云容器引擎
Autopilot
用户指南

文档版本 01

发布日期 2025-01-27

华为云计算技术有限公司

商标声明


和其他华为商标均为华为技术有限公司的商标。
本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为云计算技术有限公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为云计算技术有限公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

华为云计算技术有限公司

地址： 贵州省贵安新区黔中大道交兴功路华为云数据中心 邮编：550029

网址： https://www.huaweicloud.com/

目 录

1 集群

1.1 集群概述

1.1.1 集群基本信息

Kubernetes是一个开源的容器编排引擎，可用于容器化应用的自动化部署、 扩缩和管理。

对应用开发者而言，可以把Kubernetes看成一个集群操作系统。Kubernetes提供服务发现、伸缩、负载均衡、自愈甚至选举等功能，让开发者从基础设施相关配置中解脱出来。

集群的网络

集群的网络可以分成两部分：

- 容器网络：为集群内Pod分配IP地址，负责Pod的通信。当前CCE Autopilot集群仅支持云原生网络2.0模型。
- 服务网络：服务（Service）是用来解决访问容器的Kubernetes对象，每个Service都有一个固定的IP地址。

在创建集群时，您需要为各个网络选择合适的网段，确保各网段之间不存在重叠，每个网段下有足够的IP地址可用。

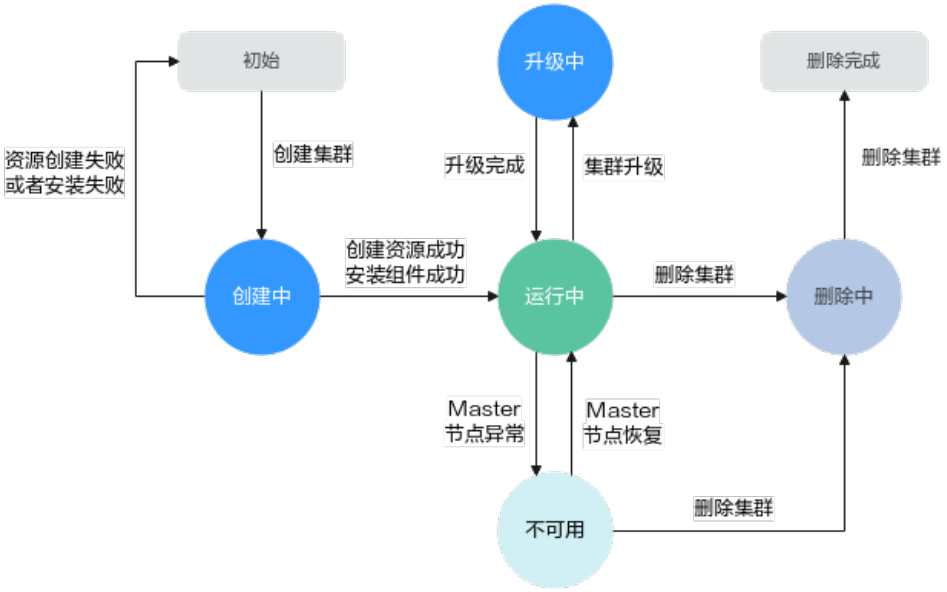
集群生命周期

表 1-1 集群状态说明

状态	说明
创建中	集群正在创建，正在申请云资源
运行中	集群正常运行
升级中	集群正在升级中

状态	说明
不可用	当前集群不可用
删除中	集群正在删除中

图 1-1 集群状态流转



集群版本发布说明

Kubernetes不断集成新特性并修复漏洞，同时CCE Autopilot集群也会定期发布其支持的Kubernetes版本，并在社区版的基础上提供更多的功能更新和维护。社区Kubernetes版本不断集成新特性、详情请参见：

- **Kubernetes版本发布记录**
- **Autopilot集群补丁版本发布记录**

为了能够方便您使用稳定又可靠的Kubernetes版本，建议您在版本维护周期结束之前升级您的Kubernetes集群。

1.1.2 Kubernetes 版本发布记录

1.1.2.1 Kubernetes 1.31 版本说明（公测）

CCE Autopilot集群严格遵循社区一致性认证，现已支持创建Kubernetes 1.31集群。本文介绍Kubernetes 1.31版本的变更说明。

索引

- **新增特性及特性增强**
- **API变更与弃用**

- **参考链接**

新增特性及特性增强

Kubernetes 1.31版本

- StatefulSet起始序号（GA）

  在Kubernetes 1.31中，StatefulSetStartOrdinal特性进阶至GA。默认情况下，StatefulSet中Pod的序号是从0开始，该特性引入后允许用户自定义Pod的起始序号。详细使用方式请参考**起始序号**。

- 弹性索引Job（GA）

  在Kubernetes 1.31中，ElasticIndexedJob特性进阶至GA。该特性允许用户在索引Job创建后修改其.spec.completions和.spec.parallelism字段，使之具备弹性伸缩能力。详细使用方式请参考**弹性索引Job**。

- Pod失效策略（GA）

  在Kubernetes 1.31中，JobPodFailurePolicy特性进阶至GA。该特性允许用户根据Pod失效的原因来分别指定处理方式（重试或者忽略），以优化避免不必要的Pod重启带来的运行成本。详细使用方式请参考**Pod失效策略**。

- Pod干扰状况（GA）

  在Kubernetes 1.31中，PodDisruptionConditions特性进阶至GA。该特性在Pod的Condition中新增了DisruptionTarget类型，表示Pod失效的原因，例如被高优先级的Pod抢占、因节点删除而被清理、被kubelet终止等。若Pod是Job或者CronJob控制器创建的，可以与**Pod失效策略**一起使用，通过该Condition定义失效时的行为。更多关于本特性的详细信息请参考**Pod干扰状况**。

- 定制资源的可选择字段（Beta）

  在Kubernetes 1.31中，CustomResourceFieldSelectors特性进阶至Beta。该特性支持对CRD配置selectableFields，并支持使用Field Selectors过滤List、Watch和DeleteCollection请求，方便用户定位或管理符合特定条件的CRD资源。详细使用方式请参考**定制资源的可选择字段**。

- Job成功策略（Beta）

  在Kubernetes 1.31中，JobSuccessPolicy特性进阶至Beta。该特性允许用户基于成功的Pod个数为Job配置成功策略。详细使用方式请参考**成功策略**。

- podAffinity中的matchLabelKeys（Beta）

  在Kubernetes 1.31中，MatchLabelKeysInPodAffinity特性进阶至Beta。该特性在podAffinity和podAntiAffinity中引入了更为精细的配置字段matchLabelKeys和mismatchLabelKeys，以解决调度器在Deployment滚动更新期间无法区分新老Pod，继而导致调度结果不符合亲和性和反亲和性预期的问题。详细使用方式请参考**matchLabelKeys**。

- ServiceAccountTokenNodeBinding（Beta）

  在Kubernetes 1.31中，ServiceAccountTokenNodeBinding特性进阶至Beta。该特性支持创建绑定到节点的ServiceAccount Token：在Token中包含节点信息的声明，并在使用Token时验证节点的存在，若节点被删除，则Token将会失效。详细使用方式请参考**手动为 ServiceAccount 创建 API 令牌**。

Kubernetes 1.30版本

- Webhook匹配表达式（GA）

在Kubernetes1.30版本中，Webhook匹配表达式特性进阶至GA。此特性允许对准入Webhook支持根据特定的条件进行匹配，更细粒度地控制Webhook的触发条件。详细使用方式请参考**动态准入控制**。

- 验证准入策略（GA）

  在Kubernetes1.30版本中，验证准入策略（ValidatingAdmissionPolicy）特性进阶至GA。该特性支持通过CEL表达式声明资源的验证准入策略。详细使用方式请参考**验证准入策略**。

- 基于ContainerResource指标的Pod水平自动扩缩容（GA）

  在Kubernetes1.30版本中，基于ContainerResource指标的Pod水平自动扩缩容特性进阶至GA。该特性允许HPA根据Pod中各个容器的资源使用情况来配置自动伸缩，而不仅是Pod的整体资源使用情况，便于为Pod中最重要的容器配置扩缩容阈值。详细使用方式请参考**容器资源指标**。

- 传统ServiceAccount令牌清理器（GA）

  在Kubernetes1.30版本中，传统ServiceAccount令牌清理器特性进阶至GA。其作为kube-controller-manager的一部分运行，每24小时检查一次，查看是否有任何自动生成的传统ServiceAccount令牌在特定时间段内（默认为一年，通过--legacy-service-account-token-clean-up-period指定）未被使用。如果有的话，清理器会将这些令牌标记为无效，并添加kubernetes.io/legacy-token-invalid-since标签，其值为当前日期。如果一个无效的令牌在特定时间段（默认为1年，通过--legacy-service-account-token-clean-up-period指定）内未被使用，清理器将会删除它。更多使用细节请参考**传统ServiceAccount令牌清理器**。

Kubernetes 1.29版本

- Service的负载均衡IP模式（Alpha）

  在Kubernetes1.29版本，Service的负载均衡IP模式以Alpha版本正式发布。其在Service的status中新增字段ipMode，用于配置集群内Service到Pod的流量转发模式。当设置为VIP时，目的地址为负载均衡IP和端口的流量将由kube-proxy重定向到目标节点，当设置为Proxy时，流量将被发送到负载均衡器，然后由负载均衡器转发到目标节点。这项特性将有助于解决流量绕过负载均衡器缩导致的负载均衡器功能缺失问题。更多使用细节请参考**Service的负载均衡IP模式**。

- NFTables代理模式（Alpha）

  在Kubernetes1.29版本，NFTables代理模式以Alpha版本正式发布。该特性允许kube-proxy运行在NFTables模式，在该模式下，kube-proxy使用内核netfilters子系统的nftables API来配置数据包转发规则。更多使用细节请参考**NFTables代理模式**。

- 未使用容器镜像的垃圾收集（Alpha）

  在Kubernetes1.29版本，未使用容器镜像的垃圾收集以Alpha版本正式发布。该特性允许用户为每个节点配置本地镜像未被使用的最长时间，超过这个时间镜像将被垃圾回收。配置方法为使用kubelet配置文件中的ImageMaximumGCAge字段。更多使用细节请参考**未使用容器镜像的垃圾收集**。

- PodLifecycleSleepAction（Alpha）

  在Kubernetes1.29版本，PodLifecycleSleepAction以Alpha版本正式发布。该特性在容器生命周期回调中引入了Sleep回调程序，可以配置让容器在启动后和停止前暂停一段指定的时间。更多使用细节请参考**PodLifecycleSleepAction**。

- KubeletSeparateDiskGC（Alpha）

  在Kubernetes1.29版本，KubeletSeparateDiskGC以Alpha版本正式发布。该特性启用后，即使在容器镜像和容器位于独立文件系统的情况下，也能进行垃圾回收。

- matchLabelKeys/mismatchLabelKeys（Alpha）

  在Kubernetes1.29版本，matchLabelKeys/mismatchLabelKeys以Alpha版本正式发布。该特性启用后，在Pod的亲和/反亲和配置中新增了matchLabelKeys/mismatchLabelKeys字段，可配置更丰富的Pod间亲和/反亲和策略。更多使用细节请参考**matchLabelKeys/mismatchLabelKeys**。

- clusterTrustBundle投射卷（Alpha）

  在Kubernetes1.29版本，clusterTrustBundle投射卷以Alpha版本正式发布。该特性启用后，支持将ClusterTrustBundle对象以自动更新的文件的形式注入卷。更多使用细节请参考**clusterTrustBundle投射卷**。

- 基于运行时类的镜像拉取（Alpha）

  在Kubernetes1.29版本中，基于运行时类的镜像拉取以Alpha版本正式发布。该特性启用后， kubelet 会通过一个元组（镜像名称，运行时处理程序）而不仅仅是镜像名称或镜像摘要来引用容器镜像。 您的容器运行时可能会根据选定的运行时处理程序调整其行为。基于运行时类来拉取镜像对于基于VM的容器会有帮助。更多使用细节请参考**基于运行时类的镜像拉取**。

- PodReadyToStartContainers状况达到Beta

  在Kubernetes1.29版本，PodReadyToStartContainers状况特性达到Beta版本。其在Pod的status中新增了一个名为PodReadyToStartContainers的Condition，该Condition为true表示Pod的沙箱已就绪，可以开始创建业务容器。该特性使得集群管理员可以更清晰和全面地查看 Pod 沙箱的创建完成和容器的就绪状态，增强了指标监控和故障排查能力。更多使用细节请参考**PodReadyToStartContainersCondition**。

- Job相关特性
  - Pod更换策略达到Beta

    在Kubernetes1.29版本，Pod更换策略特性达到Beta版本。该特性确保只有Pod达到Failed阶段（status.phase: Failed）才会被替换，而不是当删除时间戳不为空时，Pod仍处于删除过程中就重建Pod，以此避免出现2个Pod同时占用索引和节点资源。

  - 逐索引的回退限制达到Beta

    在Kubernetes1.29版本，逐索引的回退限制特性达到Beta版本。默认情况下，带索引的 Job（Indexed Job）的 Pod 失败情况会被统计下来，受 .spec.backoffLimit 字段所设置的全局重试次数限制。这意味着，如果存在某个索引值的 Pod 一直持续失败，则会 Pod 会被重新启动，直到重试次数达到限制值。 一旦达到限制值，整个 Job 将被标记为失败，并且对应某些索引的 Pod 甚至可能从不曾被启动。该特性可以在某些索引值的 Pod 失败的情况下，仍完成执行所有索引值的 Pod，并且通过避免对持续失败的、特定索引值的Pod进行不必要的重试，更好地利用计算资源。

- 原生边车容器达到Beta

  在Kubernetes1.29版本，原生边车容器特性达到Beta版本。其在initContainers中新增了restartPolicy字段，当配置为Always时表示启用边车容器。边车容器和业务容器部署在同一个Pod中，但并不会延长Pod的生命周期。边车容器常用于网络代理、日志收集等场景。更多使用细节请参考**边车容器**。

- 传统ServiceAccount令牌清理器达到Beta

  在Kubernetes1.29版本，传统ServiceAccount令牌清理器特性达到Beta版本。其作为kube-controller-manager的一部分运行，每24小时检查一次，查看是否有任何自动生成的传统ServiceAccount令牌在特定时间段内（默认为一年，通过--legacy-service-account-token-clean-up-period指定）未被使用。如果有的话，清理器会将这些令牌标记为无效，并添加kubernetes.io/legacy-token-invalid-

since标签，其值为当前日期。如果一个无效的令牌在特定时间段（默认为1年，通过--legacy-service-account-token-clean-up-period指定）内未被使用，清理器将会删除它。更多使用细节请参考**传统ServiceAccount令牌清理器**。

- DevicePluginCDIDevices达到Beta

  在Kubernetes1.29版本，DevicePluginCDIDevices特性达到Beta版本。该特性在DeviceRunContainerOptions增加CDIDevices字段，使得设备插件开发者可以直接将CDI设备名称传递给支持CDI的容器运行时。

- PodHostIPs达到Beta

  在Kubernetes1.29版本中，PodHostIPs特性达到Beta版本。该特性在Pod和downward API的Status中增加hostIPs字段，用于将节点IP地址暴露给工作负载。该字段是hostIP的双栈协议版本，第一个IP始终与hostIP相同。

- API优先级和公平性达到GA

  在Kubernetes1.29版本，API优先级和公平性（APF）特性达到GA版本。APF以更细粒度的方式对请求进行分类和隔离，提升最大并发限制，并且它还引入了空间有限的排队机制，因此在非常短暂的突发情况下，API 服务器不会拒绝任何请求。 通过使用公平排队技术从队列中分发请求，这样，一个行为不佳的控制器就不会导致其他控制器异常 （即使优先级相同）。更多使用细节请参考**API优先级和公平性**。

- APIListChunking达到GA

  在Kubernetes1.29版本，APIListChunking特性达到GA版本。该特性允许客户端在List请求中进行分页，避免一次性返回过多数据而导致的性能问题。

- PersistentVolume的阶段转换时间戳达到Beta

  在Kubernetes1.29版本，PersistentVolume的阶段转换时间戳特性达到Beta版本。该特性在PV的status中添加了一个lastPhaseTransitionTime字段，表示PV上一次phase变化的时间。通过该字段，集群管理员可以跟踪PV上次转换到不同阶段的时间，从而实现更高效、更明智的资源管理。更多使用细节请参考**PersistentVolume的阶段转换时间戳**。

- ReadWriteOncePod达到GA

  在Kubernetes1.29版本中，ReadWriteOncePod特性达到GA版本。该特性允许用户在PVC中配置访问模式为ReadWriteOncePod，确保同时只有一个 Pod能够修改存储中的数据，以防止数据冲突或损坏。更多使用细节请参考**ReadWriteOncePod**。

- CSINodeExpandSecret达到GA

  在Kubernetes1.29版本中，CSINodeExpandSecret特性达到GA版本。该特性允许在添加节点时将Secret身份验证数据传递到CSI驱动以供后者使用。

- CRD验证表达式语言达到GA

  在Kubernetes1.29版本中，CRD验证表达式语言特性达到GA版本。该特性允许用户在CRD中使用通用表达式语言（CEL）定义校验规则，相比webhook更加高效。更多使用细节请参考**CRD校验规则**。

API 变更与弃用

Kubernetes 1.31版本

在Kubernetes1.31版本中，在CustomResourceDefinition（CRD）中指定caBundle字段时，如果caBundle非空，但内容无效或不包含任何CA证书，那么该CRD将不会提供服务。CRD的caBundle设置为有效状态后，将不再允许通过更新操作将其变为无效或内容为空的状态（直接更新将报错invalid field value），以避免中断CRD的正常服务。

Kubernetes 1.30版本

- 在Kubernetes1.30版本中，kubectl移除了apply命令的prune-whitelist参数，使用prune-allowlist替代。
- 在Kubernetes1.30版本中，移除了在1.27版本已废弃的准入插件SecurityContextDeny，使用**Pod安全性准入插件**（PodSecurity）替代。

Kubernetes 1.29版本

- 在Kubernetes1.29版本中，新创建的CronJob不再支持在.spec.schedule中通过TZ或者CRON_TZ配置时区，请使用.spec.timeZone替代。已经创建的CronJob不受此影响。
- 在Kubernetes1.29版本中，移除了alpha API ClusterCIDR。
- 在Kubernetes1.29版本中，kube-apiserver新增启动参数--authentication-config，用于指定AuthenticationConfiguration文件地址，该启动参数与--oidc-*启动参数互斥。
- 在Kubernetes1.29版本中，移除了KubeSchedulerConfiguration的API版本kubescheduler.config.k8s.io/v1beta3，请迁移至kubescheduler.config.k8s.io/v1。
- 在Kubernetes1.29版本中，将CEL表达式添加到v1alpha1 AuthenticationConfiguration中。
- 在Kubernetes1.29版本中，新增对象ServiceCIDR，允许用户动态配置集群分配Service的ClusterIP时所使用的地址范围。
- 在Kubernetes1.29版本中，kube-proxy新增启动参数--conntrack-udp-timeout、--conntrack-udp-timeout-stream，可对nf_conntrack_udp_timeout和nf_conntrack_udp_timeout_stream内核参数进行设置 。
- 在Kubernetes1.29版本中，将CEL表达式的支持添加到v1alpha1 AuthenticationConfiguration的WebhookMatchCondition中。
- 在Kubernetes1.29版本中，PVC.spec.Resource的类型由原先的ResourceRequirements替换为VolumeResourceRequirements。
- 在Kubernetes1.29版本中，将PodFailurePolicyRule中的OnPodConditions转变为可选字段。
- 在Kubernetes1.29版本中，FlowSchema与PriorityLevelConfiguration的API版本flowcontrol.apiserver.k8s.io/v1beta3已升级至flowcontrol.apiserver.k8s.io/v1，并进行了以下更改
  - PriorityLevelConfiguration：.spec.limited.nominalConcurrencyShares字段在省略时自动设为默认值30，并且为了确保与1.28兼容，在1.29中v1版本该字段不允许显示指定为0。在1.30中，将允许v1版本该字段显示指定为0。flowcontrol.apiserver.k8s.io/v1beta3已废弃，并在1.32中不再支持。
- 在Kubernetes1.29版本中， 优化了kube-proxy的命令行文档，kube-proxy实际上不会将任何socket绑定到由--bind-address启动参数指定的IP。
- 在Kubernetes1.29版本中，当CSI-Node-Driver没有在运行时，NodeStageVolume操作会重试。
- 在Kubernetes1.29版本中，ValidatingAdmissionPolicy支持对CRD资源进行校验。使用该特性需开启特性门控ValidatingAdmissionPolicy。
- 在Kubernetes1.29版本中，kube-proxy新增启动参数--nf-conntrack-tcp-be-liberal，可对内核参数nf_conntrack_tcp_be_liberal进行配置。

- 在Kubernetes1.29版本中，kube-proxy新增启动参数--init-only，设置后使kube-proxy的init容器在特权模式下运行，进行一些初始化配置，然后退出。
- 在Kubernetes1.29版本中，CRI的返回体中新增容器的fileSystem字段，表示容器的fileSystem使用信息，而原先只包含镜像的fileSystem。
- 在Kubernetes1.29版本中，所有内置的CloudProvider全部默认设置为关闭，如果仍需使用，可通过配置DisableCloudProviders和DisableKubeletCloudCredentialProvider特性门控来选择性关闭或者打开。

参考链接

关于Kubernetes 1.31与其他版本的性能对比和功能演进的更多信息，请参考：

- **Kubernetes v1.31 Release Notes**
- **Kubernetes v1.30 Release Notes**
- **Kubernetes v1.29 Release Notes**

1.1.2.2 Kubernetes 1.28 版本说明

云容器引擎（CCE）严格遵循社区一致性认证，现已支持创建Kubernetes 1.28集群。本文介绍Kubernetes 1.28版本的变更说明。

索引

重要说明

- 在Kubernetes 1.28版本，调度框架发生变化，减少无用的重试，从而提高调度程序的整体性能。如果开发人员在集群中使用了自定义调度程序插件，请参见**调度框架变化**进行适配升级。
- 在Kubernetes 1.28版本，Ceph FS树内插件已在v1.28中弃用，并计划在v1.31中删除（社区没有计划进行CSI迁移）。建议使用**Ceph CSI**第三方存储驱动程序作为替代方案。
- 在Kubernetes 1.28版本，Ceph RBD树内插件已在v1.28中弃用，并计划在v1.31中删除（社区没有计划进行CSI迁移）。建议使用RBD模式的**Ceph CSI**第三方存储驱动程序作为替代方案。

新增特性及特性增强

社区特性的Alpha阶段默认禁用、Beta阶段一般默认启用、GA阶段将一直默认启用，且不能禁用（会在后续版本中删除这个开关功能）。CCE对新特性的策略与社区保持一致。

- 版本偏差策略扩展至3个版本

从1.28控制平面/1.25工作节点开始，Kubernetes版本偏差策略将支持的控制平面/工作节点偏差扩展到3个版本。 这使得节点的年度次要版本升级成为可能，同时保持受支持的次要版本。更多细节请参考**版本偏差策略**。

- 可追溯的默认StorageClass进阶至GA

  在Kubernetes 1.28版本，可追溯默认StorageClass赋值现已进阶至GA。 这项增强特性极大地改进了默认的StorageClasses为PersistentVolumeClaim（PVC）赋值的方式。

  PersistentVolume（PV）控制器已修改为：当未设置storageClassName时，自动向任何未绑定的PersistentVolumeClaim分配一个默认的StorageClass。此外，API服务器中的PersistentVolumeClaim准入验证机制也已调整为允许将值从未设置状态更改为实际的StorageClass名称。更多使用细节请参考**默认StorageClass赋值**。

- 原生边车容器（Alpha）

  在Kubernetes 1.28版本，原生边车容器以Alpha版本正式发布。其在Init容器中添加了一个新的restartPolicy字段， 该字段在SidecarContainers特性门控启用时可用。需要注意的是，原生边车容器目前仍有些问题需要解决，因此K8S社区建议仅在Alpha阶段的**短期测试集群**中使用边车功能。 更多使用细节请参考**原生边车容器**。

- 混合版本代理（Alpha）

  在Kubernetes 1.28版本，发布了用于改进集群安全升级的新机制（混合版本代理）。该特性为Alpha特性。当集群进行升级时，集群中不同版本的kube-apiserver为不同的内置资源集（组、版本、资源）提供服务。在这种情况下资源请求如果由任一可用的apiserver提供服务，请求可能会到达无法解析此请求资源的apiserver中，导致请求失败。该特性能解决该问题。（主要注意的是，CCE本身提供的升级能力即可做到无损升级，因此不存在该特性涉及的场景）。更多使用细节请参考**混合版本代理**。

- 节点非体面关闭特性达到GA

  在Kubernetes 1.28版本，节点非体面关闭特性达到GA阶段。当一个节点被关闭但没有被Kubelet的Node Shutdown Manager检测到时，StatefulSet的Pod将会停留在终止状态，并且不能移动到新运行的节点上。当用户确认该节点已经处于不可恢复的情况下，可以手动为Node打上out-of-service的污点，以使得该节点上的StatefulSet的Pod和VolumeAttachments被强制删除，并在健康的Node上创建相应的Pod。更多使用细节请参考**节点非体面关闭**。

- NodeSwap特性达到Beta

  在Kubernetes 1.28版本，NodeSwap能力进阶至Beta版本。目前仍然处于默认关闭状态，需要使用NodeSwap门控打开。该特性可以为Linux节点上运行的Kubernetes工作负载逐个节点地配置内存交换。需要注意的是，该特性虽然进阶至Beta特性，但仍然存在一些需要增强的问题和安全风险。更多使用细节请参考**NodeSwap特性**。

- Job相关特性

  在Kubernetes 1.28版本，增加了**Pod更换策略**和基于**带索引Job的回退限制**两个alpha特性。

  - Pod更换策略

    默认情况下，当Pod进入终止（Terminating）状态（例如由于抢占或驱逐机制）时，Kubernetes会立即创建一个替换的Pod，因此这时会有两个Pod同时运行。

    在Kubernetes 1.28版本中可以使用JobPodReplacementPolicy 来启用该特性。可以在Job的Spec中定义podReplacementPolicy，目前仅可设置为

Failed。在设置为Failed之后，Pod仅在达到Failed阶段时才会被替换，而不是在它们处于终止过程中（Terminating）时被替换。此外，您可以检查Job的.status.termination字段。该字段的值表示终止过程中的Job所关联的Pod数量。

- 带索引Job的回退限制

  默认情况下，带索引的Job（Indexed Job）的 Pod 失败情况会被记录下来，受.spec.backoffLimit字段所设置的全局重试次数限制。这意味着，如果存在某个索引值的Pod一直持续失败，则Pod会被重新启动，直到重试次数达到限制值。一旦达到限制值，整个Job将被标记为失败，并且对应某些索引的Pod甚至可能从不曾被启动。

  在Kubernetes 1.28版本中，可以通过启用集群的JobBackoffLimitPerIndex特性门控来启用此特性。开启之后，允许在创建带索引的Job（Indexed Job）时指定.spec.backoffLimitPerIndex字段。当某个Job的失败次数超过设定的上限时，将不再进行重试。

- CEL相关特性

  在Kubernetes 1.28版本，CEL能力进行了相应的增强。

  - CRD使用CEL进行Validate的特性进阶至Beta

    该特性在v1.25版本就已经升级为Beta版本。通过将CEL表达式直接集成在CRD中，可以使开发者在不使用Webhook的情况下解决大部分对CR实例进行验证的用例。在未来的版本，将继续扩展CEL表达式的功能，以支持默认值和CRD转换。

  - 基于CEL的准入控制进阶至Beta

    基于通用表达式语言 (CEL) 的准入控制是可定制的，对于kube-apiserver接受到的请求，可以使用CEL表达式来决定是否接受或拒绝请求，可作为Webhook准入控制的一种替代方案。在 v1.28 中，CEL准入控制被升级为Beta，同时添加了一些新功能，包括但不限于：

    - ValidatingAdmissionPolicy类型检查现在可以正确处理CEL表达式中的“authorizer”变量。
    - ValidatingAdmissionPolicy支持对messageExpression字段进行类型检查。
    - kube-controller-manager组件新增ValidatingAdmissionPolicy控制器，用来对ValidatingAdmissionPolicy中的CEL表达式做类型检查，并将原因保存在状态字段中。
    - 支持变量组合，可以在ValidatingAdmissionPolicy中定义变量，然后在定义其他变量时使用它。
    - 新增CEL库函数支持对Kubernetes的resource.Quantity类型进行解析。

- 其它特性说明

  - 在Kubernetes 1.28版本，ServiceNodePortStaticSubrange 特性为beta，允许保留静态端口范围，避免与动态分配端口冲突。具体细节请参考**为NodePort Service分配端口时避免冲突**。
  - 在Kubernetes 1.28版本，增加了alpha特性ConsistentListFromCache，允许kube-apiserver从缓存中提供一致性列表，Get和List请求可以从缓存中读取数据，而不需要从etcd中获取。
  - 在Kubernetes 1.28版本，kubelet能够配置drop-in目录（alpha特性）。该特性允许向kubelet添加对“--config-dir”标志的支持，以允许用户指定一个插

入目录，该目录将覆盖位于/etc/kubernetes/kubelet.conf位置的Kubelet的配置。

- 在Kubernetes 1.28版本，ExpandedDNSConfig升级至GA，默认会被打开。该参数用于允许扩展DNS的配置。
- 在Kubernetes 1.28版本，提供Alpha特性CRD Validation Ratcheting。该特性允许Patch或者Update请求没有更改任何不合法的字段，将允许CR验证失败。
- 在Kubernetes 1.28版本，kube-controller-manager添加了--concurrent-cron-job-syncs flag用来设置cron job controller的workers数。

API 变更与弃用

- 在Kubernetes 1.28版本，移除特性NetworkPolicyStatus，因此Network Policy不再有status属性。
- 在Kubernetes 1.28版本，Job对象中增加了新的annotationbatch.kubernetes.io/cronJob-scheduled-timestamp，表示Job的创建时间。
- 在Kubernetes 1.28版本，Job API中添加podReplacementPolicy和terminating字段，当前一旦先前创建的pod终止，Job就会立即启动替换pod。添加字段允许用户指定是在先前的Pod终止后立即更换Pod（原行为），还是在现有的Pod完全终止后才替换Pod（新行为）。这是一项Alpha级别特性，您可以通过在集群中启用**JobPodReplacementPolicy** 来启用该特性。
- 在Kubernetes 1.28版本，Job支持BackoffLimitPerIndex字段。当前使用的运行Job的策略是Job中的整个Pod共享一个Backoff机制，当Job达到Backoff的限制时，整个Job都会被标记为失败，并清理资源，包括尚未运行的index。此字段允许对单个的index设置Backoff。更多信息请参见**带索引Job的Backoff限制**。
- 在Kubernetes 1.28版本，添加ServedVersions字段到 StorageVersion API中。该变化由混合代理版本特性引入。该增加字段ServedVersions用于表明API服务器可以提供的版本。
- 在Kubernetes 1.28版本，SelfSubjectReview 添加到authentication.k8s.io/v1中，并且kubectl auth whoami走向GA。
- 在Kubernetes 1.28版本，PersistentVolume有了一个新的字段LastPhaseTransitionTime，用来保存最近一次volume转变Phase的时间。
- 在Kubernetes 1.28版本，PVC.Status中移除resizeStatus，使用AllocatedResourceStatus替代。resizeStatus表示调整存储大小操作的状态，默认为空字符串。
- 在Kubernetes 1.28版本，设置了hostNetwork: true并且定义了ports的Pods，自动设置hostport字段。
- 在Kubernetes 1.28版本，StatefulSet的Pod索引设置为Pod的标签statefulset.kubernetes.io/pod-index。
- 在Kubernetes 1.28版本，Pod的Condition字段中的PodHasNetwork重命名为PodReadyToStartContainers，用来表明网络、卷等已成功创建，sandbox pod已经创建完成，可以启动容器。
- 在Kubernetes 1.28版本，在KubeSchedulerConfiguration中添加了新的配置选项delayCacheUntilActive， 该参数为true时，非master节点的kube-scheduler不会缓存调度信息。这为非主节点的内存减缓了压力，但会导致主节点发生故障时，减慢故障转移的速度。
- 在Kubernetes 1.28版本，在admissionregistration.k8s.io/v1alpha1.ValidatingAdmissionPolicy中添加namespaceParamRef字段。

- 在Kubernetes 1.28版本，在CRD validation rules中添加reason和fieldPath，允许用户指定验证失败的原因和字段路径。
- 在Kubernetes 1.28版本，ValidatingAdmissionPolicy的CEL表达式通过namespaceObject支持namespace访问。
- 在Kubernetes 1.28版本，将API groups ValidatingAdmissionPolicy 和ValidatingAdmissionPolicyBinding提升到betav1。
- 在Kubernetes 1.28版本，ValidatingAdmissionPolicy 扩展了messageExpression字段，用来检查已解析类型。

特性门禁及命令行参数

- 在Kubernetes 1.28版本，kubelet移除了flag –short。因此kubectl version 默认输出与kubectl version –short相同。
- 在Kubernetes 1.28版本，kube-controller-manager废弃flag--volume-host-cidr-denylist和--volume-host-allow-local-loopback。--volume-host-cidr-denylist是用逗号分隔的一个CIDR范围列表，禁止使用这些地址上的卷插件。--volume-host-allow-local-loopback为false时，禁止本地回路IP地址和--volume-host-cidr-denylist中所指定的CIDR范围。
- 在Kubernetes 1.28版本，kubelet --azure-container-registry-config被弃用并在未来的版本中会被删除。 请使用--image-credential-provider-config和--image-credential-provider-bin-dir来设置。
- 在Kubernetes 1.28版本，kube-scheduler: 删除了--lock-object-namespace和--lock-object-name。 请使用--leader-elect-resource-namespace和--leader-elect-resource-name或ComponentConfig来配置这些参数。（ --lock-object-namespace用来定义锁对象的命名空间，--lock-object-name用来定义锁对象的名称 ）
- 在Kubernetes 1.28版本，KMSv1已弃用，以后只会接收安全更新。请改用KMSv2。在未来版本中，设置--feature-gates=KMSv1=true以使用已弃用的KMSv1功能。
- 在Kubernetes 1.28版本，移除了如下特性门禁：DelegateFSGroupToCSIDriver、DevicePlugins、KubeletCredentialProviders、MixedProtocolLBService、ServiceInternalTrafficPolicy、ServiceIPStaticSubrange、EndpointSliceTerminatingCondition。

参考链接

关于Kubernetes 1.28与其他版本的性能对比和功能演进的更多信息，请参考：
Kubernetes v1.28 Release Notes

1.1.2.3 Kubernetes 1.27 版本说明

云容器引擎（ CCE ）严格遵循社区一致性认证，现已支持创建Kubernetes 1.27集群。本文介绍Kubernetes 1.27版本的变更说明。

索引

主要特性

- SeccompDefault特性已进入稳定阶段

  如需使用SeccompDefault特性，您需要为每个节点的kubelet启用--seccomp-default**命令行标志**。如果启用该特性，kubelet将为所有工作负载默认使用RuntimeDefault seccomp配置文件，该配置文件由容器运行时定义，而不是使用Unconfined（禁用seccomp）模式。

- Job可变调度指令

  该特性在Kubernetes 1.22版本中引入，当前已进入稳定阶段。在大多数情况下，并行作业Pod希望在一定的约束下运行，例如希望所有Pod在同一可用区。该特性允许在Job开始前修改调度指令。您可以使用suspend字段挂起Job，在Job挂起阶段，Pod模板中的调度部分（例如节点选择器、节点亲和性、反亲和性、容忍度）允许修改。详情请参见**可变调度指令**。

- Downward API HugePages已进入稳定阶段

  在Kubernetes 1.20版本中，**Downward API**引入了*requests.hugepages-<pagesize>*和*limits.hugepages-<pagesize>*，HugePage可以和其他资源一样设置资源配额。

- Pod调度就绪态进入Beta阶段

  Pod创建后，Kubernetes调度程序会负责选择合适的节点运行pending状态的Pod。在实际使用时，一些Pod可能会由于资源不足长时间处于pending状态。这些Pod可能会影响集群中的其他组件运行（如Cluster Autoscaler）。通过指定/删除Pod的.spec.schedulingGates，您可以控制Pod何时准备好进行调度。详情请参见**Pod调度就绪态**。

- 通过Kubernetes API访问节点日志

  此功能当前处于Alpha阶段。集群管理员可以直接查询节点上的服务日志，可以帮助调试节点上运行的服务问题。如需使用此功能，请确保在该节点上启用了NodeLogQuery**特性门控**，并且kubelet配置选项enableSystemLogHandler和enableSystemLogQuery都设置为true。

- ReadWriteOncePod访问模式进入Beta阶段

  在Kubernetes 1.22版本中，PV和PVC提供了一种新的访问模式ReadWriteOncePod，该功能当前进入Beta阶段。卷可以被单个Pod以读写方式挂载。如果您想确保整个集群中只有一个Pod可以读取或写入该PVC，请使用ReadWriteOncePod访问模式，详情请参见**访问模式**。

- Pod拓扑分布约束中matchLabelKeys字段进入Beta阶段

  matchLabelKeys是一个Pod标签键的列表，用于选择需要计算分布方式的Pod集合。使用matchLabelKeys字段，您无需在变更Pod修订版本时更新pod.spec。控制器或Operator只需要将不同修订版的标签键设为不同的值。调度器将根据matchLabelKeys自动确定取值。详情请参见**Pod拓扑分布约束**。

- 快速标记SELinux卷标签功能进入Beta阶段

  默认情况下，容器运行时递归地将SELinux标签赋予所有Pod卷上的所有文件。 为了加快该过程，Kubernetes使用挂载可选项*-o context=<label>*可以立即改变卷的SELinux标签。详情请参见**快速标记SELinux卷标签**。

- VolumeManager重构进入Beta阶段

  重构的VolumeManager后，如果启用NewVolumeManagerReconstruction**特性门控**，将会在kubelet启动期间使用更有效的方式来获取已挂载卷。

- 服务器端字段校验和OpenAPI V3已进入稳定阶段

  Kubernetes 1.23中添加了对OpenAPI v3的支持，1.24版本中已进入Beta阶段，1.27已进入稳定阶段。

- 控制StatefulSet启动序号

  Kubernetes 1.26为StatefulSet引入了一个新的Alpha级别特性，可以控制Pod副本的序号。 从Kubernetes 1.27开始，此特性进入Beta阶段，序数可以从任意非负数开始。详情请参见**Kubernetes 1.27: StatefulSet启动序号简化了迁移**。

- HorizontalPodAutoscaler ContainerResource类型指标进入Beta阶段

  Kubernetes 1.20在HorizontalPodAutoscaler (HPA) 中引入了**ContainerResource类型指标**。在Kubernetes 1.27中，此特性进阶至Beta，相应的特性门控 (HPAContainerMetrics) 默认被启用。

- StatefulSet PVC自动删除进入Beta阶段

  Kubernetes v1.27提供一种新的策略机制，用于控制StatefulSets的PersistentVolumeClaims（PVCs）的生命周期。这种新的PVC保留策略允许用户指定当删除StatefulSet或者缩减StatefulSet中的副本时，是自动删除还是保留从StatefulSet规约模板生成的PVC。详情请参见**PersistentVolumeClaim保留**。

- 磁盘卷组快照

  磁盘卷组快照在Kubernetes 1.27中作为Alpha特性被引入。此特性允许用户对多个卷进行快照，以保证在发生故障时数据的一致性。 它使用标签选择器来将多个PersistentVolumeClaims分组以进行快照。 这个新特性仅支持CSI卷驱动器。详情请参见**Kubernetes 1.27：介绍用于磁盘卷组快照的新API**。

- kubectl apply裁剪更安全、更高效

  在Kubernetes 1.5版本中，kubectl apply引入了--prune标志来删除不再需要的资源，允许kubectl apply自动清理从当前配置中删除的资源。然而，现有的--prune实现存在设计缺陷，会降低性能并导致意外行为。Kubernetes 1.27中，kubectl apply提供基于ApplySet的剪裁方式，当前处于Alpha阶段，详情请参见**使用配置文件对Kubernetes对象进行声明式管理**。

- 为NodePort Service分配端口时避免冲突

  在Kubernetes 1.27中，您可以启用新的**特性门控** ServiceNodePortStaticSubrange，为NodePort Service使用不同的端口分配策略，减少冲突的风险。当前该特性处于Alpha阶段。

- 原地调整Pod资源

  在Kubernetes 1.27中，允许用户调整分配给Pod的CPU和内存资源大小，而无需重新启动容器。当前该特性处于Alpha阶段，详情请参见**纵向弹性伸缩**。

- 加快Pod启动

  在Kubernetes 1.27中进行了一系列的参数调整，以提高Pod的启动速度，例如并行镜像拉取、提高Kubelet默认API每秒查询限值等。详情请参见**Kubernetes 1.27：关于加快Pod启动的进展**。

- KMS V2进入Beta阶段

  Kubernetes中的密钥管理KMS v2 API进入Beta阶段，对KMS加密提供程序的性能进行了重大改进。详情请参见**使用KMS驱动进行数据加密**。

弃用和移除

- 在Kubernetes 1.27版本，针对卷扩展GA特性的以下特性门禁将被移除，且不得再在--feature-gates标志中引用。（**ExpandCSIVolumes，ExpandInUsePersistentVolumes，ExpandPersistentVolumes**）
- 在Kubernetes 1.27版本，移除--master-service-namespace 命令行参数。该参数支持指定在何处创建名为kubernetes的Service来表示API服务器。自v1.26版本已被弃用，1.27版本正式移除。

- 在Kubernetes 1.27版本，移除ControllerManagerLeaderMigration特性门禁。**Leader Migration**提供了一种机制，让HA集群在升级多副本的控制平面时通过在kube-controller-manager和cloud-controller-manager这两个组件之间共享的资源锁，安全地迁移“特定于云平台”的控制器。特性自v1.24正式发布，被无条件启用， 在v1.27版本中此特性门禁选项将被移除。
- 在Kubernetes 1.27版本，移除--enable-taint-manager命令行参数。该参数支持的特性基于污点的驱逐已被默认启用，且在标志被移除时也将继续被隐式启用。
- 在Kubernetes 1.27版本，移除--pod-eviction-timeout 命令行参数。弃用的命令行参数--pod-eviction-timeout将被从kube-controller-manager中移除。
- 在Kubernetes 1.27版本，移除CSI Migration特性门禁。**CSI migration**程序允许从树内卷插件移动到树外CSI驱动程序。CSI迁移自Kubernetes v1.16起正式发布，关联的CSIMigration特性门禁将在v1.27中被移除。
- 在Kubernetes 1.27版本，移除CSIInlineVolume特性门禁。**CSI Ephemeral Volume**特性允许在Pod规约中直接指定CSI卷作为临时使用场景。这些CSI卷可用于使用挂载的卷直接在Pod内注入任意状态，例如配置、Secret、身份、变量或类似信息。 此特性在v1.25中进阶至正式发布。因此，此特性门禁CSIInlineVolume将在v1.27版本中移除。
- 在Kubernetes 1.27版本，移除EphemeralContainers特性门禁。对于Kubernetes v1.27，临时容器的API支持被无条件启用；EphemeralContainers特性门禁将被移除。
- 在Kubernetes 1.27版本，移除LocalStorageCapacityIsolation特性门禁。**Local Ephemeral Storage Capacity Isolation**特性在 v1.25 中进阶至正式发布。此特性支持emptyDir卷这类Pod之间本地临时存储的容量隔离，因此可以硬性限制Pod对共享资源的消耗。如果本地临时存储的消耗超过了配置的限制，kubelet将驱逐Pod。特性门禁LocalStorageCapacityIsolation将在v1.27版本中被移除。
- 在Kubernetes 1.27版本，移除NetworkPolicyEndPort特性门禁。Kubernetes v1.25版本将NetworkPolicy中的endPort进阶至正式发布。支持endPort字段的NetworkPolicy提供程序可用于指定一系列端口以应用NetworkPolicy。
- 在Kubernetes 1.27版本，移除StatefulSetMinReadySeconds特性门禁。对于作为StatefulSet一部分的Pod，只有当Pod至少在**minReadySeconds**中指定的持续期内可用（并通过检查）时，Kubernetes才会将此Pod标记为只读。 该特性在Kubernetes v1.25中正式发布，StatefulSetMinReadySeconds特性门禁将锁定为true，并在v1.27版本中被移除。
- 在Kubernetes 1.27版本，移除IdentifyPodOS特性门禁。启用该特性门禁，您可以为Pod指定操作系统，此项特性支持自v1.25版本进入稳定。IdentifyPodOS特性门禁将在Kubernetes v1.27中被移除。
- 在Kubernetes 1.27版本，移除DaemonSetUpdateSurge特性门禁。Kubernetes v1.25版本还稳定了对DaemonSet Pod的浪涌支持，其实现是为了最大限度地减少部署期间DaemonSet的停机时间。DaemonSetUpdateSurge特性门禁将在Kubernetes v1.27中被移除。
- 在Kubernetes 1.27版本，移除--container-runtime 命令行参数。kubelet 接受一个已弃用的命令行参数--container-runtime， 并且在移除dockershim代码后，唯一有效的值将是remote。 Kubernetes v1.27将移除该参数，该参数自v1.24版本以来已被弃用。

参考链接

关于Kubernetes 1.27与其他版本的性能对比和功能演进的更多信息，请参考：**Kubernetes v1.27 Release Notes**

1.1.3 Autopilot 集群补丁版本发布记录

索引

v1.31 版本（公测）

表 1-2 v1.31 补丁版本发布说明

Autopilot集群补丁版本号	Kubernetes社区版本	特性更新	优化增强	安全漏洞修复
v1.31.1-r0	v1.31.1	CCE Autopilot支持创建1.31集群版本，有关更多信息请参见Kubernetes 1.31版本说明。	-	修复部分安全问题。

v1.28 版本

表 1-3 v1.28 补丁版本发布说明

Autopilot集群补丁版本号	Kubernetes社区版本	特性更新	优化增强	安全漏洞修复
v1.28.7-r0	v1.28.3	• 支持命名空间/工作负载绑定子网及安全组。 • 支持在SFS Turbo文件系统中创建子目录。	-	修复部分安全问题。
v1.28.6-r0	v1.28.3	• 支持工作负载实例挂载EVS存储。 • 支持命名空间/工作负载绑定子网及安全组。 • 支持自定义磁盘空间。	• 增强集群运维监控能力。 • 持续优化Pod启动时间。 • 优化autopilot大规模集群性能。	修复部分安全问题。
v1.28.5-r0	v1.28.3	• 支持创建工作负载时配置APM探针。 • 支持使用内网IP访问kube-apiserver。	-	修复部分安全问题。

Autopilot集群补丁版本号	Kubernetes社区版本	特性更新	优化增强	安全漏洞修复
v1.28.4-r0	v1.28.3	• 支持使用网络、磁盘等自定义指标创建弹性伸缩策略。 • 支持公网访问kube-apiserver。	使用YAML创建应用时自动忽略Autopilot不支持且不影响业务功能的配置参数。	修复部分安全问题。
v1.28.3-r0	v1.28.3	• Everest存储插件后台托管。 • 支持使用OBS存储卷。	-	修复部分安全问题。
v1.28.2-r0	v1.28.3	• 支持CronHPA定时扩缩容策略。 • 支持实例配置Security Context。	-	修复部分安全问题。
v1.28.1-r10	v1.28.3	CCE Autopilot支持创建1.28集群版本，有关更多信息请参见**Kubernetes 1.28版本说明**。	-	-

v1.27 版本

表 1-4 v1.27 补丁版本发布说明

Autopilot集群补丁版本号	Kubernetes社区版本	特性更新	优化增强	安全漏洞修复
v1.27.9-r0	v1.27.4	• 支持命名空间/工作负载绑定子网及安全组。 • 支持在SFS Turbo文件系统中创建子目录。	-	修复部分安全问题。
v1.27.8-r0	v1.27.4	• 支持工作负载实例挂载EVS存储。 • 支持命名空间/工作负载绑定子网及安全组。 • 支持自定义磁盘空间。	• 增强集群运维监控能力。 • 持续优化Pod启动时间。 • 优化autopilot大规模集群性能。	修复部分安全问题。

Autopilot集群补丁版本号	Kubernetes社区版本	特性更新	优化增强	安全漏洞修复
v1.27.7-r0	v1.27.4	• 支持创建工作负载时配置APM探针。 • 支持使用内网IP访问kube-apiserver。	-	修复部分安全问题。
v1.27.6-r0	v1.27.4	• 支持使用网络、磁盘等自定义指标创建弹性伸缩策略。 • 支持公网访问kube-apiserver。	使用YAML创建应用时自动忽略Autopilot不支持且不影响业务功能的配置参数。	修复部分安全问题。
v1.27.5-r0	v1.27.4	• Everest存储插件后台托管。 • 支持使用OBS存储卷。	-	修复部分安全问题。
v1.27.4-r0	v1.27.4	• 支持CronHPA定时扩缩容策略。 • 支持实例配置Security Context。	-	修复部分安全问题。
v1.27.3-r30	v1.27.4	-	支持一键配置监控告警。	修复部分安全问题。
v1.27.3-r20	v1.27.4	• 支持安装NGINX Ingress控制器插件。 • 支持安装云原生监控插件以及云原生日志插件，实现对应用指标的监控以及应用日志采集。 • 支持应用模板市场。 • 支持自定义资源(CRD)的使用。 • 支持对接Cloudshell。	优化集群创建时默认创建NAT网关以便于应用访问公网。	修复部分安全问题。
v1.27.3-r10	v1.27.4	CCE Autopilot支持创建1.27集群版本，有关更多信息请参见**Kubernetes 1.27版本说明**。	-	-

1.2 购买 CCE Autopilot 集群

CCE Autopilot 集群是基于CCI提供的具备完善的Kubernetes集群的管理平台，CCE Autopilot集群可以为您提供原生Kubernetes的各种拓展接口，同时与CCI一样让您无需创建和管理服务器节点即可直接运行容器应用，帮助您专注于应用开发，提升应用开发效率。同时，使用CCE Autopilot集群时，您只需按照应用的资源使用量付费，大大降低您的使用成本。

约束与限制

- 集群一旦创建以后，不支持变更以下项：
  - 变更集群类型。
  - 变更集群的网络配置，如所在的虚拟私有云VPC、容器子网、服务网段、kube-proxy代理模式（即服务转发模式）。
- 使用Autopilot集群时需注意周边资源的配额限制，请您预留足够的配额，每个集群占用的资源详情如下表。

服务	配额项	最小占用量	最小占用量说明	Region限额说明	配额申请
云容器引擎	集群	1	-	一个Region下每个账号可创建的集群总数限制为50个。	我的配额
虚拟私有云	VPC	1/集群	每个集群需要选择1个VPC，为集群提供隔离、私密的虚拟网络环境。	一个Region下每个账号可创建的VPC限制为5个。	
	子网	1/集群	每个集群至少需要选择1个子网用于分配容器IP。 其中集群控制面会默认占用8个IP地址，用于集群控制面部署及外部服务对接。	一个Region下每个账号可创建的子网限制为50个。	
	安全组	2/集群	每个集群会自动创建2个安全组，分别用于集群控制面和ENI的网络访问控制。	一个Region下每个账号可创建的安全组限制为100个。	
	安全组规则	7/集群	每个集群会自动创建7个的安全组规则，用于放通指定端口，以保证集群中的正常网络通信。	一个Region下每个账号可创建的安全组规则限制为1000个。	
VPC终端节点	终端节点	3/集群	每个集群至少需要预留3个终端节点配额，使集群可正常访问SWR、OBS等周边服务。	一个Region下每个账号可创建的终端节点限制为50个。	

服务	配额项	最小占用量	最小占用量说明	Region限额说明	配额申请
云解析服务	DNS内网域名	2/集群	每个集群至少需要占用2个DNS内网域名配额，用于实现集群内或跨集群服务间的正常通信。	一个Region下每个账号可创建的DNS内网域名限制为50个。	
	DNS记录集	6/集群	每个集群至少需要占用6个DNS记录集配额，用于提供集群内指定域名到IP地址或其他数据的解析规则。	一个Region下每个账号可创建的DNS记录集限制为500个。	

步骤一：登录 CCE 控制台

步骤1 登录**CCE控制台**。

步骤2 在“集群管理”页面右上角单击“购买集群”。

----结束

步骤二：配置集群

在“购买集群”页面，填写集群配置参数。

基础配置

参数	说明
集群类型	选择“CCE Autopilot集群”。
集群名称	请输入集群名称，同一账号下集群不可重名。
企业项目	该参数仅对开通企业项目的企业客户账号显示。 企业项目是一种云资源管理方式，企业项目管理服务提供统一的云资源按项目管理，以及项目内的资源管理、成员管理。了解更多企业项目相关信息，请查看**企业管理**。 选择某企业项目（如：default）后，集群及集群下的资源将创建到所选企业项目下。为方便管理资源，在集群创建成功后，建议不要修改集群下资源的企业项目。
集群版本	选择集群使用的Kubernetes版本。

网络配置

参数	说明
虚拟私有云	选择集群所在的虚拟私有云VPC，如没有可选项可以单击右侧“新建虚拟私有云”创建。集群创建后不可修改。
容器子网	选择容器所在子网，如没有可选项可以单击右侧“新建子网”创建。容器子网决定了集群下容器的数量上限，集群创建后支持新增子网。
服务网段	同一集群下容器互相访问时使用的Service资源的网段。决定了Service资源的上限。 集群创建后不可修改。
镜像访问	为确保您的集群节点可以从容器镜像服务中拉取镜像，默认使用所选的VPC中已有的终端节点，否则系统将为您新建SWR和OBS的终端节点。 终端节点将产生一定费用，详情请参见**价格计算器**。
配置SNAT	开启后您的集群可以通过NAT网关访问公网，默认使用所选的VPC中已有的NAT网关，否则系统将会为您自动创建一个默认规格的NAT网关并绑定弹性公网IP，自动配置SNAT规则。 使用NAT网关将产生一定费用，详情请参见**价格计算器**。

高级配置（可选）

参数	说明
告警中心	告警中心提供完善的集群告警能力，使您的集群在运行过程中发生故障及时预警，确保业务稳定。开启后将会创建默认告警规则，并发送告警通知到所选择的联系组。详细介绍请参见**通过告警中心一键配置告警**。
资源标签	通过为资源添加标签，可以对资源进行自定义标记，实现资源的分类。 您可以在TMS中创建“预定义标签”，预定义标签对所有支持标签功能的服务资源可见，通过使用预定义标签可以提升标签创建和迁移效率。具体请参见**创建预定义标签**。 • 标签键只能包含中文、英文字母、数字、空格和特殊字符(-_.:=+@)，长度不超过128个字符，不能以_sys_开头。资源标签键不可以为空。 • 标签值只能包含中文，英文字母、数字、空格和特殊字符(-_.:/=+@)，长度不超过255个字符。资源标签值可以为空。
集群描述	支持200个字符。

步骤三：插件选择

单击“下一步：插件选择”，选择创建集群时需要安装的插件。

基础功能

参数	说明
CoreDNS域名解析	默认安装**CoreDNS域名解析**插件，可为集群提供域名解析、连接云上DNS服务器等能力。

可观测性

参数	说明
Kubernetes Metrics Server	默认安装**Kubernetes Metrics Server**，为集群采集资源使用指标，例如容器CPU和内存使用率。
云原生监控插件	可选插件。勾选后自动安装**云原生监控插件**，为集群提供普罗指标采集能力，并将指标上报至指定的AOM实例。轻量化模式暂不支持基于自定义普罗语句的HPA，若需要相关功能，可在集群创建完成后手动安装全量的插件。
云原生日志采集插件	可选插件。勾选后自动安装**云原生日志采集插件**，将日志上报至LTS的日志采集器。集群创建完成后可在CCE日志中心页面对采集规则进行查询与管理。 LTS创建日志组免费，并每月赠送每个账号一定量免费日志采集额度，超过免费额度部分将产生费用（**价格计算器**）。关于如何采集自定义指标，请参见**收集容器日志**。

步骤四：插件配置

单击“下一步：插件配置”，配置插件。当前默认安装插件不支持配置，集群创建完成后，可前往“插件中心”修改配置。

可选插件配置如下：

可观测性

参数	说明
云原生监控插件	选择指标上报的AOM实例。如果没有可用实例，您可以单击“新建实例”进行创建。 AOM采集的基础指标免费，自定义指标将由AOM服务进行收费，详情请参见**价格详情**。

参数	说明
云原生日志采集插件	选择需要采集的日志。开启后将自动创建一个名称为k8s-log-{clusterId}的日志组，并为每个勾选的日志类型创建一个日志流。 • 容器日志：采集容器标准输出日志，对应的日志流名称为stdout-{clusterId}。 • Kubernetes事件：采集Kubernetes日志，对应的日志流名称为event-{clusterId}。 如果不开启日志采集能力。集群创建后可以前往CCE日志中心页面重新开启。 LTS创建日志组免费，并每月赠送每个账号一定量免费日志采集额度，超过免费额度部分将产生费用（**价格计算器**）。关于如何采集自定义指标，请参见**收集容器日志**。

步骤五：确认配置

参数填写完成后，单击“下一步：确认配置”，显示集群资源清单，确认无误后，单击“提交”。

集群创建预计需要5-10分钟，您可以单击“返回集群管理”进行其他操作或单击“查看集群事件列表”后查看集群详情。

相关操作

- 通过命令行工具连接集群：请参见**通过kubectl连接集群**。
- 通过kubectl对接多个集群：请参见**通过kubectl对接多个集群**

1.3 连接集群

1.3.1 通过 kubectl 连接集群

操作场景

本文将介绍如何通过kubectl连接集群。

权限说明

kubectl访问集群是通过集群上生成的配置文件（kubeconfig.json）进行认证，kubeconfig.json文件内包含用户信息，CCE根据用户信息的权限判断kubectl有权限访问哪些Kubernetes资源。即哪个用户获取的kubeconfig.json文件，kubeconfig.json就拥有哪个用户的信息，这样使用kubectl访问时就拥有这个用户的权限。

用户拥有的权限请参见**CCE Autopilot集群权限概述**。

使用 kubectl 连接集群

若您需要从客户端计算机连接到Kubernetes集群，可使用Kubernetes命令行客户端kubectl，您可登录CCE控制台，单击待连接集群名称，在集群“概览”页面查看访问地址以及kubectl的连接步骤。

图 1-2 集群连接信息

您需要先下载kubectl以及配置文件，复制到您的客户端机器，完成配置后，即可以访问Kubernetes集群。使用kubectl连接集群的步骤如下：

步骤1 准备环境

您需要准备一台**与集群同VPC**的虚拟机，并绑定弹性公网IP用于下载kubectl。

步骤2 下载kubectl

如果已经安装kubectl，则跳过此步骤，您可执行**kubectl version**命令判断是否已安装kubectl。

本文以Linux环境为例安装和配置kubectl，详情请参考**安装kubectl**。

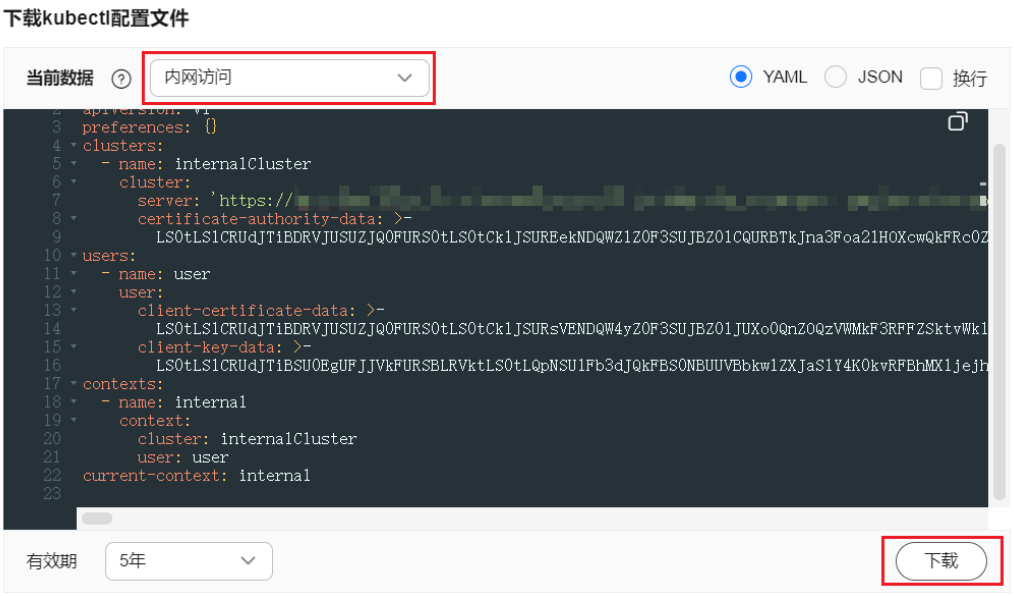
1. 登录到您的客户端机器，下载kubectl。
   ```
   cd /home
   curl -LO https://dl.k8s.io/release/{v1.25.0}/bin/linux/amd64/kubectl
   ```
   其中*{v1.25.0}*为指定的版本号，请根据集群版本进行替换。
2. 安装kubectl。
   ```
   chmod +x kubectl
   mv -f kubectl /usr/local/bin
   ```

步骤3 获取kubectl配置文件

在集群“概览”页中的“连接信息”版块，单击kubectl后的“配置”按钮，查看kubectl的连接信息，并在弹出页面中选择访问方式，然后下载对应的配置文件。

- Autopilot集群版本范围为v1.27.7-r0、v1.28.5-r0及以上：支持“内网域名访问”、“内网IP访问”和“公网访问”。
- Autopilot集群版本范围为v1.27.7-r0、v1.28.5-r0以下：支持“内网访问”和“公网访问”，其中“内网访问”仅支持使用域名，不支持使用内网IP。

图 1-3 下载配置文件



说明

- kubectl配置文件（kubeconfig）用于对接认证集群，请您妥善保存该认证凭据，防止文件泄露后，集群有被攻击的风险。
- IAM用户下载的配置文件所拥有的Kubernetes权限与CCE控制台上IAM用户所拥有的权限一致。
- 如果Linux系统里面配置了KUBECONFIG环境变量，kubectl会优先加载KUBECONFIG环境变量，而不是$home/.kube/config，使用时请注意。

步骤4 配置kubectl

以Linux环境为例配置kubectl。

1. 登录到您的客户端机器，复制**步骤3**中下载的配置文件（以kubeconfig.yaml为例）到您客户端机器的/home目录下。
2. 配置kubectl认证文件。

```
cd /home
mkdir -p $HOME/.kube
mv -f kubeconfig.yaml $HOME/.kube/config
```

3. 根据使用场景，切换kubectl的访问模式。
   - VPC网络内域名接入访问请执行：
     ```
     kubectl config use-context internal
     ```
   - 互联网接入访问请执行（集群需绑定公网地址）：
     ```
     kubectl config use-context external
     ```
   - 互联网接入访问如需开启双向认证请执行（集群需绑定公网地址）：
     ```
     kubectl config use-context externalTLSVerify
     ```

----结束

常见问题

- **Error from server Forbidden**

  使用kubectl在创建或查询Kubernetes资源时，显示如下内容：

```
# 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"
```

原因是用户没有操作该Kubernetes资源的权限，请参见**命名空间权限（Kubernetes RBAC授权）**为用户授权。

- **The connection to the server localhost:8080 was refused**

  使用kubectl在创建或查询Kubernetes资源时，显示如下内容：

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

  原因是由于该kubectl客户端未配置集群认证，请参见**配置kubectl认证文件**进行配置。

相关操作

- **通过kubectl对接多个集群**
- **通过配置kubeconfig文件实现集群权限精细化管理**

1.3.2 通过 CloudShell 连接集群

操作场景

本文将介绍如何通过CloudShell连接集群。

权限说明

在CloudShell中使用kubectl时，kubectl的权限由登录用户的权限决定。

使用 CloudShell 连接集群

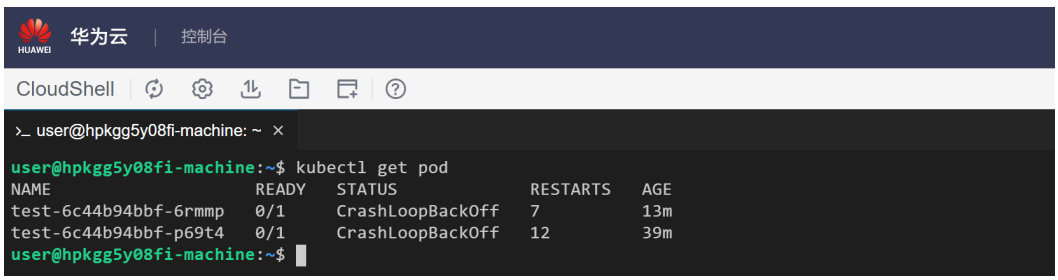
CloudShell是一款用于管理与运维云资源的网页版Shell工具，CCE支持使用CloudShell连接集群，如**图1-4**所示，单击“命令行工具”即可在CloudShell中使用kubectl访问集群。

说明

- CloudShell中kubectl证书有效期为1天，从云容器引擎重新跳转可以重置有效期。
- 集群必须安装CoreDNS才能使用CloudShell。
- CloudShell暂不支持委托账号和子项目。

图 1-4 CloudShell

图 1-5 在 CloudShell 中使用 kubectl



1.3.3 通过 X509 证书连接集群

操作场景

通过控制台获取集群证书，使用该证书可以访问Kubernetes集群。

操作步骤

步骤1 登录CCE控制台，单击集群名称进入集群。

步骤2 查看集群概览页，在右边“连接信息”下证书认证一栏，单击“下载”。

图 1-6 获取证书

步骤3 在弹出的“证书获取”窗口中，根据系统提示选择证书的过期时间并下载集群X509证书。

须知

- 下载的证书包含client.key、client.crt、ca.crt三个文件，请妥善保管您的证书，不要泄露。
- 集群中容器之间互访不需要证书。

步骤4 使用集群证书调用Kubernetes原生API。

例如使用curl命令调用接口查看Pod信息，如下所示，其中*****:5443*为集群API Server的内网或公网地址。

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

更多集群接口请参见**Kubernetes API**。

----结束

1.3.4 配置集群 API Server 公网访问

您可以为Kubernetes集群的API Server绑定弹性公网IP（EIP），使集群API Server具备公网访问能力。

为 API Server 绑定 EIP

步骤1 登录CCE控制台，单击集群名称进入集群。

步骤2 查看集群概览页，在右边“连接信息”下公网地址一栏，单击“绑定”。

步骤3 选择一个已有的弹性公网IP。如果无可用IP，可单击“创建弹性IP”前往EIP控制台进行创建。

说明

- 通过绑定EIP实现公网访问，集群存在风险，建议绑定的EIP配置DDoS高防服务或**配置API Server访问策略**。
- 绑定EIP将会短暂重启集群API Server并更新kubeconfig证书，请避免在此期间操作集群。

步骤4 单击“确定”。

----结束

配置 API Server 访问策略

集群的API Server绑定EIP将会暴露到互联网，存在被攻击的风险，建议修改集群控制节点安全组规则。

步骤1 登录CCE控制台，单击集群名称进入集群，在概览页面找到“集群ID”并复制。

步骤2 登录VPC控制台，在左侧导航栏中选择“访问控制 > 安全组”。

步骤3 在筛选栏中，选择筛选条件为“描述”，并粘贴集群ID进行筛选。

步骤4 筛选结果中将会包含多个安全组，找到控制节点的安全组（以[cce集群名称]-cce-control开头），单击“配置规则”。

步骤5 添加入方向的5443端口规则，单击“添加规则”。

您可以根据需求修改允许访问的源地址。例如，客户端需要访问API Server的IP为100.*.*.*，则可添加5443端口入方向规则的源地址为100.*.*.*。

步骤6 修改完成后单击“确认”。

----结束

1.4 管理集群

1.4.1 删除集群

注意事项

- 删除集群会删除集群下的工作负载与服务，相关业务将无法恢复。在执行操作前，请确保相关数据已完成备份或者迁移，删除完成后数据无法找回，请谨慎操作。

  部分资源不会删除：

  - Service和Ingress关联的已有ELB实例（仅删除自动创建的ELB实例）
  - 关联创建的VPC级别的资源（如终端节点、NAT网关、SNAT出网EIP）
- 在集群非运行状态（例如冻结、不可用状态）时删除集群，会残留存储、网络等关联资源，请妥善处理。

删除集群

步骤1 登录CCE控制台，在左侧导航栏中选择“集群管理”。

步骤2 找到需要删除的集群，查看集群的更多操作，并单击“删除集群”。

步骤3 在弹出的“删除集群”窗口中，根据系统提示，勾选删除集群时需要释放的资源。

- 删除集群下云存储资源（删除存储卷PV绑定的云硬盘等底层存储）。
- 删除集群下负载均衡ELB等网络资源（仅删除自动创建的ELB资源）。
- 其余关联创建的VPC级别的资源（如终端节点、NAT网关、SNAT出网EIP）在删除集群时默认保留，请确认其他集群或服务未重用该资源，并前往网络控制台进行删除。

步骤4 单击“是”，开始执行删除集群操作。

删除集群需要花费1~3分钟，请耐心等候。

----结束

1.5 升级集群

1.5.1 升级概述

云容器引擎（CCE）严格遵循社区一致性认证，每年发布3个集群版本，每个版本发布后提供至少24个月的维护周期，CCE保证维护周期内的集群版本的稳定运行。

为了保障您的服务权益，请您务必在维护周期结束之前升级您的Kubernetes集群，主动升级集群有以下好处：

- 降低安全和稳定性风险：Kubernetes版本迭代过程中，会不断修复发现的安全及稳定性漏洞，长久使用EOS版本集群会给业务带来安全和稳定性风险。
- 支持新功能和新操作系统：Kubernetes版本的迭代过程中，会不断带来新的功能、优化。
- 避免大跨度兼容风险：Kubernetes版本的迭代过程中，会不断带来API变更与功能废弃。长久未升级的集群，在需要升级时需要更大的运维保障投入。周期性的跟随升级能有效缓解版本差异累积导致的兼容性风险。建议用户每季度升级一次补丁版本，每年升级一次大版本至当前支持的最新版本。

- 更加有效的技术支持：对于EOS的Kubernetes版本集群，CCE不再提供安全补丁和问题修复，同样无法保证EOS版本集群的技术支持质量。

您可在集群列表页面确认集群的Kubernetes版本，以及当前是否有新的版本可供升级。若有需要请联系工作人员提供专家咨询服务。

集群升级路径

CCE Autopilot集群基于社区Kubernetes版本迭代演进，版本号由社区Kubernetes版本和补丁版本两部分构成，因此提供两类集群升级路径。

- Kubernetes版本升级：

Kubernetes版本号	支持升级到的Kubernetes版本号
v1.27	v1.28
v1.28	/

说明

补丁版本需要升级至最新补丁后方可进行Kubernetes版本升级，控制台将根据当前集群版本自动为您生成最佳升级路径。

- 补丁版本升级

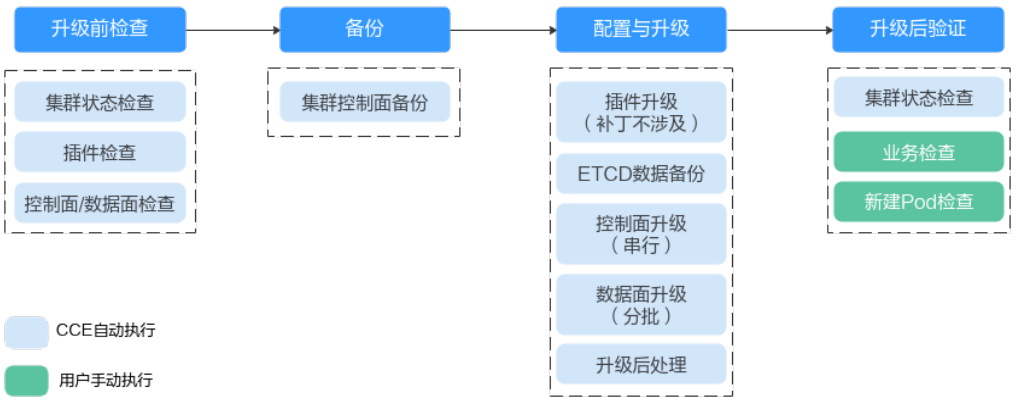
  CCE Autopilot集群采取了补丁版本管理策略，旨在不进行大版本升级的情况下，为在维集群提供新的特性、Bugfix和漏洞修复。

  新的补丁版本发布以后您可在任意补丁版本一次直升到最新补丁版本。

集群升级流程

集群升级流程包括升级前检查、备份、升级和升级后验证几个步骤，下面介绍集群升级过程中的相关流程。

图 1-7 集群升级流程



在确定集群的目标版本后，请您仔细阅读升级**注意事项**，避免升级时出现功能不兼容的问题。

1. **升级前检查**

   升级集群前，CCE会对您的集群进行必要的检查，包括集群状态、插件状态、工作负载兼容性等多方面进行检查，确保集群满足升级的条件，检查项目请参考**升级前检查异常问题排查**。如果出现检查异常项，请参考控制台中的提示进行修复。

2. **备份**

   通过硬盘快照的方式帮您备份集群控制面数据，以保存CCE组件镜像、组件配置、Etcd数据等关键数据。建议您在升级前进行备份。如果在升级过程中出现不可预期的情况，可以基于备份为您快速恢复集群。

备份方式	备份对象	备份方式	备份时间	回滚时间	说明
etcd数据备份	etcd数据	升级流程中自动备份	1-5min	2h	必选备份，升级过程中自动进行，用户无需关注
EVS快照备份	控制面数据，包括组件镜像、配置、日志以及etcd数据	通过页面一键备份（手动触发）	1-5min	20min	-

3. **配置与升级**

   执行升级前，需要对升级参数进行配置，我们已为您提供了默认配置，您也可以根据需要进行配置，升级参数配置完成后，将进入正式升级流程，对插件、控制面、数据面依次进行升级。

   - **插件升级配置**：此处列出了您的集群中已安装的插件。在集群升级过程中系统会自动升级已选择的插件，以兼容升级后的集群版本，您可以单击插件右侧的“配置”重新定义插件参数。

     说明

     插件右侧如有⚠标记，表示当前插件不能同时兼容集群升级起始和目标版本，在集群版本升级完成后将为您升级该插件 ，该插件在集群升级过程中可能无法正常使用。

4. **升级后验证**

   升级完成后，会自动为集群执行集群状态检查等，您需要手动进行业务验证、新建Pod验证等，确保升级后集群功能正常。

1.5.2 升级前须知

升级前，您可以在CCE控制台确认您的集群是否可以进行升级操作。确认方法请参见**升级概述**。

注意事项

升级集群前，您需要知晓以下事项：

- **请务必慎重并选择合适的时间段进行升级，以减少升级对您的业务带来的影响。**
- 集群升级前，请参考**Kubernetes版本发布记录**了解每个集群版本发布的特性以及差异，否则可能因为应用不兼容新集群版本而导致升级后异常。例如，您需要检查集群中是否使用了目标版本废弃的API，否则可能导致升级后调用接口失败。

集群升级时，以下几点注意事项可能会对您的业务存在影响，请您关注：

- 集群升级前，**请确认集群中未执行高危操作**，否则可能导致集群升级失败或升级后配置丢失。例如，常见的高危操作有通过ELB控制台修改CCE管理的监听器配置等。建议您通过CCE控制台修改相关配置，以便在升级时自动继承。

约束与限制

集群升级出现异常时，集群可通过备份数据进行回滚。若您在升级成功之后对集群进行了其它操作（例如变更集群规格），将无法再通过备份数据回滚。

废弃 API 说明

随着Kubernetes API的演化，API会周期性地被重组或升级，老的API会被弃用并被最终删除。以下为各Kubernetes社区版本中被废弃的API，更多已废弃的API说明请参见**已弃用API的迁移指南**。

说明

当某API被废弃时，已经创建的资源对象不受影响，但新建或编辑该资源时将出现API版本被拦截的情况。

升级备份说明

目前集群升级备份方式如下：

备份方式	备份对象	备份方式	备份时间	回滚时间	说明
etcd数据备份	etcd数据	升级流程中自动备份	1-5min	2h	必选备份，升级过程中自动进行，用户无需关注
EVS快照备份	控制面数据，包括组件镜像、配置、日志以及etcd数据	通过页面一键备份（手动触发）	1-5min	20min	-

1.5.3 手动升级

您可以通过云容器引擎管理控制台升级集群版本，以支持新特性的使用。

升级前，请先了解CCE各集群版本能够升级到的目标版本，以及升级方式和升级影响，详情请参见**升级概述**和**升级前须知**。

注意事项

- 集群升级过程中会自动升级插件到目标集群兼容的版本，升级过程中请不要卸载或者重装插件。
- 升级之前请确认所有的插件都处于运行状态，如果插件升级失败可以在插件问题修复后，重试升级。
- 若在集群升级过程中出现升级失败的提示，请参照提示信息修复问题后单击重试，若重试后仍未成功升级，请**提交工单**协助您进行修复。

更多注意事项请参见**升级前须知**。

操作步骤

集群升级步骤包括：升级前检查、备份、配置与升级、升级后处理。

步骤1 登录CCE控制台，单击集群名称进入集群。

步骤2 在左侧导航栏选择“集群升级”。

步骤3 根据当前集群版本，系统将为您生成最佳升级路径，您可以在该路径中选择需要升级的版本，确认集群版本差异、插件版本等信息，然后单击“前往升级”。

步骤4 进行升级前检查，单击“开始检查”并确认。如集群中存在异常项或风险项，请根据页面提示的检查结果进行处理，处理完成后需重新进行升级前检查。

- 异常项：请查看页面提示的解决方案并处理异常后，重新进行升级前检查。
- 风险项：表示该结果可能会影响集群升级结果，请您查看风险说明并确认您是否处于风险影响范围。如确认无风险，可单击该风险项后的“确认”按钮，手动跳过该风险项，然后重新进行升级前检查。

待升级前检查通过后，单击“下一步”。

步骤5 进行集群备份。集群升级过程中将自动进行etcd数据备份，您可手动进行控制面备份，以加快控制面升级失败时的回滚速度，如无需手动备份可直接单击“下一步”。

备份方式	备份对象	备份方式	备份时间	回滚时间	说明
etcd数据备份	etcd数据	升级流程中自动备份	1-5min	2h	必选备份，升级过程中自动进行，用户无需关注
EVS快照备份	控制面数据，包括组件镜像、配置、日志以及etcd数据	通过页面一键备份（手动触发）	1-5min	20min	-

步骤6 配置升级参数。

- **插件升级配置：**此处列出了您的集群中已安装的插件。在集群升级过程中系统会自动升级已选择的插件，以兼容升级后的集群版本，您可以单击插件右侧的“配置”重新定义插件参数。

  说明

  插件右侧如有  标记，表示当前插件不能同时兼容集群升级起始和目标版本，在集群版本升级完成后将为您升级该插件 ，该插件在集群升级过程中可能无法正常使用。

步骤7 配置完成后，单击“立即升级”按钮，并确认升级操作后集群开始升级。您可以在页面下方查看版本升级的进程。

说明

若在集群升级过程中出现升级失败的提示，请参照提示信息修复问题后重试。

步骤8 升级完成后，单击“下一步”，请根据页面提示的检查项进行升级后验证。确认所有检查项均正常后，可单击“完成”按钮，并确认完成升级后检查，详情请参见**升级后验证**。

您可以在集群列表页面查看集群当前的Kubernetes版本，确认升级成功。

----结束

常见问题

- **CCE集群升级时，升级集群插件失败如何排查解决?**

1.5.4 升级后验证

1.5.4.1 集群状态检查

检查项内容

集群升级后，需要检查集群状态是否为“运行中”状态。

检查步骤

系统会自动为您检查集群状态是否正常，您可以根据诊断结果前往集群列表页面进行确认。

解决方案

当集群状态异常时，请联系技术支持人员。

1.5.4.2 业务检查

检查项内容

集群升级完毕，由用户验证当前集群正在运行的业务是否正常。

检查步骤

业务不同，验证的方式也有所不同，建议您在升级前确认适合您业务的验证方式，并在升级前后均执行一遍。

常见的业务确认方式有：

- 业务界面可用
- 监控平台无异常告警与事件
- 关键应用进程无错误日志
- API拨测正常等

解决方案

若集群升级后您的在线业务有异常，请联系技术支持人员。

1.5.4.3 新建 Pod 检查

检查内容

检查集群升级后，是否能新建Pod。

检查步骤

请新建Pod，确保新的Pod能提供相同的业务服务。

步骤1 登录CCE控制台，单击集群名称进入集群。

步骤2 在导航栏中选择“工作负载”，单击右上角“创建工作负载”或“YAML创建”。创建工作负载的操作步骤详情请参见**创建工作负载**。

图 1-8 创建工作负载

建议您使用日常测试的镜像作为基础镜像。您可参照如下YAML部署最小应用Pod。

```
apiVersion: apps/v1
kind: Deployment
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
```

步骤3 负载创建完毕后请检查该工作负载的Pod状态是否正常。

步骤4 检查完毕后，在导航栏中选择“工作负载”，选择post-upgrade-check工作负载并单击“更多>删除”，删除该测试用工作负载。

----结束

解决方案

若Pod无法新建，或状态异常，请提交工单联系客服人员。

1.5.5 升级前检查异常问题排查

1.5.5.1 升级前检查项

集群升级前，系统将自动进行全面的升级前检查，当集群不满足升级前检查条件时将无法继续升级。为了能够更好地避免升级风险，本文提供全量的升级前检查问题及解决方案，帮助您对可能存在的升级故障进行预处理。

表 1-5 检查项列表

序号	检查项名称	检查项说明
1	升级管控检查异常处理	检查集群是否处于升级管控中。
2	插件检查异常处理	• 检查插件状态是否正常 • 检查插件是否支持目标版本
3	Helm模板检查异常处理	检查当前HelmRelease记录中是否含有目标集群版本不支持的K8s废弃API，可能导致升级后helm模板不可用。
4	Master节点SSH连通性检查异常处理	该检查通过尝试建立SSH连接，检查CCE是否能通过SSH方式连接至您的Master节点。
5	K8s废弃资源检查异常处理	检查集群是否存在对应版本已经废弃的资源。
6	cce-hpa-controller插件限制检查异常处理	检查cce-controller-hpa插件的目标版本是否存在兼容性限制。

序号	检查项名称	检查项说明
7	K8s废弃API检查异常处理	系统会扫描过去一天的审计日志，检查用户是否调用目标K8s版本已废弃的API。 **说明** 由于审计日志的时间范围有限，该检查项仅作为辅助手段，集群中可能已使用即将废弃的API，但未在过去一天的审计日志中体现，请您充分排查。
8	HTTPS类型负载均衡证书一致性检查异常处理	检查HTTPS类型负载均衡所使用的证书，是否在ELB服务侧被修改。
9	Pod配置检查	检查Pod配置是否存在兼容性限制。

1.5.5.2 升级管控检查异常处理

检查项内容

检查集群是否处于升级管控中。

解决方案

CCE基于以下几点原因，可能会暂时限制该集群的升级功能：

- 基于用户提供的信息，该集群被识别为核心重点保障的生产集群。
- 正在或即将进行其他运维任务，例如Master节点3AZ改造等。
- 集群中存在容器引擎为Docker但OS与节点池配置不同的节点，您可以重置这部分节点后再次执行升级前检查。

请根据界面日志联系技术支持人员了解限制原因并申请解除升级限制。

1.5.5.3 插件检查异常处理

检查项内容

当前检查项包括以下内容：

- 检查插件状态是否正常
- 检查插件是否支持目标版本

解决方案

- **问题场景一：插件状态异常**

  请登录CCE控制台，单击集群名称进入集群控制台，前往“插件中心”处查看并处理处于异常状态的插件。
- **问题场景二：目标集群版本不支持当前插件版本**

  检查到该插件由于兼容性等问题无法随集群自动升级。请您登录CCE控制台，在“插件中心”处进行手动升级。

- **问题场景三：插件升级到最新版本后，仍不支持目标集群版本**

  请您登录CCE控制台，单击集群名称进入集群控制台，在“插件中心”处进行手动卸载，具体插件支持版本以及替换方案可查看**插件**。

- **问题场景四：插件配置不满足在升级条件，请在插件升级页面升级插件之后重试**

  升级前检查出现以下报错：

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

  请您登录CCE控制台，在“插件中心”处手动升级插件。

1.5.5.4 Helm 模板检查异常处理

检查项内容

检查当前HelmRelease记录中是否含有目标集群版本不支持的K8s废弃API，可能导致升级后helm模板不可用。

解决方案

将HelmRelease记录中K8s废弃API转换为源版本和目标版本均兼容的API。

说明

该检查项解决方案已在升级流程中自动兼容处理，此检查不再限制。您无需关注并处理。

1.5.5.5 Master 节点 SSH 连通性检查异常处理

检查项内容

该检查通过尝试建立SSH连接，检查CCE是否能通过SSH方式连接至您的Master节点。

解决方案

SSH连通性检查可能有较低概率因为网络波动检查失败，请您优先重试升级前检查；

若重试检查仍无法通过检查，请您提交工单，联系技术支持人员排查。

1.5.5.6 K8s 废弃资源检查异常处理

检查项内容

检查集群是否存在对应版本已经废弃的资源。

解决方案

- **问题场景一： 1.25及以上集群中的service存在废弃的annotation：tolerate-unready-endpoints**

  报错日志信息如下：

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

  检查日志信息中所给出的service是否存在**"tolerate-unready-endpoints"**的annotation，如果存在则将其去掉，并在对应的service的spec中添加下列字段来替代该annotation：

```
publishNotReadyAddresses: true
```

- **问题场景二： 1.27及以上集群中的service存在废弃的annotation：service.kubernetes.io/topology-aware-hints**

  报错日志信息如下：

  ```
  some check failed in cluster upgrade: this cluster has deprecated service list: map[***] with deprecated annotation list [service.kubernetes.io/topology-aware-hints]
  ```

  检查日志信息中所给出的service是否存在**"service.kubernetes.io/topology-aware-hints"**的annotation，如果存在则将其去掉，并在对应的service中用**"service.kubernetes.io/topology-mode"**的annotation进行替换。

1.5.5.7 cce-hpa-controller 插件限制检查异常处理

检查项内容

检查cce-controller-hpa插件的目标版本是否存在兼容性限制。

解决方案

检测到目标cce-controller-hpa插件版本存在兼容性限制，需要集群安装能提供metrics api的插件，例如metrics-server；

请您在集群中安装相应metrics插件之后重试检查

1.5.5.8 K8s 废弃 API 检查异常处理

检查项内容

系统会扫描过去一天的审计日志，检查用户是否调用目标K8s版本已废弃的API。

说明

由于审计日志的时间范围有限，该检查项仅作为辅助手段，集群中可能已使用即将废弃的API，但未在过去一天的审计日志中体现，请您充分排查。

解决方案

检查说明

根据检查结果，检测到您的集群通过kubectl或其他应用调用了升级目标集群版本已废弃的API，您可在升级前进行整改，否则升级到目标版本后，该API将会被kube-apiserver拦截，影响您的使用。

v1.27版本升级v1.28版本时暂无版本兼容性差异。

1.5.5.9 HTTPS 类型负载均衡证书一致性检查异常处理

检查项内容

检查HTTPS类型负载均衡所使用的证书，是否在ELB服务侧被修改。

解决方案

该问题的出现，一般是由于用户在CCE中创建HTTPS类型Ingress后，直接在ELB证书管理功能中修改了Ingress引用的证书，导致CCE集群中存储的证书内容与ELB侧不一致，进而导致升级后ELB侧证书被覆盖。

步骤1 请登录ELB服务控制台，在“弹性负载均衡 > 证书管理”界面找到该证书，在证书描述字段中找到对应的secret_id。

图 1-9 查询证书

该secret_id即为集群中对应Secret的metadata.uid字段，可以根据该uid查询集群中Secret的名称。

您可以通过以下kubectl命令进行查询，其中*<secret_id>*请自行替换。

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

步骤2 您可以将Ingress使用的证书替换为ELB服务器证书，即可通过ELB控制台创建或编辑该证书。

1. 请登录CCE控制台，前往“服务”页面并选择“路由”页签，找到使用该证书的路由，单击“更多 > 更新”。注意，这里可能有多个Ingess引用该证书，所涉及的Ingress都需要进行更新，可以根据Ingress的yaml文件的spec.tls中secretName字段判断是否引用该Secret中的证书。

   您可以通过以下kubectl命令进行查询引用该证书的Ingress，其中*<secret_name>*请自行替换。

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

2. 在监听器配置中，选择服务器证书来源为“ELB服务器证书”，该证书可通过ELB控制台创建或编辑，单击“确定”。
3. 在“配置与密钥”界面删除对应的Secret，删除前建议先备份。

----结束

1.5.5.10 Pod 配置检查

检查项内容

检查Pod配置是否存在兼容性限制。

解决方案

问题场景：1.28.6-r0|1.27.8-r0以下版本的集群中，Pod存在系统预置的“annotation：resource.cce.io/extra-ephemeral-storage-in-GiB”，升级后可能会导致Pod创建失败或非预期的收费。

报错日志信息如下：


```
some check failed in cluster upgrade: this cluster has podlist: map[pod-xxx pod-xxx] with system preset annotation list [resource.cce.io/extra-ephemeral-storage-in-GiB]
```

检查日志信息中给出的Pod是否存在“resource.cce.io/extra-ephemeral-storage-in-GiB”的annotation。该annotation声明表示额外增加的临时存储空间大小，具体信息请参见**Pod注解**。

您可以通过以下两种方式解决该问题：

- 升级负载并删除annotation，在集群升级页面重新检查。集群升级后，新创建的Pod不再使用额外的临时存储，临时存储容量为默认的30GiB。
- 升级负载并修改annotation，将“resource.cce.io/extra-ephemeral-storage-in-GiB”设置正确的值，在集群升级页面跳过检查项。集群升级后，新创建的Pod使用额外增加的临时存储，且会收取相应的费用，收费标准请参见**计费说明**。

2 工作负载

2.1 工作负载概述

工作负载是在Kubernetes上运行的应用程序。无论您的工作负载是单个组件还是协同工作的多个组件，您都可以在Kubernetes上的一组Pod中运行它。在Kubernetes中，工作负载是对一组Pod的抽象模型，用于描述业务的运行载体，包括Deployment、StatefulSet、Job、CronJob等多种类型。

CCE Autopilot集群提供基于Kubernetes原生类型的容器部署和管理能力，支持容器工作负载部署、配置、监控、扩容、升级、卸载、服务发现及负载均衡等生命周期管理。

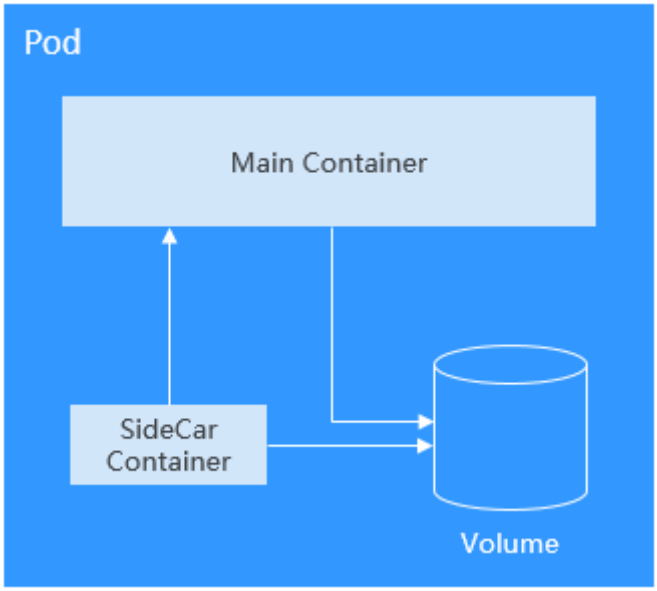
容器组（Pod）

容器组（Pod）是Kubernetes创建或部署的最小单位。一个Pod封装一个或多个容器（Container）、存储资源（Volume）、一个独立的网络IP以及管理控制容器运行方式的策略选项。

Pod使用主要分为两种方式：

- Pod中运行一个容器。这是Kubernetes最常见的用法，您可以将Pod视为单个封装的容器，但是Kubernetes是直接管理Pod而不是容器。
- Pod中运行多个需要耦合在一起工作、需要共享资源的容器。通常这种场景下应用包含一个主容器和几个辅助容器（SideCar Container），如图2-1所示，例如主容器为一个web服务器，从一个固定目录下对外提供文件服务，而辅助容器周期性的从外部下载文件存到这个固定目录下。

图 2-1 Pod

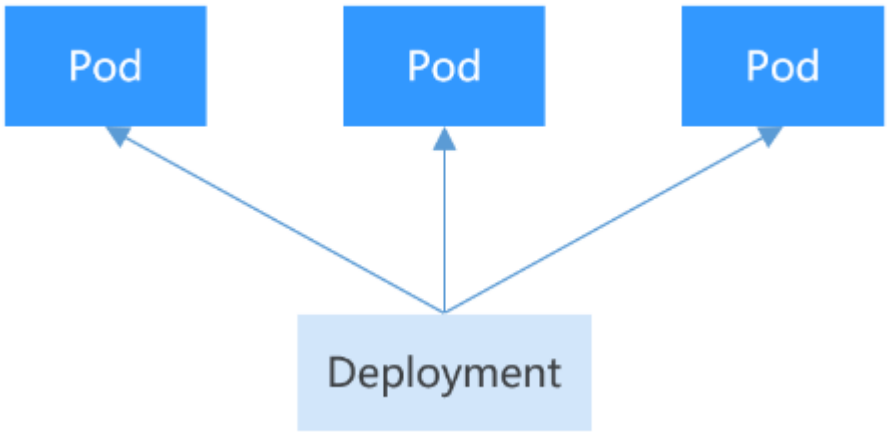


实际使用中很少直接创建Pod，而是使用Kubernetes中称为Controller的抽象层来管理Pod实例，例如Deployment和Job。Controller可以创建和管理多个Pod，提供副本管理、滚动升级和自愈能力。通常，Controller会使用Pod Template来创建相应的Pod。

无状态负载（Deployment）

Pod是Kubernetes创建或部署的最小单位，但是Pod是被设计为相对短暂的一次性实体，Pod可以被驱逐（当节点资源不足时）、随着集群的节点崩溃而消失。Kubernetes提供了Controller（控制器）来管理Pod，Controller可以创建和管理多个Pod，提供副本管理、滚动升级和自愈能力，其中最为常用的就是Deployment。

图 2-2 Deployment



一个Deployment可以包含一个或多个Pod副本，每个Pod副本的角色相同，所以系统会自动为Deployment的多个Pod副本分发请求。

Deployment集成了上线部署、滚动升级、创建副本、恢复上线的功能，在某种程度上，Deployment实现无人值守的上线，大大降低了上线过程的复杂性和操作风险。

有状态负载（StatefulSet）

Deployment控制器下的Pod都有个共同特点，那就是每个Pod除了名称和IP地址不同，其余完全相同。需要的时候，Deployment可以通过Pod模板创建新的Pod；不需要的时候，Deployment就可以删除任意一个Pod。

但是在某些场景下，这并不满足需求，比如有些分布式的场景，要求每个Pod都有自己单独的状态时，比如分布式数据库，每个Pod要求有单独的存储，这时Deployment无法满足业务需求。

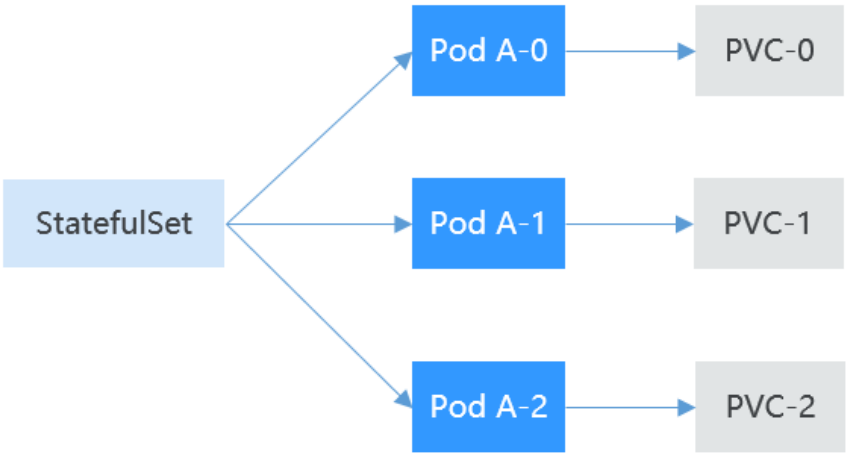
分布式有状态应用的特点主要是应用中每个部分的角色不同（即分工不同），比如数据库有主备、Pod之间有依赖，在Kubernetes中部署有状态应用对Pod有如下要求：

- Pod能够被别的Pod找到，要求Pod有固定的标识。
- 每个Pod有单独存储，Pod被删除恢复后，必须读取原来的数据，否则状态就会不一致。

Kubernetes提供了StatefulSet来解决这个问题，其具体如下：

1. StatefulSet给每个Pod提供固定名称，Pod名称增加从0-N的固定后缀，Pod重新调度后Pod名称和HostName不变。
2. StatefulSet通过Headless Service给每个Pod提供固定的访问域名。
3. StatefulSet通过创建固定标识的PVC保证Pod重新调度后还是能访问到相同的持久化数据。

图 2-3 StatefulSet



普通任务（Job）和定时任务（CronJob）

Job和CronJob是负责批量处理短暂的一次性任务（short lived one-off tasks），即仅执行一次的任务，它保证批处理任务的一个或多个Pod成功结束。

- Job：是Kubernetes用来控制批处理型任务的资源对象。批处理业务与长期伺服业务（Deployment、StatefulSet）的主要区别是批处理业务的运行有头有尾，而长期伺服业务在用户不停止的情况下永远运行。Job管理的Pod根据用户的设置把任务成功完成就自动退出（Pod自动删除）。
- CronJob：是基于时间的Job，就类似于Linux系统的crontab文件中的一行，在指定的时间周期运行指定的Job。

任务负载的这种用完即停止的特性特别适合一次性任务，比如持续集成。

工作负载生命周期说明

表 2-1 状态说明

状态	说明
运行中	所有实例都处于运行中、或实例数为0时显示此状态。
未就绪	容器处于异常、负载下实例没有正常运行时显示此状态。
处理中	负载没有进入运行状态但也没有报错时显示此状态。
可用	当多实例无状态工作负载运行过程中部分实例异常，可用实例不为0，工作负载会处于可用状态。
执行完成	任务执行完成，仅Job存在该状态。
删除中	触发删除操作后，工作负载会处于删除中状态。

2.2 创建工作负载

2.2.1 创建无状态负载（Deployment）

操作场景

在运行中始终不保存任何数据或状态的工作负载称为“无状态负载 Deployment”，例如nginx。您可以通过控制台或kubectl命令行创建无状态负载。

前提条件

- 在创建容器工作负载前，您需要存在一个可用集群。若没有可用集群，请参照**购买CCE Autopilot集群**中内容创建。
- 在创建容器工作负载前，您需要配置访问SWR和OBS服务的VPC终端节点，详情请参见**配置访问SWR和OBS服务的VPC终端节点**。
- 若工作负载需要被外网访问，请为工作负载创建负载均衡类型的服务。

  说明

  单个实例（Pod）内如果有多个容器，请确保容器使用的端口不冲突 ，否则部署会失败。

通过控制台创建

步骤1 登录CCE控制台。

步骤2 单击集群名称进入集群，在左侧选择“工作负载”，在右上角单击“创建工作负载”。

步骤3 配置工作负载的信息。

基本信息

- 负载类型：选择无状态工作负载Deployment。

- 负载名称：填写工作负载的名称。请输入1到63个字符的字符串，可以包含小写英文字母、数字和中划线（-），并以小写英文字母开头，小写英文字母或数字结尾。
- 命名空间：选择工作负载的命名空间，默认为default。您可以单击后面的“创建命名空间”，命名空间的详细介绍请参见**创建命名空间**。
- 实例数量：填写实例的数量，即工作负载Pod的数量。

容器配置

- 容器信息

  Pod中可以配置多个容器，您可以单击右侧“添加容器”为Pod配置多个容器。

  - 基本信息：配置容器的基本信息。

参数	说明
容器名称	为容器命名。
更新策略	镜像更新/拉取策略。可以勾选“总是拉取镜像”，表示每次都从镜像仓库拉取镜像；如不勾选则优使用节点已有的镜像，如果没有这个镜像再从镜像仓库拉取。
镜像名称	单击后方“选择镜像”，选择容器使用的镜像。 如果需要使用第三方镜像，请参见**使用第三方镜像**。
镜像版本	选择需要部署的镜像版本。
CPU配额	CPU资源限制值，即允许容器使用的CPU最大值，防止占用过多资源。
内存配额	内存资源限制值，即允许容器使用的内存最大值。如果超过，容器会被终止。
初始化容器（可选）	选择容器是否作为初始化（Init）容器。初始化（Init）容器不支持设置健康检查。 Init容器是一种特殊容器，可以在Pod中的其他应用容器启动之前运行。每个Pod中可以包含多个容器，同时Pod中也可以有一个或多个先于应用容器启动的Init容器，当所有的Init 容器运行完成时，Pod中的应用容器才会启动并运行。详细说明请参见**Init容器**。

  - 生命周期（可选）：在容器的生命周期的特定阶段配置需要执行的操作，例如启动命令、启动后处理和停止前处理，详情请参见**设置容器生命周期**。
  - 健康检查（可选）：根据需求选择是否设置存活探针、就绪探针及启动探针，详情请参见**设置容器健康检查**。
  - 环境变量（可选）：支持通过键值对的形式为容器运行环境设置变量，可用于把外部信息传递给Pod中运行的容器，可以在应用部署后灵活修改，详情请参见**设置环境变量**。
  - 数据存储（可选）：在容器内挂载本地存储或云存储，不同类型的存储使用场景及挂载方式不同，详情请参见**存储**。
  - 安全设置（可选）：对容器权限进行设置，保护系统和其他容器不受其影响。请输入用户ID，容器将以当前用户权限运行。

- 镜像访问凭证：用于访问镜像仓库的凭证，默认取值为default-secret，使用default-secret可访问SWR镜像仓库的镜像。default-secret详细说明请参见**default-secret**。

服务配置（可选）

服务（Service）可为Pod提供外部访问。每个Service有一个固定IP地址，Service将访问流量转发给Pod，而且Service可以为这些Pod自动实现负载均衡。

您也可以在创建完工作负载之后再创建Service，不同类型的Service概念和使用方法请参见**服务（Service）**。

高级配置（可选）

- 升级策略：指定工作负载的升级方式及升级参数，支持滚动升级和替换升级，详情请参见**设置工作负载升级策略**。
- 标签与注解：以键值对形式为工作负载Pod添加标签或注解，填写完成后需单击“确认添加”。关于标签与注解的作用及配置说明，请参见**设置标签与注解**。
- DNS配置：为工作负载单独配置DNS策略，详情请参见**工作负载DNS配置说明**。
- 性能管理配置：使用应用性能管理（APM）服务，为JAVA程序提供更精准的问题分析与定位，详情请参见**设置性能管理配置**。
- 网络配置：
  - 是否开启指定容器网络配置：开启指定容器网络配置，工作负载使用指定的容器网络配置中的容器子网跟安全组来创建，详情请参见**使用容器网络配置为命名空间/工作负载绑定子网及安全组**。
  - 指定容器网络配置名：只支持选择关联资源类型为工作负载类型的自定义容器网络配置。

步骤4 单击右下角“创建工作负载”。

----结束

通过 kubectl 命令行创建

本节以nginx工作负载为例，说明kubectl命令创建工作负载的方法。

须知

Autopilot集群暂不支持配置节点亲和与反亲和，所以当您使用kubectl命令行创建工作负载时，为避免Pod创建失败，请不要配置affinity字段。

步骤1 请参见**通过kubectl连接集群**，使用kubectl连接集群。

步骤2 创建一个名为nginx-deployment.yaml的描述文件。其中，nginx-deployment.yaml为自定义名称，您可以随意命名。

vi nginx-deployment.yaml

描述文件内容如下。此处仅为示例，deployment的详细说明请参见**kubernetes官方文档**。

```
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    #若使用“开源镜像中心”的镜像，可直接填写镜像名称；若使用“我的镜像”中的镜像，请在SWR中获取具体镜像地址。
        imagePullPolicy: Always
        name: nginx
      imagePullSecrets:
      - name: default-secret
```

以上yaml字段解释如表2-2。

表 2-2 deployment 字段详解

字段名称	字段说明	必选/可选
apiVersion	表示API的版本号。 **说明** 请根据集群版本输入： • 1.17及以上版本的集群中无状态应用apiVersion格式为**apps/v1** • 1.15及以下版本的集群中无状态应用apiVersion格式为**extensions/v1beta1**	必选
kind	创建的对象类别。	必选
metadata	资源对象的元数据定义。	必选
name	deployment的名称。	必选
spec	用户对deployment的详细描述的主体部分都在spec中给出。	必选
replicas	实例数量。	必选
selector	定义Deployment可管理的容器实例。	必选
strategy	升级类型。当前支持两种升级方式，默认为滚动升级。 • RollingUpdate：滚动升级。 • ReplaceUpdate：替换升级。	可选
template	描述创建的容器实例详细信息。	必选
metadata	元数据。	必选
labels	metadata.labels定义容器标签。	可选

字段名称	字段说明	必选/可选
spec: containers	• image（必选）：容器镜像名称。 • imagePullPolicy（可选）：获取镜像的策略，可选值包括Always（每次都尝试重新下载镜像）、Never（仅使用本地镜像）、IfNotPresent（如果本地有该镜像，则使用本地镜像，本地不存在时下载镜像），默认为Always。 • name（必选）：容器名称。	必选
imagePullSecrets	Pull镜像时使用的secret名称。若使用私有镜像，该参数为必选。 • 需要Pull SWR容器镜像仓库的镜像时，参数值固定为default-secret。 • 当Pull第三方镜像仓库的镜像时，需设置为创建的secret名称。	可选

步骤3 创建deployment。

kubectl create -f nginx-deployment.yaml

回显如下表示已开始创建deployment。

```
deployment "nginx" created
```

步骤4 查看deployment状态。

kubectl get deployment

deployment状态显示为Running，表示deployment已创建成功。

```
NAME         READY    UP-TO-DATE   AVAILABLE   AGE
nginx        1/1      1            1           4m5s
```

参数解析：

- NAME：工作负载名称。
- READY：表示工作负载的可用状态，显示为“可用Pod个数/期望Pod个数”。
- UP-TO-DATE：指当前已经完成更新的副本数。
- AVAILABLE：可用的Pod个数。
- AGE：已经运行的时间。

步骤5 若工作负载（即deployment）需要被访问，您需要设置访问方式，具体请参见**服务（Service）**创建对应服务。

----结束

相关文档

- **实现升级实例过程中的业务不中断**
- **CCE容器中域名解析的最佳实践**

2.2.2 创建有状态负载（StatefulSet）

操作场景

在运行过程中会保存数据或状态的工作负载称为“有状态工作负载（statefulset）”。例如MySQL，它需要存储产生的新数据。

因为容器可以在不同主机间迁移，所以在宿主机上并不会保存数据，这依赖于CCE提供的高可用存储卷，将存储卷挂载在容器上，从而实现有状态工作负载的数据持久化。

约束与限制

- 当您删除或扩缩有状态负载时，为保证数据安全，系统并不会删除它所关联的存储卷。
- 当您删除一个有状态负载时，为实现有状态负载中的Pod可以有序停止，请在删除之前将副本数缩容到0。
- 您需要在创建有状态负载的同时，创建一个Headless Service，用于解决有状态负载Pod互相访问的问题，详情请参见**Headless Service**。

前提条件

- 在创建容器工作负载前，您需要存在一个可用集群。若没有可用集群，请参照**购买CCE Autopilot集群**中内容创建。
- 在创建容器工作负载前，您需要配置访问SWR和OBS服务的VPC终端节点，详情请参见**配置访问SWR和OBS服务的VPC终端节点**。
- 若工作负载需要被外网访问，请为工作负载创建负载均衡类型的服务。

  说明

  单个实例（Pod）内如果有多个容器，请确保容器使用的端口不冲突，否则部署会失败。

通过控制台创建

步骤1 登录CCE控制台。

步骤2 单击集群名称进入集群，在左侧选择“工作负载”，在右上角单击“创建工作负载”。

步骤3 配置工作负载的信息。

基本信息

- 负载类型：选择有状态工作负载StatefulSet。
- 负载名称：填写工作负载的名称。请输入1到63个字符的字符串，可以包含小写英文字母、数字和中划线（-），并以小写英文字母开头，小写英文字母或数字结尾。
- 命名空间：选择工作负载的命名空间，默认为default。您可以单击后面的“创建命名空间”，命名空间的详细介绍请参见**创建命名空间**。
- 实例数量：填写实例的数量，即工作负载Pod的数量。

容器配置

- 容器信息

  Pod中可以配置多个容器，您可以单击右侧“添加容器”为Pod配置多个容器。

- 基本信息：配置容器的基本信息。

参数	说明
容器名称	为容器命名。
更新策略	镜像更新/拉取策略。可以勾选“总是拉取镜像”，表示每次都从镜像仓库拉取镜像；如不勾选则优使用节点已有的镜像，如果没有这个镜像再从镜像仓库拉取。
镜像名称	单击后方“选择镜像”，选择容器使用的镜像。 如果需要使用第三方镜像，请参见**使用第三方镜像**。
镜像版本	选择需要部署的镜像版本。
CPU配额	CPU资源限制值，即允许容器使用的CPU最大值，防止占用过多资源。
内存配额	内存资源限制值，即允许容器使用的内存最大值。如果超过，容器会被终止。
初始化容器（可选）	选择容器是否作为初始化（Init）容器。初始化（Init）容器不支持设置健康检查。 Init容器是一种特殊容器，可以在Pod中的其他应用容器启动之前运行。每个Pod中可以包含多个容器，同时Pod中也可以有一个或多个先于应用容器启动的Init容器，当所有的Init 容器运行完成时，Pod中的应用容器才会启动并运行。详细说明请参见**Init容器**。

- 生命周期（可选）：在容器的生命周期的特定阶段配置需要执行的操作，例如启动命令、启动后处理和停止前处理，详情请参见**设置容器生命周期**。
- 健康检查（可选）：根据需求选择是否设置存活探针、就绪探针及启动探针，详情请参见**设置容器健康检查**。
- 环境变量（可选）：支持通过键值对的形式为容器运行环境设置变量，可用于把外部信息传递给Pod中运行的容器，可以在应用部署后灵活修改，详情请参见**设置环境变量**。
- 数据存储（可选）：在容器内挂载本地存储或云存储，不同类型的存储使用场景及挂载方式不同。

  说明

  - 有状态负载支持“动态挂载”存储。

    动态挂载通过**volumeClaimTemplates**字段实现，并依赖于StorageClass动态创建能力。有状态工作负载通过volumeClaimTemplates字段为每一个Pod关联了一个独有的PVC，而这个PVC又会和对应的PV绑定。因此当Pod被重新调度后，仍然能够根据该PVC名称挂载原有的数据。

  - 负载创建完成后，动态挂载的存储不支持更新。
- 安全设置（可选）：对容器权限进行设置，保护系统和其他容器不受其影响。请输入用户ID，容器将以当前用户权限运行。

- 镜像访问凭证：用于访问镜像仓库的凭证，默认取值为default-secret，使用default-secret可访问SWR镜像仓库的镜像。default-secret详细说明请参见**default-secret**。

实例间发现服务配置

Headless Service用于解决StatefulSet内Pod互相访问的问题，Headless Service给每个Pod提供固定的访问域名。具体请参见**Headless Service**。

服务配置（可选）

服务（Service）可为Pod提供外部访问。每个Service有一个固定IP地址，Service将访问流量转发给Pod，而且Service可以为这些Pod自动实现负载均衡。

您也可以在创建完工作负载之后再创建Service，不同类型的Service概念和使用方法请参见**服务（Service）**。

高级配置（可选）

- 升级策略：指定工作负载的升级方式及升级参数，支持滚动升级和替换升级，详情请参见**设置工作负载升级策略**。
- 实例管理策略（podManagementPolicy）：

  对于某些分布式系统来说，StatefulSet的顺序性保证是不必要和/或者不应该的。这些系统仅仅要求唯一性和身份标志。
  - 有序策略：默认实例管理策略，有状态负载会逐个的、按顺序的进行部署、删除、伸缩实例，只有前一个实例部署Ready或者删除完成后，有状态负载才会操作后一个实例。
  - 并行策略：支持有状态负载并行创建或者删除所有的实例，有状态负载发生变更时立刻在实例上生效。
- 标签与注解：以键值对形式为工作负载Pod添加标签或注解，填写完成后需单击“确认添加”。关于标签与注解的作用及配置说明，请参见**设置标签与注解**。
- DNS配置：为工作负载单独配置DNS策略，详情请参见**工作负载DNS配置说明**。
- 性能管理配置：使用应用性能管理（APM）服务，为JAVA程序提供更精准的问题分析与定位，详情请参见**设置性能管理配置**。
- 网络配置：
  - 是否开启指定容器网络配置：开启指定容器网络配置后，工作负载使用指定的容器网络配置中的容器子网跟安全组来创建，详情请参见**使用容器网络配置为命名空间/工作负载绑定子网及安全组**。
  - 指定容器网络配置名：开启指定容器网络配置后，显示该功能。只支持选择关联资源类型为工作负载类型的自定义容器网络配置。

步骤4 单击右下角“创建工作负载”。

----结束

通过 kubectl 命令行创建

须知

Autopilot集群暂不支持配置节点亲和与反亲和，所以当您使用kubectl命令行创建工作负载时，为避免Pod创建失败，请不要配置affinity字段。

步骤1 请参见**通过kubectl连接集群**，使用kubectl连接集群。

步骤2 创建一个名为nginx-statefulset.yaml的文件。

其中，nginx-statefulset.yaml为自定义名称，您可以随意命名。

vi nginx-statefulset.yaml

以下内容仅为示例，若需要了解statefulset的详细内容，请参考**kubernetes官方文档**。

```
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
      imagePullSecrets:
        - name: default-secret
      dnsPolicy: ClusterFirst
  serviceName: nginx-svc
  replicas: 2
  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
```

步骤3 创建工作负载以及对应headless服务。

kubectl create -f nginx-statefulset.yaml

回显如下，表示有状态工作负载（stateful）已创建成功。

```
statefulset.apps/nginx created
```


kubectl create -f nginx-headless.yaml

回显如下，表示对应headless服务已创建成功。

```
service/nginx-svc created
```

步骤4 若工作负载需要被访问（集群内访问或节点访问），您需要设置访问方式，具体请参见**服务（Service）**创建对应服务。

----结束

2.2.3 创建普通任务（Job）

操作场景

普通任务是一次性运行的短任务，部署完成后即可执行。正常退出（exit 0）后，任务即执行完成。

普通任务是用来控制批处理型任务的资源对象。批处理业务与长期伺服业务（Deployment、Statefulset）的主要区别是：

批处理业务的运行有头有尾，而长期伺服业务在用户不停止的情况下永远运行。Job管理的Pod根据用户的设置把任务成功完成就自动退出了。成功完成的标志根据不同的spec.completions策略而不同，即：

- 单Pod型任务有一个Pod成功就标志完成。
- 定数成功型任务保证有N个任务全部成功。
- 工作队列型任务根据应用确认的全局成功而标志成功。

前提条件

- 在创建容器工作负载前，您需要存在一个可用集群。若没有可用集群，请参照**购买CCE Autopilot集群**中内容创建。
- 在创建容器工作负载前，您需要配置访问SWR和OBS服务的VPC终端节点，详情请参见**配置访问SWR和OBS服务的VPC终端节点**。
- 若工作负载需要被外网访问，请为工作负载创建负载均衡类型的服务。

  说明

  单个实例（Pod）内如果有多个容器，请确保容器使用的端口不冲突 ，否则部署会失败。

通过控制台创建

步骤1 登录CCE控制台。

步骤2 单击集群名称进入集群，在左侧选择“工作负载”，在右上角单击“创建工作负载”。

步骤3 配置工作负载的信息。

基本信息

- 负载类型：选择任务Job。
- 负载名称：填写工作负载的名称。请输入1到63个字符的字符串，可以包含小写英文字母、数字和中划线（-），并以小写英文字母开头，小写英文字母或数字结尾。

- 命名空间：选择工作负载的命名空间，默认为default。您可以单击后面的“创建命名空间”，命名空间的详细介绍请参见**创建命名空间**。
- 实例数量：填写实例的数量，即工作负载Pod的数量。

容器配置

- 容器信息

  Pod中可以配置多个容器，您可以单击右侧“添加容器”为Pod配置多个容器。

  - 基本信息：配置容器的基本信息。

参数	说明
容器名称	为容器命名。
更新策略	镜像更新/拉取策略。可以勾选“总是拉取镜像”，表示每次都从镜像仓库拉取镜像；如不勾选则优使用节点已有的镜像，如果没有这个镜像再从镜像仓库拉取。
镜像名称	单击后方“选择镜像”，选择容器使用的镜像。 如果需要使用第三方镜像，请参见**使用第三方镜像**。
镜像版本	选择需要部署的镜像版本。
CPU配额	CPU资源限制值，即允许容器使用的CPU最大值，防止占用过多资源。
内存配额	内存资源限制值，即允许容器使用的内存最大值。如果超过，容器会被终止。
初始化容器（可选）	选择容器是否作为初始化（Init）容器。初始化（Init）容器不支持设置健康检查。 Init容器是一种特殊容器，可以在Pod中的其他应用容器启动之前运行。每个Pod中可以包含多个容器，同时Pod中也可以有一个或多个先于应用容器启动的Init容器，当所有的Init 容器运行完成时，Pod中的应用容器才会启动并运行。详细说明请参见**Init容器**。

  - 生命周期（可选）：在容器的生命周期的特定阶段配置需要执行的操作，例如启动命令、启动后处理和停止前处理，详情请参见**设置容器生命周期**。
  - 环境变量（可选）：支持通过键值对的形式为容器运行环境设置变量，可用于把外部信息传递给Pod中运行的容器，可以在应用部署后灵活修改，详情请参见**设置环境变量**。
  - 数据存储（可选）：在容器内挂载本地存储或云存储，不同类型的存储使用场景及挂载方式不同，详情请参见**存储**。

- 镜像访问凭证：用于访问镜像仓库的凭证，默认取值为default-secret，使用default-secret可访问SWR镜像仓库的镜像。default-secret详细说明请参见**default-secret**。

高级配置（可选）

- 标签与注解：以键值对形式为工作负载Pod添加标签或注解，填写完成后需单击“确认添加”。关于标签与注解的作用及配置说明，请参见**设置标签与注解**。
- 任务设置：

  - 并行数：任务负载执行过程中允许同时创建的最大实例数，并行数应不大于实例数。
  - 超时时间（秒）：当任务执行超出该时间时，任务将会被标识为执行失败，任务下的所有实例都会被删除。为空时表示不设置超时时间。
  - 完成模式：
    - 非索引：当执行成功的Pod数达到实例数时， Job执行成功。Job中每一个Pod都是同质的，Pod之间是独立无关。
    - 索引：系统会为每个Pod分配索引值，取值为 0 到实例数-1。每个分配了索引的Pod都执行成功，则Job执行成功。索引模式下，Job中的Pod命名遵循$(job-name)-$(index)模式。
  - 挂起任务：默认任务创建后被立即执行。选择挂起任务后，任务创建后处于挂起状态；将其关闭后，任务继续执行。
- 网络配置：
  - 是否开启指定容器网络配置：开启指定容器网络配置，工作负载使用指定的容器网络配置中的容器子网跟安全组来创建，详情请参见**使用容器网络配置为命名空间/工作负载绑定子网及安全组**。
  - 指定容器网络配置名：只支持选择关联资源类型为工作负载类型的自定义容器网络配置。

步骤4 单击右下角“创建工作负载”。

----结束

使用 kubectl 创建 Job

须知

Autopilot集群暂不支持配置节点亲和与反亲和，所以当您使用kubectl命令行创建工作负载时，为避免Pod创建失败，请不要配置affinity字段。

Job的关键配置参数如下所示：

- .spec.completions表示Job结束需要成功运行的Pod个数，默认为1。
- .spec.parallelism表示并行运行的Pod的个数，默认为1。
- .spec.backoffLimit表示失败Pod的最大重试次数，超过这个次数不会继续重试。
- .spec.activeDeadlineSeconds表示Pod运行时间，一旦达到这个时间，Job及其所有的Pod都会停止。且activeDeadlineSeconds优先级高于backoffLimit，即到达activeDeadlineSeconds的Job会忽略backoffLimit的设置。

根据.spec.completions和.spec.parallelism的设置，可以将Job划分为以下几种类型。

表 2-3 任务类型

Job类型	说明	.spec.completions	.spec.parallelism
一次性Job	创建一个Pod直至其成功结束。	1	1

Job类型	说明	.spec.completions	.spec.parallelism
固定结束次数的Job	依次创建一个Pod运行直至成功结束的Pod个数到达到.spec.completions的数值。	>1	1
固定结束次数的并行Job	依次创建多个Pod运行直至成功结束的Pod个数到达到.spec.completions的数值。	>1	>1
带工作队列的并行Job	创建一个或多个Pod，从工作队列中取走对应的任务并处理，完成后将任务从队列中删除后退出，详情请参见**使用工作队列进行精细的并行处理**。	不填写	>1或=1

以下是一个Job配置示例，保存在myjob.yaml中，其计算π到2000位并打印输出。

```
apiVersion: batch/v1
kind: Job
metadata:
  name: myjob
spec:
  completions: 50        # Job结束需要运行50个Pod，这个示例中就是打印π 50次
  parallelism: 5        # 并行5个Pod
  backoffLimit: 5        # 最多重试5次
  template:
    spec:
      containers:
      - name: pi
        image: perl
        command: ["perl",  "-Mbignum=bpi", "-wle", "print bpi(2000)"]
      restartPolicy: Never
      imagePullSecrets:
        - name: default-secret
```

说明：

- apiVersion: batch/v1 是当前job的Version。
- kind: Job：指定当前资源的类型是Job。
- restartPolicy: Never：是指当前的重启策略。对于Job，只能设置为Never或者OnFailure。对于其他controller（比如Deployment）可以设置为Always。

运行该任务，如下：

步骤1 启动这个job。

```
[root@k8s-master k8s]# kubectl apply -f myjob.yaml
job.batch/myjob created
```

步骤2 查看这个job。

kubectl get job

```
[root@k8s-master k8s]# kubectl get job
NAME    COMPLETIONS   DURATION   AGE
myjob   50/50         23s        3m45s
```

completions为 50/50 表示成功运行了这个job。

步骤3 查看pod的状态。

kubectl get pod

```
[root@k8s-master k8s]# kubectl get pod
NAME         READY   STATUS      RESTARTS   AGE
myjob-29qlw   0/1     Completed   0          4m5s
...
```

状态为Completed表示这个job已经运行完成。

步骤4 查看这个pod的日志。

kubectl logs

```
# kubectl logs myjob-29qlw
3.14159265358979323846264338327950288419716939937510582097494459230781640628620899862803482534211706798214808651328230664709384460955058223172535940812848111745028410270193852110555964462294895493038196442881097566593344612847564823378678316527120190914564856692346034861045432664821339360726024914127372458700660631558817488152092096282925409171536436789259036001133053054882046652138414695194151160943305727036575959195309218611738193261179310511854807446237996274956735188575272489122793818301194912983367336244065664308602139494639522473719070217986094370277053921717629317675238467481846766940513200056812714526356082778577134275778960917363717872146844090122495343014654958537105079227968925892354201995611212902196086403441815981362977477130996051870721134999999837297804995105973173281609631859502445945534690830264252230825334468503526193118817101000313783875288658753320838142061717766914730359825349042875546873115956286388235378759375195778185778053217122680661300192787661119590921642019893809525720106548586327886593615338182796823030195203530185296899577362259941389124972177528347913151557485724245415069595082953311686172785588907509838175463746493931925506040092770167113900984882401285836160356370766010471018194295559619894676783744944825537977472684710404753464620804668425906949129331367702898915210475216205696602405803815019351125338243003558764024749647326391419927260426992279678235478163600934172164121992458631503028618297455570674983850549458858692699569092721079750930295532116534498720275596023648066549911988183479775356636980742654252786255181841757467289097777279380008164706001614524919217321721477235014144197356854816136115735255213347574184946843852332390739414333454776241686251898356948556209921922218427255025425688767179049460165346680498862723279178608578438382796797668145410095388378636095068006422512520511739298489608412848862694560424196528502221066118630674427862203919494504712371378696095636437191728746776465757396241389086583264599581339047802759 01
```

----结束

相关操作

普通任务创建完成后，您还可执行表2-4中操作。

表 2-4 其他操作

操作	操作说明
删除普通任务	1. 选择待删除的任务，单击操作列的“更多 > 删除”。 2. 单击“是”。 任务删除后将无法恢复，请谨慎操作。

2.2.4 创建定时任务（CronJob）

操作场景

定时任务是按照指定时间周期运行的短任务。使用场景为在某个固定时间点，为所有运行中的节点做时间同步。

定时任务是基于时间的Job，就类似于Linux系统的crontab，在指定的时间周期运行指定的Job，即：

- 在给定时间点只运行一次。
- 在给定时间点周期性地运行。

CronJob的典型用法如下所示：

- 在给定的时间点调度Job运行。
- 创建周期性运行的Job，例如数据库备份、发送邮件。

前提条件

- 在创建容器工作负载前，您需要存在一个可用集群。若没有可用集群，请参照**购买CCE Autopilot集群**中内容创建。
- 在创建容器工作负载前，您需要配置访问SWR和OBS服务的VPC终端节点，详情请参见**配置访问SWR和OBS服务的VPC终端节点**。
- 若工作负载需要被外网访问，请为工作负载创建负载均衡类型的服务。

  说明

  单个实例（Pod）内如果有多个容器，请确保容器使用的端口不冲突，否则部署会失败。

通过控制台创建

步骤1 登录CCE控制台。

步骤2 单击集群名称进入集群，在左侧选择“工作负载”，在右上角单击“创建工作负载”。

步骤3 配置工作负载的信息。

基本信息

- 负载类型：选择定时任务CronJob。
- 负载名称：填写工作负载的名称。请输入1到63个字符的字符串，可以包含小写英文字母、数字和中划线（-），并以小写英文字母开头，小写英文字母或数字结尾。
- 命名空间：选择工作负载的命名空间，默认为default。您可以单击后面的“创建命名空间”，命名空间的详细介绍请参见**创建命名空间**。

容器配置

- 容器信息

  Pod中可以配置多个容器，您可以单击右侧“添加容器”为Pod配置多个容器。

  - 基本信息：配置容器的基本信息。

参数	说明
容器名称	为容器命名。
更新策略	镜像更新/拉取策略。可以勾选“总是拉取镜像”，表示每次都从镜像仓库拉取镜像；如不勾选则优使用节点已有的镜像，如果没有这个镜像再从镜像仓库拉取。

参数	说明
镜像名称	单击后方“选择镜像”，选择容器使用的镜像。 如果需要使用第三方镜像，请参见**使用第三方镜像**。
镜像版本	选择需要部署的镜像版本。
CPU配额	CPU资源限制值，即允许容器使用的CPU最大值，防止占用过多资源。
内存配额	内存资源限制值，即允许容器使用的内存最大值。如果超过，容器会被终止。
初始化容器（可选）	选择容器是否作为初始化（Init）容器。初始化（Init）容器不支持设置健康检查。 Init容器是一种特殊容器，可以在Pod中的其他应用容器启动之前运行。每个Pod中可以包含多个容器，同时Pod中也可以有一个或多个先于应用容器启动的Init容器，当所有的Init 容器运行完成时，Pod中的应用容器才会启动并运行。详细说明请参见**Init容器**。

  - 生命周期（可选）：在容器的生命周期的特定阶段配置需要执行的操作，例如启动命令、启动后处理和停止前处理，详情请参见**设置容器生命周期**。
  - 环境变量（可选）：支持通过键值对的形式为容器运行环境设置变量，可用于把外部信息传递给Pod中运行的容器，可以在应用部署后灵活修改，详情请参见**设置环境变量**。
- 镜像访问凭证：用于访问镜像仓库的凭证，默认取值为default-secret，使用default-secret可访问SWR镜像仓库的镜像。default-secret详细说明请参见**default-secret**。

定时规则

- 并发策略：支持如下三种模式。
  - Forbid：在前一个任务未完成时，不创建新任务。
  - Allow：定时任务不断新建Job，会抢占集群资源。
  - Replace：已到新任务创建时间点，但前一个任务还未完成，新的任务会取代前一个任务。
- 定时规则：指定新建定时任务在何时执行，YAML中的定时规则通过CRON表达式实现。
  - 以固定周期执行定时任务，支持的周期单位为分钟、小时、日、月。例如，每30分钟执行一次任务，对应的CRON表达式为“*/30 * * * *”，执行时间将从单位范围内的0值开始计算，如**00:00:00**、**00:30:00**、**01:00:00**、**...**。
  - 以固定时间（按月）执行定时任务。例如，在每个月1日的0时0分执行任务，对应的CRON表达式为“0 0 1 */1 *”，执行时间为******-01-01 00:00:00**、******-02-01 00:00:00**、**...**。
  - 以固定时间（按周）执行定时任务。例如，在每周一的0时0分执行任务，对应的CRON表达式为“0 0 * * 1”，执行时间为******-**-01 周一 00:00:00**、******-**-08 周一 00:00:00**、**...**。
  - 自定义CRON表达式：关于CRON表达式的用法，可参考**CRON**。

说明

- 以固定时间（按月）执行定时任务时，在某月的天数不存在的情况下，任务将不会在该月执行。例如设置天数为30，而2月份没有30号，任务将跳过该月份，在3月30号继续执行。
- 由于CRON表达式的定义，这里的固定周期并非严格意义的周期。将从0开始按周期对其时间单位范围（例如单位为分钟时，则范围为0~59）进行划分，无法整除时最后一个周期会被重置。因此仅在周期能够平均划分其时间单位范围时，才能表示准确的周期。

  举个例子，周期单位为小时，因为“/2、/3、/4、/6、/8和/12”可将24小时整除，所以可以表示准确的周期；而使用其他周期时，在新的一天开始时，最后一个周期将会被重置。比如CRON式为“* */12 * * *”时为准确的周期，每天的执行时间为**00:00:00**和**12:00:00**；而CRON式为“* */13 * * *”时，每天的执行时间为**00:00:00**和**13:00:00**，在第二天0时，虽然没到13个小时的周期但还是会被刷新。

- 任务记录：可以设置保留执行成功或执行失败的任务个数，设置为0表示不保留。

高级配置（可选）

- 标签与注解：以键值对形式为工作负载Pod添加标签或注解，填写完成后需单击“确认添加”。关于标签与注解的作用及配置说明，请参见**设置标签与注解**。
- 网络配置：
  - 是否开启指定容器网络配置：开启指定容器网络配置，工作负载使用指定的容器网络配置中的容器子网跟安全组来创建，详情请参见**使用容器网络配置为命名空间/工作负载绑定子网及安全组**。
  - 指定容器网络配置名：只支持选择关联资源类型为工作负载类型的自定义容器网络配置。

步骤4 单击右下角“创建工作负载”。

----结束

使用 kubectl 创建 CronJob

须知

Autopilot集群暂不支持配置节点亲和与反亲和，所以当您使用kubectl命令行创建工作负载时，为避免Pod创建失败，请不要配置affinity字段。

CronJob的配置参数如下所示：

- .spec.schedule指定任务运行时间与周期，参数格式请参见**Cron**，例如“0 * * * *”或“@hourly”。
- .spec.jobTemplate指定需要运行的任务，格式与**使用kubectl创建Job**相同。
- .spec.startingDeadlineSeconds指定任务开始的截止期限。
- .spec.concurrencyPolicy指定任务的并发策略，支持Allow、Forbid和Replace三个选项。
  - Allow（默认）：允许并发运行Job。
  - Forbid：禁止并发运行，如果前一个还没有完成，则直接跳过下一个。
  - Replace：取消当前正在运行的Job，用一个新的来替换。

下面是一个CronJob的示例，保存在cronjob.yaml文件中。

说明

在v1.21及以上集群中，CronJob的apiVersion为**batch/v1**。

在v1.21以下集群中，CronJob的apiVersion为**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
```

运行该任务，如下：

步骤1 创建CronJob。

kubectl create -f cronjob.yaml

命令行终端显示如下信息：

```
cronjob.batch/hello created
```

步骤2 执行如下命令，查看执行情况。

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
```


须知

删除CronJob时，对应的普通任务及相关的Pod都会被删除。

----结束

相关操作

定时任务创建完成后，您还可执行表2-5中操作。

表 2-5 其他操作

操作	操作说明
编辑YAML	单击定时任务名称后的“更多 > 编辑YAML”，可修改当前任务对应的YAML文件。
停止定时任务	1. 选择待停止的任务，单击操作列的“停止”。 2. 单击“是”。
删除定时任务	1. 选择待删除的任务，单击操作列的“更多 > 删除”。 2. 单击“是”。 任务删除后将无法恢复，请谨慎操作。

2.3 配置工作负载

2.3.1 设置镜像拉取策略

创建工作负载会从镜像仓库拉取容器镜像到节点上，当前Pod重启、升级时也会拉取镜像。

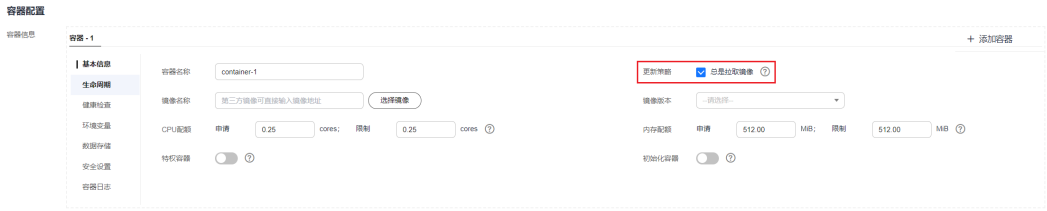
默认情况下容器镜像拉取策略imagePullPolicy是**IfNotPresent**，表示如果节点上有这个镜像就直接使用节点已有镜像，如果没有这个镜像就会从镜像仓库拉取。

容器镜像拉取策略还可以设置为**Always**，表示无论节点上是否有这个镜像，都会从镜像仓库拉取，并覆盖节点上的镜像。

```
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
```

在CCE控制台也可以设置镜像拉取策略，在创建工作负载时设置“更新策略”：勾选表示总是拉取镜像（Always），不勾选则表示按需拉取镜像（IfNotPresent）。

图 2-4 设置更新策略



须知

建议您在制作镜像时，每次制作一个新的镜像都使用一个新的Tag，如果不更新Tag只更新镜像，当拉取策略选择为IfNotPresent时，CCE会认为当前节点已经存在这个Tag的镜像，不会重新拉取。

2.3.2 使用第三方镜像

操作场景

CCE支持拉取第三方镜像仓库的镜像来创建工作负载。

通常第三方镜像仓库必须经过认证（账号密码）才能访问，而CCE中容器拉取镜像是使用密钥认证方式，这就要求在拉取镜像前先创建镜像仓库的密钥。

前提条件

使用第三方镜像时，请确保Autopilot可以正常访问私有仓库所在的网络环境，可供选择的方案如下：

- 内网访问：私有仓库位于集群所在VPC。
- 云专线或VPN访问：将私有仓库网络环境通过**云专线**或**VPN**与集群所在VPC打通。

通过界面操作

步骤1 创建第三方镜像仓库的密钥。

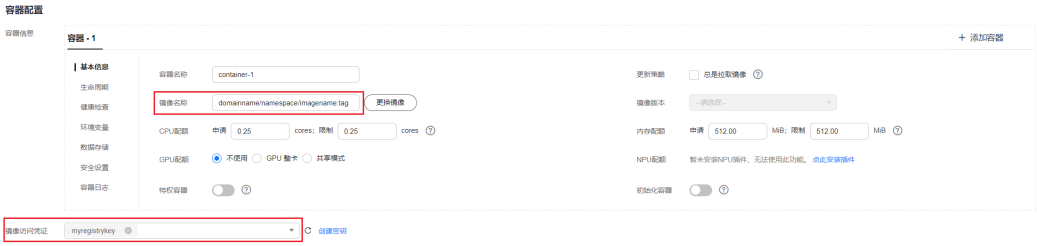
单击集群名称进入集群，在左侧导航栏选择“配置与密钥”，在右侧选择“密钥”页签，单击右上角“创建密钥”，密钥类型必须选择为kubernetes.io/dockerconfigjson。详细操作请参见**创建密钥**。

此处的“用户名”和“密码”请填写第三方镜像仓库的账号密码。

图 2-5 添加密钥

步骤2 创建工作负载时，可以在“镜像名称”中直接填写私有镜像地址，填写的格式为domainname/namespace/imagename:tag，并选择**步骤1**中创建的密钥。

图 2-6 填写私有镜像地址



步骤3 填写其他参数后，单击“创建工作负载”。

----结束

使用 kubectl 创建第三方镜像仓库的密钥

步骤1 请参见**通过kubectl连接集群**，使用kubectl连接集群。

步骤2 通过kubectl创建认证密钥 ，该密钥类型为kubernetes.io/dockerconfigjson类型。

```
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
```

其中，*myregistrykey*为密钥名称，*default*为密钥所在的命名空间，其余参数如下所示。

- DOCKER_REGISTRY_SERVER：第三方镜像仓库的地址，如“www.3rdregistry.com”或“10.10.10.10:443”。
- DOCKER_USER：第三方镜像仓库的账号。
- DOCKER_PASSWORD：第三方镜像仓库的密码。

- DOCKER_EMAIL：第三方镜像仓库的邮箱。

步骤3 创建工作负载时使用第三方镜像，具体步骤请参见如下。

kubernetes.io/dockerconfigjson类型的密钥作为私有镜像获取的认证方式，以Pod为例，创建的myregistrykey作为镜像的认证方式。

```
apiVersion: v1
kind: Pod
metadata:
  name: foo
  namespace: default
spec:
  containers:
    - name: foo
      image: www.3rdregistry.com/janedoe/awesomeapp:v1
  imagePullSecrets:
    - name: myregistrykey              #使用上面创建的密钥
```

----结束

2.3.3 设置容器生命周期

操作场景

CCE提供了回调函数，在容器的生命周期的特定阶段执行调用，比如容器在停止前希望执行某项操作，就可以注册相应的钩子函数。

目前提供的生命周期回调函数如下所示：

- **启动命令：**容器将会以该启动命令启动，请参见**启动命令**。
- **启动后处理：**容器启动后触发，请参见**启动后处理**。
- **停止前处理：**容器停止前触发。设置停止前处理，确保升级或实例删除时可提前将实例中运行的业务排水。详细请参见**停止前处理**。

启动命令

在默认情况下，镜像启动时会运行默认命令，如果想运行特定命令或重写镜像默认值，需要进行相应设置。

Docker的镜像拥有存储镜像信息的相关元数据，如果不设置生命周期命令和参数，容器运行时将运行镜像制作时提供的默认的命令和参数，Docker将这两个字段定义为ENTRYPOINT和CMD。

如果在创建工作负载时填写了容器的运行命令和参数，将会覆盖镜像构建时的默认命令ENTRYPOINT、CMD，规则如下：

表 2-6 容器如何执行命令和参数

镜像ENTRYPOINT	镜像CMD	容器运行命令	容器运行参数	最终执行
[touch]	[/root/test]	未设置	未设置	[touch /root/test]
[touch]	[/root/test]	[mkdir]	未设置	[mkdir]

镜像ENTRYPOINT	镜像CMD	容器运行命令	容器运行参数	最终执行
[touch]	[/root/test]	未设置	[/opt/test]	[touch /opt/test]
[touch]	[/root/test]	[mkdir]	[/opt/test]	[mkdir /opt/test]

步骤1 登录CCE控制台，在创建工作负载时，配置容器信息，选择“生命周期”。

步骤2 在“启动命令”页签，输入运行命令和运行参数。

表 2-7 容器启动命令

命令方式	操作步骤
运行命令	输入可执行的命令，例如“/run/server”。 若运行命令有多个，需分行书写。 **说明** 多命令时，运行命令建议用/bin/sh或其他的shell，其他全部命令作为参数来传入。
运行参数	输入控制容器运行命令参数，例如--port=8080。 若参数有多个，多个参数以换行分隔。

----结束

启动后处理

步骤1 登录CCE控制台，在创建工作负载时，配置容器信息，选择“生命周期”。

步骤2 在“启动后处理”页签，设置启动后处理的参数。

表 2-8 启动后处理-参数说明

参数	说明

命令行脚本方式	在容器中执行指定的命令，配置为需要执行的命令。命令的格式为Command Args[1] Args[2]…（Command为系统命令或者用户自定义可执行程序，如果未指定路径则在默认路径下寻找可执行程序），如果需要执行多条命令，建议采用将命令写入脚本执行的方式。**不支持后台执行和异步执行的命令。**
	如需要执行的命令如下：
	`exec:`
	`  command:`
	`  - /install.sh`
	`  - install_agent`
	请在执行脚本中填写: /install install_agent。这条命令表示容器创建成功后将执行install.sh。

----结束

停止前处理

步骤1 登录CCE控制台，在创建工作负载时，配置容器信息，选择“生命周期”。

步骤2 在“停止前处理”页签，设置停止前处理的命令。

表 2-9 停止前处理

参数	说明
命令行脚本方式	在容器中执行指定的命令，配置为需要执行的命令。命令的格式为Command Args[1] Args[2]…（ Command为系统命令或者用户自定义可执行程序，如果未指定路径则在默认路径下寻找可执行程序），如果需要执行多条命令，建议采用将命令写入脚本执行的方式。 如需要执行的命令如下： exec: command: - /uninstall.sh - uninstall_agent 请在执行脚本中填写: /uninstall uninstall_agent。这条命令表示容器结束前将执行uninstall.sh。

----结束

YAML 样例

本节以nginx为例，说明kubectl命令设置容器生命周期的方法。

在以下配置文件中，您可以看到postStart命令在容器目录/bin/bash下写了个**install.sh**命令。 preStop执行**uninstall.sh**命令。

```
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                    #启动命令
        imagePullPolicy: Always
        lifecycle:
          postStart:
            exec:
              command:
```



```
      - /bin/bash
      - install.sh              #启动后命令
    preStop:
      exec:
        command:
        - /bin/bash
        - uninstall.sh            #停止前命令
  name: nginx
imagePullSecrets:
- name: default-secret
```

2.3.4 设置容器健康检查

操作场景

健康检查是指容器运行过程中，根据用户需要，定时检查容器健康状况。若不配置健康检查，如果容器内应用程序异常，Pod将无法感知，也不会自动重启去恢复。最终导致虽然Pod状态显示正常，但Pod中的应用程序异常的情况。

Kubernetes提供了三种健康检查的探针：

- **存活探针：**livenessProbe，用于检测容器是否正常，类似于执行ps命令检查进程是否存在。如果容器的存活检查失败，集群会对该容器执行重启操作；若容器的存活检查成功则不执行任何操作。
- **就绪探针：**readinessProbe，用于检查用户业务是否就绪，如果未就绪，则不转发流量到当前实例。一些程序的启动时间可能很长，比如要加载磁盘数据或者要依赖外部的某个模块启动完成才能提供服务。这时候程序进程在，但是并不能对外提供服务。这种场景下该检查方式就非常有用。如果容器的就绪检查失败，集群会屏蔽请求访问该容器；若检查成功，则会开放对该容器的访问。
- **启动探针**：startupProbe，用于探测应用程序容器什么时候启动了。 如果配置了这类探测器，就可以控制容器在启动成功后再进行存活性和就绪检查， 确保这些存活、就绪探针不会影响应用程序的启动。 这可以用于对启动慢的容器进行存活性检测，避免它们在启动运行之前就被终止。

检查方式

- **HTTP请求检查**

  HTTP请求方式针对的是提供HTTP/HTTPS服务的容器，集群周期性地对该容器发起HTTP/HTTPS GET请求，如果HTTP/HTTPS response返回码属于200~399范围，则证明探测成功，否则探测失败。使用HTTP请求探测必须指定容器监听的端口和HTTP/HTTPS的请求路径。

  例如：提供HTTP服务的容器，HTTP检查路径为：/health-check；端口为：80；主机地址可不填，默认为容器实例IP，此处以172.16.0.186为例。那么集群会周期性地对容器发起如下请求：GET http://172.16.0.186:80/health-check。您也可以为HTTP请求添加一个或多个请求头部，例如设置请求头名称为Custom-Header，对应的值为example。

图 2-7 HTTP 请求检查

- **TCP 端口检查**

  对于提供TCP通信服务的容器，集群周期性地对该容器建立TCP连接，如果连接成功，则证明探测成功，否则探测失败。选择TCP端口探测方式，必须指定容器监听的端口。

  例如：有一个nginx容器，它的服务端口是80，对该容器配置了TCP端口探测，指定探测端口为80，那么集群会周期性地对该容器的80端口发起TCP连接，如果连接成功则证明检查成功，否则检查失败。

图 2-8 TCP 端口检查

- **执行命令检查**

  命令检查是一种强大的检查方式，该方式要求用户指定一个容器内的可执行命令，集群会周期性地在容器内执行该命令，如果命令的返回结果是0则检查成功，否则检查失败。

  对于上面提到的TCP端口检查和HTTP请求检查，都可以通过执行命令检查的方式来替代：

  - 对于TCP端口探测，可以使用程序对容器的端口尝试connect，如果connect成功，脚本返回0，否则返回-1。
  - 对于HTTP请求探测，可以使用脚本命令来对容器尝试使用wget命令进行探测。

    wget http://127.0.0.1:80/health-check

    并检查response的返回码，如果返回码在200~399 的范围，脚本返回0，否则返回-1。如下图：

图 2-9 执行命令检查

须知

- 必须把要执行的程序放在容器的镜像里面，否则会因找不到程序而执行失败。
- 如果执行的命令是一个shell脚本，由于集群在执行容器里的程序时，不在终端环境下，因此不能直接指定脚本为执行命令，需要加上脚本解析器。比如脚本是**/data/scripts/health_check.sh**，那么使用执行命令检查时，指定的程序应该是**sh /data/scripts/health_check.sh**。究其原因是集群在执行容器里的程序时，不在终端环境下。

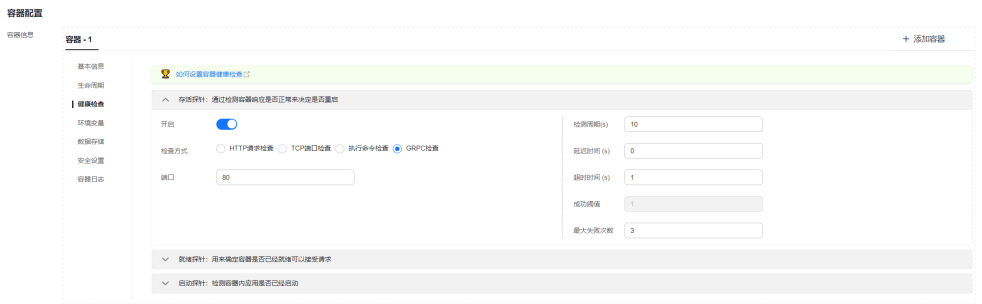
- **GRPC检查**

  GRPC检查可以为GRPC应用程序配置启动、活动和就绪探针，而无需暴露任何HTTP端点，也不需要可执行文件。Kubernetes可以通过GRPC连接到工作负载并查询其状态。

须知

- 使用GRPC检查时，您的应用需支持**GRPC健康检查协议**。
- 与HTTP和TCP探针类似，如果配置错误，都会被认作是探测失败，例如错误的端口、应用未实现健康检查协议等。

图 2-10 GRPC 检查



公共参数说明

表 2-10 公共参数说明

参数	参数说明
检测周期（periodSeconds）	探针检测周期，单位为秒。 例如，设置为30，表示每30秒检测一次。
延迟时间（initialDelaySeconds）	延迟检查时间，单位为秒，此设置与业务程序正常启动时间相关。 例如，设置为30，表明容器启动后30秒才开始健康检查，该时间是预留给业务程序启动的时间。
超时时间（timeoutSeconds）	超时时间，单位为秒。 例如，设置为10，表明执行健康检查的超时等待时间为10秒，如果超过这个时间，本次健康检查就被视为失败。若设置为0或不设置，默认超时等待时间为1秒。
成功阈值（successThreshold）	探测失败后，将状态转变为成功所需要的最小连续成功次数。例如，设置为1时，表明健康检查失败后，健康检查需要连续成功1次，才认为工作负载状态正常。 默认值是 1，最小值是 1。 存活和启动探测的这个值必须是 1。
最大失败次数（failureThreshold）	当探测失败时重试的次数。 存活探测情况下的放弃就意味着重新启动容器。就绪探测情况下的放弃Pod会被打上未就绪的标签。 默认值是 3。最小值是 1。

YAML 示例

```
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
```

2.3.5 设置环境变量

操作场景

环境变量是指容器运行环境中设定的一个变量，环境变量可以在工作负载部署后修改，为工作负载提供极大的灵活性。

CCE中设置的环境变量与Dockerfile中的“ENV”效果相同。

须知

容器启动后，容器中的内容不应修改。如果修改配置项（例如将容器应用的密码、证书、环境变量配置到容器中），当容器重启（例如节点异常重新调度Pod）后，会导致配置丢失，业务异常。

配置信息应通过入参等方式导入容器中，以免重启后配置丢失。

环境变量支持如下几种方式设置。

- **自定义**：手动填写环境变量名称及对应的参数值。
- **配置项导入**：将配置项中所有键值都导入为环境变量。
- **配置项键值导入**：将配置项中某个键的值导入作为某个环境变量的值。例如图2-11中将configmap-example这个配置项中configmap_key的值configmap_value导入为环境变量key1的值，导入后容器中将会存在一个名为key1的环境变量，其值为configmap_value。
- **密钥导入**：将密钥中所有键值都导入为环境变量。
- **密钥键值导入**：将密钥中某个键的值导入作为某个环境变量的值。例如图2-11中将secret-example这个配置项中secret_key的值secret_value导入为环境变量key2的值，导入后容器中将会存在一个名为key2的环境变量，其值为secret_value。
- **变量/变量引用**：用Pod定义的字段作为环境变量的值。例如图2-11中将此Pod的名称导入为环境变量key3的值，导入后容器中将会存在一个名为key3的环境变量，其值为该Pod的名称。
- **资源引用**：用容器定义的资源申请值或限制值作为环境变量的值。例如图2-11中将容器container-1的CPU限制值导入为环境变量key4的值，导入后容器中将会存在一个名为key4的环境变量，其值为容器container-1的CPU限制值。

添加环境变量

步骤1 登录CCE控制台。

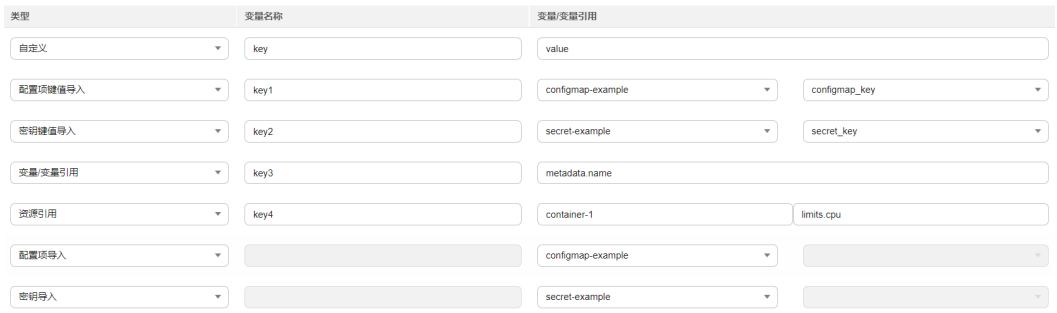
步骤2 单击集群名称进入集群，在左侧选择“工作负载”，在右上角单击“创建工作负载”。

步骤3 在创建工作负载时，在“容器配置”中修改容器信息，选择“环境变量”页签。

步骤4 设置环境变量。

- 单击“新增变量”，逐条增加环境变量，依次“配置类型”、“变量名称”和“变量/变量引用”。
- 单击“批量编辑自定义变量”，在编辑页面，按行输入自定义变量，格式为“变量名称=变量/变量引用”。

图 2-11 设置环境变量



----结束

YAML 样例

```
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                    # 自定义
              value: value
            - name: key1                   # 配置项键值导入
              valueFrom:
                configMapKeyRef:
                  name: configmap-example
                  key: configmap_key
            - name: key2                   # 密钥键值导入
              valueFrom:
                secretKeyRef:
                  name: secret-example
```



```
            key: secret_key
        - name: key3                    # 变量引用，用Pod定义的字段作为环境变量的值
          valueFrom:
            fieldRef:
              apiVersion: v1
              fieldPath: metadata.name
        - name: key4                    # 资源引用，用Container定义的字段作为环境变量的值
          valueFrom:
            resourceFieldRef:
              containerName: container1
              resource: limits.cpu
              divisor: 1
      envFrom:
        - configMapRef:                 # 配置项导入
            name: configmap-example
        - secretRef:                    # 密钥导入
            name: secret-example
    imagePullSecrets:
      - name: default-secret
```

环境变量查看

如果configmap-example和secret-example的内容如下。

```
$ 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为secret_value的base64编码
kind: Secret
...
```

则进入Pod中查看的环境变量结果如下。

```
$ 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                           # 自定义环境变量
key1=configmap_value                  # 配置项键值导入
key2=secret_value                   # 密钥键值导入
key3=env-example-695b759569-lx9jp     # Pod的metadata.name
key4=1                              # container1这个容器的limits.cpu，单位为Core，向上取整
configmap_key=configmap_value         # 配置项导入，原配置项中的键值直接会导入结果
secret_key=secret_value               # 密钥导入，原密钥中的键值直接会导入结果
```

2.3.6 设置性能管理配置

操作场景

应用性能管理服务（APM）当前支持给JAVA类工作负载提供调用链、拓扑等监控能力。您可为JAVA类工作负载安装APM探针，以提供更精准的问题分析与定位，协助您高效解决应用难题。

工作负载创建时和创建后，均可以对JAVA类工作负载监控进行设置。

前提条件

- 若您还未开通APM服务，请前往APM控制台，并参照界面提示进行开通。
- 若您还未创建APM终端节点，请根据性能管理配置页面的指引自动创建。

注意事项

- 如果删除APM终端节点，会导致数据上传到APM失败。此时如果重新创建APM终端节点，系统没有自动将指标上传到APM服务，您可以尝试重启应用。
- 删除使用APM的工作负载，不会删除APM终端节点，如需删除，请前往网络控制台删除。

操作步骤

步骤1 登录CCE控制台。

步骤2 单击集群名称进入集群，在左侧选择“工作负载”，在右上角单击“创建工作负载”。

步骤3 在创建工作负载时，在“高级设置”中找到“性能管理配置”。探针默认状态为“不启用”，您可选择“APM2.0探针”。开启后将会通过应用性能管理服务针对JAVA程序提供更精准的问题分析与定位。

说明

1. 启用APM探针后会自动新增一个名为init-javaagent（APM2.0探针）的初始化容器用来初始化探针，并分配0.25Core CPU和250MiB内存供初始化容器使用。
2. 启用APM探针后会给所有业务容器自动添加环境变量：PAAS_MONITORING_GROUP、JAVA_TOOL_OPTIONS、PAAS_CLUSTER_ID、APM_ACCESS_ADDRESS。
3. 启用APM探针后会给所有业务容器自动挂载一个名为paas-apm2（APM2.0 探针）的本地存储卷。

步骤4 填写探针相关参数。

- APM终端节点：Autopilot集群内访问APM需要创建APM终端节点，用于连接VPC网络和APM服务。关于终端节点价格详情请参见**价格计算器**。
- 探针版本：选择探针的版本。
- “探针升级策略”，默认为“重启自动升级”。
  - 重启自动升级：每次都尝试重新下载镜像。
  - 重启手动升级：如果本地有该镜像，则使用本地镜像，本地不存在时下载镜像。
- APM环境：输入APM环境名称，该参数为选填。
- APM应用：选择一个已有的APM应用。
- 子应用：输入APM子应用，该参数为选填。
- 接入密钥：将会自动获取APM服务的密钥信息，可前往APM控制台查看密钥详情。

关于APM2.0参数概念的详细说明，请参见**APM参数说明**。

步骤5 应用启动后，等待约3分钟，应用数据就会呈现在APM界面中，此时登录APM，您可以在APM上通过拓扑、调用链等进行应用性能优化，详细操作请参考**应用拓扑**。

----结束

修改性能管理配置

步骤1 登录CCE控制台。

步骤2 单击集群名称进入集群，在左侧选择“工作负载”，单击工作负载名称。

步骤3 在“性能管理配置”页签中，单击右下角“编辑”修改性能管理配置参数。

参数说明详情请参见步骤4。

----结束

2.3.7 设置工作负载升级策略

在实际应用中，升级是一个常见的场景，Deployment、StatefulSet和DaemonSet都能够很方便的支撑应用升级。

设置不同的升级策略，有如下两种。

- RollingUpdate：滚动升级，即逐步创建新Pod再删除旧Pod，为默认策略。
- Recreate：替换升级，即先把当前Pod删掉再重新创建Pod。

图 2-12 工作负载升级策略

升级参数说明

参数	说明	限制
最大浪涌（maxSurge）	与spec.replicas相比，可以有多少个Pod存在，默认值是25%。 比如spec.replicas为 4，那升级过程中就不能超过5个Pod存在，即按1个的步长升级，实际升级过程中会换算成数字，且换算会向上取整。这个值也可以直接设置成数字。	仅Deployment支持配置。
最大无效实例数（maxUnavailable）	与spec.replicas相比，可以有多少个Pod失效，也就是删除的比例，默认值是25%。 比如spec.replicas为4，那升级过程中就至少有3个Pod存在，即删除Pod的步长是1。同样这个值也可以设置成数字。	仅Deployment支持配置。
实例可用最短时间（minReadySeconds）	指定新创建的 Pod 在没有任意容器崩溃情况下的最小就绪时间， 只有超出这个时间 Pod 才被视为可用。默认值为 0（Pod 在准备就绪后立即将被视为可用）。	-

参数	说明	限制
最大保留版本数（revisionHistoryLimit）	用来设定出于回滚目的所要保留的旧 ReplicaSet 数量。 这些旧 ReplicaSet 会消耗 etcd 中的资源，并占用 kubectl get rs 的输出。 每个 Deployment 修订版本的配置都存储在其 ReplicaSets 中；因此，一旦删除了旧的 ReplicaSet， 将失去回滚到 Deployment 的对应修订版本的能力。 默认情况下，系统保留 10 个旧 ReplicaSet，但其理想值取决于新 Deployment 的频率和稳定性。	-
升级最大时长（progressDeadlineSeconds）	指定系统在报告 Deployment 进展失败 之前等待 Deployment 取得进展的秒数。 这类报告会在资源状态中体现为 Type=Progressing、Status=False、Reason=ProgressDeadlineExceeded。 Deployment 控制器将持续重试 Deployment。 将来，一旦实现了自动回滚，Deployment 控制器将在探测到这样的条件时立即回滚 Deployment。 如果指定，则此字段值需要大于 .spec.minReadySeconds 取值。	-
缩容时间窗（terminationGracePeriodSeconds）	优雅删除时间，默认为30秒，删除Pod时发送SIGTERM终止信号，然后等待容器中的应用程序终止执行，如果在terminationGracePeriodSeconds时间内未能终止，则发送SIGKILL的系统信号强行终止。	-

升级示例

Deployment的升级可以是声明式的，也就是说只需要修改Deployment的YAML定义即可，比如使用kubectl edit命令将上面Deployment中的镜像修改为nginx:alpine。修改完成后再查询ReplicaSet和Pod，发现创建了一个新的ReplicaSet，Pod也重新创建了。

```
$ 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
```

Deployment可以通过maxSurge和maxUnavailable两个参数控制升级过程中同时重新创建Pod的比例，这在很多时候是非常有用，配置如下所示。

```
spec:
  strategy:
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 0
    type: RollingUpdate
```

在前面的例子中，由于spec.replicas是2，如果maxSurge和maxUnavailable都为默认值25%，那实际升级过程中，maxSurge允许最多3个Pod存在（向上取整，2*1.25=2.5，取整为3），而maxUnavailable则不允许有Pod Unavailable（向上取整，2*0.75=1.5，取整为2），也就是说在升级过程中，一直会有2个Pod处于运行状态，每次新建一个Pod，等这个Pod创建成功后再删掉一个旧Pod，直至Pod全部为新Pod。

回滚

回滚也称为回退，即当发现升级出现问题时，让应用回到老的版本。Deployment可以非常方便的回滚到老版本。

例如上面升级的新版镜像有问题，可以执行kubectl rollout undo命令进行回滚。

```
$ kubectl rollout undo deployment nginx
deployment.apps/nginx rolled back
```

Deployment之所以能如此容易的做到回滚，是因为Deployment是通过ReplicaSet控制Pod的，升级后之前ReplicaSet都一直存在，Deployment回滚做的就是使用之前的ReplicaSet再次把Pod创建出来。Deployment中保存ReplicaSet的数量可以使用revisionHistoryLimit参数限制，默认值为10。

2.3.8 设置标签与注解

Pod 注解

Autopilot集群在创建Pod过程中提供了多种高级功能，您可以通过在控制台或YAML中添加注解（Annotation）来实现，具体可用的Annotation请参见**表2-11**。

表 2-11 Pod Annotation

功能及相关文档	参数	示例值	说明
为Pod配置QoS	kubernetes.io/ingress-bandwidth	100M	表示Pod的入口带宽大小。 用于控制进入Pod的数据传输速率，确保Pod具有处理外部请求的能力。
	kubernetes.io/egress-bandwidth	100M	表示Pod的出口带宽大小。 用于控制Pod向外发送数据的速率，影响Pod与外部服务或用户之间的通信效率。
设置可用区亲和性	node.cce.io/node-az-list	cn-east-3a,cn-east-3b	表示Pod亲和的可用区列表。 可以通过设置工作负载注解实现可用区亲和，将Pod调度到指定的可用区。

功能及相关文档	参数	示例值	说明
增加Pod的临时存储容量	resource.cce.io/extra-ephemeral-storage-in-GiB	14	表示额外增加的Pod临时存储容量，单位GiB，取值范围：0-994。 每个Pod的实际临时存储容量=30GiB（默认免费提供的临时存储容量）+额外增加的临时存储容量。 **说明** 1.28.6-r0和1.27.8-r0及以上版本的集群支持该功能，其他版本的集群需要升级才能实现。

您可通过控制台或YAML文件添加**Annotation**，具体步骤如下。

控制台添加Annotation

在控制台创建工作负载时，您可以通过“高级配置 > 标签与注解”，添加“Pod注解”，从而启用Pod的高级功能。

例如，为Pod增加14GiB的临时存储空间，可以将“Pod注解”设置为“resource.cce.io/extra-ephemeral-storage-in-GiB=14”，并单击“确认添加”。

图 2-13 为 Pod 增加临时存储空间

YAML中添加Annotation

通过YAML创建工作负载时您可以通过“annotations”参数，启用Pod的高级功能。

例如，创建Nginx工作负载时，您可以通过“annotations”为Pod增加14GiB的临时存储空间。

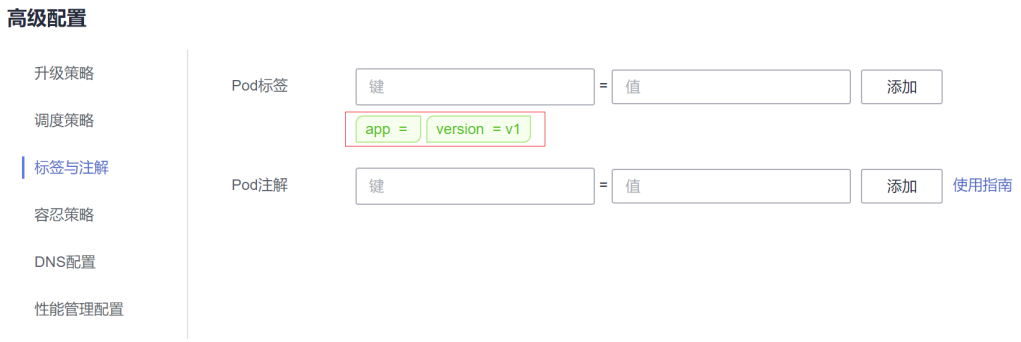
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx     # 工作负载名称
spec:
  replicas: 1     # 实例数量
  selector:
    matchLabels:  # 选择器，用于选择带有特定标签的资源
      app: nginx
  template:
    metadata:
      labels:     # 标签
        app: nginx
      annotations:
        resource.cce.io/extra-ephemeral-storage-in-GiB: '14'
    spec:
```

```
  containers:
  - image: nginx:latest   # 镜像名称：镜像版本
    name: nginx
  imagePullSecrets:
  - name: default-secret
```

Pod 标签

您可以通过Pod标签为Pod组织、选择和管理相关的资源，提高资源应用的灵活性和可维护性。

在控制台创建工作负载时，会默认为Pod添加如下标签，其中app的值为工作负载名称。



YAML示例如下：

```
...
spec:
  selector:
    matchLabels:
      app: nginx
      version: v1
  template:
    metadata:
      labels:
        app: nginx
        version: v1
    spec:
      ...
```

2.3.9 设置可用区亲和性

操作场景

可用区是指在同一个地理区域内，由于硬件、网络等因素的限制，将数据中心划分为多个独立的区域。在同一可用区内的节点之间可以通过高速网络进行快速通信，而不同可用区之间的通信则需要跨越物理距离，可能会有一定的延迟和风险。

将Pod调度到不同的可用区域中，可以提高应用程序的可用性和容错性。

设置可用区亲和性

在Autopilot集群中，您可以通过设置工作负载注解实现可用区亲和，将Pod调度到指定的可用区。

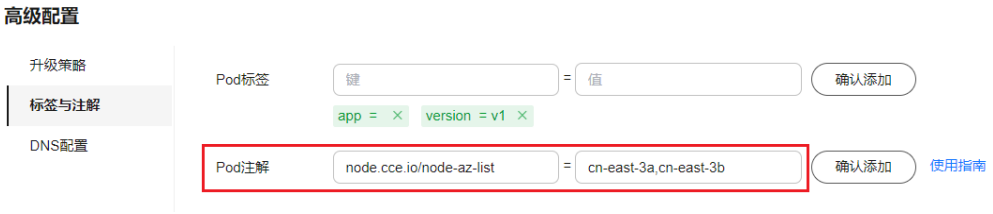
步骤1 登录CCE控制台。

步骤2 单击集群名称进入集群，在左侧选择“工作负载”，在右上角单击“创建工作负载”。

步骤3 在“高级配置”中，选择“标签与注解”，并填写以下注解。

- 键：node.cce.io/node-az-list
- 值：可用区名称，多个可用区间使用英文逗号隔开。
  不同区域的可用区名称请参见**地区和终端节点**。

图 2-14 设置可用区亲和性



步骤4 填写其他工作负载参数后，单击“创建工作负载”。

----结束

2.4 登录容器实例

操作场景

如果在使用容器的过程中遇到非预期的问题，您可登录容器进行调试。

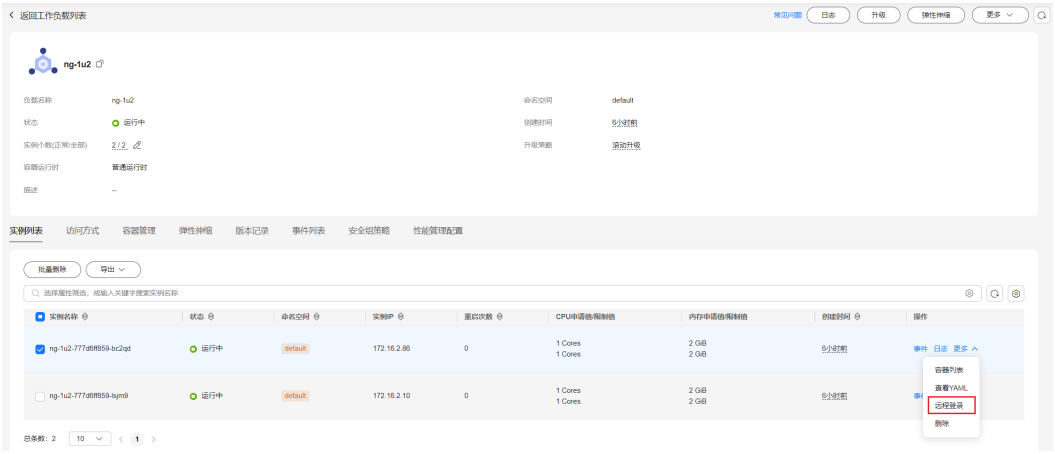
使用 CloudShell 登录容器

步骤1 登录CCE控制台，单击集群名称进入集群。

步骤2 在左侧选择“工作负载”，单击目标工作负载名称，查看工作负载的实例列表。

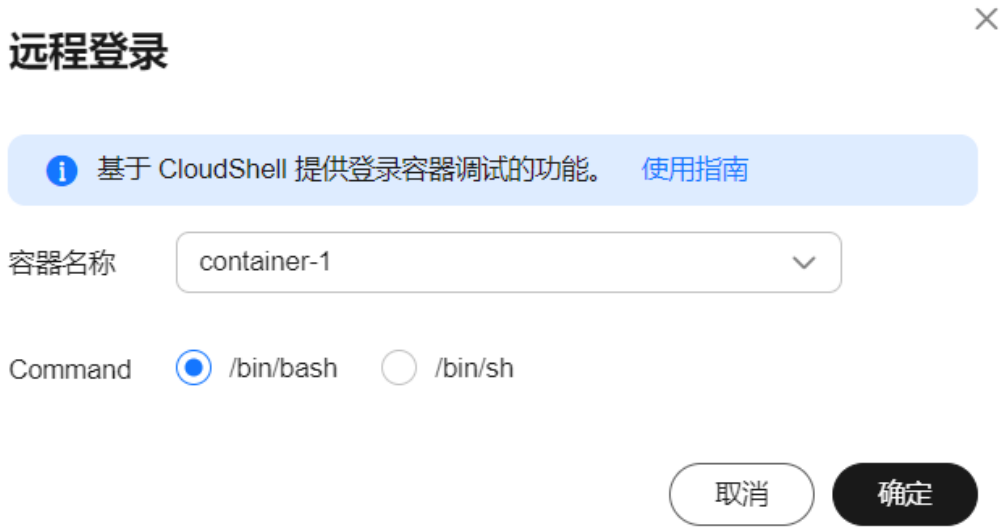
步骤3 单击目标实例操作列中的“更多 > 远程登录”。

图 2-15 登录容器



步骤4 在弹出窗口中选择要登录的容器以及命令，然后单击“确定”。

图 2-16 选择登录的容器与命令

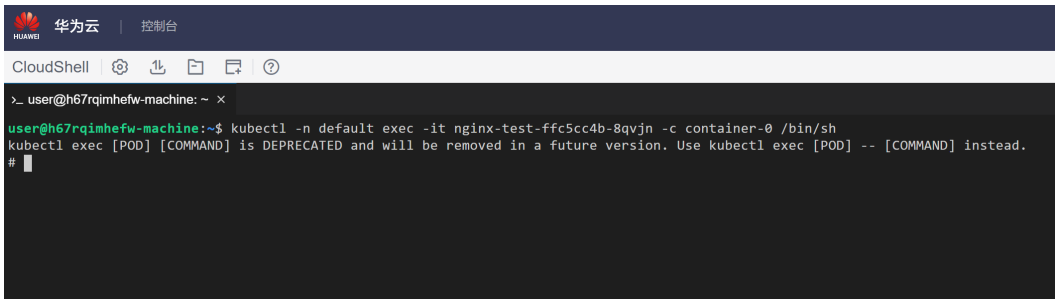


步骤5 页面会自动跳转到CloudShell，并初始化启动kubectl，然后自动执行kubectl exec命令登录到容器。

说明

请等待kubectl exec 命令自动执行后再操作，此命令出现需要一段时间 5-10秒。

图 2-17 CloudShell 页面



----结束

使用 kubectl 命令登录容器

步骤1 使用kubectl连接集群，详情请参见**通过kubectl连接集群**。

步骤2 执行以下命令，查看已创建的Pod。

```
kubectl get pod
```

示例输出如下：

```
NAME                     READY   STATUS    RESTARTS      AGE
nginx-59d89cb66f-mhljr   1/1     Running   0             11m
```

步骤3 查询该Pod中的容器名称。

```
kubectl get po nginx-59d89cb66f-mhljr -o jsonpath='{range .spec.containers[*]}{.name}{end}{"\n"}'
```

示例输出如下：

```
container-1
```

步骤4 执行以下命令，登录到nginx-59d89cb66f-mhljr这个Pod中名为container-1的容器。

```
kubectl exec -it nginx-59d89cb66f-mhljr -c container-1 -- /bin/sh
```

步骤5 如需退出容器，可执行**exit**命令。

----结束

2.5 管理工作负载和任务

操作场景

工作负载创建后，您可以对其执行升级、编辑YAML、日志、监控、回退、删除等操作。

表 2-12 工作负载/任务管理

操作	描述
日志	可查看工作负载的日志信息。
升级	可以通过更换镜像或镜像版本实现无状态工作负载、有状态工作负载的快速升级，业务无中断。
编辑YAML	可通过在线YAML编辑窗对无状态工作负载、有状态工作负载、定时任务和容器组的YAML文件进行修改和下载。普通任务的YAML文件仅支持查看、复制和下载。 **说明** 如果对已有的定时任务（CronJob）进行修改，修改之后运行的新Pod将使用新的配置，而已经运行的Pod将继续运行不会发生任何变化。
回退	无状态工作负载可以进行回退操作，仅无状态工作负载可用。
重新部署	工作负载可以进行重新部署操作，重新部署后将重启负载下的全部容器组Pod。
关闭/开启升级	无状态工作负载可以进行关闭/开启升级操作，仅无状态工作负载可用。
标签管理	标签是以key/value键值对的形式附加在工作负载上的。添加标签后，可通过标签对工作负载进行管理和选择。任务或定时任务无法使用标签管理功能。
删除	若工作负载无需再使用，您可以将工作负载或任务删除。工作负载或任务删除后，将无法恢复，请谨慎操作。
事件	查看具体实例的事件名称、事件类型、发生次数、Kubernetes事件、首次和最近发生的时间。
停止/启动	停止/启动一个定时任务，该功能仅定时任务可用。

日志

您可以通过“日志”功能查看无状态工作负载、有状态工作负载、普通任务的日志信息。本文以无状态工作负载为例说明如何查看日志。

须知

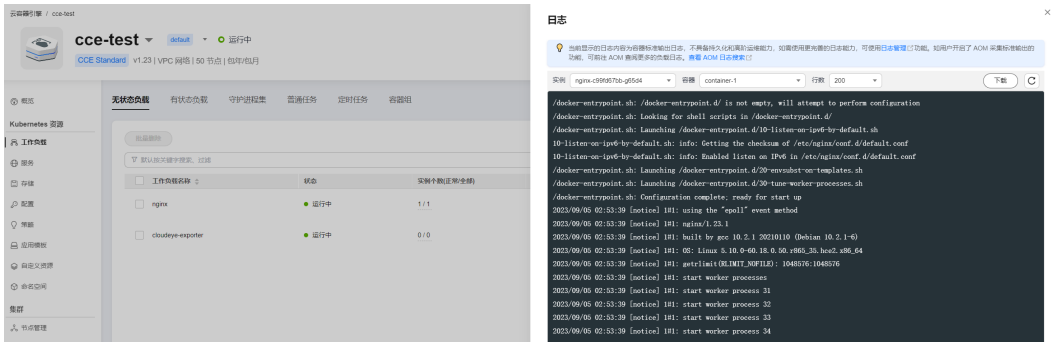
查看日志前请将浏览器与后端服务器时间调成一致。

步骤1 登录CCE控制台，进入一个已有的集群，在左侧导航栏中选择“工作负载”。

步骤2 选择“无状态负载”页签，单击工作负载后的“日志”。

在弹出的“日志”窗口中可以查看容器日志信息。

图 2-18 查看无状态工作负载日志



----结束

升级

您可以通过CCE控制台实现无状态工作负载、有状态工作负载的快速升级。

本文以无状态工作负载为例说明如何进行升级。

若需要更换镜像或镜像版本，您需要提前将镜像上传到容器镜像服务，上传方法请参见**通过Docker客户端上传镜像**。

步骤1 登录CCE控制台，进入一个已有的集群，在左侧导航栏中选择“工作负载”。

步骤2 选择“无状态负载”页签，单击待升级工作负载后的“升级”。

说明

- 暂不支持批量升级多个工作负载。
- 有状态工作负载升级时，若升级类型为替换升级，需要用户手动删除实例后才能升级成功，否则界面会始终显示“处理中”。

步骤3 请根据业务需求进行工作负载的升级，参数设置方法与创建工作负载时一致。

步骤4 更新完成后，单击“升级工作负载”，并手动确认YAML文件差异后提交升级。

----结束

编辑 YAML

可通过在线YAML编辑窗对无状态工作负载、有状态工作负载、定时任务和容器组的YAML文件进行修改和下载。普通任务的YAML文件仅支持查看、复制和下载。本文以无状态工作负载为例说明如何在线编辑YAML。

步骤1 登录CCE控制台，进入一个已有的集群，在左侧导航栏中选择“工作负载”。

步骤2 选择“无状态负载”页签，单击工作负载后的“更多 > 编辑YAML”，在弹出的“编辑YAML”窗中可对当前工作负载的YAML文件进行修改。

步骤3 单击“确定”，完成修改。

步骤4 （可选）在“编辑YAML”窗中，单击“下载”，可下载该YAML文件。

----结束

回退（仅无状态工作负载可用）

所有无状态工作负载的发布历史记录都保留在系统中，您可以回退到指定的版本。

步骤1 登录CCE控制台，进入一个已有的集群，在左侧导航栏中选择“工作负载”。

步骤2 选择“无状态负载”页签，单击待回退工作负载后的“更多 > 回退”。

步骤3 切换至“版本记录”页签，并选择回退版本，单击“回退到此版本”，并手动确认YAML文件差异后单击“确定”。

图 2-19 回退工作负载版本

----结束

重新部署

重新部署将重启负载下的全部容器组Pod。本文以无状态工作负载为例说明如何重新部署工作负载。

步骤1 登录CCE控制台，进入一个已有的集群，在左侧导航栏中选择“工作负载”。

步骤2 选择“无状态负载”页签，单击工作负载后的“更多 > 重新部署”。

步骤3 在弹出的提示框中单击“是”，即可完成工作负载的重新部署。

----结束

关闭/开启升级（仅无状态工作负载可用）

无状态工作负载可以进行“关闭/开启升级”操作。

- 关闭升级后，对负载进行的升级操作可以正常下发，但不会被应用到实例。

  如果您正在滚动升级的过程中，滚动升级会在关闭升级命令下发后停止，出现新旧实例共存的状态。
- 开启升级后，负载可以正常升级和回退，负载下的实例会与负载当前的最新信息进行一次同步，如果有不一致的，则会自动按照负载的最新信息进行升级。

须知

工作负载状态在关闭升级时无法执行回退操作。

步骤1 登录CCE控制台，进入一个已有的集群，在左侧导航栏中选择“工作负载”。

步骤2 选择“无状态负载”页签，单击工作负载后方操作栏中的“更多 > 关闭/开启升级”。

步骤3 在弹出的信息提示框中，单击“是”。

----结束

标签管理

标签是以key/value键值对的形式附加在工作负载上的。添加标签后，可通过标签对工作负载进行管理和选择。您可以给多个工作负载打标签，也可以给指定的某个工作负载打标签。

步骤1 登录CCE控制台，进入一个已有的集群，在左侧导航栏中选择“工作负载”。

步骤2 选择“无状态负载”页签，单击工作负载后方操作栏中的“更多 > 标签管理”。

步骤3 单击“添加”，输入键和值后单击“确定”。

图 2-20 标签管理

说明

标签格式要求如下：以字母和数字开头或结尾，由字母、数字、连接符（-）、下划线（_）、点号（.）组成且63字符以内。

----结束

删除工作负载/任务

若工作负载无需再使用，您可以将工作负载或任务删除。工作负载或任务删除后，将无法恢复，请谨慎操作。本文以无状态工作负载为例说明如何使用删除功能。

步骤1 登录CCE控制台，进入一个已有的集群，在左侧导航栏中选择“工作负载”。

步骤2 单击待删除工作负载后的“更多 > 删除”，删除工作负载。

请仔细阅读系统提示，删除操作无法恢复，请谨慎操作。

步骤3 单击“是”。

说明

- 若Pod所在节点不可用或者关机，负载无法删除时可以在详情页面实例列表选择强制删除。
- 请确保要删除的存储没有被其他负载使用，导入和存在快照的存储只做解关联操作。

----结束

事件

本文以无状态工作负载为例说明如何使用事件功能。任务或定时任务中的事件功能可直接单击工作负载操作栏中的“事件”按钮查看。

步骤1 登录CCE控制台，进入一个已有的集群，在左侧导航栏中选择“工作负载”。

步骤2 选择“无状态负载”页签，单击工作负载名称，可在“实例列表”中单击某个实例的“事件”按钮，查看该工作负载或具体实例的事件名称、事件类型、发生次数、Kubernetes事件、首次和最近发生的时间。

说明

事件保存时间为1小时，1小时后自动清除数据。

----结束

2.6 管理内核参数配置

CCE Autopilot是云容器引擎服务推出的Serverless版集群，同物理机系统内核隔离且互不影响。对于资深业务部署场景，内核参数调优是比较通用的方式。在安全范围内，CCE Autopilot服务允许客户根据Kubernetes社区推荐的方案，通过Pod的安全上下文（Security Context）对内核参数进行配置，极大提升用户业务部署的灵活性。如果您对securityContext概念不够熟悉，更多信息可阅读**Security Context**。

在 Linux 中，最通用的内核参数修改方式是通过sysctl接口进行配置。在Kubernetes中，也是通过Pod的sysctl安全上下文（Security Context）对内核参数进行配置，如果您对sysctl概念不够熟悉，可阅读**在 Kubernetes 集群中使用 sysctl**。安全上下文（Security Context）作用于同一个Pod内的所有容器。

CCE Autopilot服务支持修改的非安全的sysctl参数范围如下：

```
kernel.shm*,
kernel.msg*,
kernel.sem,
fs.mqueue.*,
net.*
```

须知

为了避免破坏操作系统的稳定性，请您在了解变更后果之后再修改sysctl参数。

有命名空间的sysctl参数，在未来的Linux内核版本中，可能会发生变化。

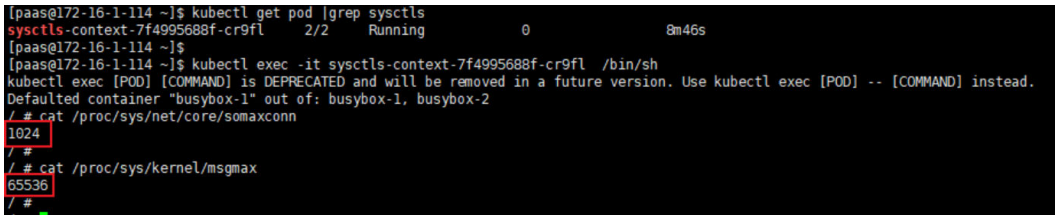
由于非安全的sysctl参数其本身具有不稳定性，在使用非安全的 sysctl 参数时可能会导致一些严重问题， 如容器的错误行为，用户需自行承担风险。

以下示例中，使用Pod SecurityContext来对两个sysctl参数kernel.msgmax和net.core.somaxconn进行设置。

```
apiVersion: v1
kind: Pod
metadata:
  name: sysctls-context-example
spec:
  securityContext:
    sysctls:
    - name: kernel.msgmax
      value: "65536"
    - name: net.core.somaxconn
      value: "1024"
  ...
```

进入容器确认配置生效：

```
kubectl exec -it podname -c container-1 -- /bin/sh
```



2.7 管理自定义资源

自定义资源定义（Custom Resource Definition，CRD）是对Kubernetes API的扩展，当默认的Kubernetes资源无法满足业务需求时，您可以通过CRD对象来定义新的资源类别。 根据CRD的定义，您可以在集群中创建自定义资源（Custom Resource，CR）来满足业务需求。 CRD允许用户创建新的资源类别的同时又不必添加新的Kubernetes API服务器，从而有效提高集群管理的灵活性。

创建 CRD

步骤1 登录CCE控制台。

步骤2 单击集群名称进入集群，在左侧选择“自定义资源”，在右上角单击“YAML创建”。

步骤3 输入YAML来新建CRD。CRD的YAML定义需要根据业务需求进行定制，详情请参见**使用CustomResourceDefinition扩展Kubernetes API**。

步骤4 单击“确定”。

----结束

查看 CRD 及其对应的资源

步骤1 登录CCE控制台。

步骤2 单击集群名称进入集群，在左侧选择“自定义资源”。

步骤3 在自定义资源页面，查看CRD或CRD对应的资源对象。

- 查看CRD及其YAML

  列表中列出了集群中所有CRD，以及对应的API组、API版本、资源作用范围，单击操作列中的“查看YAML”按钮即可查看CRD的YAML。

  您可以通过上方的搜索框，使用关键词搜索全部资源类型。

- 查看CRD对应的资源对象

  在列表中选择一个自定义资源类型，单击操作列中的“查看资源”按钮即可浏览对应的资源对象。

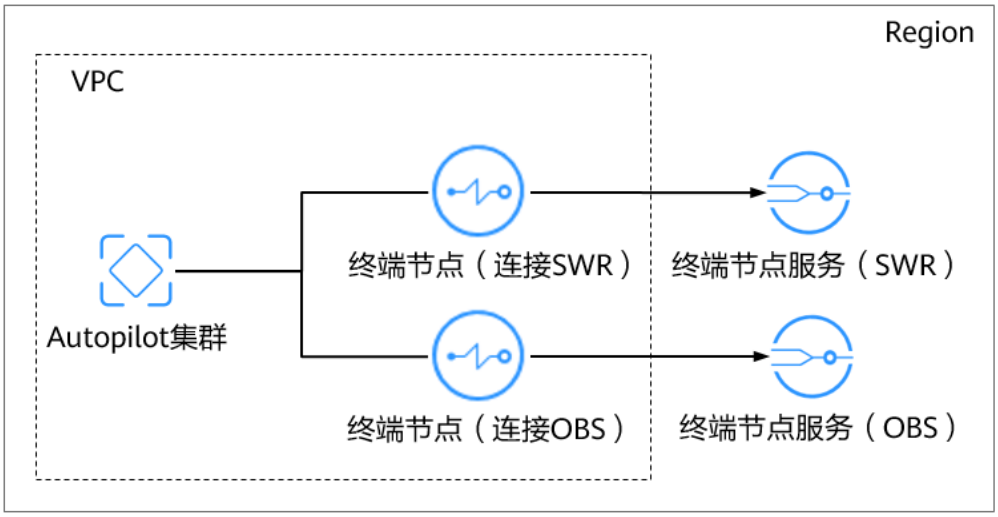
----结束

2.8 配置访问 SWR 和 OBS 服务的 VPC 终端节点

Autopilot集群中创建工作负载时，需要通过SWR服务和OBS服务的VPC终端节点来拉取镜像。

关于VPC终端节点详情请参见**什么是VPC终端节点?**

图 2-21 Autopilot 集群访问 SWR 和 OBS



配置访问 SWR 服务的 VPC 终端节点

步骤1 登录**VPC终端节点控制台**。

步骤2 在“终端节点”页面，单击“购买终端节点”。

步骤3 根据界面提示配置必选参数。

表 2-13 终端节点配置参数

参数	说明
区域	终端节点所在区域，需要与Autopilot集群所在区域保持一致。
计费方式	此处选择按需计费。
服务类别	选择“按名称查找服务”。
服务名称	参考表2-14，根据集群所在区域填写，并单击“验证”。
虚拟私有云	需要选择Autopilot集群所在的虚拟私有云。
子网	选择一个已有子网。
IPv4地址	默认可选择“自动分配IPv4地址”，您也可以根据需求选择“手动指定IP地址”。

表 2-14 SWR 服务名称

区域	名称
华南-广州友好	cn-south-4.SWR.f80386a2-ce16-4f92-9df9-20f7fc01e7a2
西南-贵阳一	com.myhuaweicloud.cn-southwest-2.swr
华南-广州	swr.cn-south-1.myhuaweicloud.com
华东-上海一	com.myhuaweicloud.cn-east-3.swr
华北-北京四	com.myhuaweicloud.cn-north-4.swr
亚太-曼谷	ap-southeast-2.SWR.ac7067e1-f8d1-4f5c-abe1-0f78960e5d4c
亚太-新加坡	com.myhuaweicloud.ap-southeast-3.swr

图 2-22 创建 SWR 服务的 VPC 终端节点

步骤4 参数配置完成，单击“立即购买”，进行规格确认。

- 规格确认无误，单击“提交”，任务提交成功。
- 参数信息配置有误，需要修改，单击“上一步”，修改参数，然后单击“提交”。

步骤5 返回终端节点列表，如果终端节点状态为“已接受”，表示终端节点已成功连接至终端节点服务。

----结束

配置访问 OBS 服务的 VPC 终端节点

步骤1 登录**VPC终端节点控制台**。

步骤2 在“终端节点”页面，单击“购买终端节点”。

步骤3 根据界面提示配置必选参数。

表 2-15 终端节点配置参数

参数	说明
区域	终端节点所在区域，需要与Autopilot集群所在区域保持一致。
计费方式	此处选择按需计费。
服务类别	选择“按名称查找服务”。
服务名称	参考**表2-16**，根据集群所在区域填写，并单击“验证”。
虚拟私有云	需要选择Autopilot集群所在的虚拟私有云。
路由表	选择一个已有的路由表。

表 2-16 OBS 服务名称

区域	名称
华南-广州友好	cn-south-4.com.myhuaweicloud.v4.obsv2
西南-贵阳一	cn-southwest-2.com.myhuaweicloud.v4.obsv2
华南-广州	cn-south-1.com.myhuaweicloud.v4.obsv2
华东-上海一	cn-east-3.com.myhuaweicloud.v4.global.obsv2
华北-北京四	cn-north-4.com.myhuaweicloud.v4.obsv2
亚太-曼谷	ap-southeast-2.myhuaweicloud.v4.obsv2
亚太-新加坡	ap-southeast-3.com.myhuaweicloud.v4.obsv2

图 2-23 创建 OBS 服务的 VPC 终端节点

步骤4 参数配置完成，单击“立即购买”，进行规格确认。

- 规格确认无误，单击“提交”，任务提交成功。
- 参数信息配置有误，需要修改，单击“上一步”，修改参数，然后单击“提交”。

步骤5 返回终端节点列表，如果终端节点状态为“已接受”，表示终端节点已成功连接至终端节点服务。

----结束

3 网络

3.1 网络概述

关于集群的网络，可以从如下两个角度进行了解：

- 集群网络是什么样的：集群中包含有哪些网络，各自的用处是什么，具体请参见**集群网络构成**。
- 集群中的Pod是如何访问的：访问Pod就是访问容器，也就是访问用户的业务，Kubernetes提供**Service**和**Ingress**来解决Pod的访问问题。本章节根据用户使用场景总结了常见的**网络访问场景**，让您能够在不同使用场景下选择合适的使用方法。

集群网络构成

集群的网络可以分成两部分：

- **容器网络**

  为集群内Pod分配IP地址。CCE Autopilot集群继承Kubernetes的IP-Per-Pod-Per-Network的容器网络模型，即每个Pod在每个网络平面下都拥有一个独立的IP地址，Pod内所有容器共享同一个网络命名空间，集群内所有Pod都在一个直接连通的扁平网络中，无需NAT可直接通过Pod的IP地址访问。Kubernetes只提供了如何为Pod提供网络的机制，并不直接负责配置Pod网络；Pod网络的具体配置操作交由具体的容器网络插件实现。容器网络插件负责为Pod配置网络并管理容器IP地址。

  当前CCE Autopilot集群仅支持云原生网络2.0容器网络模型。云原生网络2.0是自研的新一代容器网络模型，深度整合了虚拟私有云VPC的弹性网卡（Elastic Network Interface，简称ENI）和辅助弹性网卡（Sub Network Interface，简称Sub-ENI）的能力，直接从VPC网段内分配容器IP地址，支持ELB直通容器，绑定安全组，绑定弹性公网IP，享有高性能。

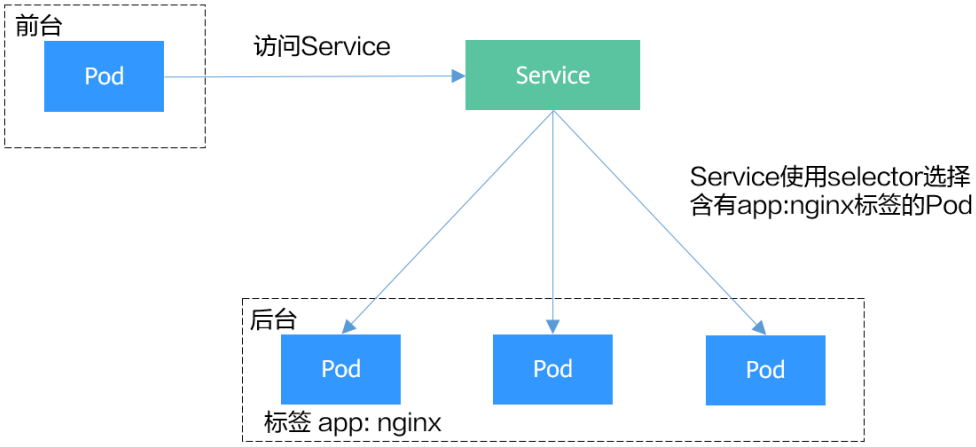
- **服务网络**

  服务（Service）是Kubernetes内的概念，每个Service都有一个固定的IP地址，在CCE上创建集群时，可以指定Service的地址段（即服务网段）。服务网段不能和容器子网重叠。服务网段是集群的ClusterIP类型服务的IP地址范围，只在集群内使用，不能在集群外使用。

Service

Service是用来解决Pod访问问题的。每个Service有一个固定IP地址，Service将访问流量转发给Pod，而且Service可以给这些Pod做负载均衡。

图 3-1 通过 Service 访问 Pod



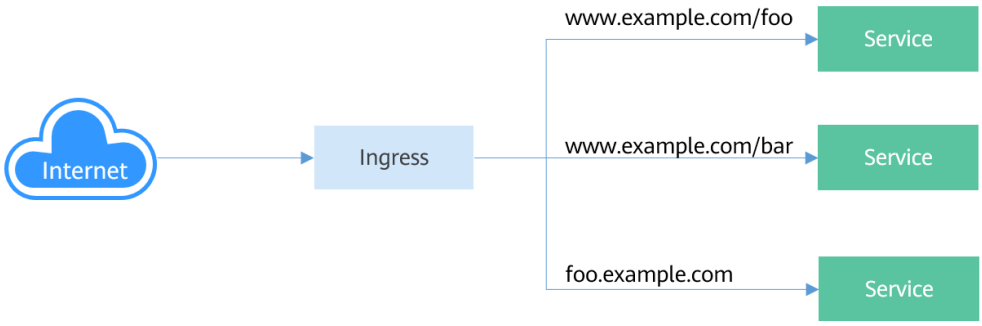
根据创建Service的类型不同，可分成如下模式：

- ClusterIP：用于在集群内部互相访问的场景，通过ClusterIP访问Service。
- LoadBalancer：用于从集群外部访问的场景，通过一个特定的LoadBalancer访问Service，这个LoadBalancer将请求直接转发到Pod的SubENI，而外部只需要访问LoadBalancer。

Ingress

Service是基于四层TCP和UDP协议转发的，而Ingress可以基于七层的HTTP和HTTPS协议转发，可以通过域名和路径做到更细粒度的划分，如下图所示。

图 3-2 Ingress-Service



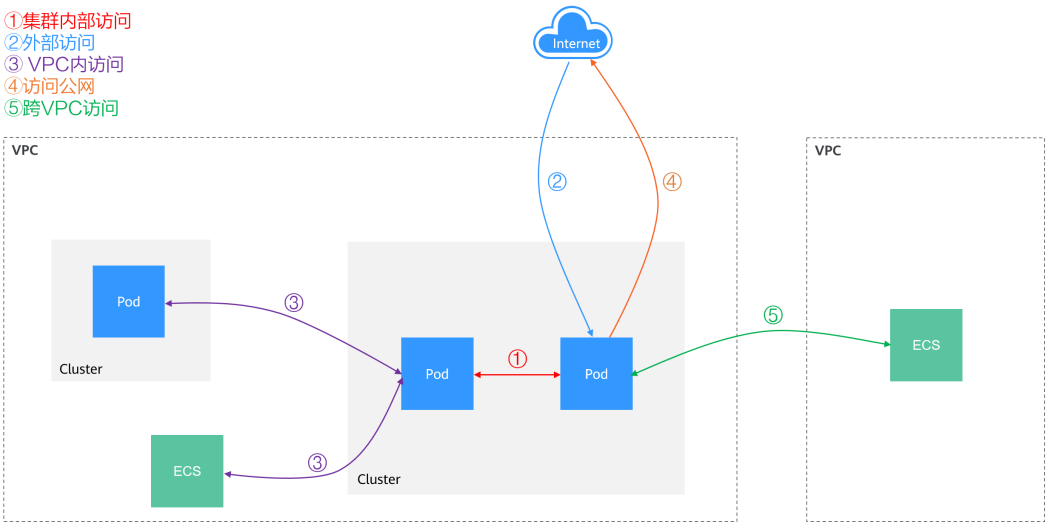
网络访问场景

工作负载网络访问可以分为如下几种场景。

- 从集群内部访问工作负载：创建ClusterIP类型的Service，通过Service访问工作负载。

- 从集群外部访问工作负载：从集群外部访问工作负载推荐使用Service（LoadBalancer类型）或Ingress访问。
  - 通过公网访问工作负载：需要LoadBalancer绑定公网IP。
  - 通过内网访问工作负载：通过LoadBalancer的内网IP即可访问工作负载。如果跨VPC需要通过对等连接等手段打通不同VPC网络。
- 工作负载访问外部网络：
  - 工作负载访问内网：负载访问内网地址，在不同容器网络模型下有不同的表现，需要注意在对端安全组放通容器网段。
  - 工作负载访问公网：访问公网有几种方法可以实现，一是给Pod IP绑定公网IP（**为Pod配置EIP**），另一个是通过NAT网关配置SNAT规则（**从容器访问公网**）。

图 3-3 网络访问示意图



3.2 容器网络

3.2.1 配置集群容器子网

操作场景

当创建CCE Autopilot集群时设置的容器子网太小，无法满足业务扩容需求时，您通过扩展集群容器子网的方法来解决。本文介绍如何为CCE Autopilot集群添加和删除容器子网。

说明

1.27.8-r0，1.28.6-r0及以上版本的集群支持删除容器子网。

为 CCE Autopilot 集群添加容器子网

步骤1 登录CCE控制台，单击CCE Autopilot集群名称，进入集群。

步骤2 在“概览”页面，找到“网络信息”版块，并单击“添加”。

图 3-4 添加容器子网

步骤3 选择同一VPC下的容器子网，您可一次性添加多个容器子网。如没有其他可用的容器子网，可前往VPC控制台创建。

图 3-5 选择容器子网

步骤4 单击“确定”。

----结束

为 CCE Autopilot 集群删除容器子网

须知

删除容器子网属高危操作，请确保当前集群中没有已经使用待删除子网的网卡。

具体步骤：您在集群概览页，在“基本信息”模块中查看集群ID，在“网络信息 > 默认容器子网”复制待删除容器子网的子网ID。在**弹性网卡**页面的“弹性网卡”和“辅助弹性网卡”列表中，通过子网ID进行筛选，如果筛选出的网卡“名称”或者“描述”里包含当前集群的ID，表示网卡被集群占用。

步骤1 登录CCE控制台，单击对应集群名称，进入集群。

步骤2 在左侧导航栏中，单击“集群 > 配置中心”，切换至“网络配置”页签。

步骤3 在“容器网络配置”中，单击default-network（默认容器子网）后的“更新”。

步骤4 在“容器子网（Pod CIDR）”中**取消选择**需要删除的容器子网，单击“确定”。

图 3-6 删除容器子网

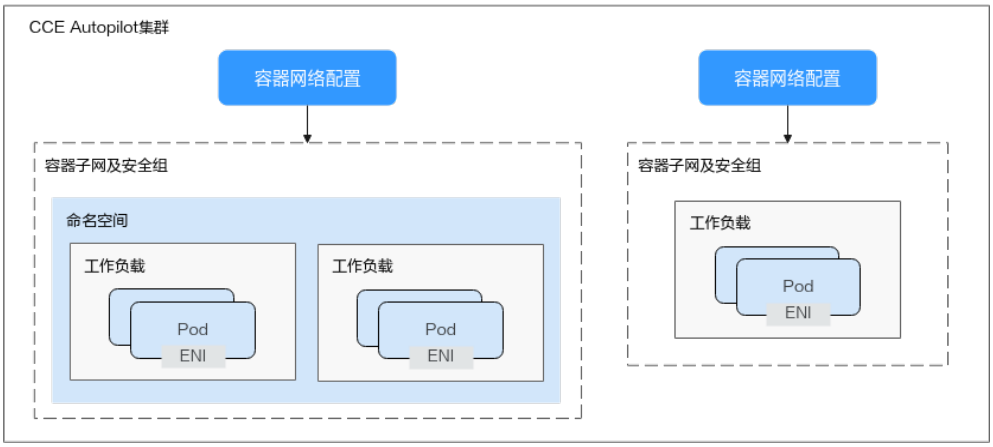
----结束

3.2.2 使用容器网络配置为命名空间/工作负载绑定子网及安全组

操作场景

CCE Autopilot集群支持以命名空间或工作负载粒度设置容器所在的容器子网及安全组，该功能通过名为NetworkAttachmentDefinition的**CRD**资源实现。如您想为指定的命名空间或工作负载配置指定的容器子网和安全组，可创建自定义容器网络配置（NetworkAttachmentDefinition），并将该容器网络配置与相应的命名空间或工作负载关联，进而实现业务的子网划分或业务安全隔离的诉求。

图 3-7 自定义容器网络配置示意图



目前容器网络配置（NetworkAttachmentDefinition）支持关联的资源类型对比如下：

表 3-1 关联资源类型对比

维度	容器网络配置（NetworkAttachmentDefinition）关联的资源类型	
	命名空间	工作负载
容器网络划分维度说明	为命名空间关联容器网络配置，该命名空间下创建的所有工作负载使用相同的子网配置跟安全组配置。	为工作负载关联容器网络配置，指定了相同容器网络配置的工作负载使用相同的子网配置跟安全组配置。
支持的CCE Autopilot集群版本	v1.27.8-r0, v1.28.6-r0及以上版本。	v1.27.8-r0, v1.28.6-r0及以上版本。
约束限制	同一个命名空间最多只能被关联到一个容器网络配置。	只能指定未关联命名空间的自定义容器网络配置。

说明

- Pod使用的容器网络配置优先级如下：Pod直接关联的容器网络配置 > Pod的命名空间关联的容器网络配置 > 集群默认容器网络配置（default-network）。
- 集群中存在默认容器网络配置default-network，对所有未配置自定义容器网络配置的Pod生效，“概览”页面的网络信息中的“默认容器子网”即为default-network中的容器子网。default-network不支持删除。
- 当集群中只有一个容器网络配置，该容器网络配置即为默认容器网络配置；当集群中有多个容器网络配置时，带有名为“yangtse.io/default-network: true”annotation的容器网络配置即为默认容器网络配置。当集群中不存在默认容器网络配置时，未关联任何容器网络配置的Pod创建后会由于无法分配到网卡而启动失败。

约束与限制

如需删除创建的自定义容器网络配置(NetworkAttachmentDefinition)，请先删除对应的命名空间下使用该配置创建的Pod（带有名为“cni.yangtse.io/network-status”的annotation），详情请参见**删除网络配置**。

创建命名空间类型的容器网络配置

创建命名空间类型的容器网络配置后，关联命名空间下创建的所有工作负载使用相同的子网配置跟安全组配置。

通过控制台创建命名空间类型的容器网络配置

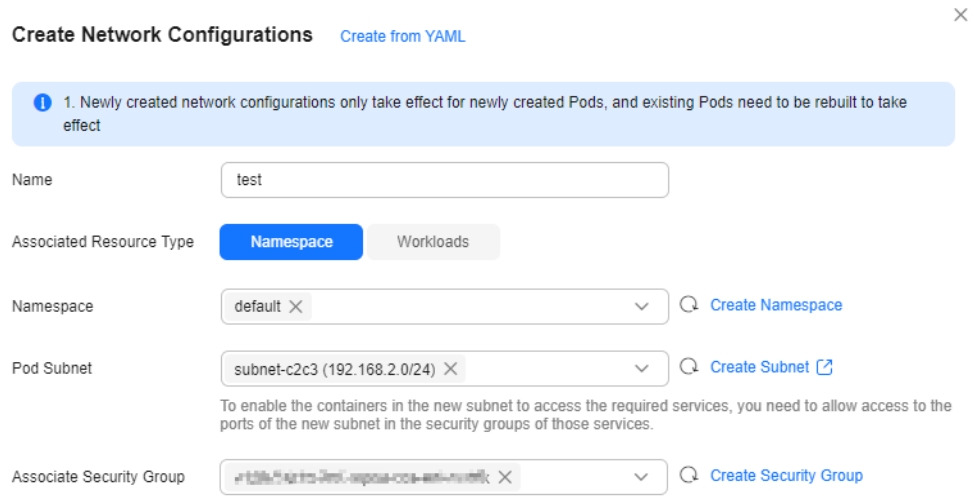
步骤1 登录CCE控制台。

步骤2 单击集群名称进入集群，在左侧选择“配置中心”，选择“网络配置”页签。

步骤3 查看“容器网络配置”，单击“添加自定义容器网络配置”，在弹窗中配置容器子网和安全组等信息。

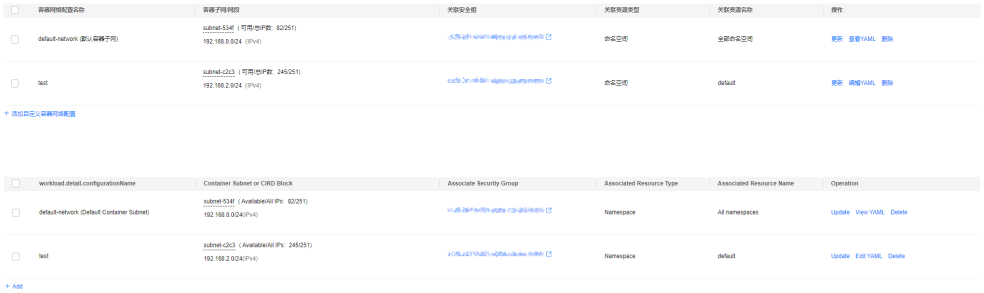
- 名称：自定义容器网络配置名称，最长支持63个字符。default-network、default、mgnt0、mgnt1四个名称为系统预留，请勿使用。
- 关联资源类型：自定义容器网络配置关联的资源类型，详情请参见**表3-1**。创建命名空间类型的容器网络配置请选择“命名空间”类型。
- 命名空间：请选择您需要关联的命名空间。不同容器网络配置之间关联的命名空间不可重复。若无命名空间可选请单击后方的“创建命名空间”进行创建。
- 容器子网：请选择子网，最多可选择100个子网。若无子网可选请单击后方的“创建子网”进行创建，创建完成后单击刷新按钮。
- 关联安全组：默认为容器ENI安全组，您也可以选择单击后方的“创建安全组”进行创建，创建完成后单击刷新按钮。最多可选择5个安全组。

图 3-8 创建命名空间类型的容器网络配置



步骤4 完成基本配置后单击“确定”，创建完成后页面自动返回到自定义容器网络配置列表，可以看到新创建的容器网络配置已在列表中。

图 3-9 容器网络配置列表



----结束

通过 kubectl 命令行创建命名空间类型的容器网络配置

创建命名空间类型的容器网络配置后，该命名空间下创建的所有工作负载使用相同的子网配置跟安全组配置。

步骤1 请参见**通过kubectl连接集群**，使用kubectl连接集群。

步骤2 创建YAML文件networkattachment-test.yaml，用于创建自定义容器网络配置并关联命名空间，文件名称可自定义。

```
vim networkattachment-test.yaml
```

文件内容如下，：

```
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"
    }
  }
 }
}
```

表 3-2 关键参数说明

参数	是否必填	参数类型	描述
apiVersion	是	String	表示API的版本号，固定为k8s.cni.cncf.io/v1。
kind	是	String	创建的对象类别，固定为NetworkAttachmentDefinition。
yangtse.io/project-id	是	String	当前所在Region的项目ID，获取方式请参见**获取项目ID**。
name	是	String	配置名称，可自定义。名称由小写字母、数字、横线（-）和点组成，且必须以字母或数字开头和结尾，最长支持63个字符。default-network、default、mgnt0、mgnt1四个名称为系统预留，请勿使用。
namespace	是	String	配置资源所在命名空间，固定为kube-system。
config	是	Object	配置内容，为json格式的字符串。 config字段数据结构具体说明，请参见**表2 config字段数据结构说明**。

表 3-3 config 字段数据结构说明

参数	是否必填	参数类型	描述
type	是	String	表示网络类型，固定为eni-neutron。
args	是	Object	安全组和子网的配置参数。 args字段数据结构具体说明，请参见**表3-4**。

参数	是否必填	参数类型	描述
selector	否	Object	选择该配置所作用的命名空间。 selector字段数据结构具体说明，请参见表3-5。

表 3-4 args 字段数据结构说明

参数	是否必填	参数类型	描述
securityGroups	否	String	需要配置的安全组ID，最多关联5条安全组。如果没有对安全组进行规划，请和default-network中的安全组保持一致。 获取方式： 登录虚拟私有云控制台，在左侧导航栏选择“访问控制 > 安全组”，单击安全组名称，在“基本信息”页签下找到“ID”字段复制即可。
subnets	是	Array of subnetID Objects	容器子网ID列表，至少需填写1个，最多可填写100个，具体格式如下： `[{"subnetID":"27d95**"},{"subnetID":"827bb**"},{"subnetID":"bdd6b**"}]` 容器子网要求与集群在同一VPC下。 获取方式： 登录虚拟私有云控制台，在左侧导航栏选择“虚拟私有云 > 子网”，单击子网名称，在“基本信息”页签下找到“子网ID”字段复制即可。

表 3-5 selector 字段数据结构说明

参数	是否必填	参数类型	描述
namespaceSelector	否	matchLabels Object	该选择器为Kubernetes标准的选择器，需填写命名空间标签，格式如下： `"matchLabels":{ "kubernetes.io/metadata.name":"default" }` 不同配置之间的命名空间不可重合。

步骤3 创建NetworkAttachmentDefinition。

```
kubectl create -f networkattachment-test.yaml
```

回显如下，表示NetworkAttachmentDefinition已创建。

```
networkattachmentdefinition.k8s.cni.cncf.io/example created
```

----结束

创建工作负载类型的容器网络配置

创建工作负载类型的容器网络配置后，需要为工作负载关联容器网络配置，指定了相同容器网络配置的工作负载使用相同的子网配置跟安全组配置。

通过控制台创建工作负载类型的容器网络配置

步骤1 登录CCE控制台。

步骤2 单击集群名称进入集群，在左侧选择“配置中心”，选择“网络配置”页签。

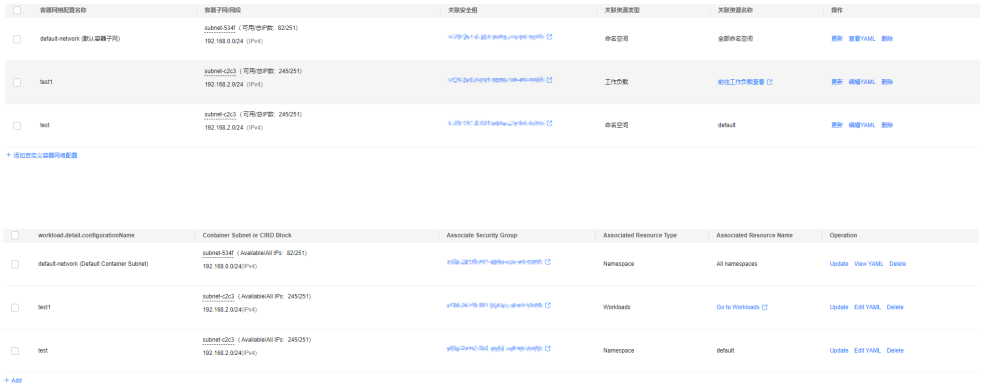
步骤3 查看“容器网络配置”，单击“添加自定义容器网络配置”，在弹窗中配置容器子网和安全组等信息。

- 名称：自定义容器网络配置名称，最长支持63个字符。default-network、default、mgnt0、mgnt1四个名称为系统预留，请勿使用。
- 关联资源类型：自定义容器网络配置关联的资源类型，详情请参见**表3-1**。创建工作负载类型的容器网络配置请选择“工作负载”类型。
- 容器子网：请选择子网，最多可选择100个子网。若无子网可选请单击后方的“创建子网”进行创建，创建完成后单击刷新按钮。
- 关联安全组：默认为容器ENI安全组，您也可以选择单击后方的“创建安全组”进行创建，创建完成后单击刷新按钮。最多可选择5个安全组。

图 3-10 创建工作负载类型的容器网络配置

步骤4 完成基本配置后单击“确定”，创建完成后页面自动返回到自定义容器网络配置列表，可以看到新创建的容器网络配置已在列表中。

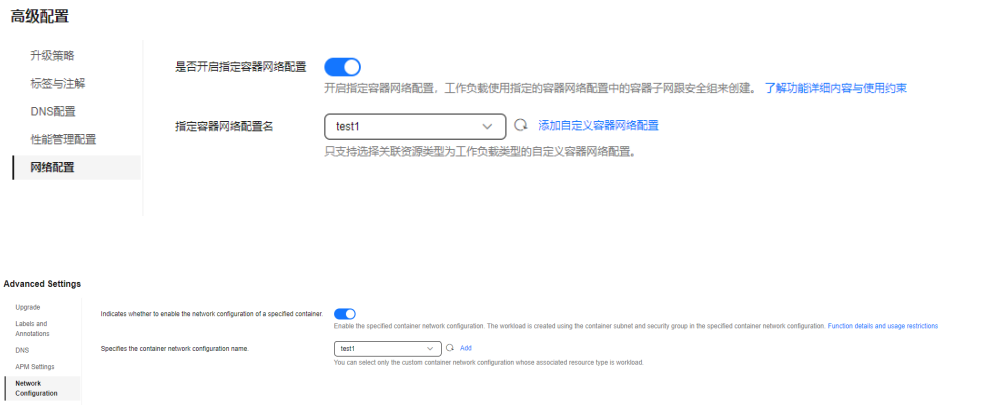
图 3-11 容器网络配置列表



步骤5 在创建工作负载时，可选择自定义的容器网络配置。

1. 在左侧导航栏中选择“工作负载”，在右侧选择“无状态负载”页签。
2. 单击页面右上角“创建工作负载”，在“高级配置”中选择“网络配置”页签，并选择是否开启指定容器网络配置。
3. 选择一个已有的容器网络配置。如果没有可用的网络配置，可单击“添加自定义容器网络配置”进行创建。

图 3-12 选择容器网络配置



4. 工作负载其余信息都配置完成后，单击“创建工作负载”。

   返回到“配置中心”，在容器网络配置列表里可以看到创建的容器网络配置关联的资源名称。

图 3-13 查看容器网络配置关联的资源

----结束

通过 kubectl 命令行创建工作负载类型的容器网络配置

创建工作负载类型的容器网络配置后，需要为工作负载关联容器网络配置，关联相同容器网络配置的工作负载将使用相同的子网配置跟安全组配置。

步骤1 请参见**通过kubectl连接集群**，使用kubectl连接集群。

步骤2 创建YAML文件networkattachment-test.yaml，用于自定义容器网络配置，文件名称可自定义。

```
vim networkattachment-test.yaml
```

文件内容如下：

```
apiVersion: k8s.cni.cncf.io/v1
kind: NetworkAttachmentDefinition
metadata:
  annotations:
    yangtse.io/project-id: 80d5a**
  name: example
  namespace: kube-system
spec:
  config: |
   {
    "type":"eni-neutron",
    "args":{
      "securityGroups":"f4983**",
      "subnets":[
        {
          "subnetID":"5594b**"
        }
      ]
    }
   }
```

表 3-6 关键参数说明

参数	是否必填	参数类型	描述
apiVersion	是	String	表示API的版本号。固定为k8s.cni.cncf.io/v1。
kind	是	String	创建的对象类别。固定为NetworkAttachmentDefinition。
yangtse.io/project-id	是	String	当前所在Region的项目ID，获取方式请参见**获取项目ID**。

参数	是否必填	参数类型	描述
name	是	String	配置名称，可自定义。名称由小写字母、数字、横线（-）和点组成，且必须以字母或数字开头和结尾，最长支持63个字符。default-network、default、mgnt0、mgnt1四个名称为系统预留，请勿使用。
namespace	是	String	配置资源所在命名空间，固定为kube-system。
config	是	Object	配置内容，为json格式的字符串。 config字段数据结构具体说明，请参见表3-7。

表 3-7 config 字段数据结构说明

参数	是否必填	参数类型	描述
type	是	String	表示网络类型，固定为eni-neutron。
args	是	Object	安全组和子网的配置参数。 args字段数据结构具体说明，请参见表3-8。

表 3-8 args 字段数据结构说明

参数	是否必填	参数类型	描述
securityGroups	否	String	需要配置的安全组ID，最多关联5条安全组。如果没有对安全组进行规划，请和default-network中的安全组保持一致。 获取方式： 登录虚拟私有云控制台，在左侧导航栏选择“访问控制 > 安全组”，单击安全组名称，在“基本信息”页签下找到“ID”字段复制即可。

参数	是否必填	参数类型	描述
subnets	是	Array of subnetID Objects	容器子网ID列表，至少需填写1个，最多可填写100个，具体格式如下： [{"subnetID":"27d95**"},{"subnetID":"827bb**"},{"subnetID":"bdd6b**"}] 容器子网要求与集群在同一VPC下。 获取方式： 登录虚拟私有云控制台，在左侧导航栏选择“虚拟私有云 > 子网”，单击子网名称，在“基本信息”页签下找到“子网ID”字段复制即可。

步骤3 创建NetworkAttachmentDefinition。

```
kubectl create -f networkattachment-test.yaml
```

回显如下，表示NetworkAttachmentDefinition已创建。

```
networkattachmentdefinition.k8s.cni.cncf.io/example created
```

步骤4 创建YAML文件deploy.yaml，用于创建工作负载并关联刚创建的容器网络配置（此处创建Deployment类型的工作负载），文件名称可自定义。

```
vim deploy.yaml
```

文件内容如下：

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
        yangtse.io/network: "example"  # 自定义容器网络配置名称，便于通过label查找所有关联此容器网络配置的Pod
      annotations:
        yangtse.io/network: "example"  # 自定义容器网络配置名称
    spec:
      containers:
        - name: container-0
          image: nginx:alpine
          resources:
            limits:
              cpu: 100m
              memory: 200Mi
            requests:
              cpu: 100m
              memory: 200Mi
      imagePullSecrets:
        - name: default-secret
```

- yangtse.io/network：指定的自定义容器网络配置名称，只能指定未关联命名空间的容器网络配置。请在label上同时添加此参数，便于通过label查找所有关联此容器网络配置的Pod。

须知

yangtse.io/network指定的自定义容器网络配置名称与networkattachment-test.yaml中name值一致。

步骤5 创建Deployment。

```
kubectl create -f deploy.yaml
```

回显如下：

```
deployment.apps/nginx created
```

----结束

验证命名空间/工作负载是否绑定容器网络配置

本节主要介绍如何确认工作负载是否成功绑定容器网络配置中的子网和安全组。如果您需要验证命名空间级别的绑定情况，可以通过该命名空间中的具体工作负载实现，验证步骤与本节一致。

步骤1 验证子网是否绑定成功。

1. 查看工作负载中Pod对应的IP地址。

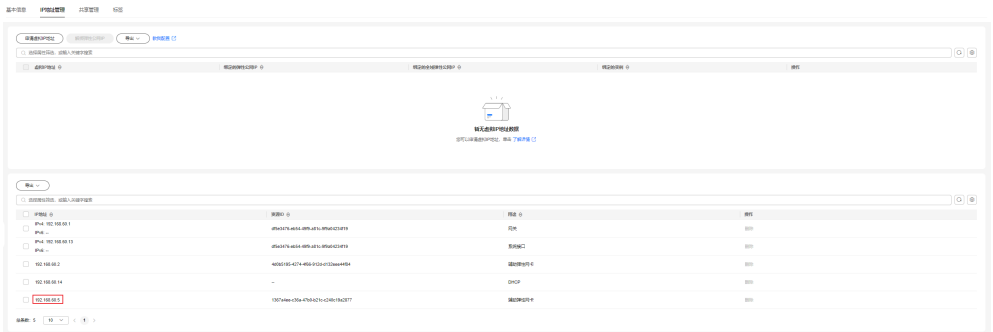
   ```
   kubectl get pod -o wide
   ```

   回显结果如下：

   ```
   NAME                     READY   STATUS    RESTARTS   AGE     IP              NODE
   NOMINATED NODE   READINESS GATES
   nginx-85dbdb8c5d-ng5bb   1/1     Running   0          5d18h   192.168.60.5    ca50a5ae-e1ef-41c6-
   b3fc-6ebcd10a1e07   <none>           <none>
   ```

2. 进入网络控制台，左侧导航栏单击“虚拟私有云 > 子网”，单击对应的子网名称。
3. 单击“IP地址管理”，“IP地址管理”中若有Pod对应的IP地址则说明子网绑定成功。

图 3-14 查看子网绑定的 IP 地址



步骤2 验证安全组是否绑定成功。

1. 返回网络控制台，左侧导航栏单击“访问控制 > 安全组”，单击对应的安全组名称。
2. 单击“关联实例”，当前页签中单击“辅助弹性网卡”。
3. “辅助弹性网卡”页签中，若私有IP地址列表有Pod对应的IP地址，则说明安全组绑定成功。

图 3-15 查看安全组绑定的 IP 地址

----结束

删除网络配置

您可以查看新添加网络配置的YAML，也可以对新添加的配置进行“删除”操作。

在删除网络配置时，需先删除该配置所对应的容器，否则将删除失败。

1. 执行以下命令筛选集群中使用该配置的Pod（其中example为示例配置名称，请自行替换）：
   ```
   kubectl get pod -A -o=jsonpath="{.items[?(@.metadata.annotations.cni\.yangtse\.io/network-status=='[{\"name\":\"example\"}]')]['metadata.namespace', 'metadata.name']}"
   ```
   返回结果中包含了该配置关联的Pod名字及命名空间。
2. 删除创建该Pod的Owner，其Owner可能为Deployment、StatefulSet、Job和CronJob类型的工作负载。
   ```
   kubectl delete deployment nginx
   ```
   回显如下：
   ```
   deployment.apps "nginx" deleted
   ```
3. 删除创建的容器网络配置。
   ```
   kubectl delete -f networkattachment-test.yaml -n kube-system
   ```
   回显如下：
   ```
   networkattachmentdefinition.k8s.cni.cncf.io "example" deleted
   ```

3.3 服务（Service）

3.3.1 集群内访问（ClusterIP）

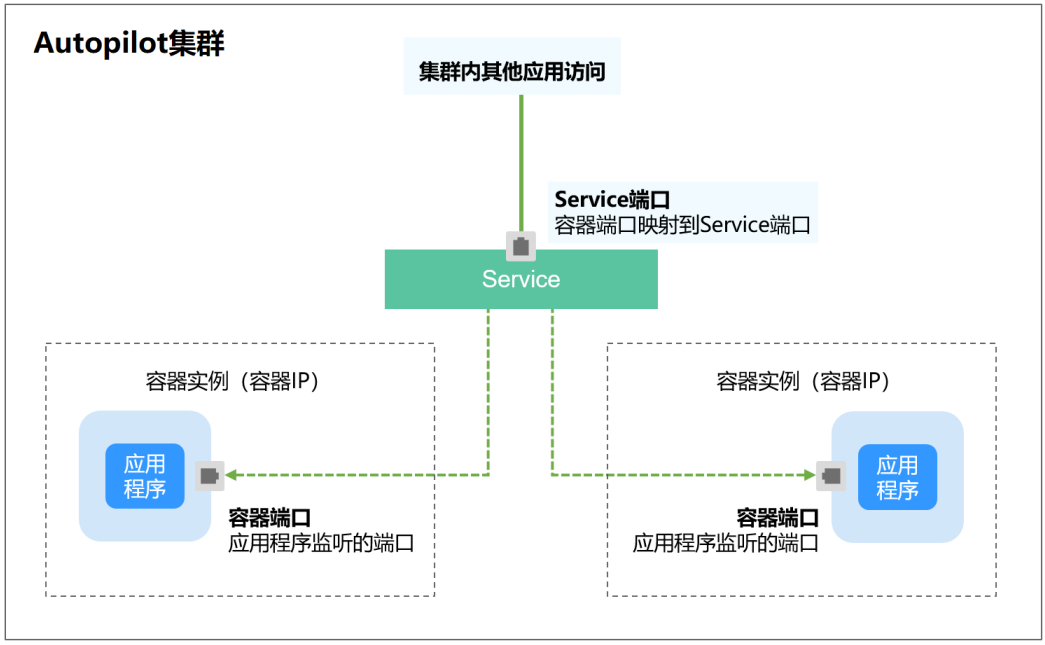
操作场景

集群内访问表示工作负载暴露给同一集群内其他工作负载访问的方式，可以通过“集群内部域名”访问。

集群内部域名格式为“<服务名称>.<工作负载所在命名空间>.svc.cluster.local:<端口号>”，例如“nginx.default.svc.cluster.local:80”。

访问通道、容器端口与访问端口映射如图3-16所示。

图 3-16 集群内访问



创建 ClusterIP 类型 Service

步骤1 登录CCE控制台，单击集群名称进入集群。

步骤2 在左侧导航栏中选择“服务”，在右上角单击“创建服务”。

步骤3 设置集群内访问参数。

- **Service名称：**自定义服务名称，可与工作负载名称保持一致。
- **访问类型**：选择“集群内访问”。
- **命名空间：**工作负载所在命名空间。
- **选择器：**添加标签，Service根据标签选择Pod，填写后单击“确认添加”。也可以引用已有工作负载的标签，单击“引用负载标签”，在弹出的窗口中选择负载，然后单击“确定”。
- **端口配置：**
  - 协议：请根据业务的协议类型选择。
  - 服务端口：Service使用的端口，端口范围为1-65535。
  - 容器端口：工作负载程序实际监听的端口，需用户确定。例如nginx默认使用80端口。

步骤4 单击“确定”，创建Service。

----结束

通过 kubectl 命令行创建

您可以通过kubectl命令行设置Service访问方式。本节以nginx为例，说明kubectl命令实现集群内访问的方法。

步骤1 请参见通过kubectl连接集群，使用kubectl连接集群。

步骤2 创建并编辑nginx-deployment.yaml和nginx-clusterip-svc.yaml文件。

其中，nginx-deployment.yaml和nginx-clusterip-svc.yaml为自定义名称，您可以随意命名。

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              # 访问Service的端口
    protocol: TCP           # 访问Service的协议，支持TCP和UDP
    targetPort: 80          # Service访问目标容器的端口，此端口与容器中运行的应用强相关，如本例中nginx镜像默认使用80端口
  selector:                 # 标签选择器，Service通过标签选择Pod，将访问Service的流量转发给Pod，此处选择带有 app:nginx 标签的Pod
    app: nginx
  type: ClusterIP           # Service的类型，ClusterIP表示在集群内访问
```

步骤3 创建工作负载。

kubectl create -f nginx-deployment.yaml

回显如下，表示工作负载已经创建。

```
deployment "nginx" created
```

kubectl get po

回显如下，工作负载状态为Running，表示工作负载已处于运行中状态。

```
NAME                     READY     STATUS              RESTARTS   AGE
nginx-2601814895-znhbr   1/1       Running             0          15s
```

步骤4 创建服务。

kubectl create -f nginx-clusterip-svc.yaml

回显如下，表示服务已开始创建。

```
service "nginx-clusterip" created
```

kubectl get svc

回显如下，表示服务已创建成功，CLUSTER-IP已生成。

```
# 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
```

步骤5 访问Service。

在集群内的容器或节点上都能够访问Service。

创建一个Pod并进入到容器内，使用curl命令访问Service的IP:Port或域名，如下所示。

其中域名后缀可以省略，在同个命名空间内可以直接使用nginx-clusterip:8080访问，跨命名空间可以使用nginx-clusterip.default:8080访问。

```
# kubectl run -i --tty --image nginx:alpine test --rm /bin/sh
If you don't 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>
...
```

----结束

3.3.2 负载均衡（LoadBalancer）

3.3.2.1 创建负载均衡类型的服务

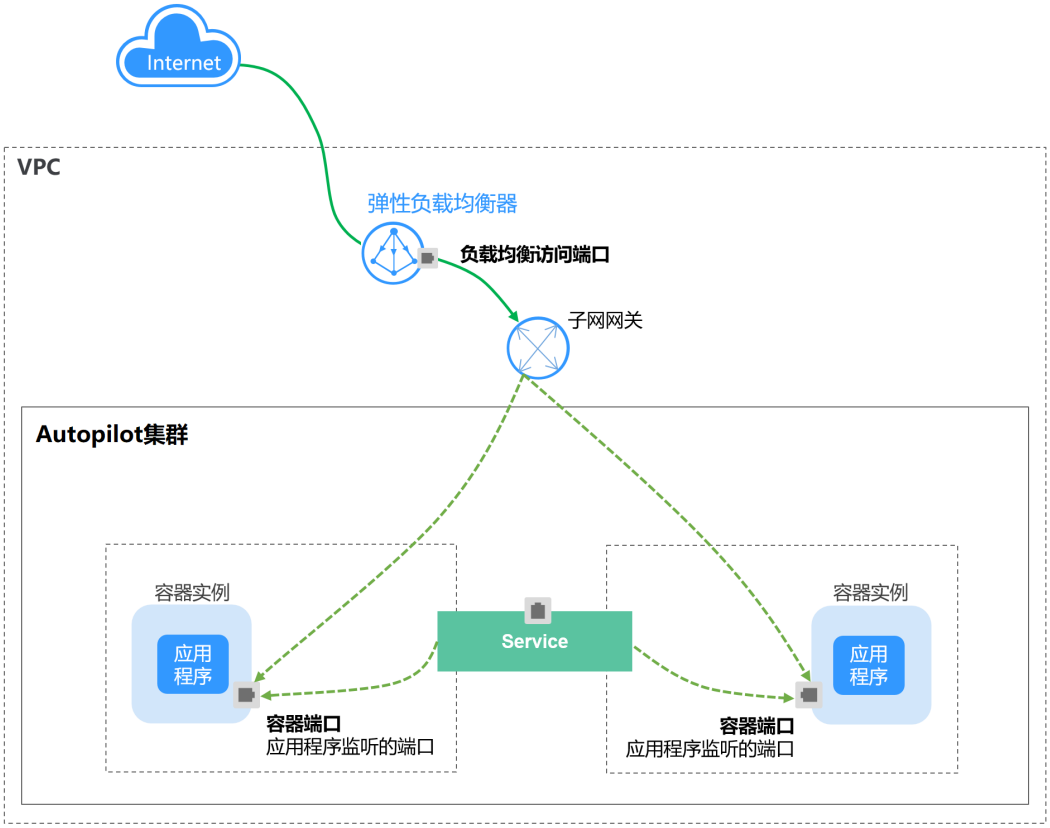
操作场景

负载均衡（LoadBalancer）类型的服务可以通过弹性负载均衡（ELB）从公网访问到工作负载，与弹性IP方式相比提供了高可靠的保障。负载均衡访问方式由公网弹性负载均衡服务地址以及设置的访问端口组成，例如“10.117.117.117:80”。

在使用**CCE Autopilot集群 + 独享型ELB实例**时，支持ELB直通Pod，使部署在容器中的业务时延降低、性能无损耗。

从集群外部访问时，从ELB直接转发到Pod；集群内部访问可通过Service转发到Pod。

图 3-17 ELB 直通容器



约束与限制

- 自动创建的ELB实例建议不要被其他资源使用，否则会在删除时被占用，导致资源残留。
- 独享型ELB网络类型必须支持私网（有私有IP地址）。如果需要Service支持HTTP，则独享型ELB规格需要为网络型（TCP/UDP）和应用型（HTTP/HTTPS）。

创建 LoadBalancer 类型 Service

步骤1 登录CCE控制台，单击集群名称进入集群。

步骤2 在左侧导航栏中选择“服务”，在右上角单击“创建服务”。

步骤3 设置参数。

- **Service名称：**自定义服务名称，可与工作负载名称保持一致。
- **访问类型**：选择“负载均衡 LoadBalancer”。
- **命名空间：**工作负载所在命名空间。
- **服务亲和：**

  集群级别：集群下所有节点的IP+访问端口均可以访问到此服务关联的负载，服务访问会因路由跳转导致一定性能损失，且无法获取到客户端源IP。
- **选择器：**添加标签，Service根据标签选择Pod，填写后单击“确认添加”。也可以引用已有工作负载的标签，单击“引用负载标签”，在弹出的窗口中选择负载，然后单击“确定”。
- **负载均衡器：**选择弹性负载均衡的类型、创建方式。

  ELB类型可选择“独享型”，独享型ELB可以根据支持的协议类型选择“网络型（TCP/UDP）”、“应用型（HTTP/HTTPS）”或“网络型（TCP/UDP）&应用型（HTTP/HTTPS）”。

  创建方式可选择“选择已有”或“自动创建”。不同创建方式的配置详情请参见**表3-9**。

表 3-9 ELB 配置

创建方式	配置
选择已有	仅支持选择与集群在同一个VPC下的ELB实例。如果没有可选的ELB实例，请单击“创建负载均衡器”跳转到ELB控制台创建。
自动创建	– 实例名称：请填写ELB名称。 – 企业项目：该参数仅对开通企业项目的企业客户账号显示。企业项目是一种云资源管理方式，企业项目管理服务提供统一的云资源按项目管理，以及项目内的资源管理、成员管理。 – 可用区：可以选择在多个可用区创建负载均衡实例，提高服务的可用性。如果业务需要考虑容灾能力，建议选择多个可用区。 – 前端子网：用于分配ELB实例对外服务的IP地址。 – 后端子网：用于与后端服务建立连接的IP地址。 – 网络型规格/应用型规格/规格： ■ 弹性规格：适用于业务用量波动较大的场景，按实际使用量收取每小时使用的容量费用。 ■ 固定规格：适用于业务用量较为稳定的场景，按固定规格折算收取每小时使用的容量费用。 – 弹性公网IP：选择“自动创建”时，可配置公网带宽的计费方式及带宽大小。 – 资源标签：通过为资源添加标签，可以对资源进行自定义标记，实现资源的分类。您可以在TMS中创建“预定义标签”，预定义标签对所有支持标签功能的服务资源可见，通过使用预定义标签可以提升标签创建和迁移效率。v1.27.5-r0、v1.28.3-r0及以上版本集群支持。

负载均衡配置：您可以单击负载均衡配置的“编辑”图标配置ELB实例的参数，在弹出窗口中配置ELB实例的参数。

- 分配策略：可选择加权轮询算法、加权最少连接或源IP算法。

  说明

  - 加权轮询算法：根据后端服务器的权重，按顺序依次将请求分发给不同的服务器。它用相应的权重表示服务器的处理性能，按照权重的高低以及轮询方式将请求分配给各服务器，相同权重的服务器处理相同数目的连接数。常用于短连接服务，例如HTTP等服务。
  - 加权最少连接：最少连接是通过当前活跃的连接数来估计服务器负载情况的一种动态调度算法。加权最少连接就是在最少连接数的基础上，根据服务器的不同处理能力，给每个服务器分配不同的权重，使其能够接受相应权值数的服务请求。常用于长连接服务，例如数据库连接等服务。
  - 源IP算法：将请求的源IP地址进行Hash运算，得到一个具体的数值，同时对后端服务器进行编号，按照运算结果将请求分发到对应编号的服务器上。这可以使得对不同源IP的访问进行负载分发，同时使得同一个客户端IP的请求始终被派发至某特定的服务器。该方式适合负载均衡无cookie功能的TCP协议。

- 会话保持类型：默认不启用。

  启用后根据监听器支持的协议类型自动选择会话保持类型，具体如下：

  - 前端协议为TCP/UDP的监听器，会话保持类型支持源IP地址。基于源IP地址的简单会话保持，即来自同一IP地址的访问请求转发到同一台后端服务器（Pod实例）上。
  - 前端协议为HTTP/HTTPS的监听器，会话保持类型支持负载均衡器cookie。

  说明

  当**分配策略**使用源IP算法时，不支持设置会话保持。

- **健康检查**：设置负载均衡的健康检查配置。
  - 全局检查：全局检查仅支持使用相同协议的端口，无法对多个使用不同协议的端口生效，建议使用“自定义检查”。
  - 自定义检查：在**端口配置**中对多种不同协议的端口设置健康检查。

表 3-10 健康检查参数

参数	说明
协议	当**端口配置**协议为TCP时，支持TCP和HTTP协议；当**端口配置**协议为UDP时，支持UDP协议。 检查路径（仅HTTP健康检查协议支持）：指定健康检查的URL地址。检查路径只能以/开头，长度范围为1-80。

参数	说明
端口	健康检查默认使用业务端口作为健康检查的端口；您也可以重新指定端口用于健康检查，重新指定端口会为服务增加一个名为cce-healthz的服务端口配置。 容器端口：使用独享型负载均衡关联ENI实例时，容器端口作为健康检查的检查端口。取值范围为1-65535。
检查周期（秒）	每次健康检查响应的最大间隔时间，取值范围为1-50。
超时时间（秒）	每次健康检查响应的最大超时时间，取值范围为1-50。
最大重试次数	健康检查最大的重试次数，取值范围为1-10。

- **端口配置：**
  - 协议：请根据业务的协议类型选择。
  - 服务端口：Service使用的端口，端口范围为1-65535。
  - 容器端口：工作负载程序实际监听的端口，需用户确定。例如nginx默认使用80端口。
  - 监听器前端协议：ELB监听器的前端协议，是客户端与负载均衡监听器建立流量分发连接所使用的协议。当选择独享型负载均衡器类型时，包含“应用型（HTTP/HTTPS）”方可支持配置HTTP/HTTPS。
  - 健康检查：**健康检查**选项设置为“自定义检查”时，可以为不同协议的端口配置健康检查，参数说明请参见**表3-10**。

    说明

    在创建LoadBalancer类型Service时，会自动生成一个随机节点端口号（NodePort）。
- **注解**：LoadBalancer类型Service有一些CCE定制的高级功能，通过注解annotations实现，具体注解的内容请参见**使用Annotation配置负载均衡**。

步骤4 单击“确定”，创建Service。

----结束

通过 kubectl 命令行创建-使用已有 ELB

您可以在创建工作负载时通过kubectl命令行设置Service访问方式。本节以nginx为例，说明kubectl命令实现负载均衡（LoadBalancer）访问的方法。

步骤1 请参见**通过kubectl连接集群**，使用kubectl连接集群。

步骤2 创建并编辑nginx-deployment.yaml以及nginx-elb-svc.yaml文件。

其中，nginx-deployment.yaml和nginx-elb-svc.yaml为自定义名称，您可以随意命名。

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

```
apiVersion: v1
kind: Service
metadata:
  name: nginx
  annotations:
    kubernetes.io/elb.id: <your_elb_id>                    # ELB ID，替换为实际值
    kubernetes.io/elb.class: performance                   # 负载均衡器类型
    kubernetes.io/elb.lb-algorithm: ROUND_ROBIN            # 负载均衡器算法
    kubernetes.io/elb.session-affinity-mode: SOURCE_IP     # 会话保持类型为源IP
    kubernetes.io/elb.session-affinity-option: '{"persistence_timeout": "30"}'    # 会话保持时间（分钟）
    kubernetes.io/elb.health-check-flag: 'on'              # 开启ELB健康检查功能
    kubernetes.io/elb.health-check-option: '{
      "protocol":"TCP",
      "delay":"5",
      "timeout":"10",
      "max_retries":"3"
    }'
spec:
  selector:
    app: nginx
  ports:
  - name: service0
    port: 80     #访问Service的端口，也是负载均衡上的监听器端口。
    protocol: TCP
    targetPort: 80  #Service访问目标容器的端口，此端口与容器中运行的应用强相关
    nodePort: 31128  #节点的端口号，如不指定，将在30000-32767范围内随机生成一个端口号
  type: LoadBalancer
```

上述示例通过Annotation（注解）实现负载均衡的一些高级功能，例如会话保持、健康检查等，对应的说明请参见**表3-11**。

除本示例中的功能外，如需了解更多高级功能相关注解及示例，请参见**使用Annotation配置负载均衡**。

表 3-11 annotations 参数

参数	是否必填	参数类型	描述
kubernetes.io/elb.id	是	String	为负载均衡实例的ID。 在关联已有ELB时：必填。 **获取方法：** 在控制台的“服务列表”中，单击“网络 > 弹性负载均衡 ELB”，单击ELB的名称，在ELB详情页的“基本信息”页签下找到“ID”字段复制即可。
kubernetes.io/elb.class	是	String	取值如下： ● performance：独享型负载均衡。 **说明** 负载均衡类型的服务对接已有的独享型ELB时，网络类型必须支持私网（有私有IP地址）。
kubernetes.io/elb.lb-algorithm	否	String	后端云服务器组的负载均衡算法，默认值为“ROUND_ROBIN”。 取值范围： ● ROUND_ROBIN：加权轮询算法。 ● LEAST_CONNECTIONS：加权最少连接算法。 ● SOURCE_IP：源IP算法。 **说明** 当该字段的取值为SOURCE_IP时，后端云服务器组绑定的后端云服务器的权重设置（weight字段）无效，且不支持开启会话保持。
kubernetes.io/elb.session-affinity-mode	否	String	支持基于源IP地址的简单会话保持，即来自同一IP地址的访问请求转发到同一台后端服务器上。 ● 不启用：不填写该参数。 ● 开启会话保持：需增加该参数，取值“SOURCE_IP”，表示基于源IP地址。 **说明** 当kubernetes.io/elb.lb-algorithm设置为“SOURCE_IP”（源IP算法）时，不支持开启会话保持。
kubernetes.io/elb.session-affinity-option	否	表3-12 Object	ELB会话保持配置选项，可设置会话保持的超时时间。

参数	是否必填	参数类型	描述
kubernetes.io/elb.health-check-flag	否	String	是否开启ELB健康检查功能。 • 开启：空值或"on" • 关闭："off" 开启时需同时填写**kubernetes.io/elb.health-check-option**字段。
kubernetes.io/elb.health-check-option	否	**表3-13** Object	ELB健康检查配置选项。

表 3-12 elb.session-affinity-option 字段数据结构说明

参数	是否必填	参数类型	描述
persistence_timeout	是	String	当elb.session-affinity-mode是“SOURCE_IP”时生效，设置会话保持的超时时间（分钟）。 默认值为："60"，取值范围：1-60。

表 3-13 elb.health-check-option 字段数据结构说明

参数	是否必填	参数类型	描述
delay	否	String	健康检查间隔（秒）。 默认值：5，取值范围：1-50
timeout	否	String	健康检查的超时时间（秒）。 默认值：10，取值范围1-50
max_retries	否	String	健康检查的最大重试次数。 默认值：3，取值范围1-10
protocol	否	String	健康检查的协议。 取值范围：“TCP”或者“HTTP”
path	否	String	健康检查的URL，协议是“HTTP”时配置。 默认值：“/” 取值范围：1-80字符

步骤3 创建工作负载。

kubectl create -f nginx-deployment.yaml

回显如下，表示工作负载已创建完成。

```
deployment/nginx created
```

kubectl get pod

回显如下，工作负载状态为Running状态，表示工作负载已运行中。

```
NAME                READY   STATUS          RESTARTS   AGE
nginx-2601814895-c1xhw   1/1     Running         0          6s
```

步骤4 创建服务。

kubectl create -f nginx-elb-svc.yaml

回显如下，表示服务已创建。

```
service/nginx created
```

kubectl get svc

回显如下，表示工作负载访问方式已设置成功，工作负载可访问。

```
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
```

步骤5 在浏览器中输入访问地址，例如输入10.78.42.242:80。10.78.42.242为负载均衡实例IP地址，80为对应界面上的访问端口。

可成功访问nginx。

图 3-18 通过负载均衡访问 nginx

----结束

通过 kubectl 命令行创建-自动创建 ELB

您可以在创建工作负载时通过kubectl命令行设置Service访问方式。本节以nginx为例，说明kubectl命令实现负载均衡（ LoadBalancer)访问的方法。

步骤1 请参见**通过kubectl连接集群**，使用kubectl连接集群。

步骤2 创建并编辑nginx-deployment.yaml以及nginx-elb-svc.yaml文件。

其中，nginx-deployment.yaml和nginx-elb-svc.yaml为自定义名称，您可以随意命名。

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

独享型负载均衡（公网访问）Service示例：

```
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",
      "vip_subnet_cidr_id": "******",
      "vip_address": "**.**.**.**",
      "elb_virsubnet_ids": ["******"],
      "available_zone": [
        "cn-north-4b"
      ],
      "l4_flavor_name": "L4_flavor.elb.s1.small"
    }'
    kubernetes.io/elb.enterpriseID: '0'       # 负载均衡所属企业项目ID
    kubernetes.io/elb.lb-algorithm: ROUND_ROBIN                 # 负载均衡器算法
    kubernetes.io/elb.session-affinity-mode: SOURCE_IP          # 会话保持类型为源IP
    kubernetes.io/elb.session-affinity-option: '{"persistence_timeout": "30"}'     # 会话保持时间（分钟）
    kubernetes.io/elb.health-check-flag: 'on'                   # 开启ELB健康检查功能
    kubernetes.io/elb.health-check-option: '{
      "protocol":"TCP",
      "delay":"5",
      "timeout":"10",
      "max_retries":"3"
    }'
    kubernetes.io/elb.tags: key1=value1,key2=value2           # 添加ELB资源标签
spec:
  selector:
    app: nginx
  ports:
  - name: cce-service-0
    targetPort: 80
    nodePort: 0
    port: 80
    protocol: TCP
  type: LoadBalancer
```

上述示例通过Annotation（注解）实现负载均衡的一些高级功能，例如会话保持、健康检查等，对应的说明请参见**表3-14**。

除本示例中的功能外，如需了解更多高级功能相关注解及示例，请参见**使用Annotation配置负载均衡**。

表 3-14 annotations 参数

参数	是否必填	参数类型	描述
kubernetes.io/elb.class	是	String	请根据不同的应用场景和功能需求选择合适的负载均衡器类型。 取值如下： • performance：独享型负载均衡。
kubernetes.io/elb.autocreate	是	**elb.autocreate** object	自动创建service关联的ELB。 **说明** 公网自动创建：负载均衡器购买弹性公网IP，允许公网和私网访问。 私网自动创建：负载均衡器不购买弹性公网IP，只允许私网访问。 **示例：** • 公网自动创建： 值为 '{"type":"public","bandwidth_name":"cce-bandwidth-1551163379627","bandwidth_chargemode":"bandwidth","bandwidth_size":5,"bandwidth_sharetype":"PER","eip_type":"5_bgp","name":"james","available_zone":["cn-east-3a"],"l4_flavor_name":"L4_flavor.elb.s1.small"}'。 • 私网自动创建： 值为 '{"type":"inner","available_zone":["cn-east-3a"],"l4_flavor_name":"L4_flavor.elb.s1.small"}'。
kubernetes.io/elb.subnet-id	-	String	为集群所在子网的ID，取值范围：1-100字符。 • Kubernetes v1.11.7-r0及以下版本的集群自动创建时为必填参数。 • Kubernetes v1.11.7-r0以上版本的集群：可不填。 获取方法请参见：**VPC子网接口与OpenStack Neutron子网接口的区别是什么?**

参数	是否必填	参数类型	描述
kubernetes.io/elb.enterpriseID	否	String	为ELB企业项目ID，选择后可以直接创建在具体的ELB企业项目下。 该字段不传（或传为字符串'0'），则将资源绑定给默认企业项目。 **获取方法：** 登录控制台后，单击顶部菜单右侧的“企业 > 项目管理”，在打开的企业项目列表中单击要加入的企业项目名称，进入企业项目详情页，找到“ID”字段复制即可。
kubernetes.io/elb.lb-algorithm	否	String	后端云服务器组的负载均衡算法，默认值为“ROUND_ROBIN”。 取值范围： • ROUND_ROBIN：加权轮询算法。 • LEAST_CONNECTIONS：加权最少连接算法。 • SOURCE_IP：源IP算法。 **说明** 当该字段的取值为SOURCE_IP时，后端云服务器组绑定的后端云服务器的权重设置（weight字段）无效，且不支持开启会话保持。
kubernetes.io/elb.session-affinity-mode	否	String	支持基于源IP地址的简单会话保持，即来自同一IP地址的访问请求转发到同一台后端服务器上。 • 不启用：不填写该参数。 • 开启会话保持：需增加该参数，取值“SOURCE_IP”，表示基于源IP地址。 **说明** 当kubernetes.io/elb.lb-algorithm设置为“SOURCE_IP”（源IP算法）时，不支持开启会话保持。
kubernetes.io/elb.session-affinity-option	否	表3-12 Object	ELB会话保持配置选项，可设置会话保持的超时时间。
kubernetes.io/elb.health-check-flag	否	String	是否开启ELB健康检查功能。 • 开启：“（空值）”或“on” • 关闭：“off” 开启时需同时填写**kubernetes.io/elb.health-check-option**字段。

参数	是否必填	参数类型	描述
kubernetes.io/elb.health-check-option	否	表3-13 Object	ELB健康检查配置选项。
kubernetes.io/elb.tags	否	String	为ELB添加资源标签，仅自动创建ELB时支持设置。 格式为key=value，同时添加多个标签时以英文逗号（,）隔开。

表 3-15 elb.autocreate 字段数据结构说明

参数	是否必填	参数类型	描述
name	否	String	自动创建的负载均衡的名称。 取值范围：只能由中文、英文字母、数字、下划线、中划线、点组成，且长度范围为1-64个字符。 默认名称：cce-lb+service.UID
type	否	String	负载均衡实例网络类型，公网或者私网。 ● public：公网型负载均衡 ● inner：私网型负载均衡 默认类型：inner
bandwidth_name	公网型负载均衡必填	String	带宽的名称，默认值为：cce-bandwidth-******。 取值范围：只能由中文、英文字母、数字、下划线、中划线、点组成，且长度范围为1-64个字符。
bandwidth_chargemode	否	String	带宽付费模式。 ● bandwidth：按带宽 ● traffic：按流量 默认类型：bandwidth

参数	是否必填	参数类型	描述
bandwidth_size	公网型负载均衡必填	Integer	带宽大小，默认1Mbit/s~2000Mbit/s，请根据Region带宽支持范围设置。 调整带宽时的最小单位会根据带宽范围不同存在差异。 ● 小于等于300Mbit/s：默认最小单位为1Mbit/s。 ● 300Mbit/s~1000Mbit/s：默认最小单位为50Mbit/s。 ● 大于1000Mbit/s：默认最小单位为500Mbit/s。
bandwidth_sharetype	公网型负载均衡必填	String	带宽共享方式。 ● PER：独享带宽
eip_type	公网型负载均衡必填	String	弹性公网IP类型。 ● 5_telcom：电信 ● 5_union：联通 ● 5_bgp：全动态BGP ● 5_sbgp：静态BGP
vip_subnet_cidr_id	否	String	指定ELB所在的子网，该子网必须属于集群所在的VPC。 如不指定，则ELB与集群在同一个子网。
vip_address	否	String	负载均衡器的内网IP。仅支持指定IPv4地址，不支持指定IPv6地址。 该IP必须为ELB所在子网网段中的IP。若不指定，自动从ELB所在子网网段中生成一个IP地址。
available_zone	是	Array of strings	负载均衡所在可用区。 可以通过**查询可用区列表**获取所有支持的可用区。 独享型负载均衡器独有字段。

参数	是否必填	参数类型	描述
l4_flavor_name	是	String	四层负载均衡实例规格名称。 可以通过**查询规格列表**获取所有支持的类型。 • 弹性规格：适用于业务用量波动较大的场景，按实际使用量收取每小时使用的容量费用。 • 固定规格：适用于业务用量较为稳定的场景，按固定规格折算收取每小时使用的容量费用。 独享型负载均衡器独有字段。
l7_flavor_name	否	String	七层负载均衡实例规格名称。 可以通过**查询规格列表**获取所有支持的类型。 • 弹性规格：适用于业务用量波动较大的场景，按实际使用量收取每小时使用的容量费用。 • 固定规格：适用于业务用量较为稳定的场景，按固定规格折算收取每小时使用的容量费用。 独享型负载均衡器独有字段，必须与l4_flavor_name对应规格的类型一致，即都为弹性规格或都为固定规格。
elb_virsubnet_ids	否	Array of strings	负载均衡后端所在子网，不填默认为集群子网。不同实例规格将占用不同数量子网IP，不建议使用其他资源（如集群）的子网网段。 独享型负载均衡器独有字段。 示例： `"elb_virsubnet_ids": [ "14567f27-8ae4-42b8-ae47-9f847a4690dd" ]`

步骤3 创建工作负载。

kubectl create -f nginx-deployment.yaml

回显如下，表示工作负载已开始创建。

```
deployment/nginx created
```

kubectl get pod

回显如下，工作负载状态为Running状态，表示工作负载已运行中。

```
NAME                     READY   STATUS    RESTARTS   AGE
nginx-2601814895-c1xhw   1/1     Running   0          6s
```

步骤4 创建服务。

kubectl create -f nginx-elb-svc.yaml

回显如下，表示服务已创建。

```
service/nginx created
```

kubectl get svc

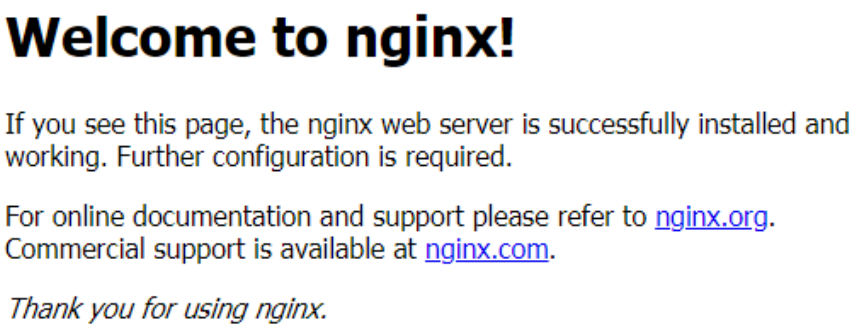
回显如下，表示工作负载访问方式已设置成功，工作负载可访问。

```
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
```

步骤5 在浏览器中输入访问地址，例如输入10.78.42.242:80。10.78.42.242为负载均衡实例IP地址，80为对应界面上的访问端口。

可成功访问nginx。

图 3-19 通过负载均衡访问 nginx



----结束

3.3.2.2 健康检查使用 UDP 协议的安全组规则说明

操作场景

Autopilot集群使用独享型ELB，当负载均衡协议为UDP时，健康检查也采用UDP协议，您需要打开ENI安全组的ICMP协议安全组规则，入方向规则放通ELB后端子网网段的源地址。

操作步骤

步骤1 登录CCE控制台，单击服务列表中的“网络 > 虚拟私有云 VPC”，在网络控制台单击“访问控制 > 安全组”。

步骤2 在界面右侧的安全组列表中找到集群的ENI安全组，名称规则默认是**{集群名}-cce-eni-{随机ID}**。

如果集群中绑定了自定义的容器安全组，请根据实际进行选择。

步骤3 单击安全组名称，在打开的页面中单击“入方向规则”页签，单击“添加规则”，添加入方向规则，详细配置请参见**图3-20**。

- 协议端口：选择ICMP的全部端口。
- 源地址：填写ELB后端子网网段。

图 3-20 添加安全组规则

步骤4 单击“确定”。

----结束

3.3.3 Headless Service

前面讲的Service解决了Pod的内外部访问问题，但还有下面这些问题没解决。

- 同时访问所有Pod
- 一个Service内部的Pod互相访问

Headless Service正是解决这个问题的，Headless Service不会创建ClusterIP，并且查询会返回所有Pod的DNS记录，这样就可查询到所有Pod的IP地址。有状态负载StatefulSet正是使用Headless Service解决Pod间互相访问的问题。

```
apiVersion: v1
kind: Service       # 对象类型为Service
metadata:
  name: nginx-headless
  labels:
    app: nginx
spec:
  ports:
    - name: nginx     # Pod间通信的端口名称
      port: 80        # Pod间通信的端口号
  selector:
    app: nginx        # 选择标签为app:nginx的Pod
  clusterIP: None     # 必须设置为None，表示Headless Service
```

执行如下命令创建Headless Service。

```
# kubectl create -f headless.yaml
service/nginx-headless created
```

创建完成后可以查询Service。

```
# kubectl get svc
NAME             TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
nginx-headless   ClusterIP   None         <none>        80/TCP    5s
```

创建一个Pod来查询DNS，可以看到能返回所有Pod的记录，这就解决了访问所有Pod的问题了。

```
$ kubectl run -i --tty --image tutum/dnsutils dnsutils --restart=Never --rm /bin/sh
If you don't 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
```

3.4 路由（Ingress）

3.4.1 ELB Ingress 管理

3.4.1.1 通过控制台创建 ELB Ingress

前提条件

- Ingress为后端工作负载提供网络访问，因此集群中需提前部署可用的工作负载。若您无可用工作负载，可参考**创建工作负载**部署工作负载。
- 为上述工作负载配置Service。

注意事项

- 建议其他资源不要使用Ingress自动创建的ELB实例，否则在删除Ingress时，ELB实例会被占用，导致资源残留。
- 添加Ingress后请在CCE页面对所选ELB实例进行配置升级和维护，不可在ELB页面对配置进行更改，否则可能导致Ingress服务异常。
- Ingress转发策略中注册的URL需与后端应用提供访问的URL一致，否则将返回404错误。
- 独享型ELB规格必须支持应用型（HTTP/HTTPS），且网络类型必须支持私网（有私有IP地址）。
- 同集群使用多个Ingress对接同一个ELB端口时，监听器的配置项（例如监听器关联的证书、监听器HTTP2属性等）均以第一个Ingress配置为准。

添加 ELB Ingress

本节以nginx作为工作负载并添加ELB Ingress为例进行说明。

步骤1 登录CCE控制台，单击集群名称进入集群。

步骤2 选择左侧导航栏的“服务”，在右侧选择“路由”页签，单击右上角“创建路由”。

步骤3 设置Ingress参数。

- **名称：**自定义Ingress名称，例如ingress-demo。
- **负载均衡器：**选择弹性负载均衡的类型、创建方式。

  ELB类型可选择“独享型”。独享型ELB规格需要支持应用型（HTTP/HTTPS），且网络类型必须支持私网。

创建方式可选择“选择已有”或“自动创建”。不同创建方式的配置详情请参见表3-16。

表 3-16 ELB 配置

创建方式	配置
选择已有	仅支持选择与集群在同一个VPC下的ELB实例。如果没有可选的ELB实例，请单击“创建负载均衡器”跳转到ELB控制台创建。
自动创建	– 实例名称：请填写ELB名称。 – 企业项目：该参数仅对开通企业项目的企业客户账号显示。企业项目是一种云资源管理方式，企业项目管理服务提供统一的云资源按项目管理，以及项目内的资源管理、成员管理。 – 可用区：可以选择在多个可用区创建负载均衡实例，提高服务的可用性。如果业务需要考虑容灾能力，建议选择多个可用区。 – 前端子网：用于分配ELB实例对外服务的IP地址。 – 后端子网：用于与后端服务建立连接的IP地址。 – 网络型规格/应用型规格/规格： ▪ 弹性规格：适用于业务用量波动较大的场景，按实际使用量收取每小时使用的容量费用。 ▪ 固定规格：适用于业务用量较为稳定的场景，按固定规格折算收取每小时使用的容量费用。 – 弹性公网IP：选择“自动创建”时，可配置公网带宽的计费方式及带宽大小。 – 资源标签：通过为资源添加标签，可以对资源进行自定义标记，实现资源的分类。您可以在TMS中创建“预定义标签”，预定义标签对所有支持标签功能的服务资源可见，通过使用预定义标签可以提升标签创建和迁移效率。v1.27.5-r0、v1.28.3-r0及以上版本集群支持。

- **监听器配置：**Ingress为负载均衡器配置监听器，监听器对负载均衡器上的请求进行监听，并分发流量。配置完成后ELB实例侧将会创建对应的监听器，名称默认为k8s_<协议类型>_<端口号>，例如“k8s_HTTP_80”。
  - 前端协议：支持HTTP和HTTPS。
  - 对外端口：开放在负载均衡服务地址的端口，可任意指定。
  - 访问控制：
    - 允许所有IP访问：不设置访问控制。
    - 白名单：仅所选IP地址组可以访问ELB地址。
    - 黑名单：所选IP地址组无法访问ELB地址。
  - 证书来源：支持TLS密钥和ELB服务器证书。
  - 服务器证书：负载均衡器创建HTTPS协议监听时需要绑定证书，以支持HTTPS数据传输加密认证。

- TLS密钥：创建密钥证书的方法请参见**创建密钥**。
- ELB服务器证书：使用在ELB服务中创建的证书。

说明

同一个ELB实例的同一个端口配置HTTPS时，一个监听器只支持配置一个密钥证书。若使用两个不同的密钥证书将两个Ingress添加到同一个ELB下的同一个监听器，ELB侧实际只生效最先添加的证书。

- SNI：SNI（Server Name Indication）是TLS的扩展协议，在该协议下允许同一个IP地址和端口号下对外提供多个基于TLS的访问域名，且不同的域名可以使用不同的安全证书。开启SNI后，允许客户端在发起TLS握手请求时就提交请求的域名信息。负载均衡收到TLS请求后，会根据请求的域名去查找证书：若找到域名对应的证书，则返回该证书认证鉴权；否则，返回缺省证书（服务器证书）认证鉴权。

说明

- 当选择HTTPS协议时，才支持配置“SNI”选项。
- 用于SNI的证书需要指定域名，每个证书只能指定一个域名。支持泛域名证书。

- 后端协议：

  当**监听器配置**为HTTP协议时，仅可选择“HTTP”。

  当**监听器配置**为HTTPS协议时，可选择“HTTP”、“HTTPS”或“GRPC”。选择GRPC协议时需开启HTTP/2，系统将自动为您添加**kubernetes.io/elb.http2-enable:true**注解。

- 高级配置：

配置	说明
获取监听器端口号	开启后可以将ELB实例的监听端口从报文的HTTP头中带到后端云服务器。
获取客户端请求端口号	开启后可以将客户端的源端口从报文的HTTP头中带到后端云服务器。
重写X-Forwarded-Host	开启后将以客户端请求头的Host重写X-Forwarded-Host传递到后端云服务器。
空闲超时时间	客户端连接空闲超时时间。在超过空闲超时时间一直没有请求， 负载均衡会暂时中断当前连接，直到下一次请求时重新建立新的连接。
请求超时时间	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，超出请求超时时间将会断开连接。

配置	说明
响应超时时间	等待后端服务器响应超时时间。请求转发后端服务器后，在等待超过响应超时时间没有响应，负载均衡将终止等待，并返回 HTTP504错误码。
开启HTTP2	客户端与LB之间的HTTPS请求的HTTP2功能的开启状态。 开启后，可提升客户端与LB间的访问性能，但LB与后端服务器间仍采用HTTP1.X协议。

- **转发策略配置：**请求的访问地址与转发规则匹配时（转发规则由域名、URL组成，例如：10.117.117.117:80/helloworld），此请求将被转发到对应的目标Service处理。单击 + 按钮可添加多条转发策略。
  - 域名：实际访问的域名地址。请确保所填写的域名已注册并备案，一旦配置了域名规则后，必须使用域名访问。
  - URL匹配规则：
    - 前缀匹配：例如映射URL为/healthz，只要符合此前缀的URL均可访问。例如/healthz/v1，/healthz/v2。
    - 精确匹配：表示只有URL完全匹配时，访问才能生效。例如映射URL为/healthz，则必须为此URL才能访问。
    - 正则匹配：按正则表达式方式匹配URL。例如正则表达式为**/[A-Za-z0-9_.-]+/test**。只要符合此规则的URL均可访问，例如/abcA9/test，/v1-Ab/test。正则匹配规则支持POSIX与Perl两种标准。
  - URL：需要注册的访问路径，例如：/healthz。

    说明

    此处添加的访问路径要求后端应用内存在相同的路径，否则转发无法生效。

    例如，Nginx应用默认的Web访问路径为“/usr/share/nginx/html”，在为Ingress转发策略添加“/test”路径时，需要应用的Web访问路径下也包含相同路径，即“/usr/share/nginx/html/test”，否则将返回404。
  - 目标服务名称：请选择已有Service或新建Service。页面列表中的查询结果已自动过滤不符合要求的Service。
  - 目标服务访问端口：可选择目标Service的访问端口。
  - 负载均衡配置：
    - 分配策略：可选择加权轮询算法、加权最少连接或源IP算法。

说明

- 加权轮询算法：根据后端服务器的权重，按顺序依次将请求分发给不同的服务器。它用相应的权重表示服务器的处理性能，按照权重的高低以及轮询方式将请求分配给各服务器，相同权重的服务器处理相同数目的连接数。常用于短连接服务，例如HTTP等服务。
- 加权最少连接：最少连接是通过当前活跃的连接数来估计服务器负载情况的一种动态调度算法。加权最少连接就是在最少连接数的基础上，根据服务器的不同处理能力，给每个服务器分配不同的权重，使其能够接受相应权值数的服务请求。常用于长连接服务，例如数据库连接等服务。
- 源IP算法：将请求的源IP地址进行Hash运算，得到一个具体的数值，同时对后端服务器进行编号，按照运算结果将请求分发到对应编号的服务器上。这可以使得对不同源IP的访问进行负载分发，同时使得同一个客户端IP的请求始终被派发至某特定的服务器。该方式适合负载均衡无cookie功能的TCP协议。

- 会话保持类型：默认不启用。支持以下类型：
  - 负载均衡器cookie：同时需填写“会话保持时间”，范围为1-1440分钟。

    说明

    当**分配策略**使用源IP算法时，不支持设置会话保持。

- 健康检查：设置负载均衡的健康检查配置，启用时支持以下配置。

参数	说明
协议	当目标服务端口配置协议为TCP时，支持TCP/HTTP/GRPC协议。 ○ 检查路径（仅HTTP/GRPC健康检查协议支持）：指定健康检查的URL地址。检查路径只能以/开头，长度范围为1-80。
端口	健康检查默认使用业务端口（Service的NodePort或容器端口）作为健康检查的端口；您也可以重新指定端口用于健康检查，重新指定端口会为服务增加一个名为cce-healthz的服务端口配置。 ○ 节点端口：如不指定将随机一个端口。取值范围为30000-32767。 ○ 容器端口：使用独享型负载均衡关联ENI实例时，容器端口作为健康检查的检查端口。取值范围为1-65535。
检查周期（秒）	每次健康检查响应的最大间隔时间，取值范围为1-50。
超时时间（秒）	每次健康检查响应的最大超时时间，取值范围为1-50。
最大重试次数	健康检查最大的重试次数，取值范围为1-10。

– 操作：可单击“删除”按钮删除该配置。

- **注解**：Ingress有一些CCE定制的高级功能，通过注解annotations实现。在使用kubectl创建时，会用到注解，具体请参见**添加Ingress-自动创建ELB**和**添加Ingress-对接已有ELB**。

步骤4 配置完成后，单击“确定”。创建完成后，在Ingress列表可查看到已添加的Ingress。

在ELB控制台可查看通过CCE自动创建的ELB，名称默认为“cce-lb-ingress.UID”。单击ELB名称进入详情页，在“监听器”页签下即可查看Ingress对应的路由设置，包括URL、监听器端口以及对应的后端服务器组端口。

须知

Ingress创建后请在CCE页面对所选ELB实例进行配置升级和维护，不要在ELB控制台对ELB实例进行维护，否则可能导致Ingress服务异常。

图 3-21 ELB 路由设置

步骤5 访问工作负载（例如名称为defaultbackend）的“/healthz”接口。

1. 获取工作负载“/healthz”接口的访问地址。访问地址由负载均衡实例IP、对外端口、映射URL组成，例如：10.**.**.**:80/healthz。
2. 在浏览器中输入“/healthz”接口的访问地址，如：http://10.**.**.**:80/healthz，即可成功访问工作负载，如**图3-22**。

图 3-22 访问 defaultbackend“/healthz”接口

----结束

相关操作

由于社区Ingress结构体中没有property属性，用户使用client-go调用创建ingress的api接口时，创建的Ingress中没有property属性。为了与社区的client-go兼容，CCE提供了相关解决方案，具体请参见**Ingress中的property字段如何实现与社区client-go兼容?** 。

3.4.1.2 通过 Kubectl 命令行创建 ELB Ingress

操作场景

本节以Nginx工作负载为例，说明通过kubectl命令添加ELB Ingress的方法。

- 如您在同一VPC下没有可用的ELB，CCE支持在添加Ingress时自动创建ELB，请参考**添加Ingress-自动创建ELB**。
- 如您已在同一VPC下提前创建了一个可用的ELB，则可参考**添加Ingress-对接已有ELB**。

前提条件

- Ingress为后端工作负载提供网络访问，因此集群中需提前部署可用的工作负载。若您无可用工作负载，可参考**创建工作负载**部署示例nginx工作负载。
- 为上述工作负载配置Service。
- 独享型ELB规格必须支持应用型（HTTP/HTTPS），且网络类型必须支持私网（有私有IP地址）。

添加 Ingress-自动创建 ELB

下面介绍如何通过kubectl命令在添加Ingress时自动创建ELB。

步骤1 请参见**通过kubectl连接集群**，使用kubectl连接集群。

步骤2 创建名为“**ingress-test.yaml**”的YAML文件，此处文件名可自定义。

vi ingress-test.yaml

独享型负载均衡（公网访问）示例：

```
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",
          "elb_virsubnet_ids":["******"],
          "available_zone": [
              "cn-north-4b"
          ],
          "l7_flavor_name": "L7_flavor.elb.s1.small"
      }'
spec:
  rules:
  - host: ''
    http:
      paths:
      - path: '/'
        backend:
          service:
            name: <your_service_name>  #替换为您的目标服务名称
```



```
        port:
          number: <your_service_port>  #替换为您的目标服务端口
      property:
        ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
      pathType: ImplementationSpecific
ingressClassName: cce
```

表 3-17 关键参数说明

参数	是否必填	参数类型	描述
kubernetes.io/elb.class	是	String	取值如下： • performance：独享型负载均衡。
ingressClassName	是	String	cce：表示使用自研ELBIngress。 通过API接口创建Ingress时必须增加该参数。
kubernetes.io/elb.port	是	Integer	界面上的对外端口，为注册到负载均衡服务地址上的端口。 取值范围：1~65535。 **说明** 部分端口为高危端口，默认被屏蔽，如21端口。
kubernetes.io/elb.subnet-id	否	String	为集群所在子网的ID，取值范围：1~100字符。 获取方法请参见：**VPC子网接口与OpenStack Neutron子网接口的区别是什么？**
kubernetes.io/elb.enterpriseID	否	String	企业项目ID，选择后可以直接创建在具体的企业项目下。 取值范围：1~100字符。 **获取方法：** 登录控制台后，单击顶部菜单右侧的“企业 > 项目管理”，在打开的企业项目列表中单击要加入的企业项目名称，进入企业项目详情页，找到“ID”字段复制即可。

参数	是否必填	参数类型	描述
kubernetes.io/elb.autocreate	是	elb.autocreate object	自动创建Ingress关联的ELB，详细字段说明参见**表3-18**。 **示例：** • 公网自动创建： 值为 '{"type":"public","bandwidth_name":"cce-bandwidth-******","bandwidth_chargemode":"bandwidth","bandwidth_size":5,"bandwidth_sharetype":"PER","eip_type":"5_bgp","name":"james"}' • 私网自动创建： 值为 '{"type":"inner", "name": "A-location-d-test"}'
kubernetes.io/elb.tags	否	String	为ELB添加资源标签，仅自动创建ELB时支持设置。 格式为key=value，同时添加多个标签时以英文逗号（,）隔开。
host	否	String	为服务访问域名配置，默认为""，表示域名全匹配。请确保所填写的域名已注册并备案，一旦配置了域名规则后，必须使用域名访问。
path	是	String	为路由路径，用户自定义设置。所有外部访问请求需要匹配host和path。 **说明** 此处添加的访问路径要求后端应用内存在相同的路径，否则转发无法生效。 例如，Nginx应用默认的Web访问路径为“/usr/share/nginx/html”，在为Ingress转发策略添加“/test”路径时，需要应用的Web访问路径下也包含相同路径，即“/usr/share/nginx/html/test”，否则将返回404。
ingress.beta.kubernetes.io/url-match-mode	否	String	路由匹配策略。 默认值为“STARTS_WITH”(前缀匹配)。 取值范围： • EQUAL_TO：精确匹配 • STARTS_WITH：前缀匹配 • REGEX：正则匹配

参数	是否必填	参数类型	描述
pathType	是	String	路径类型。 • ImplementationSpecific: 匹配方法取决于具体Ingress Controller的实现。在CCE中会使用ingress.beta.kubernetes.io/url-match-mode指定的匹配方式。 • Exact：精确匹配 URL 路径，且区分大小写。 • Prefix：前缀匹配，且区分大小写。该方式是将URL路径通过“/”分隔成多个元素 ，并且对元素进行逐个匹配。 如果URL中的每个元素均和路径匹配，则说明该URL的子路径均可以正常路由。 **说明** – Prefix匹配时每个元素均需精确匹配，如果URL的最后一个元素是请求路径中最后一个元素的子字符串，则不会匹配 。例如：/foo/bar匹配/foo/bar/baz，但不匹配/foo/barbaz。 – 通过“/”分隔元素时，若URL或请求路径以“/”结尾，将会忽略结尾的“/”。例如：/foo/bar会匹配/foo/bar/。 关于Ingress路径匹配示例，请参见**示例**。

表 3-18 elb.autocreate 字段数据结构说明

参数	是否必填	参数类型	描述
name	否	String	自动创建的负载均衡的名称。 取值范围：只能由中文、英文字母、数字、下划线、中划线、点组成，且长度范围为1-64个字符。 默认名称：cce-lb+service.UID
type	否	String	负载均衡实例网络类型，公网或者私网。 • public：公网型负载均衡 • inner：私网型负载均衡 默认类型：inner

参数	是否必填	参数类型	描述
bandwidth_name	公网型负载均衡必填	String	带宽的名称，默认值为：cce-bandwidth-******。 取值范围：只能由中文、英文字母、数字、下划线、中划线、点组成，且长度范围为1-64个字符。
bandwidth_chargemode	否	String	带宽付费模式。 ● bandwidth：按带宽 ● traffic：按流量 默认类型：bandwidth
bandwidth_size	公网型负载均衡必填	Integer	带宽大小，默认1Mbit/s~2000Mbit/s，请根据Region带宽支持范围设置。 调整带宽时的最小单位会根据带宽范围不同存在差异。 ● 小于等于300Mbit/s：默认最小单位为1Mbit/s。 ● 300Mbit/s~1000Mbit/s：默认最小单位为50Mbit/s。 ● 大于1000Mbit/s：默认最小单位为500Mbit/s。
bandwidth_sharetype	公网型负载均衡必填	String	带宽共享方式。 ● PER：独享带宽
eip_type	公网型负载均衡必填	String	弹性公网IP类型。 ● 5_telcom：电信 ● 5_union：联通 ● 5_bgp：全动态BGP ● 5_sbgp：静态BGP
vip_subnet_cidr_id	否	String	指定ELB所在的子网，该子网必须属于集群所在的VPC。 如不指定，则ELB与集群在同一个子网。
vip_address	否	String	负载均衡器的内网IP。仅支持指定IPv4地址，不支持指定IPv6地址。 该IP必须为ELB所在子网网段中的IP。若不指定，自动从ELB所在子网网段中生成一个IP地址。

参数	是否必填	参数类型	描述
available_zone	是	Array of strings	负载均衡所在可用区。 可以通过**查询可用区列表**获取所有支持的可用区。 独享型负载均衡器独有字段。
l4_flavor_name	是	String	四层负载均衡实例规格名称。 可以通过**查询规格列表**获取所有支持的类型。 • 弹性规格：适用于业务用量波动较大的场景，按实际使用量收取每小时使用的容量费用。 • 固定规格：适用于业务用量较为稳定的场景，按固定规格折算收取每小时使用的容量费用。 独享型负载均衡器独有字段。
l7_flavor_name	否	String	七层负载均衡实例规格名称。 可以通过**查询规格列表**获取所有支持的类型。 • 弹性规格：适用于业务用量波动较大的场景，按实际使用量收取每小时使用的容量费用。 • 固定规格：适用于业务用量较为稳定的场景，按固定规格折算收取每小时使用的容量费用。 独享型负载均衡器独有字段，必须与l4_flavor_name对应规格的类型一致，即都为弹性规格或都为固定规格。
elb_virsubnet_ids	否	Array of strings	负载均衡后端所在子网，不填默认为集群子网。不同实例规格将占用不同数量子网IP，不建议使用其他资源（如集群）的子网网段。 独享型负载均衡器独有字段。 示例： `"elb_virsubnet_ids": [ "14567f27-8ae4-42b8-ae47-9f847a4690dd" ]`

步骤3 创建Ingress。

kubectl create -f ingress-test.yaml

回显如下，表示Ingress服务已创建。

```
ingress/ingress-test created
```

kubectl get ingress

回显如下，表示Ingress服务创建成功，工作负载可访问。

```
NAME           HOSTS     ADDRESS          PORTS   AGE
ingress-test   *         121.**.**.**     80      10s
```

步骤4 访问工作负载（例如Nginx工作负载），在浏览器中输入访问地址http://121.**.**.**:80进行验证。

其中，121.**.**.**为统一负载均衡实例的IP地址。

----结束

添加 Ingress-对接已有 ELB

CCE支持在添加Ingress时选择对接已有的ELB。

说明

- 对接已有独享型ELB规格必须支持应用型（HTTP/HTTPS），且网络类型必须支持私网（有私有IP）。

YAML文件配置如下：

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-test
  annotations:
    kubernetes.io/elb.id: <your_elb_id>  #替换为您已有的ELB ID
    kubernetes.io/elb.ip: <your_elb_ip>  #替换为您已有的ELB IP
    kubernetes.io/elb.class: performance  #ELB类型
    kubernetes.io/elb.port: '80'
spec:
  rules:
  - host: ''
    http:
      paths:
      - path: '/'
        backend:
          service:
            name: <your_service_name>  #替换为您的目标服务名称
            port:
              number: 8080             #替换为您的目标服务端口
        property:
          ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
        pathType: ImplementationSpecific
  ingressClassName: cce
```

表 3-19 关键参数说明

参数	是否必填	参数类型	描述
kubernetes.io/elb.id	是	String	为负载均衡实例的ID，取值范围：1-100字符。 **获取方法：** 在控制台的“服务列表”中，单击“网络 > 弹性负载均衡 ELB”，单击ELB的名称，在ELB详情页的“基本信息”页签下找到“ID”字段复制即可。

参数	是否必填	参数类型	描述
kubernetes.io/elb.ip	否	String	为负载均衡实例的服务地址，公网ELB配置为公网IP，私网ELB配置为私网IP。
kubernetes.io/elb.class	是	String	负载均衡器类型。 ● performance：独享型负载均衡。 **说明** ELB Ingress对接已有的独享型ELB时，该独享型ELB必须支持应用型（HTTP/HTTPS）规格。

3.4.2 Nginx Ingress 管理

3.4.2.1 通过控制台创建 Nginx Ingress

前提条件

- Ingress为后端工作负载提供网络访问，因此集群中需提前部署可用的工作负载。若您无可用工作负载，可参考**创建工作负载**部署工作负载。
- 为上述工作负载配置ClusterIP类型Service，可参考**集群内访问（ClusterIP）**配置示例Service。
- 添加Nginx Ingress时，需在集群中提前安装NGINX Ingress 控制器，具体操作可参考**安装插件**。

约束与限制

- **不建议在ELB服务页面修改ELB实例的任何配置，否则将导致服务异常。**如果您已经误操作，请卸载Nginx Ingress插件后重装。
- Ingress转发策略中注册的URL需与后端应用提供访问的URL一致，否则将返回404错误。
- 负载均衡实例需与当前集群处于相同VPC 且为相同公网或私网类型。
- 负载均衡实例需要拥有至少两个监听器配额，且端口80和443没有被监听器占用。

添加 Nginx Ingress

本节以Nginx作为工作负载并添加Nginx Ingress为例进行说明。

步骤1 登录CCE控制台，单击集群名称进入集群。

步骤2 选择左侧导航栏的“服务”，在右侧选择“路由”页签，单击右上角“创建路由”。

步骤3 设置Ingress参数。

- **名称：**自定义Ingress名称，例如nginx-ingress-demo。
- **命名空间：**选择需要添加Ingress的命名空间。
- **对接Nginx：**集群中已安装**NGINX Ingress控制器**插件后显示此选项，未安装该插件时本选项不显示。

- **前端协议：**支持HTTP和HTTPS，安装NGINX Ingress控制器插件时预留的监听端口，默认HTTP为80，HTTPS为443。使用HTTPS需要配置相关证书。
- **证书来源：**使用证书以支持HTTPS数据传输加密认证。
  - 如果您选择“TLS密钥”，需要提前创建IngressTLS或kubernetes.io/tls类型的密钥证书，创建密钥的方法请参见**创建密钥**。
  - 如果您选择“默认证书”，NGINX Ingress控制器会使用插件默认证书进行加密认证。选择“默认证书”将使用NGINX Ingress控制器自带证书。
- **SNI：**SNI（Server Name Indication）是TLS的扩展协议，在该协议下允许同一个IP地址和端口号下对外提供多个基于TLS的访问域名，且不同的域名可以使用不同的安全证书。开启SNI后，允许客户端在发起TLS握手请求时就提交请求的域名信息。负载均衡收到TLS请求后，会根据请求的域名去查找证书：若找到域名对应的证书，则返回该证书认证鉴权；否则，返回缺省证书（服务器证书）认证鉴权。

- **转发策略配置：**请求的访问地址与转发规则匹配时（转发规则由域名、URL组成），此请求将被转发到对应的目标Service处理。单击“添加转发策略”按钮可添加多条转发策略。
  - 域名：实际访问的域名地址。请确保所填写的域名已注册并备案，在Ingress创建完成后，将域名与自动创建的负载均衡实例的IP（即Ingress访问地址的IP部分）绑定。一旦配置了域名规则，则必须使用域名访问。
  - URL匹配规则：
    - 默认：默认为前缀匹配。
    - 前缀匹配：例如映射URL为/healthz，只要符合此前缀的URL均可访问。例如/healthz/v1，/healthz/v2。
    - 精确匹配：表示只有URL完全匹配时，访问才能生效。例如映射URL为/healthz，则必须为此URL才能访问。
  - URL：需要注册的访问路径，例如：/healthz。

    说明

    - Nginx Ingress的访问路径匹配规则是基于“/”符号分隔的路径前缀匹配，并区分大小写。只要访问路径以“/”符号分隔后的子路径匹配此前缀，均可正常访问，但如果该前缀仅是子路径中的部分字符串，则不会匹配。例如URL设置为/healthz，则匹配/healthz/v1，但不匹配/healthzv1。
    - 此处添加的访问路径要求后端应用内存在相同的路径，否则转发无法生效。

      例如，Nginx应用默认的Web访问路径为“/usr/share/nginx/html”，在为Ingress转发策略添加“/test”路径时，需要应用的Web访问路径下也包含相同路径，即“/usr/share/nginx/html/test”，否则将返回404。
  - 目标服务名称：请选择已有Service或新建Service。页面列表中的查询结果已自动过滤不符合要求的Service。
  - 目标服务访问端口：可选择目标Service的访问端口。
  - 操作：可单击“删除”按钮删除该配置。
- **注解**：以“key: value”形式设置，可通过**Annotations**查询nginx-ingress支持的配置。

步骤4 配置完成后，单击“确定”。

创建完成后，在Ingress列表可查看到已添加的Ingress。

----结束

3.4.2.2 通过 Kubectl 命令行创建 Nginx Ingress

操作场景

本节以Nginx工作负载为例，说明kubectl命令添加Nginx Ingress的方法。

前提条件

- 集群必须已安装NGINX Ingress 控制器，具体操作可参考**安装插件**。
- Ingress为后端工作负载提供网络访问，因此集群中需提前部署可用的工作负载。若您无可用工作负载，可参考**创建工作负载**部署工作负载。
- 为上述工作负载配置ClusterIP类型的Service，可参考**集群内访问（ClusterIP）**配置示例Service。

添加 Nginx Ingress

步骤1 请参见**通过kubectl连接集群**，使用kubectl连接集群。

步骤2 创建名为“**ingress-test.yaml**”的YAML文件，此处文件名可自定义。

vi ingress-test.yaml

以HTTP协议访问为例，YAML文件配置如下。

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-test
spec:
  rules:
    - host: ''
      http:
        paths:
          - path: /
            backend:
              service:
                name: <your_service_name>  #替换为您的目标服务名称
                port:
                  number: <your_service_port>  #替换为您的目标服务端口
            property:
              ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
            pathType: ImplementationSpecific
  ingressClassName: nginx   # 表示使用Nginx Ingress
```

表 3-20 关键参数说明

参数	是否必填	参数类型	描述
ingressClassName	是	String	nginx：表示使用NginxIngress，未安装nginx-ingress插件时无法使用。 通过API接口创建Ingress时必须增加该参数。

参数	是否必填	参数类型	描述
host	否	String	为服务访问域名配置，默认为""，表示域名全匹配。请确保所填写的域名已注册并备案，一旦配置了域名规则后，必须使用域名访问。
path	是	String	为路由路径，用户自定义设置。所有外部访问请求需要匹配host和path。 **说明** ● Nginx Ingress的访问路径匹配规则是基于“/”符号分隔的路径前缀匹配，并区分大小写。只要访问路径以“/”符号分隔后的子路径匹配此前缀，均可正常访问，但如果该前缀仅是子路径中的部分字符串，则不会匹配。例如URL设置为/healthz，则匹配/healthz/v1，但不匹配/healthzv1。 ● 此处添加的访问路径要求后端应用内存在相同的路径，否则转发无法生效。例如，Nginx应用默认的Web访问路径为“/usr/share/nginx/html”，在为Ingress转发策略添加“/test”路径时，需要应用的Web访问路径下也包含相同路径，即“/usr/share/nginx/html/test”，否则将返回404。
ingress.beta.kubernetes.io/url-match-mode	否	String	路由匹配策略。 默认值为“STARTS_WITH”(前缀匹配)。 取值范围： ● EQUAL_TO：精确匹配 ● STARTS_WITH：前缀匹配

参数	是否必填	参数类型	描述
pathType	是	String	路径类型。 • ImplementationSpecific: 匹配方法取决于具体Ingress Controller的实现。在CCE中会使用ingress.beta.kubernetes.io/url-match-mode指定的匹配方式。 • Exact：精确匹配 URL 路径，且区分大小写。 • Prefix：前缀匹配，且区分大小写。该方式是将URL路径通过“/”分隔成多个元素 ，并且对元素进行逐个匹配。 如果URL中的每个元素均和路径匹配，则说明该URL的子路径均可以正常路由。 **说明** – Prefix匹配时每个元素均需精确匹配，如果URL的最后一个元素是请求路径中最后一个元素的子字符串，则不会匹配 。例如：/foo/bar匹配/foo/bar/baz，但不匹配/foo/barbaz。 – 通过“/”分隔元素时，若URL或请求路径以“/”结尾，将会忽略结尾的“/”。例如：/foo/bar会匹配/foo/bar/。 关于Ingress路径匹配示例，请参见示例。

步骤3 创建Ingress。

kubectl create -f ingress-test.yaml

回显如下，表示Ingress服务已创建。

```
ingress/ingress-test created
```

查看已创建的Ingress。

kubectl get ingress

回显如下，表示Ingress服务创建成功，工作负载可访问。

```
NAME           HOSTS     ADDRESS         PORTS   AGE
ingress-test   *         121.**.**.**    80      10s
```

步骤4 访问工作负载（例如Nginx工作负载），在浏览器中输入访问地址“http://121.**.**.**:80”进行验证。

其中，“121.**.**.**”为统一负载均衡实例的IP地址。

----结束

3.5 Pod 网络配置

3.5.1 为 Pod 配置 EIP

使用场景

CCE Autopilot集群中，Pod使用的是VPC的弹性网卡/辅助弹性网卡，可直接绑定弹性公网IP。

为方便用户在CCE内直接为Pod关联弹性公网IP，用户只需在创建Pod时，配置annotation（yangtse.io/pod-with-eip: "true"），弹性公网IP就会随Pod自动创建并绑定至该Pod。

约束限制

- 绑定EIP的Pod，如果要被公网成功访问，需要添加放通相应请求流量的安全组规则。
- 单个Pod只能绑定单个EIP。
- 创建Pod时，可指定相关的annotation配置EIP的属性，创建完成后，更新EIP相关的annotation均无效。
- 与Pod关联的EIP不要通过弹性公网IP的console或API直接操作（修改名称/删除/解绑/绑定/转包周期等操作），否则可能导致EIP功能异常。
- 自动创建的EIP被手动删除后，会导致网络异常，需要重建Pod。

EIP 跟随 Pod 创建

创建Pod时，填写pod-with-eip的annotation后，EIP会随Pod自动创建并绑定至该Pod。

以下示例创建一个名为nginx的无状态负载，EIP将随Pod自动创建并绑定至Pod。具体字段含义请参见**表3-22**。

- 创建Deployment时自动创建**独占带宽**类型的EIP，无需指定带宽ID，示例如下：

```
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"  # EIP跟随Pod创建
        yangtse.io/eip-bandwidth-size: "5"  # EIP带宽
        yangtse.io/eip-network-type: 5_bgp  # EIP类型
        yangtse.io/eip-charge-mode: bandwidth  # EIP计费模式
        yangtse.io/eip-bandwidth-name: <eip_bandwidth_name>  # EIP带宽名称
    spec:
      containers:
        - name: container-0
```

```
      image: nginx:alpine
      resources:
        limits:
          cpu: 250m
          memory: 500Mi
        requests:
          cpu: 250m
          memory: 500Mi
  imagePullSecrets:
    - name: default-secret
```

表 3-21 独占带宽 EIP 跟随 Pod 创建的 annotation 配置

annotation	是否可选	默认值	参数说明	取值范围
yangtse.io/pod-with-eip	必选	false	是否需要跟随Pod创建EIP并绑定到该Pod。	"false"或"true"
yangtse.io/eip-bandwidth-size	可选	5	带宽大小，单位为Mbit/s。	具体范围以各区域配置为准，根据带宽的计费类型不同可能存在差异，详情请参见弹性公网IP控制台的购买页面。 例如，“华东-上海一”区域按带宽计费类型的带宽大小范围为1Mbit/s~2000Mbit/s、按流量计费类型的带宽大小范围为1Mbit/s~300Mbit/s。
yangtse.io/eip-network-type	可选	5_bgp	公网IP类型。	具体类型以各区域配置为准，详情请参见弹性公网IP控制台的购买页面。 例如，“华东-上海一”区域支持以下类型： ● 5_bgp：全动态BGP ● 5_sbgp：静态BGP
yangtse.io/eip-charge-mode	可选	空	按流量计费或按带宽计费。 **建议填写该参数。**若该参数为空，表示不指定计费模式，则以该区域下弹性公网IP接口的默认值为准。	● bandwidth：按带宽计费 ● traffic：按流量计费

annotation	是否可选	默认值	参数说明	取值范围
yangtse.io/eip-bandwidth-name	可选	Pod名称	带宽名称。	• 1-64个字符，支持数字、字母、中文、_(下划线)、-（中划线）、.（点） • 最小长度：1 • 最大长度：64

- 创建Deployment时自动创建**共享带宽**类型的EIP，必须指定共享带宽ID，示例如下：

```
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"  # EIP跟随Pod创建
        yangtse.io/eip-network-type: 5_bgp  # EIP类型
        yangtse.io/eip-bandwidth-id: <eip_bandwidth_id>  # EIP共享带宽ID
    spec:
      containers:
        - name: container-0
          image: nginx:alpine
          resources:
            limits:
              cpu: 250m
              memory: 500Mi
            requests:
              cpu: 250m
              memory: 500Mi
      imagePullSecrets:
        - name: default-secret
```

表 3-22 共享带宽类型 EIP 跟随 Pod 创建的 annotation 配置

annotation	是否可选	默认值	参数说明	取值范围
yangtse.io/pod-with-eip	必选	false	是否需要跟随Pod创建EIP并绑定到该Pod。	"false"或"true"

annotation	是否可选	默认值	参数说明	取值范围
yangtse.io/eip-network-type	可选	5_bgp	公网IP类型。	• 5_bgp • 5_union • 5_sbgp 具体类型以各区域配置为准，详情请参见弹性公网IP控制台。
yangtse.io/eip-bandwidth-id	使用共享型带宽时必选	空	已有的带宽ID。 • 不填写该字段时，则默认使用独占带宽的EIP。独占带宽EIP的参数设置请参见表3-21。 • 填写该字段时，只允许同时指定yangtse.io/eip-network-type字段，且该字段为可选。	-

检查 Pod 的 EIP 就绪

容器网络控制器会在Pod IP分配后，为Pod绑定EIP并回写分配结果至Pod的annotation（yangtse.io/allocated-ipv4-eip），Pod业务容器的启动时间可能早于EIP分配结果回写成功时间。

您可以尝试为Pod配置init container并使用downwardAPI类型的存储卷把yangtse.io/allocated-ipv4-eip的annotation通过volume挂载到init container里，并在init container中检查EIP是否已经分配成功。您可以参考以下示例配置init container。

```
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']
...
```

EIP 跟随 Pod 删除

当Pod被删除时，自动创建的EIP会随Pod一起被删除。

3.5.2 为 Pod 配置固定 EIP

使用场景

CCE Autopilot集群支持为StatefulSet工作负载或直接创建的Pod分配固定的公网IP（EIP）。

约束限制

- 开启固定EIP功能需要和Pod自动创建EIP功能配合使用，详情请参见**为Pod配置EIP**。
- 目前只支持StatefulSet类型的Pod或直接创建的Pod固定EIP，暂不支持Deployment等其他类型的工作负载配置Pod固定EIP。
- 固定EIP创建后，生命周期内（如过期时间未到/Pod还在使用中）不支持通过Pod修改EIP属性。
- 对Pod的EIP地址无明确要求的业务不建议配置固定EIP，因为配置了固定EIP的Pod，Pod重建的耗时会略微变长。

配置固定 EIP

创建固定EIP的Pod时，填写EIP相关的annotation后，EIP会随Pod自动创建并绑定至该Pod。

以下示例创建一个名为nginx的有状态负载，EIP将随Pod自动创建并绑定至Pod。具体字段含义请参见**表3-23**。

- 创建有状态负载时固定**独占带宽**类型的EIP，无需指定带宽ID，示例如下：

```
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'  # Pod固定EIP
        yangtse.io/static-eip-expire-no-cascading: 'false'  # EIP级联删除
        yangtse.io/static-eip-expire-duration: 5m  # 固定EIP过期回收的时间间隔
        yangtse.io/pod-with-eip: 'true'  # EIP跟随Pod创建
        yangtse.io/eip-bandwidth-size: '5'  # EIP带宽
        yangtse.io/eip-network-type: 5_bgp  # EIP类型
```

```
      yangtse.io/eip-charge-mode: bandwidth    # EIP计费模式
  spec:
    containers:
      - name: container-0
        image: nginx:alpine
        resources:
          limits:
            cpu: 100m
            memory: 200Mi
          requests:
            cpu: 100m
            memory: 200Mi
    imagePullSecrets:
      - name: default-secret
```

- 创建有状态负载时固定**共享带宽**类型的EIP，必须指定带宽ID，示例如下：

```
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'  # Pod固定EIP
        yangtse.io/pod-with-eip: 'true'    # EIP跟随Pod创建
        yangtse.io/eip-network-type: 5_bgp    # EIP类型
        yangtse.io/eip-bandwidth-id: <eip_bandwidth_id>    # EIP共享带宽ID
    spec:
      containers:
        - name: container-0
          image: nginx:alpine
          resources:
            limits:
              cpu: 100m
              memory: 200Mi
            requests:
              cpu: 100m
              memory: 200Mi
      imagePullSecrets:
        - name: default-secret
```

表 3-23 Pod 固定 EIP 的 annotation 配置

annotation	是否可选	默认值	参数说明	取值范围
yangtse.io/static-eip	必选	false	是否开启Pod固定EIP，只有StatefulSet类型的Pod或无ownerReferences的Pod支持，默认不开启。	"false"或"true"

annotation	是否可选	默认值	参数说明	取值范围
yangtse.io/static-eip-expire-duration	可选	5m	删除固定EIP的Pod后，对应的固定EIP过期回收的时间间隔。	支持时间格式为Go time type，例如1h30m、5m。关于Go time type，请参见**Go time type**。
yangtse.io/static-eip-expire-no-cascading	可选	false	是否关闭StatefulSet工作负载的级联回收。 默认为false，表示StatefulSet删除后，会级联删除对应的固定EIP。如果您需要在删除StatefulSet对象后，在EIP过期回收时间内保留对应的固定EIP，用于下一次重建同名的StatefulSet再次使用对应的固定EIP，请将该参数设为true。	"false"或"true"

表 3-24 独占带宽 EIP 跟随 Pod 创建的 annotation 配置

annotation	是否可选	默认值	参数说明	取值范围
yangtse.io/pod-with-eip	必选	false	是否需要跟随Pod创建EIP并绑定到该Pod。	"false"或"true"
yangtse.io/eip-bandwidth-size	可选	5	带宽大小，单位为Mbit/s。	具体范围以各区域配置为准，根据带宽的计费类型不同可能存在差异，详情请参见弹性公网IP控制台的购买页面。 例如，“华东-上海一”区域按带宽计费类型的带宽大小范围为1Mbit/s~2000Mbit/s、按流量计费类型的带宽大小范围为1Mbit/s~300Mbit/s。

annotation	是否可选	默认值	参数说明	取值范围
yangtse.io/eip-network-type	可选	5_bgp	公网IP类型。	具体类型以各区域配置为准，详情请参见弹性公网IP控制台的购买页面。 例如，“华东-上海一”区域支持以下类型： ● 5_bgp：全动态BGP ● 5_sbgp：静态BGP
yangtse.io/eip-charge-mode	可选	空	按流量计费或按带宽计费。 **建议填写该参数。**若该参数为空，表示不指定计费模式，则以该区域下弹性公网IP接口的默认值为准。	● bandwidth：按带宽计费 ● traffic：按流量计费
yangtse.io/eip-bandwidth-name	可选	Pod名称	带宽名称。	● 1-64个字符，支持数字、字母、中文、_(下划线)、-（中划线）、.（点） ● 最小长度：1 ● 最大长度：64

表 3-25 共享带宽类型 EIP 跟随 Pod 创建的 annotation 配置

annotation	是否可选	默认值	参数说明	取值范围
yangtse.io/pod-with-eip	必选	false	是否需要跟随Pod创建EIP并绑定到该Pod。	"false"或"true"
yangtse.io/eip-network-type	可选	5_bgp	公网IP类型。	● 5_bgp ● 5_union ● 5_sbgp 具体类型以各区域配置为准，详情请参见弹性公网IP控制台。

annotation	是否可选	默认值	参数说明	取值范围
yangtse.io/eip-bandwidth-id	使用共享型带宽时必选	空	已有的带宽ID。 • 不填写该字段时，则默认使用独占带宽的EIP。独占带宽EIP的参数设置请参见表3-21。 • 填写该字段时，只允许同时指定yangtse.io/eip-network-type字段，且该字段为可选。	-

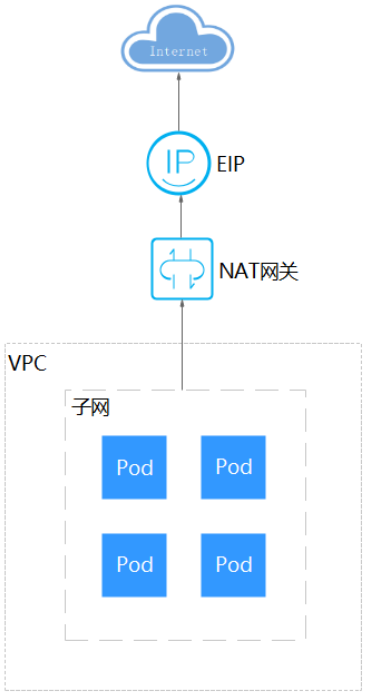
删除固定 EIP

删除Pod后，在配置的固定EIP过期时间内，如果有同名的Pod创建，EIP依旧可用。只有在EIP过期时间内没有同名Pod创建或者删除StatefulSet时开启级联删除EIP时，固定EIP才会删除。

3.6 从容器访问公网

NAT网关能够为VPC内的容器实例提供网络地址转换（Network Address Translation）服务，SNAT功能通过绑定弹性公网IP，实现私有IP向公有IP的转换，可实现VPC内的容器实例共享弹性公网IP访问Internet。其原理如图3-23所示。通过NAT网关的SNAT功能，即使VPC内的容器实例不配置弹性公网IP也可以直接访问Internet，提供超大并发数的连接服务，适用于请求量大、连接数多的服务。

图 3-23 SNAT



操作步骤

您可以通过如下步骤实现容器实例访问Internet。

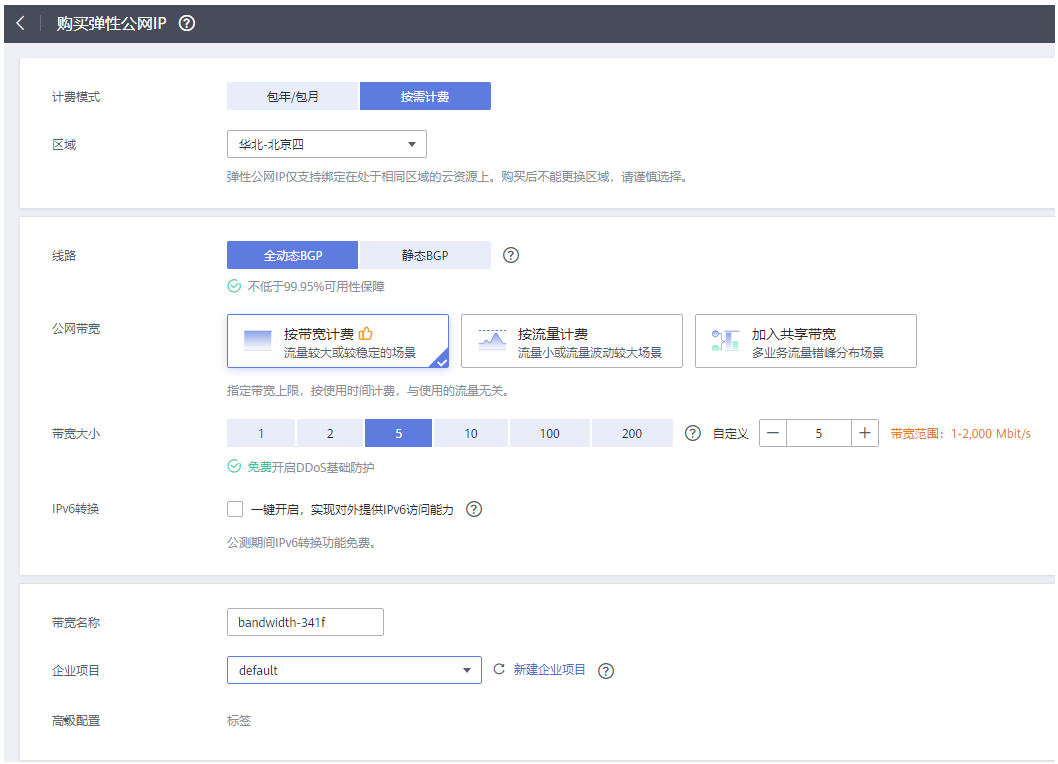
步骤1 创建弹性公网IP。

1. 登录管理控制台。
2. 在管理控制台左上角单击 ，选择区域和项目。
3. 在控制台首页，单击左上角的 ，在展开的列表中单击“网络 > 弹性公网IP”。
4. 在“弹性公网IP”界面，单击“购买弹性公网IP”。
5. 根据界面提示配置参数。

   说明

   此处“区域”需选择容器实例所在区域。

图 3-24 购买弹性公网 IP



步骤2 创建NAT网关，具体请参见**购买NAT网关**。

1. 登录管理控制台。
2. 在管理控制台左上角单击 ，选择区域和项目。
3. 在控制台首页，单击左上角的 ，在展开的列表中单击“网络 > NAT网关”。
4. 在NAT网关页面，单击右上角的“购买公网NAT网关”。
5. 根据界面提示配置参数。

说明

此处需选择集群相同的VPC。

图 3-25 购买公网 NAT 网关

步骤3 配置SNAT规则，为子网绑定弹性公网IP，具体请参见**添加SNAT规则**。

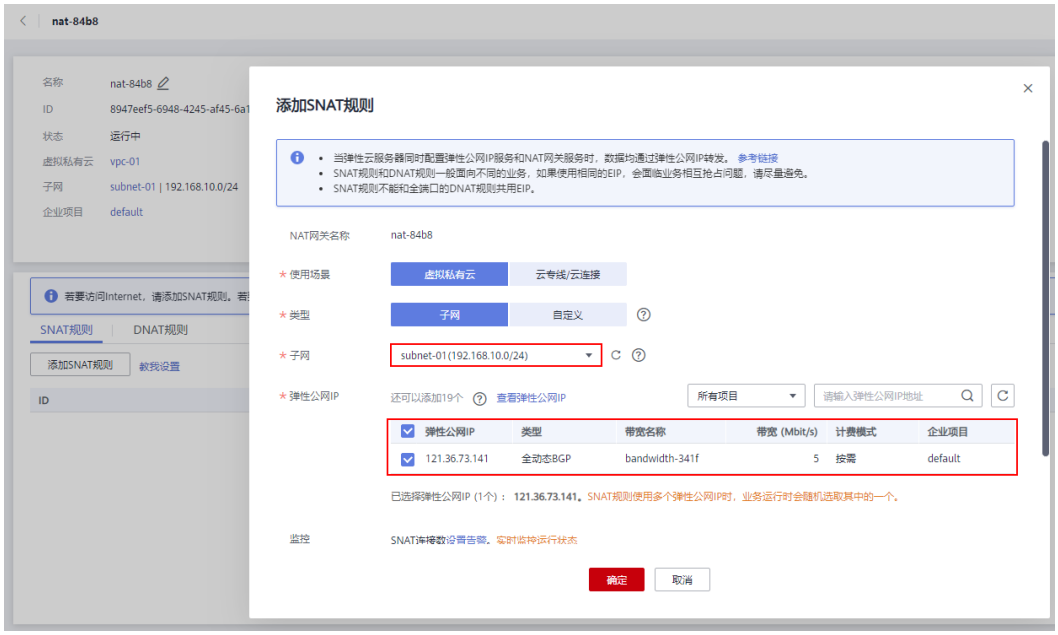
1. 登录管理控制台。
2. 在管理控制台左上角单击，选择区域和项目。
3. 在控制台首页，单击左上角的，在展开的列表中单击“网络 >NAT网关”。
4. 在NAT网关页面，单击需要添加SNAT规则的NAT网关名称。
5. 在SNAT规则页签中，单击“添加SNAT规则”。
6. 根据界面提示配置参数。

说明

SNAT规则按网段生效，此处子网需要选择容器所在子网，即创建集群时选择的容器子网。

对于存在多个网段的情况，可以创建多个SNAT规则或选择自定义网段，只要网段能包含容器子网即可。

图 3-26 配置 SNAT 规则



SNAT规则配置完成后，您就可以从容器中访问公网了，从容器中能够ping通公网。

----结束

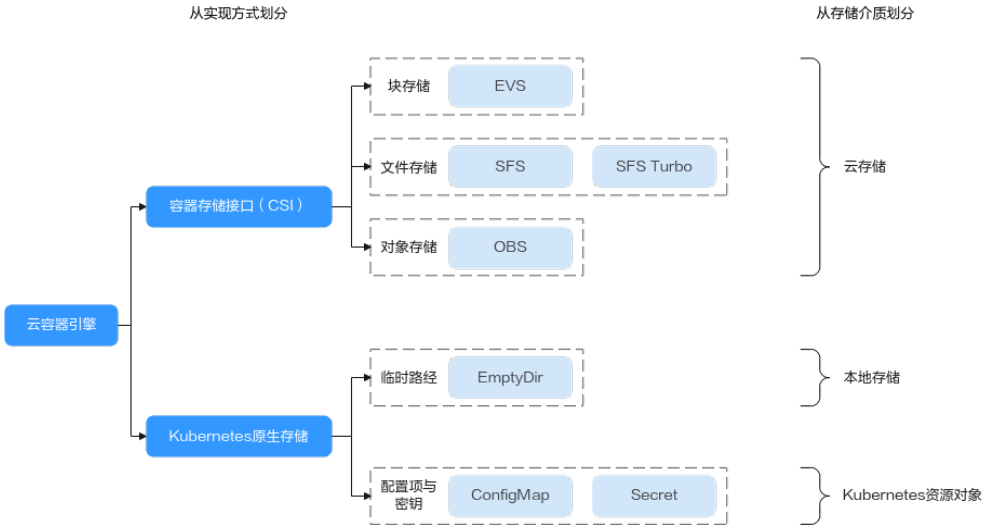
4 存储

4.1 存储概述

存储概览

CCE Autopilot集群的容器存储功能基于Kubernetes容器存储接口（CSI）实现，深度融合多种类型的云存储并全面覆盖不同的应用场景，而且完全兼容Kubernetes原生的存储服务，例如EmptyDir、Secret、ConfigMap等存储类型。

图 4-1 容器存储概览类型



Autopilot集群支持工作负载Pod绑定多种类型的存储：

- 从实现方式上划分，可以分为容器存储接口和Kubernetes原生存储。

类别	说明
容器存储接口	**Out-of-Tree**的形式，规定了标准的容器存储接口，可以允许存储供应商使用符合标准的自定义存储插件，通过PVC/PV的形式实现挂载，摒弃了以往需要将插件源码添加到Kubernetes代码仓库统一构建、编译、发布的方式。从Kubernetes 1.13开始，容器存储接口（CSI）是实现卷插件的推荐方法。
Kubernetes原生存储	In-Tree的形式，通过Kubernetes代码仓库统一构建、编译、发布。

- 从存储介质上划分，可以分为云存储、本地存储和Kubernetes资源对象。

类别	说明	应用场景
云存储	存储介质为存储供应商提供的云存储，该类别的存储卷挂载均通过PVC/PV形式。	一般用于存储可用性要求较高的数据，或部分数据需要共享的场景，例如日志保存、媒体资源存放等。 根据具体的使用场景，您可以选择合适的云存储类型，详情请参见**云存储对比**。
本地存储	仅支持临时路径（EmptyDir），该类型存储是Kubernetes原生的EmptyDir类型，生命周期与容器实例相同，并支持指定内存作为存储介质。容器实例消亡时，EmptyDir会被删除，数据会永久丢失。	用于存储非高可用数据，可在IO要求较高、延迟低的场景下使用。 详情请参见**本地存储**。
Kubernetes资源对象	ConfigMap和Secret是集群中创建的资源，属于比较特殊的存储类型，由Kubernetes API服务器上的tmpfs（基于RAM的文件系统）提供存储。	ConfigMap一般用于给Pod注入配置数据。 Secret一般用于给Pod传递敏感信息，例如密码。

云存储对比

对比维度	云硬盘EVS	文件存储SFS	极速文件存储SFS Turbo	对象存储OBS
概念	云硬盘（Elastic Volume Service）可以为云服务器提供高可靠、高性能、规格丰富并且可弹性扩展的块存储服务，可满足不同场景的业务需求，适用于分布式文件系统、开发测试、数据仓库以及高性能计算等场景。	SFS为用户提供一个完全托管的共享文件存储，能够弹性伸缩至PB规模，具备高可用性和持久性，为海量数据、高带宽型应用提供有力支持。适用于多种应用场景，包括HPC、媒体处理、文件共享、内容管理和Web服务等。	SFS Turbo为用户提供一个完全托管的共享文件存储，能够弹性伸缩至320TB规模，具备高可用性和持久性，为海量的小文件、低延迟高IOPS型应用提供有力支持。适用于多种应用场景，包括高性能网站、日志存储、压缩解压、DevOps、企业办公、容器应用等。	对象存储服务（Object Storage Service，OBS）提供海量、安全、高可靠、低成本的数据存储能力，可供用户存储任意类型和大小的数据。适合企业备份/归档、视频点播、视频监控等多种数据存储场景。
存储数据的逻辑	存放的是二进制数据，无法直接存放文件，如果需要存放文件，需要先格式化文件系统后使用。	存放的是文件，会以文件和文件夹的层次结构来整理和呈现数据。	存放的是文件，会以文件和文件夹的层次结构来整理和呈现数据。	存放的是对象，可以直接存放文件，文件会自动产生对应的系统元数据，用户也可以自定义文件的元数据。
访问方式	只能在ECS/BMS中挂载使用，不能被操作系统应用直接访问，需要格式化成文件系统进行访问。	在ECS/BMS中通过网络协议挂载使用。需要指定网络地址进行访问，也可以将网络地址映射为本地目录后进行访问。	提供标准的文件访问协议NFS（仅支持NFSv3），用户可以将现有应用和工具与SFS Turbo无缝集成。	可以通过互联网或专线访问。需要指定桶地址进行访问，使用的是HTTP和HTTPS等传输协议。

对比维度	云硬盘EVS	文件存储SFS	极速文件存储SFS Turbo	对象存储OBS
静态存储卷	支持，请参见**通过静态存储卷使用已有云硬盘**。	支持，请参见**通过静态存储卷使用已有文件存储**。	支持，请参见**通过静态存储卷使用已有极速文件存储**。	支持，请参见**通过静态存储卷使用已有对象存储**。
动态存储卷	支持，请参见**通过动态存储卷使用云硬盘**。	支持，请参见**通过动态存储卷使用文件存储**。	SFS Turbo不支持，SFS Turbo子目录支持，请参见**通过动态存储卷创建SFS Turbo子目录（推荐）**。	支持，请参见**通过动态存储卷使用对象存储**。
主要特点	非共享存储，每个云盘只能在单个Pod挂载。	共享存储，可提供高性能、高吞吐存储服务。	共享存储，可提供高性能、高带宽存储服务。	共享存储，用户态文件系统。
应用场景	HPC高性能计算、企业核心集群应用、企业应用系统和开发测试等。 **说明** 高性能计算：主要是高速率、高IOPS的需求，用于作为高性能存储，比如工业设计、能源勘探等。	HPC高性能计算、媒体处理、内容管理和Web服务、大数据和分析应用程序等。 **说明** 高性能计算：主要是高带宽的需求，用于共享文件存储，比如基因测序、图片渲染等。	高性能网站、日志存储、DevOps、企业办公等。	大数据分析、静态网站托管、在线视频点播、基因测序、智能视频监控、备份归档、企业云盘（网盘）等。
容量	TB级别	SFS 1.0：PB级别 通用文件系统（原SFS 3.0）：EB级别	通用型：TB级别	EB级别
时延	1~2ms	SFS 1.0：3~20ms 通用文件系统（原SFS 3.0）：10ms	通用型：1~5ms	10ms
最大IOPS	因规格而异，范围为2.2K~256K	SFS 1.0：2K 通用文件系统（原SFS 3.0）：百万级	通用型：最大达100K	千万级

对比维度	云硬盘EVS	文件存储SFS	极速文件存储SFS Turbo	对象存储OBS
带宽	MB/s级别	SFS 1.0：GB/s级别 通用文件系统（原SFS 3.0）：TB/s级别	通用型：最大为GB/s级别	TB/s级别

本地存储

临时路径（EmptyDir）是Kubernetes原生的EmptyDir类型，生命周期与容器实例相同，并支持指定内存作为存储介质。容器实例消亡时，EmptyDir会被删除，数据会永久丢失。详情请参见**临时路径（EmptyDir）**。

主要特点：本地临时卷，存储空间来自本地的kubelet根目录或内存。

应用场景：

- 缓存空间，例如基于磁盘的归并排序。
- 为耗时较长的计算任务提供检查点，以便任务能方便地从崩溃前状态恢复执行。
- 在Web服务器容器服务数据时，保存内容管理器容器获取的文件。

企业项目支持说明

- 自动创建存储：

  CCE Autopilot集群支持使用存储类创建云硬盘和对象存储类型PVC时指定企业项目，将创建的存储资源（云硬盘和对象存储）归属于指定的企业项目下，**企业项目可选为集群所属的企业项目或default企业项目**。

  若不指定企业项目，则创建的存储资源默认使用存储类StorageClass中指定的企业项目。对于CCE Autopilot集群提供的csi-disk和csi-obs存储类，所创建的存储资源属于default企业项目。

- 使用已有存储：

  使用PV创建PVC时，因为存储资源在创建时已经指定了企业项目，如果PVC中指定企业项目，则务必确保在PVC和PV中指定的everest.io/enterprise-project-id保持一致，否则两者无法正常绑定。

相关文档

- **存储基础知识**
- **云硬盘存储（EVS）**
- **文件存储（SFS）**
- **极速文件存储（SFS Turbo）**
- **对象存储（OBS）**

4.2 存储基础知识

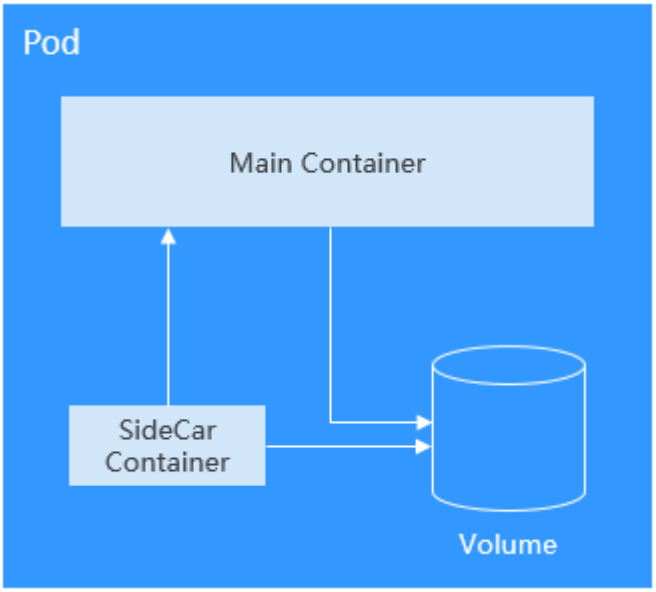
Volume（卷）

容器中的文件在磁盘上是临时存放的，这给容器中运行的较重要的应用程序带来如下两个问题：

- 当容器重建时，容器中的文件将会丢失。
- 当在一个Pod中同时运行多个容器时，容器间需要共享文件。

Kubernetes抽象出了Volume（卷）来解决以上两个问题。Kubernetes的Volume是Pod的一部分，Volume不是单独的对象，不能独立创建，只能在Pod中定义。Pod中的所有容器都可以使用Volume，但需要将Volume挂载到容器中的目录下。

实际中使用容器存储如下图所示，可将同一个Volume挂载到不同的容器中，实现不同容器间的存储共享。



存储卷的基本使用原则如下：

- 一个Pod可以挂载多个Volume。虽然单Pod可以挂载多个Volume，但是并不建议给一个Pod挂载过多卷。
- 一个Pod可以挂载多种类型的Volume。
- 每个被Pod挂载的Volume卷，可以在不同的容器间共享。
- Kubernetes环境推荐使用PVC和PV方式挂载Volume。

说明

卷（Volume）的生命周期与挂载它的Pod相同，即Pod被删除的时候，Volume也一起被删除。但是Volume里面的文件可能在Volume消失后仍然存在，这取决于Volume的类型。

Kubernetes提供了非常丰富的Volume类型，主要可分为In-Tree和Out-of-Tree两个大类：

卷（Volume）分类	描述
In-Tree	In-Tree卷是通过Kubernetes代码仓库维护的，与Kubernetes二进制文件一起构建、编译、发布，当前Kubernetes已不再接受这种模式的卷类型。 例如HostPath、EmptyDir、Secret和ConfigMap等Kubernetes原生支持的卷都属于这个类型。 而PVC（PersistentVolumeClaim）可以说是一种特殊的In-Tree卷，Kubernetes使用这种类型的卷从In-Tree模式向Out-of-Tree模式进行转换，这种类型的卷允许使用者在不同的存储供应商环境中“申请”使用底层存储创建的PV（PersistentVolume）。
Out-of-Tree	Out-of-Tree卷包括容器存储接口（CSI）和FlexVolume（已弃用），存储供应商只需遵循一定的规范即可创建自定义存储插件，创建可供Kubernetes使用的PV，而无需将插件源码添加到Kubernetes代码仓库。例如SFS、OBS等云存储都是通过在集群中安装存储驱动的形式使用的，需要在集群中创建对应的PV，然后使用PVC挂载到Pod中。

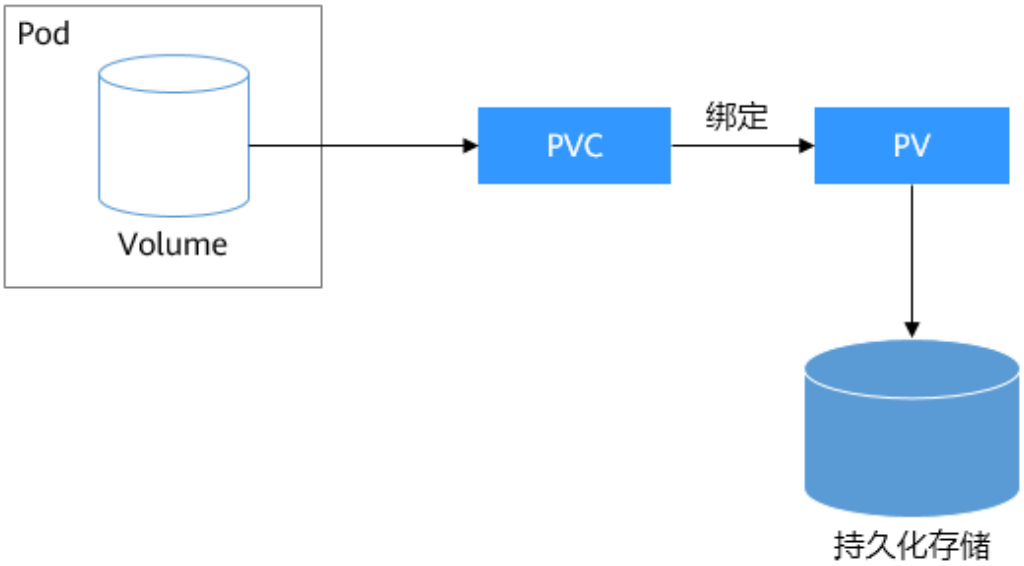
PV 与 PVC

Kubernetes抽象了PV（PersistentVolume）和PVC（PersistentVolumeClaim）来定义和使用存储，从而让使用者不用关心具体的基础设施，当需要存储资源的时候，只要像CPU和内存一样，声明要多少即可。

- PV：PV是PersistentVolume的缩写，译为持久化存储卷，描述的是一个集群里的持久化存储卷，它和节点一样，属于集群级别资源，其对象作用范围是整个Kubernetes集群。PV可以有自己的独立生命周期，不依附于Pod。
- PVC：PVC是PersistentVolumeClaim的缩写，译为持久化存储卷声明，描述的是负载对存储的申领。为应用配置存储时，需要声明一个存储需求（即PVC），Kubernetes会通过最佳匹配的方式选择一个满足需求的PV，并与PVC绑定。PVC与PV是一一对应关系，在创建PVC时，需描述请求的持久化存储的属性，比如，存储的大小、可读写权限等等。

在Pod中可以使用Volume关联PVC，即可让Pod使用到存储资源，它们之间的关系如下图所示。

图 4-2 PVC 绑定 PV



CSI

CSI（Container Storage Interface，容器存储接口）是容器标准存储接口规范，也是Kubernetes社区推荐的存储插件实现方案。

存储卷访问模式

存储卷只能以底层存储资源所支持的方式挂载到宿主系统上。例如，文件存储可以支持多个Pod读写，云硬盘只能被一个Pod读写。

- ReadWriteOnce：存储卷可以被一个Pod以读写方式挂载。
- ReadWriteMany：存储卷可以被多个Pod以读写方式挂载。

表 4-1 存储卷支持的访问模式

存储类型	ReadWriteOnce	ReadWriteMany
云硬盘EVS	√	×
文件存储SFS	×	√
对象存储OBS	×	√
极速文件存储SFS Turbo	×	√

存储卷挂载方式

通常在使用存储卷时，可以通过以下方式挂载：

可以使用PV描述已有的底层存储资源，然后通过创建PVC在Pod中使用底层存储资源。也可以使用动态创建的方式，在PVC中指定存储类（StorageClass），利用StorageClass中的Provisioner自动创建PV来绑定PVC。

表 4-2 挂载存储卷的方式

挂载方式	说明	支持的存储卷类型	其他限制
静态创建存储卷（使用已有存储）	需要手动创建底层存储（例如云硬盘、文件存储等）和PV。 控制台操作流程：首先，使用已有的底层存储创建PV。其次，创建PVC，并将PVC挂载至工作负载创建的Pod中。创建PVC时，Kubernetes会将PVC和匹配的PV进行绑定，这样就实现了工作负载访问存储服务的能力。 **说明** 若存储卷的访问模式为ReadWriteOnce，则创建工作负载时，工作负载的Pod实例数量只能设置为1。	所有存储卷均支持	无
动态创建存储卷（自动创建存储）	自动创建底层存储和PV。 控制台操作流程：首先，创建PVC，在PVC中指定StorageClass。StorageClass中的Provisioner会根据PVC的需要自动创建底层存储和PV，自动创建的PV直接与PVC绑定，挂载至对应的工作负载创建的Pod中。其次，将PVC挂载至工作负载中。 **说明** 若存储卷的访问模式为ReadWriteOnce，则创建工作负载时，工作负载的Pod实例数量只能设置为1。	云硬盘存储、对象存储、文件存储	无

挂载方式	说明	支持的存储卷类型	其他限制
动态挂载（VolumeClaimTemplate）	自动创建底层存储和PV。动态挂载能力通过卷申领模板（**volumeClaimTemplates**字段）实现，并依赖于StorageClass的动态创建PV能力。 控制台操作流程：首先，创建工作负载，在“数据存储”中选择“动态挂载”。其次，通过动态挂载的方式创建PVC，PVC指定StorageClass，StorageClass中的Provisioner会根据PVC的需要自动创建底层存储和PV。 **说明** 即使存储卷的访问模式为ReadWriteOnce，工作负载的Pod实例数量也可以设置为多个。原因在于：动态挂载可以为每一个Pod关联一个独有的PVC及PV，当Pod被重新调度后，仍然能够根据该PVC名称挂载原有的数据。	仅云硬盘存储支持	仅有状态工作负载支持

PV 回收策略

PV回收策略用于指定删除PVC时，底层卷的回收策略，支持设定Delete、Retain回收策略。

- Delete：删除PVC的动作会将PV对象从Kubernetes中移除，同时也会从外部基础设施中移除所关联的底层存储资产。

  说明

  包周期的资源无法通过Delete回收策略进行级联删除。

- Retain：当PVC对象被删除时，PV对象与底层存储资源均不会被删除，需要手动删除回收。PVC删除后PV资源状态为“已释放（Released）”，且不能直接再次被PVC绑定使用。

  您可以通过以下步骤来手动删除回收：

  a. 手动删除PersistentVolume对象。
  b. 根据情况，手动清除所关联的底层存储资源上的数据。
  c. 手动删除所关联的底层存储资源。

  如果您希望重用该底层存储资源，可以重新创建新的PersistentVolume对象。

CCE Autopilot集群还支持一种删除PVC时不删除底层存储资源的使用方法，当前仅支持使用YAML创建：PV回收策略设置为Delete，并添加annotations“everest.io/reclaim-policy: retain-volume-only”。这样在删除PVC时，PV会被删除，但底层存储资源会保留。

以云硬盘为例，YAML示例如下：

```
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>   # 替换为集群所在的区域
    failure-domain.beta.kubernetes.io/zone: <your_zone>       # 替换为云硬盘所在的可用区
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>   # 替换为集群所在的区域
    failure-domain.beta.kubernetes.io/zone: <your_zone>       # 替换为云硬盘所在的可用区
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
```

相关文档

- 更多关于Kubernetes存储的信息，请参见**Storage**。
- 更多关于CCE Autopilot集群容器存储的信息，请参见**存储概述**。

4.3 云硬盘存储（EVS）

4.3.1 云硬盘概述

为满足数据持久化的需求，CCE Autopilot集群支持将云硬盘（EVS）创建的存储卷挂载到容器的某一路径下，当容器在同一可用区内迁移时，挂载的云硬盘将一同迁移。通过云硬盘，可以将存储系统的远端文件目录挂载到容器中，数据卷中的数据将被永久保存，即使删除了容器，数据卷中的数据依然保存在存储系统中。

说明

目前，只有部分区域支持使用云硬盘，具体请以控制台为准。您可以通过**功能总览**，了解支持该功能的所有区域。

云硬盘性能规格

云硬盘性能的主要指标包括：

- IOPS：云硬盘每秒进行读写的操作次数。
- 吞吐量：云硬盘每秒成功传送的数据量，即读取和写入的数据量。
- IO读写时延：云硬盘连续两次进行读写操作所需要的最小时间间隔。

表 4-3 云硬盘性能规格

参数	极速型SSD V2	通用型SSD V2	极速型SSD	通用SSD	超高IO	高IO
云硬盘最大容量（GiB）	• 系统盘：1024 • 数据盘：32768	• 系统盘：1024 • 数据盘：32768	• 系统盘：1024 • 数据盘：32768	• 系统盘：1024 • 数据盘：32768	• 系统盘：1024 • 数据盘：32768	• 系统盘：1024 • 数据盘：32768
最大IOPS	256000	128000	128000	20000	50000	5000
最大吞吐量（MiB/s）	4000	1000	1000	250	350	150
IOPS突发上限	NA	NA	64000	8000	16000	5000
云硬盘IOPS性能计算公式	IOPS值由用户预配置，范围为100~256000，具体可配置值≤(1000*容量（GiB）)	IOPS值由用户预配置，范围为3000~128000，具体可配置值≤(500*容量（GiB）)	IOPS = min (128000, 1800 + 50 × 容量)	IOPS=min (20000, 1800 + 12 × 容量)	IOPS = min (50000, 1800 + 50 × 容量)	IOPS = min (5000, 1800 + 8 × 容量)

参数	极速型SSD V2	通用型SSD V2	极速型SSD	通用SSD	超高IO	高IO
云硬盘吞吐量性能计算公式（MiB/s）	吞吐量 ≤ min(4000，预配置IOPS/16)	吞吐量值由用户配置，范围为125~1000，具体可配置值 ≤ (IOPS/4)	吞吐量 = min(1000, 120 + 0.5 × 容量)	吞吐量 = min(250, 100 + 0.5 × 容量)	吞吐量 = min(350, 120 + 0.5 × 容量)	吞吐量 = min(150, 100 + 0.15 × 容量)
单队列访问时延（ms）	亚毫秒级	1	亚毫秒级	1	1	1~ 3
API名称	ESSD2	GPSSD2	ESSD	GPSSD	SSD	SAS

关于云硬盘性能的详细介绍，请以**磁盘类型及性能介绍**为准。

使用场景

根据使用场景不同，云硬盘类型的存储支持以下挂载方式：

- **通过静态存储卷使用已有云硬盘**：即静态创建的方式，需要先使用已有的云硬盘创建PV，然后通过PVC在工作负载中挂载存储。适用于已有可用的底层存储或底层存储需要包周期的场景。
- **通过动态存储卷使用云硬盘**：即动态创建的方式，无需预先创建云硬盘，在创建PVC时通过指定存储类（StorageClass），即可自动创建云硬盘和对应的PV对象。适用于无可用的底层存储，需要新创建的场景。
- **在有状态负载中动态挂载云硬盘存储**：仅有状态工作负载支持，可以为每一个Pod关联一个独有的PVC及PV，当Pod被重新调度后，仍然能够根据该PVC名称挂载原有的数据。适用于多实例的有状态工作负载。

计费说明

- 挂载云硬盘类型的存储卷时，通过StorageClass**自动创建**的云硬盘默认创建计费模式为“按需计费”，且不支持将云硬盘单独转包周期。如需使用包周期的云硬盘，请**使用已有的云硬盘存储**进行挂载。
- 关于云硬盘的价格信息，请参见**云硬盘计费说明**。

4.3.2 通过静态存储卷使用已有云硬盘

CCE Autopilot集群支持使用已有的云硬盘创建存储卷（PersistentVolume）。创建成功后，通过创建相应的PersistentVolumeClaim绑定当前PersistentVolume使用。适用于已有底层存储或底层存储需要包周期的场景。

前提条件

- 您已经创建好一个集群。

- 您已经创建好一块云硬盘，并且云硬盘满足以下条件：
  - 已有的云硬盘不可以是系统盘、专属盘或共享盘。
  - 云硬盘模式需选择SCSI（购买云硬盘时默认为VBD模式）。
  - 云硬盘的状态可用，且未被其他资源使用。
  - 不支持使用加密的云硬盘。
  - 仅支持选择集群所属企业项目和default企业项目下的云硬盘。
  - 不支持使用已进行分区的云硬盘。
  - 仅支持使用ext4类型的云硬盘。
- 如果您需要通过命令行创建，需要使用kubectl连接到集群，详情请参见**通过kubectl连接集群**。

约束与限制

- 目前，只有部分区域支持使用云硬盘，具体请以控制台为准。您可以通过**功能总览**，了解支持该功能的所有区域。
- 云硬盘不支持被多个工作负载、同一个工作负载的多个实例或多个任务使用。创建无状态工作负载时，若使用了EVS云硬盘，建议工作负载只选择一个实例。
- 集群版本要求在v1.27.8-r0、v1.28.6-r0及以上。如果您的集群版本不符合该要求，则需要通过集群升级使用该功能。
- 集群中每个工作负载最多挂载10个云硬盘存储，若挂载数量超过10，可能导致负载运行异常。

基于已有云硬盘创建并挂载存储卷

基于已有云硬盘创建静态存储卷，并将其挂载到工作负载中，从而实现持久化存储。本节将介绍两种方式创建并挂载静态存储卷，即控制台方式和kubectl命令行方式。

使用控制台方式

步骤1 登录CCE控制台，单击集群名称进入集群。

步骤2 静态创建存储卷声明和存储卷。

1. 在左侧导航栏选择“存储”，在右侧选择“存储卷声明”页签。单击右上角“创建存储卷声明 PVC”，在弹出的窗口中填写存储卷声明参数。

图 4-3 创建存储卷声明 PVC

参数	描述
存储卷声明类型	本文示例中选择“云硬盘”。
PVC名称	输入PVC的名称，同一命名空间下的PVC名称需唯一。
命名空间	命名空间是Kubernetes集群中的抽象概念，可以将集群中的资源或对象划分为一个组，且不同命名空间中的数据彼此隔离，您可以根据需要创建并使用命名空间。 集群创建后会默认生成default命名空间，如果没有特殊要求，可以直接选择default命名空间。
创建方式	- 已有底层存储的场景下，根据是否已经创建存储卷可选择“新建存储卷 PV”或“已有存储卷 PV”来静态创建PVC。 - 无可用底层存储的场景下，可选择“动态创建”，具体操作请参见**通过动态存储卷使用云硬盘**。 本文示例中选择“新建存储卷”，可通过控制台同时创建PV及PVC。
关联存储卷[a]	选择集群中已有的PV卷，需要提前创建PV，请参考**相关操作**中的“创建存储卷”操作。 本文示例中无需选择。
云硬盘[b]	单击“选择云硬盘”，您可以在新页面中勾选满足要求的云硬盘，并单击“确定”。
PV名称[b]	输入PV名称，同一集群内的PV名称需唯一。
访问模式[b]	云硬盘类型的存储卷仅支持ReadWriteOnce，表示存储卷可以被一个节点以读写方式挂载，详情请参见**存储卷访问模式**。

参数	描述
回收策略 b	您可以选择Delete或Retain，用于指定删除PVC时底层存储的回收策略，详情请参见**PV回收策略**。 本文示例中选择“Retain”。 **说明** 多个PV使用同一个底层存储时建议使用**Retain**，避免级联删除底层卷。

说明

a：创建方式选择“已有存储卷 PV”时可设置。

b：创建方式选择“新建存储卷 PV”时可设置。

2. 单击“创建”，将同时为您创建存储卷声明及存储卷。

   您可以在左侧导航栏中选择“存储”，在“存储卷声明”和“存储卷”页签下查看已经创建的存储卷声明和存储卷。

步骤3 创建应用。

1. 在左侧导航栏中选择“工作负载”，在右侧选择“有状态负载”页签。
2. 单击页面右上角“创建工作负载”，在“容器配置”中选择“数据存储”页签，并单击“添加存储卷 > 已有存储卷声明 (PVC)”。

   本文主要为您介绍存储卷的挂载使用，如**表4-4**，其他参数详情请参见**创建工作负载**。

   本例中将该盘挂载到容器中/data路径下，在该路径下生成的容器数据会存储到云硬盘中。

说明

由于云硬盘为非共享模式，工作负载下多个实例无法同时挂载，会导致实例启动异常。因此挂载云硬盘时，工作负载实例数需为1。

图 4-4 存储卷挂载

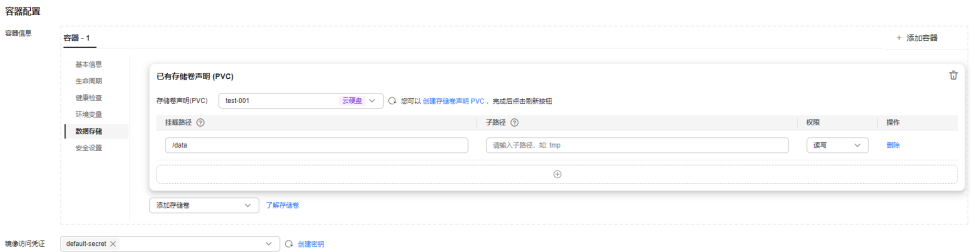


表 4-4 存储卷挂载

参数	参数说明
存储卷声明（PVC）	选择已有的云硬盘存储卷。 云硬盘存储卷无法被多个工作负载重复挂载。

参数	参数说明
挂载路径	请输入挂载路径，如：/data。 数据存储挂载到容器上的路径。请不要挂载在系统目录下，如“/”、“/var/run”等，会导致容器异常。建议挂载在空目录下，若目录不为空，请确保目录下无影响容器启动的文件，否则文件会被替换，导致容器启动异常，工作负载创建失败。 **须知** 挂载高危目录的情况下，建议使用低权限账号启动，否则可能会造成宿主机高危文件被破坏。
子路径	请输入存储卷的子路径，将存储卷中的某个路径挂载至容器，可以实现在单一Pod中使用同一个存储卷的不同文件夹。如：data，表示容器中挂载路径下的数据会存储在存储卷的data文件夹中。不填写时默认为根路径。
权限	- 只读：只能读容器路径中的数据卷。 - 读写：可修改容器路径中的数据卷，容器迁移时新写入的数据不会随之迁移，会造成数据丢失。

3. 其余信息都配置完成后，单击“创建工作负载”。

   工作负载创建成功后，容器挂载目录下的数据将会持久化保持，您可以参考**验证数据持久化**中的步骤进行验证。

----结束

使用 kubectl 命令行方式

步骤1 使用kubectl连接集群。

步骤2 创建PV。当您的集群中已存在创建完成的PV时，可跳过本步骤。

1. 创建pv-evs.yaml文件。

```
apiVersion: v1
kind: PersistentVolume
metadata:
  annotations:
    pv.kubernetes.io/provisioned-by: everest-csi-provisioner
    everest.io/reclaim-policy: retain-volume-only        # 可选字段，删除PV时可保留底层存储卷
  name: pv-evs    # PV的名称
  labels:
    failure-domain.beta.kubernetes.io/region: <your_region>   # 替换为集群所在的区域
    failure-domain.beta.kubernetes.io/zone: <your_zone>       # 替换为云硬盘所在的可用区
spec:
  accessModes:
    - ReadWriteOnce     # 访问模式，云硬盘必须为ReadWriteOnce
  capacity:
    storage: 10Gi       # 云硬盘的容量，单位为Gi，取值范围 1-32768
  csi:
    driver: disk.csi.everest.io     # 挂载依赖的存储驱动
    fsType: ext4     # 设置文件系统类型，设置为ext4
    volumeHandle: <your_volume_id>   #云硬盘的volumeID
    volumeAttributes:
      everest.io/disk-mode: SCSI          # 云硬盘的磁盘模式，仅支持SCSI
      everest.io/disk-volume-type: SAS     # 云硬盘的类型
      storage.kubernetes.io/csiProvisionerIdentity: everest-csi-provisioner
      everest.io/enterprise-project-id: <your_project_id>  # 可选字段，企业项目ID，如果指定企业项目，则创建PVC时也需要指定相同的企业项目，否则PVC无法绑定PV。
```



```
    everest.io/disk-iops: '3000'      # 可选字段，设置云硬盘的IOPS，仅云硬盘类型为GPSSD2或ESSD2时支持配置
    everest.io/disk-throughput: '125' # 可选字段，设置云硬盘的吞吐量，仅云硬盘类型为GPSSD2时支持配置
  persistentVolumeReclaimPolicy: Delete    # 回收策略
  storageClassName: csi-disk               # 存储类名称，云硬盘必须为csi-disk
```

表 4-5 关键参数说明

参数	是否必选	描述
everest.io/reclaim-policy: retain-volume-only	否	可选字段，目前仅支持配置“retain-volume-only”，回收策略为Delete时生效。 如果回收策略是Delete且当前值设置为“retain-volume-only”删除PVC回收逻辑为：删除PV，保留底层存储卷。
failure-domain.beta.kubernetes.io/region	是	集群所在的区域。 region对应的值请参见**地区和终端节点**。
failure-domain.beta.kubernetes.io/zone	是	创建云硬盘所在的可用区。 可以通过**查询可用区列表**获取所有支持的可用区。
fsType	是	设置文件系统类型，目前仅支持ext4。
volumeHandle	是	云硬盘的volumeID。 **获取方法：**在云服务器控制台，单击左侧栏目树中的“云硬盘 > 磁盘”，单击要对接的云硬盘名称进入详情页，在“概览信息”页签下单击“ID”后的复制图标即可获取云硬盘的volumeID。
everest.io/disk-volume-type	是	云硬盘类型，全大写。 – SAS：高I/O。 – SSD：超高I/O。 – GPSSD：通用型SSD。 – ESSD：极速型SSD。 – GPSSD2：通用型SSD v2，使用时需同时指定everest.io/disk-iops和everest.io/disk-throughput注解。 – ESSD2：极速型SSD v2，使用时需指定everest.io/disk-iops注解。

参数	是否必选	描述
everest.io/disk-iops	否	IOPS值由用户预配置，仅使用通用型SSD v2和极速型SSD v2类型的云硬盘支持设置。 – 通用型SSD v2类型的IOPS范围为3000~128000，最高可配置 500*容量（GiB）。 选择通用型SSD V2，当IOPS大于3000时会收取额外IOPS费用，详情请参见**价格计算器**。 – 极速型SSD v2类型的IOPS范围为100~256000，最高可配置 1000*容量（GiB）。 选择极速型SSD V2，使用IOPS会收取额外IOPS费用，详情请参见**价格计算器**。
everest.io/disk-throughput	否	吞吐量由用户预配置，仅使用通用型SSD v2类型的云硬盘支持设置。 范围为125~1000MiB/s，最高可配置IOPS/4。 吞吐量大于125MiB/s时会收取额外吞吐量费用，详情请参见**价格计算器**。
everest.io/enterprise-project-id	否	可选字段，云硬盘的企业项目ID。如果指定企业项目，则创建PVC时也需要指定相同的企业项目，否则PVC无法绑定PV。 **获取方法**：在云服务器控制台，单击左侧栏目树中的“云硬盘 > 磁盘”，单击要对接的云硬盘名称进入详情页，在“概览信息”页签下找到“管理信息”中的企业项目，单击并进入对应的企业项目控制台，复制对应的ID值即可获取云硬盘所属的企业项目的ID。
persistentVolumeReclaimPolicy	是	支持Delete、Retain回收策略，详情请参见**PV回收策略**。如果数据安全性要求较高，建议使用**Retain**以免误删数据。 **Delete**: – Delete且不设置everest.io/reclaim-policy：删除PVC，PV资源与云硬盘均被删除。 – Delete且设置everest.io/reclaim-policy=retain-volume-only：删除PVC，PV资源被删除，云硬盘资源会保留。 **Retain**：删除PVC，PV资源与底层存储资源均不会被删除，需要手动删除回收。PVC删除后PV资源状态为“已释放（Released）”，不能直接再次被PVC绑定使用。
storageClassName	是	云硬盘存储对应的存储类名称为csi-disk。

2. 执行以下命令，创建PV。
```
kubectl apply -f pv-evs.yaml
```

步骤3 创建PVC。

1. 创建pvc-evs.yaml文件。
```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-evs
  namespace: default
  annotations:
    everest.io/disk-volume-type: SAS    # 云硬盘的类型

    everest.io/enterprise-project-id: <your_project_id>  # 可选字段，企业项目ID，如果指定企业项目，则创建PVC时也需要指定相同的企业项目，否则PVC无法绑定PV。
    everest.io/disk-iops: '3000'      # 可选字段，设置云硬盘的IOPS，仅云硬盘类型为GPSSD2或ESSD2时支持配置
    everest.io/disk-throughput: '125' # 可选字段，设置云硬盘的吞吐量，仅云硬盘类型为GPSSD2时支持配置
  labels:
    failure-domain.beta.kubernetes.io/region: <your_region>   # 替换为集群所在的区域
    failure-domain.beta.kubernetes.io/zone: <your_zone>       # 替换为云硬盘所在的可用区
spec:
  accessModes:
  - ReadWriteOnce              # 云硬盘必须为ReadWriteOnce
  resources:
    requests:
      storage: 10Gi            # 云硬盘大小，取值范围 1-32768，必须和已有PV的storage大小保持一致。
  storageClassName: csi-disk    # StorageClass类型为云硬盘
  volumeName: pv-evs           # PV的名称
```

表 4-6 关键参数说明

参数	是否必选	描述
failure-domain.beta.kubernetes.io/region	是	集群所在的区域。 region对应的值请参见**地区和终端节点**，如“cn-east-3”。
failure-domain.beta.kubernetes.io/zone	是	创建云硬盘所在的可用区。 可以通过**查询可用区列表**获取所有支持的可用区。
storage	是	PVC申请容量，单位为Gi。 必须与**1**中PV的storage大小保持一致。
volumeName	是	PV的名称，必须与**1**中PV的名称一致。
storageClassName	是	存储类名称，必须与**1**中PV的存储类一致。 云硬盘存储对应的存储类名称为csi-disk。

2. 执行以下命令，创建PVC。
```
kubectl apply -f pvc-evs.yaml
```

步骤4 创建应用。

1. 创建web-evs.yaml文件，本示例中将云硬盘挂载至/data路径。
```
apiVersion: apps/v1
kind: StatefulSet
```

```
metadata:
  name: web-evs
  namespace: default
spec:
  replicas: 1            # 使用云硬盘的工作负载副本数必须是1
  selector:
    matchLabels:
      app: web-evs
  serviceName: web-evs   # Headless Service名称
  template:
    metadata:
      labels:
        app: web-evs
    spec:
      containers:
      - name: container-1
        image: nginx:latest
        volumeMounts:
        - name: pvc-disk     #卷名称，需与volumes字段中的卷名称对应
          mountPath: /data  #存储卷挂载的位置
      imagePullSecrets:
        - name: default-secret
      volumes:
        - name: pvc-disk     #卷名称，可自定义
          persistentVolumeClaim:
            claimName: pvc-evs     #已创建的PVC名称
---
apiVersion: v1
kind: Service
metadata:
  name: web-evs   # Headless Service名称
  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. 执行以下命令，创建一个挂载云硬盘存储的应用。

```
kubectl apply -f web-evs.yaml
```

工作负载创建成功后，容器挂载目录下的数据将会持久化保持，您可以参考**验证数据持久化**中的步骤进行验证。

----结束

验证数据持久化

步骤1 查看部署的应用及云硬盘文件。

1. 执行以下命令，查看已创建的Pod。

```
kubectl get pod | grep web-evs
```

预期输出如下：

```
web-evs-0                1/1     Running   0             38s
```

2. 执行以下命令，查看云硬盘是否挂载至/data路径。

```
kubectl exec web-evs-0 -- df | grep data
```

预期输出如下：

```
/dev/sdc            10255636     36888  10202364   0% /data
```

3. 执行以下命令，查看/data路径下的文件。

```
kubectl exec web-evs-0 -- ls /data
```

预期输出如下：

```
lost+found
```

步骤2 执行以下命令，在/data路径下创建static文件。

```
kubectl exec web-evs-0 --  touch /data/static
```

步骤3 执行以下命令，查看/data路径下的文件。

```
kubectl exec web-evs-0 -- ls /data
```

预期输出如下：

```
lost+found
static
```

步骤4 执行以下命令，删除名称为web-evs-0的Pod。

```
kubectl delete pod web-evs-0
```

预期输出如下：

```
pod "web-evs-0" deleted
```

步骤5 删除后，StatefulSet控制器会自动重新创建一个同名副本。执行以下命令，验证/data路径下的文件是否更改。

```
kubectl exec web-evs-0 -- ls /data
```

预期输出如下：

```
lost+found
static
```

static文件仍然存在，则说明云硬盘中的数据可持久化保存。

----结束

相关操作

您还可以执行**表4-7**中的操作。

表 4-7 其他操作

操作	说明	操作步骤
创建存储卷	通过CCE控制台单独创建PV。	1. 在左侧导航栏选择“存储”，在右侧选择“存储卷”页签。单击右上角“创建存储卷PV”，在弹出的窗口中填写存储卷声明参数。 ● 存储卷类型：选择“云硬盘”。 ● 云硬盘：单击“选择云硬盘”，在新页面中勾选满足要求的云硬盘，并单击“确定”。 ● PV名称：输入PV名称，同一集群内的PV名称需唯一。 ● 访问模式：仅支持ReadWriteOnce，表示存储卷可以被一个节点以读写方式挂载，详情请参见**存储卷访问模式**。 ● 回收策略：Delete或Retain，详情请参见**PV回收策略**。 2. 单击“创建”。
扩容云硬盘存储卷	通过CCE控制台快速扩容已挂载的云硬盘。 仅按需计费的云硬盘支持扩容，包周期的云硬盘请单击卷名称，前往存储服务扩容。	1. 在左侧导航栏选择“存储”，在右侧选择“存储卷声明”页签。单击PVC操作列的“更多 > 扩容”。 2. 输入新增容量，并单击“确定”。
事件	查看PVC或PV的事件名称、事件类型、发生次数、Kubernetes事件、首次和最近发生的时间，便于定位问题。	1. 在左侧导航栏选择“存储”，在右侧选择“存储卷声明”或“存储卷”页签。 2. 单击目标实例操作列的“事件”，即可查看1小时内的事件（事件保存时间为1小时）。
查看YAML	可对PVC或PV的YAML文件进行查看、复制和下载。	1. 在左侧导航栏选择“存储”，在右侧选择“存储卷声明”或“存储卷”页签。 2. 单击目标实例操作列的“查看YAML”，即可查看或下载YAML。

4.3.3 通过动态存储卷使用云硬盘

CCE Autopilot集群支持指定存储类（StorageClass），自动创建云硬盘类型的底层存储和对应的存储卷，适用于无可用的底层存储，需要新创建的场景。

前提条件

- 您已经创建好一个集群。
- 如果您需要通过命令行创建，需要使用kubectl连接到集群，详情请参见**通过kubectl连接集群**。

约束与限制

- 目前，只有部分区域支持使用云硬盘，具体请以控制台为准。您可以通过**功能总览**，了解支持该功能的所有区域。
- 云硬盘不支持被多个工作负载、同一个工作负载的多个实例或多个任务使用。创建无状态工作负载时，若使用了EVS云硬盘，建议工作负载只选择一个实例。
- 集群版本要求在v1.27.8-r0、v1.28.6-r0及以上。如果您的集群版本不符合该要求，则需要通过集群升级使用该功能。
- 集群中每个工作负载最多挂载10个云硬盘存储，若挂载数量超过10，可能导致负载运行异常。
- 动态创建云硬盘存储卷时支持添加资源标签，且云硬盘创建完成后无法在集群侧更新资源标签，需要前往云硬盘控制台更新。如果使用已有的云硬盘创建存储卷，也需要在云硬盘控制台添加或更新资源标签。

自动创建并挂载云硬盘和存储卷

通过StorageClass自动创建云硬盘存储和存储卷，并挂载到工作负载中，从而实现持久化存储。本节将介绍两种方式自动创建和挂载云硬盘和存储卷，即控制台方式和kubectl命令行方式。

使用控制台方式

步骤1 登录CCE控制台，单击集群名称进入集群。

步骤2 动态创建存储卷声明和存储卷。

1. 在左侧导航栏选择“存储”，在右侧选择“存储卷声明”页签。单击右上角“创建存储卷声明 PVC”，在弹出的窗口中填写存储卷声明参数。

图 4-5 创建存储卷声明 PVC

参数	描述
存储卷声明类型	本文中选择“云硬盘”。
PVC名称	输入PVC的名称，同一命名空间下的PVC名称需唯一。
命名空间	命名空间是Kubernetes集群中的抽象概念，可以将集群中的资源或对象划分为一个组，且不同命名空间中的数据彼此隔离，您可以根据需要创建并使用命名空间。 集群创建后会默认生成default命名空间，如果没有特殊要求，可以直接选择default命名空间。
创建方式	- 无可用底层存储的场景下，可选择“动态创建”，通过控制台级联创建存储卷声明PVC、存储卷PV和底层存储。 - 已有底层存储的场景下，根据是否已经创建PV可选择“新建存储卷”或“已有存储卷”，静态创建PVC，具体操作请参见**通过静态存储卷使用已有云硬盘**。 本文中选择“动态创建”。
存储类	云硬盘对应的存储类为csi-disk。

参数	描述
存储卷名称前缀（可选）	定义自动创建的底层存储名称，实际创建的底层存储名称为“存储卷名称前缀”与“PVC UID”的拼接组合，如果不填写该参数，默认前缀为“pvc”。 例如，存储卷名称前缀设置为“test”，则实际创建的底层存储名称test-{uid}。
可用区	选择云硬盘的可用区，创建后不支持更换可用区。 本示例中选择“可用区3”。
云硬盘类型	选择云硬盘类型。不同区域、不同可用区支持的云硬盘类型存在差异，请以控制台选项为准。 本示例中选择“高IO”。 **说明** 通用型SSD V2支持自定义设置IOPS和吞吐量，极速型SSD V2支持自定义设置IOPS，设置范围参见**云硬盘性能数据表**。
容量（GiB）	申请的存储卷容量大小。 本示例中设置为“10”。
访问模式	云硬盘类型的存储卷仅支持ReadWriteOnce，表示存储卷可以被一个节点以读写方式挂载，详情请参见**存储卷访问模式**。
加密	选择底层存储是否加密，使用加密时需要选择使用的加密密钥。目前，暂不支持加密。
企业项目	仅支持default、集群所在企业项目或存储类指定的企业项目。
资源标签	通过为资源添加标签，可以对资源进行自定义标记，实现资源的分类。 您可以在TMS中创建“预定义标签”，预定义标签对所有支持标签功能的服务资源可见，通过使用预定义标签可以提升标签创建和迁移效率。具体请参见**创建预定义标签**。 集群会自动创建“CCE-Cluster-ID=<集群ID>”、“CCE-Cluster-Name=<集群名称>”、“CCE-Namespace=<命名空间名称>”的系统标签，您无法自定义修改。 **说明** 云硬盘类型的动态存储卷创建完成后，不支持在CCE侧更新资源标签。如需更新云硬盘的资源标签，请前往云硬盘控制台。

2. 单击“创建”。

   您可以在左侧导航栏中选择“存储”，在“存储卷声明”和“存储卷”页签下查看已经创建的存储卷声明和存储卷。

步骤3 创建应用。

1. 在左侧导航栏中选择“工作负载”，在右侧选择“有状态负载”页签。
2. 单击页面右上角“创建工作负载”，在“容器配置”中选择“数据存储”页签，并单击“添加存储卷 > 已有存储卷声明 (PVC)”。

   本文主要为您介绍存储卷的挂载使用，如**表4-8**，其他参数详情请参见**工作负载**。

本例中将该盘挂载到容器中/data路径下，在该路径下生成的容器数据会存储到云硬盘中。

说明

由于云硬盘为非共享模式，工作负载下多个实例无法同时挂载，会导致实例启动异常。因此挂载云硬盘时，工作负载实例数需为1。

图 4-6 存储卷挂载

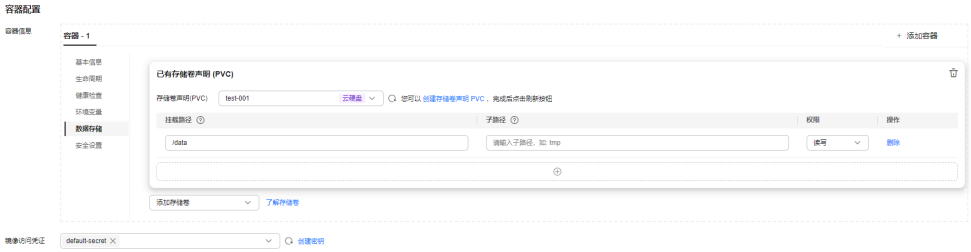


表 4-8 存储卷挂载

参数	参数说明
存储卷声明（PVC）	选择已有的云硬盘存储卷。 云硬盘存储卷无法被多个工作负载重复挂载。
挂载路径	请输入挂载路径，如：/data。 数据存储挂载到容器上的路径。请不要挂载在系统目录下，如“/”、“/var/run”等，会导致容器异常。建议挂载在空目录下，若目录不为空，请确保目录下无影响容器启动的文件，否则文件会被替换，导致容器启动异常，工作负载创建失败。 **须知** 挂载高危目录的情况下，建议使用低权限账号启动，否则可能会造成宿主机高危文件被破坏。
子路径	请输入存储卷的子路径，将存储卷中的某个路径挂载至容器，可以实现在单一Pod中使用同一个存储卷的不同文件夹。如：data，表示容器中挂载路径下的数据会存储在存储卷的data文件夹中。不填写时默认为根路径。
权限	- 只读：只能读容器路径中的数据卷。 - 读写：可修改容器路径中的数据卷，容器迁移时新写入的数据不会随之迁移，会造成数据丢失。

3. 其余信息都配置完成后，单击“创建工作负载”。

   工作负载创建成功后，容器挂载目录下的数据将会持久化保持，您可以参考**验证数据持久化**中的步骤进行验证。

----结束

使用 kubectl 命令行方式

步骤1 使用kubectl连接集群。

步骤2 使用StorageClass动态创建PVC及PV。

1. 创建pvc-evs-auto.yaml文件。

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-evs-auto
  namespace: default
  annotations:
    everest.io/disk-volume-type: SAS    # 云硬盘的类型

    everest.io/enterprise-project-id: <your_project_id>  # 可选字段，企业项目ID，如果指定企业项目，则创建PVC时也需要指定相同的企业项目，否则PVC无法绑定PV
    everest.io/disk-volume-tags: '{"key1":"value1","key2":"value2"}'  # 可选字段，用户自定义资源标签
    everest.io/disk-iops: '3000'      # 可选字段，设置云硬盘的IOPS，仅云硬盘类型为GPSSD2或ESSD2时支持配置
    everest.io/disk-throughput: '125' # 可选字段，设置云硬盘的吞吐量，仅云硬盘类型为GPSSD2时支持配置
    everest.io/csi.volume-name-prefix: test  # 可选字段，定义自动创建的底层存储名称前缀
  labels:
    failure-domain.beta.kubernetes.io/region: <your_region>   # 替换为集群所在的区域
    failure-domain.beta.kubernetes.io/zone: <your_zone>       # 替换为云硬盘所在的可用区
spec:
  accessModes:
  - ReadWriteOnce              # 云硬盘必须为ReadWriteOnce
  resources:
    requests:
      storage: 10Gi            # 云硬盘大小，取值范围 1-32768
  storageClassName: csi-disk    # StorageClass类型为云硬盘
```

表 4-9 关键参数说明

参数	是否必选	描述
failure-domain.beta.kubernetes.io/region	是	集群所在的区域。 region对应的值请参见**地区和终端节点**，如“cn-east-3”。
failure-domain.beta.kubernetes.io/zone	是	创建云硬盘所在的可用区。 可以通过**查询可用区列表**获取所有支持的可用区。
everest.io/disk-volume-type	是	云硬盘类型，全大写。不同区域、不同可用区支持的云硬盘类型存在差异，请以控制台选项为准。 – SAS：高I/O。 – SSD：超高I/O。 – GPSSD：通用型SSD。 – ESSD：极速型SSD。 – GPSSD2：通用型SSD v2，使用时需同时指定everest.io/disk-iops和everest.io/disk-throughput注解。 – ESSD2：极速型SSD v2，使用时需指定everest.io/disk-iops注解。

参数	是否必选	描述
everest.io/disk-iops	否	IOPS值由用户预配置，仅使用通用型SSD v2和极速型SSD v2类型的云硬盘支持设置。 – 通用型SSD v2类型的IOPS范围为3000~128000，最高可配置 500*容量（GiB）。 选择通用型SSD V2，当IOPS大于3000时会收取额外IOPS费用，详情请参见**价格计算器**。 – 极速型SSD v2类型的IOPS范围为100~256000，最高可配置 1000*容量（GiB）。 选择极速型SSD V2，使用IOPS会收取额外IOPS费用，详情请参见**价格计算器**。
everest.io/disk-throughput	否	吞吐量由用户预配置，仅使用通用型SSD v2类型的云硬盘支持设置。 范围为125~1000MiB/s，最高可配置IOPS/4。 吞吐量大于125MiB/s时会收取额外吞吐量费用，详情请参见**价格计算器**。
everest.io/enterprise-project-id	否	可选字段，云硬盘的企业项目ID。如果指定企业项目，则创建PVC时也需要指定相同的企业项目，否则PVC无法绑定PV。 **获取方法**：在企业项目管理控制台，单击要对接的企业项目名称，复制企业项目ID值即可。
everest.io/disk-volume-tags	否	可选字段，通过为资源添加标签，可以对资源进行自定义标记，实现资源的分类。 您可以在TMS中创建“预定义标签”，预定义标签对所有支持标签功能的服务资源可见，通过使用预定义标签可以提升标签创建和迁移效率。具体请参见**创建预定义标签**。 CCE服务会自动创建“CCE-Cluster-ID=<集群ID>”、“CCE-Cluster-Name=<集群名称>”、“CCE-Namespace=<命名空间名称>”的系统标签，您无法自定义修改。
everest.io/csi.volume-name-prefix	否	可选字段，定义自动创建的底层存储名称，实际创建的底层存储名称为“存储卷名称前缀”与“PVC UID”的拼接组合，如果不填写该参数，默认前缀为“pvc”。 取值范围：参数值长度为1~26，且必须是小写字母、数字、中划线，不能以中划线开头或结尾。 例如，存储卷名称前缀设置为“test”，则实际创建的底层存储名称test-{uid}。

参数	是否必选	描述
storage	是	PVC申请容量，单位为Gi，取值范围为1-32768。
storageClassName	是	云硬盘存储对应的存储类名称为csi-disk。

2. 执行以下命令，创建PVC。
```
kubectl apply -f pvc-evs-auto.yaml
```

步骤3 创建应用。

1. 创建web-evs-auto.yaml文件，本示例中将云硬盘挂载至/data路径。
```
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名称
  template:
    metadata:
      labels:
        app: web-evs-auto
    spec:
      containers:
      - name: container-1
        image: nginx:latest
        volumeMounts:
        - name: pvc-disk    #卷名称，需与volumes字段中的卷名称对应
          mountPath: /data  #存储卷挂载的位置
      imagePullSecrets:
        - name: default-secret
      volumes:
        - name: pvc-disk    #卷名称，可自定义
          persistentVolumeClaim:
            claimName: pvc-evs-auto    #已创建的PVC名称
---
apiVersion: v1
kind: Service
metadata:
  name: web-evs-auto   # Headless Service名称
  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. 执行以下命令，创建一个挂载云硬盘存储的应用。
```
kubectl apply -f web-evs-auto.yaml
```

工作负载创建成功后，容器挂载目录下的数据将会持久化保持，您可以参考**验证数据持久化**中的步骤进行验证。

----结束

验证数据持久化

步骤1 查看部署的应用及云硬盘文件。

1. 执行以下命令，查看已创建的Pod。
   ```
   kubectl get pod | grep web-evs-auto
   ```
   预期输出如下：
   ```
   web-evs-auto-0               1/1     Running   0             38s
   ```
2. 执行以下命令，查看云硬盘是否挂载至/data路径。
   ```
   kubectl exec web-evs-auto-0 -- df | grep data
   ```
   预期输出如下：
   ```
   /dev/sdc            10255636     36888  10202364   0% /data
   ```
3. 执行以下命令，查看/data路径下的文件。
   ```
   kubectl exec web-evs-auto-0 -- ls /data
   ```
   预期输出如下：
   ```
   lost+found
   ```

步骤2 执行以下命令，在/data路径下创建static文件。

```
kubectl exec web-evs-auto-0 --  touch /data/static
```

步骤3 执行以下命令，查看/data路径下的文件。

```
kubectl exec web-evs-auto-0 -- ls /data
```

预期输出如下：

```
lost+found
static
```

步骤4 执行以下命令，删除名称为web-evs-auto-0的Pod。

```
kubectl delete pod web-evs-auto-0
```

预期输出如下：

```
pod "web-evs-auto-0" deleted
```

步骤5 删除后，StatefulSet控制器会自动重新创建一个同名副本。执行以下命令，验证/data路径下的文件是否更改。

```
kubectl exec web-evs-auto-0 -- ls /data
```

预期输出如下：

```
lost+found
static
```

static文件仍然存在，则说明云硬盘中的数据可持久化保存。

----结束

相关操作

您还可以执行**表4-10**中的操作。

表 4-10 其他操作

操作	说明	操作步骤
扩容云硬盘存储卷	通过CCE控制台快速扩容已挂载的云硬盘。	1. 在左侧导航栏选择“存储”，在右侧选择“存储卷声明”页签。单击PVC操作列的“更多 > 扩容”。 2. 输入新增容量，并单击“确定”。
事件	查看PVC或PV的事件名称、事件类型、发生次数、Kubernetes事件、首次和最近发生的时间，便于定位问题。	1. 在左侧导航栏选择“存储”，在右侧选择“存储卷声明”或“存储卷”页签。 2. 单击目标实例操作列的“事件”，即可查看1小时内的事件（事件保存时间为1小时）。
查看YAML	可对PVC或PV的YAML文件进行查看、复制和下载。	1. 在左侧导航栏选择“存储”，在右侧选择“存储卷声明”或“存储卷”页签。 2. 单击目标实例操作列的“查看YAML”，即可查看或下载YAML。

4.3.4 在有状态负载中动态挂载云硬盘存储

使用场景

动态挂载仅可在**创建有状态负载（StatefulSet）**时使用，通过卷声明模板（**volumeClaimTemplates**字段）实现，并依赖于StorageClass的动态创建PV能力。在多实例的有状态负载中，动态挂载可以为每一个Pod关联一个独有的PVC及PV，当Pod被重新调度后，仍然能够根据该PVC名称挂载原有的数据。而在无状态工作负载的普通挂载方式中，当存储支持多点挂载（ReadWriteMany）时，工作负载下的多个Pod会被挂载到同一个底层存储中。

说明

Kubernetes不允许在更新StatefulSet时添加或删除volumeClaimTemplates字段，您只能在创建StatefulSet时设置volumeClaimTemplates。

前提条件

- 您已经创建好一个集群。
- 如果您需要通过命令行创建，需要使用kubectl连接到集群，详情请参见**通过kubectl连接集群**。

约束与限制

- 目前，只有部分区域支持使用云硬盘，具体请以控制台为准。您可以通过**功能总览**，了解支持该功能的所有区域。
- 集群版本要求在v1.27.8-r0、v1.28.6-r0及以上。如果您的集群版本不符合该要求，则需要通过集群升级使用该功能。
- 集群中每个工作负载最多挂载10个云硬盘存储，若挂载数量超过10，可能导致负载运行异常。

动态创建并挂载云硬盘存储

基于StorageClass的动态创建PV的能力，为有状态工作负载的每个Pod分配独有的PVC和PV，保证重新调度后数据的持久性。本节将介绍两种方式动态创建和挂载云硬盘存储，即控制台方式和kubectl命令行方式。

使用控制台方式

本文主要为您介绍存储卷的动态挂载使用，其他参数详情请参见**创建有状态负载（StatefulSet）**。

步骤1 登录CCE控制台，单击集群名称进入集群。

步骤2 在左侧导航栏中选择“工作负载”，在右侧选择“有状态负载”页签。

步骤3 单击页面右上角“创建工作负载”，在“容器配置”中选择“数据存储”页签，并单击“添加存储卷 > 动态挂载 (VolumeClaimTemplate)”。

步骤4 单击“创建存储卷声明 PVC”，在弹出窗口中填写存储卷声明参数。

参数填写完成后，单击“创建”。

图 4-7 创建存储卷声明 PVC

参数	描述
存储卷声明类型	本文中选择“云硬盘”。
PVC名称	输入PVC的名称。创建后将根据实例数自动增加后缀，格式为<自定义PVC名称>-<序号>，例如example-0。
命名空间	命名空间是Kubernetes集群中的抽象概念，可以将集群中的资源或对象划分为一个组，且不同命名空间中的数据彼此隔离，您可以根据需要创建并使用命名空间。 集群创建后会默认生成default命名空间，如果没有特殊要求，可以直接选择default命名空间。
创建方式	可选择“动态创建”，通过控制台级联创建存储卷声明PVC、存储卷PV和底层存储。
存储类	云硬盘对应的存储类为csi-disk。
存储卷名称前缀（可选）	定义自动创建的底层存储名称，实际创建的底层存储名称为“存储卷名称前缀”与“PVC UID”的拼接组合，如果不填写该参数，默认前缀为“pvc”。 例如，存储卷名称前缀设置为“test”，则实际创建的底层存储名称test-{uid}。
可用区	选择云硬盘的可用区，创建后不支持更换可用区。 本示例中选择“可用区3”。
云硬盘类型	选择云硬盘类型。不同区域、不同可用区支持的云硬盘类型存在差异，请以控制台选项为准。 本示例中选择“高IO”。 **说明** 通用型SSD V2支持自定义设置IOPS和吞吐量，极速型SSD V2支持自定义设置IOPS，设置范围参见**云硬盘性能数据表**。
容量（GiB）	申请的存储卷容量大小。 本示例中设置为“10”。
访问模式	云硬盘类型的存储卷仅支持ReadWriteOnce，表示存储卷可以被一个节点以读写方式挂载，详情请参见**存储卷访问模式**。
加密	选择底层存储是否加密，使用加密时需要选择使用的加密密钥。目前，暂不支持加密。
企业项目	仅支持default、集群所在企业项目或存储类指定的企业项目。

参数	描述
资源标签	通过为资源添加标签，可以对资源进行自定义标记，实现资源的分类。 您可以在资源标签管理服务中创建“预定义标签”，预定义标签对所有支持标签功能的服务资源可见，通过使用预定义标签可以提升标签创建和迁移效率。具体请参见**创建预定义标签**。 CCE服务会自动创建“CCE-Cluster-ID=<集群ID>”、“CCE-Cluster-Name=<集群名称>”、“CCE-Namespace=<命名空间名称>”的系统标签，您无法自定义修改。 **说明** 云硬盘类型的动态存储卷创建完成后，不支持在CCE侧更新资源标签。如需更新云硬盘的资源标签，请前往云硬盘控制台。

步骤5 填写挂载路径。

本例中将该盘挂载到容器中/data路径下，在该路径下生成的容器数据会存储到云硬盘中。

图 4-8 存储卷挂载

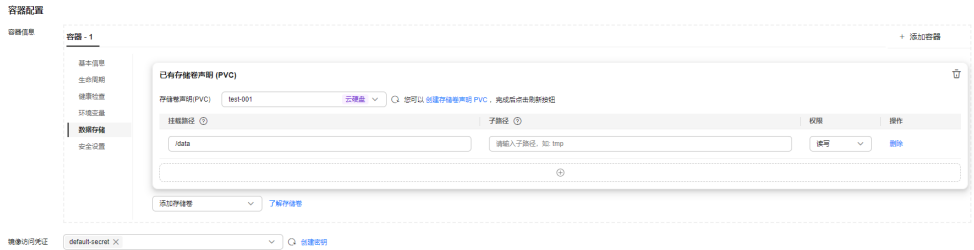


表 4-11 存储卷挂载

参数	参数说明
挂载路径	请输入挂载路径，如：/data。 数据存储挂载到容器上的路径。请不要挂载在系统目录下，如“/”、“/var/run”等，会导致容器异常。建议挂载在空目录下，若目录不为空，请确保目录下无影响容器启动的文件，否则文件会被替换，导致容器启动异常，工作负载创建失败。 **须知** 挂载高危目录的情况下 ，建议使用低权限账号启动，否则可能会造成宿主机高危文件被破坏。
子路径	请输入存储卷的子路径，将存储卷中的某个路径挂载至容器，可以实现在单一Pod中使用同一个存储卷的不同文件夹。如：data，表示容器中挂载路径下的数据会存储在存储卷的data文件夹中。不填写时默认为根路径。
权限	• 只读：只能读容器路径中的数据卷。 • 读写：可修改容器路径中的数据卷，容器迁移时新写入的数据不会随之迁移，会造成数据丢失。

步骤6 其余信息都配置完成后，单击“创建工作负载”。工作负载创建成功后，容器挂载目录下的数据将会持久化保持，您可以参考**验证数据持久化**中的步骤进行验证。

----结束

使用 kubectl 命令行方式

步骤1 使用kubectl连接集群。

步骤2 创建statefulset-evs.yaml文件，本示例中将云硬盘挂载至/data路径。

```
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          # 需与volumeClaimTemplates字段中的名称对应
              mountPath: /data        # 存储卷挂载的位置
      imagePullSecrets:
        - name: default-secret
  serviceName: statefulset-evs        # Headless Service名称
  replicas: 2
  volumeClaimTemplates:
    - apiVersion: v1
      kind: PersistentVolumeClaim
      metadata:
        name: pvc-disk
        namespace: default
        annotations:
          everest.io/disk-volume-type: SAS    # 云硬盘的类型

          everest.io/enterprise-project-id: <your_project_id>  # 可选字段，企业项目ID，如果指定企业项目，则创建PVC时也需要指定相同的企业项目，否则PVC无法绑定PV。
          everest.io/disk-volume-tags: '{"key1":"value1","key2":"value2"}' # 可选字段，用户自定义资源标签
          everest.io/disk-iops: '3000'      # 可选字段，设置云硬盘的IOPS，仅云硬盘类型为GPSSD2或ESSD2时支持配置
          everest.io/disk-throughput: '125' # 可选字段，设置云硬盘的吞吐量，仅云硬盘类型为GPSSD2时支持配置
          everest.io/csi.volume-name-prefix: test  # 可选字段，定义自动创建的底层存储名称前缀
        labels:
          failure-domain.beta.kubernetes.io/region: <your_region>   # 替换为集群所在的区域
          failure-domain.beta.kubernetes.io/zone: <your_zone>       # 替换为云硬盘所在的可用区
      spec:
        accessModes:
          - ReadWriteOnce              # 云硬盘必须为ReadWriteOnce
        resources:
          requests:
            storage: 10Gi              # 云硬盘大小，取值范围 1-32768
        storageClassName: csi-disk     # StorageClass类型为云硬盘
---
apiVersion: v1
kind: Service
metadata:
  name: statefulset-evs   # Headless Service名称
  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
```

表 4-12 关键参数说明

参数	是否必选	描述
failure-domain.beta.kubernetes.io/region	是	集群所在的区域。 region对应的值请参见**地区和终端节点**，如“cn-east-3”。
failure-domain.beta.kubernetes.io/zone	是	创建云硬盘所在的可用区。 可以通过**查询可用区列表**获取所有支持的可用区。
everest.io/disk-volume-type	是	云硬盘类型，全大写。不同区域、不同可用区支持的云硬盘类型存在差异，请以控制台选项为准。 ● SAS：高I/O。 ● SSD：超高I/O。 ● GPSSD：通用型SSD。 ● ESSD：极速型SSD。 ● GPSSD2：通用型SSD v2，使用时需同时指定everest.io/disk-iops和everest.io/disk-throughput注解。 ● ESSD2：极速型SSD v2，使用时需指定everest.io/disk-iops注解。
everest.io/disk-iops	否	IOPS值由用户预配置，仅使用通用型SSD v2和极速型SSD v2类型的云硬盘支持设置。 ● 通用型SSD v2类型的IOPS范围为3000~128000，最高可配置 500*容量（GiB）。 选择通用型SSD V2，当IOPS大于3000时会收取额外IOPS费用，详情请参见**价格计算器**。 ● 极速型SSD v2类型的IOPS范围为100~256000，最高可配置 1000*容量（GiB）。 选择极速型SSD V2，使用IOPS会收取额外IOPS费用，详情请参见**价格计算器**。

参数	是否必选	描述
everest.io/disk-throughput	否	吞吐量由用户预配置，仅使用通用型SSD v2类型的云硬盘支持设置。 范围为125~1000MiB/s，最高可配置 IOPS/4。 吞吐量大于125MiB/s时会收取额外吞吐量费用，详情请参见**价格计算器**。
everest.io/enterprise-project-id	否	可选字段，云硬盘的企业项目ID。如果指定企业项目，则创建PVC时也需要指定相同的企业项目，否则PVC无法绑定PV。 **获取方法**：在企业项目管理控制台，单击要对接的企业项目名称，复制企业项目ID值即可。
everest.io/disk-volume-tags	否	可选字段，通过为资源添加标签，可以对资源进行自定义标记，实现资源的分类。 您可以在TMS中创建“预定义标签”，预定义标签对所有支持标签功能的服务资源可见，通过使用预定义标签可以提升标签创建和迁移效率。具体请参见**创建预定义标签**。 CCE服务会自动创建“CCE-Cluster-ID=<集群ID>”、“CCE-Cluster-Name=<集群名称>”、“CCE-Namespace=<命名空间名称>”的系统标签，您无法自定义修改。
everest.io/csi.volume-name-prefix	否	可选字段，定义自动创建的底层存储名称，实际创建的底层存储名称为“存储卷名称前缀”与“PVC UID”的拼接组合，如果不填写该参数，默认前缀为“pvc”。 取值范围：参数值长度为1~26，且必须是小写字母、数字、中划线，不能以中划线开头或结尾。 例如，存储卷名称前缀设置为“test”，则实际创建的底层存储名称test-{uid}。
storage	是	PVC申请容量，单位为Gi，取值范围为1-32768。
storageClassName	是	云硬盘存储对应的存储类名称为csi-disk。

步骤3 执行以下命令，创建一个挂载云硬盘存储的应用。

```
kubectl apply -f statefulset-evs.yaml
```

工作负载创建成功后，容器挂载目录下的数据将会持久化保持，您可以参考**验证数据持久化**中的步骤进行验证。

----结束

验证数据持久化

步骤1 查看部署的应用及云硬盘文件。

1. 执行以下命令，查看已创建的Pod。


```
kubectl get pod | grep statefulset-evs
```

预期输出如下：

```
statefulset-evs-0       1/1     Running   0          45s
statefulset-evs-1       1/1     Running   0          28s
```

2. 执行以下命令，查看云硬盘是否挂载至/data路径。

```
kubectl exec statefulset-evs-0 -- df | grep data
```

预期输出如下：

```
/dev/sdd            10255636     36888  10202364   0% /data
```

3. 执行以下命令，查看/data路径下的文件。

```
kubectl exec statefulset-evs-0 -- ls /data
```

预期输出如下：

```
lost+found
```

步骤2 执行以下命令，在/data路径下创建static文件。

```
kubectl exec statefulset-evs-0 --  touch /data/static
```

步骤3 执行以下命令，查看/data路径下的文件。

```
kubectl exec statefulset-evs-0 -- ls /data
```

预期输出如下：

```
lost+found
static
```

步骤4 执行以下命令，删除名称为web-evs-auto-0的Pod。

```
kubectl delete pod statefulset-evs-0
```

预期输出如下：

```
pod "statefulset-evs-0" deleted
```

步骤5 删除后，StatefulSet控制器会自动重新创建一个同名副本。执行以下命令，验证/data路径下的文件是否更改。

```
kubectl exec statefulset-evs-0 -- ls /data
```

预期输出如下：

```
lost+found
static
```

static文件仍然存在，则说明云硬盘中的数据可持久化保存。

----结束

相关操作

您还可以执行表4-13中的操作。

表 4-13 其他操作

操作	说明	操作步骤
扩容云硬盘存储卷	通过CCE控制台快速扩容已挂载的云硬盘。	1. 在左侧导航栏选择“存储”，在右侧选择“存储卷声明”页签。单击PVC操作列的“更多 > 扩容”。 2. 输入新增容量，并单击“确定”。

操作	说明	操作步骤
事件	查看PVC或PV的事件名称、事件类型、发生次数、Kubernetes事件、首次和最近发生的时间，便于定位问题。	1. 在左侧导航栏选择“存储”，在右侧选择“存储卷声明”或“存储卷”页签。 2. 单击目标实例操作列的“事件”，即可查看1小时内的事件（事件保存时间为1小时）。
查看YAML	可对PVC或PV的YAML文件进行查看、复制和下载。	1. 在左侧导航栏选择“存储”，在右侧选择“存储卷声明”或“存储卷”页签。 2. 单击目标实例操作列的“查看YAML”，即可查看或下载YAML。

4.3.5 扩容云硬盘存储卷

当工作负载挂载的云硬盘存储卷空间不足时，您可以通过云硬盘存储卷扩容的方式解决。本文介绍如何通过控制台进行云硬盘存储卷扩容。

前提条件

您已经创建好一个集群，集群版本在v1.27.8-r0、v1.28.6-r0及以上。

操作步骤

步骤1 登录CCE控制台，单击集群名称进入集群。

步骤2 在左侧导航栏选择“存储”，在右侧选择“存储卷声明”页签。单击PVC操作列的“更多 > 扩容”。

图 4-9 扩容云硬盘

步骤3 输入新增容量，并单击“确定”。本示例中，新增容量为10GiB。

说明

磁盘不支持缩容，建议您合理选择扩容容量。

图 4-10 输入扩容容量

步骤4 验证扩容是否成功。返回“存储卷声明”页签，可以看到相应PVC容量由原来的10GiB扩容至20GiB。

----结束

4.3.6 快照与备份

CCE Autopilot集群通过云硬盘EVS服务为您提供快照功能。云硬盘快照简称快照，指云硬盘数据在某个时刻的完整复制或镜像，是一种重要的数据容灾手段。当数据丢失时，可通过快照将数据完整的恢复到快照时间点。

您可以创建快照，快速保存指定时刻云硬盘的数据。同时，您还可以通过快照创建新的云硬盘，新的云硬盘在初始状态就会具有快照中的数据。

前提条件

- 您已经创建好一个集群。
- 如果您需要通过命令行创建，需要使用kubectl连接到集群，详情请参见**通过kubectl连接集群**。

约束与限制

- 目前，只有部分区域支持使用云硬盘，具体请以控制台为准。您可以通过**功能总览**，了解支持该功能的所有区域。
- 集群版本要求在v1.27.8-r0、v1.28.6-r0及以上。如果您的集群版本不符合该要求，则需要通过集群升级使用该功能。
- 基于快照创建的云硬盘，其子类型（普通IO/高IO/超高IO等）、是否加密、磁盘模式（VBD/SCSI）、共享性(非共享/共享)、容量等都要与快照关联母盘保持一致，这些属性查询和设置出来后不能够修改。
- 只有可用或正在使用状态的磁盘能创建快照，且单个磁盘最大支持创建7个快照。
- 加密磁盘的快照数据以加密方式存放，非加密磁盘的快照数据以非加密方式存放。

使用场景

快照功能可以帮助您实现以下需求：

- **日常备份数据**

  通过对云硬盘定期创建快照，实现数据的日常备份，可以应对由于误操作、病毒以及黑客攻击等导致数据丢失或不一致的情况。

- **快速恢复数据**

  更换操作系统、应用软件升级或业务数据迁移等重大操作前，您可以创建一份或多份快照，一旦升级或迁移过程中出现问题，可以通过快照及时将业务恢复到快照创建点的数据状态。

  例如，当由于云服务器A的系统盘A发生故障而无法正常开机时，由于系统盘A已经故障，因此也无法将快照数据回滚至系统盘A。此时您可以使用系统盘A已有的快照新创建一块云硬盘B并挂载至正常运行的云服务器B上，从而云服务器B能够通过云硬盘B读取原系统盘A的数据。

  说明

  当前CCE Autopilot集群提供的快照能力与K8s社区CSI快照功能一致：只支持基于快照创建新云硬盘，不支持将快照回滚到源云硬盘。

- **快速部署多个业务**

  通过同一个快照可以快速创建出多个具有相同数据的云硬盘，从而可以同时为多种业务提供数据资源。例如数据挖掘、报表查询和开发测试等业务。这种方式既保护了原始数据，又能通过快照创建的新云硬盘快速部署其他业务，满足企业对业务数据的多元化需求。

创建快照

使用控制台创建

步骤1 登录CCE控制台。

步骤2 单击集群名称进入集群，在左侧选择“存储”，在右侧选择“快照与备份”页签。

步骤3 单击右上角“创建快照”，在弹出的窗口中设置相关参数。

图 4-11 创建快照

表 4-14 创建快照参数说明

参数	说明
快照名称	请填写快照的名称，例如“disksnap-test1”。
选择存储	选择要创建快照的PVC，仅能选择云硬盘类型PVC，例如选择“pvc”。

步骤4 单击“创建”。在“快照与备份”页签中可以看到新建的快照信息。

图 4-12 新建的快照

----结束

使用YAML创建

```
kind: VolumeSnapshot
apiVersion: snapshot.storage.k8s.io/v1beta1
metadata:
  name: disksnap-test1   # 快照名称
  namespace: default
spec:
  source:
    persistentVolumeClaimName: pvc    # PVC的名称，仅能选择云硬盘类型PVC
  volumeSnapshotClassName: csi-disk-snapclass
```

使用快照创建 PVC

通过快照创建云硬盘PVC时，磁盘类型、磁盘模式、加密属性需和快照源云硬盘保持一致。

使用控制台创建

步骤1 登录CCE控制台。

步骤2 单击集群名称进入集群，在左侧选择“存储”，在右侧选择“快照与备份”页签。

步骤3 找到需要创建PVC的快照，单击“创建存储卷声明 PVC”，并在弹出窗口中设置PVC参数。

图 4-13 利用快照创建 PVC

表 4-15 利用快照创建 PVC 的参数说明

参数	说明
PVC名称	请输入PVC名称，如“pvc-disksnap”。
存储卷名称前缀（可选）	定义自动创建的底层存储名称，实际创建的底层存储名称为“存储卷名称前缀”与“PVC UID”的拼接组合，如果不填写该参数，默认前缀为“pvc”。 例如，存储卷名称前缀设置为“test”，则实际创建的底层存储名称test-{uid}。
资源标签	通过为资源添加标签，可以对资源进行自定义标记，实现资源的分类。 您可以在TMS中创建“预定义标签”，预定义标签对所有支持标签功能的服务资源可见，通过使用预定义标签可以提升标签创建和迁移效率。具体请参见**创建预定义标签**。 CCE服务会自动创建“CCE-Cluster-ID=<集群ID>”、“CCE-Cluster-Name=<集群名称>”、“CCE-Namespace=<命名空间名称>”的系统标签，您无法自定义修改。

步骤4 单击“创建”。选择“存储卷声明”页签，可以看到新建的PVC。

图 4-14 新建的 PVC

----结束

使用YAML创建

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-disksnap    # PVC名称
  namespace: default
  annotations:
    everest.io/disk-volume-type: SSD    # 云硬盘类型，需要与快照源云硬盘保持一致
    everest.io/disk-volume-tags: '{"key1":"value1","key2":"value2"}' # 可选字段，用户自定义资源标签
    everest.io/disk-iops: '3000'     # 可选字段，设置云硬盘的IOPS，仅云硬盘类型为GPSSD2或ESSD2时支持配置
    everest.io/disk-throughput: '125' # 可选字段，设置云硬盘的吞吐量，仅云硬盘类型为GPSSD2时支持配置
  labels:
    failure-domain.beta.kubernetes.io/region: <your_region>   # 替换为云硬盘所在的区域
    failure-domain.beta.kubernetes.io/zone: <your_zone>       # 替换为云硬盘所在的可用区
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
  storageClassName: csi-disk
  dataSource:
    name: disksnap-test1            # 快照的名称
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
```

4.4 文件存储（SFS）

4.4.1 文件存储概述

文件存储介绍

CCE Autopilot支持将文件存储（SFS）创建的存储卷挂载到容器的某一路径下，以满足数据持久化需求，SFS存储卷适用于多读多写的持久化存储，适用大容量扩展以及成本敏感型的业务场景，包括媒体处理、内容管理、大数据分析和分析工作负载程序等。SFS文件系统不适合海量小文件业务，海量小文件业务推荐使用SFS Turbo文件系统。

说明

目前，只有部分区域支持使用文件存储，具体请以控制台为准。您可以通过**功能总览**，了解支持该功能的所有区域。

SFS为用户提供一个完全托管的共享文件存储，能够弹性伸缩至PB规模，具备高可用性和持久性，为海量数据、高带宽型应用提供有力支持。

- **符合标准文件协议：**用户可以将文件系统挂载给服务器，像使用本地文件目录一样。

- **数据共享：**多台服务器可挂载相同的文件系统，数据可以共享操作和访问。
- **私有网络：**数据访问必须在数据中心内部网络中。
- **容量与性能：**单文件系统容量较高（PB级），性能极佳（IO读写时延ms级）。
- **应用场景：**适用于多读多写（ReadWriteMany）场景下的各种工作负载（Deployment/StatefulSet）和普通任务（Job）使用，主要面向高性能计算、媒体处理、内容管理和Web服务、大数据和分析应用程序等场景。

文件存储性能

SFS包括通用文件系统（原SFS 3.0）和SFS容量型两种类型，其中CCE Autopilot集群仅支持使用通用文件系统（原SFS 3.0）。更多关于通用文件系统存储性能的详细介绍，请参见**文件系统类型**。

表 4-16 文件存储性能

参数	通用文件系统（原SFS 3.0）
最大带宽	1.25TB/s
最高IOPS	百万
时延	10ms
最大容量	EB

使用场景

根据使用场景不同，文件存储支持以下挂载方式：

- **通过静态存储卷使用已有文件存储**：即静态创建的方式，需要先使用已有的文件存储创建PV，然后通过PVC在工作负载中挂载存储。适用于已有可用的底层存储或底层存储需要包周期的场景。
- **通过动态存储卷使用文件存储**：即动态创建的方式，无需预先创建文件存储，在创建PVC时通过指定存储类（StorageClass），即可自动创建文件存储和对应的PV对象。适用于无可用的底层存储，需要新创建的场景。

计费说明

- 挂载文件存储类型的存储卷时，通过StorageClass**自动创建**的文件存储默认创建计费模式为“按需计费”，按实际使用的存储容量和时长收费。关于文件存储的价格信息，请参见**文件存储计费说明**。
- 如需使用包周期的文件存储，请**使用已有的文件存储**进行挂载。

4.4.2 通过静态存储卷使用已有文件存储

文件存储（SFS）是一种可共享访问，并提供按需扩展的高性能文件系统（NAS），适用大容量扩展以及成本敏感型的业务场景。CCE Autopilot集群仅支持挂载通用文件系统（原SFS3.0）存储卷，即文件存储的底层存储必须为通用文件系统（原SFS3.0）。本文介绍如何使用已有的文件存储静态创建PV和PVC，并在工作负载中实现数据持久化与共享性。

前提条件

- 如果您需要通过命令行创建，需要使用kubectl连接到集群，详情请参见**通过kubectl连接集群**。
- 您已经创建好一个通用文件系统（原SFS3.0），并且通用文件系统（原SFS3.0）与集群在同一个VPC内。
- 您已经配置通用文件系统（原SFS3.0）所需的VPC终端节点，具体配置操作请参见**配置VPC终端节点**。

约束与限制

- 目前，只有部分区域支持使用文件存储，具体请以控制台为准。您可以通过**功能总览**，了解支持该功能的所有区域。
- 支持多个PV挂载同一个通用文件系统（原SFS3.0），但有如下限制：
  - 多个不同的PVC/PV使用同一个底层通用文件系统（原SFS3.0）卷时，如果挂载至同一Pod使用，会因为PV的volumeHandle参数值相同导致无法为Pod挂载所有PVC，出现Pod无法启动的问题，请避免该使用场景。
  - PV中persistentVolumeReclaimPolicy参数建议设置为Retain，否则可能存在一个PV删除时级联删除底层卷，其他关联这个底层卷的PV会由于底层存储被删除导致使用出现异常。
  - 重复用底层存储时，建议在应用层做好多读多写的隔离保护，防止产生的数据覆盖和丢失。
- 使用通用文件系统（原SFS 3.0）存储卷时，挂载点不支持修改属组和权限。
- 使用通用文件系统（原SFS 3.0）存储卷时，创建、删除PVC和PV过程中可能存在时延，实际计费时长请以SFS侧创建、删除时刻为准。
- 通用文件系统（原SFS 3.0）存储卷在Delete回收策略下暂不支持自动回收文件存储卷。当删除PV/PVC时，需要手动删除所有文件后才可以正常删除PV和PVC。

通过控制台使用已有文件存储

步骤1 登录CCE控制台，单击集群名称进入集群。

步骤2 静态创建存储卷声明和存储卷。

1. 在左侧导航栏选择“存储”，在右侧选择“存储卷声明”页签。单击右上角“创建存储卷声明”，在弹出的窗口中填写存储卷声明参数。

参数	描述
存储卷声明类型	本文中选择“文件存储”。
PVC名称	输入PVC的名称，同一命名空间下的PVC名称需唯一。
创建方式	- 已有底层存储的场景下，根据是否已经创建PV可选择“新建存储卷”或“已有存储卷”来静态创建PVC。 - 无可用底层存储的场景下，可选择“动态创建”，具体操作请参见**通过动态存储卷使用文件存储**。 本文示例中选择“新建存储卷”，可通过控制台同时创建PV及PVC。

参数	描述
关联存储卷[a]	选择集群中已有的PV卷，需要提前创建PV，请参考**相关操作**中的“创建存储卷”操作。 本文示例中无需选择。
文件存储[b]	单击“选择文件存储”，您可以在新页面中勾选满足要求的文件存储，并单击“确定”。 **说明** 当前仅支持选择通用文件系统（原SFS 3.0）的文件存储。
PV名称[b]	输入PV名称，同一集群内的PV名称需唯一。
访问模式[b]	文件存储类型的存储卷仅支持ReadWriteMany，表示存储卷可以被多个节点以读写方式挂载，详情请参见**存储卷访问模式**。
回收策略[b]	您可以选择Delete或Retain，用于指定删除PVC时底层存储的回收策略，详情请参见**PV回收策略**。 **说明** 多个PV使用同一个底层存储时建议使用**Retain**，避免级联删除底层卷。
挂载参数[b]	输入挂载参数键值对，详情请参见**设置文件存储挂载参数**。

说明

a：创建方式选择“已有存储卷”时可设置。

b：创建方式选择“新建存储卷”时可设置。

2. 单击“创建”，将同时为您创建存储卷声明和存储卷。

   您可以在左侧导航栏中选择“存储”，在“存储卷声明”和“存储卷”页签下查看已经创建的存储卷声明和存储卷。

步骤3 创建应用。

1. 在左侧导航栏中选择“工作负载”，在右侧选择“无状态负载”页签。
2. 单击页面右上角“创建工作负载”，在“容器配置”中选择“数据存储”页签，并单击“添加存储卷 > 已有存储卷声明 (PVC)”。

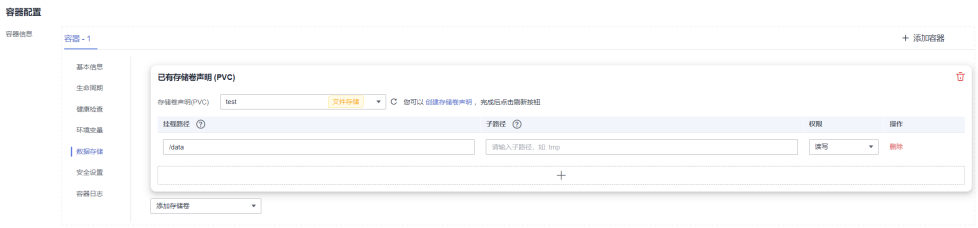
   本文主要为您介绍存储卷的挂载使用，如**表4-17**，其他参数详情请参见**工作负载**。

表 4-17 存储卷挂载

参数	参数说明
存储卷声明（PVC）	选择已有的文件存储卷。

参数	参数说明
挂载路径	请输入挂载路径，如：/tmp。 数据存储挂载到容器上的路径。请不要挂载在系统目录下，如“/”、“/var/run”等，会导致容器异常。建议挂载在空目录下，若目录不为空，请确保目录下无影响容器启动的文件，否则文件会被替换，导致容器启动异常，工作负载创建失败。 **须知** 挂载高危目录的情况下，建议使用低权限账号启动，否则可能会造成宿主机高危文件被破坏。
子路径	请输入子路径，如：tmp，表示容器中挂载路径下的数据会存储在卷的tmp文件夹中。 使用子路径挂载本地磁盘，实现在单一Pod中重复使用同一个Volume。不填写时默认为根。
权限	- 只读：只能读容器路径中的数据卷。 - 读写：可修改容器路径中的数据卷，容器迁移时新写入的数据不会随之迁移，会造成数据丢失。

本例中将该存储卷挂载到容器中/data路径下，在该路径下生成的容器数据会存储到文件存储中。



3. 其余信息都配置完成后，单击“创建工作负载”。

   工作负载创建成功后，容器挂载目录下的数据将会持久化保持，您可以参考**验证数据持久化及共享性**中的步骤进行验证。

----结束

通过 kubectl 命令行使用已有文件存储

步骤1 使用kubectl连接集群。

步骤2 创建PV。

1. 创建pv-sfs.yaml文件。

```
apiVersion: v1
kind: PersistentVolume
metadata:
  annotations:
    pv.kubernetes.io/provisioned-by: everest-csi-provisioner
    everest.io/reclaim-policy: retain-volume-only      # 可选字段，删除PV，保留底层存储卷
  name: pv-sfs    # PV的名称
spec:
  accessModes:
  - ReadWriteMany      # 访问模式，文件存储必须为ReadWriteMany
```

```
capacity:
  storage: 1Gi      # 文件存储容量大小
csi:
  driver: nas.csi.everest.io    # 挂载依赖的存储驱动
  fsType: nfs
  volumeHandle: <your_volume_id>   # 使用通用文件系统（原SFS 3.0）时填写文件存储名称
  volumeAttributes:
    everest.io/share-export-location: <your_location>  #文件存储的共享路径
    storage.kubernetes.io/csiProvisionerIdentity: everest-csi-provisioner
    everest.io/sfs-version: sfs3.0       # 使用通用文件系统（原SFS 3.0）
persistentVolumeReclaimPolicy: Retain    # 回收策略
storageClassName: csi-sfs              # 存储类名称：csi-sfs表示使用通用文件系统（原SFS 3.0）
mountOptions: []                       # 挂载参数
```

表 4-18 关键参数说明

参数	是否必选	描述
everest.io/reclaim-policy: retain-volume-only	否	可选字段 目前仅支持配置“retain-volume-only” 回收策略为Delete时生效。如果回收策略是Delete且当前值设置为“retain-volume-only”删除PVC回收逻辑为：删除PV，保留底层存储卷。
volumeHandle	是	使用通用文件系统（原SFS 3.0）文件存储：填写文件存储的名称。
everest.io/share-export-location	是	文件存储的共享路径。 共享路径格式如下： *{your_sfs30_name}*.sfs3.*{region}*.myhuaweicloud.com:/*{your_sfs30_name}*
mountOptions	是	挂载参数。 不设置时默认配置为如下配置，具体说明请参见**设置文件存储挂载参数**。 mountOptions: - vers=3 - timeo=600 - nolock - hard
persistentVolumeReclaimPolicy	是	支持Delete、Retain回收策略，详情请参见**PV回收策略**。多个PV使用同一个文件存储时建议使用**Retain**，避免级联删除底层卷。 **Delete**: – Delete且不设置everest.io/reclaim-policy：删除PVC，PV资源与文件存储均被删除。 – Delete且设置everest.io/reclaim-policy=retain-volume-only：删除PVC，PV资源被删除，文件存储资源会保留。 **Retain**：删除PVC，PV资源与底层存储资源均不会被删除，需要手动删除回收。PVC删除后PV资源状态为“已释放（Released）”，不能直接再次被PVC绑定使用。

参数	是否必选	描述
storage	是	PVC申请容量，单位为Gi。 对文件存储来说，此处仅为校验需要（不能为空和0），设置的大小不起作用，此处设定为固定值1Gi。
storageClassName	是	存储类名称，支持配置csi-sfs，表示使用通用文件系统（原SFS 3.0）文件存储。

2. 执行以下命令，创建PV。
```
kubectl apply -f pv-sfs.yaml
```

步骤3 创建PVC。

1. 创建pvc-sfs.yaml文件。
```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-sfs
  namespace: default
  annotations:
    volume.beta.kubernetes.io/storage-provisioner: everest-csi-provisioner
spec:
  accessModes:
  - ReadWriteMany                 # 文件存储必须为ReadWriteMany
  resources:
    requests:
      storage: 1Gi                # 文件存储大小
  storageClassName: csi-sfs       # 存储类名称，必须与PV的存储类一致。
  volumeName: pv-sfs              # PV的名称
```

表 4-19 关键参数说明

参数	是否必选	描述
storage	是	PVC申请容量，单位为Gi。 必须和已有PV的storage大小保持一致。
storageClassName	是	存储类名称，必须与1中PV的存储类一致。 – csi-sfs：表示使用通用文件系统（原SFS 3.0）文件存储。
volumeName	是	PV的名称，必须与1中PV的名称一致。

2. 执行以下命令，创建PVC。
```
kubectl apply -f pvc-sfs.yaml
```

步骤4 创建应用。

1. 创建web-demo.yaml文件，本示例中将文件存储挂载至/data路径。
```
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    #卷名称，需与volumes字段中的卷名称对应
        mountPath: /data  #存储卷挂载的位置
    imagePullSecrets:
      - name: default-secret
    volumes:
      - name: pvc-sfs-volume    #卷名称，可自定义
        persistentVolumeClaim:
          claimName: pvc-sfs    #已创建的PVC名称
```

2. 执行以下命令，创建一个挂载文件存储的应用。

```
kubectl apply -f web-demo.yaml
```

工作负载创建成功后，容器挂载目录下的数据将会持久化保持，您可以参考**验证数据持久化及共享性**中的步骤进行验证。

----结束

验证数据持久化及共享性

步骤1 查看部署的应用及文件。

1. 执行以下命令，查看已创建的Pod。

```
kubectl get pod | grep web-demo
```

预期输出如下：

```
web-demo-846b489584-mjhm9   1/1     Running   0          46s
web-demo-846b489584-wvv5s   1/1     Running   0          46s
```

2. 依次执行以下命令，查看Pod的/data路径下的文件。

```
kubectl exec web-demo-846b489584-mjhm9 -- ls /data
kubectl exec web-demo-846b489584-wvv5s -- ls /data
```

两个Pod均无返回结果，说明/data路径下无文件。

步骤2 执行以下命令，在/data路径下创建static文件。

```
kubectl exec web-demo-846b489584-mjhm9 --  touch /data/static
```

步骤3 执行以下命令，查看/data路径下的文件。

```
kubectl exec web-demo-846b489584-mjhm9 -- ls /data
```

预期输出如下：

```
static
```

步骤4 验证数据持久化

1. 执行以下命令，删除名称为web-demo-846b489584-mjhm9的Pod。

```
kubectl delete pod web-demo-846b489584-mjhm9
```

预期输出如下：

```
pod "web-demo-846b489584-mjhm9" deleted
```

删除后，Deployment控制器会自动重新创建一个副本。

2. 执行以下命令，查看已创建的Pod。

```
kubectl get pod | grep web-demo
```


预期输出如下，web-demo-846b489584-d4d4j为新建的Pod：

```
web-demo-846b489584-d4d4j   1/1     Running   0          110s
web-demo-846b489584-wvv5s   1/1     Running   0          7m50s
```

3. 执行以下命令，验证新建的Pod中/data路径下的文件是否更改。

```
kubectl exec web-demo-846b489584-d4d4j -- ls /data
```

预期输出如下：

```
static
```

static文件仍然存在，则说明数据可持久化保存。

步骤5 验证数据共享性

1. 执行以下命令，查看已创建的Pod。

```
kubectl get pod | grep web-demo
```

预期输出如下：

```
web-demo-846b489584-d4d4j   1/1     Running   0          7m
web-demo-846b489584-wvv5s   1/1     Running   0          13m
```

2. 执行以下命令，在任意一个Pod的/data路径下创建share文件。本例中选择名为web-demo-846b489584-d4d4j的Pod。

```
kubectl exec web-demo-846b489584-d4d4j --  touch /data/share
```

并查看该Pod中/data路径下的文件。

```
kubectl exec web-demo-846b489584-d4d4j -- ls /data
```

预期输出如下：

```
share
static
```

3. 由于写入share文件的操作未在名为web-demo-846b489584-wvv5s的Pod中执行，在该Pod中查看/data路径下是否存在文件以验证数据共享性。

```
kubectl exec web-demo-846b489584-wvv5s -- ls /data
```

预期输出如下：

```
share
static
```

如果在任意一个Pod中的/data路径下创建文件，其他Pod下的/data路径下均存在此文件，则说明两个Pod共享一个存储卷。

----结束

相关操作

您还可以执行**表4-20**中的操作。

表 4-20 其他操作

操作	说明	操作步骤
创建存储卷	通过CCE控制台单独创建PV。	1. 在左侧导航栏选择“存储”，在右侧选择“存储卷”页签。单击右上角“创建存储卷”，在弹出的窗口中填写存储卷声明参数。 • 存储卷类型：选择“文件存储”。 • 文件存储：单击“选择文件存储”，在新页面中勾选满足要求的文件存储，并单击“确定”。 • PV名称：输入PV名称，同一集群内的PV名称需唯一。 • 访问模式：仅支持ReadWriteMany，表示存储卷可以被多个节点以读写方式挂载，详情请参见**存储卷访问模式**。 • 回收策略：Delete或Retain，详情请参见**PV回收策略**。 **说明** 多个PV使用同一个底层存储时建议使用**Retain**，避免级联删除底层卷。 • 挂载参数：输入挂载参数键值对，详情请参见**设置文件存储挂载参数**。 2. 单击“创建”。
事件	查看PVC或PV的事件名称、事件类型、发生次数、Kubernetes事件、首次和最近发生的时间，便于定位问题。	1. 在左侧导航栏选择“存储”，在右侧选择“存储卷声明”或“存储卷”页签。 2. 单击目标实例操作列的“事件”，即可查看1小时内的事件（事件保存时间为1小时）。
查看YAML	可对PVC或PV的YAML文件进行查看、复制和下载。	1. 在左侧导航栏选择“存储”，在右侧选择“存储卷声明”或“存储卷”页签。 2. 单击目标实例操作列的“查看YAML”，即可查看或下载YAML。

4.4.3 通过动态存储卷使用文件存储

CCE Autopilot集群仅支持挂载通用文件系统（原SFS3.0）存储卷，即文件存储的底层存储必须为通用文件系统（原SFS3.0）。本文介绍如何通过存储类动态创建PV和PVC，并在工作负载中实现数据持久化与共享性。

前提条件

- 如果您需要通过命令行创建，需要使用kubectl连接到集群，详情请参见**通过kubectl连接集群**。
- 您已经配置通用文件系统（原SFS3.0）所需的VPC终端节点，具体配置操作请参见**配置VPC终端节点**。

约束与限制

- 目前，只有部分区域支持使用文件存储，具体请以控制台为准。您可以通过**功能总览**，了解支持该功能的所有区域。
- 使用通用文件系统（原SFS 3.0）存储卷时，挂载点不支持修改属组和权限。
- 使用通用文件系统（原SFS 3.0）存储卷时，创建、删除PVC和PV过程中可能存在时延，实际计费时长请以SFS侧创建、删除时刻为准。
- 通用文件系统（原SFS 3.0）存储卷使用Delete回收策略时，需要挂载文件系统手动删除所有文件后才可以正常删除PV和PVC。

通过控制台自动创建文件存储

步骤1 登录CCE控制台，单击集群名称进入集群。

步骤2 动态创建存储卷声明和存储卷。

1. 在左侧导航栏选择“存储”，在右侧选择“存储卷声明”页签。单击右上角“创建存储卷声明”，在弹出的窗口中填写存储卷声明参数。

参数	描述
存储卷声明类型	本文中选择“文件存储”。
PVC名称	输入PVC的名称，同一命名空间下的PVC名称需唯一。
创建方式	- 无可用底层存储的场景下，可选择“动态创建”，通过控制台级联创建存储卷声明PVC、存储卷PV和底层存储。 - 已有底层存储的场景下，根据是否已经创建PV可选择“新建存储卷”或“已有存储卷”，静态创建PVC，具体操作请参见**通过静态存储卷使用已有文件存储**。 本文中选择“动态创建”。
存储类	文件存储对应的存储类为csi-sfs。
访问模式	文件存储类型的存储卷仅支持ReadWriteMany，表示存储卷可以被多个节点以读写方式挂载，详情请参见**存储卷访问模式**。

2. 单击“创建”，将同时为您创建存储卷声明和存储卷。

   您可以在左侧导航栏中选择“存储”，在“存储卷声明”和“存储卷”页签下查看已经创建的存储卷声明和存储卷。

步骤3 创建应用。

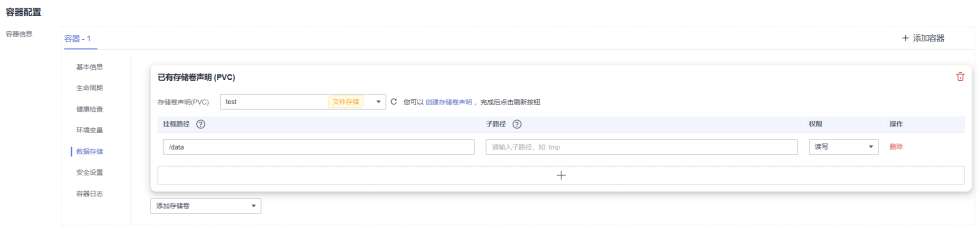
1. 在左侧导航栏中选择“工作负载”，在右侧选择“无状态负载”页签。
2. 单击页面右上角“创建工作负载”，在“容器配置”中选择“数据存储”页签，并单击“添加存储卷 > 已有存储卷声明 (PVC)”。

   本文主要为您介绍存储卷的挂载使用，如**表4-21**，其他参数详情请参见**创建工作负载**。

表 4-21 存储卷挂载

参数	参数说明
存储卷声明（PVC）	选择已有的文件存储卷。
挂载路径	请输入挂载路径，如：/tmp。 数据存储挂载到容器上的路径。请不要挂载在系统目录下，如“/”、“/var/run”等，会导致容器异常。建议挂载在空目录下，若目录不为空，请确保目录下无影响容器启动的文件，否则文件会被替换，导致容器启动异常，工作负载创建失败。 **须知** 挂载高危目录的情况下，建议使用低权限账号启动，否则可能会造成宿主机高危文件被破坏。
子路径	请输入子路径，如：tmp，表示容器中挂载路径下的数据会存储在卷的tmp文件夹中。 使用子路径挂载本地磁盘，实现在单一Pod中重复使用同一个Volume。不填写时默认为根。
权限	- 只读：只能读容器路径中的数据卷。 - 读写：可修改容器路径中的数据卷，容器迁移时新写入的数据不会随之迁移，会造成数据丢失。

本例中将该存储卷挂载到容器中/data路径下，在该路径下生成的容器数据会存储到文件存储中。



3. 其余信息都配置完成后，单击“创建工作负载”。

   工作负载创建成功后，容器挂载目录下的数据将会持久化保持，您可以参考**验证数据持久化及共享性**中的步骤进行验证。

----结束

使用 kubectl 自动创建文件存储

步骤1 使用kubectl连接集群。

步骤2 使用StorageClass动态创建PVC及PV。

1. 创建pvc-sfs-auto.yaml文件。

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-sfs-auto
  namespace: default
```

```
  annotations:
    everest.io/crypt-key-id: <your_key_id>      # 可选字段，密钥的id，使用该密钥加密文件存储
    everest.io/crypt-alias: sfs/default         # 可选字段，密钥的名称，创建加密卷时必须提供该字段
    everest.io/crypt-domain-id: <your_domain_id>  # 可选字段，指定加密卷所属租户的ID，创建加密卷时必须提供该字段
spec:
  accessModes:
    - ReadWriteMany             # 文件存储必须为ReadWriteMany
  resources:
    requests:
      storage: 1Gi              # 文件存储大小
  storageClassName: csi-sfs     # StorageClass类型为文件存储
```

表 4-22 关键参数说明

参数	是否必选	描述
storage	是	PVC申请容量，单位为Gi。 对文件存储来说，此处仅为校验需要（不能为空和0），设置的大小不起作用，此处设定为固定值1Gi。
storageClassName	是	存储类名称。 – csi-sfs：表示使用通用文件系统（原SFS 3.0）文件存储。
everest.io/crypt-key-id	否	当文件存储是加密卷时为必填，填写创建文件存储时选择的加密密钥ID，可使用自定义密钥或名为“sfs/default”的云硬盘默认密钥。 **获取方法**：在数据加密控制台，找到需要加密的密钥，复制密钥ID值即可。
everest.io/crypt-alias	否	密钥的名称，创建加密卷时必须提供该字段。 **获取方法**：在数据加密控制台，找到需要加密的密钥，复制密钥名称即可。
everest.io/crypt-domain-id	否	指定加密卷所属租户的ID，创建加密卷时必须提供该字段。 **获取方法**：在云服务器控制台，鼠标悬浮至右上角的用户名称并单击“我的凭证”，复制账号ID即可。

2. 执行以下命令，创建PVC。

```
kubectl apply -f pvc-sfs-auto.yaml
```

步骤3 创建应用。

1. 创建web-demo.yaml文件，本示例中将文件存储挂载至/data路径。

```
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    #卷名称，需与volumes字段中的卷名称对应
        mountPath: /data  #存储卷挂载的位置
    imagePullSecrets:
      - name: default-secret
    volumes:
      - name: pvc-sfs-volume    #卷名称，可自定义
        persistentVolumeClaim:
          claimName: pvc-sfs-auto    #已创建的PVC名称
```

2. 执行以下命令，创建一个挂载文件存储的应用。

```
kubectl apply -f web-demo.yaml
```

工作负载创建成功后，容器挂载目录下的数据将会持久化保持，您可以参考**验证数据持久化及共享性**中的步骤进行验证。

----结束

验证数据持久化及共享性

步骤1 查看部署的应用及文件。

1. 执行以下命令，查看已创建的Pod。

```
kubectl get pod | grep web-demo
```

预期输出如下：

```
web-demo-846b489584-mjhm9   1/1     Running   0          46s
web-demo-846b489584-wvv5s   1/1     Running   0          46s
```

2. 依次执行以下命令，查看Pod的/data路径下的文件。

```
kubectl exec web-demo-846b489584-mjhm9 -- ls /data
kubectl exec web-demo-846b489584-wvv5s -- ls /data
```

两个Pod均无返回结果，说明/data路径下无文件。

步骤2 执行以下命令，在/data路径下创建static文件。

```
kubectl exec web-demo-846b489584-mjhm9 --  touch /data/static
```

步骤3 执行以下命令，查看/data路径下的文件。

```
kubectl exec web-demo-846b489584-mjhm9 -- ls /data
```

预期输出如下：

```
static
```

步骤4 验证数据持久化

1. 执行以下命令，删除名称为web-demo-846b489584-mjhm9的Pod。

```
kubectl delete pod web-demo-846b489584-mjhm9
```

预期输出如下：

```
pod "web-demo-846b489584-mjhm9" deleted
```

删除后，Deployment控制器会自动重新创建一个副本。

2. 执行以下命令，查看已创建的Pod。

```
kubectl get pod | grep web-demo
```

预期输出如下，web-demo-846b489584-d4d4j为新建的Pod：

```
web-demo-846b489584-d4d4j   1/1     Running   0          110s
web-demo-846b489584-wvv5s   1/1     Running   0          7m50s
```

3. 执行以下命令，验证新建的Pod中/data路径下的文件是否更改。
```
kubectl exec web-demo-846b489584-d4d4j -- ls /data
```
预期输出如下：
```
static
```
static文件仍然存在，则说明数据可持久化保存。

步骤5 验证数据共享性

1. 执行以下命令，查看已创建的Pod。
```
kubectl get pod | grep web-demo
```
预期输出如下：
```
web-demo-846b489584-d4d4j   1/1     Running   0          7m
web-demo-846b489584-wvv5s   1/1     Running   0          13m
```
2. 执行以下命令，在任意一个Pod的/data路径下创建share文件。本例中选择名为web-demo-846b489584-d4d4j的Pod。
```
kubectl exec web-demo-846b489584-d4d4j --  touch /data/share
```
并查看该Pod中/data路径下的文件。
```
kubectl exec web-demo-846b489584-d4d4j -- ls /data
```
预期输出如下：
```
share
static
```
3. 由于写入share文件的操作未在名为web-demo-846b489584-wvv5s的Pod中执行，在该Pod中查看/data路径下是否存在文件以验证数据共享性。
```
kubectl exec web-demo-846b489584-wvv5s -- ls /data
```
预期输出如下：
```
share
static
```
如果在任意一个Pod中的/data路径下创建文件，其他Pod下的/data路径下均存在此文件，则说明两个Pod共享一个存储卷。

----结束

相关操作

您还可以执行表4-23中的操作。

表 4-23 其他操作

操作	说明	操作步骤
事件	查看PVC或PV的事件名称、事件类型、发生次数、Kubernetes事件、首次和最近发生的时间，便于定位问题。	1. 在左侧导航栏选择“存储”，在右侧选择“存储卷声明”或“存储卷”页签。 2. 单击目标实例操作列的“事件”，即可查看1小时内的事件（事件保存时间为1小时）。
查看YAML	可对PVC或PV的YAML文件进行查看、复制和下载。	1. 在左侧导航栏选择“存储”，在右侧选择“存储卷声明”或“存储卷”页签。 2. 单击目标实例操作列的“查看YAML”，即可查看或下载YAML。

4.4.4 设置文件存储挂载参数

本章节主要介绍如何设置文件存储的挂载参数。您可以在PV中设置挂载参数，然后通过PVC绑定PV，也可以在StorageClass中设置挂载参数，然后使用StorageClass创建PVC，动态创建出的PV会默认带有StorageClass中设置的挂载参数。

文件存储挂载参数

CCE Autopilot在挂载文件存储时默认设置了如表4-24所示的参数。

表 4-24 文件存储挂载参数

参数	参数值	描述
keep-original-ownership	无需填写	表示是否保留文件挂载点的ownership。 • 默认为不添加该参数，此时挂载文件存储时将会默认把挂载点的ownership修改为root:root。 • 如添加该参数，挂载文件存储时将保持文件系统原有的ownership。
vers	3	文件系统版本，目前只支持NFSv3。取值：3
nolock	无需填写	选择是否使用NLM协议在服务器上锁文件。当选择nolock选项时，锁对于同一主机的应用有效，对不同主机不受锁的影响。
timeo	600	NFS客户端重传请求前的等待时间(单位为0.1秒)。建议值：600。
hard/soft	无需填写	挂载方式类型。 • 取值为hard，即使用硬连接方式，若NFS请求超时，则客户端一直重新请求直至成功。 • 取值为soft，即软挂载方式挂载系统，若NFS请求超时，则客户端向调用程序返回错误。 默认为hard。
sharecache/nosharecache	无需填写	设置客户端并发挂载同一文件系统时数据缓存和属性缓存的共享方式。设置为sharecache时，多个挂载共享共享同一缓存。设为nosharecache时，每个挂载各有一个缓存。默认为sharecache。 **说明** 设置nosharecache禁用共享缓存会对性能产生一定影响。每次挂载都会重新获取挂载信息，会增加与NFS服务器的通信开销和NFS客户端的内存消耗，同时同客户端设置nosharecache存在cache不一致的风险。因此，应该根据具体情况进行权衡，以确定是否需要使用nosharecache选项。

除了以上参数外，您还可以设置其他的文件存储挂载参数，具体请参见**挂载NFS文件系统到云服务器（Linux）**。

在 PV 中设置挂载参数

在PV中设置挂载参数可以通过mountOptions字段实现，如下所示，mountOptions支持挂载的字段请参见**文件存储挂载参数**。

步骤1 使用kubectl连接集群，详情请参见**通过kubectl连接集群**。

步骤2 在PV中设置挂载参数，示例如下：

```
apiVersion: v1
kind: PersistentVolume
metadata:
  annotations:
    pv.kubernetes.io/provisioned-by: everest-csi-provisioner
    everest.io/reclaim-policy: retain-volume-only      # 可选字段，删除PV，保留底层存储卷
  name: pv-sfs
spec:
  accessModes:
  - ReadWriteMany      # 访问模式，文件存储必须为ReadWriteMany
  capacity:
    storage: 1Gi       # 文件存储容量大小
  csi:
    driver: nas.csi.everest.io    # 挂载依赖的存储驱动
    fsType: nfs
    volumeHandle: <your_volume_id>   # 使用通用文件系统（原SFS 3.0）时填写文件存储名称
    volumeAttributes:
      everest.io/share-export-location: <your_location> # 文件存储的共享路径
      storage.kubernetes.io/csiProvisionerIdentity: everest-csi-provisioner
      everest.io/sfs-version: sfs3.0       # 使用通用文件系统（原SFS 3.0）
  persistentVolumeReclaimPolicy: Retain    # 回收策略
  storageClassName: csi-sfs                # 存储类名称：csi-sfs表示使用通用文件系统（原SFS 3.0）
  mountOptions:                            # 挂载参数
  - vers=3
  - nolock
  - timeo=600
  - hard
```

步骤3 PV创建后，可以创建PVC关联PV，然后在工作负载的容器中挂载，具体操作步骤请参见**通过静态存储卷使用已有文件存储**。

步骤4 验证挂载参数是否生效。

本例中将PVC挂载至使用nginx:latest镜像的工作负载，并通过**mount -l**命令查看挂载参数是否生效。

1. 查看已挂载文件存储的Pod，本文中的示例工作负载名称为web-sfs。

   ```
   kubectl get pod | grep web-sfs
   ```

   回显如下：

   ```
   web-sfs-***   1/1     Running   0          23m
   ```

2. 执行以下命令查看挂载参数，其中web-sfs-***为示例Pod。

   ```
   kubectl exec -it web-sfs-*** -- mount -l | grep nfs
   ```

   若回显中的挂载信息与设置的挂载参数一致，说明挂载参数设置成功。

   ```
   <您的共享路径> on /data type nfs
   (rw,relatime,vers=3,rsize=1048576,wsize=1048576,namlen=255,hard,nolock,noresvport,proto=tcp,
   timeo=600,retrans=2,sec=sys,mountaddr=**.**.**.**,mountvers=3,mountport=2050,mountproto=tcp
   ,local_lock=all,addr=**.**.**.**)
   ```

----结束

在 StorageClass 中设置挂载参数

在StorageClass中设置挂载参数同样可以通过mountOptions字段实现，如下所示，mountOptions支持挂载的字段请参见**文件存储挂载参数**。

步骤1 使用kubectl连接集群，详情请参见**通过kubectl连接集群**。

步骤2 创建自定义的StorageClass，示例如下：

```
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>   # 集群所在VPC的ID
  everest.io/sfs-version: sfs3.0              # 表示使用通用文件系统（原SFS 3.0）文件存储
reclaimPolicy: Delete
volumeBindingMode: Immediate
mountOptions:                          # 挂载参数
- vers=3
- nolock
- timeo=600
- hard
```

步骤3 StorageClass设置好后，就可以使用这个StorageClass创建PVC，动态创建出的PV会默认带有StorageClass中设置的挂载参数，具体操作步骤请参见**通过动态存储卷使用文件存储**。

步骤4 验证挂载参数是否生效。

本例中将PVC挂载至使用nginx:latest镜像的工作负载，并通过**mount -l**命令查看挂载参数是否生效。

1. 查看已挂载文件存储的Pod，本文中的示例工作负载名称为web-sfs。
   ```
   kubectl get pod | grep web-sfs
   ```
   回显如下：
   ```
   web-sfs-***   1/1     Running   0          23m
   ```
2. 执行以下命令查看挂载参数，其中web-sfs-***为示例Pod。
   ```
   kubectl exec -it web-sfs-*** -- mount -l | grep nfs
   ```
   若回显中的挂载信息与设置的挂载参数一致，说明挂载参数设置成功。
   ```
   <您的共享路径> on /data type nfs
   (rw,relatime,vers=3,rsize=1048576,wsize=1048576,namlen=255,hard,nolock,noresvport,proto=tcp,
   timeo=600,retrans=2,sec=sys,mountaddr=**.**.**.**,mountvers=3,mountport=2050,mountproto=tcp
   ,local_lock=all,addr=**.**.**.**)
   ```

----结束

4.5 极速文件存储（SFS Turbo）

4.5.1 极速文件存储概述

极速文件存储介绍

CCE Autopilot集群支持将极速文件存储（SFS Turbo）及其子目录创建的存储卷挂载到容器的某一路径下，以满足数据持久化的需求。极速文件存储具有按需申请，快速供给，弹性扩展，方便灵活等特点，适用于海量小文件业务，例如DevOps、容器微服务、企业办公等应用场景。

> **说明**
>
> 目前，只有部分区域支持使用SFS Turbo，具体请以控制台为准。您可以通过**功能总览**，了解支持该功能的所有区域。

SFS Turbo为用户提供一个完全托管的共享文件存储，能够弹性伸缩至320TB规模，具备高可用性和持久性，为海量的小文件、低延迟高IOPS型应用提供有力支持。

- **符合标准文件协议：**用户可以将文件系统挂载给服务器，像使用本地文件目录一样。
- **数据共享：**多台服务器可挂载相同的文件系统，数据可以共享操作和访问。
- **私有网络：**数据访问必须在数据中心内部网络中。
- **安全隔离：**直接使用云上现有IaaS服务构建独享的云文件存储，为租户提供数据隔离保护和IOPS性能保障。
- **应用场景：**适用于多读多写（ReadWriteMany）场景下的各种工作负载（Deployment/StatefulSet）和普通任务（Job）使用，主要面向高性能网站、日志存储、DevOps、企业办公等场景。

极速文件存储性能

关于极速文件存储的性能参数，请参考**文件系统类型**。

使用场景

极速文件存储支持以下挂载方式：

- **通过静态存储卷使用已有极速文件存储**：即静态创建的方式，需要先使用已有的文件存储创建PV，然后通过PVC在工作负载中挂载存储。
- **通过动态存储卷创建SFS Turbo子目录（推荐）**：CCE Autopilot集群支持在创建PVC时动态创建SFS Turbo子目录，并支持配置子目录容量大小，实现不同工作负载共享使用SFS Turbo。

计费说明

极速文件存储不具备动态创建能力，只能挂载已创建完成的极速文件存储，您可根据需求选择按需计费或者包年包月套餐。关于极速文件存储的价格详情，请参见**极速文件存储计费说明**。

4.5.2 通过静态存储卷使用已有极速文件存储

极速文件存储（SFS Turbo）是一种具备高可用性和持久性的共享文件系统，适合海量的小文件、低延迟高IOPS的应用。使用本方案时，您需要手动创建底层存储资源SFS Turbo和存储卷PV。使用PV描述已有的底层存储资源，然后通过创建PVC在Pod中使用底层存储资源。本文介绍如何使用已有的极速文件存储（使用子目录和不涉及子目录）静态创建PV和PVC，并在工作负载中实现数据持久化与共享性。

前提条件

- 您已经创建好一个状态可用的SFS Turbo，并且SFS Turbo与集群在同一个VPC内。
- 如果您需要通过命令行创建，需要使用kubectl连接到集群，详情请参见**通过kubectl连接集群**。

约束与限制

- 目前，只有部分区域支持使用SFS Turbo，具体请以控制台为准。您可以通过**功能总览**，了解支持该功能的所有区域。
- 支持多个PV挂载同一个SFS Turbo，但有如下限制：
  - 多个不同的PVC/PV使用同一个底层SFS Turbo卷时，如果挂载至同一Pod使用，会因为PV的volumeHandle参数值相同导致无法为Pod挂载所有PVC，出现Pod无法启动的问题，请避免该使用场景。
  - PV中persistentVolumeReclaimPolicy参数建议设置为Retain，否则可能存在一个PV删除时级联删除底层卷，其他关联这个底层卷的PV会由于底层存储被删除导致使用出现异常。
  - 重复用底层存储时，建议在应用层做好多读多写的隔离保护，防止产生的数据覆盖和丢失。
- 使用SFS Turbo的子目录要求集群版本在v1.27.9-r0、v1.28.7-r0及以上。如果您的集群版本不符合该要求，则需要通过集群升级使用该功能。
- 对SFS Turbo类型的存储来说，删除集群或删除PVC时不会回收包周期的SFS Turbo资源，您需要在SFS Turbo控制台中自行回收。

通过控制台使用已有极速文件存储

步骤1 登录CCE控制台，单击集群名称进入集群。

步骤2 静态创建存储卷声明和存储卷。

1. 在左侧导航栏选择“存储”，在右侧选择“存储卷声明”页签。单击右上角“创建存储卷声明”，在弹出的窗口中填写存储卷声明参数。

参数	描述
存储卷声明类型	本文中选择“极速文件存储”。
PVC名称	输入PVC的名称，同一命名空间下的PVC名称需唯一。
命名空间	表示PVC所在命名空间。 - 若左侧导航栏选择“存储”后，页面上方“命名空间”为“全部命名空间”或“非系统命名空间”，则此处可以选择合适的命名空间。 - 若左侧导航栏选择“存储”后，页面上方“命名空间”已指定一个特定的非系统的命名空间（如default），则此处固定为该命名空间。
创建方式	根据是否已经创建PV可选择“新建存储卷”或“已有存储卷”来静态创建PVC。 本文示例中选择“新建存储卷”，可通过控制台同时创建PV及PVC。
关联存储卷[a]	选择集群中已有的PV卷，需要提前创建PV，请参考**相关操作**中的“创建存储卷”操作。 本文示例中无需选择。
极速文件存储[b]	单击“选择极速文件存储”，您可以在新页面中勾选满足要求的极速文件存储，并单击“确定”。

参数	描述
子目录[b]	选择是否使用子目录创建PV。 若选择“是”，请填写子目录绝对路径，例如/a/b。此时PV所关联的底层存储为所选定的极速文件存储实例中的一个子目录，您需要确保该子目录已存在且可用。
PV名称[b]	输入PV名称，同一集群内的PV名称需唯一。
访问模式[b]	极速文件存储类型的存储卷仅支持ReadWriteMany，表示存储卷可以被多个节点以读写方式挂载，详情请参见**存储卷访问模式**。
回收策略[b]	不使用子目录创建PV时，仅支持Retain，表示删除PVC时PV不会被同时删除，详情请参见**PV回收策略**。选择使用子目录创建PV时，支持选择Delete。
子目录回收策略[b]	删除PVC时是否保留子目录，该参数需与**PV回收策略**配合使用，**当PV回收策略为"Delete"时支持配置**。 – 保留：删除PVC，PV会被删除，但**PV关联的子目录会被保留**。 – 删除：删除PVC，**PV及其关联的子目录均会被删除**。
挂载参数[b]	输入挂载参数键值对，详情请参见**设置极速文件存储挂载参数**。

说明

a：创建方式选择“已有存储卷”时可设置。

b：创建方式选择“新建存储卷”时可设置。

2. 单击“创建”，将同时为您创建存储卷声明和存储卷。

   您可以在左侧导航栏中选择“存储”，在“存储卷声明”和“存储卷”页签下查看已经创建的存储卷声明和存储卷。

步骤3 创建工作负载。

1. 在左侧导航栏中选择“工作负载”，在右侧选择“无状态负载”页签。
2. 单击页面右上角“创建工作负载”，在“容器配置”中选择“数据存储”页签，并单击“添加存储卷 > 已有存储卷声明 (PVC)”。

   本文主要为您介绍存储卷的挂载使用，如**表4-25**，其他参数详情请参见**工作负载**。

表 4-25 存储卷挂载

参数	参数说明
存储卷声明（PVC）	选择已有的极速文件存储卷。

参数	参数说明
挂载路径	请输入挂载路径，如：/tmp。 数据存储挂载到容器上的路径。请不要挂载在系统目录下，如“/”、“/var/run”等，会导致容器异常。建议挂载在空目录下，若目录不为空，请确保目录下无影响容器启动的文件，否则文件会被替换，导致容器启动异常，工作负载创建失败。 **须知** 挂载高危目录的情况下，建议使用低权限账号启动，否则可能会造成宿主机高危文件被破坏。
子路径	表示将存储卷中的某个子目录挂载到容器的指定路径，而不是将整个存储卷挂载到容器的路径，可以实现在单一Pod中使用同一个存储卷的不同文件夹。如：tmp，表示容器中挂载路径下的数据会存储在存储卷的tmp文件夹中。不填写时默认为根路径。
权限	- 只读：只能读容器路径中的数据卷。 - 读写：可修改容器路径中的数据卷，容器迁移时新写入的数据不会随之迁移，会造成数据丢失。

本例中将该存储卷挂载到容器中/data路径下，在该路径下生成的容器数据会存储到极速文件存储中。

图 4-15 挂载存储卷

3. 其余信息都配置完成后，单击“创建工作负载”。

   工作负载创建成功后，容器挂载目录下的数据将会持久化保持，您可以参考**验证数据持久化及共享性**中的步骤进行验证。

----结束

通过 kubectl 命令行使用已有极速文件存储

您可以根据不同的使用场景选择是否使用子目录。

使用已有极速文件存储（不涉及子目录）

使用整个极速文件存储，不涉及子目录。

步骤1 使用kubectl连接集群。

步骤2 创建PV。

1. 创建pv-sfsturbo.yaml文件。

```
apiVersion: v1
kind: PersistentVolume
metadata:
  annotations:
    pv.kubernetes.io/provisioned-by: everest-csi-provisioner
  name: pv-sfsturbo    # PV的名称
spec:
  accessModes:
  - ReadWriteMany      # 访问模式，极速文件存储必须为ReadWriteMany
  capacity:
    storage: 500Gi      # 极速文件存储容量大小
  csi:
    driver: sfsturbo.csi.everest.io    # 挂载依赖的存储驱动
    fsType: nfs
    volumeHandle: <your_volume_id>  # 极速文件存储的ID
    volumeAttributes:
      everest.io/share-export-location: <your_location>  # 极速文件存储的共享路径
      everest.io/enterprise-project-id: <your_project_id>  # 极速文件存储的项目ID
      storage.kubernetes.io/csiProvisionerIdentity: everest-csi-provisioner
  persistentVolumeReclaimPolicy: Retain    # 回收策略
  storageClassName: csi-sfsturbo          # SFS Turbo存储类名称
  mountOptions: []                        # 挂载参数
```

表 4-26 关键参数说明

参数	是否必选	描述
volumeHandle	是	填写极速文件存储的ID。 **获取方法**：在CCE控制台，单击顶部的“服务列表 > 存储 > 弹性文件服务”，并选择SFS Turbo。在列表中单击对应的SFS Turbo文件存储名称，在详情页中复制“ID”后的内容即可。
everest.io/share-export-location	是	极速文件存储的共享路径。 **获取方法**：在CCE控制台，单击顶部的“服务列表 > 存储 > 弹性文件服务”，选择SFS Turbo，在弹性文件服务列表中可以看到“挂载地址”列，即为文件存储的共享路径。
everest.io/enterprise-project-id	否	极速文件存储的项目ID。当您使用的企业项目为default时，此字段固定填写为'0'。 **获取方法**：在弹性文件服务控制台，单击左侧栏目树中的“SFS Turbo”，单击要对接的SFS Turbo名称进入详情页，在“基本信息”页签下找到企业项目，单击并进入对应的企业项目控制台，复制对应的ID值即可。

参数	是否必选	描述
mountOptions	否	挂载参数。 不设置时默认配置为如下配置，具体说明请参见**设置极速文件存储挂载参数**。 mountOptions: - vers=3 - timeo=600 - nolock - hard
persistentVolumeReclaimPolicy	是	仅支持Retain回收策略，详情请参见**PV回收策略**。 **Retain**：删除PVC，PV资源与底层存储资源均不会被删除，需要手动删除回收。PVC删除后PV资源状态为“已释放（Released）”，不能直接再次被PVC绑定使用。
storage	是	PVC申请容量，单位为Gi。
storageClassName	是	极速文件存储对应的存储类名称为csi-sfsturbo。

2. 执行以下命令，创建PV。
```
kubectl apply -f pv-sfsturbo.yaml
```

步骤3 创建PVC。

1. 创建pvc-sfsturbo.yaml文件。
```
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>  # 极速文件存储的项目ID
spec:
  accessModes:
  - ReadWriteMany                # 极速文件存储必须为ReadWriteMany
  resources:
    requests:
      storage: 500Gi             # 极速文件存储大小
  storageClassName: csi-sfsturbo      # SFS Turbo存储类名称，必须与PV的存储类一致
  volumeName: pv-sfsturbo    # PV的名称
```

表 4-27 关键参数说明

参数	是否必选	描述
everest.io/enterprise-project-id	否	极速文件存储的项目ID。当您使用的企业项目为default时，此字段固定填写'0'。 **获取方法**：在弹性文件服务控制台，单击左侧栏目树中的“SFS Turbo”，单击要对接的SFS Turbo名称进入详情页，在“基本信息”页签下找到企业项目，单击并进入对应的企业项目控制台，复制对应的ID值即可。
storage	是	PVC申请容量，单位为Gi。 必须和已有PV的storage大小保持一致。
storageClassName	是	存储类名称，必须与1中PV的存储类一致。 极速文件存储对应的存储类名称为csi-sfsturbo。
volumeName	是	PV的名称，必须与1中PV名称一致。

2. 执行以下命令，创建PVC。
```
kubectl apply -f pvc-sfsturbo.yaml
```

步骤4 创建工作负载。

1. 创建web-demo.yaml文件，本示例中将极速文件存储挂载至/data路径。
```
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    #卷名称，需与volumes字段中的卷名称对应
          mountPath: /data  #存储卷挂载的位置
      imagePullSecrets:
        - name: default-secret
      volumes:
        - name: pvc-sfsturbo-volume    #卷名称，可自定义
          persistentVolumeClaim:
            claimName: pvc-sfsturbo    #已创建的PVC名称
```
2. 执行以下命令，创建一个挂载极速文件存储的工作负载。
```
kubectl apply -f web-demo.yaml
```
工作负载创建成功后，您可以尝试**验证数据持久化及共享性**。

----结束

使用已有极速文件存储的子目录

步骤1 使用kubectl连接集群。

步骤2 创建PV。

1. 创建pv-sfsturbo.yaml文件。

   示例如下：

```
apiVersion: v1
kind: PersistentVolume
metadata:
  annotations:
    pv.kubernetes.io/provisioned-by: everest-csi-provisioner
    everest.io/reclaim-policy: retain-volume-only    # 可选，当使用子目录且回收策略为Delete时使用，表示删除PVC时，PV会被删除，但PV关联的子目录会被保留
  name: pv-sfsturbo    # PV的名称
spec:
  accessModes:
  - ReadWriteMany      # 访问模式，极速文件存储必须为ReadWriteMany
  capacity:
    storage: 500Gi       # 极速文件存储容量大小
  csi:
    driver: sfsturbo.csi.everest.io    # 挂载依赖的存储驱动
    fsType: nfs
    volumeHandle: pv-sfsturbo   # 子目录场景下为pv名称
    volumeAttributes:
      everest.io/share-export-location: <sfsturbo_path>:/<absolute_path>   # 极速文件存储的共享路径+子目录
      everest.io/enterprise-project-id: <your_project_id>  # 极速文件存储的项目ID
      storage.kubernetes.io/csiProvisionerIdentity: everest-csi-provisioner
      everest.io/volume-as: absolute-path   # 可选，表示使用SFS Turbo子目录
  persistentVolumeReclaimPolicy: Retain    # 回收策略，自动创建子目录时支持设置为Delete
  storageClassName: csi-sfsturbo          # SFS Turbo存储类名称
  mountOptions: []                        # 挂载参数
```

表 4-28 关键参数说明

参数	是否必选	描述
volumeHandle	是	使用SFS Turbo子目录创建PV时，填写PV名称。
everest.io/share-export-location	是	极速文件存储子目录的共享路径。 格式为： *{sfsturbo_path}:/{absolute_path}* **获取方法**：在CCE控制台，单击顶部的“服务列表 > 存储 > 弹性文件服务”，选择SFS Turbo，在弹性文件服务列表中可以看到“共享路径”列，即为极速文件存储的共享路径。
everest.io/enterprise-project-id	否	极速文件存储的项目ID。当您使用的企业项目为default时，此字段固定填写'0'。 **获取方法**：在弹性文件服务控制台，单击左侧栏目树中的“SFS Turbo”，单击要对接的SFS Turbo名称进入详情页，在“基本信息”页签下找到企业项目，单击并进入对应的企业项目控制台，复制对应的ID值即可。

参数	是否必选	描述
mountOptions	否	挂载参数。 不设置时默认配置为如下配置，具体说明请参见**设置极速文件存储挂载参数**。 mountOptions: - vers=3 - timeo=600 - nolock - hard
persistentVolumeReclaimPolicy	是	表示回收策略，详情请参见**PV回收策略**。 – Retain：删除PVC，PV资源与底层存储资源均不会被删除，需要手动删除回收。PVC删除后PV资源状态为“已释放（Released）”，不能直接再次被PVC绑定使用。 – Delete：自动创建子目录时支持设置，表示删除PVC时，同时删除PV。
everest.io/reclaim-policy	否	删除PVC时是否保留子目录，该参数需与**PV回收策略**配合使用。仅当PV回收策略为"Delete"时生效，取值如下： – retain-volume-only：表示删除PVC时，PV会被删除，但**PV关联的子目录会被保留**。 – delete：表示删除PVC，**PV及其关联的子目录均会被删除**。 **说明** 删除子目录时，仅删除PVC参数中设置的子目录绝对路径，不会级联删除上层目录。
everest.io/volume-as	否	固定取值为“absolute-path”，表示使用动态创建SFS Turbo子目录。
storage	是	PVC申请容量，单位为Gi。使用子目录时，该参数值无实际意义，仅作校验需要（不能为空和0）。
storageClassName	是	极速文件存储对应的存储类名称为csi-sfsturbo。

2. 执行以下命令，创建PV。
```
kubectl apply -f pv-sfsturbo.yaml
```

步骤3 创建PVC。

1. 创建pvc-sfsturbo.yaml文件。
```
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>  # 极速文件存储的项目ID
spec:
```



```
accessModes:
- ReadWriteMany                # 极速文件存储必须为ReadWriteMany
resources:
  requests:
    storage: 500Gi              # 极速文件存储大小
storageClassName: csi-sfsturbo      # SFS Turbo存储类名称，必须与PV的存储类一致
volumeName: pv-sfsturbo     # PV的名称
```

表 4-29 关键参数说明

参数	是否必选	描述
everest.io/enterprise-project-id	否	极速文件存储的项目ID。当您使用的企业项目为default时，此字段固定填写'0'。 **获取方法**：在弹性文件服务控制台，单击左侧栏目树中的“SFS Turbo”，单击要对接的SFS Turbo名称进入详情页，在“基本信息”页签下找到企业项目，单击并进入对应的企业项目控制台，复制对应的ID值即可。
storage	是	PVC申请容量，单位为Gi。 必须和已有PV的storage大小保持一致。
storageClassName	是	存储类名称，必须与1中PV的存储类一致。 极速文件存储对应的存储类名称为csi-sfsturbo。
volumeName	是	PV的名称，必须与1中PV名称一致。

2. 执行以下命令，创建PVC。

```
kubectl apply -f pvc-sfsturbo.yaml
```

步骤4 创建工作负载。

1. 创建web-demo.yaml文件，本示例中将极速文件存储挂载至/data路径。

```
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     #卷名称，需与volumes字段中的卷名称对应
          mountPath: /data  #存储卷挂载的位置
      imagePullSecrets:
        - name: default-secret
      volumes:
        - name: pvc-sfsturbo-volume     #卷名称，可自定义
```

```
      persistentVolumeClaim:
        claimName: pvc-sfsturbo    #已创建的PVC名称
```

2. 执行以下命令，创建一个挂载极速文件存储的应用。

```
kubectl apply -f web-demo.yaml
```

工作负载创建成功后，您可以尝试**验证数据持久化及共享性**。

----结束

验证数据持久化及共享性

步骤1 查看部署的应用及文件。

1. 执行以下命令，查看已创建的Pod。

```
kubectl get pod | grep web-demo
```

预期输出如下：

```
web-demo-846b489584-mjhm9   1/1     Running   0          46s
web-demo-846b489584-wvv5s   1/1     Running   0          46s
```

2. 依次执行以下命令，查看Pod的/data路径下的文件。

```
kubectl exec web-demo-846b489584-mjhm9 -- ls /data
kubectl exec web-demo-846b489584-wvv5s -- ls /data
```

两个Pod均无返回结果，说明/data路径下无文件。

步骤2 执行以下命令，在/data路径下创建static文件。

```
kubectl exec web-demo-846b489584-mjhm9 --  touch /data/static
```

步骤3 执行以下命令，查看/data路径下的文件。

```
kubectl exec web-demo-846b489584-mjhm9 -- ls /data
```

预期输出如下：

```
static
```

步骤4 验证数据持久化

1. 执行以下命令，删除名称为web-demo-846b489584-mjhm9的Pod。

```
kubectl delete pod web-demo-846b489584-mjhm9
```

预期输出如下：

```
pod "web-demo-846b489584-mjhm9" deleted
```

删除后，Deployment控制器会自动重新创建一个副本。

2. 执行以下命令，查看已创建的Pod。

```
kubectl get pod | grep web-demo
```

预期输出如下，web-demo-846b489584-d4d4j为新建的Pod：

```
web-demo-846b489584-d4d4j   1/1     Running   0          110s
web-demo-846b489584-wvv5s   1/1     Running   0          7m50s
```

3. 执行以下命令，验证新建的Pod中/data路径下的文件是否更改。

```
kubectl exec web-demo-846b489584-d4d4j -- ls /data
```

预期输出如下：

```
static
```

static文件仍然存在，则说明数据可持久化保存。

步骤5 验证数据共享性

1. 执行以下命令，查看已创建的Pod。

```
kubectl get pod | grep web-demo
```

预期输出如下：

```
web-demo-846b489584-d4d4j   1/1     Running   0          7m
web-demo-846b489584-wvv5s   1/1     Running   0          13m
```


2. 执行以下命令，在任意一个Pod的/data路径下创建share文件。本例中选择名为web-demo-846b489584-d4d4j的Pod。

```
kubectl exec web-demo-846b489584-d4d4j --  touch /data/share
```

并查看该Pod中/data路径下的文件。

```
kubectl exec web-demo-846b489584-d4d4j -- ls /data
```

预期输出如下：

```
share
static
```

3. 由于写入share文件的操作未在名为web-demo-846b489584-wvv5s的Pod中执行，在该Pod中查看/data路径下是否存在文件以验证数据共享性。

```
kubectl exec web-demo-846b489584-wvv5s -- ls /data
```

预期输出如下：

```
share
static
```

如果在任意一个Pod中的/data路径下创建文件，其他Pod下的/data路径下均存在此文件，则说明两个Pod共享一个存储卷。

----结束

相关操作

您还可以执行**表4-30**中的基本操作。

表 4-30 其他操作

操作	说明	操作步骤
创建存储卷	通过CCE控制台单独创建PV。	1. 在左侧导航栏选择“存储”，在右侧选择“存储卷”页签。单击右上角“创建存储卷”，在弹出的窗口中填写存储卷声明参数。 ● 存储卷类型：选择“极速文件存储”。 ● 极速文件存储：单击“选择极速文件存储”，在新页面中勾选满足要求的极速文件存储，并单击“确定”。 ● 子目录：选择是否使用子目录创建PV。请填写子目录绝对路径，例如/a/b，并确保子目录已存在且可用。 ● PV名称：输入PV名称，同一集群内的PV名称需唯一。 ● 访问模式：仅支持ReadWriteMany，表示存储卷可以被多个节点以读写方式挂载，详情请参见**存储卷访问模式**。 ● 回收策略：不使用子目录创建PV时，仅支持Retain，详情请参见**PV回收策略**。选择使用子目录创建PV时，支持选择Delete。 ● 挂载参数：输入挂载参数键值对，详情请参见**设置极速文件存储挂载参数**。 2. 单击“创建”。

操作	说明	操作步骤
扩容极速文件存储存储卷	通过CCE控制台快速扩容已挂载的极速文件存储。	1. 在左侧导航栏选择“存储”，在右侧选择“存储卷声明”页签。单击PVC操作列的“更多 > 扩容”。 2. 输入新增容量，并单击“确定”。
事件	查看PVC或PV的事件名称、事件类型、发生次数、Kubernetes事件、首次和最近发生的时间，便于定位问题。	1. 在左侧导航栏选择“存储”，在右侧选择“存储卷声明”或“存储卷”页签。 2. 单击目标实例操作列的“事件”，即可查看1小时内的事件（事件保存时间为1小时）。
查看YAML	可对PVC或PV的YAML文件进行查看、复制和下载。	1. 在左侧导航栏选择“存储”，在右侧选择“存储卷声明”或“存储卷”页签。 2. 单击目标实例操作列的“查看YAML”，即可查看或下载YAML。

4.5.3 设置极速文件存储挂载参数

本章节主要介绍如何设置极速文件存储的挂载参数。极速文件存储仅支持在PV中设置挂载参数，然后通过创建PVC绑定PV。

极速文件存储挂载参数

CCE Autopilot在挂载极速文件存储时默认设置了如**表4-31**所示的参数。

表 4-31 极速文件存储挂载参数

参数	参数值	描述
keep-original-ownership	无需填写	表示是否保留文件挂载点的ownership。 • 默认为不添加该参数，此时挂载极速文件存储时将会默认把挂载点的ownership修改为root:root。 • 如添加该参数，挂载极速文件存储时将保持文件系统原有的ownership。
vers	3	文件系统版本，目前只支持NFSv3。取值：3
nolock	无需填写	选择是否使用NLM协议在服务器上锁文件。当选择nolock选项时，锁对于同一主机的应用有效，对不同主机不受锁的影响。
timeo	600	NFS客户端重传请求前的等待时间(单位为0.1秒)。建议值：600。

参数	参数值	描述
hard/soft	无需填写	挂载方式类型。 • 取值为hard，即使用硬连接方式，若NFS请求超时，则客户端一直重新请求直至成功。 • 取值为soft，即软挂载方式挂载系统，若NFS请求超时，则客户端向调用程序返回错误。 默认为hard。

除了以上参数外，您还可以设置其他的文件存储挂载参数，具体请参见**挂载NFS文件系统到云服务器（Linux）**。

在 PV 中设置挂载参数

在PV中设置挂载参数可以通过mountOptions字段实现，如下所示，mountOptions支持挂载的字段请参见**极速文件存储挂载参数**。

步骤1 使用kubectl连接集群，详情请参见**通过kubectl连接集群**。

步骤2 在PV中设置挂载参数，示例如下：

```
apiVersion: v1
kind: PersistentVolume
metadata:
  annotations:
    pv.kubernetes.io/provisioned-by: everest-csi-provisioner
  name: pv-sfsturbo    # PV的名称
spec:
  accessModes:
  - ReadWriteMany      # 访问模式，极速文件存储必须为ReadWriteMany
  capacity:
    storage: 500Gi       # 极速文件存储容量大小
  csi:
    driver: sfsturbo.csi.everest.io    # 挂载依赖的存储驱动
    fsType: nfs
    volumeHandle: {your_volume_id}  # 极速文件存储的ID
    volumeAttributes:
      everest.io/share-export-location: {your_location}  # 极速文件存储的共享路径
      everest.io/enterprise-project-id: {your_project_id} # 极速文件存储的项目ID
      storage.kubernetes.io/csiProvisionerIdentity: everest-csi-provisioner
  persistentVolumeReclaimPolicy: Retain    # 回收策略
  storageClassName: csi-sfsturbo           # SFS Turbo存储类名称
  mountOptions:                            # 挂载参数
  - vers=3
  - nolock
  - timeo=600
  - hard
```

步骤3 PV创建后，可以创建PVC关联PV，然后在工作负载的容器中挂载，具体操作步骤请参见**通过静态存储卷使用已有极速文件存储**。

步骤4 验证挂载参数是否生效。

本例中将PVC挂载至使用nginx:latest镜像的工作负载，并通过**mount -l**命令查看挂载参数是否生效。

1. 查看已挂载文件存储的Pod，本文中的示例工作负载名称为web-sfsturbo。
   ```
   kubectl get pod | grep web-sfsturbo
   ```
   回显如下：


```
web-sfsturbo-***  1/1     Running   0          23m
```

2. 执行以下命令查看挂载参数，其中web-sfsturbo-***为示例Pod。

```
kubectl exec -it web-sfsturbo-*** -- mount -l | grep nfs
```

若回显中的挂载信息与设置的挂载参数一致，说明挂载参数设置成功。

```
{您的挂载地址} on /data type nfs
(rw,relatime,vers=3,rsize=1048576,wsize=1048576,namlen=255,hard,nolock,noresvport,proto=tcp,
timeo=600,retrans=2,sec=sys,mountaddr=**.**.**.**,mountvers=3,mountport=20048,mountproto=tc
p,local_lock=all,addr=**.**.**.**)
```

----结束

4.5.4 通过动态存储卷创建 SFS Turbo 子目录（推荐）

通常情况下，在工作负载容器中挂载SFS Turbo类型的存储卷时，默认会将根目录挂载到容器中。而SFS Turbo的容量最小为500G，超出了大多数工作负载所需的容量，导致存储容量的浪费。为了更加经济合理地利用存储容量，CCE Autopilot集群支持在创建PVC时动态创建SFS Turbo子目录，并支持配置子目录容量大小，实现不同工作负载共享使用SFS Turbo。

前提条件

- 如果您需要通过命令行创建，需要使用kubectl连接到集群，详情请参见**通过kubectl连接集群**。
- 您已经创建好一个状态可用的SFS Turbo，并且SFS Turbo与集群在同一个VPC内。

约束与限制

- 目前，只有部分区域支持使用SFS Turbo，具体请以控制台为准。您可以通过**功能总览**，了解支持该功能的所有区域。
- 使用SFS Turbo的子目录要求集群版本要求在v1.27.9-r0、v1.28.7-r0及以上。如果您的集群版本不符合该要求，则需要通过集群升级使用该功能。
- 子目录配额约束：
  - 子目录配额小于1Gi按1Gi配置，最大不能超过对应SFS Turbo实例的大小，单次扩容步长最小1Gi，不支持缩容。
  - 子目录配额限制后，不支持取消。
  - 子目录文件数量 = 配额容量（KB）/16，上限为10亿，不支持用户单独设置。

通过控制台动态创建 SFS Turbo 子目录

步骤1 登录CCE控制台，单击集群名称进入集群。

步骤2 在左侧导航栏选择“存储”，在右侧选择“存储卷声明”页签。单击右上角“创建存储卷声明”，在弹出的窗口中填写存储卷声明参数。

参数	描述
存储卷声明类型	本文中选择“极速文件存储”。
PVC名称	输入PVC的名称，同一命名空间下的PVC名称需唯一。

参数	描述
命名空间	表示PVC所在命名空间。 • 若左侧导航栏选择“存储”后，页面上方“命名空间”为“全部命名空间”或“非系统命名空间”，则此处可以选择合适的命名空间。 • 若左侧导航栏选择“存储”后，页面上方“命名空间”已指定一个特定的非系统的命名空间（如default），则此处固定为该命名空间。
创建方式	选择“动态创建子目录”。
存储类	选择极速文件存储对应的存储类为csi-sfsturbo。
访问模式	极速文件存储类型的存储卷仅支持ReadWriteMany，表示存储卷可以被多个节点以读写方式挂载，详情请参见**存储卷访问模式**。
极速文件存储	单击“选择极速文件存储”，您可以在新页面中勾选满足要求的极速文件存储，并单击“确定”。
子目录	请填写子目录绝对路径，例如/a/b。当该子目录不存在时，将会自动创建该子目录。
子目录回收策略	删除PVC时是否保留子目录。 • 保留：删除PVC，PV会被删除，但**PV关联的子目录会被保留**。 • 删除：删除PVC，**PV及其关联的子目录均会被删除**。 **说明** 删除子目录时，仅删除PVC参数中设置的子目录绝对路径，不会级联删除上层目录。
子目录容量限制	• 不限制：不开启子目录容量大小限制。 • 限制：开启子目录容量大小限制。
容量	请输入子目录限制的容量大小，单位GiB。此选项只在开启子目录容量大小限制时有效。

步骤3 单击“创建”，将同时为您创建存储卷声明和存储卷。

您可以在左侧导航栏中选择“存储”，在“存储卷声明”和“存储卷”页签下查看已经创建的存储卷声明和存储卷。

步骤4 创建工作负载。

1. 在左侧导航栏中选择“工作负载”，在右侧选择“无状态负载”页签。
2. 单击页面右上角“创建工作负载”，在“容器配置”中选择“数据存储”页签，并单击“添加存储卷 > 已有存储卷声明 (PVC)”。

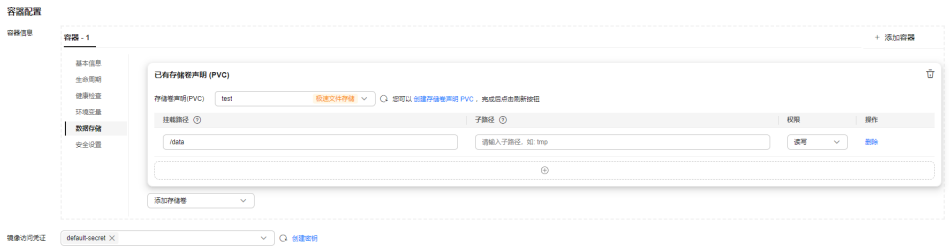
   本文主要为您介绍存储卷的挂载使用，如**表4-32**，其他参数详情请参见**工作负载**。

表 4-32 存储卷挂载

参数	参数说明
存储卷声明（PVC）	选择已有的极速文件存储卷。
挂载路径	请输入挂载路径，如：/tmp。 数据存储挂载到容器上的路径。请不要挂载在系统目录下，如“/”、“/var/run”等，会导致容器异常。建议挂载在空目录下，若目录不为空，请确保目录下无影响容器启动的文件，否则文件会被替换，导致容器启动异常，工作负载创建失败。 **须知** 挂载高危目录的情况下 ，建议使用低权限账号启动，否则可能会造成宿主机高危文件被破坏。
子路径	表示将存储卷中的某个子目录挂载到容器的指定路径，而不是将整个存储卷挂载到容器的路径，可以实现在单一Pod中使用同一个存储卷的不同文件夹。如：tmp，表示容器中挂载路径下的数据会存储在存储卷的tmp文件夹中。不填写时默认为根路径。
权限	- 只读：只能读容器路径中的数据卷。 - 读写：可修改容器路径中的数据卷，容器迁移时新写入的数据不会随之迁移，会造成数据丢失。

本例中将该存储卷挂载到容器中/data路径下，在该路径下生成的容器数据会存储到极速文件存储中。

图 4-16 挂载存储卷



3. 其余信息都配置完成后，单击“创建工作负载”。

   工作负载创建成功后，容器挂载目录下的数据将会持久化保持，您可以参考**验证数据持久化及共享性**中的步骤进行验证。

----结束

通过 kubectl 命令行动态创建 SFS Turbo 子目录

步骤1 使用kubectl连接集群。

步骤2 创建pvc-sfsturbo-subpath.yaml文件。

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-sfsturbo-subpath    # PVC的名称
  namespace: default
  annotations:
    everest.io/volume-as: absolute-path              # 表示使用SFS Turbo子目录
    everest.io/sfsturbo-share-id: <sfsturbo_id>      # SFS Turbo的ID
    everest.io/path: /a                              # 自动创建的子目录，必须为绝对路径
    everest.io/reclaim-policy: retain-volume-only    # 表示删除PVC时，PV会被删除，但PV关联的子目录会被保留
spec:
  accessModes:
    - ReadWriteMany      # SFS Turbo必须为ReadWriteMany
  resources:
    requests:
      storage: 10Gi      # 对于SFS Turbo子目录类型的PVC，此处无实际意义，仅作校验需要（不能为空和0）
  storageClassName: csi-sfsturbo      # SFS Turbo存储类名称
```

表 4-33 关键参数说明

参数	是否必选	描述
everest.io/volume-as	否	固定取值为“absolute-path”，表示使用动态创建SFS Turbo子目录。
everest.io/sfsturbo-share-id	否	SFS Turbo的ID。 **获取方法**：在CCE控制台，单击顶部的“服务列表 > 存储 > 弹性文件服务”，并选择SFS Turbo。在列表中单击对应的极速弹性文件存储名称，在详情页中复制“ID”后的内容即可。
everest.io/path	否	自动创建的子目录，必须为绝对路径。
everest.io/reclaim-policy	否	删除PVC时是否保留子目录，该参数需与**PV回收策略**配合使用。仅当PV回收策略为"Delete"时生效，取值如下： • retain-volume-only：表示删除PVC时，PV会被删除，但**PV关联的子目录会被保留**。 • delete：表示删除PVC，**PV及其关联的子目录均会被删除**。 **说明** 删除子目录时，仅删除PVC参数中设置的子目录绝对路径，不会级联删除上层目录。
storage	是	PVC申请容量，单位为Gi。 对于SFS Turbo子目录类型的PVC： • 开启配额限制时，表示子目录容量大小。 • 未开启配额限制时，无实际意义，仅作校验需要，此处可设置为固定值10Gi。
everest.io/csi.enable-sfsturbo-dir-quota	否	是否开启子目录的配额限制。“true”表示开启限制，其他值或不填表示不开启限制。

步骤3 执行以下命令，创建PVC。

```
kubectl apply -f pvc-sfsturbo-subpath.yaml
```

步骤4 创建工作负载。

1. 创建web-demo.yaml文件，本示例中将极速文件存储挂载至/data路径。

```
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     #卷名称，需与volumes字段中的卷名称对应
          mountPath: /data  #存储卷挂载的位置
      imagePullSecrets:
        - name: default-secret
      volumes:
        - name: pvc-sfsturbo-volume     #卷名称，可自定义
          persistentVolumeClaim:
            claimName: pvc-sfsturbo-subpath     #已创建的PVC名称
```

2. 执行以下命令，创建一个挂载极速文件存储的应用。

```
kubectl apply -f web-demo.yaml
```

工作负载创建成功后，您可以参考**验证数据持久化及共享性**中的步骤进行验证。

----结束

验证数据持久化及共享性

步骤1 查看部署的应用及文件。

1. 执行以下命令，查看已创建的Pod。

```
kubectl get pod | grep web-demo
```

预期输出如下：

```
web-demo-846b489584-mjhm9   1/1     Running   0          46s
web-demo-846b489584-wvv5s   1/1     Running   0          46s
```

2. 依次执行以下命令，查看Pod的/data路径下的文件。

```
kubectl exec web-demo-846b489584-mjhm9 -- ls /data
kubectl exec web-demo-846b489584-wvv5s -- ls /data
```

两个Pod均无返回结果，说明/data路径下无文件。

步骤2 执行以下命令，在/data路径下创建static文件。

```
kubectl exec web-demo-846b489584-mjhm9 --  touch /data/static
```

步骤3 执行以下命令，查看/data路径下的文件。

```
kubectl exec web-demo-846b489584-mjhm9 -- ls /data
```

预期输出如下：

```
static
```

步骤4 验证数据持久化

1. 执行以下命令，删除名称为web-demo-846b489584-mjhm9的Pod。
```
kubectl delete pod web-demo-846b489584-mjhm9
```
预期输出如下：
```
pod "web-demo-846b489584-mjhm9" deleted
```
删除后，Deployment控制器会自动重新创建一个副本。
2. 执行以下命令，查看已创建的Pod。
```
kubectl get pod | grep web-demo
```
预期输出如下，web-demo-846b489584-d4d4j为新建的Pod：
```
web-demo-846b489584-d4d4j   1/1     Running   0          110s
web-demo-846b489584-wvv5s   1/1     Running   0          7m50s
```
3. 执行以下命令，验证新建的Pod中/data路径下的文件是否更改。
```
kubectl exec web-demo-846b489584-d4d4j -- ls /data
```
预期输出如下：
```
static
```
static文件仍然存在，则说明数据可持久化保存。

步骤5 验证数据共享性

1. 执行以下命令，查看已创建的Pod。
```
kubectl get pod | grep web-demo
```
预期输出如下：
```
web-demo-846b489584-d4d4j   1/1     Running   0          7m
web-demo-846b489584-wvv5s   1/1     Running   0          13m
```
2. 执行以下命令，在任意一个Pod的/data路径下创建share文件。本例中选择名为web-demo-846b489584-d4d4j的Pod。
```
kubectl exec web-demo-846b489584-d4d4j --  touch /data/share
```
并查看该Pod中/data路径下的文件。
```
kubectl exec web-demo-846b489584-d4d4j -- ls /data
```
预期输出如下：
```
share
static
```
3. 由于写入share文件的操作未在名为web-demo-846b489584-wvv5s的Pod中执行，在该Pod中查看/data路径下是否存在文件以验证数据共享性。
```
kubectl exec web-demo-846b489584-wvv5s -- ls /data
```
预期输出如下：
```
share
static
```
如果在任意一个Pod中的/data路径下创建文件，其他Pod下的/data路径下均存在此文件，则说明两个Pod共享一个存储卷。

----结束

相关操作

您还可以执行**表4-34**中的操作。

表 4-34 其他操作

操作	说明	操作步骤
事件	查看PVC或PV的事件名称、事件类型、发生次数、Kubernetes事件、首次和最近发生的时间，便于定位问题。	1. 在左侧导航栏选择“存储”，在右侧选择“存储卷声明”或“存储卷”页签。 2. 单击目标实例操作列的“事件”，即可查看1小时内的事件（事件保存时间为1小时）。
查看YAML	可对PVC或PV的YAML文件进行查看、复制和下载。	1. 在左侧导航栏选择“存储”，在右侧选择“存储卷声明”或“存储卷”页签。 2. 单击目标实例操作列的“查看YAML”，即可查看或下载YAML。

4.5.5 SFS Turbo 动态创建子目录并挂载

背景信息

SFS Turbo容量最小500G，且不是按使用量计费。SFS Turbo挂载时默认将根目录挂载到容器，而通常情况下负载不需要这么大容量，造成浪费。

CCE支持一种在SFS Turbo下动态创建子目录的方法，能够在SFS Turbo下动态创建子目录并挂载到容器，这种方法能够共享使用SFS Turbo，从而更加经济合理的利用SFS Turbo存储容量。

目前，只有部分区域支持使用SFS Turbo，具体请以控制台为准。您可以通过**功能总览**，了解支持该功能的所有区域。

创建 subpath 类型 SFS Turbo 存储卷

说明

相比于CCE Standard/Turbo集群，CCE Autopilot集群删除SFS Turbo子目录PVC时不支持对子目录进行归档，即删除SFS Turbo子目录PVC时，若该PVC对应的PV的persistentVolumeReclaimPolicy为Delete，则对应的SFS Turbo子目录会被删除。

步骤1 创建SFS Turbo资源，选择网络时，请选择与集群相同的VPC与子网。

步骤2 新建一个StorageClass的YAML文件，例如sfsturbo-subpath-sc.yaml。

配置示例：

```
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/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
```

其中：

- name：storageclass的名称。
- mountOptions：选填字段；mount挂载参数。

  默认使用如下所示配置，具体请参见**设置挂载参数**。**此处不能配置为nolock=true，会导致挂载失败。**

  ```
  mountOptions:
  - vers=3
  - timeo=600
  - nolock
  - hard
  ```

- everest.io/volume-as：该参数需设置为“subpath”来使用subpath模式。
- everest.io/share-access-to：选填字段。subpath模式下，填写SFS Turbo资源的所在VPC的ID。
- everest.io/share-expand-type：选填字段。若SFS Turbo资源存储类型为增强版（标准型增强版、性能型增强版），设置为bandwidth。
- everest.io/share-export-location：挂载目录配置。由SFS Turbo共享路径和子目录组成，共享路径可至SFS Turbo服务页面查询，子路径由用户自定义，后续指定该StorageClass创建的PVC均位于该子目录下。
- everest.io/share-volume-type：选填字段。填写SFS Turbo的类型。标准型为STANDARD，性能型为PERFORMANCE。对于增强型需配合“everest.io/share-expand-type”字段使用，everest.io/share-expand-type设置为“bandwidth”。
- everest.io/zone：选填字段。指定SFS Turbo资源所在的可用区。
- everest.io/volume-id：SFS Turbo资源的卷ID，可至SFS Turbo界面查询。

步骤3 执行**kubectl create -f sfsturbo-subpath-sc.yaml**。

步骤4 新建一个PVC的YAML文件，sfs-turbo-test.yaml。

配置示例：

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: sfs-turbo-test
  namespace: default
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 50Gi
  storageClassName: sfsturbo-subpath-sc
  volumeMode: Filesystem
```

其中：

- name: PVC的名称。
- storageClassName：SC的名称。

- storage：subpath模式下，该参数无实际意义，容量受限于SFS Turbo资源的总容量，若SFS Turbo资源总容量不足，请及时到SFS Turbo界面扩容。

步骤5 执行**kubectl create -f sfs-turbo-test.yaml**。

----结束

说明

对subpath类型的SFS Turbo扩容时，没有实际的扩容意义。该操作不会对SFS Turbo资源进行实际的扩容，需要用户自行保证SFS Turbo的总容量不被耗尽。

创建 Deployment 挂载已有数据卷

步骤1 新建一个Deployment的YAML文件，例如deployment-test.yaml。

配置示例：

```
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
```

其中：

- name：创建的工作负载名称。
- image：工作负载的镜像。
- mountPath：容器内挂载路径，示例中挂载到“/tmp”路径。
- claimName：已有的PVC名称。

步骤2 创建Deployment负载。

kubectl create -f deployment-test.yaml

----结束

StatefulSet 动态创建 subpath 模式的数据卷

步骤1 新建一个StatefulSet的YAML文件，例如statefulset-test.yaml。

配置示例：

```
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
```

其中：

- name：创建的工作负载名称。
- image：工作负载的镜像。

- mountPath：容器内挂载路径，示例中挂载到“/tmp”路径。
- “spec.template.spec.containers.volumeMounts.name ”和“spec.volumeClaimTemplates.metadata.name”有映射关系，必须保持一致。
- storageClassName: 填写自建的SC名称。

步骤2 创建StatefulSet负载。

kubectl create -f statefulset-test.yaml

----结束

4.6 对象存储（OBS）

4.6.1 对象存储概述

对象存储介绍

对象存储服务（Object Storage Service，OBS）提供海量、安全、高可靠、低成本的数据存储能力，可供用户存储任意类型和大小的数据。适合企业备份/归档、视频点播、视频监控等多种数据存储场景。

- **标准接口：**具备标准Http Restful API接口，用户必须通过编程或第三方工具访问对象存储。
- **数据共享：**服务器、嵌入式设备、IOT设备等所有调用相同路径，均可访问共享的对象存储数据。
- **公共/私有网络：**对象存储数据允许在公网访问，满足互联网应用需求。
- **容量与性能：**容量无限制，性能较高（IO读写时延10ms级）。
- **应用场景：**适用于（基于OBS界面、OBS工具、OBS SDK等）的一次上传共享多读（ReadOnlyMany）的各种工作负载（Deployment/StatefulSet）和普通任务（Job）使用，主要面向大数据分析、静态网站托管、在线视频点播、基因测序、智能视频监控、备份归档、企业云盘（网盘）等场景。

对象存储规格

对象存储提供了多种存储类别，从而满足客户业务对存储性能、成本的不同诉求。

- 对象桶：提供高可靠、高性能、高安全、低成本的数据存储能力，无文件数量限制、容量限制。
  - 标准存储：访问时延低和吞吐量高，因而适用于有大量热点文件（平均一个月多次）或小文件（小于1MB），且需要频繁访问数据的业务场景，例如：大数据、移动应用、热点视频、社交图片等场景。
  - 低频访问存储：适用于不频繁访问（平均一年少于12次）但在需要时也要求快速访问数据的业务场景，例如：文件同步/共享、企业备份等场景。与标准存储相比，低频访问存储有相同的数据持久性、吞吐量以及访问时延，且成本较低，但是可用性略低于标准存储。
- 并行文件系统：并行文件系统（Parallel File System）是对象存储服务的子产品，是经过优化的高性能文件语义系统，主要应用于大数据场景。详细介绍请参见**什么是并行文件系统**。

关于对象存储的详细介绍，请以**对象存储类别**为准。

性能说明

容器负载挂载对象存储时，每挂载一个对象存储卷，后端会产生一个常驻进程。当负载使用对象存储数过多或大量读写对象存储文件时，常驻进程会占用大量内存，部分场景下内存消耗量参考**表4-35**，为保证负载稳定运行，建议负载使用的对象存储卷数量不超过其申请的内存GiB数量，如负载的申请的内存规格为4GiB，则建议其使用的对象存储数**不超过**4。

表 4-35 单个对象存储常驻进程内存消耗

测试项目	内存消耗
长稳运行	约50m
2并发写10M文件	约110m
4并发写10M文件	约220m
单写100G文件	约300m

使用场景

根据使用场景不同，对象存储支持以下挂载方式：

- **通过静态存储卷使用已有对象存储**：即静态创建的方式，需要先使用已有的对象存储创建PV，然后通过PVC在工作负载中挂载存储。适用于已有可用的底层存储或底层存储需要包周期的场景。
- **通过动态存储卷使用对象存储**：即动态创建的方式，无需预先创建对象存储，在创建PVC时通过指定存储类（StorageClass），即可自动创建对象存储和对应的PV对象。适用于无可用的底层存储，需要新创建的场景。

计费说明

- 挂载对象存储类型的存储卷时，通过StorageClass**自动创建**的对象存储默认创建计费模式为“按需计费”。关于对象存储的价格信息，请参见**对象存储计费说明**。
- 如需使用包周期的对象存储，请**使用已有的对象存储**进行挂载。

4.6.2 通过静态存储卷使用已有对象存储

本文介绍如何使用已有的对象存储静态创建PV和PVC，并在工作负载中实现数据持久化与共享性。

前提条件

如果您需要通过命令行创建，需要使用kubectl连接到集群，详情请参见**通过kubectl连接集群**。

约束与限制

- 使用对象存储时，挂载点不支持修改属组和权限。
- 挂载普通桶时不支持硬链接（Hard Link）。

- 支持多个PV挂载同一个对象存储，但有如下限制：
  - 多个不同的PVC/PV使用同一个底层对象存储卷时，如果挂载至同一Pod使用，会因为PV的volumeHandle参数值相同导致无法挂载，请避免该使用场景。
  - PV中persistentVolumeReclaimPolicy参数建议设置为Retain，否则可能存在一个PV删除时，级联删除底层卷，其他关联这个底层卷的PV会由于底层存储被删除导致使用出现异常。
  - 重复用底层存储时，数据一致性由您自行维护。建议在应用层做好多读多写的隔离保护，合理规划文件使用时间，避免出现多个客户端写同一个文件的情况，防止产生数据覆盖和丢失。

通过控制台使用已有对象存储

步骤1 登录CCE控制台，单击集群名称进入集群。

步骤2 静态创建存储卷声明和存储卷。

1. 在左侧导航栏选择“存储”，在右侧选择“存储卷声明”页签。单击右上角“创建存储卷声明 PVC”，在弹出的窗口中填写存储卷声明参数。

参数	描述
存储卷声明类型	本文中选择“对象存储”。
OBS终端节点	Autopilot集群内访问对象存储，需要创建OBS终端节点，用于连接VPC网络和OBS服务。
PVC名称	输入PVC的名称，同一命名空间下的PVC名称需唯一。
创建方式	- 已有底层存储的场景下，根据是否已经创建PV可选择“新建存储卷”或“已有存储卷”来静态创建PVC。 - 无可用底层存储的场景下，可选择“动态创建”，具体操作请参见**通过动态存储卷使用对象存储**。 本文示例中选择“新建存储卷”，可通过控制台同时创建PV及PVC。
关联存储卷[a]	选择集群中已有的PV卷，需要提前创建PV，请参考**相关操作**中的“创建存储卷”操作。 本文示例中无需选择。
对象存储[b]	单击“选择对象存储”，您可以在新页面中勾选满足要求的对象存储，并单击“确定”。
PV名称[b]	输入PV名称，同一集群内的PV名称需唯一。
访问模式[b]	对象存储类型的存储卷仅支持ReadWriteMany，表示存储卷可以被多个节点以读写方式挂载，详情请参见**存储卷访问模式**。
回收策略[b]	您可以选择Delete或Retain，用于指定删除PVC时底层存储的回收策略，详情请参见**PV回收策略**。 **说明** 多个PV使用同一个对象存储时建议使用**Retain**，避免级联删除底层卷。

参数	描述
访问密钥 (AK/SK)[b]	自定义密钥：如果您需要为不同OBS存储分配不同的用户权限时，可通过选择不同的Secret实现更灵活的权限控制（推荐使用）。具体使用请参见**对象存储卷挂载设置自定义访问密钥（AK/SK）**。 仅支持选择带有 secret.kubernetes.io/used-by = csi 标签的密钥，密钥类型为cfe/secure-opaque。如果无可用密钥，可单击“创建密钥”进行创建： – 名称：请输入密钥名称。 – 命名空间：密钥所在的命名空间。 – 访问密钥（AK/SK）：上传.csv格式的密钥文件，详情请参见**获取访问密钥**。
挂载参数[b]	输入挂载参数键值对，详情请参见**设置对象存储挂载参数**。

说明

a：创建方式选择“已有存储卷 PV”时可设置。

b：创建方式选择“新建存储卷 PV”时可设置。

2. 单击“创建”，将同时为您创建存储卷声明和存储卷。

   您可以在左侧导航栏中选择“存储”，在“存储卷声明”和“存储卷”页签下查看已经创建的存储卷声明和存储卷。

步骤3 创建应用。

1. 在左侧导航栏中选择“工作负载”，在右侧选择“无状态负载”页签。
2. 单击页面右上角“创建工作负载”，在“容器配置”中选择“数据存储”页签，并单击“添加存储卷 > 已有存储卷声明 (PVC)”。

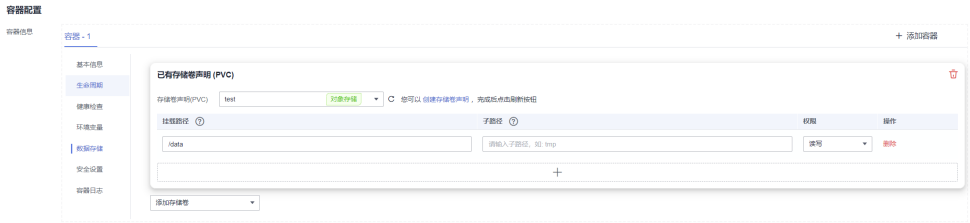
   本文主要为您介绍存储卷的挂载使用，如**表4-36**，其他参数详情请参见**创建工作负载**。

表 4-36 存储卷挂载

参数	参数说明
存储卷声明（PVC）	选择已有的对象存储卷。
挂载路径	请输入挂载路径，如：/tmp。 数据存储挂载到容器上的路径。请不要挂载在系统目录下，如“/”、“**/var/run**”等，会导致容器异常。建议挂载在空目录下，若目录不为空，请确保目录下无影响容器启动的文件，否则文件会被替换，导致容器启动异常，工作负载创建失败。 **须知** 挂载高危目录的情况下，建议使用低权限账号启动，否则可能会造成宿主机高危文件被破坏。

参数	参数说明
子路径	请输入存储卷的子路径，将存储卷中的某个路径挂载至容器，可以实现在单一Pod中使用同一个存储卷的不同文件夹。如：tmp，表示容器中挂载路径下的数据会存储在存储卷的tmp文件夹中。不填写时默认为根路径。
权限	- 只读：只能读容器路径中的数据卷。 - 读写：可修改容器路径中的数据卷，容器迁移时新写入的数据不会随之迁移，会造成数据丢失。

本例中将该存储卷挂载到容器中/data路径下，在该路径下生成的容器数据会存储到对象存储中。



3. 其余信息都配置完成后，单击“创建工作负载”。

   工作负载创建成功后，容器挂载目录下的数据将会持久化保持，您可以参考**验证数据持久化及共享性**中的步骤进行验证。

----结束

通过 kubectl 命令行使用已有对象存储

步骤1 使用kubectl连接集群。

步骤2 创建PV。

1. 创建pv-obs.yaml文件。

```
apiVersion: v1
kind: PersistentVolume
metadata:
  annotations:
    pv.kubernetes.io/provisioned-by: everest-csi-provisioner
    everest.io/reclaim-policy: retain-volume-only      # 可选字段，删除PV，保留底层存储卷
  name: pv-obs       # PV的名称
spec:
  accessModes:
  - ReadWriteMany    # 访问模式，对象存储必须为ReadWriteMany
  capacity:
    storage: 1Gi     # 对象存储容量大小
  csi:
    driver: obs.csi.everest.io        # 挂载依赖的存储驱动
    fsType: obsfs                     # 实例类型
    volumeHandle: <your_volume_id>    # 对象存储的名称
    volumeAttributes:
      storage.kubernetes.io/csiProvisionerIdentity: everest-csi-provisioner
      everest.io/obs-volume-type: STANDARD
      everest.io/region: <your_region>                    # 对象存储的区域
    nodePublishSecretRef:           # 设置对象存储的自定义密钥
      name: <your_secret_name>      # 自定义密钥的名称
      namespace: <your_namespace>   # 自定义密钥的命名空间
  persistentVolumeReclaimPolicy: Retain    # 回收策略
```

```
storageClassName: csi-obs          # 存储类名称
mountOptions: []                   # 挂载参数
```

表 4-37 关键参数说明

参数	是否必填	描述
everest.io/reclaim-policy: retain-volume-only	否	可选字段 目前仅支持配置“retain-volume-only” 如果回收策略是Delete且当前值设置为“retain-volume-only”删除PVC回收逻辑为：删除PV，保留底层存储卷。
fsType	是	实例类型，支持“obsfs”与“s3fs”。 – obsfs：并行文件系统，配套使用obsfs挂载。 – s3fs：对象桶，配套使用s3fs挂载。
volumeHandle	是	对象存储的名称。
everest.io/obs-volume-type	是	对象存储类型。 – fsType设置为s3fs时，支持STANDARD（标准桶）、WARM（低频访问桶）。 – fsType设置为obsfs时，该字段不起作用。
everest.io/region	是	OBS存储区域。 Region对应的值请参见**地区和终端节点**。

nodePublishSecretRef	否	对象存储卷挂载支持设置自定义访问密钥（AK/SK），您可以使用AK/SK创建一个Secret，然后挂载到PV。详细说明请参见**对象存储卷挂载设置自定义访问密钥（AK/SK）**。
		示例如下：
		`nodePublishSecretRef:`
		`  name: secret-demo`
		`  namespace: default`

mountOptions	否	挂载参数，具体请参见**设置对象存储挂载参数**。

参数	是否必填	描述
persistentVolumeReclaimPolicy	是	支持Delete、Retain回收策略，详情请参见**PV回收策略**。多个PV使用同一个对象存储时建议使用**Retain**，避免级联删除底层卷。 **Delete**: – Delete且不设置everest.io/reclaim-policy：删除PVC，PV资源与存储均被删除。 – Delete且设置everest.io/reclaim-policy=retain-volume-only：删除PVC，PV资源被删除，存储资源会保留。 **Retain**：删除PVC，PV资源与底层存储资源均不会被删除，需要手动删除回收。PVC删除后PV资源状态为“已释放（Released）”，不能直接再次被PVC绑定使用。
storage	是	存储容量，单位为Gi。 对对象存储来说，此处仅为校验需要（不能为空和0），设置的大小不起作用，此处设定为固定值1Gi。
storageClassName	是	对象存储对应的存储类名称为csi-obs。

2. 执行以下命令，创建PV。
```
kubectl apply -f pv-obs.yaml
```

步骤3 创建PVC。

1. 创建pvc-obs.yaml文件。
```
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>  # 自定义密钥的名称
    csi.storage.k8s.io/node-publish-secret-namespace: <your_namespace>       # 自定义密钥的命名空间
spec:
  accessModes:
  - ReadWriteMany               # 对象存储必须为ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: csi-obs      # 存储类名称，必须与PV的存储类一致。
  volumeName: pv-obs    # PV的名称
```


表 4-38 关键参数说明

参数	是否必填	描述
csi.storage.k8s.io/node-publish-secret-name	否	PV中指定的自定义密钥的名称。
csi.storage.k8s.io/node-publish-secret-namespace	否	PV中指定的自定义密钥的命名空间。
storage	是	PVC申请容量，单位为Gi。 对于对象存储来说，此处仅为校验需要（不能为空和0），设置的大小不起作用，此处设定为固定值1Gi。
storageClassName	是	存储类名称，必须与1中PV的存储类一致。 对象存储对应的存储类名称为csi-obs。
volumeName	是	PV的名称，必须与1中PV名称一致。

2. 执行以下命令，创建PVC。

```
kubectl apply -f pvc-obs.yaml
```

步骤4 创建应用。

1. 创建web-demo.yaml文件，本示例中将对象存储挂载至/data路径。

```
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    #卷名称，需与volumes字段中的卷名称对应
          mountPath: /data  #存储卷挂载的位置
      imagePullSecrets:
        - name: default-secret
      volumes:
        - name: pvc-obs-volume    #卷名称，可自定义
          persistentVolumeClaim:
            claimName: pvc-obs    #已创建的PVC名称
```

2. 执行以下命令，创建一个挂载对象存储的应用。

```
kubectl apply -f web-demo.yaml
```

工作负载创建成功后，您可以尝试**验证数据持久化及共享性**。

----结束

验证数据持久化及共享性

步骤1 查看部署的应用及文件。

1. 执行以下命令，查看已创建的Pod。
   ```
   kubectl get pod | grep web-demo
   ```
   预期输出如下：
   ```
   web-demo-846b489584-mjhm9   1/1     Running   0          46s
   web-demo-846b489584-wvv5s   1/1     Running   0          46s
   ```
2. 依次执行以下命令，查看Pod的/data路径下的文件。
   ```
   kubectl exec web-demo-846b489584-mjhm9 -- ls /data
   kubectl exec web-demo-846b489584-wvv5s -- ls /data
   ```
   两个Pod均无返回结果，说明/data路径下无文件。

步骤2 执行以下命令，在/data路径下创建static文件。

```
kubectl exec web-demo-846b489584-mjhm9 --  touch /data/static
```

步骤3 执行以下命令，查看/data路径下的文件。

```
kubectl exec web-demo-846b489584-mjhm9 -- ls /data
```

预期输出如下：

```
static
```

步骤4 验证数据持久化

1. 执行以下命令，删除名称为web-demo-846b489584-mjhm9的Pod。
   ```
   kubectl delete pod web-demo-846b489584-mjhm9
   ```
   预期输出如下：
   ```
   pod "web-demo-846b489584-mjhm9" deleted
   ```
   删除后，Deployment控制器会自动重新创建一个副本。
2. 执行以下命令，查看已创建的Pod。
   ```
   kubectl get pod | grep web-demo
   ```
   预期输出如下，web-demo-846b489584-d4d4j为新建的Pod：
   ```
   web-demo-846b489584-d4d4j   1/1     Running   0          110s
   web-demo-846b489584-wvv5s   1/1     Running   0          7m50s
   ```
3. 执行以下命令，验证新建的Pod中/data路径下的文件是否更改。
   ```
   kubectl exec web-demo-846b489584-d4d4j -- ls /data
   ```
   预期输出如下：
   ```
   static
   ```
   static文件仍然存在，则说明数据可持久化保存。

步骤5 验证数据共享性

1. 执行以下命令，查看已创建的Pod。
   ```
   kubectl get pod | grep web-demo
   ```
   预期输出如下：
   ```
   web-demo-846b489584-d4d4j   1/1     Running   0          7m
   web-demo-846b489584-wvv5s   1/1     Running   0          13m
   ```
2. 执行以下命令，在任意一个Pod的/data路径下创建share文件。本例中选择名为web-demo-846b489584-d4d4j的Pod。
   ```
   kubectl exec web-demo-846b489584-d4d4j --  touch /data/share
   ```
   并查看该Pod中/data路径下的文件。
   ```
   kubectl exec web-demo-846b489584-d4d4j -- ls /data
   ```
   预期输出如下：
   ```
   share
   static
   ```

3. 由于写入share文件的操作未在名为web-demo-846b489584-wvv5s的Pod中执行，在该Pod中查看/data路径下是否存在文件以验证数据共享性。

```
kubectl exec web-demo-846b489584-wvv5s -- ls /data
```

预期输出如下：

```
share
static
```

如果在任意一个Pod中的/data路径下创建文件，其他Pod下的/data路径下均存在此文件，则说明两个Pod共享一个存储卷。

----结束

相关操作

您还可以执行表4-39中的操作。

表 4-39 其他操作

操作	说明	操作步骤
创建存储卷	通过CCE控制台单独创建PV。	1. 在左侧导航栏选择“存储”，在右侧选择“存储卷”页签。单击右上角“创建存储卷”，在弹出的窗口中填写存储卷声明参数。 • 存储卷类型：选择“对象存储”。 • 对象存储：单击“选择对象存储”，在新页面中勾选满足要求的对象存储，并单击“确定”。 • PV名称：输入PV名称，同一集群内的PV名称需唯一。 • 访问模式：仅支持ReadWriteMany，表示存储卷可以被多个节点以读写方式挂载，详情请参见**存储卷访问模式**。 • 回收策略：Delete或Retain，详情请参见**PV回收策略**。 **说明** 多个PV使用同一个底层存储时建议使用**Retain**，避免级联删除底层卷。 • 自定义密钥：如果您需要为不同OBS存储分配不同的用户权限时，可通过选择不同的Secret实现更灵活的权限控制（推荐使用）。具体使用请参见**对象存储卷挂载设置自定义访问密钥（AK/SK）**。 仅支持选择带有 secret.kubernetes.io/used-by = csi 标签的密钥，密钥类型为 cfe/secure-opaque。如果无可用密钥，可单击“创建密钥”进行创建。 • 挂载参数：输入挂载参数键值对，详情请参见**设置对象存储挂载参数**。 2. 单击“创建”。

操作	说明	操作步骤
更新访问密钥	通过CCE控制台更新对象存储的访问密钥。	1. 在左侧导航栏选择“存储”，在右侧选择“存储卷声明”页签。单击PVC操作列的“更多 > 更新访问密钥”。 2. 上传.csv格式的密钥文件，详情请参见**获取访问密钥**。单击“确定”。
事件	查看PVC或PV的事件名称、事件类型、发生次数、Kubernetes事件、首次和最近发生的时间，便于定位问题。	1. 在左侧导航栏选择“存储”，在右侧选择“存储卷声明”或“存储卷”页签。 2. 单击目标实例操作列的“事件”，即可查看1小时内的事件（事件保存时间为1小时）。
查看YAML	可对PVC或PV的YAML文件进行查看、复制和下载。	1. 在左侧导航栏选择“存储”，在右侧选择“存储卷声明”或“存储卷”页签。 2. 单击目标实例操作列的“查看YAML”，即可查看或下载YAML。

4.6.3 通过动态存储卷使用对象存储

本文介绍如何自动创建对象存储，适用于无可用的底层存储卷，需要新创建的场景。

约束与限制

- 使用对象存储时，挂载点不支持修改属组和权限。
- 挂载普通桶时不支持硬链接（Hard Link）。
- OBS限制单用户创建100个桶，当动态创建的PVC数量较多时，容易导致桶数量超过限制，OBS桶无法创建。此种场景下建议直接调用OBS的API或SDK使用OBS，不在工作负载中挂载OBS桶。

通过控制台自动创建对象存储

步骤1 登录CCE控制台，单击集群名称进入集群。

步骤2 动态创建存储卷声明和存储卷。

1. 在左侧导航栏选择“存储”，在右侧选择“存储卷声明”页签。单击右上角“创建存储卷声明 PVC”，在弹出的窗口中填写存储卷声明参数。

参数	描述
存储卷声明类型	本文中选择“对象存储”。
OBS终端节点	Autopilot集群内访问对象存储，需要创建OBS终端节点，用于连接VPC网络和OBS服务。
PVC名称	输入PVC的名称，同一命名空间下的PVC名称需唯一。

参数	描述
创建方式	– 无可用底层存储的场景下，可选择“动态创建”，通过控制台级联创建存储卷声明PVC、存储卷PV和底层存储。 – 已有底层存储的场景下，根据是否已经创建PV可选择“新建存储卷”或“已有存储卷”，静态创建PVC，具体操作请参见**通过静态存储卷使用已有对象存储**。 本文中选择“动态创建”。
存储类	对象存储对应的存储类为csi-obs。
实例类型	– 并行文件系统：一种对象存储服务提供的高性能文件系统，提供毫秒级别访问时延，以及TB/s级别带宽和百万级别的IOPS。**推荐您使用并行文件系统**。 – 对象桶：桶（Bucket）是OBS中存储对象的容器，桶中的所有对象都处于同一逻辑层级。
对象存储类型	选择“对象桶”时，支持选择以下类别： – 标准存储：适用于有大量热点文件或小文件，且需要频繁访问（平均一个月多次）并快速获取数据的业务场景。 – 低频访问存储：适用于不频繁访问（平均一年少于12次），但需要快速获取数据的业务场景。
访问模式	对象存储类型的存储卷仅支持ReadWriteMany，表示存储卷可以被多个节点以读写方式挂载，详情请参见**存储卷访问模式**。
访问密钥(AK/SK)	自定义密钥：如果您需要为不同OBS存储分配不同的用户权限时，可通过选择不同的Secret实现更灵活的权限控制（推荐使用）。具体使用请参见**对象存储卷挂载设置自定义访问密钥（AK/SK）**。 仅支持选择带有 secret.kubernetes.io/used-by = csi 标签的密钥，密钥类型为cfe/secure-opaque。如果无可用密钥，可单击“创建密钥”进行创建： – 名称：请输入密钥名称。 – 命名空间：密钥所在的命名空间。 – 访问密钥（AK/SK）：上传.csv格式的密钥文件，详情请参见**获取访问密钥**。

2. 单击“创建”，将同时为您创建存储卷声明和存储卷。

   您可以在左侧导航栏中选择“存储”，在“存储卷声明”和“存储卷”页签下查看已经创建的存储卷声明和存储卷。

步骤3 创建应用。

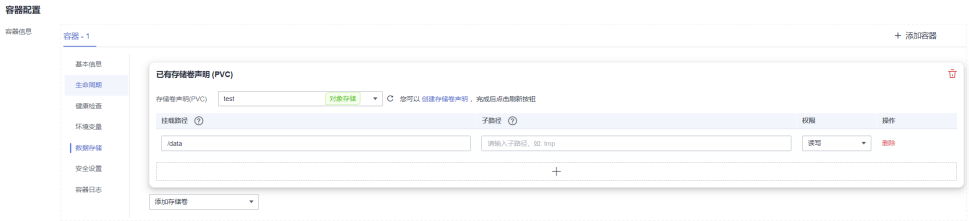
1. 在左侧导航栏中选择“工作负载”，在右侧选择“无状态负载”页签。
2. 单击页面右上角“创建工作负载”，在“容器配置”中选择“数据存储”页签，并单击“添加存储卷 > 已有存储卷声明 (PVC)”。

   本文主要为您介绍存储卷的挂载使用，如**表4-40**，其他参数详情请参见**创建工作负载**。

表 4-40 存储卷挂载

参数	参数说明
存储卷声明（PVC）	选择已有的对象存储卷。
挂载路径	请输入挂载路径，如：/tmp。 数据存储挂载到容器上的路径。请不要挂载在系统目录下，如“/”、“/var/run”等，会导致容器异常。建议挂载在空目录下，若目录不为空，请确保目录下无影响容器启动的文件，否则文件会被替换，导致容器启动异常，工作负载创建失败。 **须知** 挂载高危目录的情况下，建议使用低权限账号启动，否则可能会造成宿主机高危文件被破坏。
子路径	请输入存储卷的子路径，将存储卷中的某个路径挂载至容器，可以实现在单一Pod中使用同一个存储卷的不同文件夹。如：tmp，表示容器中挂载路径下的数据会存储在存储卷的tmp文件夹中。不填写时默认为根路径。
权限	- 只读：只能读容器路径中的数据卷。 - 读写：可修改容器路径中的数据卷，容器迁移时新写入的数据不会随之迁移，会造成数据丢失。

本例中将该存储卷挂载到容器中/data路径下，在该路径下生成的容器数据会存储到对象存储中。



3. 其余信息都配置完成后，单击“创建工作负载”。

   工作负载创建成功后，容器挂载目录下的数据将会持久化保持，您可以参考**验证数据持久化及共享性**中的步骤进行验证。

----结束

使用 kubectl 自动创建对象存储

步骤1 使用kubectl连接集群。

步骤2 使用StorageClass动态创建PVC及PV。

1. 创建pvc-obs-auto.yaml文件。

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-obs-auto
  namespace: default
```

```
  annotations:
    everest.io/obs-volume-type: STANDARD    # 对象存储类型
    csi.storage.k8s.io/fstype: obsfs        # 实例类型
    csi.storage.k8s.io/node-publish-secret-name: <your_secret_name>       # 自定义密钥的名称
    csi.storage.k8s.io/node-publish-secret-namespace: <your_namespace>    # 自定义密钥的命名空间
spec:
  accessModes:
    - ReadWriteMany             # 对象存储必须为ReadWriteMany
  resources:
    requests:
      storage: 1Gi              # 对象存储大小
  storageClassName: csi-obs     # StorageClass类型为对象存储
```

表 4-41 关键参数说明

参数	是否必选	描述
everest.io/obs-volume-type	是	对象存储类型。 – fsType设置为s3fs时，支持STANDARD（标准桶）、WARM（低频访问桶）。 – fsType设置为obsfs时，该字段不起作用。
csi.storage.k8s.io/fstype	是	实例类型，支持“obsfs”与“s3fs”。 – obsfs：并行文件系统，配套使用obsfs挂载。 – s3fs：对象桶，配套使用s3fs挂载。
csi.storage.k8s.io/node-publish-secret-name	否	自定义密钥的名称。 如果您需要为不同OBS存储分配不同的用户权限时，可通过选择不同的Secret实现更灵活的权限控制（推荐使用）。具体使用请参见**对象存储卷挂载设置自定义访问密钥（AK/SK）**。
csi.storage.k8s.io/node-publish-secret-namespace	否	自定义密钥的命名空间。
storage	是	PVC申请容量，单位为Gi。 对对象存储来说，此处仅为校验需要（不能为空和0），设置的大小不起作用，此处设定为固定值1Gi。
storageClassName	是	存储类名称，对象存储对应的存储类名称为csi-obs。

2. 执行以下命令，创建PVC。

```
kubectl apply -f pvc-obs-auto.yaml
```

步骤3 创建应用。

1. 创建web-demo.yaml文件，本示例中将对象存储挂载至/data路径。

```
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    #卷名称，需与volumes字段中的卷名称对应
        mountPath: /data  #存储卷挂载的位置
    imagePullSecrets:
      - name: default-secret
    volumes:
      - name: pvc-obs-volume    #卷名称，可自定义
        persistentVolumeClaim:
          claimName: pvc-obs-auto    #已创建的PVC名称
```

2. 执行以下命令，创建一个挂载对象存储的应用。

```
kubectl apply -f web-demo.yaml
```

工作负载创建成功后，您可以尝试**验证数据持久化及共享性**。

----结束

验证数据持久化及共享性

步骤1 查看部署的应用及文件。

1. 执行以下命令，查看已创建的Pod。

```
kubectl get pod | grep web-demo
```

预期输出如下：

```
web-demo-846b489584-mjhm9   1/1     Running   0          46s
web-demo-846b489584-wvv5s   1/1     Running   0          46s
```

2. 依次执行以下命令，查看Pod的/data路径下的文件。

```
kubectl exec web-demo-846b489584-mjhm9 -- ls /data
kubectl exec web-demo-846b489584-wvv5s -- ls /data
```

两个Pod均无返回结果，说明/data路径下无文件。

步骤2 执行以下命令，在/data路径下创建static文件。

```
kubectl exec web-demo-846b489584-mjhm9 --  touch /data/static
```

步骤3 执行以下命令，查看/data路径下的文件。

```
kubectl exec web-demo-846b489584-mjhm9 -- ls /data
```

预期输出如下：

```
static
```

步骤4 验证数据持久化

1. 执行以下命令，删除名称为web-demo-846b489584-mjhm9的Pod。

```
kubectl delete pod web-demo-846b489584-mjhm9
```

预期输出如下：

```
pod "web-demo-846b489584-mjhm9" deleted
```

删除后，Deployment控制器会自动重新创建一个副本。

2. 执行以下命令，查看已创建的Pod。

```
kubectl get pod | grep web-demo
```

预期输出如下，web-demo-846b489584-d4d4j为新建的Pod：


```
web-demo-846b489584-d4d4j   1/1     Running   0          110s
web-demo-846b489584-wvv5s   1/1     Running   0          7m50s
```

3. 执行以下命令，验证新建的Pod中/data路径下的文件是否更改。

```
kubectl exec web-demo-846b489584-d4d4j -- ls /data
```

预期输出如下：

```
static
```

static文件仍然存在，则说明数据可持久化保存。

步骤5 验证数据共享性

1. 执行以下命令，查看已创建的Pod。

```
kubectl get pod | grep web-demo
```

预期输出如下：

```
web-demo-846b489584-d4d4j   1/1     Running   0          7m
web-demo-846b489584-wvv5s   1/1     Running   0          13m
```

2. 执行以下命令，在任意一个Pod的/data路径下创建share文件。本例中选择名为web-demo-846b489584-d4d4j的Pod。

```
kubectl exec web-demo-846b489584-d4d4j --  touch /data/share
```

并查看该Pod中/data路径下的文件。

```
kubectl exec web-demo-846b489584-d4d4j -- ls /data
```

预期输出如下：

```
share
static
```

3. 由于写入share文件的操作未在名为web-demo-846b489584-wvv5s的Pod中执行，在该Pod中查看/data路径下是否存在文件以验证数据共享性。

```
kubectl exec web-demo-846b489584-wvv5s -- ls /data
```

预期输出如下：

```
share
static
```

如果在任意一个Pod中的/data路径下创建文件，其他Pod下的/data路径下均存在此文件，则说明两个Pod共享一个存储卷。

----结束

相关操作

您还可以执行表4-42中的操作。

表 4-42 其他操作

操作	说明	操作步骤
更新访问密钥	通过CCE控制台更新对象存储的访问密钥。	1. 在左侧导航栏选择“存储”，在右侧选择“存储卷声明”页签。单击PVC操作列的“更多 > 更新访问密钥”。 2. 上传.csv格式的密钥文件，详情请参见获取访问密钥。单击“确定”。

操作	说明	操作步骤
事件	查看PVC或PV的事件名称、事件类型、发生次数、Kubernetes事件、首次和最近发生的时间，便于定位问题。	1. 在左侧导航栏选择“存储”，在右侧选择“存储卷声明”或“存储卷”页签。 2. 单击目标实例操作列的“事件”，即可查看1小时内的事件（事件保存时间为1小时）。
查看YAML	可对PVC或PV的YAML文件进行查看、复制和下载。	1. 在左侧导航栏选择“存储”，在右侧选择“存储卷声明”或“存储卷”页签。 2. 单击目标实例操作列的“查看YAML”，即可查看或下载YAML。

4.6.4 设置对象存储挂载参数

本章节主要介绍如何设置对象存储的挂载参数。您可以在PV中设置挂载参数，然后通过PVC绑定PV，也可以在StorageClass中设置挂载参数，然后使用StorageClass创建PVC，动态创建出的PV会默认带有StorageClass中设置的挂载参数。

对象存储挂载参数

CCE Autopilot在挂载对象存储时默认设置了表4-43和表4-44的参数，其中表4-43中的参数不可取消。

表 4-43 默认使用且不可取消的挂载参数

参数	参数值	描述
use_ino	无需填写	使用该选项，由obsfs分配inode编号。读写模式下自动开启。
big_writes	无需填写	配置后可更改写缓存最大值大小
nonempty	无需填写	允许挂载目录非空
allow_other	无需填写	允许其他用户访问并行文件系统
no_check_certificate	无需填写	不校验服务端证书
sigv2	无需填写	签名版本。对象桶自动使用。
public_bucket	1	设置为1时匿名挂载公共桶。对象桶只读模式下自动使用。

表 4-44 默认使用且可修改的挂载参数

参数	参数值	描述
max_write	131072	仅配置big_writes的情况下才生效，推荐使用128KB。
ssl_verify_hostname	0	不根据主机名验证SSL证书。
max_background	100	可配置后台最大等待请求数。并行文件系统自动使用。
umask	0	配置文件权限的掩码。 例如，如果umask值为022，而目录最大权限为777，则设置umask后该目录权限为777 - 022 = 755，即rwxr-xr-x。

在 PV 中设置挂载参数

在PV中设置挂载参数可以通过mountOptions字段实现，如下所示，mountOptions支持挂载的字段请参见**对象存储挂载参数**。

步骤1 使用kubectl连接集群，详情请参见**通过kubectl连接集群**。

步骤2 在PV中设置挂载参数，示例如下：

```
apiVersion: v1
kind: PersistentVolume
metadata:
  annotations:
    pv.kubernetes.io/provisioned-by: everest-csi-provisioner
    everest.io/reclaim-policy: retain-volume-only      # 可选字段，删除PV，保留底层存储卷
  name: pv-obs       # PV的名称
spec:
  accessModes:
  - ReadWriteMany    # 访问模式，对象存储必须为ReadWriteMany
  capacity:
    storage: 1Gi     # 对象存储容量大小
  csi:
    driver: obs.csi.everest.io        # 挂载依赖的存储驱动
    fsType: obsfs                     # 实例类型
    volumeHandle: <your_volume_id>    # 对象存储的名称
    volumeAttributes:
      storage.kubernetes.io/csiProvisionerIdentity: everest-csi-provisioner
      everest.io/obs-volume-type: STANDARD
      everest.io/region: <your_region>                    # 对象存储的区域
      everest.io/enterprise-project-id: <your_project_id>    # 可选字段，企业项目ID，如果指定企业项目，则创建PVC时也需要指定相同的企业项目，否则PVC无法绑定PV。
    nodePublishSecretRef:             # 设置对象存储的自定义密钥
      name: <your_secret_name>       # 自定义密钥的名称
      namespace: <your_namespace>    # 自定义密钥的命名空间
  persistentVolumeReclaimPolicy: Retain    # 回收策略
  storageClassName: csi-obs             # 存储类名称
  mountOptions:                         # 挂载参数
  - umask=027
```

步骤3 PV创建后，可以创建PVC关联PV，然后在工作负载的容器中挂载，具体操作步骤请参见**通过静态存储卷使用已有对象存储**。

----结束

在 StorageClass 中设置挂载参数

在StorageClass中设置挂载参数同样可以通过mountOptions字段实现，如下所示，mountOptions支持挂载的字段请参见**对象存储挂载参数**。

步骤1 使用kubectl连接集群，详情请参见**通过kubectl连接集群**。

步骤2 创建自定义的StorageClass，示例如下：

```
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:                    # 挂载参数
- umask=027
```

步骤3 StorageClass设置好后，就可以使用这个StorageClass创建PVC，动态创建出的PV会默认带有StorageClass中设置的挂载参数，具体操作步骤请参见**通过静态存储卷使用已有对象存储**。

----结束

4.6.5 对象存储卷挂载设置自定义访问密钥（AK/SK）

背景信息

CCE提供了设置自定义访问密钥的能力，可以让IAM用户使用自己的访问密钥挂载对象存储卷，从而可以对OBS进行访问权限控制（具体请参见**OBS不同权限控制方式的区别**）。

约束与限制

对象存储卷使用自定义访问密钥（AK/SK）时，对应的AK/SK不允许删除或禁用，否则业务容器将无法访问已挂载的对象存储。

获取访问密钥

步骤1 登录控制台。

步骤2 鼠标指向界面右上角的登录用户名，在下拉列表中单击“我的凭证”。

步骤3 在左侧导航栏单击“访问密钥”。

步骤4 单击“新增访问密钥”，进入“新增访问密钥”页面。

步骤5 单击“确定”，下载访问密钥。

----结束

使用访问密钥创建 Secret

步骤1 获取访问密钥。

步骤2 对访问密钥进行base64编码（假设上文获取到的ak为“xxx”，sk为“yyy”）。

echo -n xxx|base64

echo -n yyy|base64

记录编码后的AK和SK。

步骤3 新建一个secret的yaml，如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
```

其中：

参数	描述
access.key	base64编码后的ak。
secret.key	base64编码后的sk。
name	secret的名称
namespace	secret的命名空间
secret.kubernetes.io/used-by: csi	带上这个标签才能在控制台上创建OBS PV/PVC时可见。
type	密钥类型，该值必须为cfe/secure-opaque 使用该类型，用户输入的数据会自动加密。

步骤4 创建Secret。

kubectl create -f test-user.yaml

----结束

静态创建对象存储卷时指定挂载 Secret

使用访问密钥创建Secret后，在创建PV时只需要关联上Secret，就可以使用Secret中的访问密钥（AK/SK）挂载对象存储卷。

步骤1 登录OBS控制台，创建对象存储桶，记录桶名称和存储类型，以并行文件系统为例。

步骤2 新建一个pv的yaml文件，如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: cn-north-4
      storage.kubernetes.io/csiProvisionerIdentity: everest-csi-provisioner
    volumeHandle: obs-normal-static-pv
  persistentVolumeReclaimPolicy: Delete
  storageClassName: csi-obs
```

参数	描述
nodePublishSecretRef	挂载时指定的密钥，其中 • name：指定secret的名字 • namespace：指定secret的命令空间
fsType	文件类型，支持“obsfs”与“s3fs”，取值为s3fs时创建的是obs对象桶，配套使用s3fs挂载；取值为obsfs时创建的是obs并行文件系统，配套使用obsfs挂载。
volumeHandle	对象存储的桶名称。

步骤3 创建PV。

kubectl create -f pv-example.yaml

PV创建完成后，就可以创建PVC关联PV。

步骤4 新建一个PVC的yaml文件，如pvc-example.yaml。

PVC yaml文件配置示例：

```
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
```

参数	描述
csi.storage.k8s.io/node-publish-secret-name	指定secret的名字
csi.storage.k8s.io/node-publish-secret-namespace	指定secret的命令空间

步骤5 创建PVC。

kubectl create -f pvc-example.yaml

PVC创建后，就可以创建工作负载挂载PVC使用存储。

----结束

动态创建对象存储卷时指定挂载密钥

动态创建对象存储卷时，可通过如下方法指定挂载密钥。

步骤1 新建一个pvc的yaml文件，如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
```

参数	描述
csi.storage.k8s.io/node-publish-secret-name	指定secret的名字
csi.storage.k8s.io/node-publish-secret-namespace	指定secret的命令空间

步骤2 创建PVC。

kubectl create -f pvc-example.yaml

PVC创建后，就可以创建工作负载挂载PVC使用存储。

----结束

配置验证

根据上述步骤，使用IAM用户的密钥挂载对象存储卷。假设工作负载名称为obs-secret，容器内挂载目录是/temp，IAM用户权限为CCE ReadOnlyAccess和Tenant Guest。

1. 查询工作负载实例名称。

   kubectl get po | grep obs-secret

   期望输出：

   ```
   obs-secret-5cd558f76f-vxslv        1/1     Running   0          3m22s
   ```

2. 查询挂载目录下对象，查询正常。

   kubectl exec obs-secret-5cd558f76f-vxslv -- ls -l /temp/

3. 尝试在挂载目录内写入数据，写入失败。

   kubectl exec obs-secret-5cd558f76f-vxslv -- touch /temp/test

   期望输出：

   ```
   touch: setting times of '/temp/test': No such file or directory
   command terminated with exit code 1
   ```

4. 参考桶策略配置，给挂载桶的子用户设置读写权限。

5. 再次尝试在挂载目录内写入数据，写入成功。

   kubectl exec obs-secret-5cd558f76f-vxslv -- touch /temp/test

6. 查看容器内挂载目录，验证数据写入成功。

   kubectl exec obs-secret-5cd558f76f-vxslv -- ls -l /temp/

   期望输出：

   ```
   -rwxrwxrwx 1 root root 0 Jun  7 01:52 test
   ```

4.7 临时路径（EmptyDir）

临时路径是Kubernetes原生的EmptyDir类型，生命周期与容器实例相同，并支持指定内存作为存储介质。容器实例消亡时，EmptyDir会被删除，数据会永久丢失。

通过控制台使用临时路径

步骤1 登录CCE控制台，单击集群名称进入集群。

步骤2 在左侧导航栏中选择“工作负载”，在右侧选择“无状态负载”页签。

步骤3 单击页面右上角“创建工作负载”，在“容器配置”中选择“数据存储”页签，并单击“添加存储卷 > 临时路径(EmptyDir)”。

本文主要为您介绍存储卷的挂载使用，如**表4-45**，其他参数详情请参见**创建工作负载**。

表 4-45 临时路径挂载

参数	参数说明
存储介质	开启内存： • 开启后可以使用内存提高运行速度，但存储容量受内存大小限制。适用于数据量少，读写效率要求高的场景。 • 未开启时默认存储在硬盘上，适用于数据量大，读写效率要求低的场景。 **说明** • 开启内存后请注意内存大小，如果存储容量超出内存大小会发生OOM事件。 • 使用内存时的EmptyDir的大小为Pod规格限制值的100%。 • 不使用内存的EmptyDir不会占用系统内存。
挂载路径	请输入挂载路径，如：/tmp。 数据存储挂载到容器上的路径。请不要挂载在系统目录下，如“/”、“**/var/run**”等，会导致容器异常。建议挂载在空目录下，若目录不为空，请确保目录下无影响容器启动的文件，否则文件会被替换，导致容器启动异常，工作负载创建失败。 **须知** 挂载高危目录的情况下 ，建议使用低权限账号启动，否则可能会造成宿主机高危文件被破坏。
子路径	请输入存储卷的子路径，将存储卷中的某个路径挂载至容器，可以实现在单一Pod中使用同一个存储卷的不同文件夹。如：tmp，表示容器中挂载路径下的数据会存储在存储卷的tmp文件夹中。不填写时默认为根路径。
权限	• 只读：只能读容器路径中的数据卷。 • 读写：可修改容器路径中的数据卷，容器迁移时新写入的数据不会随之迁移，会造成数据丢失。

步骤4 其余工作负载参数都配置完成后，单击“创建工作负载”。

----结束

通过 kubectl 使用临时路径

步骤1 请参见**通过kubectl连接集群**配置kubectl命令。

步骤2 创建并编辑nginx-emptydir.yaml文件。

vi nginx-emptydir.yaml

YAML文件内容如下：

```
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     # 卷名称，需与volumes字段中的卷名称对应
            mountPath: /tmp        # emptyDir挂载路径
    imagePullSecrets:
      - name: default-secret
    volumes:
      - name: vol-emptydir         # 卷名称，可自定义
        emptyDir:
          medium: Memory           # emptyDir磁盘介质：设置为Memory时，表示开启内存；设置为空时为原生默认的存储介质类型
          sizeLimit: 1Gi           # 卷容量大小
```

步骤3 创建工作负载。

kubectl apply -f nginx-emptydir.yaml

----结束

4.8 增加 Pod 的临时存储容量

工作负载中的每个Pod默认提供30GiB（IOPS上限2500，IOPS突发上限16000）的免费磁盘空间，除系统本身及平台预留资源占用外，用户实际可用的镜像、容器及临时存储的空间总大小约为20GiB。当此存储容量无法满足您的需求时，您可以通过控制台或kubectl命令行的方式自定义增加Pod临时存储容量。新增的临时存储容量按照大小和使用时长计费，更多计费信息请参考**计费说明**。

约束与限制

集群版本要求在1.27.8-r0、1.28.6-r0及以上。如果您的集群版本不符合该要求，则需要通过集群升级使用该功能。

增加临时存储容量

本节将介绍两种方式增加Pod的临时存储容量，即控制台方式和kubectl命令行方式。

使用控制台方式

本部分将向您介绍如何通过控制台增加Pod的临时存储容量，重点说明该功能的相关参数。关于工作负载的其他参数，请参见**创建工作负载**。

步骤1 登录**CCE控制台**，单击集群名称进入集群。

步骤2 在左侧导航栏中选择“工作负载”。

- 如果您需要新创建一个工作负载，并增加Pod的临时存储容量，请选择对应负载类型，单击页面右上角“创建工作负载”。
- 如果您需要增加已有的工作负载的Pod临时存储容量，请找到对应的工作负载名称，单击右侧“升级”。

步骤3 您可以通过两种方式增加临时存储容量。

表 4-46 增加临时存储容量的不同方式

方式	示例	步骤
添加“Pod注解”	**键：resource.cce.io/extra-ephemeral-storage-in-GiB** **值：10**	1. 选择“高级配置 > 标签与注解”页签。 2. 在Pod注解的“键”中输入“resource.cce.io/extra-ephemeral-storage-in-GiB”，用来表示额外增加的Pod的临时存储容量。 3. 在Pod注解的“值”中，填入需要增加的存储容量，取值范围为0-994。单击“确定添加”。 **说明** 在Pod注解中添加上述注解后，“规格确认”中的“单Pod临时存储”会由默认的30GiB变为40GiB，即“单Pod临时存储=默认的30GiB+增加的临时存储容量”。 **图 4-17** 增加 Pod 的临时存储容量  **图 4-18** 单 Pod 临时存储 

方式	示例	步骤
修改“单Pod临时存储”	**40GiB**	1. “规格确认 > 单Pod临时存储”，单击“编辑图标”。 2. 在对话框中输入您需要的容量，该容量=默认的30GiB+增加的临时存储容量，取值范围为30-1024。 **须知** 如果无法确定需要为Pod增加多少临时存储容量，您可以直接保持系统推荐值，即“ 单Pod临时存储”的取值。当选择镜像后，系统会根据选择的镜像大小和系统占用计算推荐值，计算公式：Pod临时存储推荐值=max（30GiB免费额度，系统占用+镜像大小之和*2）。 **图 4-19** 修改单 Pod 临时存储  3. 单击“确定”。

步骤4 勾选“我已知晓上述计费规则”，单击“创建工作负载”或“升级工作负载”。

----结束

使用 kubectl 命令行方式

本部分将向您介绍如何通过kubectl增加Pod的临时存储容量，重点说明该功能的相关参数。关于工作负载的其他参数，请参见**创建工作负载**。接下来，分别介绍“创建工作负载，并增加Pod的临时存储容量”和“增加已有的工作负载的Pod临时存储容量”的两种情况。

说明

该步骤涉及命令行操作，您可以使用以下两种方式进行相关操作：

- 通过集群内命令行工具进行操作，该命令行工具已经配置kubectl命令，并已连接集群。
- 通过ECS虚拟机进行操作，该ECS需与集群处于同一VPC，并**通过kubectl连接集群**。

创建工作负载，并增加Pod的临时存储容量

1. 创建YAML文件extra-ephemeral-storage-test.yaml，用于配置工作负载，文件名称可自定义。

```
vim extra-ephemeral-storage-test.yaml
```

文件内容如下：

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: extra-ephemeral-storage-test     # 工作负载名称
spec:
  replicas: 1     # 实例数量
  selector:
```



```
  matchLabels:  # 选择器，用于选择带有特定标签的资源
    app: nginx
template:
  metadata:
    labels:     # 标签
      app: nginx
    annotations:
      resource.cce.io/extra-ephemeral-storage-in-GiB: '10'
  spec:
    containers:
    - image: nginx:latest   # 镜像名称：镜像版本
      name: nginx
    imagePullSecrets:
    - name: default-secret
```

其中，“**resource.cce.io/extra-ephemeral-storage-in-GiB**”表示额外增加的Pod的临时存储容量，取值范围为0-994，例如“10”。Pod的临时存储总容量为默认30GiB与该值之和。

输入完成后，Esc键退出编辑，输入:wq保存。

2. 创建工作负载。

```
kubectl create -f extra-ephemeral-storage-test.yaml
```

3. 查询工作负载状态。

```
kubectl get deployment
```

回显如下，如果READY为1/1，则表示创建成功。此时工作负载内的Pod的临时存储容量已经变为您设置的值。

```
NAME         READY   UP-TO-DATE   AVAILABLE   AGE
nova-test    1/1     1            1           59s
```

4. 验证工作负载内的Pod的临时存储容量是否增加成功。

通过以下命令，查询工作负载对应的Pod名称，extra-ephemeral-storage-test可以换为工作负载的名称。

```
kubectl get pod | grep extra-ephemeral-storage-test
```

回显如下：

```
extra-ephemeral-storage-test-85dbdb8c5d-4vftc     1/1     Running   0          2m24s
```

通过以下命令，进入该Pod内部。

```
kubectl exec -it extra-ephemeral-storage-test-85dbdb8c5d-4vftc /bin/sh
```

查询该Pod的临时存储容量。

```
lsblk
```

回显结果如下，Pod的存储总容量为40G，说明Pod扩容成功。输入exit，按下Enter键，退出Pod。

```
NAME    MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
vda     254:0    0   40G  0 disk
├─vda1 254:1    0   59M  0 part
├─vda2 254:2    0  1.5G  0 part
├─vda3 254:3    0  1.5G  0 part
└─vda4 254:4    0   37G  0 part /etc/resolv.conf
```

增加已有的工作负载的Pod临时存储容量

1. 修改对应工作负载的YAML文件，添加resource.cce.io/extra-ephemeral-storage-in-GiB的annotations。以名为extra-ephemeral-storage-test的无状态工作负载为例。

```
kubectl edit deployment extra-ephemeral-storage-test
```

在下方位置，添加resource.cce.io/extra-ephemeral-storage-in-GiB的annotations。输入完成后，Esc键退出编辑，输入:wq保存。

```
template:
  metadata:
    creationTimestamp: null
    labels:
```



```
    app: nginx
  annotations:
    resource.cce.io/extra-ephemeral-storage-in-GiB: '10'
```

其中，“**resource.cce.io/extra-ephemeral-storage-in-GiB**”表示额外增加的Pod的临时存储容量，取值范围为0-994，例如“10”。Pod的临时存储总容量为默认30GiB与该值之和。

2. 查询工作负载状态。

```
kubectl get deployment
```

回显如下，如果READY为1/1，则表示工作负载升级成功。此时工作负载内的Pod的临时存储容量已经变为您设置的值。

```
NAME         READY   UP-TO-DATE   AVAILABLE   AGE
nova-test    1/1     1            1           59m
```

3. 验证工作负载内的Pod的临时存储容量是否增加成功。

通过以下命令，查询工作负载对应的Pod名称，extra-ephemeral-storage-test可以换为工作负载的名称。

```
kubectl get pod | grep extra-ephemeral-storage-test
```

回显如下：

```
extra-ephemeral-storage-test-85dbdb8c5d-4vftc     1/1     Running   0          60m24s
```

通过以下命令，进入该Pod内部。

```
kubectl exec -it extra-ephemeral-storage-test-85dbdb8c5d-4vftc /bin/sh
```

查询该Pod的临时存储容量。

```
lsblk
```

回显结果如下，Pod的存储总容量为40G，说明Pod扩容成功。输入exit，按下Enter键，退出Pod。

```
NAME    MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
vda     254:0    0   40G  0 disk
├─vda1 254:1    0   59M  0 part
├─vda2 254:2    0  1.5G  0 part
├─vda3 254:3    0  1.5G  0 part
└─vda4 254:4    0   37G  0 part /etc/resolv.conf
```

5 云原生观测

5.1 监控中心

5.1.1 开通监控中心

开通监控中心将在集群中安装云原生监控插件，该插件提供监控中心的指标采集功能。开通后，监控中心将采集集群中的指标并上报至AOM实例。本章节介绍如何为集群开通监控中心功能。

须知

- 开通监控中心后，集群中的指标将上报至AOM实例，AOM针对基础指标免费，自定义指标由AOM服务收费，具体请参考**价格详情**。
- 云原生监控插件在集群中运行需要消耗集群资源，请确保集群资源能够满足插件的安装。具体资源消耗可以前往“插件中心”云原生监控插件安装页面获取。

前提条件

开通监控中心前，用户需要使用具有admin用户组的账户完成对CCE及其依赖服务的委托授权。

授权方式：监控中心页面自动弹出“确认授权”页面，用户单击“确认授权”按钮后系统自动完成授权。

约束与限制

使用监控中心前，用户需要使用具有admin用户组的账户完成对CCE及其依赖服务的委托授权。授权完成后，拥有CCE Administrator角色或CCE FullAccess权限的用户可进行监控中心所有操作；拥有CCE ReadOnlyAccess权限的用户可以查看所有资源信息，但是无法进行任何操作。

开通监控中心

- **购买集群时开通**
  a. 登录云容器引擎控制台，购买集群。
  b. 在“插件选择”页面，勾选云原生监控插件。
  c. 在“插件配置”页面，选择云原生监控插件需要对接的AOM实例。如AccessCode未创建，请先创建AccessCode。

  图 5-1 启用容器监控

- **在监控中心页面开通**
  a. 在目标集群左侧导航栏选择“监控中心”。
  b. 单击“立即开通”，并选择指标上报的AOM实例。

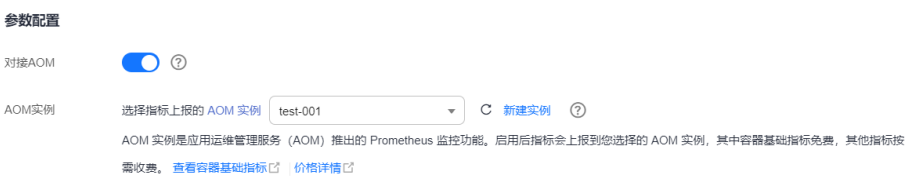
  图 5-2 开通监控中心

  c. 开通成功后，等待3-5分钟，监控数据将上报至AOM实例，随即可以使用监控中心相关功能。

- **在插件管理页面开通**
  a. 在目标集群左侧导航栏选择“插件中心”。
  b. 选择云原生监控插件，单击“安装”。
  c. 开启对接AOM开关，指标将上报至AOM实例。

  图 5-3 安装云原生监控插件

  d. 插件安装完成3-5分钟后，监控数据将上报至AOM实例，随即可以使用监控中心相关功能。

说明

如需关闭监控中心，请前往CCE控制台“插件管理”页面卸载云原生监控插件，或关闭AOM对接，即可以停止使用该功能。

5.1.2 集群监控

当您想观测整个集群的资源使用情况和健康度时，可以在“集群”页面查看，该页面提供了单个集群的监控情况，包含**资源健康概况**、**资源消耗Top统计**和**数据面监控**多维度的信息概况。

功能入口

步骤1 登录CCE控制台，单击集群名称进入集群详情页。

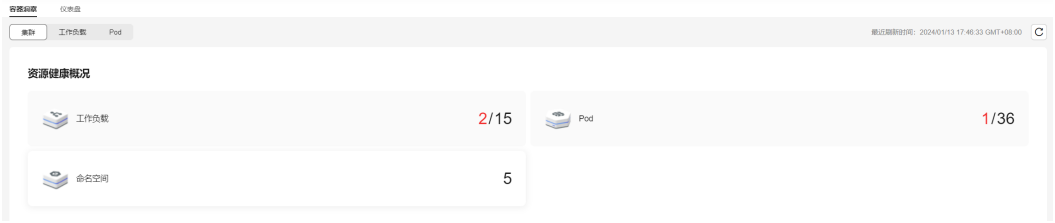
步骤2 在左侧导航栏中选择“监控中心”，单击“集群”。

----结束

资源健康概况

资源概况涵盖了工作负载和Pod资源中异常资源所占比例，以及命名空间的总数。

图 5-4 资源概况



资源消耗 Top 统计

在资源消耗Top统计中，CCE服务会将CPU使用率和内存使用率排名前五的无状态负载、有状态负载和Pod纳入统计范围，以帮助您识别资源消耗“大户”。如果您需要查看全部数据，可前往**工作负载**或**Pod**页面。

图 5-5 资源消耗 Top 统计

监控名词解释：

- CPU使用率
  工作负载CPU使用率 = 工作负载各个Pod中CPU使用率的平均值
  Pod CPU使用率 = Pod实际使用的CPU核数 / 业务容器CPU核数限制值之和

- 内存使用率
  工作负载内存使用率 = 工作负载各个Pod中内存使用率的平均值
  Pod内存使用率 = Pod实际使用的物理内存 / 业务容器物理内存限制值之和

数据面监控

此处默认统计近1小时、近8小时和近24小时的各维度资源用量。如需查看更多监控信息，请单击“查看全部监控”，跳转至“仪表盘”页面，相应指导请参见**使用仪表盘**。

说明

您可以将鼠标悬停在图表上，以便查看每分钟的监控数据。

- **Pod数量状态趋势**：实时监控集群Pod的状态。
- **Pod总重启次数趋势**：近5分钟的集群的Pod重启次数总和。

5.1.3 工作负载监控

如果您需要监控工作负载的资源使用情况，可以前往“工作负载”页面查看。该页面提供了指定集群下所有工作负载的综合信息，以及单个工作负载的详细监控数据，包括CPU/内存使用率、网络流入/流出速率等。

功能入口

步骤1 登录CCE控制台，单击集群名称进入集群详情页。

步骤2 在左侧导航栏中选择“监控中心”，单击“工作负载”。

页面呈现了所有工作负载的综合信息，如需深入了解单个工作负载的监控情况，可单击工作负载名称，进入该工作负载的“概览”页面，通过切换“Pod列表”、“监控”页签查看相应内容。

----结束

工作负载列表

工作负载列表中包含工作负载名称、状态、Pod个数（正常/全部）、命名空间、镜像名称、CPU使用率，以及内存使用率等信息。

图 5-6 工作负载列表

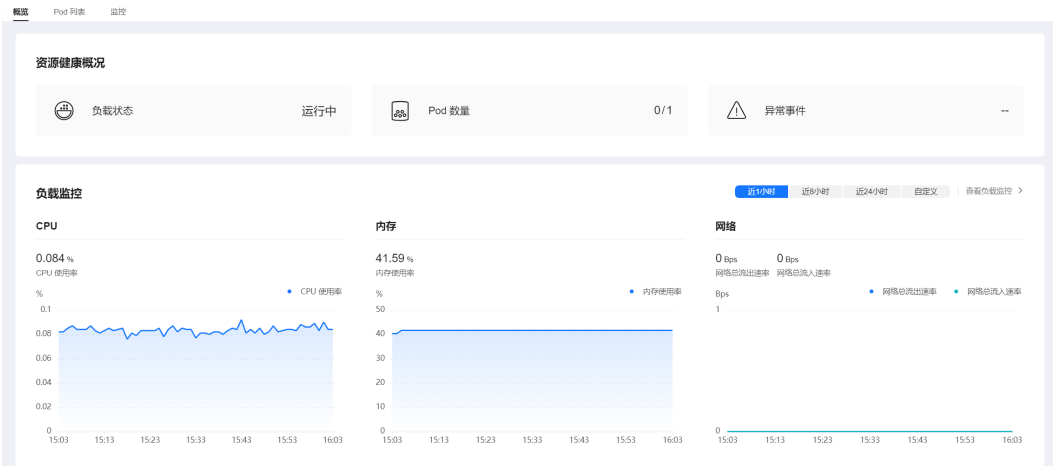
您可以利用页面右上角的工作负载类型，以及列表上方的工作负载名称、状态和命名空间进行筛选，快速定位所需的工作负载。

您也可以单击“导出”按钮来导出全部工作负载数据，或者选择部分工作负载进行导出，此时仅导出所选中的数据。导出的文件为“.xlsx”格式，文件命名中包含时间戳。

概览

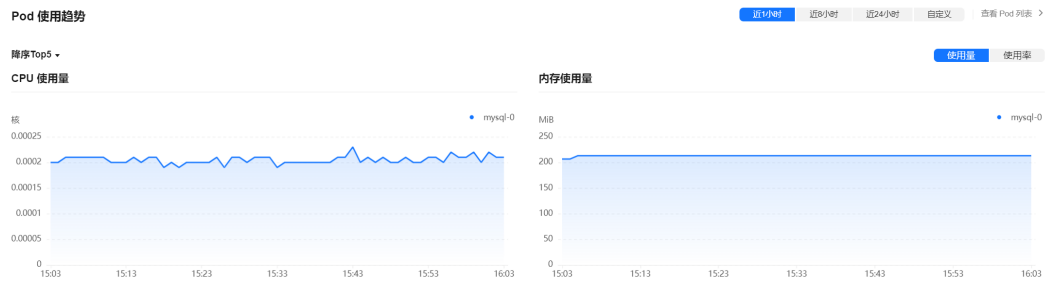
单击工作负载名称，您可以方便地查看资源概况，包括负载状态、Pod数量（异常/总数）以及异常事件。此外，还可以浏览近一小时的监控概览，其中包括CPU使用率、内存使用率和网络流入/流出速率这些常见的监控指标。

图 5-7 资源概况和监控概览



同时，概览页面还提供了Pod使用趋势功能，您可以从中了解工作负载中各Pod的CPU使用率、CPU使用量、内存使用率和内存使用量（在图表右上角切换对应指标），并且支持查看降序Top5和升序Top5数据（在图表左上角进行切换）。

图 5-8 Pod 使用趋势



如需了解更多指标，请前往**监控**页面查看。

Pod 列表

Pod列表中包含了Pod名称、状态、命名空间、Pod IP、所在节点、重启次数、CPU申请/限制、内存申请/限制，以及CPU和内存使用率等详细信息。

图 5-9 Pod 列表

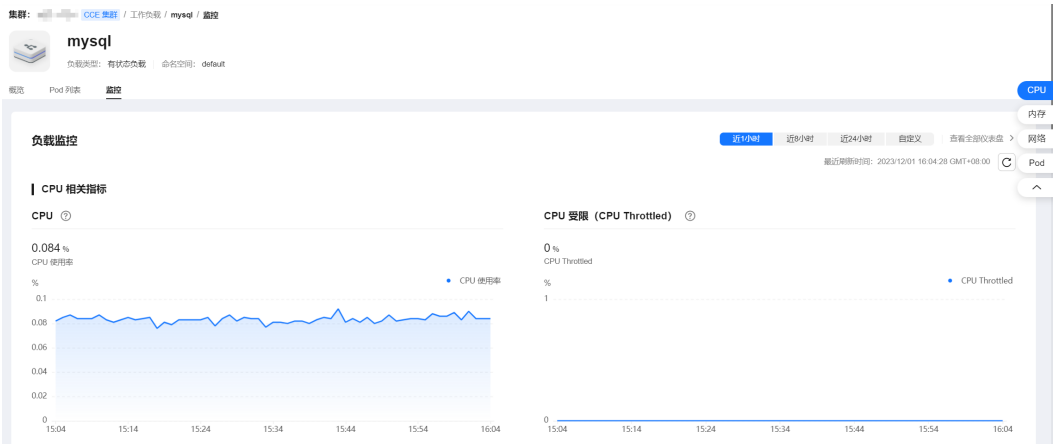
您可以通过在列表上方按照Pod名称、状态、命名空间、Pod IP和所在节点进行筛选，快速找到需要的Pod。您也可以单击“导出”按钮来导出全部Pod数据，或者选择部分Pod进行导出，此时仅导出所选中的数据。导出的文件为“.xlsx”格式，文件命名中包含时间戳。

单击实例名称可以查看实例的详细监控数据。更多相关内容，请参见**Pod监控**。

监控

在此处，您可以方便地查看工作负载在近1小时、近8小时、近24小时以及自定义时间段内各维度资源的使用情况。如需查看更多监控信息，请单击“查看全部仪表盘”，跳转至“仪表盘”页面，相应指导请参见**使用仪表盘**。

图 5-10 工作负载监控



- **CPU相关指标**
  - CPU：负载的所有 Pod 的容器在不同的时间段使用的 CPU 总量占负载的所有 Pod 的容器的 CPU Limit 总量的比例。
  - CPU 受限（CPU Throttled）：负载的所有 Pod 的容器在不同的时间段的 CPU 受限时间所占的平均比例。
- **内存相关指标**
  - 内存使用率：负载的所有 Pod 的容器在不同的时间段使用的内存总量占负载的所有 Pod 的容器的内存 Limit 总量比例。
- **网络相关指标**
  - 网络总流出速率：负载的所有 Pod 的容器在不同的时间段的每秒钟发送的总字节数。
  - 网络总流入速率：负载的所有 Pod 的容器在不同的时间段的每秒钟接收的总字节数。
  - 网络发送丢包率：负载的所有 Pod 的容器在不同的时间段的发送丢失的数据包总量占发送的数据包总量的比例。
  - 网络接收丢包率：负载的所有 Pod 的容器在不同的时间段的接收丢失的数据包总量占接收的数据包总量的比例。
- **Pod相关指标**
  - Pod CPU使用率：负载的每个 Pod 在不同的时间段的CPU使用量除以它们的 CPU Limit 量。

- Pod内存使用率：负载的每个 Pod 在不同的时间段的内存使用量除以它们的内存 Limit 量。
- Pod状态数量趋势：负载在不同的时间段分别处于不可用、未就绪、运行中、已完成或其他的状态 Pod 数量之和。
- Pod数量变化趋势：负载的 Pod（副本）在不同的时间段的数量。

5.1.4 Pod 监控

如果您需要监控Pod的资源使用情况，可以前往“Pod”页面查看。该页面提供了指定集群下所有Pod的综合信息，以及单个Pod的详细监控数据，包括CPU/内存使用率、网络流入/流出速率等。

功能入口

步骤1 登录CCE控制台，单击集群名称进入集群详情页。

步骤2 在左侧导航栏中选择“监控中心”，单击“Pod”。

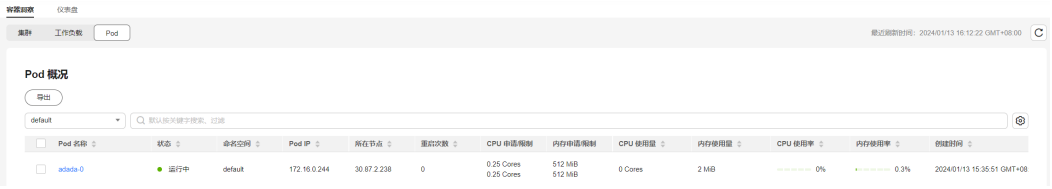
页面呈现了所有Pod的综合信息，如需深入了解单个Pod的监控情况，可单击Pod名称，进入该Pod的“概览”页面，通过切换“容器列表”、“监控”页签查看相应内容。

----结束

Pod 列表

Pod列表中包含Pod名称、状态、命名空间、Pod IP、所在节点、重启次数、CPU申请/限制、内存申请/限制、CPU使用率，以及内存使用率等信息。

图 5-11 Pod 列表



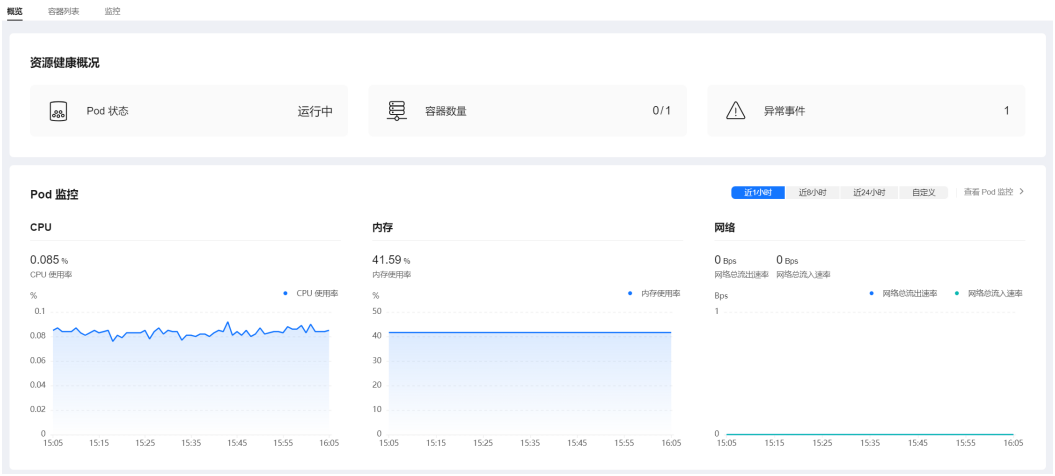
您可以利用列表上方的命名空间，以及搜索栏中的Pod名称、状态、Pod IP和所在节点进行筛选，快速定位所需的Pod。

您也可以单击“导出”按钮来导出全部Pod数据，或者选择部分Pod进行导出，此时仅导出所选中的数据。导出的文件为“.xlsx”格式，文件命名中包含时间戳。

概览

单击Pod名称，您可以方便地查看资源概况，包括Pod状态、容器数量（异常/总数）以及异常事件。此外，还可以浏览Pod近一小时的监控概览，其中包括CPU使用率、内存使用率和网络流入/流出速率这些常见的监控指标。

图 5-12 资源概况和监控概览



同时，概览页面还提供了容器使用趋势功能，您可以从中了解Pod中各容器的CPU使用率、CPU使用量、内存使用率和内存使用量（在图表右上角切换对应指标），并且支持查看降序Top5和升序Top5数据（在图表左上角进行切换）。

如需了解更多指标，请前往**监控**页面查看。

容器列表

容器列表中包含了容器名称、状态、命名空间、重启次数，以及镜像等详细信息。

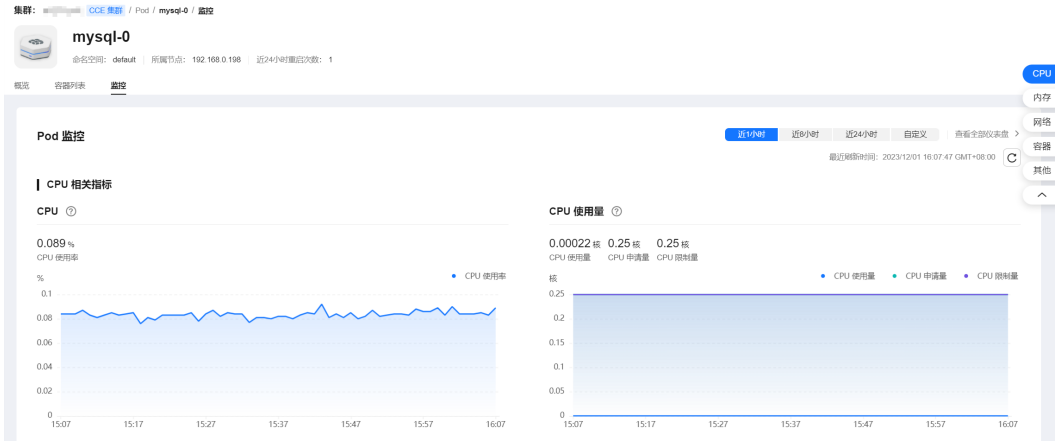
图 5-13 容器列表

您可以通过在列表上方按照容器名称、状态和命名空间进行筛选，快速找到需要的容器。您也可以单击“导出”按钮来导出全部容器数据，或者选择部分容器进行导出，此时仅导出所选中的数据。导出的文件为“.xlsx”格式，文件命名中包含时间戳。

监控

在此处，您可以方便地查看实例在近1小时、近8小时、近24小时以及自定义时间段内各维度资源的使用情况。如需查看更多监控信息，请单击“查看全部仪表盘”，跳转至“仪表盘”页面，相应指导请参见**使用仪表盘**。

图 5-14 Pod 监控



- **CPU相关指标**
  - CPU：Pod 的所有容器在不同的时间段 CPU 使用总量占 Pod 的所有容器 CPU Limit 总量的比例。
  - CPU 使用量：Pod 已经使用的 CPU 核数。
  - CPU 申请量：Pod CPU Request 值。
  - CPU 限制量：Pod CPU Limit 值，使用量接近该值时容器的 CPU 资源会被限流，影响容器性能。
- **内存相关指标**
  - 内存使用率：Pod 的所有容器在不同的时间段内存使用总量占 Pod 的所有容器内存 Limit 总量。
  - 内存使用量：Pod 已经使用的内存量。
  - 内存申请量：Pod 内存 Request 值。
  - 内存限制量：Pod 内存 Limit 值, 使用量到达该值时会导致容器 OOM。
- **网络相关指标**
  - 网络总流出速率：Pod 的所有容器每秒钟发送的总字节数。
  - 网络总流入速率：Pod 的所有容器每秒钟接收的总字节数。
- **容器相关指标**
  - 容器CPU使用率：Pod 的每个容器在不同的时间段的 CPU 使用量占它们的 CPU Limit 量的比例。
  - 容器内存使用率：Pod 的每个容器在不同的时间段的内存使用量占它们的内存 Limit 量的比例。
  - 容器CPU受限：Pod 的每个容器在不同的时间段的 CPU 受限时间所占的比例。
  - 容器网络丢包率：Pod 的每个的容器在不同的时间段接收丢失的数据包总量占接收的数据包总量的比例。
- **其他指标**
  - Pod 历史状态：Pod 在不同时间段所处的状态。
  - 容器历史状态：Pod 的每个容器在不同的时间段所处的状态。

5.1.5 仪表盘

5.1.5.1 使用仪表盘

仪表盘集合了不同视角、不同组件的高频监控指标。将不同的指标以图表的形式直观、综合性地汇集在同一个屏幕上，帮助您实时全面地掌握集群整体运行状况。

仪表盘提供了丰富的视图监控指标呈现，包括集群视图、Pod视图等等。

前提条件

- 集群处于“运行中”状态。
- 集群已开通“监控中心”。

查看/切换视图

步骤1 登录CCE控制台，单击集群名称进入集群详情页。

步骤2 在左侧导航栏中选择“监控中心”，单击“仪表盘”页签，默认展示集群视图。

步骤3 监控中心仪表盘提供了预置视图，您可单击视图名称边上的“切换视图”按钮，选择需要的视图查看监控数据。

步骤4 设置查看视图的相关参数。

步骤5 设置视图的时间窗。

在页面右上角处，选择时间段，或者自定义时间，并单击刷新界面。

----结束

5.1.5.2 集群视图

基于集群的指标和PromQL语句，提供了集群Pod数、容器数、CPU、内存、网络、磁盘等关键资源相关图表，帮助您了解整体集群的资源运行状态。接下来主要从指标说明、指标清单两个部分来进行图表的说明，其中图表中对于数值过大的字节（bytes）会换算为MB、KB、GB等。

指标说明

集群视图暴露的指标包括基础资源指标、网络指标和磁盘指标，具体说明如下：

图 5-15 基础资源图表

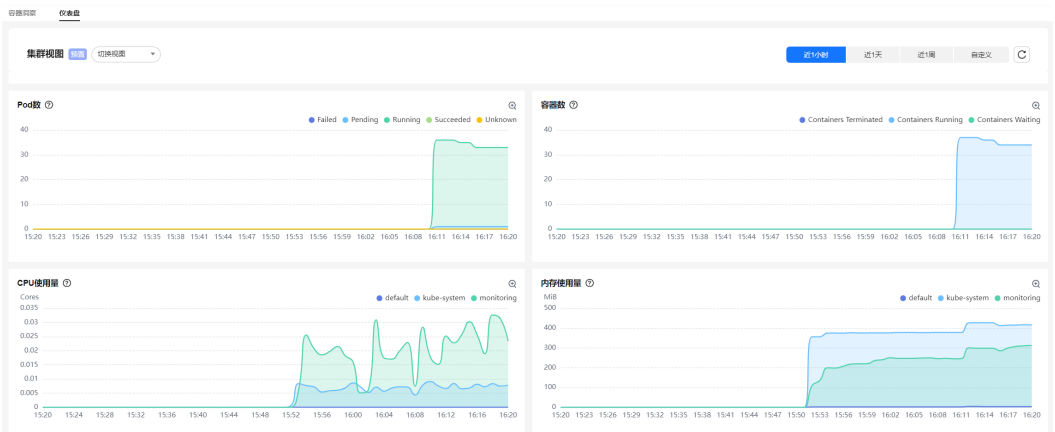


表 5-1 基础资源图表说明

指标名称	单位	说明
Pod数	个	集群中处在不同运行状态下的Pod个数（状态包含：Failed、Pending、Running、Succeeded、Unknown等）
容器数	个	集群中处在不同运行状态下的容器个数（状态包含：Containers Running、Containers Waiting、Containers Terminated等）
CPU使用量	Cores	以命名空间为粒度统计各个命名空间内的所有容器的CPU使用量之和。
内存使用量	字节	以命名空间为粒度统计各个命名空间内的所有容器的内存使用量之和。

图 5-16 网络图表

表 5-2 网络图表说明

指标名称	单位	说明
网络接收速率	字节/秒	以命名空间为粒度统计各个命名空间内的所有容器每秒接收的字节数之和。
网络发送速率	字节/秒	以命名空间为粒度统计各个命名空间内的所有容器每秒传输的字节数之和。
网络平均接收速率	字节/秒	以命名空间为粒度统计各个命名空间内的容器每秒平均接收的字节数。
网络平均发送速率	字节/秒	以命名空间为粒度统计各个命名空间内的容器每秒平均传输的字节数。

指标名称	单位	说明
接收数据包速率	个/秒	以命名空间为粒度统计各个命名空间内的所有容器每秒接收的数据包数之和。
集群发送数据包速率	个/秒	以命名空间为粒度统计各个命名空间内所有容器每秒发送的数据包数之和。
丢包速率（接收）	个/秒	以命名空间为粒度统计各个命名空间内所有容器每秒接收的数据丢包数之和。
丢包速率（发送）	个/秒	以命名空间为粒度统计各个命名空间内所有容器每秒发送的数据丢包数之和。

图 5-17 磁盘图表

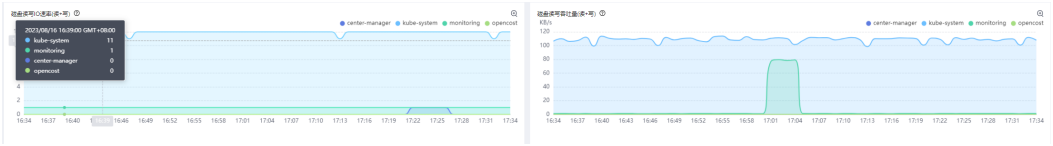


表 5-3 磁盘图表说明

指标说明	单位	说明
磁盘读写IO速率(读+写)	次数/秒	以命名空间为粒度统计各个命名空间内所有容器每秒的磁盘读写IO的次数之和。
磁盘读写吞吐量(读+写)	字节/秒	以命名空间为粒度统计各个命名空间内所有容器每秒的磁盘读写字节量之和。

指标清单

集群视图使用的指标清单如下：

表 5-4 集群视图指标清单

指标	指标类型	说明
kube_pod_container_resource_requests	gauge	容器请求的请求资源数
kube_pod_container_resource_limits	gauge	容器请求的限制资源数
kube_pod_status_phase	gauge	Pod当前阶段
kube_pod_container_status_waiting	gauge	容器是否处在waiting状态

指标	指标类型	说明
kube_pod_container_status_terminated	gauge	容器是否处在终止状态
container_cpu_usage_seconds_total	counter	容器CPU累计使用时间
container_memory_rss	gauge	RSS内存，即常驻内存集。是分配给进程使用的实际物理内存字节数，不是磁盘上缓存的虚机内存。
container_network_receive_bytes_total	counter	容器网络累积接收字节数
container_network_transmit_bytes_total	counter	容器网络累积传输字节数
container_network_receive_packets_total	counter	容器网络收到的累计数据包数
container_network_transmit_packets_total	counter	容器网络传输的累计数据包数
container_network_receive_packets_dropped_total	counter	容器网络接收时丢失的数据包数
container_network_transmit_packets_dropped_total	counter	容器网络传输时丢失的数据包数
container_fs_reads_total	counter	容器磁盘读取次数
container_fs_reads_bytes_total	counter	容器磁盘读取的总字节数

5.1.5.3 Pod 视图

从Pod视角呈现Pod维度集群资源、网络、磁盘等监控情况，帮助您详细了解Pod的运行状态。

指标说明

Pod视图暴露的指标包括Pod资源指标、Pod网络指标和Pod磁盘指标，具体说明如下：

图 5-18 Pod 资源指标

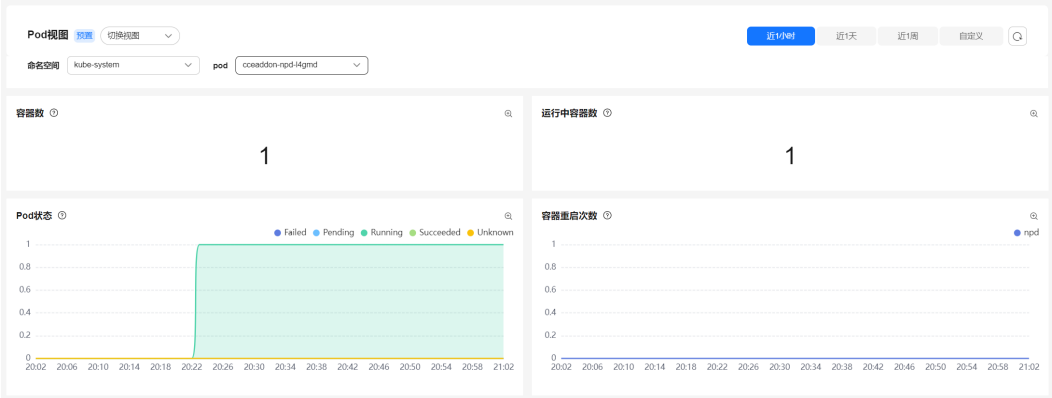


表 5-5 Pod 资源指标说明

指标名称	单位	说明
容器数	个	Pod中的容器总数
运行中容器数	个	Pod中正在运行的容器个数
Pod状态	个	处在不同状态下的Pod个数
容器重启次数	次	容器被重启的次数
CPU使用量	Cores	Pod CPU使用量
CPU 有效率&使用率	百分比	有效率：使用量/请求量；使用率：使用量/总量
内存使用量	字节	内存使用量
内存 有效率&使用率	百分比	有效率：使用量/请求量；使用率：使用量/总量
CPU Throttling	百分比	CPU节流周期限制率

图 5-19 Pod 网络指标

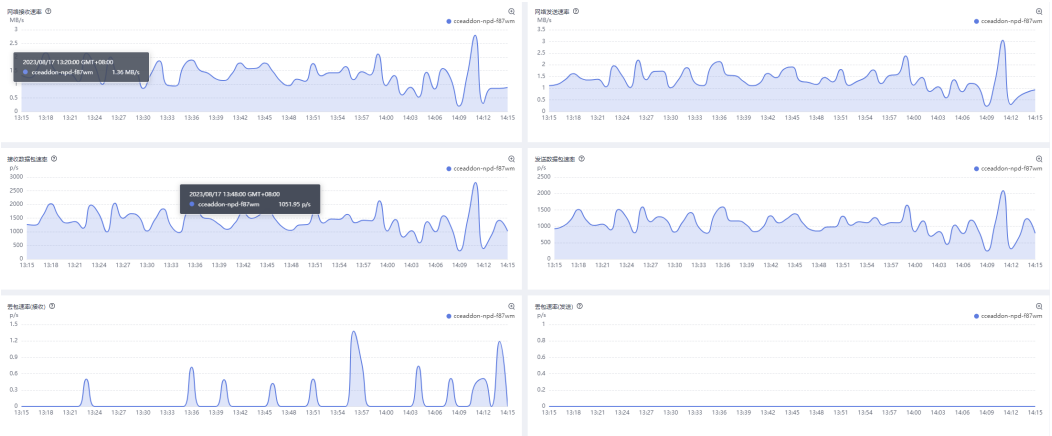


表 5-6 Pod 网络指标说明

指标名称	单位	说明
网络接收速	字节/秒	容器每秒接收的字节数
网络发送速率	字节/秒	容器每秒发送的字节数
接收数据包速率	个/秒	容器每秒接收数据包数
发送数据包速率	个/秒	容器每秒发送数据包数
丢包速率(接收)	个/秒	容器每秒接收的数据丢包数
丢包速率(发送)	个/秒	容器每秒发送的数据丢包数

图 5-20 Pod 磁盘指标

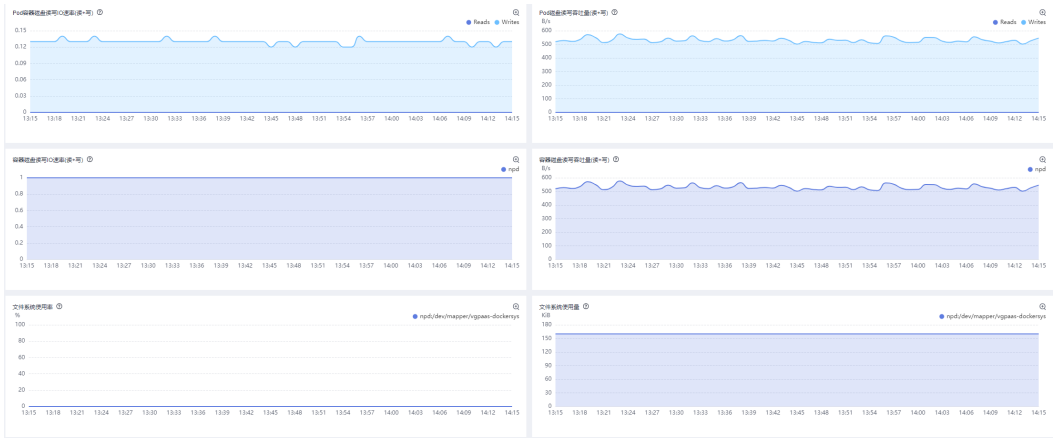


表 5-7 Pod 磁盘指标说明

指标名称	单位	说明
Pod容器磁盘读写IO速率(读+写)	次数/秒	Pod磁盘每秒读写IO次数
Pod磁盘读写吞吐量(读+写)	字节/秒	Pod磁盘每秒读写字节数
容器磁盘读写IO速率(读+写)	次数/秒	容器磁盘每秒读写IO次数
容器磁盘读写吞吐量(读+写)	字节/秒	容器磁盘每秒读写字节数
文件系统使用率	百分比	文件系统的使用率
文件系统使用量	字节	文件系统已经使用的字节数

指标清单

Pod视图使用的指标清单如下：

表 5-8 Pod 视图指标清单

指标	指标类型	说明
kube_pod_container_status_running	gauge	容器当前是否在运行中的状态
kube_pod_container_info	gauge	Pod中的容器信息
kube_pod_status_phase	gauge	Pod当前的阶段
kube_pod_container_status_restarts_total	counter	容器重启次数
container_cpu_usage_seconds_total	counter	容器CPU累计使用时间
kube_pod_container_resource_requests	gauge	容器请求的请求资源数
container_spec_cpu_quota	gauge	容器的CPU配额
container_memory_working_set_bytes	gauge	容器内存使用量
container_spec_memory_limit_bytes	gauge	容器内存限制量
container_cpu_cfs_throttled_periods_total	counter	容器限制周期间隔数
container_cpu_cfs_periods_total	counter	容器经过强制限制的周期间隔数
container_network_receive_bytes_total	counter	容器接收字节的累计计数
container_network_transmit_bytes_total	counter	容器传输字节的累计计数
container_network_receive_packets_total	counter	容器接收数据包的累计计数
container_network_transmit_packets_total	counter	容器传输数据包的累计计数
container_network_receive_packets_dropped_total	counter	容器接收丢失的数据包的累计计数
container_network_transmit_packets_dropped_total	counter	容器传输丢失的数据包的累计计数
container_fs_reads_total	counter	容器已完成磁盘读取的累计计数

指标	指标类型	说明
container_fs_writes_total	counter	容器已完成磁盘写入的累计计数
container_fs_reads_bytes_total	counter	容器读取的累计字节数
container_fs_writes_bytes_total	counter	容器写入的累计在节数
container_fs_usage_bytes	gauge	文件系统上容器已经使用的字节数
container_fs_limit_bytes	gauge	文件系统上容器限制的字节数

5.1.5.4 Kubelet 视图

Kubelet是运行在集群中每个节点上的代理程序，它提供了一些指标可以更好地了解集群的运行状态。

指标说明

Kubelet视图暴露的指标如下：

表 5-9 Kubelet 图表说明

视图名称	单位	说明
运行中Kubelet	个	集群运行中的kubelet的数量
运行中Pod	个	当前Kubelet所在节点上运行中Pod的数量
运行中容器	个	当前Kubelet所在节点上运行中容器的数量
实际卷数量	个	当前Kubelet所在节点的实际卷数量
期望卷数量	个	当前Kubelet所在节点的期望卷数量
配置错误数量	个	当前Kubelet所在节点的Kubelet配置错误数量
操作速率	次/秒	Kubelet每秒执行的操作的次数
操作错误率	次/秒	Kubelet每秒执行的操作失败的次数
操作时延	秒	Kubelet的不同操作的操作时延
Pod启动速率	次/秒	Kubelet每秒执行了pod start的次数
Pod启动时延（99分位）	秒	Kubelet执行pod start操作中99%的操作的时延分布情况
存储操作速率	次/秒	Kubelet每秒执行的存储相关操作的次数

视图名称	单位	说明
存储操作错误率	次/秒	Kubelet每秒执行的存储相关操作失败的次数
存储操作时延（99分位）	秒	Kubelet执行存储操作中99%的操作的时延分布情况
控制组管理器操作速率	次/秒	每秒执行销毁或更新操作的次数
控制组管理器操作时延（99分位）	秒	Kubelet执行销毁或更新操作中99%的操作的时延分布情况
PLEG relist速率	次/秒	Kubelet PLEG每秒执行relist的次数
PLEG relist间隔（99分位）	秒	Kubelet PLEG relist中99%的操作的间隔分布情况
PLEG relist时延（99分位）	秒	Kubelet PLEG relist中99%的操作的时延分布情况
RPC速率	次/秒	不同状态响应码的RPC请求的次数
请求时延（99分位）	秒	不同method的请求的99%的时延分布情况
内存使用量	字节	Kubelet的内存使用量
CPU使用量	字节	Kubelet的CPU使用量
Go routine数	个	Go协程数量

指标清单

Kubelet视图使用的指标清单如下：

表 5-10 Kubelet 指标说明

指标	类型	说明
storage_operation_errors_total	Counter	存储操作期间发生的错误次数
storage_operation_duration_seconds_count	Counter	存储操作的操作次数
storage_operation_duration_seconds_bucket	Histogram	存储操作的持续时间
kubelet_pod_start_duration_seconds_count	Counter	进行过pod start的数量
kubelet_pod_start_duration_seconds_bucket	Histogram	pod start的耗时分布情况
kubelet_runtime_operations_duration_seconds_bucket	Histogram	不同操作的累计操作耗时分布情况

指标	类型	说明
kubelet_runtime_operations_errors_total	Counter	不同操作的累计操作失败的数量
kubelet_node_config_error	Gauge	如果节点遇到与配置相关的错误，则此指标为true（1），否则为false（0）
volume_manager_total_volumes	Gauge	Volume Manager中的卷数
kubelet_running_containers	Gauge	当前运行的Containers数
kubelet_running_pods	Gauge	当前运行的pod数
kubelet_node_name	Gauge	节点名称，值始终为1
kubelet_runtime_operations_total	Counter	运行过程中不同的操作类型的累计操作次数
kubelet_cgroup_manager_duration_seconds_count	Counter	销毁和更新的数量
kubelet_cgroup_manager_duration_seconds_bucket	Histogram	销毁和更新操作的耗时分布情况
kubelet_pleg_relist_duration_seconds_count	Counter	PLEG relist pod不同耗时的数量
kubelet_pleg_relist_interval_seconds_bucket	Histogram	PLEG relist 间隔的分布情况
kubelet_pleg_relist_duration_seconds_bucket	Histogram	PLEG relist pod耗时的分布情况
rest_client_requests_total	Counter	请求apiserver的总次数（按照返回码code和请求类型method统计）
rest_client_request_duration_seconds_bucket	Histogram	请求apiserver的总次数（按照返回码code和请求类型method统计）的分布情况
process_resident_memory_bytes	Gauge	进程驻留内存大小（以字节为单位）
process_cpu_seconds_total	Counter	进程用户和系统 CPU 总时间（以秒为单位）
go_goroutines	Gauge	协程数量

5.1.5.5 Prometheus Agent 视图

Prometheus Agent是轻量化的容器监控模式，可以收集有关主机和应用程序的指标数据，并将数据上报并存储到AOM或三方监控平台。Prometheus Agent视图展示了Prometheus提供的一些内置指标，可用于监控和度量系统的性能和状态。

指标说明

Prometheus Agent视图暴露的指标如下：

图 5-21 Prometheus Agent 资源指标

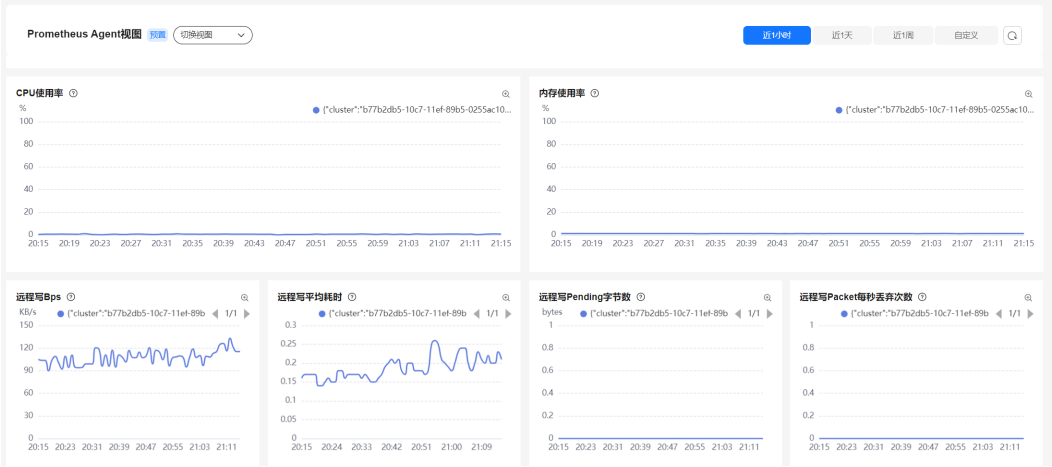


表 5-11 Prometheus Agent 视图说明

视图指标	单位	描述
CPU使用率	百分比	Prometheus Agent Pod CPU平均使用率
内存使用率	百分比	Prometheus Agent Pod 内存平均使用率
远程写Bps	字节/秒	每秒远程写入的字节数
远程写平均耗时	秒	远程写入平均耗时
远程写Pending字节数	字节	远程写入挂起的数据字节数
远程写Packet每秒丢弃次数	次	远程写入每秒丢弃的数据包数
远程写每秒错误请求次数	次	远程写每秒错误请求次数
远程写错误请求百分比	百分比	远程写错误请求百分比
远程写每秒重试次数	次	远程写每秒重试次数
采集Scrapers数量	个	采集Scrapers数量
每秒采集次数	次	每秒采集次数
采集平均耗时	秒	采集平均耗时
每秒采集读取错误次数	次	每秒采集读取错误次数
每秒采集写入错误次数	次	每秒采集写入错误次数
每秒采集大小超限次数	次	每秒采集大小超限次数

视图指标	单位	描述
发送队列读取Bps	字节/秒	发送队列每秒读的字节数
发送队列写Bps	字节/秒	发送队列每秒写的字节数
发送队列Pending大小	字节	发送队列挂起的数据字节数
每秒Block读取次数	次	发送队列每秒读block的次数
每秒Block写入次数	次	发送队列每秒写block的次数
每秒Block丢弃次数	次	发送队列每秒block丢弃的次数

指标清单

Prometheus Agent视图使用的指标清单如下：

表 5-12 Prometheus Agent 指标说明

指标名称	类型	说明
container_cpu_usage_seconds_total	Gauge	容器CPU使用量
container_memory_working_set_bytes	Gauge	容器工作内存使用量
vmagent_remotewrite_bytes_sent_total	Counter	Prometheus Agent远程写入字节数发送总计
vmagent_remotewrite_duration_seconds_sum	Summary	Prometheus Agent远程写入耗时
vmagent_remotewrite_pending_data_bytes	Gauge	Prometheus Agent远程写入挂起数据字节数
vmagent_remotewrite_packets_dropped_total	Counter	Prometheus Agent远程写入数据包丢弃总数
vmagent_remotewrite_requests_total	Counter	Prometheus Agent远程写入请求总数
vmagent_remotewrite_retries_count_total	Counter	Prometheus Agent远程写入重试次数总数
vm_promscrape_active_scrapers	Gauge	采集的分片数量
vm_promscrape_scrapes_total	Counter	采集次数
vm_promscrape_scrape_duration_seconds_sum	Summary	vmagent采集指标的耗时

指标名称	类型	说明
vm_promscrape_conn_read_errors_total	Counter	采集读取错误次数
vm_promscrape_conn_write_errors_total	Counter	采集写入错误次数
vm_promscrape_max_scrape_size_exceeded_errors_total	Counter	采集大小超过限制的次数
vm_persistentqueue_bytes_read_total	Counter	发送队列读取的总字节数
vm_persistentqueue_bytes_written_total	Counter	发送队列写入的总字节数
vm_persistentqueue_bytes_pending	Gauge	发送队列Pending的字节数
vm_persistentqueue_blocks_read_total	Counter	发送队列读block的次数
vm_persistentqueue_blocks_written_total	Counter	发送队列写block的次数
vm_persistentqueue_blocks_dropped_total	Counter	发送队列block丢弃的次数

5.2 日志中心

5.2.1 收集容器日志

云原生日志采集插件是基于开源fluent-bit和opentelemetry构建的云原生日志、Kubernetes事件采集插件。云原生日志采集插件支持基于CRD的日志采集策略，可以根据您配置的策略规则，对集群中的容器标准输出日志、容器文件日志及Kubernetes事件日志进行采集与转发。同时支持上报Kubernetes事件到AOM，用于配置事件告警，默认上报所有异常事件和部分正常事件。

约束与限制

- 每个集群限制50条日志规则。
- 不采集.gz、.tar、.zip后缀类型的日志文件，且不支持采集日志文件的软链接。
- 每个集群限制单行日志采集速率不超过10000条/秒，多行日志不超过2000条/秒。
- 如果业务容器的数据目录是通过数据卷（Volume）挂载的，插件不支持采集它的父目录，需设置采集目录为完整的数据目录。

费用说明

LTS创建日志组免费，并每月赠送每个账号一定量免费日志采集额度，超过免费额度部分将产生费用（**价格计算器**）。

使用云原生日志采集插件采集日志

步骤1 启用云原生日志采集插件日志采集功能

在集群创建时启用云原生日志采集插件日志采集

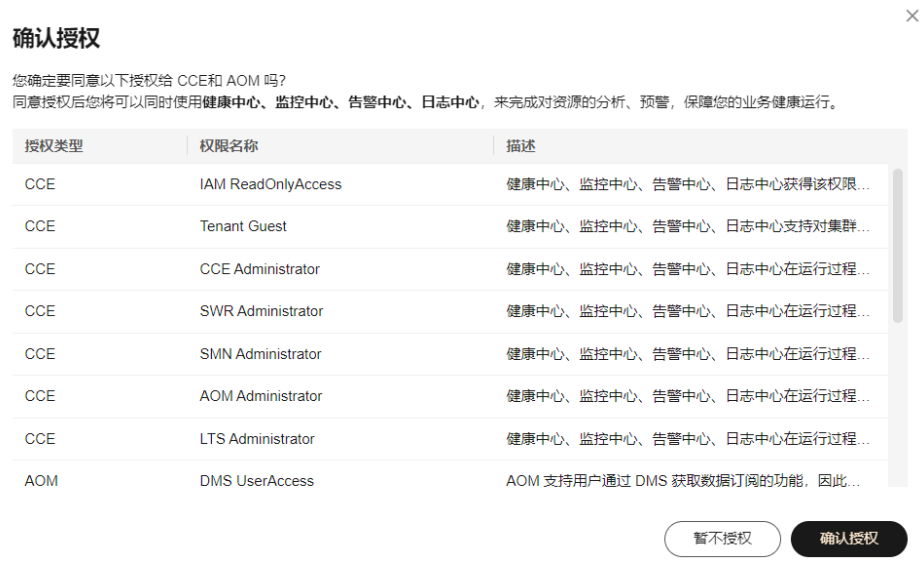
1. 登录云容器引擎（CCE）控制台。
2. 在控制台上方导航栏，单击“购买集群”。
3. 在“插件选择”页面中，选择“云原生日志采集插件”。
4. 单击右下角“下一步：插件配置”，根据需求勾选以下配置。
   - **容器日志：**开启后，将创建名为default-stdout的日志策略，并上报所有命名空间下的标准输出到云日志服务（LTS）。
   - **Kubernetes事件：**开启后，将创建名为default-event的日志策略，并上报所有命名空间下的Kubernetes事件到云日志服务（LTS）。
5. 配置完成后，单击右下角“下一步：确认配置”，在弹出的窗口中单击“确定”，完成创建。

为已有集群启用CCE 云原生日志采集插件日志采集

1. 登录云容器引擎（CCE）控制台，单击集群名称进入集群，选择左侧导航栏的“日志中心”。
2. 未进行授权的用户需要先授权，已授权的用户直接跳转下一步。

   在弹出框中单击“确认授权”。

   图 5-22 添加授权



3. 页面单击“一键开启”，等待约30秒后，页面自动跳转。
   - **采集容器标准输出：**开启后，将创建名为default-stdout的日志策略，并上报所有命名空间下的标准输出到云日志服务（LTS）。
   - **采集Kubernetes事件：**开启后，将创建名为default-event的日志策略，并上报所有命名空间下的Kubernetes事件到云日志服务（LTS）。

图 5-23 开启

步骤2 查看并配置日志采集策略。

1. 登录云容器引擎（CCE）控制台，单击集群名称进入集群，选择左侧导航栏的“日志中心”。
2. 右上角单击“日志采集策略”，将显示当前集群所有上报LTS的日志策略。

   若启用日志采集功能时勾选了采集容器标准输出和采集Kubernetes事件，将创建两个日志策略，并对接默认的LTS日志组、日志流。

   - **default-stdout**：采集标准输出。默认日志组：k8s-logs-{集群ID}；默认日志流：stdout-{集群ID}。
   - **default-event**：采集Kubernetes事件。默认日志组：k8s-logs-{集群ID}；默认日志流：event-{集群ID}。

图 5-24 查看日志策略

3. 创建日志策略：单击上方“创建日志策略”，输入要采集的配置信息。
   - **策略模板**：若启用日志采集功能时未勾选采集容器标准输出和采集Kubernetes事件，或者删除了对应的日志策略，可通过该方式重新创建默认日志策略。
   - **自定义策略**：用于配置自定义日志策略。

图 5-25 自定义策略

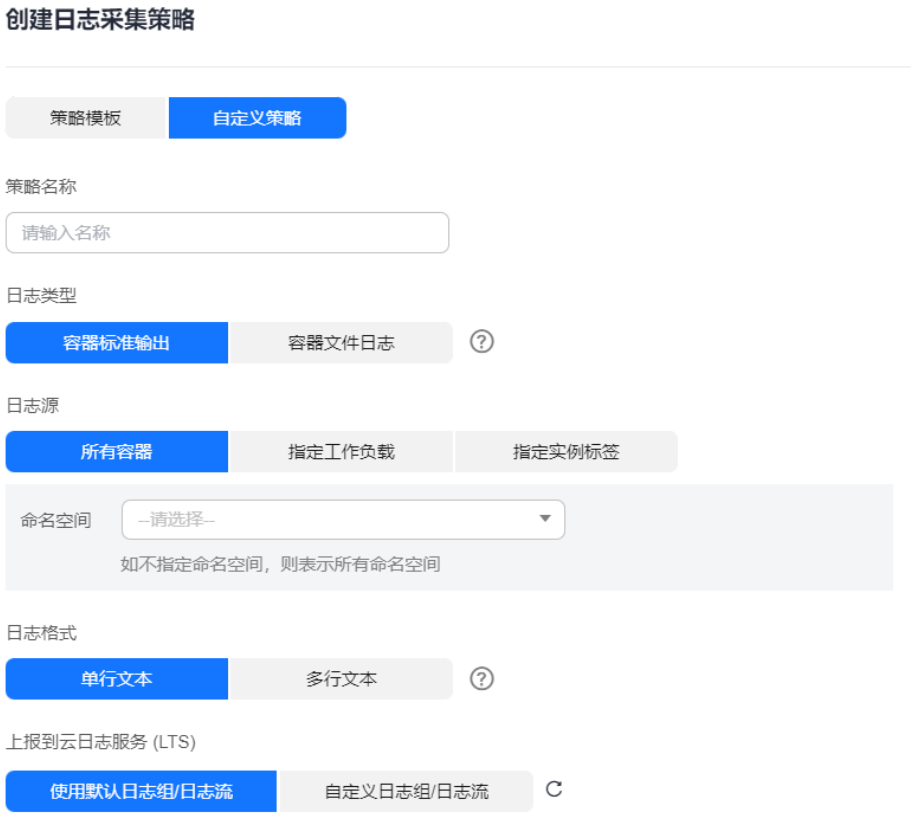


表 5-13 自定义策略参数说明

参数	说明
日志类型	指定采集哪类日志。 - **容器标准输出**：用于采集容器标准输出，可以按命名空间、工作负载名称、实例标签配置采集策略。 - **容器文件日志**：用于采集容器内的日志，可以按工作负载和实例标签配置采集策略。
日志源	采集哪些容器的日志。 - **所有容器**：可以指定采集某个命名空间的所有容器，如不指定则采集所有命名空间的容器。 - **指定工作负载**：指定采集哪些工作负载容器的日志，可以指定采集工作负载中具体容器的日志，如不指定则采集所有容器的日志。 - **指定实例标签**：根据标签指定采集哪些工作负载容器的日志，可以指定采集工作负载中具体容器的日志，如不指定则采集所有容器的日志。

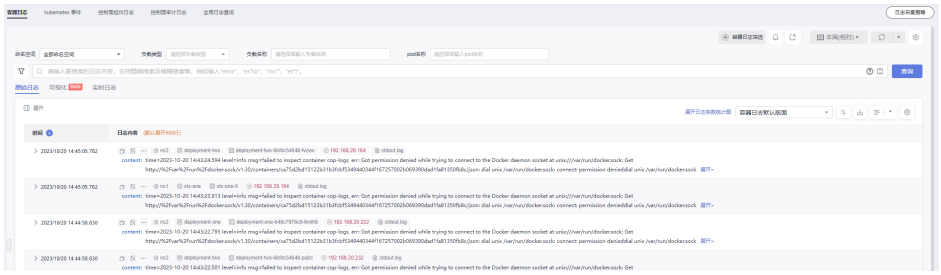
参数	说明
日志格式	– **单行文本** 每条日志仅包含一行文本，以换行符 \n 作为各条日志的分界线。 – **多行文本** 有些程序打印的日志存在一条完整的日志数据跨占多行（例如 Java 程序日志）情况，日志采集系统默认是按行采集。如果您想在日志采集系统中按整条显示日志，可以开启多行文本，采用首行正则的方式进行匹配，当选择多行文本时，需填写日志匹配格式。 例如： 需采集的日志格式如下，则需填写时间的正则匹配，在日志匹配格式处填写：\d{4}-\d{2}-\d{2} \d{2}\:\d{2}\:\d{2}.* 则下面以日期开头三行日志会作为一条完整日志。 2022-01-01 00:00:00 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)
上报到云日志服务(LTS)	用于配置日志上报的日志组和日志流。 – **使用默认日志组/日志流：**将为您自动选择默认日志组（k8s-log-{集群ID}）和默认的日志流（stdout-{集群ID}）。 – **自定义日志组/日志流：**可在下拉框选择任意日志组和日志流。 ▪ 日志组是云日志服务进行日志管理的基本单位。如果您未创建日志组，CCE会提示您进行创建，默认名称为k8s-log-{集群ID}，如 k8s-log-bb7eaa87-07dd-11ed-ab6c-0255ac1001b3。 ▪ 日志流（LogStream）：日志流是日志读写的基本单位，日志组中可以创建日志流，将不同类型的日志分类存储，方便对日志进一步分类管理。在安装插件或者根据模板创建日志策略时，会自动创建以下日志流： 容器日志：默认名称为stdout-{集群ID}，如 stdout-bb7eaa87-07dd-11ed-ab6c-0255ac1001b3。 k8s事件：默认名称为event-{集群ID}，如 event-bb7eaa87-07dd-11ed-ab6c-0255ac1001b3。

4. 编辑日志策略：单击“编辑”按钮，可对已经存在的日志策略进行修改。
5. 删除日志策略：单击“删除”按钮，可对已经存在的日志策略进行删除。

步骤3 查看日志。

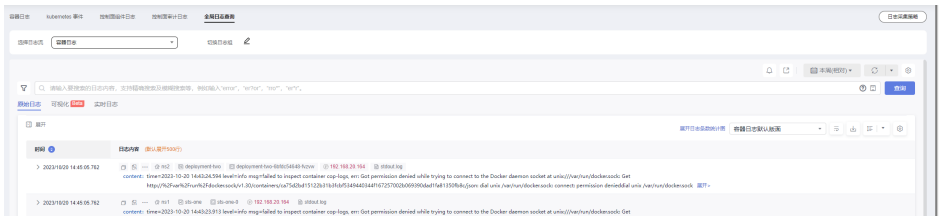
1. 登录云容器引擎（CCE）控制台，单击集群名称进入集群，选择左侧导航栏的“日志中心”。
2. 日志中心下有3个页签，支持不同类型日志查看。
   – 容器日志：显示默认日志组（k8s-log-{集群ID}）下默认日志流（stdout-{集群ID}）中的所有日志数据，支持通过工作负载搜索。

图 5-26 容器日志查询



- Kubernetes事件：显示默认日志组（k8s-log-{集群ID}）下默认日志流（event-{集群ID}）中的所有日志数据，用于查询集群产生的Kubernetes事件。
- 全局日志查询：支持查看所有日志组日志流下的日志信息。可通过选择日志流查看所选日志流中的日志信息，默认会选择集群默认日志组（k8s-log-{集群ID}），可通过单击切换日志组右侧的图标切换其他日志组。

图 5-27 全局日志查询



3. 单击右上角“日志采集策略”，单击“查看日志”，可以直接跳转至对应日志策略的日志列表。

图 5-28 查看日志

----结束

常见问题处理

- **插件中除log-operator外组件均未就绪，且出现异常事件“实例挂卷失败”。**

  解决方案：请查看log-operator日志，安装插件时，其余组件所需的配置文件需要log-operator生成，log-operator生成配置出错，会导致所有组件无法正常启动。

  日志信息如下：

  ```
  MountVolume.SetUp failed for volume "otel-collector-config-vol":configmap "log-agent-otel-collector-config" not found
  ```

- **log-operator标准输出报错：Failed to create log group, the number of log groups exceeds the quota**

示例：

```
2023/05/05 12:17:20.799 [E] call 3 times failed, resion: create group failed, projectID: xxx, groupName: k8s-log-xxx, err: create groups status code: 400, response: {"error_code":"LTS.0104","error_msg":"Failed to create log group, the number of log groups exceeds the quota"}, url: https://lts.cn-north-4.myhuaweicloud.com/v2/xxx/groups, process will retry after 45s
```

解决方案：LTS日志组有配额限制，如果出现该报错，请前往LTS下删除部分无用的日志组。限制详情见：**日志组**。

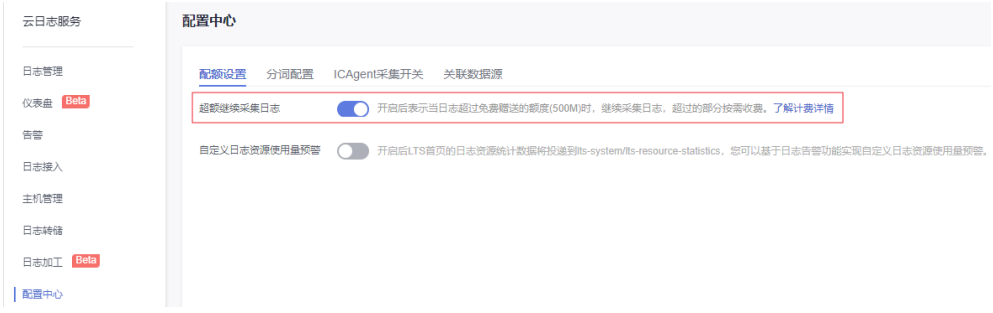
- **日志无法上报，otel组件标准输出报错：log's quota has full**

```
2023-08-16T09:03:20.067+0800    error    exporterhelper/queued_retry.go:361    Exporting failed. Try enabling retry_on_failure config option to retry on retryable errors    {"kind": "exporter", "data_type": "logs", "name": "lts/default-event", "error": "fail to push event data via lts exporter: read body {\"errorCode\":\"SVCSTG.ALS.200.210\",\"errorMessage\":\"projectid                                s quota has full!!\",\"result\":null} error"}
go.opentelemetry.io/collector/exporter/exporterhelper.(*retrySender).send
        go.opentelemetry.io/collector@v0.58.0/exporter/exporterhelper/queued_retry.go:361
go.opentelemetry.io/collector/exporter/exporterhelper.(*logsExporterWithObservability).send
        go.opentelemetry.io/collector@v0.58.0/exporter/exporterhelper/logs.go:142
go.opentelemetry.io/collector/exporter/exporterhelper.(*queuedRetrySender).send
        go.opentelemetry.io/collector@v0.58.0/exporter/exporterhelper/queued_retry.go:295
go.opentelemetry.io/collector/exporter/exporterhelper.NewLogsExporterWithContext.func2
        go.opentelemetry.io/collector@v0.58.0/exporter/exporterhelper/logs.go:122
go.opentelemetry.io/collector/consumer.ConsumeLogsFunc.ConsumeLogs
        go.opentelemetry.io/collector@v0.58.0/consumer/logs.go:36
go.opentelemetry.io/collector/service/internal/fanoutconsumer.(*logsConsumer).ConsumeLogs
        go.opentelemetry.io/collector@v0.58.0/service/internal/fanoutconsumer/logs.go:77
cieotelcol/receiver/k8seventsreceiver.(*k8seventsReceiver).handleEvent
        cieotelcol/receiver/k8seventsreceiver/receiver.go:138
cieotelcol/receiver/k8seventsreceiver.(*k8seventsReceiver).startWatch.func1
        cieotelcol/receiver/k8seventsreceiver/receiver.go:116
k8s.io/client-go/tools/cache.ResourceEventHandlerFuncs.OnAdd
        k8s.io/client-go@v0.24.3/tools/cache/controller.go:232
k8s.io/client-go/tools/cache.processDeltas
        k8s.io/client-go@v0.24.3/tools/cache/controller.go:441
k8s.io/client-go/tools/cache.newInformer.func1
```

解决方案：

云日志服务（LTS）有免费赠送的额度，超出后将收费，报错说明免费额度已用完，如果需要继续使用，请前往云日志服务控制台“配置中心”，打开“超额继续采集日志”开关。

图 5-29 配额设置



- **采集容器内日志，且采集目录配置了通配符，日志无法采集。**

  排查方法：请检查工作负载配置中Volume挂载情况，如果业务容器的数据目录是通过数据卷（Volume）挂载的，插件不支持采集它的父目录，需设置采集目录为完整的数据目录。例如/var/log/service目录是数据卷挂载的路径，则设置采集目录为/var/log或/var/log/*将采集不到该目录下的日志，需设置采集目录为/var/log/service。

解决方案：若日志生成目录为/application/logs/{应用名}/*.log，建议工作负载挂载Volume时，直接挂载/application/logs，日志策略中配置采集路径为/application/logs/*/*.log

5.2.2 收集 Kubernetes 事件

云原生日志采集插件可采集Kubernetes事件上报到云日志服务（LTS）和应用运维管理（AOM），用于保存事件信息和事件告警。

费用说明

LTS创建日志组免费，并每月赠送每个账号一定量免费日志采集额度，超过免费额度部分将产生费用（**价格计算器**）。

Kubernetes 事件上报云日志服务（LTS）

集群未安装云原生日志采集插件

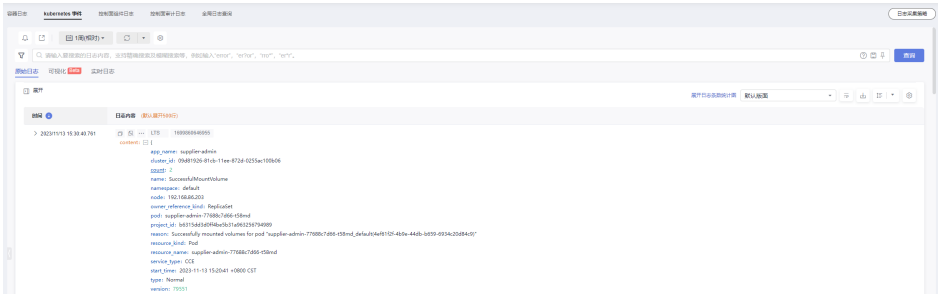
安装CCE 云原生日志采集插件时，可通过勾选采集Kubernetes事件，创建默认日志采集策略，采集所有事件上报到LTS。安装方法见：**收集容器日志**。

集群已安装云原生日志采集插件

1. 登录云容器引擎（CCE）控制台，单击集群名称进入集群，选择左侧导航栏的“日志中心”。
2. 右上角单击“日志采集策略”，将显示当前集群所有上报LTS的日志策略。
3. 单击上方“创建日志策略”，输入要采集的配置信息。

   策略模板：若安装插件时未勾选采集Kubernetes事件，或者删除了对应的日志策略，可通过该方式重新创建默认事件采集策略。

4. 事件查看：可直接在“日志中心”页面查看，选择日志策略配置的日志流名称，即可查看上报到云日志服务（LTS）的事件。



Kubernetes 事件上报应用运维管理（AOM）

安装云原生日志采集插件后，默认会将上报所有Warning级别事件以及部分Normal级别事件到应用运维管理（AOM），上报的事件可用于配置告警。

自定义事件上报

若已上报的事件不能满足需求，可通过修改配置，修改需要上报到应用运维管理（AOM）的事件。

步骤1 在集群上执行以下命令，编辑当前的事件采集配置。

kubectl edit logconfig -n kube-system default-event-aom

步骤2 根据需要修改事件采集配置。

```
apiVersion: logging.openvessel.io/v1
kind: LogConfig
metadata:
  annotations:
    helm.sh/resource-policy: keep
  name: default-event-aom
  namespace: kube-system
spec:
  inputDetail:    #采集端配置
    type: event    #采集端类型，请勿修改
    event:
      normalEvents:    #Normal级别事件采集配置
        enable: true    #是否开启Normal级别事件采集
        includeNames:    #需要采集的事件名，不指定则采集所有事件
        - NotTriggerScaleUp
        excludeNames:    #不采集的事件名，不指定则采集所有事件
        - ScaleDown
      warningEvents:    #Warning级别事件采集配置
        enable: true    #是否开启Warning级别事件采集
        includeNames:    #需要采集的事件名，不指定则采集所有事件
        - NotTriggerScaleUp
        excludeNames:    #不采集的事件名，不指定则采集所有事件
        - ScaleDown
  outputDetail:
    type: AOM    #输出端类型，请勿修改
    AOM:
      events:
      - name: DeleteNodeWithNoServer    #事件名，必选
        nameCn: 废弃节点清理    #事件对应的中文名，不填则上报的事件直接显示英文
        resourceType: Namespace    #事件对应的资源类型
        severity: Major    #事件上报到AOM后的事件级别，默认Major。可选值：Critical：紧急；Major：重要；Minor：次要；Info：提示
```

----结束

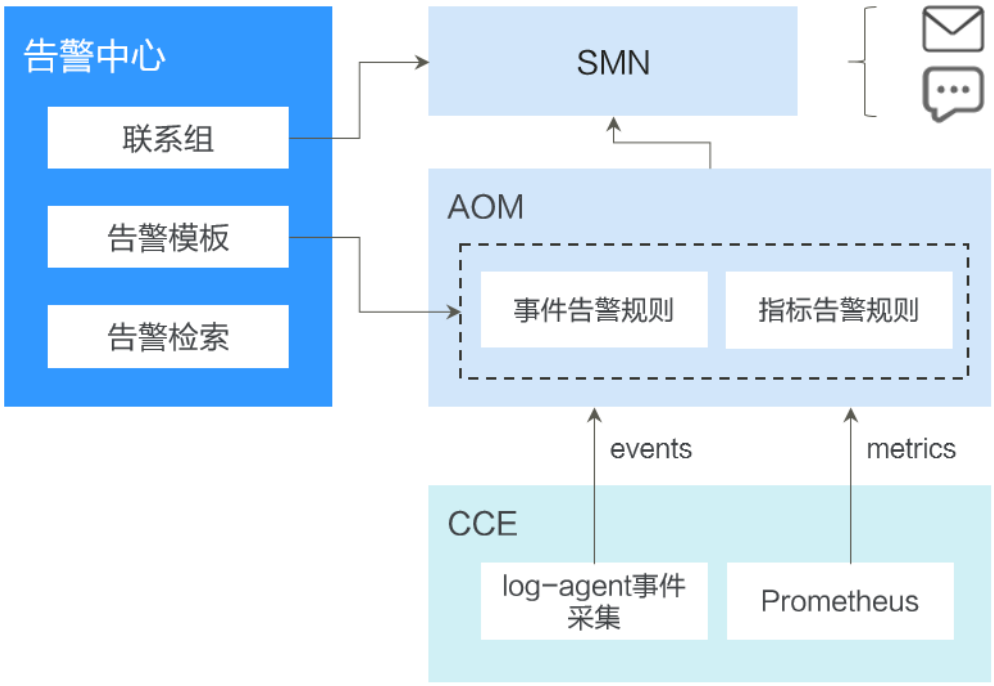
5.3 告警中心

5.3.1 告警中心概述

云原生告警是可观测性体系里面比较重要的一环。在云原生告警中，除了传统的CPU、内存等资源使用量的告警以外，还有容器重启等事件告警、应用访问失败等自定义的监控指标告警。

CCE的云原生告警能力是由AOM服务提供的，支持指标和事件的告警。同时，CCE集群详情中增加了告警中心能力，能支持快速配置资源等常用告警和告警查看。

图 5-30 告警中心架构



- 告警中心
  基于AOM服务的告警能力实现，提供集群内的告警快速检索、告警快速配置的能力。用户可以通过告警中心一键配置常用的告警规则。
- AOM服务
  华为云应用运维管理服务，是云上应用的一站式立体化运维管理平台，是云上监控、告警的基础。
- SMN服务
  华为云的消息通知服务，是云上应用发送告警或通知的依赖服务。在云原生场景中，在AOM服务触发的告警将通过SMN里面配置的短信、电子邮件、HTTP等方式发送。

5.3.2 通过告警中心一键配置告警

告警中心基于AOM告警功能，提供集群内置告警一键开启能力，在集群发生故障时能够及时发现并预警，协助您维护业务稳定性。智能告警中心可有效节省您在AOM侧手

动配置告警规则的工作量，并且内置的告警规则基于华为云容器团队大规模集群运维经验，能够满足您的日常运维所需，覆盖容器服务异常事件告警、集群相关基础资源的关键指标告警及集群中应用的指标告警。

约束与限制

仅华为云/华为账号，或者拥有CCE Administrator权限或CCE FullAccess权限的IAM用户可进行告警中心所有操作。CCE ReadOnlyAccess权限的IAM用户可以查看所有资源信息，但是无法进行任何操作。

开启告警中心

步骤1 在目标集群左侧导航栏选择“告警中心”。

步骤2 选择“告警规则”页签，单击“开启告警中心”，在弹出的页面中选择一个或多个联系组，以便分组管理订阅终端并接收告警消息。如果当前还没有联系组，请参考**配置告警通知人**进行创建。

步骤3 单击“确认”完成功能开启。

说明

告警中心中的指标类告警规则依赖云原生监控插件上报指标数据到AOM Prometheus实例，需要开通监控中心。当您的集群未安装插件或者在安装插件时未对接AOM Prometheus实例，告警中心将不会创建指标类告警规则。开通监控中心请参考**开通监控中心**。

表5-14中的事件类告警依赖日志中心开启收集Kubernetes事件的能力，详情请参见**收集Kubernetes事件**。

----结束

配置告警规则

开启智能告警中心后，可以进行告警规则的配置和管理。

步骤1 登录CCE控制台。

步骤2 在集群列表页面，单击目标集群名称进入详情页。

步骤3 在左侧导航栏选择“告警中心”，选择“告警规则”页签，在此处进行告警规则的配置和管理。

智能告警中心功能会默认生成容器场景下的告警规则模板（包含异常事件告警、异常指标告警）。告警规则被分类为若干个告警规则集，您可以为告警规则集关联多个联系组，并开启或关闭告警项。告警规则集中包含多个告警规则，一个告警规则对应单个异常的检查项。关于默认告警规则模板，请参见**表5-14**。

----结束

表 5-14 默认告警规则

告警规则类型	告警项	告警说明	告警类型	依赖项	PromQL/事件名称
负载规则集	Pod状态异常	检查Pod状态是否异常	指标类	云原生监控插件	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
	Pod频繁重启	检查Pod是否频繁重启	指标类	云原生监控插件	increase(kube_pod_container_status_restarts_total[5m]) > 3
	Deployment副本数不匹配	检查无状态负载的副本数是否匹配	指标类	云原生监控插件	(kube_deployment_spec_replicas != kube_deployment_status_replicas_available) and (changes(kube_deployment_status_replicas_updated[5m]) == 0)
	Statefulset副本数不匹配	检查有状态负载的副本数是否匹配	指标类	云原生监控插件	(kube_statefulset_status_replicas_ready != kube_statefulset_status_replicas) and (changes(kube_statefulset_status_replicas_updated[5m]) == 0)

告警规则类型	告警项	告警说明	告警类型	依赖项	PromQL/事件名称
	容器CPU使用率大于百分之八十	检查容器CPU使用率是否大于80%	指标类	云原生监控插件	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
	容器内存使用率大于百分之八十	检查容器内存使用率是否大于80%	指标类	云原生监控插件	(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
	容器状态异常	检查容器状态是否异常	指标类	云原生监控插件	sum by (namespace, pod, container, cluster_name, cluster) (kube_pod_container_status_waiting_reason) > 0
	更新负载均衡失败	检查更新负载均衡是否成功	事件类	云原生日志采集插件	不涉及
	Pod内存不足OOM	检查Pod是否OOM	事件类	节点故障检测插件 云原生日志采集插件	PodOOMKilling

告警规则类型	告警项	告警说明	告警类型	依赖项	PromQL/事件名称
集群状态规则集	集群状态不可用	检查集群状态是否可用	事件类	云原生日志采集插件	不涉及

配置告警通知人

联系组是基于**消息通知服务 SMN**的主题功能实现的，目的是为消息发布者和订阅者提供一个可以相互交流的通道。联系组包含一个或多个订阅终端，您可以通过配置告警联系组，分组管理订阅终端，接收告警信息。联系组创建完成后，需要绑定至告警规则集，这样，当有告警触发时，联系组中的订阅终端就可以收到告警消息了。

步骤1 登录CCE控制台。

步骤2 在集群列表页面，单击目标集群名称进入详情页。

步骤3 在左侧导航栏选择“告警中心”，选择“默认联系组”页签。

步骤4 单击“新建联系组”，在弹出的页面中输入联系组参数。

- **联系组名称**：输入联系组名称，创建后不可修改。名称只能包含大写字母、小写字母、数字、-和_，且必须由大写字母、小写字母或数字开头，名称长度为1~255字符。
- **告警消息显示名**：即订阅终端接收消息的标题名称。假设订阅终端为邮件，推送邮件消息时，若已设置告警消息显示名，发件人则呈现为“显示名”，若未设置告警消息显示名，发件人呈现为“username@example.com”。支持在联系组创建完成后修改告警消息显示名。
- **添加订阅终端**：您需要添加一个或多个订阅终端来接收告警消息。终端类型包括短信和邮件，选择“短信”时，请输入有效的手机号码；选择“邮件”时，请输入有效的电子邮件地址。

步骤5 单击“确定”完成联系组的创建。

返回联系组列表，订阅终端状态为“未确认”，您需要继续执行后续操作，向该终端发送订阅请求，以验证终端有效性。

步骤6 单击操作列“请求订阅”，向该终端发送订阅请求。若终端收到请求，请按照提示进行确认，确认完成后订阅终端状态将变为“已确认”。

步骤7 联系组创建并确认后，单击  图标启用联系组，实现联系组和告警规则集的绑定。

说明

告警规则集最多支持绑定5个联系组。

----结束

查看告警列表

您可以在“告警列表”页面查看最近发送的历史记录。

步骤1 登录CCE控制台。

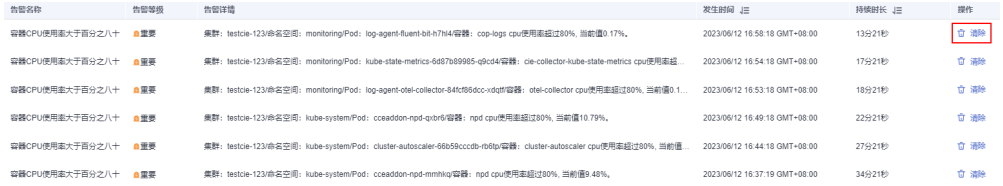
步骤2 在集群列表页面，单击目标集群名称进入详情页。

步骤3 在左侧导航栏选择“告警中心”，选择“告警列表”页签。

列表中默认展示全部待解决告警，支持按照告警关键字、告警等级，以及告警发生的时间范围筛选。同时支持查看指定筛选条件的告警在不同时间段的分布情况。

如果确认某条告警已排除，可以单击操作列的“清除”，清除后可在历史告警中查询。

图 5-31 告警列表



----结束

5.3.3 通过 CCE 配置自定义告警

当默认的告警规则无法满足您的述求时，可以创建自定义告警规则。通过在CCE中创建告警规则，您可以及时了解集群中各种资源是否存在异常。

添加指标类告警示例

说明

基于Prometheus指标的阈值告警规则，指标告警规则依赖开通监控中心，请前往监控中心一键开通。详情请参见**开通监控中心**。

步骤1 登录CCE控制台，单击集群名称进入一个已有的集群。

步骤2 在左侧导航栏选择“告警中心”，切换至“告警规则 > 自定义告警规则”页签，单击“创建告警规则”。

步骤3 设置告警规则，在创建告警规则面板填写配置。

- 规则类型：选择“指标告警”，设置基于Prometheus指标的阈值告警规则。
- 告警模板：不使用模板场景下，需填写手动规则详情。您也可以使用告警模板，快速定义告警规则（PromQL）或基于已有模板进行修改。
- 规则详情：

参数	说明	场景示例
规则名称	自定义告警规则的名称	CoreDNS内存使用率超过百分之八十
描述（可选）	添加告警规则描述。	检查CoreDNS容器内存使用率是否大于80%。

参数	说明	场景示例
告警规则（PromQL）	输入普罗查询语句。关于如何编写普罗查询语句，请参见**查询示例**。	本例中设置CoreDNS当内存使用率的最大值大于80%产生告警，示例如下： (sum(container_memory_working_set_bytes{image!="", container!="POD",namespace="kube-system",container="coredns"}) BY (cluster_name, node,container, pod, namespace, cluster) / sum(container_spec_memory_limit_bytes{namespace="kube-system", container="coredns"} > 0) BY (cluster_name, node, container, pod , namespace, cluster) * 100) > 80
告警等级	根据重要性选择告警等级，分为“紧急”、“重要”、“次要”、“提示”四个等级。	紧急
持续时长	通过下拉菜单选择告警持续时长，默认为1分钟。	1分钟
告警内容	定义告警通知中的内容，可通过“${变量}”的形式捕获Prometheus中的变量。	示例如下： 集群：${cluster_name}/命名空间：${namespace}/Pod：${pod}/容器：${container} 内存使用率超过80%, 当前值${value}%。
联系组	选择一个已有的联系组。您也可以单击“新建联系组”进行创建，配置参数详情请参见**配置告警通知人**。	CCEGroup

上述示例为kube-system空间下的CoreDNS设置一条名为“CoreDNS内存使用率超过百分之八十”的告警规则，告警等级为紧急。当内存使用率的最大值大于80%，且持续了1分钟时，给联系组CCEGroup内的所有告警联系人发送通知（通知方式为短信或邮件）。 通知内容包含集群名称、命名空间、Pod名称、容器名称以及当前的内存使用率。

- 高级设置（可选）
  - 告警标签：添加告警标识性属性，用于告警降噪分组条件，标签值可用在通知内容模板中以*$event.metadata.*标签名被引用。一共可以添加10个告警标签。
  - 告警标注：添加告警非标识性属性，标注值可用在通知内容模板中以*$event.annotations.*标注名被引用。一共可以添加10个告警标注。

步骤4 单击“确定”，然后可前往自定义告警规则列表中查看规则是否创建成功。

----结束

添加事件类告警

说明

基于事件触发的告警规则依赖开通日志中心并开启Kubernetes事件采集，前往日志中心一键开通。详情请参见**收集容器日志**。

步骤1 登录CCE控制台，单击集群名称进入一个已有的集群。

步骤2 在左侧导航栏选择“告警中心”，切换至“告警规则 > 自定义告警规则”页签，单击“创建告警规则”。

步骤3 设置告警规则，在创建告警规则面板填写配置。

- 规则类型：选择“事件告警”，设置基于事件触发的告警规则，常见事件来源为Kubernetes事件和云服务事件。
- 规则详情：

参数	说明	场景示例
规则名称	自定义告警规则的名称	ReplicaSet副本数变化
描述（可选）	添加告警规则描述。	ReplicaSet副本数在5分钟内变化次数超过3次
事件名称	输入事件的名称，该名称需要与实际产生的Kubernetes事件或云服务事件相匹配。	ScalingReplicaSet
触发方式	- 立即触发：只要事件出现即发生告警。 - 累计触发：在指定的监控周期内，累计次数满足数值要求，才会发生告警。	选择“累计触发”，并设置监控周期为“5分钟”，累计次数为“> 3”。
告警等级	根据重要性选择告警等级，分为“紧急”、“重要”、“次要”、“提示”四个等级。	次要
联系组	选择一个已有的联系组。您也可以单击“新建联系组”进行创建，配置参数详情请参见**配置告警通知人**。	CCEGroup

上述示例为ScalingReplicaSet事件设置一条名为“ReplicaSet副本数变化”的告警，告警等级为次要。当5分钟内累计次数超过3次时，CCEGroup内的所有告警联系人发送通知（通知方式为短信或邮件）。

步骤4 单击“确定”，然后可前往自定义告警规则列表中查看规则是否创建成功。

----结束

5.3.4 CCE Autopilot 集群事件列表

在集群运行过程中，CCE Autopilot集群会上报一系列事件至AOM，您可以根据自身需求添加事件类告警，监控集群数据面和控制面组件的健康状态，及时发现和解决问题，保证集群的稳定性和可靠性。

- **集群数据面事件**：集群运行过程中与用户操作相关的事件，包括工作负载、网络、存储、弹性伸缩等事件。
  - **工作负载相关事件**
  - **网络相关事件**
  - **存储相关事件**
  - **弹性伸缩相关事件**
- **集群控制面事件**：集群运行过程中控制节点上报的事件，这些事件通常是由于控制面组件的故障、升级等情况引起。

集群数据面事件

表 5-15 工作负载相关事件

类别	事件描述	事件名称	事件级别	更多说明
Pod	Pod内存不足OOM	PodOOMKilling	重要	检查Pod是否因OOM退出。 该事件依赖节点故障检测插件（1.18.41及以上版本）和云原生日志采集插件（1.3.2及以上版本）。
Pod	启动失败	FailedStart	重要	检查Pod是否启动成功。
Pod	拉取镜像失败	FailedPullImage	重要	检查Pod是否拉取镜像成功。
Pod	启动重试失败	BackOffStart	重要	检查Pod是否重启失败。
Pod	调度失败	FailedScheduling	重要	检查Pod是否调度成功。
Pod	拉取镜像重试失败	BackOffPullImage	重要	检查Pod重试拉取镜像是否成功。
Pod	创建失败	FailedCreate	重要	检查Pod创建是否成功。
Pod	状态异常	Unhealthy	次要	检查Pod健康检查是否成功。
Pod	删除失败	FailedDelete	次要	检查工作负载是否删除成功。
Pod	未拉取镜像异常	ErrImageNeverPull	次要	检查工作负载是否拉取镜像。
Pod	扩容失败	FailedScaleOut	次要	检查工作负载副本扩容是否正常。

类别	事件描述	事件名称	事件级别	更多说明
Pod	更新配置失败	FailedReconfig	次要	检查Pod更新配置是否成功。
Pod	激活失败	FailedActive	次要	检查Pod是否激活成功。
Pod	回滚失败	FailedRollback	次要	检查Pod回滚是否成功。
Pod	更新失败	FailedUpdate	次要	检查Pod更新是否成功。
Pod	缩容失败	FailedScaleIn	次要	检查Pod缩容是否失败。
Pod	重启失败	FailedRestart	次要	检查Pod重启是否成功。
Deployment	标签选择器冲突	SelectorOverlap	次要	检查集群中标签选择器是否存在冲突。
Deployment	副本集创建异常	ReplicaSetCreateError	次要	检查工作负载ReplicaSet创建副本是否正常。
Deployment	部署回滚版本未发现	DeploymentRollbackRevisionNotFound	次要	检查Deployment负载回滚版本是否存在。
Job	太多活跃Pod	TooManyActivePods	次要	检查Job达到预定的Pod数后，是否还存在活动状态的Pod。
Job	太多成功Pod	TooManySucceededPods	次要	检查Job达到预定的数量后，是否存在过多运行成功的Pod。
CronJob	查询失败	FailedGet	次要	查询CronJob是否成功。
CronJob	查询Pod列表失败	FailedList	次要	检查查询Pod列表是否成功。
CronJob	未知Job	UnexpectedJob	次要	检查CronJob是否出现未知的Job。

表 5-16 网络相关事件

类别	事件描述	事件名称	事件级别	更多说明
Service	创建负载均衡失败	CreatingLoadBalancerFailed	次要	检查创建ELB是否成功。
Service	删除负载均衡失败	DeletingLoadBalancerFailed	次要	检查删除ELB是否成功。

类别	事件描述	事件名称	事件级别	更多说明
Service	更新负载均衡失败	UpdateLoadBalancerFailed	次要	检查更新ELB是否成功。

表 5-17 存储相关事件

类别	事件描述	事件名称	事件级别	更多说明
PV	主机卸载块存储失败	DetachVolumeFailed	次要	检查卸载块存储是否成功。
PV	卷回收策略未知	VolumeUnknownReclaimPolicy	次要	检查是否指定卷回收策略。
PV	挂载数据卷失败	SetUpAtVolumeFailed	次要	检查数据卷挂载是否成功。
PV	数据卷回收失败	VolumeFailedRecycle	次要	检查数据卷是否成功回收。
PV	等待主机挂载块存储失败	WaitForAttachVolumeFailed	次要	检查节点挂载块存储是否成功。
PV	数据卷删除失败	VolumeFailedDelete	次要	检查数据卷删除是否成功。
PV	挂载盘符失败	MountDeviceFailed	次要	检查数据卷挂盘是否成功。
PV	卸载数据卷失败	TearDownAtVolumeFailed	次要	检查数据卷卸载是否成功。
PV	卸载盘符失败	UnmountDeviceFailed	次要	检查数据卷卸载盘符是否成功。
PV	主机挂载块存储失败	AttachVolumeFailed	次要	检查节点卸载块存储是否成功。
PVC	数据卷扩容失败	VolumeResizeFailed	次要	检查数据卷扩容是否成功。
PVC	卷PVC丢失	ClaimLost	次要	检查PVC卷是否正常。
PVC	创建卷失败	ProvisioningFailed	次要	检查创建数据卷是否正常。
PVC	创建卷清理失败	ProvisioningCleanupFailed	次要	检查清理数据卷是否正常。

类别	事件描述	事件名称	事件级别	更多说明
PVC	卷误绑定	ClaimMisbound	次要	检查PVC是否绑定错误的卷。

表 5-18 弹性伸缩相关事件

类别	事件描述	事件名称	事件级别	更多说明
HPA	HPA非法指标范围	InvalidTargetRange	重要	• HPA的annotations中配置了非法的extendedhpa.metrics。 • HPA的spec中metric type填写错误。
HPA	HPA获取伸缩对象失败	FailedGetScale	重要	HPA无法获取待伸缩的资源对象。
HPA	HPA计算资源扩缩副本数失败	FailedComputeMetricsReplicas	重要	一般是由于在计算需要为资源调整多少个副本数时出现了问题，例如metric-server不可用、资源指标采集失败、CPU利用率等设置不正确等。 可以通过以下命令查看详细的信息： `kubectl describe horizontalpodautoscaler <hpa-name>`
HPA	HPA获取对象指标失败	FailedGetObjectMetric	重要	获取指定对象（PVC、ConfigMaps等）的指标失败。
HPA	HPA获取Pod资源指标失败	FailedGetPodsMetric	重要	获取Pod资源指标失败（单个Pod的资源利用率）。
HPA	HPA获取集群资源指标失败	FailedGetResourceMetric	重要	获取集群资源指标失败（整个集群的资源利用率）。
HPA	HPA获取容器资源指标失败	FailedGetContainerResourceMetric	重要	获取单个容器资源指标失败。
HPA	HPA获取外部指标失败	FailedGetExternalMetric	重要	获取外部指标失败。
HPA	HPA伸缩Pod失败	FailedRescale	重要	更新待伸缩资源对象的期望副本数失败。
HPA	Pod扩缩容成功	SuccessfulRescale	次要	更新待伸缩资源对象的期望副本数成功。

类别	事件描述	事件名称	事件级别	更多说明
CronHPA	CronHPA伸缩失败	ScaleFailed	重要	CronHPA更新待伸缩资源对象的期望副本数失败。
CronHPA	CronHPA查询关联HPA失败	FailedGetHorizontalPodAutoscaler	重要	CronHPA查询关联的HPA对象失败（通常是kube-apiserver侧响应失败）。
CronHPA	CronHPA查询伸缩对象失败	FailedGetHpaScale	重要	CronHPA获取待伸缩资源对象失败。
CronHPA	CronHPA更新关联HPA失败	UpdateHPAFailed	重要	CronHPA更新关联的HPA对象失败。
CronHPA	更新HPA策略成功	UpdateHPASuccess	次要	CronHPA更新关联的HPA对象成功。
CronHPA	跳过更新HPA策略	SkipUpdateHPA	次要	CronHPA跳过更新关联的HPA对象。
CronHPA	跳过更新工作负载实例数	SkipUpdateTarget	次要	CronHPA跳过更新待伸缩资源对象的副本数。
CronHPA	更新工作负载实例数成功	UpdateTargetSuccess	次要	CronHPA更新待伸缩资源对象的副本数成功。
CustomedHPA	CustomedHPA解析冷却时间失败	FailedSetPolicySettings	重要	解析CustomedHPA的冷却时间失败。
CustomedHPA	CustomedHPA处理定时/指标规则失败	FailedSubmitRule	重要	CustomedHPA处理定时规则或指标规则失败。
CustomedHPA	CustomedHPA计算资源扩缩副本数失败	FailedComputeReplicas	重要	CustomedHPA计算指标触发资源扩缩容失败。
CustomedHPA	CustomedHPA伸缩Pod失败	FailedScale	重要	CustomedHPA更新待伸缩资源对象的期望副本数失败（通常是kube-apiserver侧响应失败）。
CustomedHPA	CustomedHPA指标扩缩容成功	MetricScaleSuccess	次要	CustomedHPA根据指标规则触发资源扩缩容成功。

类别	事件描述	事件名称	事件级别	更多说明
CustomedHPA	CustomedHPA周期扩缩容成功	CronScaleSuccess	次要	CustomedHPA根据周期规则触发资源扩缩容成功。

集群控制面事件

表 5-19 集群控制面事件

事件名称	事件ID	事件级别	事件说明
内部故障	Internal error	重要	检查集群是否出现内部故障。
检查组件状态失败或组件状态异常	Failed to check component status or components are abnormal	重要	检查集群检查组件状态是否成功，或组件状态是否异常。
集群状态不可用	Cluster status is Unavailable	重要	检查集群状态是否可用。
集群状态故障	Cluster status is Error	重要	检查集群是否出现故障。
集群状态长时间不更新	Cluster status is not updated for a long time	重要	检查集群状态是否长时间不更新。
更新集群状态失败	Failed to update cluster status	重要	检查更新集群状态是否成功。
删除不可用的Kubernetes连接失败	Failed to delete the unavailable connection of the Kubernetes cluster	重要	检查删除不可用的Kubernetes连接是否成功。
同步集群证书失败	Failed to sync the cluster cert	重要	检查同步集群证书是否成功。

6 命名空间

6.1 创建命名空间

操作场景

命名空间（Namespace）是对一组资源和对象的抽象整合。在同一个集群内可创建不同的命名空间，不同命名空间中的数据彼此隔离。使得它们既可以共享同一个集群的服务，也能够互不干扰。

例如可以将开发环境、测试环境的业务分别放在不同的命名空间。

前提条件

至少已创建一个集群。

约束与限制

每个命名空间下，创建的服务数量不能超过6000个。此处的服务对应kubernetes的service资源，即工作负载所添加的服务。

命名空间类别

命名空间按创建类型分为两大类：集群默认创建的、用户创建的。

- 集群默认创建的：集群在启动时会默认创建default、kube-public、kube-system、kube-node-lease命名空间。
  - default：所有未指定Namespace的对象都会被分配在default命名空间。
  - kube-public：此命名空间下的资源可以被所有人访问（包括未认证用户），用来部署公共插件、容器模板等。
  - kube-system：所有由Kubernetes系统创建的资源都处于这个命名空间。
  - kube-node-lease：每个节点在该命名空间中都有一个关联的“Lease”对象，该对象由节点定期更新。NodeStatus和NodeLease都被视为来自节点的心跳，在v1.13之前的版本中，节点的心跳只有NodeStatus，NodeLease特性从v1.13开始引入。NodeLease比NodeStatus更轻量级，该特性在集群规模扩展性和性能上有明显提升。

- 用户创建的：用户可以按照需要创建命名空间，例如开发环境、联调环境和测试环境分别创建对应的命名空间。或者按照不同的业务创建对应的命名空间，例如系统若分为登录和游戏服务，可以分别创建对应命名空间。

创建命名空间

步骤1 登录CCE控制台，单击集群名称进入集群。

步骤2 在左侧导航栏中选择“命名空间”，在右上角单击“创建命名空间”。

步骤3 参照表6-1设置命名空间参数。

表 6-1 命名空间基本信息

参数	参数说明
名称	新建命名空间的名称，命名必须唯一。
描述	输入对命名空间的描述信息。
配额管理	资源配额可以限制命名空间下的资源使用，进而支持以命名空间为粒度的资源划分。 **须知** **建议根据需要在命名空间中设置资源配额，避免因资源过载导致集群或节点异常。** 请输入整型数值，不输入表示不限制该资源的使用。 若您需要限制CPU或内存的配额，则创建工作负载时必须指定CPU或内存请求值。

步骤4 配置完成后，单击“确定”。

----结束

使用 kubectl 创建 Namespace

使用如下方式定义Namespace。

```
apiVersion: v1
kind: Namespace
metadata:
  name: custom-namespace
```

使用kubectl命令创建。

```
$ kubectl create -f custom-namespace.yaml
namespace/custom-namespace created
```

您还可以使用kubectl create namespace命令创建。

```
$ kubectl create namespace custom-namespace
namespace/custom-namespace created
```

6.2 管理命名空间

使用命名空间

- 创建工作负载时，您可以选择对应的命名空间，实现资源或租户的隔离。
- 查询工作负载时，选择对应的命名空间，查看对应命名空间下的所有工作负载。

命名空间使用实践

- **按照不同环境划分命名空间**

  一般情况下，工作负载发布会经历开发环境、联调环境、测试环境，最后到生产环境的过程。这个过程中不同环境部署的工作负载相同，只是在逻辑上进行了定义。分为两种做法：

  - 分别创建不同集群。

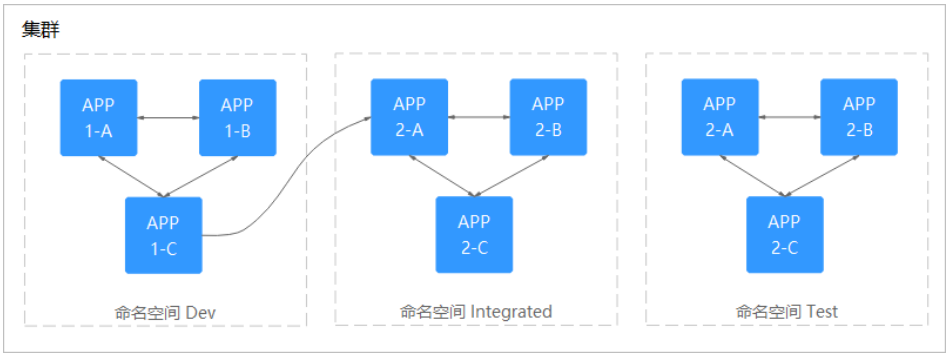
    不同集群之间，资源不能共享。同时，不同环境中的服务互访需要通过负载均衡才能实现。

  - 不同环境创建对应命名空间。

    同个命名空间下，通过服务名称（Service name）可直接访问。跨命名空间的可以通过服务名称、命名空间名称访问。

    例如下图，开发环境/联调环境/测试环境分别创建了命名空间。

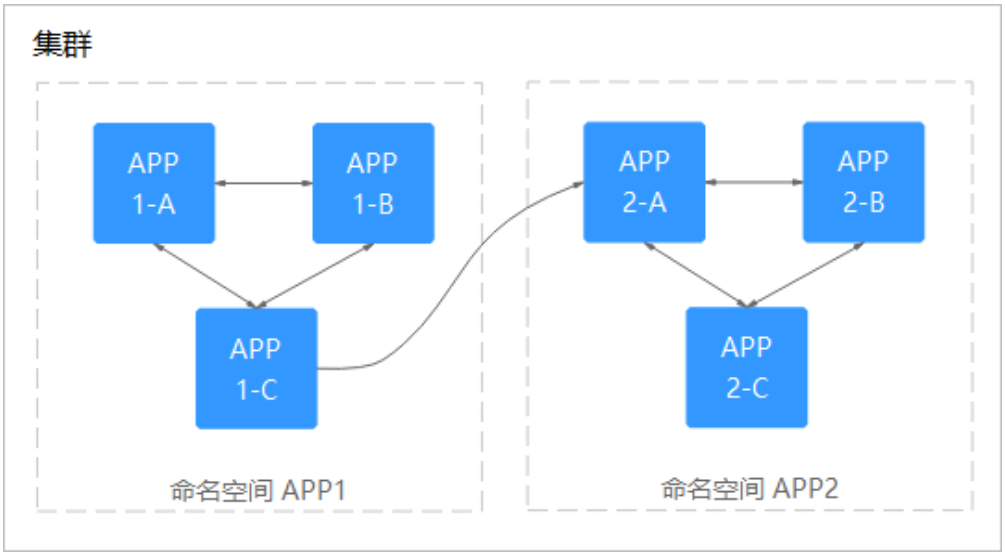
    图 6-1 不同环境创建对应命名空间

- **按照应用划分命名空间**

  对于同个环境中，应用数量较多的情况，建议进一步按照工作负载类型划分命名空间。例如下图中，按照APP1和APP2划分不同命名空间，将不同工作负载在逻辑上当做一个工作负载组进行管理。且同一个命名空间内的工作负载可以通过服务名称访问，不同命名空间下的通过服务名称、命名空间名称访问。

图 6-2 按照工作负载划分命名空间



管理命名空间标签

步骤1 登录CCE控制台，单击集群名称进入集群，在左侧选择“命名空间”。

步骤2 单击目标命名空间操作列的“更多 > 标签管理”。

步骤3 弹出的窗口中将展示命名空间已有的标签，您可根据需要进行修改。

- 添加标签：单击“添加”，填写需要增加标签的“键”和“值”，单击“确定”。
  例如，填写的键为“project”，值为“cicd”，就可以从逻辑概念表示该命名空间是用来部署CICD环境使用。
- 删除标签：单击需要删除标签前的⊖，并单击“确定”。

图 6-3 添加或删除命名空间标签

步骤4 标签修改成功后，再次进入该界面，在“标签”列下可查看到已经修改的标签。

----结束

删除命名空间

删除命名空间会删除该命名空间下所有的资源（如工作负载，短任务、配置项等），请谨慎操作。

步骤1 登录CCE控制台，进入集群。

步骤2 在左侧导航栏中选择“命名空间”，选中待删除的命名空间，单击“更多 > 删除”。

根据系统提示进行删除操作。系统内置的命名空间不支持删除。

----结束

6.3 设置资源配额及限制

通过设置命名空间级别的资源配额，实现多团队或多用户在共享集群资源的情况下限制团队、用户可以使用的资源总量，包括限制命名空间下创建某一类型对象的数量以及对象消耗计算资源（CPU、内存）的总量。

背景信息

默认情况下，运行中的Pod可以无限制地使用Node节点上的CPU和内存，这意味着任意一个Pod都可以无节制地使用集群的计算资源，某个命名空间的Pod可能会耗尽集群的所有资源。

kubernetes在一个物理集群上提供了多个虚拟集群，这些虚拟集群被称为命名空间。命名空间可用于多种工作用途，满足多用户的使用需求，通过为每个命名空间配置资源额度可以有效限制资源滥用，从而保证集群的可靠性。

您可为命名空间配置包括CPU、内存、Pod数量等资源的额度，更多信息请参见**资源配额**。

约束与限制

在Kubernetes中，外部用户及内部组件频繁的数据更新操作采用乐观并行的控制方法。通过定义资源版本（resourceVersion）实现乐观锁，资源版本字段包含在对象的元数据（metadata）中。这个字段标识了对象的内部版本号，且对象被修改时，该字段将随之修改。kube-apiserver可以通过该字段判断对象是否已经被修改。当包含resourceVersion的更新请求到达apiserver，服务器端将对比请求数据与服务器中数据的资源版本号，如果不一致，则表明在本次更新提交时，服务端对象已被修改，此时apiserver将返回冲突错误（409）。客户端需重新获取服务端数据，重新修改后再次提交到服务器端；而资源配额对每个命名空间的资源消耗总量提供限制，并且会记录集群中的资源信息，因此开启资源配额后，在大规模并发场景下创建资源冲突概率会变高，会影响批创资源性能。

操作步骤

步骤1 登录CCE控制台，单击集群名称进入集群。

步骤2 在左侧导航栏中选择“命名空间”。

步骤3 单击对应命名空间后的“管理配额”。

系统级别的命名空间kube-system、kube-public默认不支持设置资源配额。

步骤4 设置资源配额，然后单击“确定”。

须知

- 命名空间设置了CPU或内存资源配额后，创建工作负载时，必须指定CPU或内存的请求值（request）和约束值（limit），否则CCE将拒绝创建实例。若设置资源配额值为0，则不限制该资源的使用。
- 配额累计使用量包含CCE系统默认创建的资源，如default命名空间下系统默认创建的kubernetes服务（该服务可通过后端kubectl工具查看）等，故建议命名空间下的资源配额略大于实际期望值以去除系统默认创建资源的影响。

----结束

7 配置项与密钥

7.1 创建配置项

操作场景

配置项（ConfigMap）是一种用于存储工作负载所需配置信息的资源类型，内容由用户决定。配置项创建完成后，可在容器工作负载中作为文件或者环境变量使用。

配置项允许您将配置文件从容器镜像中解耦，从而增强容器工作负载的可移植性。

配置项价值如下：

- 使用配置项功能可以帮您管理不同环境、不同业务的配置。
- 方便您部署相同工作负载的不同环境，配置文件支持多版本，方便您进行更新和回滚工作负载。
- 方便您快速将您的配置以文件的形式导入到容器中。

约束与限制

- ConfigMap资源文件大小不得超过1MB。
- **静态Pod**中不可使用ConfigMap。

操作步骤

步骤1 登录CCE控制台，单击集群名称进入集群。

步骤2 在左侧导航栏中选择“配置与密钥”，在右上角单击“创建配置项”。

步骤3 填写参数。

表 7-1 新建配置参数说明

参数	参数说明
名称	新建的配置项名称，同一个命名空间里命名必须唯一。
命名空间	新建配置项所在的命名空间。若不选择，默认为default。

参数	参数说明
描述	配置项的描述信息。
配置数据	配置项的数据。 键值对形式，单击 ┼ 添加。其中值支持String、JSON和YAML格式。
标签	配置项的标签。键值对形式，输入键值对后单击“确认添加”。

步骤4 配置完成后，单击“确定”。

工作负载配置列表中会出现新创建的工作负载配置。

----结束

使用 kubectl 创建配置项

步骤1 请参见**通过kubectl连接集群**配置kubectl命令。

步骤2 创建并编辑cce-configmap.yaml文件。

vi cce-configmap.yaml

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cce-configmap
data:
  SPECIAL_LEVEL: Hello
  SPECIAL_TYPE: CCE
```

表 7-2 关键参数说明

参数	说明
apiVersion	固定值为v1。
kind	固定值为ConfigMap。
metadata.name	配置项名称，可自定义。
data	配置项的数据，需填写键值对形式。

步骤3 创建配置项。

kubectl create -f cce-configmap.yaml

查看已创建的配置项。

kubectl get cm

```
NAME            DATA      AGE
cce-configmap     3       7m
```

----结束

相关操作

配置项创建完成后，您还可以执行表7-3中的操作。

表 7-3 其他操作

操作	说明
编辑YAML	单击配置项名称后的“编辑YAML”，可编辑当前配置项的YAML文件。
更新配置	1. 选择需要更新的配置项名称，单击“更新”。 2. 根据表7-1更改信息。 3. 单击“确定”。
删除配置	选择要删除的配置项，单击“删除”。 根据系统提示删除配置。

7.2 使用配置项

配置项创建后，可在工作负载环境变量、命令行参数和数据卷三个场景使用。

- 通过配置项设置工作负载环境变量
- 通过配置项设置命令行参数
- 使用配置项挂载到工作负载数据卷

本节以下面这个ConfigMap为例，具体介绍ConfigMap的用法。

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cce-configmap
data:
  SPECIAL_LEVEL: Hello
  SPECIAL_TYPE: CCE
```

须知

- 在工作负载里使用ConfigMap时，需要工作负载和ConfigMap处于同一集群和命名空间中。
- 以数据卷挂载使用ConfigMap时，当ConfigMap被更新，Kubernetes会同时更新数据卷中的数据。

  对于以subPath形式挂载的ConfigMap数据卷，当ConfigMap被更新时，Kubernetes无法自动更新数据卷中的数据。
- 以环境变量方式使用ConfigMap时，当ConfigMap被更新，数据不会被自动更新。更新这些数据需要重新启动Pod。

通过配置项设置工作负载环境变量

使用控制台方式

步骤1 登录CCE控制台，单击集群名称进入集群。

步骤2 在左侧导航栏选择“工作负载”，单击右上角“创建工作负载”。

在创建工作负载时，在“容器配置”中找到“环境变量”，单击“新增变量”。

- **配置项导入**：选择一个配置项，将配置项中所有键值都导入为环境变量。

- **配置项键值导入**：将配置项中某个键的值导入作为某个环境变量的值。
  - 变量名称：工作负载中的环境变量名称，可自定义，默认为配置项中选择的键名。
  - 变量/变量引用：选择一个配置项及需要导入的键名，将其对应的值导入为工作负载环境变量。

  例如将cce-configmap这个配置项中“SPECIAL_LEVEL”的值“Hello”导入，作为工作负载环境变量“SPECIAL_LEVEL”的值，导入后容器中有一个名为“SPECIAL_LEVEL”的环境变量，其值为“Hello”。

步骤3 配置其他工作负载参数后，单击“创建工作负载”。

等待工作负载正常运行后，您可**登录容器**执行以下语句，查看该配置项是否已被设置为工作负载的环境变量。

```
printenv SPECIAL_LEVEL
```

示例输出如下：

```
Hello
```

----结束

使用kubectl方式

步骤1 请参见**通过kubectl连接集群**配置kubectl命令。

步骤2 创建并编辑nginx-configmap.yaml文件。

vi nginx-configmap.yaml

YAML文件内容如下：

- **配置项导入**：如果要将一个配置项中所有数据都添加到环境变量中，可以使用envFrom参数，配置项中的Key会成为工作负载中的环境变量名称。

```
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:                    # 使用envFrom来指定环境变量引用的配置项
        - configMapRef:
            name: cce-configmap     # 引用的配置项名称
      imagePullSecrets:
      - name: default-secret
```

- **配置项键值导入**：您可以在创建工作负载时将配置项设置为环境变量，使用valueFrom参数单独引用ConfigMap中的Key/Value。

```
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:                        # 设置工作负载中的环境变量
        - name: SPECIAL_LEVEL       # 工作负载中的环境变量名称
          valueFrom:                # 使用valueFrom来指定环境变量引用配置项
            configMapKeyRef:
              name: cce-configmap   # 引用的配置项名称
              key: SPECIAL_LEVEL    # 引用的配置项中的key
        - name: SPECIAL_TYPE        # 添加多个环境变量参数，可同时导入多个环境变量
          valueFrom:
            configMapKeyRef:
              name: cce-configmap
              key: SPECIAL_TYPE
      imagePullSecrets:
      - name: default-secret
```

步骤3 创建工作负载。

```
kubectl apply -f nginx-configmap.yaml
```

步骤4 创建完成后，查看Pod中的环境变量。

1. 执行以下命令，查看已创建的Pod。

   ```
   kubectl get pod | grep nginx-configmap
   ```

   预期输出如下：

   ```
   nginx-configmap-***   1/1     Running   0          2m18s
   ```

2. 执行以下命令，查看该Pod中的环境变量。

   ```
   kubectl exec nginx-configmap-*** -- printenv SPECIAL_LEVEL SPECIAL_TYPE
   ```

   预期输出如下：

   ```
   Hello
   CCE
   ```


说明该配置项已被设置为工作负载的环境变量。

----结束

通过配置项设置命令行参数

您可以使用配置项作为环境变量来设置容器中的命令或者参数值，使用环境变量替换语法$VAR_NAME来进行。

使用控制台方式

步骤1 登录CCE控制台，单击集群名称进入集群。

步骤2 在左侧导航栏选择“工作负载”，单击右上角“创建工作负载”。

在创建工作负载时，在“容器配置”中找到“环境变量”，单击“新增变量”。本例中以“配置项导入”为例。

- **配置项导入**：选择一个配置项，将配置项中所有键值都导入为环境变量。

步骤3 在“容器配置”中找到“生命周期”，在右侧选择“启动后处理”页签，并填写以下参数。

- 处理方式：命令行脚本。
- 执行命令：以下命令需分三行填写，其中*SPECIAL_LEVEL*和*SPECIAL_TYPE*为工作负载中的环境变量名，即cce-configmap配置项中的键名。

```
/bin/bash
-c
echo $SPECIAL_LEVEL $SPECIAL_TYPE > /usr/share/nginx/html/index.html
```

步骤4 配置其他工作负载参数后，单击“创建工作负载”。

等待工作负载正常运行后，您可登录容器执行以下语句，查看该配置项是否已被设置为工作负载的环境变量。

```
cat /usr/share/nginx/html/index.html
```

示例输出如下：

```
Hello CCE
```

----结束

使用kubectl方式

步骤1 请参见**通过kubectl连接集群**配置kubectl命令。

步骤2 创建并编辑nginx-configmap.yaml文件。

vi nginx-configmap.yaml

如下面的示例所示，在工作负载中导入了cce-configmap配置项，其中*SPECIAL_LEVEL*和*SPECIAL_TYPE*为工作负载中的环境变量名，即cce-configmap配置项中的键名。

```
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:                    # 使用envFrom来指定环境变量引用的配置项
        - configMapRef:
            name: cce-configmap     # 引用的配置项名称
      imagePullSecrets:
        - name: default-secret
```

步骤3 创建工作负载。

kubectl apply -f nginx-configmap.yaml

步骤4 等待工作负载正常运行后，容器中的/usr/share/nginx/html/index.html文件将被输入如下内容。

1. 执行以下命令，查看已创建的Pod。

   ```
   kubectl get pod | grep nginx-configmap
   ```

   预期输出如下：

   ```
   nginx-configmap-***   1/1     Running   0          2m18s
   ```

2. 执行以下命令，查看该Pod中的环境变量。

   ```
   kubectl exec nginx-configmap-*** -- cat /usr/share/nginx/html/index.html
   ```

   预期输出如下：

   ```
   Hello CCE
   ```

----结束

使用配置项挂载到工作负载数据卷

配置项(ConfigMap)挂载是将配置项中的数据挂载到指定的容器路径。平台提供工作负载代码和配置文件的分离，“配置项挂载”用于处理工作负载配置参数。用户需要提前创建工作负载配置，操作步骤请参见**创建配置项**。

使用控制台方式

步骤1 登录CCE控制台，单击集群名称进入集群。

步骤2 在左侧导航栏选择“工作负载”，单击右上角“创建工作负载”。

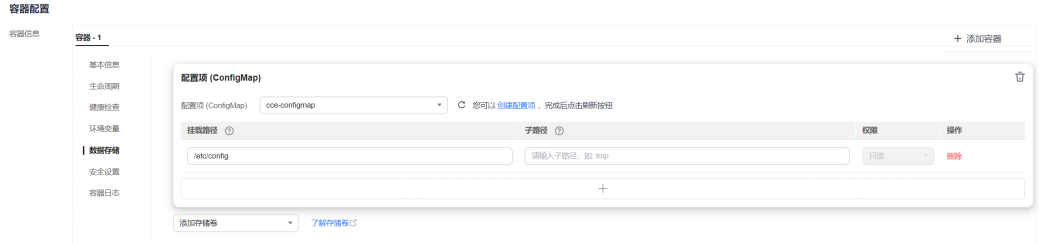
在创建工作负载时，在“容器配置”中找到“数据存储”，选择“添加存储卷 > 配置项(ConfigMap)”。

步骤3 选择配置项挂载参数，如表7-4。

表 7-4 配置项挂载

参数	参数说明
配置项	选择对应的配置项名称。 配置项需要提前创建，具体请参见创建配置项。
挂载路径	请输入挂载路径。配置项挂载完成后，会在容器中的挂载路径下生成以配置项中的key为文件名， value为文件内容的配置文件。 数据存储挂载到容器上的路径。请不要挂载在系统目录下，如“/”、“/var/run”等，会导致容器异常。建议挂载在空目录下，若目录不为空，请确保目录下无影响容器启动的文件，否则文件会被替换，导致容器启动异常，工作负载创建失败。 **须知** 挂载高危目录的情况下 ，建议使用低权限账号启动，否则可能会造成宿主机高危文件被破坏。
子路径	请输入挂载路径的子路径。 • 使用子路径挂载本地磁盘，实现在单一Pod中重复使用同一个Volume，不填写时默认为根。 • 子路径可以填写ConfigMap的键值，子路径若填写为不存在的键值则数据导入不会生效。 • 通过子路径导入的数据不会随ConfigMap的更新而动态更新。
权限	只读。只能读容器路径中的数据卷。

图 7-1 使用配置项挂载到工作负载数据卷



步骤4 其余信息都配置完成后，单击“创建工作负载”。

等待工作负载正常运行后，本示例将在/etc/config目录下生成SPECIAL_LEVEL和SPECIAL_TYPE两个文件，且文件的内容分别为Hello和CCE。

您可登录容器执行以下语句，查看容器中的SPECIAL_LEVEL或SPECIAL_TYPE文件。

```
cat /etc/config/SPECIAL_LEVEL
```

预期输出如下：

```
Hello
```

----结束

使用kubectl方式

步骤1 请参见**通过kubectl连接集群**配置kubectl命令。

步骤2 创建并编辑nginx-configmap.yaml文件。

vi nginx-configmap.yaml

如下面的示例所示，配置项挂载完成后，最终会在容器中的/etc/config目录下生成以配置项中的key为文件名， value为文件内容的配置文件。

```
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          # 挂载到/etc/config目录下
          readOnly: true
    volumes:
    - name: config-volume
      configMap:
        name: cce-configmap               # 引用的配置项名称
```

步骤3 创建工作负载。

kubectl apply -f nginx-configmap.yaml

步骤4 等待工作负载正常运行后，在/etc/config目录下会生成SPECIAL_LEVEL和SPECIAL_TYPE两个文件，且文件的内容分别为Hello和CCE。

1. 执行以下命令，查看已创建的Pod。
   ```
   kubectl get pod | grep nginx-configmap
   ```
   预期输出如下：
   ```
   nginx-configmap-***   1/1     Running   0          2m18s
   ```
2. 执行以下命令，查看该Pod中的SPECIAL_LEVEL或SPECIAL_TYPE文件。
   ```
   kubectl exec nginx-configmap-*** -- cat /etc/config/SPECIAL_LEVEL
   ```
   预期输出如下：
   ```
   Hello
   ```

----结束

7.3 创建密钥

操作场景

密钥（Secret）是一种用于存储工作负载所需要认证信息、密钥的敏感信息等的资源类型，内容由用户决定。资源创建完成后，可在容器工作负载中作为文件或者环境变量使用。

约束与限制

静态Pod中不可使用Secret。

操作步骤

步骤1 登录CCE控制台，单击集群名称进入集群。

步骤2 在左侧导航栏中选择“配置与密钥”，选择“密钥”页签，在右上角单击“创建密钥”。

步骤3 填写参数。

表 7-5 基本信息说明

参数	参数说明
名称	新建的密钥的名称，同一个命名空间内命名必须唯一。
命名空间	新建密钥所在的命名空间，默认为default。
描述	密钥的描述信息。
密钥类型	新建的密钥类型。 • Opaque：一般密钥类型。 • kubernetes.io/dockerconfigjson：存放拉取私有仓库镜像所需的认证信息。 • kubernetes.io/tls：Kubernetes的TLS密钥类型，用于存放7层负载均衡服务所需的证书。kubernetes.io/tls类型的密钥示例及说明请参见**TLS Secret**。 • IngressTLS：CCE提供的TLS密钥类型，用于存放7层负载均衡服务所需的证书。 • 其他：若需要创建其他类型的密钥，请手动输入密钥类型。

参数	参数说明
密钥数据	工作负载密钥的数据可以在容器中使用。 • 当密钥为Opaque类型时，单击＋，在弹出的窗口中输入键值对，并且可以勾选“自动Base64转码”。 • 当密钥为kubernetes.io/dockerconfigjson类型时，输入私有镜像仓库的账号和密码。 • 当密钥为kubernetes.io/tls或IngressTLS类型时，上传证书文件和私钥文件。 **说明** – 证书是自签名或CA签名过的凭据，用来进行身份认证。 – 证书请求是对签名的请求，需要使用私钥进行签名。
密钥标签	密钥的标签。键值对形式，输入键值对后单击“确认添加”。

步骤4 配置完成后，单击“确定”。

密钥列表中会出现新创建的密钥。

----结束

Secret 资源文件配置示例

本章节主要介绍Secret类型的资源描述文件的配置示例。

- Opaque类型

  定义的Secret文件secret.yaml内容如下。其中data字段以键值对的形式填写，value需要用Base64编码，Base64编码方法请参见**如何进行Base64编码**。

  ```
  apiVersion: v1
  kind: Secret
  metadata:
    name: mysecret          # secret的名称
    namespace: default      #命名空间，默认为default
  data:
    <your_key>: <your_value>  #填写键值对，其中value需要用Base64编码
  type: Opaque
  ```

- kubernetes.io/dockerconfigjson类型

  定义的Secret文件secret.yaml内容如下。其中.dockerconfigjson需要用Base64，Base64编码方法请参见**如何进行Base64编码**。

  ```
  apiVersion: v1
  kind: Secret
  metadata:
    name: mysecret          # secret的名称
    namespace: default      #命名空间，默认为default
  data:
    .dockerconfigjson: eyJh*****    #Base64编码后的内容
  type: kubernetes.io/dockerconfigjson
  ```

  获取.dockerconfigjson内容的步骤如下：

  a. 获取镜像仓库的登录信息：

     - 镜像仓库地址：本文中以address为例，请根据实际信息替换。

- 用户名：本文中以username为例，请根据实际信息替换。
- 密码：本文中以password为例，请根据实际信息替换。

b. 使用Base64将键值对username:password进行编码，获取编码后的内容填入3中。

```
echo -n "username:password" | base64
```

回显如下：

```
dXNlcm5hbWU6cGFzc3dvcmQ=
```

c. 使用Base64对以下JSON内容进行编码。

```
echo -n '{"auths":{"address":
{"username":"username","password":"password","auth":"dXNlcm5hbWU6cGFzc3dvcmQ="}}}'
| base64
```

回显如下：

```
eyJhdXRocyI6eyJhZGRyZXNzIjp7InVzZXJuYW1lIjoidXNlcm5hbWUiLCJwYXNzd29yZCI6InBhc3N3b3JkIiwiYXV0aCI6ImRYTmxjbTVoYldVNmNHRnpjM2R2Y21RPSJ9fX0=
```

编码后的内容即为.dockerconfigjson内容。

- kubernetes.io/tls类型

  其中tls.crt和tls.key需要用Base64，Base64编码方法请参见**如何进行Base64编码**。

```
kind: Secret
apiVersion: v1
metadata:
  name: mysecret          # secret的名称
  namespace: default      #命名空间，默认为default
data:
  tls.crt: LS0tLS1CRU*****FURS0tLS0t  #证书内容，需要Base64编码
  tls.key: LS0tLS1CRU*****VZLS0tLS0=  #私钥内容，需要Base64编码
type: kubernetes.io/tls
```

- IngressTLS类型

  其中tls.crt和tls.key需要用Base64，Base64编码方法请参见**如何进行Base64编码**。

```
kind: Secret
apiVersion: v1
metadata:
  name: mysecret          # secret的名称
  namespace: default      #命名空间，默认为default
data:
  tls.crt: LS0tLS1CRU*****FURS0tLS0t  #证书内容，需要Base64编码
  tls.key: LS0tLS1CRU*****VZLS0tLS0=  #私钥内容，需要Base64编码
type: IngressTLS
```

使用 kubectl 创建密钥

步骤1 请参见**通过kubectl连接集群**配置kubectl命令。

步骤2 通过Base64编码，创建并编辑cce-secret.yaml文件。

```
# echo -n "待编码内容" | base64
******
```

vi cce-secret.yaml

Opaque类型的YAML示例如下，其余类型请参见**Secret资源文件配置示例**：

```
apiVersion: v1
kind: Secret
metadata:
  name: mysecret
type: Opaque
```



```
data:
  <your_key>: <your_value>  #填写键值对，其中value需要用Base64编码
```

步骤3 创建密钥。

kubectl create -f cce-secret.yaml

创建完成后可以查询到密钥。

kubectl get secret -n default

----结束

相关操作

密钥创建完成后，您还可以执行表7-6中的操作。

说明

密钥列表中包含系统密钥资源，系统密钥资源不可更新，也不能删除，只能查看。

表 7-6 其他操作

操作	说明
编辑YAML	单击密钥名称后的“编辑YAML”，可编辑当前密钥的YAML文件。
更新密钥	1. 选择需要更新的密钥名称，单击“更新”。 2. 根据表7-5更改信息。 3. 单击“确定”。
删除密钥	选择要删除的密钥，单击“删除”。 根据系统提示删除密钥。
批量删除密钥	1. 勾选需要删除的密钥名称。 2. 单击页面左上角的“批量删除”，删除选中的密钥。 3. 根据系统提示删除密钥。

如何进行 Base64 编码

对字符串进行Base64编码，可以直接使用“echo -n 要编码的内容 | base64”命令即可，示例如下：

```
root@ubuntu:~# echo -n "待编码内容" | base64
******
```

7.4 使用密钥

密钥创建后，可在工作负载环境变量和数据卷两个场景使用。

须知

请勿对以下CCE系统使用的密钥做任何操作，详情请参见**集群系统密钥说明**。

- 请不要操作kube-system下的secrets。
- 请不要操作任何命名空间下的default-secret、paas.elb。其中，default-secret用于SWR的私有镜像拉取，paas.elb用于该命名空间下的服务对接ELB。

- **使用密钥设置工作负载的环境变量**
- **使用密钥配置工作负载的数据卷**

本节以下面这个Secret为例，具体介绍Secret的用法。

```
apiVersion: v1
kind: Secret
metadata:
  name: mysecret
type: Opaque
data:
  username: ******  #需要用Base64编码
  password: ******  #需要用Base64编码
```

须知

- 在Pod里使用密钥时，需要Pod和密钥处于同一集群和命名空间中。
- 当Secret 被更新时，Kubernetes会同时更新数据卷中的数据。

  但对于以**subPath**形式挂载的Secret数据卷，当Secret 被更新时，Kubernetes无法自动更新数据卷中的数据。

使用密钥设置工作负载的环境变量

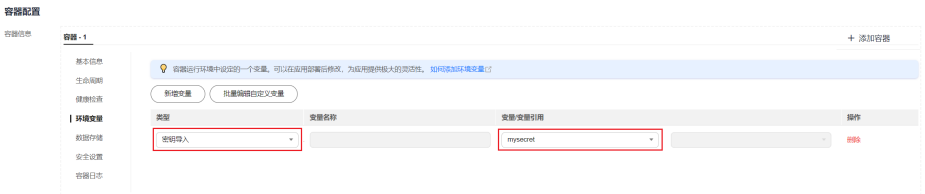
使用控制台方式

步骤1 登录CCE控制台，单击集群名称进入集群。

步骤2 在左侧导航栏选择“工作负载”，单击右上角“创建工作负载”。

在创建工作负载时，在“容器配置”中找到“环境变量”，单击“新增变量”。

- **密钥导入**：选择一个密钥，将密钥中所有键值都导入为环境变量。



- **密钥项键值导入**：将密钥中某个键的值导入作为某个环境变量的值。
  - 变量名称：工作负载中的环境变量名称，可自定义，默认为密钥中选择的键名。
  - 变量/变量引用：选择一个密钥及需要导入的键名，将其对应的值导入为工作负载环境变量。

例如将mysecret这个密钥中“username”的值导入，作为工作负载环境变量“username”的值，导入后容器中将会有一个名为“username”的环境变量。

步骤3 配置其他工作负载参数后，单击“创建工作负载”。

等待工作负载正常运行后，您可**登录容器**执行以下语句，查看该密钥是否已被设置为工作负载的环境变量。

```
printenv username
```

如输出与Secret中的内容一致，则说明该密钥已被设置为工作负载的环境变量。

----结束

使用kubectl方式

步骤1 请参见**通过kubectl连接集群**配置kubectl命令。

步骤2 创建并编辑nginx-secret.yaml文件。

vi nginx-secret.yaml

YAML文件内容如下：

- **密钥导入**：如果要将一个密钥中所有数据都添加到环境变量中，可以使用envFrom参数，密钥中的Key会成为工作负载中的环境变量名称。

```
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:                # 使用envFrom来指定环境变量引用的密钥
        - secretRef:
            name: mysecret      # 引用的密钥名称
      imagePullSecrets:
      - name: default-secret
```

- **密钥键值导入**：您可以在创建工作负载时将密钥设置为环境变量，使用valueFrom参数单独引用Secret中的Key/Value。

```
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:                          # 设置工作负载中的环境变量
      - name: SECRET_USERNAME       # 工作负载中的环境变量名称
        valueFrom:                  # 使用valueFrom来指定环境变量引用的密钥
          secretKeyRef:
            name: mysecret          # 引用的密钥名称
            key: username           # 引用的密钥中的key
      - name: SECRET_PASSWORD       # 添加多个环境变量参数，可同时导入多个环境变量
        valueFrom:
          secretKeyRef:
            name: mysecret
            key: password
    imagePullSecrets:
    - name: default-secret
```

步骤3 创建工作负载。

kubectl apply -f nginx-secret.yaml

步骤4 创建完成后，查看Pod中的环境变量。

1. 执行以下命令，查看已创建的Pod。
   ```
   kubectl get pod | grep nginx-secret
   ```
   预期输出如下：
   ```
   nginx-secret-***   1/1     Running   0          2m18s
   ```
2. 执行以下命令，查看该Pod中的环境变量。
   ```
   kubectl exec nginx-secret-*** -- printenv SPECIAL_USERNAME SPECIAL_PASSWORD
   ```
   如输出与Secret中的内容一致，则说明该密钥已被设置为工作负载的环境变量。

----结束

使用密钥配置工作负载的数据卷

密钥(Secret)挂载将密钥中的数据挂载到指定的容器路径，密钥内容由用户决定。用户需要提前创建密钥，操作步骤请参见**创建密钥**。

使用控制台方式

步骤1 登录CCE控制台，单击集群名称进入集群。

步骤2 在左侧导航栏选择“工作负载”，在右侧选择“无状态负载”页签。单击右上角“创建工作负载”。

在创建工作负载时，在“容器配置”中找到“数据存储”，选择“添加存储卷 > 密钥(Secret)”。

步骤3 选择密钥挂载参数，如**表7-7**。

表 7-7 密钥挂载

参数	参数说明
密钥	选择对应的密钥名称。 密钥需要提前创建，具体请参见**创建密钥**。
挂载路径	请输入挂载路径。密钥挂载完成后，会在容器中的挂载路径下生成以密钥中的key为文件名， value为文件内容的密钥文件。 数据存储挂载到容器上的路径。请不要挂载在系统目录下，如“/”、“**/var/run**”等，会导致容器异常。建议挂载在空目录下，若目录不为空，请确保目录下无影响容器启动的文件，否则文件会被替换，导致容器启动异常，工作负载创建失败。 **须知** 挂载高危目录的情况下 ，建议使用低权限账号启动，否则可能会造成宿主机高危文件被破坏。
子路径	请输入挂载路径的子路径。 • 使用子路径挂载本地磁盘，实现在单一Pod中重复使用同一个Volume，不填写时默认为根。 • 子路径可以填写Secret的键值，子路径若填写为不存在的键值则数据导入不会生效。 • 通过子路径导入的数据不会随Secret的更新而动态更新。
权限	只读。只能读容器路径中的数据卷。

图 7-2 使用密钥挂载到工作负载数据卷

步骤4 其余信息都配置完成后，单击“创建工作负载”。

等待工作负载正常运行后，本示例将在/etc/foo目录下生成username和password两个文件，且文件的内容分别为密钥值。

您可**登录容器**执行以下语句，查看容器中的username和password两个文件。

```
cat /etc/foo/username
```

预期输出与Secret中的内容一致。

----结束

使用kubectl方式

步骤1 请参见**通过kubectl连接集群**配置kubectl命令。

步骤2 创建并编辑nginx-secret.yaml文件。

vi nginx-secret.yaml

如下面的示例所示，mysecret密钥的username和password以文件方式保存在/etc/foo目录下。

```
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          # 挂载到/etc/foo目录下
          readOnly: true
    volumes:
    - name: foo
      secret:
        secretName: mysecret      # 引用的密钥名称
```

您还可以使用items字段控制Secret键的映射路径，例如，将username存放在容器中的/etc/foo/my-group/my-username目录下。

说明

- 使用items字段指定Secret键的映射路径后，没有被指定的键将不会被以文件形式创建。例如，下面的例子中的password键未被指定，则该文件将不会被创建。
- 如果要使用Secret中全部的键，那么必须将全部的键都列在items字段中。
- items字段中列出的所有键必须存在于相应的Secret 中。否则，该卷不被创建。

```
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          # 挂载到/etc/foo目录下
          readOnly: true
    volumes:
    - name: foo
      secret:
        secretName: mysecret      # 引用的密钥名称
        items:
        - key: username      # 引用的密钥中的键名
          path: my-group/my-username    # Secret键的映射路径
```


步骤3 创建工作负载。

kubectl apply -f nginx-secret.yaml

步骤4 等待工作负载正常运行后，在/etc/foo目录下会生成username和password两个文件。

1. 执行以下命令，查看已创建的Pod。
   ```
   kubectl get pod | grep nginx-secret
   ```
   预期输出如下：
   ```
   nginx-secret-***   1/1     Running   0          2m18s
   ```
2. 执行以下命令，查看该Pod中的username或password文件。
   ```
   kubectl exec nginx-secret-*** -- cat /etc/foo/username
   ```
   预期输出与Secret中的内容一致。

----结束

7.5 集群系统密钥说明

CCE Autopilot集群默认会在每个命名空间下创建如下密钥：

- default-secret
- paas.elb

下面将详细介绍这个几个密钥的用途。

default-secret

default-secret的类型为kubernetes.io/dockerconfigjson，其data内容是登录SWR镜像仓库的凭据，用于从SWR拉取镜像。在CCE中创建工作负载时如果需要从SWR拉取镜像，需要配置imagePullSecrets的取值为default-secret，如下所示。

```
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
```

default-secret的data数据会定期更新，且当前的data内容会在一定时间后会过期失效。您可以使用describe命令在default-secret的中查看到具体的过期时间，如下所示。

须知

在使用时请直接使用default-secret，而不要复制secret内容重新创建，因为secret里面的凭据会过期，从而导致无法拉取镜像。


```
$ 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

paas.elb的data内容是临时AK/SK数据，创建节点或自动创建ELB时均会使用该密钥。paas.elb的data数据同样会定期更新，且在一定时间后会过期失效。

实际使用中您不会直接使用paas.elb，但请不要删除paas.elb，否则会导致创建节点或ELB失败。

8 弹性伸缩

8.1 工作负载弹性伸缩

8.1.1 工作负载伸缩原理

HPA 工作原理

HPA（Horizontal Pod Autoscaler）是用来控制Pod水平伸缩的控制器，HPA周期性检查Pod的度量数据，计算满足HPA资源所配置的目标数值所需的副本数量，进而调整目标资源（如Deployment）的replicas字段。

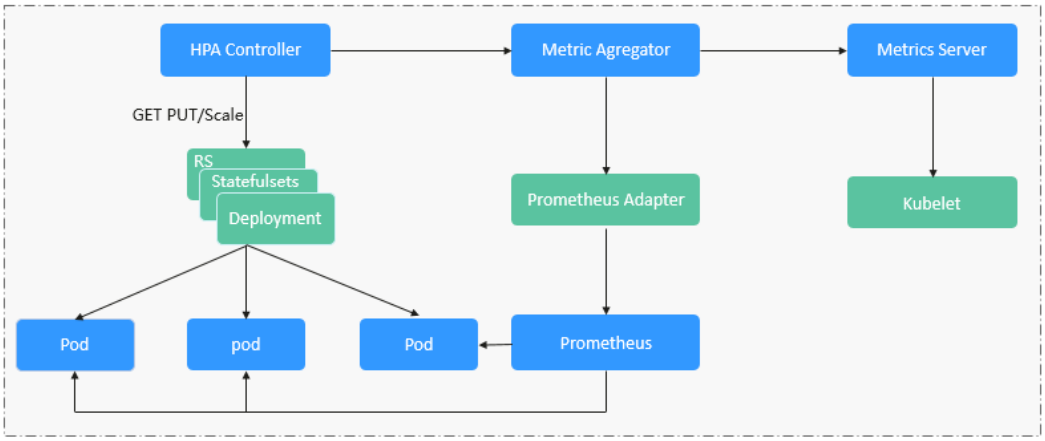
想要做到自动弹性伸缩，先决条件就是能感知到各种运行数据，例如集群节点、Pod、容器的CPU、内存使用率等等。而这些数据的监控能力Kubernetes也没有自己实现，而是通过其他项目来扩展Kubernetes的能力，Kubernetes提供**Prometheus**和**Metrics Server**插件来实现该能力：

- **Prometheus**是一套开源的系统监控报警框架，能够采集丰富的Metrics（度量数据），目前已经基本是Kubernetes的标准监控方案。
- **Metrics Server**是Kubernetes集群范围资源使用数据的聚合器。Metrics Server从kubelet公开的Summary API中采集度量数据，能够收集包括了Pod、Node、容器、Service等主要Kubernetes核心资源的度量数据，且对外提供一套标准的API。

使用HPA（Horizontal Pod Autoscaler）配合Metrics Server可以实现基于CPU和内存的自动弹性伸缩，再配合Prometheus还可以实现自定义监控指标的自动弹性伸缩。

HPA主要流程如**图8-1**所示。

图 8-1 HPA 流程图



HPA的核心有如下2个部分：

- 监控数据来源

  最早社区只提供基于CPU和Mem的HPA，随着应用越来越多搬迁到K8s上以及Prometheus的发展，开发者已经不满足于CPU和Memory，开发者需要应用自身的业务指标，或者是一些接入层的监控信息，例如：Load Balancer的QPS、网站的实时在线人数等。社区经过思考之后，定义了一套标准的Metrics API，通过聚合API对外提供服务。

  - metrics.k8s.io： 主要提供Pod和Node的CPU和Memory相关的监控指标。
  - custom.metrics.k8s.io： 主要提供Kubernetes Object相关的自定义监控指标。
  - external.metrics.k8s.io：指标来源外部，与任何的Kubernetes资源的指标无关。

- 扩缩容决策算法

  HPA controller根据当前指标和期望指标来计算缩放比例，计算公式如下：

  desiredReplicas = ceil[currentReplicas * (currentMetricValue / desiredMetricValue)]

  例如当前的指标值是200m，目标值是100m，那么按照公式计算期望的实例数就会翻倍。那么在实际过程中，可能会遇到实例数值反复伸缩，导致集群震荡。为了保证稳定性，HPA controller从以下几个方面进行优化：

  - 冷却时间：在1.11版本以及之前的版本，社区引入了horizontal-pod-autoscaler-downscale-stabilization-window和horizontal-pod-autoScaler-upscale-stabilization-window这两个启动参数代表缩容冷却时间和扩容冷却时间，这样保证在冷却时间内，跳过扩缩容。1.14版本之后引入延迟队列，保存一段时间内每一次检测的决策建议，然后根据当前所有有效的决策建议来进行决策，从而保证期望的副本数尽量小的发生变更，保证稳定性。
  - 忍受度：可以看成一个缓冲区，当实例变化范围在忍受范围之内的话，保持原有的实例数不变。

    首先定义ratio = currentMetricValue / desiredMetricValue

    当|ratio – 1.0| <= tolerance时，则会忽略，跳过scale。

    当|ratio – 1.0| > tolerance时， 就会根据之前的公式计算期望值。

    当前社区版本中默认值为0.1。

HPA是基于指标阈值进行伸缩的，常见的指标主要是 CPU、内存，也可以通过自定义指标，例如QPS、连接数等进行伸缩。但是存在一个问题：基于指标的伸缩存在一定的时延，这个时延主要包含：采集时延(分钟级) + 判断时延(分钟级) + 伸缩时延(分钟级)。这个分钟级的时延，可能会导致应用CPU飚高，响应时间变慢。为了解决这个问题，CCE提供了定时策略，对于一些有周期性变化的应用，提前扩容资源，而业务低谷时，定时回收资源。

8.1.2 HPA 策略

HPA策略即Horizontal Pod Autoscaling，是Kubernetes中实现POD水平自动伸缩的功能。该策略在Kubernetes社区HPA功能的基础上，增加了应用级别的冷却时间窗和扩缩容阈值等功能。

前提条件

使用HPA需要安装能够提供Metrics API的插件：

- **Kubernetes Metrics Server**：提供基础资源使用指标，例如容器CPU和内存使用率。
- **云原生监控插件**：根据自定义指标进行弹性伸缩需要将自定义指标聚合到Kubernetes API Server，详情请参见**使用自定义指标创建HPA策略**。

创建 HPA 策略

步骤1 在CCE控制台，单击集群名称进入集群。

步骤2 单击左侧导航栏的“策略”，切换至“HPA策略”页签，在右上角单击“创建HPA策略”。

步骤3 填写HPA基本信息。

- 策略名称：新建策略的名称，请自定义。
- 命名空间：请选择工作负载所在的命名空间。
- 关联工作负载：请选择要设置HPA策略的工作负载。

步骤4 填写HPA策略配置参数。

表 8-1 HPA 策略配置

参数	参数说明
实例范围	请输入最小实例数和最大实例数。 策略触发时，工作负载实例将在此范围内伸缩。

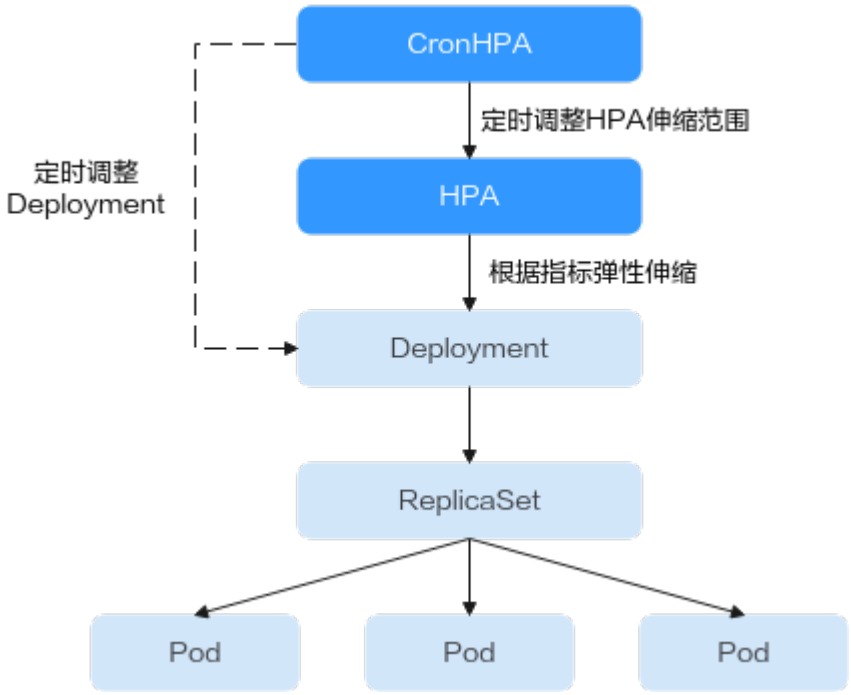
参数	参数说明
伸缩配置	• 系统默认：采用社区推荐的默认行为进行负载伸缩，详情请参见**社区默认行为说明**。 • 自定义：自定义扩/缩容配置的稳定窗口、步长、优先级等策略，实现更灵活的配置。未配置的参数将采用社区推荐的默认值。 – 禁止扩/缩容：选择是否禁止扩容或缩容。 – 稳定窗口：需要伸缩时，会在一段时间（设定的稳定窗口值）内持续检测，如在该时间段内始终需要进行伸缩（不满足设定的指标期望值）才进行伸缩，避免短时间的指标抖动造成异常。 – 步长策略：扩/缩容的步长，可设置一定时间内扩/缩容Pod数量或百分比。在存在多条策略时，可以选择使Pod数量最多或最少的策略。
系统策略	• 指标：可选择“CPU利用率”或“内存利用率”。 **说明** 利用率 = 工作负载容器组（Pod）的实际使用量 / 申请量 • 期望值：请输入期望资源平均利用率。 期望值表示所选指标的期望值，通过向上取整（当前指标值 / 期望值 × 当前实例数）来计算需要伸缩的实例数。 **说明** HPA在计算扩容、缩容实例数时，会选择最近5分钟内实例数的最大值。 • 容忍范围：指标处于范围内时不会触发伸缩，期望值必须在容忍范围之间。 当指标值大于缩容阈值且小于扩容阈值时，不会触发扩容或缩容。**阈值仅在1.15及以上版本的集群中支持**。

步骤5 设置完成后，单击“创建”。

----结束

8.1.3 CronHPA 定时策略

在一些复杂的业务场景下，可能有固定时间段高峰业务，又有日常突发高峰业务。此种情况下，用户既期望能定时弹性伸缩应对固定时间段高峰业务，又期望能根据指标弹性伸缩应对日常突发高峰业务。CCE提供CronHPA的自定义资源，实现在固定时间段对集群进行扩缩容，并且可以和HPA策略共同作用，定时调整HPA伸缩范围，实现复杂场景下的工作负载伸缩。



CronHPA支持定时调整HPA策略的最大和最小实例数，也可以直接定时调整Deployment的Pod实例数。

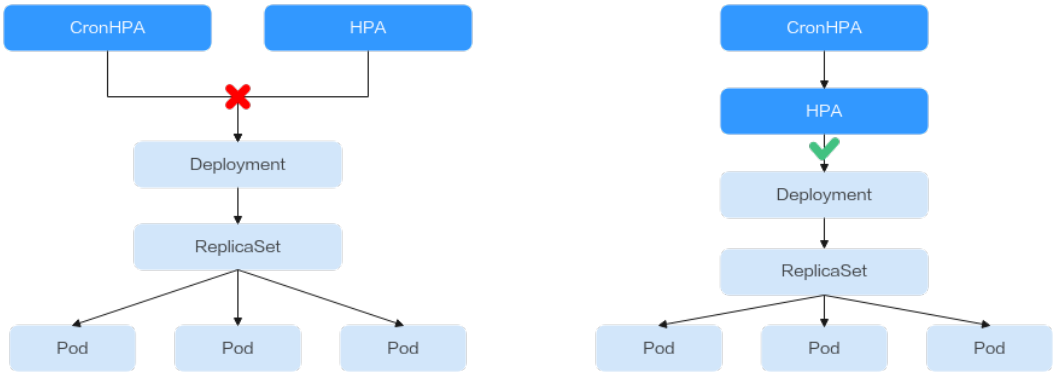
前提条件

集群中已安装**CCE容器弹性引擎**插件。

使用 CronHPA 调整 HPA 伸缩范围

CronHPA支持定时调整HPA策略的最大和最小实例数，满足复杂场景下的工作负载伸缩。

由于HPA与CronHPA均通过scaleTargetRef字段来获取伸缩对象，如果CronHPA和HPA同时设置Deployment为伸缩对象，两个伸缩策略相互独立，后执行的操作会覆盖先执行的操作，导致伸缩效果不符合预期，因此需避免这种情况发生。



在CronHPA与HPA共同使用时，CronHPA规则是在HPA策略的基础上生效的，CronHPA不会直接调整Deployment的副本数目，而是通过HPA来操作Deployment，因此了解以下参数可帮助您更好地理解其工作原理。

- CronHPA的目标实例数（targetReplicas）：表示CronHPA设定的实例数，在CronHPA生效时用于调整HPA的最大/最小实例数，从而间接调整Deployment实例数。
- HPA的最小实例数（minReplicas）：Deployment的实例数下限。
- HPA的最大实例数（maxReplicas）：Deployment的实例数上限。
- Deployment的实例数（replicas）：CronHPA策略生效之前Deployment的Pod数量。

在CronHPA规则生效时，通过比较目标实例数（targetReplicas）与实际Deployment的实例数，并结合HPA的最小实例数或最大实例数的数值大小，来调整Deployment实例数的上下限值。

图 8-2 CronHPA 扩缩容场景

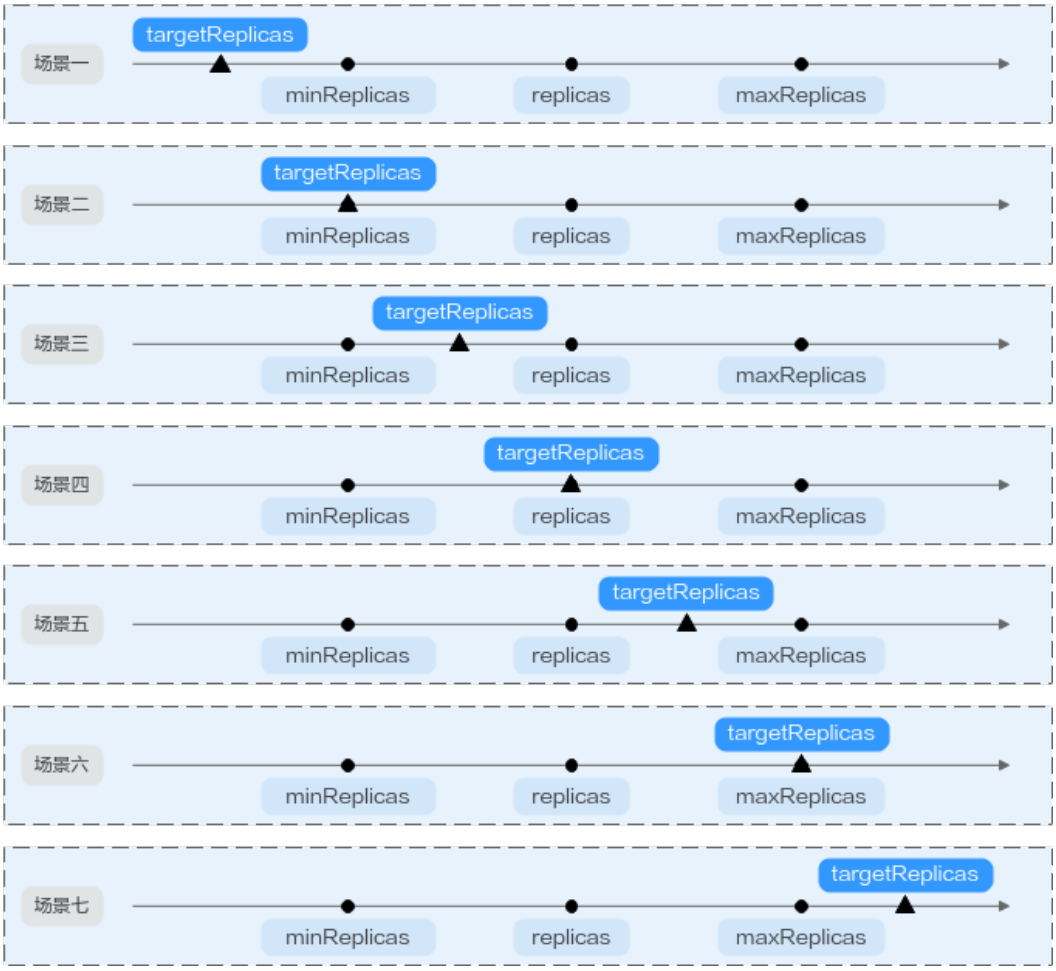


图8-2中为可能存在的扩缩容场景，如下表格以举例的形式说明了不同场景下CronHPA修改HPA的情况。

表 8-2 CronHPA 扩缩容场景

场景	场景说明	CronHPA (targetReplicas)	Deployment (replicas)	HPA (minReplicas / maxReplicas)	最终结果	操作说明
场景一	**targetReplicas** < minReplicas ≤ replicas ≤ maxReplicas	4	5	5/10	HPA: 4/10 Deployment: 5	CronHPA目标实例数低于HPA最小实例数（minReplicas）时： ● 修改HPA的最小实例数。 ● Deployment实例数无修改。
场景二	**targetReplicas** = minReplicas ≤ replicas ≤ maxReplicas	5	6	5/10	HPA: 5/10 Deployment: 6	CronHPA目标实例数等于HPA最小实例数（minReplicas）时： ● HPA的最小实例数无修改。 ● Deployment实例数无修改。
场景三	minReplicas < **targetReplicas** < replicas ≤ maxReplicas	4	5	1/10	HPA: 4/10 Deployment: 5	CronHPA目标实例数大于HPA最小实例数（minReplicas），小于Deployment实例数（replicas）时： ● 修改HPA的最小实例数。 ● Deployment实例数无修改。

场景	场景说明	CronHPA (targetReplicas)	Deployment (replicas)	HPA (minReplicas / maxReplicas)	最终结果	操作说明
场景四	minReplicas < **targetReplicas** = replicas < maxReplicas	5	5	1/10	HPA: 5/10 Deployment: 5	CronHPA目标实例数大于HPA最小实例数（minReplicas），等于Deployment实例数（replicas）时： ● 修改HPA的最小实例数。 ● Deployment实例数无修改。
场景五	minReplicas ≤ replicas < **targetReplicas** < maxReplicas	6	5	1/10	HPA: 6/10 Deployment: 6	CronHPA目标实例数大于Deployment实例数（replicas），小于HPA最大实例数（maxReplicas）时： ● 修改HPA的最小实例数。 ● 修改Deployment实例数。
场景六	minReplicas ≤ replicas < **targetReplicas** = maxReplicas	10	5	1/10	HPA: 10/10 Deployment: 10	CronHPA目标实例数大于Deployment实例数（replicas），等于HPA最大实例数（maxReplicas）时： ● 修改HPA的最小实例数。 ● 修改Deployment实例数。

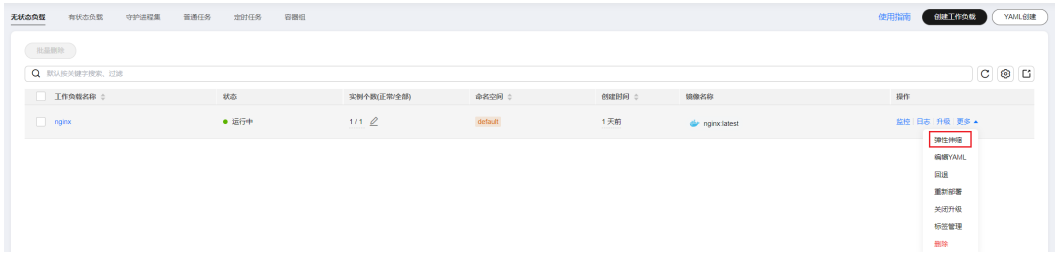
场景	场景说明	CronHPA (targetReplicas)	Deployment (replicas)	HPA (minReplicas / maxReplicas)	最终结果	操作说明
场景七	minReplicas ≤ replicas ≤ maxReplicas < **targetReplicas**	11	5	5/10	HPA: 11/11 Deployment: 11	CronHPA目标实例数大于HPA最大实例数（maxReplicas）时： • 修改HPA的最小实例数。 • 修改HPA的最大实例数。 • 修改Deployment实例数。

使用控制台创建

步骤1 在CCE控制台，单击集群名称进入集群。

步骤2 单击左侧导航栏的“工作负载”，在目标工作负载的操作列中单击“更多 > 弹性伸缩”。

图 8-3 工作负载弹性伸缩



步骤3 策略类型选择“HPA+CronHPA策略”，启用HPA策略，并同时启用CronHPA策略。

此时CronHPA会定时调整HPA策略的最大和最小实例数。

步骤4 设置HPA策略，详情请参见**HPA策略**。

图 8-4 启用 HPA 策略

表 8-3 HPA 策略配置

参数	参数说明
实例范围	请输入最小实例数和最大实例数。 策略触发时，工作负载实例将在此范围内伸缩。
伸缩配置	• 系统默认：采用社区推荐的默认行为进行负载伸缩，详情请参见**社区默认行为说明**。 • 自定义：自定义扩/缩容配置的稳定窗口、步长、优先级等策略，实现更灵活的配置。未配置的参数将采用社区推荐的默认值。 – 禁止扩/缩容：选择是否禁止扩容或缩容。 – 稳定窗口：需要伸缩时，会在一段时间（设定的稳定窗口值）内持续检测，如在该时间段内始终需要进行伸缩（不满足设定的指标期望值）才进行伸缩，避免短时间的指标抖动造成异常。 – 步长策略：扩/缩容的步长，可设置一定时间内扩/缩容Pod数量或百分比。在存在多条策略时，可以选择使Pod数量最多或最少的策略。

参数	参数说明
系统策略	• 指标：可选择“CPU利用率”或“内存利用率”。 **说明** 利用率 = 工作负载容器组（Pod）的实际使用量 / 申请量 • 期望值：请输入期望资源平均利用率。 期望值表示所选指标的期望值，通过向上取整（当前指标值 / 期望值 × 当前实例数）来计算需要伸缩的实例数。 **说明** HPA在计算扩容、缩容实例数时，会选择最近5分钟内实例数的最大值。 • 容忍范围：指标处于范围内时不会触发伸缩，期望值必须在容忍范围之间。 当指标值大于缩容阈值且小于扩容阈值时，不会触发扩容或缩容。**阈值仅在1.15及以上版本的集群中支持**。

步骤5 在CronHPA的策略规则中单击+，在弹出的窗口中设置伸缩策略参数。

图 8-5 启用 CronHPA 策略

表 8-4 CronHPA 策略参数配置

参数	参数说明
目标实例数	策略触发时，将根据实际情况调整HPA策略实例数范围，详情请参见表8-2。
触发时间	可选择每天、每周、每月或每年的具体时间点。 **说明** 触发时间基于节点所在时区进行计算。
是否启用	可选择启用或关闭该策略规则。

步骤6 填写完成上述参数，单击“确定”，您可以在列表中查看添加的策略规则。重复以上步骤，您可以添加多条策略规则，但策略的触发时间不能相同。

步骤7 设置完成后，单击“创建”。

----结束

使用kubectl命令行创建

当CronHPA与HPA兼容使用时，需要将CronHPA中的scaleTargetRef字段设置为HPA策略，而HPA策略的scaleTargetRef字段设置为Deployment，这样CronHPA策略会在固定的时间调整HPA策略的实例数量上下限，即可实现工作负载定时伸缩和弹性伸缩的兼容。

步骤1 为Deployment创建HPA策略。

```
apiVersion: autoscaling/v1
kind: HorizontalPodAutoscaler
metadata:
  name: hpa-test
  namespace: default
spec:
  maxReplicas: 10              #  最大实例数
  minReplicas: 5              #  最小实例数
  scaleTargetRef:             #  关联Deployment
    apiVersion: apps/v1
    kind: Deployment
    name: nginx
  targetCPUUtilizationPercentage: 50
```

步骤2 创建CronHPA策略，并关联**步骤1**中创建的HPA策略。

```
apiVersion: autoscaling.cce.io/v2alpha1
kind: CronHorizontalPodAutoscaler
metadata:
  name: ccetest
  namespace: default
spec:
   scaleTargetRef:                # 关联HPA策略
      apiVersion: autoscaling/v1
      kind: HorizontalPodAutoscaler
      name: hpa-test
   rules:
   - ruleName: "scale-down"
     schedule: "15 * * * *"       # 指定任务运行时间与周期，参数格式请参见Cron，例如0 * * * * 或@hourly。
     targetReplicas: 1            # 目标实例数量
     disable: false
   - ruleName: "scale-up"
     schedule: "13 * * * *"
     targetReplicas: 11
     disable: false
```

表 8-5 CronHPA 关键字段说明

字段	说明
apiVersion	API版本，固定值“autoscaling.cce.io/v2alpha1”。
kind	API类型，固定值“CronHorizontalPodAutoscaler”。
metadata.name	CronHPA策略名称。
metadata.namespace	CronHPA策略所在的命名空间。

字段	说明
spec.scaleTargetRef	指定CronHPA的扩缩容对象，可配置以下字段： • apiVersion：CronHPA扩缩容对象的API版本。 • kind：CronHPA扩缩容对象的API类型。 • name：CronHPA扩缩容对象的名称。 CronHPA支持HPA策略或Deployment，具体用法请参见**使用CronHPA调整HPA伸缩范围**或**使用CronHPA直接调整Deployment实例数量**。
spec.rules	CronHPA策略规则，可添加多个规则。每个规则可配置以下字段： • ruleName：CronHPA规则名称，该名称需唯一。 • schedule：指定任务运行时间与周期，参数格式与CronTab类似，请参见**Cron**，例如0 * * * * 或@hourly。 **说明** 触发时间基于节点所在时区进行计算。 • targetReplicas：扩缩容的Pod数目。 • disable：参数值为“true”或“false”。其中“false”表示该规则生效，“true”则表示该规则不生效。

----结束

使用 CronHPA 直接调整 Deployment 实例数量

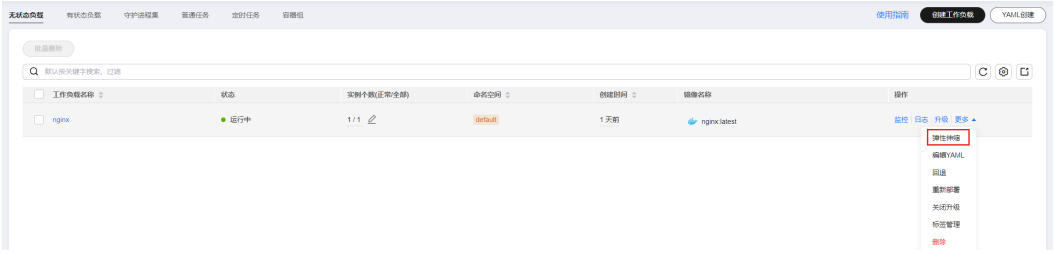
CronHPA还可以单独调整关联Deployment，定时调整Deployment的实例数，使用方法如下。

使用控制台创建

步骤1 在CCE控制台，单击集群名称进入集群。

步骤2 单击左侧导航栏的“工作负载”，在目标工作负载的操作列中单击“更多 > 弹性伸缩”。

图 8-6 工作负载弹性伸缩



步骤3 策略类型选择“HPA+CronHPA策略”，选择不启用HPA策略，并选择启用CronHPA策略。

此时CronHPA会直接定时调整工作负载的实例数。

步骤4 在CronHPA的策略规则中单击+，在弹出的窗口中设置伸缩策略参数。

图 8-7 使用 CronHPA 调整工作负载实例数

表 8-6 CronHPA 策略参数配置

参数	参数说明
目标实例数	策略触发时，工作负载实例将调整至该数值。
触发时间	可选择每天、每周、每月或每年的具体时间点。 **说明** 触发时间基于节点所在时区进行计算。
是否启用	可选择启用或关闭该策略规则。

步骤5 填写完成上述参数，单击“确定”，您可以在列表中查看添加的策略规则。重复以上步骤，您可以添加多条策略规则，但策略的触发时间不能相同。

步骤6 设置完成后，单击“创建”。

----结束

使用kubectl命令行创建

```
apiVersion: autoscaling.cce.io/v2alpha1
kind: CronHorizontalPodAutoscaler
metadata:
  name: ccetest
  namespace: default
spec:
   scaleTargetRef:           # 关联Deployment
      apiVersion: apps/v1
      kind: Deployment
      name: nginx
   rules:
   - ruleName: "scale-down"
     schedule: "08 * * * *"    # 指定任务运行时间与周期，参数格式请参见Cron，例如0 * * * * 或@hourly。
     targetReplicas: 1
```



```
    disable: false
  - ruleName: "scale-up"
    schedule: "05 * * * *"
    targetReplicas: 3
    disable: false
```

8.1.4 管理工作负载伸缩策略

操作场景

HPA策略创建完成后，可对创建的策略进行**查看编辑（控制台方式）**、**编辑（YAML方式）**、**克隆**以及**删除**等操作。

查看 HPA 策略

您可以查看HPA策略的规则、状态和事件，参照界面中的报错提示有针对性的解决异常事件。

步骤1 登录CCE控制台，单击集群名称进入集群。

步骤2 在左侧导航栏中单击“策略”，切换至“HPA策略”页签，单击要查看的HPA策略前方的 。

步骤3 在展开的区域中，可以看到规则、状态页签，单击策略操作区域的“事件”，可以看到事件页签。若策略异常，请参照界面中的报错提示进行定位处理。

说明

您还可以在工作负载详情页中查看已创建的HPA策略：

1. 登录CCE控制台，单击集群名称进入集群。
2. 在左侧导航栏中单击“工作负载”，单击工作负载名称查看详情。
3. 在该工作负载详情页的“弹性伸缩”页签下可以看到HPA策略，您在“策略”页面配置的伸缩策略也会在这里显示。

表 8-7 事件类型及名称

事件类型	事件名称	描述
正常	SuccessfulRescale	扩缩容成功
异常	InvalidTargetRange	无效的TargetRange
	InvalidSelector	无效选择器
	FailedGetObjectMetric	获取对象失败数
	FailedGetPodsMetric	获取Pod列表失败指标
	FailedGetResourceMetric	获取资源失败数
	FailedGetExternalMetric	获取外部指标失败
	InvalidMetricSourceType	无效的指标来源类型
	FailedConvertHPA	转换HPA失败
	FailedGetScale	获取比例尺失败

事件类型	事件名称	描述
	FailedComputeMetricsReplicas	计算指标副本数失败
	FailedGetScaleWindow	获取ScaleWindow失败
	FailedRescale	扩缩容失败

----结束

编辑 HPA 策略（控制台方式）

控制台的方式编辑HPA策略。

步骤1 登录CCE控制台，单击集群名称进入集群。

步骤2 在左侧导航栏中单击“策略”，切换至“HPA策略”页签，单击策略后方“操作”栏中的“更多 > 编辑”。

步骤3 在打开的“编辑HPA策略”页面中，参考**表8-1**更新策略参数。

步骤4 单击“确定”完成策略更新。

----结束

编辑 HPA 策略（YAML 方式）

步骤1 登录CCE控制台，单击集群名称进入集群。

步骤2 在左侧导航栏中单击“策略”，切换至“HPA策略”页签，单击HPA策略后方“操作”栏中的“编辑YAML”。

步骤3 在弹出的“编辑YAML”窗口中，可以对YAML进行修改和下载。

----结束

克隆 HPA 策略

复制现有HPA策略的策略配置，重新创建一个新的HPA策略，关联不同的命名空间或工作负载。

步骤1 登录CCE控制台，单击集群名称进入集群。

步骤2 在左侧导航栏中单击“策略”，切换至“HPA策略”页签，单击策略后方“操作”栏中的“更多 > 克隆”。

步骤3 在打开的“创建HPA策略”页面中，参考**步骤3**填写策略基本信息。

步骤4 单击“确定”完成创建一个新的策略。

----结束

删除 HPA 策略

步骤1 登录CCE控制台，单击集群名称进入集群。

步骤2 在左侧导航栏中单击“策略”，单击策略后方栏中的“更多 > 删除”。

步骤3 在弹出的窗口中，单击“是”完成删除操作。

----结束

9 插件

9.1 CoreDNS 域名解析

插件简介

CoreDNS域名解析插件是一款通过链式插件的方式为Kubernetes提供域名解析服务的DNS服务器。

CoreDNS是由CNCF孵化的开源软件，用于Cloud-Native环境下的DNS服务器和服务发现解决方案。CoreDNS实现了插件链式架构，能够按需组合插件，运行效率高、配置灵活。在Kubernetes集群中使用CoreDNS能够自动发现集群内的服务，并为这些服务提供域名解析。同时，通过级联云上DNS服务器，还能够为集群内的工作负载提供外部域名的解析服务。

该插件为系统资源插件，在创建集群时默认安装。

目前CoreDNS已经成为社区Kubernetes集群推荐的DNS服务器解决方案。

CoreDNS官网：**https://coredns.io/**

开源社区地址：**https://github.com/coredns/coredns**

说明

DNS详细使用方法请参见**DNS**。

约束与限制

CoreDNS域名解析插件正常运行或升级时，请确保集群中的可用节点数大于等于插件的实例数，且所有实例都处于运行状态，否则将导致插件异常或升级失败。

编辑插件

本插件为系统默认安装，若需自定义参数，可参照如下步骤进行编辑。

步骤1 登录CCE控制台，单击集群名称进入集群，在左侧导航栏中选择“插件中心”，在右侧找到**CoreDNS域名解析**插件，单击“编辑”。

步骤2 在编辑插件页面，查看“规格配置”。

表 9-1 CoreDNS 插件规格配置

参数	参数说明
实例数	插件实例的副本数量。 实例数为1时插件不具备高可用能力，当插件实例所在节点异常时可能导致插件功能无法正常使用，请谨慎选择。
容器	CoreDNS所能提供的域名解析QPS与CPU消耗成正相关，集群中的节点/容器数量增加时，CoreDNS实例承受的压力也会同步增加。

步骤3 设置插件支持的“参数配置”。

表 9-2 CoreDNS 插件参数配置

参数	参数说明
存根域设置	对自定义的域名配置域名服务器，格式为一个键值对，键为DNS后缀域名，值为一个或一组DNS IP地址，如 'acme.local --1.2.3.4,6.7.8.9'。 详情请参见**为CoreDNS配置存根域**。

参数	参数说明
高级配置	• parameterSyncStrategy：插件升级时是否配置一致性检查。 – ensureConsistent：表示启用配置一致性检查，如果集群中记录的配置和实际生效配置不一致，插件将无法升级。 – force：表示升级时忽略配置一致性检查。请您自行确保当前生效配置和原配置一致。插件升级完毕后，需恢复parameterSyncStrategy值为ensureConsistent，重新启用配置一致性检查。 – inherit：表示升级时自动继承差异配置。插件升级完毕后，需恢复parameterSyncStrategy值为ensureConsistent，重新启用配置一致性检查。 • stub_domains：存根域，您可对自定义的域名配置域名服务器，格式为一个键值对，键为DNS后缀域名，值为一个或一组DNS IP地址。 • upstream_nameservers：上游域名服务器地址。 • servers: CoreDNS 1.23.1插件版本开始开放servers配置，用户可对servers做定制化配置，详情请参见**dns-custom-nameservers**。 其中plugins为CoreDNS中各组件配置。一般场景建议保持默认配置，以免出现配置错误而导致CoreDNS整体不可用。每个plugin组件可包含"name"、"parameters"(可选)、"configBlock"(可选)配置，对应生成的Corefile配置文件中格式如下： `$name $parameters { $configBlock }` 常用plugin的说明请参见**表9-3**。更多配置详情请参见**Plugins**。 示例： `{ "servers": [ { "plugins": [ { "name": "bind", "parameters": "{$POD_IP}" }, { "name": "cache", "configBlock": "servfail 5s", "parameters": 30 }, { "name": "errors" }, { "name": "health", "parameters": "{$POD_IP}:8080" }, { "name": "ready", "{$POD_IP}:8081" }, {`...

参数	参数说明

	"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"]
	}

表 9-3 CoreDNS 主 zone 默认 plugin 配置说明

plugin名称	描述
bind	CoreDNS侦听的hostIP配置，建议保持当前默认值{$POD_IP}。详情请参见**bind**。
cache	启用DNS缓存。详情请参见**cache**。 插件版本>=1.25.10时，支持关闭servfail缓存。如需关闭servfail缓存，请将configBlock设置为"servfail 0"；否则servfail缓存的单位为s，不可省略。
errors	错误信息到标准输出。详情请参见**errors**。
health	CoreDNS健康检查配置，当前侦听{$POD_IP}:8080，请保持此默认值，否则导致coredns健康检查失败而不断重启。详情请参见**health**。

plugin名称	描述
ready	检查后端服务是否准备好接收流量，当前侦听{$POD_IP}:8081。如果后端服务没有准备好，CoreDNS将会暂停 DNS 解析服务，直到后端服务准备好为止。详情请参见**ready**。
kubernetes	CoreDNS Kubernetes插件，提供集群内服务解析能力。详情请参见**kubernetes**。
loadbalance	轮转式 DNS 负载均衡器，在应答中随机分配A、AAAA和MX记录的顺序。详情请参见**loadbalance**。
prometheus	CoreDNS自身metrics数据接口，默认zone侦听{$POD_IP}:9153，请保持此默认值，否则普罗无法采集coredns metrics数据。详情请参见**prometheus**。
forward	不在 Kubernetes 集群域内的任何查询都将转发到默认的解析器 (/etc/resolv.conf)。详情请参见**forward**。
reload	允许自动重新加载已更改的Corefile。编辑ConfigMap配置后，请等待两分钟，以使更改生效。详情请参见**reload**。

log	开启CoreDNS域名解析的日志。详情请参见**log**。
	示例如下：
	{
	"name": "log"
	}

template	设置快速应答模板，AAAA表示IPv6解析请求，rcode控制应答返回NXDOMAIN，即表示没有IPv6解析结果。详情请参见**template**。
	示例如下：
	{
	"configBlock": "rcode NXDOMAIN",
	"name": "template",
	"parameters": "ANY AAAA"
	}

步骤4 完成以上配置后，单击“确定”。

----结束

组件说明

表 9-4 coredns 组件

容器组件	说明	资源类型
coredns	该容器为提供集群域名解析服务的DNS服务器。	Deployment

Kubernetes 中的域名解析逻辑

DNS策略可以在每个pod基础上进行设置，目前，Kubernetes支持**Default、ClusterFirst、ClusterFirstWithHostNet和None**四种DNS策略，具体请参见**Service与Pod的DNS**。这些策略在pod-specific的**dnsPolicy**字段中指定。

- **“Default”**：如果dnsPolicy被设置为“Default”，则名称解析配置将从pod运行的节点继承。 自定义上游域名服务器和存根域不能够与这个策略一起使用。
- **“ClusterFirst”：**如果dnsPolicy被设置为“ClusterFirst”，任何与配置的集群域后缀不匹配的DNS查询（例如，www.kubernetes.io）将转发到从该节点继承的上游名称服务器。集群管理员可能配置了额外的存根域和上游DNS服务器。
- **“ClusterFirstWithHostNet”：**对于使用hostNetwork运行的Pod，您应该明确设置其DNS策略“ClusterFirstWithHostNet”。
- **“None”：**它允许Pod忽略Kubernetes环境中的DNS设置。应使用dnsConfigPod规范中的字段提供所有DNS设置 。

说明

- Kubernetes 1.10及以上版本，支持Default、ClusterFirst、ClusterFirstWithHostNet和None四种策略；低于Kubernetes 1.10版本，仅支持default、ClusterFirst和ClusterFirstWithHostNet三种。
- “Default”不是默认的DNS策略。如果dnsPolicy的Flag没有特别指明，则默认使用“**ClusterFirst**”。

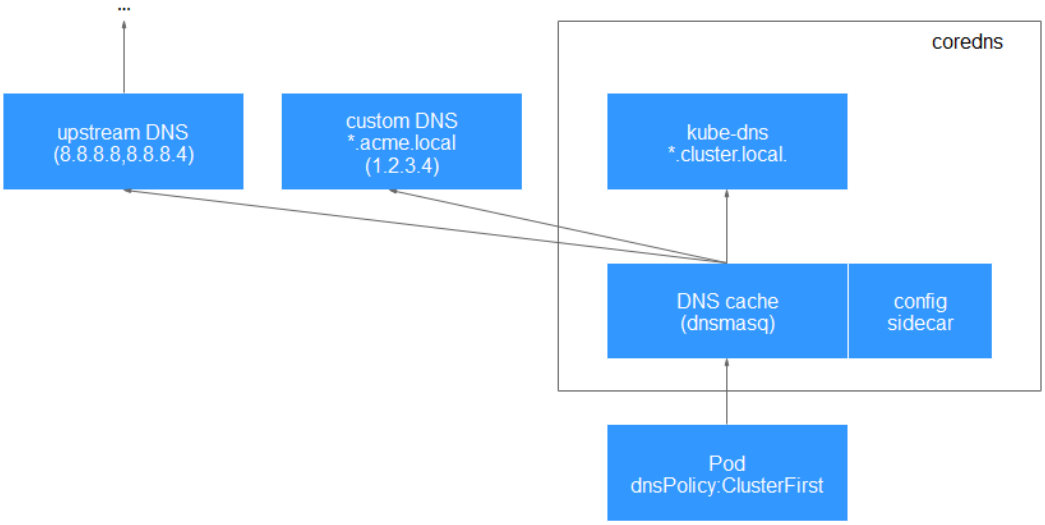
路由请求流程：

未配置存根域：没有匹配上配置的集群域名后缀的任何请求，例如“www.kubernetes.io”，将会被转发到继承自节点的上游域名服务器。

已配置存根域：如果配置了存根域和上游DNS服务器，DNS查询将基于下面的流程对请求进行路由：

1. 查询首先被发送到coredns中的DNS缓存层。
2. 从缓存层，检查请求的后缀，并根据下面的情况转发到对应的DNS上：
   - 具有集群后缀的名字（例如“.cluster.local”）：请求被发送到coredns。
   - 具有存根域后缀的名字（例如“.acme.local”）：请求被发送到配置的自定义DNS解析器（例如：监听在 1.2.3.4）。
   - 未能匹配上后缀的名字（例如“widget.com”）：请求被转发到上游DNS。

图 9-1 路由请求流程



9.2 CCE 容器存储插件（Everest）

插件简介

CCE容器存储插件（Everest）是一个云原生容器存储系统，基于CSI（即Container Storage Interface）为Kubernetes集群对接云存储服务的能力。

说明

v1.27.5-r0、v1.28.3-r0及以上版本的集群中，该插件由系统自动配置，无需手动安装或更新。

编辑插件

本插件为系统默认安装，若需自定义参数，可参照如下步骤进行编辑。

步骤1 登录CCE控制台，单击集群名称进入集群，单击左侧导航栏的“插件中心”，在右侧找到**CCE容器存储插件（Everest）**插件，单击“编辑”。

步骤2 在编辑插件页面，查看“规格配置”。

表 9-5 everest 插件规格配置

参数	参数说明
实例数	插件实例的副本数量。 实例数为1时插件不具备高可用能力，当插件实例所在节点异常时可能导致插件功能无法正常使用，请谨慎选择。
容器	CCE容器存储插件（Everest）包含everest-csi-controller和everest-csi-driver两个组件，详情请参见**组件说明**。

步骤3 设置插件支持的“参数配置”。

表 9-6 everest 插件参数配置

参数	参数说明
csi_attacher_worker_threads	CCE容器存储插件（Everest）中同时处理挂EVS卷的worker数，默认值为“60”。
csi_attacher_detach_worker_threads	CCE容器存储插件（Everest）中同时处理卸载EVS卷的worker数，默认值均为“60”。
volume_attaching_flow_ctrl	CCE容器存储插件（Everest）在1分钟内可以挂载EVS卷的最大数量，此参数的默认值“0”表示everest插件不做挂卷限制，此时挂卷性能由底层存储资源决定。
cluster_id	集群ID。
default_vpc_id	集群所在VPC的ID。
disable_auto_mount_secret	挂载对象桶/并行文件系统时，是否允许使用默认的AKSK，默认为false。
enable_node_attacher	是否开启agent侧attacher，开启后由attacher负责处理**VolumeAttachment**。
flow_control	默认为空。用户无需填写。
over_subscription	本地存储池（local_storage）的超分比。默认为80，若本地存储池为100GiB，可以超分为180GiB使用。
project_id	集群所属项目ID。

说明

CCE容器存储插件（Everest）针对大批量挂EVS卷的性能做了优化，用户可配置如下3个参数：

- csi_attacher_worker_threads
- csi_attacher_detach_worker_threads
- volume_attaching_flow_ctrl

上述三个参数由于存在关联性且与集群所在局点的底层存储资源限制有关，当您对大批量挂卷的性能有要求（大于500EVS卷/分钟）时，请联系客服，在指导下进行配置，否则可能会因为参数配置不合理导致出现everest插件运行不正常的情况。

步骤4 单击“确定”。

----结束

组件说明

表 9-7 everest 组件

容器组件	说明	资源类型
everest-csi-controller	此容器负责存储卷的创建、删除、快照、扩容、attach/detach等功能。	Deployment

容器组件	说明	资源类型
everest-csi-driver	此容器负责PV的挂载、卸载、文件系统resize等功能。	Daemon Set

9.3 Kubernetes Metrics Server

Kubernetes通过Metrics API提供资源使用指标，例如容器CPU和内存使用率。这些度量可以由用户直接访问（例如，通过使用kubectl top命令），或者由集群中的控制器（例如，Horizontal Pod Autoscaler）使用来进行决策。

Metrics Server是集群核心资源监控数据的聚合器，您可以在CCE控制台快速安装本插件。

安装本插件后，可创建HPA策略，具体请参见**HPA策略**。

社区官方项目及文档：**https://github.com/kubernetes-sigs/metrics-server**。

编辑插件

步骤1 登录CCE控制台，单击集群名称进入集群，单击左侧导航栏的“插件中心”，在右侧找到**Kubernetes Metrics Server**插件，单击“编辑”。

步骤2 在编辑插件页面，设置“规格配置”。

表 9-8 metrics-server 插件规格配置

参数	参数说明
实例数	显示插件中的实例数。
容器	显示插件容器的CPU和内存配额。

步骤3 单击“确定”。

----结束

组件说明

表 9-9 metrics-server 组件

容器组件	说明	资源类型
metrics-server	集群核心资源监控数据的聚合器，用于收集和聚合集群中通过Metrics API提供的资源使用指标。	Deployment

9.4 云原生监控插件

插件简介

云原生监控插件（kube-prometheus-stack）通过使用Prometheus-operator和Prometheus，提供简单易用的端到端Kubernetes集群监控能力。

使用kube-prometheus-stack可将监控数据与监控中心对接，在监控中心控制台查看监控数据，配置告警等。

开源社区地址：**https://github.com/prometheus/prometheus**

安装插件

步骤1 登录CCE控制台，单击集群名称进入集群，在左侧导航栏中选择“插件中心”，在右侧找到**云原生监控插件**，单击“安装”。

步骤2 “数据存储配置”开启“监控数据上报至AOM服务”，并选择“指标上报的AOM实例”。

监控数据上报至AOM服务：将普罗数据上报至 AOM 服务。开启后，可选择对应的AOM实例。采集的基础指标免费，自定义指标将由AOM服务进行收费，详情请参见**价格详情**。对接AOM需要用户具备一定权限，目前仅**华为云/华为账号**，或者在**admin**用户组下的用户支持此操作。

步骤3 根据需求进行“规格配置”。

- **普罗高可用：**高可用会在集群中将prometheus-lightweight、prometheus-operator、custom-metrics-apiserver、kube-state-metrics组件按多实例方式部署。
- **配置清单：**您可以根据需求为各组件配置CPU配额和内存配额。

步骤4 根据需求进行“参数配置”。

- **自定义指标HPA**：以服务发现的形式自动采集应用的指标用于HPA。开启后需要在目标应用添加相关配置，详情请参见**使用自定义指标创建HPA策略**。
- **采集周期：**采集时间间隔，输入值必须在10到60之间。

步骤5 完成以上配置后，单击“安装”。

插件安装完成后，根据您的使用需求，可能还需进行以下操作。

如需使用自定义指标创建弹性伸缩策略，请将Prometheus采集到的自定义指标聚合到API Server，可供HPA策略使用，详情请参见**使用自定义指标创建HPA策略**。

----结束

组件说明

安装kube-prometheus-stack插件创建的Kubernetes资源，全部都创建在monitoring命名空间下。

表 9-10 kube-prometheus-stack 组件

容器组件	说明	资源类型
prometheusOperator（负载名称：prometheus-operator）	根据自定义资源（Custom Resource Definition / CRDs）来部署和管理Prometheus Server，同时监控这些自定义资源事件的变化来做相应的处理，是整个系统的控制中心。	Deployment
prometheus-lightweight（负载名称：prometheus-lightweight）	Operator根据自定义资源Prometheus类型中定义的内容而部署Prometheus Server集群，这些自定义资源可以看作是用来管理Prometheus Server集群的StatefulSets资源。	StatefulSet
kubeStateMetrics（负载名称：kube-state-metrics）	将Prometheus的metrics数据格式转换成K8s API接口能识别的格式。 **说明** 该组件如果存在多个Pod，只会有一个Pod暴露指标。	Deployment
adapter（负载名称：custom-metrics-apiserver）	将自定义指标聚合到原生的Kubernetes API Server，与“自定义指标HPA”密切相关。仅当“自定义指标HPA”被启用时，需要安装adapter组件。	Deployment

通过 Metrics API 提供资源指标

容器和节点的资源指标，如CPU、内存使用量，可通过Kubernetes的Metrics API获得。这些指标可以直接被用户访问，比如用kubectl top命令，也可以被HPA或者CustomedHPA使用，根据资源使用率使负载弹性伸缩。

插件可为Kubernetes提供Metrics API，但默认未开启，若要将其开启，需要创建以下APIService对象：

```
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
```

可以将该对象保存为文件，命名为metrics-apiservice.yaml，然后执行以下命令：

```
kubectl create -f metrics-apiservice.yaml
```

执行kubectl top pod -n monitoring命令，若显示如下，则表示Metrics API能正常访问：


```
# kubectl top pod -n monitoring
NAME                                          CPU(cores)   MEMORY(bytes)
......
custom-metrics-apiserver-d4f556ff9-l2j2m      38m          44Mi
......
```

须知

卸载插件时，需要执行以下kubectl命令，同时删除APIService对象，否则残留的APIService资源将导致metrics-server插件安装失败。

```
kubectl delete APIService v1beta1.metrics.k8s.io
```

使用自定义指标创建 HPA 策略

步骤1 登录CCE控制台，单击集群名称进入集群。

步骤2 自定义采集规则，采集到的指标可以作为HPA策略的监控指标，用于控制工作负载弹性伸缩。

- 如果需要创建一个新的配置项，用于自定义采集规则，可以按照以下步骤进行。

  在左侧导航栏中选择“配置与密钥”，单击“YAML创建”，自定义采集规则示例如下。

  如果您需要增加多个采集规则，可在rules字段下添加多个配置，关于采集规则配置详情请参见**Metrics Discovery and Presentation Configuration**。

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: user-adapter-config  # 配置项名称，不能修改
  namespace: monitoring  # 配置项所在的命名空间
data:
  config.yaml: |-
    rules:  # 采集规则
    - seriesQuery: 'container_network_receive_bytes_total{namespace!="",pod!=""}' # 扩容需要的原始指标（指标来源kubelet）
      seriesFilters: []
      resources:
        overrides:  # 指定Pod和命名空间资源
          namespace:
            resource: namespace
          pod:
            resource: pod
      name:
        matches: container_(.*)_total  #匹配以container_开头并以_total结尾的指标。
        as: "pod_${1}_per_second"  # 采集指标的别名
      metricsQuery: sum(rate(<<.Series>>{<<.LabelMatchers>>}[30s])) by (<<.GroupBy>>) # 负载下所有容器的30s内指标变化率
```

  说明

  以上示例中，聚合时间为30s，如果该参数设置小于15s采集周期，可能会导致指标不准确。

- 如果已有一个配置项，可以通过以下步骤对其采集规则进行修改。

  在左侧导航栏中选择“配置与密钥”，切换至“monitoring”命名空间，在“配置项”页签找到对应的配置项，并单击“更新”。

  在“更新配置项”页面，单击“配置数据 > 编辑”，进而修改采集规则。

图 9-2 更新配置项

步骤3 在左侧导航栏中选择“插件中心”，在右侧找到**云原生监控插件**，单击“编辑”，并开启**“自定义指标HPA”**功能。

完成后，单击“确定”。

说明

您需要先创建**步骤2**中的自定义采集规则，然后打开**“自定义指标HPA”**功能触发插件重新部署，自定义指标采集才可以生效。

步骤4 在左侧导航栏中选择“工作负载”，找到需要创建HPA策略的工作负载单击“弹性伸缩”。您可在“自定义策略”中选择**步骤2**中的采集指标创建弹性伸缩策略。

说明

当负载创建后，请您等待Pod ready并完成第一个采集周期的指标采集后，再创建HPA策略。

图 9-3 创建 HPA 策略

----结束

9.5 云原生日志采集插件

插件简介

云原生日志采集插件（log-agent）是基于开源fluent-bit和opentelemetry构建的云原生日志、K8s事件采集插件。log-agent安装后，会自动创建fluent-bit容器组件，用于日志采集。此外，log-agent支持基于CRD的日志采集策略，可以根据您配置的策略规则，对集群中的容器标准输出日志、容器文件日志及K8s事件日志进行采集与转发。同时支持上报K8s事件到AOM，用于配置事件告警，默认上报所有异常事件和部分正常事件。采集日志的详细使用方法请参见**收集容器日志**。

说明

fluent-bit容器组件由华为云免费提供，不收取任何费用。

约束与限制

云原生日志采集插件有如下限制：

- 每个集群限制50条日志规则。
- 不采集.gz、.tar、.zip后缀类型的日志文件。
- 若容器运行时为containerd模式，容器标准输出日志中的多行配置暂不生效。
- 每个集群限制单行日志采集速率不超过10000条/秒，多行日志不超过2000条/秒。

权限说明

云原生日志采集插件中的fluent-bit组件会根据用户的采集配置，读取容器标准输出、容器内文件日志并采集。

fluent-bit组件运行需要以下权限：

- CAP_DAC_OVERRIDE：忽略文件的 DAC 访问限制。
- CAP_FOWNER：忽略文件属主 ID 必须和进程用户 ID 相匹配的限制。
- DAC_READ_SEARCH：忽略文件读及目录搜索的 DAC 访问限制。
- SYS_PTRACE：允许跟踪任何进程。

安装插件

步骤1 登录CCE控制台，单击集群名称进入集群，在左侧导航栏中选择“插件中心”，在右侧找到**云原生日志采集插件**，单击“安装”。

步骤2 在安装插件页面，设置“规格配置”。

表 9-11 插件规格配置

参数	参数说明
实例数	选择上方插件规格后，显示插件中的实例数。 选择“自定义”规格时，您可根据需求调整插件实例数。
容器	log-agent插件还包含以下容器组件，您可根据需求自定义调整规格： ● log-operator：负责解析及更新日志规则的组件。 ● otel-collector：负责集中式日志转发的组件，将fluent-bit收集的日志转发到LTS。

步骤3 完成以上配置后，单击“安装”。

----结束

组件说明

表 9-12 log-agent 组件

容器组件	说明	资源类型
fluent-bit	轻量级的日志收集器和转发器，用于采集日志。	Pod
log-operator	负责生成内部的配置文件。	Deployment
otel-collector	负责收集来自不同应用程序和服务的日志数据，集中后上报至LTS。	Deployment

9.6 NGINX Ingress 控制器

插件简介

Kubernetes通过kube-proxy服务实现了Service的对外发布及负载均衡，它的各种方式都是基于传输层实现的。在实际的互联网应用场景中，不仅要实现单纯的转发，还有更加细致的策略需求，如果使用真正的负载均衡器更会增加操作的灵活性和转发性能。

基于以上需求，Kubernetes引入了资源对象Ingress，Ingress为Service提供了可直接被集群外部访问的虚拟主机、负载均衡、SSL代理、HTTP路由等应用层转发功能。

Kubernetes官方发布了基于Nginx的Ingress控制器，CCE的NGINX Ingress控制器插件直接使用社区模板与镜像。Nginx Ingress控制器会将Ingress生成一段Nginx的配置，并将Nginx配置通过ConfigMap进行储存，这个配置会通过Kubernetes API写到Nginx的Pod中，然后完成Nginx的配置修改和更新，详细工作原理请参见**工作原理**。

开源社区地址：**https://github.com/kubernetes/ingress-nginx**

说明

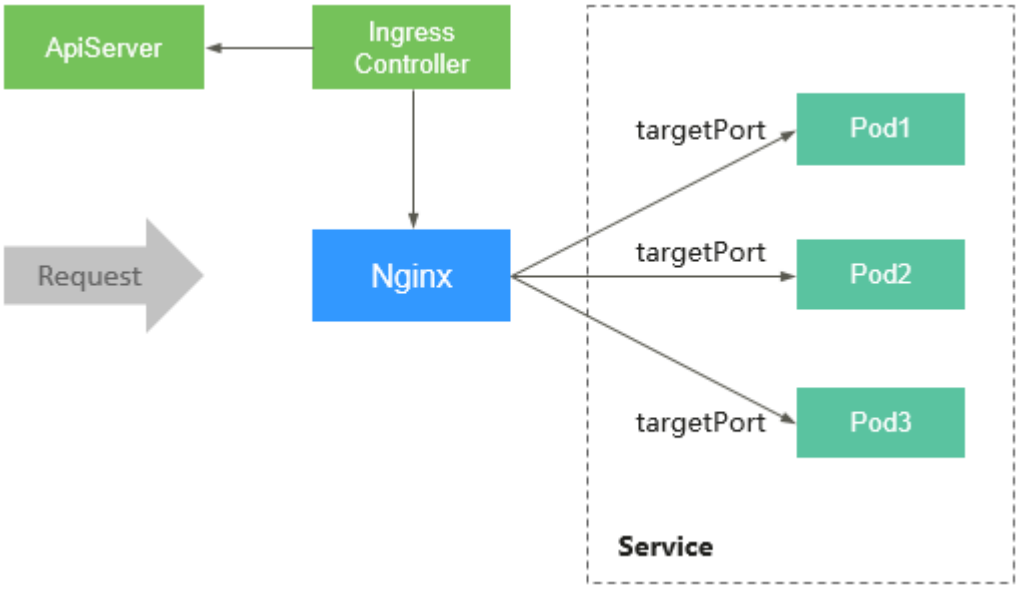
- 安装该插件时，您可以通过“nginx配置参数”添加配置，此处的设置将会全局生效，该参数直接通过配置nginx.conf生成，将影响管理的全部Ingress，相关参数可通过**ConfigMaps**查找，如果您配置的参数不包含在**ConfigMaps**所列出的选项中将不会生效。
- 安装该插件后，您在CCE控制台**创建Ingress**时可以选择对接Nginx控制器，并通过“注解”设置Nginx Ingress功能，支持的注解字段详情请参见**Annotations**。
- 请勿手动修改和删除CCE自动创建的ELB和监听器，否则将出现工作负载异常；若您已经误修改或删除，请卸载Nginx Ingress插件后重装。

工作原理

Nginx Ingress由资源对象Ingress、Ingress控制器、Nginx三部分组成，Ingress控制器用以将Ingress资源实例组装成Nginx配置文件（nginx.conf），并重新加载Nginx使变更的配置生效。当它监听到Service中Pod变化时通过动态变更的方式实现Nginx上游服务器组配置的变更，无须重新加载Nginx进程。工作原理如**图9-4**所示。

- Ingress：一组基于域名或URL把请求转发到指定Service实例的访问规则，是Kubernetes的一种资源对象，Ingress实例被存储在对象存储服务etcd中，通过接口服务被实现增、删、改、查的操作。
- Ingress控制器（Ingress Controller）：用以实时监控资源对象Ingress、Service、End-point、Secret（主要是TLS证书和Key）、Node、ConfigMap的变化，自动对Nginx进行相应的操作。
- Nginx：实现具体的应用层负载均衡及访问控制。

图 9-4 Nginx Ingress 工作原理



使用约束

- 独享型ELB规格必须支持网络型（TCP/UDP），且网络类型必须支持私网（有私有IP地址）。
- nginx-ingress在升级时会预留10s的宽限时间，用于删除ELB后端的nginx-ingress控制器。
- nginx-ingress-controller的优雅退出时间为300s，若nginx-ingress升级时存在超过300s的长连接，长连接会被断开，出现服务短暂中断。

前提条件

在创建容器工作负载前，您需要存在一个可用集群。若没有可用集群，请参照**购买CCE Autopilot集群**中的步骤创建。

安装插件

步骤1 登录CCE控制台，单击集群名称进入集群，在左侧导航栏中选择“插件中心”，在右侧找到**NGINX Ingress控制器**插件，单击“安装”。

步骤2 在安装插件页面，设置“规格配置”。

表 9-13 nginx-ingress 插件规格配置

参数	参数说明
实例数	您可根据需求调整插件实例数。
容器	您可根据需求调整插件实例的容器规格。

步骤3 设置插件支持的“参数配置”。

- **负载均衡器**：支持对接独享型负载均衡实例，如果无可用实例，请先创建。负载均衡器需要拥有至少两个监听器配额，且端口 80 和 443 没有被监听器占用。

- **开启准入校验**：针对Ingress资源的准入控制，以确保控制器能够生成有效的配置。开启后将会对Nginx类型的Ingress资源配置做准入校验，若校验失败，请求将被拦截。关于准入校验详情，请参见**准入控制**。

  说明

  - 开启准入校验后，会在一定程度上影响Ingress资源的请求响应速度。
  - 仅2.4.1及以上版本的插件支持开启准入校验。

- **nginx配置参数**：配置nginx.conf文件，将影响管理的全部Ingress，相关参数可通过**ConfigMaps**查找，如果您配置的参数不包含在**ConfigMaps**所列出的选项中将不会生效。

  此处以设置keep-alive-requests参数为例，设置保持活动连接的最大请求数为100。

  ```
  {
    "keep-alive-requests": "100"
  }
  ```

- **默认404服务**：默认使用插件自带的404服务。支持自定义404服务，填写“命名空间/服务名称”，如果服务不存在，插件会安装失败。

步骤4 单击“安装”。

----结束

组件说明

表 9-14 nginx-ingress 组件

容器组件	说明	资源类型
cceaddon-nginx-ingress-controller	基于Nginx的Ingress控制器，为集群提供灵活的路由转发能力。	Deployment
cceaddon-nginx-ingress-default-backend	Nginx的默认后端。返回“default backend - 404”。	Deployment
cceaddon-nginx-ingress-admission-create	开启webhook功能后，创建证书用于webhook验证。	Job
cceaddon-nginx-ingress-admission-patch	开启webhook功能后，将创建出的证书更新到webhook配置中。	Job

9.7 CCE 容器弹性引擎

CCE容器弹性引擎（cce-hpa-controller）插件是一款CCE自研的插件，能够基于CPU利用率、内存利用率等指标，对无状态工作负载进行弹性扩缩容。

主要功能

- 支持按照当前实例数的百分比进行扩缩容。

- 支持设置一次扩缩容的最小步长。
- 支持按照实际指标值执行不同的扩缩容动作。

约束与限制

cce-hpa-controller需要安装能够提供Metrics API的插件，您可根据集群版本和实际需求选择其中之一：

- **Kubernetes Metrics Server**：提供基础资源使用指标，例如容器CPU和内存使用率。所有集群版本均可安装。
- **云原生监控插件**：根据自定义指标进行弹性伸缩需要将自定义指标聚合到Kubernetes API Server，详情请参见**使用自定义指标创建HPA策略**。

安装插件

步骤1 登录CCE控制台，单击集群名称进入集群，单击左侧导航栏的“插件中心”，在右侧找到**CCE容器弹性引擎**插件，单击“安装”。

步骤2 在安装插件页面，设置“规格配置”。

表 9-15 cce-hpa-controller 插件规格配置

参数	参数说明
实例数	插件实例的副本数量。 实例数为1时插件不具备高可用能力，当插件实例异常时可能导致插件功能无法正常使用，请谨慎选择。
容器	选择插件规格后，显示插件容器的CPU和内存配额。 您可根据需求调整插件实例的容器规格。

步骤3 单击“安装”。

----结束

组件说明

表 9-16 cce-hpa-controller 组件

容器组件	说明	资源类型
customedhpa-controller	CCE自研的弹性伸缩组件，可基于CPU利用率、内存利用率等指标，对无状态工作负载进行弹性扩缩容。	Deployment

10 模板（Helm Chart）

10.1 使用模板时的 API 资源限制

资源	限制项	说明	推荐替代方案
namespaces	-	支持	为安全起见，Autopilot 不允许在系统管理的命名空间（如 kube-system）中部署工作负载，不可进行任何资源的创建、修改、删除、exec等。
nodes	-	支持	只支持查询，不支持增删改功能
persistentvolumeclaims	-	支持	-
persistentvolumes	-	支持	-
pods	HostPath	限制挂载本地宿主机文件到容器中	使用emptyDir或云存储
	HostNetwork	限制将宿主机端口映射到容器上	使用type=LoadBalancer的负载均衡
	HostPID	限制容器可见宿主机PID空间	用户不感知节点，无需使用
	HostIPC	限制容器进程和宿主机进程间通信	用户不感知节点，无需使用
	NodeName	限制Pod调度到特定节点	用户不感知节点，无需使用
	特权容器	不支持	-

资源	限制项	说明	推荐替代方案
	Linux capabilities	支持"SETPCAP", "MKNOD", "AUDIT_WRITE", "CHOWN", "DAC_OVERRIDE", "FOWNER", "FSETID", "KILL", "SETGID", "SETUID", "NET_BIND_SERVICE", "SYS_CHROOT", "SETFCAP", "SYS_PTRACE" 可以通过SecurityContext设置开启NET_RAW 、SYS_PTRACE 、NET_ADMIN	使用允许值
	调度的节点亲和与打散规则	限制将Pod调度到指定节点或者具有某些标签的节点上，或者将一批Pod打散到具有某些标签的节点上。 Autopilot集群中应用指定节点亲和性或nodeSelector字段不生效。	• 无需指定节点调度，但可以指定Pod到某一个可用区 • 可以将一批Pod打散到多个可用区
	Pod间亲和与反亲和配置	不生效	无需配置
	allowPrivilegeEscalation是否允许特权升级	不支持	使用默认配置
	RuntimeClassName	无需配置，上层应用（Pod除外)指定RuntimeClassName时后端将自动修改为系统默认支持的runc	无需配置
	时区同步(会挂载主机/etc/localtime)	不支持	使用默认配置
serviceaccounts	-	不支持修改系统配置、不允许绑定系统角色	使用默认配置
services	-	限制nodeport，ELB只支持独享型	使用type=LoadBalancer的负载均衡

资源	限制项	说明	推荐替代方案
daemonsets	apps	限制使用Daemonset类workload	通过Sidecar形式在Pod中部署多个镜像
deployments	apps	支持，其中限制使用的字段与Pod一致	使用允许值
replicasets	apps	支持，其中限制使用的字段与Pod一致	使用允许值
statefulsets	apps	支持，其中限制使用的字段与Pod一致	使用允许值
cronjobs	batch	支持，其中限制使用的字段与Pod一致	使用允许值
jobs	batch	支持，其中限制使用的字段与Pod一致	使用允许值
clusterrolebindings	rbac.authorization.k8s.io	支持，限制绑定系统组与系统用户，cce-service组	使用允许值
rolebindings	rbac.authorization.k8s.io	支持，限制绑定系统组与系统用户，cce-service组	使用允许值
storageclasses	storage.k8s.io	不支持创建obs、evs类型的storageclass；其他功能支持	使用允许值

10.2 通过模板部署应用

在CCE控制台上，您可以上传Helm模板包，然后在控制台安装部署，并对部署的实例进行管理。

约束与限制

- 单个用户可以上传模板的个数有限制，请以各个Region控制台界面中提示的实际值为准。
- CCE使用的Helm版本为v3.8.2，支持上传Helm v3版本语法的模板包。
- 模板若存在多个版本，则消耗对应数量的模板配额。
- 由于模板的操作权限同时具有较高的集群操作权限，因此租户应当谨慎授予用户对于模板生命周期管理的权限，包括上传模板的权限，以及创建、删除和更新模板实例的权限。

模板包规范

以下以redis为例，在准备redis模板包时根据模板包规范制作模板包。

- **命名要求**

模板包命名格式为：**{name}-{version}**.tgz，其中**{version}**为版本号，格式为“主版本号.次版本号.修订号”，如redis-0.4.2.tgz。

说明

模板名称{name}的长度不能超过64个字符。

版本号需遵循**语义化版本**规则。

- 主版本号、次版本号为必选，修订号为可选。
- 主版本号、次版本号、修订号的数值为整数，均需要≥0，且≤99。

- **目录结构**

  模板包的目录结构如下所示：

```
redis/
  templates/
  values.yaml
  README.md
  Chart.yaml
  .helmignore
```

目录说明如**表10-1**所示，带*的为必选项：

表 10-1 模板包目录说明

参数	参数说明
* templates	用于存放所有的template（模板）文件。
* values.yaml	用于描述template文件所需的配置参数。 **须知** 定义template文件配置参数时，请注意此处定义的“镜像地址”务必和容器镜像仓库中对应的镜像地址保持一致。否则创建工作负载会异常，提示镜像拉取失败。 镜像地址获取方法如下：在CCE控制台，单击左侧导航栏的“镜像仓库”，进入容器镜像服务控制台。在“我的镜像 > 自有镜像”中，单击已上传镜像的名称，在“镜像版本 ”页签的“下载指令”栏中即可获取镜像地址，单击 按钮即可复制该指令。
README.md	一个markdown文件，包括： • 描述Chart提供的工作负载或服务。 • 运行Chart的前提。 • 解释values.yaml文件中的配置。 • 安装和配置Chart的相关信息。
* Chart.yaml	模板的基本信息说明。 注：Helm v3版本apiVersion从v1切换到了v2。
.helmignore	设定在工作负载安装时不需要读取templates的某些文件或数据。

上传模板

步骤1 登录CCE控制台，单击集群名称进入集群，在左侧导航栏中选择“应用模板”，在右上角单击“上传模板”。

步骤2 单击“添加文件”，选中待上传的工作负载包后，单击“上传”。

图 10-1 上传模板包

----结束

创建模板实例

步骤1 登录CCE控制台，单击集群名称进入集群，在左侧导航栏中选择“应用模板”。

步骤2 在“我的模板”页签中，单击目标模板下的“安装”。

步骤3 参照表10-2设置安装工作负载参数。

表 10-2 安装工作负载参数说明

参数	参数说明
实例名称	新建模板实例名称，命名必须唯一。
命名空间	指定部署的命名空间。
选择版本	选择模板的版本。
配置文件	用户可以导入values.yaml文件，导入后可替换模板包中的values.yaml文件；也可直接在配置框中在线编辑模板参数。 **说明** 此处导入的values.yaml文件需符合yaml规范，即KEY:VALUE格式。对于文件中的字段不做任何限制。 导入的value.yaml的key值必须与所选的模板包的values.yaml保持一致，否则不会生效。即key不能修改。 1. 单击“添加文件”。 2. 选择对应的values.yaml文件，单击“打开”。

步骤4 配置完成后，单击“安装”。

在“模板实例”页签下可以查看模板实例的安装情况。

----结束

升级模板工作负载

步骤1 登录CCE控制台，单击集群名称进入集群，在左侧导航栏中选择“应用模板”，在右侧选择“模板实例”页签。

步骤2 单击待升级工作负载后的“升级”，设置升级模板工作负载的参数。

步骤3 选择对应的模板版本。

步骤4 参照界面提示修改模板参数。单击“升级”，再单击“提交”。

步骤5 单击“返回模板实例列表”，模板状态为“升级成功”时，表明工作负载升级成功。

----结束

回退模板工作负载

步骤1 登录CCE控制台，单击集群名称进入集群，在左侧导航栏中选择“应用模板”，在右侧选择“模板实例”页签。

步骤2 单击待回退工作负载后的“回退”，选择要回退的工作负载版本，单击“回退”。

模板工作负载列表中，状态为“回退成功”时，表明工作负载回退成功。

----结束

卸载模板工作负载

步骤1 登录CCE控制台，单击集群名称进入集群，在左侧导航栏中选择“应用模板”，在右侧选择“模板实例”页签。

步骤2 单击待卸载模板实例后的“更多 > 卸载”，确认待卸载模板实例后，单击“是”。模板实例卸载后不能恢复，请谨慎操作。

----结束

11 权限

11.1 CCE Autopilot 集群权限概述

CCE权限管理是在统一身份认证服务（IAM）与Kubernetes的角色访问控制（RBAC）的能力基础上，打造的细粒度权限管理功能，支持基于IAM的细粒度权限控制和IAM Token认证，支持集群级别、命名空间级别的权限控制，帮助用户便捷灵活的对租户下的IAM用户、用户组设定不同的操作权限。

如果您需要对CCE集群及相关资源进行精细的权限管理，例如限制不同部门的员工拥有部门内资源的细粒度权限，您可以使用CCE权限管理提供的增强能力进行多维度的权限管理。

本章节将介绍CCE权限管理机制及其涉及到的基本概念。如果当前账号已经能满足您的要求，您可以跳过本章节，不影响您使用CCE服务的其它功能。

说明

CCE权限管理相关策略适用于所有CCE产品，包括CCE Standard/Turbo/Autopilot集群。CCE Autopilot集群产品形态导致的差异，将特殊标注说明。

CCE 支持的权限管理能力

CCE的权限管理包括“集群权限”和“命名空间权限”两种能力，能够从集群和命名空间层面对用户组或用户进行细粒度授权，具体解释如下：

- **集群权限：**是基于IAM系统策略的授权，可以通过用户组功能实现IAM用户的授权。用户组是用户的集合，通过集群权限设置可以让某些用户组操作集群（如创建/删除集群、节点、节点池、模板、插件等），而让某些用户组仅能查看集群。

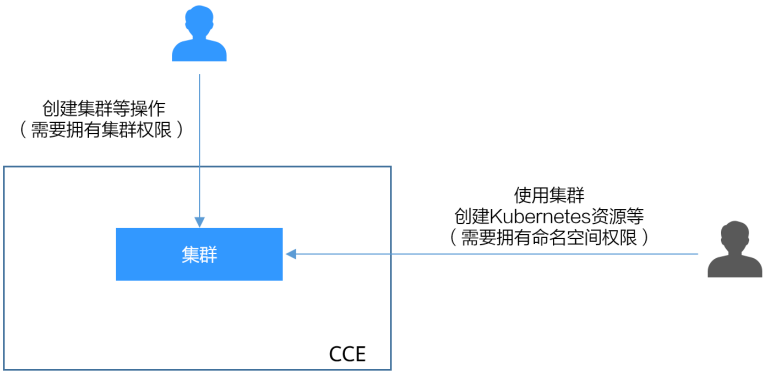
  集群权限涉及CCE非Kubernetes API，支持IAM细粒度策略和企业项目管理相关能力。

- **命名空间权限：**是基于Kubernetes RBAC（Role-Based Access Control，基于角色的访问控制）能力的授权，通过权限设置可以让不同的用户或用户组拥有操作不同Kubernetes资源的权限。同时CCE基于开源能力进行了增强，可以支持基于IAM用户或用户组粒度进行RBAC授权、IAM token直接访问API进行RBAC认证鉴权。

  命名空间权限涉及CCE和Kubernetes API，基于Kubernetes RBAC能力进行增强，支持对接IAM用户/用户组进行授权和认证鉴权，但与IAM细粒度策略独立。

CCE的权限可以从使用的阶段分为两个阶段来看，第一个阶段是创建和管理集群的权限，也就是拥有创建/删除集群、节点等资源的权限。第二个阶段是使用集群Kubernetes资源（如工作负载、Service等）的权限。

图 11-1 权限示例图



清楚了集群权限和命名空间权限后，您就可以通过这两步授权，做到精细化的权限控制。

集群权限（IAM 授权）与命名空间权限（Kubernetes RBAC 授权）的关系

拥有不同集群权限（IAM授权）的用户，其拥有的命名空间权限（Kubernetes RBAC授权）也不同。**表11-1**给出了不同用户拥有的命名空间权限详情。

表 11-1 不同用户拥有的命名空间权限

用户类型	CCE Standard/Turbo/Autopilot集群（Standard/Turbo集群版本要求1.13及以上）
拥有Tenant Administrator权限的用户（例如账号）	全部命名空间权限
拥有CCE Administrator权限的IAM用户	全部命名空间权限
拥有CCE FullAccess或者CCE ReadOnlyAccess权限的IAM用户	按Kubernetes RBAC授权
拥有Tenant Guest权限的IAM用户	按Kubernetes RBAC授权

kubectl 权限说明

您可以通过**kubectl访问集群**的Kubernetes资源，那kubectl拥有哪些Kubernetes资源的权限呢?

kubectl访问CCE集群是通过集群上生成的配置文件（kubeconfig.json）进行认证，kubeconfig.json文件内包含用户信息，CCE根据用户信息的权限判断kubectl有权限访问哪些Kubernetes资源。即哪个用户获取的kubeconfig.json文件，kubeconfig.json就拥有哪个用户的信息，这样使用kubectl访问时就拥有这个用户的权限。而用户拥有的权限就是**表11-1**所示的权限。

联邦用户支持说明

IAM支持基于SAML、OIDC协议的单点登录，如果您已经有自己的企业管理系统，同时您的用户需要使用您账号内的云服务资源，您可以使用IAM的身份提供商功能，实现用户使用企业管理系统账号单点登录，这一过程称之为联邦身份认证。

通过联邦身份认证访问的用户称为联邦用户，联邦用户相当于IAM用户。

联邦用户使用CCE时需要注意如下两点。

- 用户创建CCE集群时，会在集群中默认为该用户创建一个cluster-admin权限（管理员权限），联邦用户由于每次登录注销都会改变用户ID，所以在CCE控制台权限管理处，权限用户会显示已删除，请勿删除该权限，否则会导致鉴权失败。此种情况下建议在CCE为某个用户组创建cluster-admin权限，将联邦用户加入此用户组。
- 联邦用户不支持创建永久访问密钥AK/SK，在需要使用AK/SK的场景（如创建OBS类型PV/PVC时），只能由账号或是实体IAM用户创建密钥，共享给联邦用户。由于密钥表示用户所拥有的权限，因此建议由与联邦用户同在一个用户组的实体IAM用户创建并分享密钥。

IAM 支持的授权项

策略包含系统策略和自定义策略，如果系统策略不满足授权要求，管理员可以创建自定义策略，并通过给用户组授予自定义策略来进行精细的访问控制。策略支持的操作与API相对应，授权项列表说明如下：

- 权限：允许或拒绝某项操作。
- 对应API接口：自定义策略实际调用的API接口。
- 授权项：自定义策略中支持的Action，在自定义策略中的Action中写入授权项，可以实现授权项对应的权限功能。
- 依赖的授权项：部分Action存在对其他Action的依赖，需要将依赖的Action同时写入授权项，才能实现对应的权限功能。
- IAM项目(Project)/企业项目(Enterprise Project)：自定义策略的授权范围，包括IAM项目与企业项目。授权范围如果同时支持IAM项目和企业项目，表示此授权项对应的自定义策略，可以在IAM和企业管理两个服务中给用户组授权并生效。如果仅支持IAM项目，不支持企业项目，表示仅能在IAM中给用户组授权并生效，如果在企业管理中授权，则该自定义策略不生效。关于IAM项目与企业项目的区别，详情请参见：**IAM与企业管理的区别**。

说明

“√”表示支持，“x”表示暂不支持。

云容器引擎（CCE）支持的自定义策略授权项如下所示：

表 11-2 Cluster

权限	对应API接口	授权项（Action）	IAM项目(Project)	企业项目(Enterprise Project)
获取指定项目下的集群	**[Standard/Turbo]** GET /api/v3/projects/{project_id}/clusters **[Autopilot]** GET /autopilot/v3/projects/{project_id}/clusters	cce:cluster:list	√	√
获取指定的集群	**[Standard/Turbo]** GET /api/v3/projects/{project_id}/clusters/{cluster_id} **[Autopilot]** GET /autopilot/v3/projects/{project_id}/clusters/{cluster_id}	cce:cluster:get	√	√
创建集群	**[Standard/Turbo]** POST /api/v3/projects/{project_id}/clusters **[Autopilot]** POST /autopilot/v3/projects/{project_id}/clusters	cce:cluster:create	√	√
更新指定的集群	**[Standard/Turbo]** PUT /api/v3/projects/{project_id}/clusters/{cluster_id} **[Autopilot]** PUT /autopilot/v3/projects/{project_id}/clusters/{cluster_id}	cce:cluster:update	√	√
删除集群	**[Standard/Turbo]** DELETE /api/v3/projects/{project_id}/clusters/{cluster_id} **[Autopilot]** DELETE /autopilot/v3/projects/{project_id}/clusters/{cluster_id}	cce:cluster:delete	√	√

权限	对应API接口	授权项（Action）	IAM项目(Project)	企业项目(Enterprise Project)
升级集群	**[Standard/Turbo]** POST /api/v3/projects/{project_id}/clusters/{cluster_id}/operation/upgrade **[Autopilot]** POST /autopilot/v3/projects/{project_id}/clusters/{cluster_id}/operation/upgrade	cce:cluster:upgrade	√	√
唤醒集群	**[Standard/Turbo]** POST /api/v3/projects/{project_id}/clusters/{cluster_id}/operation/awake **[Autopilot]** 不涉及	cce:cluster:start	√	√
休眠集群	**[Standard/Turbo]** POST /api/v3/projects/{project_id}/clusters/{cluster_id}/operation/hibernate **[Autopilot]** 不涉及	cce:cluster:stop	√	√
变更集群规格	**[Standard/Turbo]** POST /api/v2/projects/{project_id}/clusters/:clusterid/resize **[Autopilot]** 不涉及	cce:cluster:resize	√	√
绑定、解绑集群公网APIServer地址	**[Standard/Turbo]** PUT /api/v3/projects/{project_id}/clusters/{cluster_id}/mastereip **[Autopilot]** PUT /autopilot/v3/projects/{project_id}/clusters/{cluster_id}/mastereip	cce:cluster:update	√	√

权限	对应API接口	授权项（Action）	IAM项目(Project)	企业项目(Enterprise Project)
获取集群证书	**[Standard/Turbo]** POST /api/v3/projects/{project_id}/clusters/{cluster_id}/clustercert **[Autopilot]** POST /autopilot/v3/projects/{project_id}/clusters/{cluster_id}/clustercert	cce:cluster:get	√	√

表 11-3 Node

权限	对应API接口	授权项	IAM项目(Project)	企业项目(Enterprise Project)
获取集群下所有节点	**[Standard/Turbo]** GET /api/v3/projects/{project_id}/clusters/{cluster_id}/nodes **[Autopilot]** 不涉及	cce:node:list	√	√
获取指定的节点	**[Standard/Turbo]** GET /api/v3/projects/{project_id}/clusters/{cluster_id}/nodes/{node_id} **[Autopilot]** 不涉及	cce:node:get	√	√
创建节点	**[Standard/Turbo]** POST /api/v3/projects/{project_id}/clusters/{cluster_id}/nodes **[Autopilot]** 不涉及	cce:node:create	√	√ **说明** 使用企业项目授权创建节点需额外添加evs:quota:get的全局权限。

权限	对应API接口	授权项	IAM项目(Project)	企业项目(Enterprise Project)
更新指定的节点	**[Standard/Turbo]** PUT /api/v3/projects/{project_id}/clusters/{cluster_id}/nodes/{node_id} **[Autopilot]** 不涉及	cce:node:update	√	√
删除节点	**[Standard/Turbo]** DELETE /api/v3/projects/{project_id}/clusters/{cluster_id}/nodes/{node_id} **[Autopilot]** 不涉及	cce:node:delete	√	√

表 11-4 Job

权限	对应API接口	授权项	IAM项目(Project)	企业项目(Enterprise Project)
获取任务信息	**[Standard/Turbo]** GET /api/v3/projects/{project_id}/jobs/{job_id} **[Autopilot]** GET /autopilot/v3/projects/{project_id}/jobs/{job_id}	cce:job:get	√	√
列出所有任务	**[Standard/Turbo]** GET /api/v2/projects/{project_id}/jobs **[Autopilot]** GET /autopilot/v2/projects/{project_id}/jobs	cce:job:list	√	√

权限	对应API接口	授权项	IAM项目(Project)	企业项目(Enterprise Project)
删除所有任务或删除单个任务	**[Standard/Turbo]** DELETE /api/v2/projects/{project_id}/jobs DELETE /api/v2/projects/{project_id}/jobs/{job_id} **[Autopilot]** DELETE /autopilot/v2/projects/{project_id}/jobs DELETE /autopilot/v2/projects/{project_id}/jobs/{job_id}	cce:job:delete	√	√

表 11-5 Nodepool

权限	对应API接口	授权项	IAM项目(Project)	企业项目(Enterprise Project)
获取集群下所有节点池	**[Standard/Turbo]** GET /api/v3/projects/{project_id}/clusters/{cluster_id}/nodepools **[Autopilot]** 不涉及	cce:nodepool:list	√	√
获取节点池	**[Standard/Turbo]** GET /api/v3/projects/{project_id}/clusters/{cluster_id}/nodepools/{nodepool_id} **[Autopilot]** 不涉及	cce:nodepool:get	√	√
创建节点池	**[Standard/Turbo]** POST /api/v3/projects/{project_id}/clusters/{cluster_id}/nodepools **[Autopilot]** 不涉及	cce:nodepool:create	√	√

权限	对应API接口	授权项	IAM项目(Project)	企业项目(Enterprise Project)
更新节点池信息	**[Standard/Turbo]** PUT /api/v3/projects/{project_id}/clusters/{cluster_id}/nodepools/{nodepool_id} **[Autopilot]** 不涉及	cce:nodepool:update	√	√
删除节点池	**[Standard/Turbo]** DELETE /api/v3/projects/{project_id}/clusters/{cluster_id}/nodepools/{nodepool_id} **[Autopilot]** 不涉及	cce:nodepool:delete	√	√

表 11-6 Chart

权限	对应API接口	授权项	IAM项目(Project)	企业项目(Enterprise Project)
更新模板	**[Standard/Turbo]** PUT /v2/charts/{id} **[Autopilot]** POST /autopilot/v2/charts	cce:chart:update	√	×
上传模板	**[Standard/Turbo]** POST /v2/charts **[Autopilot]** POST /autopilot/v2/charts	cce:chart:upload	√	×
下载模板	**[Standard/Turbo]** GET /v2/charts/{id}/archive **[Autopilot]** GET /autopilot/v2/charts/{id}/archive	cce:chart:get	√	×

权限	对应API接口	授权项	IAM项目(Project)	企业项目(Enterprise Project)
列出所有模板	**[Standard/Turbo]** GET /v2/charts **[Autopilot]** GET /autopilot/v2/charts	cce:chart:list	√	×
获取模板信息	**[Standard/Turbo]** GET /v2/charts/{id} **[Autopilot]** GET /autopilot/v2/charts/{id}	cce:chart:get	√	×
获取模板Values	**[Standard/Turbo]** GET /v2/charts/{id}/values **[Autopilot]** GET /autopilot/v2/charts/{id}/values	cce:chart:get	√	×
删除模板	**[Standard/Turbo]** DELETE /v2/charts/{id} **[Autopilot]** DELETE /autopilot/v2/charts/{id}	cce:chart:delete	√	×
获取用户模板配额	**[Standard/Turbo]** GET /v2/charts/{project_id}/quotas **[Autopilot]** GET /autopilot/v2/charts/{project_id}/quotas	cce:chart:list	√	×

表 11-7 Release

权限	对应API接口	授权项	IAM项目(Project)	企业项目(Enterprise Project)
更新升级模板实例	**[Standard/Turbo]** PUT /cce/cam/v3/clusters/{cluster_id}/namespace/{namespace}/releases/{name} PUT /v2/releases/{name}（已废弃） **[Autopilot]** PUT /autopilot/cam/v3/clusters/{cluster_id}/namespace/{namespace}/releases/{name}	cce:release:update	√	√
列出所有模板实例	**[Standard/Turbo]** GET /cce/cam/v3/clusters/{cluster_id}/releases GET /v2/releases（已废弃） **[Autopilot]** GET /autopilot/cam/v3/clusters/{cluster_id}/releases	cce:release:list	√	√
创建模板实例	**[Standard/Turbo]** POST /cce/cam/v3/clusters/{cluster_id}/releases POST /v2/releases（已废弃） **[Autopilot]** POST /autopilot/cam/v3/clusters/{cluster_id}/releases	cce:release:create	√	√
获取模板实例信息	**[Standard/Turbo]** GET /cce/cam/v3/clusters/{cluster_id}/namespace/{namespace}/releases/{name} GET /v2/releases/{name}（已废弃） **[Autopilot]** GET /autopilot/cam/v3/clusters/{cluster_id}/namespace/{namespace}/releases/{name}	cce:release:get	√	√

权限	对应API接口	授权项	IAM项目(Project)	企业项目(Enterprise Project)
查询指定模板实例历史记录	**[Standard/Turbo]** GET /cce/cam/v3/clusters/{cluster_id}/namespace/{namespace}/releases/{name}/history GET /v2/releases/{name}/history（已废弃） **[Autopilot]** GET /autopilot/cam/v3/clusters/{cluster_id}/namespace/{namespace}/releases/{name}	cce:release:get	√	√
删除模板实例	**[Standard/Turbo]** DELETE /cce/cam/v3/clusters/{cluster_id}/namespace/{namespace}/releases/{name} DELETE /v2/releases/{name}（已废弃） **[Autopilot]** DELETE /autopilot/cam/v3/clusters/{cluster_id}/namespace/{namespace}/releases/{name}	cce:release:delete	√	√

表 11-8 Storage

权限	对应API接口	授权项	IAM项目(Project)	企业项目(Enterprise Project)
创建PersistentVolumeClaim（待废弃）	**[Standard/Turbo]** POST /api/v1/namespaces/{namespace}/cloudpersistentvolumeclaims **[Autopilot]** 不涉及	cce:storage:create	√	√

权限	对应API接口	授权项	IAM项目(Project)	企业项目(Enterprise Project)
删除PersistentVolumeClaim（待废弃）	**[Standard/Turbo]** DELETE /api/v1/namespaces/{namespace}/cloudpersistentvolumeclaims/{name} **[Autopilot]** 不涉及	cce:storage:delete	√	√
列出所有磁盘	**[Standard/Turbo]** GET /storage/api/v1/namespaces/{namespace}/listvolumes **[Autopilot]** 不涉及	cce:storage:list	√	√

表 11-9 Addon

权限	对应API接口	授权项	IAM项目(Project)	企业项目(Enterprise Project)
列出所有插件模板	**[Standard/Turbo]** GET /api/v3/addontemplate **[Autopilot]** GET /autopilot/v3/addontemplates	cce:addonTemplate:get	√	x
创建插件实例	**[Standard/Turbo]** POST /api/v3/addons **[Autopilot]** POST /autopilot/v3/addons	cce:addonInstance:create	√	√
获取插件实例	**[Standard/Turbo]** GET /api/v3/addons/{id}?cluster_id={cluster_id} **[Autopilot]** GET /autopilot/v3/addons/{id}	cce:addonInstance:get	√	√

权限	对应API接口	授权项	IAM项目(Project)	企业项目(Enterprise Project)
列出所有插件实例	**[Standard/Turbo]** GET /api/v3/addons?cluster_id={cluster_id} **[Autopilot]** GET /autopilot/v3/addons?cluster_id={cluster_id}	cce:addonInstance:list	√	√
删除插件实例	**[Standard/Turbo]** DELETE /api/v3/addons/{id} **[Autopilot]** DELETE /autopilot/v3/addons/{id}	cce:addonInstance:delete	√	√
更新升级插件实例	**[Standard/Turbo]** PUT /api/v3/addons/{id} **[Autopilot]** PUT /autopilot/v3/addons/{id}	cce:addonInstance:update	√	√

表 11-10 Quota

权限	对应API接口	授权项	IAM项目(Project)	企业项目(Enterprise Project)
查询配额详情	**[Standard/Turbo]** GET /api/v3/projects/{project_id}/quotas **[Autopilot]** GET /autopilot/v3/projects/{project_id}/quotas	cce:quota:get	√	√

表 11-11 Label

权限	对应API接口	授权项	IAM项目(Project)	企业项目(Enterprise Project)
批量添加指定集群的资源标签	**[Standard/Turbo]** POST /api/v3/projects/{project_id}/clusters/{cluster_id}/tags/create **[Autopilot]** POST /autopilot/v3/projects/{project_id}/clusters/{cluster_id}/tags/create	cce:tag:operate	√	√
批量删除指定集群的资源标签	**[Standard/Turbo]** POST /api/v3/projects/{project_id}/clusters/{cluster_id}/tags/delete **[Autopilot]** POST /autopilot/v3/projects/{project_id}/clusters/{cluster_id}/tags/delete	cce:tag:operate	√	√

表 11-12 Upgrade

权限	对应API接口	授权项	IAM项目(Project)	企业项目(Enterprise Project)
获取集群升级任务详情	**[Standard/Turbo]** GET /api/v3/projects/{project_id}/clusters/{cluster_id}/operation/upgrade/tasks/{task_id} **[Autopilot]** GET /autopilot/v3/projects/{project_id}/clusters/{cluster_id}/operation/upgrade/tasks/{task_id}	cce:cluster:get	√	√

权限	对应API接口	授权项	IAM项目(Project)	企业项目(Enterprise Project)
获取集群升级任务详情列表	**[Standard/Turbo]** GET /api/v3/projects/{project_id}/clusters/{cluster_id}/operation/upgrade/tasks **[Autopilot]** GET /autopilot/v3/projects/{project_id}/clusters/{cluster_id}/operation/upgrade/tasks	cce:cluster:get	√	√
重试集群升级任务	**[Standard/Turbo]** POST /api/v3/projects/{project_id}/clusters/{cluster_id}/operation/upgrade/retry **[Autopilot]** POST /autopilot/v3/projects/{project_id}/clusters/{cluster_id}/operation/upgrade/retry	cce:cluster:upgrade	√	√
集群升级前检查	**[Standard/Turbo]** POST /api/v3/projects/{project_id}/clusters/{cluster_id}/operation/precheck **[Autopilot]** POST /autopilot/v3/projects/{project_id}/clusters/{cluster_id}/operation/precheck	cce:cluster:upgrade	√	√
获取集群升级前检查任务详情	**[Standard/Turbo]** GET /api/v3/projects/{project_id}/clusters/{cluster_id}/operation/precheck/tasks/{task_id} **[Autopilot]** GET /autopilot/v3/projects/{project_id}/clusters/{cluster_id}/operation/precheck/tasks/{task_id}	cce:cluster:get	√	√

权限	对应API接口	授权项	IAM项目(Project)	企业项目(Enterprise Project)
获取集群升级前检查任务详情列表	**[Standard/Turbo]** GET /api/v3/projects/{project_id}/clusters/{cluster_id}/operation/precheck/tasks **[Autopilot]** GET /autopilot/v3/projects/{project_id}/clusters/{cluster_id}/operation/precheck/tasks	cce:cluster:get	√	√
集群升级后确认	**[Standard/Turbo]** POST /api/v3/projects/{project_id}/clusters/{cluster_id}/operation/postcheck **[Autopilot]** POST /autopilot/v3/projects/{project_id}/clusters/{cluster_id}/operation/postcheck	cce:cluster:upgrade	√	√
集群备份	**[Standard/Turbo]** POST /api/v3.1/projects/{project_id}/clusters/{cluster_id}/operation/snapshot **[Autopilot]** POST /autopilot/v3.1/projects/{project_id}/clusters/{cluster_id}/operation/snapshot	cce:cluster:upgrade	√	√
获取集群备份任务详情列表	**[Standard/Turbo]** GET /api/v3.1/projects/{project_id}/clusters/{cluster_id}/operation/snapshot/tasks **[Autopilot]** GET /autopilot/v3.1/projects/{project_id}/clusters/{cluster_id}/operation/snapshot/tasks	cce:cluster:get	√	√

权限	对应API接口	授权项	IAM项目(Project)	企业项目(Enterprise Project)
获取集群升级相关信息	**[Standard/Turbo]** GET /api/v3/projects/{project_id}/clusters/{cluster_id}/upgradeinfo **[Autopilot]** GET /autopilot/v3/projects/{project_id}/clusters/{cluster_id}/upgradeinfo	cce:cluster:get	√	√
开启集群升级流程引导任务	**[Standard/Turbo]** POST /api/v3/projects/{project_id}/clusters/{cluster_id}/operation/upgradeworkflows **[Autopilot]** POST /autopilot/v3/projects/{project_id}/clusters/{cluster_id}/operation/upgradeworkflows	cce:cluster:upgrade	√	√
获取历史集群升级引导任务列表	**[Standard/Turbo]** GET /api/v3/projects/{project_id}/clusters/{cluster_id}/operation/upgradeworkflows **[Autopilot]** GET /autopilot/v3/projects/{project_id}/clusters/{cluster_id}/operation/upgradeworkflows	cce:cluster:get	√	√
获取指定集群升级引导任务详情	**[Standard/Turbo]** GET /api/v3/projects/{project_id}/clusters/{cluster_id}/operation/upgradeworkflows/{upgrade_workflow_id} **[Autopilot]** GET /autopilot/v3/projects/{project_id}/clusters/{cluster_id}/operation/upgradeworkflows/{upgrade_workflow_id}	cce:cluster:get	√	√

权限	对应API接口	授权项	IAM项目(Project)	企业项目(Enterprise Project)
更新指定集群升级引导任务状态	**[Standard/Turbo]** PATCH /api/v3/projects/{project_id}/clusters/{cluster_id}/operation/upgradeworkflows/{upgrade_workflow_id} **[Autopilot]** PATCH /autopilot/v3/projects/{project_id}/clusters/{cluster_id}/operation/upgradeworkflows/{upgrade_workflow_id}	cce:cluster:upgrade	√	√

11.2 集群权限（IAM 授权）

CCE集群权限是基于IAM**系统策略**和**自定义策略**的授权，可以通过用户组功能实现IAM用户的授权。

⚠ 注意

- 集群权限仅针对与集群相关的资源（如集群、节点等）有效，您必须确保同时配置了**命名空间权限**，才能有操作Kubernetes资源（如工作负载、Service等）的权限。
- 使用CCE控制台查看集群时，显示情况依赖于命名空间权限的设置情况，如果没有设置命名空间权限，则无法查看集群下的资源。

前提条件

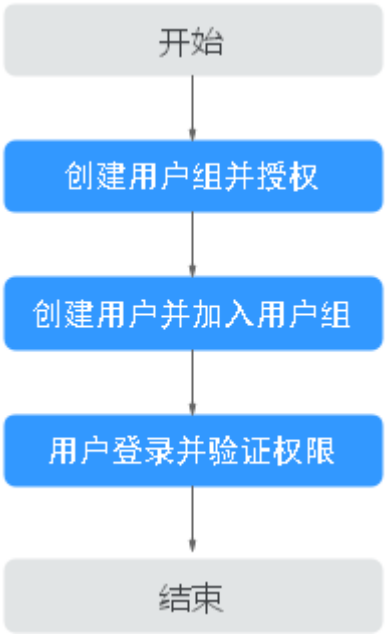
- 给用户组授权之前，请您了解用户组可以添加的CCE系统策略，并结合实际需求进行选择，CCE支持的系统策略及策略间的对比，请参见**CCE系统权限**。若您需要对除CCE之外的其它服务授权，IAM支持服务的所有策略请参见**系统权限**。
- 拥有Security Administrator（IAM除切换角色外所有权限）权限的用户（如账号默认拥有此权限），才能看见CCE控制台权限管理页面当前用户组及用户组所拥有的权限。

配置说明

CCE控制台“权限管理 > 集群权限”页面中创建用户组和具体权限设置均是跳转到IAM控制台进行具体操作，设置完后在集群权限页面能看到用户组所拥有的权限。本章节描述操作直接以IAM中操作为主，不重复介绍在CCE控制台如何跳转。

示例流程

图 11-2 给用户授予 CCE 权限流程



1. **创建用户组并授权**。
   在IAM控制台创建用户组，并授予CCE权限，例如CCE ReadOnlyAccess。

   说明

   CCE服务按区域部署，在IAM控制台授予CCE权限时请选择“区域级项目”。
2. **创建用户并加入用户组** 。
   在IAM控制台创建用户，并将其加入**1**中创建的用户组。

   须知

   通过IAM用户使用CCE时，该IAM用户需要同时支持“编程访问”和“管理控制台访问”的访问方式。
3. **用户登录** 并验证权限。
   新创建的用户登录控制台，切换至授权区域，验证权限：
   - 在“服务列表”中选择云容器引擎，进入CCE主界面尝试购买集群，如果无法无法成功操作（假设当前权限仅包含CCE ReadOnlyAccess），表示“CCE ReadOnlyAccess”已生效。
   - 在“服务列表”中选择除云容器引擎外（假设当前策略仅包含CCE ReadOnlyAccess）的任一服务，若提示权限不足，表示“CCE ReadOnlyAccess”已生效。

系统角色

角色是IAM最初提供的一种根据用户的工作职能定义权限的粗粒度授权机制。该机制以服务为粒度，提供有限的服务相关角色用于授权。角色并不能满足用户对精细化授权的要求，无法完全达到企业对权限最小化的安全管控要求。

IAM中预置的CCE系统角色为**CCE Administrator**，给用户组授予该系统角色权限时，必须同时勾选该角色依赖的其他策略才会生效，例如Tenant Guest、Server Administrator、ELB Administrator、OBS Administrator、SFS Administrator、SWR Admin、APM FullAccess。了解更多角色依赖关系，请参考：**系统权限**。

系统策略

IAM中预置的CCE系统策略当前包含**CCE FullAccess**和**CCE ReadOnlyAccess**两种策略：

- **CCE FullAccess**：系统策略，CCE服务集群相关资源的普通操作权限，不包括集群（启用Kubernetes RBAC鉴权）的命名空间权限，不包括委托授权、生成集群证书等管理员角色的特权操作。
- **CCE ReadOnlyAccess**：系统策略，CCE服务集群相关资源的只读权限，不包括集群（启用Kubernetes RBAC鉴权）的命名空间权限。

说明

CCE Standard/Turbo集群购买包周期集群、节点时，需要为用户添加**自定义策略**，额外配置费用中心服务的支付相关权限，如bss:*:*。

表11-13和**表11-14**中，“√”表示支持，“x”表示不支持。

表 11-13 CCE FullAccess 策略主要权限

操作（Action）	Action详情	CCE Standard/Turbo集群	CCE Autopilot集群	说明
cce:*:*	cce:cluster:create	√	√	创建集群
	cce:cluster:delete	√	√	删除集群
	cce:cluster:update	√	√	更新集群，如后续允许集群支持RBAC，调度参数更新等
	cce:cluster:upgrade	√	√	升级集群
	cce:cluster:start	√	×	唤醒集群
	cce:cluster:stop	√	×	休眠集群
	cce:cluster:list	√	√	查询集群列表

操作（Action）	Action详情	CCE Standard /Turbo集群	CCE Autopilot集群	说明
	cce:cluster:get	√	√	查询集群详情
	cce:node:create	√	×	添加节点
	cce:node:delete	√	×	删除节点/批量删除节点
	cce:node:update	√	×	更新节点，如更新节点名称
	cce:node:get	√	×	查询节点详情
	cce:node:list	√	×	查询节点列表
	cce:nodepool:create	√	×	创建节点池
	cce:nodepool:delete	√	×	删除节点池
	cce:nodepool:update	√	×	更新节点池信息
	cce:nodepool:get	√	×	获取节点池
	cce:nodepool:list	√	×	列出集群的所有节点池
	cce:release:create	√	√	创建模板实例
	cce:release:delete	√	√	删除模板实例
	cce:release:update	√	√	更新升级模板实例
	cce:job:list	√	√	查询任务列表（集群层面的job）
	cce:job:delete	√	√	删除任务/批量删除任务（集群层面的job）
	cce:job:get	√	√	查询任务详情（集群层面的job）
	cce:storage:create	√	√	创建存储

操作（Action）	Action详情	CCE Standard /Turbo集群	CCE Autopilot集群	说明
	cce:storage:delete	√	√	删除存储
	cce:storage:list	√	√	列出所有磁盘
	cce:addonInstance:create	√	√	创建插件实例
	cce:addonInstance:delete	√	√	删除插件实例
	cce:addonInstance:update	√	√	更新升级插件实例
	cce:addonInstance:get	√	√	获取插件实例
	cce:addonTemplate:get	√	√	获取插件模板
	cce:addonInstance:list	√	√	列出所有插件实例
	cce:addonTemplate:list	√	√	列出所有插件模板
	cce:chart:list	√	√	列出所有模板
	cce:chart:delete	√	√	删除模板
	cce:chart:update	√	√	更新模板
	cce:chart:upload	√	√	上传模板
	cce:chart:get	√	√	获取模板信息
	cce:release:get	√	√	获取模板实例信息
	cce:release:list	√	√	列出所有模板实例

操作（Action）	Action详情	CCE Standard /Turbo集群	CCE Autopilot集群	说明
	cce:userAuthorization:get	√	√	获取CCE用户授权
	cce:userAuthorization:create	√	√	创建CCE用户授权
nat:*:*	-	×	√	NAT网关服务的所有权限。
nat:*:get	-	√	√	NAT网关服务所有资源详情的查看权限。
nat:*:list	-	√	√	NAT网关服务所有资源列表的查看权限。
vpcep:*:*	-	×	√	VPCEP（VPC终端节点）的所有权限。
ecs:*:*	-	√	√	ECS（弹性云服务器）服务的所有权限。
evs:*:*	具体action详见：**云硬盘v2接口的授权信息**。	√	√	EVS（云硬盘）的所有权限。 可以将云硬盘挂载到云服务器，并可以随时扩容云硬盘容量
vpc:*:*	-	√	√	VPC（虚拟私有云，包含二代ELB）的所有权限。 创建的集群需要运行在虚拟私有云中，创建命名空间时，需要创建或关联VPC，创建在命名空间的容器都运行在VPC之内。
bms:*:get*	-	√	√	BMS（裸金属服务器）所有资源详情的查看权限。
bms:*:list*	-	√	√	BMS（裸金属服务器）所有资源列表的查看权限。
ims:*:get*	-	√	√	IMS（镜像服务）所有资源详情的查看权限。

操作（Action）	Action详情	CCE Standard /Turbo集群	CCE Autopilot集群	说明
ims:*:list*	-	√	√	IMS（镜像服务）所有资源列表的查看权限。
elb:*:get	-	√	√	ELB（弹性负载均衡）所有资源详情的查看权限。
elb:*:list	-	√	√	ELB（弹性负载均衡）所有资源列表的查看权限。
sfs:*:get*	-	√	√	SFS（弹性文件存储）所有资源详情的查看权限。
sfs:shares:ShareAction	-	√	√	SFS（弹性文件存储）资源的扩容共享。
sfsturbo:*:get*	-	√	√	SFS Turbo（极速弹性文件存储）服务所有资源详情的查看权限。
sfsturbo:shares:ShareAction	-	√	√	SFS Turbo（极速弹性文件存储）资源的扩容共享。
tms:resourceTags:list	-	√	√	TMS（标签管理服务）资源标签列表查看权限。
kps:domainKeypairs:list	-	√	√	DEW（数据加密服务）账号密钥对的SSH密钥列表查看权限。
kps:domainKeypairs:get	-	√	√	DEW（数据加密服务）账号密钥对的SSH密钥详情查看权限。
kms:cmk:get	-	√	√	DEW（数据加密服务）查看密钥信息权限。
kms:cmk:list	-	√	√	DEW（数据加密服务）查看密钥列表权限。
aom:*:get	-	√	√	AOM（应用运维管理）资源详情的查看权限。

操作（Action）	Action详情	CCE Standard /Turbo集群	CCE Autopilot集群	说明
aom:*:list	-	√	√	AOM（应用运维管理）资源列表的查看权限。
aom:autoScalingRule:*	-	√	√	AOM（应用运维管理）自动扩缩容规则的所有操作权限。
apm:icmgr:*	-	√	√	APM（应用性能管理服务）操作ICAgent权限。
lts:*:*	-	√	√	LTS（云日志服务）的所有权限。
smn:*:*	-	√	√	SMN（消息通知服务）的所有权限。

表 11-14 CCE ReadOnlyAccess 策略主要权限

操作（Action）	操作（Action）	CCE Standard /Turbo集群	CCE Autopilot集群	说明
cce:*:get	cce:cluster:get	√	√	查询集群详情
	cce:node:get	√	×	查询节点详情
	cce:job:get	√	√	查询任务详情（集群层面的job）
	cce:addonInstance:get	√	√	获取插件实例
	cce:addonTemplate:get	√	√	获取插件模板
	cce:chart:get	√	√	获取模板信息
	cce:nodepool:get	√	×	获取节点池
	cce:release:get	√	√	获取模板实例信息
	cce:userAuthorization:get	√	√	获取CCE用户授权

操作（Action）	操作（Action）	CCE Standard/Turbo集群	CCE Autopilot集群	说明
cce:*:list	cce:cluster:list	√	√	查询集群列表
	cce:node:list	√	×	查询节点列表
	cce:job:list	√	√	查询任务列表（集群层面的job）
	cce:addonInstance:list	√	√	列出所有插件实例
	cce:addonTemplate:list	√	√	列出所有插件模板
	cce:chart:list	√	√	列出所有模板
	cce:nodepool:list	√	×	列出集群的所有节点池
	cce:release:list	√	√	列出所有模板实例
	cce:storage:list	√	√	列出所有磁盘
cce:kubernetes:*	-	√	√	操作所有Kubernetes资源，具体权限请在**命名空间权限**中配置。
nat:*:get	-	√	√	NAT网关服务所有资源详情的查看权限。
nat:*:list	-	√	√	NAT网关服务所有资源列表的查看权限。
vpcep:*:get	-	×	√	VPCEP（VPC终端节点）所有资源详情的查看权限。
vpcep:*:list	-	×	√	VPCEP（VPC终端节点）所有资源列表的查看权限。
ecs:*:get	-	√	√	ECS（弹性云服务器）所有资源详情的查看权限。
ecs:*:list	-	√	√	ECS（弹性云服务器）所有资源列表的查看权限。

操作（Action）	操作（Action）	CCE Standard /Turbo集群	CCE Autopilot集群	说明
bms:*:get*	-	√	√	BMS（裸金属服务器）所有资源详情的查看权限。
bms:*:list	-	√	√	BMS（裸金属服务器）所有资源列表的查看权限。
ims:*:get*	-	√	√	IMS（镜像服务）所有资源详情的查看权限。
ims:*:list*	-	√	√	IMS（镜像服务）所有资源列表的查看权限。
evs:*:get	-	√	√	EVS（云硬盘）所有资源详情的查看权限。 可以将云硬盘挂载到云服务器，并可以随时扩容云硬盘容量
evs:*:list	-	√	√	EVS（云硬盘）所有资源列表的查看权限。
evs:*:count	-	√	√	-
vpc:*:get	-	√	√	VPC（虚拟私有云）所有资源详情的查看权限。 创建的集群需要运行在虚拟私有云中，创建命名空间时，需要创建或关联VPC，创建在命名空间的容器都运行在VPC之内。
vpc:*:list	-	√	√	VPC（虚拟私有云）所有资源列表的查看权限。
elb:*:get	-	√	√	ELB（弹性负载均衡）所有资源详情的查看权限。
elb:*:list	-	√	√	ELB（弹性负载均衡）所有资源列表的查看权限。

操作（Action）	操作（Action）	CCE Standard /Turbo集群	CCE Autopilot集群	说明
sfs:*:get*	-	√	√	SFS（弹性文件存储）所有资源详情的查看权限。
sfs:shares:ShareAction	-	√	√	SFS（弹性文件存储）资源的扩容共享。
sfsturbo:*:get*	-	√	√	SFS Turbo（极速弹性文件存储）服务所有资源详情的查看权限。
sfsturbo:shares:ShareAction	-	√	√	SFS Turbo（极速弹性文件存储）资源的扩容共享。
tms:resourceTags:list	-	√	√	TMS（标签管理服务）资源标签列表查看权限。
kps:domainKeypairs:list	-	√	√	DEW（数据加密服务）账号密钥对的SSH密钥列表查看权限。
kps:domainKeypairs:get	-	√	√	DEW（数据加密服务）账号密钥对的SSH密钥详情查看权限。
kms:cmk:get	-	√	√	DEW（数据加密服务）查看密钥信息权限。
kms:cmk:list	-	√	√	DEW（数据加密服务）查看密钥列表权限。
aom:*:get	-	√	√	AOM（应用运维管理）服务所有资源详情的查看权限。
aom:*:list	-	√	√	AOM（应用运维管理）服务所有资源列表的查看权限。
aom:autoScalingRule:*	-	√	√	AOM（应用运维管理）服务自动扩缩容规则的所有操作权限。

操作（Action）	操作（Action）	CCE Standard /Turbo集群	CCE Autopilot集群	说明
lts:*:get	-	√	√	LTS（云日志服务）的所有资源详情的查看权限。
lts:*:list	-	√	√	LTS（云日志服务）的所有资源列表的查看权限。
smn:*:get	-	√	√	SMN（消息通知服务）所有资源详情的查看权限。
smn:*:list	-	√	√	SMN（消息通知服务）所有资源列表的查看权限。

自定义策略

如果系统预置的CCE策略，不满足您的授权要求，可以创建自定义策略。自定义策略中可以添加的授权项（Action）请参考**权限策略和授权项**。

目前支持以下两种方式创建自定义策略：

- 可视化视图创建自定义策略：无需了解策略语法，按可视化视图导航栏选择云服务、操作、资源、条件等策略内容，可自动生成策略。
- JSON视图创建自定义策略：可以在选择策略模板后，根据具体需求编辑策略内容；也可以直接在编辑框内编写JSON格式的策略内容。

具体创建步骤请参见：**创建自定义策略**。本章为您介绍常用的CCE自定义策略样例。

CCE自定义策略样例：

- 示例1：创建一个名称为“test”的集群

```
{
    "Version": "1.1",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "cce:cluster:create"
            ]
        }
    ]
}
```

- 示例2：拒绝用户删除节点

  拒绝策略需要同时配合其他策略使用，否则没有实际作用。用户被授予的策略中，一个授权项的作用如果同时存在Allow和Deny，则遵循**Deny优先原则**。

  如果您给用户授予CCEFullAccess的系统策略，但不希望用户拥有CCEFullAccess中定义的删除节点权限（cce:node:delete），您可以创建一条相同Action的自定义策略，并将自定义策略的Effect设置为Deny，然后同时将CCEFullAccess和拒绝策

略授予用户，根据Deny优先原则，则用户可以对CCE执行除了删除节点外的所有操作。拒绝策略示例如下：

```
{
    "Version": "1.1",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "cce:node:delete"
            ]
        }
    ]
}
```

- 示例3：多个授权项策略

  一个自定义策略中可以包含多个授权项，且除了可以包含本服务的授权项外，还可以包含其他服务的授权项，可以包含的其他服务必须跟本服务同属性，即都是项目级服务或都是全局级服务。多个授权语句策略描述如下：

```
{
    "Version": "1.1",
    "Statement": [
        {
            "Action": [
                "ecs:cloudServers:resize",
                "ecs:cloudServers:delete",
                "ecs:cloudServers:delete",
                "ims:images:list",
                "ims:serverImages:create"
            ],
            "Effect": "Allow"
        }
    ]
}
```

CCE 集群权限与企业项目

CCE支持以集群为粒度，基于企业项目维度进行资源管理以及权限分配。

如下事项需特别注意：

- IAM项目是基于资源的物理隔离进行管理，而企业项目则是提供资源的全局逻辑分组，更符合企业实际场景，并且支持基于企业项目维度的IAM策略管理，因此推荐您使用企业项目。详细信息请参见**如何创建企业项目**。
- IAM项目与企业项目共存时，IAM将优先匹配IAM项目策略、未决则匹配企业项目策略。
- CCE集群基于已有基础资源（VPC）创建集群、节点时，请确保IAM用户在已有资源的企业项目下有相关权限，否则可能导致集群或者节点创建失败。
- 当资源不支持企业项目时，为企业项目授予该资源的权限将不会生效。

表 11-15 企业项目资源

是否支持企业项目	资源名称	说明
支持企业项目的资源	cluster	集群
	node	节点
	nodepool	节点池

是否支持企业项目	资源名称	说明
	job	任务
	tag	集群标签
	addonInstance	插件实例
	release	Helm版本
	storage	存储资源
不支持企业项目的资源	quota	集群配额
	chart	模板
	addonTemplate	插件模板

CCE 集群权限与 IAM RBAC

CCE兼容IAM传统的系统角色进行权限管理，建议您切换使用IAM的细粒度策略，避免设置过于复杂或不必要的权限管理场景。

CCE当前支持的角色如下：

- IAM的基础角色：
  - te_admin（Tenant Administrator）：可以调用除IAM外所有服务的所有API。
  - readonly（Tenant Guest）：可以调用除IAM外所有服务的只读权限的API。
- CCE的自定义管理员角色：CCE Administrator。

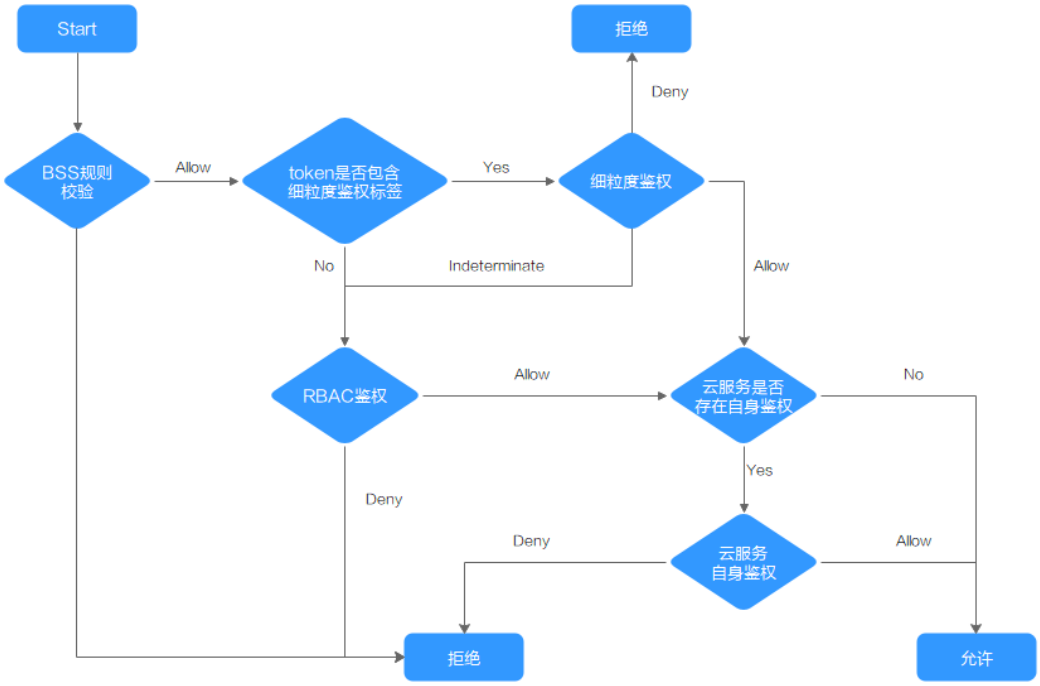
说明

如果用户有Tenant Administrator或者CCE Administrator的系统角色，则此用户拥有Kubernetes RBAC的cluster-admin权限，在集群创建后不可移除。

如果用户为集群创建者，则默认被授权Kubernetes RBAC的cluster-admin权限，此项权限可以在集群创建后被手动移除:

- 方式1：权限管理 - 命名空间权限 - 移除cluster-creator。
- 方式2：通过API或者kubectl删除资源，ClusterRoleBinding：cluster-creator。

RBAC与IAM策略共存时，CCE开放API或Console操作的后端鉴权逻辑如下：



11.3 命名空间权限（Kubernetes RBAC 授权）

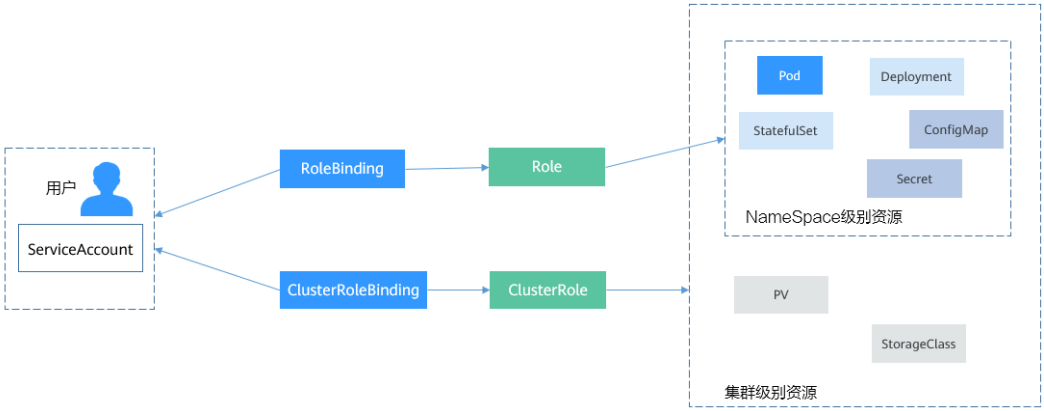
命名空间权限（kubernetes RBAC 授权）

命名空间权限是基于Kubernetes RBAC能力的授权，通过权限设置可以让不同的用户或用户组拥有操作不同Kubernetes资源的权限。Kubernetes RBAC API定义了四种类型：Role、ClusterRole、RoleBinding与ClusterRoleBinding，这四种类型之间的关系和简要说明如下：

- Role：角色，其实是定义一组对Kubernetes资源（命名空间级别）的访问规则。
- RoleBinding：角色绑定，定义了用户和角色的关系。
- ClusterRole：集群角色，其实是定义一组对Kubernetes资源（集群级别，包含全部命名空间）的访问规则。
- ClusterRoleBinding：集群角色绑定，定义了用户和集群角色的关系。

Role和ClusterRole指定了可以对哪些资源做哪些动作，RoleBinding和ClusterRoleBinding将角色绑定到特定的用户、用户组或ServiceAccount上。如下图所示。

图 11-3 角色绑定



在CCE控制台可以授予用户或用户组命名空间权限，可以对某一个命名空间或全部命名空间授权，CCE控制台默认提供如下ClusterRole。

- view（只读权限）：对全部或所选命名空间下大多数资源的只读权限。
- edit（开发权限）：对全部或所选命名空间下多数资源的读写权限。当配置在全部命名空间时能力与运维权限一致。
- admin（运维权限）：对全部命名空间下大多数资源的读写权限，对节点、存储卷，命名空间和配额管理的只读权限。
- cluster-admin（管理员权限）：对全部命名空间下所有资源的读写权限。
- drainage-editor：节点排水操作权限，可执行节点排水。
- drainage-viewer：节点排水只读权限，仅可查看节点排水状态，无法执行节点排水。
- geip-editor-role：该权限用于在集群中使用全域弹性公网IP资源，仅对资源geips具有读写权限。
- icagent-clusterRole：该权限用于生成Kubernetes事件日志上报到LTS。

除了使用上述常用的ClusterRole外，您还可以通过定义Role和RoleBinding来进一步对全局资源（如Node、PersistentVolumes、CustomResourceDefinitions等）和命名空间中不同类别资源（如Pod、Deployment、Service等）的增删改查权限进行配置，从而做到更加精细化的权限控制。

集群权限（IAM 授权）与命名空间权限（Kubernetes RBAC 授权）的关系

拥有不同集群权限（IAM授权）的用户，其拥有的命名空间权限（Kubernetes RBAC授权）不同。表11-16给出了不同用户拥有的命名空间权限详情。

表 11-16 不同用户拥有的命名空间权限

用户类型	1.13及以上版本的集群
拥有Tenant Administrator权限的用户（例如账号）	全部命名空间权限
拥有CCE Administrator权限的IAM用户	全部命名空间权限

用户类型	1.13及以上版本的集群
拥有CCE FullAccess或者CCE ReadOnlyAccess权限的IAM用户	按Kubernetes RBAC授权
拥有Tenant Guest权限的IAM用户	按Kubernetes RBAC授权

注意事项

- 任何用户创建集群后，CCE会自动为该用户添加该集群的所有命名空间的cluster-admin权限，也就是说该用户允许对集群以及所有命名空间中的全部资源进行完全控制。联邦用户由于每次登录注销都会改变用户ID，所以权限用户会显示已删除，此情况下请勿删除该权限，否则会导致鉴权失败。此种情况下建议在CCE为某个用户组创建cluster-admin权限，将联邦用户加入此用户组。
- 拥有Security Administrator（IAM除切换角色外所有权限）权限的用户（如账号所在的admin用户组默认拥有此权限），才能在CCE控制台命名空间权限页面进行授权操作。

配置命名空间权限（控制台）

CCE中的命名空间权限是基于Kubernetes RBAC能力的授权，通过权限设置可以让不同的用户或用户组拥有操作不同Kubernetes资源的权限。

步骤1 登录CCE控制台，在左侧导航栏中选择“权限管理”。

步骤2 在右边下拉列表中选择要添加权限的集群。

步骤3 在右上角单击“添加权限”，进入添加权限页面。

步骤4 在添加权限页面，确认集群名称，选择该集群下要授权使用的命名空间，例如选择“全部命名空间”，选择要授权的用户或用户组，再选择具体权限。

说明

对于没有IAM权限的用户，给其他用户和用户组配置权限时，无法选择用户和用户组，此时支持填写用户ID或用户组ID进行配置。

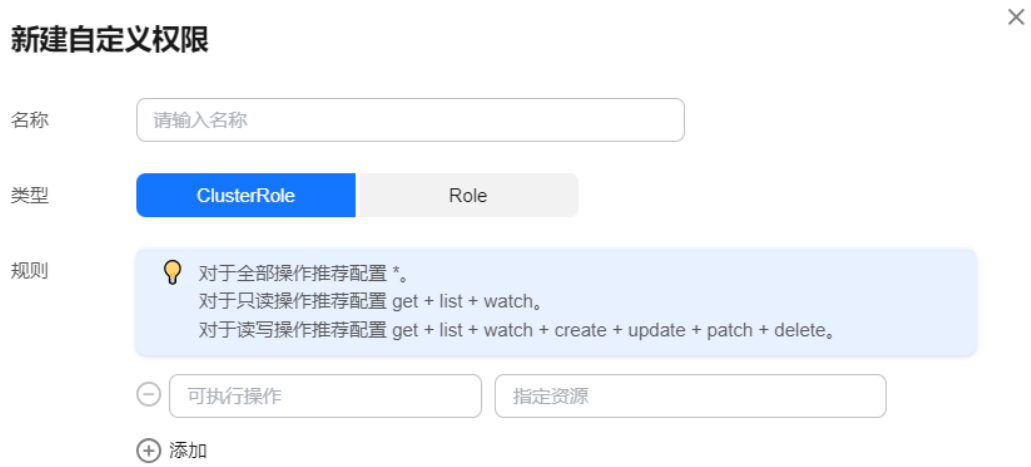
图 11-4 配置命名空间权限

其中自定义权限可以根据需要自定义，选择自定义权限后，在自定义权限一行右侧单击新建自定义权限，在弹出的窗口中填写名称并选择规则。创建完成后，在添加权限的自定义权限下拉框中可以选择。

自定义权限分为ClusterRole或Role两类，ClusterRole或Role均包含一组代表相关权限的规则，详情请参见**使用RBAC鉴权**。

- ClusterRole：ClusterRole是一个集群级别的资源，可设置集群的访问权限。
- Role：Role用于在某个命名空间内设置访问权限。当创建Role时，必须指定该Role所属的命名空间。

图 11-5 自定义权限



步骤5 单击“确定”。

----结束

自定义命名空间权限（kubectl）

说明

kubectl访问CCE集群是通过集群上生成的配置文件（kubeconfig.json）进行认证，kubeconfig.json文件内包含用户信息，CCE根据用户信息的权限判断kubectl有权限访问哪些Kubernetes资源。即哪个用户获取的kubeconfig.json文件，kubeconfig.json就拥有哪个用户的信息，这样使用kubectl访问时就拥有这个用户的权限。而用户拥有的权限就是**集群权限（IAM授权）与命名空间权限（Kubernetes RBAC授权）的关系**所示的权限。

除了使用cluster-admin、admin、edit、view这4个最常用的clusterrole外，您还可以通过定义Role和RoleBinding来进一步对命名空间中不同类别资源（如Pod、Deployment、Service等）的增删改查权限进行配置，从而做到更加精细化的权限控制。

Role的定义非常简单，指定namespace，然后就是rules规则。如下面示例中的规则就是允许对default命名空间下的Pod进行GET、LIST操作。

```
kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  namespace: default                    # 命名空间
  name: role-example
rules:
- apiGroups: [""]
```



```
resources: ["pods"]                    # 可以访问pod
verbs: ["get", "list"]                 # 可以执行GET、LIST操作
```

- apiGroups表示资源所在的API分组。
- resources表示可以操作哪些资源：pods表示可以操作Pod，其他Kubernetes的资源如deployments、configmaps等都可以操作
- verbs表示可以执行的操作：get表示查询一个Pod，list表示查询所有Pod。您还可以使用create（创建），update（更新），delete（删除）等操作词。

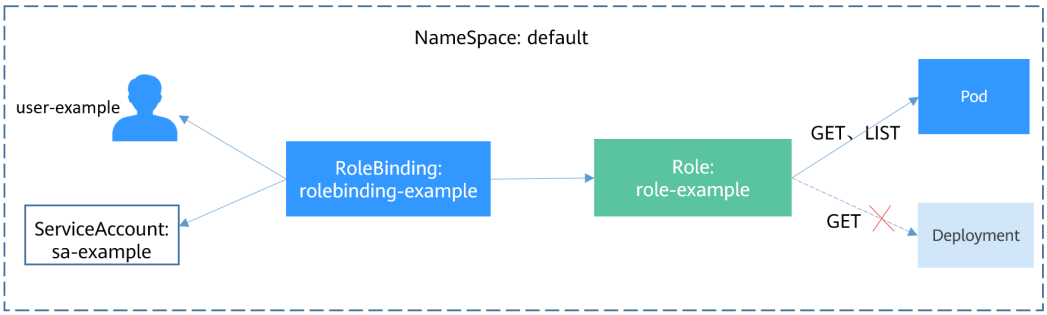
详细的类型和操作请参见**使用 RBAC 鉴权**。

有了Role之后，就可以将Role与具体的用户绑定起来，实现这个的就是RoleBinding了。如下所示。

```
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-example的用户ID
  apiGroup: rbac.authorization.k8s.io
```

这里的subjects就是将Role与IAM用户绑定起来，从而使得IAM用户获取role-example这个Role里面定义的权限，如下图所示。

图 11-6 RoleBinding 绑定 Role 和用户



subjects下用户的类型还可以是用户组，这样配置可以对用户组下所有用户生效。

```
subjects:
- kind: Group
  name: 0c96fad22880f32a3f84c009862af6f7    # 用户组ID
  apiGroup: rbac.authorization.k8s.io
```

使用IAM用户user-example连接集群，获取Pod信息，发现可获取到Pod的信息。

```
# 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
```

然后查看Deployment和Service，发现没有权限；再查询kube-system命名空间下的Pod信息，发现也没有权限。这就说明IAM用户user-example仅拥有default这个命名空间下GET和LIST Pod的权限，与前面定义的没有偏差。

```
# 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"
```

示例：授予集群管理员权限（cluster-admin）

集群全部权限可以使用cluster-admin权限，cluster-admin包含集群所有资源的权限。

图 11-7 授予集群管理员权限（cluster-admin）

如果使用kubectl查看可以看到创建了一个ClusterRoleBinding，将cluster-admin和cce-role-group这个用户组绑定了起来。

```
# 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
```

使用被授予用户连接集群，如果能正常查询PV、StorageClass的信息，则说明权限配置正常。

```
# 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
```

示例：授予命名空间运维权限（admin）

admin权限拥有命名空间大多数资源的读写权限，您可以授予用户/用户组全部命名空间admin权限。

图 11-8 授予全部命名空间运维权限（admin）

如果使用kubectl查看可以看到创建了一个RoleBinding，将admin和cce-role-group这个用户组绑定了起来。

```
# 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
```

使用被授予用户连接集群，如果能正常查询PV、StorageClass的信息，但无法创建命名空间，则说明权限配置正常。

```
# 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
```

示例：授予命名空间开发权限（edit）

edit权限拥有命名空间大多数资源的读写权限，您可以授予用户/用户组全部命名空间edit权限。

图 11-9 授予 default 命名空间开发权限（edit）

如果使用kubectl查看可以看到创建了一个RoleBinding，将edit和cce-role-group这个用户组绑定了起来，且权限范围是default这个命名空间。

```
# 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
```

使用被授予用户连接集群，您会发现可以查询和创建default命名空间的资源，但无法查询kube-system命名空间资源，也无法查询集群级别的资源。

```
# 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
```

示例：授予命名空间只读权限（view）

view权限拥有命名空间查看权限，您可以给某个或全部命名空间授权。

图 11-10 授予 default 命名空间只读权限（view）

如果使用kubectl查看可以看到创建了一个RoleBinding，将view和cce-role-group这个用户组绑定了起来，且权限范围是default这个命名空间。

```
# 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
```

使用被授予用户连接集群，您会发现可以查询default命名空间的资源，但无法创建资源。

```
# 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"
```

示例：授予某类 Kubernetes 资源权限

上面几个示例都是集群全部资源（cluster-admin）、命名空间全部资源（admin、view），也可以对某类Kubernetes资源授权，如Pod、Deployment、Service这些资源，具体请参见**自定义命名空间权限（kubectl）**。

12 配置中心

12.1 集群配置概览

集群配置中心为您提供集群的基础配置信息，包含**集群信息**、**集群配置**和**已安装插件**多维度的信息概况。

功能入口

步骤1 登录CCE控制台，单击集群名称进入集群详情页。

步骤2 在左侧导航栏中选择“配置中心”，单击“配置概览”页签。

----结束

集群信息

集群信息包括多个维度：

- **集群ID**：集群资源唯一标识，创建成功后自动生成，可用于API接口调用等场景。
- **集群名称**：集群创建成功后可以单击 进行修改集群名称。
- **集群原名**：修改集群名称后显示的集群原名，设置其他集群的名称时和该集群的原名同样不可以重复。
- **集群状态**：当前集群的运行状态，详情请参见**集群生命周期**。
- **集群类别**：显示当前集群为CCE Autopilot集群。
- **集群创建时间**：显示集群创建的时间，CCE以该时间作为计费依据。

集群配置

集群创建完成后，基本配置如下：

- **集群版本 | 补丁版本**：显示当前集群对应的Kubernetes版本号以及CCE Autopilot集群的补丁版本号。
- **网络模型**：显示当前集群的网络模型，仅支持云原生网络2.0。
- **计费模式**：显示当前集群的计费模式，仅支持按需计费。

- **企业项目**：显示集群所属的企业项目。了解更多企业项目相关信息，请查看**企业管理**。
- **操作保护**：开启操作保护后，需通过虚拟MFA、手机短信或邮箱等再次确认当前操作。请前往“IAM 服务 > 安全设置 > 敏感操作”开启操作保护。
- **资源标签**：对资源进行自定义标记，实现资源的分类。
- **集群描述**：添加自定义的集群描述，支持200个英文字符。

已安装插件

该模块可以查看集群中已安装的插件。当集群中存在可以升级的插件时，请单击“前往升级”，在插件中心页面进行查看。右上角单击“全部插件”，可以在“插件中心”页面查看更详细信息。

12.2 集群访问配置

访问方式

- **kubectl**：您需要先下载kubectl以及kubeconfig配置文件，完成配置后，即可以使用kubectl访问Kubernetes集群。详情请参见**通过kubectl连接集群**。
- **公网地址**：为Kubernetes集群的API Server绑定弹性公网IP。配置完成后，集群API Server将具有公网访问能力。

  说明

  - 绑定弹性公网IP后，集群存在公网访问风险，建议为控制节点安全组加固5443端口的入方向规则。详情请参见**如何配置集群的访问策略**。
  - 此操作将会短暂重启 kube-apiserver 并更新集群访问证书（ kubeconfig ），请避免在此期间操作集群。

认证鉴权

CCE Autopilot集群支持下载X509证书，证书中包含client.key、client.crt、ca.crt三个文件，请妥善保管您的证书，不要泄露。

如需使用证书访问集群，请参考**通过X509证书连接集群**。

12.3 网络配置

网络配置支持为您的集群配置容器网段。

集群网络配置

表 12-1 集群网络配置参数说明

参数名称	参数说明
虚拟私有云	显示集群所在虚拟私有云。 虚拟私有云（Virtual Private Cloud，简称VPC）可以为云服务器、云容器和云数据库等资源构建隔离的的虚拟网络环境，用户可以自主配置和管理。您可以自由配置VPC内的IP地址段、子网和安全组等子服务，也可以申请弹性带宽和弹性公网IP搭建业务系统。
虚拟私有云网段	显示虚拟私有云网段。
容器网络模型	显示集群的容器网络模型，仅支持云原生网络2.0。

服务配置

服务网段：用于配置集群的ClusterIP类型服务的IP地址范围。服务网段不能和容器子网网段重叠，并且只在集群内使用，不能在集群外使用。

容器网络配置

如果需要为命名空间或工作负载设置不同的容器子网或安全组，您可以创建一条策略，将容器子网/安全组与命名空间或工作负载进行绑定，详情请参见**使用容器网络配置为命名空间/工作负载绑定子网及安全组**。

- 容器绑定子网：Pod IP会被约束在特定的网段中，不同命名空间或工作负载之间可实现网络隔离。
- 容器绑定安全组：支持为同一命名空间或工作负载的Pod设置安全组规则，实现自定义访问策略。

说明

1.27.8-r0，1.28.6-r0及以上版本的CCE Autopilot集群支持删除容器子网。

12.4 监控运维配置

CCE Autopilot集群为您提供监控应用及资源的能力，支持采集各项指标及事件等数据以分析应用健康状态，您可以通过“配置中心 > 监控运维配置”统一调整监控运维参数。

您需要**开通监控中心**，以使用监控运维配置的所有功能。

监控配置

采集配置

默认采集周期：云原生监控插件的指标采集周期，默认为15秒。

对接AOM监控服务

AOM实例是应用运维管理服务（AOM）推出的Prometheus监控功能。“普罗数据上报至AOM服务”开启后，指标会上报到您选择的AOM实例，其中容器基础指标免费，其他指标按需收费。了解免费指标的更多信息，请参见**基础指标-容器指标**。

日志配置

采集配置

CCE Autopilot集群可以帮助您快速采集Kubernetes集群的日志信息，包括容器日志和Kubernetes事件。通过采集的日志信息，您可以快速进行日志的查询与分析。

说明

日志上报LTS会创建名为k8s-logs-{clusterId}的默认日志组，并收取相关的费用。LTS收费标准请参见**价格计算器**。

表 12-2 容器

日志类型	日志	LTS日志流名称	开启状态	参考文档
容器日志	容器标准输出	stdout-{clusterId}	开通业务日志采集需安装**云原生日志采集插件**。	**收集容器日志**
Kubernetes事件	Kubernetes事件	event-{clusterId}	开通业务日志采集需安装**云原生日志采集插件**。	**收集Kubernetes事件**

Kubernetes事件上报至AOM

集群中安装**云原生日志采集插件**后，Kubernetes事件默认上报至LTS，您可以通过该配置将Kubernetes事件上报至AOM。

- 异常事件上报：默认开启，会将所有异常事件上报至AOM。您可以单击“配置黑名单”，将不需要上报的事件添加至黑名单进行管理，其中“事件名称”可通过**CCE Autopilot集群事件列表**查询。
- 普通事件上报：开启后，会将普通事件上报至AOM，系统默认配置了部分需要上报的普通事件。如果您需要自定义上报的事件，可以单击“配置白名单”，将需要上报添加至白名单进行管理，其中“事件名称”可通过**CCE Autopilot集群事件列表**查询。