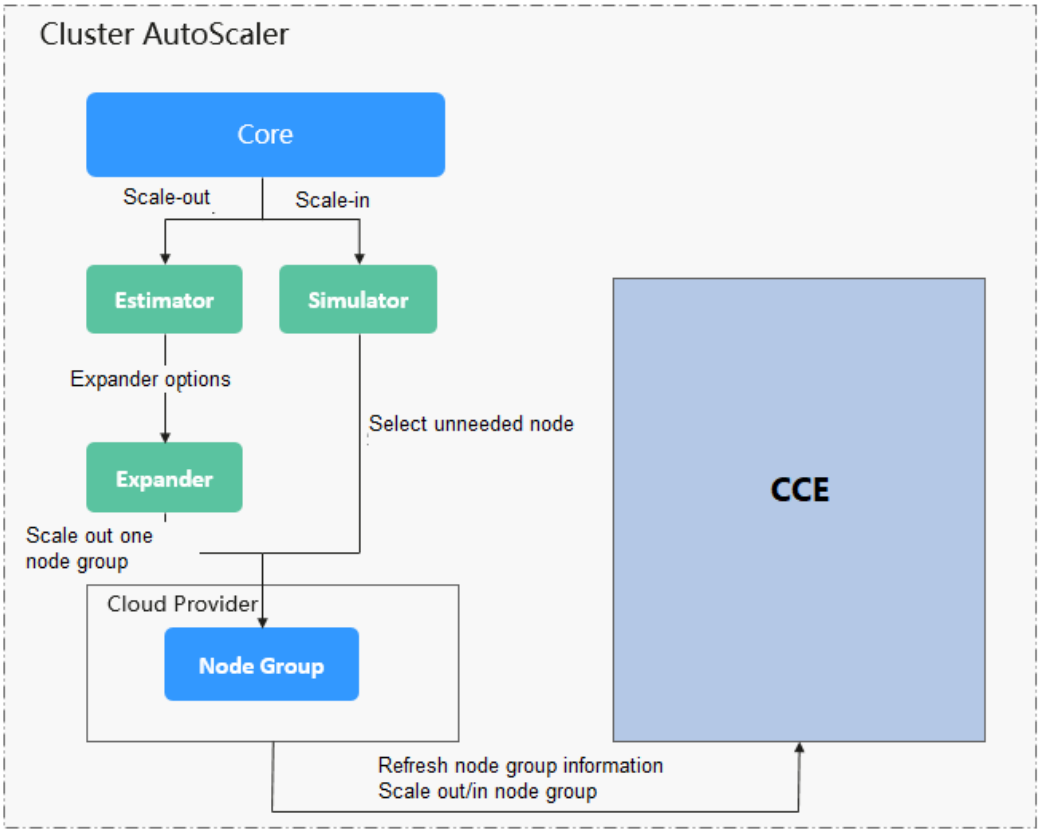
  NOTA

  Quando um nó atende às condições de dimensionamento, o Autoscaler adiciona a mancha **DeletionCandidateOfClusterAutoscaler** ao nó com antecedência para evitar que pods sejam programados para o nó. Após a desinstalação do complemento Autoscaler, se a mancha ainda existir no nó, exclua-a manualmente.

Arquitetura do autoscaler

Figura 12-5 mostra a arquitetura do autoscaler e seus módulos principais:

Figura 12-5 Arquitetura do autoscaler



Descrição

- **Estimator**: avalia o número de nós a serem adicionados a cada pool de nós para hospedar pods não agendáveis.
- **Simulator**: localiza os nós que atendem às condições de reduzir no cenário de redução.
- **Expander**: seleciona um nó ideal do pool de nós selecionado pelo estimador com base na política definida pelo usuário no cenário de expansão. Atualmente, o Expander possui as seguintes políticas:

Tabela 12-10 Políticas de Expander apoiadas pelo CCE

Política	Descrição	Cenário de aplicação	Exemplo
Random	Seleciona aleatoriamente um pool de nós programável para executar a expansão.	Esta política é normalmente utilizada como uma cópia de segurança básica para outras políticas complexas. Utilize esta política apenas se as outras políticas não puderem ser utilizadas.	Suponha que o dimensionamento automático esteja ativado para os pools de nós 1 e 2 no cluster e que o limite superior de expansão não seja atingido. A política para dimensionar o número de réplicas de uma carga de trabalho é a seguinte: 1. Os pods pendentes acionam o autoscaler para determinar o processo de expansão. 2. O autoscaler simula a fase de agendamento e avalia que os pods pendentes podem ser programados para os nós adicionados em ambos os pools de nós 1 e 2. 3. O autoscaler seleciona aleatoriamente o pool de nós 1 ou o pool de nós 2 para expansão.

Política	Descrição	Cenário de aplicação	Exemplo
most-pods	Uma política combinada. Tem precedência sobre a política aleatória. Seleciona preferencialmente o pool de nós que pode agendar a maioria dos pods após a expansão. Se vários pools de nós atenderem à condição, a política random será usada para tomar decisões adicionais.	Esta política baseia-se no número máximo de pods que podem ser agendados.	Suponha que o dimensionamento automático esteja ativado para os pools de nós 1 e 2 no cluster e que o limite superior de dimensionamento não seja atingido. A política para dimensionar o número de réplicas de uma carga de trabalho é a seguinte: 1. Os pods pendentes acionam o autoscaler para determinar o processo de expansão. 2. O autoscaler simula a fase de agendamento e avalia que alguns pods pendentes podem ser agendados para os nós adicionados nos pools de nós 1 e 2. 3. O autoscaler avalia que o pool de nós 1 pode agendar 20 novos pods e o pool de nós 2 pode agendar apenas 10 novos pods após a expansão. Portanto, o autoscaler seleciona o pool de nós 1 para dimensionamento.

Política	Descrição	Cenário de aplicação	Exemplo
least-waste	Uma política combinada. Tem precedência sobre a política aleatória. O autoscaler avalia a taxa geral de alocação de CPU ou memória dos pools de nós e seleciona o pool de nós com o mínimo de desperdício de CPU ou memória. Se vários pools de nós atenderem à condição, a política random será usada para tomar decisões adicionais.	Esta política usa a pontuação mínima de resíduo de recursos de CPU ou memória como critério de seleção. A fórmula para calcular a pontuação mínima de resíduos (wastedScore) é a seguinte: ● wastedCPU = (número total de CPUs dos nós a serem expandidos – número total de CPUs dos pods a serem agendados)/número total de CPUs dos nós a serem expandidos ● wastedMemory = (tamanho total de memória dos nós a serem expandidos – tamanho total de memória dos pods a serem agendados)/ tamanho total de memória dos nós a serem expandidos ● wastedScore = wastedCPU + wastedMemory	Suponha que o dimensionamento automático esteja ativado para os pools de nós 1 e 2 no cluster e que o limite superior de dimensionamento não seja atingido. A política para dimensionar o número de réplicas de uma carga de trabalho é a seguinte: 1. Os pods pendentes acionam o autoscaler para determinar o processo de expansão. 2. O autoscaler simula a fase de agendamento e avalia que alguns pods pendentes podem ser agendados para os nós adicionados nos pools de nós 1 e 2. 3. O autoscaler avalia que a pontuação mínima de desperdício do pool de nós 1 após a expansão é menor do que a do pool de nós 2. Portanto, o autoscaler seleciona o pool de nós 1 para expansão.

Política	Descrição	Cenário de aplicação	Exemplo
priority	Uma política combinada. As prioridades para as políticas são as seguintes: priority > least-waste > random. É uma política least-waste aprimorada configurada com base no pool de nós ou na prioridade do grupo de dimensionamento. Se vários pools de nós atenderem à condição, a política least-waste será usada para a tomada de decisões posteriores.	Essa política permite configurar e gerenciar as prioridades de pools de nós ou grupos de dimensionamento por meio do console ou da API, enquanto a política least-waste pode reduzir a taxa de desperdício de recursos em cenários comuns. Essa política tem uma aplicabilidade mais ampla e é usada como a política de seleção padrão.	Suponha que o dimensionamento automático esteja ativado para os pools de nós 1 e 2 no cluster e que o limite superior de expansão não seja atingido. A política para dimensionar o número de réplicas de uma carga de trabalho é a seguinte: 1. Os pods pendentes acionam o autoscaler para determinar o processo de expansão. 2. O autoscaler simula a fase de agendamento e avalia que alguns pods pendentes podem ser agendados para os nós adicionados nos pools de nós 1 e 2. 3. O autoscaler avalia que o pool de nós 1 tem uma prioridade mais alta do que o pool de nós 2. Portanto, o autoscaler seleciona o pool de nós 1 para expansão.

12.3.2 Criação de uma política de dimensionamento de nós

O CCE fornece dimensionamento de nó por meio do complemento **autoscaler**. Os nodes com especificações diferentes podem ser adicionados automaticamente em AZs sob demanda.

Se uma política de dimensionamento de nó e a configuração no complemento autoscaler entrarem em vigor ao mesmo tempo, por exemplo, há pods que não podem ser agendados e o valor de uma métrica atinge o limite ao mesmo tempo, expansão é realizada primeiro para os pods não programáveis.

- Se a expansão for bem-sucedida para os pods não programáveis, o sistema ignora a lógica de regra baseada em métricas e entra no próximo loop.
- Se a expansãot falhar para os pods não programáveis, a regra baseada em métrica será executada.

Pré-requisitos

Antes de usar a função de dimensionamento de nó, instale o complemento **autoscaler** da v1.13.8 ou posterior no cluster.

Restrições

- Somente os pools de nós de pagamento por uso suportam o dimensionamento automático.
- As políticas de dimensionamento automático se aplicam aos pools de nós. Quando o número de nós em um pool de nós é 0 e a política de dimensionamento é baseada no uso da CPU ou da memória, o escalonamento de nós não é acionado.
- A redução de nó causará perda de dados de PVC/PV para os **PVs locais** associado ao nó. Esses PVCs e PVs não podem ser restaurados ou usados novamente. Numa redução de nó, o pod que usa o PV local é despejado do nó. Um novo pod é criado e permanece no estado pendente. Isso ocorre porque a PVC usada pelo pod tem um rótulo de nó, devido ao qual o pod não pode ser programado.

Procedimento

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster.

Passo 2 Escolha **Node Scaling** no painel de navegação.

- Se **Uninstalled** for exibido ao lado do nome do complemento, clique em **Install**, defina os parâmetros do complemento conforme necessário e clique em **Install** para instalar o complemento.
- Se **Installed** for exibido ao lado do nome do complemento, o complemento foi instalado.

Passo 3 Clique em **Create Node Scaling Policy** no canto superior direito e defina os parâmetros da seguinte forma:

- **Policy Name**: nome da política a ser criada, que pode ser personalizada.
- **Associated Node Pools**: selecione o pool de nós a ser associado. Você pode associar vários pools de nós para usar a mesma política de dimensionamento.
- **Rules**: clique em **Add Rule**. Na caixa de diálogo exibida, defina os seguintes parâmetros:

  Rule Name: insira o nome de uma regra.

  Rule Type: você pode selecionar **Metric-based** ou **Periodic**. As diferenças entre os dois tipos são as seguintes:

  – **Metric-based**:

    Condition: selecione **CPU allocation rate** ou **Memory allocation rate** e insira um valor. O valor deve ser maior do que a porcentagem de dimensionamento configurada no complemento autoscaler.

NOTA

- Alocação de recursos (%) = recursos solicitados pelos pods no pool de nós/recursos alocáveis aos pods no pool de nós
- **Se várias regras atenderem às condições, as regras serão executadas em um dos seguintes modos:**

  Se as regras baseadas na **CPU allocation rate** e na **memory allocation rate** forem configuradas e duas ou mais regras atenderem às condições de expansão, a regra que adicionará a maioria dos nós será executada.

  Se uma regra baseada na **CPU allocation rate** e **uma regra periódica** forem configuradas e ambas atenderem às condições de expansão, uma delas será executada aleatoriamente. A regra executada primeiro (regra A) altera o pool de nós para o estado de escala. Como resultado, a outra regra (regra B) não pode ser executada. Depois que a regra A for executada e o status do pool de nós se tornar normal, a regra B não será executada.
- Se regras baseadas na **CPU allocation rate** e na **memory allocation rate** estiverem configuradas, o período de detecção de política varia de acordo com a lógica de processamento de cada loop do complemento autoscaler. A expansão é acionada quando as condições são atendidas, mas é limitada por outros fatores, como o intervalo de resfriamento e o status do pool de nós.

– **Periodic**:

  Trigger Time: você pode selecionar um ponto de tempo específico todos os dias, todas as semanas, todos os meses ou todos os anos.

Action: defina uma ação a ser executada quando a condição de gatilho for atendida.

Você pode clicar em **Add Rule** para adicionar mais políticas de dimensionamento de nó. Você pode adicionar no máximo uma regra baseada em uso de CPU e uma regra baseada em uso de memória. O número total de regras não pode exceder 10.

Passo 4 Clique em **OK**.

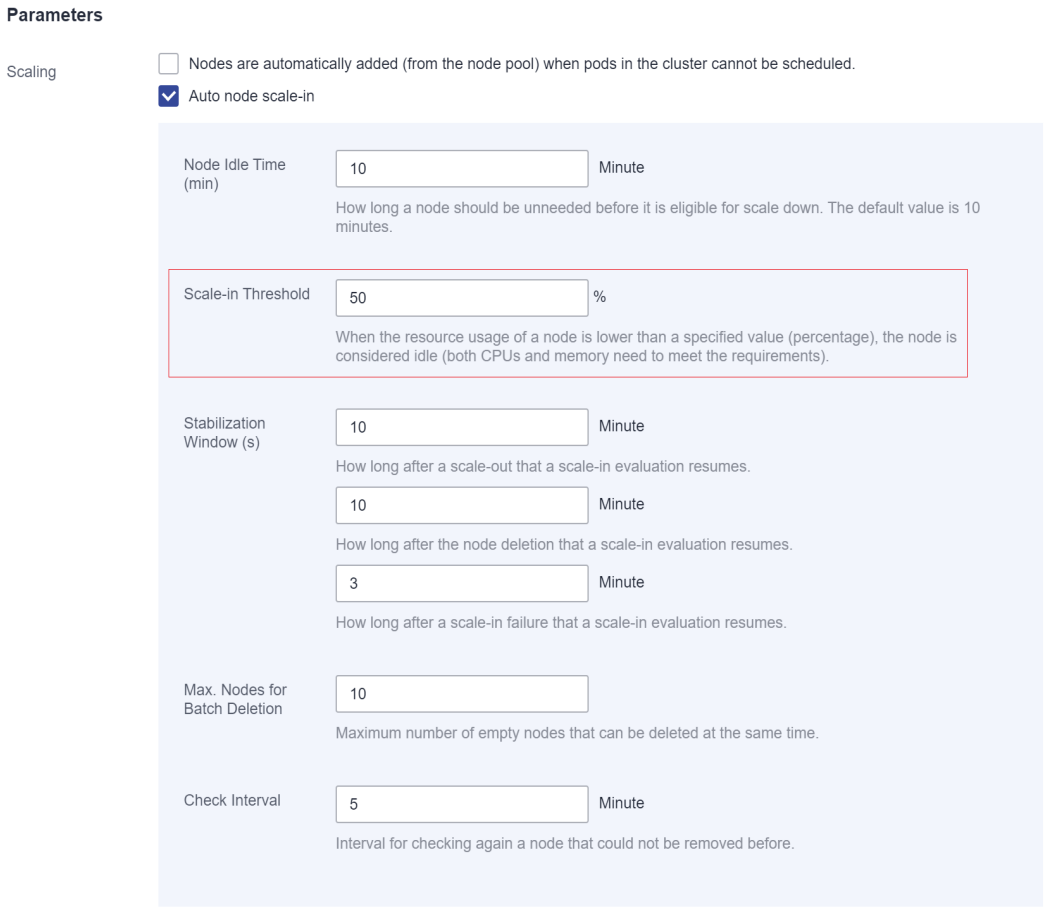
----Fim

Restrições em redução

Você pode definir políticas de expansão do nó somente ao instalar o **complemento autoscaler**.

A redução de nó pode ser acionada somente pela taxa de alocação de recursos. Quando as taxas de alocação de CPU e memória em um cluster são menores do que os limites especificados (definidos quando o complemento autoscaler é instalado ou modificado), a redução é acionada para os nós no pool de nós (essa função pode ser desativada), conforme mostrado na **Figura 12-6**

Figura 12-6 Configuração de redução automática



Exemplo YAML

Veja a seguir um exemplo YAML de uma política de dimensionamento de nó:

```
apiVersion: autoscaling.cce.io/v1alpha1
kind: HorizontalNodeAutoscaler
metadata:
  creationTimestamp: "2020-02-13T12:47:49Z"
  generation: 1
  name: xxxx
  namespace: kube-system
  resourceVersion: "11433270"
  selfLink: /apis/autoscaling.cce.io/v1alpha1/namespaces/kube-system/
horizontalnodeautoscalers/xxxx
  uid: c2bd1e1d-60aa-47b5-938c-6bf3fadbe91f
spec:
  disable: false
  rules:
  - action:
      type: ScaleUp
      unit: Node
      value: 1
    cronTrigger:
      schedule: 47 20 * * *
    disable: false
    ruleName: cronrule
    type: Cron
  - action:
      type: ScaleUp
      unit: Node
```



```
    value: 2
  disable: false
  metricTrigger:
    metricName: Cpu
    metricOperation: '>'
    metricValue: "40"
    unit: Percent
  ruleName: metricrule
  type: Metric
targetNodepoolIds:
- 7d48eca7-3419-11ea-bc29-0255ac1001a8
```

Tabela 12-11 Parâmetros principais

Parâmetro	Tipo	Descrição
spec.disable	Bool	Se ativar a política de dimensionamento. Este parâmetro tem efeito para todas as regras na política.
spec.rules	Array	Todas as regras em uma política de dimensionamento.
spec.rules[x].ruleName	String	Nome da regra.
spec.rules[x].type	String	Tipo da regra. Atualmente, **Cron** e **Metric** são suportados.
spec.rules[x].disable	Bool	Troca de regra. Atualmente, apenas **false** é suportado.
spec.rules[x].action.type	String	Tipo de ação de regra. Atualmente, apenas **ScaleUp** é suportado.
spec.rules[x].action.unit	String	Unidade de ação de regra. Atualmente, apenas **Node** é suportado.
spec.rules[x].action.value	Integer	Valor da ação da regra.
spec.rules[x].cronTrigger	/	Opcional. Este parâmetro é válido apenas em regras periódicas.
spec.rules[x].cronTrigger.schedule	String	Expressão Cron de uma regra periódica.
spec.rules[x].metricTrigger	/	Opcional. Esse parâmetro é válido somente em regras baseadas em métricas.
spec.rules[x].metricTrigger.metricName	String	Métrica de uma regra baseada em métricas. Atualmente, **Cpu** e **Memory** são suportados.
spec.rules[x].metricTrigger.metricOperation	String	Operador de comparação de uma regra baseada em métricas. Atualmente, apenas **>** é suportado.

Parâmetro	Tipo	Descrição
spec.rules[x].metricTrigger.metricValue	String	Limite métrico de uma regra baseada em métricas. O valor pode ser qualquer número inteiro de 1 a 100 e deve ser uma cadeia de caracteres.
spec.rules[x].metricTrigger.Unit	String	Unidade do limite de regra baseado em métricas. Atualmente, apenas **%** é suportado.
spec.targetNodepoolIds	Array	Todos os pools de nós associados à política de dimensionamento.
spec.targetNodepoolIds[x]	String	ID do pool de nós associado à política de dimensionamento.

12.3.3 Gerenciamento de políticas de dimensionamento de nós

Cenário

Depois que uma política de dimensionamento de nó é criada, você pode excluir, editar, desativar, ativar ou clonar a política.

Exibir uma política de dimensionamento de nó

Você pode exibir o pool de nós associado, as regras e o histórico de dimensionamento de uma política de escala de nó e corrigir falhas de acordo com as informações de erro exibidas.

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster.

Passo 2 Escolha **Node Scaling** no painel de navegação e clique em ⌄ na frente da política a ser visualizada.

Passo 3 Na área expandida, as páginas de guia **Associated Node Pools**, **Rules** e **Scaling History** são exibidas. Se a política for anormal, localize e retifique a falha com base nas informações de erro.

NOTA

Você também pode desativar ou ativar o dimensionamento automático na página **Node Pools**.

1. Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster.
2. No painel de navegação, escolha **Nodes** e alterne para a página de guia **Node Pools**.
3. Localize a linha que contém o pool de nós de destino e escolha **More** > **Configure Auto Scaling**. Na caixa de diálogo **Configure Auto Scaling**, configure os nós máximos e mínimos, bem como o período de resfriamento.

----Fim

Excluir uma política de dimensionamento de nós

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster.

Passo 2 Escolha **Node Scaling** no painel de navegação e escolha **More** > **Delete** ao lado da política a ser excluída.

Passo 3 Na caixa de diálogo **Delete Node Scaling Policy** apresentada, confirme se pretende eliminar a política.

Passo 4 Clique em **Yes** para excluir a política.

----Fim

Editar uma política de dimensionamento de nós

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster.

Passo 2 Escolha **Node Scaling** no painel de navegação e clique em **Edit** na coluna **Operation** da política a ser editada.

Passo 3 Na página **Edit Node Scaling Policy** exibida, modifique os valores de parâmetro de política listados em **Tabela 12-11**.

Passo 4 Após a conclusão da configuração, clique em **OK**.

----Fim

Clonar uma política de dimensionamento de nós

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster.

Passo 2 Escolha **Node Scaling** no painel de navegação e escolha **More** > **Clone** ao lado da política a ser clonada.

Passo 3 Na página **Clone Node Scaling Policy** exibida, determinados parâmetros foram clonados. Adicione ou modifique outros parâmetros de política com base nos requisitos de serviço.

Passo 4 Clique em **OK**.

----Fim

Ativar ou desativar uma política de dimensionamento de nó

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster.

Passo 2 Escolha **Node Scaling** no painel de navegação e clique em **Disable** na coluna **Operation** da política a ser desativada. Se a política estiver no estado desativado, clique em **Enable** na coluna **Operation** da política.

Passo 3 Na caixa de diálogo exibida, confirme se deseja desativar ou ativar a política de nó.

----Fim

12.4 Uso de HPA e CA para dimensionamento automático de cargas de trabalho e nós

Cenários de aplicações

A melhor maneira de lidar com o aumento do tráfego é ajustar automaticamente o número de máquinas com base no volume de tráfego ou no uso de recursos, o que é chamado de dimensionamento.

Ao implementar aplicações em pods, você pode configurar os recursos solicitados e os limites de recursos para os pods a fim de evitar o uso ilimitado de recursos durante os horários de pico. No entanto, depois que o limite superior for atingido, poderá ocorrer um erro de aplicação. O dimensionamento de pods pode resolver esse problema com eficiência. Se o uso de recursos no nó aumentar até certo ponto, os pods adicionados recentemente não poderão ser agendados para esse nó. Nesse caso, o CCE adicionará nós de acordo.

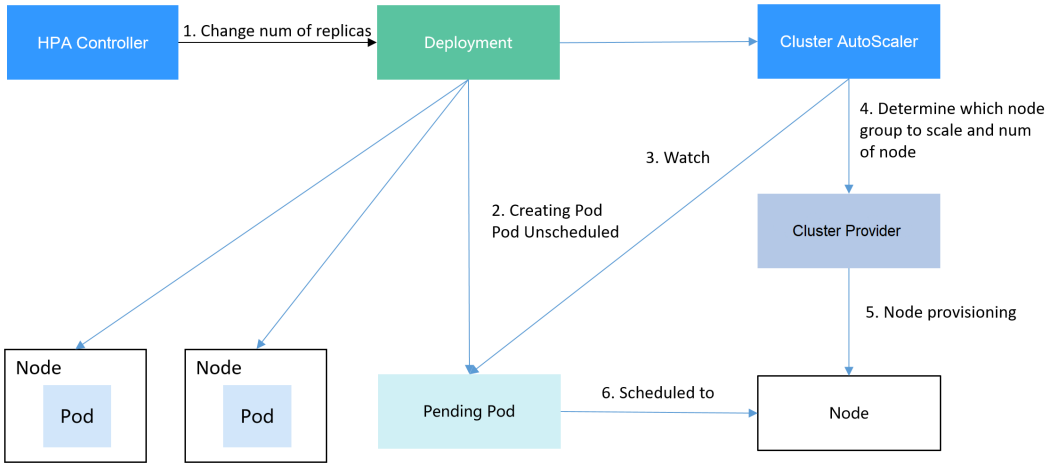
Solução

As duas principais políticas de dimensionamento automático são HPA (Dimensionamento automático de pod horizontal) e CA (Dimensionamento automático de cluster). HPA é para dimensionamento automático de carga de trabalho e CA é para dimensionamento automático de nós.

HPA e CA trabalham juntos. O HPA requer recursos de cluster suficientes para o dimensionamento bem-sucedido. Quando os recursos do cluster são insuficientes, o CA é necessário para adicionar nós. Se o HPA reduzir as cargas de trabalho, o cluster terá um grande número de recursos ociosos. Nesse caso, o CA precisa liberar nós para evitar desperdício de recursos.

Conforme mostrado em **Figura 12-7**, o HPA realiza a expansão com base nas métricas de monitoramento. Quando os recursos do cluster são insuficientes, os pods recém-criados ficam no estado Pending. Em seguida, o CA verifica esses pods pendentes e seleciona o pool de nós mais apropriado com base na política de dimensionamento configurada para dimensionar o pool de nós. Para obter detalhes sobre como o HPA e o CA funcionam, consulte **Mecanismos de dimensionamento de carga de trabalho** e **Mecanismos de dimensionamento de nós**.

Figura 12-7 Fluxos de trabalho de HPA e CA

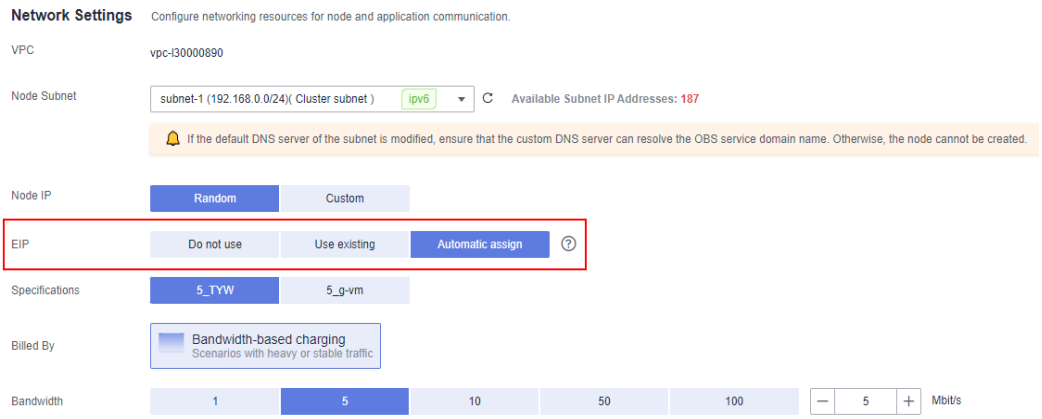


O uso do HPA e CA pode implementar facilmente o dimensionamento automático na maioria dos cenários. Além disso, o processo de dimensionamento de nós e pods pode ser facilmente observado.

Esta seção usa um exemplo para descrever o processo de dimensionamento automático usando as políticas de HPA e CA juntas.

Preparativos

Passo 1 Crie um cluster com um nó. O nó deve ter 2 núcleos de vCPUs e 4 GiB de memória, ou uma especificação superior, bem como um EIP para permitir o acesso externo. Se nenhum EIP estiver vinculado ao nó durante a criação do nó, você poderá vincular um manualmente no console do ECS após a criação do nó.



Passo 2 Instale complementos para o cluster.

- autoscaler: complemento de dimensionamento de nós
- metrics-server: um agregador de dados de uso de recursos em um cluster do Kubernetes. Ele pode coletar dados de medição dos principais recursos do Kubernetes, como pods, nós, contêineres e serviços.

Passo 3 Faça logon no nó de cluster e execute uma aplicação com uso intensivo de computação. Quando um usuário envia uma solicitação, o resultado precisa ser calculado antes de ser retornado ao usuário.

1. Crie um arquivo PHP chamado **index.php** para calcular a raiz quadrada da solicitação 1.000.000 de vezes antes de retornar **OK!**.

   ```
   vi index.php
   ```

   O conteúdo do arquivo é o seguinte:

   ```
   <?php
     $x = 0.0001;
     for ($i = 0; $i <= 1000000; $i++) {
       $x += sqrt($x);
     }
     echo "OK!";
   ?>
   ```

2. Compile um arquivo **Dockerfile** para criar uma imagem.

   ```
   vi Dockerfile
   ```

   O conteúdo é o seguinte:

   ```
   FROM php:5-apache
   COPY index.php /var/www/html/index.php
   RUN chmod a+rx index.php
   ```


3. Execute o seguinte comando para criar uma imagem chamada **hpa-example** com a tag **latest**.
   ```
   docker build -t hpa-example:latest .
   ```
4. (Opcional) Faça logon no console do SWR, escolha **Organizations** no painel de navegação e clique em **Create Organization** no canto superior direito.

   Pule esta etapa se você já tiver uma organização.
5. No painel de navegação, escolha **My Images** e clique em **Upload Through Client**. Na página exibida, clique em **Generate a temporary login command** e clique em para copiar o comando.
6. Execute o comando de logon copiado na etapa anterior no nó do cluster. Se o logon for bem-sucedido, a mensagem "Login Succeeded" será exibida.
7. Marque a imagem hpa-example.

   docker tag *{Image name 1:Tag 1}*/*{Image repository address}*/*{Organization name}*/*{Image name 2:Tag 2}*
   - *{Image name 1:Tag 1}*: nome e tag da imagem local a ser carregada.
   - *{Image repository address}*: o nome de domínio no final do comando de logon no **comando de logon**. Ele pode ser obtido no console do SWR.
   - *{Organization name}*: nome da **organização criada**.
   - *{Image name 2:Tag 2}*: nome e tag da imagem desejada a serem exibidos no console do SWR.

   O seguinte é um exemplo:

   docker tag hpa-example:latest swr.ap-southeast-1.myhuaweicloud.com/cloud-develop/hpa-example:latest
8. Envie a imagem para o repositório de imagens.

   docker push *{Image repository address}*/*{Organization name}*/*{Image name 2:Tag 2}*

   O seguinte é um exemplo:

   docker push swr.ap-southeast-1.myhuaweicloud.com/cloud-develop/hpa-example:latest

   As informações a seguir serão retornadas após um envio bem-sucedido:
   ```
   6d6b9812c8ae: Pushed
   ...
   fe4c16cbf7a4: Pushed
   latest: digest: sha256:eb7e3bbd*** size: **
   ```
   Para visualizar a imagem enviada, acesse o console do SWR e atualize a página **My Images**.

----Fim

Criação de um pool de nós e de uma política de dimensionamento de nós

Passo 1 Faça logon no console do CCE, acesse o cluster criado, clique em **Nodes** à esquerda, clique na guia **Node Pools** e clique em **Create Node Pool** no canto superior direito.

Passo 2 Configure o pool de nós.

- **Nodes**: defina como **1**, indicando que um nó é criado por padrão quando um pool de nós é criado.
- **Specifications**: 2 vCPUs | 4 GiB

Mantenha os padrões para outros parâmetros. Para obter detalhes, consulte **Criação de um pool de nós**.

Passo 3 Localize a linha que contém o pool de nós recentemente criado e clique em **Auto Scaling** no canto superior direito. Para obter detalhes, consulte **Criação de uma política de dimensionamento de nós**.

Se o complemento CCE Cluster Autoscaler não estiver instalado no cluster, instale-o primeiro. Para obter detalhes, consulte **CCE Cluster Autoscaler**.

- **Automatic scale-out**: se essa função estiver ativada, os nós em um pool de nós serão adicionados automaticamente com base na carga do cluster.
- **Customized scale-out rules.**: clique em **Add Rule**. Na caixa de diálogo exibida, configure os parâmetros. Se a taxa de alocação de CPU for maior que 70%, um nó será adicionado a cada pool de nós associado. Uma política de dimensionamento de nós precisa ser associada a um pool de nós. Vários pools de nós podem ser associados. Quando você precisar dimensionar nós, o nó com especificações adequadas será adicionado ou reduzido do pool de nós com base no princípio do desperdício mínimo.
- **Automatic scale-in**: se essa função estiver ativada, os nós em um pool de nós serão excluídos automaticamente com base na carga do cluster. Por exemplo, acione a redução quando a utilização de recursos do nó for inferior a 50%.
- **AS Configuration**: modifique o intervalo de quantidade de nós. Durante o dimensionamento automático, o número de nós em um pool de nós está sempre dentro do intervalo de quantidade configurado.
- **AS Object**: ative o dimensionamento automático para especificações de nós em um pool de nós.

Passo 4 Clique em **OK**.

----Fim

Criação de uma carga de trabalho

Use a imagem hpa-example para criar um Deployment com uma réplica. O caminho da imagem está relacionado à organização carregada no repositório SWR e precisa ser substituído pelo valor real.

```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: hpa-example
spec:
  replicas: 1
  selector:
    matchLabels:
      app: hpa-example
  template:
    metadata:
      labels:
        app: hpa-example
    spec:
      containers:
      - name: container-1
        image: 'hpa-example:latest' # Replace it with the address of the image
you uploaded to SWR.
        resources:
          limits:                  # The value of limits must be the same as that
of requests to prevent flapping during scaling.
            cpu: 500m
            memory: 200Mi
```



```
          requests:
            cpu: 500m
            memory: 200Mi
      imagePullSecrets:
      - name: default-secret
```

Em seguida, crie um Serviço de NodePort para a carga de trabalho para que a carga de trabalho possa ser acessada de redes externas.

NOTA

Para permitir acesso externo aos Serviços de NodePort, aloque um EIP para o nó no cluster. Após a alocação, sincronize os dados do nó. Para obter detalhes, consulte **Sincronização de dados com servidores em nuvem**. Se o nó já estiver vinculado a um EIP, você não precisará criar um.

Como alternativa, você pode criar um Serviço com um balanceador de carga do ELB para acesso externo. Para obter detalhes, consulte **Uso do kubectl para criar um serviço (criação automática de um balanceador de carga)**.

```
kind: Service
apiVersion: v1
metadata:
  name: hpa-example
spec:
  ports:
    - name: cce-service-0
      protocol: TCP
      port: 80
      targetPort: 80
      nodePort: 31144
  selector:
    app: hpa-example
  type: NodePort
```

Criação de uma política de HPA

Crie uma política HPA. Conforme mostrado abaixo, a política está associada à carga de trabalho hpa-example e o uso da CPU de destino é de 50%.

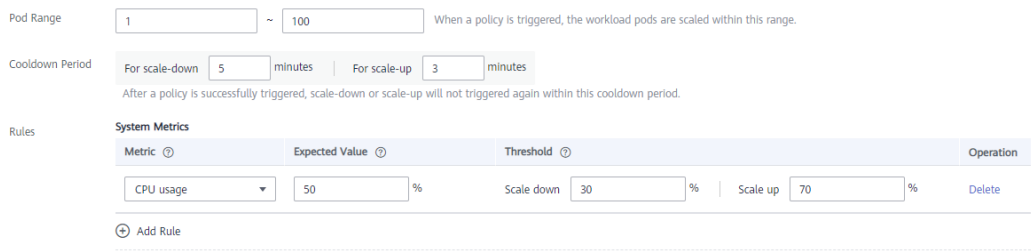
Há duas outras anotações. Uma anotação define os limites da CPU, indicando que o dimensionamento não é realizado quando o uso da CPU está entre 30% e 70% para evitar o impacto causado por pequenas flutuações. A outra é a janela de tempo de dimensionamento, indicando que, depois que a política for executada com sucesso, uma operação de dimensionamento não será acionada novamente nesse intervalo de resfriamento para evitar o impacto causado pela flutuação de curto prazo.

```
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
  name: hpa-policy
  annotations:
    extendedhpa.metrics:
'[{"type":"Resource","name":"cpu","targetType":"Utilization","targetRange":
{"low":"30","high":"70"}}]'
    extendedhpa.option: '{"downscaleWindow":"5m","upscaleWindow":"3m"}'
spec:
  scaleTargetRef:
    kind: Deployment
    name: hpa-example
    apiVersion: apps/v1
  minReplicas: 1
  maxReplicas: 100
  metrics:
    - type: Resource
      resource:
        name: cpu
```



```
      target:
        type: Utilization
        averageUtilization: 50
```

Configure os parâmetros da seguinte forma se você estiver usando o console.



Observação do processo de dimensionamento automático

Passo 1 Verifique o status do nó do cluster. No exemplo a seguir, há dois nós.

```
# kubectl get node
NAME            STATUS   ROLES    AGE     VERSION
192.168.0.183   Ready    <none>   2m20s   v1.17.9-r0-CCE21.1.1.3.B001-17.36.8
192.168.0.26    Ready    <none>   55m     v1.17.9-r0-CCE21.1.1.3.B001-17.36.8
```

Verifique a política de HPA. O uso da CPU da carga de trabalho de destino é de 0%.

```
# kubectl get hpa hpa-policy
NAME         REFERENCE                TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
hpa-policy   Deployment/hpa-example   0%/50%    1         100       1          4m
```

Passo 2 Execute o seguinte comando para acessar a carga de trabalho. No comando a seguir, {ip:port} indica o endereço de acesso da carga de trabalho, que pode ser consultado na página de detalhes da carga de trabalho.

while true;do wget -q -O- http://*{ip:port}*; done

> **NOTA**
>
> Se nenhum EIP for exibido, o nó de cluster não foi atribuído a nenhum EIP. Aloque um, vincule ao nó e sincronize os dados do nó. Para obter detalhes, consulte **Sincronização de dados com servidores em nuvem**.

Observe o processo de dimensionamento da carga de trabalho.

```
# kubectl get hpa hpa-policy --watch
NAME         REFERENCE                TARGETS    MINPODS   MAXPODS   REPLICAS
AGE
hpa-policy   Deployment/hpa-example   0%/50%     1         100       1          4m
hpa-policy   Deployment/hpa-example   190%/50%   1         100       1
4m23s
hpa-policy   Deployment/hpa-example   190%/50%   1         100       4
4m31s
hpa-policy   Deployment/hpa-example   200%/50%   1         100       4
5m16s
hpa-policy   Deployment/hpa-example   200%/50%   1         100       4
6m16s
hpa-policy   Deployment/hpa-example   85%/50%    1         100       4
7m16s
hpa-policy   Deployment/hpa-example   81%/50%    1         100       4
8m16s
hpa-policy   Deployment/hpa-example   81%/50%    1         100       7
8m31s
hpa-policy   Deployment/hpa-example   57%/50%    1         100       7
```



```
9m16s
hpa-policy   Deployment/hpa-example   51%/50%    1         100       7
10m
hpa-policy   Deployment/hpa-example   58%/50%    1         100       7
11m
```

Você pode ver que o uso da CPU da carga de trabalho é de 190% em 4m23s, o que excede o valor de destino. Nesse caso, o dimensionamento é acionado para expandir a carga de trabalho para quatro réplicas/pods. Nos vários minutos seguintes, o uso da CPU não diminui até 7m16s. Isso ocorre porque os novos pods podem não ser criados com sucesso. A causa possível é que os recursos são insuficientes e os pods estão no estado pendente. Durante esse período, os nós estão sendo expandidos.

Aos 7m16s, o uso da CPU diminui, indicando que os pods foram criados com sucesso e começam a suportar tráfego. O uso da CPU diminui para 81% a 8m, ainda maior do que o valor de destino (50%) e o alto limite (70%). Portanto, 7 pods são adicionados a 9m16s e o uso da CPU diminui para 51%, o que está dentro do intervalo de 30% a 70%. A partir de então, o número de pods permanece 7.

Na saída a seguir, você pode ver o processo de dimensionamento da carga de trabalho e a hora em que a política HPA entra em vigor.

```
# kubectl describe deploy hpa-example
...
Events:
  Type    Reason             Age    From                   Message
  ----    ------             ----   ----                   -------
  Normal  ScalingReplicaSet  25m    deployment-controller  Scaled up replica set
hpa-example-79dd795485 to 1
  Normal  ScalingReplicaSet  20m    deployment-controller  Scaled up replica set
hpa-example-79dd795485 to 4
  Normal  ScalingReplicaSet  16m    deployment-controller  Scaled up replica set
hpa-example-79dd795485 to 7
# kubectl describe hpa hpa-policy
...
Events:
  Type    Reason             Age    From                       Message
  ----    ------             ----   ----                       -------
  Normal  SuccessfulRescale  20m    horizontal-pod-autoscaler  New size: 4;
reason: cpu resource utilization (percentage of request) above target
  Normal  SuccessfulRescale  16m    horizontal-pod-autoscaler  New size: 7;
reason: cpu resource utilization (percentage of request) above target
```

Verifique o número de nós. A saída a seguir mostra que dois nós foram adicionados.

```
# kubectl get node
NAME            STATUS   ROLES    AGE     VERSION
192.168.0.120   Ready    <none>   3m5s    v1.17.9-r0-CCE21.1.1.3.B001-17.36.8
192.168.0.136   Ready    <none>   6m58s   v1.17.9-r0-CCE21.1.1.3.B001-17.36.8
192.168.0.183   Ready    <none>   18m     v1.17.9-r0-CCE21.1.1.3.B001-17.36.8
192.168.0.26    Ready    <none>   71m     v1.17.9-r0-CCE21.1.1.3.B001-17.36.8
```

Você também pode visualizar o histórico de dimensionamento no console. Por exemplo, a política de CA é executada uma vez quando a taxa de alocação de CPU no cluster é maior que 70% e o número de nós no pool de nós é aumentado de 2 para 3. O novo nó é adicionado automaticamente pelo autoscaler com base no estado pendente dos pods na fase inicial do HPA.

O processo de dimensionamento do nó é o seguinte:

1. Depois que o número de pods muda para 4, os pods ficam no estado Pendente devido a recursos insuficientes. Como resultado, a política de expansão padrão do complemento autoscaler é acionada e o número de nós é aumentado em um.

2. A expansão do segundo nó é acionada porque a taxa de alocação de CPU no cluster é superior a 70%. Como resultado, o número de nós aumenta em um, que é registrado no histórico de dimensionamento no console. O dimensionamento com base na taxa de alocação garante que o cluster tenha recursos suficientes.

Passo 3 Pare de acessar a carga de trabalho e verifique o número de pods.

```
# kubectl get hpa hpa-policy --watch
NAME         REFERENCE                TARGETS    MINPODS   MAXPODS   REPLICAS
AGE
hpa-policy   Deployment/hpa-example   50%/50%    1         100       7
12m
hpa-policy   Deployment/hpa-example   21%/50%    1         100       7
13m
hpa-policy   Deployment/hpa-example   0%/50%     1         100       7
14m
hpa-policy   Deployment/hpa-example   0%/50%     1         100       7
18m
hpa-policy   Deployment/hpa-example   0%/50%     1         100       3
18m
hpa-policy   Deployment/hpa-example   0%/50%     1         100       3
19m
hpa-policy   Deployment/hpa-example   0%/50%     1         100       3
19m
hpa-policy   Deployment/hpa-example   0%/50%     1         100       3
19m
hpa-policy   Deployment/hpa-example   0%/50%     1         100       3
19m
hpa-policy   Deployment/hpa-example   0%/50%     1         100       3
23m
hpa-policy   Deployment/hpa-example   0%/50%     1         100       3
23m
hpa-policy   Deployment/hpa-example   0%/50%     1         100       1
23m
```

Você pode ver que o uso da CPU é de 21% a 13m. O número de pods é reduzido para 3 a 18m e depois para 1 a 23m.

Na saída a seguir, você pode ver o processo de dimensionamento da carga de trabalho e a hora em que a política HPA entra em vigor.

```
# kubectl describe deploy hpa-example
...
Events:
  Type    Reason             Age    From                   Message
  ----    ------             ----   ----                   -------
  Normal  ScalingReplicaSet  25m    deployment-controller  Scaled up replica set
hpa-example-79dd795485 to 1
  Normal  ScalingReplicaSet  20m    deployment-controller  Scaled up replica set
hpa-example-79dd795485 to 4
  Normal  ScalingReplicaSet  16m    deployment-controller  Scaled up replica set
hpa-example-79dd795485 to 7
  Normal  ScalingReplicaSet  6m28s  deployment-controller  Scaled down replica
set hpa-example-79dd795485 to 3
  Normal  ScalingReplicaSet  72s    deployment-controller  Scaled down replica
set hpa-example-79dd795485 to 1
# kubectl describe hpa hpa-policy
...
Events:
  Type    Reason             Age    From                       Message
  ----    ------             ----   ----                       -------
  Normal  SuccessfulRescale  20m    horizontal-pod-autoscaler  New size: 4;
reason: cpu resource utilization (percentage of request) above target
  Normal  SuccessfulRescale  16m    horizontal-pod-autoscaler  New size: 7;
reason: cpu resource utilization (percentage of request) above target
  Normal  SuccessfulRescale  6m45s  horizontal-pod-autoscaler  New size: 3;
reason: All metrics below target
  Normal  SuccessfulRescale  90s    horizontal-pod-autoscaler  New size: 1;
reason: All metrics below target
```


Você também pode exibir o histórico de execução da política HPA no console. Aguarde até que um nó seja reduzido.

A razão pela qual os outros dois nós no pool de nós não são reduzidos é que ambos têm pods no namespace do sistema kube (e esses pods não são criados por DaemonSets). Para obter detalhes, consulte **Mecanismos de dimensionamento de nós**.

----Fim

Resumo

O uso do HPA e CA pode implementar facilmente o dimensionamento automático na maioria dos cenários. Além disso, o processo de dimensionamento de nós e pods pode ser facilmente observado.

13 Complementos

13.1 Visão geral

O CCE fornece vários tipos de complementos para estender as funções de cluster e atender aos requisitos de recursos. Você pode instalar complementos conforme necessário.

AVISO

O CCE usa gráficos Helm para implementar complementos. Para modificar ou atualizar um complemento, execute operações na página **Add-ons** ou use APIs de gerenciamento de complementos abertas. Não modifique diretamente os recursos relacionados aos complementos em segundo plano. Caso contrário, exceções de complemento ou outros problemas inesperados podem ocorrer.

Tabela 13-1 Lista de complementos

Nome do complemento	Descrição
CoreDNS	CoreDNS é um servidor DNS que fornece resolução de nome de domínio para clusters do Kubernetes por meio de um complemento de cadeia.
Armazenamento do contêiner do CCE (Everest)	O complemento de armazenamento do CCE Everest é um sistema de armazenamento de contêineres nativo da nuvem, que permite que clusters do Kubernetes v1.15.6 ou posterior usem armazenamento em nuvem por meio da Interface de armazenamento de contêiner (CSI).
Detector de problema de nó do CCE	O detector de problemas de nó do CCE (NPD) é um complemento que monitora eventos anormais de nós de cluster e se conecta a uma plataforma de monitoramento de terceiros. É um daemon em execução em cada nó. Ele coleta problemas de nó de diferentes daemons e os reporta ao servidor da API. O complemento NPD pode ser executado como um daemon ou DaemonSet.

Nome do complemento	Descrição
Kubernetes Dashboard	Dashboard do Kubernetes é uma interface de usuário de uso geral baseada na Web para clusters do Kubernetes e integra todos os comandos que podem ser usados na CLI. Ele permite que os usuários gerenciem aplicações em execução em um cluster e solucionem falhas, além de gerenciar o próprio cluster.
Autoscaler de cluster do CCE	O complemento Autoscaler redimensiona um cluster com base no status de agendamento do pod e no uso de recursos.
Kubernetes Metrics Server	metrics-server é um agregador para monitoramento de dados de recursos principais do cluster.
HPA de CCE avançado	cce-hpa-controller é um complemento desenvolvido pelo CCE, que pode ser usado para dimensionar de forma flexível as Implantações com base em métricas como uso de CPU e uso de memória.
Prometheus (EOM)	Prometheus é uma estrutura de monitoramento e alerta de sistema de código aberto. O CCE permite que você instale rapidamente o Prometheus como um complemento.
web-terminal (EOM)	web-terminal é um complemento que permite que você use kubectl em uma interface Web. Ele pode se conectar ao Linux usando o WebSocket através de um navegador e fornece APIs para integração em sistemas independentes. Ele pode ser usado diretamente como um serviço para obter informações através do banco de dados de gerenciamento de configuração (CMDB) e fazer logon no servidor.
Mecanismo de estouro de nuvem do CCE para CCI	O complemento virtual-kubelet é uma implementação do projeto de código aberto Virtual Kubelet. Quando ocorrem picos de tráfego momentâneos em clusters de CCE, o virtual-kubelet usa **Cloud Container Instance (CCI)** para criar pods para Implementações, StatefulSets e tarefas, eliminando a sobrecarga de redimensionamento de clusters do CCE.
Suíte IA do CCE (GPU NVIDIA)	NVIDIA GPU é um complemento de gerenciamento de dispositivos que suporta GPUs em contêineres. Suporta apenas drivers NVIDIA.
Suíte de IA do CCE (Ascend NPU)	NVIDIA GPU é um complemento de gerenciamento de dispositivos que suporta Huawei NPUs em contêineres.
Volcano scheduler	Volcano é um programador para computação de alto desempenho de uso geral, como agendamento de tarefas, gerenciamento de chips heterogêneos e gerenciamento de execução de tarefas, atendendo aos usuários finais por meio de estruturas de computação para diferentes indústrias, como IA, Big Data, sequenciamento de genes e renderização.
Nginx Ingress controller	Nginx Ingress controller encaminha dados de aplicações, como os dados de hosts virtuais, balanceadores de carga, proxy SSL e roteamento HTTP para serviços que podem ser acessados diretamente fora de um cluster.

Nome do complemento	Descrição
Secrets Manager do CCE para DEW	O complemento dew-provider é usado para interconectar-se com o **Data Encryption Workshop (DEW)**, que permite montar segredos armazenados fora de um cluster (DEW para armazenar informações sensíveis) para pods. Desta forma, as informações confidenciais podem ser dissociadas do ambiente de cluster, o que impede o vazamento de informações causado pela codificação do programa ou pela configuração de texto simples.
Exportador de métricas de rede do CCE	dolphin é um complemento para monitorar e gerenciar o tráfego de rede de contêineres. Este complemento coleta quantos pacotes IPv4 e bytes são recebidos e enviados (incluindo aqueles enviados para a Internet) e permite que você obtenha rótulos de pod. dolphin suporta várias tarefas de monitoramento, permite selecionar métricas de monitoramento e usa um PodSelector para selecionar back-ends de monitoramento. As informações de monitoramento foram adaptadas para Prometheus. Você pode chamar a API do Prometheus para visualizar os dados de monitoramento.
NodeLocal DNSCache	O DNSCache do NodeLocal melhora o desempenho do DNS do cluster executando proxies de cache de DNS como DaemonSets em nós do cluster.

Ciclo de vida do complemento

Um ciclo de vida de complemento envolve todos os status do complemento, desde a instalação até a desinstalação.

Tabela 13-2 Status de complemento

Status	Atributo	Descrição
Running	Estado estável	O complemento está sendo executado corretamente, todas as instâncias do complemento são implantadas corretamente e o complemento pode ser usado corretamente.
Partially ready	Estado estável	O complemento está sendo executado corretamente, mas algumas instâncias do complemento não são implementadas corretamente. Nesse estado, as funções do complemento podem estar indisponíveis.
Unavailable	Estado estável	O complemento funciona mal e todas as instâncias do complemento não são implementadas corretamente.
Installing	Estado intermediário	O complemento está sendo implementado. Se todas as instâncias não puderem ser agendadas devido à configuração incorreta do complemento ou recursos insuficientes, o sistema definirá o status do complemento como **Unavailable** 10 minutos depois.

Status	Atributo	Descrição
Installation failed	Estado estável	Falhou na instalação do complemento. Desinstale-o e tente novamente.
Upgrading	Estado intermediário	O complemento está sendo atualizado.
Upgrade failed	Estado estável	O complemento de atualização falhou. Atualize-o novamente ou desinstale-o e tente novamente.
Rolling back	Estado intermediário	O complemento está revertendo.
Rollback failed	Estado estável	Falhou na reversão do complemento. Tente novamente a reversão ou desinstale-a e tente novamente.
Deleting	Estado intermediário	O complemento está a ser eliminado. Se esse estado permanece por muito tempo, ocorre uma exceção.
Deletion failed	Estado estável	Falhou ao excluir complemento. Tente novamente.
Unknown	Estado estável	Nenhum gráfico de complementos encontrado.

NOTA

Quando um complemento está em um estado intermediário, como **Installing** ou **Deleting**, você não tem permissão para editar ou desinstalar o complemento.

Se o status do complemento for desconhecido e o **status.Reason** retornado é "don't install the addon in this cluster", o segredo associado com a release de Helm do complemento no cluster é normalmente excluído por engano. Nesse caso, desinstale o complemento e reinstale-o com as mesmas configurações.

Operações relacionadas

Você pode executar as operações listadas em **Tabela 13-3** na página **Add-ons**.

Tabela 13-3 Operações relacionadas

Operação	Descrição	Procedimento
Install	Instalar um complemento especificado.	1. Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster. No painel de navegação, escolha **Add-ons**. 2. Clique em **Install** sob o complemento de destino. Cada complemento tem parâmetros de configuração diferentes. Para obter detalhes, consulte o capítulo correspondente. 3. Clique em **OK**.
Upgrade	Atualizar um complemento para a nova versão.	1. Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster. No painel de navegação, escolha **Add-ons**. 2. Se um add-on pode ser atualizado, o botão **Upgrade** é exibido abaixo dele. Clique em **Upgrade**. Cada complemento tem parâmetros de configuração diferentes. Para obter detalhes, consulte o capítulo correspondente. 3. Clique em **OK**.
Edit	Editar parâmetros de complemento.	1. Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster. No painel de navegação, escolha **Add-ons**. 2. Clique em **Edit**sob o complemento de destino. Cada complemento tem parâmetros de configuração diferentes. Para obter detalhes, consulte o capítulo correspondente. 3. Clique em **OK**.
Uninstall	Desinstalar um complemento do cluster.	1. Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster. No painel de navegação, escolha **Add-ons**. 2. Clique em **Uninstall** sob o complemento de destino. 3. Na caixa de diálogo exibida, clique em **Yes**. Esta operação não pode ser desfeita.

13.2 CoreDNS

Introdução

CoreDNS é um servidor DNS que fornece resolução de nome de domínio para clusters do Kubernetes por meio de um complemento de cadeia.

O CoreDNS é um software de código aberto e faz parte da CNCF. Ele fornece um meio para os serviços em nuvem descobrirem uns aos outros em implementações nativas da nuvem. Cada um dos plug-ins encadeados pelo CoreDNS fornece uma função de DNS específica. Você pode integrar o CoreDNS apenas com os plug-ins necessários para torná-lo rápido, eficiente e flexível. Quando usado em um cluster do Kubernetes, o CoreDNS pode descobrir automaticamente serviços no cluster e fornecer resolução de nome de domínio para esses serviços. Ao trabalhar com o servidor DNS, o CoreDNS pode resolver nomes de domínio externos para cargas de trabalho em um cluster.

Este extra é instalado por predefinição durante a criação do cluster.

O Kubernetes apoia o CoreDNS como o DNS padrão oficial para todos os clusters daqui para frente.

Site oficial do CoreDNS: **https://coredns.io/**

Comunidade de código aberto: **https://github.com/coredns/coredns**

NOTA

Para mais detalhes, consulte **DNS**.

Restrições

Para executar o CoreDNS corretamente ou fazer upgrade do CoreDNS em um cluster, certifique-se de que o número de nós disponíveis no cluster seja maior ou igual ao número de instâncias do CoreDNS e que todas as instâncias do CoreDNS estejam em execução. Caso contrário, o complemento irá funcionar mal ou a atualização irá falhar.

Instalar o complemento

Este complemento foi instalado por padrão. Se for desinstalado devido a alguns motivos, você pode reinstalá-lo executando as seguintes etapas:

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster. Escolha **Add-ons** no painel de navegação, localize **CoreDNS** à direita e clique em **Install**.

Passo 2 Na página **Install Add-on**, configure as especificações.

Tabela 13-4 Parâmetros de CoreDNS

Parâmetro	Descrição
Pods	Número de pods para o complemento. A alta disponibilidade não é possível com um único pod de complemento. Se ocorrer um erro no nó em que o pod do complemento é executado, o complemento falhará.
Containers	As consultas por segundo (QPS) do complemento CoreDNS estão positivamente correlacionadas com o consumo de CPU. Se o número de nós ou contêineres no cluster aumentar, os pods do CoreDNS suportarão cargas de trabalho mais pesadas. Ajuste o número dos pods do CoreDNS e suas cotas de CPU e memória com base na escala de cluster. Para mais detalhes, consulte **Tabela 13-5**.

Tabela 13-5 Cotas recomendadas do CoreDNS

Nós	QPS recomendado	Pods	VCPUs solicitadas	Limite da vCPU	Memória solicitada	Limite de memória
50	2500	2	500m	500m	512 MiB	512 MiB
200	5000	2	1000m	1000m	1024 MiB	1024 MiB
1000	10000	2	2000m	2000m	2048 MiB	2048 MiB
2000	20000	4	2000m	2000m	2048 MiB	2048 MiB

Passo 3 Configure os parâmetros do complemento.

Tabela 13-6 Parâmetros do complemento CoreDNS

Parâmetro	Descrição
Stub Domain	Um servidor de nomes de domínio para um nome de domínio personalizado. O formato é um par chave-valor. A chave é um sufixo de nome de domínio e o valor é um ou mais endereços IP DNS, por exemplo, **acme.local -- 1.2.3.4,6.7.8.9**. Para mais detalhes, consulte **Configurar o domínio de stub para CoreDNS**.

Parâmetro	Descrição

Advance Config	• **parameterSyncStrategy**: indica se a verificação de consistência deve ser configurada quando o extra é atualizado.
	– **ensureConsistent**: indica que a verificação de consistência da configuração está ativada. Se a configuração registrada no cluster for inconsistente com a configuração real, o complemento não poderá ser atualizado.
	– **force**: indica que a verificação de consistência da configuração é ignorada durante uma atualização. Nesse caso, você deve garantir que a configuração efetiva atual seja a mesma da configuração original. Depois que o complemento é atualizado, restaure o valor de **parameterSyncStrategy** para **ensureConsistent** para habilitar a verificação de consistência de configuração novamente.
	– **inherit**: indica que as configurações personalizadas são herdadas automaticamente durante uma atualização. Depois que o complemento é atualizado, restaure o valor de **parameterSyncStrategy** para **ensureConsistent** para habilitar a verificação de consistência de configuração novamente.
	• **stub_domains**: subdomínios, que permitem configurar um servidor de nomes de domínio para um nome de domínio personalizado. Um subdomínio está no formato de um par chave-valor, onde a chave é o sufixo de um nome de domínio do DNS e o valor é um ou mais endereços IP do DNS.
	• **upstream_nameservers**: endereço IP do servidor DNS upstream.
	• **servers**: servidores de nomes, que estão disponíveis no CoreDNS v1.23.1 e versões posteriores. Você pode personalizar servidores de nomes. Para obter detalhes, consulte **dns-custom-nameservers**. **plugins** indica a configuração de cada componente no CoreDNS. Mantenha as configurações padrão normalmente para evitar que o CoreDNS fique indisponível devido a erros de configuração. Cada componente do plug-in contém **name**, **parameters** (opcional) e **configBlock** (opcional). O formato do Corefile gerado é o seguinte:
	`$name  $parameters {`
	`$configBlock`
	`}`
	Tabela 13-7 descreve plugins comuns. Para detalhes, veja **Plug-ins**.
	Exemplo:
	`{`
	`"servers": [`
	`{`
	`"plugins": [`
	`{`
	`"name": "bind",`
	`"parameters": "{$POD_IP}"`
	`},`
	`{`
	`"name": "cache",`
	`"parameters": 30`
	`},`
	`{`
	`"name": "errors"`
	`},`
	`{`

Parâmetro	Descrição

	"name": "health",
	"parameters": "{$POD_IP}:8080"
	},
	{
	"name": "ready",
	"{$POD_IP}:8081"
	},
	{
	"configBlock": "pods insecure
	\nfallthrough in-addr.arpa ip6.arpa",
	"name": "kubernetes",
	"parameters": "cluster.local in-addr.arpa
	ip6.arpa"
	},
	{
	"name": "loadbalance",
	"parameters": "round_robin"
	},
	{
	"name": "prometheus",
	"parameters": "{$POD_IP}:9153"
	},
	{
	"configBlock": "policy random",
	"name": "forward",
	"parameters": ". /etc/resolv.conf"
	},
	{
	"name": "reload"
	}
	],
	"port": 5353,
	"zones": [
	{
	"zone": "."
	}
	]
	}
	],
	"stub_domains": {
	"acme.local": [
	"1.2.3.4",
	"6.7.8.9"
	]
	},
	"upstream_nameservers": ["8.8.8.8", "8.8.4.4"]
	}

Tabela 13-7 Configuração padrão do plug-in da zona de CoreDNS ativa

Nome do plug-in	Descrição
bind	Endereço IP do host escutado pelo CoreDNS. Mantenha o valor padrão **{$POD_IP}**. Para obter detalhes, consulte **bind**.
cache	Ativa o cache do DNS. Para obter detalhes, consulte **cache**.
errors	Erros são registrados no stdout. Para obter detalhes, consulte **errors**.

Nome do plug-in	Descrição
health	Verificação de integridade do CoreDNS. {$POD_IP}:8080 é escutado. Mantenha a configuração padrão. Caso contrário, a verificação de integridade do CoreDNS falhará e o complemento será reiniciado repetidamente. Para mais detalhes, veja **health**.
ready	Se o servidor back-end está pronto para receber tráfego. {$POD_IP}:8081 é escutado. Se o servidor back-end não estiver pronto, o CoreDNS suspenderá a resolução de DNS até que o servidor back-end esteja pronto. Para obter detalhes, veja **ready**.
kubernetes	O plug-in CoreDNS do Kubernetes, que fornece o recurso de análise de serviços em um cluster. Para detalhes, veja **kubernetes**.
loadbalance	Balanceador de carga do DNS round-robin que randomiza a ordem dos registros A, AAAA e MX em uma resposta. Para obter detalhes, consulte **loadbalance**.
prometheus	API para obter métricas do CoreDNS. {$POD_IP}:9153 é escutado na zona padrão. Mantenha a configuração padrão. Caso contrário, a Prometheus não poderá coletar métricas do CoreDNS. Para detalhes, veja **Prometheus**.
forward	Encaminha quaisquer consultas que não estejam no domínio do cluster do Kubernetes para resolvedores predefinidos (**/etc/resolv.conf**). Para obter detalhes, consulte **forward**.
reload	Recarrega automaticamente Corefiles modificados. Depois de modificar um ConfigMap, aguarde dois minutos para que a modificação tenha efeito. Para obter detalhes, consulte **reload**.

log	Habilita o registro em logs do CoreDNS. Para mais detalhes, consulte **log**.
	Exemplo:
	`{`
	`  "name": "log"`
	`}`

template	Um modelo de resposta rápida, onde **AAAA** indica uma solicitação IPv6. Se **NXDOMAIN** é retornado em uma resposta **rcode**, nenhum resultado de resolução IPv6 é retornado. Para obter detalhes, consulte **template**.
	Exemplo:
	`{`
	`  "configBlock": "rcode NXDOMAIN",`
	`  "name": "template",`
	`  "parameters": "ANY AAAA"`
	`}`

Passo 4 Configure as políticas de agendamento para o complemento.

NOTA

- As políticas de agendamento não entram em vigor em instâncias complementares do tipo DaemonSet.
- Ao configurar a implementação de várias AZs ou a afinidade de nó, verifique se há nós que atendem à política de agendamento e se os recursos são suficientes no cluster. Caso contrário, o complemento não pode ser executado.

Tabela 13-8 Configurações para programação de complementos

Parâmetro	Descrição
Multi-AZ Deployment	• **Preferred**: os pods de Implementação do complemento serão agendados preferencialmente para nós em diferentes AZs. Se todos os nós no cluster forem implementados na mesma AZ, os pods serão agendados para essa AZ. • **Forcible**: os pods de Implementação do complemento serão forçosamente agendados para nós em diferentes AZs. Se houver menos AZs do que pods, os pods extras não funcionarão.
Node Affinity	• **Incompatibility**: a afinidade de nó está desabilitada para o complemento. • **Node Affinity**: especifique os nós em que o complemento é implementado. Se você não especificar os nós, o complemento será agendado aleatoriamente com base na política de agendamento de cluster padrão. • **Specified Node Pool Scheduling**: especifique o pool de nós em que o complemento é implementado. Se você não especificar o pool de nós, o complemento será agendado aleatoriamente com base na política de agendamento de cluster padrão. • **Custom Policies**: insira os rótulos dos nós em que o complemento será implementado para políticas de agendamento mais flexíveis. Se você não especificar rótulos de nó, o complemento será agendado aleatoriamente com base na política de agendamento de cluster padrão. Se várias políticas de afinidade personalizadas estiverem configuradas, certifique-se de que existem nós que atendam a todas as políticas de afinidade no cluster. Caso contrário, o complemento não pode ser executado.
Taints and Tolerations	O uso de manchas e tolerâncias permite (não forçosamente) que Implementação do complemento seja agendada para um nó com as manchas correspondentes e controla as políticas de despejo de Implementação depois que o nó onde a Implementação está localizada é contaminado. O complemento adiciona a política de tolerância padrão para as manchas **node.kubernetes.io/not-ready** e **node.kubernetes.io/unreachable**, respectivamente. A janela de tempo de tolerância é 60s. Para mais detalhes, consulte **Manchas e tolerâncias**.

Passo 5 Clique em **Install**.

----Fim

Componentes

Tabela 13-9 Componentes do CoreDNS

Componente	Descrição	Tipo de recurso
CoreDNS	Servidor DNS para clusters	Implementação

Como funciona a resolução de nomes de domínio no Kubernetes?

As políticas de DNS podem ser configuradas para cada pod. O Kubernetes é compatível com as políticas de DNS **Default**, **ClusterFirst**, **ClusterFirstWithHostNet** e **None**. Para obter detalhes, consulte **DNS para Serviços e pods**. Essas políticas são especificadas no campo **dnsPolicy** no pod específico.

- **Default**: os pods herdam a configuração de resolução de nome do nó em que os pods são executados. O servidor DNS upstream personalizado e o domínio de stub não podem ser usados junto com esta política.
- **ClusterFirst**: qualquer consulta de DNS que não corresponda ao sufixo de domínio de cluster configurado, como **www.kubernetes.io**, é encaminhada para o servidor de nomes upstream herdado do nó. Os administradores de cluster podem ter domínios de stub extra e servidores DNS upstream configurados.
- **ClusterFirstWithHostNet**: para pods em execução com hostNetwork, defina a política de DNS **ClusterFirstWithHostNet**.
- **None**: ela permite que um pod ignore as configurações de DNS do ambiente do Kubernetes. Todas as configurações de DNS devem ser fornecidas usando o campo **dnsPolicy** no pod específico.

NOTA

- Os clusters do Kubernetes v1.10 e posteriores são compatíveis com **Default**, **ClusterFirst**, **ClusterFirstWithHostNet** e **None**. Os clusters anteriores ao Kubernetes v1.10 suportam apenas **Default**, **ClusterFirst** e **ClusterFirstWithHostNet**.
- **Default** não é a política de DNS padrão. Se **dnsPolicy** não for explicitamente especificado, será usado **ClusterFirst**.

Roteamento

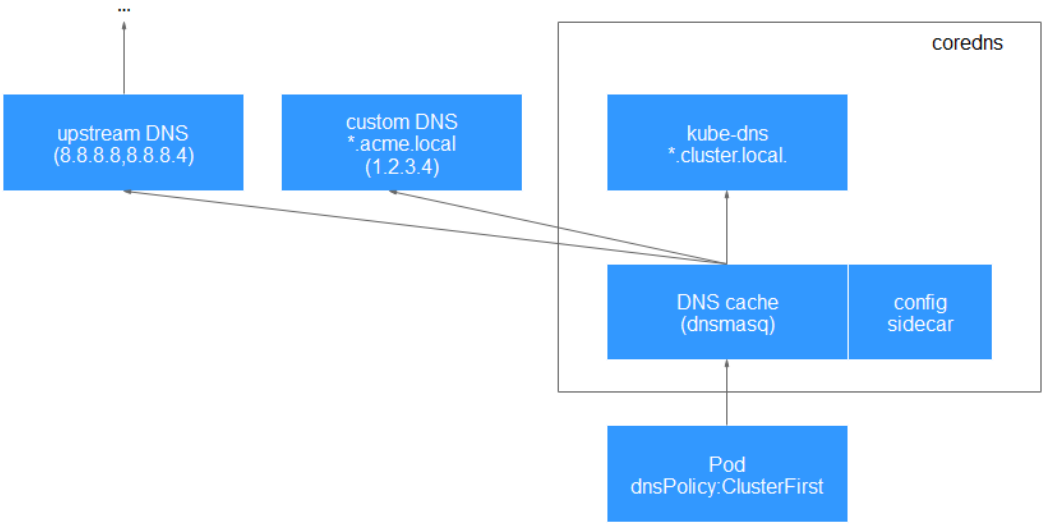
Without stub domain configurations: qualquer consulta que não corresponda ao sufixo de domínio de cluster configurado, como **www.kubernetes.io**, é encaminhada para o servidor DNS upstream herdado do nó.

With stub domain configurations: se os domínios de stub e servidores DNS upstream estiverem configurados, as consultas de DNS serão roteadas de acordo com o seguinte fluxo:

1. A consulta é enviada primeiro para a camada de cache do DNS no CoreDNS.
2. A partir da camada de cache, o sufixo da solicitação é examinado e, em seguida, a solicitação é encaminhada para o DNS correspondente:

- Nomes com o sufixo de cluster, por exemplo, **.cluster.local**: a solicitação é enviada ao CoreDNS.
- Nomes com o sufixo de domínio de stub, por exemplo, **.acme.local**: a solicitação é enviada para o resolvedor do DNS personalizado configurado que escuta, por exemplo, em 1.2.3.4.
- Nomes que não correspondem ao sufixo (por exemplo, **widget.com**): a solicitação é encaminhada para o DNS upstream.

Figura 13-1 Roteamento



13.3 Armazenamento do contêiner do CCE (Everest)

Introdução

O Everest é um sistema de armazenamento de contêineres nativo da nuvem, que permite que clusters do Kubernetes v1.15.6 ou posterior acessem serviços de armazenamento em nuvem por meio do CSI.

Everest é um complemento de recursos do sistema. Ele é instalado por padrão quando um cluster do Kubernetes v1.15 ou posterior é criado.

Restrições

- Se o cluster for atualizado da v1.13 para a v1.15, **storage-driver** será substituído pelo Everest (v1.1.6 ou posterior) para armazenamento em contêiner. A aquisição não afeta as funções de armazenamento originais.
- Na versão 1.2.0 do complemento Everest, **a autenticação por chave** é otimizada quando o OBS é usado. Depois que o complemento Everest for atualizado de uma versão anterior à 1.2.0, reinicie todas as cargas de trabalho que usam o OBS no cluster. Caso contrário, as cargas de trabalho talvez não consigam usar o OBS.
- Por predefinição, este extra é instalado em **clusters de v1.15 e posteriores**. Para clusters v1.13 e anteriores, o complemento **storage-driver** é instalado por padrão.
- Os nós que executam Huawei Cloud EulerOS 1.1 suportam os complementos Everest de v2.x.x (v2.1.9 ou posterior) e v1.2.x (v1.2.70 ou posterior), mas não suportam os complementos de v1.3.x.

Instalar o complemento

Este complemento foi instalado por padrão. Se for desinstalado devido a alguns motivos, você pode reinstalá-lo executando as seguintes etapas:

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster. Clique em **Add-ons** no painel de navegação, localize **CCE Container Storage (Everest)** à direita e clique em **Install**.

Passo 2 Na página **Install Add-on**, configure as especificações.

Tabela 13-10 Parâmetros do Everest

Parâmetro	Descrição
Instances	Número de instâncias para o complemento. A alta disponibilidade não é possível com uma única instância do complemento. Se ocorrer um erro no nó em que a instância do complemento é executada, o complemento falhará.
Containers	O complemento Everest contém os componentes Everest-csi-controller e everest-csi-driver. Para mais detalhes, consulte **Componentes**. As especificações dos componentes do complemento podem ser personalizadas com base nas suas necessidades. Retenha os valores padrão solicitados de CPU e memória dos componentes do complemento. Os valores limite podem ser ajustados com base no número de nós de cluster e PVCs. Para obter detalhes sobre as sugestões de configuração, consulte **Tabela 13-11**. Em cenários não típicos, as fórmulas para estimar os valores de limite são as seguintes: ● everest-csi-controller – Limite da CPU: 250m para 200 ou menos nós, 350m para 1000 nós e 500m para 2000 nós – Limite de memória = (200 Mi + número de nós x 1 Mi + número de PVCs x 0,2 Mi) x 1,2 ● everest-csi-driver – Limite da CPU: 300m para 200 ou menos nós, 500m para 1000 nós e 800m para 2000 nós – Limite de memória: 300 Mi para 200 ou menos nós, 600 Mi para 1000 nós e 900 Mi para 2000 nós

Tabela 13-11 Limites de configuração recomendados em cenários típicos

Cenário de configuração			everest-csi-controller		everest-csi-driver	
Nós	PVs/ PVCs	Instâncias do complemento	vCPUs (limite = solicitada)	Memória (limite = solicitada)	vCPUs (limite = solicitada)	Memória (limite = solicitada)
50	1000	2	250m	600 MiB	300m	300 MiB
200	1000	2	250m	1 GiB	300m	300 MiB
1000	1000	2	350m	2 GiB	500m	600 MiB
1000	5000	2	450m	3 GiB	500m	600 MiB
2000	5000	2	550m	4 GiB	800m	900 MiB
2000	10000	2	650m	5 GiB	800m	900 MiB

Passo 3 Configure os parâmetros do complemento.

Tabela 13-12 Parâmetros do Everest

Parâmetro	Descrição
csi_attacher_worker_threads	Número de nós de trabalho que podem ser processados simultaneamente pelo Everest para anexar volumes do EVS. O valor padrão é **60**.
csi_attacher_detach_worker_threads	Número de nós de trabalho que podem ser processados simultaneamente pelo Everest para desanexar volumes do EVS. O valor padrão é **60**.
volume_attaching_flow_ctrl	Número máximo de volumes do EVS que podem ser anexados pelo complemento Everest em 1 minuto. O valor padrão é **0**, indicando que o desempenho de anexar volumes do EVS é determinado pelos recursos de armazenamento subjacentes.
cluster_id	ID do cluster
default_vpc_id	ID da VPC à qual o cluster pertence
disable_auto_mount_secret	Se a AK/SK padrão pode ser usada quando um bucket de objetos ou um sistema de arquivos paralelo é montado. O valor padrão é **false**.
enable_node_attacher	Se deve habilitar o anexador no agente para processar o **VolumeAttachment**.
flow_control	Este campo é deixado em branco por padrão. Você não precisa configurar esse parâmetro.

Parâmetro	Descrição
over_subscription	Taxa de comprometimento excessivo do pool de armazenamento local (**local_storage**). O valor padrão é **80**. Se o tamanho do pool de armazenamento local for de 100 GB, ele poderá estar comprometido em excesso para 180 GB.
project_id	ID do projeto ao qual um cluster pertence

NOTA

No Everest 1.2.26 ou posterior, o desempenho de anexar um grande número de volumes do EVS foi otimizado. Os seguintes parâmetros podem ser configurados:

- csi_attacher_worker_threads
- csi_attacher_detach_worker_threads
- volume_attaching_flow_ctrl

Os parâmetros anteriores são associados uns aos outros e são restringidos pelos recursos de armazenamento subjacentes na região onde o cluster está localizado. Para anexar um grande número de volumes (mais de 500 volumes do EVS por minuto), entre em contato com o atendimento ao cliente e configure os parâmetros sob sua orientação para evitar que o complemento Everest seja executado anormalmente devido a configurações de parâmetros incorretas.

Passo 4 Configure as políticas de agendamento para o complemento.

NOTA

- As políticas de agendamento não entram em vigor em instâncias complementares do tipo DaemonSet.
- Ao configurar a implementação de várias AZs ou a afinidade de nó, verifique se há nós que atendem à política de agendamento e se os recursos são suficientes no cluster. Caso contrário, o complemento não pode ser executado.

Tabela 13-13 Configurações para programação de complementos

Parâmetro	Descrição
Multi-AZ Deployment	- **Preferred**: os pods de Implementação do complemento serão agendados preferencialmente para nós em diferentes AZs. Se todos os nós no cluster forem implementados na mesma AZ, os pods serão agendados para essa AZ. - **Forcible**: os pods de Implementação do complemento serão forçosamente agendados para nós em diferentes AZs. Se houver menos AZs do que pods, os pods extras não funcionarão.

Parâmetro	Descrição
Node Affinity	• **Incompatibility**: a afinidade de nó está desabilitada para o complemento. • **Node Affinity**: especifique os nós em que o complemento é implementado. Se você não especificar os nós, o complemento será agendado aleatoriamente com base na política de agendamento de cluster padrão. • **Specified Node Pool Scheduling**: especifique o pool de nós em que o complemento é implementado. Se você não especificar o pool de nós, o complemento será agendado aleatoriamente com base na política de agendamento de cluster padrão. • **Custom Policies**: insira os rótulos dos nós em que o complemento será implementado para políticas de agendamento mais flexíveis. Se você não especificar rótulos de nó, o complemento será agendado aleatoriamente com base na política de agendamento de cluster padrão. Se várias políticas de afinidade personalizadas estiverem configuradas, certifique-se de que existem nós que atendam a todas as políticas de afinidade no cluster. Caso contrário, o complemento não pode ser executado.
Taints and Tolerations	O uso de manchas e tolerâncias permite (não forçosamente) que Implementação do complemento seja agendada para um nó com as manchas correspondentes e controla as políticas de despejo de Implementação depois que o nó onde a Implementação está localizada é contaminado. O complemento adiciona a política de tolerância padrão para as manchas **node.kubernetes.io/not-ready** e **node.kubernetes.io/unreachable**, respectivamente. A janela de tempo de tolerância é 60s. Para mais detalhes, consulte **Manchas e tolerâncias**.

Passo 5 Clique em **Install**.

----Fim

Componentes

Tabela 13-14 Componentes do Everest

Componente	Descrição	Tipo de recurso
everest-csi-controller	Usado para criar, excluir, capturar instantâneos, expandir, anexar e desanexar volumes de armazenamento. Se a versão do cluster for 1.19 ou posterior e a versão do complemento for 1.2.*x*, o pod do componente everest-csi-controller também terá um contêiner everest-localvolume-manager por padrão. Esse contêiner gerencia a criação de pools de armazenamento de LVM e PVs locais no nó.	Implementação
everest-csi-driver	Usado para montar e desmontar PVs e redimensionar sistemas de arquivos. Se a versão do complemento for 1.2.*x* e a região onde o cluster está localizado suportar node-attacher, o pod do componente everest-csi-driver também contém um contêiner everest-node-attacher. Este contêiner é responsável pela anexação do EVS distribuído. Este item de configuração está disponível em algumas regiões.	DaemonSet

13.4 Detector de problema de nó do CCE

Introdução

O detector de problemas de nó do CCE (NPD) é um complemento que monitora eventos anormais de nós de cluster e se conecta a uma plataforma de monitoramento de terceiros. É um daemon em execução em cada nó. Ele coleta problemas de nó de diferentes daemons e os reporta ao servidor da API. O complemento NPD pode ser executado como um daemon ou DaemonSet.

Para obter mais informações, consulte **node-problem-detector**.

Restrições

- Ao usar esse complemento, não formate ou particione discos de nó.
- Cada processo de NPD ocupa 30 m de CPU e 100 MB de memória.

Permissões

Para monitorar os logs do kernel, o complemento NPD precisa ler o host **/dev/kmsg**. Portanto, o modo privilegiado deve ser ativado. Para mais detalhes, veja **privileged**.

Além disso, o CCE mitiga os riscos de acordo com o princípio do privilégio mínimo. Apenas os seguintes privilégios estão disponíveis para a execução do NPD:

- cap_dac_read_search: permissão para acessar **/run/log/journal**.
- cap_sys_admin: permissão para acessar **/dev/kmsg**.

Instalar o complemento

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster. Escolha **Add-ons** no painel de navegação, encontre **CCE Node Problem Detector** na direita, e clique em **Install**.

Passo 2 Na página **Install Add-on**, configure as especificações.

Tabela 13-15 Configuração do NPD

Parâmetro	Descrição
Add-on Specifications	As especificações podem ser **Custom**.
Instances	Se você selecionar **Custom**, poderá ajustar o número de pods conforme necessário.
Containers	Se você selecionar **Custom**, poderá ajustar as especificações do contêiner conforme necessário.

Passo 3 Configure os parâmetros do complemento.

Somente as versões v1.16.0 e posteriores suportam as configurações.

Tabela 13-16 Parâmetros do NPD

Parâmetro	Descrição
common.image.pullPolicy	Uma política de extração de imagens. O valor padrão é **IfNotPresent**.
feature_gates	Um portão de recurso
npc.maxTaintedNode	O número máximo de nós que o NPC pode adicionar manchas quando uma única falha ocorre em vários nós para minimizar o impacto O valor pode estar no formato int ou percentual.
npc.nodeAffinity	Afinidade do nó do controlador

Passo 4 Configure as políticas de agendamento para o complemento.

NOTA

- As políticas de agendamento não entram em vigor em instâncias complementares do tipo DaemonSet.
- Ao configurar a implementação de várias AZs ou a afinidade de nó, verifique se há nós que atendem à política de agendamento e se os recursos são suficientes no cluster. Caso contrário, o complemento não pode ser executado.

Tabela 13-17 Configurações para programação de complementos

Parâmetro	Descrição
Multi-AZ Deployment	• **Preferred**: os pods de Implementação do complemento serão agendados preferencialmente para nós em diferentes AZs. Se todos os nós no cluster forem implementados na mesma AZ, os pods serão agendados para essa AZ. • **Forcible**: os pods de Implementação do complemento serão forçosamente agendados para nós em diferentes AZs. Se houver menos AZs do que pods, os pods extras não funcionarão.
Node Affinity	• **Incompatibility**: a afinidade de nó está desabilitada para o complemento. • **Node Affinity**: especifique os nós em que o complemento é implementado. Se você não especificar os nós, o complemento será agendado aleatoriamente com base na política de agendamento de cluster padrão. • **Specified Node Pool Scheduling**: especifique o pool de nós em que o complemento é implementado. Se você não especificar o pool de nós, o complemento será agendado aleatoriamente com base na política de agendamento de cluster padrão. • **Custom Policies**: insira os rótulos dos nós em que o complemento será implementado para políticas de agendamento mais flexíveis. Se você não especificar rótulos de nó, o complemento será agendado aleatoriamente com base na política de agendamento de cluster padrão. Se várias políticas de afinidade personalizadas estiverem configuradas, certifique-se de que existem nós que atendam a todas as políticas de afinidade no cluster. Caso contrário, o complemento não pode ser executado.
Taints and Tolerations	O uso de manchas e tolerâncias permite (não forçosamente) que Implementação do complemento seja agendada para um nó com as manchas correspondentes e controla as políticas de despejo de Implementação depois que o nó onde a Implementação está localizada é contaminado. O complemento adiciona a política de tolerância padrão para as manchas **node.kubernetes.io/not-ready** e **node.kubernetes.io/unreachable**, respectivamente. A janela de tempo de tolerância é 60s. Para mais detalhes, consulte **Manchas e tolerâncias**.

Passo 5 Clique em **Install**.

----Fim

Componentes

Tabela 13-18 Componentes do NPD

Componente	Descrição	Tipo de recurso
node-problem-controller	Isolar falhas basicamente com base em resultados de detecção de falhas.	Implementação
node-problem-detector	Detectar falhas de nó.	DaemonSet

Itens de verificação de NPD

NOTA

Os itens de verificação são suportados apenas em 1.16.0 e versões posteriores.

Verifique os itens de cobertura de eventos e status.

- Relacionados a eventos

  Para itens de verificação relacionados a eventos, quando ocorre um problema, o NPD relata um evento para o servidor da API. O tipo de evento pode ser **Normal** (evento normal) ou **Warning** (evento anormal).

  Tabela 13-19 Itens de verificação relacionados a eventos

Item de verificação	Função	Descrição
OOMKilling	Ouvir os logs do kernel e verifique se os eventos OOM ocorrem e são relatados. Cenário típico: quando o uso de memória de um processo em um contêiner excede o limite, a OOM é acionada e o processo é encerrado.	Evento de aviso Objeto de escuta: **/dev/kmsg** Regra de correspondência: "Killed process \\d+ (.+) total-vm:\\d+kB, anon-rss:\\d+kB, file-rss:\\d +kB.*"
TaskHung	Ouvir os logs do kernel e verifique se os eventos taskHung ocorrem e são relatados. Cenário típico: a suspensão de I/O de disco causa a suspensão do processo.	Evento de aviso Objeto de escuta: **/dev/kmsg** Regra de correspondência: "task \\S+:\\w+ blocked for more than \\w+ seconds\\."

Item de verificação	Função	Descrição
ReadonlyFilesystem	Verificar se o erro **Remount root filesystem read-only** ocorre no kernel do sistema ouvindo os logs do kernel. Cenário típico: um usuário separa um disco de dados de um nó por engano no ECS, e as aplicações gravam dados continuamente no ponto de montagem do disco de dados. Como resultado, ocorre um erro de I/O no kernel e o disco é montado novamente como um disco de somente leitura. **NOTA** Se o rootfs dos node pods for do tipo mapeador de dispositivos, ocorrerá um erro no thin pool se um disco de dados for desanexado. Isso afetará o NPD e o NPD não poderá detectar falhas de nó.	Evento de aviso Objeto de escuta: **/dev/kmsg** Regra de correspondência: **Remounting filesystem read-only**

- Status-related

  Para itens de verificação relacionados ao status, quando ocorre um problema, o NPD relata um evento para o servidor da API e altera o status do nó de forma síncrona. Esta função pode ser usada em conjunto com o **isolamento de falhas do Node-problem-controller** para isolar nós.

  Se o período de verificação não for especificado nos seguintes itens de verificação, o período padrão será de 30 segundos.

  Tabela 13-20 Verificar componentes do sistema

Item de verificação	Função	Descrição
Erro no componente de rede do contêiner CNIProblem	Verificar o status dos componentes da CNI (componentes da rede do contêiner).	Nenhuma
Erro no componente de tempo de execução do contêiner CRIProblem	Verificar o status do Docker e do contêiner dos componentes de CRI (componentes de tempo de execução do contêiner).	Objeto de verificar: Docker ou containerd

Item de verificação	Função	Descrição
Reinícios frequentes do Kubelet FrequentKubeletRestart	Retroceder periodicamente os logs do sistema para verificar se o componente importante de Kubelet reinicia com frequência.	● Limite padrão: 10 reinícios em 10 minutos Se o Kubelet for reiniciado 10 vezes em 10 minutos, isso indica que o sistema é reiniciado com frequência e um alarme de falha é gerado. ● Objeto de escuta: logs no diretório **/run/log/journal** **NOTA** Os sistemas operacionais Ubuntu e HCE 2.0 não suportam os itens de verificação anteriores devido a formatos de log incompatíveis.
Reinícios frequentes do Docker FrequentDockerRestart	Retroceder periodicamente os logs do sistema para verificar se o tempo de execução do contêiner Docker é reiniciado com frequência.	
Reinícios frequentes do containerd FrequentContainerdRestart	Retroceder periodicamente os logs do sistema para verificar se o tempo de execução do contêiner containerd reinicia com frequência.	
erro de kubelet KubeletProblem	Verificar o status do componente importante de Kubelet.	Nenhuma
erro de kube-proxy KubeProxyProblem	Verificar o status do componente importante de kube-proxy.	Nenhuma

Tabela 13-21 Verificar métricas do sistema

Item de verificação	Função	Descrição
Tabela conntrack cheia ConntrackFullProblem	Verificar se a tabela conntrack está cheia.	● Limite padrão: 90% ● Utilização: **nf_conntrack_count** ● Valor máximo: **nf_conntrack_max**
Recursos de disco insuficientes DiskProblem	Verificar o uso do disco do sistema e dos discos de dados do CCE (incluindo o disco lógico de CRI e o disco lógico de kubelet) no nó.	● Limite padrão: 90% ● Origem: `df -h` Atualmente, discos de dados adicionais não são suportados.

Item de verificação	Função	Descrição
Manipuladores de arquivo insuficientes FDProblem	Verificar se os manipuladores do arquivo FD estão esgotados.	● Limite padrão: 90% ● Utilização: o primeiro valor em **/proc/sys/fs/file-nr** ● Valor máximo: o terceiro valor em **/proc/sys/fs/file-nr**
Memória de nó insuficiente MemoryProblem	Verificar se a memória está gasta.	● Limite padrão: 80% ● Uso: **MemTotal-MemAvailable** em **/proc/meminfo** ● Valor máximo: **MemTotal** em **/proc/meminfo**
Recursos de processo insuficientes PIDProblem	Verificar se os recursos do processo PID estão esgotados.	● Limite padrão: 90% ● Utilização: **nr_threads in /proc/loadavg** ● Valor máximo: menor valor entre **/proc/sys/kernel/pid_max** e **/proc/sys/kernel/threads-max**.

Tabela 13-22 Verificar o armazenamento

Item de verificação	Função	Descrição
Disco de somente leitura DiskReadonly	Execute periodicamente testes de gravação no disco do sistema e nos discos de dados do CCE (incluindo o disco lógico de CRI e o disco lógico de Kubelet) do nó para verificar a disponibilidade dos discos importantes.	Caminhos de detecção: ● /mnt/paas/kubernetes/kubelet/ ● /var/lib/docker/ ● /var/lib/containerd/ ● /var/paas/sys/log/cceaddon-npd/ O arquivo temporário **npd-disk-write-ping** é gerado no caminho de detecção. Atualmente, discos de dados adicionais não são suportados.

Item de verificação	Função	Descrição
Erro do pool de armazenamento emptyDir EmptyDirVolumeGroupStatusError	Verificar se o grupo de volume efêmero no nó é normal. Impacto: o pod que depende do pool de armazenamento não pode gravar dados no volume temporário. O volume temporário é montado novamente como um sistema de arquivos somente leitura pelo kernel devido a um erro de I/O. Cenário típico: ao criar um nó, um usuário configura dois discos de dados como um pool de armazenamento de volume temporário. O usuário exclui alguns discos de dados por engano. Como resultado, o pool de armazenamento se torna anormal.	● Período da detecção: 30s ● Origem: `vgs -o vg_name, vg_attr` ● Princípio: verifique se o VG (storage pool) está no estado P. Se sim, alguns PVs (discos de dados) são perdidos. ● Agendamento conjunto: o agendador pode identificar automaticamente um erro de pool de armazenamento PV e impedir que pods que dependem do pool de armazenamento sejam gendados para o nó. ● Cenário excepcional: o complemento NPD não pode detectar a perda de todos os PVs (discos de dados), resultando na perda de VGs (pools de armazenamento). Nesse caso, o kubelet isola automaticamente o nó, detecta a perda de VGs (pools de armazenamento) e atualiza os recursos correspondentes em **nodestatus.allocatable** para **0**. Isso impede que os pods que dependem do pool de armazenamento sejam agendados para o nó. O dano de um único PV não pode ser detectado por este item de verificação, mas pelo item de verificação ReadonlyFilesystem.
Erro do pool de armazenamento PV LocalPvVolumeGroupStatusError	Verificar o grupo PV no nó. Impacto: os pods que dependem do pool de armazenamento não podem gravar dados no volume persistente. O volume persistente é remontado como um sistema de arquivos somente leitura pelo kernel devido a um erro de I/O. Cenário típico: ao criar um nó, um usuário configura dois discos de dados como um pool de armazenamento de volume persistente. Alguns discos de dados são apagados por engano.	

Item de verificação	Função	Descrição
Erro do ponto de montagem MountPointProblem	Verificar o ponto de montagem no nó. Definição excepcional: você não pode acessar o ponto de montagem executando o comando **cd**. Cenário típico: Network File System (NFS), por exemplo, obsfs e s3fs é montado em um nó. Quando a conexão é anormal devido a exceções de servidor do NFS de rede ou de par, todos os processos que acessam o ponto de montagem são suspensos. Por exemplo, durante uma atualização de cluster, um kubelet é reiniciado e todos os pontos de montagem são verificados. Se o ponto de montagem anormal for detectado, a atualização falhará.	Alternativamente, você pode executar o seguinte comando: `for dir in `df -h | grep -v "Mounted on" | awk "{print \\$NF}"`;do cd $dir; done && echo "ok"`
I/O de disco suspensa DiskHung	Verificar se a suspensão de I/O ocorre em todos os discos no nó, ou seja, se as operações de leitura e gravação de I/O não são respondidas. Definição de suspensão de I/O: o sistema não responde a solicitações de I/O de disco e alguns processos estão no estado D. Cenário típico: os discos não podem responder devido a drivers de disco rígido do sistema operacional anormais ou falhas graves na rede subjacente.	● Objeto de verificar: todos os discos de dados ● Origem: /proc/diskstat Alternativamente, você pode executar o seguinte comando: `iostat -xmt 1` ● Limite: – Uso médio: ioutil >= 0,99 – Comprimento médio da fila de I/O: avgqu-sz >= 1 – Volume médio de transferência de I/O: iops (w/s) + ioth (wMB/s) <= 1 **NOTA** Em alguns sistemas operacionais, nenhum dado é alterado durante a I/O. Nesse caso, calcule o uso do tempo de I/O da CPU. O valor de iowait deve ser maior que 0,8.

Item de verificação	Função	Descrição
I/O de disco lenta DiskSlow	Verificar se todos os discos no nó têm I/Os lentas, ou seja, se as I/Os respondem lentamente. Cenário típico: os discos EVS têm I/Os lentas devido à flutuação da rede.	● Objeto de verificar: todos os discos de dados ● Origem: /proc/diskstat Alternativamente, você pode executar o seguinte comando: `iostat -xmt 1` ● Limite padrão: Latência média de I/O: await >= 5000 ms **NOTA** Se as solicitações de I/O não forem respondidas e os dados **await** não forem atualizados, esse item de verificação será inválido.

Tabela 13-23 Outros itens de verificação

Item de verificação	Função	Descrição
NTP anormal NTPProblem	Verificar se o serviço de sincronização de relógio de nó ntpd ou chronyd está sendo executado corretamente e se um desvio de tempo do sistema é causado.	Limite de deslocamento de relógio padrão: 8000 ms
Erro no processo D ProcessD	Verificar se há um processo D no nó.	Limite padrão: 10 processos anormais detectados por três vezes consecutivas Origem: ● /proc/{PID}/stat ● Como alternativa, você pode executar o comando **ps aux**. Cenário excepcional: o item de verificação ProcessD ignora os processos D residentes (heartbeat e update) dos quais o driver SDI no nó do BMS depende.
Erro de processo Z ProcessZ	Verificar se o nó tem processos no estado Z.	

Item de verificação	Função	Descrição
Erro de ResolvConf ResolvConfFileProblem	Verificar se o arquivo ResolvConf foi perdido. Verificar se o arquivo ResolvConf está normal. Definição excepcional: nenhum servidor de resolução de nome de domínio upstream (nameserver) está incluído.	Objeto: **/etc/resolv.conf**
Evento agendado existente ScheduledEvent	Verificar se existem eventos de migração ao vivo agendados no nó. Um evento de plano de migração ao vivo geralmente é acionado por uma falha de hardware e é um método de retificação automática de falhas na camada IaaS. Cenário típico: o host está defeituoso. Por exemplo, o ventilador está danificado ou o disco tem setores defeituosos. Como resultado, a migração ao vivo é acionada para VMs.	Origem: ● http://169.254.169.254/meta-data/latest/events/scheduled Este item de verificação é um recurso Alpha e está desativado por padrão.

O componente de kubelet tem os seguintes itens de verificação padrão, que têm bugs ou defeitos. Você pode corrigi-los atualizando o cluster ou usando o NPD.

Tabela 13-24 Itens de verificação padrão do kubelet

Item de verificação	Função	Descrição
Recursos de PID insuficientes. PIDPressure	Verificar se os PIDs são suficientes.	● Intervalo: 10 segundos ● Limite: 90% ● Defeito: na versão 1.23.1 da comunidade e versões anteriores, este item de verificação torna-se inválido quando mais de 65535 PIDs são usados. Para obter detalhes, consulte **issue 107107**. Na versão 1.24 da comunidade e versões anteriores, thread-max não é considerado neste item de verificação.
Memória insuficiente MemoryPressure	Verificar se a memória alocável para os contêineres é suficiente.	● Intervalo: 10 segundos ● Limite: máx. 100 MiB ● Alocável = memória total de um nó – memória reservada de um nó ● Defeito: esse item de verificação verifica apenas a memória consumida pelos contêineres e não considera a consumida por outros elementos no nó.
Recursos de disco insuficientes DiskPressure	Verificar o uso do disco e o uso de inodes dos discos do kubelet e do Docker.	● Intervalo: 10 segundos ● Limite: 90%

Isolamento de falha do Node-problem-controller

NOTA

O isolamento de falhas é suportado apenas por complementos de 1.16.0 e versões posteriores.

Por padrão, se vários nós se tornarem defeituosos, o NPC adiciona manchas a até 10% dos nós. Você pode definir **npc.maxTaintedNode** para aumentar o limite.

O plug-in NPD de código aberto fornece detecção de falhas, mas não isolamento de falhas. O CCE aprimora o controlador de problemas de nó (NPC) baseado no NPD de código aberto. Esse componente é implementado com base no **node controller** do Kubernetes. Para falhas relatadas pelo NPD, o NPC adiciona automaticamente manchas aos nós para isolamento de falhas de nó.

Tabela 13-25 Parâmetros

Parâmetro	Descrição	Padrão
npc.enable	Se ativar o NPC Este parâmetro não é suportado em versões 1.18.0 ou posteriores.	true
npc.maxTaintedNode	O número máximo de nós que o NPC pode adicionar manchas quando uma única falha ocorre em vários nós para minimizar o impacto. O formato int e o formato percentual são suportados.	10% Intervalo de valores: ● O valor está no formato int e varia de 1 a infinito. ● O valor varia de 1% a 100%, em porcentagem. O valor mínimo deste parâmetro multiplicado pelo número de nós de cluster é 1.
npc.nodeAffinity	Afinidade do nó do controlador	N/D

Exibir eventos do NPD

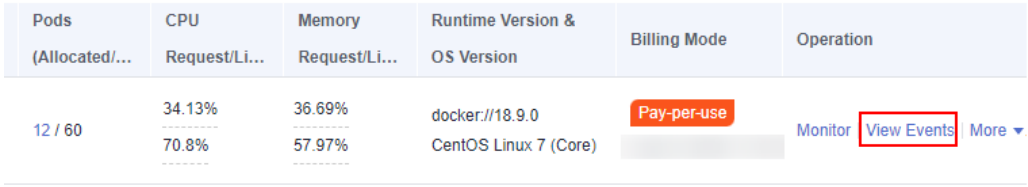
Os eventos relatados pelo complemento NPD podem ser consultados na página **Nodes**.

Passo 1 Efetue logon no console do CCE.

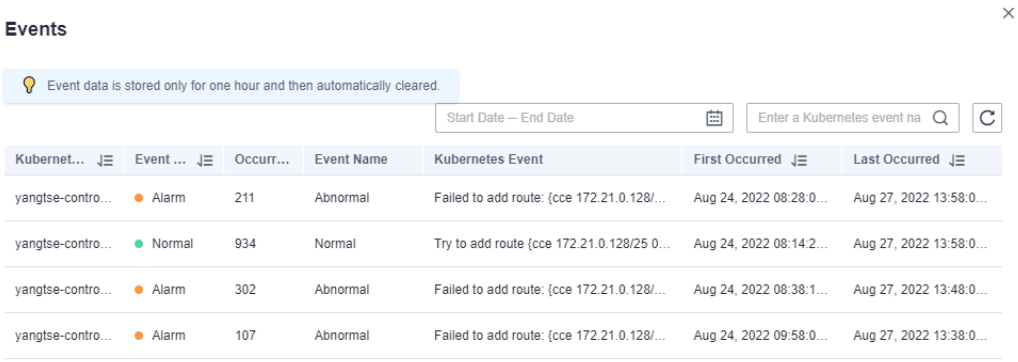
Passo 2 Clique no nome do cluster para acessar o console do cluster. Escolha **Nodes** no painel de navegação.

Passo 3 Localize a linha que contém o nó de destino e clique em **View Events**.

Figura 13-2 Exibir eventos do nó



Você pode obter eventos na página exibida.



----Fim

Alarmes de AOM

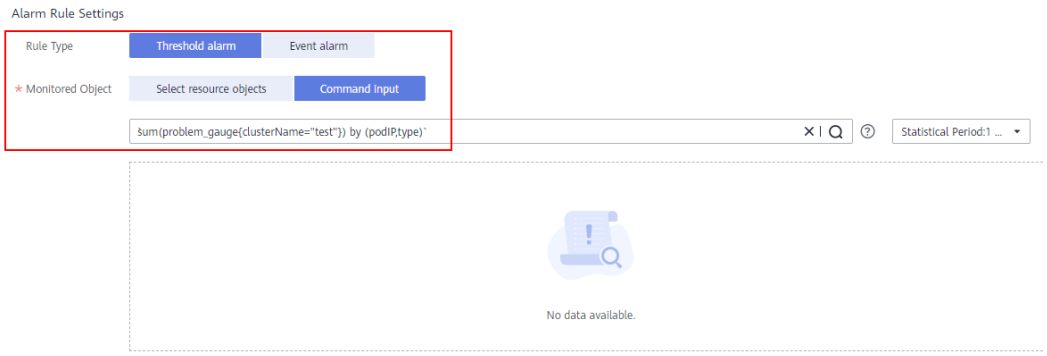
Para itens de verificação de NPD relacionados ao status, você pode configurar o Application Operations Management (AOM) para converter status anormais em alarmes de AOM e notificá-lo via mensagem SMS ou e-mail.

Passo 1 Efetue logon no console do AOM.

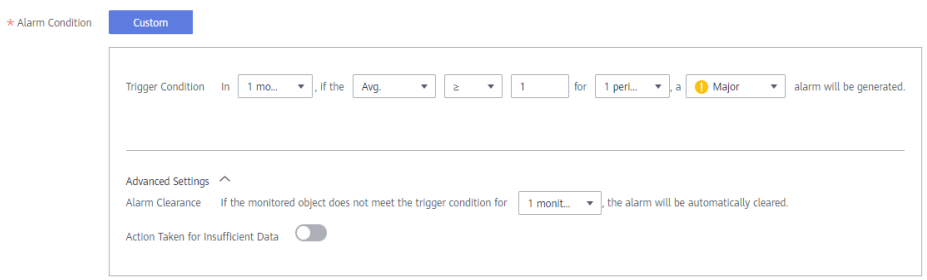
Passo 2 No painel de navegação, escolha **Alarm Center** > **Alarm Rules** e clique em **Create Alarm Rule**.

Passo 3 Defina uma regra de alarme.

- **Rule Type**: selecione **Threshold alarm**.
- **Monitored Object**: selecione **Command input**.
- Digite ` **sum(problem_gauge{clusterName="test"}) by (podIP,type)**` na caixa de texto.



- **Alarm Condition**: acionar um alarme importante se o valor médio for maior ou igual a 1 por um tempo consecutivo em um período de monitoramento.



- (Opcional) **Alarm notification**: para ser notificado de alarmes por e-mail ou mensagem SMS, configure as regras de ação para a regra de alarme. Se nenhuma regra de ação estiver disponível, você poderá criar uma.

----Fim

Coletar métricas de Prometheus

O pod do daemon de NPD expõe dados métricos do Prometheus na porta 19901. Por padrão, o pod de NPD é adicionado com a anotação **metrics.alpha.kubernetes.io/custom-endpoints: '[{"api":"prometheus","path":"/metrics","port":"19901","names":""}]'**. Você pode criar um coletor de Prometheus para identificar e obter métricas de NPD do **http://{{NpdPodIP}}:{{NpdPodPort}}/metrics**.

NOTA

Se a versão do complemento NPD for anterior a 1.16.5, a porta exposta das métricas do Prometheus é **20257**.

Atualmente, os dados da métrica incluem **problem_counter** e **problem_gauge**, como mostrado abaixo.

```
# HELP problem_counter Number of times a specific type of problem have occurred.
# TYPE problem_counter counter
problem_counter{reason="DockerHung"} 0
problem_counter{reason="DockerStart"} 0
problem_counter{reason="EmptyDirVolumeGroupStatusError"} 0
...
# HELP problem_gauge Whether a specific type of problem is affecting the node or
not.
# TYPE problem_gauge gauge
problem_gauge{reason="CNIIsDown",type="CNIProblem"} 0
problem_gauge{reason="CNIIsUp",type="CNIProblem"} 0
problem_gauge{reason="CRIIsDown",type="CRIProblem"} 0
problem_gauge{reason="CRIIsUp",type="CRIProblem"} 0
..
```

13.5 Kubernetes Dashboard

Introdução

O Kubernetes Dashboard é uma interface de usuário baseada na Web de propósito geral para clusters do Kubernetes. Ele permite que os usuários gerenciem aplicações em execução no cluster e solucionem problemas, bem como gerenciem o próprio cluster, executando comandos.

Com o Kubernetes Dashboard, você pode:

- Implementar aplicações em contêiner em um cluster do Kubernetes.
- Diagnosticar problemas de aplicações em contêiner.
- Gerenciar recursos de cluster.
- Exibir aplicações em execução em um cluster.
- Criar e modificar recursos do Kubernetes (como Implementações, tarefas e DaemonSets).
- Verificar os erros que ocorrem em um cluster.

Por exemplo, você pode escalar uma Implementação, executar uma atualização contínua, reiniciar um pod ou implementar uma nova aplicação.

Comunidade de código aberto: **https://github.com/kubernetes/dashboard**

Instalar o complemento

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster. Escolha **Add-ons** no painel de navegação, localize **Kubernetes Dashboard** à direita e clique em **Install**.

Passo 2 Na página **Configuration**, configure os seguintes parâmetros:

- **Certificate Configuration**: configurar um certificado para o dashboard.
  - Usar uma certificação personalizada
    - **Certificate File**: clique em para exibir o arquivo de certificado de exemplo.
    - **Private Key**: clique em para ver o exemplo de chave privada.
  - Usar um certificado padrão

AVISO

O certificado padrão gerado pelo dashboard é inválido, o que afeta o acesso normal ao dashboard por meio de um navegador. Recomendamos que você faça o upload manual de um certificado válido para que o navegador possa verificar seu acesso e proteger sua conexão.

Passo 3 Clique em **Install**.

----Fim

Acessar o complemento dashboard

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster. Escolha **Add-ons** no painel de navegação. Na página exibida, verifique se o complemento dashboard está no estado **Running** e clique em **Access**.

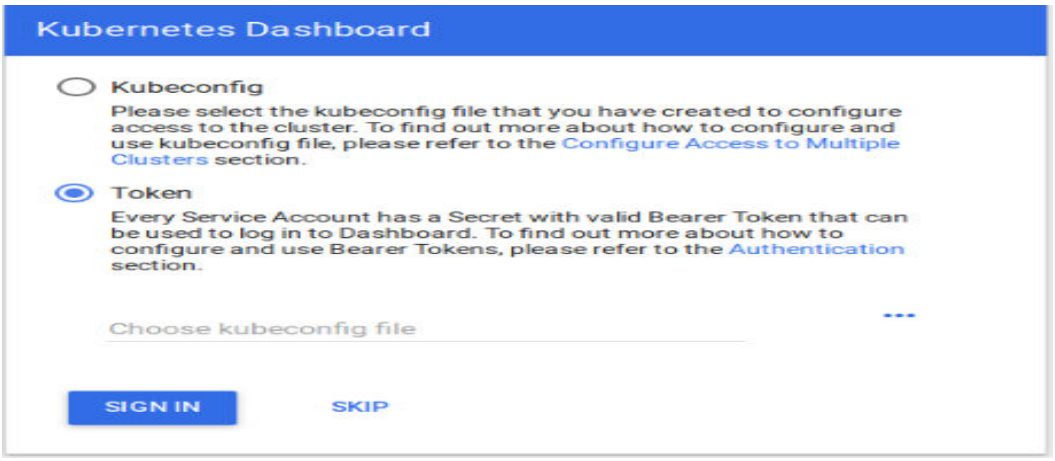
Passo 2 Copie o token na caixa de diálogo exibida.

Passo 3 Na página de logon do dashboard, selecione **Token**, cole o token copiado e clique em **SIGN IN**.

NOTA

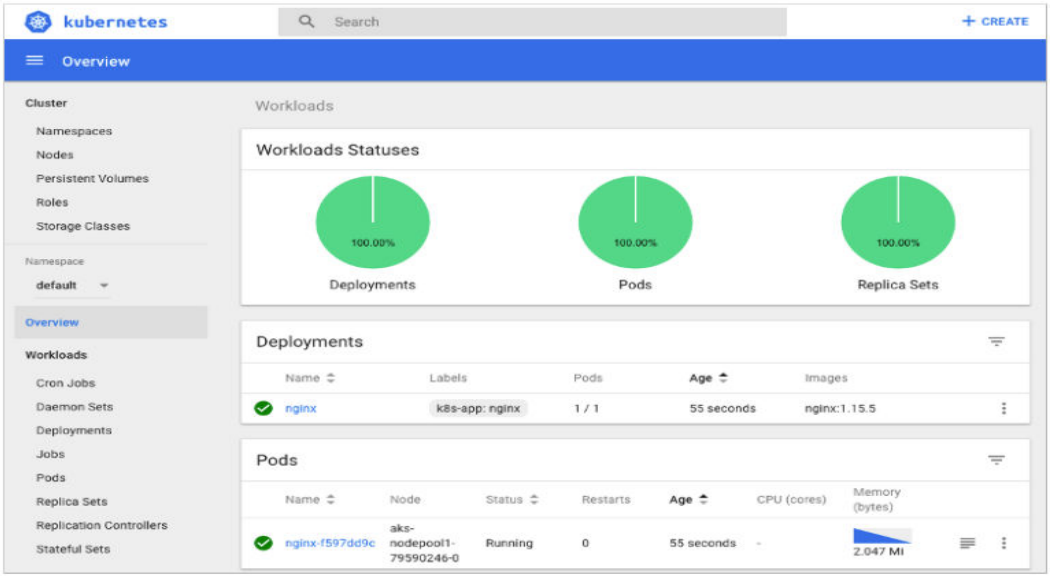
Por padrão, este complemento não suporta logon usando kubeconfig autenticado por certificado. Você é aconselhado a usar o modo token para logon. Para obter detalhes, consulte **https://github.com/kubernetes/dashboard/issues/2474#issuecomment-348912376**.

Figura 13-3 Logon do token



Passo 4 Veja a página do dashboard como mostrado em **Figura 13-4**.

Figura 13-4 Página de dashboard



----Fim

Modificar permissões

Depois que o dashboard é instalado, a função inicial só pode exibir a maioria dos recursos que são exibidos no dashboard. Para solicitar as permissões para executar outras operações no dashboard, modifique os recursos de autorização do RBAC em segundo plano.

Procedimento

Modifique a regra **kubernetes-dashboard-minimal** no ClusterRole.

Para obter detalhes sobre como usar a autorização do RBAC, visite **https://kubernetes.io/docs/reference/access-authn-authz/rbac/**.

Componentes

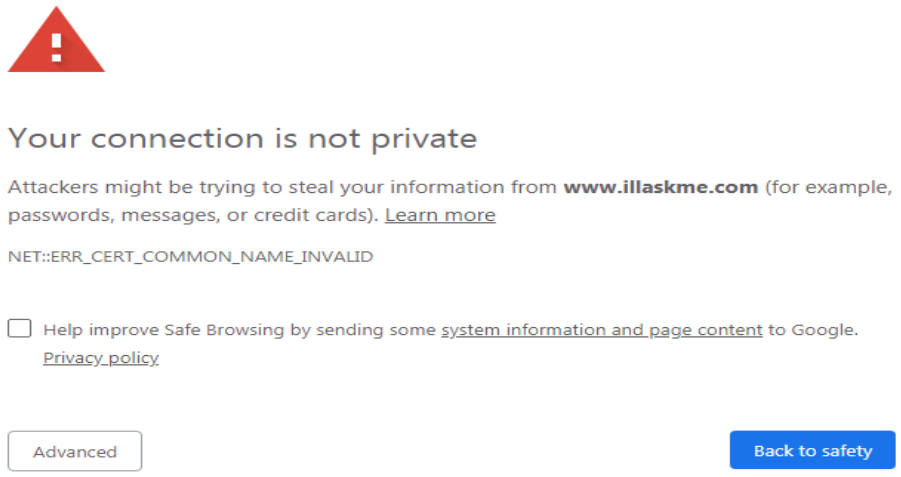
Tabela 13-26 Componentes do dashboard

Componente	Descrição	Tipo de recurso
Dashboard	IU de monitoramento visualizada	Implementação

Solução de problemas de acesso

Quando o Google Chrome é usado para acessar o dashboard, a mensagem de erro "ERR_CERT_INVALID", em vez da página de logon, é exibida. A causa possível é que o certificado padrão gerado pelo dashboard não passa na verificação do Google Chrome. Existem duas soluções para este problema:

Figura 13-5 Mensagem de erro exibida no Google Chrome



- Solução 1: usar o navegador Firefox para acessar o dashboard. Na área **Exceptions** da página **Proxy Settings**, adicione o endereço do dashboard aos endereços que ignorarão o servidor proxy. Em seguida, a página de logon do dashboard será exibida.
- Solução 2: iniciar o Google Chrome com o sinalizador **--ignore-certificate-errors** para ignorar o erro de certificado.

  Windows: salve o endereço do ashboard. Feche todas as janelas ativas do Google Chrome. Pressione a tecla do Windows + R para exibir a caixa de diálogo **Run**. Digite **chrome --ignore-certificate-errors** na caixa de diálogo **Run** para abrir uma nova janela do Google Chrome. Na barra de endereços, digite o endereço do dashboard para abrir a página de logon.

13.6 Autoscaler de cluster do CCE

Introdução

O Autoscaler é um controlador importante do Kubernetes. Ele suporta escalabilidade de microsserviços e é fundamental para o design sem servidor.

Quando o uso da CPU ou da memória de um microsserviço é muito alto, o escalonamento automático de pods horizontais é acionado para adicionar pods e reduzir a carga. Esses pods podem ser reduzidos automaticamente quando a carga é baixa, permitindo que o microsserviço seja executado da forma mais eficiente possível.

O CCE simplifica a criação, a atualização e o dimensionamento manual de clusters do Kubernetes, nos quais as cargas de tráfego mudam ao longo do tempo. Para equilibrar o uso de recursos e o desempenho da carga de trabalho dos nós, o Kubernetes introduz o complemento Autoscaler para ajustar automaticamente o número de nós de um cluster com base no uso de recursos necessário para cargas de trabalho implementadas no cluster. Para mais detalhes, consulte **Criação de uma política de dimensionamento de nós**.

Comunidade de código aberto: **https://github.com/kubernetes/autoscaler**

Como funciona o complemento

Autoscaler controla expansão e redução automáticas.

- **Expansão automática**

  Você pode escolher um dos seguintes métodos:

  – Se os pods em um cluster não puderem ser agendados devido a nós de trabalho insuficientes, o dimensionamento do cluster será acionado para adicionar nós. Os nós a serem adicionados têm a mesma especificação configurada para o pool de nós ao qual os nós pertencem.

    Expansão automática será realizada quando:

    - Os recursos de nó são insuficientes.
    - Nenhuma política de afinidade de nó é definida na configuração de agendamento do pod. Se um nó tiver sido configurado como um nó de afinidade para pods, nenhum nó não será adicionado automaticamente quando os pods não puderem ser agendados. Para obter detalhes sobre como configurar a política de afinidade de nó, consulte **Política de agendamento (afinidade/antiafinidade)**.

  – Quando o cluster atende à política de dimensionamento do nó, a expansão de cluster também é acionada. Para mais detalhes, consulte **Criação de uma política de dimensionamento de nós**.

  NOTA

  O complemento segue a política "No Less, No More". Por exemplo, se três núcleos forem necessários para criar um pod e o sistema oferecer suporte a nós de quatro e oito núcleos, o Autoscaler criará preferencialmente um nó de quatro núcleos.

- **Redução automática**

Quando um nó de cluster fica ocioso por um período (10 minutos por padrão), o dimensionamento de cluster é acionado e o nó é excluído automaticamente. No entanto, um nó não pode ser excluído de um cluster se os seguintes pods existirem:

- Pods que não atendem aos requisitos específicos definidos nos Orçamentos de interrupção de pods (**PodDisruptionBudget**)
- Pods que não podem ser programados para outros nós devido a restrições, como políticas de afinidade e antiafinidade
- Pods que têm a anotação **cluster-autoscaler.kubernetes.io/safe-to-evict: 'false'**
- Pods (exceto aqueles criados por DaemonSets no namespace do kube-system) que existem no namespace do kube-system no nó
- Pods que não são criados pelo controlador (Implementação/ReplicaSet/tarefa/StatefulSet)

NOTA

Quando um nó atende às condições de dimensionamento, o Autoscaler adiciona a mancha **DeletionCandidateOfClusterAutoscaler** ao nó com antecedência para evitar que pods sejam programados para o nó. Após a desinstalação do complemento Autoscaler, se a mancha ainda existir no nó, exclua-a manualmente.

Restrições

- Verifique se há recursos suficientes para instalar o complemento.
- Somente os **nós de VM com pagamento por uso** podem ser adicionados ou removidos pelo Autoscaler.
- O pool de nós padrão não oferece suporte ao dimensionamento automático. Para mais detalhes, consulte **Descrição de DefaultPool**.
- Quando o Autoscaler é usado, algumas manchas ou anotações podem afetar o dimensionamento automático. Portanto, não use as seguintes manchas ou anotações em clusters:
  - **ignore-taint.cluster-autoscaler.kubernetes.io**: a mancha funciona em nós. O Autoscaler do Kubernetes nativo suporta proteção contra expansões anormais e avalia periodicamente a proporção de nós disponíveis no cluster. Quando a proporção de nós não prontos exceder 45%, a proteção será acionada. Nesse caso, todos os nós com a mancha **ignore-taint.cluster-autoscaler.kubernetes.io** no cluster são filtrados do modelo do Autoscaler e registrados como nós não prontos, que afeta o dimensionamento do cluster.
  - **cluster-autoscaler.kubernetes.io/enable-ds-eviction**: a anotação funciona em pods, o que determina se os pods de DaemonSet podem ser removidos pelo Autoscaler. Para mais detalhes, veja **Rótulos, anotações e manchas bem conhecidos**.

Instalar o complemento

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster. Escolha **Add-ons** no painel de navegação, localize o **CCE Cluster Autoscaler** à direita e clique em **Install**.

Passo 2 Na página **Install Add-on**, configure as especificações.

Tabela 13-27 Configuração de especificações

Parâmetro	Descrição
Pods	Número de pods para o complemento. A alta disponibilidade não é possível com um único pod. Se ocorrer um erro no nó em que o pod do complemento é executado, o complemento falhará.
Containers	Ajuste o número dos pods do Autoscaler e suas cotas de CPU e memória com base na escala de cluster. Para mais detalhes, consulte **Tabela 13-28**.

Tabela 13-28 Cotas recomendadas do Autoscaler

Nós	Pods	VCPUs solicitadas	Limite da vCPU	Memória solicitada	Limite de memória
50	2	1000m	1000m	1000 MiB	1000 MiB
200	2	4000m	4000m	2000 MiB	2000 MiB
1000	2	8000m	8000m	8000 MiB	8000 MiB
2000	2	8000m	8000m	8000 MiB	8000 MiB

Passo 3 Configure os parâmetros do complemento.

Tabela 13-29 Parâmetros

Parâmetro	Descrição
Total Nodes	Número máximo de nós que podem ser gerenciados pelo cluster, dentro dos quais o dimensionamento do cluster é realizado.
Total CPUs	Soma máxima de núcleos de CPU de todos os nós em um cluster, dentro do qual a escalabilidade do cluster é executada.
Total Memory (GB)	A soma máxima da memória de todos os nós em um cluster, dentro do qual a expansão de cluster é executado.

Passo 4 Configure as políticas de agendamento para o complemento.

NOTA

- As políticas de agendamento não entram em vigor em instâncias complementares do tipo DaemonSet.
- Ao configurar a implementação de várias AZs ou a afinidade de nó, verifique se há nós que atendem à política de agendamento e se os recursos são suficientes no cluster. Caso contrário, o complemento não pode ser executado.

Tabela 13-30 Configurações para programação de complementos

Parâmetro	Descrição
Multi-AZ Deployment	● **Preferred**: os pods de Implementação do complemento serão agendados preferencialmente para nós em diferentes AZs. Se todos os nós no cluster forem implementados na mesma AZ, os pods serão agendados para essa AZ. ● **Forcible**: os pods de Implementação do complemento serão forçosamente agendados para nós em diferentes AZs. Se houver menos AZs do que pods, os pods extras não funcionarão.
Node Affinity	● **Incompatibility**: a afinidade de nó está desabilitada para o complemento. ● **Node Affinity**: especifique os nós em que o complemento é implementado. Se você não especificar os nós, o complemento será agendado aleatoriamente com base na política de agendamento de cluster padrão. ● **Specified Node Pool Scheduling**: especifique o pool de nós em que o complemento é implementado. Se você não especificar o pool de nós, o complemento será agendado aleatoriamente com base na política de agendamento de cluster padrão. ● **Custom Policies**: insira os rótulos dos nós em que o complemento será implementado para políticas de agendamento mais flexíveis. Se você não especificar rótulos de nó, o complemento será agendado aleatoriamente com base na política de agendamento de cluster padrão. Se várias políticas de afinidade personalizadas estiverem configuradas, certifique-se de que existem nós que atendam a todas as políticas de afinidade no cluster. Caso contrário, o complemento não pode ser executado.
Taints and Tolerations	O uso de manchas e tolerâncias permite (não forçosamente) que Implementação do complemento seja agendada para um nó com as manchas correspondentes e controla as políticas de despejo de Implementação depois que o nó onde a Implementação está localizada é contaminado. O complemento adiciona a política de tolerância padrão para as manchas **node.kubernetes.io/not-ready** e **node.kubernetes.io/unreachable**, respectivamente. A janela de tempo de tolerância é 60s. Para mais detalhes, consulte **Manchas e tolerâncias**.

Passo 5 Após a conclusão da configuração, clique em **Install**.

----Fim

Componentes

Tabela 13-31 Autoscaler

Componente	Descrição	Tipo de recurso
Autoscaler	Dimensionamento automático para clusters do Kubernetes	Implementação

Período de resfriamento de redução

Os intervalos de resfriamento de redução podem ser configurados nas configurações do pool de nós e nas configurações do complemento Autoscaler.

Intervalo de resfriamento de redução configurado em um pool de nós

Esse intervalo indica o período durante o qual os nós adicionados ao pool de nós atual após uma operação de expansão não podem ser excluídos. Esse intervalo entra em vigor no nível do pool de nós.

Intervalo de resfriamento de redução configurado no complemento Autoscaler

O intervalo após uma expansão indica o período durante o qual todo o cluster não pode ser dimensionado após o complemento Autoscaler disparar a expansão (devido aos pods, métricas e políticas de dimensionamento não programáveis). Esse intervalo entra em vigor no nível do cluster.

O intervalo depois que um nó é excluído indica o período durante o qual o cluster não pode ser dimensionado após o complemento Autoscaler disparar a redução. Esse intervalo entra em vigor em todo o cluster.

O intervalo após uma falha de redução indica o período durante o qual o cluster não pode ser dimensionado após o complemento Autoscaler disparar a redução. Esse intervalo entra em vigor em todo o cluster.

13.7 Nginx Ingress controller

Introdução

O Kubernetes usa o kube-proxy para expor Serviços e fornecer balanceamento de carga. A implementação está na camada de transporte. Quando se trata de aplicações da Internet, onde um bucket de informações é gerado, o encaminhamento precisa ser mais refinado, controlado de forma precisa e flexível por políticas e balanceadores de carga para oferecer maior desempenho.

É aqui que entram os ingresses. Ingresses fornecem funções de encaminhamento da camada de aplicação, como hosts virtuais, balanceamento de carga, proxy SSL e roteamento HTTP, para Serviços que podem ser acessados diretamente fora de um cluster.

O Kubernetes lançou oficialmente o controlador de ingress baseado em Nginx. O Nginx Ingress controller do CCE usa diretamente modelos e imagens da comunidade. O Nginx Ingress controller gera a configuração do Nginx e armazena a configuração usando ConfigMap. A configuração será gravada nos pods do Nginx por meio da API do Kubernetes.

Desta forma, a configuração do Nginx é modificada e atualizada. Para mais detalhes, consulte **Como funciona o nginx-ingress**.

Você pode visitar a **comunidade de código aberto** para obter mais informações.

NOTA

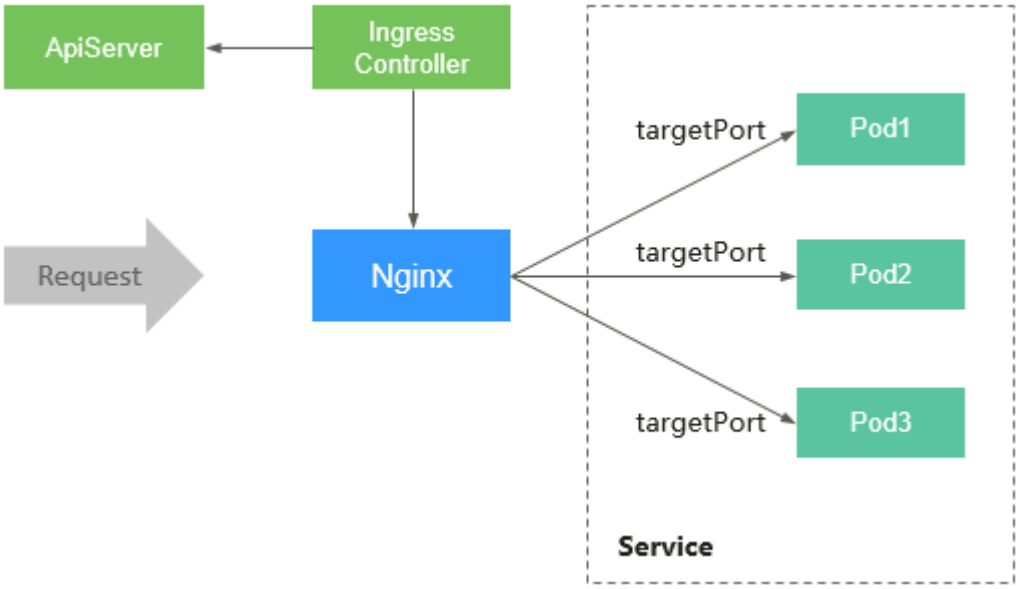
- Ao instalar o complemento, você pode adicionar configurações definindo a configuração do Nginx. As configurações entram em vigor globalmente. Esse parâmetro é gerado pela configuração do arquivo **nginx.conf** e afeta todos os ingresses gerenciados. Você pode pesquisar parâmetros relacionados no **ConfigMap**. Se os parâmetros configurados não estiverem incluídos nas opções listadas no ConfigMap, as configurações não terão efeito.
- Depois que o complemento for instalado, você poderá se interconectar com o Nginx e adicionar **annotations** do Kubernetes a um ingress específico para personalizar seu comportamento ao **criar um ingress** no console do CCE. Para obter detalhes sobre o campo **annotations** do Kubernetes, consulte **Anotações**.
- Não modifique ou exclua manualmente o balanceador de carga e o ouvinte que são criados automaticamente pelo CCE. Caso contrário, a carga de trabalho será anormal. Se você modificou ou excluiu-os por engano, desinstale o complemento nginx-ingress e reinstale-o.

Como funciona o nginx-ingress

nginx-ingress consiste no objeto de ingress, no controlador de ingress e no Nginx. O controlador de ingress monta os ingresses no arquivo de configuração do Nginx (nginx.conf) e recarrega o Nginx para que as configurações alteradas tenham efeito. Quando ele detecta que o pod em um Serviço muda, ele altera dinamicamente a configuração do grupo de servidores upstream do Nginx. Nesse caso, o processo de Nginx não precisa ser recarregado. **Figura 13-6** mostra como funciona o nginx-ingress.

- Um ingress é um grupo de regras de acesso que encaminha solicitações para Serviços especificados com base em nomes de domínio ou URLs. Ingresses são armazenados no serviço de armazenamento de objetos etcd e são adicionados, excluídos, modificados e consultados por meio de APIs.
- O controlador de ingress monitora as alterações de objetos de recursos, como ingresses, Serviços, pontos de extremidade, segredos (principalmente certificados e chaves TLS), nós e ConfigMaps em tempo real e executa operações automaticamente no Nginx.
- O Nginx implementa balanceamento de carga e controle de acesso na camada de aplicação.

Figura 13-6 Princípios de funcionamento do nginx-ingress



Restrições

- Esse complemento pode ser instalado somente em clusters v1.15 ou posterior.
- **kubernetes.io/ingress.class: "nginx"** deve ser adicionado à anotação do Ingress criado por meio da API.
- Os balanceadores de carga dedicados devem ser do tipo de rede (TCP/UDP) que suportam redes privadas (com um IP privado).
- O nó onde o nginx-ingress-controller está em execução e os contêineres em execução no nó não podem acessar o Nginx Ingress. Nesse caso, execute a implementação de antiafinidade para as cargas de trabalho e o nginx-ingress-controller. Para mais detalhes, consulte **Implementação de antiafinidade para cargas de trabalho e nginx-ingress-controller**.

Pré-requisitos

Antes de criar uma carga de trabalho, você deve ter um cluster disponível. Se nenhum cluster estiver disponível, crie um de acordo com **Compra de um cluster do CCE**.

Instalar o complemento

NOTA

- A vulnerabilidade **CVE-2021-25746** foi corrigida no nginx-ingress-controller da v1.2.0 (correspondente ao complemento do CCE nginx-ingress 2.1.0). **Regras** são adicionadas para desabilitar algumas anotações propensas a acesso não autorizado.
- A vulnerabilidade **CVE-2021-25745** foi corrigida no nginx-ingress-controller da v1.2.0 (correspondente ao complemento do CCE nginx-ingress 2.1.0). **Regras** são adicionadas para desabilitar alguns caminhos de acesso propensos a acesso não autorizado.

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster. Escolha **Add-ons** no painel de navegação, localize **NGINX Ingress Controller** na direita e clique em **Install**.

Passo 2 Na página **Install Add-on**, configure as especificações.

Tabela 13-32 Configuração de nginx-ingress

Parâmetro	Descrição
Add-on Specifications	As especificações podem ser **Custom**.
Instances	Se você selecionar **Custom**, poderá ajustar o número de pods conforme necessário.
Containers	Se você selecionar **Custom**, poderá ajustar as especificações do contêiner conforme necessário.

Passo 3 Configure os parâmetros do complemento.

- **Load Balancer**: selecione um balanceador de carga compartilhado ou dedicado. Se nenhum balanceador de carga estiver disponível, crie um. O balanceador de carga tem pelo menos dois ouvintes, e as portas 80 e 443 não são ocupadas por ouvintes.
- **Nginx Parameters**: a configuração do arquivo **nginx.conf** afetará todos os ingresses gerenciados. Você pode pesquisar parâmetros relacionados através do **ConfigMaps**. Se os parâmetros configurados não estiverem incluídos nas opções listadas nesses ConfigMaps, os parâmetros não terão efeito.

  Por exemplo, você pode usar o parâmetro **keep-alive-requests** para descrever como definir o número máximo de solicitações para manter conexões ativas como 100.

  ```
  {
      "keep-alive-requests": "100"
  }
  ```

- **Default 404 Service**: por padrão, o serviço 404 fornecido pelo complemento é usado. Para personalizar o serviço 404, insira o namespace/nome do serviço. Se o serviço não existir, a instalação do complemento falhará.

Passo 4 Configure as políticas de agendamento para o complemento.

NOTA

- As políticas de agendamento não entram em vigor em instâncias complementares do tipo DaemonSet.
- Ao configurar a implementação de várias AZs ou a afinidade de nó, verifique se há nós que atendem à política de agendamento e se os recursos são suficientes no cluster. Caso contrário, o complemento não pode ser executado.

Tabela 13-33 Configurações para programação de complementos

Parâmetro	Descrição
Multi-AZ Deployment	- **Preferred**: os pods de Implementação do complemento serão agendados preferencialmente para nós em diferentes AZs. Se todos os nós no cluster forem implementados na mesma AZ, os pods serão agendados para essa AZ. - **Forcible**: os pods de Implementação do complemento serão forçosamente agendados para nós em diferentes AZs. Se houver menos AZs do que pods, os pods extras não funcionarão.

Parâmetro	Descrição
Node Affinity	● **Incompatibility**: a afinidade de nó está desabilitada para o complemento. ● **Node Affinity**: especifique os nós em que o complemento é implementado. Se você não especificar os nós, o complemento será agendado aleatoriamente com base na política de agendamento de cluster padrão. ● **Specified Node Pool Scheduling**: especifique o pool de nós em que o complemento é implementado. Se você não especificar o pool de nós, o complemento será agendado aleatoriamente com base na política de agendamento de cluster padrão. ● **Custom Policies**: insira os rótulos dos nós em que o complemento será implementado para políticas de agendamento mais flexíveis. Se você não especificar rótulos de nó, o complemento será agendado aleatoriamente com base na política de agendamento de cluster padrão. Se várias políticas de afinidade personalizadas estiverem configuradas, certifique-se de que existem nós que atendam a todas as políticas de afinidade no cluster. Caso contrário, o complemento não pode ser executado.
Taints and Tolerations	O uso de manchas e tolerâncias permite (não forçosamente) que Implementação do complemento seja agendada para um nó com as manchas correspondentes e controla as políticas de despejo de Implementação depois que o nó onde a Implementação está localizada é contaminado. O complemento adiciona a política de tolerância padrão para as manchas **node.kubernetes.io/not-ready** e **node.kubernetes.io/unreachable**, respectivamente. A janela de tempo de tolerância é 60s. Para mais detalhes, consulte **Manchas e tolerâncias**.

Passo 5 Clique em **Install**.

----Fim

Componentes

Tabela 13-34 componente de nginx-ingress

Componente	Descrição	Tipo de recurso
nginx-ingress	Nginx Ingress controller, que fornece roteamento e encaminhamento flexíveis para clusters	Implementação

Implementação de antiafinidade para cargas de trabalho e nginx-ingress-controller

O nó onde o nginx-ingress-controller está em execução e os contêineres em execução no nó não podem acessar o Nginx Ingress. Para evitar esse problema, configure uma regra de antiafinidade para informar ao agendador para não co-localizar a carga de trabalho e o nginx-ingress-controller no mesmo nó.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  strategy:
    type: RollingUpdate
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - image: nginx:aplpine
        imagePullPolicy: IfNotPresent
        name: nginx
      imagePullSecrets:
      - name: default-secret

                    affinity:

                            podAntiAffinity:

                              requiredDuringSchedulingIgnoredDuringExecution:

                                - labelSelector:

                                    matchExpressions:
                    # Use the labels of nginx-ingress-controller to implement
anti-affinity.
                                      - key: app

                                        operator: In

                                        values:

                                          - nginx-ingress

                                      - key: component

                                        operator: In

                                        values:
```



```
                - controller

      namespaces:

        - kube-system

      topologyKey: kubernetes.io/hostname
```

13.8 Kubernetes Metrics Server

A partir da versão 1.8, o Kubernetes fornece métricas de uso de recursos, como o uso da CPU e da memória do contêiner, por meio da Metrics API. Essas métricas podem ser acessadas diretamente pelos usuários (por exemplo, usando o comando **kubectl top**) ou usadas por controladores (por exemplo, Horizontal Pod Autoscaler) em um cluster para tomada de decisões. O componente específico é metrics-server, que é usado para substituir o heapster para fornecer as funções semelhantes. O heapster foi gradualmente abandonado desde a v1.11.

metrics-server é um agregador para monitoramento de dados de recursos principais do cluster. Você pode instalar rapidamente este complemento no console do CCE.

Depois de instalar este complemento, pode criar políticas HPA. Para mais detalhes, consulte **Criação de uma política HPA para dimensionamento automático da carga de trabalho**.

O projeto oficial da comunidade e a documentação estão disponíveis em **https://github.com/kubernetes-sigs/metrics-server**.

Instalar o complemento

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster. Escolha **Add-ons** no painel de navegação, localize **Kubernetes Metrics Server** à direita e clique em **Install**.

Passo 2 Na página **Install Add-on**, configure as especificações.

Tabela 13-35 Configuração de metrics-server

Parâmetro	Descrição
Add-on Specifications	Selecione **Single**, **Custom** ou **HA** para **Add-on Specifications**.
Instances	Número de pods que serão criados para corresponder às especificações do complemento selecionado. Se você selecionar **Custom**, poderá ajustar o número de pods conforme necessário.
Containers	As cotas de CPU e memória do contêiner permitidas para as especificações adicionais selecionadas. Se você selecionar **Custom**, poderá ajustar as especificações do contêiner conforme necessário.

Passo 3 Configure as políticas de agendamento para o complemento.

NOTA

- As políticas de agendamento não entram em vigor em instâncias complementares do tipo DaemonSet.
- Ao configurar a implementação de várias AZs ou a afinidade de nó, verifique se há nós que atendem à política de agendamento e se os recursos são suficientes no cluster. Caso contrário, o complemento não pode ser executado.

Tabela 13-36 Configurações para programação de complementos

Parâmetro	Descrição
Multi-AZ Deployment	• **Preferred**: os pods de Implementação do complemento serão agendados preferencialmente para nós em diferentes AZs. Se todos os nós no cluster forem implementados na mesma AZ, os pods serão agendados para essa AZ. • **Forcible**: os pods de Implementação do complemento serão forçosamente agendados para nós em diferentes AZs. Se houver menos AZs do que pods, os pods extras não funcionarão.
Node Affinity	• **Incompatibility**: a afinidade de nó está desabilitada para o complemento. • **Node Affinity**: especifique os nós em que o complemento é implementado. Se você não especificar os nós, o complemento será agendado aleatoriamente com base na política de agendamento de cluster padrão. • **Specified Node Pool Scheduling**: especifique o pool de nós em que o complemento é implementado. Se você não especificar o pool de nós, o complemento será agendado aleatoriamente com base na política de agendamento de cluster padrão. • **Custom Policies**: insira os rótulos dos nós em que o complemento será implementado para políticas de agendamento mais flexíveis. Se você não especificar rótulos de nó, o complemento será agendado aleatoriamente com base na política de agendamento de cluster padrão. Se várias políticas de afinidade personalizadas estiverem configuradas, certifique-se de que existem nós que atendam a todas as políticas de afinidade no cluster. Caso contrário, o complemento não pode ser executado.
Taints and Tolerations	O uso de manchas e tolerâncias permite (não forçosamente) que Implementação do complemento seja agendada para um nó com as manchas correspondentes e controla as políticas de despejo de Implementação depois que o nó onde a Implementação está localizada é contaminado. O complemento adiciona a política de tolerância padrão para as manchas **node.kubernetes.io/not-ready** e **node.kubernetes.io/unreachable**, respectivamente. A janela de tempo de tolerância é 60s. Para mais detalhes, consulte **Manchas e tolerâncias**.

Passo 4 Clique em **Install**.

----Fim

Componentes

Tabela 13-37 Componentes de metrics-server

Componente	Descrição	Tipo de recurso
metrics-server	Agregador para os dados monitorados dos recursos principais do cluster, que é usado para coletar e agregar métricas de uso de recursos obtidas por meio da Metrics API no cluster	Implementação

13.9 HPA de CCE avançado

cce-hpa-controller é um complemento desenvolvido pelo CCE, que pode ser usado para dimensionar de forma flexível as Implantações com base em métricas como uso de CPU e uso de memória.

Depois de instalar este complemento, pode criar políticas CustomedHPA. Para mais detalhes, consulte **Criação de uma política CustomedHPA para o dimensionamento automático da carga de trabalho**.

Principais funções

- O dimensionamento pode ser realizado com base na porcentagem do número atual de pods.
- A etapa mínima de dimensionamento pode ser definida.
- Diferentes operações de escala podem ser executadas com base nos valores métricos reais.

Restrições

- Esse complemento pode ser instalado somente em clusters da v1.15 ou posterior.
- Se a versão do cce-hpa-controller for anterior a 1.2.11, o complemento **Prometheus** deve ser instalado. Se a versão do cce-hpa-controller for 1.2.11 ou posterior, os complementos que podem fornecer a API de métricas devem ser instalados. Selecione um dos seguintes complementos com base na versão do cluster e nos requisitos reais.
  - **Kubernetes Metrics Server**: fornece métricas básicas de uso de recursos, como uso de CPU e memória do contêiner. É suportado por todas as versões de cluster.
  - **Monitoramento de cluster da nuvem nativa**: fornece métricas personalizadas além das métricas básicas de recursos. Registre o Prometheus como o serviço que fornece a API de métricas. Para mais detalhes, consulte **Fornecer métricas de recursos por meio de Metrics API**. Este suplemento suporta clusters v1.17 ou posterior.
  - **Prometheus (EOM)**: fornece métricas personalizadas além das métricas básicas de recursos. Registre o Prometheus como o serviço que fornece a API de métricas.

Para mais detalhes, consulte **Fornecer métricas de recursos por meio de Metrics API**. Este suplemento suporta apenas clusters de v1.21 ou anterior.

Instalar o complemento

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster. Clique em **Add-ons** no painel de navegação, localize **CCE Advanced HPA** à direita e clique em **Install**.

Passo 2 Na página **Install Add-on**, configure as especificações.

Tabela 13-38 Configuração de cce-hpa-controller

Parâmetro	Descrição
Add-on Specifications	Selecione **Single** ou **Custom** para **Add-on Specifications**. **NOTA** Os complementos de instância única são usados apenas para verificação de serviço. Em implementações comerciais, selecione **Custom** com base nas especificações do cluster. As especificações do cce-hpa-controller são decididas pelo número total de contêineres no cluster e pelo número de políticas de escala. É aconselhável configurar 500m de CPU e 1.000 MiB de memória para cada 5.000 contêineres e 100m de CPU e 500 MiB de memória para cada 1.000 políticas de dimensionamento.
Instances	Número de pods que serão criados para corresponder às especificações do complemento selecionado. Se você selecionar **Custom**, poderá ajustar o número de pods conforme necessário.
Containers	As cotas de CPU e memória do contêiner permitidas para as especificações adicionais selecionadas. Se você selecionar **Custom**, poderá ajustar as especificações do contêiner conforme necessário.

Passo 3 Selecione **Single** ou **Custom** para **Add-on Specifications**.

- **Pods**: defina o número de pods com base nos requisitos de serviço.
- **Containers**: defina uma cota de contêiner adequada com base nos requisitos de serviço.

Passo 4 Configure as políticas de agendamento para o complemento.

NOTA

- As políticas de agendamento não entram em vigor em instâncias complementares do tipo DaemonSet.
- Ao configurar a implementação de várias AZs ou a afinidade de nó, verifique se há nós que atendem à política de agendamento e se os recursos são suficientes no cluster. Caso contrário, o complemento não pode ser executado.

Tabela 13-39 Configurações para programação de complementos

Parâmetro	Descrição
Multi-AZ Deployment	● **Preferred**: os pods de Implementação do complemento serão agendados preferencialmente para nós em diferentes AZs. Se todos os nós no cluster forem implementados na mesma AZ, os pods serão agendados para essa AZ. ● **Forcible**: os pods de Implementação do complemento serão forçosamente agendados para nós em diferentes AZs. Se houver menos AZs do que pods, os pods extras não funcionarão.
Node Affinity	● **Incompatibility**: a afinidade de nó está desabilitada para o complemento. ● **Node Affinity**: especifique os nós em que o complemento é implementado. Se você não especificar os nós, o complemento será agendado aleatoriamente com base na política de agendamento de cluster padrão. ● **Specified Node Pool Scheduling**: especifique o pool de nós em que o complemento é implementado. Se você não especificar o pool de nós, o complemento será agendado aleatoriamente com base na política de agendamento de cluster padrão. ● **Custom Policies**: insira os rótulos dos nós em que o complemento será implementado para políticas de agendamento mais flexíveis. Se você não especificar rótulos de nó, o complemento será agendado aleatoriamente com base na política de agendamento de cluster padrão. Se várias políticas de afinidade personalizadas estiverem configuradas, certifique-se de que existem nós que atendam a todas as políticas de afinidade no cluster. Caso contrário, o complemento não pode ser executado.
Taints and Tolerations	O uso de manchas e tolerâncias permite (não forçosamente) que Implementação do complemento seja agendada para um nó com as manchas correspondentes e controla as políticas de despejo de Implementação depois que o nó onde a Implementação está localizada é contaminado. O complemento adiciona a política de tolerância padrão para as manchas **node.kubernetes.io/not-ready** e **node.kubernetes.io/unreachable**, respectivamente. A janela de tempo de tolerância é 60s. Para mais detalhes, consulte **Manchas e tolerâncias**.

Passo 5 Clique em **Install**.

----Fim

Componentes

Tabela 13-40 Componentes do cce-hpa-controller

Componente	Descrição	Tipo de recurso
customedhpa-controller	Componente de escalonamento automático do CCE, que escala para dentro ou para fora Implementações com base em métricas como uso da CPU e uso da memória	Implementação

13.10 Mecanismo de estouro de nuvem do CCE para CCI

O mecanismo de estouro de nuvem do CCE para CCI, virtual-kubelet, é desenvolvido com base no Virtual Kubelet, uma implementação de kubelet do Kubernetes de código aberto que se disfarça como um kubelet para conectar o Kubernetes a outras APIs. virtual-kubelet é usado principalmente para estender as APIs do Kubernetes a serviços de contêiner sem servidor, como CCI da Huawei Cloud.

Para ser específico, o kubelet virtual chama a **Cloud Container Instance (CCI)** para criar e gerenciar pods para as Implementações, os StatefulSets e as tarefas implementadas no CCE durante picos de serviço de curto prazo. Desta forma, você pode reduzir o consumo causado pelo escalonamento de cluster.

Comunidade de código aberto: **https://github.com/virtual-kubelet/virtual-kubelet**

O complemento virtual-kubelet fornece as seguintes funções:

- **Cria pods em segundos.** virtual-kubelet cria pods automaticamente na **CCI**, eliminando a sobrecarga de redimensionamento do cluster do CCE.
- Funciona perfeitamente com **SWR** para você usar imagens públicas e privadas.
- Suporta sincronização de eventos, monitoramento, registro em logs, execução de comandos remotos em pods e visualização de status para pods da CCI.
- Permite que os usuários visualizem as informações de capacidade sobre nós elásticos virtuais.
- Suporta conectividade entre pods do CCE e CCI por meio de Serviços.

Restrições

- Somente os clusters padrão do CCE que usam VPCs e clusters do CCE Turbo (usando virtual-kubelet 1.2.5 ou posterior) são compatíveis. Clusters de Arm não são suportados. As instâncias do complemento não serão implementadas nos nós de Arm, se houver, em execução no cluster.
- Os pods do CCE agendados para CCI oferecem suporte aos volumes de ConfigMap, Secret, EmptyDir, DownwardAPI, Projected e PersistentVolumeClaims, e os volumes de DownwardAPI e Projected só podem ser usados no virtual-kubelet 1.3.25 e versões posteriores.
  - emptyDir: subcaminhos não são suportados.
  - PersistentVolumeClaims: somente os tipos de armazenamento em nuvem do SFS e SFS Turbo e as classes de armazenamento CSI são compatíveis.

  - Projected: se uma origem do tipo serviceAccountToken estiver configurada, o token no segredo service-account-token correspondente será montado depois que o pod for agendado para a CCI. O token é válido por um longo tempo e não tem público esperado. Ou seja, as configurações de **expirationSeconds** e **audience** não têm efeito.

- DaemonSets e pods que usam o modo HostNetwork não podem ser agendados para CCI.
- Os pods implementados no CCE e na CCI só podem se comunicar por meio dos Serviços ClusterIP. Os Serviços do CCE ClusterIP não podem ser acessados no init-container.
- Não especifique a porta de verificação de integridade ao interconectar com um Serviço LoadBalancer ou ingress para cargas de trabalho executadas no CCE e na CCI.

  1. Em um cluster do CCE, os contêineres da CCI e os contêineres do CCE usam portas de back-end diferentes registradas no ELB. Se você especificar uma porta de verificação de integridade, algumas verificações de integridade do back-end serão anormais.

  2. Quando clusters diferentes usarem um serviço para se conectar ao ouvinte do mesmo balanceador de carga do ELB, verifique se o modo de verificação de integridade escolhido não afetará o acesso ao serviço.

  3. Permita acesso de **100.125.0.0/16** ao interconectar com um Serviço LoadBalancer compartilhado para cargas de trabalho executadas em CCE e CCI.

- A sub-rede onde o cluster está localizado não pode se sobrepor com 10.247.0.0/16. Caso contrário, a sub-rede entra em conflito com o bloco CIDR de serviço no espaço para nomes da CCI.
- Antes de usar o virtual-kubelet, vá para o console da CCI para conceder à CCI permissão para usar o CCE.
- Depois que o complemento virtual-kubelet é instalado, um namespace chamado "cce-burst-*cluster ID*" é criado na CCI e gerenciado pelo complemento. Não use esse namespace ao criar pods manualmente na CCI.
- O uso de recursos do complemento varia de acordo com o volume de serviço dimensionado para CCI. Os pods, segredos, congfigMaps, PVs e PVCs solicitados pelo serviço ocupam os recursos da VM. Recomendamos que você avalie o uso do serviço e se inscreva para VMs com base nas seguintes especificações: para 1.000 pods e 1.000 ConfigMaps (300 KB), nós com 2 vCPUs e 4 GiB de memória são recomendados. Para 2.000 pods e 2.000 ConfigMaps são recomendados nós com 4 vCPUs e 8 GiB de memória. Para 4.000 pods e 4.000 ConfigMaps são recomendados nós com 8 vCPUs e 16 GiB de memória.
- **Se o agendamento dos recursos dimensionados para CCI falhar, o nó de virtual-kubelet será bloqueado por meia hora, durante a qual os recursos não poderão ser agendados para CCI.** Você pode usar o kubectl para verificar o status do nó do virtual-kubelet no console do cluster do CCE. Se o nó tiver sido bloqueado, desbloqueie-o manualmente.

```
user@imkrz4xzkz30alq-machine:~$ kubectl get node
NAME              STATUS                     ROLES            AGE    VERSION
192.168.182.101   Ready                      <none>           33d    v1.23.0-CCE23.0.1
virtual-kubelet   Ready,SchedulingDisabled   virtual-kubelet  4d5h   v1.19.16-v1.3.4-145-g8114c8a-dev
user@imkrz4xzkz30alq-machine:~$
user@imkrz4xzkz30alq-machine:~$
user@imkrz4xzkz30alq-machine:~$ kubectl uncordon virtual-kubelet
node/virtual-kubelet uncordoned
user@imkrz4xzkz30alq-machine:~$ kubectl get node
NAME              STATUS   ROLES            AGE    VERSION
192.168.182.101   Ready    <none>           33d    v1.23.0-CCE23.0.1
virtual-kubelet   Ready    virtual-kubelet  4d5h   v1.19.16-v1.3.4-145-g8114c8a-dev
user@imkrz4xzkz30alq-machine:~$
```

- As restrições ao uso do gerenciamento de logs para coletar os logs de pods dimensionados para CCI são as seguintes:

- Somente logs em caminhos de arquivo de contêiner podem ser coletados.
- A rede deve ser ativada quando o complemento é instalado porque a rede de Serviço do virtual-kubelet é necessária.
- Os pods dimensionados para CCI não oferecem suporte à atualização a quente das políticas de log. Depois que uma política de log for atualizada, reimplemente os pods dimensionados para CCI para aplicar a modificação.
- A coleta de logs consome memória de pod. É uma boa prática reservar 50 MB de memória para que um pod seja associado a uma política de log. Se o pod estiver associado a várias políticas de log, reserve 5 MB adicionais de memória sempre que uma política de log for associada.
- Um log maior que 250 KB não pode ser coletado.
- Os logs não podem ser coletados de diretórios de montagem especificados, como os diretórios **system**, **device**, **cgroup** e **tmpfs**.
- Os nomes dos arquivos de log a serem coletados associados à mesma política de log no mesmo contêiner devem ser exclusivos. Se houver vários arquivos com o mesmo nome, o coletor coletará apenas o primeiro que detectar.
- Se o nome de um arquivo de log exceder 190 caracteres, o arquivo de log não será coletado.

Cálculo e restrições nas especificações do pod

As especificações do pod são as seguintes:

- As solicitações e os limites de um recurso são definidos com o mesmo valor para todos os contêineres de aplicação e inicialização. Se um limite for configurado, o limite será aplicado. Se nenhum limite for configurado, a solicitação será aplicada.
- Se as solicitações não forem iguais aos limites, o valor maior dos dois itens a seguir será aplicado:
  - Soma dos limites definidos para um recurso em todos os contêineres de aplicação
  - Limite máximo de um recurso em todos os contêineres de inicialização

Restrições às especificações do pod:

- O número de vCPUs em um pod deve ser maior que 0.
- Após o **ajuste automático de recursos**, o número de vCPUs no pod é 48 ou 64, ou varia de 0,25 a 32, a memória é um múltiplo inteiro de 1 GiB e varia de 1 GiB a 512 GiB, e a relação entre CPU e memória varia de 1:2 a 1:8.

Ajuste automático de recursos

Se os pods a serem dimensionados para CCI não atenderem às especificações da CCI e as especificações configuradas não forem superiores a 32 vCPUs e 256 GiB de memória, o virtual-kubelet ajustará automaticamente os recursos do pod.

As regras são as seguintes:

1. Aumente o número de vCPUs de cada contêiner de aplicação e de inicialização (exceto contêineres BestEffort) no pod para um múltiplo inteiro de 0,25 e defina a memória para um valor maior ou igual a 0,2 GiB.
2. Ajuste continuamente os recursos do pod até que o número de vCPUs do pod seja maior que 32 ou a memória seja maior que 256 GiB.

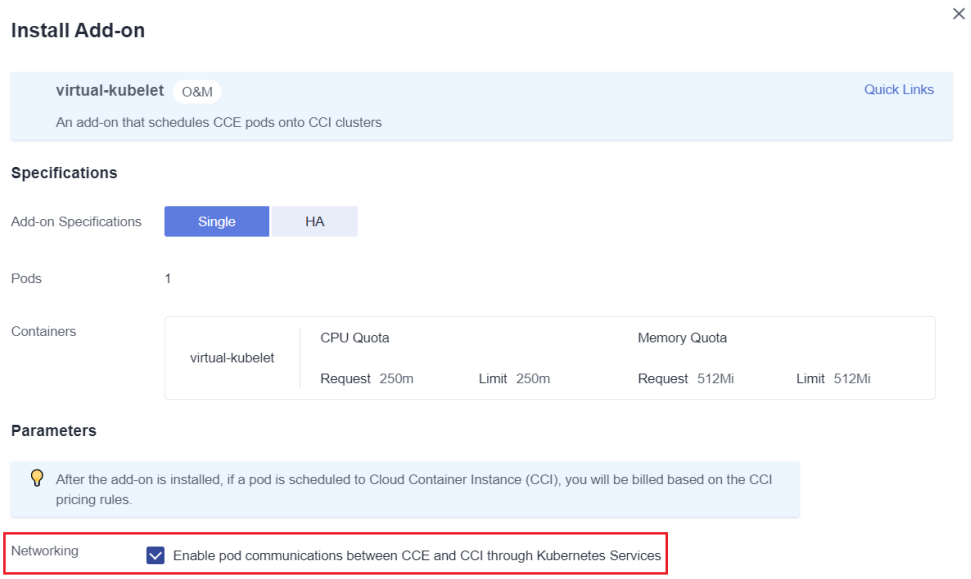
3. Aumente a memória do pod para um múltiplo inteiro de 1 GiB.
4. Se a proporção de memória de pod/vCPUs for menor que 2, aumente a memória de pod para um valor maior ou igual ao dobro do número de vCPUs, bem como um múltiplo inteiro de 1 GiB. Se a proporção de memória de pod/vCPUs for maior que 8, aumente o número de vCPUs de pod para ser maior ou igual a 1/8 da memória, bem como um múltiplo inteiro de 0,25.
5. Os incrementos de vCPU e memória causados pelo ajuste são adicionados ao primeiro contêiner de aplicação não BestEffort.

Instalar o complemento

Passo 1 No console do CCE, clique no nome do cluster para acessar o cluster. Clique em **Add-ons** no painel de navegação, localize **virtual-kubelet** à direita e clique em **Install**.

Passo 2 Na página **Install Add-on**, selecione **Networking** para habilitar as comunicações de pod entre o CCE e a CCI por meio dos Serviços do Kubernetes.

Figura 13-7 Selecionar rede



Passo 3 Clique em **Install**.

NOTA

Depois que a rede é selecionada, um sidecar é injetado no pod em execução na CCI para oferecer suporte ao acesso ao Serviço. O número de contêineres em execução consultados é um a mais do que o número definido, o que é normal.

----Fim

Usar rótulos para definir políticas de dimensionamento

Após instalar o complemento virtual-kubelet, adicione o rótulo **virtual-kubelet.io/burst-to-cci** à carga de trabalho.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: test
```



```
  namespace: default
  labels:
                        virtual-kubelet.io/burst-to-cci: 'auto'    # Scale pods
to CCI.
spec:
  replicas: 2
  selector:
    matchLabels:
      app: test
  template:
    metadata:
      labels:
        app: test
    spec:
      containers:
        - image: 'nginx:perl'
          name: container-0
          resources:
            requests:
              cpu: 250m
              memory: 512Mi
            limits:
              cpu: 250m
              memory: 512Mi
          volumeMounts: []
      imagePullSecrets:
        - name: default-secret
```

NOTA

Adicione o seguinte campo aos **labels** de uma carga de trabalho ou pod no arquivo YAML:

```
virtual-kubelet.io/burst-to-cci: "auto"
```

Os seguintes valores são suportados:

- **auto**: o sistema determina automaticamente se os pods devem ser dimensionados para CCI com base no resultado de pontuação real do agendador. **TaintToleration** agenda preferencialmente os pods para os nós do CCE.
- **localPrefer**: os pods são agendados para a CCI quando os recursos do cluster são insuficientes.
- **enforce**: os pods são agendados à força para CCI.
- **off**: os pods não estão agendados para a CCI.

Usar perfis para gerenciar pods na nuvem e fora dela

Você pode usar perfis para configurar e gerenciar pods, associar perfis a pods usando labelSelector e configurar a política de alocação dos pods associados para alocar ou limitar o número de pods na e fora da nuvem.

Restrições

- Você pode usar o perfil para configurar e gerenciar a política de pods. burst-to-cci ainda é compatível e é anterior ao perfil.
- localPrefer não pode ser configurado para **local** e **cci** ao mesmo tempo.
- As políticas auto e localPrefer podem ser associadas a pods que não foram associados a perfis. A política de imposição não pode ser associada a pods que não foram associados a perfis.
- Atualmente, quando a política localPrefer é configurada no perfil, a configuração do número máximo de local pode se tornar inválida em cenários extremos para evitar problemas globais.
- Quando uma implantação está realizando atualização contínua, é recomendável definir **maxSurge** para um valor tão pequeno quanto possível (por exemplo, 0) para evitar que o

volume de programação da região onde **maxNum** é limitado seja menor do que o esperado.

- Um pod pode ser associado a apenas um perfil, ou seja, o perfil com o maior grau de associação. Depois que um pod é criado, se o rótulo do pod for modificado, o pod não corresponde ao perfil original. Nesse caso, o pod seleciona o perfil com o maior grau de associação para associação. Você pode avaliar o grau de associação de acordo com as seguintes instruções:
  - Calcule a soma de **matchLabels** e **matchExpressions** em **labelSelector** com base em **obejectLables** no perfil. O perfil com a maior soma tem o maior grau de associação.
  - Se dois perfis forem filtrados, selecione aquele cujo nome tenha a menor ordem alfabética.

Descrição do uso

Configurar **local maxNum** e **scaleDownPriority**

```
apiVersion: scheduling.cci.io/v1
kind: ScheduleProfile
metadata:
  name: test-cci-profile
  namespace: default
spec:
  objectLabels:
    matchLabels:
      app: nginx
                    strategy: localPrefer
  location:
                    local:
                    maxNum: 20 # Currently, maxNum cannot be configured for local
and cci at the same time.
                    scaleDownPriority: 10
    cci: {}
status:
  phase: initialized
  restrict:
                    local: 20 # restrict is configured based on location. That
is, local and cci do not appear at the same time.
  used:
    local: 20
    cci: 0
```

Configurar **cci maxNum** e **scaleDownPriority**

```
apiVersion: scheduling.cci.io/v1
kind: ScheduleProfile
metadata:
  name: test-cci-profile
  namespace: default
spec:
  objectLabels:
    matchLabels:
      app: nginx
                strategy: localPrefer
  location:
    local: {}
                cci:
                maxNum: 20 # Currently, maxNum cannot be configured for local and
cci at the same time.
                scaleDownPriority: 10
status:
  phase: initialized
  restrict:
                cci: 20 # restrict is configured based on location. That is,
local and cci do not appear at the same time.
  used:
```



```
    local: 0
    cci: 20
```

Parâmetros:

- **strategy**: política de agendamento. O valor pode ser **auto**, **enforce** ou **localPrefer**.
- O número máximo de pods no IDC off-line e na nuvem e a prioridade de dimensionamento do **scaleDownPriority** podem ser configurados no **location**. O valor de **maxNum** varia de 0 a int32, e o valor de **scaleDownPriority** varia de -100 a 100.

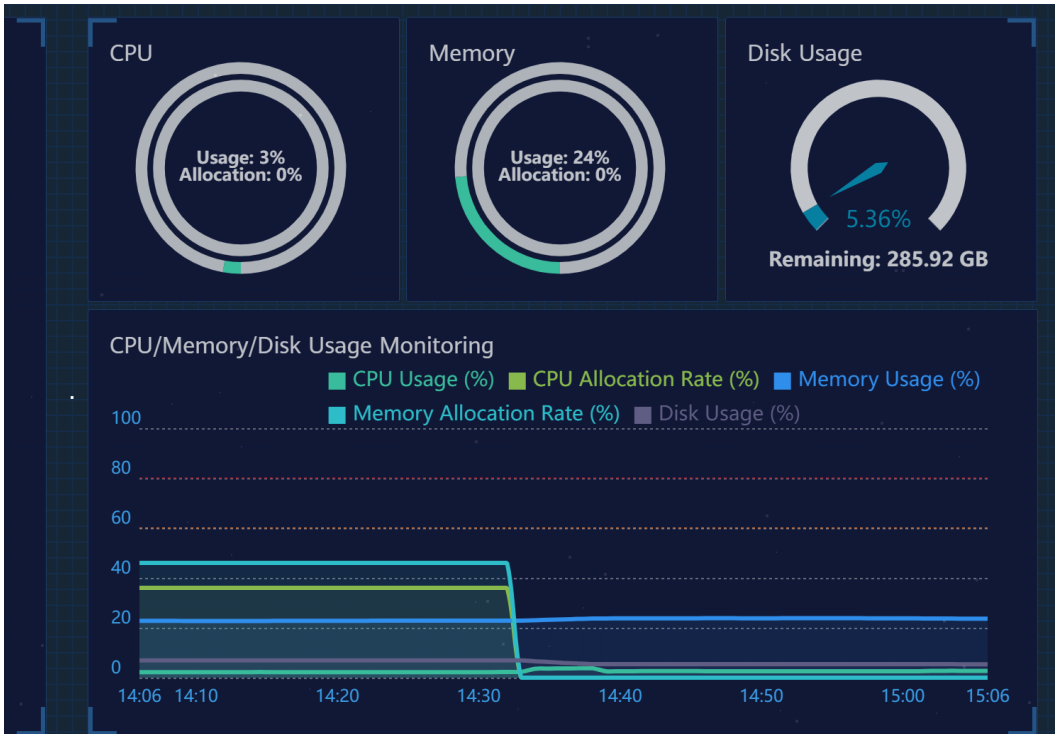
Crie uma Implementação, use **selector** para selecionar o pod que contém **app:nginx** e associe o pod ao perfil criado.

```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: nginx
spec:
  replicas: 10
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: container-1
          image: nginx:latest
          imagePullPolicy: IfNotPresent
          resources:
            requests:
              cpu: 250m
              memory: 512Mi
            limits:
              cpu: 250m
              memory: 512Mi
      imagePullSecrets:
        - name: default-secret
```

Impacto do virtual-kubelet no monitoramento de recursos disponíveis

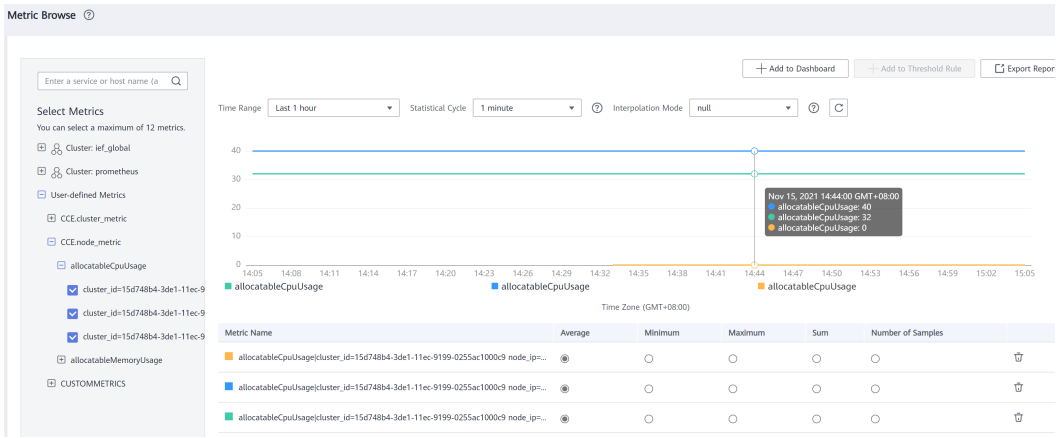
Depois que o complemento virtual-kubelet é instalado, o sistema usa recursos da CCI como recursos de nó de cluster. É equivalente a um nó super-grande para monitoramento. Nesse caso, os recursos da CCI são calculados juntos nos recursos alocáveis na página de monitoramento de cluster do CCE. Conforme mostrado na figura a seguir, após a instalação do complemento virtual-kubelet, a taxa de alocação de CPU/memória diminui drasticamente.

Figura 13-8 Monitoramento do cluster



Você pode exibir a taxa de alocação de um nó específico na página **Nodes** ou no console do AOM.

Figura 13-9 Exibir recursos alocáveis de um nó no console do AOM



13.11 Suíte IA do CCE (GPU NVIDIA)

Introdução

NVIDIA GPU é umcomplemento de gerenciamento de dispositivos que suporta GPUs em contêineres. Para usar nós de GPU em um cluster, esse complemento deve estar instalado.

Restrições

- O driver a ser baixado deve ser um arquivo **.run**.
- Somente os drivers NVIDIA Tesla são suportados, não os drivers GRID.
- Ao instalar ou reinstalar o complemento, verifique se o endereço de download do driver está correto e acessível. O CCE não verifica a validade do endereço.
- O complemento gpu-beta somente permite que você baixe o driver e execute o script de instalação. O status do complemento indica apenas como o complemento está sendo executado, não se o driver foi instalado com êxito.
- O CCE não garante a compatibilidade entre a versão do driver da GPU e a versão da biblioteca CDUA da sua aplicação. Você precisa verificar a compatibilidade por si mesmo.
- Se uma imagem de sistema operacional personalizada tiver um driver de GPU instalado, o CCE não poderá garantir que o driver da GPU seja compatível com outros componentes da GPU, como os componentes de monitoramento usados no CCE.

Instalar o complemento

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster. Escolha **Add-ons** no painel de navegação, localize **CCE AI Suite (NVIDIA GPU)** à direita e clique em **Install**.

Passo 2 Na página **Install Add-on**, configure as especificações.

Tabela 13-41 Especificações de complemento

Parâmetro	Descrição
Add-on Specifications	Selecione **Default** ou **Custom**.
Containers	As cotas de CPU e memória do contêiner permitidas para as especificações adicionais selecionadas. Se você selecionar **Custom**, poderá ajustar as especificações do contêiner conforme necessário.

Passo 3 Configure os parâmetros do complemento.

- **NVIDIA Driver**: insira o link para baixar o driver NVIDIA. Todos os nós de GPU no cluster usarão esse driver.

AVISO

- Se o link de download for um endereço de rede pública, por exemplo, **https://us.download.nvidia.com/tesla/470.103.01/NVIDIA-Linux-x86_64-470.103.01.run**, vinculam um EIP a cada nó da GPU. Para obter detalhes sobre como obter o link do driver, consulte **Obter o link do driver da rede pública**.
- Se o link de download for um URL do OBS, não será necessário vincular um EIP aos nós da GPU. Para obter detalhes sobre como obter o link do driver, consulte **Obter o link do driver do OBS**.
- Certifique-se de que a versão do driver NVIDIA corresponda ao nó da GPU.
- Depois que a versão do driver for alterada, reinicie o nó para que a alteração tenha efeito.
- Use o driver de versão 470 ou posterior para a Huawei Cloud EulerOS 2.0 no qual o Linux Kernel 5.x é construído e o driver 515 ou posterior para o Ubuntu 22.04.

Passo 4 Clique em **Install**.

NOTA

A desinstalação do complemento limpará o driver da GPU nos nós. Como resultado, os pods de GPU recém-programados para os nós não podem ser executados corretamente, mas os pods de GPU em execução não são afetados.

----Fim

Verificar o complemento

Depois que o complemento for instalado, execute o comando **nvidia-smi** no nó da GPU e no contêiner que agenda os recursos da GPU para verificar a disponibilidade do dispositivo e do driver da GPU.

- Nó da GPU:

```
# If the add-on version is earlier than 2.0.0, run the following command:
cd /opt/cloud/cce/nvidia/bin && ./nvidia-smi

# If the add-on version is 2.0.0 or later and the driver installation path is changed, run the following command:
cd /usr/local/nvidia/bin && ./nvidia-smi
```

- Contêiner:

```
cd /usr/local/nvidia/bin && ./nvidia-smi
```

Se as informações da GPU forem retornadas, o dispositivo está disponível e o complemento foi instalado.

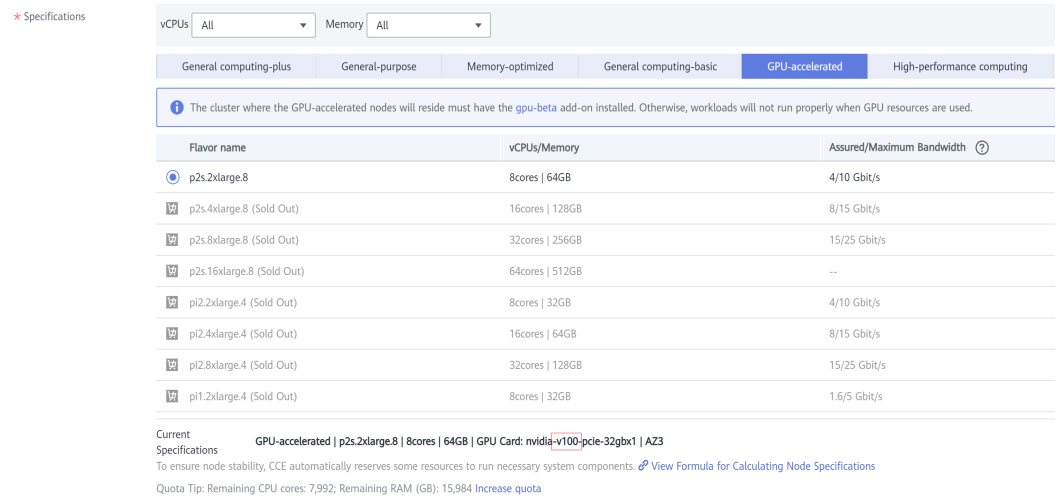

```
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 440.118.02   Driver Version: 440.118.02   CUDA Version: 10.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla V100-SXM2...  Off  | 00000000:21:01.0 Off |                    0 |
| N/A   31C    P0    23W / 300W |      0MiB / 16160MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

Obter o link do driver da rede pública

Passo 1 Efetue logon no console do CCE.

Passo 2 Clique em **Create Node** e selecione o nó da GPU a ser criado na área **Specifications**. O modelo da placa da GPU do nó é exibido na parte inferior da página.

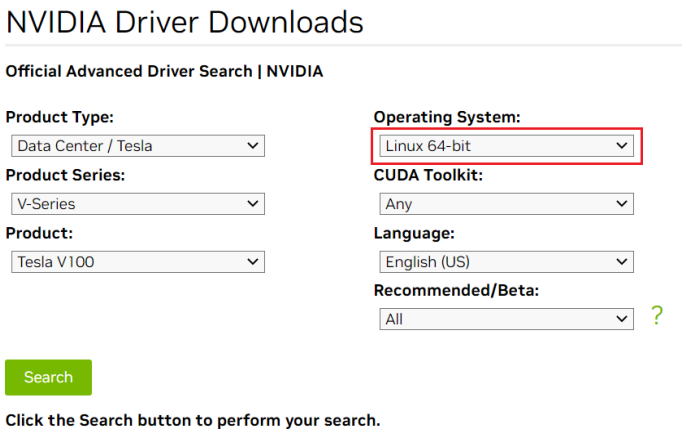
Figura 13-10 Visualizar o modelo da placa da GPU



Passo 3 Visite ***https://www.nvidia.com/Download/Find.aspx?lang=en***.

Passo 4 Selecione as informações do driver na página **NVIDIA Driver Downloads**, como mostrado na **Figura 13-11**. **Operating System** deve ser **Linux 64-bit**.

Figura 13-11 Definir parâmetros



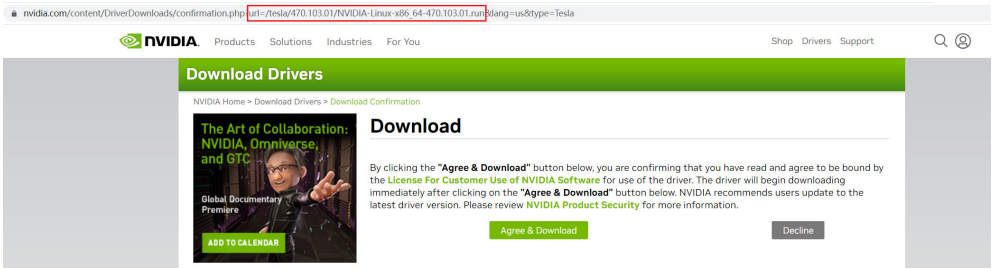
Passo 5 Depois de confirmar as informações do driver, clique em **SEARCH**. Uma página é exibida, mostrando as informações do driver, como mostrado na **Figura 13-12**. Clique em **Download**.

Figura 13-12 Informação do driver

Passo 6 Obtenha o link do driver de uma das seguintes maneiras:

- Método 1: como mostrado na **Figura 13-13**, encontre *url=/tesla/470.103.01/NVIDIA-Linux-x86_64-470.103.01.run* na caixa de endereço do navegador. Em seguida, complemente-o para obter o link do driver **https://us.download.nvidia.com/tesla/470.103.01/NVIDIA-Linux-x86_64-470.103.01.run**. Usando esse método, você deve vincular um EIP a cada nó da GPU.
- Método 2: como mostrado na **Figura 13-13**, clique em **AGREE & DOWNLOAD** para baixar o driver. Em seguida, faça o upload do driver para o OBS e registre o URL do OBS. Usando esse método, você não precisa vincular um EIP aos nós da GPU.

Figura 13-13 Obtenção do link



----Fim

Obter o link do driver do OBS

Passo 1 Carregue o driver para o OBS e defina o arquivo de driver para leitura pública. Para obter detalhes, consulte **Carregamento de um objeto**.

NOTA

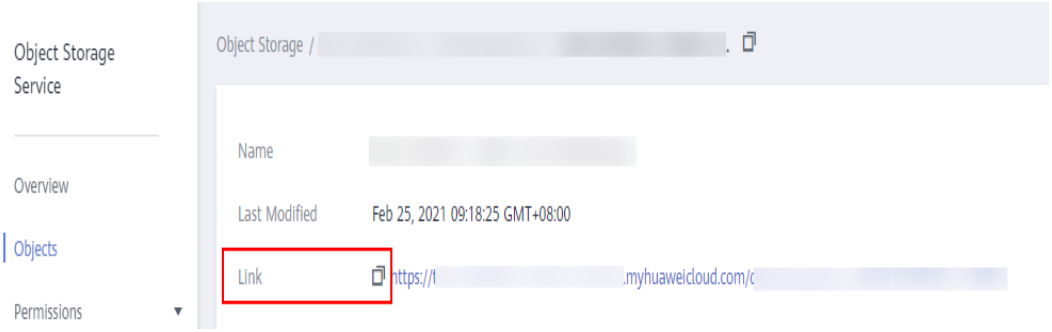
Quando o nó for reiniciado, o driver será baixado e instalado novamente. Certifique-se de que o link do bucket do OBS do driver seja válido.

Passo 2 Na lista de buckets, clique em um nome de intervalo e, em seguida, a página **Overview** do bucket é exibida.

Passo 3 No painel de navegação, escolha **Objects**.

Passo 4 Selecione o nome do objeto de destino e copie o link do driver na página de detalhes do objeto.

Figura 13-14 Copiar um link do OBS



----Fim

Componentes

Tabela 13-42 Componente da GPU

Componente	Descrição	Tipo de recurso
nvidia-driver-installer	Usado para instalar um driver NVIDIA em nós de GPU.	DaemonSet

Links úteis

- **Como corrigir falhas quando o driver NVIDIA é usado para iniciar contêineres em nós de GPU?**
- **O que fazer se ocorrerem exceções de nó de GPU?**
- **Agendamento da GPU**

13.12 Suíte de IA do CCE (Ascend NPU)

Introdução

O Ascend NPU é um complemento de gerenciamento de dispositivos que suporta NPUs da Huawei em contêineres.

Depois que esse complemento for instalado, você poderá criar nós acelerados pelo Ascend para processar de forma rápida e eficiente a inferência e o reconhecimento de imagem.

Restrições

- Para usar nós acelerados pelo Ascend em um cluster, o complemento Ascend NPU deve estar instalado.
- Depois que um nó acelerado por IA for migrado, o nó será redefinido. Reinstale manualmente o driver NPU.

Instalar o complemento

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster. Escolha **Add-ons** no painel de navegação, localize **CCE AI Suite (Ascend NPU)** à direita e clique em **Install**.

Passo 2 Defina os parâmetros de NPU. O complemento usa os seguintes parâmetros por padrão. As configurações de NPU padrão fornecidas pelo complemento podem satisfazer a maioria dos cenários e não requerem alterações.

```
{
    "check_frequency_failed_threshold": 100,
    "check_frequency_fall_times": 3,
    "check_frequency_gate": false,
    "check_frequency_recover_threshold": 100,
    "check_frequency_rise_times": 2,
    "container_path": "/usr/local/HiAI_unused",
    "host_path": "/usr/local/HiAI_unused"
}
```

Passo 3 Clique em **Install**.

----Fim

Componentes

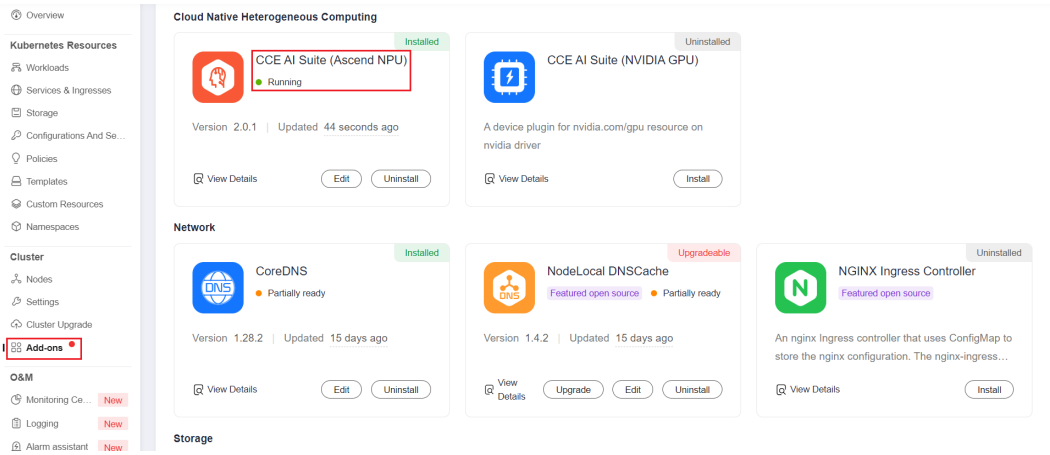
Tabela 13-43 Componentes de huawei-npu

Componente	Descrição	Tipo de recurso
npu-driver-installer	Usado para instalar um driver NPU em nós de NPU.	DaemonSet

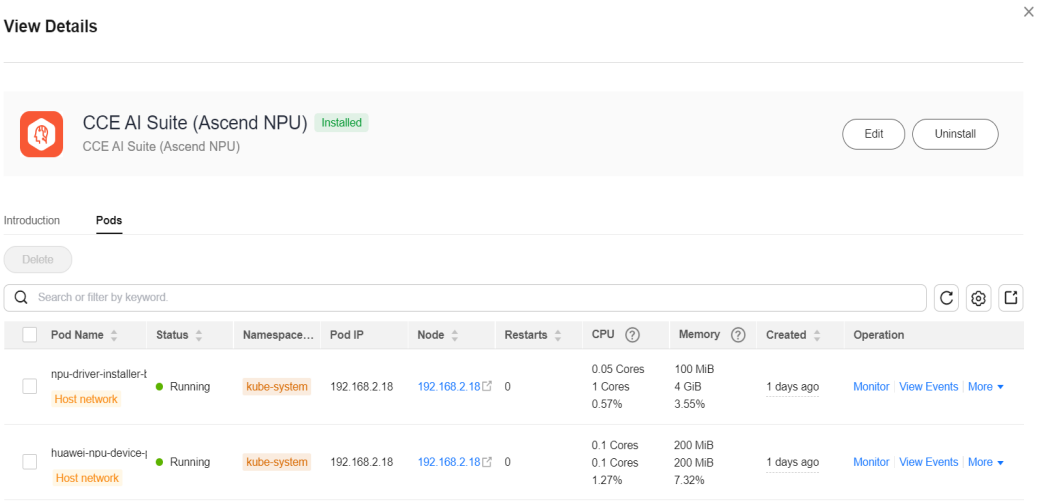
Como verificar se o driver NPU foi instalado em um nó

Depois de garantir que o driver foi instalado com êxito, reinicie o nó para que o driver entre em vigor. Caso contrário, o driver não poderá entrar em vigor e os recursos de NPU não estarão disponíveis. Para verificar se o driver está instalado, execute as seguintes operações:

Passo 1 Na página **Add-ons**, clique em **CCE AI Suite (Ascend NPU)**.



Passo 2 Verifique se o nó em que o npu-driver-installer está implementado está no estado **Running**.



> **NOTA**
>
> Se o nó for reiniciado antes da instalação do driver NPU, a instalação do driver poderá falhar e uma mensagem será exibida na página **Nodes** do cluster indicando que o driver Ascend não está pronto. Nesse caso, desinstale o driver NPU do nó e reinicie o **npu-driver-installer** para reinstalar o driver NPU. Depois de confirmar que o driver está instalado, reinicie o nó. Para obter detalhes sobre como desinstalar o driver, consulte **Desinstalar o driver NPU**.

----Fim

Desinstalar o driver NPU

Faça logon no nó, obtenha os registros de operação do driver no arquivo **/var/log/ascend_seclog/operation.log** e localize o pacote de execução do driver usado na última instalação. Se o arquivo lof não existir, o driver será instalado usando o pacote combinado de **npu_x86_latest.run** ou **npu_arm_latest.run**. Depois de encontrar o pacote de instalação do driver, execute o comando **bash {run *package name*} --uninstall** para desinstalar o driver e reiniciar o nó conforme solicitado.

Passo 1 Efetue logon no nó em que o driver NPU precisa ser desinstalado e localize o arquivo **/var/log/ascend_seclog/operation.log**.

Passo 2 Se o arquivo **/var/log/ascend_seclog/operation.log** puder ser encontrado, visualize o registro de instalação do driver para encontrar o registro de instalação do driver.

```
[root@y00379955-w-ai1s-e25 ~]# ll /var/log/ascend_seclog/operation.log
-rw-r----- 1 root root 285 Dec  1 20:00 /var/log/ascend_seclog/operation.log
[root@y00379955-w-ai1s-e25 ~]# cat /var/log/ascend_seclog/operation.log
Install SUGGESTION root 2022-12-01 19:53:47 127.0.0.1 Ascend310-hdk-npu-driver 6.0.rc1 linux-x86-64.run success install_type=full; cmdlist=--quiet --full.
```

Se o arquivo **/var/log/ascend_seclog/operation.log** não puder ser encontrado, o driver pode ser instalado usando o pacote combinado de **npu_x86_latest.run** ou **npu_arm_latest.run**. Você pode confirmar isso verificando se o diretório **/usr/local/HiAI/driver/** existe.

> **NOTA**
>
> O pacote combinado do driver NPU é armazenado no diretório **/root/d310_driver**, e outros pacotes de instalação do driver são armazenados no diretório **/root/npu-drivers**.

Passo 3 Depois de encontrar o pacote de instalação do driver, execute o comando **bash {run *package path*} --uninstall** para desinstalar o driver. O seguinte usa **Ascend310-hdk-npu-driver_6.0.rc1_linux-x86-64.run** como um exemplo:

```
bash /root/npu-drivers/Ascend310-hdk-npu-driver_6.0.rc1_linux-x86-64.run --uninstall
```

```
[root@y00379955-w-ai1s-e25 npu-drivers]# ./Ascend310-hdk-npu-driver_6.0.rc1_linux-x86-64.run --uninstall
Verifying archive integrity...  100%   SHA256 checksums are OK. All good.
Uncompressing ASCEND DRIVER RUN PACKAGE  100%
[Driver] [2022-12-01 19:59:53] [INFO]Start time: 2022-12-01 19:59:53
[Driver] [2022-12-01 19:59:53] [INFO]LogFile: /var/log/ascend_seclog/ascend_install.log
[Driver] [2022-12-01 19:59:53] [INFO]OperationLogFile: /var/log/ascend_seclog/operation.log
[Driver] [2022-12-01 19:59:53] [INFO]base version is 22.0.3.


[Driver] [2022-12-01 20:00:04] [INFO]Driver package uninstalled successfully! Reboot needed for uninstallation to take effect!
[Driver] [2022-12-01 20:00:04] [INFO]End time: 2022-12-01 20:00:04
```

Passo 4 Reinicie o nó conforme solicitado. (A instalação e a desinstalação do driver NPU atual terão efeito somente depois que o nó for reiniciado.)

----Fim

13.13 Volcano scheduler

Introdução

Volcano é uma plataforma de processamento em lote baseada em Kubernetes. Ele fornece uma série de recursos necessários para aprendizado de máquina, aprendizado profundo, bioinformática, genômica e outras aplicações de Big data, como um poderoso complemento aos recursos do Kubernetes.

O Volcano fornece recursos de computação gerais, como agendamento de tarefas de alto desempenho, gerenciamento de chips heterogêneos e gerenciamento de execução de tarefas. Ele acessa as estruturas de computação para vários setores, como IA, Big data, gene e renderização, e agenda até 1000 pods por segundo para usuários finais, melhorando significativamente a eficiência de agendamento e a utilização de recursos.

O Volcano fornece agendamento de tarefas, gerenciamento de tarefas e gerenciamento de filas para aplicações de computação. Suas principais características são as seguintes:

- Diversas estruturas de computação, como TensorFlow e Spark, podem ser executadas no Kubernetes em contêineres. APIs comuns para tarefas de computação em lotes por meio de CRD, vários plug-ins e gerenciamento avançado do ciclo de vida da tarefa são fornecidos.
- Os recursos avançados de agendamento são fornecidos para computação em lote e cenários de computação de alto desempenho, incluindo agendamento de grupo, agendamento de prioridade preemptiva, embalagem, reserva de recursos e topologia de tarefas.
- As filas podem ser gerenciadas com eficiência para agendar tarefas. Recursos complexos de agendamento de tarefas, como prioridade de fila e filas de vários níveis, são suportados.

Volcano foi disponibilizado no GitHub em **https://github.com/volcano-sh/volcano**.

Instale e configure o complemento de Volcano em clusters do CCE. Para mais detalhes, consulte **Agendamento de Volcano**.

NOTA

Ao usar o Volcano como um agendador, use-o para agendar todas as cargas de trabalho no cluster. Isso evita conflitos de agendamento de recursos causados pelo trabalho simultâneo de vários agendadores.

Instalar o complemento

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster. Escolha **Add-ons** no painel de navegação, localize **Volcano Scheduler** à direita e clique em **Install**.

Passo 2 Na página **Install Add-on**, configure as especificações.

Tabela 13-44 Configuração do Volcano

Parâmetro	Descrição
Add-on Specifications	Selecione **Single**, **Custom** ou **HA** para **Add-on Specifications**.
Instances	Número de pods que serão criados para corresponder às especificações do complemento selecionado. Se você selecionar **Custom**, poderá ajustar o número de pods conforme necessário.
Containers	As cotas de CPU e memória do contêiner permitidas para as especificações adicionais selecionadas. Se você selecionar **Custom**, os valores recomendados para **volcano-controller** e **volcano-scheduler** serão os seguintes: ● Se o número de nós for menor que 100, mantenha a configuração padrão. As vCPUs solicitadas são 500m e o limite é 2000m. A memória solicitada é de 500 MiB e o limite é de 2000 MiB. ● Se o número de nós for maior que 100, aumente as vCPUs solicitadas em 500m e a memória solicitada em 1000 MiB cada vez que 100 nós (10.000 pods) forem adicionados. Aumente o limite de vCPU em 1500m e o limite de memória em 1000 MiB. **NOTA** Fórmula recomendada para calcular o valor solicitado: – VCPUs solicitadas: calcule o número de nós de destino multiplicado pelo número de pods de destino, execute a pesquisa de interpolação com base no número de nós no cluster multiplicado pelo número de pods de destino em **Tabela 13-45** e arredonde o valor da solicitação e o valor limite mais próximo das especificações. Por exemplo, para 2000 nós e 20.000 pods, número de nós de destino x número de pods de destino = 40 milhões, o que é próximo da especificação de 700/70.000 (número de nós do cluster x número de pods = 49 milhões). De acordo com a tabela a seguir, defina as vCPUs solicitadas como 4000m e o valor limite como 5500m. – Memória solicitada: recomenda-se que 2,4 GiB de memória seja alocada a cada 1000 nós e 1 GiB de memória seja alocada a cada 10.000 pods. A memória solicitada é a soma desses dois valores. (O valor obtido pode ser diferente do valor recomendado em **Tabela 13-45**. Você pode usar qualquer um deles.) Memória solicitada = número de nós de destino/1000 x 2,4 GiB + número de pods de destino/10.000 x 1 GiB Por exemplo, para 2000 nós e 20 000 pods, a memória solicitada é de 6,8 GiB (2000/1000 x 2,4 GiB + 20.000/10.000 x 1 GiB).

Tabela 13-45 Valores recomendados para volcano-controller e volcano-scheduler

Nós/pods em um cluster	vCPUs solicitadas (m)	Limite da vCPU (m)	Memória solicitada (MiB)	Limite de memória (MiB)
50/5000	500	2000	500	2000

Nós/pods em um cluster	vCPUs solicitadas (m)	Limite da vCPU (m)	Memória solicitada (MiB)	Limite de memória (MiB)
100/10.000	1000	2500	1500	2500
200/20.000	1500	3000	2500	3500
300/30.000	2000	3500	3500	4500
400/40.000	2500	4000	4500	5500
500/50.000	3000	4500	5500	6500
600/60.000	3500	5000	6500	7500
700/70.000	4000	5500	7500	8500

Passo 3 Configure parâmetros avançados de complemento.

Configure os parâmetros do Volcano scheduler padrão. Para mais detalhes, consulte **Tabela 13-47**.

```
colocation_enable: ''
default_scheduler_conf:
  actions: 'allocate, backfill'
  tiers:
    - plugins:
        - name: 'priority'
        - name: 'gang'
        - name: 'conformance'
        - name: 'lifecycle'
          arguments:
            lifecycle.MaxGrade: 10
            lifecycle.MaxScore: 200.0
            lifecycle.SaturatedTresh: 1.0
            lifecycle.WindowSize: 10
    - plugins:
        - name: 'drf'
        - name: 'predicates'
        - name: 'nodeorder'
    - plugins:
        - name: 'cce-gpu-topology-predicate'
        - name: 'cce-gpu-topology-priority'
        - name: 'cce-gpu'
    - plugins:
        - name: 'nodelocalvolume'
        - name: 'nodeemptydirvolume'
        - name: 'nodeCSIscheduling'
        - name: 'networkresource'
tolerations:
  - effect: NoExecute
    key: node.kubernetes.io/not-ready
    operator: Exists
    tolerationSeconds: 60
  - effect: NoExecute
    key: node.kubernetes.io/unreachable
    operator: Exists
    tolerationSeconds: 60
```


Tabela 13-46 Parâmetros avançados de configuração de Volcano

Plug-in	Função	Descrição	Demonstração
colocation_enable	Se habilitar a implementação híbrida.	Valor: ● **true**: híbrida ativada ● **false**: híbrida desativada	Nenhuma
default_scheduler_conf	Usado para agendar pods. Consiste em uma série de ações e plug-ins e apresenta alta escalabilidade. Você pode especificar e implementar ações e plug-ins com base em seus requisitos.	Consiste em actions e tiers. ● **actions**: define os tipos e a sequência de ações a serem executadas pelo agendador. ● **tiers**: configura a lista de plugins.	Nenhuma
actions	Ações a serem executadas em cada fase de programação. A sequência de ação configurada é a sequência de execução do agendador. Para obter detalhes, consulte **Ações**. O agendador percorre todas as tarefas a serem programadas e executa ações como enqueue, allocate, preempt, e backfill na sequência configurada para encontrar o nó mais apropriado para cada tarefa.	As seguintes opções são compatíveis: ● **enqueue** usa uma série de algoritmos de filtragem para filtrar as tarefas a serem agendadas e as envia para a fila para aguardar o agendamento. Após essa ação, o status da tarefa muda de **pending** para **inqueue**. ● **allocate**: seleciona o nó mais adequado com base em uma série de algoritmos de pré-seleção e seleção. ● **preempt**: executa o agendamento de preempção para tarefas com prioridades mais altas na mesma fila com base em regras de prioridade. ● **backfill**: agenda as tarefas pendentes o máximo possível para maximizar a utilização dos recursos do nó.	`actions: 'allocate, backfill'` **NOTA** Ao configurar **actions**, use **preempt** ou **enqueue**.

Plug-in	Função	Descrição	Demonstração
plugins	Detalhes de implementação de algoritmos em ações com base em diferentes cenários. Para detalhes, veja **Plug-ins**.	Para mais detalhes, consulte **Tabela 13-47**.	Nenhuma

tolerations	Tolerância do complemento a manchas de nós.	Por padrão, o complemento pode ser executado em nós com o **node.kubernetes.io/not-ready** ou **node.kubernetes.io/unreachable** e o valor do efeito de mancha é **NoExecute**, mas será despejado em 60 segundos.	`tolerations:`
			`  - effect: NoExecute`
			`  key: node.kubernetes.io/not-ready`
			`  operator: Exists`
			`  tolerationSeconds: 60`
			`  - effect: NoExecute`
			`  key: node.kubernetes.io/unreachable`
			`  operator: Exists`
			`  tolerationSeconds: 60`

Tabela 13-47 Plug-ins suportados

Plug-in	Função	Descrição	Demonstração

binpack	Programe pods para nós com alto uso de recursos (não alocando pods para nós carregados de luz) para reduzir fragmentos de recursos.	**arguments**:
		● **binpack.weight**: peso do plug-in binpack.
		● **binpack.cpu**: proporção de CPUs para todos os recursos. O valor padrão do parâmetro é **1**.
		● **binpack.memory**: proporção de recursos de memória para todos os recursos. O valor padrão do parâmetro é **1**.
		● **binpack.resources**: outros tipos de recursos personalizados solicitados pelo pod, por exemplo, **nvidia.com/gpu**. Vários tipos podem ser configurados e separados por vírgulas (,).
		● **binpack.resources.**<*your_resource*>: peso do seu recurso personalizado em todos os recursos. Vários tipos de recursos podem ser adicionados. <*your_resource*> indica o tipo de recurso definido em **binpack.resources**, por exemplo, **binpack.resources.nvidia.com/gpu**.
			`- plugins:`
			`  - name: binpack`
			`  arguments:`
			`  binpack.weight: 10`
			`  binpack.cpu: 1`
			`  binpack.memory: 1`
			
			`binpack.resources: nvidia.com/gpu, example.com/foo`
			
			`binpack.resources.nvidia.com/gpu: 2`
			
			`binpack.resources.example.com/foo: 3`

conformance	Evite que pods principais, como os pods no namespace **kube-system** sejam antecipados.	Nenhuma	`- plugins:`
			`  - name: 'priority'`
			`  - name: 'gang'`
			`  enablePreemptable: false`
			`  - name: 'conformance'`

Plug-in	Função	Descrição	Demonstração

lifecycle	Ao coletar estatísticas sobre regras de escala de serviço, os pods com ciclos de vida semelhantes são preferencialmente programados para o mesmo nó. Com a capacidade de dimensionamento horizontal do Autoscaler, os recursos podem ser rapidamente dimensionados e liberados, reduzindo custos e melhorando a utilização de recursos.
	1. Coleta estatísticas sobre o ciclo de vida dos pods na carga de serviço e programa pods com ciclos de vida semelhantes para o mesmo nó.
	2. Para um cluster configurado com uma política de dimensionamento automático, ajuste a anotação de reduzir do nó para reduzir preferencialmente no nó com baixo uso.
		arguments:
		● **lifecycle.WindowSize**: o valor é um inteiro maior ou igual a 1 e o padrão é **10**.
		Registre o número de vezes que o número de réplicas é alterado. Se a carga mudar regularmente e periodicamente, diminua o valor. Se a carga for alterada de forma irregular e o número de réplicas for alterado com frequência, aumente o valor. Se o valor for muito grande, o período de aprendizado é prolongado e muitos eventos são registrados.
		● **lifecycle.MaxGrade**: o valor é um inteiro maior ou igual a 3 e o padrão é **3**.
		Ele indica níveis de réplicas. Por exemplo, se o valor for definido como **3**, as réplicas serão classificadas em três níveis. Se a carga mudar regularmente e periodicamente, diminua o valor. Se a carga mudar de forma irregular, aumente o valor. Definir um valor excessivamente pequeno pode resultar em previsões de ciclo de vida imprecisas.
		● **lifecycle.MaxScore**: número de ponto flutuante float64. O valor deve ser superior ou igual a 50.0. O valor padrão é **200.0**.
		Pontuação máxima (equivalente ao peso) do plug-in do ciclo de vida.
			`- plugins:`
			`  - name: priority`
			`  - name: gang`
			`  enablePreemptable: false`
			`  - name: conformance`
			`  - name: lifecycle`
			`  arguments:`
			`lifecycle.MaxGrade: 10`
			`lifecycle.MaxScore: 200.0`
			`lifecycle.SaturatedTresh: 1.0`
			`lifecycle.WindowSize: 10`
			NOTA
			● Para nós que não desejam ser reduzidos, marque-os manualmente como nós de período longo e adicione a anotação **volcano.sh/long-lifecycle-node: true** para eles. Para um nó não marcado, o plug-in de ciclo de vida marca automaticamente o nó com base no ciclo de vida da carga no nó.
			● O valor padrão de **MaxScore** é **200.0**, que é o dobro do peso de outros plug-ins. Quando o plug-in do ciclo de vida não tem efeito óbvio ou conflitos com outros plug-ins, desabilite outros plug-ins ou aumente o valor do **MaxScore**.
			● Depois que o agendador é reiniciado, o plug-in do ciclo de vida precisa gravar novamente a alteração de carga. O efeito de agendamento ideal pode ser alcançado somente após vários períodos de estatísticas serem coletados.

Plug-in	Função	Descrição	Demonstração
		● **lifecycle.SaturatedTresh**: número de ponto flutuante float64. Se o valor for menor que 0,5, use **0.5**. Se o valor for maior que 1, use **1**. O valor padrão é **0.8**. Limite para determinar se o uso do nó é muito alto. Se o uso do nó exceder o limite, o agendador preferencialmente agendará tarefas para outros nós.	

Plug-in	Função	Descrição	Demonstração

Gang	Considere um grupo de pods como um todo para alocação de recursos. Este plug-in verifica se o número de pods agendados em uma tarefa atende aos requisitos mínimos para a execução da tarefa. Se sim, todos os pods na tarefa serão agendados. Se não, os pods não serão agendados.
	NOTA
	Se uma política de agendamento de grupo for usada, se os recursos restantes no cluster forem maiores ou iguais à metade do número mínimo de recursos para a execução de uma tarefa, mas menores do que o mínimo de recursos para a execução da tarefa, as expansões do Autoscaler não serão acionadas.
		● **enablePreemptable**:
		– **true**: preempção ativada
		– **false**: preempção desativada
		● **enableJobStarving**:
		– **true**: os recursos são antecipados com base na configuração **minAvailable** das tarefas.
		– **false**: os recursos são antecipados com base em réplicas de tarefa.
		NOTA
		– O valor padrão de **minAvailable** para cargas de trabalho nativas do Kubernetes (como Implementações) é **1**. É uma boa prática definir **enableJobStarving** como **false**.
		– Em cenários de IA e Big data, você pode especificar o valor de **minAvailable** ao criar um vcjob. É uma boa prática definir **enableJobStarving** como **true**.
		– Nas versões do Volcano anteriores à v1.11.5, **enableJobStarving** é definido como **true** por padrão. Em versões do Volcano posteriores à v1.11.5, **enableJobStarving** é definido como **false** por padrão.
			`- plugins:`
			`  - name: priority`
			`  - name: gang`
			`    enablePreemptable: false`
			`    enableJobStarving: false`
			`  - name: conformance`

priority	Agendamento baseado em prioridades de carregamento personalizadas.	Nenhuma	`- plugins:`
			`  - name: priority`
			`  - name: gang`
			`    enablePreemptable: false`
			`  - name: conformance`

Plug-in	Função	Descrição	Demonstração

overcommit	Os recursos em um cluster são programados após serem acumulados em um determinado múltiplo para melhorar a eficiência do enfileiramento da carga de trabalho. Se todas as cargas de trabalho forem Implementações, remova este plug-in ou defina o fator de aumento para **2.0**.
	NOTA
	Este plug-in é suportado no Volcano 1.6.5 e versões posteriores.
		arguments:
		● **overcommit-factor**: fator de inflação, cujo valor padrão é **1.2**.
			`- plugins:`
			`  - name: overcommit`
			`  arguments:`
			`  overcommit-factor: 2.0`

drf	O algoritmo de agendamento Dominant Resource Fairness (DRF), que agenda tarefas com base em seu compartilhamento de recursos dominante. Tarefas com uma parcela de recursos menor serão agendadas com uma prioridade mais alta.	Nenhuma	`- plugins:`
			`  - name: 'drf'`
			`  - name: 'predicates'`
			`  - name: 'nodeorder'`

Plug-in	Função	Descrição	Demonstração

predicates	Determine se uma tarefa está vinculada a um nó usando uma série de algoritmos de avaliação, como afinidade nó/pod, tolerância a manchas, repetição de nó, limites de volume e correspondência de zona de volume.	Nenhuma	`- plugins:`
			`  - name: 'drf'`
			`  - name: 'predicates'`
			`  - name: 'nodeorder'`

Plug-in	Função	Descrição	Demonstração

nodeorder	Um algoritmo comum para selecionar nós. Os nós são pontuados na alocação de recursos simulada para encontrar o nó mais adequado para a tarefa atual.	Parâmetros de pontuação:
		● **nodeaffinity.weight**: os pods são agendados com base na afinidade do nó. O padrão deste parâmetro é **2**.
		● **podaffinity.weight**: os pods são agendados com base na afinidade do pod. O padrão deste parâmetro é **2**.
		● **leastrequested.weight**: os pods são programados para o nó com os recursos menos solicitados. O padrão deste parâmetro é **1**.
		● **balancedresource.weight**: os pods são agendados para o nó com alocação de recursos balanceada. O padrão deste parâmetro é **1**.
		● **mostrequested.weight**: os pods são agendados para o nó com os recursos mais solicitados. O padrão deste parâmetro é **0**.
		● **tainttoleration.weight**: os pods são programados para o nó com uma alta tolerância a manchas. O padrão deste parâmetro é **3**.
		● **imagelocality.weight**: os pods são agendados para o nó onde existem as imagens necessárias. O padrão deste parâmetro é **1**.
		● **selectorspread.weight**: os pods são agendados uniformemente para diferentes nós. O padrão deste parâmetro é **0**.
		● **podtopologyspread.weight**: os pods são
			`- plugins:`
			`  - name: nodeorder`
			`  arguments:`
			
			`leastrequested.weight: 1`
			
			`mostrequested.weight: 0`
			
			`nodeaffinity.weight: 2`
			
			`podaffinity.weight: 2`
			
			`balancedresource.weight: 1`
			
			`tainttoleration.weight: 3`
			
			`imagelocality.weight: 1`
			
			`podtopologyspread.weight: 2`

Plug-in	Função	Descrição	Demonstração
		programados com base na topologia do pod. O padrão deste parâmetro é **2**.	

cce-gpu-topology-predicate	Algoritmo de pré-seleção de agendamento de topologia de GPU	Nenhuma	`- plugins:`
			`  - **name: 'cce-gpu-topology-predicate'**`
			`  - name: 'cce-gpu-topology-priority'`
			`  - name: 'cce-gpu'`

cce-gpu-topology-priority	Algoritmo de prioridade de agendamento de topologia de GPU	Nenhuma	`- plugins:`
			`  - name: 'cce-gpu-topology-predicate'`
			`  - **name: 'cce-gpu-topology-priority'**`
			`  - name: 'cce-gpu'`

Plug-in	Função	Descrição	Demonstração

cce-gpu	Alocação de recursos de GPU que suporta configurações de GPU decimais trabalhando com o complemento gpu.
	NOTA
	● O plug-in da versão 1.10.5 ou posterior não suporta este complemento. Use xGPU em vez disso.
	● O pré-requisito para configurar GPUs decimais é que os nós da GPU no cluster estejam no modo compartilhado. Para obter detalhes sobre como verificar se o compartilhamento de GPU está desabilitado no cluster, consulte o parâmetro **enable-gpu-share** em **Gerenciamento de configuração de cluster**.
		Nenhuma	`- plugins:`
			`  - name: 'cce-gpu-topology-predicate'`
			`  - name: 'cce-gpu-topology-priority'`
			`  - name: 'cce-gpu'`

numa-aware	Agendamento de afinidade NUMA. Para obter detalhes, consulte **Agendamento de afinidade NUMA**.	**arguments**:
		● **weight**: peso do plug-in numa-aware
			`- plugins:`
			`  - name: 'nodelocalvolume'`
			`  - name: 'nodeemptydirvolume'`
			`  - name: 'nodeCSIscheduling'`
			`  - name: 'networkresource'`
			`  arguments:`
			`  NetworkType: vpc-router`
			`  - name: numa-aware`
			`  arguments:`
			`  weight: 10`

Plug-in	Função	Descrição	Demonstração

networkresource	O nó de requisito da ENI pode ser pré-selecionado e filtrado. Os parâmetros são transferidos pelo CCE e não precisam ser configurados manualmente.	**arguments**:
		● **NetworkType**: tipo de rede (**eni** ou **vpc-router**)
			`- plugins:`
			`  - name: 'nodelocalvolume'`
			`  - name: 'nodeemptydirvolume'`
			`  - name: 'nodeCSIscheduling'`
			`  - **name: 'networkresource'**`
			`  arguments:`
			`  NetworkType: vpc-router`

nodelocalvolume	Filtrar nós que não atendem aos requisitos de volume local.	Nenhuma	`- plugins:`
			`  - **name: 'nodelocalvolume'**`
			`  - name: 'nodeemptydirvolume'`
			`  - name: 'nodeCSIscheduling'`
			`  - name: 'networkresource'`

nodeemptydirvolume	Filtrar nós que não atendam aos requisitos de vazioDir.	Nenhuma	`- plugins:`
			`  - name: 'nodelocalvolume'`
			`  - **name: 'nodeemptydirvolume'**`
			`  - name: 'nodeCSIscheduling'`
			`  - name: 'networkresource'`

nodeCSIscheduling	Filtrar nós com mau funcionamento do Everest.	Nenhuma	`- plugins:`
			`  - name: 'nodelocalvolume'`
			`  - name: 'nodeemptydirvolume'`
			`  - **name: 'nodeCSIscheduling'**`
			`  - name: 'networkresource'`

Passo 4 Configure as políticas de agendamento para o complemento.

NOTA

- As políticas de agendamento não entram em vigor em instâncias complementares do tipo DaemonSet.
- Ao configurar a implementação de várias AZs ou a afinidade de nó, verifique se há nós que atendem à política de agendamento e se os recursos são suficientes no cluster. Caso contrário, o complemento não pode ser executado.

Tabela 13-48 Configurações para programação de complementos

Parâmetro	Descrição
Implementação de várias AZs	● **Preferred**: os pods de implantação do complemento serão agendados preferencialmente para nós em diferentes AZs. Se todos os nós no cluster forem implementados na mesma AZ, os pods serão agendados para essa AZ. ● **Forcible**: os pods de implementação do complemento serão forçosamente agendados para nós em diferentes AZs. Se houver menos AZs do que pods, os pods extras não funcionarão.
Node Affinity	● **Incompatibility**: a afinidade de nó está desabilitada para o complemento. ● **Node Affinity**: especifique os nós em que o complemento é implementado. Se você não especificar os nós, o complemento será agendado aleatoriamente com base na política de agendamento de cluster padrão. ● **Specified Node Pool Scheduling**: especifique o pool de nós em que o complemento é implementado. Se você não especificar o pool de nós, o complemento será agendado aleatoriamente com base na política de agendamento de cluster padrão. ● **Custom Policies**: insira os rótulos dos nós em que o complemento será implementado para políticas de agendamento mais flexíveis. Se você não especificar rótulos de nó, o complemento será agendado aleatoriamente com base na política de agendamento de cluster padrão. Se várias políticas de afinidade personalizadas estiverem configuradas, certifique-se de que existem nós que atendam a todas as políticas de afinidade no cluster. Caso contrário, o complemento não pode ser executado.
Taints and Tolerations	O uso de manchas e tolerâncias permite (não forçosamente) que a Implementação do complemento seja agendada para um nó com as manchas correspondentes e controla as políticas de despejo de Implementação depois que o nó onde a Implementação está localizada é contaminado. O complemento adiciona a política de tolerância padrão para as manchas **node.kubernetes.io/not-ready** e **node.kubernetes.io/unreachable**, respectivamente. A janela de tempo de tolerância é 60s. Para mais detalhes, consulte **Manchas e tolerâncias**.

Passo 5 Clique em **Install**.

----Fim

Componentes

Tabela 13-49 Componentes do Volcano

Componente	Descrição	Tipo de recurso
volcano-scheduler	Agendar pods.	Implementação
volcano-controller	Sincronizar CRDs.	Implementação
volcano-admission	Servidor Webhook, que verifica e modifica recursos como pods e tarefas	Implementação
volcano-agent	Agente híbrido nativo da nuvem, que é usado para garantia de QoS de nó, intermitência de CPU e superassinatura de recursos dinâmicos	DaemonSet
resource-exporter	Relatar as informações de topologia NUMA dos nós.	DaemonSet

Modificar as configurações do volcano-scheduler usando o console

volcano-scheduler é o componente responsável pelo agendamento do pod. Consiste em uma série de ações e plug-ins. As ações devem ser executadas em todas as etapas. Plug-ins fornecem os detalhes do algoritmo de ação em diferentes cenários. volcano-scheduler é altamente escalável. Você pode especificar e implementar ações e plug-ins com base em seus requisitos.

Volcano permite que você configure o agendador durante a instalação, atualização e edição. A configuração será sincronizada com o volcano-scheduler-configmap.

Esta seção descreve como configurar o volcano-scheduler.

NOTA

Somente o Volcano da v1.7.1 e posteriores suportam essa função. Na nova página do plug-in, opções como **plugins.eas_service** e **resource_exporter_enable** são substituídas por **default_scheduler_conf**.

Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster. Escolha **Add-ons** no painel de navegação. À direita da página, localize **Volcano Scheduler** e clique em **Install** ou **Upgrade**. Na área **Parameters**, configure os parâmetros do Volcano.

- Usar **resource_exporter**:

```
{
    "ca_cert": "",
    "default_scheduler_conf": {
        "actions": "allocate, backfill",
        "tiers": [
            {
                "plugins": [
                    {
                        "name": "priority"
                    },
```



```
                    {
                        "name": "gang"
                    },
                    {
                        "name": "conformance"
                    }
                ]
            },
            {
                "plugins": [
                    {
                        "name": "drf"
                    },
                    {
                        "name": "predicates"
                    },
                    {
                        "name": "nodeorder"
                    }
                ]
            },
            {
                "plugins": [
                    {
                        "name": "cce-gpu-topology-predicate"
                    },
                    {
                        "name": "cce-gpu-topology-priority"
                    },
                    {
                        "name": "cce-gpu"
                    },
                    {
                        "name": "numa-aware" # add this also enable
resource_exporter
                    }
                ]
            },
            {
                "plugins": [
                    {
                        "name": "nodelocalvolume"
                    },
                    {
                        "name": "nodeemptydirvolume"
                    },
                    {
                        "name": "nodeCSIscheduling"
                    },
                    {
                        "name": "networkresource"
                    }
                ]
            }
        ]
    },
    "server_cert": "",
    "server_key": ""
}
```

depois que esta função estiver ativada, você pode usar as funções de numa-aware e resource_exporter.

- Usar **eas_service**:

```
{
    "ca_cert": "",
    "default_scheduler_conf": {
        "actions": "allocate, backfill",
        "tiers": [
            {
```



```
                "plugins": [
                    {
                        "name": "priority"
                    },
                    {
                        "name": "gang"
                    },
                    {
                        "name": "conformance"
                    }
                ]
            },
            {
                "plugins": [
                    {
                        "name": "drf"
                    },
                    {
                        "name": "predicates"
                    },
                    {
                        "name": "nodeorder"
                    }
                ]
            },
            {
                "plugins": [
                    {
                        "name": "cce-gpu-topology-predicate"
                    },
                    {
                        "name": "cce-gpu-topology-priority"
                    },
                    {
                        "name": "cce-gpu"
                    },
                    {
                        "name": "eas",
                        "custom": {
                            "availability_zone_id": "",
                            "driver_id": "",
                            "endpoint": "",
                            "flavor_id": "",
                            "network_type": "",
                            "network_virtual_subnet_id": "",
                            "pool_id": "",
                            "project_id": "",
                            "secret_name": "eas-service-secret"
                        }
                    }
                ]
            },
            {
                "plugins": [
                    {
                        "name": "nodelocalvolume"
                    },
                    {
                        "name": "nodeemptydirvolume"
                    },
                    {
                        "name": "nodeCSIscheduling"
                    },
                    {
                        "name": "networkresource"
                    }
                ]
            }
        ]
```



```
    },
    "server_cert": "",
    "server_key": ""
}
```

- Usar **ief**:

```
{
    "ca_cert": "",
    "default_scheduler_conf": {
        "actions": "allocate, backfill",
        "tiers": [
            {
                "plugins": [
                    {
                        "name": "priority"
                    },
                    {
                        "name": "gang"
                    },
                    {
                        "name": "conformance"
                    }
                ]
            },
            {
                "plugins": [
                    {
                        "name": "drf"
                    },
                    {
                        "name": "predicates"
                    },
                    {
                        "name": "nodeorder"
                    }
                ]
            },
            {
                "plugins": [
                    {
                        "name": "cce-gpu-topology-predicate"
                    },
                    {
                        "name": "cce-gpu-topology-priority"
                    },
                    {
                        "name": "cce-gpu"
                    },
                    {
                        "name": "ief",
                        "enableBestNode": true
                    }
                ]
            },
            {
                "plugins": [
                    {
                        "name": "nodelocalvolume"
                    },
                    {
                        "name": "nodeemptydirvolume"
                    },
                    {
                        "name": "nodeCSIscheduling"
                    },
                    {
                        "name": "networkresource"
                    }
                ]
            }
```



```
            ]
        },
        "server_cert": "",
        "server_key": ""
    }
```

Reter as configurações originais de volcano-scheduler-configmap

Se você quiser usar a configuração original depois que o plug-in for atualizado, execute as seguintes etapas:

Passo 1 Verifique e faça backup da configuração original do volcano-scheduler-configmap.

Exemplo:

```
# kubectl edit cm volcano-scheduler-configmap -n kube-system
apiVersion: v1
data:
  default-scheduler.conf: |-
    actions: "enqueue, allocate, backfill"
    tiers:
    - plugins:
      - name: priority
      - name: gang
      - name: conformance
    - plugins:
      - name: drf
      - name: predicates
      - name: nodeorder
      - name: binpack
        arguments:
          binpack.cpu: 100
          binpack.weight: 10
          binpack.resources: nvidia.com/gpu
          binpack.resources.nvidia.com/gpu: 10000
    - plugins:
      - name: cce-gpu-topology-predicate
      - name: cce-gpu-topology-priority
      - name: cce-gpu
    - plugins:
      - name: nodelocalvolume
      - name: nodeemptydirvolume
      - name: nodeCSIscheduling
      - name: networkresource
```

Passo 2 Insira o conteúdo personalizado na área **Parameters** no console.

```
{
    "ca_cert": "",
    "default_scheduler_conf": {
        "actions": "enqueue, allocate, backfill",
        "tiers": [
            {
                "plugins": [
                    {
                        "name": "priority"
                    },
                    {
                        "name": "gang"
                    },
                    {
                        "name": "conformance"
                    }
                ]
            },
            {
                "plugins": [
                    {
                        "name": "drf"
```



```
                    },
                    {
                        "name": "predicates"
                    },
                    {
                        "name": "nodeorder"
                    },
                    {
                        "name": "binpack",
                        "arguments": {
                            "binpack.cpu": 100,
                            "binpack.weight": 10,
                            "binpack.resources": "nvidia.com/gpu",
                            "binpack.resources.nvidia.com/gpu": 10000
                        }
                    }
                ]
            },
            {
                "plugins": [
                    {
                        "name": "cce-gpu-topology-predicate"
                    },
                    {
                        "name": "cce-gpu-topology-priority"
                    },
                    {
                        "name": "cce-gpu"
                    }
                ]
            },
            {
                "plugins": [
                    {
                        "name": "nodelocalvolume"
                    },
                    {
                        "name": "nodeemptydirvolume"
                    },
                    {
                        "name": "nodeCSIscheduling"
                    },
                    {
                        "name": "networkresource"
                    }
                ]
            }
        ]
    },
    "server_cert": "",
    "server_key": ""
}
```

NOTA

Quando esta função for usada, o conteúdo original em volcano-scheduler-configmap será sobrescrito. Portanto, você deve verificar se o volcano-scheduler-configmap foi modificado durante a atualização. Se sim, sincronize a modificação com a página de atualização.

----Fim

Desinstalar o complemento de Volcano

Depois que o complemento for desinstalado, todos os recursos personalizados do Volcano (**Tabela 13-50**) serão excluídos, incluindo os recursos criados. Reinstalar o complemento não herdará nem restaurará as tarefas antes da desinstalação. É uma boa prática desinstalar o

complemento do Volcano somente quando nenhum recurso personalizado do Volcano estiver sendo usado no cluster.

Tabela 13-50 Recursos personalizados do Volcano

Item	Grupo de API	Versão da API	Nível de recurso
Comando	bus.volcano.sh	v1alpha1	Namespaced
Job	batch.volcano.sh	v1alpha1	Namespaced
Numatopology	nodeinfo.volcano.sh	v1alpha1	Cluster
PodGroup	scheduling.volcano.sh	v1beta1	Namespaced
Queue	scheduling.volcano.sh	v1beta1	Cluster

Operações relacionadas

- **Excesso de assinaturas de recursos dinâmicos**
- **Agendamento de afinidade NUMA**

13.14 Secrets Manager do CCE para DEW

Introdução

O complemento dew-provider é usado para interconectar-se com o **Data Encryption Workshop (DEW)**, que permite montar segredos armazenados fora de um cluster (DEW para armazenar informações sensíveis) para pods. Desta forma, as informações confidenciais podem ser dissociadas do ambiente de cluster, o que impede o vazamento de informações causado pela codificação do programa ou pela configuração de texto simples.

Restrições

- O DEW inclui o Key Management Service (KMS), Cloud Secret Management Service (CSMS) e Key Pair Service (KPS). Atualmente, o complemento dew-provider pode se interconectar apenas com o CSMS.
- O complemento dew-provider pode ser instalado somente em clusters v1.19 ou posteriores.
- O complemento dew-provider pode ser instalado em clusters padrão do CCE e em clusters do CCE Turbo.
- Um máximo de 500 objetos SecretProviderClass podem ser criados.
- Quando o complemento é desinstalado, os recursos de CRD relacionados são excluídos de acordo. Mesmo que o complemento seja reinstalado, o objeto SecretProviderClass original não está disponível. Se quiser usar os recursos originais do SecretProviderClass depois que o complemento for desinstalado e reinstalado, crie-os manualmente novamente.

Como funciona o complemento

- Montagem básica: depois que o complemento dew-provider for instalado, você poderá criar um objeto SecretProviderClass e declarar e referenciar o volume em um pod. Quando o pod é iniciado, o segredo declarado no objeto SecretProviderClass é montado no pod.
- Rotação programada: depois que um pod for executado corretamente, se o segredo declarado no objeto SPC e armazenado no CSMS for atualizado, os valores de segredo mais recentes poderão ser atualizados para o pod por meio de rotação programada. Ao usar esse recurso, defina a versão secreta como a **latest**.
- Consciência em tempo real das alterações do SPC: depois que um pod for executado corretamente, se um usuário modificar o segredo declarado no objeto SPC (por exemplo, um segredo é adicionado ou o número da versão é alterado), o complemento pode detectar a mudança em tempo real e atualizar o segredo para o pod.

Instalar o complemento

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster. Clique em **Add-ons** no painel de navegação, localize **CCE Secrets Manager for DEW** à direita e clique em **Install**.

Passo 2 Na página **Install Add-on**, configure os parâmetros na área **Parameters**, conforme listado na tabela a seguir.

Parâmetro	Descrição
rotation_poll_interval	Intervalo de rotação, em unidade de m (em vez de min). O intervalo de rotação indica o intervalo para enviar uma solicitação ao CSMS e obter o último segredo. O intervalo adequado é [1m, 1440m]. O valor padrão é **2m**.

Passo 3 Clique em **Install**.

Depois que o complemento for instalado, selecione o cluster e clique em **Add-ons** no painel de navegação. Na página exibida, visualize o complemento na área **Add-ons Installed**.

----Fim

Componentes

Tabela 13-51 Componentes do dew-provider

Componente	Descrição	Tipo de recurso
dew-provider	Um componente que obtém segredos especificados do CSMS e os monta nos pods	DaemonSet

Componente	Descrição	Tipo de recurso
secrets-store-csi-driver	Um componente que mantém dois CRDs, SecretProviderClass (SPC) e SecretProviderClassPodStatus (spcPodStatus). **SPC** é usado para descrever o segredo em que os usuários estão interessados (como a versão secreta e o nome). Ele é criado por usuários e será referenciado em pods. **spcPodStatus** é usado para rastrear as relações de vinculação entre pods e segredos. Ele é criado automaticamente pelo csi-driver e não requer operação manual. Um pod corresponde a um spcPodStatus. Depois que um pod é iniciado, um spcPodStatus é gerado para o pod. Quando o ciclo de vida do pod termina, o spcPodStatus é excluído de acordo.	DaemonSet

Uso de complemento

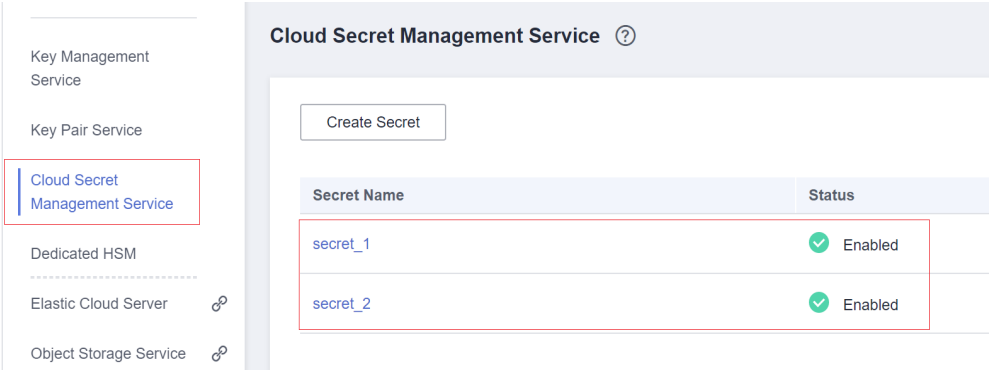
Passo 1 Crie um ServiceAccount.

1. Crie um objeto ServiceAccount, **que declara os nomes de segredo que podem ser usados pelos serviços. Se um usuário fizer referência a um segredo que não esteja declarado aqui, a montagem falhará. Como resultado, o pod não pode ser executado**.

   Crie o arquivo **serviceaccount.yaml** com base no modelo abaixo e declare os nomes de segredos que podem ser usados pelos serviços no campo **cce.io/dew-resource**. Aqui, **secret_1** e **secret_2** são declarados, indicando que o serviço tem permissão para referenciar dois segredos. Em operações subsequentes, se o usuário fizer referência a **secret_3** no serviço, a verificação falhará. Como resultado, o segredo não pode ser montado e o pod não pode ser executado.

   ```
   apiVersion: v1
   kind: ServiceAccount
   metadata:
     name: nginx-spc-sa
     annotations:
       cce.io/dew-resource: "[\"secret_1\",\"secret_2\"]"  #secrets that allow
   pod to use
   ```

   Certifique-se de que os segredos declarados aqui existam no CSCM, conforme mostrado na figura a seguir. Caso contrário, mesmo que a verificação seja bem-sucedida, ocorre um erro quando o segredo correspondente é obtido do CSCM. Como resultado, o pod não pode funcionar corretamente.

2. Execute o seguinte comando para criar a ServiceAccount:

   kubectl apply -f serviceaccount.yaml

3. Verifique se o objeto ServiceAccount foi criado com êxito.

```
$ kubectl get sa
NAME       SECRETS   AGE
default         1         18d   # This is the default ServiceAccount object
of the system.
nginx-spc-sa    1         19s   # This is the newly created ServiceAccount
object.
```

   Um objeto ServiceAccount chamado **nginx-spc-sa** foi criado. Este objeto será referenciado em pods.

Passo 2 Crie um SecretProviderClass.

1. O objeto SecretProviderClass é usado para descrever as informações secretas (como a versão e o nome) que os usuários estão interessados. É criado por usuários e será referenciado em pods.

   Crie o arquivo **secretproviderclass.yaml** usando o modelo abaixo. Preste atenção ao campo **objects** em **parameters**, que é uma matriz usada para declarar o segredo a ser montado.

```
apiVersion: secrets-store.csi.x-k8s.io/v1
kind: SecretProviderClass
metadata:
  name: spc-test
spec:
  provider: cce     # The value is fixed at cce.
  parameters:
    objects: |
          - objectName: "secret_1"
            objectVersion: "v1"
            objectType: "csms"
```

Parâmetro	Tipo	Obrigatório	Descrição
objectName	String	Sim	Nome da credencial. Se vários nomes de objetos estiverem definidos no mesmo SecretProviderClass, os nomes de objetos devem ser únicos. Caso contrário, a montagem falhará.
objectAlias	String	Não	Nome do arquivo do segredo escrito no contêiner. Se este parâmetro não for especificado, o nome do arquivo do segredo escrito no recipiente é o valor de **objectName** por padrão. Se este parâmetro for especificado, o valor deve ser diferente de **objectName** e dos valores de **objectAlias** e **objectName** de outros segredos. Caso contrário, a montagem falhará.
objectType	String	Sim	Tipo de segredo. Atualmente, apenas **csms** é suportado. Outros valores são inválidos.

Parâmetro	Tipo	Obrigatório	Descrição
objectVersion	String	Sim	Versão de segredo. – Especifique uma versão, por exemplo, v1. – Use a versão mais recente (mais recente). Quando **objectVersion** é definido como **latest**, se o segredo correspondente no CSCM for atualizado, ele será atualizado para o pod após um determinado intervalo (**rotation_poll_interval**).

2. Execute o seguinte comando para criar um objeto SecretProviderClass:

   kubectl apply -f secretproviderclass.yaml

3. Verifique se o objeto SecretProviderClass foi criado.

   ```
   $ kubectl get spc
   NAME   AGE
   spc-test   20h
   ```

   Um objeto SecretProviderClass chamado **spc-test** é criado. Este objeto será referenciado em pods posteriormente.

Passo 3 Crie um pod.

A seguir, descrevemos como criar uma aplicação de Nginx.

1. Defina uma carga de trabalho, faça referência ao objeto ServiceAccount criado em **serviceAccountName** e faça referência ao objeto SPC criado em **secretProviderClass**, especifique o caminho de montagem do recipiente em **mountPath**. (Não especifique diretórios especiais como **/** e **/var/run**. Caso contrário, o contêiner pode falhar ao ser iniciado.)

   ```
   apiVersion: apps/v1
   kind: Deployment
   metadata:
     name: nginx-spc
     labels:
       app: nginx
   spec:
     replicas: 1
     selector:
       matchLabels:
         app: nginx
     template:
       metadata:
         labels:
           app: nginx
       spec:
         serviceAccountName: nginx-spc-sa   # Reference the created
   ServiceAccount.
         volumes:
           - name: secrets-store-inline
             csi:
               driver: secrets-store.csi.k8s.io
               readOnly: true
               volumeAttributes:
                 secretProviderClass: "spc-test"  # Reference the created SPC.
         containers:
           - name: nginx-spc
             image: nginx:alpine
             imagePullPolicy: IfNotPresent
             volumeMounts:
   ```



```
            - name: secrets-store-inline
              mountPath: "/mnt/secrets-store"  # Define the mount path of
secrets in the container.
              readOnly: true
      imagePullSecrets:
        - name: default-secret
```

2. Crie um pod.
   ```
   kubectl apply -f deployment.yaml
   ```
3. Verifique se o pod foi criado.
   ```
   $ kubectl get pod
   NAME                      READY   STATUS    RESTARTS   AGE
   nginx-spc-67c9d5b594-642np     1/1     Running    0            20s
   ```
4. Acesse o contêiner e verifique se o segredo especificado está escrito corretamente. Por exemplo:
   ```
   $ kubectl exec -ti nginx-spc-67c9d5b594-642np -- /bin/bash
   root@nginx-spc-67c9d5b594-642np:/#
   root@nginx-spc-67c9d5b594-642np:/# cd /mnt/secrets-store/
   root@nginx-spc-67c9d5b594-642np:/mnt/secrets-store#
   root@nginx-spc-67c9d5b594-642np:/mnt/secrets-store# ls
   secret_1
   ```

   A saída do comando mostra que secret_1 declarado no objeto SPC foi gravado no pod.

   Além disso, você pode obter **spcPodStatus** para verificar a relação de vinculação entre pods e segredos. Por exemplo:

   ```
   $ kubectl get spcps
   NAME                                                                    AGE
   nginx-spc-67c9d5b594-642np-default-spc-test   103s
   $ kubectl get spcps nginx-spc-67c9d5b594-642np-default-spc-test -o yaml
   ......
   status:
   mounted: true
   objects:    # Mounted secret
   - id: secret_1
   version: v1
   podName: nginx-spc-67c9d5b594-642np   # Pod that references the SPC object
   secretProviderClassName: spc-test               # SPC object
   targetPath: /mnt/paas/kubernetes/kubelet/pods/6dd29596-5b78-44fb-9d4c-
   a5027c420617/volumes/kubernetes.io~csi/secrets-store-inline/mount
   ```

----Fim

Rotação programada

Como descrito anteriormente, você pode usar este complemento para completar os segredos de montagem, ou seja, você pode gravar os segredos armazenados no CSMS em um pod.

Para alterar a versão secreta declarada no objecto SPC para **latest**, execute o seguinte comando:

```
apiVersion: secrets-store.csi.x-k8s.io/v1
kind: SecretProviderClass
metadata:
  name: spc-test
spec:
  provider: cce
  parameters:
    objects: |
          - objectName: "secret_1"
            objectVersion: "latest"  # change "v1"to "latest"
            objectType: "csms"
```

Depois que o objeto SPC é atualizado, o complemento envia periodicamente uma solicitação ao CSMS para obter o valor de secret_1 da versão mais recente e atualiza o valor para o pod

que faz referência ao objeto SPC. O intervalo para que o complemento envie periodicamente solicitações é especificado por **rotation_poll_interval** definido em **Instalar o complemento**.

Detecção em tempo real de alterações de SPC

As alterações do SPC já são detectadas em tempo real no **Uso de complemento** e na **Rotação programada**. Para demonstração, adicione **secret_2** ao objeto SPC da seguinte forma:

```
apiVersion: secrets-store.csi.x-k8s.io/v1
kind: SecretProviderClass
metadata:
  name: spc-test
spec:
  provider: cce
  parameters:
    objects: |
          - objectName: "secret_1"
            objectVersion: "latest"
            objectType: "csms"
          - objectName: "secret_2"
            objectVersion: "v1"
            objectType: "csms"
```

Depois que o objeto SPC é atualizado, o novo **secret_2** é montado rapidamente no pod que faz referência ao objeto SPC.

Exibir logs de componentes

Veja o pod onde o complemento é executado.

```
$ kubectl get pod -n kube-system
NAME                          READY   STATUS      RESTARTS   AGE
csi-secrets-store-76tj2       3/3     Running     0           11h
dew-provider-hm5fq            1/1     Running     0           11h
```

Veja os logs do pod do componente dew-provider.

```
$ kubectl logs dew-provider-hm5fq -n kube-system
...Log information omitted...
...
```

Visualize os logs do pod do componente csi-secrets-store. Como o pod do componente csi-secrets-store contém vários contêineres, você deve executar o comando **-c** para especificar um contêiner ao exibir os logs do pod. O contêiner de armazenamento de segredos é o principal contêiner de serviço do complemento e contém a maioria dos logs.

```
$ kubectl logs csi-secrets-store-76tj2 -c secrets-store -n kube-system
...Log information omitted...
...
```

13.15 Exportador de métricas de rede do CCE

Introdução

dolphin é um complemento para monitorar e gerenciar o tráfego de rede de contêineres. golfinho da versão atual coleta estatísticas de tráfego de Kata e contêineres comuns em clusters do CCE Turbo.

Este complemento recolhe quantos pacotes e bytes de IPv4 são recebidos e enviados (incluindo aqueles enviados para a rede pública). PodSelectors podem ser usados para

selecionar back-ends de monitoramento para suportar várias tarefas de monitoramento e métricas de monitoramento opcionais. Você também pode obter informações de rótulo de vagens. As informações de monitoramento foram adaptadas ao formato de Prometheus. Você pode chamar a API do Prometheus para visualizar os dados de monitoramento.

Restrições

- Este complemento pode ser instalado somente em clusters do CCE Turbo da versão 1.19 ou posterior. Seus pods podem ser implementados apenas em nós que executam o EulerOS e não podem ser implementados em nós Arm. \
- Esse complemento pode ser instalado em nós que usam o mecanismo de contêiner ou Docker. Em nós em contêineres, ele pode rastrear atualizações de pods em tempo real. Nos nós do Docker, ele pode consultar atualizações de pods no modo de polling.
- Somente as estatísticas de tráfego de contêineres seguros (Kata como o tempo de execução do contêiner) e de contêineres comuns (runC como o tempo de execução do contêiner) em um cluster do CCE Turbo podem ser coletadas.
- Depois que o complemento é instalado, o tráfego não é monitorado por padrão. Crie um CR para configurar uma tarefa de monitoramento para monitoramento de tráfego.
- Verifique se há recursos suficientes no nó para instalar o complemento.
- A origem dos rótulos de monitoramento e dos rótulos de usuário já deve estar disponível antes que um pod seja criado.

Instalar o complemento

Passo 1 Efetue logon no console do CCE e clique no nome do cluster do Turbo CCE para acessar o cluster. Clique em **Add-ons** no painel de navegação, localize **CCE Network Metrics Exporter** à direita e clique em **Install**.

Passo 2 Na página Install Add-on, visualize a configuração do complemento.

Nenhum parâmetro pode ser configurado para o complemento atual.

Passo 3 Clique em **Install**.

Depois que o complemento for instalado, selecione o cluster e clique em **Add-ons** no painel de navegação. Na página exibida, visualize o complemento na área **Add-ons Installed**.

----Fim

Componentes

Tabela 13-52 Componente de dolphin

Componente	Descrição	Tipo de recurso
dolphin	Usado para monitorar o tráfego de rede de contêiner de clusters do CCE Turbo	DaemonSet

Entrega de uma tarefa de monitoramento

Você pode entregar uma tarefa de monitoramento criando um CR. Atualmente, um CR pode ser criado chamando uma API ou usando o comando **kubectl apply** depois de fazer logon em

um nó de trabalho. Em versões posteriores, um CR pode ser criado no console. Um CR representa uma tarefa de monitoramento e fornece parâmetros opcionais, como **selector**, **podLabel** e **ip4Tx**. Para obter detalhes, consulte o modelo de criação de CR abaixo.

```
apiVersion: crd.dolphin.io/v1
kind: MonitorPolicy
metadata:
    name: example-task            # Monitoring task name.
    namespace: kube-system        # The value must be kube-system. This field is
mandatory.
spec:
    selector:                     # (Optional) Backend monitored by the dolphin
add-on, for example, labelSelector. By default, all containers on the node are
monitored.
    matchLabels:
      app: nginx
    matchExpressions:
      - key: app
        operator: In
        values:
          - nginx
  podLabel: [app]               # (Optional) Pod label.
  ip4Tx:                        # (Optional) Indicates whether to collect
statistics about the number of sent IPv4 packets and the number of sent IPv4
bytes. This function is disabled by default.
    enable: true
    ip4Rx:                        # (Optional) Indicates whether to collect
statistics about the number of received IPv4 packets and the number of received
IPv4 bytes. This function is disabled by default.
    enable: true
    ip4TxInternet:                # (Optional) Indicates whether to collect
statistics about the number of sent IPv4 packets and the number of sent IPv4
bytes. This function is disabled by default.
    enable: true
```

PodLabel: você pode inserir os rótulos de vários pods e separá-los com vírgulas (,), por exemplo, [app, version].

Os rótulos devem obedecer às seguintes regras. A expressão regular correspondente é (^[a-zA-Z_]$)|(^([a-zA-Z][a-zA-Z0-9_]|_[a-zA-Z0-9])([a-zA-Z0-9_]){0,254}$).

- Um máximo de cinco rótulos podem ser inseridos. Cada rótulo contém no máximo 256 caracteres.
- O valor não pode começar com um dígito ou sublinhado duplo (_).
- O formato de uma única etiqueta deve estar em conformidade com A-Za-z_0-9.

Exemplo 1

```
apiVersion: crd.dolphin.io/v1
kind: MonitorPolicy
metadata:
  name: example-task
  namespace: kube-system
spec:
  podLabel: [app]
  ip4Tx:
    enable: true
```

No exemplo anterior, o nome da tarefa de monitoramento é **example-task**, que monitora todos os pods em um nó e gera o número de pacotes de IPv4 enviados e o número de bytes enviados. Se o contêiner monitorado contiver o rótulo **app**, as informações de valor-chave do rótulo correspondente serão transportadas nas métricas de monitoramento. Caso contrário, o valor do rótulo correspondente é **not found**.

Exemplo 2


```
apiVersion: crd.dolphin.io/v1
kind: MonitorPolicy
metadata:
  name: example-task
  namespace: kube-system
spec:
  selector:
    matchLabels:
      app: nginx
  podLabel: [test, app]
  ip4Tx:
    enable: true
  ip4Rx:
    enable: true
  ip4TxInternet:
    enable: true
```

No exemplo anterior, o nome da tarefa de monitoramento é **example-task**, que monitora todos os pods que atendem ao labelselector com app=nginx em um nó e gera as seis métricas. Se o contêiner monitorado contiver rótulos **test** e **app**, as informações de valor-chave do rótulo correspondente serão transportadas nas métricas de monitoramento. Caso contrário, o valor do rótulo correspondente é **not found**.

Você pode criar, modificar e excluir tarefas de monitoramento no formato anterior. Atualmente, é possível criar no máximo 10 tarefas de monitoramento. Quando várias tarefas de monitoramento correspondem ao mesmo back-end de monitoramento, cada back-end de monitoramento gera a métrica de monitoramento específica para o número de tarefas de monitoramento.

NOTA

- Se você modificar ou excluir uma tarefa de monitoramento, os dados de monitoramento coletados pela tarefa de monitoramento serão perdidos. Portanto, tenha cuidado ao realizar esta operação.
- Depois que o complemento é desinstalado, o CR da tarefa de monitoramento é removido junto com o complemento.

Verificar estatísticas de tráfego

Os dados de monitoramento coletados por este complemento são exportados no formato exportador de Prometheus, que pode ser obtido de uma das seguintes maneiras:

- Instale o complemento Prometheus, que se interconecta automaticamente com o complemento dolphin e periodicamente coleta informações de monitoramento.
- Acesse diretamente a porta de serviço 10001 fornecida pelo complemento dolphin, por exemplo, http://{POD_IP}:10001/metrics.

Observe que, se você acessar a porta de serviço do dolphin em um nó, permita o acesso do grupo de segurança do nó e do pod.

Você pode instalar o complemento Prometheus para visualizar as informações de monitoramento. Para obter detalhes sobre como usar o complemento Prometheus, consulte **Monitoramento de métricas personalizadas usando o Prometheus**.

Tabela 13-53 Métricas de monitoramento suportadas

Métrica	Parâmetro
Número de pacotes IPv4 enviados para a rede pública	ip4_send_pkt_internet

Métrica	Parâmetro
Número de bytes IPv4 enviados para a rede pública	ip4_send_byte_internet
Número de pacotes IPv4 recebidos	ip4_rcv_pkt
Número de bytes IPv4 recebidos	ip4_rcv_byte
Número de pacotes IPv4 enviados	ip4_send_pkt
Número de bytes IPv4 enviados	ip4_send_byte

- Exemplo 1 (número de pacotes IPv4 enviados para a rede pública):
  ```
  dolphin_ip4_send_pkt_internet{app="nginx",pod="default/nginx-66c9c65dbf-zjg24",task="kube-system/example-task "} 241
  ```
  No exemplo anterior, o namespace do pod é **default**, o nome do pod é **nginx-66c9c65dbf-zjg24**, o rótulo é **app** e o valor é **nginx**. Essa métrica é criada monitorando a tarefa **example-task**, e o número de pacotes IPv4 enviados pelo pod para a rede pública é **241**.
- Exemplo 2 (número de bytes IPv4 enviados para a rede pública):
  ```
  dolphin_ip4_send_byte_internet{app="nginx",pod="default/nginx-66c9c65dbf-zjg24",task="kube-system/example-task" } 23618
  ```
  No exemplo anterior, o namespace do pod é **default**, o nome do pod é **nginx-66c9c65dbf-zjg24**, o rótulo é **app** e o valor é **nginx**. Essa métrica é criada pelo monitoramento da tarefa **example-task**, e o número de bytes IPv4 enviados pelo pod para a rede pública é **23618**.
- Exemplo 3 (número de pacotes IPv4 enviados):
  ```
  dolphin_ip4_send_pkt{app="nginx",pod="default/nginx-66c9c65dbf-zjg24",task="kube-system/example-task "} 379
  ```
  No exemplo anterior, o namespace do pod é **default**, o nome do pod é **nginx-66c9c65dbf-zjg24**, o rótulo é **app** e o valor é **nginx**. Essa métrica é criada monitorando a tarefa **example-task**, e o número de pacotes IPv4 enviados pelo pod é **379**.
- Exemplo 4 (número de bytes IPv4 enviados):
  ```
  dolphin_ip4_send_byte{app="nginx",pod="default/nginx-66c9c65dbf-zjg24",task="kube-system/example-task "} 33129
  ```
  No exemplo anterior, o namespace do pod é **default**, o nome do pod é **nginx-66c9c65dbf-zjg24**, o rótulo é **app** e o valor é **nginx**. Essa métrica é criada pelo monitoramento da tarefa **example-task**, e o número de bytes IPv4 enviados pelo pod é **33129**.
- Exemplo 5 (número de pacotes IPv4 recebidos):
  ```
  dolphin_ip4_rcv_pkt{app="nginx",pod="default/nginx-66c9c65dbf-zjg24",task="kube-system/example-task "} 464
  ```
  No exemplo anterior, o namespace do pod é **default**, o nome do pod é **nginx-66c9c65dbf-zjg24**, o rótulo é **app** e o valor é **nginx**. Esta métrica é criada monitorando a tarefa **example-task**, e o número de pacotes do IPv4 recebidos pelo pod é **464**.
- Exemplo 6 (número de bytes IPv4 recebidos):
  ```
  dolphin_ip4_rcv_byte{app="nginx",pod="default/nginx-66c9c65dbf-zjg24",task="kube-system/example-task "} 34654
  ```
  No exemplo anterior, o namespace do pod é **default**, o nome do pod é **nginx-66c9c65dbf-zjg24**, o rótulo é **app** e o valor é **nginx**. Essa métrica é criada

monitorando a tarefa **example-task**, e o número de bytes IPv4 recebidos pelo pod é **34654**.

NOTA

Se o contêiner não contiver o rótulo especificado, o valor do rótulo no corpo da resposta será **not found**. O formato é o seguinte:

dolphin_ip4_send_byte_internet{test="not found", pod="default/nginx-66c9c65dbf-zjg24",task="default" } 23618

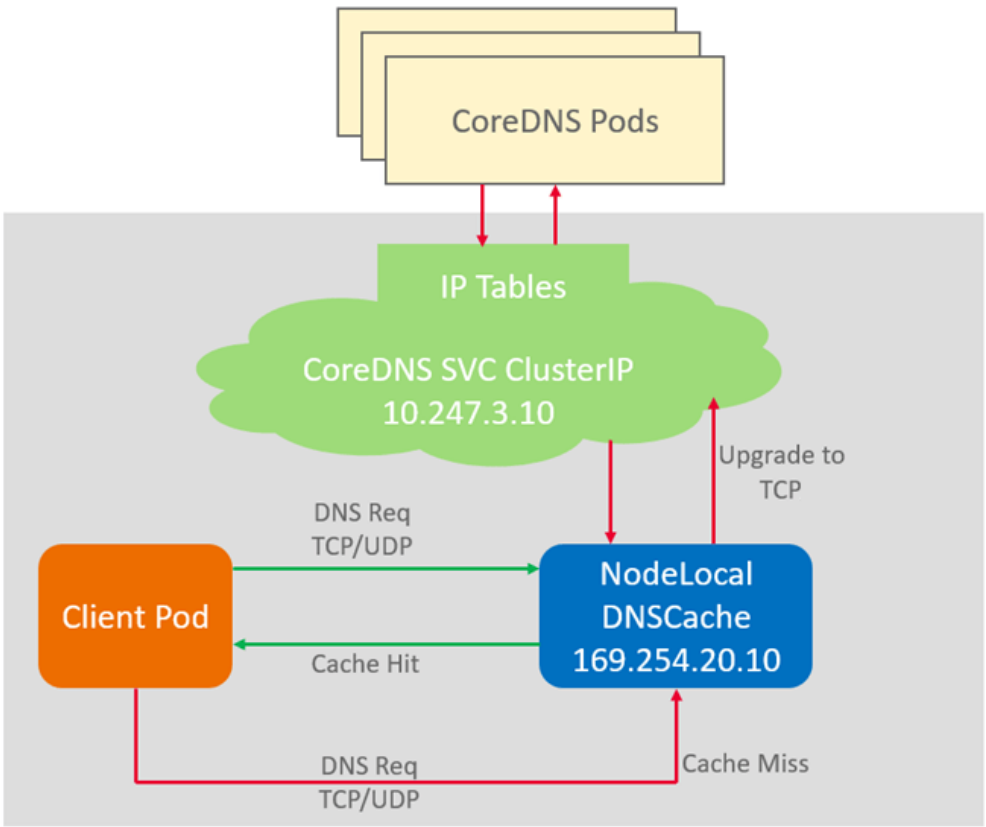
13.16 NodeLocal DNSCache

Introdução

NodeLocal DNSCache é um complemento desenvolvido com base na comunidade **NodeLocal DNSCache**. Esse complemento funciona como um DaemonSet para executar o proxy de cache do DNS em nós de cluster para melhorar o desempenho do DNS do cluster.

Comunidade de código aberto: **https://github.com/kubernetes/dns**

Figura 13-15 Caminho de consulta de NodeLocal DNSCache



Restrições

- Esse recurso está disponível apenas para clusters de v1.19 ou posterior.

Instalar o complemento

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster. Escolha **Add-ons** no painel de navegação, localize **NodeLocal DNSCache** à direita e clique em **Install**.

Passo 2 Na página **Install Add-on**, configure as especificações.

Tabela 13-54 Especificações de complemento

Parâmetro	Descrição
Add-on Specifications	Selecione **Standalone**, **HA** ou **Custom** para **Add-on Specifications**.
Instances	Número de pods que serão criados para corresponder às especificações do complemento selecionado. Se você selecionar **Custom**, poderá ajustar o número de pods conforme necessário.
Containers	As cotas de CPU e memória do contêiner permitidas para as especificações adicionais selecionadas. Se você selecionar **Custom**, poderá ajustar as especificações do contêiner conforme necessário.

Passo 3 Configure os parâmetros do complemento.

- **enable_dnsconfig_admission**: depois que essa função for ativada, um controlador de injeção dinâmica de DNSConfig será criado. O controlador intercepta solicitações de criação de pods no namespace rotulado com **node-localdns-injection=enabled** com base em Admission Webhook e configura automaticamente **Pod dnsConfig** que usa o cache do DNS. Se essa função estiver desabilitada ou se o pod pertencer a um namespace que não seja de destino, será necessário configurar manualmente DNSConfig para o pod.
- **Target Namespace**: esse parâmetro está disponível depois que **DNSConfig Automatic Injection** está ativada. Somente NodeLocal DNSCache da v1.3.0 ou posterior suporta essa função.
  - **All Enabled**: o CCE adiciona o rótulo **node-local-dns-injection=enabled** a todos os namespaces criados, excluindo os internos (como o **kube-system**), identifica solicitações de criação de namespace e adiciona automaticamente o rótulo aos namespaces recém-criados.
  - **Manual configuration**: você deve adicionar manualmente o rótulo **node-local-dns-injection=enabled** aos namespaces que exigem a injeção de DNSConfig. Para mais detalhes, consulte **Gerenciar rótulos de namespace**.

Passo 4 Configure as políticas de agendamento para o complemento.

NOTA

- As políticas de agendamento não entram em vigor em instâncias complementares do tipo DaemonSet.
- Ao configurar a implementação de várias AZs ou a afinidade de nó, verifique se há nós que atendem à política de agendamento e se os recursos são suficientes no cluster. Caso contrário, o complemento não pode ser executado.

Tabela 13-55 Configurações para programação de complementos

Parâmetro	Descrição
Multi-AZ Deployment	• **Preferred**: os pods de Implementação do complemento serão agendados preferencialmente para nós em diferentes AZs. Se todos os nós no cluster forem implementados na mesma AZ, os pods serão agendados para essa AZ. • **Forcible**: os pods de Implementação do complemento serão forçosamente agendados para nós em diferentes AZs. Se houver menos AZs do que pods, os pods extras não funcionarão.
Node Affinity	• **Incompatibility**: a afinidade de nó está desabilitada para o complemento. • **Node Affinity**: especifique os nós em que o complemento é implementado. Se você não especificar os nós, o complemento será agendado aleatoriamente com base na política de agendamento de cluster padrão. • **Specified Node Pool Scheduling**: especifique o pool de nós em que o complemento é implementado. Se você não especificar o pool de nós, o complemento será agendado aleatoriamente com base na política de agendamento de cluster padrão. • **Custom Policies**: insira os rótulos dos nós em que o complemento será implementado para políticas de agendamento mais flexíveis. Se você não especificar rótulos de nó, o complemento será agendado aleatoriamente com base na política de agendamento de cluster padrão. Se várias políticas de afinidade personalizadas estiverem configuradas, certifique-se de que existem nós que atendam a todas as políticas de afinidade no cluster. Caso contrário, o complemento não pode ser executado.
Taints and Tolerations	O uso de manchas e tolerâncias permite (não forçosamente) que Implementação do complemento seja agendada para um nó com as manchas correspondentes e controla as políticas de despejo de Implementação depois que o nó onde a Implementação está localizada é contaminado. O complemento adiciona a política de tolerância padrão para as manchas **node.kubernetes.io/not-ready** e **node.kubernetes.io/unreachable**, respectivamente. A janela de tempo de tolerância é 60s. Para mais detalhes, consulte **Manchas e tolerâncias**.

Passo 5 Clique em **Install**.

----Fim

Componentes

Tabela 13-56 Componentes do NodeLocal DNSCache

Componente	Descrição	Tipo de recurso
node-local-dns-admission-controller	Injeção automática de DNSConfig	Implementação
node-local-dns-cache	Proxy de cache do DNS em nós para melhorar o desempenho do DNS do cluster	DaemonSet

Usar o DNSCache do NodeLocal

Por padrão, as solicitações de aplicativos são enviadas por meio do proxy de CoreDNS. Para usar node-local-dns como proxy de cache de DNS, use um dos seguintes métodos:

- Injeção automática: configure automaticamente o campo **dnsConfig** do pod ao criar o pod. (Pods não podem ser injetados automaticamente em namespaces do sistema, como kube-system.)
- Configuração manual: configure manualmente o campo **dnsConfig** do pod.

Injeção automática

Devem ser satisfeitas as seguintes condições:

- **A injeção automática de DNSConfig** foi ativada durante a instalação do complemento.
- O rótulo **node-local-dns-injection=enabled** foi adicionado ao namespace. Por exemplo, execute o seguinte comando para adicionar o rótulo ao namespace **default**:

  kubectl label namespace *default* **node-local-dns-injection=enabled**
- O novo pod não é executado em namespaces do sistema, como namespace de kube-system e kube-public.
- O rótulo **node-local-dns-injection=disabled** para desabilitar a injeção de DNS não é adicionado ao novo pod.
- O novo pod usa a rede de host e **DNSPolicy** é **ClusterFirstWithHostNet**. Como alternativa, o pod não usa a rede host e **DNSPolicy** é **ClusterFirst**.

Depois que a injeção automática é ativada, as seguintes configurações do **dnsConfig** são adicionadas automaticamente ao pod criado. Além do endereço DNSCache do NodeLocal 169.254.20.10, o endereço de CoreDNS 10.247.3.10 é adicionado a **nameservers**, garantindo alta disponibilidade do servidor DNS do serviço.

```
...
  dnsConfig:
    nameservers:
      - 169.254.20.10
      - 10.247.3.10
    searches:
      - default.svc.cluster.local
      - svc.cluster.local
      - cluster.local
    options:
```



```
      - name: timeout
        value: ''
      - name: ndots
        value: '5'
      - name: single-request-reopen
...
```

Configuração manual

Adicione manualmente as configurações de **dnsConfig** ao pod.

Crie um pod e adicione o endereço IP do DNSCache do NodeLocal 169.254.20.10 à configuração de nameservers de DNSConfig.

NOTA

Os endereços de DNSCache do NodeLocal de diferentes tipos de cluster são os seguintes:

- Cluster do CCE: 169.254.20.10
- Cluster do CCE Turbo: 169.254.1.1

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - image: nginx:alpine
    name: container-0
  dnsConfig:
    nameservers:
    - 169.254.20.10
    - 10.247.3.10
    searches:
    - default.svc.cluster.local
    - svc.cluster.local
    - cluster.local
    options:
    - name: ndots
      value: '2'
  imagePullSecrets:
  - name: default-secret
```

Desinstalar o complemento

A desinstalação do complemento afetará os pods que usaram o endereço node-local-dns para a resolução de nomes de domínio. Antes de desinstalar o complemento, exclua o rótulo **node-local-dns-injection=enabled** dos namespaces envolvidos e exclua e recrie os pods com esse rótulo.

Passo 1 Verifique o complemento.

1. Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster. Escolha **Add-ons** no painel de navegação, localize **NodeLocal DNSCache** à direita e clique em **Edit**.
2. Na área **Parameters**, verifique se **DNSConfig Automatic Injection** está ativada.

   Se **DNSConfig Automatic Injection** tiver sido ativada:

   a. No painel de navegação, escolha **Namespaces**.

   b. Localize as linhas que contêm os namespaces com o rótulo **node-local-dns-injection=enabled** e exclua o rótulo. Para mais detalhes, consulte **Gerenciar rótulos de namespace**.

c. Exclua os pods nesses namespaces e recrie pods.

Se **DNSConfig Automatic Injection** não tiver sido ativada:

a. Use o kubectl para acessar o cluster.

b. Verifique os pods com DNSConfig injetada manualmente. Se vários namespaces estiverem envolvidos, verifique todos os pods nesses namespaces.

Por exemplo, para verificar pods no namespace **default**, execute o seguinte comando:

```
kubectl get pod -n default -o yaml
```

c. Remova manualmente DNSConfig e recriar pods.

Passo 2 Desinstale NodeLocal DNSCache.

1. No painel de navegação, escolha **Add-ons**. Localize **NodeLocal DNSCache** e clique em **Uninstall**.
2. Na caixa de diálogo exibida, clique em **Yes**.

----Fim

Links úteis

Uso do DNSCache do NodeLocal para melhorar o desempenho do DNS

13.17 Monitoramento de cluster da nuvem nativa

Introdução

kube-prometheus-stack usa o Prometheus-operator e o Prometheus para fornecer monitoramento de cluster do Kubernetes de ponta a ponta fácil de usar.

Este complemento permite que os dados de monitoramento sejam interconectados com Análise inteligente de contêineres (CIA) para que você possa ver os dados de monitoramento e configurar alarmes no console da CIA.

Comunidade de código aberto: **https://github.com/prometheus/prometheus**

Restrições

- Por padrão, o componente kube-state-metrics do complemento não coleta rótulos e anotações de recursos do Kubernetes. Para coletar esses rótulos e anotações, habilite manualmente a função de coleta nos parâmetros de inicialização e verifique se as métricas correspondentes são adicionadas à lista branca de coleta de ServiceMonitor denominada **kube-state-metrics**. Para mais detalhes, consulte **Coletar todos os rótulos e anotações de um pod**.

Permissões

O componente node-exporter do complemento kube-prometheus-stack precisa ler os dados de informações do Docker do diretório **/var/run/docker.sock** no host para monitorar o espaço em disco do Docker.

A seguinte permissão é necessária para executar node-exporter:

- cap_dac_override: lê os dados de informações do Docker.

Instalar o complemento

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster. Escolha **Add-ons** no painel de navegação, localize **Cloud Native Cluster Monitoring** à direita e clique em **Install**.

Passo 2 Na página **Install Add-on**, configure as especificações.

- **Containers**: instância do componente criada pelo complemento. Para mais detalhes, consulte **Componentes**. Você pode selecionar ou personalizar uma especificação conforme necessário.

Passo 3 Configure os parâmetros relacionados.

- **Connect to Third Party**: para relatar dados do Prometheus a um sistema de monitoramento de terceiros, insira o endereço e o token do sistema de monitoramento de terceiros e determine se deve ignorar a autenticação de certificado.
- **Prometheus HA**: os componentes Prometheus-server, Prometheus-operator, thanos-query, custom-metrics-apiserver e alertmanager são implementados no modo de várias instâncias no cluster.
- **Install Grafana**: use Grafana para visualizar dados de monitoramento. Grafana cria um volume de armazenamento de 5 GiB por padrão. Desinstalar o complemento **não excluirá esse volume**. O nome de usuário e a senha padrão para o primeiro logon são **admin**. Você será solicitado a alterar a senha imediatamente após o logon.
- **Collection Period**: período de coleta de dados de monitoramento.
- **Data Retention**: período de retenção dos dados de monitoramento.
- **Storage**: selecione o tipo e o tamanho do disco para armazenar dados de monitoramento. Desinstalar o complemento não excluirá esse volume.

  NOTA

  Uma PVC disponível chamado **pvc-prometheus-server** existe no namespace **monitoring** e será usado como fonte de armazenamento.

Passo 4 Clique em **Install**.

Depois que o complemento for instalado, talvez seja necessário executar as seguintes operações:

- Para usar esse complemento para fornecer métricas de recursos do sistema (como uso de CPU e memória) para o dimensionamento automático da carga de trabalho, ative a Metric API. Para mais detalhes, consulte **Fornecer métricas de recursos por meio de Metrics API**. Após a configuração, use o Prometheus para coletar métricas de recursos do sistema sem instalar repetidamente o complemento metrics-server.
- Para usar métricas personalizadas para criar uma política de AS, execute operações em **Agregar métricas personalizadas ao servidor de API do Kubernetes**. Para mais detalhes, consulte **Monitoramento de métricas personalizadas usando o Prometheus**.
- Se você também instalou o grafana ao instalar o complemento, poderá visualizar o painel. Para mais detalhes, consulte **Acessar ao Grafana**.

----Fim

Componentes

Todos os recursos do Kubernetes criados durante a instalação do complemento kube-prometheus-stack são criados no namespace denominado **monitoring**.

Tabela 13-57 Componentes de kube-prometheus-stack

Componente	Descrição	Tipo de recurso
prometheusOperator (nome da carga de trabalho: prometheus-operator)	Implementa e gerencia o Prometheus Server baseado em CustomResourceDefinitions (CRDs) e monitora e processa os eventos relacionados a esses CRDs. É o centro de controle de todo o sistema.	Implementação
prometheus (nome da carga de trabalho: prometheus-server)	Um cluster de Prometheus Server implementado pelo operador com base nos CRDs de Prometheus que podem ser considerados StatefulSets.	StatefulSet
alertmanager (nome da carga de trabalho: alertmanager-alertmanager)	Central de alarme do complemento. Ele recebe alarmes enviados pelo Prometheus e gerencia informações de alarme por meio da deduplicação, do agrupamento e da distribuição.	StatefulSet
thanosSidecar	Disponível apenas no modo HA. É executado com o prometheus-server no mesmo pod para implementar o armazenamento persistente de dados métricos do Prometheus.	Contêiner
thanosQuery	Disponível apenas no modo HA. Entrada para a consulta PromQL quando o Prometheus estiver em cenários de HA. Ele pode excluir métricas duplicadas de Store ou Prometheus.	Implementação
adapter (nome da carga de trabalho: custom-metrics-apiserver)	Agrega métricas personalizadas ao servidor da API do Kubernetes nativo.	Implementação
kubeStateMetrics (nome da carga de trabalho: kube-state-metrics)	Converte os dados de métrica do Prometheus em um formato que pode ser identificado pelas APIs do Kubernetes. Por padrão, o componente kube-state-metrics não coleta todos os rótulos e anotações dos recursos do Kubernetes. Para coletar todos os rótulos e anotações, consulte **Coletar todos os rótulos e anotações de um pod**. **NOTA** Se os componentes forem executados em vários pods, apenas um pod fornecerá métricas.	Implementação

Componente	Descrição	Tipo de recurso
nodeExporter (nome da carga de trabalho: node-exporter)	Implementado em cada nó para coletar dados de monitoramento de nó.	DaemonSet
grafana (nome da carga de trabalho: grafana)	Visualiza dados de monitoramento. Grafana cria um volume de armazenamento de 5 GiB por padrão. Desinstalar o complemento não excluirá esse volume.	Implementação
clusterProblemDetector (nome da carga de trabalho: cluster-problem-detector)	Monitora exceções de cluster.	Implementação

Fornecer métricas de recursos por meio de Metrics API

Métricas de recursos de contêineres e nós, como uso de CPU e memória, podem ser obtidas por meio da Metrics API do Kubernetes. As métricas de recursos podem ser acessadas diretamente, por exemplo, usando o comando **kubectl top** ou usadas pelas políticas HPA ou CustomedHPA para dimensionamento automático.

O complemento pode fornecer a Metrics API ao Kubernetes que está desativada por padrão. Para ativar a API, crie o seguinte objeto de APIService:

```
apiVersion: apiregistration.k8s.io/v1
kind: APIService
metadata:
  labels:
    app: custom-metrics-apiserver
    release: cceaddon-prometheus
  name: v1beta1.metrics.k8s.io
spec:
  group: metrics.k8s.io
  groupPriorityMinimum: 100
  insecureSkipTLSVerify: true
  service:
    name: custom-metrics-apiserver
    namespace: monitoring
    port: 443
  version: v1beta1
  versionPriority: 100
```

Você pode salvar o objeto como um arquivo, nomeá-lo como **metrics-apiservice.yaml** e executar o seguinte comando:

```
kubectl create -f metrics-apiservice.yaml
```

Execute o comando de monitoramento **kubectl top pod -n monitoring**. Se as seguintes informações forem exibidas, a Metrics API poderá ser acessada:

```
# kubectl top pod -n monitoring
NAME                                                  CPU(cores)
MEMORY(bytes)
......
```



```
custom-metrics-apiserver-d4f556ff9-l2j2m                  38m          44Mi
......
```

AVISO

Para desinstalar o complemento, execute o seguinte comando kubectl e exclua o objeto de APIService. Caso contrário, o complemento metrics-server não pode ser instalado devido a recursos residuais do APIService.

```
kubectl delete APIService v1beta1.metrics.k8s.io
```

Agregar métricas personalizadas ao servidor de API do Kubernetes

Por padrão, o kube-prometheus-stack da nova versão não agrega métricas personalizadas ao servidor de API do Kubernetes, e as regras de agregação de métricas não são mais configuradas no item de configuração **user-adapter-config** (ou **adapter-config** em versões anteriores). Para habilitar o dimensionamento HPA com base em métricas personalizadas, configure manualmente as regras de agregação de métricas. Para obter detalhes, consulte **Detecção de métricas e configuração de apresentação**. Se você atualizar o complemento, as configurações originais serão herdadas.

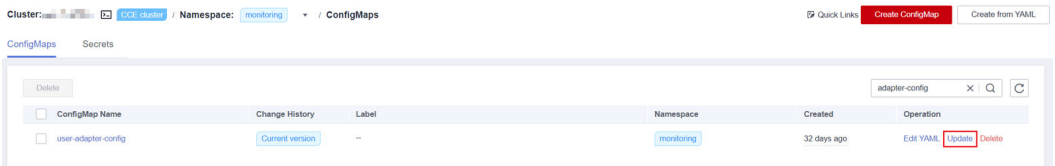
AVISO

Para usar o prometheus para monitorar métricas personalizadas, o aplicativo precisa fornecer uma API de monitoramento de métricas. Para mais detalhes, consulte **Coleta de dados de monitoramento de Prometheus**.

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster. No painel de navegação, escolha **ConfigMaps and Secrets**.

Passo 2 Alterne para o namespace **monitoring**, localize o ConfigMap **user-adapter-config** (**adapter-config** em versões anteriores) na guia **ConfigMaps** e clique em **Update**.

Figura 13-16 Atualizar um ConfigMap



Passo 3 Em **Data**, clique em **Edit** para o arquivo **config.yaml** para adicionar uma regra de coleta de métrica personalizada no campo **rules**. Clique em **OK**.

Você pode adicionar várias regras de coleta adicionando várias configurações no campo **rules**. Para obter detalhes, consulte **Detecção de métricas e configuração de apresentação**.

Exemplo de regra de métrica personalizada:

```
rules:
# The rule matches the accumulated cAdvisor metric in seconds.
- seriesQuery: '{__name__=~"^container_.*",container!="POD",namespace!="",pod!
=""}'
  resources:
    # Specify pod and namespace resources.
    overrides:
```

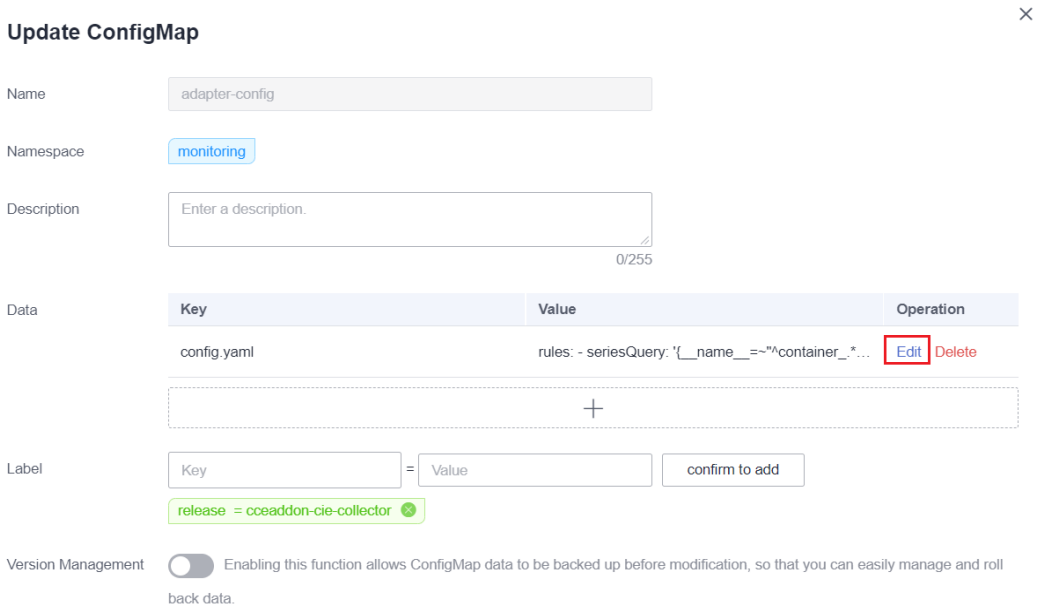


```
      namespace:
        resource: namespace
      pod:
        resource: pod
  name:
    # Delete the container_ prefix and _seconds_total suffix, and use the content
captured in .* as the metric name.
    matches: "^container_(.*)_seconds_total$"
  # Query metrics. .Series and .LabelMatchers are available in the Go language.
Use separators << and >> to avoid conflicts with the Prometheus query language.
  metricsQuery: 'sum(rate(<<.Series>>{<<.LabelMatchers>>,container!="POD"}[2m]))
by (<<.GroupBy>>)'
```

NOTA

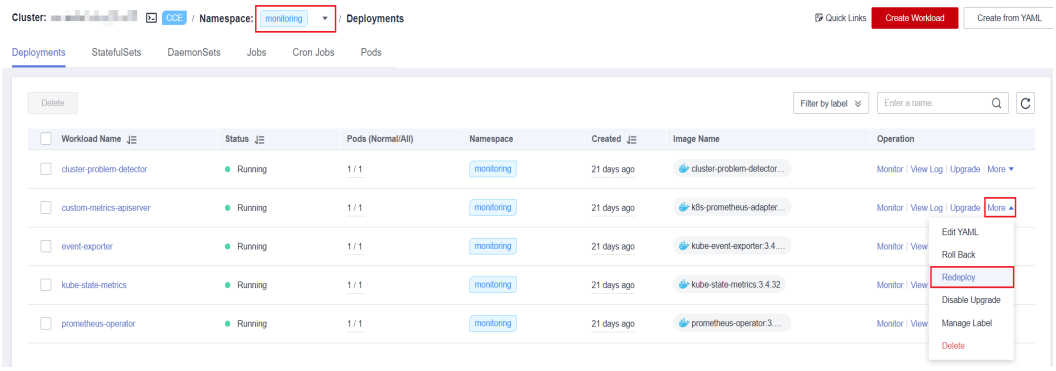
No exemplo anterior, apenas as métricas básicas do pod são coletadas. Para coletar métricas personalizadas, consulte o **guia oficial** para adicionar ou modificar regras.

Figura 13-17 Modificar dados do ConfigMap



Passo 4 Reimplemente a carga de trabalho **custom-metrics-apiserver** no namespace **monitoring**.

Figura 13-18 Reimplementação de custom-metrics-apiserver



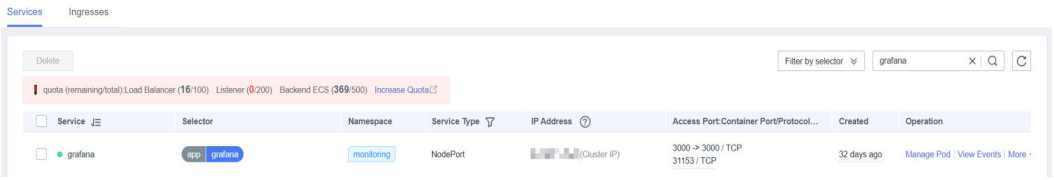
----Fim

Acessar ao Grafana

Se o Grafana for instalado durante a instalação do complemento, você poderá acessar o nó através do Serviço chamado **grafana**, que é um Serviço NodePort. Se o nó for acessado a partir de uma rede externa, você poderá vincular um EIP ao nó e acessar o nó através da porta do nó.

Como mostrado na figura a seguir, o endereço de acesso é **http://{{*Node EIP*}}:30913**.

Figura 13-19 Endereço para acessar Grafana



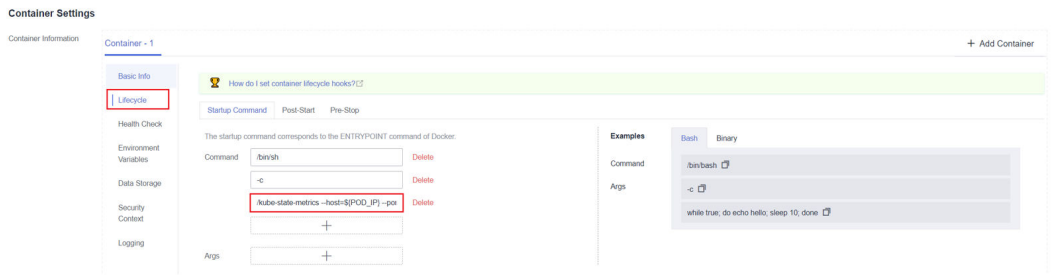
Coletar todos os rótulos e anotações de um pod

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster. No painel de navegação, escolha **Workloads**.

Passo 2 Alterne para o namespace **monitoring**, localize a carga de trabalho **kube-state-metrics** na página de guia **Deployments** e clique em **Upgrade** na coluna **Operation**.

Passo 3 Na área **Lifecycle** das configurações de contêiner, edite o comando de inicialização.

Figura 13-20 Editar o comando de inicialização



Adicione as seguintes informações ao final do parâmetro de inicialização **kube-state-metrics** original:

```
--metric-labels-allowlist=pods=[*],nodes=[node,failure-domain.beta.kubernetes.io/
zone,topology.kubernetes.io/zone]
```

Para coletar anotações, adicione parâmetros nos parâmetros de inicialização da mesma maneira.

```
--metric-annotations-allowlist=pods=[*],nodes=[node,failure-
domain.beta.kubernetes.io/zone,topology.kubernetes.io/zone]
```

AVISO

Ao editar o comando de inicialização, não modifique outros parâmetros de inicialização originais. Caso contrário, o componente pode ser anormal.

Passo 4 **kube-state-metrics** começa a coletar os rótulos/anotações de pods e nós e verifica se **kube_pod_labels/kube_pod_annotations** está na tarefa de coleta de CloudScope.

```
kubectl get servicemonitor kube-state-metrics -nmonitoring -oyaml |
kube_pod_labels
```

----Fim

Para obter mais parâmetros de inicialização do kube-state-metrics, consulte **kube-state-metrics/cli-arguments**.

13.18 Registro de logs da nuvem nativa

Introdução

log-agent é um complemento de coleta de log da nuvem nativa construído com base em OpenTelemetry e fluent-bit de código aberto. Ele suporta políticas de coleta de logs baseadas em CRD, coleta e encaminha logs de saída padrão, logs de arquivos de contêiner, logs de nó e logs de eventos do Kubernetes de contêineres em um cluster.

Depois que o complemento log-agent é instalado, os logs de stdout e os eventos do Kubernetes são coletados por padrão. Para obter detalhes sobre como usar o log-agent para coletar logs, consulte **Uso do log-agent para coletar logs de contêiner**.

Restrições

As restrições sobre o uso do complemento log-agent são as seguintes:

- O log-agent está disponível somente em clusters da v1.17 ou posterior.
- Um máximo de 50 regras de log podem ser configuradas para cada cluster.
- O log-agent não pode coletar arquivos de log .gz, .tar ou .zip.
- Durante a coleta de log de arquivo de contêiner, se o driver de armazenamento Device Mapper for usado em nós, o caminho deverá ser o caminho de montagem do disco de dados do nó.
- Se o tempo de execução do contêiner for containerd, os logs de stdout não podem ser multilinhas.
- Em cada cluster, a taxa de coleta de uma única linha de logs não pode exceder 10.000 de registros por segundo e a taxa de coleta de várias linhas de logs não pode exceder 2.000 registros por segundo.

Permissões

O componente fluent-bit do complemento log-agent lê e coleta os logs de stdout em cada nó, logs de arquivo em pods e logs de nó com base na configuração de coleta.

As seguintes permissões são necessárias para executar o componente fluent-bit:

- CAP_DAC_OVERRIDE: ignora as restrições de controle de acesso discricionário (DAC) em arquivos.
- CAP_FOWNER: ignora as restrições que o ID do proprietário do arquivo deve corresponder ao ID do usuário do processo.
- DAC_READ_SEARCH: ignora as restrições do DAC na leitura de arquivos e na pesquisa de catálogo.

- SYS_PTRACE: permite que todos os processos sejam rastreados.

Instalar o complemento

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster. Escolha **Add-ons** no painel de navegação, localize **Cloud Native Logging** à direita e clique em **Install**.

Passo 2 Na página **Install Add-on**, configure as especificações.

Tabela 13-58 Especificações de complemento

Parâmetro	Descrição
Add-on Specifications	As especificações do complemento podem ser do tipo **Low**, **High** ou **custom-resources**.
Instances	Número de pods que serão criados para corresponder às especificações do complemento selecionado. Se você selecionar **Custom**, poderá ajustar o número de pods conforme necessário.
Containers	O complemento log-agent contém os seguintes contêineres, cujas especificações podem ser ajustadas conforme necessário: ● **fluent-bit**: coletor de log, que é instalado em cada nó como um DaemonSet. ● **cop-logs**: gera e atualiza arquivos de configuração no lado da coleção. ● **log-operator**: analisa e atualiza regras de log. ● **otel-collector**: encaminha logs coletados por **fluent-bit** para LTS.

Passo 3 Configure os parâmetros do complemento.

Um grupo de logs chamado **k8s-log-{*Cluster ID*}** e fluxos de log serão criados automaticamente para coletar e relatar logs com base nas regras de coleta de logs que você configurou.

- Default log rule: por padrão, os logs de stdout e os eventos do Kubernetes são coletados.
- **Access Keys (AK/SK)**: para obter detalhes sobre como obter as chaves de acesso, consulte **Chaves de acesso**.

Passo 4 Configure as políticas de agendamento para o complemento.

NOTA

As políticas de agendamento não entram em vigor em instâncias complementares do tipo DaemonSet.

Tabela 13-59 Configurações para programação de complementos

Parâmetro	Descrição
Implementação de várias AZs	● **Preferred**: os pods da Implementação do complemento serão agendados preferencialmente para nós em diferentes AZs. Se todos os nós no cluster forem implementados na mesma AZ, os pods serão agendados para essa AZ. ● **Forcible**: os pods da Implementação do complemento serão forçosamente agendados para nós em diferentes AZs. Se houver menos AZs do que pods, os pods extras não funcionarão.

Passo 5 Clique em **Install**.

----Fim

Componentes

Tabela 13-60 Componentes do log-agent

Componente	Descrição	Tipo de recurso
fluent-bit	Coletor de logs leve e encaminhador implementados em cada nó para coletar logs	DaemonSet
cop-logs	Usado para gerar links suaves para arquivos coletados e executar no mesmo pod como fluent-bit	DaemonSet
log-operator	Usado para gerar arquivos de configuração internos	Implementação
otel-collector	Usado para coletar logs de aplicações e serviços e relatar os logs para o LTS	Implementação

13.19 e-backup (EOM)

Introdução

O complemento e-backup oferece backup e restauração de cluster. Ele faz backup de dados de aplicações e dados de serviço para o OBS e fornece backup de dados locais e remotos.

Restrições

- Não adicione, exclua ou modifique o cluster durante o backup/restauração. Caso contrário, o backup/restauração pode falhar ou ficar incompleto.
- Se alterar o cluster, é recomendável aguardar 15 minutos até que o cluster esteja estável e, em seguida, executar a operação de backup.
- Quando os snapshots de disco EVS são usados para backup, apenas os PVs do EVS são suportados e as restrições de snapshot se aplicam (por exemplo, a restauração entre AZs não é suportada). O preço é o mesmo dos snapshots de disco EVS.

- Quando restic é usado para backup, os dados dos PVs do EVS, SFS, SFS Turbo e OBS são copiados e carregados no repositório de backup do OBS.
- restic cria um snapshot para os dados no ponto de tempo de backup e carrega os dados, o que não afeta a leitura e a gravação de dados subsequentes. No entanto, restic não verifica o conteúdo do arquivo e a consistência do serviço. Aplicam-se restrições de restic.
- A memória ocupada pelo restic está relacionada ao tamanho dos dados de PV copiados pela primeira vez. Se o tamanho dos dados for maior que 500 GB, é recomendável usar os métodos de migração fornecidos pelos serviços de armazenamento em nuvem. Se você usar este complemento, poderá modificar as cotas de recursos do contêiner de restic consultando o guia de operação.
- Você pode usar Hooks para garantir a consistência dos dados de serviço para aplicações com estado durante o backup, por exemplo, sincronizando dados de memória para arquivos.
- Durante a restauração, você pode ajustar as configurações para se adaptar às diferenças de ambiente antes e depois da migração.
  - Uma aplicação pode ser restaurada do namespace original para outro namespace especificado. No entanto, confirme se a aplicação não é acessada por meio de um Serviço fixo durante a restauração.
  - Você pode alterar o endereço de imagem (repo) da aplicação para outro caminho de imagem. O nome e a tag da imagem permanecem inalterados durante a restauração.
  - Você pode alterar o nome da classe de armazenamento usada pela aplicação para uma nova. Observe que os recursos de armazenamento de back-end devem ser do mesmo tipo, por exemplo, de armazenamento em bloco para armazenamento em bloco.
- Aplicam-se restrições Velero e restic. Por exemplo, durante a restauração, o Serviço limpará o ClusterIP para se adaptar melhor às diferenças entre os clusters do Kubernetes de origem e de destino.

Instalar o complemento

Passo 1 Efetue logon no console do CCE. No painel de navegação, escolha **Add-ons**. Localize o complemento e-backup e clique em **Install**.

Passo 2 Na página **Install Add-on**, selecione o cluster, defina parâmetros e clique em **Install**.

O seguinte parâmetro é suportado:

volumeWorkerNum: número de tarefas de backup de volume simultâneas. O valor padrão é **3**.

----Fim

Usar o complemento

O e-backup usa buckets do OBS como local de armazenamento de backup. Antes de fazer backup dos dados, execute operações em **Preparar chaves** e **Criar um local de armazenamento**.

Os backups podem ser **imediatos** e **agendados**. As restaurações podem ser **imediatos**.

Preparar chaves

Passo 1 Obtenha uma chave de acesso.

Faça logon no console do CCE, mova o cursor para o nome de usuário no canto superior direito e escolha **My Credentials**. No painel de navegação à esquerda, escolha **Access Keys**. Na página exibida, clique em **Add Access Key**.

Passo 2 Crie um arquivo de chave e formatá-lo em uma cadeia usando Base64.

```
# Create a key file.
$ vi credential-for-huawei-obs
HUAWEI_CLOUD_ACCESS_KEY_ID=your_access_key
HUAWEI_CLOUD_SECRET_ACCESS_KEY=your_secret_key

# Use Base64 to format the string.
$ base64 -w 0 credential-for-huawei-obs
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXHWOBS
```

Passo 3 Crie um segredo.

Crie um segredo usando o seguinte conteúdo de YAML:

```
apiVersion: v1
kind: Secret
metadata:
  labels:
    secret.everest.io/backup: 'true'   # The secret is used by e-backup to access 
the backup storage location.
  name: secret-secure-opaque
  namespace: velero                  # The value must be velero. The secret must 
be in the same namespace as e-backup.
type: cfe/secure-opaque
data:
  # String obtained after the credential file is Base64-encoded.
  cloud: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXHWOBS
```

- O segredo deve estar no mesmo namespace do e-backup, ou seja, **velero**.
- **secret.data** armazena o segredo para acessar o OBS. A chave deve ser **cloud**, e o valor é a cadeia Base64 codificada em **Passo 2**. Geralmente, a cadeia codificada em Base64 exibida contém quebras de linha. Excluí-las manualmente ao escrever a cadeia em **secret.data**.
- O segredo deve ser rotulado **secret.everest.io/backup: true**, indicando que o segredo é usado para gerenciar o local de armazenamento de backup.

----Fim

Criar um local de armazenamento

Crie um objeto de recurso do Kubernetes usado pelo e-backup como local de armazenamento de backup para obter e detectar informações sobre o OBS de back-end.

```
apiVersion: velero.io/v1
kind: BackupStorageLocation
metadata:
  name: backup-location-001
  namespace: velero            # The object must be in the same namespace as e-
backup.
spec:
  config:
    endpoint: obs.{regionname}.myhuaweicloud.com    # OBS endpoint
  credential:
    name: secret-secure-opaque   # Name of the created secret
    key: cloud                   # Key in secret.data
  objectStorage:
```

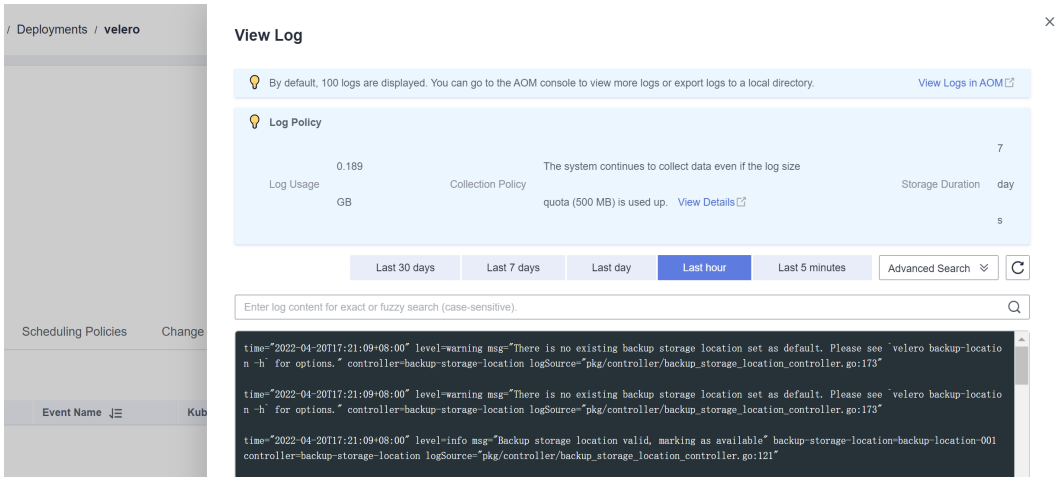
```
    bucket: tools-cce         # OBS bucket name
    prefix: for-backup        #Subpath name
  provider: huawei            # Uses the OBS service.
```

- O campo **prefix** é opcional e outros campos são obrigatórios. O valor de **provider** é fixado em **huawei**.
- Você pode obter o ponto de extremidade de **Regiões e pontos de extremidade**. Certifique-se de que todos os nós no cluster possam acessar o ponto de extremidade. Se o ponto de extremidade não carrega um cabeçalho de protocolo (http ou https), **https** é usado por padrão.
- Defina corretamente **name** e **key** na credencial. Caso contrário, o e-backup não poderá acessar o local de armazenamento.

Após a conclusão da criação, aguarde 30 segundos para verificação e sincronização do local de armazenamento de backup. Em seguida, verifique se **PHASE** está **Available**. O local está disponível somente quando o valor é **Available**.

```
$ kubectl get backupstoragelocations.velero.io backup-location-001 -n velero
NAME                  PHASE       LAST VALIDATED   AGE   DEFAULT
backup-location-001   Available   23s              23m
```

Se **PHASE** não estiver **Available** por um longo tempo, você poderá exibir os logs de e-backup para localizar a falha. Depois que o e-backup é instalado, uma carga de trabalho chamada **velero** é criada no namespace de **velero**, registrada nos logs do velero.



Backup imediato

O processo de backup começa imediatamente e pára após a conclusão. Esse modo é comumente usado para clonagem e migração.

Você pode usar o manifesto Backup abaixo e executar **kubectl create** para criar uma tarefa de backup.

```
apiVersion: velero.io/v1
kind: Backup
metadata:
  name: backup-01
  namespace: velero
spec:
  includedNamespaces:
  - nginx
  - mysql
  labelSelector:
    matchExpressions:
```



```
    - key: direction
      operator: In
      values:
      - back
      - front
    matchLabels:
      app: nginx
      backup: velero
runMode: Normal
appData:
  volumes: Restic
hooks:
  resources:
  - name: hook01
    includedNamespaces:
    - nginx
    labelSelector: {}
    pre:
    - exec:
        command:
        - /bin/sh
        - -c
        - echo hello > hello.txt && echo goodbye > goodbye.txt
        container: container-0
        onError: Fail
        timeout: 30s
    post:
    - exec:
        command:
        - /bin/sh
        - -c
        - echo hello > hello.txt && echo goodbye > goodbye.txt
        container: container-0
        onError: Fail
        timeout: 30s
storageLocation: backup-location-001
ttl: 720h0m0s
```

Descrição do parâmetro:

- Parâmetros de backup
  - **storageLocation**: (**obrigatório**) nome do local de armazenamento de backup onde os dados a serem copiados são armazenados.
  - **ttl**: duração para armazenar backups no local, após o qual os backups são excluídos. O valor deve estar no formato especificado. **h**, **m** e **s** indicam hora, minuto e segundo, respectivamente. Por exemplo, **24h** indica um dia e **3h4m5s** indica três horas, quatro minutos e cinco segundos. O valor padrão é **720h0m0s** (30 dias).
- Filtragem de recursos: os seguintes parâmetros são usados como filtros. A interseção desses campos, se todos configurados, é usada para filtrar todos os recursos no cluster.
  - **includedNamespaces** e **excludedNamespaces**: se deve fazer backup de recursos em determinados namespaces. Esses dois parâmetros entram em conflito entre si. Escolha um para configurar. Por padrão, todos os namespaces são selecionados.
  - **labelSelector**: faz backup de recursos com rótulos específicos. O princípio de funcionamento é o mesmo do Kubernetes.
  - **runMode**: (**obrigatório**) modo de backup. As opções de valor incluem **Normal** (fazendo backup de aplicações e dados), **AppOnly** (fazendo backup de aplicações somente), **DataOnly** (fazendo backup de dados somente) e **DryRun** (sem fazer backup de aplicações e dados; apenas para verificação).
- Backup de dados do serviço: os dados de serviço gerados podem ser copiados através de snapshots do Everest (suportado apenas quando os PVs do EVS como os volumes de

dados) e backups restic (que fazem backup de todos os volumes de dados, exceto os do hostPath). Esses dois modos podem ser usados juntos.

  - **appData**: modo de backup de dados de PV. O valor pode ser **Restic** ou **Snapshot** (não usado por padrão). O modo **Snapshot** entra em vigor somente quando o armazenamento suporta snapshots e o plug-in de snapshot de CSI é implementado no cluster.

- **hook**: os hooks são os comandos executados antes ou depois de um backup para gerenciar com precisão seus backups. Um hook é semelhante ao comando **kubectl exec** e se aplica apenas a pods.
  - **includedNamespaces** e **excludedNamespaces**: se deve executar um gancho em pods em determinados namespaces. Esses dois parâmetros entram em conflito entre si. Escolha um para configurar. Por padrão, todos os namespaces são selecionados.
  - **labelSelector**: executa um hook em pods com certos rótulos. O princípio de funcionamento é o mesmo do Kubernetes.
  - **command**: comando a ser executado.
  - **container**: nome do contêiner no qual o comando é executado. O padrão é o primeiro contêiner quando há vários contêineres no pod.
  - **onError**: ação a ser tomada quando o hook falha ao ser executado. O valor pode ser **Continue** ou **Fail**. Padrão para **Fail**.
  - **Continue** indica que as operações subsequentes continuam independentemente das falhas de execução do hook. **Fail** indica que as operações subsequentes não continuarão após uma falha de execução do hook.
  - **timeout**: tempo limite de execução do hook, após o qual o hook falha. Padrão para 30s.

  Falhas de hook afetam apenas pods. O backup de outros objetos, como Serviços, não é afetado.

  Os hooks não estão disponíveis globalmente. Se o pod para executar um hook não estiver selecionado como o objeto de backup, o hook não será executado. Pode-se considerar que você filtre ainda mais os objetos a serem copiados por meio de **includedNamespaces** ou **excludedNamespaces**.

  NOTA

  Todos os itens configuráveis são descritos acima. A seguir, algumas **sugestões de configuração de backup**.

  - Retenha backups por dia (24 horas).
  - Use **includeNamespace** para especificar o escopo de backup porque, na maioria dos casos, as aplicações são implementados em um namespace específico. Use **labelSelector** para controlar objetos de backup com mais precisão. Antes disso, todos os objetos de destino devem ter rótulos correspondentes. Use **includeNamespace** e **labelSelector** juntos pode satisfazer a maioria dos cenários.
  - Ao usar o Restic para fazer backup de dados de serviço, se você não estiver familiarizado com o modo OUT/IN, poderá ignorar a adição de anotações aos pods que exigem backup de volume. Em vez disso, defina **defaultVolumesToRestic** como **true** para fazer backup dos dados de serviço dos volumes de pod. O valor **false** indica que não há backups.
  - Use hooks para controlar com precisão seus backups. Evite tarefas de longa duração. Não opere diretamente o sistema de arquivos ao executar os comandos no hook.

Após a conclusão do backup, execute os seguintes comandos para exibir o status do backup (**status**):

```
$ kubectl -n velero get backups backup-01 -o yaml | grep  "phase"
  phase: Completed
```

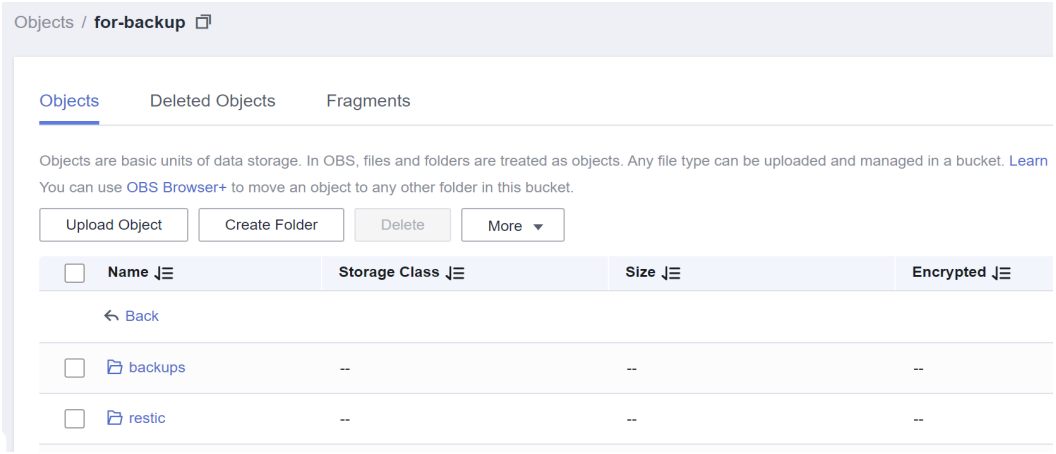


```
$ kubectl -n velero get backups backup-01 -o yaml
......
status:
  ......
```

Status de backup

- **FailedValidation**: o manifesto de backup está configurado incorretamente. Verifique **Backup.Status.ValidationErrors** para encontrar a causa.
- **InProgress**: o backup está em andamento.
- **Completed**: o backup está concluído e não ocorre nenhum erro.
- **PartiallyFailed**: o backup está concluído, mas um erro (como erro de execução do hook) ocorre durante o backup de determinados objetos.
- **Failed**: o backup falha e ocorre um erro que afeta todo o processo.
- **Deleting**: o backup está sendo excluído.

Após a conclusão do backup inicial, as pastas **backups** e **restic** são exibidas no bucket do OBS.



Os logs de backup são armazenados em um bucket do OBS. Suponha que o nome do backup é **backup-001**. Vá para o console do OBS, localize o local de armazenamento com base no nome do bucket configurado e no nome do subcaminho, vá para o diretório **backups/backup-01** e encontre o arquivo **backup-01-logs.gz**. Em seguida, baixe, descomprima e visualize os logs.

Backup periódico

O backup dos dados é feito periodicamente conforme configurado. Esse modo é comumente usado para recuperação de desastres.

Você pode usar o manifesto Schedule abaixo e executar o comando **kubectl create** para criar um cronograma. Você pode rotular o cronograma conforme necessário. Os rótulos que você adicionar no manifesto serão anexados aos backups criados pelo cronograma. Depois que um cronograma é criado em um cluster, um backup é executado imediatamente. Em seguida, os dados são copiados periodicamente, conforme especificado.

```
apiVersion: velero.io/v1
kind: Schedule
metadata:
  name: schedule-backup-001
  namespace: velero
```



```
spec:
  schedule: 0 */10 * * *
  template:
    runMode: Normal
    hooks: {}
    includedNamespaces:
    - nginx
    - mysql
    labelSelector:
      matchExpressions:
      - key: direction
        operator: In
        values:
        - back
        - front
      matchLabels:
        app: nginx
        backup: velero
    storageLocation: backup-location-001
    ttl: 720h0m0s
```

Descrição do parâmetro:

- **schedule**: tempo de execução de backups periódicos. O formato @every e expressões cron padrão do Linux são suportados.
  - @every *N*Unit: **N** é um inteiro positivo. As unidades **s**, **m** e **h** representam segundos, minutos e horas, respectivamente. Por exemplo, **@every 2h30m** indica que o backup é acionado a cada 2 horas e 30 minutos.
  - Expressão Cron. os cinco valores representam minutos, horas, dia do mês, mês e dia da semana, respectivamente.
- **template**: manifesto de backup, que é o mesmo que **spec** em **Backup imediato**.

Excluir um backup

Você pode excluir os objetos de backup e objetos relacionados (como backups, restaurações e agendamentos) de um cluster e excluir backups do local de armazenamento quando uma grande quantidade de dados de backup é gerada.

Você pode usar o manifesto DeleteBackupRequest abaixo e executar o comando **kubectl create** para criar uma solicitação de exclusão de backup.

```
apiVersion: velero.io/v1
kind: DeleteBackupRequest
metadata:
  name: backup-001-delete
  namespace: velero
spec:
  backupName: backup-001  # Name of the backup to be deleted.
```

Consulte o status.

```
$ kubectl -n velero get deletebackuprequests backup-001-delete -o yaml | grep "
phase"
   phase: InProgress
```

- **InProgress**: a tarefa de exclusão está em andamento.
- **Processed**: a tarefa de exclusão foi processada.

⚠ CUIDADO

- O estado **Processed** indica que o e-backup processou a tarefa, mas pode não concluí-la. Você pode verificar os erros no campo **deletebackuprequest.status.errors**. Se o e-backup processar corretamente e completamente a tarefa de exclusão, o objeto **DeleteBackupRequest** também será excluído.
- Não exclua manualmente o conteúdo no local de armazenamento (bucket de OBS).

Restauração imediata

Use um backup imediato como fonte de dados e restaure os dados em outro namespace ou cluster. Este modo aplica-se a todos os cenários.

Você pode usar o manifesto Restore abaixo e executar o comando **kubectl create** para criar uma solicitação de exclusão de backup.

```
apiVersion: velero.io/v1
kind: Restore
metadata:
  name: restore-01
  namespace: velero
spec:
  backupName: backup-01
  hooks:
    resources:
    - name: restore-hook-1
      includedNamespaces:
      - mysql
      labelSelector: {}
      postHooks:
      - init:
          initContainers:
          - name: restore-hook-init1
            image: alpine:latest
            volumeMounts:
            - mountPath: /restores/pvc1-vm
              name: pvc1-vm
            command:
            - /bin/ash
            - -c
            - echo -n "FOOBARBAZ" >> /restores/pvc1-vm/foobarbaz
          - name: restore-hook-init2
            image: alpine:latest
            volumeMounts:
            - mountPath: /restores/pvc2-vm
              name: pvc2-vm
            command:
            - /bin/ash
            - -c
            - echo -n "DEADFEED" >> /restores/pvc2-vm/deadfeed
      - exec:
          execTimeout: 1m
          waitTimeout: 5m
          onError: Fail
          container: mysql
          command:
          - /bin/bash
          - '-c'
          - 'while ! mysql_isready; do sleep 1; done'
      - exec:
          container: mysql
          waitTimeout: 6m
          execTimeout: 1m
          onError: Continue
```



```
          command:
          - /bin/bash
          - '-c'
          - 'mysql < /backup/backup.sql'
includedNamespaces:
- nginx
- mysql
namespaceMapping:
  nginx: nginx-another
  mysql: mysql-another
labelSelector: {}
preserveNodePorts: false
storageClassMapping:
  disk: csi-disk
  obs: csi-obs
imageRepositoryMapping:
  quay.io/coreos:  swr.ap-southeast-1.myhuaweicloud.com/everest
```

Descrição do parâmetro:

- Fonte de dados

  backupName: (**obrigatório**) backup imediato que é usado como fonte de dados.
- Parâmetros de filtragem de recursos: semelhantes aos de **Backup imediato**.
- Processamento personalizado
  - **namespaceMapping**: restaura os dados de backup para outro namespace. O valor é um mapeamento no formato de *Source: Target*. O novo namespace não precisa existir no cluster de destino.
  - **storageClassMapping**: altera o storageClassName usado por recursos de backup como PVs e PVCs. Os tipos de storageClass devem ser os mesmos.
  - **imageRepositoryMapping**: altera o campo **images** do backup. Ele é usado para mapeamento de repositório, excluindo a alteração do nome da imagem e da tag (para evitar que a migração e a atualização sejam acopladas). Por exemplo, depois de migrar **quay.io/coreos/etcd:2.5** para SWR, você pode usar **swr.ap-southeast-1.myhuaweicloud.com/everest/etcd:2.5** no repositório de imagens local. O formato de configuração é o seguinte: **quay.io/coreos: swr.ap-southeast-1.myhuaweicloud.com/everest**
  - **preserveNodePorts**: se você definir esse parâmetro como **false**, o sistema preservará apenas as nodePorts configuradas, não as geradas automaticamente pelo Serviço.
- **hooks**: você pode adicionar init hooks (usados para adicionar initContainers ao pod) e exec hooks (usados para executar alguns comandos). Para obter detalhes sobre como configurar um init hook, consulte a definição de initContainers no Kubernetes. A seguir, descrevemos a configuração geral do hook e os parâmetros de um exec hook.
  - **includedNamespaces** e **excludedNamespaces**: se deve executar um gancho em pods em determinados namespaces. Esses dois parâmetros entram em conflito entre si. Escolha um para configurar. Por padrão, todos os namespaces são selecionados.
  - **labelSelector**: executa um hook em pods com certos rótulos. O princípio de funcionamento é o mesmo do Kubernetes.
  - **command**: comando a ser executado.
  - **container**: nome do contêiner no qual o comando é executado. O padrão é o primeiro contêiner quando há vários contêineres no pod.
  - **onError**: ação a ser tomada quando o hook falha ao ser executado. O valor pode ser **Continue** ou **Fail**. Padrão para **Fail**.

  - **Continue** indica que as operações subsequentes continuam independentemente das falhas de execução do hook. **Fail** indica que as operações subsequentes não continuarão após uma falha de execução do hook.
  - **execTimeout**: tempo limite de execução do hook, após o qual o hook falha. Padrão para 30s.
  - **waitTimeout**: período de tempo limite desde o momento em que o e-backup se prepara para executar o hook até o momento em que o contêiner começa a executar o hook. Se esse período for excedido, o hook falha. O valor padrão é 0s, indicando que não há limite de tempo limite.

NOTA

- Selecione uma fonte de dados correta e verifique se o backup está no estado **Completed**.
- Defina os parâmetros relacionados à filtragem de recursos somente quando necessário.
- Os dados de serviço são restaurados por e-backup com base no modo de backup selecionado. Não são necessárias configurações ou operações manuais.
- Para obter detalhes sobre como usar hooks, consulte as sugestões de uso em *Backup imediato*. Você pode pular **waitTimeout**, a menos que necessário.
- É aconselhável restaurar o que foi feito backup em um novo namespace para evitar erros de configuração que possam desabilitar a aplicação restaurada.

Depois que a restauração for concluída, execute os seguintes comandos para exibir o status da restauração (**status**):

```
$ kubectl -n velero get restores restore-01 -o yaml | grep " phase"
  phase: Completed

$ kubectl -n velero get restores restore-01 -o yaml
......
status:
  ......
```

Descrição de status

- **FailedValidation**: o manifesto de restauração está configurado incorretamente. Verifique **Restore.Status.ValidationErrors** para encontrar a causa.
- **InProgress**: a restauração está em andamento.
- **Completed**: a restauração está concluída e não ocorre nenhum erro.
- **PartiallyFailed**: a restauração está concluída, mas ocorre um erro (como erro de execução de hook) durante a restauração de determinados objetos.
- **Failed**: a restauração falha e ocorre um erro que afeta todo o processo.

Verifique os logs, avisos e erros gerados durante a restauração.

Suponha que o nome da restauração é **restore-01**. Vá para o console do OBS, localize o local de armazenamento com base no nome do bucket configurado e no nome do subcaminho e vá para o diretório **restores/restore-01**. Os dois arquivos a seguir existem:

- **restore-01-logs.gz**: arquivo de log, que pode ser baixado, descompactado e visualizado.
- **restore-01-results.gz**: restaure o arquivo de resultados, incluindo avisos e erros.

13.20 web-terminal (EOM)

O complemento web-terminal é um servidor de terminal leve que permite usar kubectl na interface do usuário da web. Ele fornece uma interface de linha de comando remota (CLI) via

navegador da Web e HTTP, e pode ser facilmente integrado a um sistema independente. Você pode acessar diretamente o complemento como um serviço para obter informações e fazer logon em um servidor através de cmdb.

O web-terminal pode ser executado em todos os sistemas operacionais suportados pelo Node.js e não depende de módulos locais. É rápido e fácil de instalar e suporta várias sessões.

Comunidade de código aberto: **https://github.com/rabchev/web-terminal**

Restrições

- Esse complemento só pode ser instalado em clusters da v1.21 ou anterior. Clusters de Arm não são suportados.
- web-terminal não é mais evoluído. Use o CloudShell em vez disso.
- Ao instalar o web-terminal para usar o kubectl, você deve fazer logon usando sua conta na nuvem ou como um usuário do IAM com a permissão CCE Administrator. Para obter detalhes sobre como controlar a permissão de kubectl, consulte **Controle de permissões de web-terminal**.
- O complemento web-terminal pode ser usado somente após a instalação de CoreDNS em um cluster.

Precauções

O complemento web-terminal pode ser usado para gerenciar clusters do CCE. Mantenha a senha de logon segura para evitar operações inesperadas.

Instalar o complemento

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster. Escolha **Add-ons** no painel de navegação, localize **web-terminal** à direita e clique em **Install**.

Passo 2 Configure os seguintes parâmetros:

- **Access Mode**: o valor é fixado em **NodePort**. O complemento web-terminal é acessado no modo NodePort por padrão e só pode ser usado se qualquer nó no cluster tiver um EIP. Se esse tipo de acesso for selecionado, um EIP deve ser vinculado ao cluster onde o web-terminal será instalado.
- **Username**: o valor padrão é **root** e não pode ser alterado.
- **Password**: senha para fazer logon no web-terminal. Mantenha segura a senha. O complemento web-terminal pode ser usado para gerenciar clusters do CCE. Mantenha a senha de logon segura para evitar operações inesperadas.
- **Confirm Password**: insira a nova senha novamente.

Passo 3 Clique em **Install**.

----Fim

Conexão a um cluster usando o complemento web-terminal

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster. Escolha **Add-ons** no painel de navegação.

Passo 2 Encontre **web-terminal** à direita e clique em **Access**.

----Fim

Controle de permissões de web-terminal

Depois que o web-terminal é instalado, o kubectl usa cluster-admin de ClusterRole por padrão e pode operar recursos do Kubernetes no cluster. Para alterar para outro ClusterRole manualmente, você pode executar **kubectl edit clusterrolebinding web-terminal** para modificar a ServiceAccount do web-terminal.

Para obter detalhes sobre ClusterRole e o ClusterRoleBinding consulte **Permissões de namespace (com base no RBAC do Kubernetes)**.

AVISO

- Permissões de web-terminal configuradas manualmente podem ser redefinidas depois que o complemento web-terminal é atualizado. É aconselhável fazer backup das configurações antes da atualização.
- Antes de usar o kubectl para modificar ClusterRoleBindings, verifique se o kubectl foi configurado com as permissões necessárias.

13.21 Prometheus (EOM)

Introdução

Prometheus é uma estrutura de monitoramento e alerta de sistema de código aberto. Ele é derivado do sistema de monitoramento borgmon do Google, que foi criado por ex-funcionários do Google que trabalhavam na SoundCloud em 2012. O Prometheus foi desenvolvido como um projeto comunitário de código aberto e lançado oficialmente em 2015. Em 2016, o Prometheus juntou-se oficialmente à Cloud Native Computing Foundation, depois do Kubernetes.

O CCE permite que você instale rapidamente o Prometheus como um complemento.

Site oficial do Prometheus **https://prometheus.io/**

Comunidade de código aberto: **https://github.com/prometheus/prometheus**

Restrições

O complemento Prometheus é suportado apenas em clusters v1.21 e anteriores.

Recursos

Como uma estrutura de monitoramento de última geração, o Prometheus possui os seguintes recursos:

- Modelo de dados multidimensional poderoso
  a. Os dados de séries temporais são identificados pelo nome da métrica e pelo par chave-valor.

b. Os rótulos multidimensionais podem ser definidos para todas as métricas.
c. Os modelos de dados não requerem cadeias de caracteres separadas por pontos.
d. Os modelos de dados podem ser agregados, cortados e fatiados.
e. O formato de ponto flutuante duplo é suportado. Os rótulos podem ser todos definidos para unicode.

- Instrução de consulta flexível e poderosa (PromQL): uma instrução de consulta suporta adição, multiplicação e conexão para várias métricas.
- Gerenciamento fácil: o Prometheus server é um arquivo binário separado que pode funcionar localmente. Não depende de armazenamento distribuído.
- Eficiente: cada ponto de amostragem ocupa apenas 3,5 bytes, e um servidor de Prometheus pode processar milhões de métricas.
- O modo pull é usado para coletar dados de séries temporais, o que facilita testes locais e evita que servidores defeituosos enviem métricas ruins.
- Os dados de séries temporais podem ser enviados para o Prometheus server no modo push gateway.
- Os usuários podem obter os alvos monitorados por meio da descoberta de serviço ou da configuração estática.
- Várias GUIs visuais estão disponíveis.
- Dimensionamento fácil

Instalar o complemento

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster. Escolha **Add-ons** no painel de navegação, localize **Prometheus** à direita e clique em **Install**.

Passo 2 Na etapa **Configuration**, defina os seguintes parâmetros:

Tabela 13-61 Parâmetros adicionais do Prometheus

Parâmetro	Descrição
Add-on Specifications	Selecione uma especificação de complemento com base nos requisitos de serviço. As opções são as seguintes: ● **Demo(<= 100 contêineres)**: o tipo de especificação aplica-se ao ambiente de demonstração de experiência e função. Nesta especificação, o Prometheus ocupa poucos recursos, mas tem capacidades limitadas de processamento. É aconselhável usar esta especificação quando o número de contêineres no cluster não exceder 100. ● **Small(<= 2000 contêineres)**: é aconselhável usar esta especificação quando o número de contêineres no cluster não exceder 2.000. ● **Medium(<= 5000 contêineres)**: é aconselhável usar esta especificação quando o número de contêineres no cluster não exceder 5.000. ● **Large(> 5000 contêineres)**: é aconselhável usar esta especificação quando o número de contêineres no cluster exceder 5 000.
Instances	Número de pods que serão criados para corresponder às especificações do complemento selecionado. O múmero não pode ser modificado.
Container	As cotas de CPU e memória do contêiner permitidas para as especificações adicionais selecionadas. As cotas não podem ser modificadas.
Data Retention (days)	Número de dias para armazenar dados de monitoramento personalizados. O valor padrão é 15 dias.
Storage	Os discos rígidos na nuvem podem ser usados como armazenamento. Defina os seguintes parâmetros conforme solicitado: ● **AZ**: defina este parâmetro com base nos requisitos do site. Uma AZ é uma região física onde os recursos usam fontes de alimentação e redes independentes. As AZs são fisicamente isoladas, mas interconectadas por meio de uma rede interna. ● **Disk Type**: I/O comum, I/O alta e I/O ultra-alta são suportadas. ● **Capacity**: insira a capacidade de armazenamento com base nos requisitos de serviço. O valor padrão é 10 GB. **NOTA** Se já existir uma PVC no monitoramento de namespace, o armazenamento configurado será usado como a origem do armazenamento.

Passo 3 Clique em **Install**. Após a instalação, o complemento implementa as seguintes instâncias no cluster.

- prometheus-operator: implementa e gerencia o Prometheus Server baseado em CustomResourceDefinitions (CRDs) e monitora e processa os eventos relacionados a esses CRDs. É o centro de controle de todo o sistema.
- prometheus (server): um cluster de Prometheus Server implementado pelo operador com base nos CRDs de Prometheus que podem ser considerados StatefulSets.
- prometheus-kube-state-metrics: converte os dados de métrica do Prometheus em um formato que pode ser identificado pelas APIs do Kubernetes.
- custom-metrics-apiserver: agrega métricas personalizadas ao servidor nativo da API do Kubernetes.
- prometheus-node-exporter: implementado em cada nó para coletar dados de monitoramento de nó.
- grafana: visualiza dados de monitoramento.

----Fim

Fornecer métricas de recursos por meio de Metrics API

Métricas de recursos de contêineres e nós, como uso de CPU e memória, podem ser obtidas por meio da Metrics API do Kubernetes. As métricas de recursos podem ser acessadas diretamente, por exemplo, usando o comando **kubectl top** ou usadas pelas políticas HPA ou CustomedHPA para dimensionamento automático.

O complemento pode fornecer a Metrics API ao Kubernetes que está desativada por padrão. Para ativar a API, crie o seguinte objeto de APIService:

```
apiVersion: apiregistration.k8s.io/v1
kind: APIService
metadata:
  labels:
    app: custom-metrics-apiserver
    release: cceaddon-prometheus
  name: v1beta1.metrics.k8s.io
spec:
  group: metrics.k8s.io
  groupPriorityMinimum: 100
  insecureSkipTLSVerify: true
  service:
    name: custom-metrics-apiserver
    namespace: monitoring
    port: 443
  version: v1beta1
  versionPriority: 100
```

Você pode salvar o objeto como um arquivo, nomeá-lo como **metrics-apiservice.yaml** e executar o seguinte comando:

```
kubectl create -f metrics-apiservice.yaml
```

Execute o comando de monitoramento **kubectl top pod -n monitoring**. Se as seguintes informações forem exibidas, a Metrics API poderá ser acessada:

```
# kubectl top pod -n monitoring
NAME                                                  CPU(cores)
MEMORY(bytes)
......
custom-metrics-apiserver-d4f556ff9-l2j2m              38m          44Mi
......
```


AVISO

Para desinstalar o complemento, execute o seguinte comando kubectl e exclua o objeto de APIService. Caso contrário, o complemento metrics-server não pode ser instalado devido a recursos residuais do APIService.

```
kubectl delete APIService v1beta1.metrics.k8s.io
```

Referência

- Para obter detalhes sobre os conceitos e configurações do Prometheus, consulte a **Documentação oficial do Prometheus**.
- Para obter detalhes sobre como instalar o Node Exporter, consulte o **node_exporter GitHub**.

13.22 FlexVolume (preterido)

Introdução

O FlexVolume, também chamado de driver de armazenamento, funciona como um plug-in padrão do Kubernetes FlexVolume para permitir que contêineres usem recursos de armazenamento do EVS, SFS, OBS e SFS Turbo. Ao instalar e atualizar o driver de armazenamento, você pode instalar e atualizar rapidamente os recursos de armazenamento em nuvem.

O FlexVolume é um complemento de recursos do sistema. Ele é instalado por padrão quando um cluster do Kubernetes v1.13 ou anterior é criado.

Restrições

- Para clusters criados no CCE, o Kubernetes v1.15.11 é uma versão de transição na qual o complemento FlexVolume é compatível com o complemento CSI (**Everest**). Clusters de v1.17 e versões posteriores não suportam mais o FlexVolume. Use o complemento Everest.
- O complemento FlexVolume será mantido pelos desenvolvedores do Kubernetes, mas novas funcionalidades serão adicionadas apenas ao **Everest**. Não crie mais armazenamento do CCE que use o complemento FlexVolume (driver de armazenamento). Caso contrário, o armazenamento pode apresentar mau funcionamento.
- Esse complemento pode ser instalado somente em **clusters da v1.13 ou anterior**. Por padrão, o complemento **Everest** é instalado quando clusters da v1.15 ou posterior são criados.

  NOTA

  Em um cluster v1.13 ou anterior, quando uma atualização ou correção de bug estiver disponível para funcionalidades de armazenamento, você só precisará instalar ou atualizar o complemento do driver de armazenamento. Não é necessário atualizar o cluster ou criar um cluster.

Instalar o complemento

Este extra foi instalado por predefinição. Se for desinstalado devido a alguns motivos, você pode reinstalá-lo executando as seguintes etapas:

Se o driver de armazenamento não estiver instalado em um cluster, execute as seguintes etapas para instalá-lo:

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster. Escolha **Add-ons** no painel de navegação, localize **storage-driver** à direita e clique em **Install**.

Passo 2 Clique em **Install** para instalar o complemento. Observe que o driver de armazenamento não tem parâmetros configuráveis e pode ser instalado diretamente.

----Fim

14 Gráfico do Helm

14.1 Visão geral

O CCE fornece um console para gerenciar gráficos do Helm, ajudando você a implementar facilmente aplicações usando os gráficos e gerenciar aplicações no console. CCE usa Helm v3.8.2 e suporta o upload de pacotes de gráficos do Helm v3. Para mais detalhes, consulte **Implementação de uma aplicação a partir de um gráfico**.

Você também pode usar o cliente de Helm para implementar aplicações diretamente. Se você usar o cliente de Helm para implementar aplicações, o controle de versão não será suportado. Você pode usar Helm v2 ou Helm v3. Para mais detalhes, veja **Implementação de uma aplicação através do cliente de Helm v2** e **Implementação de uma aplicação através do cliente de Helm v3**.

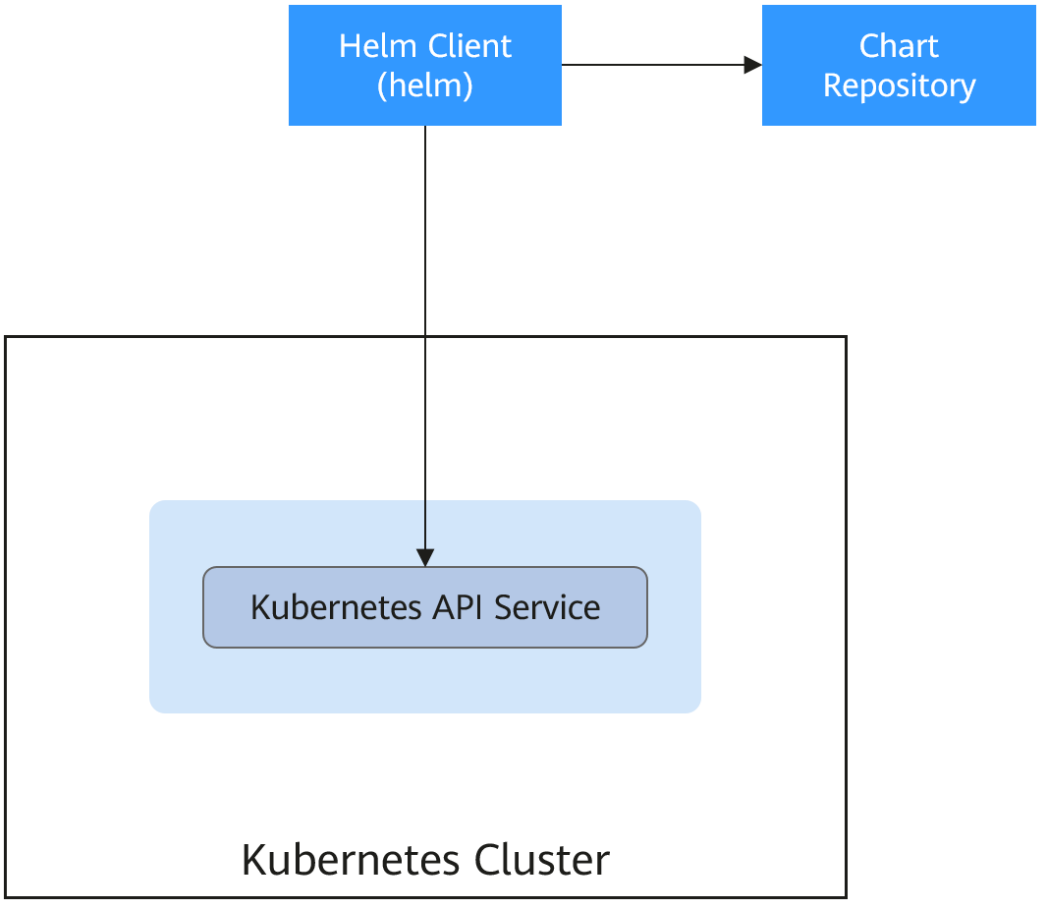
Helm

Helm é um gerenciador de pacotes para Kubernetes e gerencia gráficos. Um gráfico de Helm é uma série de arquivos YAML usados para encapsular aplicações do Kubernetes nativo. Ao implementar uma aplicação, você pode personalizar alguns metadados da aplicação para facilitar a distribuição da aplicação. Os liberadores de aplicações podem usar o Helm para empacotar aplicações, gerenciar dependências de aplicações e versões de aplicações e liberar aplicações para o repositório de software. Depois de usar o Helm, os usuários não precisam compilar arquivos complexos de implementação de aplicações. Eles podem facilmente pesquisar, instalar, atualizar, reverter e desinstalar aplicações no Kubernetes.

A relação entre Helm e Kubernetes é a seguinte:

- Helm <–> Kubernetes
- Apt <–> Ubuntu
- Yum <–> CentOS
- Pip <–> Python

A figura a seguir mostra a arquitetura da solução:



O Helm pode ajudar na orquestração de aplicações para o Kubernetes:

- Gerencia, edita e atualiza um grande número de arquivos de configuração do Kubernetes.
- Implementa uma aplicação de Kubernetes complexa que contém um grande número de arquivos de configuração.
- Compartilha e reutiliza configurações e aplicações do Kubernetes.
- Suporta vários ambientes com modelos de configuração baseados em parâmetros.
- Gerencia a liberação de aplicações, incluindo reverter a aplicação, encontrar diferenças (usando o comando **diff**) e visualizar o histórico de lançamentos.
- Controla fases em um ciclo de implementação.
- Testa e verifica a versão lançada.

14.2 Implementação de uma aplicação a partir de um gráfico

No console do CCE, você pode fazer upload de um pacote de gráfico Helm, implementá-lo e gerenciar os pods implementados.

AVISO

O CCE mudou gradualmente para o Helm v3 desde setembro de 2022. Os gráficos Helm v2 não são mais suportados no console. Se você não pode mudar para Helm v3 por enquanto, você pode usar o cliente de Helm v2 para gerenciar gráficos Helm v2 em segundo plano.

Restrições

- O número de gráficos que podem ser carregados por um único usuário é limitado. O valor exibido no console de cada região é a quantidade permitida.
- O CCE usa Helm v3.8.2 e permite o upload de pacotes de gráficos Helm v3.
- Um gráfico com várias versões consome a mesma quantidade de parte da cota do gráfico.
- Os usuários com permissões de operação de gráfico podem executar várias operações em clusters. Portanto, tenha cuidado ao atribuir aos usuários as permissões de gerenciamento do ciclo de vida do gráfico, incluindo o upload de gráficos e a criação, exclusão e atualização de versões do gráfico.

Especificações de gráfico

A carga de trabalho do Redis é usada como exemplo para ilustrar as especificações do gráfico.

- **Requisito de nomeação**

  Um pacote de gráficos é nomeado no formato de **{name}-{version}**.tgz, onde **{version}** indica o número da versão no formato de *Major version number*.*Minor version number*.*Revision number*, por exemplo, **redis-0.4.2.tgz**.

  NOTA

  O nome de gráfico {name} pode conter um máximo de 64 caracteres.

  O número da versão deve estar em conformidade com as regras de **controle de versão semântico**.

  - Os números da versão principal e secundária são obrigatórios, e o número de revisão é opcional.
  - Os números de versão principal e secundária e o número de revisão devem ser inteiros, maior ou igual a 0, e menor ou igual a 99.

- **Estrutura de diretórios**

  A estrutura de diretórios de um gráfico é a seguinte:

  ```
  redis/
    templates/
    values.yaml
    README.md
    Chart.yaml
    .helmignore
  ```

  Conforme listado em **Tabela 14-1**, os parâmetros marcados com * são obrigatórios.

Tabela 14-1 Parâmetros na estrutura de diretórios de um gráfico

Parâmetro	Descrição
* templates	Armazena todos os modelos.

Parâmetro	Descrição
* values.yaml	Descreve os parâmetros de configuração exigidos pelos modelos. **AVISO** Certifique-se de que o endereço de imagem definido no arquivo **values.yaml** é o mesmo que o endereço de imagem no repositório de imagens de contêiner. Caso contrário, uma exceção ocorre quando você cria uma carga de trabalho e o sistema exibe uma mensagem indicando que a imagem não pode ser extraída. Para obter o endereço da imagem, execute as seguintes operações: efetue logon no console do CCE. No painel de navegação, escolha **Image Repository** para acessar o console do SWR. Escolha **My Images** > **Private Images** e clique no nome da imagem carregada. Na página de guia **Image Tags**, obtenha o endereço da imagem a partir do comando pull. Você pode clicar em  para copiar o comando na coluna **Image Pull Command**.
README.md	Um arquivo de markdown, incluindo: ● A carga de trabalho ou serviços fornecidos pelo gráfico. ● Pré-requisitos para executar o gráfico. ● Configurações no arquivo **values.yaml**. ● Informações sobre a instalação e configuração do gráfico.
* Chart.yaml	Informações básicas sobre o gráfico. Observação: a versão da API do Helm v3 foi alterada de v1 para v2.
.helmignore	Arquivos ou dados que não precisam ler modelos durante a instalação da carga de trabalho.

Carregar um gráfico

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster. Escolha **Charts** no painel de navegação e clique em **Upload Chart** no canto superior direito.

Passo 2 Clique em **Select File**, selecione o gráfico a ser carregado e clique em **Upload**.

NOTA

Quando você carrega um gráfico, a regra de nomeação do bucket do OBS é alterada de cce-charts-{*region*}-{**domain_name**} para cce-charts-{*region*}-{**domain_id**}. Na regra de nomeação antiga, o sistema converte o valor **domain_name** em uma cadeia de Base64 e usa os primeiros 63 caracteres. Se não for possível localizar o gráfico no bucket do OBS com o novo nome, procure o bucket com o nome antigo.

----Fim

Criar uma release

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster. No painel de navegação, escolha **Charts**.

Passo 2 Na página de guia **My Charts**, clique em **Install** do gráfico de destino.

Passo 3 Defina os parâmetros de instalação da carga de trabalho referindo-se a **Tabela 14-2**.

Tabela 14-2 Parâmetros de instalação

Parâmetro	Descrição
Instance	Nome exclusivo da release do gráfico.
Namespace	Namespace ao qual a carga de trabalho será implementada.
Select Version	Versão de um gráfico.
Configuration File	Você pode importar e substituir o arquivo **values.yaml** ou editar diretamente os parâmetros do gráfico on-line. **NOTA** Um arquivo **values.yaml** importado deve estar em conformidade com as especificações de YAML, ou seja, formato KEY:VALUE. Os campos no arquivo não são restritos. O valor da chave do values.yaml importado deve ser o mesmo do pacote de gráficos selecionado. Caso contrário, o values.yaml não terá efeito. Ou seja, a chave não pode ser alterada. 1. Clique em **Select File**. 2. Selecione o arquivo **values.yaml** correspondente e clique em **Open**.

Passo 4 Clique em **Install**.

Na página de guia **Releases**, você pode exibir o status da instalação da release.

----Fim

Atualizar uma carga de trabalho baseada em gráfico

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster. Escolha **Charts** no painel de navegação e clique na guia **Releases**.

Passo 2 Clique em **Upgrade** na linha em que reside a carga de trabalho desejada e defina os parâmetros para a carga de trabalho.

Passo 3 Selecione uma versão do gráfico para **Chart Version**.

Passo 4 Siga os prompts para modificar os parâmetros do gráfico. Clique em **Upgrade** e, em seguida, clique em **Submit**.

Passo 5 Clique em **Back to Release List**. Se o status do gráfico for alterado para **Upgrade successful**, a carga de trabalho será atualizada com êxito.

----Fim

Reverter uma carga de trabalho baseada em gráficos

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster. Escolha **Charts** no painel de navegação e clique na guia **Releases**.

Passo 2 Clique em **More** > **Roll Back** para a carga de trabalho a ser revertida, selecione a versão da carga de trabalho e clique em **Roll back to this version**.

Na lista de cargas de trabalho, se o status for **Rollback successful**, a carga de trabalho será revertida com sucesso.

----Fim

Desinstalar uma carga de trabalho baseada em gráfico

Passo 1 Efetue logon no console do CCE e clique no nome do cluster para acessar o console do cluster. Escolha **Charts** no painel de navegação e clique na guia **Releases**.

Passo 2 Clique em **More** > **Uninstall** ao lado da versão a ser desinstalada e clique em **Yes**. Tenha cuidado ao executar esta operação porque as versões não podem ser restauradas após serem desinstaladas.

----Fim

14.3 Diferenças entre Helm v2 e Helm v3 e soluções de adaptação

Helm v2 pára na versão 2.17.0. Atualmente, o Helm v3 é o padrão na comunidade Helm. Você é aconselhado a mudar seus gráficos para o **formato de Helm v3** o mais rápido possível.

Alterações desde Helm v2:

1. **Remoção de tiller**

   O Helm v3 é mais simples e fácil de usar. Ele remove o timão e se conecta diretamente ao servidor da API usando o kubeconfig, simplificando o modelo de segurança.

2. **Estratégia de atualização aprimorada: patches de mesclagem estratégicos de 3 vias**

   Helm v2 usou um patch de mesclagem estratégico de duas vias. Durante uma atualização, ele comparou o manifesto do gráfico mais recente com o manifesto do gráfico proposto para determinar quais alterações precisavam ser aplicadas aos recursos no Kubernetes. Se as alterações foram aplicadas ao cluster fora de banda (como durante uma edição kubectl), essas alterações não foram consideradas. Isso resultou na impossibilidade de os recursos reverterem para o estado anterior.

   O Helm v3 usa um patch de mesclagem estratégico de três vias. Helm considera o manifesto antigo, seu estado vivo e o novo manifesto ao gerar um patch. O Helm compara o estado ativo atual com o estado ativo do manifesto antigo, verifica se o novo manifesto foi modificado e complementa automaticamente o novo manifesto para gerar o patch de atualização final.

   Para obter detalhes e exemplos, consulte **https://v3.helm.sh/docs/faq/changes_since_helm2**.

3. **Segredos como o driver de armazenamento padrão**

   O Helm v2 usava o ConfigMaps por padrão para armazenar informações de release. No Helm v3, Segredos agora são usados como o driver de armazenamento padrão.

4. **Nomes de release agora estão sob escopo para o namespace**

   No Helm v2, as informações sobre cada lançamento foram armazenadas no mesmo namespace que Tiller. Na prática, isso significava que, uma vez que um nome fosse usado por um release, nenhum outro release poderia usar o mesmo nome, mesmo que fosse implementado em um namespace diferente. No Helm v3, as informações sobre um release específico agora são armazenadas no mesmo namespace da versão em si. Isso

significa que o nome da versão pode ser usado em diferentes namespaces. O namespace da aplicação é o mesmo que o do release.

5. **Alteração do modo de verificação**

   Helm v3 verifica o formato do gráfico de forma mais rigorosa. Por exemplo, o Helm v3 aumenta a apiVersion em Chart.yaml de v1 para v2. Para o Chart.yaml de v2, apiVersion deve ser definida como v1. Depois de instalar o cliente Helm v3, você pode executar o comando **helm lint** para verificar se o formato do gráfico está em conformidade com as especificações do Helm v3.

   Solução de adaptação: adapte o gráfico de Helm v3 com base no documento oficial Helm **https://helm.sh/docs/topics/charts/**. O campo **apiVersion** é obrigatório.

6. **Remoção do hook crd-install**

   O hook **crd-install** foi removido em favor do diretório **crds/** no Helm v3. Note que os recursos no diretório **crds/** são implementados somente durante a instalação do release e não são atualizados durante a atualização. Quando os recursos são excluídos, os recursos são mantidos no diretório **crds/**. Se o CRD já existir, ele será ignorado com um aviso durante a instalação repetida.

   Solução de adaptação: de acordo com o **documento de Helm**, você pode manter seu CRD no diretório **crds/** ou em um gráfico separado. Helm não pode atualizar ou excluir o CRD. Portanto, é aconselhável colocar o CRD em um gráfico e, em seguida, colocar quaisquer recursos que usem esse CRD em outro gráfico.

7. **Recursos que não são criados usando Helm não são forçosamente atualizados. Releases não são forçosamente atualizados por padrão.**

   A lógica de atualização forçada do Helm v3 é alterada. Depois que a atualização falha, o sistema não apaga e recria o Helm v3. Em vez disso, o sistema usa diretamente a lógica **put**. Portanto, a atualização da versão do CCE usa a lógica de atualização não forçada por padrão. Recursos que não podem ser atualizados por meio de patches tornarão a versão incapaz de ser atualizada. Se uma versão com o mesmo nome existir no ambiente e não tiver a tag inicial **app.kubernetes.io/managed-by: Helm** of Helm v3, uma mensagem de conflito é exibida.

   Solução de adaptação: exclua recursos relacionados e crie-os usando o Helm.

8. **Limite de release de registros históricos**

   Somente as versões mais recentes de 10 releases são mantidas por padrão.

Para mais mudanças e detalhes, veja os documentos oficiais do Helm.

- Diferenças entre Helm v2 e Helm v3: **https://v3.helm.sh/docs/faq/changes_since_helm2**
- Como migrar do Helm v2 para o Helm v3: **https://helm.sh/docs/topics/v2_v3_migration**

14.4 Implementação de uma aplicação através do cliente de Helm v2

AVISO

O CCE mudou gradualmente para o Helm v3 desde setembro de 2022. Os gráficos de Helm v2 não são mais suportados no console. Se você não pode mudar para Helm v3 por enquanto, você pode usar o cliente de Helm v2 para gerenciar gráficos de Helm v2 em segundo plano.

Pré-requisitos

O cluster do Kubernetes criado no CCE foi conectado ao kubectl. Para mais detalhes, consulte **Usar o kubectl**.

Precauções

O CCE tentará converter releases de v2 para v3. Se você excluir um release do Helm v2 em segundo plano, as informações de release ainda serão exibidas na página de gráficos no console do CCE. Nesse caso, exclua-o.

Instalar Helm v2

Esta seção usa o Helm v2.17.0 como um exemplo.

Para outras versões, visite **https://github.com/helm/helm/releases**.

Passo 1 Faça download do cliente de Helm da VM conectada ao cluster.

```
wget https://get.helm.sh/helm-v2.17.0-linux-amd64.tar.gz
```

Passo 2 Descomprima o pacote de Helm.

```
tar -xzvf helm-v2.17.0-linux-amd64.tar.gz
```

Passo 3 Copie o Helm para o caminho do sistema, por exemplo, **/usr/local/bin/helm**.

```
mv linux-amd64/helm /usr/local/bin/helm
```

Passo 4 O RBAC está habilitado no servidor de API do Kubernetes. Crie o nome da conta de serviço **tiller** do tiller e atribua cluster-admin, um ClusterRole do sistema, ao tiller. Crie uma conta de recurso de tiller da seguinte forma:

vim tiller-rbac.yaml

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: tiller
  namespace: kube-system
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: tiller
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
```



```
subjects:
  - kind: ServiceAccount
    name: tiller
    namespace: kube-system
```

Passo 5 Implemente a conta de recurso do tiller.

```
kubectl apply -f tiller-rbac.yaml
```

Passo 6 Inicialize o Helm e implementar o pod de tiller.

```
helm init --service-account tiller --skip-refresh
```

Passo 7 Consulte o status.

```
kubectl get pod -n kube-system -l app=helm
```

Saída do comando:

```
NAME                             READY   STATUS    RESTARTS   AGE
tiller-deploy-7b56c8dfb7-fxk5g   1/1     Running   1          23h
```

Passo 8 Consulte a versão do Helm.

```
# helm version
Client: &version.Version{SemVer:"v2.17.0",
GitCommit:"a690bad98af45b015bd3da1a41f6218b1a451dbe", GitTreeState:"clean"}
Server: &version.Version{SemVer:"v2.17.0",
GitCommit:"a690bad98af45b015bd3da1a41f6218b1a451dbe", GitTreeState:"clean"}
```

----Fim

Instalar o gráfico do Helm

Se os gráficos fornecidos pelo CCE não atenderem aos requisitos, baixe um gráfico e instale-o.

Você pode obter o gráfico necessário no diretório **stable** neste **site**, baixar o gráfico e enviá-lo para o nó.

1. Baixe e descomprima o gráfico obtido. Geralmente, o gráfico está no formato ZIP.
   ```
   unzip chart.zip
   ```
2. Instale o gráfico de Helm.
   ```
   helm install aerospike/
   ```
3. Após a conclusão da instalação, execute o comando **helm list** para verificar o status dos releases do gráfico.

Problemas comuns

- A seguinte mensagem de erro é exibida após a execução do comando **helm version**:
  ```
  Client:
  &version.Version{SemVer:"v2.17.0",
  GitCommit:"a690bad98af45b015bd3da1a41f6218b1a451dbe", GitTreeState:"clean"}
  E0718 11:46:10.132102    7023 portforward.go:332] an error occurred
  forwarding 41458 -> 44134: error forwarding port 44134 to pod
  d566b78f997eea6c4b1c0322b34ce8052c6c2001e8edff243647748464cd7919, uid : unable
  to do port forwarding: socat not found.
  Error: cannot connect to Tiller
  ```

  As informações anteriores são exibidas porque o socat não está instalado. Execute o seguinte comando para instalar o socat:

  ```
  yum install socat -y
  ```

- Quando você executa o comando **yum install socat –y** em um nó executando o EulerOS 2.9 e a seguinte mensagem de erro é exibida:

  No match for argument: socat

Error: Unable to find a match: socat

Isso indica que a imagem de socat não está contida. Baixe o pacote RPM do **https://repo.huaweicloud.com/openeuler/openEuler-20.03-LTS-SP3/everything/x86_64/Packages/socat-1.7.3.2-8.oe1.x86_64.rpm** e execute o seguinte comando para instalar o socat:

rpm -i socat-1.7.3.2-8.oe1.x86_64.rpm

- Quando o socat tiver sido instalado e a seguinte mensagem de erro for exibida após a execução do comando **helm version**:

```
test@local:~/k8s/helm/test$ helm version
Client: &version.Version{SemVer:"v3.3.0",
GitCommit:"021cb0ac1a1b2f888144ef5a67b8dab6c2d45be6", GitTreeState:"clean"}
Error: cannot connect to Tiller
```

  O gráfico do Helm lê o certificado de configuração do arquivo **.Kube/config** para se comunicar com o Kubernetes. O erro anterior indica que a configuração do kubectl está incorreta. Nesse caso, reconecte o cluster ao kubectl. Para mais detalhes, consulte **Usar o kubectl**.

- O armazenamento falha ao ser criado depois que você se conecta aos serviços de armazenamento em nuvem.

  Esse problema pode ser causado pelo campo **annotation** na PVC criada. Altere o nome do gráfico e instale-o novamente.

- Se o kubectl não estiver configurado corretamente, a seguinte mensagem de erro será exibida após a execução do comando **helm install**:

```
[root@prometheus-57046 ~]# helm install prometheus/ --generate-name
WARNING: This chart is deprecated
Error: Kubernetes cluster unreachable: Get "http://localhost:8080/version?
timeout=32s": dial tcp [::1]:8080: connect: connection refused
```

  Solução: configure kubeconfig para o nó. Para mais detalhes, consulte **Usar o kubectl**.

14.5 Implementação de uma aplicação através do cliente de Helm v3

Pré-requisitos

O cluster do Kubernetes criado no CCE foi conectado ao kubectl. Para mais detalhes, consulte **Usar o kubectl**.

Instalar Helm v3

Esta seção usa o Helm v3.3.0 como um exemplo.

Para outras versões, visite **https://github.com/helm/helm/releases**.

Passo 1 Faça download do cliente de Helm da VM conectada ao cluster.

```
wget https://get.helm.sh/helm-v3.3.0-linux-amd64.tar.gz
```

Passo 2 Descomprima o pacote de Helm.

```
tar -xzvf helm-v3.3.0-linux-amd64.tar.gz
```

Passo 3 Copie o Helm para o caminho do sistema, por exemplo, **/usr/local/bin/helm**.

```
mv linux-amd64/helm /usr/local/bin/helm
```

Passo 4 Consulte a versão do Helm.


```
helm version
version.BuildInfo{Version:"v3.3.0",
GitCommit:"e29ce2a54e96cd02ccfce88bee4f58bb6e2a28b6", GitTreeState:"clean",
GoVersion:"go1.13.4"}
```

----Fim

Instalar o gráfico do Helm

Você pode usar o Helm para instalar um gráfico. Antes de usar o Helm, você pode precisar entender os seguintes conceitos para usar melhor o Helm:

- Gráfico: contém definições de recursos e um grande número de arquivos de configuração de aplicações de Kubernetes.
- Repositório: armazena gráficos compartilhados. Você pode baixar gráficos do repositório para um caminho local para instalação ou instalá-los on-line.
- Release: resultado da execução depois que um gráfico é instalado em um cluster do Kubernetes usando o Helm. Um gráfico pode ser instalado várias vezes em um cluster. Uma nova release será criada para cada instalação. Um gráfico MySQL é usado como exemplo. Para executar dois bancos de dados em um cluster, instale o gráfico duas vezes. Cada banco de dados tem sua própria release e nome de release.

Para obter mais detalhes, consulte **Uso do Helm**.

Passo 1 Procure um gráfico no repositório do **Artifact Hub** recomendado pelo Helm e configure o repositório do Helm.

```
helm repo add {repo_name} {repo_addr}
```

A seguir, o **gráfico de WordPress** é usado como exemplo:

```
helm repo add bitnami https://charts.bitnami.com/bitnami
```

Passo 2 Execute o comando **helm install** para instalar o gráfico.

- Instalação padrão: este é o método mais simples, que requer apenas dois parâmetros.

  ```
  helm install {release_name} {chart_name}
  ```

  Por exemplo, para instalar o WordPress, o gráfico de WordPress adicionado no **passo 1** é **bitnami/wordpress**, e o nome da release é **my-wordpress**.

  ```
  helm install my-wordpress bitnami/wordpress
  ```

- Instalação personalizada: a instalação padrão usa as configurações padrão no gráfico. Use instalação personalizada para configurações de parâmetros personalizadas. Execute o comando **helm show values** *{chart_name}* para exibir as opções configuráveis do gráfico. Por exemplo, para exibir os itens configuráveis do WordPress execute o seguinte comando:

  ```
  helm show values bitnami/wordpress
  ```

  Sobrescreva os parâmetros especificados executando os seguintes comandos:

  ```
  helm install my-wordpress bitnami/wordpress \
       --set mariadb.primary.persistence.enabled=true \
       --set mariadb.primary.persistence.storageClass=csi-disk \
       --set mariadb.primary.persistence.size=10Gi \
       --set persistence.enabled=false
  ```

Passo 3 Exiba a release do gráfico instalado.

```
helm list
```

----Fim

Problemas comuns

- A seguinte mensagem de erro é exibida após a execução do comando **helm version**:

```
Client:
&version.Version{SemVer:"v3.3.0",
GitCommit:"012cb0ac1a1b2f888144ef5a67b8dab6c2d45be6", GitTreeState:"clean"}
E0718 11:46:10.132102    7023 portforward.go:332] an error occurred
forwarding 41458 -> 44134: error forwarding port 44134 to pod
d566b78f997eea6c4b1c0322b34ce8052c6c2001e8edff243647748464cd7919, uid : unable
to do port forwarding: socat not found.
Error: cannot connect to Tiller
```

  As informações anteriores são exibidas porque o socat não está instalado. Execute o seguinte comando para instalar o socat:

```
yum install socat -y
```

- Quando você executa o comando **yum install socat –y** em um nó executando o EulerOS 2.9 ou EulerOS da Huawei Cloud, a seguinte mensagem de erro é exibida:

```
No match for argument: socat
Error: Unable to find a match: socat
```

  A imagem do nó não contém a imagem do socat. Baixe manualmente o gráfico RPM e execute o seguinte comando para instalá-lo:

```
rpm -i socat-1.7.3.2-8.oe1.x86_64.rpm
```

  NOTA

  Faça o download do gráfico RPM da imagem socat em:

  - EulerOS 2.9: **https://repo.huaweicloud.com/openeuler/openEuler-20.03-LTS-SP3/everything/x86_64/Packages/socat-1.7.3.2-8.oe1.x86_64.rpm**
  - EulerOS 1.1 da Huawei Cloud: **https://repo.huaweicloud.com/hce/1.1/os/x86_64/Packages/socat-1.7.3.2-2.hce1c.x86_64.rpm**
  - EulerOS 2.0 da Huawei Cloud: **https://repo.huaweicloud.com/hce/2.0/os/x86_64/Packages/socat-1.7.3.2-8.hce2.x86_64.rpm**

- Quando o socat tiver sido instalado e a seguinte mensagem de erro for exibida após a execução do comando **helm version**:

```
$ helm version
Client: &version.Version{SemVer:"v3.3.0",
GitCommit:"021cb0ac1a1b2f888144ef5a67b8dab6c2d45be6", GitTreeState:"clean"}
Error: cannot connect to Tiller
```

  O gráfico do Helm lê o certificado de configuração em **.Kube/config** para se comunicar com o Kubernetes. O erro anterior indica que a configuração do kubectl está incorreta. Nesse caso, reconecte o cluster ao kubectl. Para mais detalhes, consulte **Usar o kubectl**.

- O armazenamento falha ao ser criado depois que você se conecta aos serviços de armazenamento em nuvem.

  Esse problema pode ser causado pelo campo **annotation** na PVC criada. Altere o nome do gráfico e instale-o novamente.

- Se o kubectl não estiver configurado corretamente, a seguinte mensagem de erro será exibida após a execução do comando **helm install**:

```
# helm install prometheus/ --generate-name
WARNING: This chart is deprecated
Error: Kubernetes cluster unreachable: Get "http://localhost:8080/version?
timeout=32s": dial tcp [::1]:8080: connect: connection refused
```

  Solução: configure kubeconfig para o nó. Para mais detalhes, consulte **Usar o kubectl**.

14.6 Conversão um release do Helm v2 para v3

Contexto

O CCE suporta totalmente o Helm v3. Esta seção orienta você a converter um release de Helm v2 para Helm v3. Helm v3 descarta ou reconstrói algumas funções Helm v2 na camada inferior. Portanto, a conversão é arriscada até certo ponto. A simulação é necessária antes da conversão.

Para obter detalhes, consulte a **documentação da comunidade**.

Precauções

- Informações de release das lojas Helm v2 em ConfigMaps. Helm v3 faz isso em segredos.
- Quando você consulta, atualiza ou opera um release do Helm v2 no console do CCE, o CCE tentará converter o release para v3. Se você operar em segundo plano, converta o release seguindo as instruções abaixo.

Processo de conversão (sem usar o cliente de Helm v3)

Passo 1 Faça o download do plug-in de conversão helm 2to3 no nó do CCE.

```
wget https://github.com/helm/helm-2to3/releases/download/v0.10.2/
helm-2to3_0.10.2_linux_amd64.tar.gz
```

Passo 2 Descompacte o pacote de plug-in.

```
tar -xzvf helm-2to3_0.10.2_linux_amd64.tar.gz
```

Passo 3 Realize a conversão simulada.

Tome o release test-convert como exemplo. Execute o seguinte comando para simular a conversão: se as informações a seguir forem exibidas, a simulação é bem-sucedida.

```
# ./2to3 convert --dry-run --tiller-out-cluster -s configmaps test-convert
NOTE: This is in dry-run mode, the following actions will not be executed.
Run without --dry-run to take the actions described below:
Release "test-convert" will be converted from Helm v2 to Helm v3.
[Helm 3] Release "test-convert" will be created.
[Helm 3] ReleaseVersion "test-convert.v1" will be created.
```

Passo 4 Realize a conversão. Se as informações a seguir forem exibidas, a conversão será bem-sucedida.

```
# ./2to3 convert --tiller-out-cluster -s configmaps test-convert
Release "test-convert" will be converted from Helm v2 to Helm v3.
[Helm 3] Release "test-convert" will be created.
[Helm 3] ReleaseVersion "test-convert.v1" will be created.
[Helm 3] ReleaseVersion "test-convert.v1" created.
[Helm 3] Release "test-convert" created.
Release "test-convert" was converted successfully from Helm v2 to Helm v3.
Note: The v2 release information still remains and should be removed to avoid
conflicts with the migrated v3 release.
v2 release information should only be removed using `helm 2to3` cleanup and when
all releases have been migrated over.
```

Passo 5 Após a conclusão da conversão, simule a liberação de recursos. Após a simulação, limpe os recursos de release de v2.

Liberação simulada:


```
# ./2to3 cleanup --dry-run --tiller-out-cluster -s configmaps --name test-convert
NOTE: This is in dry-run mode, the following actions will not be executed.
Run without --dry-run to take the actions described below:
WARNING: "Release 'test-convert' Data" will be removed.

[Cleanup/confirm] Are you sure you want to cleanup Helm v2 data? [y/N]: y
Helm v2 data will be cleaned up.
[Helm 2] Release 'test-convert' will be deleted.
[Helm 2] ReleaseVersion "test-convert.v1" will be deleted.
```

Liberação formal:

```
# ./2to3 cleanup --tiller-out-cluster -s configmaps --name test-convert
WARNING: "Release 'test-convert' Data" will be removed.

[Cleanup/confirm] Are you sure you want to cleanup Helm v2 data? [y/N]: y
Helm v2 data will be cleaned up.
[Helm 2] Release 'test-convert' will be deleted.
[Helm 2] ReleaseVersion "test-convert.v1" will be deleted.
[Helm 2] ReleaseVersion "test-convert.v1" d
```

----Fim

Processo de conversão (usando o cliente de Helm v3)

Passo 1 Instale o cliente de Helm v3. Para mais detalhes, consulte **Instalar Helm v3**.

Passo 2 Instale o plug-in de conversão.

```
# helm plugin install https://github.com/helm/helm-2to3
Downloading and installing helm-2to3 v0.10.2 ...
https://github.com/helm/helm-2to3/releases/download/v0.10.2/
helm-2to3_0.10.2_linux_amd64.tar.gz
Installed plugin: 2to3
```

Passo 3 Verifique se o plug-in foi instalado

```
# helm plugin list
NAME VERSION
DESCRIPTION
2to3  0.10.2      migrate and cleanup Helm v2 configuration and releases in-place
to Helm v3
```

Passo 4 Realize a conversão simulada.

Tome o release test-convert como exemplo. Execute o seguinte comando para simular a conversão: se as informações a seguir forem exibidas, a conversão simulada será bem-sucedida.

```
# helm 2to3 convert --dry-run --tiller-out-cluster -s configmaps test-convert
NOTE: This is in dry-run mode, the following actions will not be executed.
Run without --dry-run to take the actions described below:
Release "test-convert" will be converted from Helm v2 to Helm v3.
[Helm 3] Release "test-convert" will be created.
[Helm 3] ReleaseVersion "test-convert.v1" will be created.
```

Passo 5 Realize a conversão. Se as informações a seguir forem exibidas, a conversão será bem-sucedida.

```
# helm 2to3 convert --tiller-out-cluster -s configmaps test-convert
Release "test-convert" will be converted from Helm v2 to Helm v3.
[Helm 3] Release "test-convert" will be created.
[Helm 3] ReleaseVersion "test-convert.v1" will be created.
[Helm 3] ReleaseVersion "test-convert.v1" created.
[Helm 3] Release "test-convert" created.
Release "test-convert" was converted successfully from Helm v2 to Helm v3.
Note: The v2 release information still remains and should be removed to avoid
conflicts with the migrated v3 release.
```



```
v2 release information should only be removed using `helm 2to3` cleanup and when
all releases have been migrated over.
```

Passo 6 Após a conversão, você pode visualizar a versão convertida executando **helm list**.

```
# helm list
NAME                NAMESPACE    REVISION UPDATED
STATUS      CHART              APP VERSION
test-convert      default   1               2022-08-29 06:56:28.166918487 +0000
UTC       deployed    test-helmold-1
```

Passo 7 Após a conclusão da conversão, simule a liberação de recursos. Após a simulação, limpe os recursos de release de v2.

Liberação simulada:

```
# helm 2to3 cleanup --dry-run --tiller-out-cluster -s configmaps --name test-
convert
NOTE: This is in dry-run mode, the following actions will not be executed.
Run without --dry-run to take the actions described below:
WARNING: "Release 'test-convert' Data" will be removed.

[Cleanup/confirm] Are you sure you want to cleanup Helm v2 data? [y/N]: y
Helm v2 data will be cleaned up.
[Helm 2] Release 'test-convert' will be deleted.
[Helm 2] ReleaseVersion "test-convert.v1" will be deleted.
```

Liberação formal:

```
# helm 2to3 cleanup --tiller-out-cluster -s configmaps --name test-convert
WARNING: "Release 'test-convert' Data" will be removed.

[Cleanup/confirm] Are you sure you want to cleanup Helm v2 data? [y/N]: y
Helm v2 data will be cleaned up.
[Helm 2] Release 'test-convert' will be deleted.
[Helm 2] ReleaseVersion "test-convert.v1" will be deleted.
[Helm 2] ReleaseVersion "test-convert.v1" deleted.
[Helm 2] Release 'test-convert' deleted.
Helm v2 data was cleaned up successfully.
```

----Fim

15 Permissões

15.1 Visão geral de permissões

O gerenciamento de permissões do CCE permite que você atribua permissões a usuários e grupos de usuários do IAM em suas contas de locatário. O CCE combina as vantagens do Identity and Access Management (IAM) e da autorização do Controle de Acesso Baseado em Função (RBAC) do Kubernetes para fornecer uma variedade de métodos de autorização, incluindo autorização refinada do IAM, autorização de token do IAM, autorização com escopo de cluster e autorização em todo o namespace.

O CCE permite que você gerencie permissões em clusters e recursos relacionados com uma granularidade mais fina, por exemplo, para controlar o acesso de funcionários em diferentes departamentos aos recursos da nuvem.

Esta seção descreve o mecanismo de gerenciamento de permissões do CCE e conceitos relacionados. Se sua conta atendeu aos seus requisitos de serviço, você pode ignorar as configurações neste capítulo.

Gerenciamento de permissões do CCE

As permissões do CCE são descritas a seguir:

- **Permissões em nível de cluster**: o gerenciamento de permissões em nível de cluster evolui a partir do recurso de autorização de política do sistema do IAM. Os usuários do IAM no mesmo grupo de usuários têm as mesmas permissões. No IAM, você pode configurar políticas do sistema para descrever quais grupos de usuários do IAM podem executar quais operações em recursos de cluster. Por exemplo, você pode conceder ao grupo de usuários A para criar e excluir o cluster X, adicionar um nó ou instalar um complemento, enquanto concede ao grupo de usuários B para exibir informações sobre o cluster X.

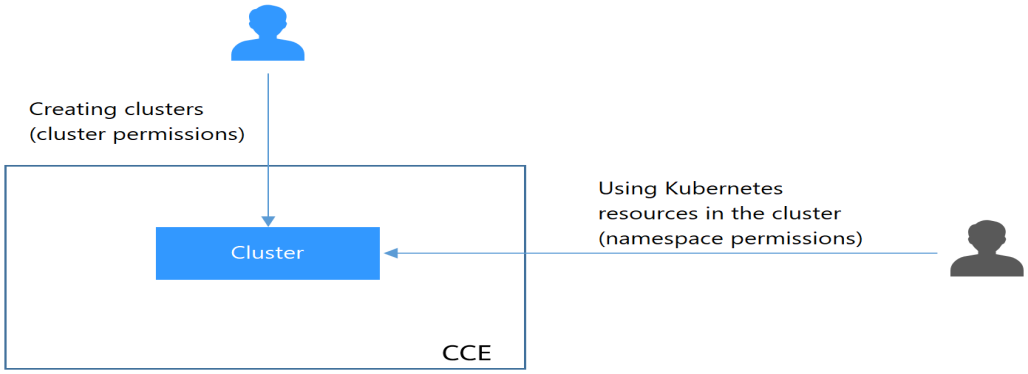
  As permissões em nível de cluster envolvem APIs não Kubernetes em clusters do CCE e oferecem suporte a políticas de IAM refinadas e gerenciamento de projetos empresariais.

- **Permissões em nível de namespace**: você pode regular o acesso de usuários ou grupos de usuários aos recursos do Kubernetes em um único namespace com base em suas funções do RBAC do Kubernetes. O CCE também foi aprimorada com base em recursos de código aberto. Ele suporta autorização do RBAC com base no usuário ou grupo de usuários do IAM e autenticação do RBAC no acesso a APIs usando tokens do IAM.

As permissões em nível de namespace envolvem as APIs do Kubernetes do CCE e são aprimoradas com base nos recursos do RBAC do Kubernetes. As permissões em nível de namespace podem ser concedidas a usuários ou grupos de usuários do IAM para autenticação e autorização, mas são independentes de políticas do IAM refinadas.

Geralmente, você configura permissões do CCE em duas encenações. O primeiro é criar e gerenciar clusters e recursos relacionados, como nós. O segundo é criar e usar recursos do Kubernetes no cluster, como cargas de trabalho e Serviços.

Figura 15-1 Ilustração sobre permissões do CCE



Essas permissões permitem que você gerencie usuários de recursos com uma granularidade mais fina.

Permissões de cluster (baseadas no IAM) e permissões de namespace (baseadas no Kubernetes RBAC)

Usuários com diferentes permissões de cluster (atribuídos usando o IAM) têm diferentes permissões de namespace (atribuídos usando o Kubernetes RBAC). **Tabela 15-1** lista as permissões de namespace de diferentes usuários.

Tabela 15-1 Diferenças nas permissões de namespace

Usuário	Clusters da v1.13 e posterior
Usuário com permissões de Tenant Administrator (por exemplo, uma conta)	Todas as permissões de namespace
Usuário do IAM com a função CCE Administrator	Todas as permissões de namespace
Usuário do IAM com a função CCE FullAccess ou CCE ReadOnlyAccess	Requer autorização do Kubernetes RBAC.
Usuário do IAM com a função Tenant Guest	Requer autorização do Kubernetes RBAC.

Permissões do kubectl

Você pode usar **kubectl** para acessar recursos do Kubernetes em um cluster.

Quando você acessa um cluster usando kubectl, o CCE usa o arquivo kubeconfig.json gerado no cluster para autenticação. Esse arquivo contém informações do usuário, com base no qual o CCE determina quais recursos do Kubernetes podem ser acessados pelo kubectl. As permissões registradas em um arquivo kubeconfig.json variam de usuário para usuário. As permissões que um usuário possui estão listadas em **Tabela 15-1**.

Usuários federados

O IAM fornece a função de provedor de identidade para implementar a autenticação de identidade federada com base em SAML (Security Assertion Markup Language) ou OpenID Connect. Essa função permite que os usuários em seu sistema de gerenciamento acessem a plataforma em nuvem por meio de logon único (SSO).

Os usuários que fazem logon por meio da autenticação de identidade federada são chamados de usuários federados. Os usuários federados são equivalentes aos usuários do IAM.

Preste atenção ao seguinte para que os usuários federados usem o CCE:

- Quando um usuário cria um cluster do CCE, a permissão cluster-admin é concedida ao usuário por padrão. O ID de usuário de um usuário federado é alterado a cada logon e logout. Portanto, o usuário é exibido como excluído na página **Permissions** do console do CCE. Não exclua manualmente a permissão, caso contrário, a autenticação falhará. Nesse caso, é aconselhável conceder a permissão cluster-admin a um grupo de usuários no CCE e adicionar usuários federados ao grupo de usuários.
- Os usuários federados não podem criar chaves de acesso permanentes (AKs/SKs). Em cenários em que as AKs/SKs são necessários (por exemplo, ao criar PVs/PVCs relacionados ao OBS), somente você ou um usuário do IAM podem criar as AKs/SK e compartilhá-los com os usuários federados. Uma chave de acesso contém as permissões concedidas a um usuário, portanto, é recomendável que o usuário federado solicite que um usuário do IAM no mesmo grupo crie uma chave de acesso.

Ações suportadas no IAM

O IAM fornece políticas definidas pelo sistema que podem ser usadas diretamente no IAM. Você também pode criar políticas personalizadas para complementar políticas definidas pelo sistema para um controle de acesso mais refinado. As operações suportadas pelas políticas são específicas das APIs. Seguem-se conceitos comuns relacionados com as políticas:

- Permissão: uma declaração em uma política que permite ou nega certas operações.
- APIs: as APIs REST que podem ser chamadas em uma política personalizada.
- Ações: adicionadas a uma política personalizada para controlar permissões para operações específicas.
- Ações dependentes: ações das quais uma ação específica depende para ter efeito. Ao atribuir permissões para a ação a um usuário, você também precisa atribuir permissões para as ações dependentes.
- Projeto do IAM/Projeto empresarial: uma ação em uma política personalizada pode ser aplicada a um ou a ambos esses projetos. As políticas que contêm ações que suportam projetos do IAM e da empresa podem ser atribuídas a grupos de usuários e entrar em vigor no IAM e no Enterprise Management. As políticas que contêm apenas ações que suportam projetos do IAM podem ser atribuídas a grupos de usuários e só entram em vigor para o IAM. Essas políticas não terão efeito se forem atribuídas a grupos de usuários no Enterprise Management. Para obter detalhes sobre as diferenças entre o IAM e os projetos empresariais, consulte **Quais são as diferenças entre o IAM e o Enterprise Management?**

> NOTA
>
> A marca de seleção (√) e o símbolo da cruz (x), respectivamente, indicam que uma ação tem efeito ou não para o tipo de projeto correspondente.

O CCE suporta as seguintes ações que podem ser definidas nas políticas personalizadas:

Tabela 15-2 Cluster

Permissão	APIs	Ação	Projeto do IAM	Projeto empresarial
Listar clusters em um projeto especificado	GET /api/v3/projects/{project_id}/clusters	cce:cluster:list	√	√
Obter informações sobre um cluster especificado	GET /api/v3/projects/{project_id}/clusters/{cluster_id}	cce:cluster:get	√	√
Criar um cluster	POST /api/v3/projects/{project_id}/clusters	cce:cluster:create	√	√
Atualizar informações sobre um cluster especificado	PUT /api/v3/projects/{project_id}/clusters/{cluster_id}	cce:cluster:update	√	√
Excluir um cluster	DELETE /api/v3/projects/{project_id}/clusters/{cluster_id}	cce:cluster:delete	√	√
Atualizar um cluster	POST /api/v2/projects/:projectid/clusters/:clusterid/upgrade	cce:cluster:upgrade	√	√
Despertar um cluster	POST /api/v3/projects/{project_id}/clusters/{cluster_id}/operation/awake	cce:cluster:start	√	√
Hibernar um cluster	POST /api/v3/projects/{project_id}/clusters/{cluster_id}/operation/hibernate	cce:cluster:stop	√	√
Alterar as especificações de um cluster	POST /api/v2/projects/{project_id}/clusters/:clusterid/resize	cce:cluster:resize	√	√
Obter um certificado de cluster	POST /api/v3/projects/{project_id}/clusters/{cluster_id}/clustercert	cce:cluster:get	√	√

Tabela 15-3 Nó

Permissão	API	Ação	Projeto do IAM	Projeto empresarial
Obter informações sobre todos os nós em um cluster	GET /api/v3/projects/{project_id}/clusters/{cluster_id}/nodes	cce:node:list	√	√
Obter informações sobre um nó especificado	GET /api/v3/projects/{project_id}/clusters/{cluster_id}/nodes/{node_id}	cce:node:get	√	√
Criar um nó	POST /api/v3/projects/{project_id}/clusters/{cluster_id}/nodes	cce:node:create	√	√
Atualizar informações sobre um nó especificado	PUT /api/v3/projects/{project_id}/clusters/{cluster_id}/nodes/{node_id}	cce:node:update	√	√
Excluir um nó	DELETE /api/v3/projects/{project_id}/clusters/{cluster_id}/nodes/{node_id}	cce:node:delete	√	√

Tabela 15-4 Tarefa

Permissão	API	Ação	Projeto do IAM	Projeto empresarial
Obter informações sobre uma tarefa	GET /api/v3/projects/{project_id}/jobs/{job_id}	cce:job:get	√	√
Listar todas as tarefas	GET /api/v2/projects/{project_id}/jobs	cce:job:list	√	√
Excluir uma ou todas as tarefas	DELETE /api/v2/projects/{project_id}/jobs DELETE /api/v2/projects/{project_id}/jobs/{job_id}	cce:job:delete	√	√

Tabela 15-5 Nodepool

Permissão	API	Ação	Projeto do IAM	Projeto empresarial
Obter informações sobre todos os pools de nós em um cluster especificado	GET /api/v3/projects/{project_id}/clusters/{cluster_id}/nodepools	cce:nodepool:list	√	√
Obter informações sobre um pool de nós	GET /api/v3/projects/{project_id}/clusters/{cluster_id}/nodepools/{nodepool_id}	cce:nodepool:get	√	√
Criar um pool de nós	POST /api/v3/projects/{project_id}/clusters/{cluster_id}/nodepools	cce:nodepool:create	√	√
Atualizar informações sobre um pool de nós	PUT /api/v3/projects/{project_id}/clusters/{cluster_id}/nodepools/{nodepool_id}	cce:nodepool:update	√	√
Excluir um pool de nós	DELETE /api/v3/projects/{project_id}/clusters/{cluster_id}/nodepools/{nodepool_id}	cce:nodepool:delete	√	√

Tabela 15-6 Gráfico

Permissão	API	Ação	Projeto do IAM	Projeto empresarial
Atualizar um gráfico	PUT /v2/charts/{id}	cce:chart:update	√	x
Carregar um gráfico	POST /v2/charts	cce:chart:upload	√	x
Listar todos os gráficos	GET /v2/charts	cce:chart:list	√	x
Obter informações sobre gráfico	GET /v2/charts/{id}	cce:chart:get	√	x
Excluir um gráfico	DELETE /v2/charts/{id}	cce:chart:delete	√	x

Tabela 15-7 Release

Permissão	API	Ação	Projeto do IAM	Projeto empresarial
Atualizar um release	PUT /v2/releases/{name}	cce:release:update	√	√
Listar todos os releases	GET /v2/releases	cce:release:list	√	√
Criar uma release	POST /v2/releases	cce:release:create	√	√
Obter informações sobre um release	GET /v2/releases/{name}	cce:release:get	√	√
Excluir uma release	DELETE /v2/releases/{name}	cce:release:delete	√	√

Tabela 15-8 Armazenamento

Permissão	API	Ação	Projeto do IAM	Projeto empresarial
Criar uma PersistentVolumeClaim	POST /api/v1/namespaces/{namespace}/cloudpersistentvolume-claims	cce:storage:create	√	√
Excluir uma PersistentVolumeClaim	DELETE /api/v1/namespaces/{namespace}/cloudpersistentvolume-claims/{name}	cce:storage:delete	√	√
Listar todos os volumes	GET /storage/api/v1/namespaces/{namespace}/listvolumes	cce:storage:list	√	√

Tabela 15-9 Complemento

Permissão	API	Ação	Projeto do IAM	Projeto empresarial
Criar uma instância de complemento	POST /api/v3/addons	cce:addonInstance:create	√	√

Permissão	API	Ação	Projeto do IAM	Projeto empresarial
Obter informações sobre uma instância de complemento	GET /api/v3/addons/{id}?cluster_id={cluster_id}	cce:addonInstance:get	√	√
Listar todas as instâncias de complemento	GET /api/v3/addons?cluster_id={cluster_id}	cce:addonInstance:list	√	√
Excluir uma instância de complemento	DELETE /api/v3/addons/{id}?cluster_id={cluster_id}	cce:addonInstance:delete	√	√
Atualizar uma instância de complemento	PUT /api/v3/addons/{id}	cce:addonInstance:update	√	√

Tabela 15-10 Cota

Permissão	API	Ação	Projeto do IAM	Projeto empresarial
Consultar detalhes da cota	GET /api/v3/projects/{project_id}/quotas	cce:quota:get	√	√

15.2 Permissões de cluster (baseadas no IAM)

As permissões no nível do cluster do CCE são atribuídas com base nas **políticas do sistema do IAM** e nas **políticas personalizadas**. Você pode usar grupos de usuários para atribuir permissões a usuários do IAM.

⚠ CUIDADO

- As permissões de cluster são concedidas aos usuários para operar apenas recursos relacionados a clusters (como clusters e nós). Para operar recursos do Kubernetes, como cargas de trabalho e serviços, você deve receber **permissões de namespace** ao mesmo tempo.
- Ao visualizar um cluster no console do CCE, as informações exibidas dependem das permissões de namespace. Se você não tiver permissões de namespace, não poderá exibir os recursos no cluster. Para mais detalhes, consulte **Dependência de permissão do console do CCE**.

Pré-requisitos

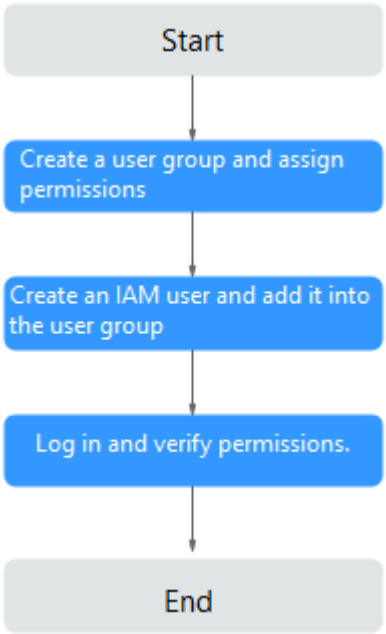
- Antes de conceder permissões a grupos de usuários, familiarize-se com as políticas do sistema listadas em **Permissões**. Para as políticas do sistema de outros serviços, consulte **Permissões do sistema**.
- Um usuário com a função de Security Administrator (por exemplo, sua conta) tem todas as permissões do IAM, exceto a troca de função. Somente esses usuários podem exibir grupos de usuários e suas permissões na página **Permissions** no console do CCE.

Configuração

No console do CCE, ao escolher **Permissions** > **Cluster-Level Permissions** para criar um grupo de usuários, você será direcionado ao console do IAM para concluir o processo. Depois que o grupo de usuários é criado e suas permissões são configuradas, você pode exibir as informações na página de guia **Cluster-Level Permissions**. Esta seção descreve as operações no IAM.

Fluxo do processo

Figura 15-2 Processo de atribuição de permissões do CCE



1. **Crie um grupo de usuários e atribua permissões** a ele.

   Crie um grupo de usuários no console do IAM e atribua permissões do CCE, por exemplo, a política **CCEReadOnlyAccess** ao grupo.

   NOTA

   O CCE é implementado por região. No console do IAM, selecione **Region-specific projects** ao atribuir permissões do CCE.

2. **Crie um usuário e adicione-o a um grupo de usuários.**

   Crie um usuário no console do IAM e adicione o usuário ao grupo criado em **1**.

3. **Fazer logon** e verificar as permissões.

   Efetue logon no console de gerenciamento como o usuário que você criou e verifique se o usuário tem as permissões atribuídas.

   – Faça logon no console de gerenciamento, alterne para o console do CCE e compre um cluster. Se você não conseguir fazê-lo (supondo que apenas a permissão **CCEReadOnlyAccess** é atribuída), a política **CCEReadOnlyAccess** entra em vigor.

   – Alterne para o console de qualquer outro serviço. Se aparecer uma mensagem indicando que você não tem as permissões necessárias para acessar o serviço, a política **CCEReadOnlyAccess** entrará em vigor.

Funções definidas pelo sistema

As funções são um tipo de mecanismo de autorização grosseiro que define permissões de nível de serviço com base nas responsabilidades do usuário. Apenas um número limitado de funções em nível de serviço está disponível para autorização. As funções não são ideais para autorização refinada e acesso de privilégio mínimo.

A função de sistema predefinida para CCE no IAM é **CCEAdministrator**. Ao atribuir essa função a um grupo de usuários, você também deve selecionar outras funções e políticas das quais essa função depende, como **Tenant Guest**, **Server Administrator**, **ELB Administrator**, **OBS Administrator**, **SFS Administrator**, **SWR Admin** e **APM FullAccess**. Para obter mais informações sobre dependências, consulte **Permissões do sistema**.

Políticas definidas pelo sistema

As políticas de sistema predefinidas para CCE no IAM são **CCEFullAccess** e **CCEReadOnlyAccess**.

- **CCE FullAccess**: permissões de operação comuns em recursos de cluster do CCE, excluindo as permissões em nível de namespace para os clusters (com RBAC do Kubernetes ativado) e as operações de administrador privilegiado, como configuração de agência e geração de certificados de cluster
- **CCE ReadOnlyAccess**: permissões para visualizar recursos de cluster do CCE, excluindo as permissões de nível de namespace dos clusters (com RBAC do Kubernetes ativado)

Tabela 15-11 Permissões atribuídas por CCEFullAccess

Ação	Ação específica	Descrição
cce:*:*	cce:cluster:create	Criar um cluster.
	cce:cluster:delete	Excluir um cluster.
	cce:cluster:update	Atualizar um cluster. Por exemplo, atualize os parâmetros de agendamento do nó do cluster e forneça suporte RBAC aos clusters.
	cce:cluster:upgrade	Atualizar um cluster.
	cce:cluster:start	Despertar um cluster.

Ação	Ação específica	Descrição
	cce:cluster:stop	Hibernar um cluster.
	cce:cluster:list	Listar todos os clusters.
	cce:cluster:get	Consultar detalhes do cluster.
	cce:node:create	Adicionar um nó.
	cce:node:delete	Excluir um ou mais nós.
	cce:node:update	Atualizar um nó. Por exemplo, atualize o nome do nó.
	cce:node:get	Consultar detalhes do nó.
	cce:node:list	Listar todos os nós.
	cce:nodepool:create	Criar um pool de nós.
	cce:nodepool:delete	Excluir um pool de nós.
	cce:nodepool:update	Atualizar informações sobre um pool de nós.
	cce:nodepool:get	Obter informações sobre um pool de nós.
	cce:nodepool:list	Listar todos os pools de nós em um cluster.
	cce:release:create	Criar um release.
	cce:release:delete	Excluir um release.
	cce:release:update	Atualizar um release.
	cce:job:list	Listar todas as tarefas de cluster.
	cce:job:delete	Excluir uma ou mais tarefas de cluster.
	cce:job:get	Consultar uma tarefa de cluster especificada.
	cce:storage:create	Criar um volume de armazenamento.
	cce:storage:delete	Excluir um volume de armazenamento.
	cce:storage:list	Listar todos os volumes.
	cce:addonInstance:create	Criar uma instância de complemento.
	cce:addonInstance:delete	Excluir uma instância de complemento.
	cce:addonInstance:update	Atualizar uma instância de complemento.
	cce:addonInstance:get	Consultar detalhes da instância do complemento.

Ação	Ação específica	Descrição
	cce:addonTemplate:get	Consultar detalhes do modelo de complemento.
	cce:addonInstance:list	Listar todas as instâncias do complemento.
	cce:addonTemplate:list	Listar todos os modelos de complementos.
	cce:chart:list	Listar todos os gráficos.
	cce:chart:delete	Eliminar um gráfico.
	cce:chart:update	Atualizar um gráfico.
	cce:chart:upload	Fazer upload de um gráfico.
	cce:chart:get	Obter informações sobre um gráfico.
	cce:release:get	Obter informações sobre um release.
	cce:release:list	Listar todos os releases.
	cce:userAuthorization:get	Obter autorização de usuário do CCE.
	cce:userAuthorization:create	Criar autorização de usuário do CCE.
ecs:*:*	Nenhuma	Executar todas as operações em um Elastic Cloud Server (ECS).
evs:*:*	-	Executar todas as operações em Elastic Volume Service (EVS). Os discos EVS podem ser anexados a servidores em nuvem e dimensionados para uma capacidade maior sempre que necessário.
vpc:*:*	Nenhuma	Executar todas as operações em Virtual Private Cloud (VPC), incluindo ELBs. Um cluster deve ser executado em uma VPC. Ao criar um namespace, crie ou associe uma VPC para o namespace para que todos os contêineres no namespace sejam executados na VPC.
sfs:*:get*	Nenhuma	Visualizar detalhes do recurso do Scalable File Service (SFS).
sfs:shares:ShareAction	Nenhuma	Compartilhar recursos do SFS para dimensionamento.
aom:*:get	Nenhuma	Exibir detalhes do recurso do Application Operations Management (AOM).
aom:*:list	Nenhuma	Listar recursos do AOM.

Ação	Ação específica	Descrição
aom:autoScalingRule:*	Nenhuma	Executar todas as operações nas regras de escala automática do AOM.
apm:icmgr:*	Nenhuma	Executar operações no ICAgent no Application Performance Management (APM).
lts:*:*	Nenhuma	Executar todas as operações no Log Tank Service (LTS).

Tabela 15-12 Permissões atribuídas por CCEReadOnlyAccess

Ação	Ação específica	Descrição
cce:*:get	cce:cluster:get	Consultar detalhes do cluster.
	cce:node:get	Consultar detalhes do nó.
	cce:job:get	Consultar uma tarefa de cluster especificada.
	cce:addonInstance:get	Consultar detalhes da instância do complemento.
	cce:addonTemplate:get	Consultar detalhes do modelo de complemento.
	cce:chart:get	Obter informações sobre um gráfico.
	cce:nodepool:get	Obter informações sobre um pool de nós.
	cce:release:get	Obter informações sobre um release.
	cce:userAuthorization:get	Obter autorização de usuário do CCE.
cce:*:list	cce:cluster:list	Listar todos os clusters.
	cce:node:list	Listar todos os nós.
	cce:job:list	Listar todas as tarefas de cluster.
	cce:addonInstance:list	Listar todas as instâncias do complemento.
	cce:addonTemplate:list	Listar todos os modelos de complementos.
	cce:chart:list	Listar todos os gráficos.
	cce:nodepool:list	Listar todos os pools de nós em um cluster.
	cce:release:list	Listar todos os releases.
	cce:storage:list	Listar todos os volumes.

Ação	Ação específica	Descrição
cce:kubernetes:*	Nenhuma	Executar operações em todos os recursos do Kubernetes. Para obter detalhes, consulte **Permissões de namespace**.
ecs:*:get	Nenhuma	Exibir detalhes sobre todos os recursos do ECS. Um ECS com vários discos EVS é um nó de cluster no CCE.
ecs:*:list	Nenhuma	Listar todos os recursos do ECS.
bms:*:get*	Nenhuma	Exibir detalhes sobre todos os recursos do BMS.
bms:*:list	Nenhuma	Listar todos os recursos do BMS.
evs:*:get	Nenhuma	Exibir detalhes de todos os recursos de disco EVS. Os discos EVS podem ser anexados a servidores em nuvem e dimensionados para uma capacidade maior sempre que necessário.
evs:*:list	Nenhuma	Listar todos os recursos do EVS.
evs:*:count	Nenhuma	Nenhuma
vpc:*:get	Nenhuma	Exibir detalhes de todos os recursos da VPC (incluindo ELBs). Um cluster deve ser executado em uma VPC. Ao criar um namespace, crie ou associe uma VPC para o namespace para que todos os contêineres no namespace sejam executados na VPC.
vpc:*:list	Nenhuma	Listar todos os recursos da VPC (incluindo ELBs).
sfs:*:get*	Nenhuma	Exibir detalhes do recurso do SFS.
sfs:shares:ShareAction	Nenhuma	Compartilhar recursos do SFS para dimensionamento.
aom:*:get	Nenhuma	Exibir detalhes do recurso do AOM.
aom:*:list	Nenhuma	Listar todos os recursos do AOM.
aom:autoScalingRule:*	Nenhuma	Executar todas as operações nas regras de escala automática do AOM.
lts:*:get	Nenhuma	Exibir detalhes sobre todos os recursos do LTS.
lts:*:list	Nenhuma	Listar todos os recursos do LTS.

Políticas personalizadas

Políticas personalizadas podem ser criadas como um suplemento para as políticas definidas pelo sistema do CCE. Para as ações que podem ser adicionadas a políticas personalizadas, consulte **Políticas de permissões e ações suportadas**.

Você pode criar políticas personalizadas de uma das seguintes maneiras:

- Editor visual: selecione serviços em nuvem, ações, recursos e condições de solicitação. Isso não requer conhecimento de sintaxe de política.
- JSON: edite políticas de JSON do rascunho ou com base em uma política existente.

Para obter detalhes, consulte **Criação de uma política personalizada**. Esta seção fornece exemplos de políticas do CCE personalizadas comuns.

Exemplos de políticas personalizadas:

- Exemplo 1: criar um cluster chamado **test**

```
{
    "Version": "1.1",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "cce:cluster:create"
            ]
        }
    ]
}
```

- Exemplo 2: negar exclusão de nó

  Uma política com apenas permissões "Deny" deve ser usada em conjunto com outras políticas para entrar em vigor. Se as permissões atribuídas a um usuário contiverem "Allow" e "Deny", as permissões "Deny" terão precedência sobre as permissões "Allow".

  O método a seguir pode ser usado se você precisar atribuir permissões da política **CCEFullAccess** a um usuário, mas quiser impedir que o usuário exclua nós. (**cce:node:delete**). Crie uma política personalizada para negar a exclusão de nós e anexe ambas as políticas ao grupo ao qual o usuário pertence. Em seguida, o usuário pode executar todas as operações no CCE, exceto a exclusão de nós. O seguinte é um exemplo de uma política de negação:

```
{
    "Version": "1.1",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "cce:node:delete"
            ]
        }
    ]
}
```

- Exemplo 3: definir as permissões para vários serviços em uma política

  Uma política personalizada pode conter as ações de vários serviços que são do tipo global ou de nível de projeto. Veja a seguir um exemplo de política que contém ações de vários serviços:

```
{
    "Version": "1.1",
    "Statement": [
        {
```



```
            "Action": [
                "ecs:cloudServers:resize",
                "ecs:cloudServers:delete",
                "ecs:cloudServers:delete",
                "ims:images:list",
                "ims:serverImages:create"
            ],
            "Effect": "Allow"
        }
    ]
}
```

Permissões de cluster do CCE e projetos empresariais

O CCE oferece suporte ao gerenciamento de recursos e à alocação de permissões por cluster e projeto empresariais.

Note que:

- Os projetos do IAM são baseados no isolamento físico de recursos, enquanto os projetos empresariais fornecem grupos lógicos globais de recursos, que atendem melhor aos requisitos reais das empresas. Além disso, as políticas do IAM podem ser gerenciadas com base em projetos empresariais. Portanto, é aconselhável usar projetos empresariais para gerenciamento de permissões. Para obter detalhes, consulte **Criação de um projeto empresarial**.
- Quando há projetos do IAM e projetos empresariais, o IAM corresponde preferencialmente às políticas de projeto do IAM.
- Ao criar um cluster ou nó usando recursos de nuvem comprados, verifique se os usuários do IAM receberam as permissões necessárias no projeto empresarial para usar esses recursos. Caso contrário, o cluster ou nó pode falhar ao ser criado.
- Se um recurso não suportar projetos empresariais, as permissões concedidas ao recurso não terão efeito.

Tipo de recurso	Nome do recurso	Descrição
Apoiar projetos empresariais	cluster	Cluster
	node	Nó
	nodepool	Pool de nós
	job	Tarefa
	tag	Rótulo do cluster
	addonInstance	Instância de complemento
	release	Versão de Helm
	storage	Armazenamento
Não apoiar projetos empresariais	quota	Cota do cluster
	chart	Gráfico
	addonTemplate	Modelo de complemento

Permissões do cluster do CCE e RBAC do IAM

O CCE é compatível com as funções do sistema do IAM para gerenciamento de permissões. Recomendamos que você use políticas refinadas fornecidas pelo IAM para simplificar o gerenciamento de permissões.

O CCE suporta as seguintes funções:

- Funções básicas do IAM:
  - te_admin (Tenant Administrator): os usuários com essa função podem chamar todas as APIs de todos os serviços, exceto o IAM.
  - readonly (Tenant Guest): os usuários com essa função podem chamar APIs com as permissões de somente leitura de todos os serviços, exceto o IAM.
- Função de administrador do CCE personalizada: CCE Administrator
- As APIs do CCE são compatíveis com três funções de sistema legadas de **ServiceStage** (SvcStg Administrator, SvcStg Developer e SvcStg Operator). Atualmente, o CCE e ServiceStage se adaptaram totalmente às políticas refinadas do IAM para gerenciamento de permissões. Portanto, não é aconselhável usar essas funções herdadas para o gerenciamento de permissões. Essas funções ServiceStage são descritas a seguir:
  - SvcStg Administrator: essa função tem as mesmas permissões que a função CCE Administrator, exceto que os usuários com essa função não têm as permissões no nível do namespace por padrão (RBAC do Kubernetes).
  - SvcStg Developer: essa função tem as mesmas permissões que a função CCE Administrator, exceto que o usuário com essa função não tem a permissão de nível de namespace por padrão (RBAC do Kubernetes).
  - SvcStg Operator: essa função tem as permissões somente leitura no CCE, excluindo as permissões no nível do namespace dos clusters por padrão.

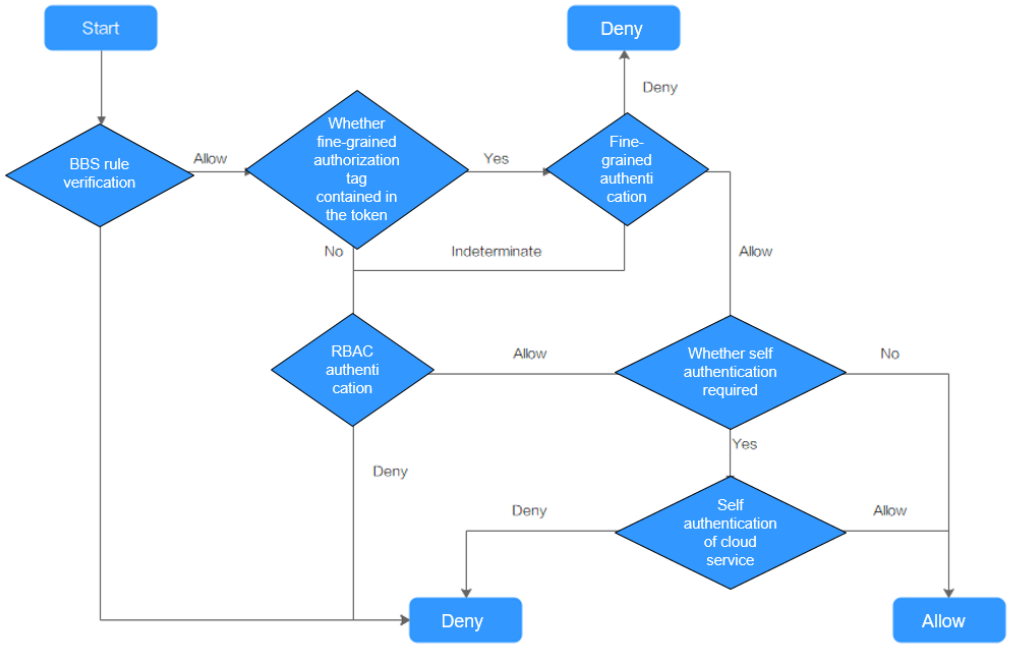
  Para obter detalhes, consulte **Permissões.**.

  NOTA

  - Tenant Administrator e Tenant Guest são funções especiais do sistema do IAM. Depois que qualquer sistema ou política personalizada for configurada, Tenant Administrator e Tenant Guest entrarão em vigor como políticas do sistema para obter compatibilidade com cenários RBAC e ABAC do IAM.
  - Se um usuário tiver a função de sistema Tenant Administrator ou CCE Administrator, ele terá as permissões de administrador de cluster no RBAC do Kubernetes e as permissões não poderão ser removidas após a criação do cluster.

    Se o usuário for o criador do cluster, as permissões cluster-admin no RBAC do Kubernetes serão concedidas ao usuário por padrão. As permissões podem ser removidas manualmente após a criação do cluster.
    - Método 1: escolha **Permissions Management** > **Namespace-Level Permissions** > **Delete** na mesma função que cluster-creator no console do CCE.
    - Método 2: exclua **ClusterRoleBinding: cluster-creator** por meio da API ou kubectl.

Quando as políticas de RBAC e IAM coexistem, a lógica de autenticação de back-end para APIs abertas ou operações de console no CCE é a seguinte:



⚠ CUIDADO

Certas APIs do CCE envolvem permissões em nível de namespace ou operações de chave e, portanto, exigem permissões especiais:

Usar clusterCert para obter o cluster kubeconfig: cceadm/teadmin

15.3 Permissões de namespace (com base no RBAC do Kubernetes)

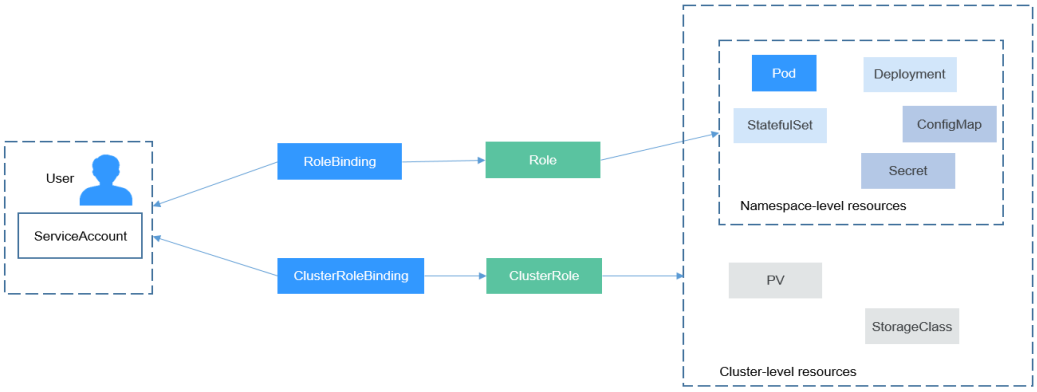
Permissões de namespace (com base no RBAC do Kubernetes)

Você pode regular o acesso de usuários ou grupos de usuários aos recursos do Kubernetes em um único namespace com base em suas funções do RBAC do Kubernetes. A API do RBAC declara quatro tipos de objetos do Kubernetes: Role, ClusterRole, RoleBinding e ClusterRoleBinding, que são descritos a seguir:

- Role: define um conjunto de regras para acessar recursos do Kubernetes em um namespace.
- RoleBinding: define a relação entre usuários e funções.
- ClusterRole: define um conjunto de regras para acessar recursos do Kubernetes em um cluster (incluindo todos os namespaces).
- ClusterRoleBinding: define a relação entre usuários e funções de cluster.

Role e ClusterRole especificam ações que podem ser executadas em recursos específicos. RoleBinding e ClusterRoleBinding vinculam funções a usuários, grupos de usuários ou ServiceAccounts específicos. Ilustração:

Figura 15-3 Vinculação de função



No console do CCE, você pode atribuir permissões a um usuário ou grupo de usuários para acessar recursos em um ou vários namespaces. Por padrão, o console do CCE fornece os seguintes ClusterRoles:

- view (somente leitura): permissão somente leitura na maioria dos recursos em todos os namespaces ou selecionados.
- editar (desenvolvimento): permissões de leitura e gravação na maioria dos recursos em todos os namespaces ou selecionados. Se esse ClusterRole estiver configurado para todos os namespaces, sua capacidade será a mesma da permissão O&M.
- admin (O&M): permissões de leitura e gravação na maioria dos recursos em todos os namespaces e permissão somente leitura em nós, volumes de armazenamento, namespaces e gerenciamento de cotas.
- cluster-admin (administrador): permissões de leitura e gravação em todos os recursos em todos os namespaces.
- drainage-editor: drenar um nó.
- drainage-viewer: visualizar o status de drenagem de nodal, mas não pode drenar um nó.

Permissões de cluster (baseadas no IAM) e permissões de namespace (baseadas no Kubernetes RBAC)

Usuários com diferentes permissões de cluster (atribuídos usando o IAM) têm diferentes permissões de namespace (atribuídos usando o Kubernetes RBAC). **Tabela 15-13** lista as permissões de namespace de diferentes usuários.

Tabela 15-13 Diferenças nas permissões de namespace

Usuário	Clusters da v1.13 e posterior
Usuário com permissões de Tenant Administrator (por exemplo, uma conta)	Todas as permissões de namespace
Usuário do IAM com a função CCE Administrator	Todas as permissões de namespace
Usuário do IAM com a função CCE FullAccess ou CCE ReadOnlyAccess	Requer autorização do Kubernetes RBAC.

Usuário	Clusters da v1.13 e posterior
Usuário do IAM com a função Tenant Guest	Requer autorização do Kubernetes RBAC.

Precauções

- Depois que você cria um cluster, o CCE atribui automaticamente a permissão cluster-admin a você, o que significa que você tem controle total sobre todos os recursos em todos os namespaces do cluster. O ID de um usuário federado é alterado a cada logon e logout. Portanto, o usuário com as permissões é exibido como excluído. Nesse caso, não exclua as permissões. Caso contrário, a autenticação falha. É aconselhável conceder a permissão cluster-admin a um grupo de usuários no CCE e adicionar usuários federados ao grupo de usuários.
- Um usuário com a função Security Administrator tem todas as permissões do IAM, exceto a troca de função. Por exemplo, uma conta no grupo de usuários admin tem essa função por padrão. Somente esses usuários podem atribuir permissões na página **Permissions** no console do CCE.

Configurar permissões de namespace (no console)

Você pode regular o acesso de usuários ou grupos de usuários aos recursos do Kubernetes em um único namespace com base em suas funções do RBAC do Kubernetes.

Passo 1 Efetue logon no console do CCE. No painel de navegação, escolha **Permissions**.

Passo 2 Selecione um cluster para o qual você deseja adicionar permissões na lista suspensa à direita.

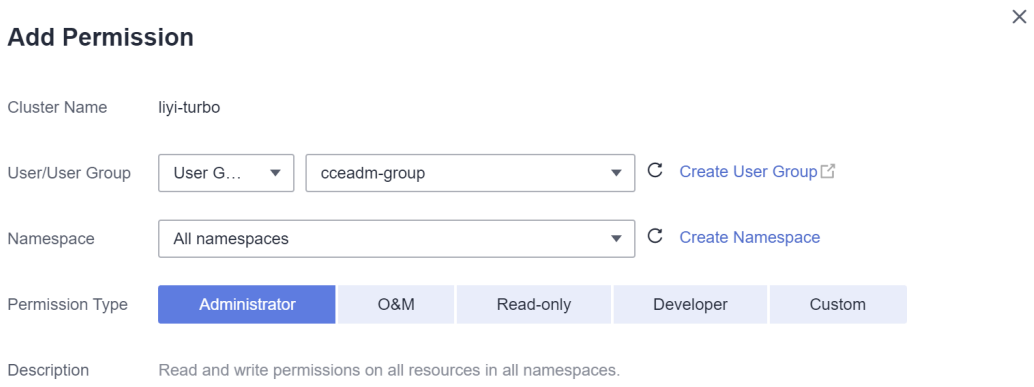
Passo 3 Clique em **Add Permissions** no canto superior direito.

Passo 4 Confirme o nome do cluster e selecione o namespace para o qual atribuir permissões. Por exemplo, selecione **All namespaces**, o usuário de destino ou grupo de usuários e selecione as permissões.

NOTA

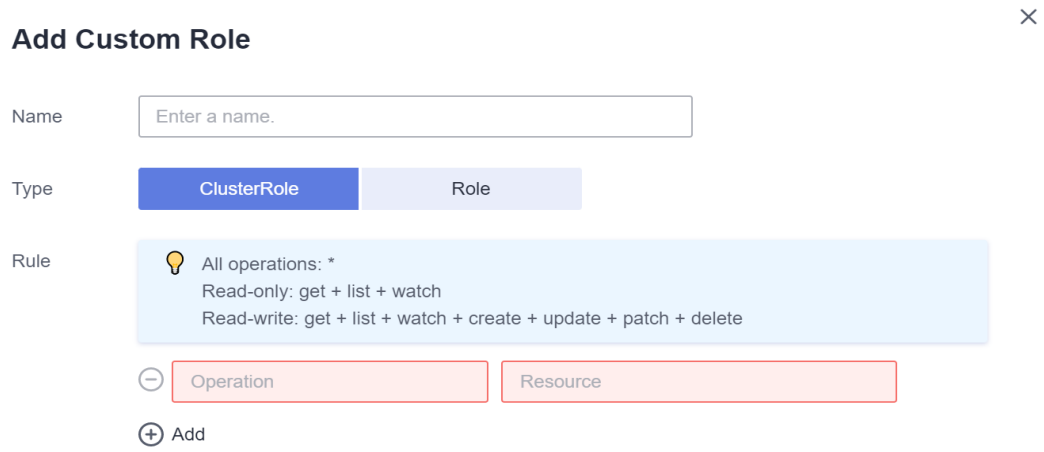
Se você não tiver permissões do IAM, não poderá selecionar usuários ou grupos de usuários ao configurar permissões para outros usuários ou grupos de usuários. Nesse caso, você pode inserir um ID de usuário ou um ID de grupo de usuários.

Figura 15-4 Configurar permissões de namespace



As permissões podem ser personalizadas conforme necessário. Depois de selecionar **Custom** para **Permission Type**, clique em **Add Custom Role** à direita do parâmetro **Custom**. Na caixa de diálogo exibida, insira um nome e selecione uma regra. Depois que a regra personalizada for criada, você poderá selecionar um valor na caixa de listagem suspensa **Custom**.

Figura 15-5 Permissão personalizada



Passo 5 Clique em **OK**.

----Fim

Usar o kubectl para configurar permissões de namespace

NOTA

Quando você acessa um cluster usando kubectl, o CCE usa **kubeconfig.json** gerado no cluster para autenticação. Esse arquivo contém informações do usuário, com base no qual o CCE determina quais recursos do Kubernetes podem ser acessados pelo kubectl. As permissões registradas em um arquivo kubeconfig.json variam de usuário para usuário. As permissões que um usuário possui estão listadas em **Permissões de cluster (baseadas no IAM) e permissões de namespace (baseadas no Kubernetes RBAC)**.

Além de cluster-admin, admin, edit e view, você pode definir Roles e RoleBindings para configurar as permissões para adicionar, excluir, modificar e consultar recursos, como pods, Implementações e Serviços, no namespace.

O procedimento para criar um Role é muito simples. Para ser específico, especifique um namespace e defina regras. As regras no exemplo a seguir são para permitir operações GET e LIST em pods no namespace padrão.

```
kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  namespace: default                              # Namespace
  name: role-example
rules:
- apiGroups: [""]
  resources: ["pods"]                             # The pod can be accessed.
  verbs: ["get", "list"]                          # The GET and LIST operations can
be performed.
```

- **apiGroups** indica o grupo de API ao qual o recurso pertence.
- **resources** indica os recursos que podem ser operados. Pods, Implementações, ConfigMaps e outros recursos do Kubernetes são suportados.

- **verbs** indica as operações que podem ser realizadas. **get** indica a consulta de um objeto específico e **list** indica a listagem de todos os objetos de um determinado tipo. Outras opções de valor incluem **create**, **update** e **delete**.

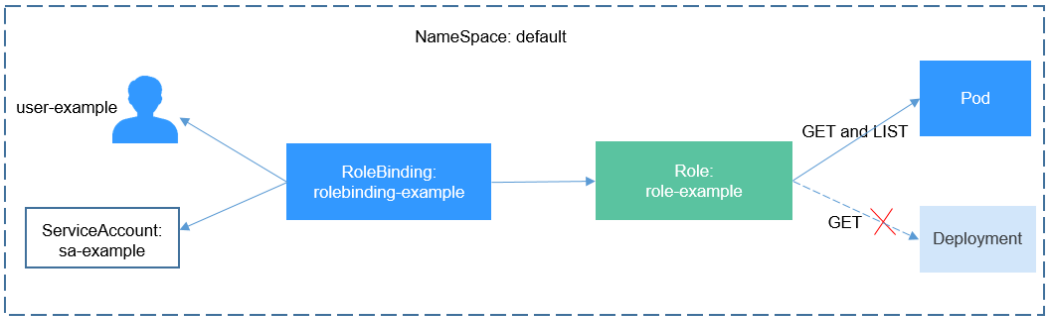
Para obter detalhes, consulte **Uso da autorização do RBAC**.

Depois de criar um Role, você pode vincular o Role a um usuário específico, que é chamado de RoleBinding. O seguinte mostra um exemplo:

```
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: RoleBinding-example
  namespace: default
  annotations:
    CCE.com/IAM: 'true'
roleRef:
  kind: Role
  name: role-example
  apiGroup: rbac.authorization.k8s.io
subjects:
- kind: User
  name: 0c97ac3cb280f4d91fa7c0096739e1f8 # User ID of the user-example
  apiGroup: rbac.authorization.k8s.io
```

A seção **subjects** vincula uma Role a um usuário do IAM para que o usuário do IAM possa obter as permissões definidas na Role, conforme mostrado na figura a seguir.

Figura 15-6 Vinculação de um role a um usuário



Você também pode especificar um grupo de usuários na seção **subjects**. Nesse caso, todos os usuários no grupo de usuários obtêm as permissões definidas na Role.

```
subjects:
- kind: Group
  name: 0c96fad22880f32a3f84c009862af6f7    # User group ID
  apiGroup: rbac.authorization.k8s.io
```

Use o exemplo de usuário do usuário do IAM para se conectar ao cluster e obter as informações do pod. A seguir, um exemplo das informações do pod retornado.

```
# kubectl get pod
NAME                                  READY   STATUS    RESTARTS   AGE
deployment-389584-2-6f6bd4c574-2n9rk  1/1     Running   0          4d7h
deployment-389584-2-6f6bd4c574-7s5qw  1/1     Running   0          4d7h
deployment-3895841-746b97b455-86g77   1/1     Running   0          4d7h
deployment-3895841-746b97b455-twvpn   1/1     Running   0          4d7h
nginx-658dff48ff-7rkph                1/1     Running   0          4d9h
nginx-658dff48ff-njdhj                1/1     Running   0          4d9h
# kubectl get pod nginx-658dff48ff-7rkph
NAME                     READY   STATUS    RESTARTS   AGE
nginx-658dff48ff-7rkph   1/1     Running   0          4d9h
```

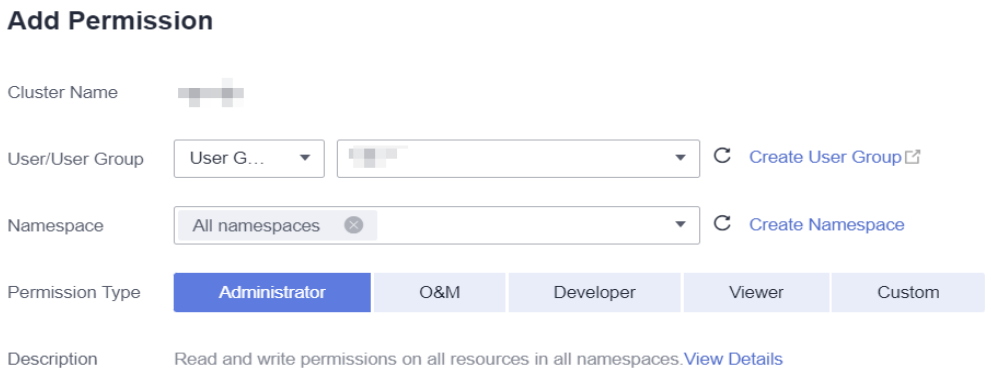

Tente consultar Implementações e Serviços no namespace. A saída mostra que **user-example** não tem as permissões necessárias. Tente consultar os pods no kube-system do namespace. A saída mostra que **user-example** também não tem as permissões necessárias. Isso indica que o usuário do IAM **user-example** tem apenas as permissões GET e LIST Pod no namespace padrão, que é o mesmo que o esperado.

```
# kubectl get deploy
Error from server (Forbidden): deployments.apps is forbidden: User
"0c97ac3cb280f4d91fa7c0096739e1f8" cannot list resource "deployments" in API
group "apps" in the namespace "default"
# kubectl get svc
Error from server (Forbidden): services is forbidden: User
"0c97ac3cb280f4d91fa7c0096739e1f8" cannot list resource "services" in API group
"" in the namespace "default"
# kubectl get pod --namespace=kube-system
Error from server (Forbidden): pods is forbidden: User
"0c97ac3cb280f4d91fa7c0096739e1f8" cannot list resource "pods" in API group "" in
the namespace "kube-system"
```

Exemplo: atribuir permissões de administrador de cluster (cluster-admin)

Você pode usar a função cluster-admin para atribuir todas as permissões em um cluster. Esta função contém as permissões para todos os recursos de cluster.

Figura 15-7 Atribuir permissões de administrador de cluster (cluster-admin)



No exemplo a seguir, um ClusterRoleBinding foi criado e vincula a função cluster-admin ao grupo de usuários **cce-role-group**.

```
# kubectl get clusterrolebinding
NAME
ROLE                              AGE
clusterrole_cluster-admin_group0c96fad22880f32a3f84c009862af6f7   ClusterRole/
cluster-admin      61s

# kubectl get clusterrolebinding clusterrole_cluster-
admin_group0c96fad22880f32a3f84c009862af6f7 -oyaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  annotations:
    CCE.com/IAM: "true"
  creationTimestamp: "2021-06-23T09:15:22Z"
  name: clusterrole_cluster-admin_group0c96fad22880f32a3f84c009862af6f7
  resourceVersion: "36659058"
  selfLink: /apis/rbac.authorization.k8s.io/v1/clusterrolebindings/
clusterrole_cluster-admin_group0c96fad22880f32a3f84c009862af6f7
  uid: d6cd43e9-b4ca-4b56-bc52-e36346fc1320
```



```
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: Group
  name: 0c96fad22880f32a3f84c009862af6f7
```

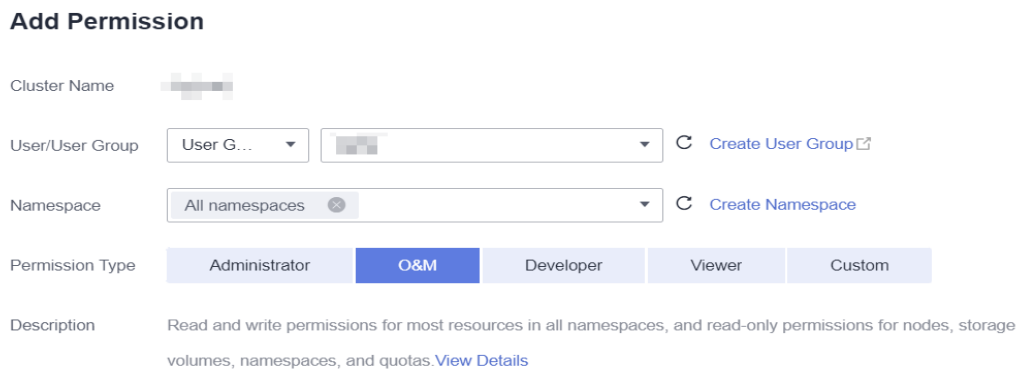
Conecte-se ao cluster como um usuário autorizado. Se os PVs e StorageClasses puderem ser consultados, a configuração de permissão entrará em vigor.

```
# kubectl get pv
No resources found
# kubectl get sc
NAME                PROVISIONER                     RECLAIMPOLICY
VOLUMEBINDINGMODE      ALLOWVOLUMEEXPANSION   AGE
csi-disk            everest-csi-provisioner         Delete
Immediate              true                    75d
csi-disk-topology   everest-csi-provisioner         Delete
WaitForFirstConsumer   true                    75d
csi-nas             everest-csi-provisioner         Delete
Immediate              true                    75d
csi-obs             everest-csi-provisioner         Delete
Immediate              false                   75d
csi-sfsturbo        everest-csi-provisioner         Delete
Immediate              true                    75d
```

Exemplo: atribuir permissões de O&M de namespace (admin)

A função admin tem as permissões de leitura e gravação na maioria dos recursos de namespace. Você pode conceder a permissão admin em todos os namespaces para um usuário ou grupo de usuários.

Figura 15-8 Atribuir permissões de O&M em todos os namespaces (admin)



No exemplo a seguir, um RoleBinding foi criado e vincula a função admin ao grupo de usuários **cce-role-group**.

```
# kubectl get rolebinding
NAME                                                            ROLE                AGE
clusterrole_admin_group0c96fad22880f32a3f84c009862af6f7   ClusterRole/admin   18s
# kubectl get rolebinding clusterrole_admin_group0c96fad22880f32a3f84c009862af6f7
-oyaml
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  annotations:
    CCE.com/IAM: "true"
```



```
  creationTimestamp: "2021-06-24T01:30:08Z"
  name: clusterrole_admin_group0c96fad22880f32a3f84c009862af6f7
  resourceVersion: "36963685"
  selfLink: /apis/rbac.authorization.k8s.io/v1/namespaces/default/rolebindings/
clusterrole_admin_group0c96fad22880f32a3f84c009862af6f7
  uid: 6c6f46a6-8584-47da-83f5-9eef1f7b75d6
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: admin
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: Group
  name: 0c96fad22880f32a3f84c009862af6f7
```

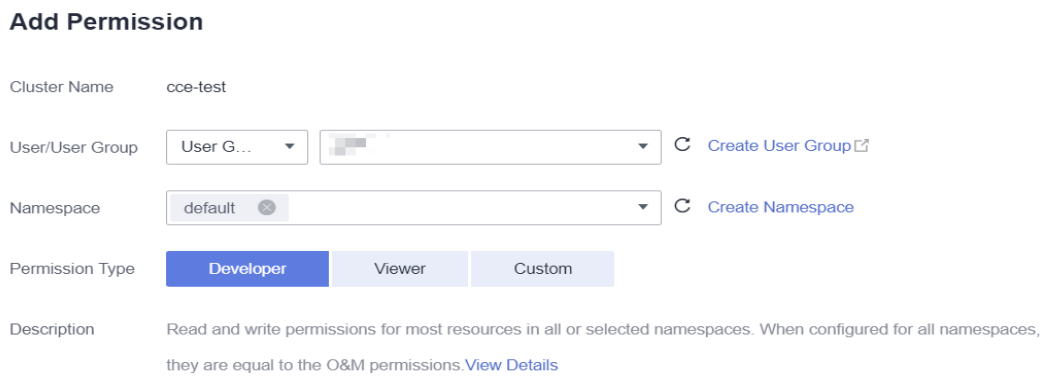
Conecte-se ao cluster como um usuário autorizado. Se os PVs e StorageClasses puderem ser consultados, mas um namespace não puder ser criado, a configuração de permissão entrará em vigor.

```
# kubectl get pv
No resources found
# kubectl get sc
NAME                PROVISIONER                     RECLAIMPOLICY
VOLUMEBINDINGMODE      ALLOWVOLUMEEXPANSION   AGE
csi-disk            everest-csi-provisioner         Delete
Immediate              true                    75d
csi-disk-topology   everest-csi-provisioner         Delete
WaitForFirstConsumer   true                    75d
csi-nas             everest-csi-provisioner         Delete
Immediate              true                    75d
csi-obs             everest-csi-provisioner         Delete
Immediate              false                   75d
csi-sfsturbo        everest-csi-provisioner         Delete
Immediate              true                    75d
# kubectl apply -f namespaces.yaml
Error from server (Forbidden): namespaces is forbidden: User
"0c97ac3cb280f4d91fa7c0096739e1f8" cannot create resource "namespaces" in API
group "" at the cluster scope
```

Exemplo: atribuir permissões de desenvolvedor de namespace (edit)

A função edit tem as permissões de leitura e gravação na maioria dos recursos de namespace. Você pode conceder a permissão edit em todos os namespaces a um usuário ou grupo de usuários.

Figura 15-9 Atribuir permissões de desenvolvedor no namespace padrão (edit)



No exemplo a seguir, um RoleBinding foi criado, a função de edição é vinculada ao grupo de usuários **cce-role-group** e o namespace de destino é o namespace padrão.


```
# kubectl get rolebinding
NAME                                                            ROLE                AGE
clusterrole_admin_group0c96fad22880f32a3f84c009862af6f7   ClusterRole/admin   18s
# kubectl get rolebinding clusterrole_admin_group0c96fad22880f32a3f84c009862af6f7
-oyaml
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  annotations:
    CCE.com/IAM: "true"
  creationTimestamp: "2021-06-24T01:30:08Z"
  name: clusterrole_admin_group0c96fad22880f32a3f84c009862af6f7
  namespace: default
  resourceVersion: "36963685"
  selfLink: /apis/rbac.authorization.k8s.io/v1/namespaces/default/rolebindings/
clusterrole_admin_group0c96fad22880f32a3f84c009862af6f7
  uid: 6c6f46a6-8584-47da-83f5-9eef1f7b75d6
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: edit
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: Group
  name: 0c96fad22880f32a3f84c009862af6f7
```

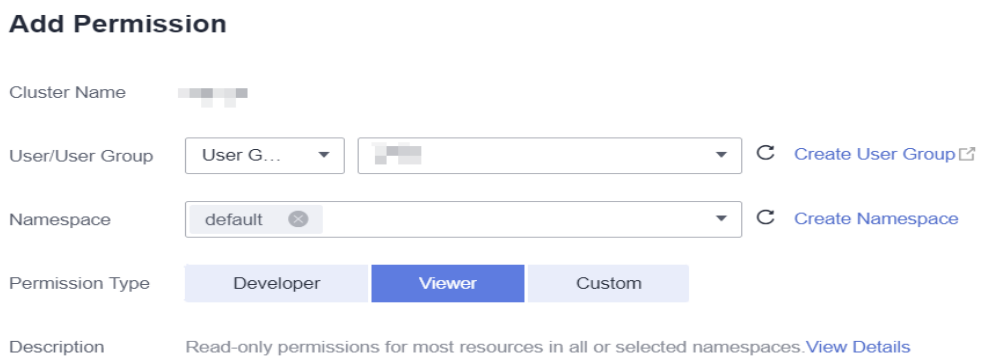
Conecte-se ao cluster como um usuário autorizado. Neste exemplo, você pode criar e obter recursos no namespace padrão, mas não pode consultar recursos no namespace do kube-system ou recursos de cluster.

```
# kubectl get pod
NAME                     READY   STATUS    RESTARTS   AGE
test-568d96f4f8-brdrp    1/1     Running   0          33m
test-568d96f4f8-cgjqp    1/1     Running   0          33m
# kubectl get pod -nkube-system
Error from server (Forbidden): pods is forbidden: User
"0c97ac3cb280f4d91fa7c0096739e1f8" cannot list resource "pods" in API group "" in
the namespace "kube-system"
# kubectl get pv
Error from server (Forbidden): persistentvolumes is forbidden: User
"0c97ac3cb280f4d91fa7c0096739e1f8" cannot list resource "persistentvolumes" in
API group "" at the cluster scope
```

Exemplo: atribuir permissões de namespace somente leitura (view)

A função view tem as permissões somente leitura em um namespace. Você pode atribuir permissões aos usuários para exibir um ou vários namespaces.

Figura 15-10 Atribuir permissões de namespace somente leitura (view)



Na saída de kubectl de exemplo a seguir, um RoleBinding foi criado, a função view é vinculada ao grupo de usuários **cce-role-group** e o namespace de destino é o namespace padrão.

```
# kubectl get rolebinding
NAME                                                    ROLE               AGE
clusterrole_view_group0c96fad22880f32a3f84c009862af6f7   ClusterRole/view   7s

# kubectl get rolebinding clusterrole_view_group0c96fad22880f32a3f84c009862af6f7 -
oyaml
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  annotations:
    CCE.com/IAM: "true"
  creationTimestamp: "2021-06-24T01:36:53Z"
  name: clusterrole_view_group0c96fad22880f32a3f84c009862af6f7
  namespace: default
  resourceVersion: "36965800"
  selfLink: /apis/rbac.authorization.k8s.io/v1/namespaces/default/rolebindings/
clusterrole_view_group0c96fad22880f32a3f84c009862af6f7
  uid: b86e2507-e735-494c-be55-c41a0c4ef0dd
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: view
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: Group
  name: 0c96fad22880f32a3f84c009862af6f7
```

Conecte-se ao cluster como um usuário autorizado. Neste exemplo, você pode consultar recursos no namespace padrão, mas não pode criar recursos.

```
# kubectl get pod
NAME                    READY   STATUS    RESTARTS   AGE
test-568d96f4f8-brdrp   1/1     Running   0          40m
test-568d96f4f8-cgjqp   1/1     Running   0          40m
# kubectl run -i --tty --image tutum/dnsutils dnsutils --restart=Never --
rm /bin/sh
Error from server (Forbidden): pods is forbidden: User
"0c97ac3cb280f4d91fa7c0096739e1f8" cannot create resource "pods" in API group ""
in the namespace "default"
```

Exemplo: atribuir permissões para um objeto de recurso específico do Kubernetes

Você pode atribuir permissões a um objeto de recurso específico do Kubernetes, como pod, Implementação e Serviço. Para mais detalhes, consulte **Usar o kubectl para configurar permissões de namespace**.

15.4 Exemplo: projetar e configurar permissões para usuários em um departamento

Visão geral

O modo convencional de agendamento de tarefas distribuídas está sendo substituído pelo Kubernetes. O CCE permite que você implemente, gerencie e dimensione facilmente aplicações em contêiner na nuvem, fornecendo suporte para você usar o Kubernetes.

Para ajudar os administradores empresariais a gerenciar permissões de recursos em clusters, o CCE fornece políticas de permissão e medidas de gerenciamento multidimensionais e refinadas. As permissões do CCE são descritas a seguir:

- **Cluster-level permissions**: permitindo que um grupo de usuários execute operações em clusters, nós, pools de nós, gráficos e complementos. Essas permissões são atribuídas com base nas políticas do sistema do IAM.
- **Namespace-level permissions**: permitindo que um usuário ou grupo de usuários execute operações em recursos do Kubernetes, como cargas de trabalho, rede, armazenamento e namespaces. Essas permissões são atribuídas com base no Kubernetes RBAC.

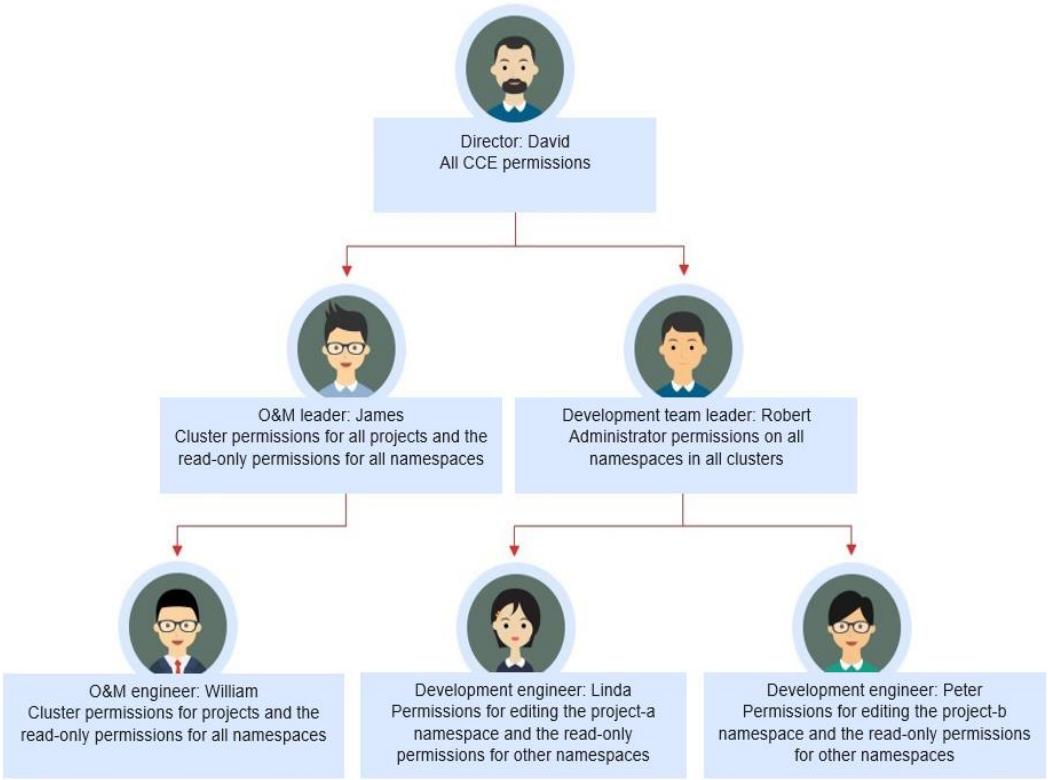
Permissões de cluster e permissões de namespace são independentes uma da outra, mas devem ser usadas juntas. As permissões definidas para um grupo de usuários aplicam-se a todos os usuários no grupo de usuários. Quando várias permissões são adicionadas a um usuário ou grupo de usuários, elas entram em vigor ao mesmo tempo (o conjunto de união é usado).

Design de permissão

A seguir, a empresa X é usada como exemplo.

Geralmente, uma empresa tem vários departamentos ou projetos, e cada departamento tem vários membros. Projete como as permissões devem ser atribuídas a diferentes grupos e projetos e defina um nome de usuário para cada membro para facilitar a configuração de permissões e grupos de usuários subsequentes.

A figura a seguir mostra a estrutura organizacional de um departamento em uma empresa e as permissões a serem atribuídas a cada membro:



Diretor: o David

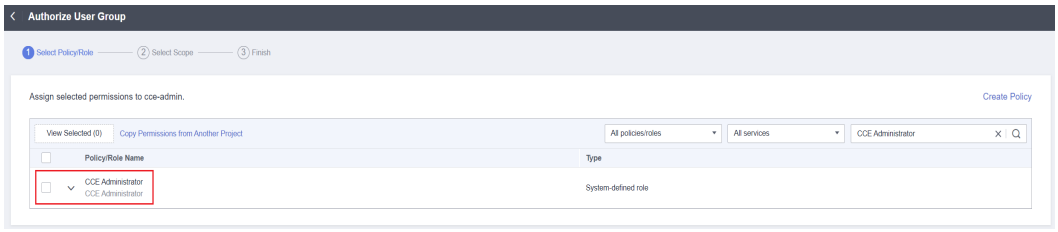
David é diretor de departamento da empresa X. Para atribuir a ele todas as permissões de CCE (permissões de cluster e namespace), crie o grupo de usuários **cce-admin** para David no console do IAM e atribua a função de CCE Administrator.

NOTA

CCE Administrator: esta função tem todas as permissões do CCE. Você não precisa atribuir outras permissões.

CCE FullAccess e CCE ReadOnlyAccess: essas políticas estão relacionadas às permissões de gerenciamento de clusters e configuradas apenas para recursos relacionados a clusters (como clusters e nós). Você também deve configurar permissões de namespace para executar operações em recursos do Kubernetes (como cargas de trabalho e Serviços).

Figura 15-11 Atribuir permissões ao grupo de usuários ao qual David pertence

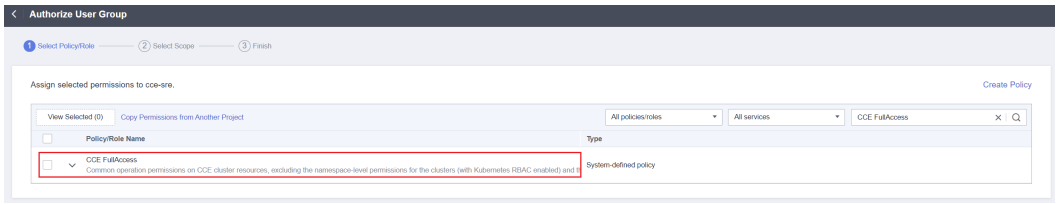


Líder de O&M: James

James é o líder da equipe de O&M do departamento. Ele precisa das permissões de cluster para todos os projetos e das permissões somente leitura para todos os namespaces.

Para atribuir as permissões, crie um grupo de usuários chamado **cce-sre** no console do IAM e adicione James a esse grupo de usuários. Em seguida, atribua CCE FullAccess ao grupo de usuários **cce-sre** para permitir que ele execute operações em clusters em todos os projetos.

Figura 15-12 Atribuir permissões ao grupo de usuários ao qual James pertence



Atribuir permissões somente leitura em todos os clusters e namespaces a todos os líderes de equipe e engenheiros

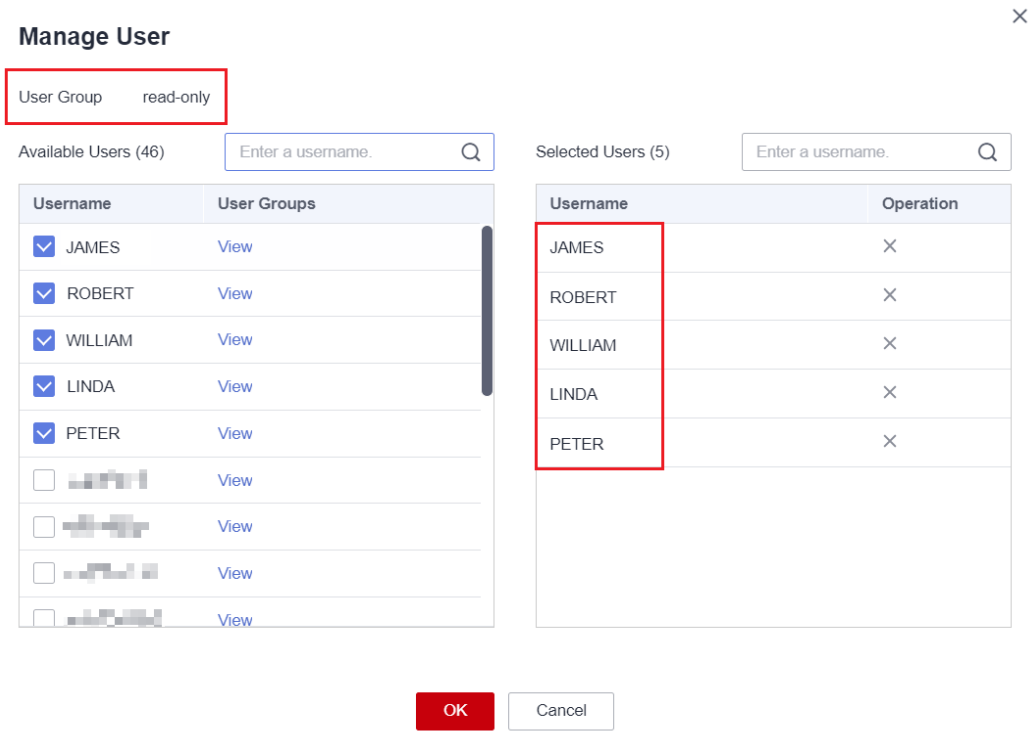
Você pode criar um grupo de usuários somente leitura chamado **read_only** no console do IAM e adicionar usuários ao grupo de usuários.

- Embora os engenheiros de desenvolvimento Linda e Peter não exijam permissões de gerenciamento de cluster, eles ainda precisam visualizar dados no console do CCE. Por conseguinte, a permissão de cluster de somente leitura é necessária.
- Para o engenheiro de O&M William, atribua a ele a permissão somente leitura em clusters nesta etapa.

- O líder da equipe de O&M, James, já tem as permissões de gerenciamento em todos os clusters. Você pode adicioná-lo ao grupo de usuários **read_only** para atribuir a permissão somente leitura em clusters a ele.

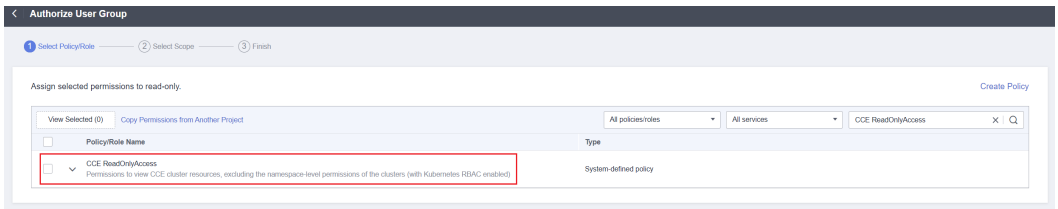
Os usuários James, Robert, William, Linda e Peter são adicionados ao grupo de usuários **read_only**.

Figura 15-13 Adicionar usuários ao grupo de usuários read_only



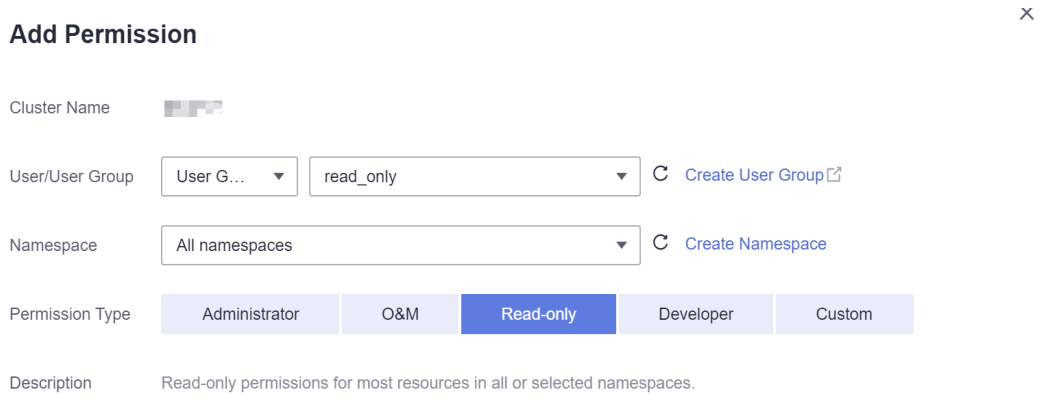
Atribua a permissão somente leitura em clusters ao grupo de usuários **read_only**.

Figura 15-14 Atribuir a permissão somente leitura em clusters ao grupo de usuários



Retorne ao console do CCE e adicione a permissão somente leitura em namespaces ao grupo de usuários **read_only** ao qual os cinco usuários pertencem. Escolha **Permissions** no console do CCE, e atribua a política somente leitura ao grupo de usuários **read_only** para cada conjunto.

Figura 15-15 Atribuir a permissão somente leitura em namespaces ao grupo de usuários



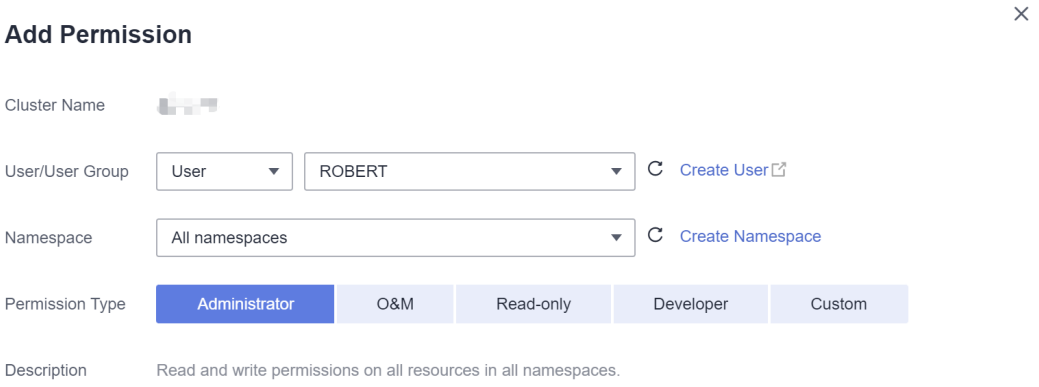
Após a conclusão da configuração, James tem as permissões de gerenciamento de cluster para todos os projetos e as permissões somente leitura em todos os namespaces, e Robert, William, Linda e Peter têm a permissão read-only em todos os clusters e namespaces.

Líder da equipe de desenvolvimento: Robert

Nas etapas anteriores, o Robert recebeu a permissão somente leitura em todos os clusters e namespaces. Agora, atribua as permissões de administrador em todos os namespaces para o Robert.

Portanto, atribua as permissões de administrador em todos os namespaces em todos os clusters para o Robert.

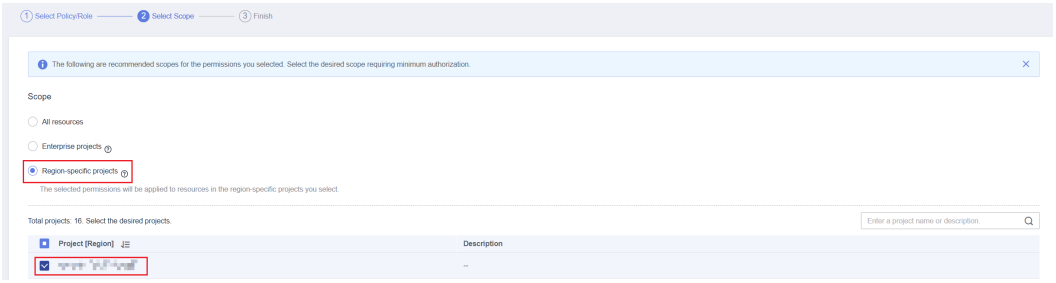
Figura 15-16 Atribuir permissões de administrador em namespaces ao Robert



Engenheiro de O&M: William

Nas etapas anteriores, o William recebeu a permissão somente leitura em todos os clusters e namespaces. Ele também exige as permissões de gerenciamento de cluster em sua região. Portanto, você pode fazer logon no console do IAM, criar um grupo de usuários chamado **cce-sre-b4** e atribuir CCE FullAccess ao William para sua região.

Figura 15-17 Atribuir as permissões de gerenciamento de cluster para a região Beijing4 ao grupo de usuários ao qual WILLIAM pertence

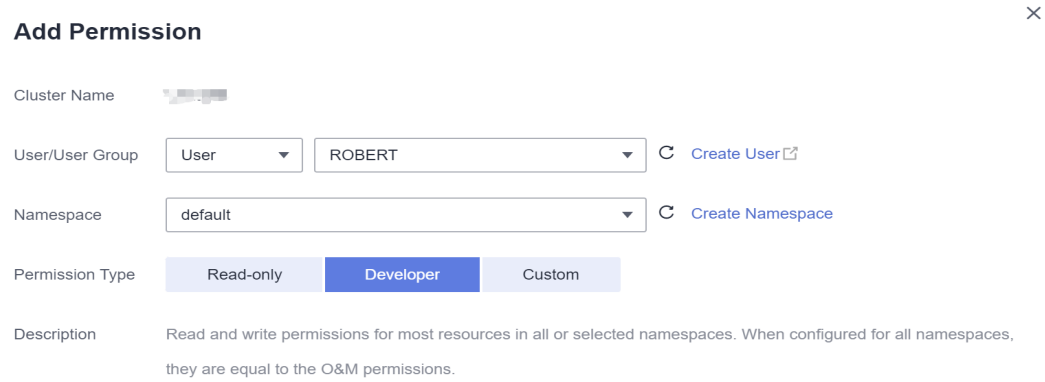


Agora, William tem as permissões de gerenciamento de cluster para sua região e a permissão somente leitura em todos os namespaces.

Engenheiros de desenvolvimento: Linda e Peter

Nas etapas anteriores, Linda e Peter receberam a permissão somente leitura em clusters e namespaces. Portanto, você só precisa atribuir a política de editar a eles.

Figura 15-18 Atribuir a política de edição em namespaces



Até agora, todas as permissões necessárias são atribuídas aos membros do departamento.

15.5 Dependência de permissão do console do CCE

Algumas políticas de permissões de CCE dependem das políticas de outros serviços de nuvem. Para exibir ou usar outros recursos de nuvem no console do CCE, ative o recurso de controle de acesso à política de sistema do IAM e atribua políticas de dependência para os outros serviços de nuvem.

- As políticas de dependência são atribuídas com base na política CCE FullAccess ou CCE ReadOnlyAccess que você configura. Para obter detalhes, consulte **Permissões de cluster (baseadas no IAM)**.
- Somente usuários e grupos de usuários com permissões de namespace podem obter o acesso de exibição aos recursos em clusters.
  - Se um usuário receber acesso de exibição a todos os namespaces de um cluster, ele poderá exibir todos os recursos de namespace (exceto segredos) no cluster. Para exibir segredos no cluster, o usuário deve obter a função **admin** ou **edit** em todos os namespaces do cluster.

- As políticas CustomedHPA e HPA entram em vigor somente após as permissões cluster-admin serem configuradas para o namespace.
- A função **view** em um único namespace permite que os usuários visualizem recursos somente no namespace especificado.

Configuração da política de dependência

Para conceder a um usuário do IAM permissões para visualizar ou usar recursos de outros serviços em nuvem no console do CCE, primeiro você deve conceder as políticas CCE Administrator, CCE FullAccess ou CCE ReadOnlyAccess ao grupo de usuários ao qual o usuário pertence e, em seguida, conceder as políticas de dependência listadas em **Tabela 15-14** ao usuário. Essas políticas de dependência permitirão que o usuário do IAM acesse recursos de outros serviços em nuvem.

NOTA

Os projetos empresariais podem agrupar e gerenciar recursos em diferentes projetos de uma empresa. Os recursos são, assim, isolados. O IAM permite que você implemente uma autorização refinada. É altamente recomendável que você use o IAM para o gerenciamento de permissões.

Se você usar um projeto empresarial para definir permissões para usuários do IAM, as seguintes restrições se aplicam:

- No console do CCE, os projetos empresariais não podem chamar a API usada para obter dados de monitoramento do AOM para monitoramento de cluster. Portanto, os usuários do IAM nesses projetos empresariais não podem consultar dados de monitoramento.
- No console do CCE, os projetos empresariais não podem chamar a API para consultar o par de chaves criado durante a criação do nó. Portanto, os usuários do IAM nesses projetos empresariais não podem usar o modo de logon de par de chaves. Somente o modo de logon por senha é suportado.
- No console do CCE, não há suporte para projetos empresariais durante a criação do modelo. Portanto, os subusuários do projeto empresarial não podem usar o gerenciamento de modelo.
- Depois de atribuir a permissão **CCE FullAccess** a um projeto empresarial, configure a ação **ecs:availabilityZones:list** no console do IAM para que os usuários do projeto empresarial possam criar nós. Caso contrário, o sistema irá avisar que eles não têm a permissão.

O CCE oferece suporte à configuração de permissões refinadas, mas tem as seguintes restrições:

- O AOM não suporta monitoramento em nível de recurso. Depois que as permissões de operação em recursos específicos são configuradas usando a função refinada de gerenciamento de recursos de cluster do IAM, os usuários do IAM podem visualizar as informações de monitoramento de cluster na página **Dashboard** do console do CCE, mas não podem visualizar os dados em métricas não refinadas.

Tabela 15-14 Políticas de dependência

Função do console	Serviços dependentes	Funções ou políticas necessárias
Visão geral de cluster	Application Operations Management (AOM)	• Um usuário do IAM com a permissão CCE Administrator atribuída pode usar essa função somente após a permissão AOM FullAccess ser atribuída. • Os usuários do IAM ReadOnlyAccess, CCE FullAccess ou CCE ReadOnlyAccess atribuídos podem usar diretamente essa função.

Função do console	Serviços dependentes	Funções ou políticas necessárias
Gerenciamento da carga de trabalho	Elastic Load Balance (ELB) Application Performance Management (APM) Application Operations Management (AOM) NAT Gateway Object Storage Service (OBS) Scalable File Service (SFS)	Exceto nos seguintes casos, o usuário não precisa de nenhuma função adicional para criar cargas de trabalho. ● Para criar um Serviço usando o ELB, você deve ter a ELB FullAccess ou ELB Administrator mais as permissões VPC Administrator atribuídas. ● Para usar uma sonda Java, você deve ter as permissõesAOM FullAccess e APM FullAccess atribuídas. ● Para criar um Serviço usando NAT Gateway, você deve ter a permissãoNAT Gateway Administrator atribuída. ● Para usar o OBS, você deve ter a permissão OBS Administrator atribuída globalmente. **NOTA** Devido ao cache, leva cerca de 13 minutos para que a política RBAC entre em vigor após ser concedida a usuários, projetos empresariais e grupos de usuários. Depois que uma política de sistema relacionada ao OBS é concedida, leva cerca de 5 minutos para que a política entre em vigor. ● Para usar o SFS, você deve ter a permissão SFS FullAccess atribuída.
Gerenciamento de cluster	Application Operations Management (AOM) Central de cobrança (BSS)	● Expansão ou aumento automáticos de escala requer a política AOM FullAccess. ● Alterar o modo de faturamento para anual/mensal requer a função BSS Administrator.
Gerenciamento de nó	Elastic Cloud Server (ECS)	Se a permissão atribuída a um usuário do IAM for CCE Administrator, criar ou excluir um nó exigirá a política ECS FullAccess ou ECS Administrator e a política VPC Administrator.

Função do console	Serviços dependentes	Funções ou políticas necessárias
Rede	Elastic Load Balance (ELB) NAT Gateway	Exceto nos seguintes casos, o usuário não precisa de nenhuma função adicional para criar um Serviço. ● Para criar um Serviço usando o ELB, você deve ter a ELB FullAccess ou ELB Administrator mais as permissões VPC Administrator atribuídas. ● Para criar um Serviço usando o NAT Gateway, você deve ter a permissão NAT Administrator atribuída.
Armazenamento do contêiner	Object Storage Service (OBS) Scalable File Service (SFS) SFS Turbo	● Para usar o OBS, você deve ter a permissão OBS Administrator atribuída globalmente. **NOTA** Devido ao cache, leva cerca de 13 minutos para que a política RBAC entre em vigor após ser concedida a usuários, projetos empresariais e grupos de usuários. Depois que uma política de sistema relacionada ao OBS é concedida, leva cerca de 5 minutos para que a política entre em vigor. ● Para usar o SFS, você deve ter a permissão SFS FullAccess atribuída. ● Usar o SFS Turbo requer a função SFS Turbo Admin. A função CCE Administrator é necessária para importar dispositivos de armazenamento.
Gerenciamento de namespace	/	/
Gestão de gráficos	/	As contas de nuvem e os usuários do IAM com a permissão CCE Administrator atribuída podem usar essa função.
Gerenciamento de complementos	/	As contas de nuvem e os usuários do IAM com a permissão Administrador CCE, CCE FullAccess ou CCE ReadOnlyAccess podem usar essa função.

Função do console	Serviços dependentes	Funções ou políticas necessárias
Gerenciamento de permissões	/	• Para contas de nuvem, nenhuma política/função adicional é necessária. • Os usuários do IAM com a permissão CCE Administrator ou Security Administrator global atribuída podem usar essa função. • Os usuários do IAM com a permissão CCE FullAccess ou CCE ReadOnlyAccess podem acessar o namespace. Além disso, os usuários do IAM devem ter as **permissões de administrador (cluster-admin)** no namespace.
ConfigMaps e segredos	/	• A criação de ConfigMaps não requer nenhuma política adicional. • A visualização de segredos requer que as permissões cluster-admin, admin ou edit sejam configuradas para o namespace. A política DEW KeypairFullAccess ou DEW KeypairReadOnlyAccess devem ser atribuídos para serviços dependentes.
Central de ajuda	/	/
Mudar para outros serviços relacionados	Software Repository for Container (SWR) Application Operations Management (AOM) Multi-Cloud Container Platform (MCP)	O console do CCE fornece links para outros serviços relacionados. Para exibir ou usar esses serviços, um usuário do IAM deve receber as permissões necessárias para os serviços.

15.6 Segurança de pod

15.6.1 Configuração de uma política de segurança de pod

Uma política de segurança de pods (PSP) é um recurso no nível de cluster que controla aspectos de segurança confidenciais da especificação do pod. O objeto da **PodSecurityPolicy** no Kubernetes define um grupo de condições que um pod deve cumprir para ser aceito pelo sistema, bem como os valores padrão dos campos relacionados.

Por padrão, o componente de controle de acesso da PSP é habilitado para clusters de v1.17.17 e uma PSP padrão global chamado **psp-global** é criado. Você pode modificar a política padrão (mas não excluí-la). Você também pode criar uma PSP e vinculá-la à configuração RBAC.

NOTA

- Além da PSP padrão global, o sistema configura PSPs independentes para componentes do sistema no namespace kube-system. Modificar a configuração psp-global não afeta a criação de pods no namespace kube-system.
- A PodSecurityPolicy foi preterido no Kubernetes v1.21 e removido do Kubernetes na v1.25. Você pode usar a admissão de segurança do pod como um substituto para a PodSecurityPolicy. Para mais detalhes, consulte **Configuração de admissão de segurança do pod**.

Modificar a PSP padrão global

Antes de modificar a PSP padrão global, certifique-se de que um cluster do CCE foi criado e conectado usando kubectl.

Passo 1 Execute o seguinte comando:

kubectl edit psp psp-global

Passo 2 Modifique os parâmetros necessários, conforme mostrado na **Tabela 15-15**.

Tabela 15-15 Configuração de PSP

Item	Descrição
privileged	Inicia o contêiner privilegiado.
hostPID hostIPC	Usa o namespace do host.
hostNetwork hostPorts	Utiliza a rede e a porta do host.
volumes	Especifica o tipo de volume montado que pode ser usado.
allowedHostPaths	Especifica o caminho do host no qual um volume hostPath pode ser montado. O campo **pathPrefix** especifica o grupo de prefixos de caminho de host no qual um volume hostPath pode ser montado.
allowedFlexVolumes	Especifica o driver FlexVolume que pode ser usado.
fsGroup	Configura o ID de grupo suplementar usado pelo volume montado no pod.
readOnlyRootFilesystem	Os pods só podem ser iniciados usando um sistema de arquivos raiz somente leitura.
runAsUser runAsGroup supplementalGroups	Especifica o ID do usuário, o ID do grupo primário e o ID do grupo suplementar para os contêineres iniciais em um pod.
allowPrivilegeEscalation defaultAllowPrivilegeEscala-tion	Especifica se **allowPrivilegeEscalation** pode ser definido como **true** em um pod. Essa configuração controla o uso do Setuid e se os programas podem usar chamadas de sistema privilegiadas adicionais.

Item	Descrição
defaultAddCapabilities requiredDropCapabilities allowedCapabilities	Controla os recursos do Linux usados em pods.
seLinux	Controla a configuração do seLinux usado em pods.
allowedProcMountTypes	Controla os ProcMountTypes que podem ser usados por pods.
annotations	Configura AppArmor e Seccomp usados por contêineres em um pod.
forbiddenSysctls allowedUnsafeSysctls	Controla a configuração do Sysctl usado por contêineres em um pod.

----Fim

Exemplo de ativação de Sysctls inseguros na política de segurança do pod

Você pode configurar permitido-unsafe-sysctls para um pool de nós. Para os clusters do CCE de **v1.17.17** e versões posteriores, adicione configurações em **allowedUnsafeSysctls** da política de segurança do pod para fazer a configuração entrar em vigor. Para mais detalhes, consulte **Tabela 15-15**.

Além de modificar a política de segurança global do pod, você pode adicionar novas políticas de segurança do pod. Por exemplo, habilite os sysctls inseguros **net.core.somaxconn**. Veja a seguir um exemplo de adição de uma política de segurança de pods:

```
apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
  annotations:
    seccomp.security.alpha.kubernetes.io/allowedProfileNames: '*'
  name: sysctl-psp
spec:
  allowedUnsafeSysctls:
  - net.core.somaxconn
  allowPrivilegeEscalation: true
  allowedCapabilities:
  - '*'
  fsGroup:
    rule: RunAsAny
  hostIPC: true
  hostNetwork: true
  hostPID: true
  hostPorts:
  - max: 65535
    min: 0
  privileged: true
  runAsGroup:
    rule: RunAsAny
  runAsUser:
    rule: RunAsAny
  seLinux:
    rule: RunAsAny
  supplementalGroups:
    rule: RunAsAny
  volumes:
```



```
  - '*'
---
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: sysctl-psp
rules:
  - apiGroups:
      - "*"
    resources:
      - podsecuritypolicies
    resourceNames:
      - sysctl-psp
    verbs:
      - use

---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: sysctl-psp
roleRef:
  kind: ClusterRole
  name: sysctl-psp
  apiGroup: rbac.authorization.k8s.io
subjects:
- kind: Group
  name: system:authenticated
  apiGroup: rbac.authorization.k8s.io
```

Restaurar a PSP original

Se você modificou a política de segurança do pod padrão e deseja restaurar a política de segurança do pod original, execute as seguintes operações.

Passo 1 Crie um arquivo de descrição de política chamado **policy.yaml**. **policy.yaml** é um nome de arquivo de exemplo. Você pode renomeá-lo conforme necessário.

vi policy.yaml

O conteúdo do arquivo de descrição é o seguinte:

```
apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
  name: psp-global
  annotations:
    seccomp.security.alpha.kubernetes.io/allowedProfileNames: '*'
spec:
  privileged: true
  allowPrivilegeEscalation: true
  allowedCapabilities:
    - '*'
  volumes:
    - '*'
  hostNetwork: true
  hostPorts:
    - min: 0
      max: 65535
  hostIPC: true
  hostPID: true
  runAsUser:
    rule: 'RunAsAny'
  seLinux:
    rule: 'RunAsAny'
  supplementalGroups:
    rule: 'RunAsAny'
  fsGroup:
```



```
    rule: 'RunAsAny'

---
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: psp-global
rules:
  - apiGroups:
      - "*"
    resources:
      - podsecuritypolicies
    resourceNames:
      - psp-global
    verbs:
      - use

---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: psp-global
roleRef:
  kind: ClusterRole
  name: psp-global
  apiGroup: rbac.authorization.k8s.io
subjects:
- kind: Group
  name: system:authenticated
  apiGroup: rbac.authorization.k8s.io
```

Passo 2 Execute o seguinte comando:

kubectl apply -f policy.yaml

----Fim

15.6.2 Configuração de admissão de segurança do pod

Antes de usar a **admissão de segurança do pod**, entenda os **Padrões de segurança do pod de Kubernetes**. Esses padrões definem diferentes níveis de isolamento para vagens. Eles permitem que você defina como deseja restringir o comportamento dos pods de forma clara e consistente. O Kubernetes oferece um controlador de admissão de segurança de pod embutido para aplicar os padrões de segurança do pod. As restrições de segurança do pod são aplicadas no nível do namespace quando os pods são criados.

O padrão de segurança do pod define três níveis de política de segurança:

Tabela 15-16 Níveis de política de segurança do pod

Nível	Descrição
privileged	Política irrestrita, fornecendo o mais amplo nível possível de permissões, geralmente destinada a cargas de trabalho no nível do sistema e da infraestrutura gerenciadas por usuários privilegiados e confiáveis, como CNIs e drivers de armazenamento.
baseline	Política minimamente restritiva que impede escalações de privilégios conhecidas, geralmente direcionadas a cargas de trabalho não críticas. Esta política desativa recursos como hostNetwork e hostPID.

Nível	Descrição
restricted	Política fortemente restrita, seguindo as práticas recomendadas atuais de endurecimento do Pod.

A admissão de segurança do pod é aplicada no nível do namespace. O controlador restringe o contexto de segurança e outros parâmetros no pod ou contêiner no namespace. A política privilegiada não verifica o campo **securityContext** do pod e do contêiner. As políticas de linha de base e restritas têm requisitos diferentes em **securityContext**. Para obter detalhes, consulte **Padrões de segurança do pod**.

Configurar o contexto de segurança: **configure um contexto de segurança para um pod ou contêiner**

Rótulos de admissão de segurança de pod

O Kubernetes define três tipos de rótulos para admissão de segurança de pods (consulte **Tabela 15-17**). Você pode definir esses rótulos em um namespace para definir o nível de padrão de segurança do pod a ser usado. No entanto, não altere o nível do padrão de segurança do pod em namespaces do sistema, como o kube-system. Caso contrário, os pods no namespace do sistema podem estar com defeito.

Tabela 15-17 Rótulos de admissão de segurança de pod

Modo	Objeto de destino	Descrição
enforce	Pods	Violações de política farão com que o pod seja rejeitado.
audit	Cargas de trabalho (como Implementação e tarefa)	As violações de política acionarão a adição de uma anotação de auditoria ao evento registrado no log de auditoria, mas são permitidas de outra forma.
warn	Cargas de trabalho (como Implementação e tarefa)	Violações de política acionarão um aviso voltado para o usuário, mas de outra forma são permitidas.

NOTA

Os pods geralmente são criados indiretamente, criando um objeto de carga de trabalho, como uma Implementação ou tarefa. Para ajudar a detectar violações precocemente, os modos de auditoria e aviso são aplicados aos recursos da carga de trabalho. No entanto, o modo de imposição é aplicado somente aos objetos de pod resultantes.

Impor a admissão de segurança de pod com rótulos de namespace

Você pode rotular namespaces para impor os padrões de segurança do pod. Suponha que um espaço de nomes está configurado da seguinte forma:


```
apiVersion: v1
kind: Namespace
metadata:
  name: my-baseline-namespace
  labels:
    pod-security.kubernetes.io/enforce: privileged
    pod-security.kubernetes.io/enforce-version: v1.25
    pod-security.kubernetes.io/audit: baseline
    pod-security.kubernetes.io/audit-version: v1.25
    pod-security.kubernetes.io/warn: restricted
    pod-security.kubernetes.io/warn-version: v1.25

    # The label can be in either of the following formats:
    # pod-security.kubernetes.io/<MODE>: <LEVEL>
    # pod-security.kubernetes.io/<MODE>-version: <VERSION>
    # The audit and warn modes inform you of which security behaviors are
violated by the load.
```

Os rótulos de namespace indicam qual nível de política aplicar para o modo. Para cada modo, há dois rótulos que determinam a política usada:

- pod-security.kubernetes.io/<MODE>: <LEVEL>
  - <MODE>: deve ser **enforce**, **audit** ou **warn**. Para obter detalhes sobre os modos, consulte **Tabela 15-17**.
  - <LEVEL>: deve ser **privileged**, **baseline** ou **restricted**. Para obter detalhes sobre os níveis, consulte **Tabela 15-16**.
- pod-security.kubernetes.io/<MODE>-version: <VERSION>

  Opcional, que fixa a política em uma determinada versão do Kubernetes.
  - <MODE>: deve ser **enforce**, **audit** ou **warn**. Para obter detalhes sobre os modos, consulte **Tabela 15-17**.
  - <VERSION>: número da versão do Kubernetes. Por exemplo, v1.25. Você também pode usar **latest**.

Se os pods forem implementados no namespace anterior, as seguintes restrições de segurança se aplicam:

1. A verificação no modo de impor é ignorada (modo impor + nível privilegiado).
2. As restrições relacionadas com a política de linha de base são verificadas (modo de auditoria + nível de linha de base). Ou seja, se o pod ou o contêiner violar a política, o evento correspondente será registrado no log de auditoria.
3. Restrições relacionadas à política restrita são verificadas (modo de aviso + nível restrito). Ou seja, se o pod ou contêiner violar a política, o usuário receberá um alarme ao criar o pod.

Migrar política de segurança do pod para a admissão de segurança do pod

Se você usar políticas de segurança de pods em um cluster anterior à v1.25 e precisar substituí-las pela admissão de segurança de pods em um cluster v1.25 ou posterior, siga o guia em **Migração de PodSecurityPolicy para o controlador de admissão interno de PodSecurity**.

AVISO

1. A admissão de segurança do pod suporta apenas três modos de isolamento, menos flexíveis do que as políticas de segurança do pod. Se você precisar de mais controle sobre restrições específicas, será necessário usar um Webhook de validação de admissão para impor essas políticas.
2. Admissão de segurança de pod é um controlador de admissão não-mutante, o que significa que não irá modificar pods antes de validá-los. Se você estava confiando neste aspecto da PSP, você precisará modificar o contexto de segurança em suas cargas de trabalho ou usar um Webhook de admissão com mutação para fazer essas alterações.
3. A PSP permite que você vincule políticas diferentes a contas de serviço diferentes. Essa abordagem tem muitas armadilhas e não é recomendada, mas se você precisar desse recurso de qualquer maneira, precisará usar um webhook de terceiros.
4. Não aplique a admissão de segurança do pod aos namespaces onde os componentes do CCE, como kube-system, kube-public e kube-node-lease, são implementados. Caso contrário, os componentes do CCE e funções adicionais serão anormais.

Documentação

- **Admissão de segurança de pod**
- **Mapeamento de PodSecurityPolicies para padrões de segurança de pods**
- **Aplicação de padrões de segurança do pod com rótulos de namespace**
- **Aplicação de padrões de segurança do pod configurando o controlador de admissão embutido**

15.7 Melhoria da segurança do token da conta de serviço

Em clusters anteriores à v1.21, um token é obtido montando o segredo da conta de serviço em um pod. Os tokens obtidos dessa maneira são permanentes. Essa abordagem não é mais recomendada a partir da versão 1.21. As contas de serviço interromperão a criação automática de segredos em clusters a partir da versão 1.25.

Em clusters da versão 1.21 ou posterior, você pode usar a API **TokenRequest** para obter o token e usar o volume projetado para montar o token no pod. Tais tokens são válidos por um período fixo (uma hora por padrão). Antes da expiração, o Kubelet atualiza o token para garantir que o pod sempre use um token válido. Quando o pod de montagem é excluído, o token se torna automaticamente inválido. Essa abordagem é implementada pelo recurso **BoundServiceAccountTokenVolume** para melhorar a segurança do token da conta de serviço. Clusters da v1.21 ou posterior habilitam essa abordagem por padrão.

Para uma transição suave, a comunidade estende o período de validade do token para um ano por padrão. Após um ano, o token se torna inválido e os clientes que não suportam o recarregamento de certificados não podem acessar o servidor da API. Recomenda-se que os clientes de versões anteriores sejam atualizados o mais rápido possível. Caso contrário, podem ocorrer falhas de serviço.

Se você usar um cliente do Kubernetes de uma versão a ser desatualizada, o recarregamento do certificado poderá falhar. As versões das bibliotecas de cliente de Kubernetes oficialmente suportadas capazes de recarregar tokens são as seguintes:

- Go: >= v0.15.7
- Python: >= v12.0.0
- Java: >= v9.0.0
- Javascript: >= v0.10.3
- Ruby: master branch
- Haskell: v0.3.0.0
- C#: >= 7.0.5

Para mais detalhes, visite **https://github.com/kubernetes/enhancements/tree/master/keps/sig-auth/1205-bound-service-account-tokens**.

NOTA

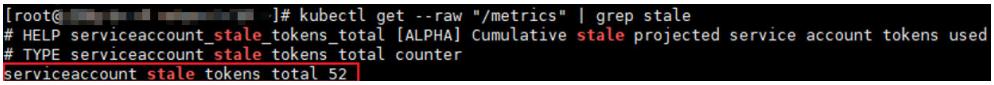
Se você precisar de um token que nunca expire, também poderá **gerenciar manualmente segredos para contas de serviço**. Embora um token de conta de serviço permanente possa ser criado manualmente, é aconselhável usar um token de curta duração chamando a API **TokenRequest** para maior segurança.

Diagnóstico

Execute as seguintes etapas para verificar os clusters de CCE v1.21 ou posterior:

1. Verifique as versões de complemento.
   - Se você estiver usando o complemento prometheus v2.23.34 ou anterior, atualize-o para v2.23.34 ou posterior.
   - Se você estiver usando o complemento npd v1.15.0 ou anterior, atualize-o para a versão mais recente.
2. Use o kubectl para conectar-se ao cluster e execute o comando **kubectl get --raw "/metrics" | grep stale** para obter as métricas. Verifique a métrica chamada **serviceaccount_stale_tokens_total**.

   Se o valor for maior que 0, algumas cargas de trabalho no cluster podem estar usando uma versão de client-go anterior. Nesse caso, verifique se esse problema ocorre nas aplicações implementados. Se sim, atualize o client-go para a versão especificada pela comunidade o mais rápido possível. A versão deve ser pelo menos duas versões principais do cluster do CCE. Por exemplo, se a versão do cluster for 1.23, a versão da biblioteca de dependência do Kubernetes deve ser pelo menos 1.19.



15.8 Descrição da atribuição do sistema

O CCE trabalha em estreita colaboração com vários serviços de nuvem para oferecer suporte a funções de computação, armazenamento, rede e monitoramento. Quando você faz logon no console do CCE pela primeira vez, o CCE solicita automaticamente permissões para acessar esses serviços de nuvem na região onde você executa suas aplicações. Especificamente:

- Serviços de computação

  Quando você cria um nó em um cluster, um servidor de nuvem é criado de acordo. O pré-requisito é que o CCE tenha obtido as permissões para acessar o Elastic Cloud Service (ECS) e Bare Metal Server (BMS).

- Serviços de armazenamento

  O CCE permite que você monte armazenamento em nós e contêineres em um cluster. O pré-requisito é que o CCE tenha obtido as permissões para acessar serviços como Elastic Volume Service (EVS), Scalable File Service (SFS) e Object Storage Service (OBS).

- Serviços de rede

  O CCE permite que os contêineres em um cluster sejam publicados como serviços que podem ser acessados por sistemas externos. O pré-requisito é que o CCE tenha obtido as permissões para acessar serviços como Virtual Private Cloud (VPC) e Elastic Load Balance (ELB).

- Serviços de contêineres e monitoramento

  O CCE suporta funções como extração de imagem de contêiner, monitoramento e registro em logs. O pré-requisito é que o CCE tenha obtido as permissões para acessar serviços como o SoftWare Repository for Container (SWR) e Application Operations Management (AOM).

Depois de concordar com a atribuição, o CCE cria automaticamente uma agência no IAM para delegar outras permissões de operação de recursos em sua conta ao CCE da Huawei Cloud. Para obter detalhes, consulte **Delegação de conta**.

As agências criadas automaticamente pelo CCE são as seguintes:

- **cce_admin_trust**
- **cce_cluster_agency**

cce_admin_trust

A agência cce_admin_trust tem as permissões de administrador do locatário. O administrador de locatário tem as permissões em todos os serviços de nuvem, exceto o IAM, que são usados para chamar os serviços de nuvem dos quais o CCE depende. A delegação tem efeito apenas na região atual.

Para usar o CCE em várias regiões, solicite permissões de recursos de nuvem em cada região. Você pode acessar o console do IAM, escolher **Agencies** e clicar em **cce_admin_trust** para exibir os registros de delegação de cada região.

NOTA

CCE pode falhar a execução como esperado se a função Tenant Administrator não for atribuída. Portanto, não exclua nem modifique a agência **cce_admin_trust** ao usar o CCE.

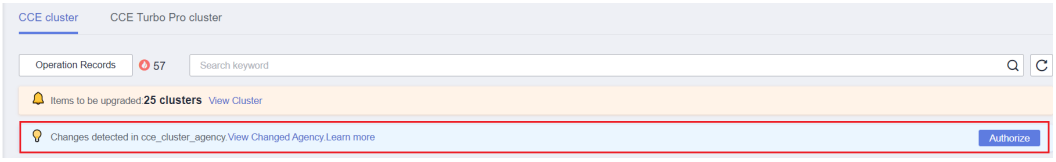
cce_cluster_agency

A agência cce_cluster_agency contém apenas as permissões de operação de recursos de serviço de nuvem exigidas pelos componentes do CCE. Ela gera credenciais de acesso temporárias usadas por componentes em clusters do CCE.

NOTA

- A agência cce_cluster_agency oferece suporte a clusters v1.21 ou posterior.
- Quando você cria a agência cce_cluster_agency, uma política personalizada chamada **CCE cluster policies** é criada automaticamente. Não exclua esta política.

Se as permissões da agência cce_cluster_agency forem diferentes das esperadas pelo CCE, o console exibirá uma mensagem indicando que as permissões foram alteradas e que você precisa autorizar novamente a agência.



A agência cce_cluster_agency pode ser reautorizada nos seguintes cenários:

- As permissões das quais os componentes do CCE dependem podem mudar com as versões. Por exemplo, se um novo componente depender de novas permissões, o CCE atualizará a lista de permissões esperadas e você precisará conceder permissões para cce_cluster_agency novamente.
- Quando você modifica manualmente as permissões da agência cce_cluster_agency, as permissões da agência são diferentes daquelas esperadas pelo CCE. Nesse caso, uma mensagem é exibida, solicitando que você autorize novamente a agência. Se você autorizar novamente a agência, as permissões modificadas manualmente podem se tornar inválidas.

16 Gerenciamento do armazenamento: FlexVolume (preterido)

16.1 Visão geral de FlexVolume

No armazenamento de contêineres, você pode usar diferentes tipos de volumes e montá-los em contêineres em pods quantos desejar.

No CCE, o armazenamento de contêineres é apoiado por objetos nativos do Kubernetes, como hostPath, segredo e ConfigMap e por serviços de armazenamento em nuvem.

Os clusters do CCE **1.13 e versões anteriores** usam o complemento **storage-driver** para se conectar a serviços de armazenamento em nuvem para dar suporte ao driver de FlexVolume do Kubernetes para armazenamento de contêiner. O driver de FlexVolume foi preterido em favor da Interface de armazenamento de contêiner (CSI). **O complemento everest para CSI é instalado em clusters do CCE de 1.15 e versões posteriores por padrão.** Para mais detalhes, consulte **Visão geral**.

NOTA

- Em clusters de CCE anteriores ao Kubernetes 1.13, a expansão de capacidade de ponta a ponta do armazenamento de contêineres não é suportada, e a capacidade de PVC é inconsistente com a capacidade de armazenamento.
- **Em um cluster v1.13 ou anterior**, quando uma atualização ou correção de bug estiver disponível para funcionalidades de armazenamento, você só precisará instalar ou atualizar o complemento storage-driver. Não é necessário atualizar o cluster ou criar um cluster.

Restrições

- Para clusters criados no CCE, o Kubernetes v1.15.11 é uma versão de transição na qual o plug-in de FlexVolume (**storage-driver**) é compatível com o plug-in de CSI (**everest**). Clusters de v1.17 e versões posteriores não suportam mais o FlexVolume. Use o complemento everest.
- O plug-in de FlexVolume storage-driver será mantido pelos desenvolvedores do Kubernetes, mas novas funcionalidades serão adicionadas apenas a **Armazenamento do contêiner do CCE (Everest)** da CSI. Não crie mais armazenamento que acesse o storage-driver de FlexVolume no CCE. Caso contrário, os recursos de armazenamento podem não funcionar corretamente.

Verificar complementos de armazenamento

Passo 1 Efetue logon no console do CCE.

Passo 2 Na árvore de navegação à esquerda, clique em **Add-ons**.

Passo 3 Clique na guia **Add-on Instance**.

Passo 4 Selecione um cluster no canto superior direito. O complemento de armazenamento padrão instalado durante a criação do cluster é exibido.

----Fim

16.2 Alteração da classe de armazenamento usada por um cluster de v1.15 de FlexVolume para CSI Everest

Em clusters posteriores a v1.15.11-r1, o CSI (o complemento everest) assumiu todas as funções do fuxi FlexVolume (o complemento do driver de armazenamento) para gerenciar o armazenamento de contêineres. É aconselhável usar o CSI Everest.

Para migrar seus volumes de armazenamento, crie um PV estático para associar ao armazenamento subjacente original e, em seguida, crie um PVC para associar a esse PV estático. Ao atualizar sua aplicação, monte o novo PVC no caminho de montagem original para migrar os volumes de armazenamento.

ATENÇÃO

Os serviços serão interrompidos durante a migração. Portanto, planeje adequadamente a migração e faça backup dos dados.

Procedimento

Passo 1 (Opcional) Faça backup dos dados para evitar a perda de dados em caso de exceções.

Passo 2 Configure um arquivo YAML do PV no formato CSI de acordo com o PV no formato FlexVolume e associe o PV ao armazenamento existente.

Para ser específico, execute os seguintes comandos para configurar o arquivo pv-example.yaml, que é usado para criar um PV.

touch pv-example.yaml

vi pv-example.yaml

Exemplo de configuração de **um PV para um volume do EVS**:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  labels:
    failure-domain.beta.kubernetes.io/region: ap-southeast-1
    failure-domain.beta.kubernetes.io/zone: <zone name>
  annotations:
    pv.kubernetes.io/provisioned-by: everest-csi-provisioner
  name: pv-evs-example
spec:
  accessModes:
```



```
  - ReadWriteOnce
  capacity:
    storage: 10Gi
  csi:
    driver: disk.csi.everest.io
    fsType: ext4
    volumeAttributes:
      everest.io/disk-mode: SCSI
      everest.io/disk-volume-type: SAS
      storage.kubernetes.io/csiProvisionerIdentity: everest-csi-provisioner
    volumeHandle: 0992dbda-6340-470e-a74e-4f0db288ed82
  persistentVolumeReclaimPolicy: Delete
  storageClassName: csi-disk
```

Preste atenção aos campos em negrito e vermelho. Os parâmetros são descritos a seguir:

Tabela 16-1 Parâmetros de configuração de volume do EVS

Parâmetro	Descrição
failure-domain.beta.kubernetes.io/region	Região onde o disco do EVS está localizado. Use o mesmo valor que o do PV do FlexVolume.
failure-domain.beta.kubernetes.io/zone	AZ onde o disco do EVS está localizado. Use o mesmo valor que o do PV do FlexVolume.
name	Nome do PV, que deve ser exclusivo no cluster.
storage	Capacidade de volume do EVS na unidade de Gi. Use o valor de **spec.capacity.storage** do PV de FlexVolume.
driver	Driver de armazenamento usado para anexar o volume. Defina o driver como **disk.csi.everest.io** para o volume do EVS.
volumeHandle	ID do volume do disco do EVS. Use o valor de **spec.flexVolume.options.volumeID** do PV do FlexVolume.
everest.io/disk-mode	Modo de disco do EVS. Use o valor de **spec.flexVolume.options.disk-mode** do PV do FlexVolume.
everest.io/disk-volume-type	Tipos de disco do EVS. Atualmente, há suporte para I/O alta (SAS) e I/O ultra-alta (SSD). Use o valor de **kubernetes.io/volumetype** na classe de armazenamento correspondente a **spec.storageClassName** do PV de FlexVolume.
storageClassName	Nome da classe de armazenamento do Kubernetes associada ao volume de armazenamento. Defina este campo como **csi-disk** para discos do EVS.

Exemplo de configuração de **um PV para um volume do SFS**:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-sfs-example
  annotations:
    pv.kubernetes.io/provisioned-by: everest-csi-provisioner
```



```
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 10Gi
  csi:
    driver: nas.csi.everest.io
    fsType: nfs
    volumeAttributes:
      everest.io/share-export-location: sfs-nas01.ap-
southeast-1.myhuaweicloud.com:/share-436304e8
      storage.kubernetes.io/csiProvisionerIdentity: everest-csi-provisioner
    volumeHandle: 682f00bb-ace0-41d8-9b3e-913c9aa6b695
  persistentVolumeReclaimPolicy: Delete
  storageClassName: csi-nas
```

Preste atenção aos campos em negrito e vermelho. Os parâmetros são descritos a seguir:

Tabela 16-2 Parâmetros de configuração de volume do SFS

Parâmetro	Descrição
name	Nome do PV, que deve ser exclusivo no cluster.
storage	Tamanho do armazenamento de arquivos na unidade de Gi. Use o valor de **spec.capacity.storage** do PV de FlexVolume.
driver	Driver de armazenamento usado para anexar o volume. Defina o driver como **nas.csi.everest.io** para o sistema de arquivos.
everest.io/share-export-location	Caminho compartilhado do sistema de arquivos. Use o valor de **spec.flexVolume.options.deviceMountPath** do PV de FlexVolume.
volumeHandle	ID do sistema de arquivos. Use o valor de **spec.flexVolume.options.volumeID** do PV do FlexVolume.
storageClassName	Nome da classe de armazenamento do Kubernetes. Defina este campo como **csi-nas**.

Exemplo de configuração de **um PV para um volume do OBS**:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-obs-example
  annotations:
    pv.kubernetes.io/provisioned-by: everest-csi-provisioner
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 1Gi
  csi:
    driver: obs.csi.everest.io
    fsType: s3fs
    volumeAttributes:
      everest.io/obs-volume-type: STANDARD
      everest.io/region: ap-southeast-1
      storage.kubernetes.io/csiProvisionerIdentity: everest-csi-provisioner
    volumeHandle: obs-normal-static-pv
```



```
  persistentVolumeReclaimPolicy: Delete
  storageClassName: csi-obs
```

Preste atenção aos campos em negrito e vermelho. Os parâmetros são descritos a seguir:

Tabela 16-3 Parâmetros de configuração de volume do OBS

Parâmetro	Descrição
name	Nome do PV, que deve ser exclusivo no cluster.
storage	Capacidade de armazenamento, na unidade de Gi. Defina esse parâmetro como o valor fixo **1Gi**.
driver	Driver de armazenamento usado para anexar o volume. Defina o driver como **obs.csi.everest.io** para o volume do OBS.
fsType	Tipo de arquivo. As opções de valor são **obsfs** ou **s3fs**. Se o valor for **s3fs**, um bucket do OBS será criado e montado usando s3fs. Se o valor for **obsfs**, um sistema de arquivos paralelo do OBS será criado e montado usando obsfs. Defina este parâmetro de acordo com o valor de **spec.flexVolume.options.posix** do PV de FlexVolume. Se o valor de **spec.flexVolume.options.posix** for **true**, defina esse parâmetro como **obsfs**. Se o valor for **false**, defina este parâmetro como **s3fs**.
everest.io/obs-volume-type	Classe de armazenamento, incluindo **STANDARD** (bucket padrão) e **WARM** (bucket de acesso infrequente). Defina este parâmetro de acordo com o valor de **spec.flexVolume.options.storage_class** do PV do FlexVolume. Se o valor de **spec.flexVolume.options.storage_class** for **standard**, defina este parâmetro como **STANDARD**. Se o valor for **standard_ia**, defina este parâmetro como **WARM**.
everest.io/region	Região onde o bucket do OBS está localizado. Use o valor de **spec.flexVolume.options.region** do PV do FlexVolume.
volumeHandle	Nome do bucket do OBS. Use o valor de **spec.flexVolume.options.volumeID** do PV do FlexVolume.
storageClassName	Nome da classe de armazenamento do Kubernetes. Defina este campo como **csi-obs**.

Exemplo de configuração de **um PV para um volume do SFS Turbo**:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-efs-example
  annotations:
    pv.kubernetes.io/provisioned-by: everest-csi-provisioner
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 10Gi
  csi:
```



```
    driver: sfsturbo.csi.everest.io
    fsType: nfs
    volumeAttributes:
      everest.io/share-export-location: 192.168.0.169:/
      storage.kubernetes.io/csiProvisionerIdentity: everest-csi-provisioner
    volumeHandle: 8962a2a2-a583-4b7f-bb74-fe76712d8414
  persistentVolumeReclaimPolicy: Delete
  storageClassName: csi-sfsturbo
```

Preste atenção aos campos em negrito e vermelho. Os parâmetros são descritos a seguir:

Tabela 16-4 Parâmetros de configuração do volume do SFS Turbo

Parâmetro	Descrição
name	Nome do PV, que deve ser exclusivo no cluster.
storage	Tamanho do sistema de arquivos. Use o valor de **spec.capacity.storage** do PV de FlexVolume.
driver	Driver de armazenamento usado para anexar o volume. Defina-o como **sfsturbo.csi.everest.io**.
everest.io/share-export-location	Caminho compartilhado do volume do SFS Turbo. Use o valor de **spec.flexVolume.options.deviceMountPath** do PV de FlexVolume.
volumeHandle	ID do volume do SFS Turbo. Use o valor de **spec.flexVolume.options.volumeID** do PV do FlexVolume.
storageClassName	Nome da classe de armazenamento do Kubernetes. Defina este campo como **csi-sfsturbo** para volumes do SFS Turbo.

Passo 3 Configure um arquivo YAML do PVC no formato CSI de acordo com o PVC no formato FlexVolume e associe o PVC ao PV criado em **Passo 2**.

Para ser específico, execute os seguintes comandos para configurar o arquivo pvc-example.yaml, que é usado para criar um PVC.

touch pvc-example.yaml

vi pvc-example.yaml

Exemplo de configuração de **um PVC para um volume do EVS**:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  labels:
    failure-domain.beta.kubernetes.io/region: ap-southeast-1
    failure-domain.beta.kubernetes.io/zone: <zone name>
  annotations:
    everest.io/disk-volume-type: SAS
    volume.beta.kubernetes.io/storage-provisioner: everest-csi-provisioner
  name: pvc-evs-example
  namespace: default
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
```



```
      storage: 10Gi
  volumeName:  pv-evs-example
  storageClassName: csi-disk
```

Preste atenção aos campos em negrito e vermelho. Os parâmetros são descritos a seguir:

Tabela 16-5 Parâmetros de configuração de PVC para um volume do EVS

Parâmetro	Descrição
failure-domain.beta.kubernetes.io/region	Região onde o cluster está localizado. Use o mesmo valor que o do PVC do FlexVolume.
failure-domain.beta.kubernetes.io/zone	AZ onde o disco do EVS é implementado. Use o mesmo valor que o do PVC do FlexVolume.
everest.io/disk-volume-type	Classe de armazenamento do disco do EVS. O valor pode ser **SAS** ou **SSD**. Defina este parâmetro para o mesmo valor do PV criado em **Passo 2**.
name	Nome do PVC, que deve ser exclusivo no namespace. O valor deve ser exclusivo no namespace. (Se o PVC for criado dinamicamente por uma aplicação com estado, o valor desse parâmetro deverá ser igual ao nome do PVC de FlexVolume.)
namespace	Namespace ao qual o PVC pertence. Use o mesmo valor que o do PVC do FlexVolume.
storage	Capacidade solicitada do PVC, que deve ser igual ao tamanho de armazenamento do PV existente.
volumeName	Nome do PV. Defina este parâmetro como o nome do PV estático em **Passo 2**.
storageClassName	Nome da classe de armazenamento do Kubernetes. Defina este campo como **csi-disk** para discos do EVS.

Exemplo de configuração de **um PVC para um volume do SFS**:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    volume.beta.kubernetes.io/storage-provisioner: everest-csi-provisioner
  name: pvc-sfs-example
  namespace: default
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 10Gi
  storageClassName: csi-nas
  volumeName: pv-sfs-example
```

Preste atenção aos campos em negrito e vermelho. Os parâmetros são descritos a seguir:

Tabela 16-6 Parâmetros de configuração de PVC para um volume do SFS

Parâmetro	Descrição
name	Nome do PVC, que deve ser exclusivo no namespace. O valor deve ser exclusivo no namespace. (Se o PVC for criado dinamicamente por uma aplicação com estado, o valor desse parâmetro deverá ser igual ao nome do PVC de FlexVolume.)
namespace	Namespace ao qual o PVC pertence. Use o mesmo valor que o do PVC do FlexVolume.
storage	Capacidade de armazenamento, na unidade de Gi. O valor deve ser o mesmo que o tamanho de armazenamento do PV existente.
storageClassName	Defina este campo como **csi-nas**.
volumeName	Nome do PV. Defina este parâmetro como o nome do PV estático em **Passo 2**.

Exemplo de configuração de **um PVC para um volume do OBS**:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    volume.beta.kubernetes.io/storage-provisioner: everest-csi-provisioner
    everest.io/obs-volume-type: STANDARD
    csi.storage.k8s.io/fstype: s3fs
  name: pvc-obs-example
  namespace: default
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: csi-obs
  volumeName: pv-obs-example
```

Preste atenção aos campos em negrito e vermelho. Os parâmetros são descritos a seguir:

Tabela 16-7 Parâmetros de configuração de PVC para um volume do OBS

Parâmetro	Descrição
everest.io/obs-volume-type	Tipo de volume do OBS, que pode ser **STANDARD** (bucket padrão) e **WARM** (bucket de acesso infrequente). Defina este parâmetro para o mesmo valor do PV criado em **Passo 2**.
csi.storage.k8s.io/fstype	Tipo de arquivo, que pode ser **obsfs** ou **s3fs**. O valor deve ser o mesmo que o de **fsType** do PV do volume do OBS estático.
name	Nome do PVC, que deve ser exclusivo no namespace. O valor deve ser exclusivo no namespace. (Se o PVC for criado dinamicamente por uma aplicação com estado, o valor desse parâmetro deverá ser igual ao nome do PVC de FlexVolume.)

Parâmetro	Descrição
namespace	Namespace ao qual o PVC pertence. Use o mesmo valor que o do PVC do FlexVolume.
storage	Capacidade de armazenamento, na unidade de Gi. Defina esse parâmetro como o valor fixo **1Gi**.
storageClassName	Nome da classe de armazenamento do Kubernetes. Defina este campo como **csi-obs**.
volumeName	Nome do PV. Defina este parâmetro como o nome do PV estático criado em **Passo 2**.

Exemplo de configuração de **um PVC para um volume do SFS Turbo**:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    volume.beta.kubernetes.io/storage-provisioner: everest-csi-provisioner
  name: pvc-efs-example
  namespace: default
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 10Gi
  storageClassName: csi-sfsturbo
  volumeName: pv-efs-example
```

Preste atenção aos campos em negrito e vermelho. Os parâmetros são descritos a seguir:

Tabela 16-8 Parâmetros de configuração de PVC para um volume do SFS Turbo

Parâmetro	Descrição
name	Nome do PVC, que deve ser exclusivo no namespace. O valor deve ser exclusivo no namespace. (Se o PVC for criado dinamicamente por uma aplicação com estado, o valor desse parâmetro deverá ser igual ao nome do PVC de FlexVolume.)
namespace	Namespace ao qual o PVC pertence. Use o mesmo valor que o do PVC do FlexVolume.
storageClassName	Nome da classe de armazenamento do Kubernetes. Defina este campo como **csi-sfsturbo**.
storage	Capacidade de armazenamento, na unidade de Gi. O valor deve ser o mesmo que o tamanho de armazenamento do PV existente.
volumeName	Nome do PV. Defina este parâmetro como o nome do PV estático criado em **Passo 2**.

Passo 4 Atualize a carga de trabalho para usar um novo PVC.

Para implementações

1. Execute os comandos **kubectl create -f** para criar um PV e um PVC.

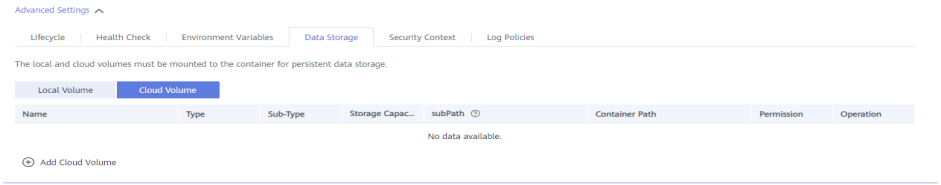
   kubectl create -f pv-example.yaml

   kubectl create -f pvc-example.yaml

   NOTA

   Substitua o nome de arquivo de exemplo **pvc-example.yaml** nos comandos anteriores pelos nomes dos arquivos YAML configurados em **Passo 2** e **Passo 3**.

2. Vá para o console do CCE. Na página de atualização da carga de trabalho, clique em **Upgrade** > **Advanced Settings** > **Data Storage** > **Cloud Storage**.



3. Desinstale o armazenamento antigo e adicione o PVC no formato CSI. Mantenha o caminho de montagem original no contêiner.
4. Clique em **Submit**.
5. Aguarde até que os pods estejam funcionando.

Para StatefulSets que usam armazenamento existente

1. Execute os comandos **kubectl create -f** para criar um PV e um PVC.

   kubectl create -f pv-example.yaml

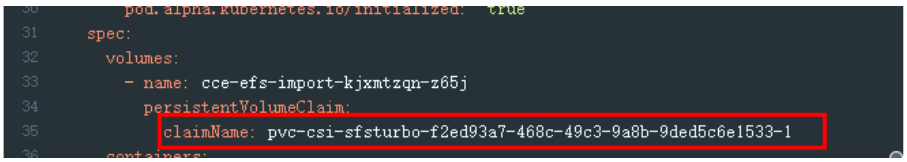
   kubectl create -f pvc-example.yaml

   NOTA

   Substitua o nome de arquivo de exemplo **pvc-example.yaml** nos comandos anteriores pelos nomes dos arquivos YAML configurados em **Passo 2** e **Passo 3**.

2. Execute o comando **kubectl edit** para editar o StatefulSet e usar o PVC recém-criado.

   kubectl edit sts sts-example -n xxx



   NOTA

   Substitua **sts-example** no comando anterior pelo nome real do StatefulSet a ser atualizado. **xxx** indica o namespace ao qual o StatefulSet pertence.

3. Aguarde até que os pods estejam funcionando.

NOTA

O console atual não oferece suporte à operação de adição de novo armazenamento em nuvem para o StatefulSets. Use os comandos kubectl para substituir o armazenamento pelo PVC recém-criado.

Para StatefulSets que usam armazenamento alocado dinamicamente

1. Faça backup do PV e do PVC no formato flexVolume usado pelo StatefulSet.

   kubectl get pvc xxx -n {namespaces} -oyaml > pvc-backup.yaml

   kubectl get pv xxx -n {namespaces} -oyaml > pv-backup.yaml

2. Altere o número de pods para **0**.
3. Na página de armazenamento, desassocie o PVC de flexVolume usado pelo StatefulSet.
4. Execute os comandos **kubectl create -f** para criar um PV e um PVC.

   kubectl create -f pv-example.yaml

   kubectl create -f pvc-example.yaml

   NOTA

   Substitua o nome de arquivo de exemplo **pvc-example.yaml** nos comandos anteriores pelos nomes dos arquivos YAML configurados em **Passo 2** e **Passo 3**.

5. Altere o número de pods de volta ao valor original e aguarde até que os pods estejam em execução.

NOTA

A alocação dinâmica de armazenamento para StatefulSets é obtida usando **volumeClaimTemplates**. Este campo não pode ser modificado pelo Kubernetes. Portanto, os dados não podem ser migrados usando um novo PVC.

A regra de nomeação de PVC do **volumeClaimTemplates** é fixa. Quando existe um PVC que atende à regra de nomeação, esse PVC é usado.

Portanto, desassocie o PVC original primeiro e, em seguida, crie um PVC com o mesmo nome no formato CSI.

6. (Opcional) Recrie a aplicação com estado para garantir que um PVC de CSI seja usado quando a aplicação for dimensionada. Caso contrário, PVCs de FlexVolume são usados no dimensionamento.

- Execute o seguinte comando para obter o arquivo YAML do StatefulSet:

kubectl get sts xxx -n {namespaces} -oyaml > sts.yaml

- Execute o seguinte comando para fazer backup do arquivo YAML do StatefulSet:

cp sts.yaml sts-backup.yaml

- Modifique a definição de **volumeClaimTemplates** no arquivo YAML do StatefulSet.

vi sts.yaml

Exemplo de configuração de **volumeClaimTemplates para um volume do EVS**:

```
volumeClaimTemplates:
  - metadata:
      name: pvc-161070049798261342
      namespace: default
      creationTimestamp: null
      annotations:
        everest.io/disk-volume-type: SAS
    spec:
      accessModes:
        - ReadWriteOnce
      resources:
        requests:
          storage: 10Gi
      storageClassName: csi-disk
```

O valor do parâmetro deve ser o mesmo que o PVC do volume do EVS criado em **Passo 3**.

Exemplo de configuração de **volumeClaimTemplates para um volume do SFS**:

```
volumeClaimTemplates:
  - metadata:
      name: pvc-161063441560279697
      namespace: default
      creationTimestamp: null
    spec:
      accessModes:
        - ReadWriteMany
      resources:
        requests:
          storage: 10Gi
      storageClassName: csi-nas
```

O valor do parâmetro deve ser o mesmo que o PVC do volume do SFS criado em **Passo 3**.

Exemplo de configuração de **volumeClaimTemplates para um volume do OBS**:

```
volumeClaimTemplates:
  - metadata:
      name: pvc-161070100417416148
      namespace: default
      creationTimestamp: null
      annotations:
        csi.storage.k8s.io/fstype: s3fs
        everest.io/obs-volume-type: STANDARD
    spec:
      accessModes:
        - ReadWriteMany
      resources:
        requests:
          storage: 1Gi
      storageClassName: csi-obs
```

O valor do parâmetro deve ser o mesmo que o PVC do volume do OBS criado em **Passo 3**.

- Exclua o StatefulSet.

kubectl delete sts xxx -n {namespaces}

- Crie o StatefulSet.

kubectl create -f sts.yaml

Passo 5 Verifique as funções de serviço.

1. Verifique se a aplicação está sendo executada corretamente.
2. Verifique se o armazenamento de dados está normal.

NOTA

Se uma reversão for necessária, execute **Passo 4**. Selecione o PVC no formato FlexVolume e atualize a aplicação.

Passo 6 Desinstale o PVC no formato FlexVolume.

Se a aplicação funcionar normalmente, desvincule o PVC no formato FlexVolume na página de gerenciamento de armazenamento.

Você também pode executar o comando kubectl para excluir o PVC e o PV do formato FlexVolume.

⚠ CUIDADO

Antes de excluir um PV, altere o persistentVolumeReclaimPolicy do PV para **Retain**. Caso contrário, o armazenamento subjacente será recuperado após a exclusão do PV.

Se o cluster tiver sido atualizado antes da migração de armazenamento, os PVs poderão não ser excluídos. Você pode remover o campo de proteção do PV **finalizers** para excluir PVs.

kubectl patch pv {pv_name} -p '{"metadata":{"finalizers":null}}'

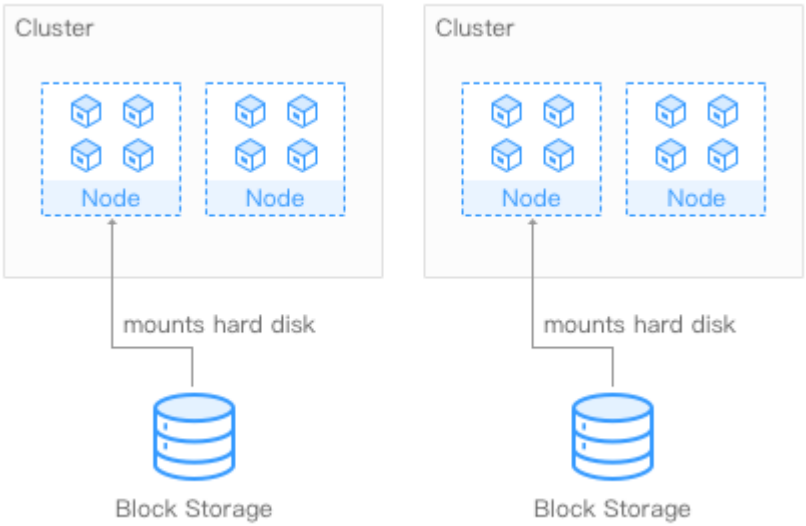
----Fim

16.3 Uso de discos EVS como volumes de armazenamento

16.3.1 Visão geral

Para obter armazenamento persistente, o CCE permite que você monte os volumes de armazenamento criados a partir de discos EVS (Elastic Volume Service) em um caminho de um contêiner. Quando o contêiner é migrado, os volumes do EVS montados também são migrados. Usando volumes do EVS, você pode montar o diretório de arquivos remotos do sistema de armazenamento em um contêiner para que os dados no volume de dados sejam permanentemente preservados, mesmo quando o contêiner for excluído.

Figura 16-1 Montagem de volumes EVS para CCE



Descrição

- **Fácil de usar**: semelhante à formatação de discos para servidores locais em layouts tradicionais, você pode formatar o armazenamento em blocos (discos) montados em servidores em nuvem e criar sistemas de arquivos neles.
- **Isolamento de dados**: cada servidor usa um dispositivo de armazenamento de blocos (disco) independente.
- **Rede privada**: o usuário pode acessar dados apenas em redes privadas de data centers.

- **Capacidade e desempenho**: a capacidade de um único volume é limitada (nível de TB), mas o desempenho é excelente (latência de I/O de leitura/gravação em nível de ms).
- **Restrição**: discos EVS com partições ou sistemas de arquivos não ext4 não podem ser importados.
- **Aplicações**: HPC, aplicações principais corporativas executados em clusters, sistemas de aplicações corporativas e desenvolvimento e teste. Esses volumes são frequentemente usados por Implementações e tarefas de pod único ou exclusivamente por cada pod em um StatefulSet. Os discos EVS são armazenamentos não compartilhados e não podem ser conectados a vários nós ao mesmo tempo. Se dois pods estiverem configurados para usar o mesmo disco EVS e os dois pods estiverem programados para nós diferentes, um pod não poderá ser iniciado porque o disco EVS não pode ser conectado a ele.

16.3.2 (kubectl) Criação automática de um disco EVS

Observações e restrições

O exemplo de configuração a seguir se aplica a clusters do Kubernetes 1.13 ou anterior.

Procedimento

Passo 1 Use o kubectl para se conectar ao cluster. Para mais detalhes, consulte **Conexão a um cluster usando o kubectl**.

Passo 2 Execute os comandos a seguir para configurar o arquivo **pvc-evs-auto-example.yaml**, que é usado para criar uma PVC.

touch pvc-evs-auto-example.yaml

vi pvc-evs-auto-example.yaml

Exemplo de arquivo YAML para clusters de v1.9, v1.11 e v1.13:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-evs-auto-example
  namespace: default
  annotations:
    volume.beta.kubernetes.io/storage-class: sas
  labels:
    failure-domain.beta.kubernetes.io/region: ap-southeast-1
    failure-domain.beta.kubernetes.io/zone: ap-southeast-1a
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
```

Tabela 16-9 Parâmetros principais

Parâmetro	Descrição
volume.beta.kubernetes.io/storage-class	Tipos de disco EVS. O valor está em minúsculas.
failure-domain.beta.kubernetes.io/region	Região onde o cluster está localizado.

Parâmetro	Descrição
failure-domain.beta.kubernetes.io/zone	AZ onde o volume do EVS é criado. Deve ser a mesma que a AZ planejada para a carga de trabalho.
storage	Capacidade de armazenamento na unidade de Gi.
accessModes	Modo de leitura/gravação do volume. Você pode definir esse parâmetro para **ReadWriteMany** (volume compartilhado) e **ReadWriteOnce** (volume não compartilhado).

Passo 3 Execute o seguinte comando para criar uma PVC.

kubectl create -f pvc-evs-auto-example.yaml

Depois que o comando é executado, um disco EVS é criado na partição onde o cluster está localizado. Escolha **Storage** > **EVS** para exibir o disco EVS. Em alternativa, pode ver o disco EVS com base no nome do volume no console do EVS.

----Fim

16.3.3 (kubectl) Criação de um PV a partir de um disco EVS existente

Observações e restrições

O exemplo de configuração a seguir se aplica a clusters do Kubernetes 1.13 ou anterior.

Procedimento

Passo 1 Efetue logon no console do EVS, crie um disco EVS e registre o ID do volume, a capacidade e o tipo de disco do disco EVS.

Passo 2 Use o kubectl para se conectar ao cluster. Para mais detalhes, consulte **Conexão a um cluster usando o kubectl**.

Passo 3 Crie dois arquivos YAML para criar o PV (PersistentVolume) e a PVC (PersistentVolumeClaim). Suponha que os nomes dos arquivos sejam **pv-evs-example.yaml** e **pvc-evs-example.yaml**.

touch pv-evs-example.yaml pvc-evs-example.yaml

Versão do cluster do Kubernetes	Descrição	Exemplo de YAML
1.11.7 ≤ versão K8s ≤ 1.13	Clusters da v1.11.7 a v1.13	**Exemplo de YAML**
1.11 ≤ versão K8s < 1.11.7	Clusters da v1.11 a v1.11.7	**Exemplo de YAML**
Versão K8s = 1.9	Clusters da v1.9	**Exemplo de YAML**

Clusters da v1.11.7 a v1.13

- **Exemplo de arquivo YAML para o PV:**

```
apiVersion: v1
kind: PersistentVolume
metadata:
  labels:
    failure-domain.beta.kubernetes.io/region: ap-southeast-1
    failure-domain.beta.kubernetes.io/zone: ap-southeast-1a
  annotations:
    pv.kubernetes.io/provisioned-by: flexvolume-huawei.com/fuxivol
  name: pv-evs-example
spec:
  accessModes:
  - ReadWriteOnce
  capacity:
    storage: 10Gi
  claimRef:
    apiVersion: v1
    kind: PersistentVolumeClaim
    name: pvc-evs-example
    namespace: default
  flexVolume:
    driver: huawei.com/fuxivol
    fsType: ext4
    options:
      disk-mode: SCSI
      fsType: ext4
      volumeID: 0992dbda-6340-470e-a74e-4f0db288ed82
  persistentVolumeReclaimPolicy: Delete
  storageClassName: sas
```

Tabela 16-10 Parâmetros principais

Parâmetro	Descrição
failure-domain.beta.kubernetes.io/region	Região onde o cluster está localizado.
failure-domain.beta.kubernetes.io/zone	AZ onde o volume do EVS é criado. Deve ser a mesma que a AZ planejada para a carga de trabalho.
storage	Capacidade de volume do SVE na unidade de Gi.
storageClassName	Tipos de disco EVS. Valores compatíveis: High I/O (SAS) and Ultra-high I/O (SSD)
driver	Driver de armazenamento. Para discos EVS, defina este parâmetro como **huawei.com/fuxivol**.
volumeID	ID do volume do disco EVS. Para obter o ID do volume, entre no console do CCE, escolha **Resource Management** > **Storage**, clique no nome da PVC na página de guia **EVS** e copie o ID da PVC na página de detalhes da PVC.

Parâmetro	Descrição
disk-mode	Tipo de dispositivo do disco EVS. O valor é **VBD** ou **SCSI**. Para clusters do CCE anteriores a v1.11.7, você não precisa definir esse campo. O valor padrão é **VBD**. Esse campo é obrigatório para clusters do CCE de v1.11.7 a v1.13 que usam Linux x86. Como os volumes do EVS fornecidos dinamicamente por uma PVC são criados a partir de discos SCSI EVS, é aconselhável escolher **SCSI** ao criar volumes manualmente (PVs estáticos). Volumes no modo VBD ainda podem ser usados após upgrades de cluster.
spec.claimRef.apiVersion	O valor é fixado em **v1**.
spec.claimRef.kind	O valor é fixado em **PersistentVolumeClaim**.
spec.claimRef.name	Nome da PVC. O valor é o mesmo que o nome da PVC criada na próxima etapa.
spec.claimRef.namespace	Namespace da PVC. O valor é o mesmo que o namespace da PVC criada na próxima etapa.

- **Exemplo de arquivo YAML para a PVC:**

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    volume.beta.kubernetes.io/storage-class: sas
    volume.beta.kubernetes.io/storage-provisioner: flexvolume-huawei.com/
fuxivol
  labels:
    failure-domain.beta.kubernetes.io/region: ap-southeast-1
    failure-domain.beta.kubernetes.io/zone: ap-southeast-1a
  name: pvc-evs-example
  namespace: default
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
  volumeName: pv-evs-example
```

Tabela 16-11 Parâmetros principais

Parâmetro	Descrição
volume.beta.kubernetes.io/storage-class	Classe de armazenamento, que deve ser a mesma do PV existente.
volume.beta.kubernetes.io/storage-provisioner	O campo deve ser definido como **flexvolume-huawei.com/fuxivol**.

Parâmetro	Descrição
failure-domain.beta.kubernetes.io/region	Região onde o cluster está localizado.
failure-domain.beta.kubernetes.io/zone	AZ onde o volume do EVS é criado. Deve ser a mesma que a AZ planejada para a carga de trabalho.
storage	Capacidade solicitada na PVC, em Gi. O valor deve ser o mesmo que o tamanho de armazenamento do PV existente.
volumeName	Nome do PV.

Clusters da v1.11 a v1.11.7

- **Exemplo de arquivo YAML para o PV:**

```
apiVersion: v1
kind: PersistentVolume
metadata:
  labels:
    failure-domain.beta.kubernetes.io/region: ap-southeast-1
    failure-domain.beta.kubernetes.io/zone:  ap-southeast-1a
  name: pv-evs-example
spec:
  accessModes:
  - ReadWriteOnce
  capacity:
    storage: 10Gi
  flexVolume:
    driver: huawei.com/fuxivol
    fsType: ext4
    options:
      fsType: ext4
      volumeID: 0992dbda-6340-470e-a74e-4f0db288ed82
  persistentVolumeReclaimPolicy: Delete
  storageClassName: sas
```

Tabela 16-12 Parâmetros principais

Parâmetro	Descrição
failure-domain.beta.kubernetes.io/region	Região onde o cluster está localizado.
failure-domain.beta.kubernetes.io/zone	AZ onde o volume do EVS é criado. Deve ser a mesma que a AZ planejada para a carga de trabalho.
storage	Capacidade de volume do SVE na unidade de Gi.
storageClassName	Tipos de disco EVS. Valores compatíveis: High I/O (SAS) and Ultra-high I/O (SSD)

Parâmetro	Descrição
driver	Driver de armazenamento. Para discos EVS, defina este parâmetro como **huawei.com/fuxivol**.
volumeID	ID do volume do disco EVS. Para obter o ID do volume, entre no console do CCE, escolha **Resource Management** > **Storage**, clique no nome da PVC na página de guia **EVS** e copie o ID da PVC na página de detalhes da PVC.
disk-mode	Tipo de dispositivo do disco EVS. O valor é **VBD** ou **SCSI**. Para clusters do CCE anteriores a v1.11.7, você não precisa definir esse campo. O valor padrão é **VBD**. Esse campo é obrigatório para clusters do CCE de v1.11.7 a v1.13 que usam Linux x86. Como os volumes do EVS fornecidos dinamicamente por uma PVC são criados a partir de discos SCSI EVS, é aconselhável escolher **SCSI** ao criar volumes manualmente (PVs estáticos). Volumes no modo VBD ainda podem ser usados após upgrades de cluster.

- **Exemplo de arquivo YAML para a PVC:**

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    volume.beta.kubernetes.io/storage-class: sas
    volume.beta.kubernetes.io/storage-provisioner: flexvolume-huawei.com/
fuxivol
  labels:
    failure-domain.beta.kubernetes.io/region: ap-southeast-1
    failure-domain.beta.kubernetes.io/zone: ap-southeast-1a
  name: pvc-evs-example
  namespace: default
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
  volumeName: pv-evs-example
```

Tabela 16-13 Parâmetros principais

Parâmetro	Descrição
volume.beta.kubernetes.io/storage-class	Classe de armazenamento. O valor pode ser **sas** ou **ssd**. O valor deve ser o mesmo do PV existente.

Parâmetro	Descrição
volume.beta.kubernetes.io/storage-provisioner	O campo deve ser definido como **flexvolume-huawei.com/fuxivol**.
failure-domain.beta.kubernetes.io/region	Região onde o cluster está localizado.
failure-domain.beta.kubernetes.io/zone	AZ onde o volume do EVS é criado. Deve ser a mesma que a AZ planejada para a carga de trabalho.
storage	Capacidade solicitada na PVC, em Gi. O valor deve ser o mesmo que o tamanho de armazenamento do PV existente.
volumeName	Nome do PV.

Clusters da v1.9

- **Exemplo de arquivo YAML para o PV:**

```
apiVersion: v1
kind: PersistentVolume
metadata:
  labels:
    failure-domain.beta.kubernetes.io/region: ap-southeast-1
    failure-domain.beta.kubernetes.io/zone:  ap-southeast-1a
  name: pv-evs-example
  namespace: default
spec:
  accessModes:
  - ReadWriteOnce
  capacity:
    storage: 10Gi
  flexVolume:
    driver: huawei.com/fuxivol
    fsType: ext4
    options:
      fsType: ext4
      kubernetes.io/namespace: default
      volumeID: 0992dbda-6340-470e-a74e-4f0db288ed82
  persistentVolumeReclaimPolicy: Delete
  storageClassName: sas
```

Tabela 16-14 Parâmetros principais

Parâmetro	Descrição
failure-domain.beta.kubernetes.io/region	Região onde o cluster está localizado.
failure-domain.beta.kubernetes.io/zone	AZ onde o volume do EVS é criado. Deve ser a mesma que a AZ planejada para a carga de trabalho.
storage	Capacidade de volume do SVE na unidade de Gi.

Parâmetro	Descrição
storageClassName	Tipos de disco EVS. Valores compatíveis: High I/O (SAS) and Ultra-high I/O (SSD)
driver	Driver de armazenamento. Para discos EVS, defina este parâmetro como **huawei.com/fuxivol**.
volumeID	ID do volume do disco EVS. Para obter o ID do volume, entre no console do CCE, escolha **Resource Management** > **Storage**, clique no nome da PVC na página de guia **EVS** e copie o ID da PVC na página de detalhes da PVC.
disk-mode	Tipo de dispositivo do disco EVS. O valor é **VBD** ou **SCSI**. Para clusters do CCE anteriores a v1.11.7, você não precisa definir esse campo. O valor padrão é **VBD**. Esse campo é obrigatório para clusters do CCE de v1.11.7 a v1.13 que usam Linux x86. Como os volumes do EVS fornecidos dinamicamente por uma PVC são criados a partir de discos SCSI EVS, é aconselhável escolher **SCSI** ao criar volumes manualmente (PVs estáticos). Volumes no modo VBD ainda podem ser usados após upgrades de cluster.

- **Exemplo de arquivo YAML para a PVC:**

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    volume.beta.kubernetes.io/storage-class: sas
    volume.beta.kubernetes.io/storage-provisioner: flexvolume-huawei.com/
fuxivol
  labels:
    failure-domain.beta.kubernetes.io/region: ap-southeast-1
    failure-domain.beta.kubernetes.io/zone: ap-southeast-1a
  name: pvc-evs-example
  namespace: default
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
  volumeName: pv-evs-example
  volumeNamespace: default
```


Tabela 16-15 Parâmetros principais

Parâmetro	Descrição
volume.beta.kubernetes.io/storage-class	Classe de armazenamento, que deve ser a mesma do PV existente.
volume.beta.kubernetes.io/storage-provisioner	O campo deve ser definido como **flexvolume-huawei.com/fuxivol**.
failure-domain.beta.kubernetes.io/region	Região onde o cluster está localizado.
failure-domain.beta.kubernetes.io/zone	AZ onde o volume do EVS é criado. Deve ser a mesma que a AZ planejada para a carga de trabalho.
storage	Capacidade solicitada na PVC, em Gi. O valor deve ser o mesmo que o tamanho de armazenamento do PV existente.
volumeName	Nome do PV.

Passo 4 Crie um PV.

kubectl create -f pv-evs-example.yaml

Passo 5 Crie uma PVC.

kubectl create -f pvc-evs-example.yaml

Depois que a operação for bem-sucedida, escolha **Resource Management** > **Storage** para exibir a PVC criada. Você também pode visualizar o disco EVS por nome no console do EVS.

Passo 6 (Opcional) Adicione os metadados associados ao cluster para garantir que o disco do EVS associado ao PV estático montado não seja excluído quando o nó ou o cluster for excluído.

⚠ CUIDADO

Se você pular esta etapa neste exemplo ou ao criar um PV ou PVC estático, certifique-se de que o disco EVS associado ao PV estático tenha sido desvinculado do nó antes de excluir o nó.

1. Obtenha o token do locatário. Para obter detalhes, consulte **Obtenção de um token de usuário**.
2. Obtenha o endereço de acesso EVS **EVS_ENDPOINT**. Para obter detalhes, consulte **Regiões e pontos de extremidade**.
3. Adicione os metadados associados ao cluster ao disco EVS que suporta o PV estático.

```
curl -X POST ${EVS_ENDPOINT}/v2/${project_id}/volumes/${volume_id}/metadata --
insecure \
    -d '{"metadata":{"cluster_id": "${cluster_id}", "namespace": "$
{pvc_namespace}"}}' \
    -H 'Accept:application/json' -H 'Content-Type:application/
json;charset=utf8' \
    -H 'X-Auth-Token:${TOKEN}'
```


Tabela 16-16 Parâmetros principais

Parâmetro	Descrição
EVS_ENDPOINT	Endereço de acesso ao EVS. Defina este parâmetro para o valor obtido em **2**.
project_id	ID do projeto.
volume_id	ID do disco EVS associado. Defina este parâmetro como **volume_id** do PV estático a ser criado. Você também pode efetuar logon no console do EVS, clicar no nome do disco EVS a ser importado e obter o ID em **Summary** na página de detalhes do disco, conforme mostrado em **Figura 16-2**.
cluster_id	ID do cluster onde o PV do EVS será criado. No console do CCE, escolha **Resource Management** > **Clusters**. Clique no nome do cluster a ser associado. Na página de detalhes do cluster, obtenha o ID do cluster, conforme mostrado em **Figura 16-3**.
pvc_namespace	Namespace onde a PVC deve ser vinculada.
TOKEN	Token do usuário. Defina este parâmetro para o valor obtido em **1**.

Figura 16-2 Obter o ID do disco

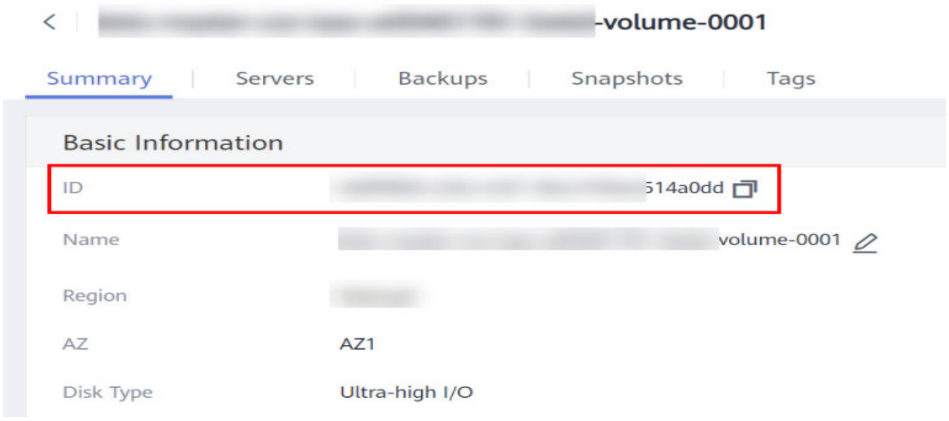
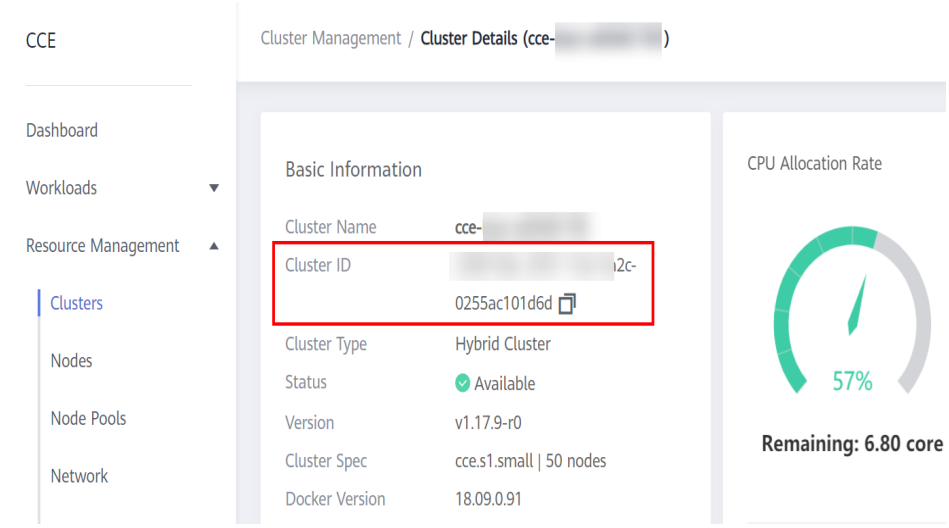


Figura 16-3 Obter o ID do cluster



Por exemplo, execute os seguintes comandos:

```
curl -X POST https://evs.ap-
southeast-1.myhuaweicloud.com:443/v2/060576866680d5762f52c0150e726aa7/volumes/
69c9619d-174c-4c41-837e-31b892604e14/metadata --insecure \
    -d '{"metadata":{"cluster_id": "71e8277e-80c7-11ea-925c-0255ac100442",
"namespace": "default"}}' \
    -H 'Accept:application/json' -H 'Content-Type:application/
json;charset=utf8' \
    -H 'X-Auth-Token:MIIPe******IsIm1ldG
```

Depois que a solicitação for executada, execute os seguintes comandos para verificar se o disco EVS foi associado aos metadados do cluster:

```
curl -X GET ${EVS_ENDPOINT}/v2/${project_id}/volumes/${volume_id}/metadata --
insecure \
    -H 'X-Auth-Token:${TOKEN}'
```

Por exemplo, execute os seguintes comandos:

```
curl -X GET https://evs.ap-
southeast-1.myhuaweicloud.com/v2/060576866680d5762f52c0150e726aa7/volumes/
69c9619d-174c-4c41-837e-31b892604e14/metadata --insecure \
    -H 'X-Auth-Token:MIIPeAYJ***9t1c31ASaQ=='
```

A saída do comando exibe os metadados atuais do disco EVS.

```
{
    "metadata": {
        "namespace": "default",
        "cluster_id": "71e8277e-80c7-11ea-925c-0255ac100442",
        "hw:passthrough": "true"
    }
}
```

----Fim

16.3.4 (kubectl) Criação de um pod montado com um volume do EVS

Observações e restrições

O exemplo de configuração a seguir se aplica a clusters do Kubernetes 1.13 ou anterior.

Procedimento

Passo 1 Use o kubectl para se conectar ao cluster. Para mais detalhes, consulte **Conexão a um cluster usando o kubectl**.

Passo 2 Execute os seguintes comandos para configurar o arquivo **evs-deployment-example.yaml**, que é usado para criar uma Implementação.

touch evs-deployment-example.yaml

vi evs-deployment-example.yaml

Exemplo de montagem de um volume do EVS em uma Implementação (volume compartilhado, baseada em PVC):

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: evs-deployment-example
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: evs-deployment-example
  template:
    metadata:
      labels:
        app: evs-deployment-example
    spec:
      containers:
      - image: nginx
        name: container-0
        volumeMounts:
        - mountPath: /tmp
          name: pvc-evs-example
      imagePullSecrets:
        - name: default-secret
      restartPolicy: Always
      volumes:
      - name: pvc-evs-example
        persistentVolumeClaim:
          claimName: pvc-evs-auto-example
```

Tabela 16-17 Parâmetros principais

Parâmetro primário	Parâmetro	Descrição
spec.template.spec.containers.volumeMounts	name	Nome do volume montado para o contêiner.
spec.template.spec.containers.volumeMounts	mountPath	Caminho de montagem do contêiner. Neste exemplo, o volume é montado no diretório **/tmp**.
spec.template.spec.volumes	name	Nome do volume.
spec.template.spec.volumes.persistentVolumeClaim	claimName	Nome de uma PVC existente.

> **NOTA**
>
> **spec.template.spec.containers.volumeMounts.name** e **spec.template.spec.volumes.name** devem ser consistentes porque eles têm uma relação de mapeamento.

Montagem de um volume do EVS num StatefulSet (baseado em modelo de PVC, não compartilhado):

Exemplo de YAML:

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: deploy-evs-sas-in
spec:
  replicas: 1
  selector:
    matchLabels:
      app: deploy-evs-sata-in
  template:
    metadata:
      labels:
        app: deploy-evs-sata-in
        failure-domain.beta.kubernetes.io/region: ap-southeast-1
        failure-domain.beta.kubernetes.io/zone: ap-southeast-1a
    spec:
      containers:
        - name: container-0
          image: 'nginx:1.12-alpine-perl'
          volumeMounts:
            - name: bs-sas-mountoptionpvc
              mountPath: /tmp
      imagePullSecrets:
        - name: default-secret
  volumeClaimTemplates:
    - metadata:
        name: bs-sas-mountoptionpvc
        annotations:
          volume.beta.kubernetes.io/storage-class: sas
          volume.beta.kubernetes.io/storage-provisioner: flexvolume-huawei.com/
fuxivol
      spec:
        accessModes:
          - ReadWriteOnce
        resources:
          requests:
            storage: 10Gi
  serviceName: wwww
```

Tabela 16-18 Parâmetros principais

Parâmetro primário	Parâmetro	Descrição
metadata	name	Nome da carga de trabalho criada.
spec.template.spec.containers	image	Imagem da carga de trabalho.
spec.template.spec.containers.volumeMount	mountPath	Caminho de montagem do contêiner. Neste exemplo, o volume é montado no diretório **/tmp**.
spec	serviceName	Serviço correspondente à carga de trabalho. Para obter detalhes sobre como criar um Serviço, consulte **Criação de um StatefulSet**.

NOTA

spec.template.spec.containers.volumeMounts.name e **spec.volumeClaimTemplates.metadata.name** devem ser consistentes porque têm uma relação de mapeamento.

Passo 3 Execute o seguinte comando para criar o pod:

kubectl create -f evs-deployment-example.yaml

Após a conclusão da criação, entre no console do CCE. No painel de navegação, escolha **Resource Management** > **Storage** > **EVS**. Em seguida, clique no nome da PVC. Na página de detalhes da PVC, você pode ver a relação de vinculação entre o volume do EVS e a PVC.

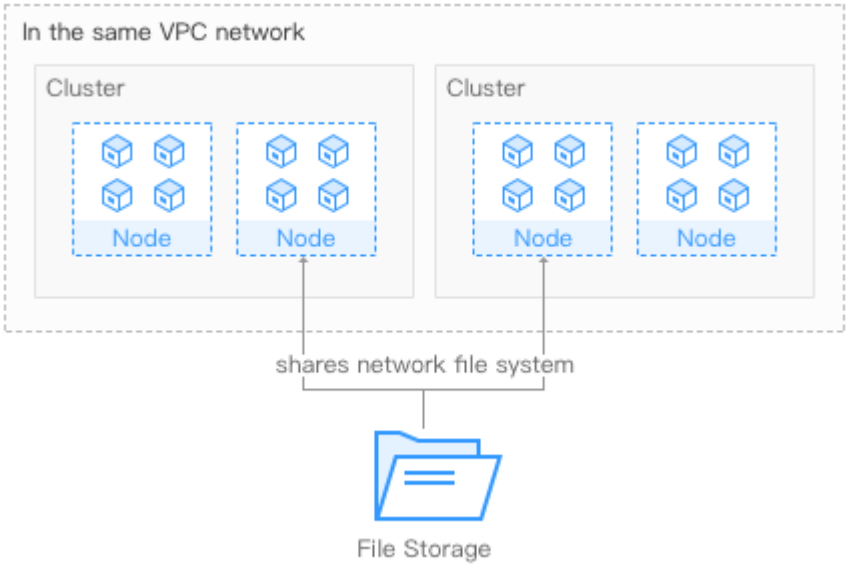
----Fim

16.4 Usar sistemas de arquivos do SFS Turbo como volumes de armazenamento

16.4.1 Visão geral

O CCE permite montar um volume criado a partir de um sistema de arquivos do SFS Turbo em um contêiner para armazenar dados persistentemente. Provisionado sob demanda e rápido, o SFS Turbo é adequado para cenários de OA corporativo, DevOps e microsserviços de contêineres.

Figura 16-4 Montagem de volumes do SFS Turbo no CCE



Descrição

- **Protocolos de arquivo padrão**: você pode montar sistemas de arquivos como volumes em servidores, o mesmo que usar diretórios locais.
- **Compartilhamento de dados**: o mesmo sistema de arquivos pode ser montado em vários servidores, para que os dados possam ser compartilhados.

- **Rede privada**: o usuário pode acessar dados apenas em redes privadas de data centers.
- **Isolamento de dados**: o serviço de armazenamento na nuvem fornece armazenamento exclusivo de arquivos na nuvem, que oferece isolamento de dados e garante o desempenho de IOPS.
- **Casos de uso**: Implementações/StatefulSets no modo ReadWriteMany, DaemonSets e tarefas criadas para sites de alto tráfego, armazenamento de logs, DevOps e aplicações de OA corporativo

16.4.2 (kubectl) Criação de um PV a partir de um sistema de arquivos do SFS Turbo existente

Cenário

O CCE permite que você use um sistema de arquivos do SFS Turbo existente para criar um PersistentVolume (PV). Após a criação ser bem-sucedida, você pode criar uma PersistentVolumeClaim (PVC) e vinculá-la ao PV.

Observações e restrições

O exemplo de configuração a seguir se aplica a clusters do Kubernetes 1.13 ou anterior.

Procedimento

Passo 1 Faça logon no console do SFS, crie um sistema de arquivos e registre o ID do sistema de arquivos, o caminho compartilhado e a capacidade.

Passo 2 Use o kubectl para se conectar ao cluster. Para mais detalhes, consulte **Conexão a um cluster usando o kubectl**.

Passo 3 Crie dois arquivos YAML para criar o PV e a PVC. Suponha que os nomes dos arquivos sejam **pv-efs-example.yaml** e **pvc-efs-example.yaml**.

touch pv-efs-example.yaml pvc-efs-example.yaml

- **Exemplo de arquivo YAML para o PV:**

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-efs-example
  annotations:
    pv.kubernetes.io/provisioned-by: flexvolume-huawei.com/fuxiefs
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 100Gi
  claimRef:
    apiVersion: v1
    kind: PersistentVolumeClaim
    name: pvc-efs-example
    namespace: default
  flexVolume:
    driver: huawei.com/fuxiefs
    fsType: efs
    options:
      deviceMountPath: <your_deviceMountPath>  # Shared storage path of your
SFS Turbo file.
      fsType: efs
      volumeID: 8962a2a2-a583-4b7f-bb74-fe76712d8414
```



```
  persistentVolumeReclaimPolicy: Delete
  storageClassName: efs-standard
```

Tabela 16-19 Parâmetros principais

Parâmetro	Descrição
driver	Driver de armazenamento usado para montar o volume. Defina-o para **huawei.com/fuxiefs**.
deviceMountPath	Caminho compartilhado do volume do SFS Turbo.
volumeID	ID do volume do SFS Turbo. Para obter o ID, entre no console do CCE, escolha **Resource Management** > **Storage**, clique no nome da PVC na página de guia **SFS Turbo** e copie o ID da PVC na página de detalhes da PVC.
storage	Tamanho do sistema de arquivos.
storageClassName	Tipo de volume suportado pelo SFS Turbo. O valor pode ser **efs-standard** e **efs-performance**. Atualmente, o SFS Turbo não oferece suporte à criação dinâmica; portanto, esse parâmetro não é usado por enquanto.
spec.claimRef.apiVersion	O valor é fixado em **v1**.
spec.claimRef.kind	O valor é fixado em **PersistentVolumeClaim**.
spec.claimRef.name	O valor é o mesmo que o nome da PVC criada na próxima etapa.
spec.claimRef.namespace	O valor é o mesmo que o namespace da PVC criada na próxima etapa.

- **Exemplo de arquivo YAML para a PVC:**

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    volume.beta.kubernetes.io/storage-class: efs-standard
    volume.beta.kubernetes.io/storage-provisioner: flexvolume-huawei.com/
fuxiefs
  name: pvc-efs-example
  namespace: default
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 100Gi
  volumeName: pv-efs-example
```


Tabela 16-20 Parâmetros principais

Parâmetro	Descrição
volume.beta.kubernetes.io/storage-class	Modo de leitura/gravação suportado pelo SFS Turbo. O valor pode ser **efs-standard** ou **efs-performance**. O valor deve ser o mesmo do PV existente.
volume.beta.kubernetes.io/storage-provisioner	O campo deve ser definido como **flexvolume-huawei.com/fuxiefs**.
storage	Capacidade de armazenamento, na unidade de Gi. O valor deve ser o mesmo que o tamanho de armazenamento do PV existente.
volumeName	Nome do PV.

NOTA

A VPC à qual o sistema de arquivos do SFS Turbo pertence deve ser a mesma VPC da VM do ECS planejada para a carga de trabalho. As portas 111, 445, 2049, 2051 e 20048 devem ser habilitadas nos grupos de segurança.

Passo 4 Crie o PV.

kubectl create -f pv-efs-example.yaml

Passo 5 Crie a PVC.

kubectl create -f pvc-efs-example.yaml

----Fim

16.4.3 (kubectl) Criação de uma Implementação montada com um volume do SFS Turbo

Cenário

Depois que um volume do SFS Turbo é criado ou importado para o CCE, você pode montar o volume em uma carga de trabalho.

Observações e restrições

O exemplo de configuração a seguir se aplica a clusters do Kubernetes 1.13 ou anterior.

Procedimento

Passo 1 Use o kubectl para se conectar ao cluster. Para mais detalhes, consulte **Conexão a um cluster usando o kubectl**.

Passo 2 Execute os seguintes comandos para configurar o arquivo **efs-deployment-example.yaml**, que é usado para criar uma Implementação:

touch efs-deployment-example.yaml

vi efs-deployment-example.yaml

Exemplo de montagem de um volume do SFS Turbo em uma Implementação (volume compartilhado baseado em PVC):

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: efs-deployment-example                                  # Workload name
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: efs-deployment-example
  template:
    metadata:
      labels:
        app: efs-deployment-example
    spec:
      containers:
      - image: nginx
        name: container-0
        volumeMounts:
        - mountPath: /tmp                                       # Mount path
          name: pvc-efs-example
      restartPolicy: Always
      imagePullSecrets:
        - name: default-secret
      volumes:
      - name: pvc-efs-example
        persistentVolumeClaim:
          claimName: pvc-sfs-auto-example                       # PVC name
```

Tabela 16-21 Parâmetros principais

Parâmetro	Descrição
name	Nome da Implementação criada.
app	Nome da aplicação em execução na Implementação.
mountPath	Monte o caminho no contêiner. Neste exemplo, o caminho de montagem é **/tmp**.

NOTA

spec.template.spec.containers.volumeMounts.name e **spec.template.spec.volumes.name** devem ser consistentes porque eles têm uma relação de mapeamento.

Passo 3 Execute o seguinte comando para criar o pod:

kubectl create -f efs-deployment-example.yaml

Após a conclusão da criação, escolha **Storage** > **SFS Turbo** no console do CCE e clique no nome da PVC. Na página de detalhes da PVC, você pode exibir a relação de vinculação entre o SFS Turbo e a PVC.

----Fim

16.4.4 (kubectl) Criação de um StatefulSet montado com um volume do SFS Turbo

Cenário

O CCE permite que você use um volume do SFS Turbo existente para criar um StatefulSet.

Observações e restrições

O exemplo de configuração a seguir se aplica a clusters do Kubernetes 1.13 ou anterior.

Procedimento

Passo 1 Crie um volume do SFS Turbo e registre o nome do volume.

Passo 2 Use o kubectl para se conectar ao cluster. Para mais detalhes, consulte **Conexão a um cluster usando o kubectl**.

Passo 3 Crie um arquivo YAML para criar a carga de trabalho. Suponha que o nome do arquivo é **efs-statefulset-example.yaml**.

touch efs-statefulset-example.yaml

vi efs-statefulset-example.yaml

Exemplo de YAML:

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: efs-statefulset-example
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: efs-statefulset-example
  template:
    metadata:
      annotations:
        metrics.alpha.kubernetes.io/custom-endpoints:
'[{"api":"","path":"","port":"","names":""}]'
        pod.alpha.kubernetes.io/initialized: 'true'
      labels:
        app: efs-statefulset-example
    spec:
      containers:
        - image: 'nginx:1.0.0'
          name: container-0
          resources:
            requests: {}
            limits: {}
          env:
            - name: PAAS_APP_NAME
              value: efs-statefulset-example
            - name: PAAS_NAMESPACE
              value: default
            - name: PAAS_PROJECT_ID
              value: b18296881cc34f929baa8b9e95abf88b
          volumeMounts:
            - name: efs-statefulset-example
              mountPath: /tmp
              readOnly: false
```



```
            subPath: ''
    imagePullSecrets:
      - name: default-secret
    terminationGracePeriodSeconds: 30
    volumes:
      - persistentVolumeClaim:
          claimName: cce-efs-import-jnr481gm-3y5o
        name: efs-statefulset-example
    affinity: {}
    tolerations:
      - key: node.kubernetes.io/not-ready
        operator: Exists
        effect: NoExecute
        tolerationSeconds: 300
      - key: node.kubernetes.io/unreachable
        operator: Exists
        effect: NoExecute
        tolerationSeconds: 300
podManagementPolicy: OrderedReady
serviceName: test
updateStrategy:
  type: RollingUpdate
```

Tabela 16-22 Parâmetros principais

Parâmetro	Descrição
replicas	Número de pods.
name	Nome da carga de trabalho criada.
image	Imagem utilizada pela carga de trabalho.
mountPath	Caminho de montagem no contêiner.
serviceName	Serviço correspondente à carga de trabalho. Para obter detalhes sobre como criar um Serviço, consulte **Criação de um StatefulSet**.
claimName	Nome de uma PVC existente.

NOTA

spec.template.spec.containers.volumeMounts.name e **spec.template.spec.volumes.name** devem ser consistentes porque eles têm uma relação de mapeamento.

Passo 4 Crie o StatefulSet.

kubectl create -f efs-statefulset-example.yaml

----Fim

16.5 Uso de buckets do OBS como volumes de armazenamento

16.5.1 Visão geral

O CCE permite que você monte um volume criado de um bucket do Object Storage Service (OBS) em um contêiner para armazenar dados persistentemente. O armazenamento de objetos é comumente usado em cargas de trabalho na nuvem, análise de dados, análise de conteúdo e objetos de hotspot.

Figura 16-5 Montagem de volumes do OBS para CCE

Observações e restrições

Os contêineres seguros não suportam volumes do OBS.

Um único usuário pode criar no máximo 100 buckets do OBS no console. Se você tiver um grande número de cargas de trabalho do CCE e quiser montar um bucket do OBS para cada carga de trabalho, poderá facilmente ficar sem buckets. Nesse cenário, é recomendável usar o OBS por meio da API ou do SDK do OBS e não montar intervalos do OBS na carga de trabalho no console.

Classe de armazenamento

O armazenamento de objetos oferece três classes de armazenamento, Standard, Infrequent Access e Archive, para atender a diferentes requisitos de desempenho e custos de armazenamento.

- A classe de armazenamento Standard apresenta baixa latência de acesso e alta taxa de transferência. É, portanto, aplicável ao armazenar um grande número de arquivos quentes (frequentemente acessados todos os meses) ou arquivos pequenos (menos de 1 MB). Os cenários de aplicações incluem análise de Big Data, aplicativos móveis, vídeos quentes e processamento de imagens em mídias sociais.
- A classe de armazenamento Infrequent Access é ideal para armazenar dados que são acessados semi-frequentemente (menos de 12 vezes por ano), com requisitos de resposta rápida. Os cenários de aplicações incluem sincronização ou compartilhamento de arquivos e backup em nível empresarial. Ela oferece a mesma durabilidade, latência de acesso e taxa de transferência que a classe de armazenamento Standard, mas a um custo

menor. No entanto, a classe de armazenamento Infrequent Access tem menor disponibilidade do que a classe de armazenamento padrão.

- A classe de armazenamento Archive é adequada para arquivar dados raramente acessados (em média, uma vez por ano). Os cenários de aplicações incluem arquivamento de dados e backups de dados de longo prazo. A classe de armazenamento Archive é segura e durável a um baixo custo acessível, que pode ser usada para substituir bibliotecas de fitas. No entanto, pode levar horas para restaurar dados da classe de armazenamento Archive.

Descrição

- **APIs padrão**: com as APIs RESTful HTTP, o OBS permite que você use ferramentas de cliente ou ferramentas de terceiros para acessar o armazenamento de objetos.
- **Compartilhamento de dados**: servidores, dispositivos incorporados e dispositivos IoT podem usar o mesmo caminho para acessar dados de objetos compartilhados no OBS.
- **Redes públicas/privadas**: o OBS permite que os dados sejam acessados a partir de redes públicas para atender aos requisitos de aplicações da Internet.
- **Capacidade e desempenho**: sem limite de capacidade; alto desempenho (latência de I/O de leitura/gravação em até 10 ms).
- **Casos de uso**: as Implementações/StatefulSets no modo ReadOnlyMany e tarefas criadas para análise de Big Data, hospedagem de sites estáticos, vídeo on-line sob demanda (VoD), sequenciamento de genes, vigilância por vídeo inteligente, backup e arquivamento e caixas de nuvem empresariais (discos da Web). Você pode criar armazenamento de objetos usando o console do OBS, as ferramentas e os SDKs.

Referência

Os clusters do CCE também podem ser montados com buckets OBS de locatários de terceiros, incluindo sistemas de arquivos paralelos OBS (de preferência) e buckets de objetos do OBS. Para obter detalhes, consulte **Montagem de um bucket de armazenamento de objetos de um locatário de terceiros**.

16.5.2 (kubectl) Criação automática de um volume do OBS

Cenário

Durante o uso do OBS, os baldes esperados do OBS podem ser criados e montados automaticamente como volumes. Atualmente, os buckets OBS de acesso padrão e infrequente são suportados, que correspondem a **obs-standard** e **obs-standard-ia**, respectivamente.

Observações e restrições

O exemplo de configuração a seguir se aplica a clusters do Kubernetes 1.13 ou anterior.

Procedimento

Passo 1 Use o kubectl para se conectar ao cluster. Para mais detalhes, consulte **Conexão a um cluster usando o kubectl**.

Passo 2 Execute os comandos a seguir para configurar o arquivo **pvc-obs-auto-example.yaml**, que é usado para criar uma PVC.

touch pvc-obs-auto-example.yaml

vi pvc-obs-auto-example.yaml

Exemplo de YAML:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    volume.beta.kubernetes.io/storage-class: obs-standard  # OBS bucket type. The
value can be obs-standard (standard) or obs-standard-ia (infrequent access).
  name: pvc-obs-auto-example  # PVC name
  namespace: default
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi   # Storage capacity in the unit of Gi. For OBS buckets, this
parameter is used only for verification (fixed to 1, cannot be empty or 0). Any
value you set does not take effect for OBS buckets.
```

Tabela 16-23 Parâmetros principais

Parâmetro	Descrição
volume.beta.kubernetes.io/ storage-class	Tipo de bucket. Atualmente, **obs-standard** e **obs-standard-ia** são suportados.
name	Nome da PVC a ser criada.
accessModes	Apenas **ReadWriteMany** é suportado. **ReadWriteOnly** não é suportado.
storage	Capacidade de armazenamento na unidade de Gi. Para buckets do OBS, esse campo é usado apenas para verificação (não pode estar vazio ou 0). Seu valor é fixado em **1**, e qualquer valor que você definir não terá efeito para buckets do OBS.

Passo 3 Execute o seguinte comando para criar uma PVC:

kubectl create -f pvc-obs-auto-example.yaml

Depois que o comando é executado, um bucket do OBS é criado na VPC à qual o cluster pertence. Você pode clicar no nome do bucket em **Storage** > **OBS** para exibir o bucket ou visualizá-lo no console do OBS.

----Fim

16.5.3 (kubectl) Criação de um PV a partir de um bucket do OBS existente

Cenário

O CCE permite que você use um bucket do OBS existente para criar um PV (PersistentVolume). Você pode criar uma PVC (PersistentVolumeClaim) e ligá-la ao PV.

Observações e restrições

O exemplo de configuração a seguir se aplica a clusters do Kubernetes 1.13 ou anterior.

Procedimento

Passo 1 Faça logon no console do OBS, crie um bucket do OBS e registre o nome do bucket e a classe de armazenamento.

Passo 2 Use o kubectl para se conectar ao cluster. Para mais detalhes, consulte **Conexão a um cluster usando o kubectl**.

Passo 3 Crie dois arquivos YAML para criar o PV e a PVC. Suponha que os nomes dos arquivos sejam **pv-obs-example.yaml** e **pvc-obs-example.yaml**.

touch pv-obs-example.yaml pvc-obs-example.yaml

Versão do cluster do Kubernetes	Descrição	Exemplo de YAML
1.11 ≤ K8s versão< 1.13	Clusters da v1.11 a v1.13	**Exemplo de YAML**
K8s versão = 1.9	Clusters da v1.9	**Exemplo de YAML**

Clusters da v1.11 a v1.13

- **Exemplo de arquivo YAML para o PV:**

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-obs-example
  annotations:
    pv.kubernetes.io/provisioned-by: flexvolume-huawei.com/fuxiobs
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 1Gi
  claimRef:
    apiVersion: v1
    kind: PersistentVolumeClaim
    name: pvc-obs-example
    namespace: default
  flexVolume:
    driver: huawei.com/fuxiobs
    fsType: obs
    options:
      fsType: obs
      region: ap-southeast-1
      storage_class: STANDARD
      volumeID: test-obs
  persistentVolumeReclaimPolicy: Delete
  storageClassName: obs-standard
```


Tabela 16-24 Parâmetros principais

Parâmetro	Descrição
driver	Driver de armazenamento usado para montar o volume. Defina o driver para **huawei.com/fuxiobs** para o volume do OBS.
storage_class	Classe de armazenamento, incluindo **STANDARD** (bucket padrão) e **STANDARD_IA** (bucket de acesso infrequente).
region	Região onde o cluster está localizado.
volumeID	Nome do bucket do OBS. Para obter o ID, entre no console do CCE, escolha **Resource Management** > **Storage**, clique no nome da PVC na página de guia **OBS** e copie o ID da PVC na página da guia **PV Details**.
storage	Capacidade de armazenamento, em Gi. O valor é fixado em **1Gi**.
storageClassName	Classe de armazenamento suportada pelo OBS, incluindo **obs-standard** (bucket padrão) e **obs-standard-ia** (bucket de acesso infrequente).
spec.claimRef.apiVersion	O valor é fixado em **v1**.
spec.claimRef.kind	O valor é fixado em **PersistentVolumeClaim**.
spec.claimRef.name	O valor é o mesmo que o nome da PVC criada na próxima etapa.
spec.claimRef.namespace	O valor é o mesmo que o namespace da PVC criada na próxima etapa.

- **Exemplo de arquivo YAML para a PVC:**

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    volume.beta.kubernetes.io/storage-class: obs-standard
    volume.beta.kubernetes.io/storage-provisioner: flexvolume-huawei.com/
fuxiobs
  name: pvc-obs-example
  namespace: default
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  volumeName: pv-obs-example
```


Tabela 16-25 Parâmetros principais

Parâmetro	Descrição
volume.beta.kubernetes.io/storage-class	Classe de armazenamento suportada pelo OBS, incluindo **obs-standard** e **obs-standard-ia**.
volume.beta.kubernetes.io/storage-provisioner	Deve ser definido como **flexvolume-huawei.com/fuxiobs**.
volumeName	Nome do PV.
storage	Capacidade de armazenamento, em Gi. O valor é fixado em **1Gi**.

Clusters da v1.9

- **Exemplo de arquivo YAML para o PV:**

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-obs-example
  namespace: default
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 1Gi
  flexVolume:
    driver: huawei.com/fuxiobs
    fsType: obs
    options:
      fsType: obs
      kubernetes.io/namespace: default
      region: ap-southeast-1
      storage_class: STANDARD
      volumeID: test-obs
  persistentVolumeReclaimPolicy: Delete
  storageClassName: obs-standard
```

Tabela 16-26 Parâmetros principais

Parâmetro	Descrição
driver	Driver de armazenamento usado para montar o volume. Defina o driver para **huawei.com/fuxiobs** para o volume do OBS.
storage_class	Classe de armazenamento, incluindo **STANDARD** (bucket padrão) e **STANDARD_IA** (bucket de acesso infrequente).
region	Região onde o cluster está localizado.
volumeID	Nome do bucket do OBS. Para obter o ID, entre no console do CCE, escolha **Resource Management** > **Storage**, clique no nome da PVC na página de guia **OBS** e copie o ID da PVC na página da guia **PV Details**.

Parâmetro	Descrição
storage	Capacidade de armazenamento, em Gi. O valor é fixado em **1Gi**.
storageClassName	Classe de armazenamento suportada pelo OBS, incluindo **obs-standard** (bucket padrão) e **obs-standard-ia** (bucket de acesso infrequente).

- **Exemplo de arquivo YAML para a PVC:**

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    volume.beta.kubernetes.io/storage-class: obs-standard
    volume.beta.kubernetes.io/storage-provisioner: flexvolume-huawei.com/
fuxiobs
  name: pvc-obs-example
  namespace: default
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  volumeName: pv-obs-example
  volumeNamespace: default
```

Tabela 16-27 Parâmetros principais

Parâmetro	Descrição
volume.beta.kubernetes.io/ storage-class	Classe de armazenamento suportada pelo OBS, incluindo **obs-standard** e **obs-standard-ia**.
volume.beta.kubernetes.io/ storage-provisioner	Deve ser definido como **flexvolume-huawei.com/ fuxiobs**.
volumeName	Nome do PV.
storage	Capacidade de armazenamento, em Gi. O valor é fixado em **1Gi**.

Passo 4 Crie um PV.

kubectl create -f pv-obs-example.yaml

Passo 5 Crie uma PVC.

kubectl create -f pvc-obs-example.yaml

----Fim

16.5.4 (kubectl) Criação de uma Implementação montada com um volume do OBS

Cenário

Depois que um volume do OBS é criado ou importado para o CCE, você pode montá-lo para uma carga de trabalho.

Observações e restrições

O exemplo de configuração a seguir se aplica a clusters do Kubernetes 1.13 ou anterior.

Procedimento

Passo 1 Use o kubectl para se conectar ao cluster. Para mais detalhes, consulte **Conexão a um cluster usando o kubectl**.

Passo 2 Execute os comandos a seguir para configurar o arquivo **obs-deployment-example.yaml**, que é usado para criar um pod.

touch obs-deployment-example.yaml

vi obs-deployment-example.yaml

Exemplo de montar um volume do OBS para uma Implementação (volume compartilhado, baseada em PVC):

```
apiVersion: apps/v1
kind: Deployment
metadata:
   name: obs-deployment-example                      # Workload name
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: obs-deployment-example
  template:
    metadata:
      labels:
        app: obs-deployment-example
    spec:
      containers:
      - image: nginx
        name: container-0
        volumeMounts:
        - mountPath: /tmp                       # Mount path
          name: pvc-obs-example
      restartPolicy: Always
      imagePullSecrets:
        - name: default-secret
      volumes:
      - name: pvc-obs-example
        persistentVolumeClaim:
          claimName: pvc-obs-auto-example       # PVC name
```

Tabela 16-28 Parâmetros principais

Parâmetro	Descrição
name	Nome do pod a ser criado.

Parâmetro	Descrição
app	Nome da aplicação em execução no pod.
mountPath	Caminho de montagem no contêiner.

NOTA

spec.template.spec.containers.volumeMounts.name e **spec.template.spec.volumes.name** devem ser consistentes porque eles têm uma relação de mapeamento.

Exemplo de montar um volume do OBS para um StatefulSet (volume dedicado, baseado em modelo de PVC):

Exemplo de YAML:

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: deploy-obs-standard-in
  namespace: default
  generation: 1
  labels:
    appgroup: ''
spec:
  replicas: 1
  selector:
    matchLabels:
      app: deploy-obs-standard-in
  template:
    metadata:
      labels:
        app: deploy-obs-standard-in
      annotations:
        metrics.alpha.kubernetes.io/custom-endpoints:
'[{"api":"","path":"","port":"","names":""}]'
        pod.alpha.kubernetes.io/initialized: 'true'
    spec:
      containers:
        - name: container-0
          image: 'nginx:1.12-alpine-perl'
          env:
            - name: PAAS_APP_NAME
              value: deploy-obs-standard-in
            - name: PAAS_NAMESPACE
              value: default
            - name: PAAS_PROJECT_ID
              value: a2cd8e998dca42e98a41f596c636dbda
          resources: {}
          volumeMounts:
            - name: obs-bs-standard-mountoptionpvc
              mountPath: /tmp
          terminationMessagePath: /dev/termination-log
          terminationMessagePolicy: File
          imagePullPolicy: IfNotPresent
      restartPolicy: Always
      terminationGracePeriodSeconds: 30
      dnsPolicy: ClusterFirst
      securityContext: {}
      imagePullSecrets:
        - name: default-secret
      affinity: {}
      schedulerName: default-scheduler
  volumeClaimTemplates:
    - metadata:
```



```
        name: obs-bs-standard-mountoptionpvc
        annotations:
          volume.beta.kubernetes.io/storage-class: obs-standard
          volume.beta.kubernetes.io/storage-provisioner: flexvolume-huawei.com/
fuxiobs
      spec:
        accessModes:
          - ReadWriteMany
        resources:
          requests:
            storage: 1Gi
  serviceName: wwww
  podManagementPolicy: OrderedReady
  updateStrategy:
    type: RollingUpdate
  revisionHistoryLimit: 10
```

Tabela 16-29 Parâmetros principais

Parâmetro	Descrição
name	Nome da carga de trabalho criada.
image	Imagem da carga de trabalho.
mountPath	Caminho de montagem no contêiner. Neste exemplo, o volume é montado no diretório **/tmp**.
serviceName	Serviço correspondente à carga de trabalho. Para obter detalhes sobre como criar um Serviço, consulte **Criação de um StatefulSet**.

NOTA

spec.template.spec.containers.volumeMounts.name e **spec.volumeClaimTemplates.metadata.name** devem ser consistentes porque têm uma relação de mapeamento.

Passo 3 Execute o seguinte comando para criar o pod:

kubectl create -f obs-deployment-example.yaml

Após a conclusão da criação, escolha **Storage** > **OBS** no console do CCE e clique no nome da PVC. Na página de detalhes da PVC, você pode ver a relação de vinculação entre o serviço OBS e a PVC.

----Fim

16.5.5 (kubectl) Criação de um StatefulSet montado com um volume do OBS

Cenário

O CCE permite que você use um volume do OBS existente para criar um StatefulSet por meio de uma PVC (PersistentVolumeClaim).

Observações e restrições

O exemplo de configuração a seguir se aplica a clusters do Kubernetes 1.13 ou anterior.

Procedimento

Passo 1 Crie um volume do OBS consultando **(kubectl) Criação automática de um volume do OBS** e obtendo o nome da PVC.

Passo 2 Use o kubectl para se conectar ao cluster. Para mais detalhes, consulte **Conexão a um cluster usando o kubectl**.

Passo 3 Crie um arquivo YAML para criar a carga de trabalho. Suponha que o nome do arquivo é **obs-statefulset-example.yaml**.

touch obs-statefulset-example.yaml

vi obs-statefulset-example.yaml

Exemplo de YAML:

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: obs-statefulset-example
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: obs-statefulset-example
  serviceName: qwqq
  template:
    metadata:
      annotations:
        metrics.alpha.kubernetes.io/custom-endpoints:
'[{"api":"","path":"","port":"","names":""}]'
        pod.alpha.kubernetes.io/initialized: "true"
      creationTimestamp: null
      labels:
        app: obs-statefulset-example
    spec:
      affinity: {}
      containers:
        image: nginx:latest
        imagePullPolicy: Always
        name: container-0
        volumeMounts:
        - mountPath: /tmp
          name: pvc-obs-example
      imagePullSecrets:
      - name: default-secret
      volumes:
        - name: pvc-obs-example
          persistentVolumeClaim:
            claimName: cce-obs-demo
```

Tabela 16-30 Parâmetros principais

Parâmetro	Descrição
replicas	Número de pods.
name	Nome da carga de trabalho criada.

Parâmetro	Descrição
image	Imagem utilizada pela carga de trabalho.
mountPath	Monte o caminho no contêiner.
serviceName	Serviço correspondente à carga de trabalho. Para obter detalhes sobre como criar um Serviço, consulte **Criação de um StatefulSet**.
claimName	Nome de uma PVC existente.

Passo 4 Crie o StatefulSet.

kubectl create -f obs-statefulset-example.yaml

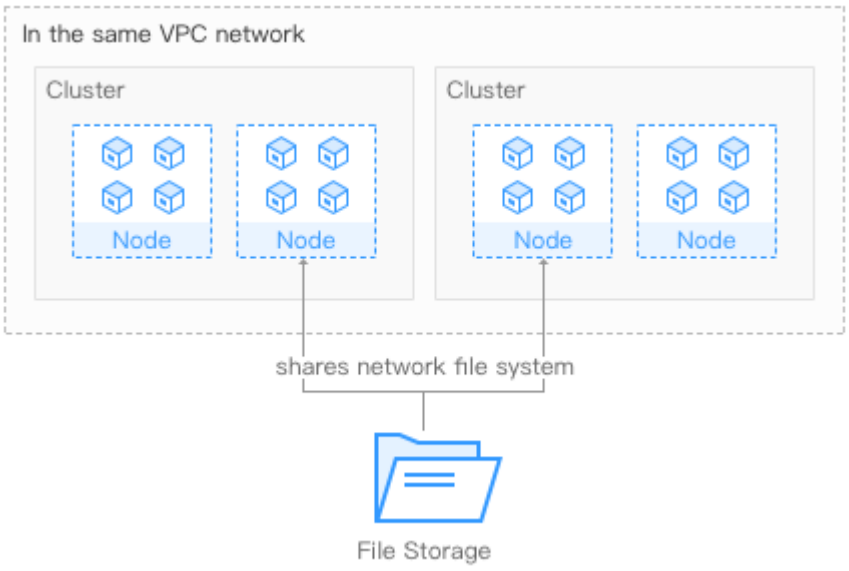
----Fim

16.6 Usar sistemas de arquivos do SFS como volumes de armazenamento

16.6.1 Visão geral

O CCE permite que você monte um volume criado a partir de um sistema de arquivos do Scalable File Service (SFS) em um contêiner para armazenar dados persistentemente. Volumes do SFS são comumente usados em cenários de ReadWriteMany, como processamento de mídia, gerenciamento de conteúdo, análise de Big Data e análise de processos de carga de trabalho.

Figura 16-6 Montagem de volumes do SFS para CCE



Descrição

- **Protocolos de arquivo padrão**: você pode montar sistemas de arquivos como volumes em servidores, o mesmo que usar diretórios locais.
- **Compartilhamento de dados**: o mesmo sistema de arquivos pode ser montado em vários servidores, para que os dados possam ser compartilhados.
- **Rede privada**: o usuário pode acessar dados apenas em redes privadas de data centers.
- **Capacidade e desempenho**: a capacidade de um único sistema de arquivos é alta (nível PB) e o desempenho é excelente (latência de I/O de leitura/gravação em nível ms).
- **Casos de uso**: Implementações/StatefulSets no modo ReadWriteMany e tarefas criadas para computação de alto desempenho (HPC), processamento de mídia, gerenciamento de conteúdo, serviços Web, análise de Big Data e análise de processos de carga de trabalho

Para obter detalhes, consulte **Visão geral do serviço SFS**.

16.6.2 (kubectl) Criação automática de um volume do SFS

NOTA

Atualmente, os sistemas de arquivos do SFS estão esgotados e as PVCs não podem ser criadas automaticamente usando a classe de armazenamento.

Observações e restrições

O exemplo de configuração a seguir se aplica a clusters do Kubernetes 1.13 ou anterior.

Procedimento

Passo 1 Use o kubectl para se conectar ao cluster. Para mais detalhes, consulte **Conexão a um cluster usando o kubectl**.

Passo 2 Execute os comandos a seguir para configurar o arquivo **pvc-sfs-auto-example.yaml**, que é usado para criar uma PVC.

touch pvc-sfs-auto-example.yaml

vi pvc-sfs-auto-example.yaml

Exemplo de arquivo YAML:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    volume.beta.kubernetes.io/storage-class: nfs-rw
  name: pvc-sfs-auto-example
  namespace: default
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 10Gi
```


Tabela 16-31 Parâmetros principais

Parâmetro	Descrição
volume.beta.kubernetes.io/ storage-class	Classe de armazenamento de arquivos. Atualmente, o tipo de protocolo de arquivo padrão (nfs-rw) é suportado.
name	Nome da PVC a ser criada.
accessModes	Apenas **ReadWriteMany** é suportado. **ReadWriteOnly** não é suportado.
storage	Capacidade de armazenamento na unidade de Gi.

Passo 3 Execute o seguinte comando para criar uma PVC:

kubectl create -f pvc-sfs-auto-example.yaml

Depois que o comando é executado, um sistema de arquivos é criado na VPC à qual o cluster pertence. Escolha **Storage** > **SFS** no console do CCE ou faça logon no console do SFS para ver o sistema de arquivos.

----Fim

16.6.3 (kubectl) Criação de um PV a partir de um sistema de arquivos do SFS existente

Observações e restrições

O exemplo de configuração a seguir se aplica a clusters do Kubernetes 1.13 ou anterior.

Procedimento

Passo 1 Faça logon no console do SFS, crie um sistema de arquivos e registre o ID do sistema de arquivos, o caminho compartilhado e a capacidade.

Passo 2 Use o kubectl para se conectar ao cluster. Para mais detalhes, consulte **Conexão a um cluster usando o kubectl**.

Passo 3 Crie dois arquivos YAML para criar o PV e a PVC. Suponha que os nomes dos arquivos sejam **pv-sfs-example.yaml** e **pvc-sfs-example.yaml**.

touch pv-sfs-example.yaml pvc-sfs-example.yaml

Versão do cluster do Kubernetes	Descrição	Exemplo de YAML
1.11 ≤ K8s versão < 1.13	Clusters da v1.11 a v1.13	**Exemplo de YAML**
K8s versão = 1.9	Clusters da v1.9	**Exemplo de YAML**

Clusters da v1.11 a v1.13

- **Exemplo de arquivo YAML para o PV:**

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-sfs-example
  annotations:
    pv.kubernetes.io/provisioned-by: flexvolume-huawei.com/fuxinfs
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 10Gi
  claimRef:
    apiVersion: v1
    kind: PersistentVolumeClaim
    name: pvc-sfs-example
    namespace: default
  flexVolume:
    driver: huawei.com/fuxinfs
    fsType: nfs
    options:
      deviceMountPath: <your_deviceMountPath>  # Shared storage path of your
file.
      fsType: nfs
      volumeID: f6976f9e-2493-419b-97ca-d7816008d91c
  persistentVolumeReclaimPolicy: Delete
  storageClassName: nfs-rw
```

Tabela 16-32 Parâmetros principais

Parâmetro	Descrição
driver	Driver de armazenamento usado para montar o volume. Defina o driver para **huawei.com/fuxinfs** para o sistema de arquivos.
deviceMountPath	Caminho compartilhado do sistema de arquivos. No console de gerenciamento, escolha **Service List** > **Storage** > **Scalable File Service**. Você pode obter o caminho compartilhado do sistema de arquivos na coluna **Mount Address**, conforme mostrado na **Figura 16-7**.
volumeID	ID do sistema de arquivos. Para obter o ID, entre no console do CCE, escolha **Resource Management** > **Storage**, clique no nome da PVC na página de guia **SFS** e copie o ID da PVC na página de detalhes da PVC.
storage	Tamanho do sistema de arquivos.
storageClassName	Modo de leitura/gravação suportado pelo sistema de arquivos. Atualmente, **nfs-rw** e o**nfs-ro** são suportados.
spec.claimRef.apiVersion	O valor é fixado em **v1**.
spec.claimRef.kind	O valor é fixado em **PersistentVolumeClaim**.
spec.claimRef.name	O valor é o mesmo que o nome da PVC criado na próxima etapa.

Parâmetro	Descrição
spec.claimRef.namespace	Namespace da PVC. O valor é o mesmo que o namespace da PVC criado na próxima etapa.

- **Exemplo de arquivo YAML para a PVC:**

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    volume.beta.kubernetes.io/storage-class: nfs-rw
    volume.beta.kubernetes.io/storage-provisioner: flexvolume-huawei.com/
fuxinfs
  name: pvc-sfs-example
  namespace: default
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 10Gi
  volumeName: pv-sfs-example
```

Tabela 16-33 Parâmetros principais

Parâmetro	Descrição
volume.beta.kubernetes.io/storage-class	Modo de leitura/gravação suportado pelo sistema de arquivos. **nfs-rw** e **nfs-ro** são suportados. O valor deve ser o mesmo do PV existente.
volume.beta.kubernetes.io/storage-provisioner	Deve ser definido como **flexvolume-huawei.com/fuxinfs**.
storage	Capacidade de armazenamento, na unidade de Gi. O valor deve ser o mesmo que o tamanho de armazenamento do PV existente.
volumeName	Nome do PV.

Clusters da v1.9

- **Exemplo de arquivo YAML para o PV:**

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-sfs-example
  namespace: default
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 10Gi
  flexVolume:
    driver: huawei.com/fuxinfs
    fsType: nfs
    options:
      deviceMountPath: <your_deviceMountPath>  # Shared storage path of your
file.
      fsType: nfs
      kubernetes.io/namespace: default
```



```
    volumeID: f6976f9e-2493-419b-97ca-d7816008d91c
 persistentVolumeReclaimPolicy: Delete
 storageClassName: nfs-rw
```

Tabela 16-34 Parâmetros principais

Parâmetro	Descrição
driver	Driver de armazenamento usado para montar o volume. Defina o driver para **huawei.com/fuxinfs** para o sistema de arquivos.
deviceMountPath	Caminho compartilhado do sistema de arquivos. No console de gerenciamento, escolha **Service List** > **Storage** > **Scalable File Service**. Você pode obter o caminho compartilhado do sistema de arquivos na coluna **Mount Address**, conforme mostrado na **Figura 16-7**.
volumeID	ID do sistema de arquivos. Para obter o ID, entre no console do CCE, escolha **Resource Management** > **Storage**, clique no nome da PVC na página de guia **SFS** e copie o ID da PVC na página de detalhes da PVC.
storage	Tamanho do sistema de arquivos.
storageClassName	Modo de leitura/gravação suportado pelo sistema de arquivos. Atualmente, **nfs-rw** e o**nfs-ro** são suportados.

- **Exemplo de arquivo YAML para a PVC:**

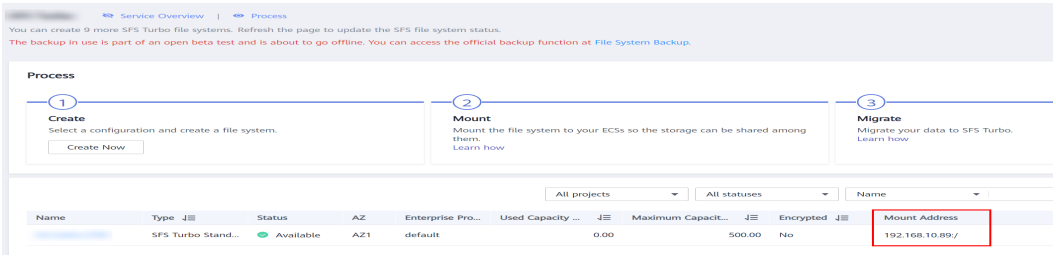
```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    volume.beta.kubernetes.io/storage-class: nfs-rw
    volume.beta.kubernetes.io/storage-provisioner: flexvolume-huawei.com/
fuxinfs
  name: pvc-sfs-example
  namespace: default
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 10Gi
  volumeName: pv-sfs-example
  volumeNamespace: default
```

Tabela 16-35 Parâmetros principais

Parâmetro	Descrição
volume.beta.kubernetes.io/ storage-class	Modo de leitura/gravação suportado pelo sistema de arquivos. **nfs-rw** e **nfs-ro** são suportados. O valor deve ser o mesmo do PV existente.
volume.beta.kubernetes.io/ storage-provisioner	O campo deve ser definido como **flexvolume-huawei.com/fuxinfs**.

Parâmetro	Descrição
storage	Capacidade de armazenamento, na unidade de Gi. O valor deve ser o mesmo que o tamanho de armazenamento do PV existente.
volumeName	Nome do PV.

Figura 16-7 SFS - endereço de montagem do sistema de arquivos



NOTA

A VPC à qual o sistema de arquivos pertence deve ser a mesma VPC da VM do ECS para a qual a carga de trabalho está planejada.

Passo 4 Crie um PV.

kubectl create -f pv-sfs-example.yaml

Passo 5 Crie uma PVC.

kubectl create -f pvc-sfs-example.yaml

----Fim

16.6.4 (kubectl) Criação de uma Implementação montada com um volume do SFS

Cenário

Depois que um volume do SFS é criado ou importado para o CCE, você pode montá-lo em uma carga de trabalho.

Observações e restrições

O exemplo de configuração a seguir se aplica a clusters do Kubernetes 1.13 ou anterior.

Procedimento

Passo 1 Use o kubectl para se conectar ao cluster. Para mais detalhes, consulte **Conexão a um cluster usando o kubectl**.

Passo 2 Execute os comandos a seguir para configurar o arquivo **sfs-deployment-example.yaml**, que é usado para criar um pod.

touch sfs-deployment-example.yaml

vi sfs-deployment-example.yaml

Exemplo de montagem de um volume do SFS em uma Implementação (volume compartilhado baseado em PVC):

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: sfs-deployment-example                                 # Workload name
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: sfs-deployment-example
  template:
    metadata:
      labels:
        app: sfs-deployment-example
    spec:
      containers:
      - image: nginx
        name: container-0
        volumeMounts:
        - mountPath: /tmp                                      # Mount path
          name: pvc-sfs-example
      imagePullSecrets:
        - name: default-secret
      restartPolicy: Always
      volumes:
      - name: pvc-sfs-example
        persistentVolumeClaim:
          claimName: pvc-sfs-auto-example                      # PVC name
```

Tabela 16-36 Parâmetros principais

Parâmetro primário	Parâmetro	Descrição
metadata	name	Nome do pod a ser criado.
spec.template.spec.containers.volumeMounts	mountPath	Monte o caminho no contêiner. Neste exemplo, o caminho de montagem é **/tmp**.
spec.template.spec.volumes.persistentVolumeClaim	claimName	Nome de uma PVC existente.

NOTA

spec.template.spec.containers.volumeMounts.name e **spec.template.spec.volumes.name** devem ser consistentes porque eles têm uma relação de mapeamento.

Exemplo de montagem de um volume do SFS em um StatefulSet dedicado (volume baseado em modelo de PVC):

NOTA

Atualmente, os sistemas de arquivos do SFS estão esgotados e não podem ser usados exclusivamente pela definição do modelo de PVC.

Exemplo de YAML:

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: deploy-sfs-nfs-rw-in
  namespace: default
  labels:
    appgroup: ''
spec:
  replicas: 2
  selector:
    matchLabels:
      app: deploy-sfs-nfs-rw-in
  template:
    metadata:
      labels:
        app: deploy-sfs-nfs-rw-in
    spec:
      containers:
        - name: container-0
          image: 'nginx:1.12-alpine-perl'
          volumeMounts:
            - name: bs-nfs-rw-mountoptionpvc
              mountPath: /aaa
      imagePullSecrets:
        - name: default-secret
  volumeClaimTemplates:
    - metadata:
        name: bs-nfs-rw-mountoptionpvc
        annotations:
          volume.beta.kubernetes.io/storage-class: nfs-rw
          volume.beta.kubernetes.io/storage-provisioner: flexvolume-huawei.com/
fuxinfs
      spec:
        accessModes:
          - ReadWriteMany
        resources:
          requests:
            storage: 1Gi
  serviceName: wwww
```

Tabela 16-37 Parâmetros principais

Parâmetro primário	Parâmetro	Descrição
metadata	name	Nome da carga de trabalho criada.
spec.template.spec.containers	image	Imagem da carga de trabalho.
spec.template.spec.containers.volumeMount	mountPath	Monte o caminho no contêiner. Neste exemplo, o caminho de montagem é **/tmp**.
spec	serviceName	Serviço correspondente à carga de trabalho. Para obter detalhes sobre como criar um Serviço, consulte **Criação de um StatefulSet**.

NOTA

spec.template.spec.containers.volumeMounts.name e **spec.volumeClaimTemplates.metadata.name** devem ser consistentes porque têm uma relação de mapeamento.

Passo 3 Execute o seguinte comando para criar o pod:

kubectl create -f sfs-deployment-example.yaml

Após a conclusão da criação, entre no console do CCE. No painel de navegação, escolha **Resource Management** > **Storage** > **SFS**. Clique no nome da PVC. Na página de detalhes da PVC, você pode exibir a relação de vinculação entre o SFS e a PVC.

----Fim

16.6.5 (kubectl) Criação de um StatefulSet montado com um volume do SFS

Cenário

O CCE permite que você use um volume do SFS existente para criar um StatefulSet por meio de uma PersistentVolumeClaim (PVC).

Observações e restrições

O exemplo de configuração a seguir se aplica a clusters do Kubernetes 1.13 ou anterior.

Procedimento

Passo 1 Crie um volume SFS referindo-se a **(kubectl) Criação automática de um volume do SFS** eregistre o nome do volume.

Passo 2 Use o kubectl para se conectar ao cluster. Para mais detalhes, consulte **Conexão a um cluster usando o kubectl**.

Passo 3 Crie um arquivo YAML para criar a carga de trabalho. Suponha que o nome do arquivo é **sfs-statefulset-example.yaml**.

touch sfs-statefulset-example.yaml

vi sfs-statefulset-example.yaml

Exemplo de YAML:

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: sfs-statefulset-example
  namespace: default
spec:
  replicas: 2
  selector:
    matchLabels:
      app: sfs-statefulset-example
  serviceName: qwqq
  template:
    metadata:
      annotations:
        metrics.alpha.kubernetes.io/custom-endpoints:
'[{"api":"","path":"","port":"","names":""}]'
        pod.alpha.kubernetes.io/initialized: "true"
      labels:
        app: sfs-statefulset-example
    spec:
      affinity: {}
      containers:
```



```
      - image: nginx:latest
        name: container-0
        volumeMounts:
        - mountPath: /tmp
          name: pvc-sfs-example
      imagePullSecrets:
      - name: default-secret
      volumes:
        - name: pvc-sfs-example
          persistentVolumeClaim:
            claimName: cce-sfs-demo
```

Tabela 16-38 Parâmetros principais

Parâmetro primário	Parâmetro	Descrição
spec	replicas	Número de pods.
metadata	name	Nome da carga de trabalho criada.
spec.template.spec.containers	image	Imagem utilizada pela carga de trabalho.
spec.template.spec.containers.volumeMounts	mountPath	Caminho de montagem no contêiner.
spec	serviceName	Serviço correspondente à carga de trabalho. Para obter detalhes sobre como criar um Serviço, consulte **Criação de um StatefulSet**.
spec.template.spec.volumes.persistentVolumeClaim	claimName	Nome de uma PVC existente.

NOTA

spec.template.spec.containers.volumeMounts.name e **spec.template.spec.volumes.name** devem ser consistentes porque eles têm uma relação de mapeamento.

Passo 4 Crie o StatefulSet.

kubectl create -f sfs-statefulset-example .yaml

----Fim

