Cloud Container Instance (CCI)

Best Practices

Issue	01
Date	2025-08-12

HUAWEI CLOUD COMPUTING TECHNOLOGIES CO., LTD.

Trademarks and Permissions


and other Huawei trademarks are the property of Huawei Technologies Co., Ltd.
All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between Huawei Cloud and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Huawei Cloud Computing Technologies Co., Ltd.

Address: Huawei Cloud Data Center Jiaoxinggong Road
Qianzhong Avenue
Gui'an New District
Gui Zhou 550029
People's Republic of China

Website: https://www.huaweicloud.com/intl/en-us/

Contents

1 Workload Creation

1.1 Deploying WordPress Using the CCI Console

Cloud Container Instance (CCI) provides a serverless container engine, eliminating the need to manage clusters or servers. CCI delivers container agility and high performance with only three steps. CCI enables you to create Deployments. It enhances container security isolation and supports fast workload deployment, elastic load balancing, and auto scaling based on the Kubernetes workload model.

Creating a Namespace

Step 1 Log in to the **CCI 2.0 console**.

Step 2 In the navigation pane, choose **Namespaces**.

Step 3 On the **Namespaces** page, click **Create Namespace** in the upper right corner.

Step 4 Enter a name for the namespace.

NOTE

- The name of each namespace must be unique.
- Enter 1 to 63 characters starting and ending with a lowercase letter or digit. Only lowercase letters, digits, and hyphens (-) are allowed.

Step 5 (Optional) Specify monitoring settings.

Parameter	Description
AOM (Optional)	If this option is enabled, you need to select an AOM instance.

Step 6 Configure the network plane.

Table 1-1 Network plane settings

Parameter	Description
IPv6	If this option is enabled, IPv4/IPv6 dual stack is supported.
VPC	Select the VPC where the workloads are running. If no VPC is available, create one first. The VPC cannot be changed once selected. Recommended CIDR blocks: 10.0.0.0/8-22, 172.16.0.0/12-22, and 192.168.0.0/16-22 **NOTICE** • You cannot set the VPC CIDR block and subnet CIDR block to 10.247.0.0/16, because this CIDR block is reserved for workloads. If you select this CIDR block, there may be IP address conflicts, which may result in workload creation failure or service unavailability. If you do not need to access pods through workloads, you can select this CIDR block. • After the namespace is created, you can choose **Namespaces** in the navigation pane and view the VPC and subnet in the **Subnet** column.
Subnet	Select the subnet where the workloads are running. If no subnet is available, create one first. The subnet cannot be changed once selected. • A certain number of IP addresses (**10** by default) in the subnet will be warmed up for the namespace. • You can set the number of IP addresses to be warmed up in **Advanced Settings**. • If warming up IP addresses for the namespace is enabled, the VPC and subnet can only be deleted after the namespace is deleted. **NOTE** Ensure that there are sufficient available IP addresses in the subnet. If IP addresses are insufficient, workload creation will fail.
Security Group	Select a security group. If no security group is available, create one first. The security group cannot be changed once selected.

Step 7 (Optional) Specify advanced settings.

Each namespace provides an IP pool. You can specify the pool size to reduce the duration for assigning IP addresses and speed up the workload creation.

For example, 200 pods are running routinely, and 200 IP addresses are required in the IP pool. During peak hours, the IP pool instantly scales out to provide 65,535 IP addresses. After a specified interval (for example, 23 hours), the IP addresses that exceed the pool size (65535 – 200 = 65335) will be recycled.

Table 1-2 (Optional) Advanced namespace settings

Parameter	Description
IP Pool Warm-up for Namespace	• An IP pool is provided for each namespace, with the number of IP addresses you specify here. IP addresses will be assigned in advance to accelerate workload creation. • An IP pool can contain a maximum of 65,535 IP addresses. • When using general-computing pods, you are advised to configure an appropriate size for the IP pool based on service requirements to accelerate workload startup. • Configure the number of IP addresses to be assigned properly. If the number of IP addresses exceeds the number of available IP addresses in the subnet, other services will be affected.
IP Address Recycling Interval (h)	Pre-assigned IP addresses that become idle can be recycled within the duration you specify here. **NOTE** Recycling mechanism: • Recycling time: The **yangtse.io/warm-pool-recycle-interval** field configured on the network determines when the IP addresses can be recycled. If **yangtse.io/warm-pool-recycle-interval** is set to **24**, the IP addresses can only be recycled 24 hours later. • Recycling rate: A maximum of 50 IP addresses can be recycled at a time. This prevents IP addresses from being repeatedly assigned or released due to fast or frequent recycling.

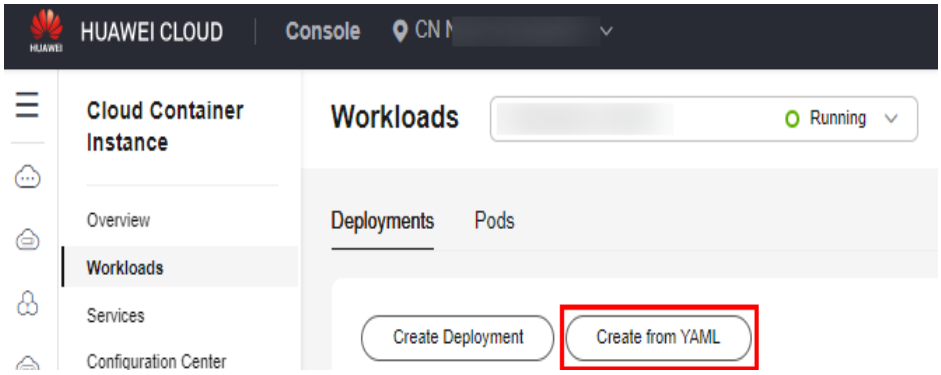
Step 8 Click **OK**.

You can view the VPC and subnet on the namespace details page.

----End

Creating a Deployment Using YAML

Step 1 Log in to the CCI console. In the navigation pane, choose **Workloads**. On the **Deployments** tab, click **Create from YAML**.



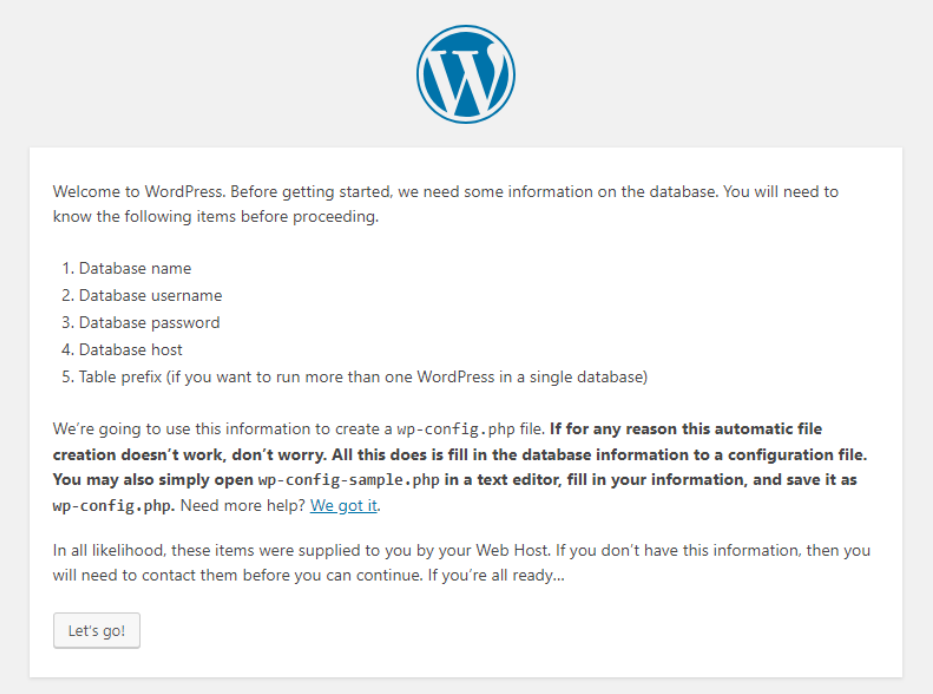
Step 2 Specify basic information. The following is an example YAML file:

```
kind: Deployment
apiVersion: cci/v2
metadata:
  name: wordpress
spec:
  replicas: 1
  selector:
    matchLabels:
      app: wordpress
  template:
    metadata:
      labels:
        app: wordpress
    spec:
      containers:
        - name: wordpress
          image: wordpress:latest
          ports:
          - containerPort: 80
          resources:
            limits:
              cpu: 500m
              memory: 1Gi
            requests:
              cpu: 500m
              memory: 1Gi
      dnsPolicy: Default
```

Step 3 Create a Service of the LoadBalancer type. The load balancer associated with the Service must have an EIP bound. For details, see **Public Network Access**. The following is an example YAML file:

```
kind: Service
apiVersion: cci/v2
metadata:
  name: service-wordpress
  annotations:
    kubernetes.io/elb.class: elb
    kubernetes.io/elb.id: '${elb_id}'
spec:
  ports:
    - name: service-wordpress-port
      protocol: TCP
      port: 80
      targetPort: 80
  selector:
    app: wordpress
  type: LoadBalancer
```

Step 4 Use a browser to access the EIP in the access address.



----End

1.2 Configuring Dockerfile Parameters for CCI

Scenario

A Dockerfile is often used to build an image. A Dockerfile is a text document that contains all the commands a user could call on the command line to assemble an image.

This topic describes how to apply the Dockerfile configurations in CCI.

Using Dockerfile Parameters in CCI

The following uses an example to describe the relationship between Dockerfile parameters and CCI.

```
FROM ubuntu:16.04

ENV VERSION 1.0

VOLUME /var/lib/app

EXPOSE 80

ENTRYPOINT ["./entrypoint.sh"]
CMD ["start"]
```

In this example, the Dockerfile contains common parameters, including **ENV**, **VOLUME**, **EXPOSE**, **ENTRYPOINT**, and **CMD**. These parameters can be configured for CCI as follows:

```
kind: Deployment
apiVersion: cci/v2
metadata:
```



```
  name: wordpress
spec:
  replicas: 1
  selector:
    matchLabels:
      app: wordpress
  template:
    metadata:
      labels:
        app: wordpress
    spec:
      containers:
      - name: wordpress
        image: wordpress:latest
        command: ["/entrypoint.sh"] # ENTRYPOINT
        args: ["start"] # CMD
        ports: # EXPOSE
        - containerPort: 80
        env: # ENV
        - name: VERSION
          value: "1.0"
        resources:
          limits:
            cpu: 500m
            memory: 1Gi
          requests:
            cpu: 500m
            memory: 1Gi
        volumeMounts: # VOLUME
        - name: cache-volume
          mountPath: /cache
      volumes:
      - name: cache-volume
        emptyDir:
          sizeLimit: 1Gi
          medium: Memory
      dnsPolicy: Default
```


2 Workload Management

2.1 Performing Graceful Rolling Upgrades for CCI Applications

Scenario

When you deploy a workload in CCI to run an application, the application is exposed through a LoadBalancer Service, which has a dedicated load balancer associated to allow traffic to the containers. When there is a rolling upgrade or auto scaling, the pods may fail to work with the load balancer, and 5xx errors are returned. This topic guides you to configure container probes and readiness time for graceful rolling upgrades and auto scaling.

Table 2-1 Scenarios

Scenario	Solution
Deployment upgrade	Rolling upgrade + Liveness or readiness probe + Graceful termination
Deployment scale-in	Liveness or readiness probe + Graceful termination

- Rolling upgrade

  Perform a rolling upgrade to update the pods for the Deployment. In this mode, pods are updated one by one, not all at once. In this way, you can control the update speed and the number of concurrent pods to ensure that services are not interrupted during the upgrade. For example, you can configure the **maxSurge** and **maxUnavailable** parameters to control the number of new pods and the number of old pods concurrently. Ensure that there is always a workload that can provide services during the upgrade.

  Here is an example. Suppose that **maxSurge** and **maxUnavailable** are set to **25%** and the number of replicas of a Deployment is 2. In the actual upgrade, **maxSurge** allows a maximum of three pods (rounded up to 3 from 2 × 1.25 = 2.5), and **maxUnavailable** does not allow any unavailable pods (rounded

down to 0 from 2 × 0.25 = 0.5). This means that two pods are running during the upgrade. Each time a pod is created, an old pod will be deleted until all pods are replaced by new ones.

- Liveness or readiness probe

  A liveness probe can be used to determine when to restart a container. If the liveness check of a container fails, CCI restarts the container to improve application availability.

  A readiness probe can know when the container is ready to receive traffic. If the pod is not ready, it will be disassociated from the load balancer.

- Graceful termination

  During workload upgrade or scale-in, the pod is disassociated from the load balancer, but the connections of the requests that are still being handled will be retained. If the backend service pods exit immediately after receiving the end signal, the in-progress requests may fail or some traffic may be forwarded to pods that have already exited, leading to traffic loss. To avoid this problem, you are advised to configure the graceful termination time (**terminationGracePeriodSeconds**) and PreStop hook for the pod.

  The default value of **terminationGracePeriodSeconds** is 30 seconds. When a pod is deleted, the SIGTERM signal is sent, and the system waits for the application in the container to terminate. If the application is not terminated within the period specified by **terminationGracePeriodSeconds**, an SIGKILL signal is sent to forcibly terminate the application.

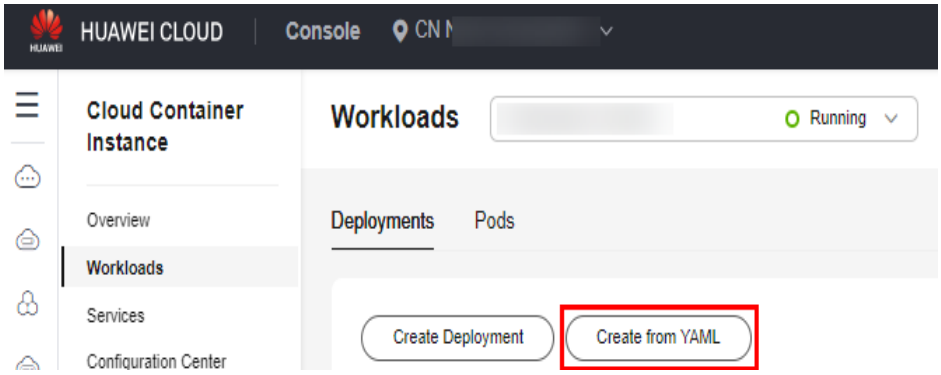
  If the SIGTERM signal is not handled in the service code, you can configure a PreStop hook for the pod to continue working for a period of time after the pod is removed to avoid traffic loss.

  The SIGTERM signal should be handled in the service code for graceful termination. You can also use a PreStop to make the pod to sleep for a while and stop the processes in the container until kube-proxy completes rule synchronization. Because the sum of the time required by the PreStop hook and the time when the processes are stopped may exceed 30s, you can specify **terminationGracePeriodSeconds** as needed to prevent the pod from being forcibly stopped by the SIGKILL signal too early.

Procedure

An Nginx Deployment is used as an example to describe how to perform a graceful rolling upgrade or auto scaling in CCI.

Step 1 Log in to the CCI console. In the navigation pane, choose **Workloads**. On the **Deployments** tab, click **Create from YAML**.



Step 2 Create a Deployment. The following is an example YAML file:

```
kind: Deployment
apiVersion: cci/v2
metadata:
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:latest
          ports:
          - containerPort: 80
          resources:
            limits:
              cpu: 500m
              memory: 1Gi
            requests:
              cpu: 500m
              memory: 1Gi
          livenessProbe:           # Liveness probe
            httpGet:               # An HTTP request is used to check the containers.
              path: /              # The HTTP health check path is /.
              port: 80                 # The health check port is 80.
            initailDelySeconds: 5
            periodSeconds: 5
          readinessProbe:          # Readiness probe. This is used to check whether the container is ready. If the container is not ready, traffic is not forwarded to it.
            httpGet:
              path: /
              port: 80
            initialDelaySeconds: 5
            periodSeconds: 5
          lifecycle:
            preStop:               # PreStop hook. This configuration ensures that the container can provide services for external systems during the exit.
              exec:
                command:
                - /bin/bash
                - '-c'
                - sleep 30
      dnsPolicy: Default
      imagePullSecrets:
        - name: imagepull-secret
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxUnavailable: 0          # Maximum number of unavailable pods that can be tolerated during the rolling upgrade
      maxSurge: 100%             # Percentage of redundant pods during the rolling upgrade
  minReadySeconds: 10            # Minimum readiness time. A pod is considered available only when the minimum readiness time is exceeded without any of its containers crashing.
```

NOTE

- The recommended value of **minReadySeconds** is the sum of the expected time for starting the container and the duration from the time when the pod is associated with the load balancer to the time when the pod receives the traffic.
- The value of **minReadySeconds** must be smaller than that of **sleep** to ensure that the new container is ready before the old container stops and exits.

Step 3 Test the upgrade and auto scaling.

Prepare a client outside the cluster and configure the script **detection_script.sh** with the following content (**100.85.125.90:80** indicates the public network address for accessing the Service):

```
#! /bin/bash
for (( ; ; ))
do
        curl -I 100.85.125.90:80 | grep "200 OK"
        if [ $? -ne 0 ]; then
                echo "response error!"
                exit 1
        fi
done
```

Step 4 Run the script **bash detection_script.sh**. Log in to the CCI console. In the navigation pane, choose **Workloads**. On the **Deployments** tab, select the target workload and click **Edit YAML** to trigger a rolling upgrade.

If the access to the application is not interrupted and the returned responses are all **200 OK**, a graceful upgrade is performed.

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0   612    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
HTTP/1.1 200 OK
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0   612    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
HTTP/1.1 200 OK
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0   612    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
HTTP/1.1 200 OK
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0   612    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
HTTP/1.1 200 OK
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0   612    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
HTTP/1.1 200 OK
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0   612    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
HTTP/1.1 200 OK
```

----End

2.2 Exposing Basic Pod Information to Containers Through Environment Variables

If you want a pod to expose its basic information to the container running in the pod, you can use the Kubernetes ***Downward API*** to inject environment variables. This topic describes how to add environment variables to the definition of a Deployment or a pod to obtain the namespace, name, UID, IP address, region, and AZ of the pod.

When CCI creates a pod and allocates it to a node, the region and AZ information of the node is added to the pod's annotations.

In this case, the format of the pod's annotations is as follows:

```
apiVersion: cci/v2
 kind: Pod
 metadata:
   annotations:
     topology.kubernetes.io/region: "{{region}}"  # Region of the node
     topology.kubernetes.io/zone: "{{available-zone}}" #AZ of the node
```

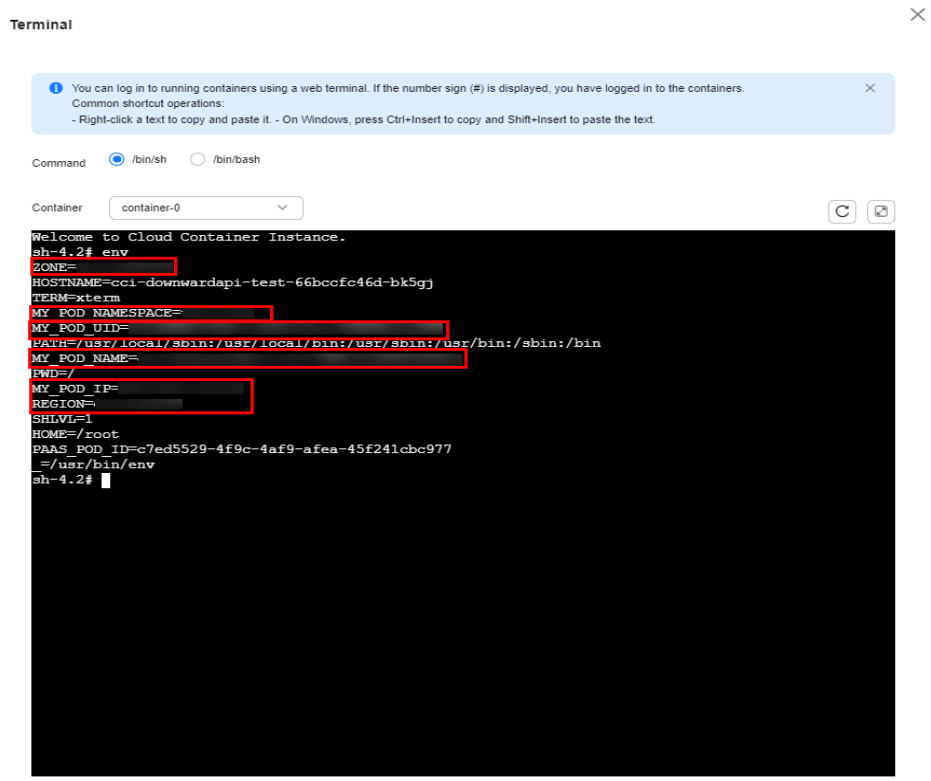
Deployment Configuration Example

The following example shows how to use environment variables to obtain basic pod information.

```
kind: Deployment
apiVersion: cci/v2
metadata:
  name: cci-downwardapi-test
  namespace: cci-test # Enter a specific namespace.
spec:
  replicas: 2
  selector:
    matchLabels:
      app: cci-downwardapi-test
  template:
    metadata:
      labels:
        app: cci-downwardapi-test
    spec:
      containers:
        - name: container-0
          image: 'library/euleros:latest'
          command:
            - /bin/bash
            - '-c'
            - while true; do echo hello; sleep 10; done
          env:
          - name: MY_POD_UID
            valueFrom:
              fieldRef:
                fieldPath: metadata.uid
          - name: MY_POD_NAME
            valueFrom:
              fieldRef:
                fieldPath: metadata.name
          - name: MY_POD_NAMESPACE
            valueFrom:
              fieldRef:
                fieldPath: metadata.namespace
          - name: MY_POD_IP
            valueFrom:
              fieldRef:
                fieldPath: status.podIP
          - name: REGION
            valueFrom:
              fieldRef:
                fieldPath: metadata.annotations['topology.kubernetes.io/region']
          - name: ZONE
            valueFrom:
              fieldRef:
                fieldPath: metadata.annotations['topology.kubernetes.io/zone']
          resources:
            limits:
              cpu: 500m
              memory: 1Gi
            requests:
              cpu: 500m
              memory: 1Gi
```


When the Deployment starts, you can view the pod information exposed to the container through environment variables.

Figure 2-1 Basic pod Information



2.3 Configuring Kernel Parameters

CCI provides an industry-leading serverless container platform. Kernel parameter optimization is a common practice in advanced service deployment scenarios. In a safe situation, CCI allows you to configure kernel parameters through a security context of a pod based on the solution recommended by the Kubernetes community, greatly improving the flexibility of service deployment. For more information, see ***Security Context***.

In Linux, kernel parameters are usually configured through the sysctl interface. In Kubernetes, kernel parameters are configured through the sysctl security context of the pod. For more information of sysctl, see ***Using sysctls in a Kubernetes Cluster***. The security context is applied to all containers in the pod.

CCI allows you to modify the following kernel parameters:

```
"kernel.shm_rmid_forced",
"kernel.shmall",
"kernel.shmmax",
"kernel.shmmni",
"kernel.msgmax",
"kernel.msgmnb",
"kernel.msgmni",
"kernel.sem",
"fs.mqueue.msg_default",
"fs.mqueue.msg_max",
"fs.mqueue.msgsize_default",
```



```
"fs.mqueue.msgsize_max",
"fs.mqueue.queues_max",
"net.core.busy_poll",
"net.core.busy_read",
"net.core.default_qdisc",
"net.core.dev_weight",
"net.core.dev_weight_rx_bias",
"net.core.dev_weight_tx_bias",
"net.core.fb_tunnels_only_for_init_net",
"net.core.flow_limit_cpu_bitmap",
"net.core.flow_limit_table_len",
"net.core.max_skb_frags",
"net.core.message_burst",
"net.core.message_cost",
"net.core.netdev_budget",
"net.core.netdev_budget_usecs",
"net.core.netdev_max_backlog",
"net.core.netdev_rss_key",
"net.core.netdev_tstamp_prequeue",
"net.core.optmem_max",
"net.core.rmem_default",
"net.core.rmem_max",
"net.core.rps_sock_flow_entries",
"net.core.somaxconn",
"net.core.tstamp_allow_data",
"net.core.warnings",
"net.core.wmem_default",
"net.core.wmem_max",
"net.core.xfrm_acq_expires",
"net.core.xfrm_aevent_etime",
"net.core.xfrm_aevent_rseqth",
"net.core.xfrm_larval_drop",
"net.ipv4.conf.all.accept_local",
"net.ipv4.conf.all.accept_redirects",
"net.ipv4.conf.all.accept_source_route",
"net.ipv4.conf.all.arp_accept",
"net.ipv4.conf.all.arp_announce",
"net.ipv4.conf.all.arp_filter",
"net.ipv4.conf.all.arp_ignore",
"net.ipv4.conf.all.arp_notify",
"net.ipv4.conf.all.bc_forwarding",
"net.ipv4.conf.all.bootp_relay",
"net.ipv4.conf.all.disable_policy",
"net.ipv4.conf.all.disable_xfrm",
"net.ipv4.conf.all.drop_gratuitous_arp",
"net.ipv4.conf.all.drop_unicast_in_l2_multicast",
"net.ipv4.conf.all.force_igmp_version",
"net.ipv4.conf.all.forwarding",
"net.ipv4.conf.all.igmpv2_unsolicited_report_interval",
"net.ipv4.conf.all.igmpv3_unsolicited_report_interval",
"net.ipv4.conf.all.ignore_routes_with_linkdown",
"net.ipv4.conf.all.log_martians",
"net.ipv4.conf.all.mc_forwarding",
"net.ipv4.conf.all.medium_id",
"net.ipv4.conf.all.promote_secondaries",
"net.ipv4.conf.all.proxy_arp",
"net.ipv4.conf.all.proxy_arp_pvlan",
"net.ipv4.conf.all.route_localnet",
"net.ipv4.conf.all.rp_filter",
"net.ipv4.conf.all.secure_redirects",
"net.ipv4.conf.all.send_redirects",
"net.ipv4.conf.all.shared_media",
"net.ipv4.conf.all.src_valid_mark",
"net.ipv4.conf.all.tag",
"net.ipv4.conf.default.accept_local",
"net.ipv4.conf.default.accept_redirects",
"net.ipv4.conf.default.accept_source_route",
"net.ipv4.conf.default.arp_accept",
"net.ipv4.conf.default.arp_announce",
```



```
            "net.ipv4.conf.default.arp_filter",
            "net.ipv4.conf.default.arp_ignore",
            "net.ipv4.conf.default.arp_notify",
            "net.ipv4.conf.default.bc_forwarding",
            "net.ipv4.conf.default.bootp_relay",
            "net.ipv4.conf.default.disable_policy",
            "net.ipv4.conf.default.disable_xfrm",
            "net.ipv4.conf.default.drop_gratuitous_arp",
            "net.ipv4.conf.default.drop_unicast_in_l2_multicast",
            "net.ipv4.conf.default.force_igmp_version",
            "net.ipv4.conf.default.forwarding",
            "net.ipv4.conf.default.igmpv2_unsolicited_report_interval",
            "net.ipv4.conf.default.igmpv3_unsolicited_report_interval",
            "net.ipv4.conf.default.ignore_routes_with_linkdown",
            "net.ipv4.conf.default.log_martians",
            "net.ipv4.conf.default.mc_forwarding",
            "net.ipv4.conf.default.medium_id",
            "net.ipv4.conf.default.promote_secondaries",
            "net.ipv4.conf.default.proxy_arp",
            "net.ipv4.conf.default.proxy_arp_pvlan",
            "net.ipv4.conf.default.route_localnet",
            "net.ipv4.conf.default.rp_filter",
            "net.ipv4.conf.default.secure_redirects",
            "net.ipv4.conf.default.send_redirects",
            "net.ipv4.conf.default.shared_media",
            "net.ipv4.conf.default.src_valid_mark",
            "net.ipv4.conf.default.tag",
            "net.ipv4.conf.eth0.accept_local",
            "net.ipv4.conf.eth0.accept_redirects",
            "net.ipv4.conf.eth0.accept_source_route",
            "net.ipv4.conf.eth0.arp_accept",
            "net.ipv4.conf.eth0.arp_announce",
            "net.ipv4.conf.eth0.arp_filter",
            "net.ipv4.conf.eth0.arp_ignore",
            "net.ipv4.conf.eth0.arp_notify",
            "net.ipv4.conf.eth0.bc_forwarding",
            "net.ipv4.conf.eth0.bootp_relay",
            "net.ipv4.conf.eth0.disable_policy",
            "net.ipv4.conf.eth0.disable_xfrm",
            "net.ipv4.conf.eth0.drop_gratuitous_arp",
            "net.ipv4.conf.eth0.drop_unicast_in_l2_multicast",
            "net.ipv4.conf.eth0.force_igmp_version",
            "net.ipv4.conf.eth0.forwarding",
            "net.ipv4.conf.eth0.igmpv2_unsolicited_report_interval",
            "net.ipv4.conf.eth0.igmpv3_unsolicited_report_interval",
            "net.ipv4.conf.eth0.ignore_routes_with_linkdown",
            "net.ipv4.conf.eth0.log_martians",
            "net.ipv4.conf.eth0.mc_forwarding",
            "net.ipv4.conf.eth0.medium_id",
            "net.ipv4.conf.eth0.promote_secondaries",
            "net.ipv4.conf.eth0.proxy_arp",
            "net.ipv4.conf.eth0.proxy_arp_pvlan",
            "net.ipv4.conf.eth0.route_localnet",
            "net.ipv4.conf.eth0.rp_filter",
            "net.ipv4.conf.eth0.secure_redirects",
            "net.ipv4.conf.eth0.send_redirects",
            "net.ipv4.conf.eth0.shared_media",
            "net.ipv4.conf.eth0.src_valid_mark",
            "net.ipv4.conf.eth0.tag",
            "net.ipv4.conf.lo.accept_local",
            "net.ipv4.conf.lo.accept_redirects",
            "net.ipv4.conf.lo.accept_source_route",
            "net.ipv4.conf.lo.arp_accept",
            "net.ipv4.conf.lo.arp_announce",
            "net.ipv4.conf.lo.arp_filter",
            "net.ipv4.conf.lo.arp_ignore",
            "net.ipv4.conf.lo.arp_notify",
            "net.ipv4.conf.lo.bc_forwarding",
            "net.ipv4.conf.lo.bootp_relay",
```



```
"net.ipv4.conf.lo.disable_policy",
"net.ipv4.conf.lo.disable_xfrm",
"net.ipv4.conf.lo.drop_gratuitous_arp",
"net.ipv4.conf.lo.drop_unicast_in_l2_multicast",
"net.ipv4.conf.lo.force_igmp_version",
"net.ipv4.conf.lo.forwarding",
"net.ipv4.conf.lo.igmpv2_unsolicited_report_interval",
"net.ipv4.conf.lo.igmpv3_unsolicited_report_interval",
"net.ipv4.conf.lo.ignore_routes_with_linkdown",
"net.ipv4.conf.lo.log_martians",
"net.ipv4.conf.lo.mc_forwarding",
"net.ipv4.conf.lo.medium_id",
"net.ipv4.conf.lo.promote_secondaries",
"net.ipv4.conf.lo.proxy_arp",
"net.ipv4.conf.lo.proxy_arp_pvlan",
"net.ipv4.conf.lo.route_localnet",
"net.ipv4.conf.lo.rp_filter",
"net.ipv4.conf.lo.secure_redirects",
"net.ipv4.conf.lo.send_redirects",
"net.ipv4.conf.lo.shared_media",
"net.ipv4.conf.lo.src_valid_mark",
"net.ipv4.conf.lo.tag",
"net.ipv4.fwmark_reflect",
"net.ipv4.icmp_echo_ignore_all",
"net.ipv4.icmp_echo_ignore_broadcasts",
"net.ipv4.icmp_errors_use_inbound_ifaddr",
"net.ipv4.icmp_ignore_bogus_error_responses",
"net.ipv4.icmp_msgs_burst",
"net.ipv4.icmp_msgs_per_sec",
"net.ipv4.icmp_ratelimit",
"net.ipv4.icmp_ratemask",
"net.ipv4.igmp_link_local_mcast_reports",
"net.ipv4.igmp_max_memberships",
"net.ipv4.igmp_max_msf",
"net.ipv4.igmp_qrv",
"net.ipv4.inet_peer_maxttl",
"net.ipv4.inet_peer_minttl",
"net.ipv4.inet_peer_threshold",
"net.ipv4.ip_default_ttl",
"net.ipv4.ip_dynaddr",
"net.ipv4.ip_early_demux",
"net.ipv4.ip_forward",
"net.ipv4.ip_forward_update_priority",
"net.ipv4.ip_forward_use_pmtu",
"net.ipv4.ip_local_port_range",
"net.ipv4.ip_local_reserved_ports",
"net.ipv4.ip_no_pmtu_disc",
"net.ipv4.ip_nonlocal_bind",
"net.ipv4.ip_unprivileged_port_start",
"net.ipv4.ipfrag_high_thresh",
"net.ipv4.ipfrag_low_thresh",
"net.ipv4.ipfrag_max_dist",
"net.ipv4.ipfrag_secret_interval",
"net.ipv4.ipfrag_time",
"net.ipv4.neigh.default.anycast_delay",
"net.ipv4.neigh.default.app_solicit",
"net.ipv4.neigh.default.base_reachable_time",
"net.ipv4.neigh.default.base_reachable_time_ms",
"net.ipv4.neigh.default.delay_first_probe_time",
"net.ipv4.neigh.default.gc_interval",
"net.ipv4.neigh.default.gc_stale_time",
"net.ipv4.neigh.default.gc_thresh1",
"net.ipv4.neigh.default.gc_thresh2",
"net.ipv4.neigh.default.gc_thresh3",
"net.ipv4.neigh.default.locktime",
"net.ipv4.neigh.default.mcast_resolicit",
"net.ipv4.neigh.default.mcast_solicit",
"net.ipv4.neigh.default.proxy_delay",
"net.ipv4.neigh.default.proxy_qlen",
```



```
"net.ipv4.neigh.default.retrans_time",
"net.ipv4.neigh.default.retrans_time_ms",
"net.ipv4.neigh.default.ucast_solicit",
"net.ipv4.neigh.default.unres_qlen",
"net.ipv4.neigh.default.unres_qlen_bytes",
"net.ipv4.neigh.eth0.anycast_delay",
"net.ipv4.neigh.eth0.app_solicit",
"net.ipv4.neigh.eth0.base_reachable_time",
"net.ipv4.neigh.eth0.base_reachable_time_ms",
"net.ipv4.neigh.eth0.delay_first_probe_time",
"net.ipv4.neigh.eth0.gc_stale_time",
"net.ipv4.neigh.eth0.locktime",
"net.ipv4.neigh.eth0.mcast_resolicit",
"net.ipv4.neigh.eth0.mcast_solicit",
"net.ipv4.neigh.eth0.proxy_delay",
"net.ipv4.neigh.eth0.proxy_qlen",
"net.ipv4.neigh.eth0.retrans_time",
"net.ipv4.neigh.eth0.retrans_time_ms",
"net.ipv4.neigh.eth0.ucast_solicit",
"net.ipv4.neigh.eth0.unres_qlen",
"net.ipv4.neigh.eth0.unres_qlen_bytes",
"net.ipv4.neigh.lo.anycast_delay",
"net.ipv4.neigh.lo.app_solicit",
"net.ipv4.neigh.lo.base_reachable_time",
"net.ipv4.neigh.lo.base_reachable_time_ms",
"net.ipv4.neigh.lo.delay_first_probe_time",
"net.ipv4.neigh.lo.gc_stale_time",
"net.ipv4.neigh.lo.locktime",
"net.ipv4.neigh.lo.mcast_resolicit",
"net.ipv4.neigh.lo.mcast_solicit",
"net.ipv4.neigh.lo.proxy_delay",
"net.ipv4.neigh.lo.proxy_qlen",
"net.ipv4.neigh.lo.retrans_time",
"net.ipv4.neigh.lo.retrans_time_ms",
"net.ipv4.neigh.lo.ucast_solicit",
"net.ipv4.neigh.lo.unres_qlen",
"net.ipv4.neigh.lo.unres_qlen_bytes",
"net.ipv4.ping_group_range",
"net.ipv4.route.error_burst",
"net.ipv4.route.error_cost",
"net.ipv4.route.gc_elasticity",
"net.ipv4.route.gc_interval",
"net.ipv4.route.gc_min_interval",
"net.ipv4.route.gc_min_interval_ms",
"net.ipv4.route.gc_thresh",
"net.ipv4.route.gc_timeout",
"net.ipv4.route.max_size",
"net.ipv4.route.min_adv_mss",
"net.ipv4.route.min_pmtu",
"net.ipv4.route.mtu_expires",
"net.ipv4.route.redirect_load",
"net.ipv4.route.redirect_number",
"net.ipv4.route.redirect_silence",
"net.ipv4.tcp_abort_on_overflow",
"net.ipv4.tcp_adv_win_scale",
"net.ipv4.tcp_allowed_congestion_control",
"net.ipv4.tcp_app_win",
"net.ipv4.tcp_autocorking",
"net.ipv4.tcp_available_congestion_control",
"net.ipv4.tcp_available_ulp",
"net.ipv4.tcp_base_mss",
"net.ipv4.tcp_challenge_ack_limit",
"net.ipv4.tcp_comp_sack_delay_ns",
"net.ipv4.tcp_comp_sack_nr",
"net.ipv4.tcp_congestion_control",
"net.ipv4.tcp_dsack",
"net.ipv4.tcp_early_demux",
"net.ipv4.tcp_early_retrans",
"net.ipv4.tcp_ecn",
```



```
            "net.ipv4.tcp_ecn_fallback",
            "net.ipv4.tcp_fack",
            "net.ipv4.tcp_fastopen",
            "net.ipv4.tcp_fastopen_blackhole_timeout_sec",
            "net.ipv4.tcp_fastopen_key",
            "net.ipv4.tcp_fin_timeout",
            "net.ipv4.tcp_frto",
            "net.ipv4.tcp_fwmark_accept",
            "net.ipv4.tcp_invalid_ratelimit",
            "net.ipv4.tcp_keepalive_intvl",
            "net.ipv4.tcp_keepalive_probes",
            "net.ipv4.tcp_keepalive_time",
            "net.ipv4.tcp_limit_output_bytes",
            "net.ipv4.tcp_low_latency",
            "net.ipv4.tcp_max_orphans",
            "net.ipv4.tcp_max_reordering",
            "net.ipv4.tcp_max_syn_backlog",
            "net.ipv4.tcp_max_tw_buckets",
            "net.ipv4.tcp_mem",
            "net.ipv4.tcp_min_rtt_wlen",
            "net.ipv4.tcp_min_snd_mss",
            "net.ipv4.tcp_min_tso_segs",
            "net.ipv4.tcp_moderate_rcvbuf",
            "net.ipv4.tcp_mtu_probing",
            "net.ipv4.tcp_no_metrics_save",
            "net.ipv4.tcp_notsent_lowat",
            "net.ipv4.tcp_orphan_retries",
            "net.ipv4.tcp_pacing_ca_ratio",
            "net.ipv4.tcp_pacing_ss_ratio",
            "net.ipv4.tcp_probe_interval",
            "net.ipv4.tcp_probe_threshold",
            "net.ipv4.tcp_recovery",
            "net.ipv4.tcp_reordering",
            "net.ipv4.tcp_retrans_collapse",
            "net.ipv4.tcp_retries1",
            "net.ipv4.tcp_retries2",
            "net.ipv4.tcp_rfc1337",
            "net.ipv4.tcp_rmem",
            "net.ipv4.tcp_sack",
            "net.ipv4.tcp_slow_start_after_idle",
            "net.ipv4.tcp_stdurg",
            "net.ipv4.tcp_syn_retries",
            "net.ipv4.tcp_synack_retries",
            "net.ipv4.tcp_syncookies",
            "net.ipv4.tcp_thin_linear_timeouts",
            "net.ipv4.tcp_timestamps",
            "net.ipv4.tcp_tso_win_divisor",
            "net.ipv4.tcp_tw_reuse",
            "net.ipv4.tcp_window_scaling",
            "net.ipv4.tcp_wmem",
            "net.ipv4.tcp_workaround_signed_windows",
            "net.ipv4.udp_early_demux",
            "net.ipv4.udp_mem",
            "net.ipv4.udp_rmem_min",
            "net.ipv4.udp_wmem_min",
            "net.ipv4.xfrm4_gc_thresh",
            "net.nf_conntrack_max",
            "net.unix.max_dgram_qlen"
```

In the following example, the pod's **securityContext** is used to set the sysctl parameters **net.core.somaxconn** and **net.ipv4.tcp_tw_reuse**.

```
apiVersion: cci/v2
kind:Pod
metadata:
    name: xxxxx
    namespace: auto-test-namespace
spec:
  securityContext:
    sysctls:
```



```
      - name: net.core.somaxconn
        value: "65536"
      - name: net.ipv4.tcp_tw_reuse
        value: "1"
      ...
...
```

Go to the container to check whether the configuration takes effect.

```
[root@master-2 ~]# kubectl  get pod -n auto-test-namespace
NAME                                          READY     STATUS    RESTARTS   AGE
cci-deployment-20225241-76dff9f854-6fwlm   1/1       Running   0          15m
cci-deployment-20225241-76dff9f854-nwst7   1/1       Running   0          29m
[root@master-2 ~]# kubectl  exec -it cci-deployment-20225241-76dff9f854-nwst7 /bin/bash -n auto-test-namespace
root@cci-deployment-20225241-76dff9f854-nwst7:/#
root@cci-deployment-20225241-76dff9f854-nwst7:/# cat /proc/sys/net/core/somaxconn
65536
root@cci-deployment-20225241-76dff9f854-nwst7:/# cat /proc/sys/net/ipv4/tcp_tw_reuse
1
root@cci-deployment-20225241-76dff9f854-nwst7:/#
```

2.4 Resizing /dev/shm

Scenario

/dev/shm consists of a temporary file system (tmpfs). tmpfs is a memory-based file system implemented in Linux/Unix and has high read/write efficiency.

In CCI, the default size of **/dev/shm** is 64 MB, which cannot meet customer requirements. You can change its size for data exchange between processes or temporary data storage.

This topic shows how to resize **/dev/shm** by setting memory-backed emptyDir or configuring **securityContext** and **mount** commands.

Constraints

- **/dev/shm** uses a memory-based tmpfs to temporarily store data. Data is not retained after the container is restarted.
- You can use either of the following methods to change the size of **/dev/shm**. However, do not use both methods in one pod.
- The emptyDir uses the memory requested by the pod and does not occupy extra resources.
- Writing data in **/dev/shm** is equivalent to requesting memory. In this scenario, you need to evaluate the memory usage of processes. When the sum of the memory requested by processes in the container and the data volume in the emptyDir exceeds the memory limit of the container, memory overflow occurs.
- When resizing **/dev/shm**, set the size to 50% of the pod's memory request.

Resizing /dev/shm Using Memory-backed emptyDir

emptyDir is applicable to temporary data storage, disaster recovery, and runtime data sharing. It will be deleted upon deletion or transfer of workload pods.

CCI allows you to mount memory-backed emptyDir. You can specify the memory size allocated to the emptyDir and mount it to the **/dev/shm** directory in the container to resize **/dev/shm**.


```
apiVersion: cci/v2
kind: Pod
metadata:
 name: pod-emptydir-name
spec:
 containers:
  - image: 'library/ubuntu:latest'
    volumeMounts:
     - name: volume-emptydir1
       mountPath: /dev/shm
    name: container-0
    resources:
     limits:
      cpu: '4'
      memory: 8Gi
     requests:
      cpu: '4'
      memory: 8Gi
 volumes:
  - emptyDir:
     medium: Memory
     sizeLimit: 4Gi
    name: volume-emptydir1
```

After the pod is started, run the **df -h** command to go to the **/dev/shm** directory. If the following information is displayed, the size is successfully modified.

Figure 2-2 /dev/shm directory details

```
root@pod-emptydir-name:/# df -h
Filesystem      Size  Used Avail Use% Mounted on
overlay          27G  161M   25G   1% /
tmpfs            64M     0   64M   0% /dev
tmpfs           4.0G     0  4.0G   0% /sys/fs/cgroup
tmpfs           4.0G     0  4.0G   0% /dev/shm
overlay          27G  161M   25G   1% /etc/hosts
/dev/vda4        27G  161M   25G   1% /etc/hostname
tmpfs           4.0G     0  4.0G   0% /proc/acpi
tmpfs           4.0G     0  4.0G   0% /proc/scsi
tmpfs           4.0G     0  4.0G   0% /sys/firmware
```

2.5 Configuring Transparent Huge Pages

Scenario

Transparent Huge Pages (THP) is a memory management feature in the Linux kernel. It is designed to compact multiple small pages (usually 4 KB) into huge pages (usually 2 MB). THP dynamically adjusts the page size to optimize memory management and reduces the number of page table lookups and Translation Lookaside Buffer (TLB) misses to reduce memory access latency. However, if there is memory fragmentation or high overhead of background processes, THP may increase memory usage.

Advantages of THP

- Reduced TLB misses

  THP compacts multiple small pages (usually 4 KB) into huge pages (usually 2 MB), significantly reducing TLB misses. TLB misses cause the processor to access the page table for address translation, increasing the memory access

latency. For memory-intensive applications, THP reduces the memory access latency and significantly improves application performance.

- Reduced resource consumption for memory management by kernel

  When the size of a memory page increases, the number of pages to be maintained in the same physical memory is greatly reduced. This reduces the resource consumption for the kernel to manage memory and the time of page table queries. The page table query time is proportional to the page table hierarchy and the number of pages. The decrease in the number of pages improves the query efficiency and the overall system performance.

Disadvantages of THP

- Higher memory usage

  In an environment where memory is frequently allocated and released, THP may cause memory fragmentation, which increases the memory usage. When there are a large number of small pages that cannot be compacted into huge pages, THP may not work effectively. As a result, memory allocation fails or performance deteriorates. Memory fragmentation is more obvious in memory-intensive workloads.

- Potential OOM error

  The resource allocation of THP may cause the actual memory usage to be inconsistent with the application requirements. For example, an application requires only 8 KB of memory (two small pages), but the kernel may allocate 2 MB of memory (a transparent huge page). When the system memory is insufficient, the extra memory usage may trigger an out of memory (OOM) error, affecting system stability.

THP Configuration Suggestions

- When an application frequently requests and releases small memory blocks (for example, 4-KB pages), THP may frequently attempt to merge and split pages. This dynamic management process requires additional compute resources, which may significantly increase the memory management overhead. In addition, frequent merging and splitting operations may cause memory fragmentation, further reducing memory utilization. For this reason, you are advised not to enable THP in this scenario to avoid performance deterioration.
- If the application does not have high performance requirements and the system memory is insufficient, you can disable THP to reduce memory usage. The resource allocation of THP may cause the actual memory usage to be inconsistent with the application requirements, and the resident set size (RSS) increases. After THP is disabled, the system uses small 4-KB pages for management. Although the number of TLB misses may increase, there is less memory fragmentation, and the memory usage can be reduced. This avoids the OOM error caused by insufficient memory.

Constraints

- CCI allows you to configure THP policies by using annotations when creating a workload. If you modify the THP policy by editing the YAML file for the Deployment, the pods will be recreated.

- If THP policies are enabled:
  - The default value of **transparent_hugepage/defrag** is **madvise**. The memory is only reclaimed from the area specified by **madvise (MADV_HUGEPAGE)**.
  - The default value of **transparent_hugepage/khugepaged/defrag** is **1**, indicating that khugepaged for memory defragmentation is enabled by default.

Procedure

1. Log in to the **CCI 2.0 console**.
2. In the navigation pane, choose **Workloads**. On the **Deployments** tab, click **Create from YAML**.
3. Define the Deployment. The following is an example YAML file:

```
kind: Deployment
apiVersion: cci/v2
metadata:
  name: deploy-example
  namespace: test-namespace
spec:
  replicas: 1
  selector:
    matchLabels:
      app: deploy-example
  template:
    metadata:
      labels:
        app: deploy-example
      annotations:
        system.cci.io/transparent_hugepage.enabled: never
    spec:
      containers:
        - name: deploy-example
          image: nginx
          resources:
            limits:
              cpu: 500m
              memory: 1Gi
            requests:
              cpu: 500m
              memory: 1Gi
      dnsPolicy: Default
      imagePullSecrets:
        - name: imagepull-secret
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxUnavailable: 0
      maxSurge: 100%
```


Table 2-2 Description of key fields

Field	Type	Description
system.cci.io/transparent_hugepage.enabled	String	• **always** (default): enables THP globally. • **madvise**: enables THP globally only in the area specified by **madvise (MADV_HUGEPAGE)**. • **never**: disables THP globally.

4. Click **OK**.

2.6 Configuring Multiple Network Interfaces and EIPs for a Pod

Scenario

When a pod needs multiple network interfaces to manage network traffic, you can configure them for the pod on CCI. This feature allows you to create multiple networks in the same namespace and configure the network and EIP used by each network interface in a pod annotation.

Feature Description

- When multiple networks are created, you must specify a network in the namespace as the default network (by setting **spec.defaultNetwork** to **true**). If there is only one network in the namespace, the network is identified as the default network.
- If there are multiple networks in a namespace, the default network cannot be changed to a non-default network.
- If there are multiple networks in a namespace, you can delete the default network configuration. However, you need to reconfigure a new default network after the deletion. Otherwise, the network function is abnormal.
- When multiple networks are created, the subnets of the networks cannot overlap, and the subnets configured for all networks must be in the same VPC.
- Deployments cannot have multiple network interfaces and EIPs. Only pods can have multiple network interfaces and EIPs.
- You can configure multiple network interfaces and EIPs for pods through the **yangtse.io/multi-eip-ids** or **k8s.v1.cni.cncf.io/networks** annotation. The two annotations cannot be configured at the same time.
- Only general-computing-lite pods can have multiple network interfaces and EIPs.
- A pod can have a maximum of five network interfaces.
- If a pod is configured with multiple network interfaces, only the IP address of the primary network interface is displayed. (The IP address displayed in

status.podIP and **status.podIPs** is the IP address of the first network interface.)

Constraints

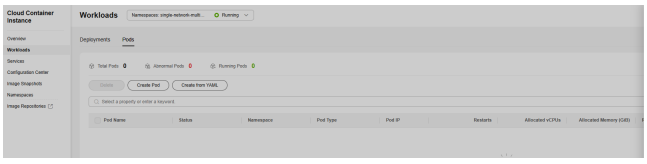
- Pods with multiple network interfaces may take a little longer to start. Plan and use the network interfaces appropriately.
- If there are multiple network interfaces and EIPs, only IPv4 addresses are supported.
- If a pod is configured with multiple network interfaces and EIPs, the primary network interface is used by default. You need to synchronize the special configurations (such as EIP access to the public network and security group network segment bypass) of other network interfaces to the primary network interface. If the special configurations are not synchronized, there may be problems such as image pull failure and volume mounting failure.
- If a pod is configured with multiple network interfaces and EIPs, the **cci.io/image-snapshot-create-if-not-present** annotation cannot be used for automatic image snapshot creation.
- If a pod is configured with multiple network interfaces and EIPs, the annotations for configuring a single EIP (including **yangtse.io/eip-bandwidth-id**, **yangtse.io/eip-bandwidth-size**, **yangtse.io/eip-network-type**, **yangtse.io/eip-charge-mode**, **yangtse.io/eip-bandwidth-name**, **yangtse.io/pod-with-eip** and **yangtse.io/eip-id**) are not supported.
- If IP addresses in the subnet in the primary network are used up, a non-primary network will not be selected.

Procedure

1. Call the CCI API to set the default network and create multiple networks.

```
apiVersion: yangtse/v2
kind: Network
metadata:
  name: test-network
  namespace: test
spec:
  defaultNetwork: true          // If defaultNetwork is set to true, the network is the default network.
  networkType: underlay_neutron
  subnets:
  - subnetID: ${subnet1}
  - subnetID: ${subnet2}
```

2. Log in to the **CCI 2.0 console**.
3. In the navigation pane, choose **Workloads**. On the **Pods** tab, click **Create from YAML**.



4. Define the pod. The following is an example YAML file:

NOTE

The **k8s.v1.cni.cncf.io/networks** field is used as an example. If the **yangtse.io/multi-eip-ids** field is required, see **Table 2-3**.


```
kind: Pod
apiVersion: cci/v2
metadata:
  name: pod-muti-eip-test
  annotations:
    k8s.v1.cni.cncf.io/networks: '[{"name":"second-network","interface":"eth0"},{"name":"default-network","interface":"eth1"},{"name":"default-network","interface":"eth2"},{"name":"second-network","interface":"eth3"}]'
    resource.cci.io/instance-type: general-computing-lite
    resource.cci.io/pod-size-specs: 0.25_0.5
spec:
  containers:
    - name: init-myservice
      image: nginx:latest
      ports:
        - containerPort: 80
          protocol: TCP
      resources:
        limits:
          cpu: 250m
          memory: 512Mi
        requests:
          cpu: 250m
          memory: 512Mi
  restartPolicy: Always
  terminationGracePeriodSeconds: 30
  dnsPolicy: Default
  securityContext: {}
  imagePullSecrets:
    - name: imagepull-secret
```

Table 2-3 yangtse.io/multi-eip-ids and **k8s.v1.cni.cncf.io/networks** description

Field	Value Type	Format	Description	Example Value
yangtse.io/multi-eip-ids	String	eip-id1,eip-id2,eip-id3,eip-id4	• EIP IDs. Use commas (,) to separate multiple IDs. Each EIP ID is unique. The EIPs are not bound to any resources.	yangtse.io/multi-eip-ids: eip-id1,eip-id2,eip-id3,eip-id4

Field	Value Type	Format	Description	Example Value

k8s.v1.cni.cncf.io/networks	String	[
		{
		"name": "second-network",
		"interface": "eth1",
		"eip": {
		"id": "eip-id",
		"bandwidth-size": "5",
		"network-type": "5_g-vm",
		"charge-mode": "bandwidth",
		"bandwidth-name": "eip-myself",
		"bandwidth-id": "bandwidth-id"
		}
		}
		]
			• **name**: (optional) The default value is **""**. If this parameter is not specified, the default network is used. The network must be created in advance.
			• **interface**: (optional) indicates the name of the network interface. The default value is **""**. If this parameter is not specified, the network interface name is constructed based on the array sequence. For example, if the first element in the array is **eth0**, the network interface name is **eth0**. The network interface names in the pod must be set to **eth0** and **eth1** in sequence.
			• **eip**: (optional) indicates the EIP associated with the network
				k8s.v1.cni.cncf.io/networks: [{"name":"default-network","interface":"eth0","eip":{"id":"eip-id"}},{"name":"second-network","interface":"eth1","eip":{"bandwidth-size":"5","network-type":"5_g-vm","charge-mode":"bandwidth","bandwidth-name":"eip-myself"}},{"name":"third-network","interface":"eth2","eip":{"bandwidth-id":"bandwidth-id"}},{"name":"network-no-eip","interface":"eth3"}]

Field	Value Type	Format	Description	Example Value
			interface of the pod. If this parameter is not specified, no EIP needs to be configured for the network interface. A network interface can only be associated with one EIP ID. Multiple network interfaces cannot be associated with the same EIP ID. • **id**: (optional) indicates the ID of an EIP that has not been bound to any resource. If this parameter is not specified, an existing EIP ID is not used. If this parameter is specified, the **bandwidth-id**, **bandwidth-size**, **bandwidth-name**, **network-type**, and **charge-mode** parameters become invalid. • **bandwidth-id**: (optional) indicates the ID	

Field	Value Type	Format	Description	Example Value
			of a shared bandwidth. If this parameter is specified, EIPs use a shared bandwidth. If this parameter is not specified, dedicated bandwidths are used. If this parameter is specified, the **bandwidth-size**, **bandwidth-name**, and **charge-mode** parameters become invalid. • **bandwidth-size**: (optional) indicates the dedicated bandwidth range, in Mbit/s. The default value is **5**. If this parameter is specified, the specified bandwidth range will be used. This parameter is used together with **network-type**, **charge-mode**, and **bandwidth-name**. The bandwidth ranges vary by region. For	

Field	Value Type	Format	Description	Example Value
			details, see the EIP console. • **network-type**: (optional) indicates a public IP address. The default value is **5_bgp**. If this parameter is specified, an EIP will be assigned. This parameter is used together with **bandwidth-size**, **charge-mode**, and **bandwidth-name**. The types vary by region. For details, see the EIP console. • **charge-mode**: (optional) indicates the billing mode. If this parameter is specified, an EIP with the specified billing mode will be assigned. This parameter is used together with **bandwidth-size**, **network-type**, and **bandwidth-name**. The default value varies	

Field	Value Type	Format	Description	Example Value
			depending on the region configuration. – **bandwidth**: billed by bandwidth – **traffic**: billed by traffic • **bandwidth-name**: (optional) indicates the name of the dedicated bandwidth. The default value is randomly generated. After this parameter is specified, a dedicated bandwidth with the specified name will be assigned. This parameter is used together with **bandwidth-size**, **network-type**, and **charge-mode**. The following configuration is for your reference: – The value can contain 1 to 64 characters and must start and end with a digit	

Field	Value Type	Format	Description	Example Value
			or letter. Only digits, letters, underscores (_), hyphens (-), and periods (.) are allowed. – Minimum characters: 1 – Maximum characters: 64	

5. Click **OK**.

3 Storage Management

3.1 Adding Ephemeral Storage Capacity

This topic describes how to add ephemeral storage capacity when a large amount of data is written or a large image is used.

Scenario

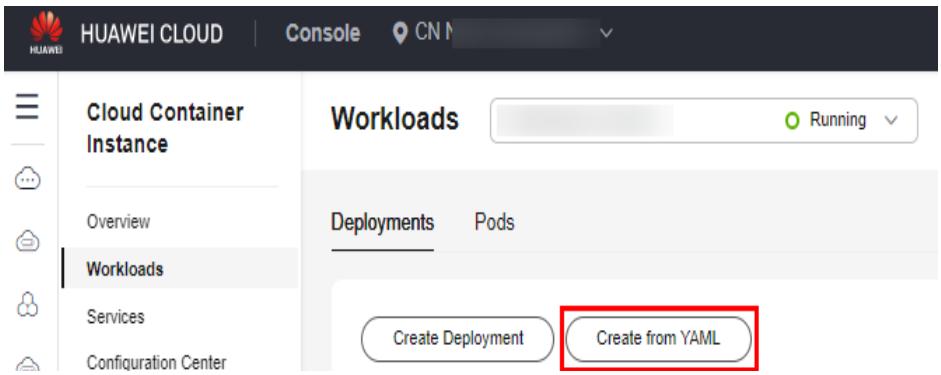
If the pod needs to write a large amount of data to rootfs or emptyDir or if the image size is greater than 30 GiB, you need to expand the ephemeral storage capacity.

Precautions

The maximum ephemeral storage capacity is 994 GiB.

Procedure

Step 1 Log in to the CCI console. In the navigation pane, choose **Workloads**. On the **Deployments** tab, click **Create from YAML**.



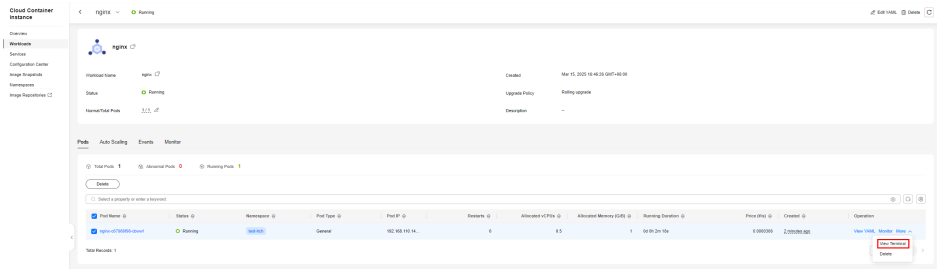
Step 2 Fill in the YAML file as follows:

```
kind: Deployment
apiVersion: cci/v2
metadata:
```



```
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:latest
          resources:
            limits:
              cpu: 500m
              memory: 1Gi
            requests:
              cpu: 500m
              memory: 1Gi
      dnsPolicy: Default
      extraEphemeralStorage:
        sizeInGiB: 10 # Ephemeral storage capacity to be added, in GiB
```

Step 3 Go to the workload details page. On the **Pods** tab, locate the target pod and click **View Terminal** in the **Operation** column.



Step 4 Enter **lsblk** and press **Enter** to check the system disk size after the expansion. (The ephemeral storage capacity is 30 GiB by default.)

----End

4 Image Management

4.1 Using Image Snapshots to Accelerate Image Pull

Scenario

Image snapshots provide a more efficient way to start up containers. Using an image snapshot to create a pod accelerates image pull and reduces the time required for starting up the containers. For details about how to use image snapshots, see **Using Image Snapshots**.

Procedure

Step 1 Log in to the CCI console. In the navigation pane, choose **Image Snapshots**. On the displayed page, click **Create from YAML**. The following is an example YAML file:

```
apiVersion: cci/v2
kind: ImageSnapshot
metadata:
  name: imagesnapshot-a
spec:
  buildingConfig:
    namespace: default  # Namespace
  imageSnapshotSize: 20 # Image snapshot size, in GiB
  ttlDaysAfterCreated: 100 # Image validity period
  images:
    - image: nginx:stable-alpine-perl # Image
  registries: []
```

Step 2 In the navigation pane, choose **Workloads**. On the **Deployments** tab, click **Create from YAML** to create a workload. Add **cci.io/image-snapshot-auto-match: 'true'** in **annotations**. The following is an example:

```
kind: Deployment
apiVersion: cci/v2
metadata:
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
```



```
metadata:
  labels:
    app: nginx
  annotations:
    cci.io/image-snapshot-auto-match: 'true' # Automatic matching of image snapshots
spec:
  containers:
    - name: nginx
      image: nginx:stable-alpine-perl
      resources:
        requests:
          cpu: 500m
          memory: 1Gi
```

Step 3 Verify that the pod annotations contain the **cci.io/image-snapshot-detail** and **cci.io/imagesnapshot-volume** fields, the matched image snapshot is displayed, and the size of the mounted data disk is the same as that of the image snapshot.

----End

4.2 Using a Third-Party Image to Create a Pod

This topic describes how to create a pod using an image pulled from a third-party repository.

Scenario

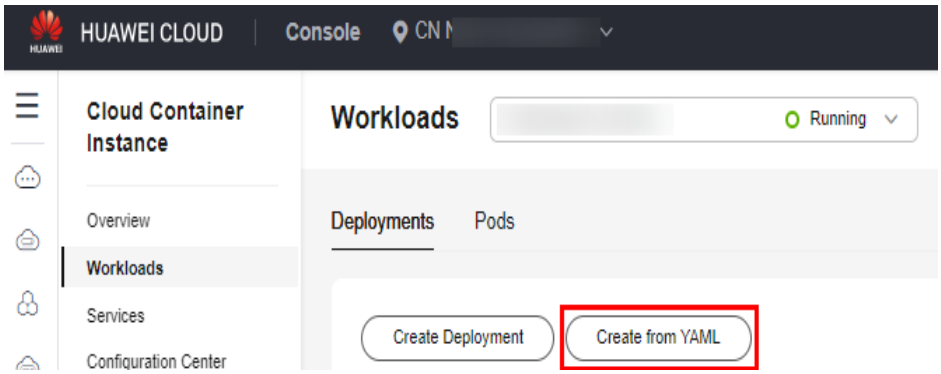
An image in a third-party repository is required.

Constraints

An EIP must be configured for Harbor first.

Procedure

Step 1 Log in to the CCI console. In the navigation pane, choose **Workloads**. On the **Deployments** tab, click **Create from YAML**.



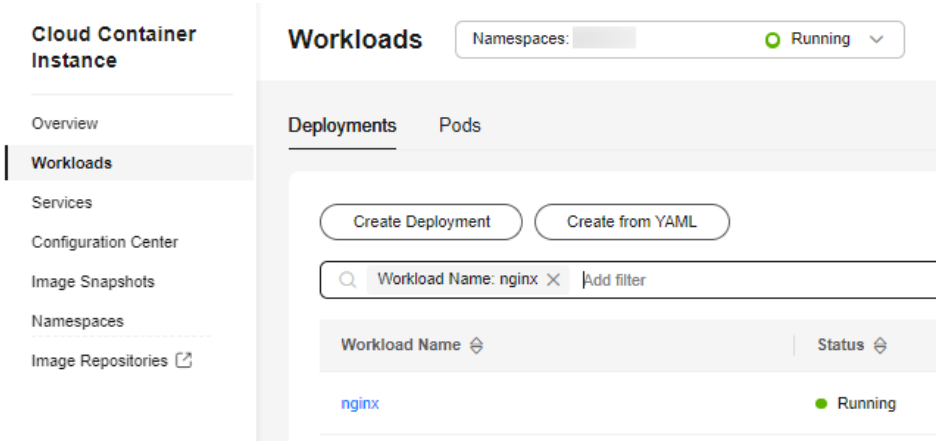
Step 2 Fill in the YAML file as follows:

```
kind: Deployment
apiVersion: cci/v2
metadata:
  name: nginx
spec:
```



```
replicas: 1
selector:
  matchLabels:
    app: nginx
template:
  metadata:
    labels:
      app: nginx
    annotations:
      cci.io/insecure-registries: 100.85.XXX.XXX # EIP bound to Harbor. This IP address is used to pull images from the self-managed image repository.
      yangtse.io/pod-with-eip: 'true' # Controls whether the pod can access the EIP.
  spec:
    containers:
      - name: nginx
        image: 100.85.XXX.XXX/library/nginx:stable-alpine-perl # Image address in Harbor
        resources:
          requests:
            cpu: 500m
            memory: 1Gi
```

Step 3 Verify that the workload is in the **Running** state.



----End

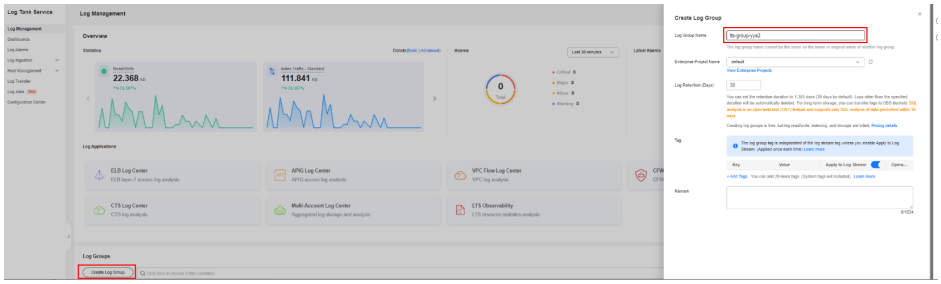
5 Log Monitoring

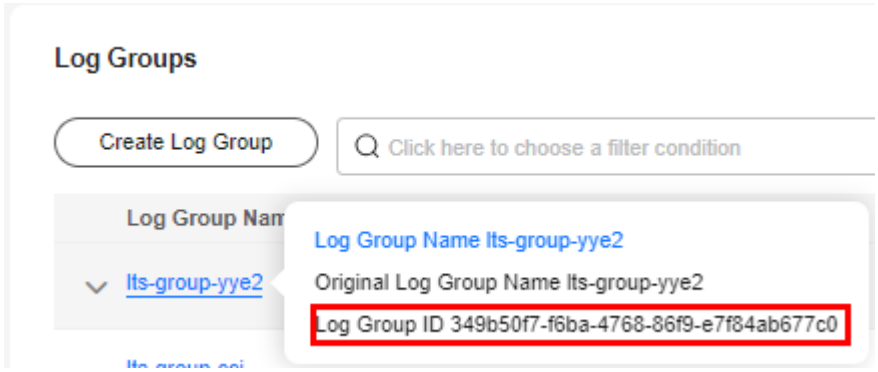
5.1 Reporting Logs to LTS

Scenario

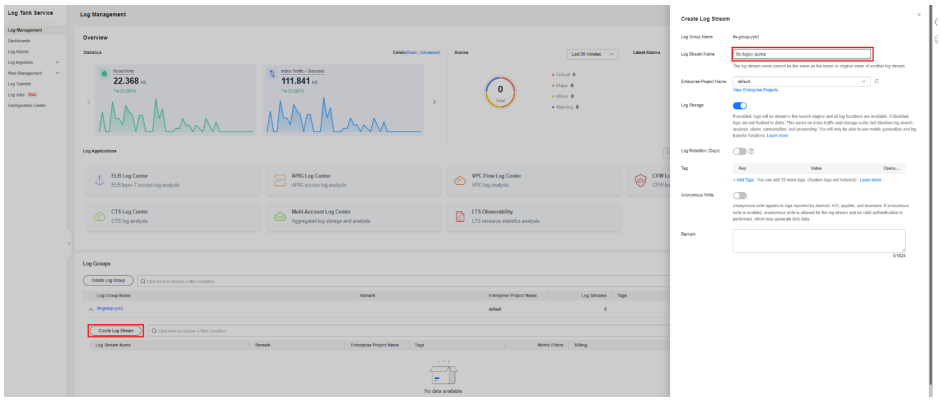
Log Tank Service (LTS) analyzes and processes massive amounts of log data to maximize the availability and performance of cloud services and applications. CCI reports logs of containers in each pod to LTS.

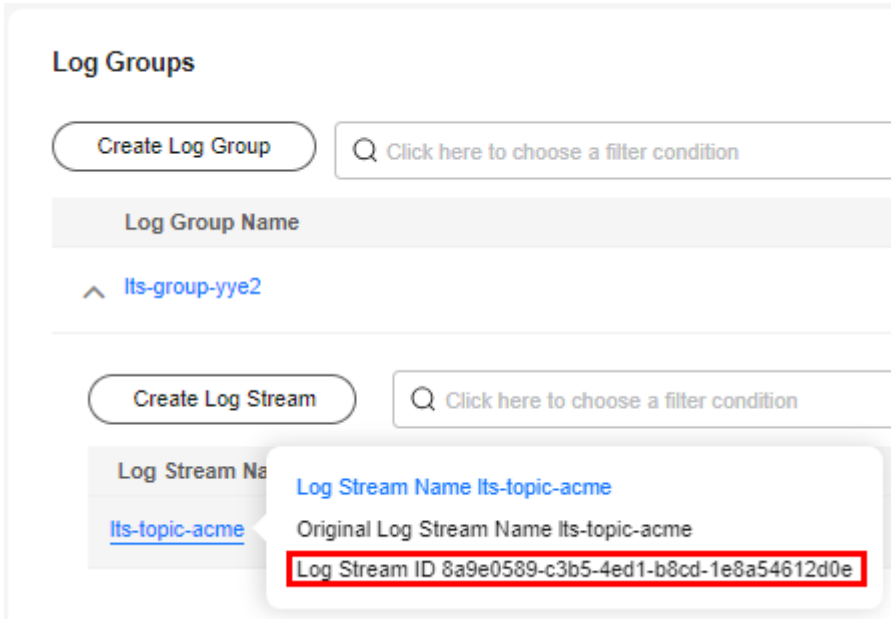
Step 1 Log in to the LTS console and create a log group.



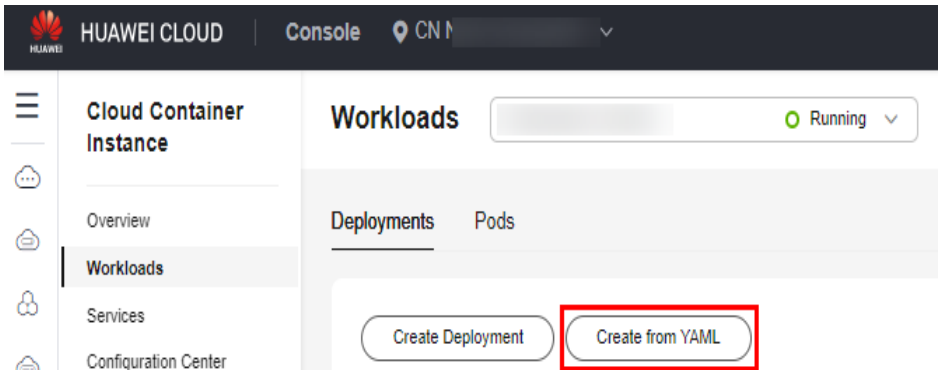


Step 2 Create a log stream in the created log group.





Step 3 Log in to the CCI console. In the navigation pane, choose **Workloads**. On the **Deployments** tab, click **Create from YAML**.



Step 4 Edit the YAML file.

```
apiVersion: cci/v2
kind: Deployment
metadata:
  name: test
spec:
```

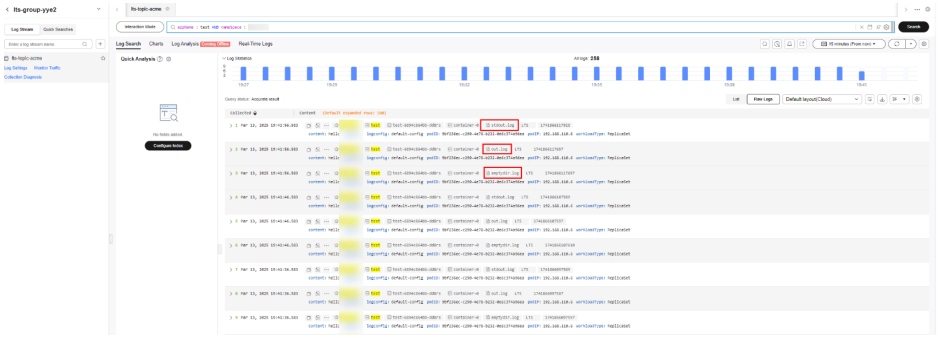


```
  replicas: 1
  selector:
    matchLabels:
      app: test
  template:
    metadata:
      annotations:
        logconf.k8s.io/fluent-bit-log-type: lts      # Where logs are reported
        logconfigs.logging.openvessel.io: "{\"default-config\":{\"container_files\":{\"container-0\":\"stdout.log;/
root/out.log;/data/emptydir-volume/*.log\"},\"regulation\":\"/(?<log>\\\\d+-\\\\d+-\\\\d+ \\\\d+:\\\\d+:\\\\d
+.*)/\",\"lts-log-info\":{\"349b50f7-f6ba-4768-86f9-e7f84ab677c0\":\"8a9e0589-c3b5-4ed1-
b8cd-1e8a54612d0e\"}}}"  # Log reporting configuration
      labels:
        app: test
    spec:
      containers:
      - image: centos:latest
        command: ['sh', '-c', "while true; do echo hello; touch /root/out.log; echo hello >> /root/out.log; touch /
data/emptydir-volume/emptydir.log; echo hello >> /data/emptydir-volume/emptydir.log; sleep 10; done"]
# Command used to simulate log writing
        volumeMounts:
        - name: emptydir-volume
          mountPath: /data/emptydir-volume
        - name: emptydir-memory-volume
          mountPath: /data/emptydir-memory-volume
        name: container-0
        resources:
          limits:
            cpu: 100m
            memory: 100Mi
          requests:
            cpu: 100m
            memory: 100Mi
      volumes:
      - name: emptydir-volume
        emptyDir: {}
      - name: emptydir-memory-volume
        emptyDir:
          sizeLimit: 1Gi
          medium: Memory
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 0
```

In **annotations**, **logconfigs.logging.openvessel.io** specifies configuration items for log reporting, and the configuration items are in JSON format. The following describes each configuration item:

```
{
    "default-config": { // Configuration name. You can change it as needed.
        "container_files": { // You can set the log collection paths of multiple containers. stdout.log indicates
standard output. /root/out.log indicates the text logs in rootfs (volumes included). /data/emptydir-xxx/
*.log indicates the directories in rootfs (volumes included).
            "container-0": "stdout.log;/root/out.log;/data/emptydir-volume/*.log",
            ...
        },
        "regulation": "/(?<log>\\d+-\\d+-\\d+ \\d+:\\d+:\\d+.*)/",  // Regular expression matching rule for
collecting multi-line logs
        "lts-log-info": { // Only one log group and one log stream are allowed.
            "f938a11a-678e-4631-80a0-3667be1280f7": "a248c18b-61ed-4de2-8f7f-c042b78bb0fd" // Log
group ID and log stream ID in the format of {log-group-ID}:{log-stream-ID}
        }
    },
    "other-config": { // Other settings
        ...
    }
}
```


Step 5 Wait until the workload enters the running state and click **View Log** in the **Operation** column to view the logs reported by the pod.



----End

