ALIBABA CLOUD

Alibaba Cloud

Container Service for Kubernetes

User Guide for Serverless Kubernetes Clusters

Document Version: 20220711

Alibaba Cloud

Legal disclaimer

Alibaba Cloud reminds you to carefully read and fully understand the terms and conditions of this legal disclaimer before you read or use this document. If you have read or used this document, it shall be deemed as your total acceptance of this legal disclaimer.

1. You shall download and obtain this document from the Alibaba Cloud website or other Alibaba Cloud-authorized channels, and use this document for your own legal business activities only. The content of this document is considered confidential information of Alibaba Cloud. You shall strictly abide by the confidentiality obligations. No part of this document shall be disclosed or provided to any third party for use without the prior written consent of Alibaba Cloud.

2. No part of this document shall be excerpted, translated, reproduced, transmitted, or disseminated by any organization, company or individual in any form or by any means without the prior written consent of Alibaba Cloud.

3. The content of this document may be changed because of product version upgrade, adjustment, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and an updated version of this document will be released through Alibaba Cloud-authorized channels from time to time. You should pay attention to the version changes of this document as they occur and download and obtain the most up-to-date version of this document from Alibaba Cloud-authorized channels.

4. This document serves only as a reference guide for your use of Alibaba Cloud products and services. Alibaba Cloud provides this document based on the "status quo", "being defective", and "existing functions" of its products and services. Alibaba Cloud makes every effort to provide relevant operational guidance based on existing technologies. However, Alibaba Cloud hereby makes a clear statement that it in no way guarantees the accuracy, integrity, applicability, and reliability of the content of this document, either explicitly or implicitly. Alibaba Cloud shall not take legal responsibility for any errors or lost profits incurred by any organization, company, or individual arising from download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, take responsibility for any indirect, consequential, punitive, contingent, special, or punitive damages, including lost profits arising from the use or trust in this document (even if Alibaba Cloud has been notified of the possibility of such a loss).

5. By law, all the contents in Alibaba Cloud documents, including but not limited to pictures, architecture design, page layout, and text description, are intellectual property of Alibaba Cloud and/or its affiliates. This intellectual property includes, but is not limited to, trademark rights, patent rights, copyrights, and trade secrets. No part of this document shall be used, modified, reproduced, publicly transmitted, changed, disseminated, distributed, or published without the prior written consent of Alibaba Cloud and/or its affiliates. The names owned by Alibaba Cloud shall not be used, published, or reproduced for marketing, advertising, promotion, or other purposes without the prior written consent of Alibaba Cloud. The names owned by Alibaba Cloud include, but are not limited to, "Alibaba Cloud", "Aliyun", "HiChina", and other brands of Alibaba Cloud and/or its affiliates, which appear separately or in combination, as well as the auxiliary signs and patterns of the preceding brands, or anything similar to the company names, trade names, trademarks, product or service names, domain names, patterns, logos, marks, signs, or special descriptions that third parties identify as Alibaba Cloud and/or its affiliates.

6. Please directly contact Alibaba Cloud for any errors of this document.

Document conventions

Style	Description	Example
⚠ Danger	A danger notice indicates a situation that will cause major system changes, faults, physical injuries, and other adverse results.	⚠ **Danger:** Resetting will result in the loss of user configuration data.
🔔 Warning	A warning notice indicates a situation that may cause major system changes, faults, physical injuries, and other adverse results.	🔔 **Warning:** Restarting will cause business interruption. About 10 minutes are required to restart an instance.
📢 Notice	A caution notice indicates warning information, supplementary instructions, and other content that the user must understand.	📢 **Notice:** If the weight is set to 0, the server no longer receives new requests.
? Note	A note indicates supplemental instructions, best practices, tips, and other content.	? **Note:** You can use Ctrl + A to select all files.
>	Closing angle brackets are used to indicate a multi-level menu cascade.	Click **Settings> Network> Set network type**.
Bold	Bold formatting is used for buttons , menus, page names, and other UI elements.	Click **OK**.
`Courier font`	Courier font is used for commands	Run the `cd /d C:/window` command to enter the Windows system folder.
Italic	Italic formatting is used for parameters and variables.	`bae log list --instanceid` *Instance_ID*
[] or [a|b]	This format is used for an optional value, where only one item can be selected.	`ipconfig [-all|-t]`
{} or {a|b}	This format is used for a required value, where only one item can be selected.	`switch {active|stand}`

Table of Contents

1.ASK overview

This topic provides an overview of serverless Kubernetes (ASK) clusters, including the benefits and use scenarios. This topic also compares ASK clusters with Container Service for Kubernetes (ACK) clusters. This helps you quickly get started with ASK clusters.

Benefits

- **O&M-free**: You can deploy an application in an ASK cluster within a few seconds. You can focus on application development without the need to manage nodes.
- **Auto scaling**: You do not need to perform capacity planning for the cluster. ASK automatically scales resources based on your workload requirements.
- **Kubernetes compatibility**: ASK supports Kubernetes-native resources such as Services, Ingresses, and Helm charts. This allows you to seamlessly migrate Kubernetes applications.
- **Secure isolation**: Pods are created based on elastic container instances. Pods of different applications are isolated from each other to prevent mutual interference.
- **Cost-effectiveness**: Pods are created based on your business requirements. You are charged based on the resources used by your applications. You are not charged for idle resources. In addition, the serverless architecture helps reduce O&M costs.
- **Integration and interconnection**: Containerized applications in ASK clusters support seamless integration with Alibaba Cloud fundamentals. These applications can communicate with existing applications and databases in the virtual private cloud (VPC) where the cluster is deployed. The containerized applications can also communicate with VM-based applications.

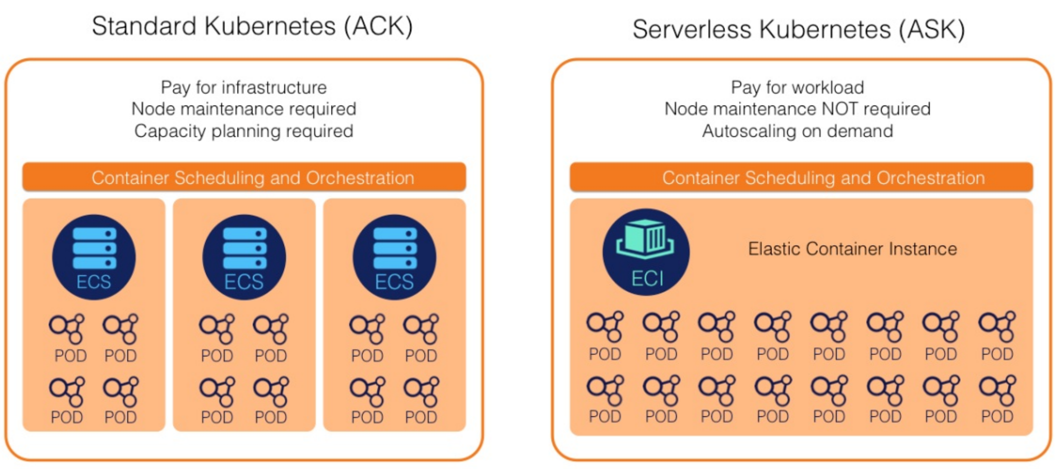
Pricing

When you use ASK clusters, you are charged for pods instead of nodes. The fees of pods are calculated based on the pricing of Elastic Container Instance. For more information, see Elastic Container Instance pricing overview.

You are also charged for other resources used in the clusters, such as Server Load Balancer (SLB) and PrivateZone. For more information about the pricing, see the following references:

- ALB billable items
- PrivateZone pricing

Comparison between ASK and ACK


Standard Kubernetes (ACK)
Pay for infrastructure
Node maintenance required
Capacity planning required
Container Scheduling and Orchestration
ECS
POD
Serverless Kubernetes (ASK)
Pay for workload
Node maintenance NOT required
Autoscaling on demand
Container Scheduling and Orchestration
ECI
Elastic Container Instance
POD

Application scenario

- **Application management**

  In ASK clusters, you do not need to manage or maintain nodes, or perform capacity planning. This reduces the costs of infrastructure management and O&M.

- **Dynamic scaling**

  For workloads that have periodic traffic patterns, such as online education and e-commence applications, ASK clusters can automatically scale resources based on workload requirements. This way, the computing costs and idle resources are reduced, and traffic spikes can be handled in a more efficient manner.

- **Data computing**

  To meet computing requirements of applications such as Spark, ASK clusters can start a large number of pods within a short period of time to process tasks. When the tasks are terminated, the pods are automatically released to stop billing. This dramatically reduces the overall computing costs. For more information, see Use ASK to create Spark tasks.

- **CI/CD**

  You can use ASK clusters to build continuous integration (CI) environments by using tools such as Jenkins or GitLab-Runner. You can set up an application delivery pipeline that covers stages such as source code compilation, image building and pushing, and application deployment. The continuous integration tasks are isolated from each other for enhanced security. You do not need to maintain specific resource pools. This reduces computing costs. For more information, see Elastic and cost-effective CI/CD based on ASK.

- **CronJobs**

  You can run CronJobs in ASK clusters. Billing automatically stops when the jobs are terminated. You do not need to maintain specific resource pools. This avoids resource waste.

2.ASK Billing

Serverless Kubernetes (ASK) clusters are classified into standard ASK clusters and professional ASK clusters. The two types of ASK clusters have different billable items and billing rules. This topic describes the billing rules of standard ASK clusters and professional ASK clusters.

Billing of standard ASK clusters

- When you use ASK clusters, you are charged for pods instead of nodes. The fees of pods are calculated based on the pricing of Elastic Container Instance. For more information, see Elastic Container Instance billing overview.
- You are also charged for other services used in the clusters, such as Server Load Balancer (SLB) and Alibaba Cloud DNS PrivateZone. For more information about the pricing, see the following references:
    - ALB billable items
    - Alibaba Cloud DNS PrivateZone pricing

Billing of professional ASK clusters

- Compared with standard ASK clusters, professional ASK clusters charge you an extra fee for cluster management.

Billing method	Price
Pay-as-you-go	ASK Pro clusters are in public preview and you can use ASK Pro clusters free of charge.

- When you use ASK clusters, you are charged for pods instead of nodes. The fees of pods are calculated based on the pricing of Elastic Container Instance. For more information, see Elastic Container Instance billing overview.
- You are also charged for other services used in the clusters, such as SLB and Alibaba Cloud DNS PrivateZone. For more information about the pricing, see the following references:
    - ALB billable items
    - Alibaba Cloud DNS PrivateZone pricing

Related information

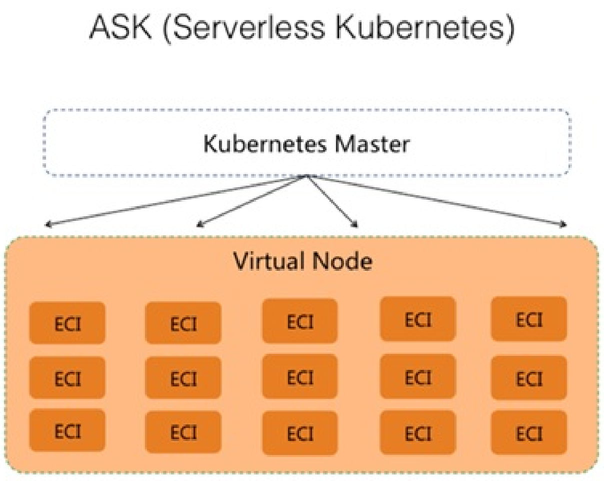
- ASK overview
- ASK Pro cluster overview
- Billing

3.Features

This topic describes the features provided by serverless Kubernetes (ASK) clusters. For more efficient management on ASK clusters, we recommend that you first read and understand this topic.

Virtual nodes

In ASK clusters, pods are deployed on virtual nodes. Virtual nodes enable seamless integration between Kubernetes clusters and elastic container instances (ECIs). Virtual nodes eliminate the limits of underlying computing resources and provide great elasticity for your clusters.



You can view information about virtual nodes in your clusters. Virtual nodes do not occupy computing resources and do not require manual operations for management. For more information about virtual nodes, see Deploy the virtual node controller and use it to create Elastic Container Instance-based pods.

Pod isolation

Each pod in an ASK cluster runs in a secure and isolated container runtime built on an ECI. The underlying compute resources of different ECIs are completely isolated by lightweight virtual sandboxes. ECIs do not affect each other.

The underlying compute resources of ECIs run Alibaba Cloud Linux2. ASK clusters serve as serverless container services. You cannot access the underlying operating system of ECIs.

Pod configurations

Pods are created based on ECIs and support multiple native Kubernetes features, including multiple container startup, environment variables, restart policies, health check commands, volume mounting, and pre-stop commands. You can run the `kubectl logs` command to view container logs and run the `kubectl exec` command to manage containers.

ASK clusters support various annotations to extend the features of pods. For more information, see Elastic Container Instance overview.

Workloads

- ASK clusters support native Kubernetes workloads such as Deployments, StatefulSets, Jobs, CronJobs, pods, and CustomResourceDefinitions (CRDs).
- ASK clusters do not support DaemonSets because ASK clusters do not support the features that are

related to nodes.

Auto scaling

ASK clusters do not contain real nodes. You do not need to be concerned about node capacity planning or cluster expansion by using cluster-autoscaler. You only need to scale your application based on your business requirements. We recommend that you configure Horizontal Pod Autoscaler (HPA) or CronHPA policies to adjust the number of pods based on your business requirements.

Network management

By default, pods use the host network mode. Each pod must be assigned an elastic network interface (ENI) by the VSwitch to enable communications with the Elastic Compute Service (ECS) instances and Relational Database Service (RDS) instances in the virtual private cloud (VPC) where the cluster is deployed.

- Service
  - You can create LoadBalancer type Services.
  - You cannot create NodePort type Services because ASK clusters do not support the features that are related to nodes.
- Ingress
  - SLB Ingress: allows you to forward traffic at Layer 7 based on Server Load Balancer (SLB) instances without deploying controllers. For more information, see ingress demo.
  - NGINX Ingress: allows you to create NGINX Ingresses after nginx-ingress-controller is deployed. For more information, see ingress-nginx demo.
- Service discovery

  To use the service discovery feature within a cluster, enable PrivateZone when you create the cluster.
- EIP

  You can bind elastic IP addresses (EIPs) to ECI pods. You can enable an ECI pod to automatically create an EIP or bind an existing EIP to the ECI pod.

Storage management

You can mount disks of Alibaba Cloud or Network Attached Storage (NAS) file systems provided by Apsara File Storage NAS to pods.

- Disks of Alibaba Cloud
  - To mount a disk through FlexVolume, you do not need to install FlexVolume. You can mount a disk of Alibaba Cloud by specifying the disk ID. For more information, see disk-flexvolume-static.yaml demo. You can also dynamically mount disks. For more information, see disk-flexvolume-dynamic.yaml demo.
  - To dynamically mount disks through persistent volumes (PVs) and persistent volume claims (PVCs), you must first install disk-controller. For more information, see disk-pvc-dynamic.yaml demo.
- NAS file systems of Apsara File Storage NAS
  - To mount NAS file systems through Network File System (NFS), see nas-nfsvolume.yaml demo.
  - To statically mount NAS file systems through FlexVolume, you can directly specify the mount target without the need to install FlexVolume. For more information, see nas-flexvolume.yaml demo.

- To statically mount NAS file systems through PVs and PVCs, you must first install disk-controller. For more information, see nas-pvc.yaml demo.

Log management

In ASK clusters, stdout logs and text logs are collected from pods without the need to deploy Logtail as a DaemonSet. For more information, see Collect log files by using Log Service.

ConfigMaps and Secrets

Supports ConfigMaps and Secrets. You can mount ConfigMaps and Secrets as volumes.

Chart management

Supports chart deployment in App Catalog to create various native Kubernetes applications.

4.Quick start

4.1. ASK quick start

This topic describes how to create a serverless Kubernetes (ASK) cluster in the Container Service for Kubernetes (ACK) console. This topic also describes how to create an application from an image and view the containers in an ASK cluster.

Prerequisites

Step 1: Create an ASK cluster

Note This section describes only the key parameters that are required to create an ASK cluster. You can configure other parameters and install components based on your business requirements. For more information, see Create an ASK cluster.

1.
2.
3.
4. On the page that appears, click the **Serverless Kubernetes** tab and configure the following parameters.

Parameter	Description	Example
Cluster Name	Enter a name for the cluster. **Note** The name must be 1 to 63 characters in length, and can contain digits, letters, and hyphens (-).	ask-hangzhou
		Standard edition
Region	Select a region to deploy the cluster.	China (Hangzhou)

Parameter	Description	Example
VPC	Specify the virtual private cloud (VPC) in which you want to deploy the cluster. Kubernetes clusters support only VPCs. You can select **Create VPC** or **Select Existing VPC**. ◦ **Create VPC**: If you select this option, ACK automatically creates a VPC and a NAT gateway in the VPC. ACK also configures SNAT rules on the NAT gateway. ◦ **Select Existing VPC**: If you select this option, you must select a VPC from the VPC drop-down list and select vSwitches in the vSwitch section. If you want to enable Internet access for the cluster to download container images or perform other operations, you must configure a NAT gateway. We recommend that you upload container images to a Container Registry instance in the region in which the cluster is deployed. This way, you can pull the images over the VPC. For more information, see Create and manage a VPC.	Select Existing VPC

Parameter	Description	Example
Configure SNAT	Specify whether to automatically create a NAT gateway and configure SNAT rules on the NAT gateway. This parameter is required only if you set **VPC** to **Create VPC**. **Note** After you select **Create VPC**, you can specify whether to automatically create a NAT gateway and configure SNAT rules on the NAT gateway. If you clear this check box, you must manually create a NAT gateway and configure SNAT rules on the NAT gateway. Otherwise, the cluster deployed in the VPC cannot access the Internet. For more information, see Create and manage Internet NAT gateways.	Configure SNAT for VPC
Service CIDR		172.21.0.0/20
		Expose API Server with EIP
Terms of Service	You must read and select **Terms of Service for Serverless Kubernetes** before you create the cluster.	Select Terms of Service

5. After you configure the cluster, click **Next:Component Configurations** in lower-right part of the page to configure components.

Parameter	Description	Example

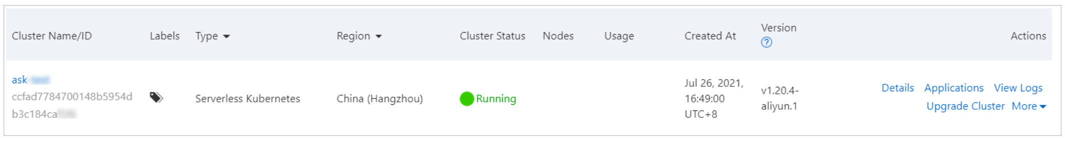
Parameter	Description	Example
Service Discovery	Specify whether to enable the service discovery feature. Default value: **Disable**. For more information, see ASK Service discovery and DNS. ◦ You can select **PrivateZone** to enable the PrivateZone service. ◦ You can select **CoreDNS** to enable the CoreDNS service.	Disable
Ingress	Specify whether to install an Ingress controller. By default, **Do Not Install** is selected. ◦ If you select **Nginx Ingress**, the NGINX Ingress controller is automatically installed. For more information about the NGINX Ingress controller, see Advanced Ingress configurations. ◦ If you select **ALB Ingress**, the Application Load Balancer (ALB) Ingress controller is automatically installed. For more information about how to access Services in a cluster by using ALB Ingresses, see Access Services by using an ALB Ingress.	Do Not Install
Monitoring Service	Specify whether to install the metrics-server component. By default, **Install metrics-server** is not selected. We recommend that you select **Install metrics-server** to enable the monitoring service.	Install metrics-server

Parameter	Description	Example
Log Service	Specify whether to enable Log Service. You can select an existing Log Service project or create one. By default, **Enable Log Service** is selected. This allows you to use Log Service with simple configurations when you create an application. For more information, see Collect log files by using Log Service.	Create Project
Knative	Specify whether to enable Knative. By default, **Enable Knative** is not selected. For more information, see Knative overview.	Enable Knative

6. After you configure components, click **Next:Confirm Order** in lower-right part of the page.
7. On the **Confirm Order** wizard page, select **Terms of Service for Serverless Kubernetes** in the Term of Service section and click **Create Cluster**.

What to do next

- After the cluster is created, you can find the cluster on the Clusters page in the console.



Cluster Name/ID	Labels	Type	Region	Cluster Status	Nodes	Usage	Created At	Version	Actions
ask [redacted] ccfad7784700148b5954d b3c184ca[redacted]		Serverless Kubernetes	China (Hangzhou)	● Running			Jul 26, 2021, 16:49:00 UTC+8	v1.20.4-aliyun.1	Details Applications View Logs Upgrade Cluster More

- On the **Clusters** page, find the created cluster and click **Details** in the Actions column. On the details page, click the **Basic Information** tab to view the basic information about the cluster and click the **Connection Information** tab to view the information about how to connect to the cluster.

Step 2: Create an application from an image

Note This section describes only the key parameters that are required to create an application from an image. For more information, see Create an application from an image.

1. Configure basic information

1.
2.
3. On the **Basic Information** wizard page, configure the basic settings of the application.

Parameter	Description	Example
Name	Enter a name for the application.	serverless-app-deployment

Parameter	Description	Example
Type	Specify the type of the resource object. Valid values include **Deployment** and **StatefulSet**.	Deployment

4. Click **Next**.

2. Configure container settings

1. In the **General** section of the **Container** wizard page, configure the basic settings of the container.

Parameter	Description	Example
Image Name	To select an image, click **Select Image**. In the Select Image dialog box, select an image and click **OK**. You can also enter the address of an image that is stored in a private registry. The image address must be in the following format: `domainname/namespace/imagename:tag`.	In this example, an image that is stored on a Container Registry Personal Edition instance is used. The instance is deployed in the **China (Hangzhou)** region.
Image Version	Click **Select Image Version** and select an image version. If you do not specify an image version, the latest image version is used.	In this example, no image version is specified.

2. In the **Ports** section, click **Add** to configure one or more container ports.

Parameter	Description	Example
Name	Enter a name for the port.	nginx
Container Port	Specify the container port that you want to expose. The port number must be from 1 to 65535.	80
Protocol	Valid values: **TCP** and **UDP**.	TCP

3. Click **Next**.

3. Configure advanced settings

1. In the **Access Control** section of the **Advanced** wizard page, configure a Service to expose the backend pod.

Click **Create** on the right side of **Services**. In the Create Service dialog box, configure the following parameters.

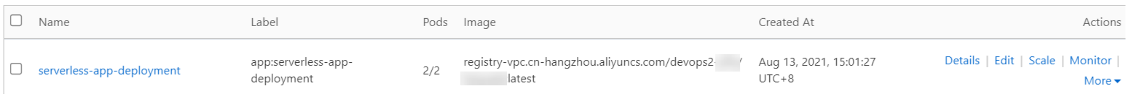
Parameter	Description	Example
Name	Enter a name for the Service.	serverless-app-svc
Type	The type of Service. This parameter determines how the Service is accessed.	Select **Server Load Balancer**. Then, select **Public Access** and **Create SLB Instance**. You can click **Modify** to change the Server Load Balancer (SLB) instance specification based on your business requirements. In this example, the default specification Small I (slb.s1.small) is used.
Port Mapping	Specify a Service port and a container port. The container port must be the same as the one that is exposed in the backend pod.	In this example, the Service port is set to 8080 and the container port is set to 80.

You can find the created Service in the **Access Control** section. You can click **Update** or **Delete** to change the configurations.

2. Click **Create**.

4. Access the NGINX application

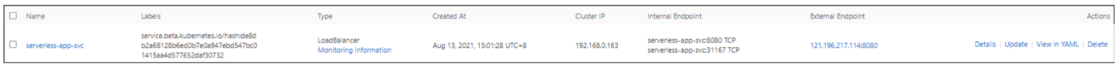
1. On the **Complete** wizard page, click **View Details**. You can find **serverless-app-deployment** on the Deployments page.



Name	Label	Pods	Image	Created At	Actions
serverless-app-deployment	app:serverless-app-deployment	2/2	registry-vpc.cn-hangzhou.aliyuncs.com/devops2[redacted]/[redacted]latest	Aug 13, 2021, 15:01:27 UTC+8	Details | Edit | Scale | Monitor | More

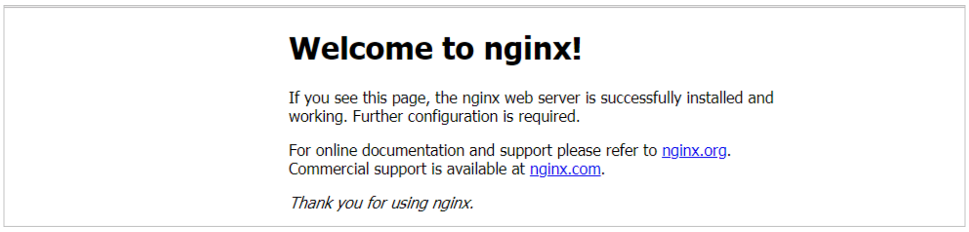
2. Access the NGINX application.

   i. In the left-side navigation pane of the cluster details page, choose . On the Services page, you can find the **serverless-app-svc** Service.



Name	Labels	Type	Created At	Cluster IP	Internal Endpoint	External Endpoint	Actions
serverless-app-svc	service.beta.kubernetes.io/hash:de8db2a68128b6ed0b7e0a947ebd547bc01415aa4d577652daf30732	LoadBalancer Monitoring information	Aug 13, 2021, 15:01:28 UTC+8	192.168.0.163	serverless-app-svc:8080 TCP serverless-app-svc:31167 TCP	121.196.217.114:8080	Details | Update | View in YAML | Delete

ii. Click the address in the **External Endpoint** column to access the NGINX application by using a browser.



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

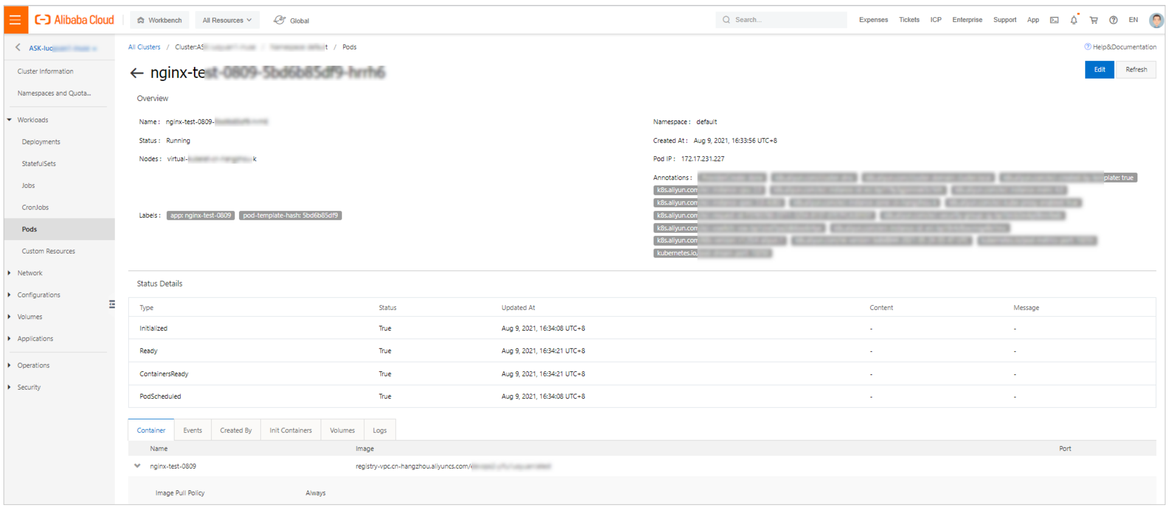
Note For more information about how to access an application by using an Ingress, see Access Services by using an ALB Ingress.

Step 3: View the pod

1.
2. Select the pod that you want to view and click **Details** in the **Actions** column.

   Note You can modify or delete the pod based on your needs. We recommend that you use templates to manage pods that are created by using templates. Do not directly modify or delete these pods.

3. On the details page of the pod, you can view the detailed information about the pod.



All Clusters / Cluster:AS[redacted]t / Pods

nginx-te[redacted]

Overview

Name : nginx-test-0809-[redacted]
Status : Running
Nodes : virtual-[redacted]k
Labels : app: nginx-test-0809 pod-template-hash: 5bd6b85df9

Namespace : default
Created At : Aug 9, 2021, 16:33:56 UTC+8
Pod IP : 172.17.231.227
Annotations : [redacted]plate: true
k8s.aliyun.com[redacted]
k8s.aliyun.com[redacted]
k8s.aliyun.com[redacted]
k8s.aliyun.com[redacted]
k8s.aliyun.com[redacted]
kubernetes.io/[redacted]

Status Details

Type	Status	Updated At	Content	Message
Initialized	True	Aug 9, 2021, 16:34:08 UTC+8	-	-
Ready	True	Aug 9, 2021, 16:34:21 UTC+8	-	-
ContainersReady	True	Aug 9, 2021, 16:34:21 UTC+8	-	-
PodScheduled	True	Aug 9, 2021, 16:34:08 UTC+8	-	-

Container | Events | Created By | Init Containers | Volumes | Logs

Name	Image	Port
nginx-test-0809	registry-vpc.cn-hangzhou.aliyuncs.com/[redacted]	

Image Pull Policy Always

4.2. Deploy NGINX applications in ASK clusters

You do not need to create or manage nodes in serverless Kubernetes (ASK) clusters. This frees you from maintaining cloud resources for your applications and allows you to focus on your business development instead of managing the underlying infrastructure. This topic describes how to deploy web applications in ASK clusters.

Prerequisites

Create an ASK cluster

Context

ASK clusters support standard Kubernetes semantics and API operations. You can create cluster resources with a few clicks, such as Deployments, StatefulSets, Jobs, Services, Ingresses, and custom resources. You can also use Helm to deploy a variety of Kubernetes-native applications in ASK clusters.

Procedure

1. Connect to ACK clusters by using kubectl.
2. Create a YAML file named *nginx.yaml* and copy the following content into the file:

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  ports:
  - port: 80
    protocol: TCP
  selector:
    app: nginx
  type: LoadBalancer
---
apiVersion: apps/v1beta2
kind: Deployment
metadata:
  name: nginx-deploy
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image:  nginx:alpine
        ports:
        - containerPort: 80
        resources:
          requests:
            cpu: "2"
            memory: "4Gi"
```

3. Run the following command to deploy an NGINX application in the ASK cluster:

```
kubectl apply -f nginx.yaml
```

Expected output:

```
service/nginx-service created
deployment.apps/nginx-deploy created
```

4. Check the states of the deployed pods and Services. You can use the IP address of the created Server Load Balancer (SLB) instance to access the NGINX application.
   i. Run the following command to query the states of the application pods:

   ```
   kubectl get pod
   ```

   Expected output:

   ```
   nginx-deploy-55d8dcf755-bxk8n   1/1     Running   0          36s
   nginx-deploy-55d8dcf755-rchhq   1/1     Running   0          36s
   ```

   ii. Run the following command to query the states of created Services:

   ```
   kubectl get svc
   ```

   Expected output:

   ```
   NAME            TYPE           CLUSTER-IP     EXTERNAL-IP   PORT(S)        AGE
   kubernetes      ClusterIP      172.**.*.*     <none>        443/TCP        10d
   nginx-service   LoadBalancer   172.19.*.***   47.57.**.**   80:32278/TCP   39s
   ```

   iii. Run the following command to use the IP address of the created SLB instance to access the NGINX application:

   ```
   curl 47.57.**.**
   ```

   Expected output:

   ```
   <!DOCTYPE html>
   <html>
   <head>
   <title>Welcome to nginx!</title>
   ...
   </html>
   ```

5. Scale out the created Deployment.
   i. Run the following command to query the created Deployment:

   ```
   kubectl get deploy
   ```

   Expected output:

   ```
   NAME           READY   UP-TO-DATE   AVAILABLE   AGE
   nginx-deploy   2/2     2            2           9m32s
   ```

   ii. Run the following command to scale out the created Deployment:

   ```
   kubectl scale deploy nginx-deploy --replicas=10
   ```

   Expected output:

   ```
   deployment.extensions/nginx-deploy scaled
   ```

iii. Run the following command to query the pods after the NGINX application is scaled out:

```
kubectl get pod
```

Expected output:

```
NAME                             READY   STATUS    RESTARTS   AGE
nginx-deploy-55d8dcf755-8jlz2    1/1     Running   0          39s
nginx-deploy-55d8dcf755-9jbzk    1/1     Running   0          39s
nginx-deploy-55d8dcf755-bqhcz    1/1     Running   0          38s
nginx-deploy-55d8dcf755-bxk8n    1/1     Running   0          10m
nginx-deploy-55d8dcf755-cn6x9    1/1     Running   0          38s
nginx-deploy-55d8dcf755-jsqjn    1/1     Running   0          38s
nginx-deploy-55d8dcf755-lhp8l    1/1     Running   0          38s
nginx-deploy-55d8dcf755-r2clb    1/1     Running   0          38s
nginx-deploy-55d8dcf755-rchhq    1/1     Running   0          10m
nginx-deploy-55d8dcf755-xspnt    1/1     Running   0          38s
```

6. You can use Horizontal Pod Autoscaler (HPA) to automatically scale out the Deployment when the CPU usage exceeds the scale-out threshold. After HPA is enabled, more pod replicas are added for the Deployment when the CPU usage exceeds the scale-out threshold.

    i. Run the following command to scale in the number of pod replicas to one for the Deployment:

    ```
    kubectl scale deploy nginx-deploy --replicas=1
    ```

    Expected output:

    ```
    deployment.extensions/nginx-deploy scaled
    ```

    ii. Run the following command to query the state of the application pod:

    ```
    kubectl get pod
    ```

    Expected output:

    ```
    NAME                             READY   STATUS    RESTARTS   AGE
    nginx-deploy-55d8dcf755-rchhq    1/1     Running   0          16m
    ```

    iii. Run the following command to configure the scaling settings of HPA:

    ```
    kubectl autoscale deployment nginx-deploy --cpu-percent=50 --min=1 --max=10
    ```

    Expected output:

    ```
    horizontalpodautoscaler.autoscaling/nginx-deploy autoscaled
    ```

    iv. Run the following command to query the state of HPA:

    ```
    kubectl get hpa
    ```

    Expected output:

    ```
    NAME           REFERENCE                  TARGETS   MINPODS   MAXPODS   REPLICAS   A
    GE
    nginx-deploy   Deployment/nginx-deploy    0%/30%    1         10        1          3
    5s
    ```

5.ECI Pod

5.1. Elastic Container Instance overview

This topic describes the configurations of an Elastic Container Instance-based pod and the limits on Elastic Container Instance. You can customize the following configurations for an elastic container instance-based pod: security isolation, CPU and memory resources and specifications, image pulling, storage, network modes, and log collection.

Prerequisites

- A virtual node is deployed in a Container Service for Kubernetes (ACK) cluster or a serverless Kubernetes (ASK) cluster is created. For more information, see Step 1: Deploy ack-virtual-node in ACK clusters and Create an ASK cluster.
- Elastic Container Instance is activated. You can log on to the Elastic Container Instance console to activate the service.

Isolation

Elastic Container Instance provides a secure and reliable runtime environment for containers of serverless applications. Elastic container instances are completely isolated from one another because sandboxed containers run on lightweight virtual machines. Elastic container instances can be scheduled to run on different physical machines. This ensures high availability.

CPU and memory resources and specifications

Resources are applied and charged in the following ways:

- **Specify CPU and memory resources**
  - **Specify CPU and memory resources for containers**: In the ACK console, you can specify CPU and memory resources for a container by setting the `resources and limits` parameters. If you do not set the parameters, 2 CPU core and 4 GB of memory are allocated to each container. The amount of resources that are required by an elastic container instance refers to the total amount of resources that are required by all containers in the elastic container instance. If the specified specification is not supported, Elastic Container Instance automatically adjusts the CPU and memory resources as required. For example, if 2 CPU cores and 3 GB of memory are specified for all containers, the allocated resources are adjusted to 2 CPU cores and 4 GB of memory. However, if more than four CPU cores are specified for all containers, the allocation is not adjusted.
  - **Specify CPU and memory resources for elastic container instances**: You can specify CPU and memory resources for an elastic container instance by adding annotations. In this case, the CPU and memory resources required by an elastic container instance are not equal to the total amount of resources that are required by all containers. The elastic container instance is created and billed based on the specified resources. After you specify CPU and memory resources for an elastic container instance, you do not need to set the `resource request` and `resource limit` of each container. Containers in the elastic container instance can use computing resources to the maximum extent.

  The following table describes the supported CPU and memory specifications.

vCPU	Memory
.25 vCPU	0.5 GB, 1 GB
.5 vCPU	1 GB, 2 GB
1 vCPU	2 GB, 4 GB, 8 GB
2 vCPU	2 GB, 4 GB, 8 GB, 16 GB
4 vCPU	4 GB, 8 GB, 16 GB, 32 GB
8 vCPU	8 GB, 16 GB, 32 GB, 64 GB
12 vCPU	12 GB, 24 GB, 48 GB, 96 GB
16 vCPU	16 GB, 32 GB, 64 GB, 128 GB
24 vCPU	48 GB, 96 GB, 192 GB
32 vCPU	64 GB, 128 GB, 256 GB
52 vCPU	96 GB, 192 GB, 384 GB
64 vCPU	128 GB, 256 GB, 512 GB

The billing period of CPU and memory resources starts when external storage is mounted to an ECI or container images are downloaded. The billing period ends when the ECI stops running (in the Succeeded or Failed state).

Calculation formula: Elastic Container Instance fee = (Number of CPU cores × Price per CPU core + Memory size × Price per GB) × Elastic Container Instance running duration. Elastic Container Instance is billed on a per-second basis. The following table describes the unit price of each billable item.

Item	Unit Price	Unit price per hour
CPU (per vCPU per second)	USD 0.0000077	USD 0.02772 per hour
Memory (per GB per second)	USD 0.00000096	USD 0.003456 per hour

- **Specify ECS instance types for pods**

  You can specify Elastic Compute Service (ECS) instance types to create elastic container instances with specific capabilities. For example, you can use instance family ecs.sn1ne to create an elastic container instance with enhanced network performance. For more information, see ECS instance families and Pay-as-you-go ECS pricing based on regions .

Note When you specify an ECS instance type to create an elastic container instance, the computing resources consumed by the elastic container instance are billed at the rate of the specified ECS instance type. The following ECS instance families are supported:

- General purpose instance families: g6e, g6, g5, and sn2ne
- Compute optimized instance families: c6e, c6a, c6, c5, and sn1ne
- Memory optimized instance families: r6e, r6, r5, se1ne, and se1
- Compute intensive instance family: ic5
- Compute optimized instance families with high clock speeds: hfc6 and hfc5
- General purpose instance families with high clock speeds: hfg6 and hfg5
- GPU-accelerated compute optimized instance families: gn6i, gn6v, gn5i, and gn5
- Big data instance family with enhanced network performance: d1ne
- Instance families with local SSDs: i2 and i2g
- Burstable instance families: t6 and t5
- Shared performance instance families: s6, xn4, n4, mn4, and e4

Calculation formula: Elastic Container Instance fee = Unit price of ECS specification specified by ECS × Elastic Container Instance running duration. Elastic Container Instance is billed on a per-second basis.

When you specify ECS instance types for elastic container instances, you can use reserved instances to deduct the Elastic Container Instance cost. For more information, see Reserved instance overview.

The Elastic Container Instance costs after the reserved instances are applied are close to the costs of subscription ECS instances.

- **Preemptible instance**

  You can use preemptible instances by adding annotations. Using preemptible instances can significantly reduce the computing costs.

Image pulling

When an Elastic Container Instance starts, the containerd runtime of the elastic container instance pulls container images from a remote image repository. To allow an elastic container instance to pull public images, you must configure a Network Address Translation (NAT) gateway for the virtual private cloud (VPC) where the elastic container instance is deployed. You can also associate an elastic IP address (EIP) with the elastic container instance to pull public images. We recommend that you store container images in Container Registry to reduce the time that is required to pull images through a VPC. In addition, You can pull private images from Container Registry without a password. This ensures high efficiency for image pulling.

Elastic Container Instance supports image snapshots. Elastic container instances cache the pulled container images as snapshots, and then use these snapshots to launch containers. This prevents repetitive pulling requests for the same image from the remote image repository, which saves time and costs. This feature is suitable for the use of large images.

Storage methods

Elastic Container Instance supports the following storage methods:

- FlexVolume:

- You can set the `disk volume Id` parameter to mount a disk volume by using the same method that is used for FlexVolume.
- You can set the `disk volume Id` parameter to mount a disk volume by using the same method that is used for FlexVolume.
- You can use FlexVolume to dynamically create `disk volumes` when you create an elastic container instance. You can also specify the size of the `disk volume` and whether to retain the `disk volume` when the elastic container instance stops running.

- Network File System (NFS): For more information, see Example.
- Persistent volumes (PVs) and persistent volume claims (PVCs): For more information, see Example.

Network mode

By default, Elastic Container Instance-based pods use the host network mode. Each pod is assigned an elastic network interface (ENI) by a vSwitch.

In a Kubernetes cluster, you can use the following methods to enable an Elastic Container Instance-based pod to communicate with pods that run on ECS instance-based nodes.

- Attach Elastic Container Instance-based pods to LoadBalancer type Services: You can attach both Elastic Container Instance-based pods and pods that run on ECS instances-based nodes to LoadBalancer type Services.
- Access ClusterIP type Services: An Elastic Container Instance-based pod can access the IP address of a cluster.
- Associate EIPs with Elastic Container Instance-based pods: You can associate EIPs with Elastic Container Instance-based pods. You can enable an Elastic Container Instance-based pod to automatically create an EIP or associate an existing EIP with the Elastic Container Instance-based pod.

Log collection

You can set environment variables of an elastic container instance to collect `stdout` files or log files and import them to Log Service. In most cases, you do not need to deploy a `sidecar` container that works as Logtail.

List of supported annotations

Notice You must add annotations in the pod configurations. Do not add annotations in the Deployment configurations.

Annotation	Description	Example
k8s.aliyun.com/eci-use-specs	Specifies the instance types and specifications that can be used. You can specify multiple instance types. If a specified instance type is out of stock, the system selects another instance type to create instances. You can use the ${cpu}-${mem}Gi format or specify instance types.	"k8s.aliyun.com/eci-use-specs": "2-4Gi,4-8Gi,ecs.c6.xlarge"

Annotation	Description	Example
k8s.aliyun.com/eci-vswitch	Sets a vSwitch for an elastic container instance.	"k8s.aliyun.com/eci-vswitch" : "${your_vsw_id}"
k8s.aliyun.com/eci-security-group	Sets a security group for an elastic container instance.	"k8s.aliyun.com/eci-security-group" : "${your_security_group_id}"
k8s.aliyun.com/eci-resource-group-id	Sets a resource group to which an elastic container instance belongs.	"k8s.aliyun.com/eci-resource-group-id" : "${your_resource_group_id}"
k8s.aliyun.com/eci-ram-role-name	Sets the Resource Access Management (RAM) role for an elastic container instance to allow it to access other Alibaba Cloud services.	"k8s.aliyun.com/eci-ram-role-name" : "${your_ram_role_name}"
k8s.aliyun.com/eci-image-snapshot-id	Specifies the ID of a cached image to accelerate elastic container instance creation.	k8s.aliyun.com/eci-image-snapshot-id: "${your_image_cache_id}"
k8s.aliyun.com/eci-image-cache	Automatically matches existing cached images. The default value: false.	k8s.aliyun.com/eci-image-cache: "true"
k8s.aliyun.com/eci-with-eip	Creates an EIP and associates it with an elastic container instance.	"k8s.aliyun.com/eci-with-eip": "true"
k8s.aliyun.com/eip-bandwidth	Sets the bandwidth for an EIP. The default bandwidth is 5 Mbit/s.	"k8s.aliyun.com/eci-with-eip": "true""k8s.aliyun.com/eip-bandwidth": 10
k8s.aliyun.com/eci-eip-instanceid	Associates an existing EIP with an elastic container instance.	"k8s.aliyun.com/eci-eip-instanceid": "${your_eip_Instance_Id}"
k8s.aliyun.com/eci-spot-strategy	If you set this parameter to SpotAsPriceGo, the system automatically bids based on the current market price. SpotWithPriceLimit: specifies the highest bid for the preemptible instance.	k8s.aliyun.com/eci-spot-strategy: "SpotAsPriceGo"
k8s.aliyun.com/eci-spot-price-limit	Valid only when k8s.aliyun.com/eci-spot-strategy is set to SpotWithPriceLimit. Sets the highest price within one hour. The price can be accurate to three decimal places.	k8s.aliyun.com/eci-spot-price-limit: "0.250"

Annotation	Description	Example
k8s.aliyun.com/eci-ntp-server	Sets one or more Network Time Protocol (NTP) servers.	k8s.aliyun.com/eci-ntp-server: 100.100.5.1,100.100.5.2 # The IP addresses of the NTP servers.
k8s.aliyun.com/eci-set-diskvolume	Converts the volume to a dynamic disk. For example, emptyDir can be converted to dynamic disks. The format is $volumeName:$type:$size.	k8s.aliyun.com/eci-set-diskvolume: "cache-volume:ext4:500Gi"

Limits on Elastic Container Instance

Elastic container instances and virtual nodes support most pod features. However, Elastic container instance do not support the following features:

- Run DaemonSet pods on virtual nodes.
- Set the hostPath and hostPID parameters.
- Add privileged permissions.
- Create NodePort type Services.
- Set network policies.

5.2. Specify CPU and memory specifications to create an elastic container instance

When you create an elastic container instance, you can specify CPU and memory specifications for the instance or for each container in the instance. This topic describes how to specify the container specifications and pod specifications of an elastic container instance.

Specify container specifications for an elastic container instance

You can specify the CPU and memory specifications for each container. The amount of resources required by an elastic container instance refers to the total amount of resources required by all containers in the elastic container instance.

Each elastic container instance supports up to 20 containers. You can customize the resource specifications for each container. The total amount of resources that are specified for the elastic container instance cannot exceed the upper limits of CPU and memory resources.

- If the total amount of resources that are specified for the elastic container instance exceeds the upper limits of CPU and memory resources, the elastic container instance automatically adjusts the CPU and memory resources as required. Resource fees are charged based on the adjusted CPU and memory resources.
- For containers that are allocated more than four vCPUs, you must strictly declare the specifications of the CPU and memory resources to avoid resource allocation adjustment and waste of compute resources. Otherwise, an error message is returned, which indicates that the specifications are invalid.

The following table describes the supported CPU and memory specifications.

vCPU	Memory
0.25 vCPU	0.5 GB, 1 GB
0.5 vCPU	1 GB, 2 GB
1 vCPU	2 GB, 4 GB, 8 GB
2 vCPU	2 GB, 4 GB, 8 GB, 16 GB
4 vCPU	4 GB, 8 GB, 16 GB, 32 GB
8 vCPU	8 GB, 16 GB, 32 GB, 64 GB
12 vCPU	12 GB, 24 GB, 48 GB, 96 GB
16 vCPU	16 GB, 32 GB, 64 GB, 128 GB
24 vCPU	48 GB, 96 GB, 192 GB
32 vCPU	64 GB, 128 GB, 256 GB
52 vCPU	96 GB, 192 GB, 384 GB
64 vCPU	128 GB, 256 GB, 512 GB

Examples

For a Kubernetes-native container, you can directly configure the resource `request` of the `container` .

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
        ports:
        - containerPort: 80
        resources:
          requests:
            cpu: "500m"
            memory: "1024Mi"
      - name: busybox
        image: busybox:latest
        ports:
        - containerPort: 80
        resources:
          requests:
            cpu: "500m"
            memory: "1024Mi"
```

Specify the pod specifications for an elastic container instance

When you specify CPU and memory specifications for an elastic container instance, the instance attempts to use multiple Elastic Compute Service (ECS) instance types to match the specifications. This allows you to improve the scalability and success rate of creation than with only one ECS instance type. This provides the following benefits:

- You do not need to limit resources for containers in your elastic container instance. When you specify container resources for an elastic container instance, you do not need to configure the resource `request` or `limit`. This allows the containers in the elastic container instance to share the requested resources to the maximum extent.
- In genomics computing and Istio scenarios, the service framework automatically adds sidecar containers to pods. You can seamlessly integrate the elastic container instance with the service framework by explicitly specifying the instance specifications.

Examples

You can set annotations: k8s.aliyun.com/eci-use-specs in the pod configuration to specify multiple ECS instance types. Separate multiple ECS instance types with commas (,).

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment-basic
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
      annotations:
        k8s.aliyun.com/eci-use-specs : "2-4Gi"
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
        ports:
        - containerPort: 80
      - name: busybox
        image: busybox:latest
        ports:
        - containerPort: 80
        resources:
          limits:
            cpu: "500m"
            memory: "1024Mi"
```

5.3. Create an ECI from a specified ECS instance type

In some scenarios, custom needs are required, such as GPU capabilities, enhanced network capabilities, high clock speed, and local storage. You can create an Elastic Container Instance (ECI) from a specified Elastic Compute Service (ECS) instance type. This topic describes how to create an ECI from a specified ECS instance type.

Descriptions of ECS instance types

For detailed information about ECI-specific ECS instance types, see the descriptions of ECS instance types. The unit price of an ECI is the same as that of the corresponding ECS instance. Both are billed on a per-second basis. For more information, see ECS price calculator.

You can check the ECS instance types supported in each region and zone in Overview of ECS instance types available to purchase in each region and zone. The following ECS instance families are supported:

- g6 and g5, general purpose instance families (the CPU-to-memory ratio is 1:4). sn2ne, general purpose instance family (the CPU-to-memory ratio is 1:4) with enhanced network performance

- c6 and c5, compute optimized instance families (the CPU-to-memory ratio is 1:2). sn1ne: compute optimized instance family (the CPU-to-memory ratio is 1:2) with enhanced network performance
- r6 and r5, memory optimized instance families (the CPU-to-memory ratio is 1:8). se1ne, memory optimized instance family (the CPU-to-memory ratio is 1:8) with enhanced network performance
- ic5, compute intensive instance family (the CPU-to-memory ratio is 1:1)
- hfc6 and hfc5, compute optimized instance families with high clock speed (the CPU-to-memory ratio is 1:2)
- hfg6 and hfg5, general purpose instance families with high clock speed (the CPU-to-memory ratio is 1:4)
- gn6i, gn6v, gn5i, and gn5, compute optimized instance families with GPU capabilities (local storage is not supported)
- t5 and t6, burstable instance families

Examples

You can set annotations: k8s.aliyun.com/eci-use-specs in the definition of a pod to specify multiple ECS instance types. Separate ECS instance types with commas (,).

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
      annotations:
        k8s.aliyun.com/eci-use-specs : "ecs.c5.large"  #Replace the value with the ECS inst
ance types as needed.
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
        ports:
        - containerPort: 80
```

5.4. Use preemptible instances

Preemptible instances are cost-effective. You can bid for idle resources of Alibaba Cloud, obtain the resources, and then run containers until the container resources are reclaimed due to higher bids from other customers. This reduces the costs of Elastic Container Instances (ECIs) in some scenarios. This topic describes how to use preemptible instances.

Context

Preemptible instances are billed at a lower price compared with pay-as-you-go instances. The actual price of a preemptible instance changes based on the supply and demand. A preemptible instance is billed based on the actual period of usage. For more information, see Overview.

A preemptible instance has a guaranteed period of one hour after it is created. During this period, the instance is not released even if your bid is lower than the current market price. This ensures that the instance can run your services without interruptions. After the guaranteed period ends, the system checks the market price and the stock of the instance type every five minutes. If the market price is higher than your bid or if the stock of the instance type is insufficient, your preemptible instance is released.

- After your preemptible instance keeps running for one hour, it may be reclaimed by the system due to price fluctuations or insufficient resources.
- The system will notify you three minutes before a preemptible instance is released.
- Released preemptible instances are reclaimed. The instance information is retained. After a preemptible instance is reclaimed, the billing stops and its state changes to Expired.

When you use a preemptible instance, we recommend that you follow these suggestions:

- When you set your bid, consider market price fluctuations. Only if your bid is higher than bids from other customers, the request is accepted and processed, and the created instance will not be released due to price fluctuations. In addition, you must submit the price based on the expected requirements of your workloads.
- We recommend that you store important data in a storage medium that is not affected by the release of preemptible instances. For example, you can use a cloud disk that will not be released along with the preemptible instance, or choose external storage such as Network-attached storage (NAS).

You can create a preemptible instance in one of the following ways:

- Specify the type of Elastic Compute Service (ECS) instance.

  The preemptible instance is billed based on the market price and discount for a pay-as-you-go ECS instance of the specified instance type at the time of purchase.

- Specify CPU and memory specifications.

  ECI automatically matches ECS instance types that meet the price requirement. The market price of the selected instance type is adopted as the base market price. Discounts are applied based on the base market price instead of the pay-as-you-go price of CPU and memory resources of the corresponding ECI. Specifying CPU and memory specifications is equivalent to specifying the ECS instance type.

Scenarios

Preemptible instances are ideal for stateless applications, such as scalable web services, image rendering, big data analytics, and large-scale parallel computing. Preemptible instances can be applied to applications that require a high level of distribution, scalability, and fault tolerance capabilities. Preemptible instances help reduce costs and increase the throughput of these applications.

You can use preemptible instances in the following scenarios:

- Real-time analytics
- Big data processing
- Scalable websites

- Image and media encoding
- Scientific computing
- Geospatial surveys
- Web crawlers
- Tests

Procedure

You can set annotations: k8s.aliyun.com/eci-spot-strategy and annotations: k8s.aliyun.com/eci-spot-price-limit in a pod.

- SpotAsPriceGo: automatically submits bids based on the up-to-date market price.
- SpotWithPriceLimit: specifies the highest bid for the preemptible instance.

Details:

- Set `k8s.aliyun.com/eci-spot-strategy` to `"SpotAsPriceGo"` .
- `k8s.aliyun.com/eci-spot-price-limit` takes effect only when `k8s.aliyun.com/eci-spot-strategy` is set to `"SpotWithPriceLimit"` .

  Set the highest bid within an hour for the preemptible instance. The price can be accurate to three decimal places.

  - `k8s.aliyun.com/eci-spot-strategy: "SpotAsPriceGo"`
  - `k8s.aliyun.com/eci-spot-price-limit: "0.250"`

Deployment example

- SpotAsPriceGo strategy

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
      annotations:
        k8s.aliyun.com/eci-use-specs: "2-4Gi"  #Specify an ECS instance type or CPU and m
emory specifications based on requirements.
        k8s.aliyun.com/eci-spot-strategy: "SpotAsPriceGo"  #A value of SpotAsPriceGo spec
ifies a preemptible instance that is priced at the market price at the time of purchase.

    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
        ports:
        - containerPort: 80
```

- SpotWithPriceLimit strategy

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
      annotations:
        k8s.aliyun.com/eci-use-specs: "ecs.c5.large"  #Specify an ECS instance type or CP
U and memory specifications as needed.
        k8s.aliyun.com/eci-spot-strategy: "SpotWithPriceLimit"  #A value of SpotWithPrice
Limit specifies a preemptible instance with the highest hourly price.
        k8s.aliyun.com/eci-spot-price-limit: "0.250" #The highest hourly price.
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
        ports:
        - containerPort: 80
```

5.5. Use GPU-accelerated elastic container instances

GPU-accelerated elastic container instances feature built-in GPUs and Compute Unified Device Architecture (CUDA) drivers. Therefore, when you use a GPU-accelerated elastic container instance, you need to only use a base image that is preinstalled with software such as CUDA Toolkit. You do not need to manually install the GPU driver. This topic describes how to use a GPU-accelerated elastic container instance.

Context

The GPU driver version supported by GPU-accelerated elastic container instances is NVIDIA 460.73.01. The CUDA Toolkit version supported by GPU-accelerated elastic container instances is 11.2. For more information about CUDA Toolkit, see NVIDIA CUDA.

You can specify GPU-accelerated Elastic Compute Service (ECS) instance types to create GPU-accelerated elastic container instances. The following GPU-accelerated ECS instance types are supported:

- gn6v, a GPU-accelerated compute-optimized instance family that uses NVIDIA V100 GPUs. This instance family includes a variety of instance types, such as ecs.gn6v-c8g1.2xlarge.
- gn6i, a GPU-accelerated compute-optimized instance family that uses NVIDIA T4 GPUs. This instance family includes a variety of instance types, such as ecs.gn6i-c4g1.xlarge.

- gn5, a GPU-accelerated compute-optimized instance family that uses NVIDIA P100 GPUs. This instance family includes a variety of instance types, such as ecs.gn5-c4g1.xlarge.
- gn5i, a GPU-accelerated compute-optimized instance family that uses NVIDIA P4 GPUs. This instance family includes a variety of instance types, such as ecs.gn5i-c2g1.large.

For more information about GPU-accelerated ECS instance types, see Instance family.

Procedure

Add `annotations: k8s.aliyun.com/eci-use-specs` to the pod configuration.

- Specify the instance type that you want to use in the `annotations` field of the pod `metadata` .
- Specify the amount of GPU resources in the `resources` field of the container configuration.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-gpu-demo
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
      annotations:
        k8s.aliyun.com/eci-use-specs: ecs.gn5i-c4g1.xlarge
    spec:
      containers:
      - name: nginx
        image: registry-vpc.cn-beijing.aliyuncs.com/eci_open/nginx:1.15.10
        resources:
            limits:
              nvidia.com/gpu: '1'
        ports:
        - containerPort: 80
```

5.6. Use AMD instances

AMD instances are built on the SHENLONG architecture and offloads virtualization features to dedicated hardware. This provides predictable and consistent ultra-high performance and reduces virtualization overheads. This topic describes how to use AMD instances.

Scenarios

- Video encoding and decoding
- Receiving and sending large numbers of network packets
- Web frontend servers

- Frontend servers of massively multiplayer online (MMO) games
- Application testing and development, such as DevOps

Specifications

Elastic Container Instance allows you to specify an Elastic Compute Service (ECS) instance type to create AMD instances. The **c6a** compute-optimized instance family is supported. For example, you can specify the `ecs.c6a.large` instance type. For more information, see Instance family.

Usage notes

Add `annotations: k8s.aliyun.com/eci-use-specs` to the pod configuration.

Add the following `annotations` to the `metadata` field of the pod configuration to specify the instance type.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
      annotations:
        k8s.aliyun.com/eci-use-specs : "ecs.c6a.xlarge"  #Replace the value with the requir
ed ECS instance type.
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
        ports:
        - containerPort: 80
```

5.7. Use reserved instances

When you use an elastic container instance to provide long-term services, you can use reserved instances to offset the cost of the elastic container instance. This topic describes how to use reserved instances.

Prerequisites

1. Reserved instances are purchased based on your business requirements. For more information, see Purchase reserved instances.
2. Reserved instances are checked and managed. For more information, see Split a reserved instance.

Note Reserved instances are automatically applied to pay-as-you-go elastic container instances based on match rules. For more information about match rules, see Match between reserved instances and pay-as-you-go instances.

How to use reserved instances

A reserved instance is applicable only when you create an elastic container instance based on a specified Elastic Compute Service (ECS) instance type. For more information, see Create an ECI from a specified ECS instance type.

Add the following `annotation` to the `template` field of pod configurations. In this example, the reserved instance is used to purchase an `ecs.c5.large` instance.

```
annotations:
    k8s.aliyun.com/eci-instance-type : "ecs.c5.large"  # Replace the value with the actual
ECS instance type in the following format: ecs.instance family.instance specification. For
example, ecs.c6.3xlarge.
```

Note `annotations` must be added to the `spec` field of pod configurations. Specify the ECS instance type based on your requirements. For more information about the supported ECS instance types, see Instance family.

Deployment example

`annotations` must be added to the `metadata` field of pod configurations.

Note The zone of the reserved instance must be the same as that of your cluster. Otherwise, the reserved instance cannot match the elastic container instance created in your serverless Kubernetes (ASK) cluster.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
      annotations:
        k8s.aliyun.com/eci-instance-type : "ecs.c5.large"  # Replace the value with the act
ual ECS instance type based on your requirements.
      containers:
      - name: nginx
        image: nginx:1.7.9
        ports:
        - containerPort: 80
```

View the billing and application details about a reserved instance

1.
2.
3.
4. Find the reserved instance that you want to view and click **View Bill** in the **Actions** column.
5. On the **Manage Reserved Instances** page, click the **Details** tab.

   You can view the usage details of the reserved instance. These details show the ECS instance or elastic container instance whose fees are offset by the reserved instance during every billing hour.

   Note One computing unit equals one vCPU. The applicable duration (hours) equals the number of computing units multiplied by the number of hours.

5.8. Create ECIs across zones

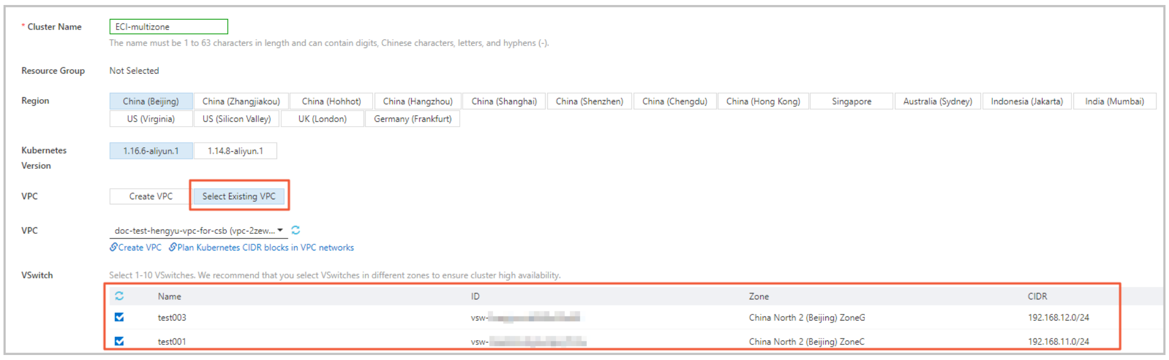
The system may exhaust instances of specified types or vSwitch IP resources in a zone when it attempts to create sufficient instances to handle traffic spikes or process a large number of jobs in parallel. As a result, you fail to create Elastic Container Instances (ECIs) in the zone. In this case, you can deploy a serverless Kubernetes (ASK) cluster across multiple zones to prevent resource shortage.

Context

- An ECI distributes the requests for creating pods to all vSwitches. This balances the heavy load.
- If the zone of a vSwitch does not have sufficient resources to create the required pods, the system selects another vSwitch.

Procedure

1. When you create an ASK cluster, you can select an existing virtual private cloud (VPC) and specify multiple vSwitches deployed in different zones. For more information about how to create a cluster, see Create an ASK cluster.



* Cluster Name: ECI-multizone
The name must be 1 to 63 characters in length and can contain digits, Chinese characters, letters, and hyphens (-).

Resource Group: Not Selected

Region: China (Beijing) | China (Zhangjiakou) | China (Hohhot) | China (Hangzhou) | China (Shanghai) | China (Shenzhen) | China (Chengdu) | China (Hong Kong) | Singapore | Australia (Sydney) | Indonesia (Jakarta) | India (Mumbai) | US (Virginia) | US (Silicon Valley) | UK (London) | Germany (Frankfurt)

Kubernetes Version: 1.16.6-aliyun.1 | 1.14.8-aliyun.1

VPC: Create VPC | Select Existing VPC

VPC: doc-test-hengyu-vpc-for-csb (vpc-2zew...
Create VPC Plan Kubernetes CIDR blocks in VPC networks

VSwitch: Select 1-10 VSwitches. We recommend that you select VSwitches in different zones to ensure cluster high availability.

	Name	ID	Zone	CIDR
✓	test003	vsw-[redacted]	China North 2 (Beijing) ZoneG	192.168.12.0/24
✓	test001	vsw-[redacted]	China North 2 (Beijing) ZoneC	192.168.11.0/24

2. Modify the configuration of zones for a cluster.

   After the ASK cluster is created, modify the vSwitch configuration for the cluster. You do not need to recreate the cluster. Run the following command and modify the `vswitch` field in `kube-system/eci-profile configmap` based on your business requirements. The modification immediately takes effect.

```
kubectl -n kube-system edit cm eci-profile
```

```
apiVersion: v1
data:
  kube-proxy: "true"
  privatezone: "true"
  quota-cpu: "192000"
  quota-memory: 640Ti
  quota-pods: "4000"
  region: cn-hangzhou
  resourcegroup: ""
  securitygroup: sg-xxx
  vpc: vpc-xxx
  vswitch: vsw-xxx,vsw-yyy,vsw-zzz
kind: ConfigMap
```

FAQ

- How do I add nodes to an ASK cluster?

  The concept of node is not applied to ASK clusters. To deploy ECIs in other zones, you can add a vSwitch of another zone to the eci-profile ConfigMap of the kube-system namespace. Separate vSwitches with commas (,).

- How do I deploy pods in different zones?

  By default, pods are randomly scheduled to zones that have sufficient resources.

  Match virtual kubelet. You can deploy ECIs of different instance types across zones by using the following three annotations:

  - `- k8s.aliyun.com/eci-schedule-strategy: " VSwitchOrdered"`

- `- k8s.aliyun.com/eci-vswitch: "vsw-11111, vsw-22222"`
- `- k8s.aliyun.com/eci-use-specs: "ecs.c5.4xlarge, ecs.c6.4xlarge,2-4Gi"`

`VSwitchOrdered` specifies that ECIs are created based on the vSwitch list in the `k8s.aliyun.com/eci-vswitch` annotation. You can set `k8s.aliyun.com/eci-schedule-strategy` to `VSwitchRandom` or `VSwitchOrdered`. When you create an ECI, the ECI is randomly scheduled to a zone or scheduled based on the vSwitch list specified by the `k8s.aliyun.com/eci-vswitch` annotation.

5.9. ECI Pod Annotation

When you create elastic container instance-based pods in a Kubernetes cluster, you can add annotations to the pods to use the features of Elastic Container Instance. Make sure that the annotations that you want to add comply with the Kubernetes syntax. This topic describes the annotations that are supported by elastic container instance-based pods. This topic also provides examples on how to configure the annotations.

The following table describes the annotations that are supported by elastic container instance-based pods.

Note

- The annotations described in the following table are applicable only to the pods that are scheduled to virtual nodes. These pods run on elastic container instances. The annotations cannot be added to the pods that are scheduled to regular nodes.
- Add annotations to the metadata field of the pods. For example, when you configure a Deployment, add annotations in the spec.template.metadata field.

Annotation	Example	Description	Reference
k8s.aliyun.com/eci-security-group	sg-bp1dktddjsg5nktv****	The ID of the security group.	Configure a security group
k8s.aliyun.com/eci-vswitch	vsw-bp1xpiowfm5vo8o3c****	The IDs of the vSwitches. You can specify multiple vSwitches across zones.	Specify multiple zones to create an elastic container instance-based pod
k8s.aliyun.com/eci-schedule-strategy	VSwitchOrdered	The multi-zone scheduling policy. Valid values: • VSwitchOrdered: Resources in the specified zones are scheduled based on the order in which the vSwitches are specified. • VSwitchRandom: Resources in the specified zones are randomly scheduled.	

Annotation	Example	Description	Reference
k8s.aliyun.com/eci-ram-role-name	AliyunECIContainerGroupRole	The Resource Access Management (RAM) role that elastic container instances assume to access other Alibaba Cloud services.	None. The following section describes the details.
k8s.aliyun.com/eci-use-specs	2-4Gi,4-8Gi,ecs.c6.xlarge	The types of elastic container instances. You can specify multiple elastic container instance specifications, such as the number of vCPUs and the memory size. You can also specify an ECS instance type.	Specify multiple instance specifications to create an elastic container instance
k8s.aliyun.com/eci-spot-strategy	SpotAsPriceGo	The bidding policy of the preemptible instance. Valid values: • SpotAsPriceGo: The instance is billed at the market price at the time of purchase. • SpotWithPriceLimit: You must specify the highest price that you want to pay for the preemptible instance.	Create a preemptible instance
k8s.aliyun.com/eci-spot-price-limit	0.5	The highest price of the preemptible instance. This parameter is valid only when k8s.aliyun.com/eci-spot-strategy is set to SpotWithPriceLimit.	
k8s.aliyun.com/eci-cpu-option-core	2	The number of physical CPU cores.	Customize CPU options
k8s.aliyun.com/eci-cpu-option-ht	1	The number of threads per core.	
k8s.aliyun.com/eci-reschedule-enable	"true"	Specifies whether to enable rescheduling for elastic container instances.	None. The following section describes the details.
k8s.aliyun.com/pod-fail-on-create-err	"true"	Specifies whether to put the elastic container instances that cannot be created into the Failed state.	None. The following section describes the details.

Annotation	Example	Description	Reference
k8s.aliyun.com/eci-image-snapshot-id	imc-2zebxkiifuyzzlhl****	The ID of the image cache. **Note** To use an image cache to create an elastic container instance, you can specify the image cache that you want to use or enable automatic matching for image caches. We recommend that you enable automatic matching for image caches.	Use the ImageCache CRD to accelerate pod creation
k8s.aliyun.com/eci-image-cache	"true"	Specifies whether to enable automatic matching for image caches. **Note** To use an image cache to create an elastic container instance, you can specify the image cache that you want to use or enable automatic matching for image caches. We recommend that you enable automatic matching for image caches.	
k8s.aliyun.com/acr-instance-id	cri-j36zhodptmyq****	The ID of the Container Registry Enterprise Edition instance. You can specify a Container Registry Enterprise Edition instance that resides in a region different from the region of the elastic container instance. To do this, you must add the region name of the Container Registry Enterprise Edition instance before the ID of the Container Registry Enterprise Edition instance. Example: `"cn-beijng:cri-j36zhodptmyq****"` .	Pull images from a Container Registry Enterprise Edition instance without a password

Annotation	Example	Description	Reference
k8s.aliyun.com/eci-eip-instanceid	eip-bp1q5n8cq4p7f6dzu****	The ID of the elastic IP address (EIP).	Enable Internet access
k8s.aliyun.com/eci-with-eip	"true"	Specifies whether to automatically create an EIP and associate the EIP with the elastic container instance.	
k8s.aliyun.com/eip-bandwidth	5	The bandwidth of the EIP.	
k8s.aliyun.com/eip-common-bandwidth-package-id	cbwp-2zeukbj916scmj51m****	The ID of the EIP bandwidth plan.	
k8s.aliyun.com/eip-isp	BGP	The line type of the EIP. This annotation is applicable only to pay-as-you-go EIPs. Valid values: • BGP: BGP (Multi-ISP) line • BGP_PRO: BGP (Multi-ISP) Pro line	
k8s.aliyun.com/eip-internet-charge-type	PayByBandwidth	The metering method of the EIP. Valid values: • PayByBandwidth: Pay-by-bandwidth • PayByTraffic: Pay-by-traffic	
k8s.aliyun.com/eci-enable-ipv6	"true"	Specifies whether to assign IPv6 addresses.	Assign an IPv6 address to an elastic container instance
kubernetes.io/ingress-bandwidth	40M	The inbound bandwidth.	Set bandwidth throttling for an elastic container instance
kubernetes.io/egress-bandwidth	20M	The outbound bandwidth.	

Annotation	Example	Description	Reference
k8s.aliyun.com/eci-extra-ephemeral-storage	50Gi	The temporary storage capacity.	Customize the temporary storage capacity
k8s.aliyun.com/eci-core-pattern	/pod/data/dump/core	The directory in which core dump files are stored.	View core dump files
k8s.aliyun.com/eci-ntp-server	100.100.*.*	The IP address of the Network Time Protocol (NTP) server.	Configure the NTP service for pods
k8s.aliyun.com/plain-http-registry	"harbor***.pre.com,192.168.XX.XX:5000,reg***.test.com:80"	The IP address of the self-managed image repository. When you create an elastic container instance by using an image in a self-managed image repository over HTTP, you must add this annotation to the instance. This allows Elastic Container Instance to pull the image over HTTP. This prevents image pull failures caused by different protocols.	Use self-managed image repositories
k8s.aliyun.com/insecure-registry	"harbor***.pre.com,192.168.XX.XX:5000,reg***.test.com:80"	The endpoint of the self-managed image repository. When you create an elastic container instance by using an image in a self-managed image repository that uses a self-signed certificate, you must add this annotation to the instance to skip the certificate authentication. This prevents image pull failures caused by certificate authentication failures.	

Configure a RAM role

You can add an annotation to configure a RAM role for a pod and grant the pod the permissions to access Alibaba Cloud services.

Notice

- Before you add the annotation, you must create a RAM role and grant permissions to the RAM role. When you create the RAM role, make sure that the trusted service of the RAM role is ECS.
- If you use a RAM user, make sure that the RAM user has the `ram:passRole` permission.

Example:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: set-ram-role
  labels:
    app: vk
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
        annotations:
            k8s.aliyun.com/eci-ram-role-name : "${your_ram_role_name}"
        labels:
            app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:latest
      nodeName: virtual-kubelet
```

Configure rescheduling for elastic container instances

Pods may fail to be scheduled to virtual nodes. You can add an annotation to enable rescheduling for pods. This ensures that the system keeps scheduling pods instead of returning failures even if the asynchronous scheduling fails.

Example:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: set-eci
  labels:
    app: vk
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
        annotations:
            k8s.aliyun.com/eci-reschedule-enable: "true"    # Enable rescheduling for elast
ic container instances.
        labels:
            app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:latest
      nodeName: virtual-kubelet
```

Put the pods that fail to be created into the Failed state

By default, if an error occurs when a pod is being created, the system attempts to create the pod for a specified number of times. If the pod fails to be created after the specified number of times, the pod changes to the Pending state. You may want pods to enter the Failed state if the pods fail to be created for specific Jobs. In this case, you can add an annotation to put a pod that fails to be created into the Failed state.

Example:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: set-pod-fail-on-create-err
  labels:
    app: vk
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
        annotations:
            k8s.aliyun.com/pod-fail-on-create-err: "true"  # Set the state to Failed if the
pod fails to be created.
        labels:
            app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:latest
      nodeName: virtual-kubelet
```

5.10. Configure NTP for pods

This topic describes how to configure the Network Time Protocol (NTP) service for pods that run on a Virtual Kubelet node. The NTP service synchronizes the time between containers and the NTP server.

Prerequisites

Virtual Kubelet is updated to the latest version. For more information, see Update Virtual Kubelet.

Background information

The operations for updating Virtual Kubelet vary based on the type of Kubernetes cluster that you use.

- For serverless Kubernetes (ASK) clusters, the system automatically updates Virtual Kubelet.
- Container Service for Kubernetes (ACK) clusters include ACK managed clusters and ACK dedicated clusters. For ACK managed clusters, the system automatically updates Virtual Kubelet. For ACK dedicated clusters, you must manually update Virtual Kubelet.
- For self-managed clusters that are deployed on Elastic Compute Service (ECS) instances or on-premises clusters, you must manually update Virtual Kubelet.

Procedure

Add the `k8s.aliyun.com/eci-ntp-server` annotation to the pod. The annotation specifies the IP address of the NTP server that you want to use.

1. Create a YAML file to configure the NTP service.

   ```
   vim set-ntp-pod.yaml
   ```

The YAML file contains the following content:

```
apiVersion: v1
kind: Pod
metadata:
  annotations:
    k8s.aliyun.com/eci-ntp-server: 10.10.5.1 # The IP address of the NTP server.
  name: set-custom-ntp
spec:
  nodeName: virtual-kubelet
  containers:
    - image: centos:latest
      command:
        - sleep
        - "3600"
      imagePullPolicy: IfNotPresent
      name: centos
```

2. Use the configurations in the YAML file for the pod.

```
kubectl apply -f set-ntp-pod.yaml
```

Verify the result

Log on to the container and check whether the NTP service works as expected.

1. Query the information about the pod.

```
kubectl get pod/set-custom-ntp
```

Expected output:

```
NAME                      READY    STATUS     RESTARTS    AGE
set-custom-ntp    1/1          Running    0                  7m20s
```

2. Log on to the container.

```
kubectl exec set-custom-ntp -it -- bash
```

3. Query the source from which the time of container is synchronized.

```
chronyc sources
```

If the IP address of the NTP server is returned, the NTP service works as expected. Expected output:

```
210 Number of sources = 1
MS Name/IP address         Stratum Poll Reach LastRx Last                            sa
mple
===============================================================================
^* 10.10.5.1                    2           6     377     35       +40us[ +135us] +/-
14ms
```

5.11. Specify time zones for pods

This topic describes how to configure different time zones for pods that run on Virtual Kubelet. When you deploy pods for an application, you can specify different time zones for the pods.

Prerequisites

Virtual Kubelet is updated to the latest version. For more information, see Update Virtual Kubelet.

Background information

The method that is used to update Virtual Kubelet varies based on the type of the Kubernetes cluster:

- Serverless Kubernetes (ASK) cluster: You must contact the administrator to update Virtual Kubelet.
- ACK managed cluster: You must manually update Virtual Kubelet.
- ACK dedicated cluster: You must manually update Virtual Kubelet.
- Self-managed cluster: You must manually update Virtual Kubelet.

Procedure

1. Create a ConfigMap to import the time zone settings that you want to use.

   For other time zones, use the corresponding files in the /usr/share/zoneinfo/Asia/ directory. Example:

   ```
   kubectl create configmap tz --from-file=/usr/share/zoneinfo/Asia/Shanghai
   ```

2. Create a YAML file to configure the time zone.

   ```
   vim set-timezone.yaml
   ```

   Mount the ConfigMap to the /etc/localtime/Shanghai directory. Example:

   ```
   apiVersion: v1
   kind: Pod
   metadata:
     name: timezone
   spec:
     containers:
     - name: timezone
       image: registry-vpc.cn-beijing.aliyuncs.com/eci_open/busybox:1.30
       command: [ "sleep", "10000" ]
       volumeMounts:
         - name: tz
           mountPath: /etc/localtime
           subPath: Shanghai
     volumes:
       - name: tz
         configMap:
           name: tz
     nodeSelector:
       type: virtual-kubelet
     tolerations:
     - key: virtual-kubelet.io/provider
       operator: Exists
   ```

3. Deploy a pod with the YAML file that you created.

```
kubectl apply -f set-timezone.yaml
```

Verify the result

Log on to the container and check whether the time zone of the container is as you configured.

1. Query information about the pod.

   ```
   kubectl get pod/timezone
   ```

   Expected output:

   ```
   NAME          READY    STATUS     RESTARTS    AGE
   timezone      1/1      Running    0           7m20s
   ```

2. Log on to the container.

   ```
   kubectl exec timezone -it -- sh
   ```

3. Query the time zone of the container.

   ```
   date -R
   ```

   If the time returned is consistent with the time zone that you configured, the time zone is configured for the container. Expected output:

   ```
   Fri, 01 May 2020 10:00:11 +0800
   ```

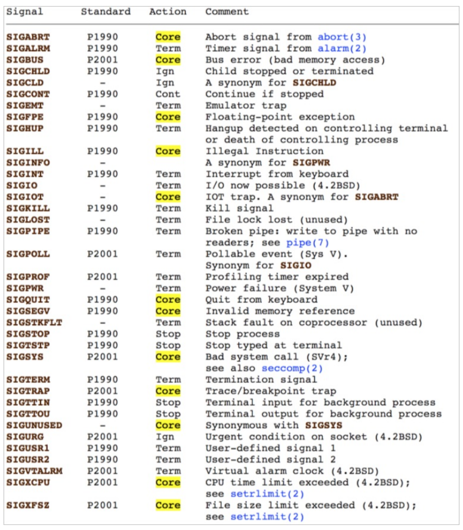
5.12. Enable the core dump feature

This topic describes how to enable the core dump feature. You can view and analyze a core dump file to identify the cause of the issue when a container is unexpectedly terminated.

Background information

In Linux, if a program unexpectedly terminates or exits, the operating system records the state of the memory that is allocated to the program and saves the state to a file. This process is called a core dump. You can view and analyze the core dump file to identify the cause of an issue.

The following figure shows the signals that indicate core dumps in Linux. The values of the signals in the Action column are Core.



Signal	Standard	Action	Comment
SIGABRT	P1990	Core	Abort signal from abort(3)
SIGALRM	P1990	Term	Timer signal from alarm(2)
SIGBUS	P2001	Core	Bus error (bad memory access)
SIGCHLD	P1990	Ign	Child stopped or terminated
SIGCLD	-	Ign	A synonym for SIGCHLD
SIGCONT	P1990	Cont	Continue if stopped
SIGEMT	-	Term	Emulator trap
SIGFPE	P1990	Core	Floating-point exception
SIGHUP	P1990	Term	Hangup detected on controlling terminal or death of controlling process
SIGILL	P1990	Core	Illegal Instruction
SIGINFO	-		A synonym for SIGPWR
SIGINT	P1990	Term	Interrupt from keyboard
SIGIO	-	Term	I/O now possible (4.2BSD)
SIGIOT	-	Core	IOT trap. A synonym for SIGABRT
SIGKILL	P1990	Term	Kill signal
SIGLOST	-	Term	File lock lost (unused)
SIGPIPE	P1990	Term	Broken pipe: write to pipe with no readers; see pipe(7)
SIGPOLL	P2001	Term	Pollable event (Sys V). Synonym for SIGIO
SIGPROF	P2001	Term	Profiling timer expired
SIGPWR	-	Term	Power failure (System V)
SIGQUIT	P1990	Core	Quit from keyboard
SIGSEGV	P1990	Core	Invalid memory reference
SIGSTKFLT	-	Term	Stack fault on coprocessor (unused)
SIGSTOP	P1990	Stop	Stop process
SIGTSTP	P1990	Stop	Stop typed at terminal
SIGSYS	P2001	Core	Bad system call (SVr4); see also seccomp(2)
SIGTERM	P1990	Term	Termination signal
SIGTRAP	P2001	Core	Trace/breakpoint trap
SIGTTIN	P1990	Stop	Terminal input for background process
SIGTTOU	P1990	Stop	Terminal output for background process
SIGUNUSED	-	Core	Synonymous with SIGSYS
SIGURG	P2001	Ign	Urgent condition on socket (4.2BSD)
SIGUSR1	P1990	Term	User-defined signal 1
SIGUSR2	P1990	Term	User-defined signal 2
SIGVTALRM	P2001	Term	Virtual alarm clock (4.2BSD)
SIGXCPU	P2001	Core	CPU time limit exceeded (4.2BSD); see setrlimit(2)
SIGXFSZ	P2001	Core	File size limit exceeded (4.2BSD); see setrlimit(2)

For more information, see Core dump file.

Overview

By default, the core dump feature is disabled in elastic container instances to prevent service unavailability that is caused by excessive disk usage. You can use one of the following methods to enable the core dump feature:

- Method 1: Specify a path to store core files

  Elastic Container Instance allows you to use an external storage device to store core files. After you specify the storage device, the core dump feature is automatically enabled for the elastic container instance. If a container unexpectedly terminates or exits, the core dump feature is triggered to generate a core file. The core file is saved to the specified external storage device.

- Method 2: Execute a core dump O&M task

  After you enable the core dump feature, the system generates an O&M task. If a container unexpectedly terminates or exits, the core dump feature is triggered to generate a core file. The core file is automatically saved to an Object Storage Service (OSS) bucket.

Note

- Method 1 requires an external storage device. You can use this method to generate core files even if the program is unstable. However, if the program repeatedly restarts, a large number of core files may be generated.
- Method 2 is easy to use, but has limits on the validity periods and regions that you specify. Method 2 is suitable for temporary program debugging and diagnostics.
  - An O&M task that is generated by using Method 2 can be executed only once. After the O&M task is executed and a core file is generated, the core dump feature is disabled. The validity period of the O&M task is 12 hours.
  - Method 2 is based on OSS and Message Service (MNS). You cannot use Method 2 in the following regions because these regions support Elastic Container Instance but do not support OSS or MNS: China (Beijing), China (Ulanqab), China (Heyuan), China (Guangzhou), and China (Nanjing).

Method 1: Specify a path to store core files

Description

Elastic Container Instance allows you to specify the path in which core files are stored. After you specify a path, the core dump feature is automatically enabled. Core files are used to analyze issues offline. In most cases, core files are stored on external storage devices instead of the local storage of containers. This prevents the loss of core files if the containers terminate. You can use one of the following methods to specify a path to store core files:

Notice

The path cannot start with a vertical bar (`|`). You cannot use core dump files to configure executable programs.

- Kubernetes

  When you create a pod in Kubernetes, you can add an annotation to specify the path where core files are saved.

```
annotations:
    k8s.aliyun.com/eci-core-pattern: "/xx/xx/core"
```

- OpenAPI

  When you call the CreateContainerGroup API operation to create an elastic container instance, you can use the CorePattern parameter to specify the path where core files are saved:

```
CorePattern = "/xx/xx/core"
```

Example

You can call the following API operation to mount Apsara File Storage (NAS) file systems as external storage devices to store core files.

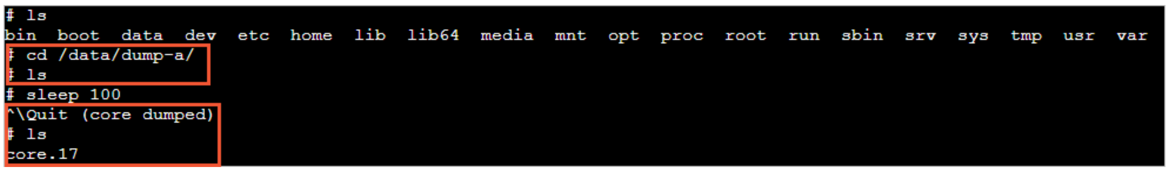
1. Create Elastic Container Instance A, mount the NAS file system to Elastic Container Instance A, and specify a path to store core files.

   When you call the CreateContainerGroup API operation to create Elastic Container Instance A, configure the following parameters to mount the `/dump/` directory of the NAS file system to the `/data/dump-a/` directory of the container and specify the `/data/dump/core` path to store core files.

```
'Volume.1.Name': 'volume1',
'Volume.1.Type': 'NFSVolume',
'Volume.1.NFSVolume.Path': '/dump/',
'Volume.1.NFSVolume.Server': '143b24****-gfn3.cn-beijing.nas.aliyuncs.com',

'Container.1.VolumeMount.1.Name': 'volume1',
'Container.1.VolumeMount.1.MountPath': '/data/dump-a/',

'CorePattern':'/data/dump-a/core',
```

2. Trigger the core dump feature in a directory of the container on Elastic Container Instance A.

   For example, after you run the `sleep 100` command in the container, you can press the `Ctrl` key and the `\` key at the same time to trigger the core dump feature. The core file is saved to the `/data/dump-a/` path of the container.



```
# ls
bin  boot  data  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
# cd /data/dump-a/
# ls
# sleep 100
^\Quit (core dumped)
# ls
core.17
```

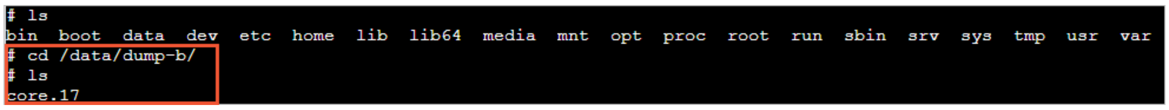
3. Release Elastic Container Instance A.
4. Mount the same NAS file system to Elastic Container Instance B.

   When you call the CreateContainerGroup API operation to create Elastic Container Instance B, specify the following parameters to mount the `/dump/` directory of the NAS file system to the `/data/dump-b/` directory of the container.

```
'Volume.1.Name': 'volume1',
'Volume.1.Type': 'NFSVolume',
'Volume.1.NFSVolume.Path': '/dump/',
'Volume.1.NFSVolume.Server': '143b24****-gfn3.cn-beijing.nas.aliyuncs.com',

'Container.1.VolumeMount.1.Name': 'dvolume1',
'Container.1.VolumeMount.1.MountPath': '/data/dump-b/',
```

5. View the core file in the container of Elastic Container Instance B.

   The following figure shows that the core file is stored in the `/data/dump-b/` path of the container. The core file is not lost after Elastic Container Instance A is released because the core file is stored in an external storage device.



```
# ls
bin  boot  data  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
# cd /data/dump-b/
# ls
core.17
```

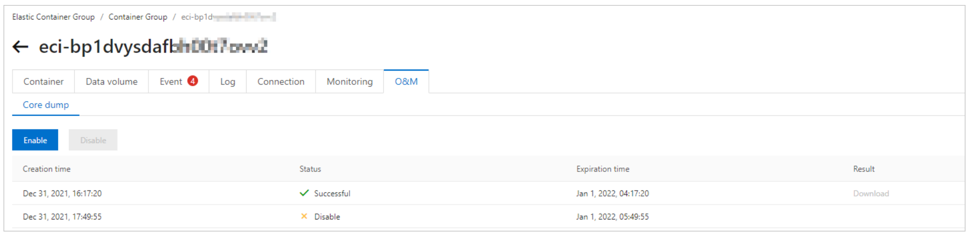
Method 2: Execute a core dump O&M task

Use the Elastic Container Instance console

1. Log on to the Elastic Container Instance console and create an elastic container instance.
2. Enable the core dump feature.
   i. Click the instance ID to go to the instance details page.

ii. Click the **O&M** tab and click **Enable**.

After you click **Enable**, the system generates an O&M task. Before the core dump feature is triggered, the task is in the **Pending** state.



Elastic Container Group / Container Group / eci-bp1d[redacted]

← eci-bp1dvysdafb[redacted]

Container | Data volume | Event 4 | Log | Connection | Monitoring | O&M

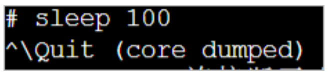
Core dump

Enable | Disable

Creation time	Status	Expiration time	Result
Dec 31, 2021, 16:17:20	✓ Successful	Jan 1, 2022, 04:17:20	Download
Dec 31, 2021, 17:49:55	✕ Disable	Jan 1, 2022, 05:49:55	

3. Trigger the core dump feature.

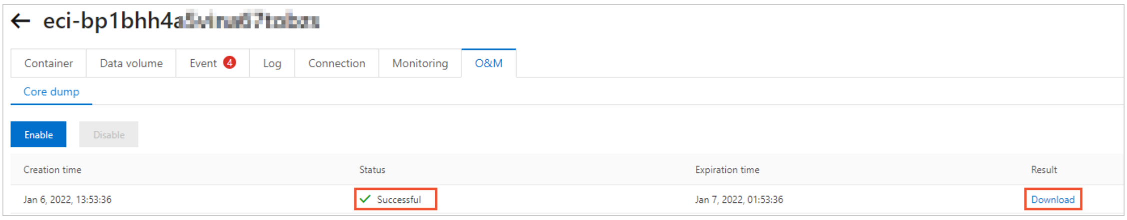
After you run the following `sleep 100` command in the container, press the `Ctrl` key and the `\` key at the same time to trigger the core dump feature. A core file is automatically generated and saved to an OSS bucket.



```
# sleep 100
^\Quit (core dumped)
```

4. Download the core file.

On the **O&M** tab of the instance details page, you can view the O&M task that is generated for the core dump. After the core dump feature is triggered and the system generates a core file, the status of the O&M task changes to **Successful**. Click **Download** in the Result column to download the core file to your on-premises machine.



← eci-bp1bhh4a[redacted]

Container | Data volume | Event 4 | Log | Connection | Monitoring | O&M

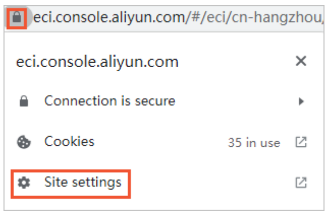
Core dump

Enable | Disable

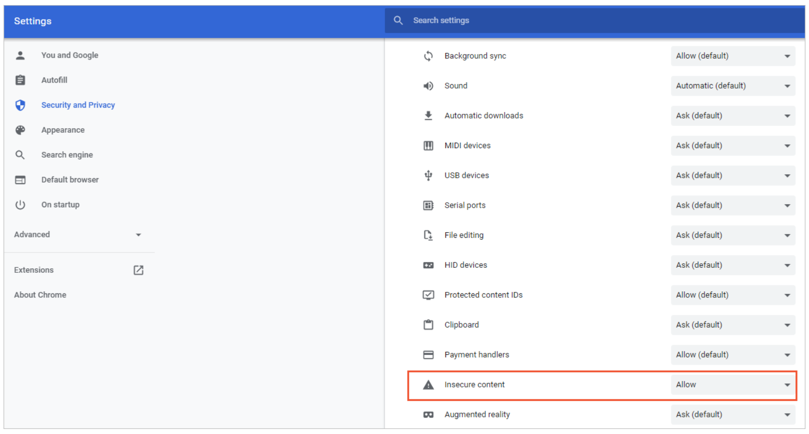
Creation time	Status	Expiration time	Result
Jan 6, 2022, 13:53:36	✓ Successful	Jan 7, 2022, 01:53:36	Download

If the system does not respond, check the website permission settings of your browser. For example, if you use Google Chrome, you can use the following method to allow the download:

i. In the Elastic Container Instance console, click the 🔒 icon in the address bar of your browser, and then select **Site settings**.



ii. Change the settings of the configuration item **Insecure content** to **Allow**.



Settings

Search settings

- You and Google
- Autofill
- Security and Privacy
- Appearance
- Search engine
- Default browser
- On startup

Advanced

Extensions

About Chrome

Background sync	Allow (default)
Sound	Automatic (default)
Automatic downloads	Ask (default)
MIDI devices	Ask (default)
USB devices	Ask (default)
Serial ports	Ask (default)
File editing	Ask (default)
HID devices	Ask (default)
Protected content IDs	Allow (default)
Clipboard	Ask (default)
Payment handlers	Allow (default)
Insecure content	Allow
Augmented reality	Ask (default)

OpenAPI

1. Create an elastic container instance.

   When you call the CreateContainerGroup API operation to create an elastic container instance, do not specify the CorePattern parameter.

2. Enable the core dump feature.

   Call the CreateInstanceOpsTask API operation to create an O&M task. Set OpsType to `coredump` and OpsValue to `enable` to enable the core dump feature. For more information, see CreateInstanceOpsTask.

3. Trigger the core dump feature.

4. Download the core file.

   Call the DescribeInstanceOpsRecords API operation to query the result of the O&M task that is executed. The ResultContent response parameter specifies the OSS path of the core file. You can use the path to download the core file.

6.Cluster

6.1. Create an ASK cluster

This topic describes how to create a serverless Kubernetes (ASK) cluster in the Container Service for Kubernetes (ACK) console.

Prerequisites

- ACK and related cloud services are activated. Required default roles are assigned to ACK. For more information, see Quick start for first-time users.
- Elastic Container Instance is activated in the Elastic Container Instance console.

Procedure

1.
2.
3.
4. On the page that appears, click the **Serverless Kubernetes** tab and set the parameters.

Parameter	Description
Cluster Name	Enter a name for the cluster. Note The name must be 1 to 63 characters in length, and can contain digits, letters, and hyphens (-).
	Select **Professional** to create an ASK Pro cluster. For more information, see ASK Pro cluster overview.
Region	Select a region to deploy the cluster.
Kubernetes Version	The Kubernetes versions that are supported by ASK are displayed.
	You can select Enable to create an ASK cluster that has IPv4/IPv6 dual stack enabled. This feature is in public preview. To use this feature, . Notice ◦ Only 1.20.11-aliyun.1 and later versions support IPv4/IPv6 dual stack. ◦ To enable IPv4/IPv6 dual stack for an ASK cluster, you must first enable IPv4/IPv6 dual stack for the virtual private cloud (VPC) where you want to deploy the cluster.

Parameter	Description
VPC	Set the VPC where you want to deploy the cluster. Kubernetes clusters support only VPCs. You can select **Create VPC** or **Selecting Existing VPC**. ◦ **Create VPC**: If you select this option, ACK automatically creates a VPC and a NAT gateway in the VPC. ACK also configures SNAT rules on the NAT gateway. ◦ **Select Existing VPC**: If you select this option, you must select a VPC from the VPC drop-down list and select vSwitches in the vSwitch section. If you want to enable Internet access, for example, to download container images, you must configure a NAT gateway. We recommend that you upload container images to a Container Registry instance in the region where the cluster is deployed. This way, you can pull images through the VPC. For more information, see Create and manage a VPC.
Zone	Select the zone where you want to deploy the cluster.
Configure SNAT	Specify whether to automatically create a NAT gateway and configure SNAT rules on the NAT gateway. This parameter is required only when you select **Create VPC** for **VPC**. **Note** After you select **Create VPC**, you can select or clear Configure SNAT. If you clear this check box, you must manually create a NAT gateway and configure SNAT rules on the NAT gateway. Otherwise, clusters that are deployed in the VPC cannot access the Internet. For more information, see Create and manage Internet NAT gateways.
Service CIDR	

Parameter	Description
	By default, an internal-facing Server Load Balancer (SLB) instance is created for the Kubernetes API server of an ASK cluster. You can modify the specification of the SLB instance. For more information, see Instance types and specifications. **Notice** If you delete the SLB instance, you cannot access the Kubernetes API server of the cluster. Select or clear **Expose API Server with EIP**. The Kubernetes API server provides multiple HTTP-based RESTful APIs, which can be used to create, delete, modify, query, and monitor resources, such as pods and Services. ◦ If you select this check box, an elastic IP address (EIP) is created and associated with the SLB instance. In this case, the Kubernetes API server of the cluster is exposed to the Internet through port 6443 of the EIP. You can use kubeconfig files to connect to and manage the cluster over the Internet. ◦ If you clear this check box, no EIP is created. You can connect to and manage the cluster by using kubeconfig files only from within the VPC. For more information, see Control public access to the API server of a cluster.
Service Discovery	Configure service discovery for the cluster. You can select **Disable**, **PrivateZone**, or **CoreDNS**. **Note** ◦ Alibaba Cloud DNS PrivateZone is a DNS resolution service for private domain names within VPCs. You can use Alibaba Cloud DNS PrivateZone to resolve private domain names to IP addresses in one or more VPCs. ◦ CoreDNS is a flexible and scalable DNS server that serves as a standard service discovery component in Kubernetes.
Ingress	Specify whether to install an Ingress controller. You can select **Do Not Install**, **Nginx Ingress**, or **ALB Ingress**. ◦ Nginx Ingress: The NGINX Ingress controller is optimized based on open source ingress-nginx and provides flexible and reliable routing services based on Ingresses. For more information, see Overview. ◦ ALB Ingress: The Application Load Balancer (ALB) Ingress controller is compatible with the NGINX Ingress controller, and provides improved traffic routing capabilities based on ALB instances. The ALB Ingress controller supports complex routing, automatic certificate discovery, and HTTP, HTTPS, and QUIC protocols. The ALB Ingress controller meets the requirements of cloud-native applications for ultra-high elasticity and balancing of heavy traffic loads at Layer 7. For more information, see ALB Ingress overview.

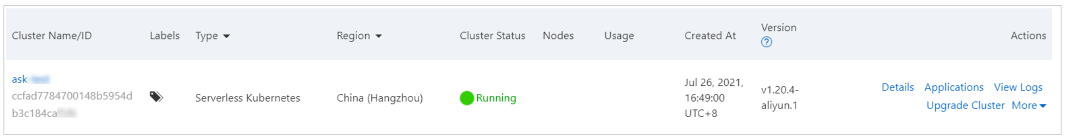
Parameter	Description
Monitoring Service	Specify whether to install metrics-server as the basic monitoring component in the cluster. The metrics-server component is a resource monitoring tool that ACK develops based on open source Metrics Server. metrics-server collects resource usage metrics for all pods in your cluster and enables Horizontal Pod Autoscaler (HPA) to work based on the collected metrics. You can call the Metrics API to retrieve monitoring metrics.
Log Service	Specify whether to enable Log Service. You can select an existing Log Service project or create one. If Log Service is disabled, you cannot use the cluster auditing feature. For more information about Log Service, see Getting Started.
Knative	Specify whether to enable Knative. Knative is a Kubernetes-based serverless framework. Knative is intended for developing a cloud-native, cross-platform orchestration standard for serverless applications. For more information, see Overview.
Deletion Protection	Specify whether to enable deletion protection for the cluster. Deletion protection prevents the cluster from being accidentally deleted in the console or by calling the API. This prevents user errors.
Resource Group	Move the pointer over **All Resources** at the top of the page and select the resource group to which the cluster belongs. The name of the selected resource group is displayed.
Labels	Add labels to the cluster. Enter a key and a value, and then click **Add**. Note ◦ Key is required. *Value* is optional. ◦ Keys are not case-sensitive. A key must be 1 to 64 characters in length, and cannot start with aliyun, http://, or https://. ◦ *Values* are not case-sensitive. A value cannot exceed 128 characters in length, and cannot contain http:// or https://. A value can be empty. ◦ The keys of labels that are added to the same resource must be unique. If you add a label with a used key, this label overwrites the label that uses the same key. ◦ If you add more than 20 labels to a resource, all labels become invalid. You must remove excess labels for the remaining labels to take effect.
Terms of Service	You must read and select **Terms of Service for Serverless Kubernetes** before you create the cluster.

5. On the right side of the page, click **Create Cluster**. In the **Confirm** message, click **OK** to start the

deployment.

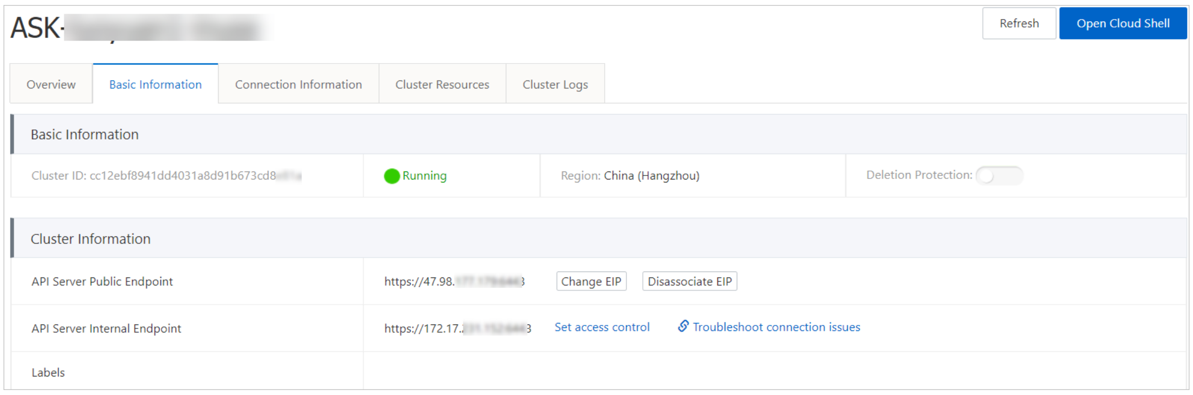
What's next

- After the cluster is created, you can find the ASK cluster on the Clusters page in the console.



Cluster Name/ID	Labels	Type	Region	Cluster Status	Nodes	Usage	Created At	Version	Actions
ask-[redacted] ccfad7784700148b5954d b3c184ca[redacted]		Serverless Kubernetes	China (Hangzhou)	● Running			Jul 26, 2021, 16:49:00 UTC+8	v1.20.4-aliyun.1	Details Applications View Logs Upgrade Cluster More

- On the **Clusters** page, find the cluster that you created and click **Details** in the Actions column. On the details page, click the **Basic Information** tab to view basic information about the cluster and click the **Connection Information** tab to view information about how to connect to the cluster.



Basic Information

Cluster ID: cc12ebf8941dd4031a8d91b673cd8[redacted]	● Running	Region: China (Hangzhou)	Deletion Protection:

Cluster Information

API Server Public Endpoint	https://47.98.[redacted]3 Change EIP Disassociate EIP
API Server Internal Endpoint	https://172.17.[redacted]3 Set access control Troubleshoot connection issues
Labels	

6.2. Create an ASK cluster by using Alibaba Cloud CLI

Alibaba Cloud command-line interface (CLI) is a management tool that is developed based on Alibaba Cloud APIs. You can use Alibaba Cloud CLI to call the APIs of Alibaba Cloud services.

Prerequisites

Before you use Alibaba Cloud CLI, you must configure the settings that are required to call Alibaba Cloud resources, including the credentials, regions, and language. For more information, see Overview.

Install and set up Alibaba Cloud CLI and kubectl

By default, Cloud Shell is preinstalled with Alibaba Cloud CLI and configured with the account information. You do not need to make any configuration changes. If you do not use Cloud Shell, you must install and set up Alibaba Cloud CLI and kubectl.

1. Install Alibaba Cloud CLI.
   - Install Alibaba Cloud CLI in Linux. For more information, see Linux.
   - Install Alibaba Cloud CLI in macOS.
     - macOS.

- You can also use a package manager to install Alibaba Cloud CLI in macOS. For more information, see Homebrew. After the package manager is installed, run the following command to install Alibaba Cloud CLI:

  ```
  brew install aliyun-cli
  ```

◦ For more information about how to install Alibaba Cloud CLI in Windows, see Windows.

2. Configure Alibaba Cloud CLI. Run the following command to create environment variables that are used to store identity information:

   ```
   aliyun configure
   ```

   Expected output:

   ```
   Configuring profile 'default' in 'AK' authenticate mode...
   Access Key Id []: ************
   Access Key Secret []: ************
   Default Region Id []: cn-beijing
   Default Output Format [json]: json (Only support json)
   Default Language [zh|en] en:
   Saving profile[default] ...Done.
   Configure Done!!!
   ..............888888888888888888888 ........=8888888888888888888D=.............
   ...........88888888888888888888888 ..........D8888888888888888888888I...........
   .........,88888888888888ZI:
   ..........................=Z88D8888888888D..................+88888888 ..............
   ..........................88888888D.........
   .........+88888888 .......Welcome to use Alibaba Cloud.......O8888888D.........
   .........+88888888 ............. ************* ..............O8888888D................
   ..+88888888 .... Command Line Interface(Reloaded) ....O8888888D.........
   .........+88888888...........................................88888888D................
   ...D888888888888DO+. ..........................?ND888888888888D.........
   ...........O8888888888888888888888..........D8888888888888888888888=...........
   ............ .:D888888888888888888.........78888888888888888888O .............
   ```

Install and set up a kubectl client. For more information, see Install and set up kubectl.

Create an ASK cluster

1. Create a working directory and a file named `create.json` in the directory.

   The following template is an example of the `create.json` file:

```
POST /clusters HTTP/1.1
<Common request parameters>
{
    "cluster_type":"Ask",
    "name":"test-ask",
    "region_id":"cn-hangzhou",
    "endpoint_public_access":false,
    "private_zone":false,
    "nat_gateway":true,
    "tags":[
        {
            "key":"k-aa",
            "value":"v-aa"
        }
    ],
    "deletion_protection":false,
    "addons":[
        {
            "name":"logtail-ds"
        }
    ],
    "zone_id":"cn-hangzhou-i"
}
```

The preceding sample template describes the configurations that are used to create a serverless Kubernetes (ASK) cluster. For more information, see Create an ASK cluster.

2. Run the following command to create an ASK cluster:

```
aliyun cs  POST /clusters --header "Content-Type=application/json" --body "$(cat create
.json)"
```

Expected output:

```
{
    "cluster_id": "*************************",
    "instanceId": "*************************",
    "request_id": "**********-****-****-****-************",
    "task_id": "*-*************"
}
```

3. After the ASK cluster is created, run the following command to query information about the cluster:

```
aliyun cs GET /clusters/<YOUR-CLUSTER-ID>
```

Expected output:

```
  {
      "-": "PayByTraffic",
      "cluster_healthy":
      "",
      "cluster_id": "*************************",
      "cluster_spec":
      "",
      "cluster_type": "Ask"
```

```
        "cluster_type": "Ask",
        "created":
        "2020-07-23T10:02:18+08:00",
        "current_version": "v1.16.6-aliyun.1",
        "data_disk_category":
        "cloud",
        "data_disk_size": 0,
        "deletion_protection": false,
        "docker_version":
        "",
        "enabled_migration": false,
        "external_loadbalancer_id": "lb-*********",
        "gw_bridge":
        "",
        "init_version": "v1.16.6-aliyun.1",
        "instance_type":
        "",
        "name": "test-serverless-k8s",
        "need_update_agent": false,
        "network_mode":
        "vpc",
        "node_status": "",
        "private_zone": false,
        "profile":
        "ask.v2",
        "region_id":
        "cn-beijing",
        "resource_group_id": "rg-*********",
        "security_group_id":
        "sg-*********",
        "size":
        0,
        "state": "running",
        "subnet_cidr":
        "172.16.0.0/16",
        "swarm_mode": false,
        "tags": [
            {
                "key":
        "env",
                "value": "test"
            }
        ],
        "updated":
        "2020-07-23T10:05:11+08:00",
        "vpc_id": "vpc-*********",
        "vswitch_cidr":
        "",
        "vswitch_id": "vsw-*********",
        "worker_ram_role_name":
        ""
    }
```

4. Run the following command to query the cluster configurations:

```
KUBECONFIG=<YOUR-LOCAL-KUBECONFIG-PATH>
aliyun cs GET /k8s/$cluster_id/user_config | jq -r '.config' > $KUBECONFIG
```

Expected output:

```
NAME              STATUS   AGE
default           Active   7m43s
kube-node-lease   Active   7m45s
kube-public       Active   7m45s
kube-system       Active   7m45s
```

Test the ASK cluster

1. Run the following command to deploy an NGINX application in the ASK cluster:

```
kubectl run nginx --image=registry-vpc.cn-shenzhen.aliyuncs.com/acs-sample/nginx:latest
```

Expected output:

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

2. Run the following command to query the state of the NGINX application:

```
kubectl get deploy nginx
```

Expected output:

```
 NAME    READY   UP-TO-DATE   AVAILABLE   AGE
 nginx   1/1     1            1           58s
```

Delete and release resources

- Run the following command to delete the NGINX application:

```
kubectl delete deploy nginx
```

Expected output:

```
deployment.extensions "nginx" deleted
```

- You can also run the following command to delete the ASK cluster and release the relevant resources, such as the virtual private cloud (VPC) where the cluster is deployed:

```
aliyun cs DELETE /clusters/************
```

Related information

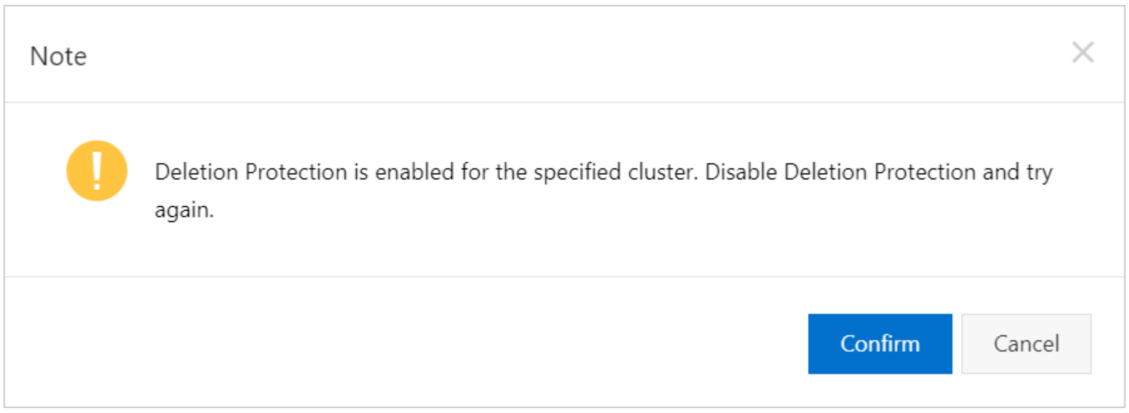
- Overview
- What is Alibaba Cloud CLI?
- What is Cloud Shell?

6.3. Delete an ASK cluster

This topic describes how to delete a serverless Kubernetes (ASK) cluster in the Container Service for Kubernetes (ACK) console.

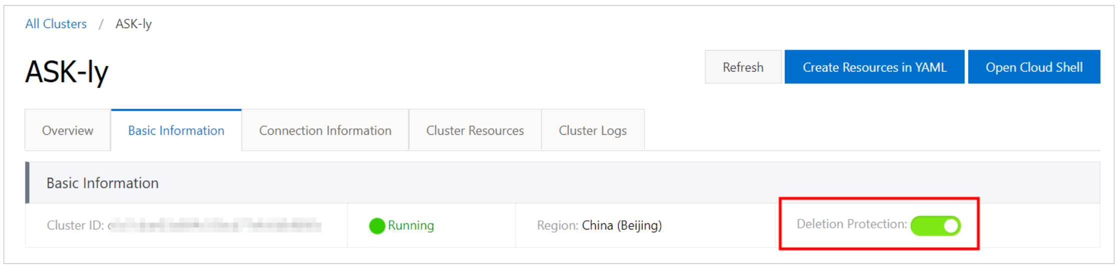
Procedure

1.
2.
3. On the **Clusters** page, find the cluster that you want to delete and choose **More > Delete** in the **Actions** column.
4. If the deletion protection feature is enabled for the cluster, click the name of the cluster or click **Details** in the **Actions** column.


Note

Deletion Protection is enabled for the specified cluster. Disable Deletion Protection and try again.

   i. On the Cluster Information page, click the **Basic Information** tab.


All Clusters / ASK-ly

ASK-ly

Basic Information

Cluster ID: [redacted] Running Region: China (Beijing) Deletion Protection:

   ii. Turn off **Deletion Protection**.
   iii. Return to the **Clusters** page.
   iv. On the **Clusters** page, find the cluster that you want to delete and choose **More > Delete** in the **Actions** column.
5. In the **Delete Cluster** dialog box, read and select **I understand the above information and want to delete the specified cluster**, and then click **OK**.

> Notice If you have created elastic container instances to run pods in the cluster, you must go to the **Deployments** page and delete all pods that run on the elastic container instances. Then, perform the preceding steps to delete the ASK cluster.

6.4. Manage and access clusters

6.4.1. Connect to an ACK cluster by using kubectl

To connect to a Container Service for Kubernetes (ACK) cluster from your on-premises machine, we recommend that you use the kubectl command-line tool. This topic describes how to use kubectl to connect to an ACK cluster.

Procedure

1. Download the latest kubectl client from the Kubernetes version description page.
2. Install and set up a kubectl client.

   For more information, see Install and set up kubectl.
3. Configure the cluster credentials.

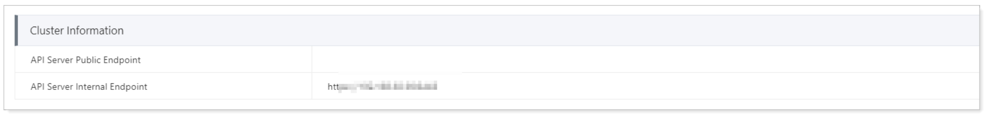
   You can view cluster credentials on the details page of a cluster.

   i.

   ii.

   iii.

   iv. (Optional)Click the **Basic Information** tab. In the **Cluster Information** section, you can view the public and internal IP addresses that are used to connect to the cluster.

   Cluster Information

API Server Public Endpoint	
API Server Internal Endpoint	htt[redacted]

   v. Click the **Connection Information** tab and copy the cluster credentials to a local file on your on-premises machine.

   You can create a *$HOME/.kube/config* file and copy the cluster credentials to this file. This is the default file from which kubectl obtains the cluster credentials. You can also create another file, for example, `/tmp/kubeconfig`, and copy the cluster credentials to the file. Then, run the *export KUBECONFIG=/tmp/kubeconfig* command.

   vi. Run the following command to check whether the kubectl client is connected to the cluster:

   ```
   kubectl get namespace
   ```

   Expected output:

   ```
   NAME              STATUS   AGE
   default           Active   3d6h
   kube-node-lease   Active   3d6h
   kube-public       Active   3d6h
   kube-system       Active   3d6h
   ```

What's next

After the preceding operations are complete, you can use kubectl to connect to the ACK cluster from your on-premises machine.

6.4.2. Use kubectl on Cloud Shell to manage ASK clusters

This topic describes how to use kubectl on Cloud Shell to manage clusters of Container Service for Kubernetes (ASK) in the ACK console.

Prerequisites

ASK quick start.

Context

To manage ASK clusters by using kubectl, you must install and set up the kubectl client. For more information, see Connect to ACK clusters by using kubectl. You can also use kubectl on Cloud Shell in the ACK console to manage ASK clusters.

Procedure

1.
2.
3. On the **Clusters** page, find the cluster that you want to manage and choose **More > Open Cloud Shell** in the **Actions** column.

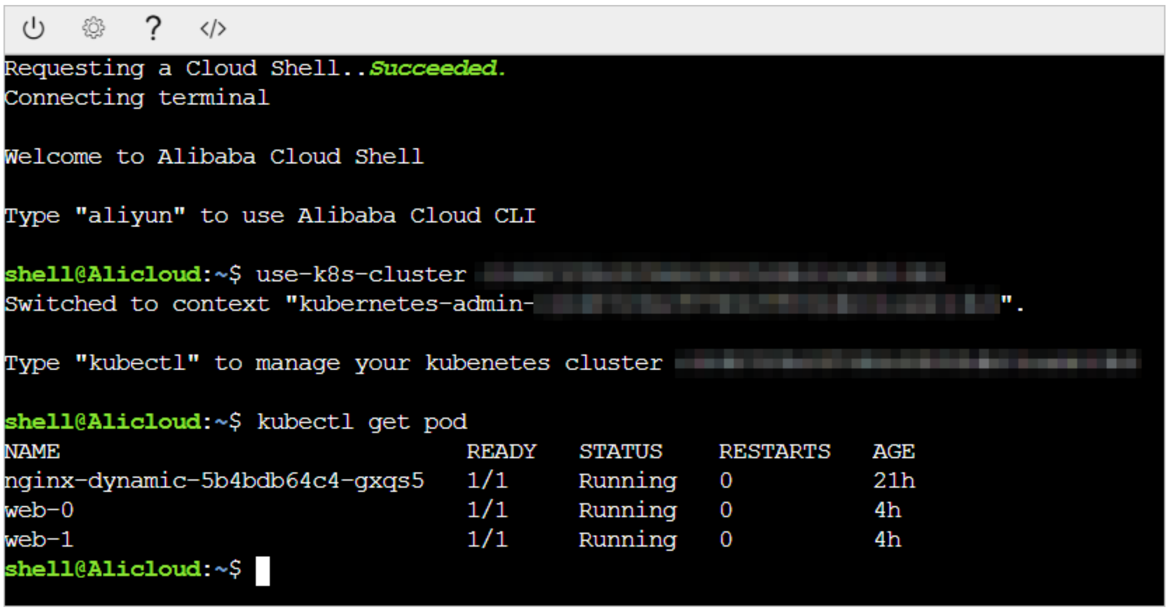
   Note

   After Cloud Shell is opened, perform the following steps:

   - On the **Authorization** page, click **OK**. You will obtain an AccessKey pair for a temporary session.
   - Click the icon and select **Mount File Storage**. You can **create and mount** a Network Attached Storage (NAS) file system to the cluster or **skip** the process.

4. Use kubectl on Cloud Shell in the ACK console to manage ASK clusters.

   Note When you open Cloud Shell, Cloud Shell automatically reads the *kubeconfig* file of the cluster. Then, you can use kubectl to manage the cluster.



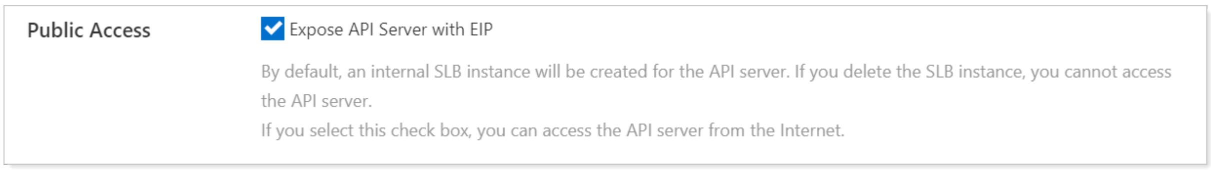
```
Requesting a Cloud Shell..Succeeded.
Connecting terminal

Welcome to Alibaba Cloud Shell

Type "aliyun" to use Alibaba Cloud CLI

shell@Alicloud:~$ use-k8s-cluster [redacted]
Switched to context "kubernetes-admin-[redacted]".

Type "kubectl" to manage your kubenetes cluster [redacted]

shell@Alicloud:~$ kubectl get pod
NAME                              READY   STATUS    RESTARTS   AGE
nginx-dynamic-5b4bdb64c4-gxqs5    1/1     Running   0          21h
web-0                             1/1     Running   0          4h
web-1                             1/1     Running   0          4h
shell@Alicloud:~$
```

6.4.3. Access the Kubernetes API server over the Internet

Serverless Kubernetes (ASK) allows you to use an elastic IP address (EIP) to expose the API server of a cluster. After you use an EIP to expose the API server of a cluster, you can access the API server of the cluster over the Internet. This topic describes how to associate an EIP with the API server when you create a cluster.

Associate an EIP with the API server when you create a cluster

When you create a cluster, select **Expose API Server with EIP** to enable public access to the API server of the cluster. For more information about how to create a cluster, see ASK quick start.



Public Access

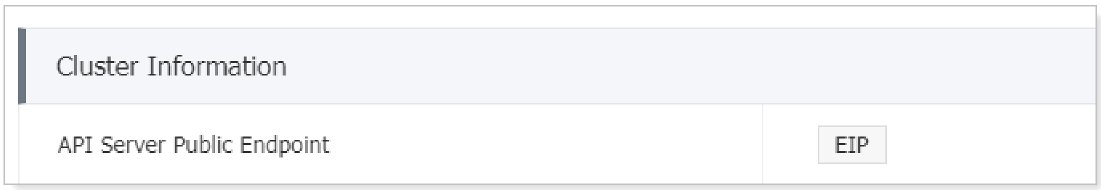
☑ Expose API Server with EIP

By default, an internal SLB instance will be created for the API server. If you delete the SLB instance, you cannot access the API server.
If you select this check box, you can access the API server from the Internet.

Associate an EIP with the Kubernetes API server of an existing cluster

If you do not select Expose API Server with EIP when you create a cluster, you can perform the following steps to associate an EIP with the API server of the cluster:

1.
2.
3.
4. In the left-side navigation pane, click **Cluster Information**.
5. On the **Cluster Information** page, click the **Basic Information** tab. In the **Cluster Information** section, click **EIP**.



Cluster Information

API Server Public Endpoint | EIP

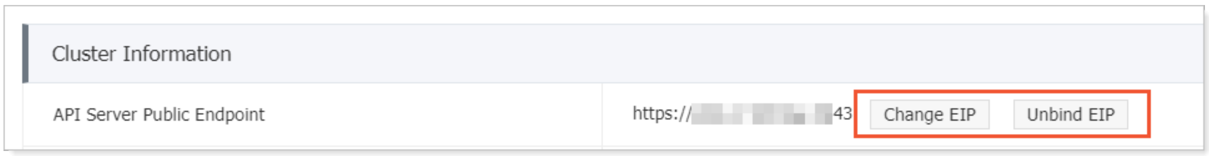
6. In the **EIP** dialog box, select an EIP and click **OK**.

   After the EIP is associated with the API server, a public IP address appears on the right side of the API Server Public Endpoint field.

   Notice The API server restarts after you associate an EIP with the API server. We recommend that you do not perform operations on the cluster during the restart process.

What's next

After an EIP is associated with the API server, you can change or disassociate the EIP. After you disassociate the EIP from the API server of a cluster, the API server cannot be accessed over the Internet.



Cluster Information

API Server Public Endpoint | https://[redacted]43 | Change EIP | Unbind EIP

6.5. Best practices

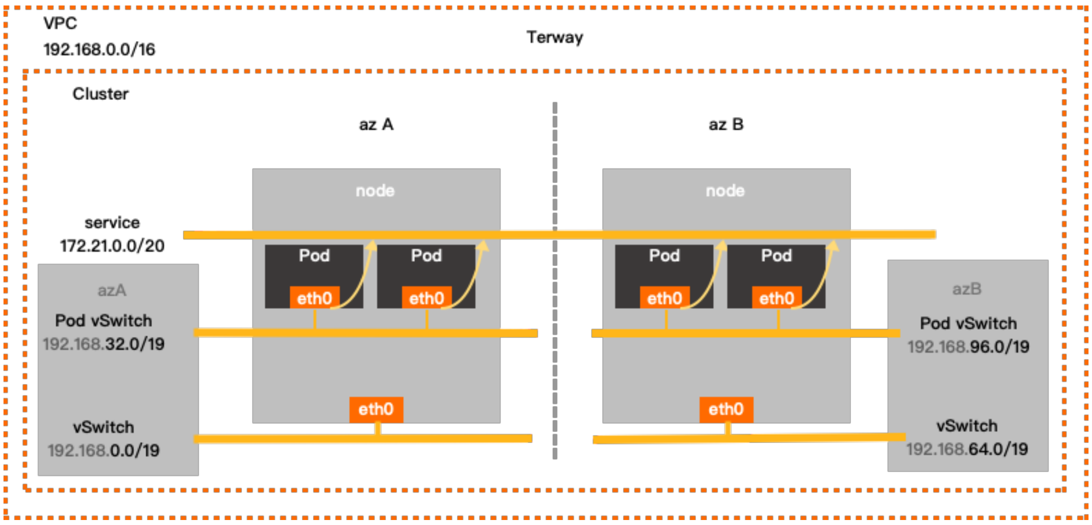
6.5.1. Plan CIDR blocks for an ACK cluster

When you create a Container Service for Kubernetes (ACK) cluster, you must specify a virtual private cloud (VPC), vSwitches, the CIDR block of pods, and the CIDR block of Services. Therefore, we recommend that you plan the IP address of each Elastic Compute Service (ECS) instance in the cluster, the CIDR block of pods, and the CIDR block of Services before you create an ACK cluster. This topic describes how to plan CIDR blocks for an ACK cluster deployed in a VPC and how each CIDR block is used.

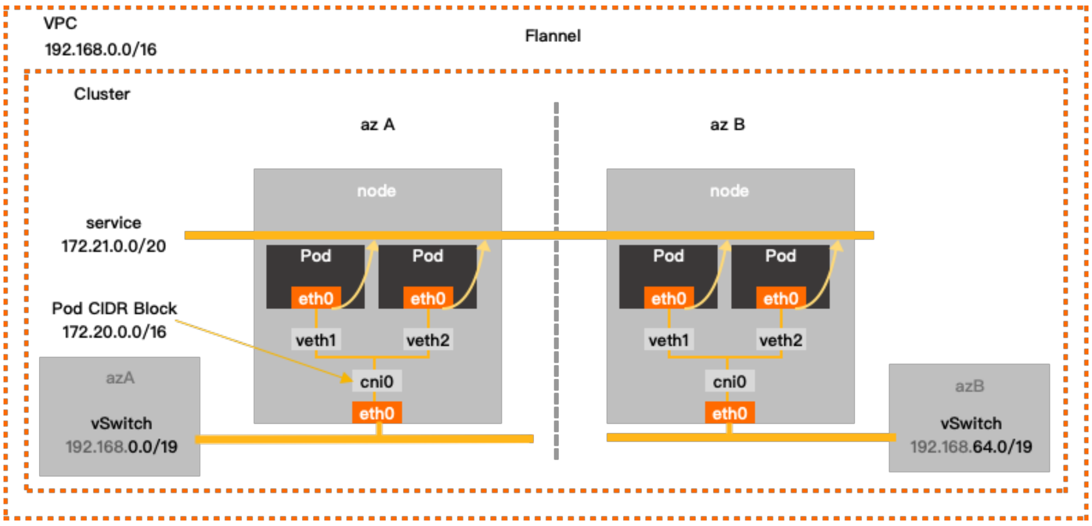
Network architectures of VPC-connected Kubernetes clusters

Before you create a VPC, you must plan the CIDR block of the VPC and the CIDR blocks of vSwitches in the VPC. Before you create an ACK cluster, you must plan the CIDR block of pods and the CIDR block of Services. ACK supports the Terway and Flannel plug-ins. The following figures show the network architectures of ACK clusters that use Terway and Flannel.

Terway



Flannel



To install Terway or Flannel in an ACK cluster, you must specify CIDR blocks and other parameters as described in the following table.

Parameter	Terway	Flannel
VPC	When you create a VPC, you must select a CIDR block for the VPC. Valid values: 10.0.0.0/8, 172.16.0.0/12, and 192.168.0.0/16. IPv6 CIDR blocks are assigned by the VPC after you enable IPv6 for the VPC. If you want to enable IPv6 for containers, select Terway as the network plug-in.	
vSwitch	The IP addresses of ECS instances are assigned from vSwitches. This allows nodes in a cluster to communicate with each other. The CIDR blocks that you specify when you create vSwitches in the VPC must be subsets of the VPC CIDR block. This means that the vSwitch CIDR blocks must fall within or be the same as the VPC CIDR block. When you set this parameter, take note of the following items: • Select one or more vSwitches in the VPC. • IP addresses from the CIDR block of a vSwitch are allocated to the ECS instances that are attached to the vSwitch. • You can create multiple vSwitches in a VPC. However, the CIDR blocks of these vSwitches cannot overlap with each other. • A node vSwitch and the vSwitch that assigns IP addresses to pods on the node must be in the same zone. For more information about zones, see Regions and zones.	The IP addresses of ECS instances are assigned from vSwitches. This allows nodes in a cluster to communicate with each other. The CIDR blocks that you specify when you create vSwitches in the VPC must be subsets of the VPC CIDR block. This means that the vSwitch CIDR blocks must fall within or be the same as the VPC CIDR block. When you set this parameter, take note of the following items: • Select one or more vSwitches in the VPC. • IP addresses from the CIDR block of a vSwitch are allocated to the ECS instances that are attached to the vSwitch. • You can create multiple vSwitches in a VPC. However, the CIDR blocks of these vSwitches cannot overlap with each other.

Parameter	Terway	Flannel
Pod vSwitch	The IP addresses of pods are assigned from the CIDR block of the pod vSwitches. This allows pods to communicate with each other. A pod is a group of containers in a Kubernetes cluster. Each pod has an IP address. The CIDR blocks that you specify when you create pod vSwitches in the VPC must be subsets of the VPC CIDR block. When you set this parameter, take note of the following items: • Select one or more vSwitches in the VPC. • In an ACK cluster that has Terway installed, the IP addresses of pods are allocated from pod vSwitches. • The CIDR blocks of pod vSwitches cannot overlap with the CIDR block specified by **Service CIDR**. • A node vSwitch and the vSwitch that assigns IP addresses to pods on the node must be in the same zone. For more information about zones, see Regions and zones.	You do not need to set this parameter if you install Flannel in an ACK cluster.
Pod CIDR Block	You do not need to set this parameter if you install Terway in an ACK cluster.	The IP addresses of pods are allocated from the pod CIDR block. This allows pods to communicate with each other. A pod is a group of containers in a Kubernetes cluster. Each pod has an IP address. When you set this parameter, take note of the following items: • Enter a CIDR block in the Pod CIDR Block field. • The CIDR block of pods cannot overlap with the CIDR blocks of **vSwitches**. • The CIDR block of pods cannot overlap with the CIDR block specified by **Service CIDR**. For example, if the VPC CIDR block is 172.16.0.0/12, the CIDR block of pods cannot be 172.16.0.0/16 or 172.17.0.0/16, because these CIDR blocks are subsets of 172.16.0.0/12.

Parameter	Terway	Flannel
Service CIDR	The CIDR block of Services. Service is an abstraction in Kubernetes. Each `ClusterIP` Service has an IP address. When you set this parameter, take note of the following items: • The IP address of a Service is effective only within the Kubernetes cluster. • The CIDR block of Services cannot overlap with the CIDR blocks of **vSwitches**. • The CIDR block of Services cannot overlap with the CIDR blocks of **Pod vSwitches**.	The CIDR block of Services. Service is an abstraction in Kubernetes. Each `ClusterIP` Service has an IP address. When you set this parameter, take note of the following items: • The IP address of a Service is effective only within the Kubernetes cluster. • The CIDR block of Services cannot overlap with the CIDR blocks of **vSwitches**. • The CIDR block of Services cannot overlap with **Pod CIDR Block**.
Service IPv6 CIDR	If you enable IPv4/IPv6 dual-stack, you must specify an IPv6 CIDR block for Services. When you set this parameter, take note of the following items: • You must specify a Unique Local Unicast Address (ULA) space that falls within the address range fc00::/7. The prefix must be 112 bits to 120 bits in length. • We recommend that you specify an IPv6 CIDR block that has the same number of IP addresses as the Service CIDR block.	You do not need to set this parameter if you install Flannel in an ACK cluster.

Precautions

Network Planning

To use Kubernetes clusters that are supported by ACK on Alibaba Cloud, you must first set up networks for the clusters based on the cluster sizes and business scenarios. You can refer to the following tables to set up networks for Kubernetes clusters. Change specifications as needed in unspecified scenarios.

Cluster size	Scenario	VPC	Zone
< 100 nodes	Regular business.	Single VPC	1
Unlimited	Cross-zone deployment is required.	Single VPC	≥ 2
Unlimited	High reliability and cross-region deployment are required.	Multiple VPCs	≥ 2

The following tables describe how to plan CIDR blocks for clusters that use Flannel or Terway.

- **Clusters that use Flannel**

VPC CIDR block	vSwitch CIDR block	Pod CIDR block	Service CIDR block	Maximum number of pod IP addresses
192.168.0.0/16	192.168.0.0/24	172.20.0.0/16	172.21.0.0/20	65536

- **Clusters that use Terway**
    - Exclusive elastic network interface (ENI) mode or IPVLAN mode is enabled

VPC CIDR block	vSwitch CIDR block	CIDR block of pod vSwitches	Service CIDR block	Maximum number of pod IP addresses
192.168.0.0/16	192.168.0.0/19	192.168.32.0/19	172.21.0.0/20	8192

    - **Multi-zone deployment**

VPC CIDR block	vSwitch CIDR block	CIDR block of pod vSwitches	Service CIDR block	Maximum number of pod IP addresses
192.168.0.0/16	Zone I 192.168.0.0/19	192.168.32.0/19	172.21.0.0/20	8192
	Zone J 192.168.64.0/19	192.168.96.0/19		8192

Plan the network of a VPC

Plan CIDR blocks for clusters

How to plan CIDR blocks

- **Scenario 1: One VPC and one Kubernetes cluster**

  This is the simplest scenario. The CIDR block of a VPC is specified when you create the VPC. When you create a cluster in the VPC, make sure that the CIDR block of pods and the CIDR block of Services do not overlap with the VPC CIDR block.

- **Scenario 2: One VPC and multiple Kubernetes clusters**

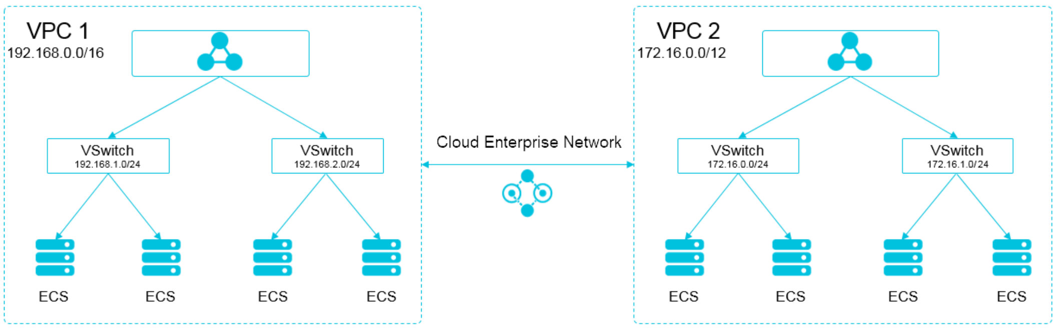
  You want to create more than one cluster in a VPC.

    - The CIDR block of the VPC is specified when you create the VPC. When you create clusters in the VPC, make sure that the VPC CIDR block, Service CIDR block, and pod CIDR block of each cluster do not overlap with one another.
    - The Service CIDR blocks of the clusters can overlap with each other. However, the pod CIDR blocks cannot overlap with each other.
    - In the default network mode (Flannel), the packets of pods must be forwarded by the VPC router. ACK automatically generates a route table for each destination pod CIDR block on the VPC router.

> **Note** In this case, a pod in one cluster can communicate with the pods and ECS instances in another cluster. However, the pod cannot communicate with the Services in another cluster.

- **Scenario 3: Two connected VPCs**

  If two VPCs are connected, you can use the route table of one VPC to specify the packets that you want to send to the other VPC. The CIDR block of VPC 1 is 192.168.0.0/16 and the CIDR block of VPC 2 is 172.16.0.0/12, as shown in the following figure. You can use the route table of VPC 1 to forward all packets that are destined for 172.16.0.0/12 to VPC 2.

  Connected VPCs

VPC	CIDR block	Destination CIDR block	Destination VPC
VPC 1	192.168.0.0/16	172.16.0.0/12	VPC 2
VPC 2	172.16.0.0/12	192.168.0.0/16	VPC 1

  In this scenario, make sure that the following conditions are met when you create a cluster in VPC 1 or VPC 2:

  - The CIDR blocks of the cluster do not overlap with the CIDR block of VPC 1.
  - The CIDR blocks of the cluster do not overlap with the CIDR block of VPC 2.
  - The CIDR blocks of the cluster do not overlap with those of other clusters.
  - The CIDR blocks of the cluster do not overlap with those of pods.
  - The CIDR blocks of the cluster do not overlap with those of Services.

  In this example, you can set the pod CIDR block of the cluster to a subset of 10.0.0.0/8.

  > **Note** All IP addresses in the destination CIDR block of VPC 2 can be considered in use. Therefore, the CIDR blocks of the cluster cannot overlap with the destination CIDR block.

  To access pods in VPC 1 from VPC 2, you must configure a route in VPC 2. The route must point to the pod CIDR block of a cluster in VPC 1.

- **Scenario 4: A VPC connected to a data center**

  If a VPC is connected to a data center, packets of specific CIDR blocks are routed to the data center. In this case, the pod CIDR block of a cluster in the VPC cannot overlap with these CIDR blocks. To access pods in the VPC from the data center, you must configure a route in the data center to enable VBR-to-VPC peering connection.

Related information

- Overview
- Work wit h Terway
- Use net work policies
- FAQ about net work management

7.ASK Pro Cluster

7.1. ASK Pro cluster overview

Professional serverless Kubernetes (ASK Pro) clusters are developed based on ASK standard clusters and provide higher reliability and security in large-scale production environments for enterprise users. ASK Pro clusters are also covered by service level agreements (SLAs) that include compensation clauses.

Overview

ASK Pro clusters are developed based on ASK standard clusters. This type of cluster inherits all benefits of ASK standard clusters, such as compatibility with the open source version, automated O&M, and scaling within seconds. In addition, ASK Pro clusters provide higher reliability and security, and are covered by SLA terms for compensation. ASK Pro clusters are suitable for enterprises that require higher security and stability for large-scale workloads in production environments.

Scenarios

- Internet enterprises. These enterprises deploy their business on a large scale and require business management with high stability, security, and observability.
- Big data computing enterprises. These enterprises deploy large-scale data computing services, high-performance data processing services, and other services with high elasticity. These services require clusters with high stability, high performance, and efficient computing capabilities.
- International enterprises that run their business in China. These enterprises prioritize security and services that provide SLAs with compensation clauses.
- Financial enterprises. These enterprises require SLAs with compensation clauses.

Features

- Managed master nodes with high reliability: The management of large-scale clusters is supported. etcd is a reliable store for disaster recovery and data restoration. etcd uses cold backups and hot backups to ensure data availability for ASK Pro clusters. Key metrics are collected for you to gain insights from control components. This allows you to detect potential risks.
- Clusters with higher security: By default, etcd uses encrypted disks in the control plane. In the data plane, kms-plugin is installed to encrypt Kubernetes Secrets. Container Service for Kubernetes (ACK) provides security management for this type of cluster. The advanced security management feature allows you to inspect containers in the running state and enable auto repairing.
- SLA guarantees: ASK Pro clusters are covered by the SLA that supports compensation clauses. A level of 99.95% uptime is guaranteed for the cluster API server.

Pricing

Billing of professional ASK clusters

Comparison

The following table compares ASK Pro clusters with ASK standard clusters.

Category	Feature	ASK	
		ASK Pro cluster	ASK standard cluster
Cluster size	N/A	Up to 10,000 elastic container instances	Up to 1,000 elastic container instances
SLA	N/A	99.95% (supports compensation)	99.90% (does not support compensation)
API Server	Custom parameter settings	✓	✗
	Availability monitoring	✓	✗
ETCD	High-frequency cold backups, high-frequency hot backups, and geo-disaster recovery	✓	✗
	Observability metrics	✓	✗
Security management	The advanced security management feature that supports data encryption. For more information, see Use KMS to encrypt Kubernetes Secrets.	✓	✗

7.2. Use KMS to encrypt Kubernetes Secrets

In professional serverless Kubernetes (ASK) clusters, you can use keys that are created in Key Management Service (KMS) to encrypt Kubernetes Secrets. This topic describes how to use a key that is managed by KMS to encrypt Secrets for a professional ASK cluster.

Prerequisites

Context

Note Secret encryption is supported only by existing professional ASK clusters. This feature cannot be enabled in newly created professional ASK clusters.

Enable Secret encryption for an existing professional ASK cluster

1.
2.
3. On the **Clusters** page, click the name of the professional ASK cluster for which you want to enable Secret encryption.
4. On the details page of the cluster, click the **Basic Information** tab. In the **Basic Information** section, turn on **Secret Encryption**.

   Note If you log on to the console as a Resource Access Management (RAM) user, make sure that the RAM user is assigned the administrator role or O&M engineer role to manage the cluster. For more information, see Assign RBAC roles to RAM users or RAM roles.

   If the status of the cluster changes from **Updating** to **Running**, the Secret encryption feature is enabled for the cluster.

Disable Secret encryption for an existing professional ASK cluster

1.
2.
3. On the **Clusters** page, click the name of the professional ASK cluster for which you want to disable Secret encryption.
4. On the details page of the cluster, click the **Basic Information** tab. In the **Basic Information** section, turn off **Secret Encryption**.

   Note If you log on to the console as a RAM user, make sure that the RAM user is assigned the administrator role or O&M engineer role to manage the cluster. For more information, see Assign RBAC roles to RAM users or RAM roles.

   If the status of the cluster changes from **Updating** to **Running**, the Secret encryption feature is disabled for the cluster.

Use automatic key rotation to encrypt Secrets

You can use the automatic key rotation feature provided by KMS to encrypt Secrets. During a key rotation, the system still uses the original key version to encrypt existing Secrets. For more information, see Automatic key rotation.

To force the system to use the new key version to encrypt existing Secrets, run the following command after the key is rotated:

```
kubectl get secrets --all-namespaces -o json | kubectl annotate --overwrite -f - encryption
-key-rotation-time="TIME"
```

Note Replace `TIME` with the string that indicates when the key rotation was performed, for example, 20220101-010101.

8.Images

8.1. Pull images from a Container Registry Enterprise Edition instance without a password

You can pull images from a Container Registry Enterprise Edition instance without a password. This accelerates image pulling. This topic describes how to pull images from a Container Registry Enterprise Edition instance without a password.

Prerequisites

Make sure that the following operations are completed:

- Container Registry is activated. A Resource Access Management (RAM) role is authorized to access Container Registry.
- A Container Registry Enterprise Edition instance is created and an image repository is configured. For more information, see Use a Container Registry Enterprise Edition instance to push and pull images.

Background information

Container Registry provides Container Registry Enterprise Edition instances and default instances. Container Registry Enterprise Edition is an enterprise-grade platform used to manage the lifecycle of cloud-native application artifacts. These artifacts include container images, Helm charts, and Open Container Initiative (OCI) artifacts. Container Registry Enterprise Edition seamlessly integrates with Container Service for Kubernetes (ACK) and helps simplify application delivery for enterprises in large-scale business deployment scenarios. For more information, see What is Container Registry?

You can pull an image from an image repository of a Container Registry instance by using one of the following methods:

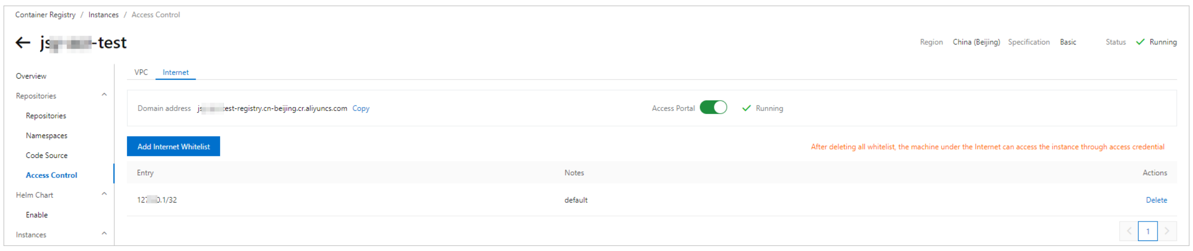
- If a Container Registry image belongs to the same account as the elastic container instance, you can pull the image without a password.
- If an image is a Docker image instead of a Container Registry image, you cannot pull the image without a password. When you call the API operation to create an elastic container instance, you can use the ImageRegistryCredential parameter to specify a password.

Pull images from a Container Registry Enterprise Edition instance without a password

In the Container Registry console, find the instance that you want to manage and configure network access control based on the following information:

- Over the Internet

  After you enable Internet access, you can access images in the Container Registry Enterprise Edition instance across regions by using public endpoints. For more information, see Configure access over the Internet.



js[redacted]-test

Region China (Beijing) Specification Basic Status ✓ Running

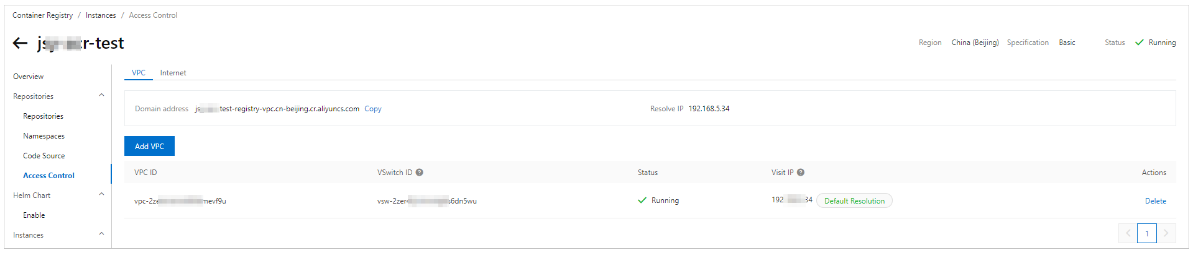
Domain address js[redacted]test-registry.cn-beijing.cr.aliyuncs.com Access Portal ✓ Running

After deleting all whitelist, the machine under the Internet can access the instance through access credential

Entry	Notes	Actions
12[redacted]0.1/32	default	Delete

- Over virtual private clouds (VPCs)

  If you want to access a Container Registry Enterprise Edition instance over VPCs, you must connect the Container Registry Enterprise Edition instance to the VPCs. For more information, see Configure access over VPCs.



js[redacted]r-test

Region China (Beijing) Specification Basic Status ✓ Running

Domain address js[redacted]test-registry-vpc.cn-beijing.cr.aliyuncs.com Resolve IP 192.168.5.34

VPC ID	VSwitch ID	Status	Visit IP	Actions
vpc-2ze[redacted]mevf9u	vsw-2zer[redacted]s6dn5wu	✓ Running	192[redacted]34 Default Resolution	Delete

After you configure the Container Registry Enterprise Edition instance, you can record the instance information such as the instance ID, instance name, and endpoint.

Use Kubernetes to pull images from a Container Registry Enterprise Edition instance without a password

You can add annotations to specify the Container Registry Enterprise Edition instance from which you want to pull images.

Note

You can specify only one Container Registry Enterprise Edition instance when you use Kubernetes. If you have multiple Container Registry Enterprise Edition instances that contain different images, we recommend that you push the images to one Container Registry Enterprise Edition instance. If you want to configure multiple Container Registry Enterprise Edition instances, we recommend that you call the API operation.

Example:

1. Prepare the YAML file.

   The following YAML file named test_cri.yaml is used as an example:

```
apiVersion: v1
kind: Pod
metadata:
  annotations:
    k8s.aliyun.com/acr-instance-id: "cri-j36zhodptmyq****"      # Specify the ID of the
Container Registry Enterprise Edition instance.
  name: cri-test
spec:
  containers:
  - image: test****-registry.cn-beijing.cr.aliyuncs.com/eci_test/nginx:1.0   # Pull an
image over the Internet.
    imagePullPolicy: Always
    name: nginx
  restartPolicy: Never
```

Note

You can pull images from a Container Registry Enterprise Edition instance that resides in a region different from the region of the elastic container instance-based pod. To do this, you must add the region ID of the Container Registry Enterprise Edition instance before the ID of the Container Registry Enterprise Edition instance. Example: `k8s.aliyun.com/acr-instance-id: "cn-beijing:cri-j36zhodptmyq****"` .

2. Create a pod.

```
kubectl apply -f test_cri.yaml
```

Call the API operation to pull images from a Container Registry Enterprise Edition instance without a password

When you call the CreateContainerGroup API operation to create an elastic container instance, you can use the AcrRegistryInfo parameters to pull images from a Container Registry Enterprise Edition instance without a password. The following table describes the parameters. For more information, see CreateContainerGroup.

Note

When you use the AcrRegistryInfo parameters to pull images from a Container Registry Enterprise Edition instance without a password, you must specify the AcrRegistryInfo.N.InstanceId parameter.

Parameter	Type	Example	Description
AcrRegistryInfo.N.RegionId	String	cn-beijing	The region ID of the Container Registry Enterprise Edition instance.
AcrRegistryInfo.N.InstanceId	String	cri-nwj395hgf6f3****	The ID of the Container Registry Enterprise Edition instance.

Parameter	Type	Example	Description
AcrRegistryInfo.N.Domain.N	RepeatList	test****-registry.cn-beijing.cr.aliyuncs.com	The endpoints of the Container Registry Enterprise Edition instance. By default, all endpoints of the Container Registry Enterprise Edition instance are displayed. You can specify one or more endpoints. Separate multiple endpoints with commas (,).
AcrRegistryInfo.N.InstanceName	String	test****	The name of the Container Registry Enterprise Edition instance.

The following examples demonstrate how to specify the AcrRegistryInfo parameters:

- Example 1: Specify the region ID, ID, name, and endpoints of the Container Registry Enterprise Edition instance.

```
'Container.1.Image': 'test****-registry.cn-beijing.cr.aliyuncs.com/eci_test/nginx:1.0',
'Container.1.Name': 'c1',
'Container.2.Image': 'test****-registry-vpc.cn-beijing.cr.aliyuncs.com/eci_test/nginx:1.0
',
'Container.2.Name': 'c2',

#AcrRegistryInfo
'AcrRegistryInfo.1.RegionId':'cn-beijing',
'AcrRegistryInfo.1.InstanceId': 'cri-nwj395hg********',
'AcrRegistryInfo.1.Domain.1': 'test****-registry-vpc.cn-beijing.cr.aliyuncs.com',
'AcrRegistryInfo.1.Domain.2': 'test****-registry.cn-beijing.cr.aliyuncs.com'
```

- Example 2: Specify the ID and name of the Container Registry Enterprise Edition instance.

```
'Container.1.Image': 'test****-registry.cn-beijing.cr.aliyuncs.com/eci_test/nginx:1.0',
'Container.1.Name': 'c1',
'Container.2.Image': 'test****-registry-vpc.cn-beijing.cr.aliyuncs.com/eci_test/nginx:1.0
',
'Container.2.Name': 'c2',

#AcrRegistryInfo
'AcrRegistryInfo.1.InstanceId': 'cri-nwj395hg********',
'AcrRegistryInfo.1.InstanceName': 'test****'
```

- Example 3: Specify only the ID of the Container Registry Enterprise Edition instance.

```
'Container.1.Image': 'test****-registry.cn-beijing.cr.aliyuncs.com/eci_test/nginx:1.0',
'Container.1.Name': 'c1',
'Container.2.Image': 'test****-registry-vpc.cn-beijing.cr.aliyuncs.com/eci_test/nginx:1.0
',
'Container.2.Name': 'c2',

#AcrRegistryInfo
'AcrRegistryInfo.1.InstanceId': 'cri-nwj395hg********'
```

You can also use SDKs to specify the AcrRegistryInfo parameters. The following sample code provides an example on how to use the SDK for Python to specify the AcrRegistryInfo parameters.

```
#!/usr/bin/env python
#coding=utf-8

from aliyunsdkcore.client import AcsClient
from aliyunsdkcore.acs_exception.exceptions import ClientException
from aliyunsdkcore.acs_exception.exceptions import ServerException
from aliyunsdkeci.request.v20180808.CreateContainerGroupRequest import CreateContainerGroup
Request

client = AcsClient('<accessKeyId>', '<accessSecret>', 'cn-beijing')

request = CreateContainerGroupRequest()
request.set_accept_format('json')

request.set_SecurityGroupId("sg-2zeh4cev9y7ulbr*****")
request.set_VSwitchId("vsw-2zejlv7xjnw61w6z*****")
request.set_ContainerGroupName("test-cri")
request.set_Containers([
  {
    "Image": "test****-registry.cn-beijing.cr.aliyuncs.com/eci_test/nginx:1.0",
    "Name": "nginx"
  },
  {
    "Image": "test****-registry-vpc.cn-beijing.cr.aliyuncs.com/eci_test/nginx:1.0",
    "Name": "nginx2"
  }
])
request.set_AcrRegistryInfos([
  {
    "RegionId": "cn-beijing",
    "InstanceId": "cri-nwj395hgf6f*****",
    "Domains": [
      "test****-registry-vpc.cn-beijing.cr.aliyuncs.com",
      "test****-registry.cn-beijing.cr.aliyuncs.com"
    ]
  }
])

response = client.do_action_with_exception(request)
# python2:  print(response)
print(str(response, encoding='utf-8'))
```

8.2. Use image caches to accelerate the creation of pods

Alibaba Cloud provides the image caching feature of Elastic Container Instance by using the ImageCache CustomResourceDefinition (CRD). You can use the image caching feature of Elastic Container Instance to accelerate the creation of pods in Kubernetes. This topic describes how to use image caches to accelerate the creation of pods.

Configuration description

An image cache is a cluster-level resource that you can use to accelerate the creation of pods in different namespaces.

You can enable automatic matching of an image cache or specify an image cache to accelerate the creation of pods by using annotations. You can add the following annotations to the metadata field in the pod configuration file:

- k8s.aliyun.com/eci-image-cache: Automatically matches the most suitable image cache and uses it to accelerate the creation of the pod. If no image cache is matched, the system automatically creates an image cache when the system creates the pod.
- k8s.aliyun.com/eci-image-snapshot-id: Specify an image cache to accelerate the creation of the pod.

Note

If both `k8s.aliyun.com/eci-image-cache` and `k8s.aliyun.com/eci-image-snapshot-id` are added, the k8s.aliyun.com/eci-image-snapshot-id annotation prevails. For more information, see Overview of the image caching feature.

When you create a pod by using an image cache, take note of the following items:

- We recommend that you specify an image in an image cache for the container to improve the matching degree.
- Set ImagePullPolicy of the container to IfNotPresent to prevent the system from repeatedly downloading image layers.

Automatic matching of an image cache

When you create a pod, you can add an annotation to enable automatic matching of an image cache to accelerate the creation of the pod. Elastic Container Instance selects the most suitable image cache from existing image caches based on the matching policies. The system selects the image cache based on the following order: the matching degree of the image, the size of the image, and the point in time when the image is created.

Note

If no image cache is matched, the system automatically creates an image cache when the system creates the pod.

- Deployment example

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
      annotations:
        k8s.aliyun.com/eci-image-cache: "true"   # Enables automatic matching of an image
cache.
    spec:
      nodeName: virtual-kubelet
      containers:
      - name: nginx
        image: nginx:1.7.9
        imagePullPolicy: IfNotPresent
```

- Pod example

```
apiVersion: v1
kind: Pod
metadata:
  annotations:
    k8s.aliyun.com/eci-image-cache: "true"   # Enables automatic matching of an image cac
he.
  name: nginx-auto-match
spec:
  containers:
  - image: nginx:1.7.9
    imagePullPolicy: IfNotPresent
    name: nginx
    resources:
      limits:
        cpu: 300m
        memory: 200Mi
      requests:
        cpu: 200m
        memory: 100Mi
  nodeName: virtual-kubelet
```

Specify an image cache

When you create a pod, you can add an annotation to specify an image cache that you want to use to accelerate the creation of the pod.

Notice

Make sure that the specified image cache is in the Ready state. Otherwise, the pod fails to be created.

- Deployment example

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
      annotations:
        k8s.aliyun.com/eci-image-snapshot-id: imc-2ze5tm5gehgtiiga****  # Specifies an im
age cache.
    spec:
      nodeName: virtual-kubelet
      containers:
      - name: nginx
        image: nginx:1.7.9
        imagePullPolicy: IfNotPresent
```

- Pod example

```
apiVersion: v1
kind: Pod
metadata:
  annotations:
    k8s.aliyun.com/eci-image-snapshot-id: imc-2ze5tm5gehgtiiga****  # Specifies an image
cache.
  name: nginx-imagecache-id
spec:
  containers:
  - image: nginx:1.7.9
    imagePullPolicy: IfNotPresent
    name: nginx
    resources:
      limits:
        cpu: 300m
        memory: 200Mi
      requests:
        cpu: 200m
        memory: 100Mi
  nodeName: virtual-kubelet
```

9.Application

9.1. Manage applications by using the CLI

This topic describes how to use the command-line interface (CLI) to create applications or view containers in which applications run.

Prerequisites

The kubectl client is connected to the serverless Kubernetes (ASK) cluster that you want to manage. For more information, see Connect to an ACK cluster by using kubectl.

Create an application by using the CLI

Run the following command to start a container. An NGINX web server is used in this example.

```
kubectl run nginx --image=registry.cn-hangzhou.aliyuncs.com/spacexnice/netdia:latest
```

Run the following command to create a service entry for the container. The `--type=LoadBalancer` setting in the command indicates that a Server Load Balancer (SLB) instance is created for the NGINX container.

```
kubectl expose deployment nginx --port=80 --target-port=80 --type=LoadBalancer
```

View containers by using the CLI

Run the following command to list all running containers in the default namespace:

```
kubectl get pods
```

```
NAME                                READY     STATUS    RESTARTS   AGE
nginx-2721357637-d****              1/1       Running   1          9h
```

9.2. Create an application from an image

This topic describes how to create an application from an image in the Container Service for Kubernetes (ACK) console.

Prerequisites

A serverless Kubernetes (ASK) cluster is created. For more information, see Create an ASK cluster.

Step 1: Configure basic settings

1.

2.
3.
4.
5.
6. On the **Basic Information** wizard page, configure the basic settings.

Parameter	Description
Name	The name of the application.
Replicas	Specify the number of pods that are provisioned for the application.
Type	The type of application, for example, **Deployment** or **StatefulSet**.
Label	Add a label to the application. A label is used to identify the application.
Annotations	Add an annotation to the application.

7. Click **Next**. Proceed to the **Container** wizard page.

Step 2: Configure the containers

On the **Container** wizard page, set the configurations of the containers for the application. The configurations include the container image, computing resources, container ports, environment variables, health checks, lifecycle, and volumes.

Note At the top of the **Container** wizard page, you can click **Add Container** to add containers to the pod.

1. In the **General** section, configure the basic settings of the container.

Parameter	Description
Image Name	You can click **Select Image**. In the dialog box that appears, select an image and click **OK**. You can also enter the address of a private registry. The registry address must be in the following format: `domainname/namespace/imagename:tag`.
	Click **Select Image Version** and select an image version. If you do not specify an image version, the latest image version is used.

Parameter	Description
Image Version	**Always Pull Images**: If you do not select this check box, ACK caches the pulled image. This improves the efficiency of application deployments. If the specified image version is the same as the cached image version, ACK creates the application from the cached image. Therefore, when you update the application code, if you do not change the image version for reasons such as to support the upper-layer workloads, the previously cached image is used. If you select this check box, ACK pulls the image from the repository each time the application is deployed. This ensures that the latest image and code are used.
	Set Image Pull Secret: Click **Set Image Pull Secret** to configure a Secret that is used to pull images. You must configure a Secret if you want to pull images from a private repository. For more information, see Create an application by using an image pull Secret.
Required Resources	The amount of CPU and memory resources that are reserved for this application. These resources are exclusive to the container. This prevents the application from becoming unavailable when other services or processes compete for computing resources.
Container Start Parameter	◦ stdin: Pass stdin to the container. ◦ tty: Stdin is a TeleTYpewriter (TTY).
Privileged Container	◦ If you select Privileged Container, privileged=true is set for the container and the privilege mode is enabled. ◦ If you do not select Privileged Container, privileged=false is set for the container and the privilege mode is disabled.
Init Container	If you select Init Container, an init container is created. An init container provides tools for pod management. For more information, see init containers.

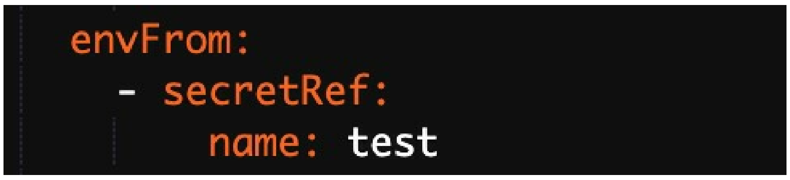
2. In the **Ports** section, click **Add** to configure one or more container ports.
   - Name: Enter a name for the container port.
   - Container Port: Enter the container port that you want to open. Enter a port number from 1 to 65535.
   - Protocol: Select TCP or UDP.
3. In the **Environments** section, click **Add** to configure one or more environment variables.

   You can configure environment variables for pods in key-value pairs. Environment variables are used to pass pod configurations to containers. For more information, see Pod variables.

   - Type: Select the type of environment variable, for example, **Custom** or **ConfigMaps**. If you select ConfigMaps or Secrets, you can pass all data in the selected ConfigMap or Secret to the container environment variables. In this example, a Secret is selected.

     Select **Secrets** from the Type drop-down list and select a Secret from the **Value/ValueFrom** drop-down list. By default, all data in the selected Secret is passed to the environment variable.

In this case, the YAML file that is used to deploy the application contains the settings that reference all data in the specified Secret.



```
envFrom:
  - secretRef:
      name: test
```

  - Variable Key: Specify the name of the environment variable.
  - Value/ValueFrom: Specify a reference that is used to define the environment variable.

4. In the **Health Check** section, you can enable **liveness** and **readiness** probes as needed.

   For more information about health checks, see Configure Liveness, Readiness and Startup Probes.

   - Liveness probes are used to detect when to restart the container.
   - Readiness probes are used to determine whether the container is ready to accept traffic.

Request type	Parameter and description
HTTP	Sends an HTTP GET request to the container. You can set the following parameters: ◦ Protocol: Select HTTP or HTTPS. ◦ Path: the requested path on the server. ◦ Port: the container port that you want to open. Enter a port number from 1 to 65535. ◦ HTTP Header: the custom headers in the HTTP request. Duplicate headers are allowed. Key-value pairs are supported. ◦ Initial Delay (s): the initialDelaySeconds field in the YAML file. This field specifies the time period (in seconds) that the system must wait before it can send a probe to the container after the container is started. Default value: 3. ◦ Period (s): the periodSeconds field in the YAML file. This field specifies the interval (in seconds) at which probes are performed. Default value: 10. Minimum value: 1. ◦ Timeout (s): the timeoutSeconds field in the YAML file. This field specifies the time (in seconds) after which a probe times out. Default value: 1. Minimum value: 1. ◦ Healthy Threshold: the minimum number of times that an unhealthy container must consecutively pass health checks before it is considered healthy. Default value: 1. Minimum value: 1. For liveness probes, this parameter must be set to 1. ◦ Unhealthy Threshold: the minimum number of times that a healthy container must consecutively fail health checks before it is considered unhealthy. Default value: 3. Minimum value: 1.

Request type	Parameter and description
TCP	Sends a TCP socket to the container. kubelet attempts to open the socket on the specified port. If the connection can be established, the container is considered healthy. Otherwise, the container is considered unhealthy. You can set the following parameters: ◦ Port: the container port that you want to open. Enter a port number from 1 to 65535. ◦ Initial Delay (s): the initialDelaySeconds field in the YAML file. This field specifies the time period (in seconds) that the system must wait before it can send a probe to the container after the container is started. Default value: 15. ◦ Period (s): the periodSeconds field in the YAML file. This field specifies the interval (in seconds) at which probes are performed. Default value: 10. Minimum value: 1. ◦ Timeout (s): the timeoutSeconds field in the YAML file. This field specifies the time (in seconds) after which a probe times out. Default value: 1. Minimum value: 1. ◦ Healthy Threshold: the minimum number of times that an unhealthy container must consecutively pass health checks before it is considered healthy. Default value: 1. Minimum value: 1. For liveness probes, this parameter must be set to 1. ◦ Unhealthy Threshold: the minimum number of times that a healthy container must consecutively fail health checks before it is considered unhealthy. Default value: 3. Minimum value: 1.
Command	Runs a probe command in the container to check the health status of the container. You can set the following parameters: ◦ Command: Enter the probe command that is run to check the health status of the container. ◦ Initial Delay (s): the initialDelaySeconds field in the YAML file. This field specifies the time period (in seconds) that the system must wait before it can send a probe to the container after the container is started. Default value: 5. ◦ Period (s): the periodSeconds field in the YAML file. This field specifies the time interval (in seconds) at which probles are performed. Default value: 10. Minimum value: 1. ◦ Timeout (s): the timeoutSeconds field in the YAML file. This field specifies the time (in seconds) after which a probe times out. Default value: 1. Minimum value: 1. ◦ Healthy Threshold: the minimum number of times that an unhealthy container must consecutively pass health checks before it is considered healthy. Default value: 1. Minimum value: 1. For liveness probes, this parameter must be set to 1. ◦ Unhealthy Threshold: the minimum number of times that a healthy container must consecutively fail health checks before it is considered unhealthy. Default value: 3. Minimum value: 1.

5. In the **Lifecycle** section, set the lifecycle of the container.

You can set the following parameters to configure the lifecycle of the container: Start, Post Start, and Pre Stop. For more information, see Configure the lifecycle of a container.

- **Start**: Set the command and parameter that take effect before the container starts.
- **Post Start**: Set the command that takes effect after the container starts.
- **Pre Stop**: Set the command that takes effect before the container stops.

6. In the **Volume** section, add on-premises storage volumes or persistent volume claims (PVCs).

   The following types of storage volumes are supported:

   i. On-premises storage volume
   ii. PVC
   iii. NAS
   iv. Disk

7. Click **Next** to configure advanced settings.

Step 3: Configure advanced settings

On the **Advanced** wizard page, configure access control, scaling configurations, labels, and annotations.

1. In the **Access Control** section, configure the access control settings to expose the backend pods.

   Note

   You can configure access control settings based on your business requirements:

   - Internal applications: For applications that run inside the cluster, you can create a Service of the **ClusterIP** or **NodePort** type to enable internal communication.
   - External applications: For applications that are exposed to the Internet, you can configure access control by using one of the following methods:
     - Create a LoadBalancer Service. When you create a Service, set Type to **Server Load Balancer**. You can select or create a Server Load Balancer (SLB) instance for the Service and use the Service to expose your application to the Internet.
     - Create an Ingress and use the Ingress to expose your application to the Internet. For more information, see Ingress.

   You can specify how the backend pods are exposed to the Internet. In this example, a ClusterIP Service and an Ingress are created to expose the NGINX application to the Internet.

   - To create a Service, click **Create** on the right side of **Services**. In the Create Service dialog box, set the parameters.

Parameter	Description
Name	The name of the Service. In this example, nginx-svc is used.

Parameter	Description
Type	The type of Service. This parameter determines how the Service is accessed. In this example, **Cluster IP** is selected. ■ **Cluster IP**: exposes the Service by using an internal IP address of the cluster. If you select this type, the Service is accessible only within the cluster. This is the default type. **Note** The **Headless Service** check box is available only when you set Type to **Cluster IP**. ■ **Server Load Balancer**: exposes the Service by using an SLB instance. If you select this type, you can enable internal or external access to the Service. SLB instances can be used to route requests to NodePort and ClusterIP Services. ■ Create SLB Instance: You can click **Modify** to change the specification of the SLB instance. ■ Use Existing SLB Instance: You can select an existing SLB instance from the drop-down list. **Note** You can create an SLB instance or use an existing SLB instance. You can also associate an SLB instance with more than one Service. However, you must take note of the following limits: ■ If you use an existing SLB instance, the listeners of the SLB instance overwrite the listeners of the Service. ■ If an SLB instance is created along with a Service, you cannot use this SLB instance to expose other Services. Otherwise, the SLB instance may be deleted. You can reuse only SLB instances that are manually created in the console or by calling the API. ■ An SLB instance must listen on different Service ports if the SLB instance exposes more than one Service. Otherwise, port conflicts may occur. ■ When an SLB instance exposes multiple Services, the names of listeners and vServer groups are used as unique identifiers in Kubernetes. Do not modify the names of listeners and vServer groups. ■ You cannot use an SLB instance to expose Services across clusters.
Port Mapping	Specify a Service port and a container port. The container port must be the same as the one that is exposed in the backend pod.

Parameter	Description
External Traffic Policy	■ Local: routes network traffic to only the node where the Service is deployed. ■ Cluster: routes network traffic to pods on other nodes. **Note** The **External Traffic Policy** parameter is available only when you set Type to **Node Port** or **Server Load Balancer**.
Annotations	Add one or more annotations to the Service to modify the configuration of the SLB instance. For example, `service.beta.kubernetes.io/alicloud-loadbalancer-bandwidth:20` specifies that the maximum bandwidth of the Service is 20 Mbit/s. This limits the amount of traffic that flows through the Service. For more information, see Use annotations to configure load balancing.
Label	Add one or more labels to the Service. Labels are used to identify the Service.

- To create an Ingress, click **Create** on the right side of **Ingresses**. In the Create dialog box, set the parameters.

  For more information, see Create an Ingress.

  Notice When you deploy an application from an image, you can create an Ingress for only one Service. In this example, the name of a virtual host is used as the test domain name. You must add the following mapping rule to the hosts file to map the domain name to the IP address of the Ingress. The entry is in the format of *<Ingress external endpoint>* + *<Ingress domain name>*. In actual scenarios, use a domain name that has obtained an Internet Content Provider (ICP) number.

  ```
  101.37.XX.XX    foo.bar.com    # The IP address of the Ingress.
  ```

Parameter	Description
Name	Enter a name for the Ingress. In this example, nginx-ingress is used.

Parameter	Description
Rules	Ingress rules are used to enable access to a specified Service in a cluster. For more information, see Create an Ingress. ▪ **Domain**: Enter the domain name of the Ingress. In this example, the test domain name `foo.bar.com` is used. ▪ **Path**: Enter the Service URL. The default path is the root path /. The default path is used in this example. Each path is associated with a backend Service. SLB forwards traffic to a backend Service only when inbound requests match the domain name and path. ▪ **Service**: Set the Service name and port. In this example, nginx-svc is used. ▪ Select **EnableTLS** to enable TLS. For more information, see Advanced NGINX Ingress configurations.
Weight	Set the weight for each Service in the path. Each weight is calculated as a percentage value. Default value: 100.
Annotations	▪ Click **Rewrite Annotation** to add a rewrite annotation for the Ingress. For example, `nginx.ingress.kubernetes.io/rewrite-target:/` specifies that */path* is redirected to the root path /. The root path can be recognized by the backend Service. ▪ You can also click **Add**, and enter the name and value of an annotation. For more information, see Annotations.
Labels	Add labels to describe the characteristics of the Ingress.

After the Service and Ingress are created, you can find the newly created Service and Ingress in the **Access Control** section. Click **Update** or **Delete** to modify the Ingress configurations.

2. In the **Scaling** section, specify whether to enable **HPA** to automatically scale the number of pods based on the CPU and memory usage.

   ACK supports the auto scaling of pods. This allows you to automatically adjust the number of pods based on the CPU and memory usage.

   Note To enable HPA, you must configure the required resources for the container. Otherwise, HPA does not take effect.

Parameter	Description
Metric	Select CPU Usage or Memory Usage. The selected resource type must be the same as that specified in the Required Resources field.
Condition	Specify the resource usage threshold. HPA triggers scale-out activities when the threshold is exceeded.

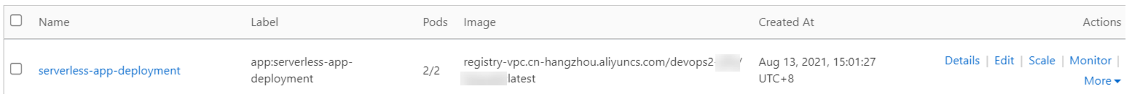
Parameter	Description
Max. Replicas	Specify the maximum number of pod replicas to which the application can be scaled.
Min. Replicas	Specify the minimum number of pod replicas that must run.

3. In the **Labels,Annotations** section, click **Add** to configure labels and annotations for the pod.
   - Pod Labels: Add a label to the pod. A label is used to identify the application.
   - Pod Annotations: Add an annotation to the pod.
4. Click **Create**.

Step 4: Check the application

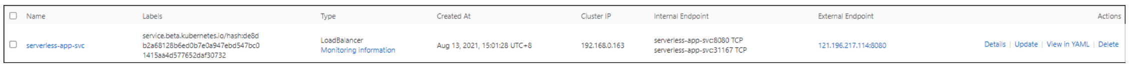
On the **Complete** wizard page, you can view the created application.

1. On the **Complete** wizard page, click **View Details**. On the Deployments page, you can find the newly created application named serverless-app-svc.



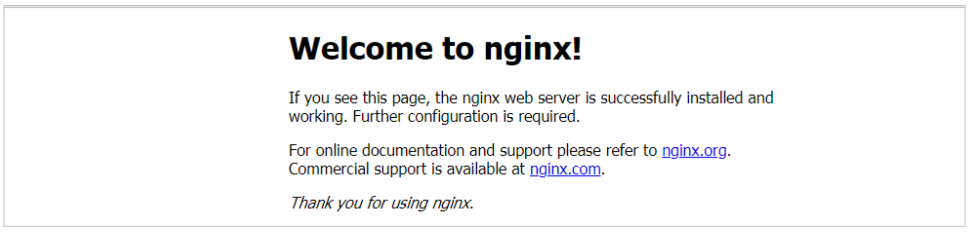
☐	Name	Label	Pods	Image	Created At	Actions
☐	serverless-app-deployment	app:serverless-app-deployment	2/2	registry-vpc.cn-hangzhou.aliyuncs.com/devops2[redacted]/[redacted]latest	Aug 13, 2021, 15:01:27 UTC+8	Details | Edit | Scale | Monitor | More

2. In the left-side navigation pane of the details page of the cluster, choose **Network > Services**. On the Services page, you can find the newly created Service named serverless-app-svc.



☐	Name	Labels	Type	Created At	Cluster IP	Internal Endpoint	External Endpoint	Actions
☐	serverless-app-svc	service.beta.kubernetes.io/hash:de8d b2a68128b6ed0b7e0a947ebd547bc0 1415aa4d577652daf30732	LoadBalancer Monitoring information	Aug 13, 2021, 15:01:28 UTC+8	192.168.0.163	serverless-app-svc:8080 TCP serverless-app-svc:31167 TCP	121.196.217.114:8080	Details | Update | View in YAML | Delete

3. To visit the NGINX welcome page, you can use your browser to access the external endpoint of the Service.



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

9.3. Create a Service

This topic describes how to create a Service in a serverless Kubernetes (ASK) cluster.

Prerequisites

A serverless Kubernetes (ASK) cluster is created. For more information, see Create an ASK cluster.

Context

A Kubernetes Service, also known as a microservice, is an abstraction that defines a logical set of pods and a policy that is used to access the pods. A label selector is used to determine which set of pods is targeted by a Service.

Each pod in Kubernetes clusters has its own IP address. However, pods are frequently created and deleted. Therefore, it is not highly available to directly expose pods to external access. Services decouple the frontend from the backend. The frontend clients do not need to be aware of which backend pods are used. This provides a loosely-decoupled microservice architecture.

For more information, see Kubernetes Services.

Step 1: Create a Deployment

Create a Deployment from an image. In this example, a Deployment named serverless-app-deployment is created. For more information, see Create an application from an image.

Step 2: Create a Service

1.
2.
3.
4.
5. In the upper-right corner of the page, click **Create**.
6. In the Create Service dialog box, set the following parameters:
   - **Name**: the name of the Service. In this example, nginx-svc is used.
   - **Type**: the type of the Service. This parameter specifies how the Service is accessed.
     - Cluster IP: the ClusterIP Service. This type of Service exposes the Service through an internal IP address of the cluster. If you select this option, the Service is accessible only within the cluster. This is the default value.

       Note The **Headless Service** option is available only when you select **Cluster IP**.

  - Server Load Balancer: the LoadBalancer Service. This type of Service is accessed by using Server Load Balancer (SLB) instances. If you select this option, you can enable internal or external access to the Service. An SLB instance can route access to NodePort and ClusterIP Services.

    Note You can create an SLB instance or use an existing SLB instance. You can also associate an SLB instance with more than one Service. However, you must take note of the following limits:

    - If you use an existing SLB instance, the listeners of the SLB instance overwrite the listeners of the Service.
    - If an SLB instance is created along with a Service, you cannot use this SLB instance to expose other Services. Otherwise, the SLB instance may be deleted. You can reuse only SLB instances that are manually created in the console or by calling the API.
    - The SLB instance must listen on different Service ports if the SLB instance exposes more than one Service. Otherwise, port conflicts may occur.
    - When an SLB instance exposes multiple Services, the names of listeners and vServer groups are used as unique identifiers in Kubernetes. Do not modify the names of listeners and vServer groups.
    - You cannot use an SLB instance to expose Services across clusters.

- **Backend**: Select the backend object for the Service. In this example, the previously created serverless-app-deployment application is selected. If you do not associate the Service with a backend object, no Endpoint object will be created. You can also manually bind the Service to an Endpoint object. For more information, see Create a Service without selectors.
- **External Traffic Policy**: Select Local or Cluster.

  Note **External Traffic Policy** is available only when you select **Node Port** or **Server Load Balancer**.

- **Port Mapping**: Set the Service port and container port. The container port must be the same as the one that is exposed by the backend pod.
- **Annotations**: Add annotations to the Service and configure parameters for the SLB instance. For example, an annotation of `service.beta.kubernetes.io/alicloud-loadbalancer-bandwidth:20` specifies that the Service bandwidth is set to 20 Mbit/s. This allows you to limit the traffic that flows through the Service. For more information about annotations, see Use annotations to configure load balancing.
- **Label**: Add labels to the Service.

7. Click **Create**. After the nginx-svc Service is created, it appears on the Services page.

9.4. Delete a Service

This topic describes how to delete a Service in the Container Service for Kubernetes (ACK) console.

Prerequisites

- A serverless Kubernetes (ASK) cluster is created. For more information, see Create an ASK cluster.

- A Service is created. For more information, see Create a Service.

Procedure

1.
2.
3.
4.
5. On the Services page, find the Service that you want to delete and click **Delete** in the **Actions** column.
6. In the **Note** message, click **Confirm**. The Service is deleted and disappears from the Services page.

9.5. View pods

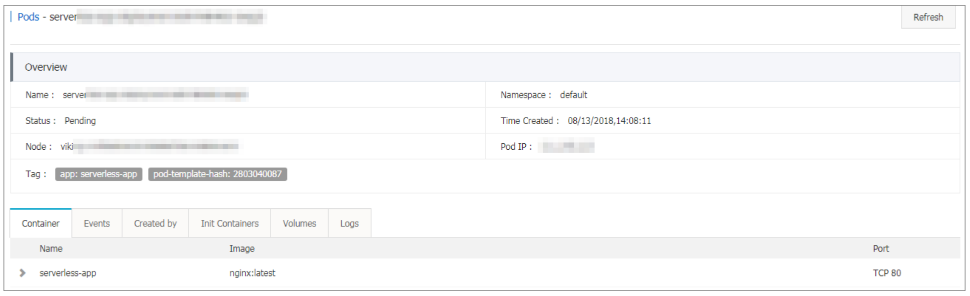
This topic describes how to view the pods of a serverless Kubernetes (ASK) cluster on the Pods tab in the Container Service for Kubernetes (ACK) console.

Procedure

1.
2.
3.
4.
5. Click the **Pods** tab.
6. On the Pods tab, find the pod that you want to view and click View **Details** on the right side of the page.

   > **Note** You can update or delete pods on the Pods tab. We recommend that you use Deployments to manage pods if they are created by Deployments.

7. On the details page of the pod, you can view detailed information about the pod.



Pods - server[redacted]

Refresh

Overview

Name : server[redacted]
Namespace : default
Status : Pending
Time Created : 08/13/2018,14:08:11
Node : viki[redacted]
Pod IP : [redacted]
Tag : app: serverless-app pod-template-hash: 2803040087

Container | Events | Created by | Init Containers | Volumes | Logs

Name	Image	Port
serverless-app	nginx:latest	TCP 80

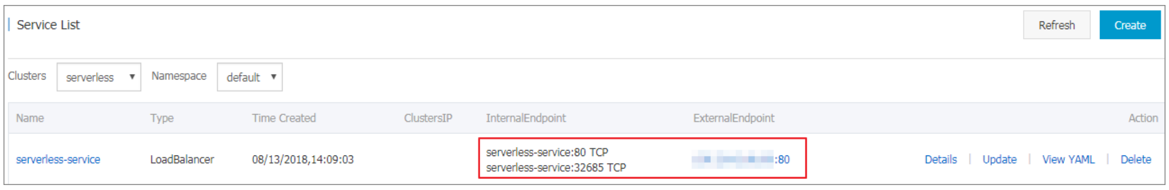
9.6. View a Service

If you create an application that provides external services, Kubernetes Dashboard automatically creates a LoadBalancer Service that uses a Server Load Balancer (SLB) instance. This allows you to direct external access to pods inside the cluster. This topic describes how to view a Service in the Container Service for Kubernetes (ACK) console.

Procedure

1.
2.
3.
4.
5. On the **Services** page, you can view the Services that are created in the cluster.

   You can view information about a Service, such as the name, type, creation time, cluster IP address, and external endpoint.



Service List

Refresh Create

Clusters serverless Namespace default

Name	Type	Time Created	ClustersIP	InternalEndpoint	ExternalEndpoint	Action
serverless-service	LoadBalancer	08/13/2018,14:09:03		serverless-service:80 TCP serverless-service:32685 TCP	[redacted]:80	Details | Update | View YAML | Delete

10.ConfigMaps and Secrets

10.1. Create a ConfigMap

This topic describes how to create a ConfigMap in the Container Service for Kubernetes (ACK) console.

Procedure

1.
2.
3.
4.
5. On the **ConfigMap** tab, click **Create**.
6. Enter a name for the ConfigMap, click **+ Add** to enter the keys and values.
    - **ConfigMap Name**: required. The name of the ConfigMap. The name can contain lowercase letters, digits, hyphens (-), and periods (.). Other resource objects must refer to the name of the ConfigMap so that they can obtain the configurations.
    - **ConfigMap**: Enter keys and values. You can also click **Import** in the upper-right corner of the Create pane to create the ConfigMap from a file.
7. Click **OK**.

Result

After the ConfigMap is created, you can view the created ConfigMap on the ConfigMap tab.

11.Storage management-CSI

11.1. CSI overview

You can use the Container Storage Interface (CSI) plug-in to mount Alibaba Cloud disks and Apsara File Storage NAS (NAS) file systems as volumes to serverless Kubernetes (ASK) clusters. This topic introduces the CSI plug-in and describes the volumes that are supported by the CSI plug-in.

Volumes supported by the CSI plug-in

The CSI plug-in supports statically provisioned volumes and dynamically provisioned volumes. The following table describes the types of volumes that different storage services support.

Alibaba Cloud storage service	Statically provisioned volume	Dynamically provisioned volume
Alibaba Cloud disks	You can use the CSI plug-in to mount a statically provisioned disk volume with a pair of persistent volume (PV) and persistent volume claim (PVC).	Supported
Apsara File Storage NAS	You can use the CSI plug-in to mount a statically provisioned NAS volume with a pair of PV and PVC.	Not supported

Note

- We recommend that you use the CSI plug-in in newly created ASK clusters. ASK will continuously update the CSI plug-in to support more features that are provided by the open source version of CSI.
- The regions that are supported by the CSI plug-in are the same as the regions that are available for Kubernetes 1.16. For more information, see Supported regions.

CSI

The Kubernetes community recommends the CSI plug-in. The CSI plug-in that is used in ASK clusters supports the features provided by the open source version of CSI. The CSI plug-in consists of two components: CSI-Plugin and CSI-Provisioner. For more information, see alibaba-cloud-csi-driver.

Component	Feature	How to install
CSI-Plugin	Allows you to mount and unmount volumes. By default, ASK allows you to mount Alibaba Cloud disks and NAS file systems as volumes.	CSI-Plugin is automatically installed by default.

Component	Feature	How to install
CSI-Provisioner	Allows ASK to automatically create disk volumes.	For more information about how to install CSI-Provisioner, see Install and update CSI-Provisioner.

11.2. Install and update CSI-Provisioner

CSI-Provisioner can automatically create disk volumes. This topic describes how to install and update CSI-Provisioner in a serverless Kubernetes (ASK) cluster.

Prerequisites

- An ASK cluster is created. For more information, see Create an ASK cluster.
- A kubectl client is connected to the cluster. For more information, see Connect to an ACK cluster by using kubectl.

Install CSI-Provisioner

You must manually install CSI-Provisioner in an ASK cluster. If CSI-Provisioner is not installed, you can manually install it in the cluster.

- If the Kubernetes version of the ASK cluster is earlier than 1.22, refer to csi-provisioner for the YAML template of CSI-Provisioner.
- If the Kubernetes version of the ASK cluster is 1.22, refer to csi-provisioner-1.22 for the YAML template of CSI-Provisioner.

Verify the installation

Run the following command to check whether CSI-Provisioner is installed:

```
kubectl get pod -n kube-system | grep csi-provisioner
```

Expected output:

```
NAME               READY    STATUS     RESTARTS    AGE
csi-provisioner    1/1      Running    0           14d
```

The output shows that the pod is in the Running state, which indicates that CSI-Provisioner is installed.

Update CSI-Provisioner

To update CSI-Provisioner, you must change the image address in the YAML template of CSI-Provisioner based on the release notes. For more information about the CSI-Provisioner version details, see csi-provisioner.

11.3. Disk volumes

11.3.1. Disk volume overview

You can create volumes to mount Alibaba Cloud disks to a Container Service for Kubernetes (ACK) cluster. You can use the Contain Storage Interface (CSI) plug-in provided by Alibaba Cloud to mount disks by creating persistent volumes (PVs) and persistent volume claims (PVCs). A PV can be statically or dynamically provisioned. This topic describes the features, disk specifications, use scenarios, limits, and billing rules of disk volumes.

Features

Alibaba Cloud disks are block-level data storage resources for Elastic Compute Service (ECS). Alibaba Cloud disks provide low latency, high performance, high durability, and high reliability. Alibaba Cloud disks use a distributed triplicate mechanism to ensure data reliability for ECS instances. If service disruptions occur within a zone due to hardware errors, data in the zone is automatically replicated to an unaffected disk in another zone to ensure data availability.

- Enhanced SSDs (ESSDs): ESSDs are based on the next-generation distributed block storage architecture and use the 25 Gigabit Ethernet and remote direct memory access (RDMA) technologies. ESSDs provide low-latency input and output. Each ESSD can provide up to 1,000,000 random read/write IOPS. For more information, see ESSDs.

  We recommend that you use ESSDs for scenarios such as online transactional processing (OLTP) databases, NoSQL databases, and Elasticsearch, Logstash, and Kibana (ELK) distributed logs.

- Standard SSDs: Standard SSDs are high-performance disks that provide consistent high random IOPS and high data reliability.

  We recommend that you use standard SSDs for scenarios such as I/O-intensive applications, small and medium-sized relational databases, and NoSQL databases.

- Ultra disks: Ultra disks are cost-effective and provide medium random IOPS and high data reliability.

  We recommend that you use ultra disks as system disks for scenarios such as development and testing.

- Basic disks: Basic disks are the previous generation of disks and are unavailable for purchase.

Disk specifications

The following table describes the performance of disks of different categories.

For more information about disk performance, see EBS performance.

Scenarios

You can use one of the following methods to mount disks if you want to use the disks to store application data:

- Use a statically provisioned disk volume
- Use a dynamically provisioned disk volume

Usage notes

- We recommend that you mount a disk by using a StatefulSet. If you use a Deployment to mount disks, you must set the number of replicated pods to 1. The Deployment cannot guarantee the sequence in which the mounting and unmounting operations are performed. New pods may be launched before the disks are unmounted from the original pods. In this case, the system may fail to

mount the disks to the new pods. In addition, due to the update policy of Deployments, the system may fail to mount the disks to the new pods when the pods are restarted. Therefore, we recommend that you do not use a Deployment to mount disks.
- Make sure that the type of disk matches the ECS instance type that is used in your cluster before you mount a disk. For more information about the matching rules between disk types and ECS instance types, see Instance family.
- You can mount at most 16 disks to each node. The maximum capacity of each disk is 32 TiB.

Billing method

- Only pay-as-you-go disks can be mounted. If you change the billing method of an ECS instance in the cluster from pay-as-you-go to subscription, you cannot change the billing method of its disks to subscription. Otherwise, the disks cannot be mounted to the cluster.
- For more information, visit the ECS product page.

For more information, see Billing methods.

11.3.2. Use a statically provisioned disk volume

Alibaba Cloud disks are block-level data storage resources for Elastic Compute Service (ECS). Alibaba Cloud disks provide low latency, high performance, high durability, and high reliability. Serverless Kubernetes (ASK) allows you to use the Container Storage Interface (CSI) plug-in to statically and dynamically provision disk volumes. This topic describes how to use the CSI plug-in to mount a statically provisioned disk volume and how to enable persistent storage by using a statically provisioned disk volume.

Prerequisites

- An ASK cluster is created. For more information, see Create an ASK cluster.
- A pay-as-you-go disk is created. The disk ID is `d-wz92s6d95go6ki9x****`. For more information, see Create a disk.

  Note The minimum capacity of an ultra disk, an SSD, or an Enhanced SSD (ESSD) is 20 GiB.

- A kubectl client is connected to your cluster. For more information, see Connect to an ACK cluster by using kubectl.

Context

Scenarios:

- You want to create applications that require high disk I/O and do not require data sharing. The applications can use storage services such as MySQL and Redis.
- You want to write log data at a high speed in high concurrency or heavy traffic loads scenarios.
- You want to persist data in a way that is independent of the pod lifecycle.

To mount a disk as a statically provisioned volume, make sure that you have purchased a disk.

Manually create a persistent volume (PV) and a persistent volume claim (PVC) that are used to statically provision a disk.

Limits

- Alibaba Cloud disks cannot be shared. A disk can be mounted only to one pod.

- A disk can be mounted only to a node that is deployed in the same zone as the disk.

Mount a statically provisioned disk volume in the ACK console

Step 1: Create a PV

1.
2.
3.
4.
5. In the upper-right corner of the **Persistent Volumes** page, click **Create**.
6. In the **Create PV** dialog box, configure the following parameters.

Parameter	Description
PV Type	You can select **Cloud Disk** or **NAS**. In this example, **Cloud Disk** is selected.
Volume Plug-in	Select **CSI**.
Access Mode	The default setting is **ReadWriteOnce**.
Disk ID	Select a mountable disk that is deployed in the same region and zone as your cluster.
File System Type	Select the file system of the disk. Valid values: **ext4**, **ext3**, **xfs**, and **vfat**. Default value: **ext4**.
Label	Add labels to the PV.

7. Click **Create**.

Step 2: Create a PVC

1.
2. In the upper-right corner of the **Persistent Volume Claims** page, click **Create**.
3. In the **Create PVC** dialog box, configure the following parameters.

Parameter	Description
PVC Type	You can select **Cloud Disk** or **NAS**. In this example, **Cloud Disk** is selected.
Name	The name of the PVC. The name must be unique in the namespace.

Parameter	Description
Allocation Mode	In this example, Existing Volumes is selected. Note If no PV is created, you can set **Allocation Mode** to **Create Volume** and configure the required parameters to create a PV. For more information, see Create a PV.
Existing Volumes	Click **Select PV**. In the dialog box that appears, find the PV that you want to use and click **Select** in the Actions column.
Capacity	The capacity claimed by the PVC. Note The capacity of a PV cannot be greater than the capacity of the disk that is associated with the PV.
Access Mode	The default setting is **ReadWriteOnce**.

4. Click **Create**.

   After the PVC is created, you can view the PVC in the list of PVCs. The PVC is bound to the specified PV.

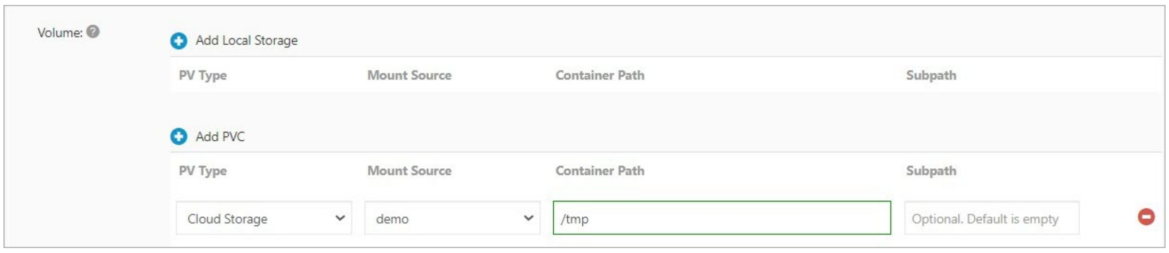
Step 3: Deploy an application

1.
2.
3. Configure the application parameters.

   This example shows how to configure the volume parameters. For more information about other parameters, see Use a StatefulSet to create a stateful application.

   You can configure local storage volumes and cloud storage volumes for an ACK cluster. In this example, **Cloud Storage** is selected.

   Mount the disk volume that is created in this example to the */tmp* path of the container. After the disk volume is mounted, the container data that is generated in the /tmp path is stored in the disk volume.



PV Type	Mount Source	Container Path	Subpath
Cloud Storage	demo	/tmp	Optional. Default is empty

4. Configure other parameters and click **Create**.

   After the application is created, you can use the volume to store application data.

Mount a statically provisioned disk volume by using kubectl

Step 1: Create a PV

1. Use the following template to create a *pv-static.yaml* file:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: csi-pv
  labels:
    alicloud-pvname: static-disk-pv
spec:
  capacity:
    storage: 25Gi
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  csi:
    driver: diskplugin.csi.alibabacloud.com
    volumeHandle: "<your-disk-id>"
  nodeAffinity:
    required:
      nodeSelectorTerms:
      - matchExpressions:
        - key: topology.diskplugin.csi.alibabacloud.com/zone
          operator: In
          values:
          - "<your-node-zone-id>"
```

Parameter	Description
name	The name of the PV.
labels	The labels that you want to add to the PV
storage	The available storage of the disk
accessModes	The access mode of the PV
persistentVolumeReclaimPolicy	The reclaim policy of the PV
driver	The type of driver. This parameter is set to `diskplugin.csi.alibabacloud.com`. This value indicates that the Alibaba Cloud CSI plug-in is used.
volumeHandle	The ID of the disk that is associated with the PV

Parameter	Description
nodeAffinity	Information about the zone to which the PV and PVC belong. You can configure this parameter to specify the zone to which the pod that uses the PV and PVC is scheduled.

2. Run the following command to create the PV:

```
kubectl create -f pv-static.yaml
```

3. View the PV that you created.

   i.

   ii.

   iii.

   iv. In the left-side navigation pane of the details page, choose **Volumes > Persistent Volumes**.

   On the **Persistent Volumes** page, verify that the newly created PV is displayed.

Step 2: Create a PVC

1. Use the following template to create a *pvc-static.yaml* file:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: csi-pvc
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 25Gi
  selector:
    matchLabels:
      alicloud-pvname: static-disk-pv
```

Parameter	Description
`name`	The name of the PVC.
`accessModes`	The access mode of the PVC
`storage`	The capacity claimed by the PVC. The claimed capacity cannot exceed the capacity of the PV that is bound to the PVC.
`matchLabels`	The labels that are used to select a PV and bind the PV to the PVC. The labels must be the same as those of the PV.

2. Run the following command to create the PVC:

```
kubectl create -f pvc-static.yaml
```

3. In the left-side navigation pane of the details page, choose **Volumes > Persistent Volume Claims**.

   On the **Persistent Volume Claims** page, verify that the newly created PVC is displayed.

Step 3: Deploy an application

In this example, a web application is created and mounted with the PVC you created.

1. Use the following template to create a *web.yaml* file:

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: web
spec:
  selector:
    matchLabels:
      app: nginx
  serviceName: "nginx"
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx
        ports:
        - containerPort: 80
          name: web
        volumeMounts:
        - name: pvc-disk
          mountPath: /data
      volumes:
        - name: pvc-disk
          persistentVolumeClaim:
            claimName: csi-pvc
```

   - mountPath: the path where you want to mount the disk in the container.
   - claimName: the name of the PVC, which is used to associate the application with the PVC.

2. Run the following command to create an application and mount the statically provisioned PV by using the PVC:

```
kubectl apply -f web.yaml
```

3. In the left-side navigation pane of the details page, choose **Workloads > StatefulSets**.

   On the **SratefulSets** page, verify that the newly created web application is displayed.

Verify that the statically provisioned disk can be used to persist data

1. View the pods that run the web application and the files in the mounted disk.

   i. Run the following command to query the pods that run the web application:

      ```
      kubectl get pod | grep web
      ```

      Expected output:

      ```
      web-1**** 1/1 Running 0 32s
      ```

   ii. Run the following command to check whether a new disk is mounted to the */data* path:

      ```
      kubectl exec web-1**** df | grep data
      ```

      Expected output:

      ```
      /dev/vdf 20511312 45080 20449848 1% /data
      ```

   iii. Run the following command to query files in the */data* path:

      ```
      kubectl exec web-1**** ls /data
      ```

      Expected output:

      ```
      lost+found
      ```

2. Run the following command to create a file named *static* in the */data* path:

   ```
   kubectl exec web-1**** touch /data/static
   ```

3. Run the following command to query files in the */data* path:

   ```
    kubectl exec web-1**** ls /data
   ```

   Expected output:

   ```
   lost+found
   static
   ```

4. Run the following command to delete the `web-1****` pod:

   ```
   kubectl delete pod web-1****
   ```

   Expected output:

   ```
   pod "web-1****" deleted
   ```

5. Verify that the file still exists in the disk after the pod is deleted.

   i. Run the following command to query the pod that is recreated:

      ```
      kubectl get pod
      ```

      Expected output:

      ```
      NAME        READY   STATUS    RESTARTS    AGE
      web-2****   1/1     Running   0           14s
      ```

ii. Run the following command to query files in the */data* path:

```
kubectl exec web-2**** ls /data
```

Expected output:

```
lost+found
static
```

The *static* file still exists in the disk. This indicates that data is persisted to the statically provisioned disk.

11.3.3. Use a dynamically provisioned disk volume

Alibaba Cloud disks are block-level data storage resources for Elastic Compute Service (ECS). Alibaba Cloud disks provide low latency, high performance, high durability, and high reliability. Serverless Kubernetes (ASK) allows you to use the Container Storage Interface (CSI) plug-in to create dynamically provisioned disk volumes. This topic describes how to use a dynamically provisioned disk volume and how to verify that a dynamically provisioned disk volume can be used to persist data.

Prerequisites

- An ASK cluster is created. For more information, see Create an ASK cluster.
- A kubectl client is connected to the cluster. For more information, see Connect to an ACK cluster by using kubectl.
- The CSI plug-in is manually installed. For more information, see Install and update CSI-Provisioner.

Context

For more information about StorageClasses, see Configure StorageClasses.

Use a dynamically provisioned disk volume in the ACK console

Step 1: Create a StorageClass

1.
2.
3.
4.
5. In the upper-right corner of the **StorageClasses** page, click **Create**.
6. In the **Create** dialog box, configure the parameters.

   The following table describes some of the parameters.

Parameter	Description
Name	The name of the StorageClass. The name must start with a lowercase letter and can contain only lowercase letters, digits, periods (.), and hyphens (-).

Parameter	Description
PV Type	Select **Cloud Disk**.
Volume Plug-in	Select **CSI**.
Parameter	By default, a type parameter is added and set to cloud_essd. The type parameter specifies the disk types. The valid values of the type parameter are cloud_efficiency, cloud_ssd, cloud_essd, and available. You can specify one or more values. However, if you specify available, you cannot specify other values. For example, you can specify `type: cloud_efficiency, cloud_ssd, cloud_essd`. This indicates that the system attempts to create a disk of the specified types in sequence. The system stops trying if a disk is created. If you set the type parameter to available, the system first attempts to create a standard SSD. If the attempt fails, the system attempts to create an ultra disk. The system stops trying if a disk is created. **Note** Some ECS instance types do not support enhanced SSDs. For more information, see FAQ. You can add custom parameters. For example, you can add the zoneId parameter to specify the IDs of the zones where you want to create the disk. If your cluster is deployed in a single zone, set the value to the ID of the zone. Example: `cn-beijing-a`. If your cluster is deployed across zones, you can set the zoneId parameter to multiple zone IDs based on your business requirements. Example: `cn-beijing-a, cn-beijing-b`.
Reclaim Policy	The reclaim policy of the disk. By default, this parameter is set to Delete. You can also set this parameter to Retain. ◦ Delete mode: When persistent volume claims (PVCs) are deleted, the related persistent volumes (PVs) and disks are also deleted. ◦ Retain mode: When PVCs are deleted, the related PVs and disks are retained. The PVs and disk data can only be manually deleted. If you require higher data security, we recommend that you use the Retain mode to prevent data loss caused by user errors.
Binding Mode	The binding mode of the disk. Default value: Immediate, which indicates that the system creates a disk before it creates a pod.

7. After you set the parameters, click **Create**.
   You can find the created StorageClass in the **StorageClasses** list.

Step 2: Create a PVC

1.
2. In the upper-right corner of the **Persistent Volume Claims** page, click **Create**.
3. In the **Create PVC** dialog box, configure the following parameters.

Parameter	Description
PVC Type	You can select Cloud Disk or NAS. In this example, Cloud Disk is selected.
Name	The name of the PVC. The name must be unique in the namespace.
Allocation Mode	In this example, **Use StorageClass** is selected.
Existing Storage Class	Click **Select**. In the **Select Storage Class** dialog box, find the StorageClass that you want to use and click **Select** in the **Actions** column.
Capacity	The capacity claimed by the PVC.
Access Mode	By default, this parameter is set to ReadWriteOnce.

4. Click **Create**.

   After the PVC is created, you can view the PVC in the **PVC** list. The PVC is bound to a PV.

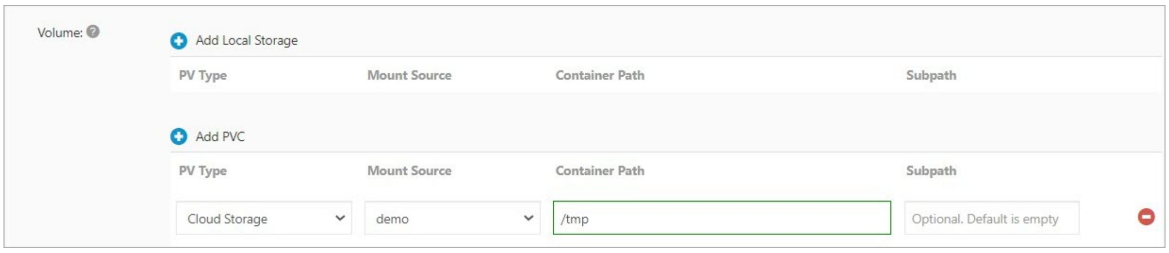
Step 3: Create an application

1.
2.
3. Configure the application parameters.

   This example shows how to configure the volume parameters. For more information about other parameters, see Use a StatefulSet to create a stateful application.

   You can configure local storage volumes and cloud storage volumes for an ACK cluster. In this example, **Cloud Storage** is selected.

   Mount the disk volume that is created in this example to the */tmp* path of the container. After the disk volume is mounted, the container data that is generated in the /tmp path is stored in the disk volume.



PV Type	Mount Source	Container Path	Subpath
Cloud Storage	demo	/tmp	Optional. Default is empty

4. Configure other parameters and click **Create**.

   After the application is created, you can use the volume to store application data.

Use a dynamically provisioned disk volume by using kubectl

Step 1: Create a StorageClass

If your cluster is deployed across multiple zones, you can create a StorageClass that requires the system to create a disk before it creates a pod.

1. Use the following template to create a file named *storage-class-csi.yaml*:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: alicloud-disk-ssd-b
provisioner: diskplugin.csi.alibabacloud.com
parameters:
  type: cloud_ssd
  regionId: cn-beijing
  zoneId: cn-beijing-b
  encrypted: false
reclaimPolicy: Retain
allowVolumeExpansion: true
volumeBindingMode: Immediate
```

Parameter	Description
provisioner	Set this parameter to `diskplugin.csi.alibabacloud.com`. This indicates that the provisioner plug-in for Alibaba Cloud disks is used to create the StorageClass.
type	The type of disk. The valid values are cloud_efficiency, cloud_ssd, cloud_essd, and available. You can specify one or more values. However, if you specify available, you cannot specify other values. For example, you can set this parameter to `type: cloud_efficiency, cloud_ssd, cloud_essd`. This indicates that the system attempts to create a disk of the specified types in sequence. The system stops trying if a disk is created. If you set this parameter to available, the system first attempts to create a standard SSD. If the attempt fails, the system attempts to create an ultra disk. The system stops trying if a disk is created. **Note** Some ECS instance types do not support enhanced SSDs. For more information, see FAQ.
(Optional)regionId	Optional. The region where you want to create a disk.
(Optional)zoneId	Optional. The zone where you want to create a disk.
(Optional)encrypted	Optional. This parameter specifies whether the disk is encrypted. Default value: false. This indicates that the created disk is not encrypted.
reclaimPolicy	The reclaim policy of the disk. By default, this parameter is set to Delete. You can also set this parameter to Retain. ◦ Delete mode: When PVCs are deleted, the related PVs and disks are also deleted. ◦ Retain mode: When PVCs are deleted, the related PVs and disks are retained. The PVs and disk data can only be manually deleted. If you require higher data security, we recommend that you use the Retain mode to avoid data loss caused by user errors.

Parameter	Description
allowVolumeExpansion	If you set this parameter to true, the disk can be automatically expanded.
volumeBindingMode	The binding mode of the disk. Default value: Immediate, which indicates that the system creates a disk before it creates a pod.

2. Run the following command to create a StorageClass:

```
kubectl apply -f storage-class-csi.yaml
```

3. View the created StorageClass.

   i.

   ii.

   iii.

   iv. In the left-side navigation pane of the details page, choose **Volumes > StorageClasses**.

   You can view the created StorageClass on the **StorageClasses** page.

Step 2: Create a PVC

1. Use the following template to create a file named *pvc-ssd.yaml*:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: disk-pvc
spec:
  accessModes:
  - ReadWriteOnce
  volumeMode: Filesystem
  resources:
    requests:
      storage: 25Gi
  storageClassName: alicloud-disk-ssd
```

Parameter	Description
name	The name of the PVC.
accessModes	The access mode of the PVC.
(Optional)volumeMode	Optional. The volume mode of the disk. Valid values: Filesystem and Block. Default value: Filesystem.
storageClassName	The name of the StorageClass that you want to associate with the PVC.
storage	The disk size claimed by the PVC. The minimum capacity is 20 GiB.

2. Run the following command to create the PVC:

```
kubectl create -f pvc-ssd.yaml
```

3. View the created PVC.

   In the left-side navigation pane of the details page, choose **Volumes > Persistent Volume Claims**. You can view the created PVC on the **Persistent Volume Claims** page.

Step 3: Create an application

1. Create a file named *pvc-dynamic.yaml*.

   Use the following template to create an application named *nginx-dynamic* and mount the PVC to the application:

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: nginx-dynamic
spec:
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx
        ports:
        - containerPort: 80
          name: web
        volumeMounts:
        - name: pvc-disk
          mountPath: /data
      volumes:
        - name: pvc-disk
          persistentVolumeClaim:
            claimName: disk-ssd
```

Parameter	Description
mountPath	The path to which the disk is mounted.
claimName	The name of the PVC that is mounted to the application.

2. Run the following command to create an application and mount the PVC to the application:

```
kubectl create -f pvc-dynamic.yaml
```

3. View the deployed application.

   In the left-side navigation pane of the details page, choose **Workloads > StatefulSets**. You can find the created application on the **StatefulSets** page.

11.3.4. Configure StorageClasses

This topic describes different types of StorageClass for serverless Kubernetes (ASK) clusters. This topic also describes how to choose proper StorageClass types to meet your business requirements and how to configure a default StorageClass.

StorageClass

The following types of StorageClass are supported in an ASK cluster.

- alicloud-disk-efficiency: ultra disk.
- alicloud-disk-ssd: standard SSD.
- alicloud-disk-essd: ESSD.
- alicloud-disk-available: a high-availability mode. In this mode, the system attempts to create an SSD in priority. If the SSD resources are exhausted, the system attempts to create an ultra disk.

  Notice For versions earlier than alicloud-csi-provisioner V1.14.8.39-0d749258-aliyun, the system attempts to create a disk in the priority order of enhanced SSD, SSD, and ultra disk.

The four types of StorageClass are suitable for single-zone clusters.

If you specify volumeBindingMode: Immediate and specify only one zone ID in the zoneId parameter of the StorageClass configurations, a cloud disk is created in the specified zone.

Default StorageClass

Kubernetes provides the default StorageClass feature. If a persistent volume claim (PVC) does not specify a StorageClass, the default StorageClass is used to provision a persistent volume (PV) for the PVC. For more information, see Default StorageClass.

Note

- The default StorageClass applies to all PVCs. Exercise caution if your cluster has PVCs for multiple storage media. For example, the default StorageClass may create a cloud disk as the PV for a PVC that claims an Apsara File Storage NAS (NAS) file system. To avoid such situations, ASK clusters do not support the default StorageClass feature. To configure a default StorageClass, perform the following steps.
- You can configure only one default StorageClass for each cluster. If you configure more than one default StorageClass for a cluster, all default StorageClasses become invalid.

1. Configure a default StorageClass.

   Run the following command to set alicloud-disk-ssd as a default StorageClass:

   ```
   kubectl patch storageclass alicloud-disk-ssd -p '{"metadata": {"annotations":{"storagec
   lass.kubernetes.io/is-default-class":"true"}}}'
   ```

   After the default StorageClass is configured, alicloud-disk-ssd is marked as (default)

   ```
   kubectl get sc
   ```

   The following output is returned:

```
NAME                          PROVISIONER                       AGE
alicloud-disk-ssd (default)   diskplugin.csi.alibabacloud.com   96m
```

2. Use the default StroageClass.
    i. The following code provides an example on how to create a PVC without specifying a StorageClass:

    ```
    apiVersion: v1
    kind: PersistentVolumeClaim
    metadata:
      name: disk-pvc
    spec:
      accessModes:
      - ReadWriteOnce
      resources:
        requests:
          storage: 20Gi
    ```

    The cluster automatically creates a cloud disk as the PV based on the default StorageClass alicloud-disk-ssd.

    ```
    kubectl get pvc
    ```

    The following output is returned:

    ```
    NAME       STATUS   VOLUME                   CAPACITY   ACCESS MODES   STORAGECLASS
    AGE
    disk-pvc   Bound    d-bp18pbai447qverm3ttq   20Gi       RWO            alicloud-dis
    k-ssd   49s
    ```

What's next

You can also run the following command to disable the default StorageClass:

```
kubectl patch storageclass alicloud-disk-ssd -p '{"metadata": {"annotations":{"storageclass
.kubernetes.io/is-default-class":"false"}}}'
```

11.4. NAS volumes

11.4.1. NAS volume overview

You can mount Apsara File Storage NAS (NAS) volumes to serverless Kubernetes (ASK) clusters. This topic describes the features, storage types, use scenarios, limits, and billing rules of NAS volumes.

Features

NAS is a cloud service that provides a file storage solution for compute nodes, such as Elastic Compute Service (ECS) instances, nodes in Elastic High-Performance Computing (E-HPC) clusters, and nodes in ACK clusters. NAS is a distributed file storage solution that provides shared access, scalability, high reliability, and high performance.

NAS uses Portable Operating System Interface of UNIX (POSIX)-based APIs and is compatible with native operating systems. NAS provides shared access, ensures data consistency, and implements mutual exclusion by using locks. NAS provides scalable file systems and allows simultaneous access to a NAS file system from multiple ECS instances. The storage capacity of a NAS file system scales in or out when you add or remove files. NAS provides shared data sources for workloads and applications that run on multiple ECS instances or servers.

NAS file system types

NAS provides the following file system types: General-purpose NAS Capacity, General-purpose NAS Performance, and Extreme NAS. For more information, see NAS types.

Scenarios

- NAS provides shared storage. You can mount NAS file systems as statically provisioned volumes to meet the requirements of diverse scenarios.
- NAS file systems can be mounted to ASK clusters only as statically provisioned volumes. Dynamic provisioning is not supported by NAS file systems. For more information about how to mount a statically provisioned NAS volume, see Mount a statically provisioned NAS volume.

Usage notes

- Apsara File Storage NAS is a shared storage service. A persistent volume claim (PVC) that is used to mount a NAS file system can be shared among pods.
- Do not delete the mount target before you unmount the NAS file system. Otherwise, the operating system hang error may occur.
- After a mount target is created, wait until the **state** of the mount target changes to **Available**.
- We recommend that you use the NFSv3 file sharing protocol.
- General-purpose and Extreme NAS file systems have different limits on mounting scenarios, the number of file systems, and file sharing protocols. For more information, see Limits.

Billing

For more information about the billing rules of NAS, see NAS billing.

11.4.2. Mount a statically provisioned NAS volume

Apsara File Storage NAS (NAS) is a distributed file system that supports shared access, elastic scaling, high reliability, and high performance. This topic describes how to mount a statically provisioned NAS volume, and how to enable persistent storage and shared storage by using a statically provisioned NAS volume.

Prerequisites

- A serverless Kubernetes (ASK) cluster is created. For more information, see Create an ASK cluster.
- A NAS file system is created. For more information, see 创建文件系统.

  If you want to encrypt data in a NAS volume, configure the encryption settings when you create the NAS file system.
- A mount target is created for the NAS file system. For more information, see Manage mount targets.

The mount target and the cluster node to which you want to mount the NAS file system must belong to the same virtual private cloud (VPC).

- A kubectl client is connected to your cluster. For more information, see Connect to an ACK cluster by using kubectl.

Scenarios

- Your application requires high disk I/O.
- You want to share files across hosts. For example, you want to use a NAS file system as a file server.

Usage notes

- To mount an Extreme NAS file system, set the `path` parameter of the NAS volume to a subdirectory of */share*. For example, you can specify the */share/path1* subdirectory when you mount an Extreme NAS file system to a pod.
- If a NAS file system is mounted to multiple pods, the data in the file system is shared by the pods. In this case, the application must be able to synchronize data across the pods if the data in the NAS file system is modified by multiple pods.

  Note You cannot grant permissions to access the / directory (root directory) of the NAS file system. The user account and user group to which the directory belongs cannot be modified.

- If the securityContext.fsgroup parameter is set in the application template, kubelet performs the `chmod` or `chown` operation after the volume is mounted. This increases the mounting time.

  Note For more information about how to speed up the mounting process when the securityContext.fsgroup parameter is set, see Why does it require a long time to mount a NAS volume?.

Mount a statically provisioned NAS volume in the console

Step 1: Create a PV

1.
2.
3.
4.
5. In the upper-right corner of the **Persistent Volumes** page, click **Create**.
6. In the **Create PV** dialog box, configure the following parameters.

Parameter	Description
PV Type	You can select **Cloud Disk** or **NAS**. In this example, **NAS** is selected.
Name	The name of the PV that you want to create. The name must be unique in the cluster. In this example, **pv-nas** is used.
Volume Plug-in	Select **CSI**.

Parameter	Description
Capacity	The capacity of the PV. A NAS file system provides unlimited capacity. This parameter does not limit the storage usage of the NAS file system but defines the capacity of the PV.
Access Mode	You can select **ReadWriteMany** or **ReadWriteOnce**. Default value: **ReadWriteMany**.
Mount Target Domain Name	You can select **Select Mount Target** to select a mount target or select **Custom** to enter a mount target.
Show Advanced Options	○ **Subdirectory**: the subdirectory of the NAS file system that you want to mount. The subdirectory must start with a forward slash (/). After you set this parameter, the PV is mounted to the subdirectory. ■ If the specified subdirectory does not exist, the system automatically creates the subdirectory in the NAS file system and mounts the subdirectory to the cluster. ■ If you do not set this parameter, the root directory of the NAS file system is mounted. ■ If you want to mount an Extreme NAS file system, the subdirectory must be under the */share* directory. ○ **Version**: the version of the PV.
Label	Add labels to the PV.

7. Click **Create**.

Step 2: Create a PVC

1.
2. In the upper-right corner of the **Persistent Volume Claims** page, click **Create**.
3. In the **Create PVC** dialog box, configure the required parameters.

Parameter	Description
PVC Type	You can select **Cloud Disk** or **NAS**. In this example, **NAS** is selected.
Name	The name of the persistent volume claim (PVC). The name must be unique in the cluster.
Allocation Mode	In this example, Existing Volumes is selected. **Note** If no PV is created, you can set **Allocation Mode** to **Create Volume** and set the required parameters to create a PV. For more information, see Create a PV.

Parameter	Description
Existing Volumes	Click **Select PV**. Find the PV that you want to use and click **Select** in the Actions column.
Capacity	The capacity claimed by the PVC. **Note** The capacity claimed by the PVC cannot exceed the capacity of the PV that is bound to the PVC.

4. Click **Create**.

   After the PVC is created, you can view the PVC in the PVCs list. The PVC is bound to the corresponding PV.

Step 3: Create an application

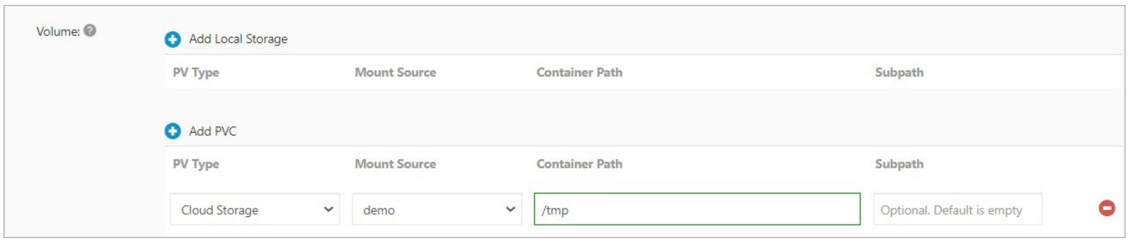
1.
2.
3. Configure the application parameters.

   This example shows how to configure the volume parameters. For more information about other parameters, see Create a stateless application by using a Deployment.

   You can add local volumes and cloud volumes.

   - **Add Local Storage**: You can select ConfigMap, Secret, or EmptyDir from the PV Type drop-down list. Then, set the Mount Source and Container Path parameters to mount the volume to a container path. For more information, see Volumes.
   - **Add PVC**: You can add cloud volumes.

   In this example, a NAS volume is mounted to the */tmp* path in the container.



4. Set other parameters and click **Create**.

   After the application is created, you can use the volume to store application data.

Mount a statically provisioned NAS volume by using kubectl

1. Run the following command to create a statically provisioned PV:

```
kubectl create -f pv-nas.yaml
```

   The following YAML template provides an example on how to create a statically provisioned PV:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-nas
  labels:
    alicloud-pvname: pv-nas
spec:
  capacity:
    storage: 5Gi
  accessModes:
    - ReadWriteMany
  csi:
    driver: nasplugin.csi.alibabacloud.com
    volumeHandle: pv-nas
    volumeAttributes:
      server: "2564f4****-ysu87.cn-shenzhen.nas.aliyuncs.com"
      path: "/csi"
  mountOptions:
  - nolock,tcp,noresvport
  - vers=3
```

Parameter	Description
name	The name of the PV.
labels	The labels that you want to add to the PV
storage	The capacity of the NAS volume.
accessModes	The access mode of the PV.
driver	The type of driver. In this example, the parameter is set to `nasplugin.csi.alibabacloud.com`. This indicates that the NAS Container Storage Interface (CSI) plug-in provided by Alibaba Cloud is used.
volumeHandle	The unique identifier of the PV. If multiple PVs are used, the identifier of each PV must be unique.
server	The mount target of the NAS file system.
path	The subdirectory of the NAS file system that you want to mount. If you want to mount an Extreme NAS file system, the subdirectory must be under the */share* directory.
vers	The version of the Network File System (NFS) protocol. We recommend that you use NFSv3. Extreme NAS file systems support only NFSv3.

2. Run the following command to create a PVC used for static provisioning:

   When you create a PVC of the NAS type, set the `selector` parameter to specify how to select a

PV and bind it to the PVC.

```
kubectl create -f pvc-nas.yaml
```

The following YAML template provides an example on how to create a PVC used for static provisioning:

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-nas
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 5Gi
  selector:
    matchLabels:
      alicloud-pvname: pv-nas
```

Parameter	Description
name	The name of the PVC.
accessModes	The access mode of the PVC.
storage	The capacity claimed by the PVC. The claimed capacity cannot exceed the capacity of the PV that is bound to the PVC.
matchLabels	The labels are used to select a PV and bind it to the PVC.

3. Run the following command to create an application named **nas-static** and mount the created PVC to the application:

```
kubectl create -f nas.yaml
```

The following YAML template provides an example of the *nas.yaml* file that is used to create the **nas-static** application.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nas-static
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx
        ports:
        - containerPort: 80
        volumeMounts:
          - name: pvc-nas
            mountPath: "/data"
      volumes:
        - name: pvc-nas
          persistentVolumeClaim:
            claimName: pvc-nas
```

Parameter	Description
mountPath	The path of the container to which the NAS volume is mounted.
claimName	The name of the PVC that is mounted to the application.

4. Run the following command to query the pods that run the application:

```
kubectl get pod
```

Expected output:

```
NAME                              READY   STATUS    RESTARTS   AGE
nas-static-5b5cdb85f6-n****       1/1     Running   0          32s
nas-static-c5bb4746c-4****        1/1     Running   0          32s
```

Verify that the NAS file system can be used to persist data

1. Query the pods that run the application and the files in the mounted NAS file system.

i. Run the following command to query the pods that run the application:

```
kubectl get pod
```

Expected output:

```
NAME                              READY   STATUS    RESTARTS   AGE
nas-static-5b5cdb85f6-n****       1/1     Running   0          32s
nas-static-c5bb4746c-4****        1/1     Running   0          32s
```

ii. Run the following command to query files in the */data* path of a pod. The pod `nas-static-5b5cdb85f6-n****` is used as an example:

```
kubectl exec nas-static-5b5cdb85f6-n**** ls /data
```

No output is returned. This indicates that no file is stored in the */data* path.

2. Run the following command to create a file named *nas* in the */data* path of the pod `nas-static-5b5cdb85f6-n****`:

```
kubectl exec nas-static-5b5cdb85f6-n**** touch /data/nas
```

3. Run the following command to query files in the */data* path of the pod `nas-static-5b5cdb85f6-n****`:

```
kubectl exec nas-static-5b5cdb85f6-n**** ls /data
```

Expected output:

```
nas
```

4. Run the following command to delete the pod:

```
kubectl delete pod nas-static-5b5cdb85f6-n****
```

5. Open another command-line interface (CLI) and run the following command to view how the pod is deleted and recreated:

```
kubectl get pod -w -l app=nginx
```

6. Verify that the file still exists after the pod is deleted.

i. Run the following command to query the name of the recreated pod:

```
kubectl get pod
```

Expected output:

```
NAME                                    READY   STATUS    RESTARTS   AGE
nas-static-5b5cdb85f6-n****       1/1     Running   0          32s
nas-static-c5bb4746c-4****        1/1     Running   0          32s
```

ii. Run the following command to query files in the */data* path of the pod `nas-static-5b5cdb85f6-n****`:

```
kubectl exec nas-static-5b5cdb85f6-n**** ls /data
```

Expected output:

```
nas
```

The *nas* file still exists in the /data path. This indicates that data is persisted to the NAS file system.

Verify that data in the NAS file system can be shared across pods

1. Query the pods that run the application and the files in the mounted NAS file system.

   i. Run the following command to query the pods that run the application:

   ```
   kubectl get pod
   ```

   Expected output:

   ```
   NAME                           READY   STATUS    RESTARTS   AGE
   nas-static-5b5cdb85f6-n****    1/1     Running   0          32s
   nas-static-c5bb4746c-4****     1/1     Running   0          32s
   ```

   ii. Run the following command to query files in the */data* path of each pod:

   ```
   kubectl exec nas-static-5b5cdb85f6-n**** ls /data
   kubectl exec nas-static-c5bb4746c-4**** ls /data
   ```

2. Run the following command to create a file named *nas* in the */data* path of a pod:

   ```
   kubectl exec nas-static-5b5cdb85f6-n**** touch /data/nas
   ```

3. Run the following command to query files in the */data* path of each pod:

   i. Run the following command to query files in the */data* path of the pod `nas-static-5b5cdb85f6-n****`:

   ```
   kubectl exec nas-static-5b5cdb85f6-n**** ls /data
   ```

   Expected output:

   ```
   nas
   ```

   ii. Run the following command to query files in the */data* path of the pod `nas-static-c5bb4746c-4****`:

   ```
   kubectl exec nas-static-c5bb4746c-4**** ls /data
   ```

   Expected output:

   ```
   nas
   ```

   When you create a file in the */data* path of one pod, you can also find the file in the */data* path of the other pod. This indicates that data in the NAS file system is shared by the two pods.

12.Storage management-Flexvolume

12.1. Disk volumes

12.1.1. Disk volume overview

You can mount disk volumes to serverless Kubernetes (ASK) clusters. This topic describes the features, disk specifications, use scenarios, usage notes, and billing rules of disk volumes.

Usage notes

You can mount both statically and dynamically provisioned disk volumes to ASK clusters.

- For more information about how to mount a statically provisioned disk volume, see Statically provision disks.
- For more information about how to mount a dynamically provisioned disk volume, see Dynamically provision a disk volume by using the CLI or Use a dynamically provisioned disk volume in the console.

12.1.2. Statically provision disks

Prerequisites

Use a disk through a PV and a PVC

1. Create a persistent volume (PV) of the disk type.

   You can create the PV in the Container Service for Kubernetes (ACK) console or by using a YAML file.

   -
   - You can also create the PV in the ACK console.

     a.

     b.

     c. In the left-side navigation pane of the details page, choose **Volumes > Persistent Volumes**.

     d.

     e.

     f.

2.

3.

12.1.3. Dynamically provision a disk volume by using the CLI

To dynamically provision a disk as a persistent volume (PV), you must manually create a StorageClass, and set the storageClassName field in a persistent volume claim (PVC) to specify the disk type.

Create a StorageClass with a specified zone ID (zoneId)

1. Create a *storage-class.yaml* file and copy the following content to the file:

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: alicloud-disk-ssd-hangzhou-b
provisioner: alicloud/disk
parameters:
  type: cloud_ssd
  regionId: cn-hangzhou
  zoneId: cn-hangzhou-b
reclaimPolicy: Retain
```

The following table describes the parameters.

Parameter	Description
`provisioner`	Set the value to alicloud/disk. This is a volume plug-in used to provision disks.
`type`	The disk type. Valid values include cloud_efficiency, cloud_ssd, cloud_essd, and available. If you set this parameter to available, the system attempts to create a disk in the following order: enhanced SSD (ESSD), standard SSD, and ultra disk. The system keeps trying until a disk is created.
`regionId`	The region where you want to create the disk.
`reclaimPolicy`	The policy to reclaim the disk. By default, this parameter is set to Delete. You can also set this parameter to Retain. If you require high data security, we recommend that you set this parameter to Retain to avoid data loss caused by user errors.
`zoneId`	The zone where you want to create the disk. For a multi-zone cluster, you can specify multiple zones. Example: `zoneId: cn-hangzhou-a,cn-hangzhou-b,cn-hangzhou-c`
`encrypted`	Optional. This parameter specifies whether the disk is encrypted. By default, this parameter is set to false. This specifies that the disk is not encrypted.

2. Run the following command to create a StorageClass:

```
kubectl apply -f storage-class.yaml
```

Create a StorageClass in WaitForFirstConsumer mode

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
   name: alicloud-disk-topology-ssd
provisioner: alicloud/disk
parameters:
    type: cloud_ssd
reclaimPolicy: Retain
volumeBindingMode: WaitForFirstConsumer
```

Note

- If you do not create a StorageClass in WaitForFirstConsumer mode and the zoneid parameter is not set, a PV is created in the zone where the Disk-Controller component is deployed.
- If you do not create a StorageClass in WaitForFirstConsumer mode but the zoneid parameter is set, the system attempts to create a PV in the specified zones based on the round-robin algorithm.
- If you create a StorageClass in WaitForFirstConsumer mode, a disk is created for the node to which the pod that consumes the PVC is scheduled. The disk is created in the zone of the scheduled pod.

Create a PVC

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: disk-ssd
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: alicloud-disk-ssd-hangzhou-b
  resources:
    requests:
      storage: 20Gi
---
kind: Pod
apiVersion: v1
metadata:
  name: disk-pod-ssd
spec:
  containers:
  - name: disk-pod
    image: nginx
    volumeMounts:
      - name: disk-pvc
        mountPath: "/mnt"
  restartPolicy: "Never"
  volumes:
    - name: disk-pvc
      persistentVolumeClaim:
        claimName: disk-ssd
```

The following default settings are included:

In a multi-zone cluster, you must manually create a StorageClass to specify the zone where a disk is created.

By default, the following types of StorageClass are provided for single-zone clusters:

- alicloud-disk-efficiency: ultra disk.
- alicloud-disk-ssd: standard SSD.
- alicloud-disk-essd: ESSD.
- alicloud-disk-available: a high-availability mode. In this mode, the system first attempts to create a standard SSD. If SSD resources are exhausted, the system attempts to create an ultra disk.

  Notice For alicloud-disk-controller versions earlier than v1.14.8.44-c23b62c5-aliyun, the system attempts to create a disk in the following order: ESSD, standard SSD, and ultra disk. The system keeps trying until a disk is created.

- alicloud-disk-topology: creates a disk in WaitForFirstConsumer mode.

Create a multi-instance StatefulSet by using a disk

You can use the volumeClaimTemplates parameter to dynamically create multiple PVCs and PVs and bind the PVs to PVCs.

```
apiVersion: v1
kind: Service
metadata:
  name: nginx
  labels:
    app: nginx
spec:
  ports:
  - port: 80
    name: web
  clusterIP: None
  selector:
    app: nginx
---
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: web
spec:
  selector:
    matchLabels:
      app: nginx
  serviceName: "nginx"
  replicas: 2
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx
        ports:
        - containerPort: 80
          name: web
        volumeMounts:
        - name: disk-ssd
          mountPath: /data
  volumeClaimTemplates:
  - metadata:
      name: disk-ssd
    spec:
      accessModes: [ "ReadWriteOnce" ]
      storageClassName: "alicloud-disk-ssd-hangzhou-b"
      resources:
        requests:
          storage: 20Gi
```

You can also use dynamically provisioned disk volumes in the Container Service for Kubernetes (ACK) console. For more information, see Use a dynamically provisioned disk volume in the ACK console.

12.1.4. Use a dynamically provisioned disk volume in the console

This topic describes how to use a dynamically provisioned disk volume in the Container Service for Kubernetes (ACK) console.

Prerequisites

A serverless Kubernetes (ASK) cluster is created. For more information, see Create an ASK cluster.

Step 1: Create a StorageClass

1.
2.
3.
4.
5. On the **StorageClasses** page, click **Create**. In the **Create** dialog box that appears, set the following parameters:
    - **Name**: the name of the disk.
    - **PV Type**: Set the value to Cloud Disk. This specifies that the provisioner plug-in for Alibaba Cloud disks is used to create the StorageClass.
    - **Volume Plug-in**: In this example, **Flexvolume** is selected.
    - **Parameter**: In this example, the parameters are type and zoneid.
        - type: the disk type. Valid values: cloud_efficiency, cloud_ssd, cloud_essd, and available. If you set this parameter to available, the system attempts to create a disk until it succeeds. The system selects a disk type in sequence from this list: an enhanced SSD (ESSD), a standard SSD, and an ultra disk. The system keeps trying until a disk is created.
        - zoneid: This parameter specifies the region where the disk is created.

          For a multi-zone cluster, you can specify multiple zones. Example:

          ```
          zoneid: cn-hangzhou-a,cn-hangzhou-b,cn-hangzhou-c
          ```
        - encrypted: optional. This parameter specifies whether the disk to be created is encrypted. By default, this parameter is set to false. This specifies that the disk to be created is not encrypted.
    - **Reclaim Policy**: the policy that is used to reclaim a disk. By default, this parameter is set to Delete. You can also set this parameter to Retain. If you require higher data security, we recommend that you set this parameter to Retain to avoid data loss caused by user errors.
    - **Binding Mode**: Valid values: Immediate and WaitForFirstConsumer. Default value: Immediate.
    - **Mount Options**: When you mount a volume, you can add multiple mount options.
6. Click **Create**.

Step 2: Create a PVC

1.

2.
3.
4.
5. In the upper-right corner of the **Persistent Volume Claims** page, click **Create**. In the **Create PVC** dialog box, set the required parameters.
    - **PVC Type**: **Cloud Disk**, **NAS**, and **OSS** are supported. In this example, **Cloud Disk** is selected.
    - **Name**: the name of the persistent volume claim (PVC). The name must be unique in the namespace.
    - **Allocation Mode**: **Use StorageClass**, **Existing Volumes**, and **Create Volume** are supported. In this example, **Use StorageClass** is selected.
    - **Existing Storage Class**: Click **Select**. Find the StorageClass that you want to use and click **Select** in the Actions column.
    - **Capacity**: the capacity of the PVC.

      Note The capacity of the PVC cannot exceed the capacity of the disk.

    - **Access Mode**: The default value is ReadWriteOnce.
6. Click **Create**.
   After the PVC is created, the PVC named test-cloud appears in the list of PVCs. The PVC is associated with the specified PV.

Step 3: Create an application that has the PVC mounted

1.
2.
3.
4.
5.
6. Set the parameters that are required to create a Deployment.

   This example shows how to set the volume parameters. For more information about other parameters, see Create a stateless application by using a Deployment.

   You can add local volumes and cloud volumes.

    - **Local Storage**: You can select HostPath, ConfigMap, Secret, or EmptyDir. The source directory or file is mounted to a path in the container. For more information, see Volumes.
    - **Cloud Storage**: supports the following types of persistent volumes (PVs): disks, Apsara File Storage NAS (NAS) file systems, and Object Storage Service (OSS).

   In this example, a PV is created from a disk, and the PV is mounted to the */tmp* path in the container. The data that is generated in this path is stored in the disk.
7. Set other parameters and click **Create**.

   After the disk volume is created, you can use the disk volume.

You can also run commands to create a dynamically provisioned disk volume. For more information, see Dynamically provision a disk volume by using the CLI.

12.2. NAS volumes

12.2.1. NAS volume overview

You can use Apsara File Storage NAS (NAS) volumes in Container Service for Kubernetes (ACK) clusters. This topic describes the features, types, use scenarios, limits, and billing rules of NAS volumes.

Features

NAS is a cloud service that provides a file storage solution for compute nodes, such as Elastic Compute Service (ECS) instances, nodes in Elastic High-Performance Computing (E-HPC) clusters, and nodes in ACK clusters. NAS is a distributed file storage solution that provides shared access, scalability, high reliability, and high performance.

NAS uses Portable Operating System Interface of UNIX (POSIX)-based APIs and is compatible with native operating systems. NAS provides shared access, ensures data consistency, and implements mutual exclusion by using locks. NAS provides scalable file systems and allows simultaneous access to a NAS file system from multiple ECS instances. The storage capacity of a NAS file system scales in or out when you add or remove files. NAS provides shared data sources for workloads and applications that run on multiple ECS instances or servers.

NAS file system types

NAS provides the following file system types: General-purpose NAS Capacity, General-purpose NAS Performance, and Extreme NAS. For more information, see NAS types.

Use scenarios

- NAS provides shared storage. You can mount NAS file systems as statically provisioned volumes to meet the requirements of diverse scenarios.
- In serverless Kubernetes (ASK) clusters, you can mount only statically provisioned NAS volumes. Dynamically provisioned NAS volumes are not supported. For more information about how to mount statically provisioned NAS volumes, see Mount a statically provisioned NAS volume.

Precautions

- NAS provides shared storage. A persistent volume claim (PVC) that is mounted with a NAS file system can be shared among pods.
- Do not delete the mount target if the related NAS file system is still mounted. Otherwise, the operating system hang may occur.
- After a mount target is created, wait until the **status** of the mount target changes to **Available**.
- We recommend that you use NFSv3.
- We recommend that you upgrade FlexVolume to the latest version before you use NAS volumes.
- Extreme NAS file systems support only NFSv3. You must specify the `nolock` parameter when you mount the file systems.

Billing

For more information about the billing rules of NAS, see NAS billing.

12.2.2. Mount a statically provisioned NAS volume

You can use the FlexVolume plug-in provided by Alibaba Cloud to mount Apsara File Storage NAS (NAS) file systems to Container Service for Kubernetes (ACK) clusters. This topic describes how to mount a statically provisioned NAS volume.

Prerequisites

- A NAS file system is created and a mount target is added to the file system. To create a NAS file system and add a mount target, log on to the NAS console. The mount target of the NAS file system and your cluster are deployed in the same virtual private cloud (VPC).
- FlexVolume is upgraded to the latest version.
- A kubectl client is connected to the cluster. For more information, see Connect to ACK clusters by using kubectl.

Context

After FlexVolume installed in the cluster, you can mount NAS file systems by using persistent volumes (PVs) and persistent volume claims (PVCs).

Procedure

You can mount a NAS file system by using a PV and a PVC.

1. Create a PV.

   You can create the PV in the ACK console or by using a YAML file.

   - Create a PV by using a YAML file.

     Use the following *nas-pv.yaml* file to create a PV:

     ```
     apiVersion: v1
     kind: PersistentVolume
     metadata:
       name: pv-nas
     spec:
       capacity:
         storage: 5Gi
       storageClassName: nas
       accessModes:
         - ReadWriteMany
       flexVolume:
         driver: "alicloud/nas"
         options:
           server: "0cd8b4a576-u****.cn-hangzhou.nas.aliyuncs.com"
           path: "/k8s"
           vers: "3"
           options: "nolock,tcp,noresvport"
     ```

   - Create the PV in the ACK console.

     a.

b.

c.

d. In the left-side navigation pane of the cluster details page, choose **Volumes > Persistent Volumes**.

e. On the **Volumes** page, click **Create** in the upper-right corner of the page.

f. In the Create PV dialog box, set the parameters.

Parameter	Description
PV Type	In this example, NAS is selected.
Volume Name	The name of the PV that you want to create. The name must be unique in the cluster. In this example, **pv-nas** is used.
Volume Plug-in	In this example, Flexvolume is selected. For more information about volume plug-ins, see Differences between the CSI and FlexVolume plug-ins.
Capacity	The capacity of the PV that you want to create. The capacity of the PV cannot exceed that of the NAS file system to be mounted.
Access Mode	Default value: ReadWriteMany.
Mount Target Domain Name	The domain name of the mount target that is used to mount the NAS file system to the cluster. For more information about how to manage the mount targets of a NAS file system, see Manage mount targets.
Subdirectory	Enter a subdirectory in the NAS file system. The subdirectory must start with a forward slash (/). If this parameter is set, the PV will be mounted to the specified subdirectory. ▪ If the specified subdirectory does not exist, the system automatically creates the subdirectory in the NAS file system and mounts the subdirectory to the cluster. ▪ If you do not set this parameter, the root directory of the NAS file system is mounted. ▪ If you specify a subdirectory of an Extreme NAS file system, the subdirectory must start with */share*.

Parameter	Description
Permissions	The access permissions on the mounted directory. For example, you can set this parameter to 755, 644, or 777. **Note** ▪ You can set access permissions only on subdirectories. ▪ If the mounted directory stores a large number of files, we recommend that you do not set this parameter. Otherwise, the process of running the chmod command may require an excessive amount of time. If the mounted directory is a subdirectory, this parameter is optional. ▪ If you do not set this parameter, the original permissions on the mounted directory are used. ▪ Take note of the following items when you set the permissions: ▪ For FlexVolume versions earlier than V1.14.6.15-8d3b7e7-aliyun, use the recursive mode when you configure permission settings. The permissions on all files and directories under the mounted directory will be modified. ▪ For FlexVolume V1.14.6.15-8d3b7e7-aliyun and later, set the **chmod (Change Mode)** parameter to configure permission settings.
chmod (Change Mode)	The change mode of access permissions. Valid values: Non-recursive and Recursive. ▪ Non-recursive: The permission changes apply only to the mounted directory. The subdirectories and files in the mounted directory are not affected. ▪ Recursive mode: The permission changes apply to the mounted directory, and the subdirectories and files in the mounted directory. **Note** If you select the recursive mode for a mounted directory that contains a large number of files, the process of running the chmod command may require an excessive amount of time. The mount or unmount operation may fail. Exercise caution when you set this parameter.
Version	The version of the NFS protocol. We recommend that you use NFSv3. Extreme NAS file systems support only NFSv3.

Parameter	Description
Labels	Add labels to the PV.

g. Click **Create**.

2. Create a PVC.

   Use the following *nas-pvc.yaml* file to create a PVC:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-nas
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: nas
  resources:
    requests:
      storage: 5Gi
```

3. Create a pod.

   Use the following *nas-pod.yaml* file to create a pod:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nas-static
  labels:
    app: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx
        ports:
        - containerPort: 80
        volumeMounts:
          - name: pvc-nas
            mountPath: /data
      volumes:
        - name: pvc-nas
          persistentVolumeClaim:
            claimName: pvc-nas
```

12.3. Create a PVC

Prerequisites

- A serverless Kubernetes (ASK) cluster is created. For more information, see ASK quick start.
-

Procedure

1.
2.
3.
4.
5.
6.
7.

12.4. Use a PVC

In the Container Service for Kubernetes (ACK) console, you can create an application from an image or a template. When you create the application, you can specify a persistent volume claim (PVC) that the application uses to request physical storage. In this example, an application is created from an image. You can also create an application from a template and specify a PVC in the template. For more information, see Disk volume overview.

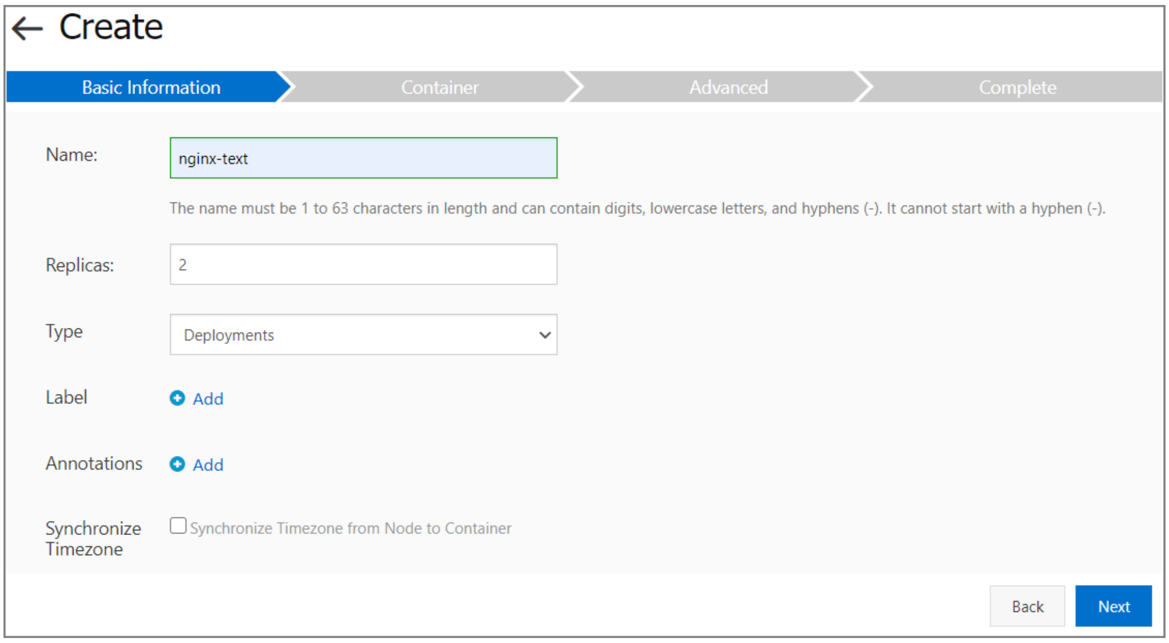
Prerequisites

- A serverless Kubernetes (ASK) cluster is created. For more information, see ASK quick start.
- A PVC is created. In this example, a PVC named pvc-disk is created based on cloud storage. For more information, see Create a PVC.

Procedure

1.
2.
3.
4.
5.
6. On the **Basic Information** wizard page, configure the basic settings.

   For more information, see Configure basic settings.



Name: nginx-text

The name must be 1 to 63 characters in length and can contain digits, lowercase letters, and hyphens (-). It cannot start with a hyphen (-).

Replicas: 2

Type: Deployments

Label: Add

Annotations: Add

Synchronize Timezone: ☐ Synchronize Timezone from Node to Container

7. On the **Container** wizard page, select an image and configure a data volume based on cloud storage. Cloud disks, NAS file systems, and Object Storage Service (OSS) buckets can be specified as cloud storage. In this example, select the pvc-disk PVC and click **Next**. For more information, see Configure the containers.
8. On the **Advanced** wizard page, create a service for the test-nginx application and click **Create**.
9. After the application is created, you are redirected to the **Creation Task Submitted** page. You can click **View Details** to view application details.

   The Basic Information page of the newly created test-nginx application appears by default.
10. On the **Pods** tab, find the pod to which the application belongs and click **View Details**.
11. You are redirected to the Overview page of the pod. Click the **Volumes** tab. Verify that the pod is associated with the pvc-disk PVC.

12.5. FAQ about FlexVolume

This topic provides answers to some frequently asked questions about disk volumes, Apsara File Storage NAS (NAS) volumes, and Object Storage Service (OSS) volumes.

Type	Question
FAQ about storage	• What do I do when a volume fails to be mounted? • How do I view the log related to storage services? • What do I do if I find kubelet contains a pod log that is not managed by ACK?

Type	Question
FAQ about disk volumes	• Why does a timeout error occur when I mount a disk to a node? • Why does a zone error occur when I mount a disk to an Elastic Compute Service (ECS) instance? • Why does an input/output error occur in a disk after a system upgrade? • Why does the system prompt The specified disk is not a portable disk when I unmount a disk? • Why does the system prompt had volume node affinity conflict when I launch a pod that has a disk mounted? • Why does the system prompt can't find disk when I launch a pod that has a disk mounted? • Why does the system prompt disk size is not supported when I dynamically provision a PV?
FAQ about NAS volumes	• Why does it require a long time to mount a NAS volume? • Why does a timeout error occur when I mount a NAS volume? • Why does the system prompt chown: option not permitted when I mount a NAS volume? • What do I do if I fail to mount a NAS volume? • What do I do if the task queue of alicloud-nas-controller is full and PVs cannot be created when I use a dynamically provisioned NAS volume?
FAQ about OSS volumes	• Why do I fail to mount an OSS volume? • Why does the directory in an OSS volume that is mounted to a container become unavailable after the cluster that runs the pod is upgraded? • Why does it require a long time to mount an OSS volume?

What do I do when a volume fails to be mounted?

Check whether FlexVolume and the plug-in for dynamic volume provisioning are installed.

Method 1: Check whether FlexVolume is installed

Run the following command to query information about pods:

```
kubectl get pod -n kube-system | grep flexvolume
```

Expected output:

```
flexvolume-4wh8s              1/1       Running    0          8d
flexvolume-65z49              1/1       Running    0          8d
flexvolume-bpc6s              1/1       Running    0          8d
flexvolume-l8pml              1/1       Running    0          8d
flexvolume-mzkpv              1/1       Running    0          8d
flexvolume-wbfhv              1/1       Running    0          8d
flexvolume-xf5cs              1/1       Running    0          8d
```

Check whether the FlexVolume pods are in the Running state. Check whether the number of running FlexVolume pods equals the number of nodes in the cluster.

If the FlexVolume pods are not in the Running state, troubleshoot the issue based on the log of the FlexVolume plug-in.

Method 2: Check whether the plug-in for dynamic volume provisioning is installed

To mount a dynamically provisioned disk volume, you must install the plug-in for dynamic volume provisioning. Run the following command to query the pod on which the plug-in runs:

```
kubectl get pod -n kube-system | grep alicloud-disk
```

Expected output:

```
alicloud-disk-controller-8679c9fc76-lq6zb     1/1 Running   0   7d
```

If the pod is not in the Running state, troubleshoot the issue based on the log of the plug-in.

How do I view the log related to storage services?

You can view the logs of FlexVolume, the provisioner plug-in, and kubelet.

Method 1: Print the log of FlexVolume on Master Node 1

Run the following command to query the pod on which an error occurs:

```
kubectl get pod -n kube-system | grep flexvolume
```

Run the following commands in sequence to print the log of the pod on which the error occurs:

```
kubectl logs flexvolume-4wh8s -n kube-system
kubectl describe pod flexvolume-4wh8s -n kube-system
```

Note The last several entries of the returned pod information indicate the status of the pod. You can analyze the error based on the log data.

View the logs of the disk driver, NAS driver, and OSS driver.

Perform the following steps to view the logs on the host node. If no PV is mounted to a pod, query the IP address of the node that hosts the pod.

```
kubectl describe pod nginx-97dc96f7b-xbx8t | grep Node
```

Expected output:

```
Node: cn-hangzhou.i-bp19myla3uvnt6zi****/192.168.XX.XX
Node-Selectors:  <none>
```

Log on to the node and print the logs of the disk driver, NAS driver, and OSS driver.

```
ssh 192.168.XX.XX
ls /var/log/alicloud/flexvolume*
```

Expected output:

```
flexvolume_disk.log  flexvolume_nas.log  flexvolume_o#ss.log
```

Method 2: Print the log of the provisioner plug-in on Master Node 1

Run the following command to query the pod on which an error occurs:

```
kubectl get pod -n kube-system | grep alicloud-disk
```

Run the following commands in sequence to print the log of the pod on which the error occurs:

```
kubectl logs alicloud-disk-controller-8679c9fc76-lq6zb -n kube-system
kubectl describe pod alicloud-disk-controller-8679c9fc76-lq6zb -n kube-system
```

Note The last several entries of the returned pod information indicate the status of the pod. You can analyze the error based on the log data.

Method 3: Print the log of kubelet

If no PV is mounted to a pod, query the IP address of the node that hosts the pod.

```
kubectl describe pod nginx-97dc96f7b-xbx8t | grep Node
```

Expected output:

```
Node: cn-hangzhou.i-bp19myla3uvnt6zi****/192.168.XX.XX
Node-Selectors:  <none>
```

Log on to the node and print the log of kubelet.

```
ssh 192.168.XX.XX
journalctl -u kubelet -r -n 1000 &> kubelet.log
```

Note In the preceding command, -n specifies the number of log entries to return.

The preceding steps describe how to print the logs of the FlexVolume, provisioner, and kubelet plug-ins. If the problem persists, contact the technical support team of Alibaba Cloud and provide related log information.

What do I do if I find kubelet contains a pod log that is not managed by ACK?

When a pod exceptionally exits, the mount target is not removed when the system unmounts the PV. As a result, the system fails to delete the pod. kubelet cannot collect all volume garbage. You must manually remove invalid mount targets or run a script to automate garbage collection.

Run the following script on the failed node to remove invalid mount targets:

```
wget https://raw.githubusercontent.com/AliyunContainerService/kubernetes-issues-solution/ma
ster/kubelet/kubelet.sh
sh kubelet.sh
```

13.Network management

13.1. Service Management

13.1.1. Considerations for configuring a LoadBalancer type Service

When you specify `Type=LoadBalancer` for a Service, Cloud Controller Manager (CCM) creates and configures Server Load Balancer (SLB) resources for the Service, including SLB instances, listeners, and VServer groups. This topic describes the considerations for configuring a LoadBalancer type Service and the policies that are used by CCM to update SLB resources.

Policies that are used by CCM to update SLB resources

Container Service for Kubernetes (ACK) allows you to specify an existing SLB instance for a Service. You can also use CCM to automatically create one for the Service. The two methods use different policies to update SLB resources. The following table describes the differences.

Resource object	Existing SLB instance	SLB instance created and managed by CCM
SLB	Use the following annotation to specify an existing SLB instance for a Service: `service.beta.kubernetes.io/alibaba-cloud-loadbalancer-id` • CCM uses the specified SLB instance to enable load balancing. You can use other annotations to configure the SLB instance. CCM automatically creates VServer groups for the instance. • When the Service is deleted, CCM does not delete the existing SLB instance that is specified in the annotation.	• CCM automatically creates, configures, and manages SLB resources based on the Service configuration, including the SLB instance, listeners, and VServer groups. • When the Service is deleted, CCM deletes the created SLB instance.
Listener	Use the following annotation to configure listeners: `service.beta.kubernetes.io/alibaba-cloud-loadbalancer-force-override-listeners:` . • If you set the annotation to false, CCM does not configure or manage listeners for the SLB instance. • If you set the annotation to true, CCM configures and manages listeners for the SLB instance based on the Service configuration. If the SLB instance has existing listeners, CCM creates new listeners to replace the existing ones.	CCM configures listeners for the SLB instance based on the Service configuration.

Resource object	Existing SLB instance	SLB instance created and managed by CCM
VServer group	When the endpoint of an Elastic Compute Service (ECS) instance in a VServer group for a Service changes or the cluster nodes are changed, CCM updates the VServer groups. • The policies for updating VServer groups vary based on the mode of the Service. ◦ If `spec.externalTrafficPolicy = Cluster` is specified for a Service, CCM adds all cluster nodes to the VServer groups of the SLB instance. If node labels are specified in the Service configuration, CCM adds cluster nodes that have the specified labels to the VServer groups of the SLB instance. **Notice** SLB limits the number of VServer groups to which an ECS instance can be added. If a Service is in Cluster mode, the quota is consumed at a high rate. When the quota is used up, Service reconciliation fails. To fix this issue, set Local mode for a Service. ◦ If `spec.externalTrafficPolicy = Local` is specified for a Service, CCM adds only the nodes where the pods that are related to the Service are deployed to the VServer groups of the SLB instance. This can reduce the consumption rate of the resources. Source IP addresses can also be retained in Layer 4 load balancing. • CCM does not add master nodes of a cluster to the VServer groups of an SLB instance. • Assume that you have run the `kubectl drain` command to remove a node from a cluster, or run the `kubectl cordon` command to mark a node as unschedulable. By default, CCM does not remove such a node from VServer groups of an SLB instance. To remove such a node, set annotation `service.beta.kubernetes.io/alibaba-cloud-loadbalancer-remove-unscheduled-backend` to `on`. **Notice** By default, CCM versions earlier than V1.9.3.164-g2105d2e-aliyun remove drained nodes or unschedulable nodes from the backend of the SLB instance.	

Considerations for reusing an existing SLB instance

- **Before you reuse an existing SLB instance, check whether the instance meets the following requirements:**
  - The SLB instance that you want to reuse is created in the SLB console. You cannot reuse an SLB instance that is created by CCM.
  - If you want to reuse an internal-facing SLB instance, the SLB instance and the cluster must be deployed in the same virtual private cloud (VPC).
- CCM configures SLB instances only for `LoadBalancer` type Services.

  Notice If you change `Type=LoadBalancer` to `Type!=LoadBalancer` for a Service, CCM deletes the configuration of the SLB instance from the Service. In this case, you cannot access the Service by using the SLB instance.

- When specific conditions are met, CCM uses a declarative API to automatically update the configuration of an SLB instance based on the Service configuration. If you set annotation `service.beta.kubernetes.io/alibaba-cloud-loadbalancer-force-override-listeners:` to `true` for a Service, CCM may overwrite the listener configuration modifications that are made in the SLB console.

> **Notice** If the SLB instance is created or managed by CCM, we recommend that you do not modify the configuration of an SLB instance in the SLB console. Otherwise, CCM may overwrite the configuration and the Service may be unavailable.

Considerations for using CCM to configure an SLB instance

- CCM configures SLB instances only for `LoadBalancer` type Services.

  > **Notice** If you change `Type=LoadBalancer` to `Type!=LoadBalancer` for a Service, CCM deletes the SLB instance that CCM previously created for the Service.

- When specific conditions are met, CCM uses a declarative API to automatically update the configuration of an SLB instance based on the Service configuration. CCM may overwrite the listener configuration modifications that are made in the SLB console.

  > **Notice** If the SLB instance is created or managed by CCM, we recommend that you do not modify the configuration of an SLB instance in the SLB console. Otherwise, CCM may overwrite the configuration and the Service may be unavailable.

Resource quotas

VPC

- A node in a cluster is mapped to a route entry in a route table. By default, each route table for a VPC contains a maximum of 48 entries. If the number of nodes in a cluster exceeds 48, .

  > **Note** In the ticket, describe your request to modify parameter `vpc_quota_route_entrys_num`. After the request is accepted, the maximum number of custom entries that can be created in one route table is increased.

- For more information about VPC resource quotas, see 限制与配额.

  To query the VPC resource quotas, go to the Quota Management page in the VPC console.

SLB

- CCM creates an SLB instance for a `LoadBalancer` type Service. By default, you can retain a maximum of 60 SLB instances under each account. To create more SLB instances, .

  > **Note** In the ticket, describe your request to modify parameter `slb_quota_instances_num`. After the request is accepted, the maximum number of SLB instances that can be retained under your account is increased.

- CCM adds ECS instances to the VServer groups of an SLB instance based on the Service configuration.

- By default, an ECS instance can be added to a maximum of 50 VServer groups. To add the ECS instance to more VServer groups, .

  Note In the ticket, describe your request to modify parameter `slb_quota_backendservers_num`. After the request is accepted, the maximum number of VServer groups to which an ECS instance can be added is increased.

- By default, a maximum of 200 backend servers can be added to an SLB instance. To add more backend servers, submit a ticket.

  Note In the ticket, describe your request to modify parameter `slb_quota_backendservers_num`. After the request is accepted, the maximum number of backend servers that can be added to an SLB instance is increased.

- CCM creates listeners based on the ports that are specified for a Service. By default, a maximum of 50 listeners can be created for an SLB instance. To create more listeners, submit a ticket.

  Note In the ticket, describe your request to modify parameter `slb_quota_listeners_num`. After the request is accepted, the maximum number of listeners that can be created for an SLB instance is increased.

- For more information about SLB resource quotas, see Limits.

  To query the SLB resource quotas, go to the Quota Management page in the SLB console.

13.1.2. Use annotations to configure load balancing

You can add annotations to the YAML file of a Service to configure load balancing. This topic describes how to use annotations to configure load balancing. You can configure Server Load Balancer (SLB) instance, listeners, and backend server groups.

Usage notes

- The values of annotations are case-sensitive.
- Starting from September 11, 2019, the keyword `alicloud` in `annotations` is changed to `alibaba-cloud`.

  Example:

  Before the change: `service.beta.kubernetes.io/alicloud-loadbalancer-id`

  After the change: `service.beta.kubernetes.io/alibaba-cloud-loadbalancer-id`

  The system still supports annotations that use the `alicloud` keyword. You do not need to modify the existing annotations.

SLB

Common annotations that are used to configure load balancing

- Create an Internet-facing SLB instance

```
apiVersion: v1
kind: Service
metadata:
  name: nginx
  namespace: default
spec:
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
    run: nginx
  type: LoadBalancer
```

- Create an internal-facing SLB instance

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-address-type: "intranet"
  name: nginx
  namespace: default
spec:
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
    run: nginx
  type: LoadBalancer
```

- Create an SLB instance of a specified specification

  For more information about the specifications of SLB instances, see CreateLoadBalancer. Use the following template to create a LoadBalancer Service and an SLB instance of a specified specification. You can also use the template to update the specification of an existing SLB instance.

  Notice If you modify the specification of the SLB instance in the SLB console, the modification may be restored by the cloud controller manager (CCM). Proceed with caution.

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-spec: "slb.s1.small"
  name: nginx
  namespace: default
spec:
  ports:
  - port: 443
    protocol: TCP
    targetPort: 443
  selector:
    run: nginx
  type: LoadBalancer
```

- Use an existing SLB instance
  - By default, the CCM does not overwrite the listeners of an existing SLB instance. To overwrite the listeners of an existing SLB instance, set the annotation `service.beta.kubernetes.io/alibaba-cloud-loadbalancer-force-override-listeners` to *true*.

    Note The following list explains why the CCM does not overwrite the listeners of an existing SLB instance:

    - If the listeners of the SLB instance are associated with applications, service interruptions may occur after the configurations of the listeners are overwritten.
    - The CCM supports limited backend configurations and cannot handle complex configurations. If you require complex backend configurations, you can manually configure listeners in the SLB console without overwriting the existing listeners.

    In both cases, we recommend that you do not overwrite the listeners of existing SLB instances. However, you can overwrite an existing listener if the port of the listener is no longer in use.

  - You cannot add tags to an existing SLB instance by using the annotation `service.beta.kubernetes.io/alibaba-cloud-loadbalancer-additional-resource-tags`.

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-id: "${YOUR_LOADBALACER_ID}"
  name: nginx
  namespace: default
spec:
  ports:
  - port: 443
    protocol: TCP
    targetPort: 443
  selector:
    run: nginx
  type: LoadBalancer
```

- Use an existing SLB instance and forcefully overwrite the listeners of the SLB instance

Use the following template to create a LoadBalancer Service and forcefully overwrite an existing listener of the SLB instance. If you specify a different listener port number, the original listener is deleted.

Notice If you use an existing SLB instance and set `forceoverride` to `true`, do not associate a listener of the SLB instance with multiple Services. Otherwise, a configuration conflict occurs when these Services overwrite the configuration of the listener.

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-id: "${YOUR_LOADBALACER_ID}"
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-force-override-listeners: "true
"
  name: nginx
  namespace: default
spec:
  ports:
  - port: 443
    protocol: TCP
    targetPort: 443
  selector:
    run: nginx
  type: LoadBalancer
```

- Specify the primary zone and secondary zone when you create an SLB instance
  - SLB instances cannot be deployed across zones in some regions, such as the Indonesia (Jakarta) region.
  - After you specify the primary zone and secondary zone for an SLB instance, the zones cannot be changed.

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-master-zoneid: "ap-southeast-5a
"
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-slave-zoneid: "ap-southeast-5a"
  name: nginx
  namespace: default
spec:
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
    run: nginx
  type: LoadBalancer
```

- Create a pay-by-bandwidth SLB instance

- Only Internet-facing SLB instances support the pay-by-bandwidth billing method.

  For more information about limits on billing methods for Internet-facing SLB instances, see Change the billing method of an Internet-facing SLB instance.

- All annotations in the following template are required.

  Modify the annotation values based on your business requirements.

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-charge-type: "paybybandwidth"
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-bandwidth: "2"
  name: nginx
  namespace: default
spec:
  ports:
  - port: 443
    protocol: TCP
    targetPort: 443
  selector:
    run: nginx
  type: LoadBalancer
```

Note The annotation service.beta.kubernetes.io/alibaba-cloud-loadbalancer-bandwidth specifies the bandwidth limit.

- Specify a vSwitch for an SLB instance
  - Obtain the ID of a vSwitch in the Virtual Private Cloud (VPC) console. Then, use the annotations in the following template to specify the vSwitch for an SLB instance.
  - The specified vSwitch and the cluster must be deployed in the same VPC.
  - All annotations in the following template are required.

```
apiVersion: v1
kind: Service
metadata:
  annotations:
   service.beta.kubernetes.io/alibaba-cloud-loadbalancer-address-type: "intranet"
   service.beta.kubernetes.io/alibaba-cloud-loadbalancer-vswitch-id: "${YOUR_VSWITCH_ID}"
  name: nginx
  namespace: default
spec:
  ports:
  - port: 443
    protocol: TCP
    targetPort: 443
  selector:
    run: nginx
  type: LoadBalancer
```

- Add tags to an SLB instance

Separate multiple tags with commas (,). Example: `"k1=v1,k2=v2"` .

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-additional-resource-tags: "Key1
=Value1,Key2=Value2"
  name: nginx
  namespace: default
spec:
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
    run: nginx
  type: LoadBalancer
```

- Create an IPv6 SLB instance
  - The kube-proxy mode must be set to IPVS.
  - The assigned IPv6 address can be used only in an IPv6 network.
  - You cannot change the IP address type after you create the SLB instance.

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-ip-version: "ipv6"
  name: nginx
spec:
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: nginx
  type: LoadBalancer
```

- Enable deletion protection for an SLB instance

  By default, deletion protection is enabled for SLB instances.

> **Notice** If you manually enable deletion protection in the SLB console for an SLB instance that is created for a LoadBalancer Service, you can run the `kubectl delete svc {your-svc-name}` command to delete the SLB instance.

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-delete-protection: "on"
  name: nginx
spec:
  externalTrafficPolicy: Local
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: nginx
  type: LoadBalancer
```

- Enable the configuration read-only mode for an SLB instance

  By default, the configuration read-only mode is enabled for an SLB instance.

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-modification-protection: "Conso
leProtection"
  name: nginx
spec:
  externalTrafficPolicy: Local
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: nginx
  type: LoadBalancer
```

- Specify the name of an SLB instance

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-name: "your-svc-name"
  name: nginx
spec:
  externalTrafficPolicy: Local
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: nginx
  type: LoadBalancer
```

- Specify a resource group to which an SLB instance belongs

  Log on to the Resource Management console to obtain the ID of a resource group. Then, use the annotation in the following template to specify the resource group to which the SLB instance belongs.

  > **Note** You cannot modify the resource group after the SLB instance is created.

  ```
  apiVersion: v1
  kind: Service
  metadata:
    annotations:
      service.beta.kubernetes.io/alibaba-cloud-loadbalancer-resource-group-id: "rg-xxxx"
    name: nginx
  spec:
    externalTrafficPolicy: Local
    ports:
    - port: 80
      protocol: TCP
      targetPort: 80
    selector:
      app: nginx
    type: LoadBalancer
  ```

- Configure a hostname for the Service
  - `your_service_name` must comply with the naming conventions of domain names.
  - After you add the annotation, the external IP address of the Service is changed from the default SLB IP address to `your_service_name`. If you access the SLB IP address from within the cluster, the request is first forwarded to the corresponding SLB instance.
  - After you add the annotation, if the listener protocol is TCP or UDP and you access the SLB IP address from within the cluster, a loopback issue occurs. For more information, see Why am I unable to access an SLB instance?
  - This annotation does not automatically associate a domain name with the SLB instance. If you want to associate a domain name with the SLB instance, log on to the Domains console, purchase a domain name, and then associate the domain name with the SLB instance. For more information, see Purchase a domain name.

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-protocol-port: "http:80"
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-hostname: "${your_service_hostn
ame}"
  name: nginx-svc
  namespace: default
spec:
  ports:
  - name: http
    port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: nginx
  type: LoadBalancer
```

Expected output:

```
NAME         TYPE           CLUSTER-IP       EXTERNAL-IP           PORT(S)
AGE
nginx-svc    loadBalancer   47.100.XX.XX     www.example.com       80:30248/TCP,443:3267
0/TCP    10s
```

Listeners

Common annotations that are used to configure listeners

- Set the session persistence period for a TCP-based SLB instance
    - The annotation `service.beta.kubernetes.io/alibaba-cloud-loadbalancer-persistence-timeout` applies only to TCP listeners.
    - If an SLB instance has multiple TCP listeners, the configuration applies to all the TCP listeners.

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-persistence-timeout: "1800"
  name: nginx
  namespace: default
spec:
  ports:
  - port: 443
    protocol: TCP
    targetPort: 443
  selector:
    run: nginx
  type: LoadBalancer
```

- Enable session persistence for HTTP or HTTPS listeners by inserting a cookie

- The annotations in the following template apply only to HTTP-based and HTTPS-based SLB instances.
- If an SLB instance has multiple HTTP or HTTPS listeners, the configuration applies to all the HTTP or HTTPS listeners.
- To configure session persistence by inserting a cookie, all annotations in the following template are required.

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-sticky-session: "on"
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-sticky-session-type: "insert"
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-cookie-timeout: "1800"
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-protocol-port: "http:80"
  name: nginx
  namespace: default
spec:
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
    run: nginx
  type: LoadBalancer
```

- Enable access control for an SLB instance
  - Create an access control list (ACL) in the SLB console and record the ACL ID. Then, use the annotations in the following template to enable access control for an SLB instance.
  - A whitelist allows access only from specified IP addresses. A blacklist blocks access only from specific IP addresses.
  - All annotations in the following template are required.

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-acl-status: "on"
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-acl-id: "${YOUR_ACL_ID}"
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-acl-type: "white"
  name: nginx
  namespace: default
spec:
  ports:
  - port: 443
    protocol: TCP
    targetPort: 443
  selector:
    run: nginx
  type: LoadBalancer
```

- Configure port forwarding for an SLB instance

- Port forwarding allows an SLB instance to forward requests from an HTTP port to an HTTPS port.
- Create a certificate in the SLB console and record the certificate ID. Then, use the following annotations to configure port forwarding for an SLB instance.
- All annotations in the following template are required.

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-protocol-port: "https:443,http:
80"
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-cert-id: "${YOUR_CERT_ID}"
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-forward-port: "80:443"
  name: nginx
  namespace: default
spec:
  ports:
  - name: https
    port: 443
    protocol: TCP
    targetPort: 443
  - name: http
    port: 80
    protocol: TCP
    targetPort: 80
  selector:
    run: nginx
  type: LoadBalancer
```

- Set the scheduling algorithm for an SLB instance
  - rr: Requests are distributed to backend servers in sequence. This is the default scheduling algorithm.
  - wrr: Backend servers with higher weights receive more requests than backend servers with lower weights.

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-scheduler: "wrr"
  name: nginx
  namespace: default
spec:
  ports:
  - port: 443
    protocol: TCP
    targetPort: 443
  selector:
    run: nginx
  type: LoadBalancer
```

- Create a UDP listener

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-protocol-port: "udp:80"
  name: nginx
  namespace: default
spec:
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
    run: nginx
  type: LoadBalancer
```

- Create an HTTP listener

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-protocol-port: "http:80"
  name: nginx
  namespace: default
spec:
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
    run: nginx
  type: LoadBalancer
```

- Create an HTTPS listener

  You must first create a certificate in the SLB console. Then, you can use the certificate ID and the following template to create a LoadBalancer Service and an HTTPS-based SLB instance.

  Note HTTPS listeners of the SLB instance decrypt HTTPS requests into HTTP requests and forward the requests to pods on the backend servers.

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-protocol-port: "https:443"
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-cert-id: "${YOUR_CERT_ID}"
  name: nginx
  namespace: default
spec:
  ports:
  - port: 443
    protocol: TCP
    targetPort: 80
  selector:
    run: nginx
  type: LoadBalancer
```

- Create a listener that has health checks enabled
  - Enable TCP health checks
    - By default, the health check feature is enabled for TCP listeners and cannot be disabled. The annotation `service.beta.kubernetes.io/alibaba-cloud-loadbalancer-health-check-flag` does not take effect.
    - To enable TCP health checks, all annotations in the following template are required.

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-health-check-type: "tcp"
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-health-check-connect-timeout:
"8"
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-healthy-threshold: "4"
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-unhealthy-threshold: "4"
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-health-check-interval: "3"
  name: nginx
  namespace: default
spec:
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
    run: nginx
  type: LoadBalancer
```

- Enable UDP health checks
  - By default, the health check feature is enabled for UDP listeners and cannot be disabled. The annotation `service.beta.kubernetes.io/alibaba-cloud-loadbalancer-health-check-flag` does not take effect.
  - To enable UDP health checks, all annotations in the following template are required.

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-protocol-port: "udp:80"
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-health-check-interval: "5"
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-health-check-connect-timeout:
"10"
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-healthy-threshold: "3"
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-unhealthy-threshold: "3"
  name: nginx
  namespace: default
spec:
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
    run: nginx
  type: LoadBalancer
```

- Enable HTTP health checks

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-health-check-flag: "on"
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-health-check-type: "http"
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-health-check-uri: "/test/inde
x.html"
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-healthy-threshold: "4"
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-unhealthy-threshold: "4"
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-health-check-timeout: "10"
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-health-check-interval: "3"
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-protocol-port: "http:80"
    # Specify the health check method. This annotation is optional.
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-health-check-method: "head"
  name: nginx
  namespace: default
spec:
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
    run: nginx
  type: LoadBalancer
```

- Enable connection draining for a listener.

  Only TCP and UDP are supported. To enable connection draining for a listener, all annotations in the following template are required.

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-connection-drain: "on"
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-connection-drain-timeout: "30"
  name: nginx
  namespace: default
spec:
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
    run: nginx
  type: LoadBalancer
```

- Use the X-Forwarded-Proto header to identify the listener protocol of an SLB instance.

  Only HTTP and HTTPS are supported.

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-protocol-port: "http:80"
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-xforwardedfor-proto: "on"
  name: nginx
  namespace: default
spec:
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
    run: nginx
  type: LoadBalancer
```

- Specify the timeout period of idle connections for a listener

  Only HTTP and HTTPS are supported.

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-protocol-port: "http:80"
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-idle-timeout: "30"
  name: nginx
  namespace: default
spec:
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
    run: nginx
  type: LoadBalancer
```

- Disable HTTP/2 for a listener.

  Only HTTPS is supported.

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-protocol-port: "https:443"
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-cert-id: "${YOUR_CERT_ID}"
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-http2-enabled: "off"
  name: nginx
  namespace: default
spec:
  ports:
  - port: 443
    protocol: TCP
    targetPort: 80
  selector:
    run: nginx
  type: LoadBalancer
```

- Specify the timeout period of requests for a listener.

  Only HTTP and HTTPS are supported.

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-protocol-port: "http:80"
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-request-timeout: "60"
  name: nginx
  namespace: default
spec:
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
    run: nginx
  type: LoadBalancer
```

- Specify the timeout period of connections for a listener.

  Only TCP is supported.

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-established-timeout: "60"
  name: nginx
  namespace: default
spec:
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
    run: nginx
  type: LoadBalancer
```

Backend server groups

Common annotations that are used to configure backend servers

- Add worker nodes that have specified labels as the backend servers of an SLB instance

  Separate multiple labels with commas (,). Example: `"k1=v1,k2=v2"`. A node must have all the specified labels before you can add the node as a backend server.

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-backend-label: "failure-domain.
beta.kubernetes.io/zone=ap-southeast-5a"
  name: nginx
  namespace: default
spec:
  ports:
  - port: 443
    protocol: TCP
    targetPort: 443
  selector:
    run: nginx
  type: LoadBalancer
```

- Add the nodes where pods that run the Service are deployed as the backend servers of an SLB instance
  - By default, `externalTrafficPolicy` is set to Cluster for a Service. In Cluster mode, all nodes in the cluster are added as the backend servers of the SLB instance. In Local mode, only nodes where the pods are deployed are added as the backend servers of the SLB instance.

- In Local mode, you must set the scheduling algorithm to weighted round-robin (WRR).

  Note

  For CCM 1.9.3.164-g2105d2e-aliyun and later, node weights are calculated based on the number of pods that run on each node for Services whose externalTrafficPolicy is set to **Local**. For more information about node weight calculation, see How does the CCM calculate node weights in Local mode?.

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-scheduler: "wrr"
  name: nginx
  namespace: default
spec:
  externalTrafficPolicy: Local
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
    run: nginx
  type: LoadBalancer
```

- Remove backend servers in the **Unschedulable** state from an SLB instance
  - You can run the `kubectl cordon` and `kubectl drain` commands to set a node to the **Unschedulable** state. By default, the annotation `service.beta.kubernetes.io/alibaba-cloud-loadbalancer-remove-unscheduled-backend` is set to off. In this case, nodes in the **Unschedulable** state are not removed from the backend server groups of an SLB instance.
  - To remove backend servers in the **Unschedulable** state from an SLB instance, set the annotation `service.beta.kubernetes.io/alibaba-cloud-loadbalancer-remove-unscheduled-backend` to on.

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-remove-unscheduled-backend: "on
"
  name: nginx
spec:
  externalTrafficPolicy: Local
  ports:
  - name: http
    port: 30080
    protocol: TCP
    targetPort: 80
  selector:
    app: nginx
  type: LoadBalancer
```

- Add pods that are assigned elastic network interfaces (ENIs) as the backend servers of an SLB instance

  When the Terway network plug-in is used, you can use the annotation `service.beta.kubernetes.io/backend-type: "eni"` to add pods that are assigned ENIs as the backend servers of an SLB instance. This improves network forwarding performance.

  ```
  apiVersion: v1
  kind: Service
  metadata:
    annotations:
      service.beta.kubernetes.io/backend-type: "eni"
    name: nginx
  spec:
    ports:
    - name: http
      port: 30080
      protocol: TCP
      targetPort: 80
    selector:
      app: nginx
    type: LoadBalancer
  ```

  Note You can also change `eni` in `service.beta.kubernetes.io/backend-type: "eni"` to `ecs`. This allows you to add Elastic Compute Service (ECS) instances as the backend severs of an SLB instance.

- Reuse an existing vServer group

  The annotation `service.beta.kubernetes.io/alibaba-cloud-loadbalancer-vgroup-port` can be used to reuse an existing vServer group that is added to an SLB instance. This annotation takes effect only when the SLB instance is reused. For more information, see Use the CCM to deploy services across clusters.

- Set weights for Services to enable weighted round robin

  When a reused SLB instance is shared among multiple Services, the annotation `service.beta.kubernetes.io/alibaba-cloud-loadbalancer-weight` can be used to set the weight of each Service to enable weighted round robin. This annotation takes effect only when an existing vServer group is reused. For more information, see Use the CCM to deploy services across clusters.

Common annotations

- **Common annotations that are used to configure SLB instances**

Annotation	Type	Description	Default	Supported CCM version

Annotation	Type	Description	Default	Supported CCM version
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-address-type	string	The type of SLB instance. Valid values: *internet* and *intranet*. ◦ *internet*: accesses the Service over the Internet. This is the default value. The **address type** of the SLB instance must be set to **Internet**. ◦ *intranet*: accesses the Service over an internal network. The **address type** of the SLB instance must be set to **intranet**.	*internet*	CCM 1.9.3 and later
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-charge-type	string	The billing method of an SLB instance. Valid values: *paybytraffic* and *paybybandwidth*.	*paybytraffic*	CCM 1.9.3 and later
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-id	string	The ID of the SLB instance. You can specify an existing SLB instance by using the annotation service.beta.kubernetes.io/alibaba-cloud-loadbalancer-id. By default, if you specify an existing SLB instance, the CCM does not overwrite the listeners of the SLB instance. To overwrite the listeners, set the annotation service.beta.kubernetes.io/alibaba-cloud-loadbalancer-force-override-listeners to *true*.	None	CCM 1.9.3.81-gca19cd4-aliyun and later
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-spec	string	The specification of the SLB instance. For more information, see CreateLoadBalancer.	slb.s1.small	CCM 1.9.3 and later
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-master-zoneid	string	The ID of the zone for the primary backend server.	None	CCM 1.9.3.10-gfb99107-aliyun and later

Annotation	Type	Description	Default	Supported CCM version
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-slave-zoneid	string	The ID of the zone for the secondary backend servers.	None	CCM 1.9.3.10-gfb99107-aliyun and later
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-force-override-listeners	string	Specifies whether to overwrite the listeners of an existing SLB instance that is specified for a Service.	*false*: does not overwrite the listeners of the existing SLB instance.	CCM 1.9.3.81-gca19cd4-aliyun and later
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-bandwidth	string	The bandwidth of an SLB instance. This annotation applies only to Internet-facing SLB instances.	*50*	CCM 1.9.3.10-gfb99107-aliyun and later
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-vswitch-id	string	The ID of the vSwitch to which an SLB instance belongs. To set this annotation, you must set the annotation service.beta.kubernetes.io/alibaba-cloud-loadbalancer-address-type to intranet.	None	CCM 1.9.3 and later
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-additional-resource-tags	string	The tags that you want to add to an SLB instance. Separate multiple tags with commas (,). Example: `"k1=v1,k2=v2"` .	None	CCM 1.9.3 and later
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-ip-version	string	The IP protocol of an SLB instance. Valid values: ipv4 and ipv6.	ipv4	CCM 1.9.3.220-g24b1885-aliyun and later
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-delete-protection	string	Specifies whether to enable deletion protection for an SLB instance. Valid values: on and off.	on	CCM 1.9.3.313-g748f81e-aliyun and later

Annotation	Type	Description	Default	Supported CCM version
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-modification-protection	string	Specifies whether to enable the configuration read-only mode for an SLB instance. Valid values: ConsoleProtection and NonProtection.	ConsoleProtection	CCM 1.9.3.313-g748f81e-aliyun and later
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-resource-group-id	string	The ID of the resource group to which an SLB instance belongs.	None	CCM 1.9.3.313-g748f81e-aliyun and later
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-name	string	The name of the SLB instance.	None	CCM 1.9.3.313-g748f81e-aliyun and later
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-hostname	string	Specifies a hostname for the Service. The hostname must comply with the naming conventions of domain names.	None	CCM 2.3.0 and later

- Common annotations that are used to configure listeners

Annotation	Type	Description	Default	Supported CCM version
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-scheduler	string	The scheduling algorithm. Valid values: *wrr and rr*. ◦ *wrr*: Backend servers with higher weights receive more requests than backend servers with lower weights. ◦ *rr*: Requests are distributed to backend servers in sequence. This is the default scheduling algorithm.	*rr*	CCM 1.9.3 and later

Annotation	Type	Description	Default	Supported CCM version
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-protocol-port	string	The listener port. Separate multiple ports with commas (,). Example: `https:443,http:80` .	None	CCM 1.9.3 and later
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-persistence-timeout	string	The session persistence period. Unit: seconds. This annotation applies only to TCP listeners. Valid values: *0 to 3600*. Default value: *0*. By default, session persistence is disabled. For more information, see CreateLoadBalancerTCPListener.	*0*	CCM 1.9.3 and later
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-sticky-session	string	Specifies whether to enable session persistence. Valid values: *on and off*. **Note** This annotation applies only to HTTP and HTTPS listeners. For more information, see CreateLoadBalancerHTTPListener and CreateLoadBalancerHTTPSListener.	*off*	CCM 1.9.3 and later

Annotation	Type	Description	Default	Supported CCM version
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-sticky-session-type	string	The method that is used to process cookies. Valid values: ◦ *insert*: inserts a cookie. ◦ *server*: rewrites a cookie. **Note** ◦ This annotation applies only to HTTP and HTTPS listeners. ◦ If the annotation service.beta.kubernetes.io/alibaba-cloud-loadbalancer-sticky-session is set to *on*, you must specify this annotation. For more information, see CreateLoadBalancerHTTPListener and CreateLoadBalancerHTTPSListener.	None	CCM 1.9.3 and later

Annotation	Type	Description	Default	Supported CCM version
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-cookie-timeout	string	The timeout period of a cookie. Valid values: *1 to 86400*. Unit: seconds. **Note** If service.beta.kubernetes.io/alibaba-cloud-loadbalancer-sticky-session is set to *on* and service.beta.kubernetes.io/alibaba-cloud-loadbalancer-sticky-session-type is set to *insert*, you must specify this annotation. For more information, see CreateLoadBalancerHTTPListener and CreateLoadBalancerHTTPSListener.	None	CCM 1.9.3 and later

Annotation	Type	Description	Default	Supported CCM version
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-cookie	string	The name of a cookie configured on a backend server. The name must be 1 to 200 characters in length, and can contain only ASCII letters and digits. It cannot contain commas (,), semicolons (;), or space characters. It cannot start with a dollar sign ($). **Note** If service.beta.kubernetes.io/alibaba-cloud-loadbalancer-sticky-session is set to *on* and service.beta.kubernetes.io/alibaba-cloud-loadbalancer-sticky-session-type is set to *server*, you must specify this annotation. For more information, see CreateLoadBalancerHTTPListener and CreateLoadBalancerHTTPSListener.	None	CCM 1.9.3 and later
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-cert-id	string	The ID of a certificate for an SLB instance. You must first upload the certificate in the SLB console.	None	CCM 1.9.3.164-g2105d2e-aliyun and later
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-health-check-flag	string	Valid values: *on and off*. ◦ Default value for TCP listeners: on. You cannot modify the value. ◦ Default value for HTTP listeners: off.	Default value: *off*. This annotation is optional for TCP listeners. By default, the health check feature is enabled for TCP listeners. This feature cannot be disabled.	CCM 1.9.3 and later

Annotation	Type	Description	Default	Supported CCM version
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-health-check-type	string	The type of health check. Valid values: *tcp and http*. For more information, see CreateLoadBalancerTCPListener.	*tcp*	CCM 1.9.3 and later
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-health-check-uri	string	The URI that is used for health checks. **Note** If the type of health check is TCP, this annotation is optional. For more information, see CreateLoadBalancerTCPListener.	None	CCM 1.9.3 and later
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-health-check-connect-port	string	The port that is used for health checks. Valid values: 1 to 65535.	None	CCM 1.9.3 and later
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-healthy-threshold	string	The number of consecutive successful health checks that must occur before an unhealthy backend server is declared healthy. Valid values: 2 to 10. For more information, see CreateLoadBalancerTCPListener.	*3*	CCM 1.9.3 and later

Annotation	Type	Description	Default	Supported CCM version
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-unhealthy-threshold	string	The number of consecutive failed health checks that must occur before a healthy backend server is declared unhealthy. Valid values: *2~10* For more information, see CreateLoadBalancerTCPListener.	*3*	CCM 1.9.3 and later
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-health-check-interval	string	The interval between two consecutive health checks. Unit: seconds. Valid values: *1 to 50*. Unit: seconds. For more information, see CreateLoadBalancerTCPListener.	*2*	CCM 1.9.3 and later
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-health-check-	string	The timeout period of a health check probe. This annotation applies to TCP health checks. If a backend server does not respond within the specified timeout period, the server fails to pass the health check. Valid values: *1 to 300*. Unit: seconds.	*5*	CCM 1.9.3 and later

Annotation	Type	Description	Default	Supported CCM version
connect-timeout		**Note** If the value of service.beta.kubernetes.io/alibaba-cloud-loadbalancer-health-check-connect-timeout is smaller than that of service.beta.kubernetes.io/alibaba-cloud-loadbalancer-health-check-interval, service.beta.kubernetes.io/alibaba-cloud-loadbalancer-health-check-connect-timeout does not take effect. In this case, the value of service.beta.kubernetes.io/alibaba-cloud-loadbalancer-health-check-interval is used as the timeout period. For more information, see CreateLoadBalancerTCPListener.		

Annotation	Type	Description	Default	Supported CCM version
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-health-check-timeout	string	The timeout period of a health check probe. This annotation applies to HTTP health checks. If a backend server does not respond within the specified timeout period, the server fails to pass the health check. Valid values: *1 to 300*. Unit: seconds. **Note** If the value of service.beta.kubernetes.io/alibaba-cloud-loadbalancer-health-check-timeout is smaller than that of service.beta.kubernetes.io/alibaba-cloud-loadbalancer-health-check-interval, service.beta.kubernetes.io/alibaba-cloud-loadbalancer-health-check-timeout does not take effect. In this case, the value of service.beta.kubernetes.io/alibaba-cloud-loadbalancer-health-check-interval is used as the timeout period. For more information, see CreateLoadBalancerTCPListener.	*5*	CCM 1.9.3 and later

Annotation	Type	Description	Default	Supported CCM version
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-health-check-domain	string	The domain name that is used for health checks. ◦ *$_ip*: the private IP addresses of backend servers. If you do not set this annotation or set the annotation to $_ip, the SLB instance uses the private IP address of each backend server as the domain name for health checks. ◦ *domain*: The domain name must be 1 to 80 characters in length, and can contain only letters, digits, periods (.), and hyphens (-).	None	CCM 1.9.3 and later
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-health-check-httpcode	string	The HTTP status code that is returned when a health check is successful. Separate multiple HTTP status codes with commas (,). Valid values: ◦ *http_2xx* ◦ *http_3xx* ◦ *http_4xx* ◦ *http_5xx* Default value: *http_2xx*.	*http_2xx*	CCM 1.9.3 and later
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-acl-status	string	Specifies whether to enable access control for a listener. Valid values: *on and off*.	*off*	CCM 1.9.3.164-g2105d2e-aliyun and later
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-acl-id	string	The ID of the network access control list (ACL) that is associated with a listener. If the annotation service.beta.kubernetes.io/alibaba-cloud-loadbalancer-acl-status is set to on, this annotation is required.	None	CCM 1.9.3.164-g2105d2e-aliyun and later

Annotation	Type	Description	Default	Supported CCM version
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-acl-type	string	The type of network ACL. Valid values: *white and black*. ◦ *white*: specifies the network ACL as a whitelist. Only requests from the IP addresses or CIDR blocks in the network ACL are forwarded. Whitelists apply to scenarios when you want to allow only specified IP addresses to access an application. Your service may be adversely affected if the whitelist is not properly configured. After a whitelist is set, the SLB instance forwards only requests from the IP addresses in the whitelist. If a whitelist is configured but no IP address is added to the whitelist, the listener forwards all requests. ◦ *black*: specifies the network ACL as a blacklist. All requests from the IP addresses or CIDR blocks in the network ACL are rejected. Blacklists apply to scenarios when you want to block access from specified IP addresses to an application. If a blacklist is configured but no IP address is added to the blacklist, the listener forwards all requests. If the annotation service.beta.kubernetes.io/alibaba-cloud-loadbalancer-acl-status is set to on, this annotation is required.	None	CCM 1.9.3.164-g2105d2e-aliyun and later

Annotation	Type	Description	Default	Supported CCM version
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-forward-port	string	Redirects HTTP requests to a specified HTTPS port. Example: `80:443` .	None	CCM 1.9.3.164-g2105d2e-aliyun and later
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-connection-drain	string	Specifies whether to enable connection draining. Valid values: *on and off*.	None	CCM 2.0.1 and later
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-connection-drain-timeout	string	The timeout period of connection draining. Unit: seconds. Value values: *10 to 900*.	None	CCM 2.0.1 and later
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-xforwardedfor-proto	string	Specifies whether to use the X-Forwarded-Proto header to obtain the listener protocol of the SLB instance. Valid values: *on and off*.	off	CCM 2.1.0 and later
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-idle-timeout	string	Specifies the timeout period of idle connections for a listener. Unit: seconds. Valid values: *1 to 60*.	15	CCM 2.1.0 and later
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-http2-enabled	string	Specifies whether to enable HTTP/2. Valid values: *on and off*.	on	CCM 2.1.0 and later
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-established-timeout	string	The timeout period of connections. Only TCP is supported. Unit: seconds. Value values: 10 to 900. For more information, see CreateLoadBalancerTCPListener.	None	CCM 2.3.0 and later

Annotation	Type	Description	Default	Supported CCM version
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-request-timeout	string	The timeout period of requests. HTTP and HTTPS are supported. Unit: seconds. Valid values: 1 to 180. If no response is received from the backend server within the timeout period, SLB returns an HTTP 504 error code to the client.	60	CCM 2.3.0 and later
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-health-check-method	string	The health check method used for HTTP listeners. Valid values: `head` and `get`.	None	CCM 2.3.0 and later

- **Common annotations that are used to configure backend server groups**

Annotation	Type	Description	Default	Supported CCM version
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-backend-label	string	Specifies that worker nodes that have matching labels are added as the backend servers of an SLB instance.	None	CCM 1.9.3 and later
externalTrafficPolicy	string	The policy that is used to add nodes as backend servers. Valid values: ◦ *Cluster*: adds all nodes as backend servers. ◦ *Local*: adds the nodes where the pods are deployed as backend servers.	*Cluster*	CCM 1.9.3 and later
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-remove-unscheduled-backend	string	Removes backend servers that are in the Unschedulable state from an SLB instance. Valid values*: on and off*.	off	CCM 1.9.3.164-g2105d2e-aliyun and later

Annotation	Type	Description	Default	Supported CCM version
service.beta.kubernetes.io/backend-type	string	The type of backend servers that are added to an SLB instance. Valid values: ◦ `eni` : adds pods as the backend servers of an SLB instance. This parameter takes effect only in Terway network mode. This improves network forwarding performance. ◦ `ecs` : adds ECS instances as the backend servers of an SLB instance.	When the Flannel network plug-in is used, the default value is `ecs` . When the Terway network plug-in is used: ◦ The default value is `ecs` . This applies to Container Service for Kubernetes (ACK) clusters that were created before August 10, 2020. ◦ The default value is `eni` . This applies to ACK clusters that are created after August 10, 2020.	CCM 1.9.3.164-g2105d2e-aliyun and later
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-vgroup-port	string	Reuses an existing vServer group. This annotation takes effect only when the existing SLB instance is reused. Separate multiple ports and vServer groups with commas (,).	None	CCM 2.0.1 and later

Annotation	Type	Description	Default	Supported CCM version
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-weight	string	Specifies the weight of a Service when an SLB instance is shared by multiple Services. This annotation takes effect only when the existing vServer group is reused.	None	CCM 2.0.1 and later

Related information

- What do I do if the annotations of a Service do not take effect?
- Service FAQ

13.1.3. Use an existing SLB instance to expose an application

Container Service for Kubernetes (ACK) allows you to use a Server Load Balancer (SLB) instance to expose a Service. To access the Service from outside the cluster, you can use the domain name of the SLB instance or the connection string `<IP:Service port>`. To access the Service from within the cluster, you can use the connection string `<Service name:Service port>`. This topic describes how to use an existing SLB instance to expose an application. An NGINX application is used as an example.

Prerequisites

An SLB instance is created by using the SLB console. The SLB instance is deployed in the region where the cluster is created. For more information about how to create an SLB instance, see Create a CLB instance.

Context

By default, cloud controller manager (CCM) v1.9.3 and later do not automatically configure listeners for existing SLB instances. You can add the annotation `service.beta.kubernetes.io/alibaba-cloud-loadbalancer-force-override-listeners: "true"` to enable CCM to configure listeners. You can also manually configure listeners for an SLB instance.

You can use the following methods to check the CCM version:

- **Console**: Log on to the ACK console and check the CCM version on the Add-ons page.
    i. Log on to .
    ii. In the left-side navigation pane, click **Clusters**.
    iii. On the **Clusters** page, find the cluster that you want to manage, and choose **More > Manage System Components** in the Actions column. On the **Add-ons** page, check the CCM version on the Core Components tab.
- **kubectl**: Run the following command to check the CCM version. This method applies to only dedicated Kubernetes clusters.

```
kubectl get pod -n kube-system -o yaml|grep image:|grep cloud-con|uniq
```

Precautions

- When you use an existing SLB instance to expose an application, take note of the following limits:
    - The SLB instance must be created by using the SLB console. You cannot reuse SLB instances that are automatically created by CCM.
    - To reuse an internal-facing SLB instance in a cluster, the SLB instance and the cluster must be deployed in the same virtual private cloud (VPC).
    - The network type of the SLB instance must be consistent with the connection method of the Service. If the Service supports **public access** (`service.beta.kubernetes.io/alibaba-cloud-loadbalancer-address-type: "internet"`), the **network type** of the SLB instance must be **Internet-facing**. If the Service supports **internal access** (`service.beta.kubernetes.io/alibaba-cloud-loadbalancer-address-type: "intranet"`), the **network type** of the SLB instance must be **internal-facing**.
    - The SLB instance must listen on different Service ports if the SLB instance exposes more than one Service.
- CCM configures SLB instances only for `Type=LoadBalancer` Services. CCM does not configure SLB instances for other types of Services.

  Notice When a Service is changed from `Type=LoadBalancer` to another type, CCM deletes the configurations that are added to the SLB instance of the Service. As a result, you can no longer use the SLB instance to access the Service.

- CCM uses a declarative API. CCM automatically updates the configurations of an SLB instance to match the configurations of the exposed Service when specific conditions are met. If you set `service.beta.kubernetes.io/alibaba-cloud-loadbalancer-force-override-listeners:` to `true` , the configuration modifications that you add in the SLB console may be overwritten.

  Notice Do not use the SLB console to modify the configurations of the SLB instance that is created and managed by CCM. Otherwise, the modifications may be overwritten and the Service may become inaccessible.

SLB resource quotas

- CCM creates SLB instances for `Type=LoadBalancer` Services. By default, you can have a maximum of 60 SLB instances within each Alibaba Cloud account. To create more than 60 SLB instances, .

  Note In the ticket, specify that you want to modify the `slb_quota_instances_num` parameter to create more SLB instances.

- CCM automatically creates SLB listeners that use Service ports. By default, each SLB instance supports a maximum of 50 listeners. To create more than 50 listeners for an SLB instance, submit a ticket.

  Note In the ticket, specify that you want to modify the `slb_quota_listeners_num` parameter to create more listeners for each SLB instance.

- CCM automatically adds Elastic Compute Service (ECS) instances to backend server groups of an SLB instance based on the Service configurations.

- By default, an ECS instance can be added to up to 50 backend server groups. To add an ECS instance to more than 50 server groups, .

  Note In the ticket, specify that you want to modify the `slb_quota_backendserver_attached_num` parameter to add an ECS instance to more server groups.

- By default, you can add up to 200 backend servers to an SLB instance. To add more backend servers to an SLB instance, submit a ticket.

  Note In the ticket, specify that you want to modify the `slb_quota_backendservers_num` parameter to add more backend servers to an SLB instance.

For more information about SLB resource quotas, see Limits. To query SLB resource quotas, go to the Quota Management page in the SLB console.

Step 1: Deploy a sample application

The following section describes how to use the **kubectl** command-line tool to deploy an application. For more information about how to deploy an application by using the ACK console, see Create a stateless application by using a Deployment.

1. Use the following YAML template to create a *my-nginx.yaml* file:

```
apiVersion: apps/v1 # for versions before 1.8.0 use apps/v1beta1
kind: Deployment
metadata:
  name: my-nginx    #The name of the sample application.
  labels:
    app: nginx
spec:
  replicas: 3       #The number of replicas.
  selector:
    matchLabels:
      app: nginx     #You must specify the same value in the selector of the Service th
at is used to expose the application.
  template:
    metadata:
      labels:
        app: nginx
    spec:
    #  nodeSelector:
    #    env: test-team
      containers:
      - name: nginx
        image: registry.aliyuncs.com/acs/netdia:latest     #Replace the value with the
image address. Format: <image_name:tags>.
        ports:
        - containerPort: 80                                #The port must be exposed in
the Service.
```

2. Run the following command to deploy the my-nginx application:

```
kubectl apply -f my-nginx.yaml
```

3. Run the following command to check the state of the application:

```
kubectl get deployment my-nginx
```

Sample response:

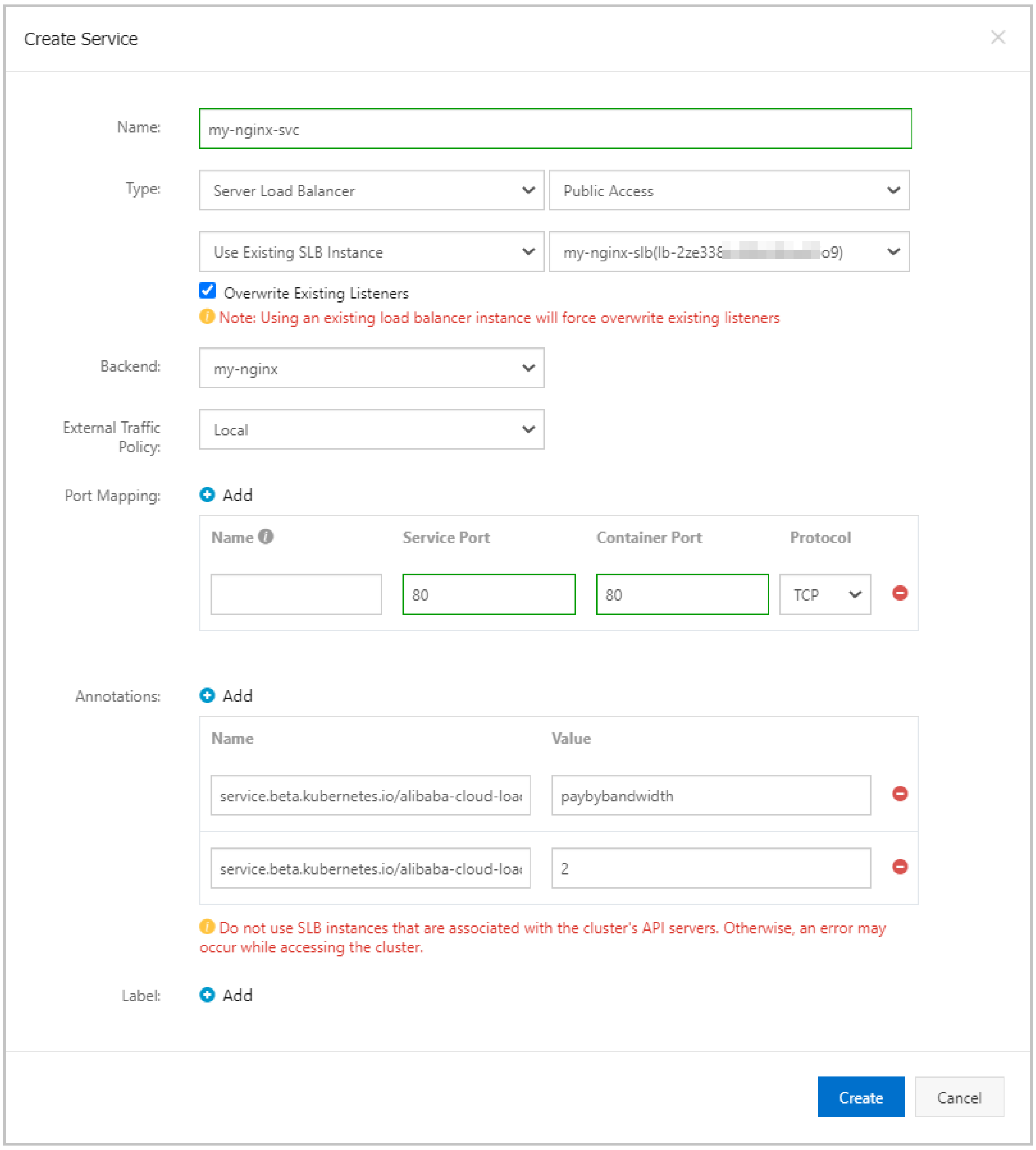
```
NAME        READY   UP-TO-DATE    AVAILABLE    AGE
my-nginx    3/3     3             3            50s
```

Step 2: Use an existing SLB instance to expose the application

You can use the ACK console or **kubectl** to create a LoadBalancer Service. After the Service is created, you can use the Service to expose the application.

Use the ACK console

1.
2.
3.
4.
5. On the **Services** page, click **Create** in the upper-right corner of the page.
6. In the **Create Service** dialog box, set the required parameters.


Create Service

Name: my-nginx-svc

Type: Server Load Balancer | Public Access

Use Existing SLB Instance | my-nginx-slb(lb-2ze338[illegible]o9)

☑ Overwrite Existing Listeners

Note: Using an existing load balancer instance will force overwrite existing listeners

Backend: my-nginx

External Traffic Policy: Local

Port Mapping: Add

Name	Service Port	Container Port	Protocol
	80	80	TCP

Annotations: Add

Name	Value
service.beta.kubernetes.io/alibaba-cloud-loa	paybybandwidth
service.beta.kubernetes.io/alibaba-cloud-loa	2

Do not use SLB instances that are associated with the cluster's API servers. Otherwise, an error may occur while accessing the cluster.

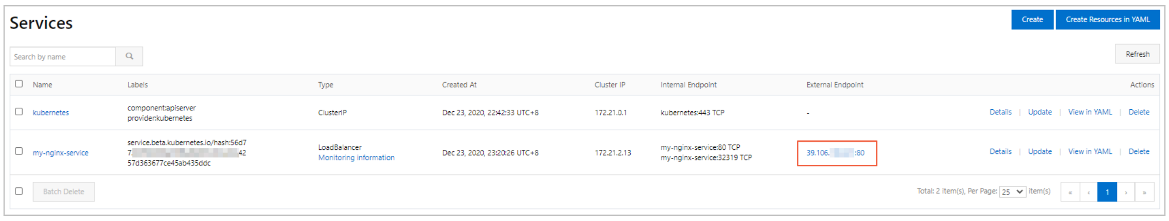
Label: Add

Parameter	Description
Name	Enter a name for the Service. my-nginx-svc is used in this example.

Parameter	Description
Type	Select the type of the Service. This parameter determines how the Service is accessed. ◦ Choose **Server Load Balancer** -> **Public Access** -> **Use Existing SLB Instance** and select an SLB instance from the drop-down list. ◦ **Overwrite Existing Listeners**: Specify whether to overwrite the listeners of the selected SLB instance. If you select this check box but the SLB instance does not have listeners, the system automatically creates listeners for the SLB instance. In this example, the SLB instance is newly created and therefore you must select this check box to create listeners for the SLB instance. Note ▪ If the listeners of the SLB instance are associated with applications, service interruptions may occur after the configurations of the listeners are overwritten. ▪ CCM supports limited backend configurations and cannot handle complex configurations. If you require complex backend configurations, you can manually modify the listeners in the SLB console without overwriting the existing configurations of the listeners. In both cases, we recommend that you do not overwrite the configurations of the listeners. However, you can overwrite the configuration of a listener if the port of the listener is no longer used.
Backend	Select the application that you want to associate with the Service. The my-nginx application is selected in this example. If you do not associate the Service with a backend, no Endpoint object is created. You can manually associate the Service with a backend. For more information, see services-without-selectors.
External Traffic Policy	Select a policy to distribute external network traffic. **Local** is selected in this example. ◦ **Local**: routes network traffic to only pods on the node where the Service is deployed. ◦ **Cluster**: routes network traffic to pods on other nodes in the cluster. Note The **External Traffic Policy** parameter is available only if you set Type to **Node Port** or **Server Load Balancer**.
Port Mapping	Specify a Service port and a container port. The Service port corresponds to the `port` field in the YAML file and the container port corresponds to the `targetPort` field in the YAML file. The container port must be the same as the one that is exposed in the backend pod. Both ports are set to 80 in this example.

Parameter	Description
Annotations	Add one or more annotations to the Service to modify the configuration of the SLB instance. You can select **Custom Annotation** or **Alibaba Cloud Annotation** from the Type drop-down list. In this example, the billing method is set to pay-by-bandwidth and the maximum bandwidth is set to 2 Mbit/s to limit the amount of traffic that flows through the Service. For more information, see Use annotations to configure load balancing. ◦ **Type**: Alibaba Cloud Annotation ◦ **Name**: In this example, two annotations are created with the following names: `service.beta.kubernetes.io/alibaba-cloud-loadbalancer-charge-type` and `service.beta.kubernetes.io/alicloud-loadbalancer-bandwidth` . ◦ **Value**: In this example, the values of the annotations are set to paybybandwidth and 2.
Label	Add one or more labels to the Service. Labels are used to identify the Service.

7. Click **Create**.
   On the **Services** page, you can view the created Service.



Services

	Name	Labels	Type	Created At	Cluster IP	Internal Endpoint	External Endpoint	Actions
☐	kubernetes	component:apiserver provider:kubernetes	ClusterIP	Dec 23, 2020, 22:42:33 UTC+8	172.21.0.1	kubernetes:443 TCP	-	Details | Update | View in YAML | Delete
☐	my-nginx-service	service.beta.kubernetes.io/hash:56d7 7.[redacted]42 57d363677ce45ab435ddc	LoadBalancer Monitoring information	Dec 23, 2020, 23:20:26 UTC+8	172.21.2.13	my-nginx-service:80 TCP my-nginx-service:32319 TCP	39.106.[redacted]:80	Details | Update | View in YAML | Delete

Total: 2 item(s), Per Page: 25 item(s)

8. Click **39.106.XX.XX:80** in the **External Endpoint** column to access the sample application.

Use kubectl

1. Use the following YAML template to create a *my-nginx-svc.yaml* file.
   - Replace the value *${YOUR_LB_ID}* of `service.beta.kubernetes.io/alibaba-cloud-loadbalancer-id` with the ID of the SLB instance that is created in the SLB console.
   - If you use an existing SLB instance, CCM does not create listeners for the SLB instance or overwrite the listeners of the SLB instance by default. If you want CCM to create new listeners or overwrite existing listeners, you can set `service.beta.kubernetes.io/alibaba-cloud-loadbalancer-force-override-listeners` to `true` . In this example, the SLB instance is newly created and therefore you must set this annotation to `true` to create listeners for the SLB instance. For more information, see Use annotations to configure load balancing.
   - To associate the Service with the backend application, set selector to the value of matchLabels in the *my-nginx.yaml* file. The value is `app: nginx` in this example.

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-id: ${YOUR_LB_ID}
    service.beta.kubernetes.io/alicloud-loadbalancer-force-override-listeners: 'true'
  labels:
    app: nignx
  name: my-nginx-svc
  namespace: default
spec:
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: nginx
  type: LoadBalancer
```

2. Run the following command to create a Service named my-nginx-svc and use the Service to expose the application:

```
kubectl apply -f my-nginx-svc.yaml
```

3. Run the following command to verify that the LoadBalancer Service is created:

```
kubectl get svc my-nginx-svc
```

Sample response:

```
NAME           TYPE           CLUSTER-IP     EXTERNAL-IP      PORT(S)        AGE
my-nginx-svc   LoadBalancer   172.21.XX.XX    39.106.XX.XX     80:30471/TCP    5m
```

4. Run the **curl <YOUR-External-IP>** command to access the sample application. Replace *<YOUR-External-IP>* with the IP address displayed in the `EXTERNAL-IP` column.

```
curl 39.106.XX.XX
```

Sample response:

```
<!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>
```

Related information

- Use annotations to configure load balancing
- Use an automatically created SLB instance to expose an application

13.1.4. Create a Service

Each pod in Kubernetes clusters has its own IP address. However, pods are frequently created and deleted. Therefore, it is not practical to directly expose pods to external access. Services decouple the frontend from the backend, which provides a loosely-coupled microservice architecture. This topic describes how to create, update, and delete Services by using the Container Service for Kubernetes (ACK) console and kubectl.

Prerequisites

A serverless Kubernetes (ASK) cluster is created. For more information, see ASK quick start.

Context

A Kubernetes Service, also known as a microservice, is an abstraction that defines a logical set of pods and a policy that is used to access the pods. A label selector is used to determine which set of pods is targeted by a Service.

Each pod in Kubernetes clusters has its own IP address. However, pods are frequently created and deleted. Therefore, it is not highly available to directly expose pods to external access. Services decouple the frontend from the backend. The frontend clients do not need to be aware of which backend pods are used. This provides a loosely-decoupled microservice architecture.

For more information, see Kubernetes Services.

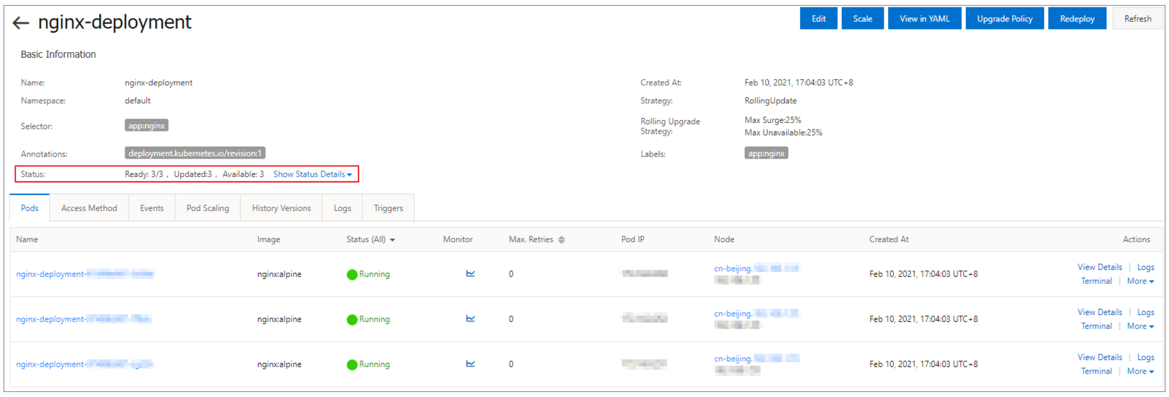
Step 1: Create a Deployment

1.
2.
3.
4.
5. On the **Deployments** page, click **Create from YAML** in the upper-right corner of the page.
6. Select the sample template or enter a custom template, and then click **Create**.

   In this example, the Deployment template for NGINX applications is selected.

```
apiVersion: apps/v1
kind: Deployment
metadata:
   name: nginx-deployment-basic
   labels:
     app: nginx
spec:
   replicas: 2
   selector:
     matchLabels:
       app: nginx
   template:
     metadata:
       labels:
         app: nginx
     spec:
       containers:
       - name: nginx
         image: nginx:1.7.9                # replace it with your exactly <image_name:t
ags>
         ports:
         - containerPort: 80               ## Expose this port in the Service.
```

7. On the Deployments page, find the created Deployment, and click the name of the Deployment or click **Details** in the Actions column. On the page that appears, you can view the status of the Deployment.



Basic Information

Name:	nginx-deployment	Created At:	Feb 10, 2021, 17:04:03 UTC+8
Namespace:	default	Strategy:	RollingUpdate
Selector:	app:nginx	Rolling Upgrade Strategy:	Max Surge:25% Max Unavailable:25%
Annotations:	deployment.kubernetes.io/revision:1	Labels:	app:nginx
Status:	Ready: 3/3 , Updated:3 , Available: 3 Show Status Details		

Name	Image	Status (All)	Monitor	Max. Retries	Pod IP	Node	Created At	Actions
nginx-deployment-[redacted]	nginx:alpine	● Running		0	[redacted]	cn-beijing.[redacted]	Feb 10, 2021, 17:04:03 UTC+8	View Details | Logs Terminal | More
nginx-deployment-[redacted]	nginx:alpine	● Running		0	[redacted]	cn-beijing.[redacted]	Feb 10, 2021, 17:04:03 UTC+8	View Details | Logs Terminal | More
nginx-deployment-[redacted]	nginx:alpine	● Running		0	[redacted]	cn-beijing.[redacted]	Feb 10, 2021, 17:04:03 UTC+8	View Details | Logs Terminal | More

Step 2: Create a Service

1.

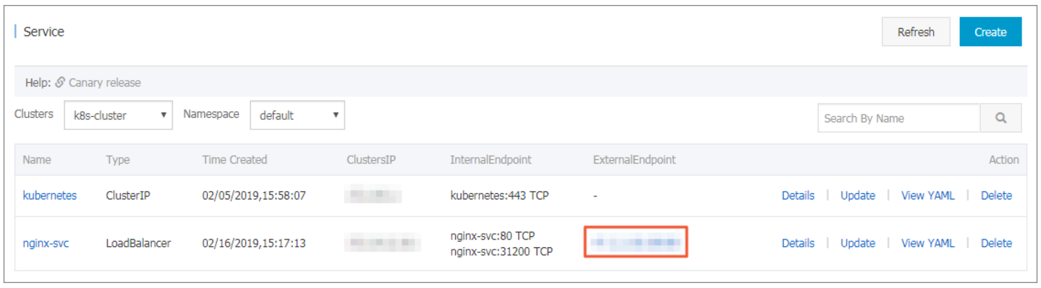
2. On the Services page, click **Create** in the upper-right corner of the page.
3. In the **Create Service** dialog box, set the parameters.

Parameter	Description
Name	Enter a name for the Service.
Type	The type of Service. This parameter determines how the Service is accessed. Valid values: ◦ Cluster IP: the ClusterIP Service. This type of Service is exposed by using the internal IP address of the cluster. This is the default value. If you select this option, the Service is accessible only from within the cluster. **Note** The **Headless Service** checkbox is available only if you set Type to **Cluster IP**. If you select this check box, you can use a headless Service to interface with other service discovery mechanisms, without being tied to the implementation of service discovery in Kubernetes. ◦ Server Load Balancer: the LoadBalancer Service. This type of Service is accessed by using Server Load Balancer (SLB) instances. If you select this option, you can enable internal or external access to the Service. LoadBalancer Services can be used to route requests to NodePort and ClusterIP Services. ▪ Create SLB Instance: You can click **Modify** to change the specification of the SLB instance. ▪ Use Existing SLB Instance: You can select an existing SLB instance. **Note** You can create an SLB instance or use an existing SLB instance. You can also associate an SLB instance with more than one Service. However, you must take note of the following limits: ▪ If you use an existing SLB instance, the listeners of the SLB instance overwrite those of the Service. ▪ If an SLB instance is created along with a Service, you cannot reuse this SLB instance when you create other Services. Otherwise, the SLB instance may be deleted. You can reuse only SLB instances that are manually created in the console or by calling the API. ▪ An SLB instance must listen on different Service ports if the SLB instance exposes more than one Service. Otherwise, port conflicts may occur. ▪ When you use an SLB instance, the names of listeners and vServer groups are used as unique identifiers in Kubernetes. Do not modify the names of listeners and vServer groups. ▪ You cannot use an SLB instance to expose Services across clusters.

Parameter	Description
Backend	Select the backend application that you want to associate with the Service. If you do not associate the Service with a backend application, no Endpoint object is created. You can also manually associate the Service with a backend application. For more information, see services-without-selectors.
External Traffic Policy	Select a policy to distribute external traffic. ◦ **Local**: This policy routes traffic only to pods on the node where the Service is deployed. ◦ **Cluster**: This policy can route traffic to pods on other nodes in the cluster. **Note** The **External Traffic Policy** parameter is available only when you set Type to **Node Port** or **Server Load Balancer**.
Port Mapping	Specify a Service port and a container port. The Service port corresponds to the `port` field in the YAML file and the container port corresponds to the `targetPort` field in the YAML file. The container port must be the same as the one that is exposed in the backend pod.
Annotations	Add one or more annotations to the Service to configure the SLB instance. You can select **Custom Annotation** or **Alibaba Cloud Annotation** from the Type drop-down list. For example, the annotation `service.beta.kubernetes.io/alicloud-loadbalancer-bandwidth:2` specifies that the maximum bandwidth of the Service is 2 Mbit/s. This limits the amount of traffic that flows through the Service.
Label	Add one or more labels to the Service. The labels are used to identify the Service.

4. Click **Create**.

   On the Services page, you can view the created Service in the Service list.



Clusters: k8s-cluster Namespace: default

Name	Type	Time Created	ClustersIP	InternalEndpoint	ExternalEndpoint	Action
kubernetes	ClusterIP	02/05/2019,15:58:07	[redacted]	kubernetes:443 TCP	-	Details | Update | View YAML | Delete
nginx-svc	LoadBalancer	02/16/2019,15:17:13	[redacted]	nginx-svc:80 TCP nginx-svc:31200 TCP	[redacted]	Details | Update | View YAML | Delete

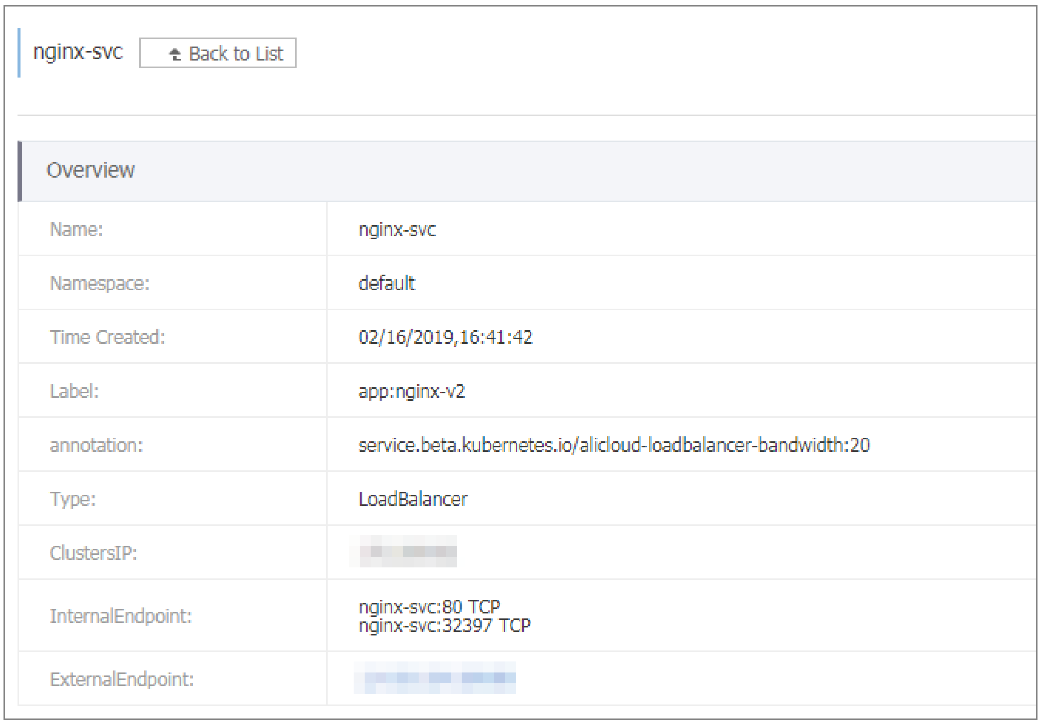
13.1.5. Manage Services

Each pod in Kubernetes clusters has its own IP address. However, pods are frequently created and deleted. Therefore, it is not practical to directly expose pods to external access. The Service resource decouples the frontend from the backend, which provides a loosely-coupled microservice architecture. This topic describes how to view, update, and delete a Service.

View a Service

1.
2.
3.
4.
5. Specify a cluster and a namespace, find the service that you want to view, and then click **Details** in the Actions column of the service.

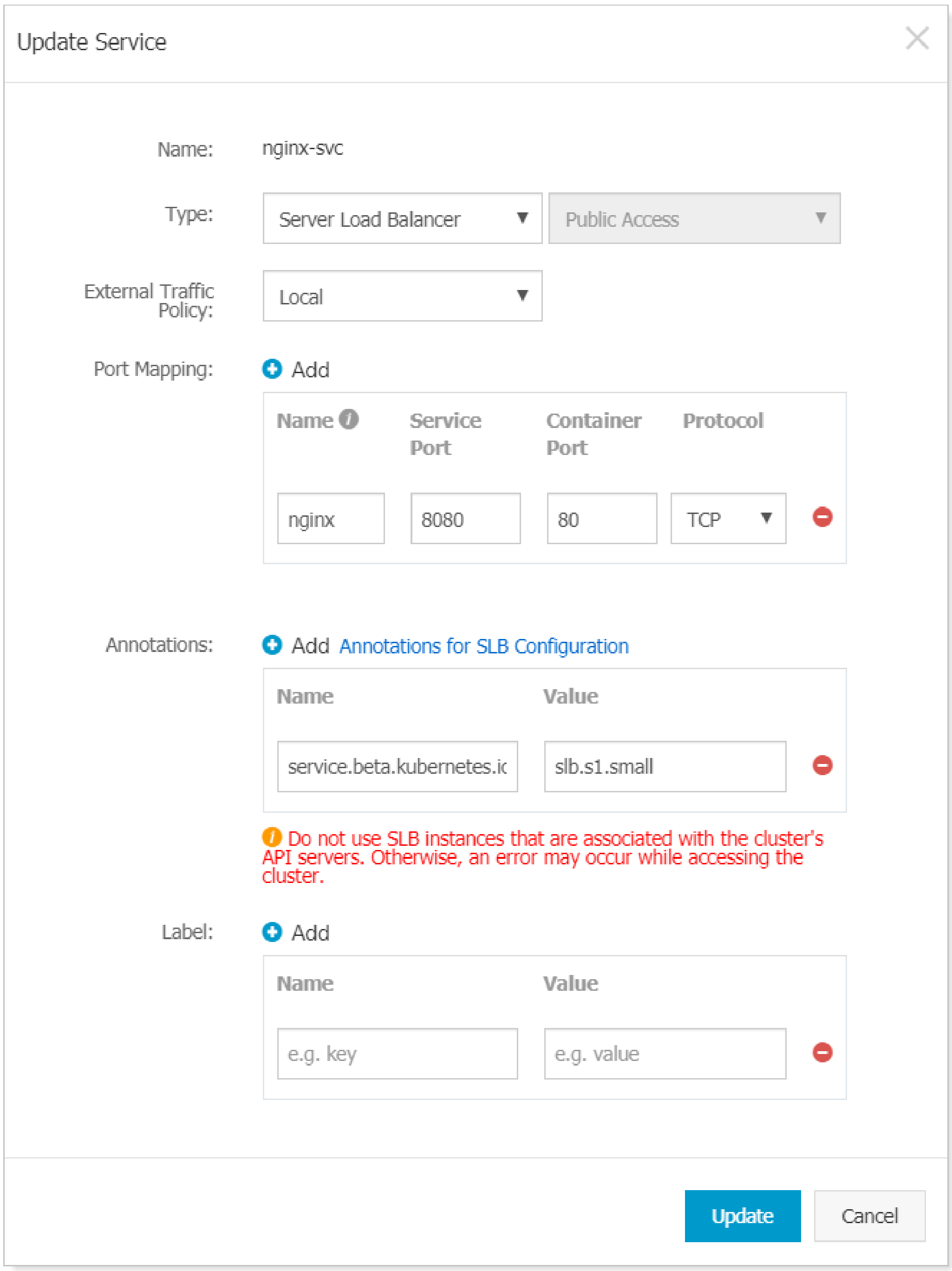
   You can view the information such as the name, type, creation time, cluster IP address, and external endpoint of the service. The following figure shows the external endpoint (IP address) that is assigned to the service. To access the NGINX application, click this IP address.



Name:	nginx-svc
Namespace:	default
Time Created:	02/16/2019,16:41:42
Label:	app:nginx-v2
annotation:	service.beta.kubernetes.io/alicloud-loadbalancer-bandwidth:20
Type:	LoadBalancer
ClustersIP:	[redacted]
InternalEndpoint:	nginx-svc:80 TCP nginx-svc:32397 TCP
ExternalEndpoint:	[redacted]

Update a Service

1.
2.
3.
4.
5. Specify a cluster and a namespace, find the service that you want to update, and then click **Update** in the Actions column of the service. In this example, the nginx-svc service is updated.
6. In the dialog box that appears, modify the configurations based on your business requirements and click **Update**.


Update Service

Name: nginx-svc

Type: Server Load Balancer | Public Access

External Traffic Policy: Local

Port Mapping: Add

Name	Service Port	Container Port	Protocol
nginx	8080	80	TCP

Annotations: Add Annotations for SLB Configuration

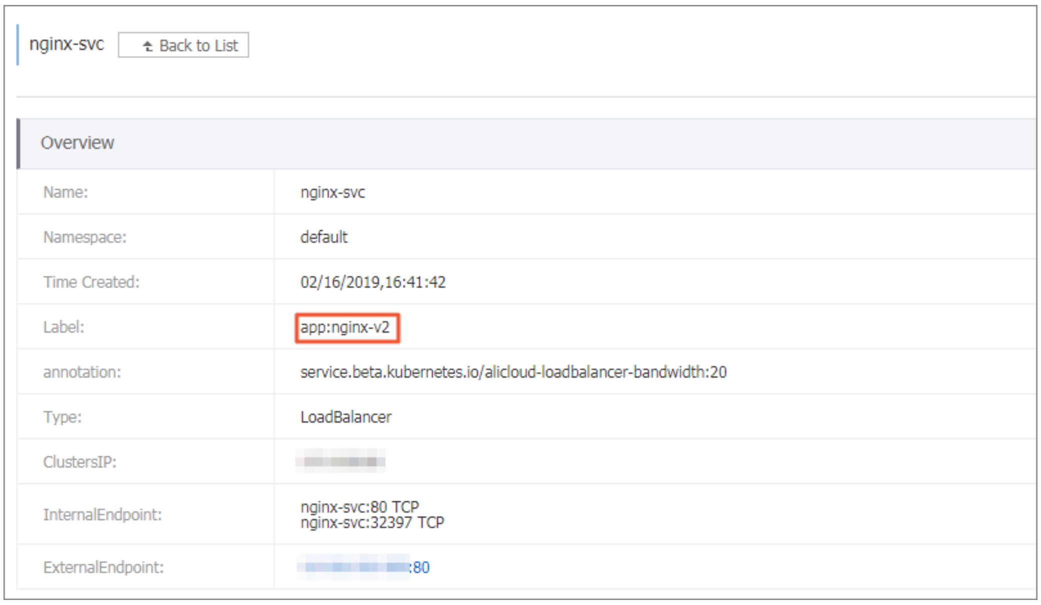
Name	Value
service.beta.kubernetes.ic	slb.s1.small

Do not use SLB instances that are associated with the cluster's API servers. Otherwise, an error may occur while accessing the cluster.

Label: Add

Name	Value
e.g. key	e.g. value

7. Find the service on the Services page and click **Details** in the Actions column to view configuration changes. In this example, the label of the service is updated.


nginx-svc
Back to List

Overview

Name:	nginx-svc
Namespace:	default
Time Created:	02/16/2019,16:41:42
Label:	app:nginx-v2
annotation:	service.beta.kubernetes.io/alicloud-loadbalancer-bandwidth:20
Type:	LoadBalancer
ClustersIP:	[redacted]
InternalEndpoint:	nginx-svc:80 TCP nginx-svc:32397 TCP
ExternalEndpoint:	[redacted]:80

Delete a Service

1.
2.
3.
4.
5. Find the service that you want to delete and click **Delete** in the Actions column of the service.
6. In the message that appears, click **Confirm**. The service is deleted and disappears from the Services page.

13.2. Compare NGINX Ingresses and ALB Ingresses

Item	Nginx Ingress	ALB Ingress

Item	Nginx Ingress	ALB Ingress
Service positioning	• Provides traffic management and advanced routing features at Layer 7. • A cluster component that supports highly customized configuration.	• Provides traffic management and advanced routing features at Layer 7. • Runs at the application layer, provides deep integration with containers, and supports different release policies, such as canary release, A/B testing, blue-green deployment, and traffic distribution by ratio. • Provides ultra-large capacities and supports auto scaling and automated O&M. • Supports integration with multiple cloud services, such as Web Application Firewall (WAF), Function Compute, PrivateLink, and transit routers.
Architecture	Provides extended features based on NGINX and Lua.	• Developed based on the Cloud Network Management platform. • Developed based on the CyberStar platform and supports auto scaling.
Basic routing	• Supports content-based routing. • Supports HTTP rewrites, redirects, overwrites, and throttling.	• Supports routing based on content and source IP addresses. • Supports HTTP rewrites, redirects, overwrites, throttling, cross-origin resource sharing (CORS), and session persistence. • Supports inbound and outbound forwarding rules.
Protocol	HTTP and HTTPS are supported.	• HTTP and HTTPS are supported. • Quick UDP Internet Connections (QUIC), WebSocket, WSS, and gRPC are supported.

Item	Nginx Ingress	ALB Ingress
Configuration change	• Processes are reloaded when you change the certificate. This may interrupt persistent connections. • Configuration changes other than certificate changes are performed by using hot updates based on Lua. • Processes are reloaded when you change the configuration of the Lua plug-in.	Allows you to change the configuration by calling API operations. This method is more efficient than using the list-watch mechanism to modify the configuration.
Authentication	• Supports Basic Auth-based authentication. • Supports the OAuth protocol.	Supports TLS-based authentication.
Performance	• Requires manual tuning to optimize system parameters and NGINX parameters. • Requires proper configurations on the number of replicated pods and the amount of resources.	• Supports one million QPS per instance. • Supports tens of millions of connections per instance. • Uses SSL hardware for acceleration.
Observability	• Allows you to collect the access log. • Allows you to configure Prometheus monitoring.	• Allows you to collect the access log by using Log Service. • Allows you to collect metrics by using CloudMonitor. • Allows you to configure alerting based on CloudMonitor.
O&M	• Supports manual O&M for the component. • Allows you to configure Horizontal Pod Autoscaler (HPA)-based scaling. • Allows you to specify computing resource specifications for optimization.	• Fully managed and O&M-free. • Supports auto scaling and automated configuration and provides ultra-large capacities. • Supports auto scaling for handling traffic spikes.

Item	Nginx Ingress	ALB Ingress
Security	• Supports HTTPS. • Supports blacklists and whitelists.	• Supports end-to-end data transfer over HTTPS, server Name Indication (SNI) for multiple certificates, Rivest-Shamir-Adleman (RSA) and elliptic-curve cryptography (ECC) certificates, TLS 1.3, and TLS cipher suites. • Supports WAF. • Supports Anti-DDoS. • Supports blacklists and whitelists.
Service governance	• Supports service discovery in Kubernetes clusters. • Supports canary releases. • Supports traffic throttling for high availability.	• Supports service discovery in Kubernetes clusters. • Supports canary releases. • Supports traffic throttling for high availability.
Extended features	Supports Lua for configuring extended features.	Supports AScript for configuring extended features. For more information, see Overview of AScript.
Cloud-native support	• Supports NGINX Service Mesh • A component that requires manual maintenance and can be used in ACK clusters and ASK clusters.	• Supports multiple cloud services, such as WAF, Function Compute, PrivateLink, and transit routers. • A managed component that can be used in Container Service for Kubernetes (ACK) clusters and serverless Kubernetes (ASK) clusters.

13.3. ALB Ingress Management

13.3.1. ALB Ingress overview

This topic introduces Ingresses and the Application Load Balancer (ALB) Ingress controller, and describes how the ALB Ingress controller works.

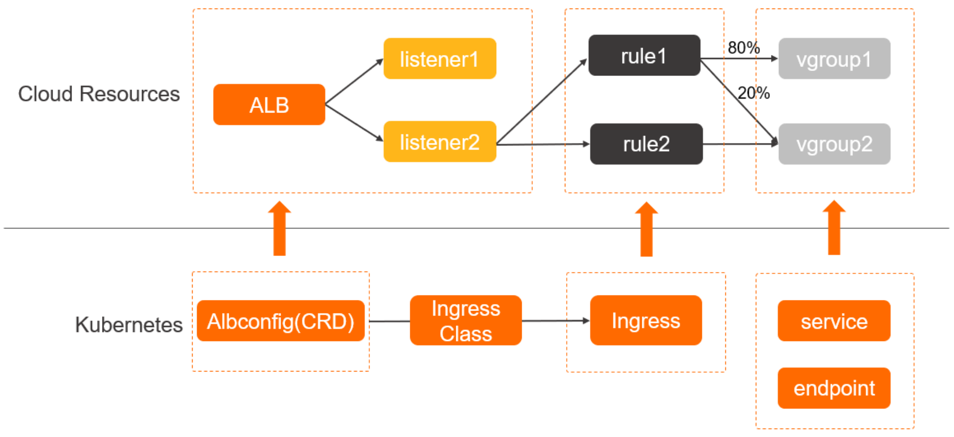
Introduction to Ingresses

In a Kubernetes cluster, an Ingress functions as an access point that exposes Services in the cluster. It distributes most of the network traffic that is destined for the Services in the cluster. An Ingress is a Kubernetes resource object that is used to enable external access to Services in a Kubernetes cluster. You can configure forwarding rules for an Ingress to route network traffic to backend pods of different Services.

How the ALB Ingress controller works

The ALB Ingress controller retrieves the changes to Ingresses from the API server and dynamically generates AlbConfig objects when Ingresses changes are detected. Then, the ALB Ingress controller performs the following operations in sequence: create ALB instances, configure listeners, create Ingress rules, and configure backend server groups. The Service, Ingress, and AlbConfig objects interact with each other in the following ways:

- A Service is an abstraction of an application that is deployed in a group of replicated pods.
- An Ingress contains reverse proxy rules. It controls to which Services HTTP or HTTPS requests are routed. For example, an Ingress routes requests to different Services based on the hosts and URLs in the requests.
- An AlbConfig object is a CustomResourceDefinition (CRD) object that the ALB Ingress controller uses to configure ALB instances and listeners. An AlbConfig object corresponds to one ALB instance.



How the ALB Ingress controller works

Notice ALB instances that serve Ingresses are fully managed by the ALB Ingress controller. To avoid service interruptions caused by Ingress errors, we recommend that you do not modify these ALB instances in the ALB console. For more information about the quotas related to ALB, see Limits.

The ALB Ingress controller is compatible with the NGINX Ingress controller, and provides improved traffic routing capabilities based on ALB instances. The ALB Ingress controller supports complex routing, automatic certificate discovery, and HTTP, HTTPS, and QUIC protocols. The ALB Ingress controller meets the requirements of cloud-native applications for ultra-high elasticity and balancing of heavy traffic loads at Layer 7.

References

- Access Services by using an ALB Ingress
- Manage the ALB Ingress controller

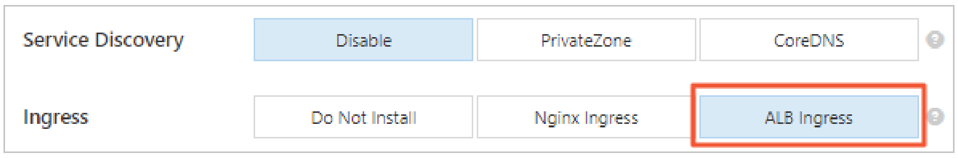
13.3.2. Manage the ALB Ingress controller

Serverless Kubernetes (ASK) clusters provide the managed Application Load Balancer (ALB) Ingress controller that implements Layer 7 forwarding rules based on ALB. This topic describes how to install the ALB Ingress controller in an ASK cluster and uninstall it from the cluster.

Install the ALB Ingress controller

Method 1: Install the ALB Ingress controller when you create an ASK cluster

When you create an ASK cluster, select **ALB Ingress** in the **Ingress** section. For more information, see ASK quick start.



Service Discovery: [✓ Disable] PrivateZone CoreDNS

Ingress: Do Not Install Nginx Ingress [✓ ALB Ingress]

Method 2: Install the ALB Ingress controller on the Add-ons page

1.
2.
3.
4.
5. Click the **Networking** tab, find **ALB Ingress Controller**, and then click **Install**.

> **Note** The ALB Ingress controller is not supported in the China (Hohhot), China (Heyuan), and UK (London) regions.

Uninstall the ALB Ingress controller

1.
2.
3.
4.
5. Click the **Networking** tab, find ALB Ingress Controller, and then click **Uninstall**.

13.3.3. Access Services by using an ALB Ingress

Application Load Balancer (ALB) Ingresses are compatible with NGINX Ingresses, and provide improved traffic routing capabilities based on ALB instances. ALB Ingresses support complex routing, automatic certificate discovery, and the HTTP, HTTPS, and Quick UDP Internet Connection (QUIC) protocols. ALB Ingresses meet the requirements of cloud-native applications for ultra-high elasticity and balancing of heavy traffic loads at Layer 7. This topic describes how to use an ALB Ingress to access Services.

Prerequisites

- A serverless Kubernetes (ASK) cluster whose Kubernetes version is 1.18 or later is created. A NAT gateway is configured for the virtual private cloud (VPC) where the cluster is deployed. This allows the cluster to download container images over the Internet. For more information, see ASK quick start.

- The kubectl client is connected to the ASK cluster. For more information, see Connect to an ACK cluster by using kubectl.

Context

An Ingress provides a collection of rules that manage external access to Services in a cluster. You can configure forwarding rules to assign Services different externally-accessible URLs. However, NGINX Ingresses and Layer 4 Server Load Balancer (SLB) Ingresses cannot meet the requirements of cloud-native applications, such as complex routing, multiple application layer protocols support (such as QUIC), and balancing of heavy traffic loads at Layer 7.

Step 1: Create an AlbConfig object

1. Create a file named *alb-test.yaml* and copy the following content into the file. The file is used to create an AlbConfig Object.

```
apiVersion: alibabacloud.com/v1
kind: AlbConfig
metadata:
  name: default
spec:
  config:
    name: alb-test
    addressType: Internet
    zoneMappings:
    - vSwitchId: vsw-uf6ccg2a9g71hx8go****
    - vSwitchId: vsw-uf6nun9tql5t8nh15****
```

Parameter	Description
spec.config.name	The name of the ALB. This parameter is optional.
addressType	The type of IP address that the ALB instance uses to provide services. This parameter is required. Valid values: ◦ Internet: The ALB instance uses a public IP address. The domain name of the Ingress is resolved to the public IP address of the ALB instance. Therefore, the ALB instance is accessible over the Internet. This is the default value. ◦ Intranet: The ALB instance uses a private IP address. The domain name of the Ingress is resolved to the private IP address of the ALB instance. Therefore, the ALB instance is accessible only within the virtual private cloud (VPC) where the ALB instance is deployed.

Parameter	Description
zoneMappings	The IDs of the vSwitches that are used by the ALB Ingress. You must specify at least two vSwitch IDs and the vSwitches must be deployed in different zones. This parameter is required. For more information about the regions and zones that are supported by ALB Ingresses, see Supported regions and zones.

2. Run the following command to create an AlbConfig object:

```
kubectl apply -f alb-test.yaml
```

Expected output:

```
AlbConfig.alibabacloud.com/default created
```

3. Create a file named *alb.yaml* and copy the following content into the file:

○Clusters that run Kubernetes versions earlier than 1.19 [✓ ◉Clusters that run Kubernetes 1.19 and later versions]

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: cafe-ingress
spec:
  ingressClassName: alb
  rules:
   - host: demo.domain.ingress.top
     http:
      paths:
      # Configure a context path.
      - path: /tea
        backend:
          serviceName: tea-svc
          servicePort: 80
      # Configure a context path.
      - path: /coffee
        backend:
          serviceName: coffee-svc
          servicePort: 80
```

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: cafe-ingress
spec:
  ingressClassName: alb
  rules:
   - host: demo.domain.ingress.top
     http:
      paths:
      # Configure a context path.
      - path: /tea
        pathType: ImplementationSpecific
        backend:
          service:
            name: tea-svc
            port:
              number: 80
      # Configure a context path.
      - path: /coffee
        pathType: ImplementationSpecific
        backend:
          service:
            name: coffee-svc
            port:
              number: 80
```

4. Run the following command to create an IngressClass:

   ```
   kubectl apply -f alb.yaml
   ```

   Expected output:

   ```
   ingressclass.networking.k8s.io/alb created
   ```

Step 2: Deploy applications

1. Create a *cafe-service.yaml* file and copy the following content into the file. The file is used to deploy two Deployments named `coffee` and `tea` and two Services named `coffee` and `tea`.

   ```
   apiVersion: apps/v1
   kind: Deployment
   metadata:
     name: coffee
   spec:
     replicas: 2
     selector:
       matchLabels:
         app: coffee
     template:
       metadata:
         labels:
           app: coffee
   ```

```
    spec:
      containers:
      - name: coffee
        image: registry.cn-hangzhou.aliyuncs.com/acs-sample/nginxdemos:latest
        ports:
        - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  name: coffee-svc
spec:
  ports:
  - port: 80
    targetPort: 80
    protocol: TCP
  selector:
    app: coffee
  clusterIP: None
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: tea
spec:
  replicas: 1
  selector:
    matchLabels:
      app: tea
  template:
    metadata:
      labels:
        app: tea
    spec:
      containers:
      - name: tea
        image: registry.cn-hangzhou.aliyuncs.com/acs-sample/nginxdemos:latest
        ports:
        - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  name: tea-svc
  labels:
spec:
  ports:
  - port: 80
    targetPort: 80
    protocol: TCP
  selector:
    app: tea
  clusterIP: None
```

2. Run the following command to deploy the Deployments and Services:

```
kubectl apply -f cafe-service.yaml
```

Expected output:

```
deployment "coffee" created
service "coffee-svc" created
deployment "tea" created
service "tea-svc" created
```

3. Run the following command to query the status of the Services that you created:

```
kubectl get svc,deploy
```

Expected output:

```
NAME              TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)    AGE
svc/coffee-svc    ClusterIP   <none>       <none>        80/TCP     1m
svc/tea-svc       ClusterIP   <none>       <none>        80/TCP     1m
NAME             DESIRED    CURRENT    UP-TO-DATE    AVAILABLE    AGE
deploy/coffee    2          2          2             2            1m
deploy/tea       1          1          1             1            1m
```

Step 3: Configure an Ingress

1. Create a *cafe-ingress.yaml* and copy the following content to the file:
2. Run the following command to configure an externally-accessible domain name and a `path` for the `coffee` and `tea` Services separately:

```
kubectl apply -f cafe-ingress.yaml
```

Expected output:

```
ingress "cafe-ingress" created
```

3. Run the following command to query the IP address of the ALB instance:

```
kubectl get ing
```

Expected output:

```
NAME           CLASS     HOSTS                          ADDRESS
PORTS    AGE
cafe-ingress   alb       demo.domain.ingress.top        alb-m551oo2zn63yov****.cn-hangzho
u.alb.aliyuncs.com   80       50s
```

Step 4: Access the service

- After you obtain the IP address of the ALB instance, use one of the following methods to access the `coffee` Service:
  - Access the `coffee` Service by using a browser.

- Access the `coffee` Service by using a CLI.

```
curl -H Host:demo.domain.ingress.top http://alb-lhwdm5c9h8lrcm****.cn-hangzhou.alb.aliy
uncs.com/coffee
```

- After you obtain the IP address of the ALB instance, use one of the following methods to access the `tea` Service:
  - Access the `tea` Service by using a browser.
  - Access the `tea` Service by using a CLI.

```
curl -H Host:demo.domain.ingress.top http://alb-lhwdm5c9h8lrcm****.cn-hangzhou.alb.aliy
uncs.com/tea
```

13.3.4. Advanced ALB Ingress configurations

An Ingress is an API object that you can use to provide Layer 7 load balancing to manage external access to Services in a serverless Kubernetes (ASK) cluster. This topic describes how to use ALB Ingresses to forward requests to backend server groups based on domain names and URL paths, redirect HTTP requests to HTTPS, and implement canary releases.

Prerequisites

- A serverless Kubernetes (ASK) cluster is created. You must configure a network address translation (NAT) gateway for the virtual private cloud (VPC) where the cluster is created so that the cluster can download container images from the Internet. For more information, see ASK quick start.
-
- For more information about how to create an AlbConfig object, see Create an AlbConfig.

Table of contents

This topic describes the following advanced ALB Ingress configurations:

Forward requests based on domain names

Perform the following steps to create a simple Ingress with or without a domain name to forward requests.

- Create a simple Ingress with a domain name.

i. Use the following template to create a Deployment, a Service, and an Ingress. Requests to the domain name of the Ingress are forwarded to the Service.

○ Clusters that run Kubernetes versions earlier than 1.19 ○ Clusters that run Kubernetes 1.19 and later versions

```
apiVersion: v1
kind: Service
metadata:
  name: demo-service
  namespace: default
spec:
  ports:
    - name: port1
      port: 80
      protocol: TCP
      targetPort: 8080
  selector:
    app: demo
  sessionAffinity: None
  type: ClusterIP
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: demo
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: demo
  template:
    metadata:
      labels:
        app: demo
    spec:
      containers:
        - image: registry.cn-hangzhou.aliyuncs.com/alb-sample/cafe:v1
          imagePullPolicy: IfNotPresent
          name: demo
          ports:
            - containerPort: 8080
              protocol: TCP
---
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: demo
  namespace: default
spec:
  ingressClassName: alb
  rules:
    - host: demo.domain.ingress.top  # The domain name of the Ingress.
      http:
        paths:
          - backend:
              serviceName: demo-service
              servicePort: 80
            path: /hello
            pathType: ImplementationSpecific
```

```
apiVersion: v1
kind: Service
metadata:
  name: demo-service
  namespace: default
spec:
  ports:
    - name: port1
      port: 80
      protocol: TCP
      targetPort: 8080
  selector:
    app: demo
  sessionAffinity: None
  type: ClusterIP
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: demo
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: demo
  template:
    metadata:
      labels:
        app: demo
    spec:
      containers:
        - image: registry.cn-hangzhou.aliyuncs.com/alb-sample/cafe:v1
          imagePullPolicy: IfNotPresent
          name: demo
          ports:
            - containerPort: 8080
              protocol: TCP
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: demo
  namespace: default
spec:
  ingressClassName: alb
  rules:
```

```
  - host: demo.domain.ingress.top
    http:
      paths:
       - backend:
           service:
            name: demo-service
             port:
               number: 80
         path: /hello
         pathType: ImplementationSpecific
```

ii.

-

Configure automatic certificate discovery

The ALB Ingress controller supports automatic certificate discovery. You must first create a certificate in the Certificate Management Service console. Then, specify the domain name of the certificate in the Transport Layer Security (TLS) configurations of the Ingress. This way, the ALB Ingress controller can automatically match and discover the certificate based on the TLS configurations of the Ingress.

1.
2.
3. Add the following setting to the YAML template of the Ingress to specify the domain name in the certificate that you created:

```
tls:
  - hosts:
    - demo.alb.ingress.top
```

Examples

○ Clusters that run Kubernetes versions earlier than 1.19 [✓ Clusters that run Kubernetes 1.19 and later versions]

```
apiVersion: v1
kind: Service
metadata:
  name: demo-service-https
  namespace: default
spec:
  ports:
    - name: port1
      port: 443
      protocol: TCP
      targetPort: 8080
  selector:
    app: demo-cafe
  sessionAffinity: None
  type: ClusterIP
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: demo-cafe
  name: demo-cafe
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: demo-cafe
  template:
    metadata:
      labels:
        app: demo-cafe
    spec:
      containers:
        - image: registry.cn-hangzhou.aliyuncs.com/alb-sample/cafe:v1
          imagePullPolicy: IfNotPresent
          name: demo-cafe
          ports:
            - containerPort: 8080
              protocol: TCP
---
apiVersion: networking.k8s.io/v1beta1
 kind: Ingress
 metadata:
   name: demo-https
   namespace: default
 spec:
   ingressClassName: alb
   tls:
   - hosts:
     - demo.alb.ingress.top
   rules:
     - host: demo.alb.ingress.top
       http:
         paths:
           - backend:
               serviceName: demo-service-https
               servicePort: 443
             path: /
             pathType: Prefix
```

```
apiVersion: v1
kind: Service
metadata:
  name: demo-service-https
  namespace: default
spec:
  ports:
    - name: port1
      port: 443
      protocol: TCP
      targetPort: 8080
  selector:
    app: demo-cafe
  sessionAffinity: None
  type: ClusterIP
  type: ClusterIP
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: demo-cafe
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: demo-cafe
  template:
    metadata:
      labels:
        app: demo-cafe
    spec:
      containers:
        - image: registry.cn-hangzhou.aliyuncs.com/alb-sample/cafe:v1
          imagePullPolicy: IfNotPresent
          name: demo-cafe
          ports:
            - containerPort: 8080
              protocol: TCP
---
apiVersion: networking.k8s.io/v1
 kind: Ingress
 metadata:
   name: demo-https
   namespace: default
 spec:
   ingressClassName: alb
   tls:
   - hosts:
     - demo.alb.ingress.top
   rules:
     - host: demo.alb.ingress.top
       http:
         paths:
           - backend:
               service:
                 name: demo-service-https
                 port:
                   number: 443
             path: /
             pathType: Prefix
```

4. Run the following command to query the certificate:

```
curl https://demo.alb.ingress.top/tea
```

Expected output:

```
null
```

Redirect HTTP requests to HTTPS

To redirect HTTP requests to HTTPS, you can add the `alb.ingress.kubernetes.io/ssl-redirect: "true"` annotation to the ALB Ingress configurations. This way, HTTP requests are redirected to HTTPS port 443.

Example:

○ Clusters that run Kubernetes versions earlier than 1.19 [✓ Clusters that run Kubernetes 1.19 and later versions]

```
apiVersion: v1
kind: Service
metadata:
  name: demo-service-ssl
  namespace: default
spec:
  ports:
    - name: port1
      port: 80
      protocol: TCP
      targetPort: 8080
  selector:
    app: demo-ssl
  sessionAffinity: None
  type: ClusterIP
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: demo-ssl
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: demo-ssl
  template:
    metadata:
      labels:
        app: demo-ssl
    spec:
      containers:
        - image: registry.cn-hangzhou.aliyuncs.com/alb-sample/cafe:v1
          imagePullPolicy: IfNotPresent
          name: demo-ssl
          ports:
            - containerPort: 8080
              protocol: TCP
---
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  annotations:
    alb.ingress.kubernetes.io/ssl-redirect: "true"
  name: demo-ssl
  name: demo-ssl
  namespace: default
spec:
  ingressClassName: alb
  tls:
  - hosts:
    - ssl.alb.ingress.top
  rules:
    - host: ssl.alb.ingress.top
      http:
        paths:
          - backend:
              serviceName: demo-service-ssl
              servicePort: 80
            path: /
            pathType: Prefix
```

```
apiVersion: v1
kind: Service
metadata:
  name: demo-service-ssl
  namespace: default
spec:
  ports:
    - name: port1
      port: 80
      protocol: TCP
      targetPort: 8080
  selector:
    app: demo-ssl
  sessionAffinity: None
  type: ClusterIP
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: demo-ssl
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: demo-ssl
  template:
    metadata:
      labels:
        app: demo-ssl
    spec:
      containers:
        - image: registry.cn-hangzhou.aliyuncs.com/alb-sample/cafe:v1
          imagePullPolicy: IfNotPresent
          name: demo-ssl
          ports:
            - containerPort: 8080
```

```
              protocol: TCP
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  annotations:
    alb.ingress.kubernetes.io/ssl-redirect: "true"
  name: demo-ssl
  namespace: default
spec:
  ingressClassName: alb
  tls:
  - hosts:
    - ssl.alb.ingress.top
  rules:
    - host: ssl.alb.ingress.top
      http:
        paths:
          - backend:
              service:
                name: demo-service-ssl
                port:
                  number: 80
            path: /
            pathType: Prefix
```

13.4. SLB Ingress management

13.4.1. Overview

This topic introduces Ingresses and describes how Ingress controllers and Server Load Balancer (SLB) Ingress controllers work.

Note SLB Ingress controllers will soon be discontinued. We recommend that you use Application Load Balancer (ALB) Ingress controllers. For more information, see ALB Ingress overview.

Introduction to Ingresses

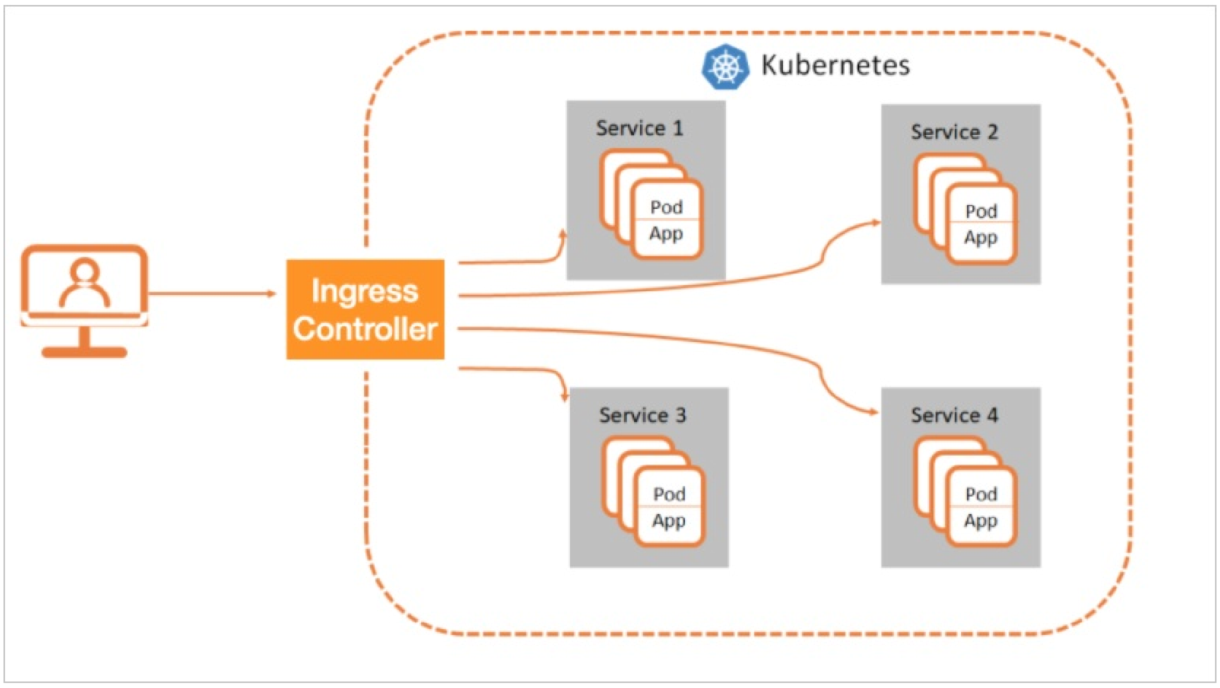
How an Ingress controller works

To ensure that the Ingress resource in a cluster works as expected, you must deploy an Ingress controller in the cluster to parse the Ingress rules. After an Ingress controller receives a request that matches an Ingress rule, the Ingress controller routes the request to the corresponding Service. Then, the Service forwards the request to pods and the pods process the request. In a Kubernetes cluster, Services, Ingresses, and Ingress controllers work in the following process:

- A Service is an abstraction of an application that is deployed in a group of replicated pods.
- An Ingress contains reverse proxy rules. It controls to which Services HTTP or HTTPS requests are routed. For example, an Ingress routes requests to different Services based on the hosts and URLs in the requests.

- An Ingress controller is a reverse proxy program that parses Ingress rules. If changes are made to the Ingress rules, the Ingress controller updates the Ingress rules accordingly. After an Ingress controller receives a request, it redirects the request to a Service based on the Ingress rules.

The changes to Ingresses in a Kubernetes cluster are updated to Ingress controllers through the cluster API server. The configurations of SLB instances are dynamically generated based on these changes. Then, Ingress rules are updated based on the configurations.



How an SLB Ingress controller works

Note SLB Ingress controllers will soon be discontinued. We recommend that you use ALB Ingress controllers. For more information, see ALB Ingress overview.

Serverless Kubernetes (ASK) clusters provide fully managed SLB Ingress controllers based on the forwarding capability of SLB instances at Layer 7. This reduces the Infrastructure as a service (IaaS) cost and provides Ingresses with the disaster recovery capability that is intended for cloud services.

- If you do not specify an SLB instance, ASK automatically creates an Internet-facing SLB instance. ASK does not create an internal-facing SLB instance for an ASK cluster.
- You can use existing Internet-facing or internal-facing SLB instances only by specifying instance IDs in annotations.
    - If no existing Internet-facing SLB instance is available, you can create one. Then, specify the ID of the newly created SLB instance in annotations.
    - If no existing internal-facing SLB instance is available, you can create one. Then, specify the ID of the newly created SLB instance in annotations.
- If you use an existing SLB instance, it must be a high-performance SLB instance that supports elastic network interfaces (ENIs). In addition, make sure that port 80 and port 443 are not used by other Services.
- SLB instances listen on port 80 for HTTP requests and listen on port 443 for HTTPS requests.
- SLB instances automatically use the certificate stored in the **Secret** that is specified in the

configurations of the first Ingress created in the cluster. If no **Secret** is specified in the configurations of this Ingress, the default **fake** certificate is used.

Notice SLB instances that serve Ingresses are fully managed by SLB Ingress controllers. To avoid service interruptions caused by Ingress errors, we recommend that you do not modify these SLB instances in the SLB console.

Related information

- Overview

13.4.2. Manage SLB Ingress controllers

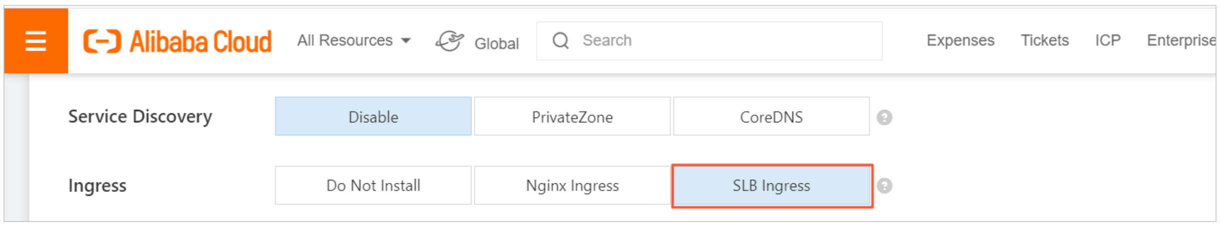
Serverless Kubernetes (ASK) clusters provide managed Server Load Balancer (SLB) Ingress controllers that can route network traffic based on Layer 7 forwarding rules. This topic describes how to install an SLB Ingress controller in an ASK cluster and how to uninstall an SLB Ingress controller from an ASK cluster.

Note SLB Ingress controllers will soon be discontinued. We recommend that you use Application Load Balancer (ALB) Ingress controllers. For more information, see ALB Ingress overview.

Install an SLB Ingress controller

Method 1: Install the SLB Ingress controller when you create an ASK cluster

When you create an ASK cluster, select **SLB Ingress** in the **Ingress** section. For more information, see ASK quick start.



Service Discovery: Disable | PrivateZone | CoreDNS

Ingress: Do Not Install | Nginx Ingress | SLB Ingress

Method 2: Install the SLB Ingress controller from the Add-ons page

1.
2.
3.
4.
5. Click the **Others** tab, find **SLB Ingress Controller**, and then click **Install**.

Uninstall the SLB Ingress controller

1.
2.
3.
4.
5. Click the **Others** tab, find SLB Ingress Controller, and then click **Uninstall**.

Delete the Service that is associated with the SLB instance

1.
2. On the **Services** page, select **kube-system** from the **Namespace** drop-down list.
3. Find the **alb-ingress-lb** Service and click **Delete** in the Actions column.
4. In the **Tips** dialog box, click **Confirm**.

13.4.3. Use the automatically created SLB instance

If you do not use an annotation to specify an existing Server Load Balancer (SLB) instance when you create an Ingress, the system automatically creates an Internet-facing SLB instance. This topic describes how to use the automatically created SLB instance with an Ingress to set up forwarding.

Prerequisites

- A serverless Kubernetes (ASK) cluster is created. You must configure a network address translation (NAT) gateway for the virtual private cloud (VPC) where the cluster is created so that the cluster can download container images from the Internet. For more information, see ASK quick start.
- The kubectl client is connected to the ASK cluster. For more information, see Connect to an ACK cluster by using kubectl.

Procedure

Step 1: Deploy applications

1. Create the *cafe-service.yaml* file, copy the following code to the file, and run the `kubectl apply -f cafe-service.yaml` command to deploy a **coffee** application and a **tea** application:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: coffee
spec:
  replicas: 2
  selector:
    matchLabels:
      app: coffee
  template:
    metadata:
      labels:
        app: coffee
    spec:
      containers:
      - name: coffee
        image: registry.cn-hangzhou.aliyuncs.com/acs-sample/nginxdemos:latest
        ports:
        - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  name: coffee-svc
spec:
  ports:
  - port: 80
    targetPort: 80
    protocol: TCP
  selector:
    app: coffee
  clusterIP: None
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: tea
spec:
  replicas: 1
  selector:
    matchLabels:
      app: tea
  template:
    metadata:
      labels:
        app: tea
    spec:
      containers:
      - name: tea
        image: registry.cn-hangzhou.aliyuncs.com/acs-sample/nginxdemos:latest
        ports:
        - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  name: tea-svc
  labels:
spec:
  ports:
  - port: 80
    targetPort: 80
    protocol: TCP
  selector:
    app: tea
  clusterIP: None
```

The following output is returned:

```
deployment "coffee" created
service "coffee-svc" created
deployment "tea" created
service "tea-svc" created
```

2. Run the following command to view the application status:

```
kubectl get svc,deploy
```

The following output is returned:

```
NAME              TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
svc/coffee-svc    ClusterIP   <none>       <none>        80/TCP    1m
svc/tea-svc       ClusterIP   <none>       <none>        80/TCP    1m
NAME             DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
deploy/coffee    2         2         2            2           1m
deploy/tea       1         1         1            1           1m
```

Step 2: Configure an Ingress

1. Create the *cafe-ingress.yaml* file, copy the following code to the file, and run the `kubectl apply -f cafe-ingress.yaml` command to configure an Ingress. The Ingress specifies the domain name and path that are used to expose the **coffee** and **tea** applications to the Internet.

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: cafe-ingress
spec:
  ingressClassName: slb
  rules:
  # Configure a Layer 7 domain name.
  - host: foo.bar.com
    http:
      paths:
      # Configure a context path.
      - path: /tea
        backend:
          service:
            name: tea-svc
            port:
              number: 80
        pathType: ImplementationSpecific
      # Configure a context path.
      - path: /coffee
        backend:
          service:
            name: coffee-svc
            port:
              number: 80
        pathType: ImplementationSpecific
```

The following output is returned:

```
ingress "cafe-ingress" created
```

2. Run the following command to obtain the IP address of the SLB instance:

```
kubectl get ing
```

The following output is returned:

```
NAME           HOSTS         ADDRESS           PORTS     AGE
cafe-ingress   foo.bar.com   139.168.XX.XX     80        1m
```

Step 3: Access the applications

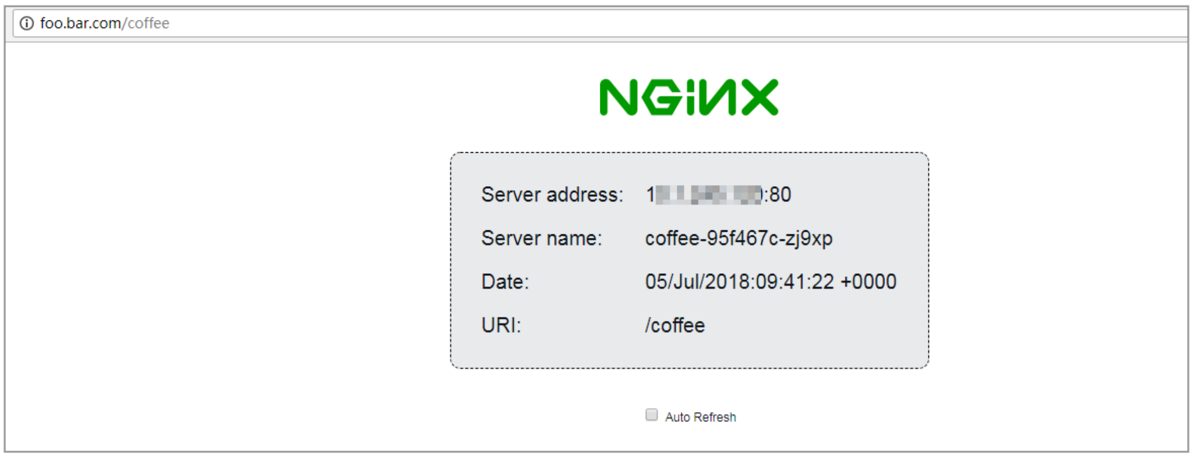
Note You must resolve the domain name to the IP address of the SLB instance.

In this example, the following DNS rule is added for the domain name to enable access to the applications. We recommend that you apply for an Internet Content Provider (ICP) number for the domain name if the domain name is used in the production environment.

```
139.168.XX.XX     foo.bar.com
```

Access the **coffee** application.

- Access the **coffee** application by using a browser.



foo.bar.com/coffee

NGINX

Server address: 1[redacted]:80
Server name: coffee-95f467c-zj9xp
Date: 05/Jul/2018:09:41:22 +0000
URI: /coffee

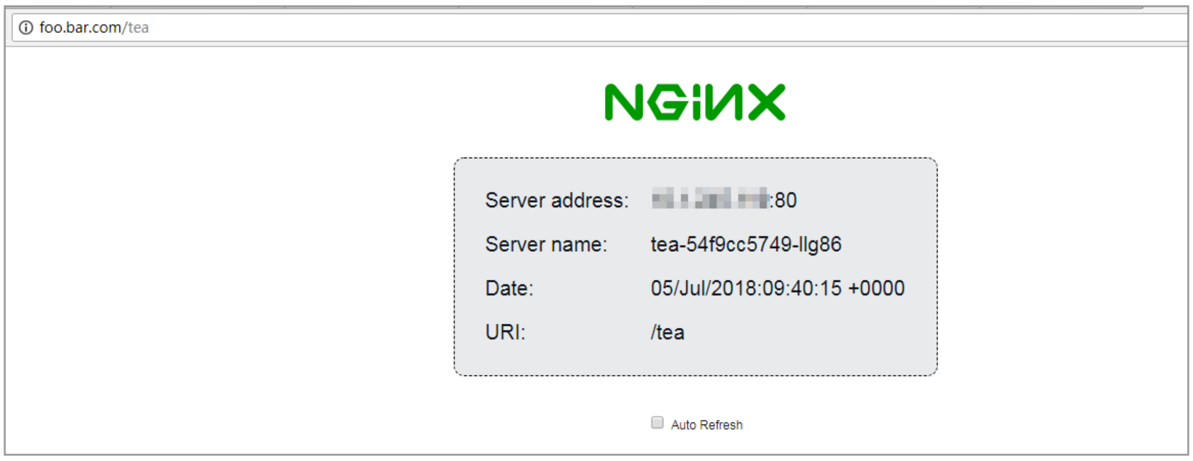
Auto Refresh

- Access the **coffee** application by using the command line tool.

```
curl -H "Host: foo.bar.com" http://139.168.XX.XX/coffee
```

Access the **tea** application.

- Access the **tea** application by using a browser.



foo.bar.com/tea

NGINX

Server address: [redacted]:80
Server name: tea-54f9cc5749-llg86
Date: 05/Jul/2018:09:40:15 +0000
URI: /tea

Auto Refresh

- Access the **tea** application by using the command line tool.

```
curl -H "Host: foo.bar.com" http://139.168.XX.XX/tea
```

13.4.4. Use an existing SLB instance

This topic describes how to use an existing Server Load Balancer (SLB) instance with an Ingress to set up forwarding. You can use an annotation to specify an SLB instance by ID when you create an Ingress.

Prerequisites

- A serverless Kubernetes (ASK) cluster is created. You must configure a network address translation (NAT) gateway for the virtual private cloud (VPC) where the cluster is created so that the cluster can download container images from the Internet. For more information, see ASK quick start.
- The kubectl client is connected to the ASK cluster. For more information, see Connect to an ACK cluster by using kubectl.
- A high-performance SLB instance is created in the virtual private cloud (VPC) where your serverless Kubernetes (ASK) cluster is deployed. High-performance SLB instances support elastic network interfaces (ENIs).
  - If you have an SLB instance in the VPC where your ASK cluster is deployed, you can log on to the SLB console and obtain the ID of the SLB instance on the **Instances** page.
  - If you have no SLB instance, you must create a high-performance SLB instance in the VPC where your ASK cluster is deployed. For example, you can create a high-performance SLB instance of the *slb.s2.small* instance type. The SLB instance can be internal-facing or Internet-facing. For more information, see Create a CLB instance.
  - In this example, an Internet-facing SLB instance is used.

Procedure

Note The Ingress controller automatically opens port 80 and 443 on the SLB instance. Make sure that port 80 and 443 are not used by other services.

Step 1: Deploy an application

1. Create a *tomcat-service.yml* file and copy the following code to the file, Then, run the `kubectl apply -f tomcat-service.yml` command to deploy a **tomcat** application for testing:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: tomcat
spec:
  replicas: 1
  selector:
    matchLabels:
      run: tomcat
  template:
    metadata:
      labels:
        run: tomcat
    spec:
      containers:
      - image: tomcat:7.0
        imagePullPolicy: Always
        name: tomcat
        ports:
        - containerPort: 8080
          protocol: TCP
      restartPolicy: Always
---
apiVersion: v1
kind: Service
metadata:
  name: tomcat
spec:
  ports:
  - port: 8080
    protocol: TCP
    targetPort: 8080
  selector:
    run: tomcat
  clusterIP: None
```

Expected output:

```
deployment "tomcat" created
service "tomcat" created
```

2. Run the following command to query the status of the application:

```
kubectl get svc,deploy tomcat
```

Expected output:

```
NAME          TYPE        CLUSTER-IP    EXTERNAL-IP    PORT(S)    AGE
svc/tomcat    ClusterIP   <none>        <none>         8080/TCP   1m
NAME             DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
deploy/tomcat    1         1         1            1           1m
```

Step 2: Create an Ingress

1. Create a *tomcat-ingress.yml* file and copy the following code to the file. Then, run the `kubectl a`

`pply -f tomcat-ingress.yml` command to create an Ingress:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: tomcat-ingress
  annotations:
    # Specify the ID of the existing SLB instance that you want to use.
    service.beta.kubernetes.io/alicloud-loadbalancer-id: lb-xxxxxxxxxx            ##Rep
lace lb-xxxxxxxxxx with the ID of your SLB instance.
    service.beta.kubernetes.io/alicloud-loadbalancer-force-override-listeners: "true"
    The ID of the certificate that you want to use.
    service.beta.kubernetes.io/alicloud-loadbalancer-cert-id: "624f2e60-62b6-11ea-95a3-
2af160c0****"
spec:
  ingressClassName: slb
  rules:
  # Configure a Layer 7 domain name.
  - host: bar.foo.com
    http:
      paths:
      # Configure a context path.
      - path: /
        backend:
          service:
            name: tomcat
            port:
              number: 8080
        pathType: ImplementationSpecific
```

Expected output:

```
ingress "tomcat-ingress" created
```

2. Run the following command to obtain the IP address of the SLB instance:

```
kubectl get ing tomcat-ingress
```

Expected output:

```
NAME              HOSTS          ADDRESS         PORTS     AGE
tomcat-ingress    bar.foo.com    47.168.XX.XX    80, 443   1m
```

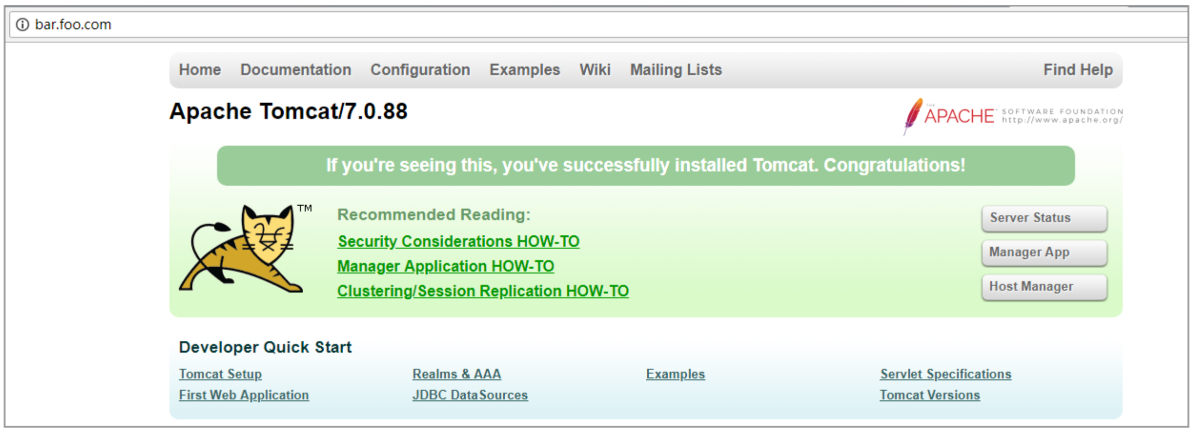
Step 3: Access the application

Note You must resolve the domain name to the IP address of the SLB instance.

In this example, the following DNS rule is created for the domain name to enable access to the test application. We recommend that you apply for an Internet Content Provider (ICP) number for the domain name if the domain name is used in the production environment.

```
47.168.XX.XX    bar.foo.com
```

- Access the **tomcat** application by using a browser.



- Access the **tomcat** application by using the CLI.

```
curl -k -H "Host: bar.foo.com" https://47.168.XX.XX
```

13.4.5. Use a Secret to configure TLS to enable HTTPS access

This topic describes how to use a Secret to configure a TLS certificate to enable HTTPS access to an Ingress.

Prerequisites

- A serverless Kubernetes (ASK) cluster is created. You must configure a network address translation (NAT) gateway for the virtual private cloud (VPC) where the cluster is created so that the cluster can download container images from the Internet. For more information, see ASK quick start.
- The kubectl client is connected to the ASK cluster. For more information, see Connect to an ACK cluster by using kubectl.

Procedure

Step 1: Deploy applications

1. Create the *cafe-service.yaml* file, copy the following code to the file, and run the `kubectl apply -f cafe-service.yaml` command to deploy a **coffee** application and a **tea** application:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: coffee
spec:
  replicas: 2
  selector:
    matchLabels:
      app: coffee
  template:
    metadata:
      labels:
        app: coffee
    spec:
      containers:
```

```
      - name: coffee
        image: registry.cn-hangzhou.aliyuncs.com/acs-sample/nginxdemos:latest
        ports:
        - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  name: coffee-svc
spec:
  ports:
  - port: 80
    targetPort: 80
    protocol: TCP
  selector:
    app: coffee
  clusterIP: None
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: tea
spec:
  replicas: 1
  selector:
    matchLabels:
      app: tea
  template:
    metadata:
      labels:
        app: tea
    spec:
      containers:
      - name: tea
        image: registry.cn-hangzhou.aliyuncs.com/acs-sample/nginxdemos:latest
        ports:
        - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  name: tea-svc
  labels:
spec:
  ports:
  - port: 80
    targetPort: 80
    protocol: TCP
  selector:
    app: tea
  clusterIP: None
```

The following output is returned:

```
deployment "coffee" created
service "coffee-svc" created
deployment "tea" created
service "tea-svc" created
```

2. Run the following command to view the application status:

   ```
   kubectl get svc,deploy
   ```

   The following output is returned:

   ```
   NAME              TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
   svc/coffee-svc    ClusterIP   <none>       <none>        80/TCP    1m
   svc/tea-svc       ClusterIP   <none>       <none>        80/TCP    1m
   NAME             DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
   deploy/coffee    2         2         2            2           1m
   deploy/tea       1         1         1            1           1m
   ```

Step 2: Configure a TLS certificate

You must configure a TLS certificate to enable HTTPS access.

1. Run the following command to generate a TLS certificate:

   ```
   openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout tls.key -out tls.crt -subj
   "/CN=bar.foo.com/O=bar.foo.com"
   ```

   The following output is returned:

   ```
   Generating a 2048 bit RSA private key
   ..........................+++
   ....................................................................+++
   writing new private key to 'tls.key'
   -----
   ```

2. Run the following command to create a **Secret** based on the TLS certificate that you generated:

   ```
   kubectl create secret tls cert-example --key tls.key --cert tls.crt
   ```

   The following output is returned:

   ```
   secret/cert-example created
   ```

3. Run the following command to view the Secret that contains the TLS certificate information:

   ```
   kubectl get secret cert-example
   ```

   The following output is returned:

   ```
   NAME           TYPE                DATA      AGE
   cert-example   kubernetes.io/tls   2         12s
   ```

Notice

- The Ingress controller automatically initializes the default HTTPS certificate for the SLB instance based on the first TLS certificate of the Ingress. You can modify the certificate only by modifying the **Secret** that is referenced by the Ingress. You cannot modify the certificate by using the SLB console. You cannot configure multiple certificates. If you want to use multiple certificates for multiple domains, you can reuse SLB instances and specify an SLB instance for each certificate.
- Modifications in the SLB console may cause errors to the Ingress. Do not modify the configurations by using the SLB console.

Step 3: Configure an Ingress

1. Create the *cafe-ingress.yaml* file, copy the following code to the file, and run the `kubectl apply -f cafe-ingress.yaml` command to configure an Ingress. The Ingress specifies the domain name and path that are used to expose the **coffee** and **tea** applications to the Internet.

```
# cat  cafe-ingress.yaml
apiVersion: extensions/v1
kind: Ingress
metadata:
  name: cafe-ingress
spec:
  tls:
  - hosts:
    - foo.bar.com
    secretName: cert-example
  rules:
  # Configure a Layer 7 domain name.
  - host: foo.bar.com
    http:
      paths:
      # Configure a context path.
      - path: /tea
        backend:
          service:
            name: tea-svc
            port:
              number: 80
        pathType: ImplementationSpecific
      # Configure a context path.
      - path: /coffee
        backend:
          service:
            name: coffee-svc
            port:
              number: 80
        pathType: ImplementationSpecific
```

The following output is returned:

```
ingress "cafe-ingress" created
```

2. Run the following command to obtain the IP address of the SLB instance:

   ```
   kubectl get ing
   ```

   The following output is returned:

   ```
   NAME            HOSTS          ADDRESS          PORTS      AGE
   cafe-ingress    foo.bar.com    139.168.XX.XX    80         1m
   ```

Step 4: Access the applications

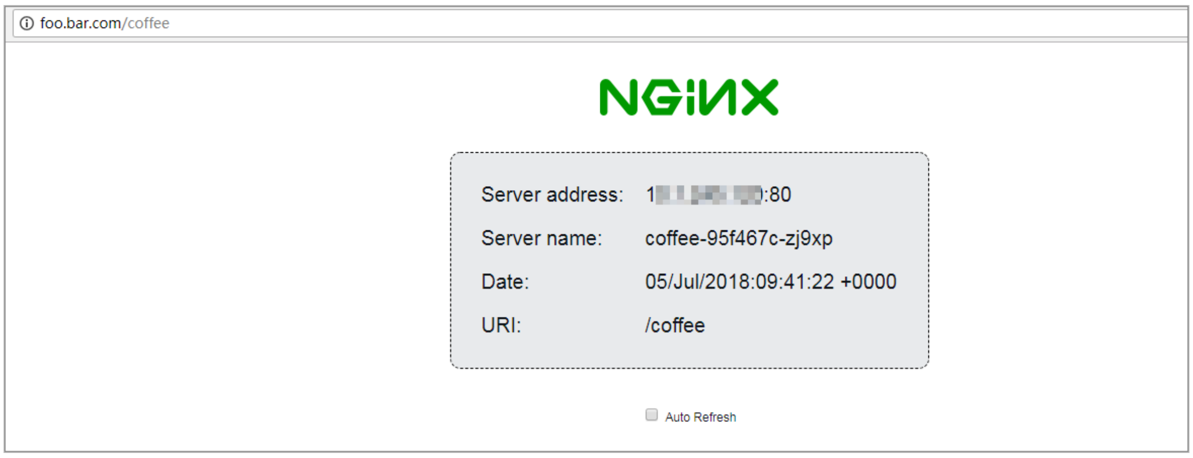
Note You must resolve the domain name to the IP address of the SLB instance.

In this example, the following DNS rule is added for the domain name to enable access to the applications. We recommend that you apply for an Internet Content Provider (ICP) number for the domain name if the domain name is used in the production environment.

```
139.168.XX.XX    foo.bar.com
```

Access the **coffee** application.

- Access the **coffee** application by using a browser.


foo.bar.com/coffee

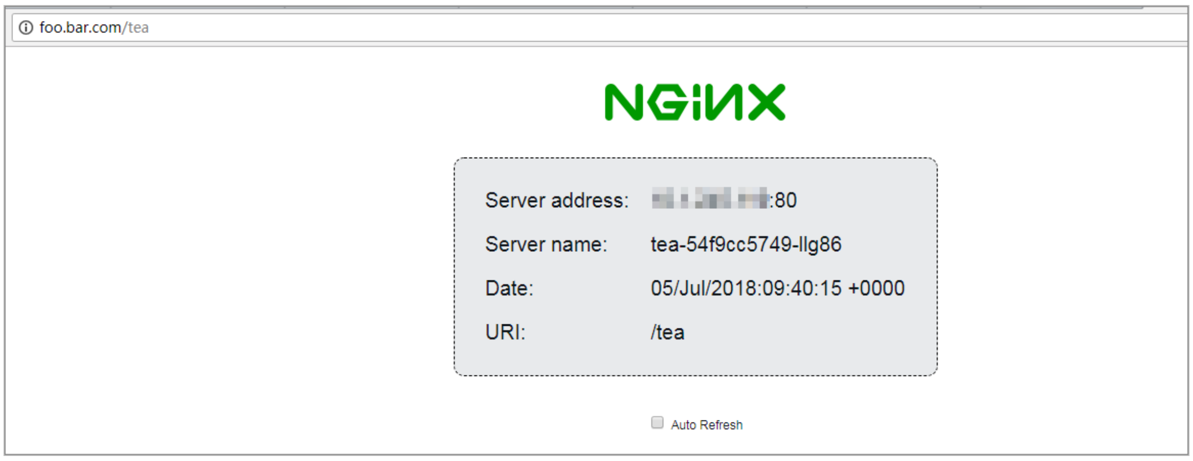
Server address: 1[redacted]:80
Server name: coffee-95f467c-zj9xp
Date: 05/Jul/2018:09:41:22 +0000
URI: /coffee

- Access the **coffee** application by using the command line tool.

```
curl -H "Host: foo.bar.com" http://139.168.XX.XX/coffee
```

Access the **tea** application.

- Access the **tea** application by using a browser.


foo.bar.com/tea

Server address: [redacted]:80
Server name: tea-54f9cc5749-llg86
Date: 05/Jul/2018:09:40:15 +0000
URI: /tea

- Access the **tea** application by using the command line tool.

```
curl -H "Host: foo.bar.com" https://139.168.XX.XX/tea -k
```

13.5. NGINX Ingress management

13.5.1. Overview

This topic introduces Ingresses and describes how Ingress controllers work. It also introduces Ingresses in serverless Kubernetes (ASK) clusters.

Introduction to Ingress

How an Ingress controller works

Ingresses in ASK clusters

ASK clusters provide NGINX Ingress controllers that are optimized on top of the open source version. You can choose to install an NGINX Ingress controller when you create an ASK cluster. You can also manually install it on the Add-ons page in the console after the cluster is created.

Related information

- Create an Ingress
- Advanced Ingress configurations
- Overview

13.5.2. Install the NGINX Ingress controller

Alibaba Cloud serverless Kubernetes (ASK) clusters provide the NGINX Ingress controller, which is optimized based on the open source NGINX Ingress controller. This topic describes how to install the NGINX Ingress controller in an ASK cluster.

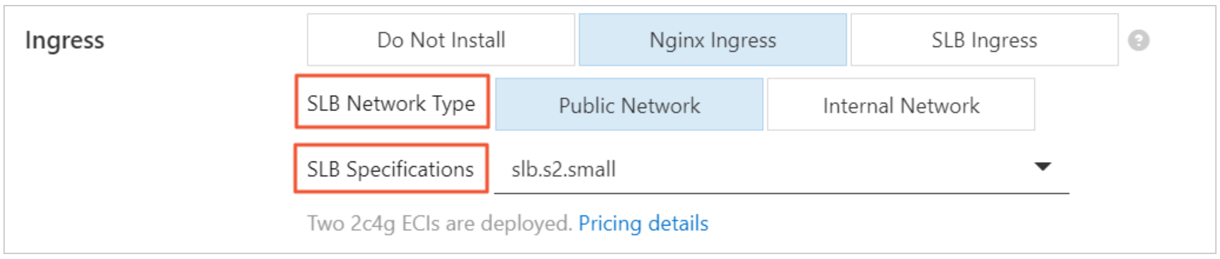
Procedure

Method 1: Install the NGINX Ingress controller when you create a cluster

When you create an ASK cluster, select **Nginx Ingress** in the **Ingress** section. For more information, see ASK quick start.

Note To install the NGINX Ingress controller, the following two conditions must be met:

- Two elastic container instances that have at least 2 CPU cores and 4 GiB of memory each are used to deploy services.
- One Server Load Balancer (SLB) instance is created. You can set the network type of the SLB instance to Public Network or Internal Network and select the SLB instance specifications based on your business requirements.



Method 2: Install the NGINX Ingress controller on the Add-ons page

1.
2.
3.
4.
5. Click the **Networking** tab. Select **Nginx Ingress Controller** and click **Install**.

13.5.3. Create an Ingress

An Ingress is a Kubernetes resource object that is used to enable external access to Services in a Kubernetes cluster. Container Service for Kubernetes (ACK) allows you to use Ingresses to configure multiple forwarding rules for handling requests to pods in a Kubernetes cluster. This topic describes how to create, view, update, and delete an Ingress in the ACK console or by using kubectl.

Prerequisites

Create an ACK managed cluster

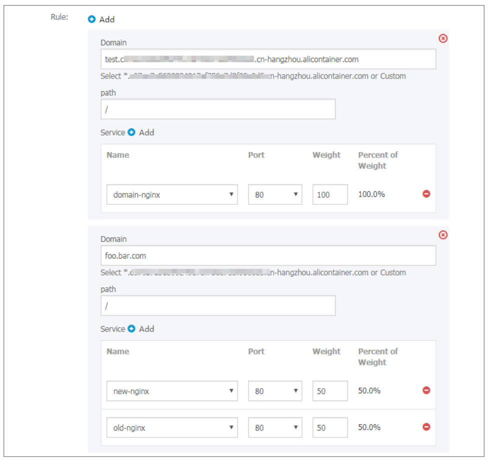
Manage Ingresses in the ACK console

Create an Ingress

1.
2.
3.
4.
5. On the **Ingresses** page, click **Create**. In the **Create** dialog box, set the name of the Ingress. In this example, the Ingress is named nginx-ingress.
6. Configure Ingress rules.

   Ingress rules are used to manage external access to Services in the cluster. Ingress rules can be HTTP or HTTPS rules. You can configure the following items in Ingress rules: domain name (virtual hostname), URL path, Service name, port, and weight.

In this example, a complex rule is added to configure Services for the default domain name and virtual hostname of the cluster. Traffic is routed based on domain names.



Rule: Add

Domain

test.c[redacted].cn-hangzhou.alicontainer.com

Select *.[redacted]n-hangzhou.alicontainer.com or Custom

path

/

Service Add

Name	Port	Weight	Percent of Weight
domain-nginx	80	100	100.0%

Domain

foo.bar.com

Select *.[redacted]n-hangzhou.alicontainer.com or Custom

path

/

Service Add

Name	Port	Weight	Percent of Weight
new-nginx	80	50	50.0%
old-nginx	80	50	50.0%

- Create a simple Ingress that uses the default domain name to provide external access.
  - Domain: Enter the default domain name of the cluster. In this example, the default domain name is `test.[cluster-id].[region-id].alicontainer.com` .

    In the Create dialog box, the default domain name of the cluster is displayed in the format of `*.[cluster-id].[region-id].alicontainer.com` . You can also obtain the default domain name from the details page of the cluster.
  - Services: Set the names, paths, and port numbers of the backend Services that you want to access.
    - Path: You can enter the URL for accessing the backend Services. The default path is the root path /. In this example, the default path is used. Each path is associated with a backend Service. Server Load Balancer (SLB) forwards traffic to a backend Service only when inbound requests match the domain name and path.
    - Services: Set the names, port numbers, and weights of the backend Services that you want to access. You can specify the same path for multiple Services. The Ingress then distributes network traffic to these Services based on the Service weights.
- Create a simple fanout Ingress that uses multiple domain names. In this example, a virtual hostname is used as the test domain name to provide external access. Route weights are specified for two backend Services and canary release settings are configured for one of the Services. In your production environment, you can use a domain name that has obtained an Internet Content Provider (ICP) number to provide external access.
  - Domain: Enter the test domain name. In this example, the test domain name is `foo.bar.com` .

    You must add the following domain name mapping rule to the hosts file:

    ```
    118.178.XX.XX foo.bar.com       ##Map the domain name to the IP address of the Ingr
    ess.
    ```

- **Services**: Set the names, paths, port numbers, and weights of the backend Services that you want to access.
  - Path: Enter the URL of the backend Service. In this example, the default root path / is used.
  - Name: In this example, select the nginx-new and nginx-old Services.
  - Port: In this example, port 80 is open.
  - Weight: Set a weight for each backend Service. The weight is a percentage value. The default value is 100. In this example, the weight for each backend Service is 50. This means that the two backend Services have the same weight.

7. Configure Transport Layer Security (TLS).

   Select **EnableTLS** to enable TLS and configure a secure Ingress. For more information, see Configure an Ingress.

   - You can select an existing Secret.

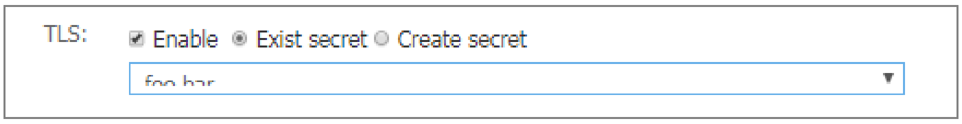
     a. Log on to a master node. Run the following command to create a file named *tls.key* and another file named *tls.crt*.

     ```
     openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout tls.key -out tls.crt
     -subj "/CN=foo.bar.com/O=foo.bar.com"
     ```

     b. Create a Secret.

     ```
     kubectl create secret tls foo.bar --key tls.key --cert tls.crt
     ```

     c. Run the `kubectl get secret` command to check whether the Secret is created. Then, you can select the newly created *foo.bar* Secret in the console.



TLS: ☑ Enable ● Exist secret ○ Create secret

foo.bar

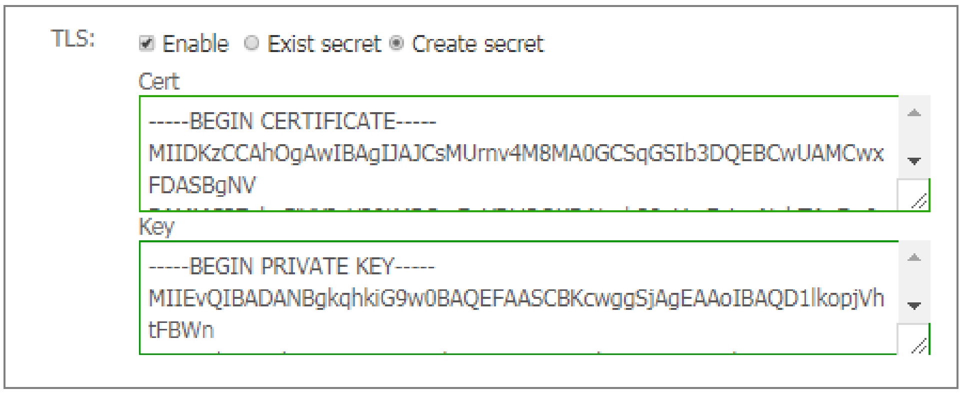
   - You can use the TLS private key and certificate to create a Secret.

     a. Log on to a master node. Run the following command to create a file named *tls.key* and another file named *tls.crt*.

     ```
     openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout tls.key -out tls.crt
     -subj "/CN=foo.bar.com/O=foo.bar.com"
     ```

     b. Run the `vim tls.key` and `vim tls.crt` commands to obtain the newly generated private key and certificate.

c. Paste the certificate to the Cert field and the private key to the Key field.



TLS: ☑ Enable ○ Exist secret ● Create secret

Cert

```
-----BEGIN CERTIFICATE-----
MIIDKzCCAhOgAwIBAgIJAJCsMUrnv4M8MA0GCSqGSIb3DQEBCwUAMCwx
FDASBgNV
```

Key

```
-----BEGIN PRIVATE KEY-----
MIIEvQIBADANBgkqhkiG9w0BAQEFAASCBKcwggSjAgEAAoIBAQD1lkopjVh
tFBWn
```

8. Configure canary release settings.

   ACK supports multiple traffic splitting methods. This allows you to select suitable solutions for specific scenarios, such as canary releases and A/B testing.

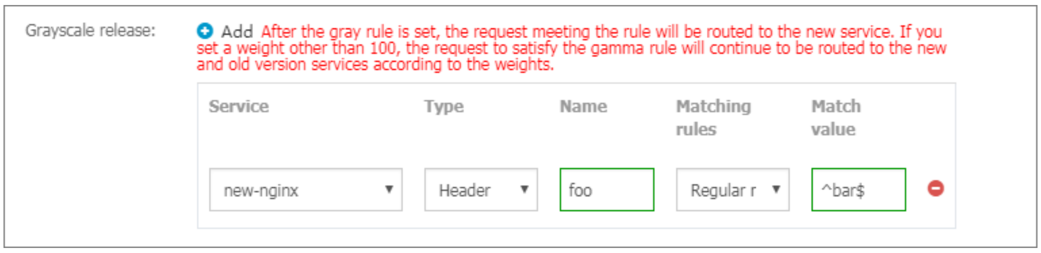
   i. Traffic splitting based on request headers.

   ii. Traffic splitting based on cookies.

   iii. Traffic splitting based on query parameters.

   After canary release settings are configured, only requests that match the specified rules are routed to the new-nginx Service. If the weight of new-nginx is lower than 100%, requests that match the specified rules are routed to the Service based on the Service weight.

   In this example, a rule is added to filter requests based on request headers. Only requests with headers that match the `foo=^bar$` regular expression can access new-nginx.



Grayscale release: Add After the gray rule is set, the request meeting the rule will be routed to the new service. If you set a weight other than 100, the request to satisfy the gamma rule will continue to be routed to the new and old version services according to the weights.

Service	Type	Name	Matching rules	Match value
new-nginx	Header	foo	Regular r	^bar$

   - **Services**: Specify the Services to be accessed.
   - **Type**: Select the type of matching rule. Valid values: Header, Cookie, and Query.
   - **Name and Match Value**: Specify the names and matching values of custom request fields in key-value pairs.
   - **Matching Rule**: Regular expressions and exact matches are supported.

   Note You can configure canary release settings for only two Services.

9. Configure annotations.

   Click **Add** on the right side of Annotations. In the **Type** drop-down list, you can select a type of annotation based on the following description:

   - **Custom Annotation**: Enter names and values as key-value pairs for the annotation. For more information, see Annotations.

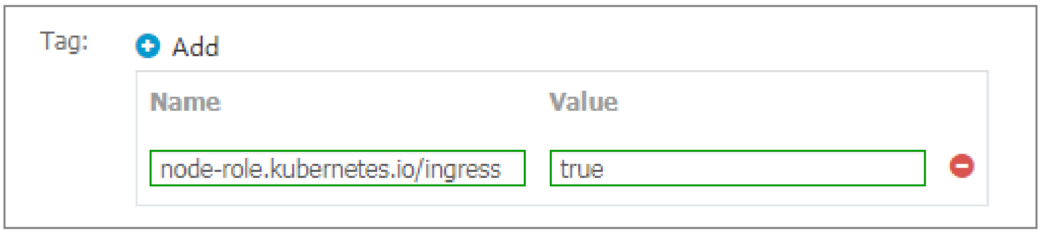
- **Ingress-NGINX**: Select annotations by name.

  You can add an annotation to redirect inbound traffic. For example, `nginx.ingress.kubernetes.io/rewrite-target: /` specifies that requests to */path* are redirected to the root path */*. The root path can be recognized by backend Services.

  > **Note** In this example, no path is configured for the backend Services. Therefore, you do not need to configure rewrite annotations. Rewrite annotations allow the Ingress to forward traffic through the root path to the backend Services. This avoids 404 errors that are caused by invalid paths.

10. Add labels.

    Add labels to describe the Ingress.

Name	Value
node-role.kubernetes.io/ingress	true

11. Click **Create**. You are redirected to the Ingresses page.

    After the Ingress is created, you can find the nginx-ingress Ingress on the Ingresses page.

View an Ingress

1.
2.
3.
4.
5. On the **Ingresses** page, find the Ingress that you want to view and click **Details** in the **Actions** column.

   On the details page, you can view detailed information about the Ingress.

Update an Ingress

1.
2.
3.
4.
5. On the **Ingresses** page, find the Ingress that you want to update and click **Update** in the **Actions** column.
6. In the **Update** dialog box, modify the parameters based on your requirements and click **Update**.

Delete an Ingress

1.
2.
3.

4.
5. On the **Ingresses** page, find the Ingress that you want to delete and choose **More > Delete** in the **Actions** column.
6. In the **Note** dialog box, click **Confirm**.

Manage Ingresses by using kubectl

Create an Ingress

1. Create a Deployment and a Service.

   You must create a Service to provide external access before you can create an Ingress.

   i. Create a file named *test-deployment-service.yaml* and copy the following content into the file.

      The following YAML template is used to create a Deployment named test-web1 and a Service named web1-service:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: test-web1
  labels:
    app: test-web1
spec:
  replicas: 1
  selector:
    matchLabels:
      app: test-web1
  template:
    metadata:
      labels:
        app: test-web1
    spec:
      containers:
      - name: test-web1
        imagePullPolicy: IfNotPresent
        image: registry.cn-hangzhou.aliyuncs.com/yilong/ingress-test:web1
        ports:
        - containerPort: 8080
---
apiVersion: v1
kind: Service
metadata:
  name: web1-service
spec:
  type: ClusterIP
  selector:
    app: test-web1
  ports:
    - port: 8080
      targetPort: 8080
```

ii. Run the following command to create the Deployment and Service:

```
kubectl apply -f test-deployment-service.yaml
```

2. Create an Ingress.

   i. Create a file named *test-ingress.yaml* and copy the following content into the file:

   - `name` : the name of the Ingress. In this example, the name is set to test-ingress.
   - `host` : the domain name that allows external access to the backend Service.
   - `path` : the URL for external access. SLB forwards traffic to the `backend` Service only when inbound requests match the `host` and `path` settings.
   - `backend` : the name and port number of the backend Service.
     - Service name: the name of the `backend` Service.
     - Service port: the Service port that is exposed.

   Clusters of Kubernetes versions earlier than 1.19 Clusters of Kubernetes 1.19 or later

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: test-ingress
  namespace: default
spec:
  rules:
  - host: test-ingress.com
    http:
      paths:
      - path: /foo
        backend:
          serviceName: web1-service
          servicePort: 8080
      - path: /bar
        backend:
          serviceName: web1-service
          servicePort: 8080
```

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: test-ingress
  namespace: default
spec:
  rules:
  - host: test-ingress.com
    http:
      paths:
      - path: /foo
        backend:
          service:
            name: web1-service
            port:
              number: 8080
        pathType: ImplementationSpecific
      - path: /bar
        backend:
          service:
            name: web1-service
            port:
              number: 8080
        pathType: ImplementationSpecific
```

ii. Run the following command to create the Ingress:

```
kubectl apply -f test-ingress.yaml
```

View Ingresses

Run the following command to view Ingresses:

```
kubectl get ingress
```

Update an Ingress

Run the following command to update an Ingress:

```
kubectl edit ingress <Ingress name>
```

Delete an Ingress

Run the following command to delete an Ingress:

```
kubectl delete ingress <Ingress name>
```

13.5.4. Advanced Ingress configurations

Prerequisites

Use cert-manager to apply for a free TLS certificate

cert-manager is an open source cloud-native tool used to manage certificates. You can use cert-manager to apply for TLS certificates for Kubernetes clusters and enable auto-renewal for the certificates. This section describes how to apply for a free certificate and enable auto-renewal for the certificate by using cert-manager.

1. Run the following command to deploy cert-manager:

   ```
   kubectl apply -f https://raw.githubusercontent.com/AliyunContainerService/serverless-k8s-examples/master/cert-manager/ask-cert-manager.yaml
   ```

   Note The YAML template referenced in the preceding command can be used to deploy cert-manager only in ASK clusters. For more information about how to deploy cert-manager in ACK clusters, see Use cert-manager to apply for a free TLS certificate.

2.
3.
4.
5.
6.
7.
8.

13.6. ASK Service discovery and DNS

Serverless Kubernetes (ASK) requires a DNS system to resolve the domain names of Services within ASK clusters. This topic describes how DNS resolution works in Kubernetes clusters. This topic also describes how to use CoreDNS and Alibaba Cloud DNS PrivateZone to resolve the domain names of Services within ASK clusters.

How DNS resolution works in Kubernetes clusters

Pods in ASK clusters send DNS queries based on the DNS configuration file named */etc/resolv.conf*. By default, the configuration file contains the following content:

```
# The nameserver parameter specifies the DNS servers to which the pods send DNS queries.
# If you use Alibaba Cloud DNS PrivateZone, the DNS servers of the virtual private cloud (VPC) where the cluster is deployed are specified as the nameservers. The IP addresses of the nameservers are 100.100.2.136 and 100.100.2.138.
# If you use CoreDNS, the IP address of the kube-dns Service is specified as the nameserver by default.
nameserver 172.xx.x.xx
# The search parameter specifies the search domain suffixes that are appended to the Service that you want to access. If the client pod belongs to the kube-system namespace, the following search domain suffixes are used:
search kube-system.svc.cluster.local svc.cluster.local cluster.local
# Other parameters.
options ndots:5
```

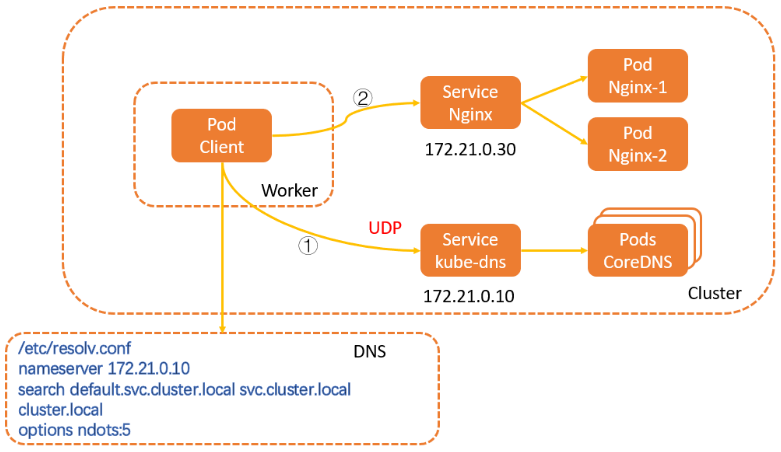
You can use CoreDNS or Alibaba Cloud DNS PrivateZone to implement DNS-based Service discovery in ASK clusters:

- CoreDNS: If you use CoreDNS in an ASK cluster, the cluster deploys a set of CoreDNS pods for DNS resolution.
- Alibaba Cloud DNS PrivateZone: If you use Alibaba Cloud DNS PrivateZone in an ASK cluster, pods in the cluster use Alibaba Cloud DNS PrivateZone for DNS resolution.

CoreDNS

CoreDNS is a DNS resolver for Kubernetes clusters. CoreDNS can resolve custom internal domain names and external domain names. CoreDNS provides a variety of plug-ins that you can use to configure custom DNS settings and customize host records, Canonical Name (CNAME) records, and rewrite rules for Kubernetes clusters. The CoreDNS project is hosted by Cloud Native Computing Foundation (CNCF), which also hosts Kubernetes. For more information, see CNCF. For more information about CoreDNS, see CoreDNS: DNS and Service Discovery.

CoreDNS is exposed as the kube-dns Service. The following figure shows how a client pod uses CoreDNS to access an NGINX Service.

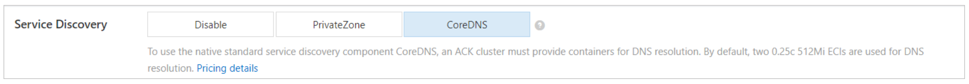


1. When a client pod attempts to access an NGINX Service, the pod sends a request to the DNS server that is specified in the DNS configuration file */etc/resolv.conf*. In this example, the DNS server address is 172.21.0.10, which is the IP address of the kube-dns Service. The result of the resolution is 172.21.0.30.
2. The client pod sends another request to 172.21.0.30, which is the IP address of the NGINX Service. Then, the request is forwarded to the backend pods named Nginx-1 and Nginx-2.

For more information about DNS resolution in Kubernetes clusters, see How DNS resolution works in ACK clusters.

How to enable CoreDNS

- You can specify and enable CoreDNS for DNS-based Service discovery when you create an ASK cluster in the .



Service Discovery: Disable | PrivateZone | [✓ CoreDNS]

To use the native standard service discovery component CoreDNS, an ACK cluster must provide containers for DNS resolution. By default, two 0.25c 512Mi ECIs are used for DNS resolution. Pricing details

- If you use an existing ASK cluster, you can go to the cluster details page and choose **Operations > Add-ons**. On the Add-ons page, click the **Networking** tab and click **Install** in the CoreDNS section. For more information, see Manage system components.

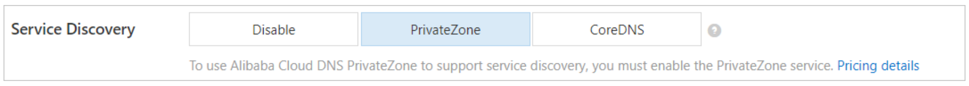
Alibaba Cloud DNS PrivateZone

Alibaba Cloud DNS PrivateZone dynamically monitors the Services and endpoints in each namespace of a Kubernetes cluster and automatically registers the domain names of the Services and the endpoints with Alibaba Cloud DNS PrivateZone. Pods in an ASK cluster use Alibaba Cloud DNS PrivateZone to resolve domain names for Service discovery in Kubernetes.

For more information about how to use Alibaba Cloud DNS PrivateZone, see Use the service discovery feature based on Alibaba Cloud DNS PrivateZone in ASK clusters.

How to enable Alibaba Cloud DNS PrivateZone

- You can specify and enable Alibaba Cloud DNS PrivateZone for DNS-based Service discovery when you create an ASK cluster.



Service Discovery: Disable | [✓ PrivateZone] | CoreDNS

To use Alibaba Cloud DNS PrivateZone to support service discovery, you must enable the PrivateZone service. Pricing details

- If you use an existing ASK cluster, you can set `enablePrivateZone` to true in the `eci-profile` ConfigMap to enable Alibaba Cloud DNS PrivateZone.

Related topics

- Configure DNS resolution
- Best practices for DNS services

13.7. Best practices

13.7.1. Use the service discovery feature based on Alibaba Cloud DNS PrivateZone in ASK clusters

Serverless Kubernetes (ASK) clusters support service discovery for intranet Services, headless Services, and ClusterIP type Services.

Prerequisites

- Alibaba Cloud DNS PrivateZone is activated in the Alibaba Cloud DNS console.
- Create an ASK cluster.
- You are connected to the ASK cluster. For more information, see Connect to an ACK cluster by using kubectl.

Context

Alibaba Cloud DNS PrivateZone is a private domain name resolution and management service based on Virtual Private Cloud (VPC). You can map a private domain name to an IP address in one or more custom VPCs. Your private domain names are not accessible in other network environments.

Procedure

1. Create a Deployment and a Service.

   You can copy the following content into a YAML file and run the `kubectl create -f nginx-service-ack.yaml` command to create a Deployment and a Service.

```
apiVersion: v1
kind: Service
```

```
metadata:
  name: nginx-headless-service
spec:
  ports:
  - port: 80
    protocol: TCP
  selector:
    app: nginx
  clusterIP: None
---
apiVersion: v1
kind: Service
metadata:
  name: nginx-clusterip-service
spec:
  ports:
  - port: 80
    protocol: TCP
  selector:
    app: nginx
  type: ClusterIP
---
apiVersion: v1
kind: Service
metadata:
  name: nginx-intranet-service
  annotations:
    service.beta.kubernetes.io/alicloud-loadbalancer-address-type: intranet
spec:
  ports:
  - port: 80
    protocol: TCP
  selector:
    app: nginx
  type: LoadBalancer
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
```

```
        image:  nginx:alpine
        ports:
        - containerPort: 80
```

2. Run the following command to view the state of the created application:

   ```
   kubectl get svc,pod,deployment
   ```

3. Log on to the Alibaba Cloud DNS console.
4. In the left-side navigation pane, click **PrivateZone** to go to the **All Zones** tab.
5. On the All Zones tab, find the zone that you want to manage and click **Configure** in the Actions column. The resolution settings page appears.

   Note Each record is in the `$svc.$ns` format and corresponds to an IP address. The following resolution rules apply:

   - A LoadBalancer type Service corresponds to only one resolution record in Alibaba Cloud DNS PrivateZone. The record corresponds to the IP address of the related Server Load Balancer (SLB) instance.
   - A ClusterIP type Service corresponds to only one resolution record in Alibaba Cloud DNS PrivateZone. The record corresponds to the IP address of the cluster.
   - A headless Service corresponds to multiple resolution records in Alibaba Cloud DNS PrivateZone. These records correspond to the IP addresses of the backend pods.

   You can access a Service by using the private domain name in the VPC.

   - Long domain name: You can use a long domain name of the `$svc.$ns.svc.cluster.local.$clusterId` format to access Services that are synchronized from other clusters to Alibaba Cloud DNS PrivateZone.
   - Short domain name: You can use a short domain name of the `$svc` format to access Services in the same namespace. You can use a short domain name of the `$svc.$ns` format to access Services in other namespaces.

   For more information, see serverless-k8s-examples.

13.7.2. Configure a security group

Security groups act as virtual firewalls to provide Stateful Packet Inspection (SPI) and packet filtering capabilities and define security domains in the cloud. You can add security group rules to control inbound and outbound traffic for elastic container instances within security groups.

Security group overview

Security group definition

A security group is a logically isolated group of instances within the same region that are mutually trusted and share the same security requirements. Security group rules control access to or from the Internet or internal network for the elastic container instances in the security group.

Note

- Each security group can manage multiple elastic container instances within the same region.
- Each elastic container instance must belong to a single security group.

Security group types

Security groups are classified into basic security groups and advanced security groups. By default, the following rules are added when a security group is created:

- Inbound rules that allow access on ports 80, 443, 22, and 3389, and an inbound rule that allows access over Internet Control Message Protocol (ICMP) on all ports. These rules can be modified.
- An outbound rule that allows all access on all ports.

The following table describes the differences in the features of basic and advanced security groups.

Feature	Basic security group	Advanced security group
Access control policy when the security group has no rules	• Inbound: denies all access requests. • Outbound: allows all access requests.	• Inbound: denies all access requests. • Outbound: denies all access requests.
Maximum number of private IP addresses	2,000	65,536
Mutual access between instances within the same security group	By default, instances within the same security group can access each other over the internal network.	By default, instances within the same security group are isolated from each other over the internal network. You must manually add security group rules to allow the instances to access each other over the internal network.
Control on access to or from other security groups	Rules can be added to control access to or from other security groups.	Rules cannot be added to control access to or from other security groups.

Notice

If your business requires a large number of elastic container instances and high O&M efficiency, we recommend that you use advanced security groups. Compared with basic security groups, advanced security groups can accommodate more elastic container instances and make it easier to configure security group rules.

Security group rules

Rules can be added to security groups to control inbound and outbound traffic. A security group rule is defined by attributes such as the direction, action, protocol type, port range, and authorization object. Take note of the following items about security group rules:

- The combined number of inbound and outbound rules in each security group cannot exceed 200.
- Follow the principle of least privilege when you add security group rules. Example:
  - Specify a single port such as port 80 in the format of 80/80, instead of a port range such as ports 1 to 80 in the format of 1/80.
  - 0.0.0.0/0 indicates all IP addresses. Do not set it as the authorization object unless necessary.

For more information, see Overview.

Specify a security group

When you create an elastic container instance, you must specify a security group for the instance.

Notice

You cannot change the security group for an elastic container instance. To use an elastic container instance within a different security group, create a new elastic container instance in that security group.

Specify security groups for elastic container instances in Kubernetes clusters

When you use Elastic Container Instance based on Virtual Kubelet in Kubernetes scenarios, all elastic container instances within a cluster are added to the default security group configured by Virtual Kubelet. You can specify a security group for an elastic container instance based on your business requirements.

- **Cluster**

  You can run the kubectl edit command to modify the eci-profile ConfigMap of a cluster and change the default security group ID in the data section for the elastic container instances in the cluster.

  Note

  Virtual Kubelet 2.0.0.90-15deb126e-aliyun and later allow modifications to eci-profile for hot updates. If your Virtual Kubelet version is earlier than 2.0.0.90-15deb126e-aliyun, we recommend that you upgrade Virtual Kubelet.

  ```
  kubectl edit configmap eci-profile -n kube-system
  ```

  Modify the securityGroupId field in the data section. Sample code:

```
data:
  enableClusterIp: "true"
  enableHybridMode: "false"
  enablePrivateZone: "false"
  resourceGroupId: ""
  securityGroupId: sg-2ze0b9o8pjjzts4h**** # Specify a security group ID.
  selectors: ""
  vSwitchIds: vsw-2zeet2ksvw7f14ryz****,vsw-2ze94pjtfuj9vaymf****
  vpcId: vpc-2zeghwzptn5zii0w7****
```

- **Specify security groups for elastic container instances**

  You can add annotations to the metadata section in pod configurations to specify a security group for an elastic container instance. Sample code:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: demo
  labels:
    app: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
        annotations:

            k8s.aliyun.com/eci-security-group: "sg-bp1dktddjsg5nktv****"      # Specify a
security group ID.

        labels:
            app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:latest
```

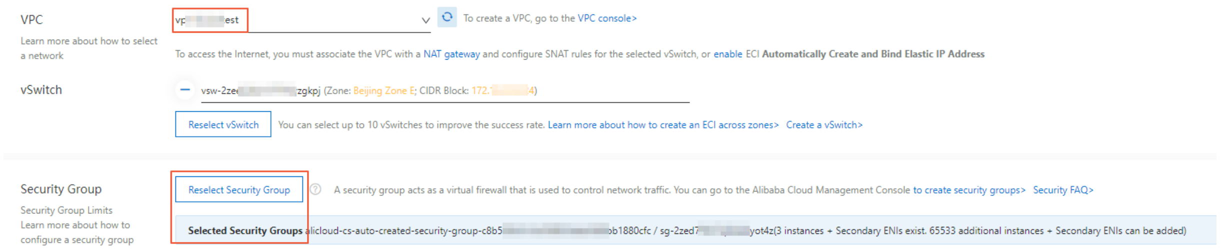
Specify security groups for elastic container instances by calling the API

When you call the CreateContainerGroup operation to create an elastic container instance, you can use the SecurityGroupId parameter to specify a security group. The following table describes the SecurityGroupId parameter. For more information, see CreateContainerGroup.

Parameter	Type	Example	Description
SecurityGroupId	String	sg-uf66jeqopgqa9hdn****	The ID of the security group

Specify security groups for elastic container instances in the console

When you create an elastic container instance on the instance buy page in the Elastic Container Instance console, you can specify a security group for the instance.



VPC
Learn more about how to select a network

vp[redacted]est

To create a VPC, go to the VPC console>

To access the Internet, you must associate the VPC with a NAT gateway and configure SNAT rules for the selected vSwitch, or enable ECI **Automatically Create and Bind Elastic IP Address**

vSwitch

vsw-2ze[redacted]zgkpj (Zone: Beijing Zone E; CIDR Block: 172.[redacted]4)

Reselect vSwitch   You can select up to 10 vSwitches to improve the success rate. Learn more about how to create an ECI across zones> Create a vSwitch>

Security Group
Security Group Limits
Learn more about how to configure a security group

Reselect Security Group   A security group acts as a virtual firewall that is used to control network traffic. You can go to the Alibaba Cloud Management Console to create security groups> Security FAQ>

Selected Security Groups alicloud-cs-auto-created-security-group-c8b5[redacted]b1880cfc / sg-2zed7[redacted]yot4z(3 instances + Secondary ENIs exist. 65533 additional instances + Secondary ENIs can be added)

Add a security group rule

You can add rules to a security group to control inbound and outbound traffic for the elastic container instances in the security group. Example:

- If your elastic container instance needs to communicate with a network outside the security group to which the instance belongs, you must add a security group rule to allow the instance to access the network.
- When attacks performed by request sources are detected, you can add security group rules to block access from the sources.

For more information about how to add security group rules, see Add security group rules.

14.Log management

14.1. Enable Log Service

This topic describes how to enable Log Service for a serverless Kubernetes (ASK) cluster.

Prerequisites

An ASK cluster is created. For more information, see Create an ASK cluster.

Procedure

To enable Log Service for an ASK cluster, perform the following operations.

1. Run the following command to modify the *eci-profile* file:

   ```
   kubectl -n kube-system edit configmap eci-profile
   ```

2. Set the `enableLogController` parameter in the *eci-profile* file to `true`, save the file, and exit.

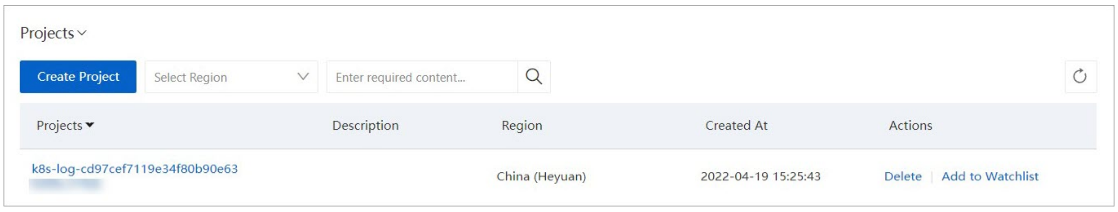
   ```
   apiVersion: v1
   data:
     #...
     enableLogController: "true"
     #...
   kind: ConfigMap
   metadata:
     name: eci-profile
     namespace: kube-system
   ```

3. Check whether Log Service is enabled for the ASK cluster.

   i. Log on to the Log Service console.

   ii. In the **Projects** section, check whether a project is created for the ASK cluster.

   If a project is created for the ASK cluster as shown in the following figure, Log Service is enabled for the ASK cluster.



Projects	Description	Region	Created At	Actions
k8s-log-cd97cef7119e34f80b90e63 [redacted]		China (Heyuan)	2022-04-19 15:25:43	Delete | Add to Watchlist

   ASK can import the standard output streams and log files of application pods to Log Service.

14.2. Collect log files by using Log Service

This topic describes how to use Log Service to collect stdout files and log files from application containers in a serverless Kubernetes (ASK) cluster.

Prerequisites

- An ASK cluster is created. For more information, see Create an ASK cluster.
- Log Service is enabled for the ASK cluster. For more information, see Enable Log Service.

Step 1: Configure log collection by using a YAML template

1.
2.
3.
4.
5. On the **Deployments** page, click **Create from YAML** in the upper-right corner.
6. Create a custom template and copy the following content to the template.

   YAML templates comply with the Kubernetes syntax. You must specify environment variables in the env field to collect log files from containers. The following code block is an example of a Deployment for collecting log files:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: alpine
  name: alpine
spec:
  replicas: 2
  selector:
    matchLabels:
      app: alpine
  template:
    metadata:
      labels:
        app: alpine
    spec:
      containers:
      - image: alpine
        imagePullPolicy: Always
        args:
        - ping
        - 127.0.0.1
        name: alpine
        env:
        # Specify environment variables.
        # Specify a Log Service project. If you want to use the default Log Service pro
ject for the cluster, do not specify the variable.
        - name: aliyun_logs_test-stdout_project
          value: k8s-log-xxx
        - name: aliyun_logs_test-file_project
          value: k8s-log-xxx
        # Specify a node group. If you want to use the default node group for the defau
lt Log Service project, do not specify the variable.
        - name: aliyun_logs_test-stdout_machinegroup
          value: k8s-group-app-alpine
        - name: aliyun_logs_test-file_machinegroup
          value: k8s-group-app-alpine
        # Specify a Logstore that is used to store the collected stout and stderr files
. In this example, the test-stdout Logstore is used.
        - name: aliyun_logs_test-stdout
          value: stdout
        # Specify a Logstore that is used to store the log files collected from the /lo
g/*.log directory. In this example, the test-file Logstore is used.
        - name: aliyun_logs_test-file
          value: /log/*.log
        ######### The retention period of log files that are collected to a Logstore. T
his variable takes effect only for the specified Logstore. ###########
        - name: aliyun_logs_test-stdout_ttl
          value: "7"
        ######### The number of shards in a Logstore. This variable takes effect only f
or the specified Logstore. ###########
        - name: aliyun_logs_test-stdout_shard
          value: "2"
```

Configure the following variables in sequence based on your requirements:

- Configure log collection by specifying environment variables. Make sure that all specified environment variables use the `aliyun_logs_` prefix. Add log collection configurations in the following format:

  ```
  - name: aliyun_logs_test-stdout
    value: stdout
  ```

  In the preceding YAML template, two environment variables are added to the log collection configuration. Environment variable `aliyun_logs_test-stdout` indicates that a `Logstore` named `test-stdout` is created to store the `stdout` files collected from containers. This way, the stdout files of containers are collected to the `Logstore` named `test-stdout`.

- If you specify a log path other than `stdout`, you must collect stdout files to the specified log path.

7. After you modify the YAML template, click **Create** to submit the configurations.

   After the Deployment is created, you can run the following command to query the status of pods:

   ```
   kubectl get Pods
   ```

   Expected output:

   ```
   NAME                        READY     STATUS    RESTARTS    AGE      IP           NODE
   alpine-76d978dbdd-g****     1/1       Running   0           21m      10.1.XX.XX   viking-
   c619c41329e624975a7bb50527180****
   alpine-76d978dbdd-v****     1/1       Running   0           21m      10.1.XX.XX   viking-
   c619c41329e624975a7bb50527180****
   ```

Step 2: Configure advanced settings in the env field

Step 3: View log data

1.
2.
3. In the Logstore list, find the Logstore that is specified when you configure log collection, click the ▦ icon, and then select **Search & Analysis** from the drop-down list.

   In this example, find the `test-stdout` Logstore, click the ▦ icon, and then select **Search & Analysis** from the drop-down list. On the page that appears, you can check the `stdout` files that are collected from elastic container instances.

14.3. Collect the log of a Job

This topic describes how to collect the log of a Job to Log Service.

Prerequisites

- A serverless Kubernetes (ASK) cluster is created. For more information, see ASK quick start.
- A virtual node is deployed in the cluster. For more information, see Deploy the virtual node controller and use it to create Elastic Container Instance-based pods.

- An Apsara File Storage NAS (NAS) file system is created and a mount target is added. For more information, see 创建文件系统 and Manage mount targets.

> **Note** If Log Service is enabled for the cluster, you can mount a volume to the Job to collect log data. Then, you can configure environment variables to synchronize the log data to Log Service. For more information, see Configure log collection for an elastic container instance.

Procedure

> **Notice**
>
> If Elastic Compute Service (ECS) instances are used in a cluster, DaemonSets can be used to collect the standard output of a Job. If elastic container instances are used in a cluster, you cannot use DaemonSets to collect log data. After a Job is complete, the pods created by the Job immediately exit. However, the log of the Job may not be collected. To resolve this problem, you can use the following method:
>
> Mount a NAS file system to the Job and store the log to the NAS file system. Then, mount the NAS file system to a pod that is used to import the log to Log Service.

1. Create a file named *job.yaml* by using a kubectl client.

   The following YAML template is used to create a Job to calculate π:

```
apiVersion: batch/v1
kind: Job
metadata:
  name: pi
spec:
  template:
    spec:
      containers:
      - name: pi
        image: resouer/ubuntu-bc
        command: ["sh", "-c", "echo 'scale=1000; 4*a(1)' | bc -l > /eci/a.log 2>&1"] #S
aves the output to the specified file.
        volumeMounts:
        - name: log-volume
          mountPath: /eci
          readOnly: false
      restartPolicy: Never
      volumes:
      - name: log-volume
        nfs:
            path: /eci
            server: 04edd48c7c-****.cn-hangzhou.nas.aliyuncs.com
            readOnly: false
  backoffLimit: 4
```

2. Deploy the Job to a virtual node.

Note For more information, see Deploy the virtual node controller and use it to create Elastic Container Instance-based pods.

```
kubectl apply -f job.yaml -n fvt-eci
```

3. Run the following command to view the states of the pods:

```
kubectl get pod -n fvt-eci
```

4. Create a file named *log-collection.yaml* by using the kubectl client and copy the following content to the file. Then, run the command to create a pod and mount the NAS file system to the pod to collect the log of the Job.

```
apiVersion: v1
kind: Pod
metadata:
  name: log-collection
spec:
  containers:
  - image: nginx:latest
    name: log-collection
    command: ['/bin/sh', '-c', 'echo &(cat /eci/a.log)'] #Prints the log of the Job.
    volumeMounts:
    - mountPath: /eci
      name: log-volume
  restartPolicy: Never
  volumes:
  - name: log-volume
    nfs:
      server: 04edd48c7c-****.cn-hangzhou.nas.aliyuncs.com
      path: /eci
      readOnly: false
```

14.4. Collect logs by using sidecar containers

Alibaba Cloud Log Service allows you to collect logs of containers in elastic container instances by using sidecar containers. This topic describes how to deploy a sidecar container and configure Logtail to collect logs of containers.

Prerequisites

- An ASK cluster is created. For more information, see Create an ASK cluster.
- Log Service is activated for the ASK cluster.

  If Log Service is not activated, you can activate Log Service as prompted when you log on to the Log Service console.

Background information

Alibaba Cloud Log Service allows you to collect logs of containers in elastic container instances by using sidecar containers. You can create a sidecar container and application container in an elastic container instance. The sidecar container is used to run a logging agent and collect the logs of the application container.

Notice

To use a sidecar container to collect container logs, you must enable Logtail. Logtail and the application container must share the same directory that is used to store log files. This way, Logtail can monitor the changes to log files in the shared directory and collect logs after the application container writes log data to the shared directory.

You can use one of the following methods to collect logs:

- Standard outputs

  To collect standard outputs, you must use the stdlog volume of the elastic container instance. When you create a pod, you can mount the stdlog volume to the sidecar container. The sidecar container can access standard outputs that are collected by the basic components of the elastic container instance as files.

- Text files

  To collect text files, you must use a shared volume in a pod. A volume can be mounted to multiple containers in a pod. The sidecar container can write the outputs of the application container to text files in the corresponding volume.

Step 1: Deploy a sidecar container

1. Create a Deployment that contains a sidecar container.

   The content in the following YAML template provides an example of the Deployment. Replace the placeholder variables with the values based on your requirements.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: nginx-log-sidecar-demo
  name: nginx-log-sidecar-demo
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx-log-sidecar-demo
  template:
    metadata:
      labels:
        app: nginx-log-sidecar-demo
    spec:
      containers:
        - name: nginx-log-demo
          image: registry-vpc.${RegionId}.aliyuncs.com/log-service/docker-log-test:late
st
          command:
            - /bin/mock_log
          args:
            - '--log-type=nginx'
            - '--stdout=false'
            - '--stderr=true'
            - '--path=/var/log/nginx/access.log'
            - '--total-count=100000000'
            - '--logs-per-sec=100'
          imagePullPolicy: Always
          volumeMounts:
            - mountPath: /var/log/nginx
              name: nginx-log
        - name: logtail
          image: registry-vpc.${RegionId}.aliyuncs.com/log-service/logtail:latest
          env:
            - name: ALIYUN_LOGTAIL_USER_ID
              value: "${Aliuid}"
            - name: ALIYUN_LOGTAIL_USER_DEFINED_ID
              value: nginx-log-sidecar
            - name: ALIYUN_LOGTAIL_CONFIG
              value: /etc/ilogtail/conf/${RegionId}/ilogtail_config.json
            - name: aliyun_logs_machinegroup
              value: k8s-group-app-alpine
          imagePullPolicy: Always
          volumeMounts:
            - mountPath: /var/log/nginx
              name: nginx-log
            - mountPath: /stdlog
              name: stdlog
      volumes:
        - emptyDir: {}        # Collect text files to the emptyDir volume.
          name: nginx-log
        - name: stdlog        # Collect standard outputs to the stdlog volume.
          flexVolume:
            driver: alicloud/pod-stdlog
```

2. Run the following command to query information about the pods:

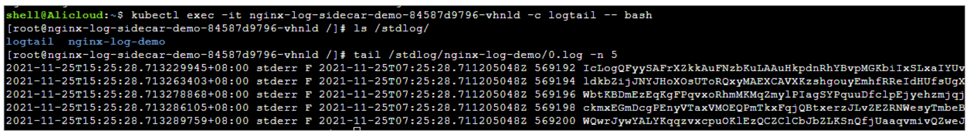
```
kubectl get pods -l app=nginx-log-sidecar-demo
```

Expected output:

```
NAME                                      READY   STATUS    RESTARTS   AGE
nginx-log-sidecar-demo-84587d9796-krn5z   2/2     Running   0          32m
nginx-log-sidecar-demo-84587d9796-vhnld   2/2     Running   0          32m
```

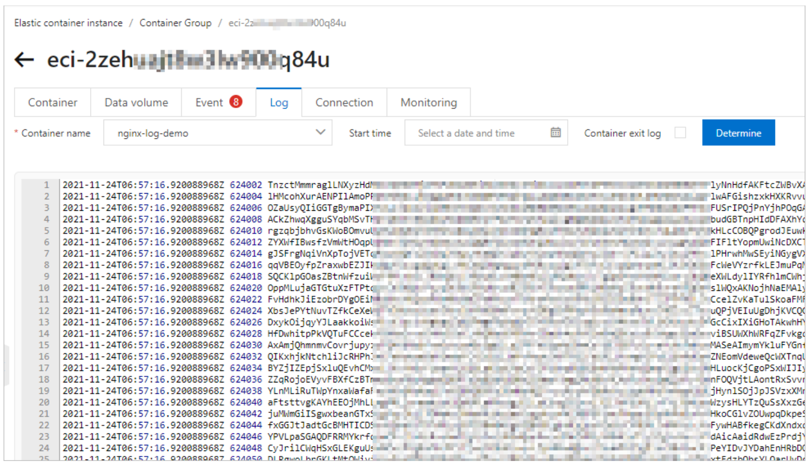
3. View logs.

- View logs by running the kubectl command



```
shell@Alicloud:~$ kubectl exec -it nginx-log-sidecar-demo-84587d9796-vhnld -c logtail -- bash
[root@nginx-log-sidecar-demo-84587d9796-vhnld /]# ls /stdlog/
logtail  nginx-log-demo
[root@nginx-log-sidecar-demo-84587d9796-vhnld /]# tail /stdlog/nginx-log-demo/0.log -n 5
2021-11-25T15:25:28.713229945+08:00 stderr F 2021-11-25T07:25:28.711205048Z 569192 IcLogQFyySAFrXZkkAuFNzbKuLAAuHkpdnRhYBvpMGKbiIxSLxaIYUv
2021-11-25T15:25:28.713263403+08:00 stderr F 2021-11-25T07:25:28.711205048Z 569194 ldkbZijJNYJHoXOsUToRQxyMAEXCAVXKzshgouyEmhfRReIdHUfsUgX
2021-11-25T15:25:28.713278868+08:00 stderr F 2021-11-25T07:25:28.711205048Z 569196 WbtKBDmEzEqKgFPqvxoRhmMKMqZmylPIagSYPquuDfclpEjyehzmjqj
2021-11-25T15:25:28.713286105+08:00 stderr F 2021-11-25T07:25:28.711205048Z 569198 ckmxEGmDcgPEnyVTaxVMOEQPmTkxFqjQBtxerzJLvZEZRNWesyTmbeB
2021-11-25T15:25:28.713289759+08:00 stderr F 2021-11-25T07:25:28.711205048Z 569200 WQwrJywYALYKqqzvxcpuOKlEzQCZClCbJbZLKSnQfjUaaqvmivQZweJ
```

- View logs in the Elastic Container Instance console


Elastic container instance / Container Group / eci-2z[redacted]00q84u

← eci-2zeh[redacted]q84u

* Container name: nginx-log-demo
Start time: Select a date and time
Container exit log ☐
Determine

```
2021-11-24T06:57:16.920088968Z 624002 TnzctMmmragILNXyzHd[redacted]lyNnHdfAKFtcZWBvX
2021-11-24T06:57:16.920088968Z 624004 1HMcohXurAENPIlAmoP[redacted]lwAFGishzxkHXKRvv
2021-11-24T06:57:16.920088968Z 624006 OZaUsyQIiGGTgBymaPI[redacted]FUSrIPQjPnYjhPOqG
2021-11-24T06:57:16.920088968Z 624008 ACkZhwqXgguSYqbMSvT[redacted]budGBTnpHIdDFAXhY
2021-11-24T06:57:16.920088968Z 624010 rgzqbjbhvGsKWoBOmvu[redacted]kHLcCOBQPgrodJEuw
2021-11-24T06:57:16.920088968Z 624012 ZYXWfIBwsfzVmWtHOqp[redacted]FIFltYopmUwiNcDXC
2021-11-24T06:57:16.920088968Z 624014 gJSFrgNqiVnXpTojVET[redacted]lPHrwhMwSEyiNGygV
2021-11-24T06:57:16.920088968Z 624016 qqVBEOyfpZraxwbEZJI[redacted]FcWeVYzrfkLEJmuPq
2021-11-24T06:57:16.920088968Z 624018 SQCKlpGOasZBtnWfzui[redacted]eXWLdylIYRfhlmCWh
2021-11-24T06:57:16.920088968Z 624020 OppMLujaGTGtuXzFTPt[redacted]slWQxAKNojhNaEMAl
2021-11-24T06:57:16.920088968Z 624022 FvHdhkJiEzobrDYgOEi[redacted]CcelZvKaTulSkoaFM
2021-11-24T06:57:16.920088968Z 624024 XbsJePYtNuvTZfkCeXe[redacted]uQPjVEIuUgDhjKVCQ
2021-11-24T06:57:16.920088968Z 624026 DxykOijqyYJLaakkoiW[redacted]GcCixIXiGHoTAkwhH
2021-11-24T06:57:16.920088968Z 624028 HfDwhitpPkVQTuFCCce[redacted]viBSUWXhWRFqZFvkg
2021-11-24T06:57:16.920088968Z 624030 AxAmjQhmnmvCovrjupy[redacted]MASeAImymYkluFYGn
2021-11-24T06:57:16.920088968Z 624032 QIKxhjkNtchliJcRHPh[redacted]ZNEomVdeweQcWXTnq
2021-11-24T06:57:16.920088968Z 624034 BYZjIZEpjSxluQEvhCM[redacted]HLuocKjCgoPSxWIJI
2021-11-24T06:57:16.920088968Z 624036 ZZqRojoEVyvFBXfCzBT[redacted]nFOQVjtLAontRxSvv
2021-11-24T06:57:16.920088968Z 624038 YLnMLiRuTWpYnxaWafa[redacted]jHynlSOjJpJSVzxXM
2021-11-24T06:57:16.920088968Z 624040 aFtsttvgKAYhEEOjMhL[redacted]WzysHLYTzQuSsXxzG
2021-11-24T06:57:16.920088968Z 624042 juMWmGiISgwxbeanGTx[redacted]HkoCGlvZOUwpqDkpe
2021-11-24T06:57:16.920088968Z 624044 fxGGJtJadtGcBMHTICD[redacted]FywHABfkegCKdXndx
2021-11-24T06:57:16.920088968Z 624046 YPVLpaSGAQDFRRMYkrf[redacted]dAicAaidRdwEzPrdj
2021-11-24T06:57:16.920088968Z 624048 CyJrilCWqHSxGLEKguU[redacted]PeYIDvJYDahEnHRbD
```

Step 2: Configure Logtail to collect logs

After you deploy the sidecar container, you need to configure Logtail in the Log Service console to collect logs.

1. Log on to the Log Service console.
2. In the **Import Data** section, click **RegEx - Text Log**.
3. On the Specify Logstore wizard page, set the parameters and click **Next**.

   Select a project and Logstore. If no project or Logstore is available, you can click **Create Now** and create one.

   Note

   By default, the system creates a project named `k8s-log-{The ID of the Kubernetes cluster}` for each Kubernetes cluster.

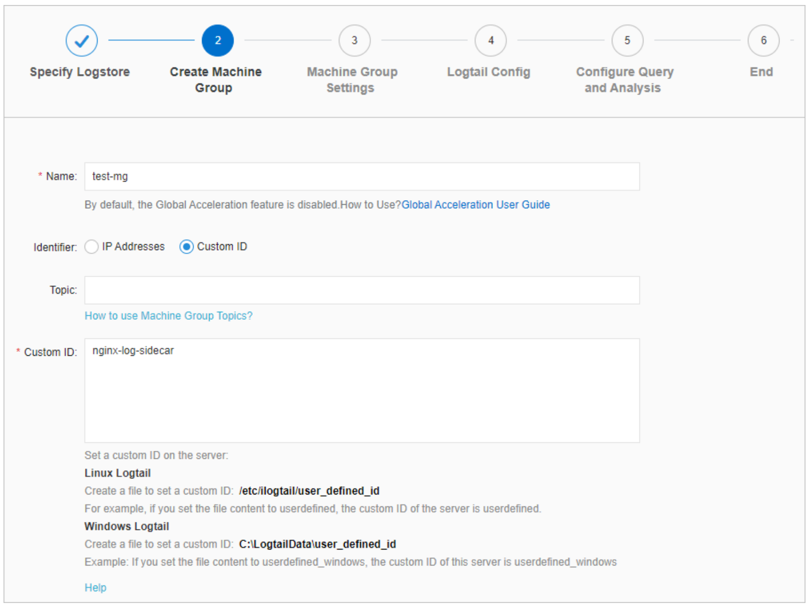
4. (Optional) Create a machine group.

   If a machine group is available, click **Use Existing Machine Groups** to skip this step.

   i. On the Create Machine Group wizard page, follow the instructions to check whether a machine group is created and click **Complete Installation**.

ii. Configure the machine group parameters and click **Next**.

Select **Custom ID** for the Identifier parameter. Enter the value of `ALIYUN_LOGTAIL_USER_DEFINED_ID` that you set in Step 1 in the **Custom ID** section. In this example, the value is `nginx-log-sidecar`.



* Name: test-mg

By default, the Global Acceleration feature is disabled.How to Use?Global Acceleration User Guide

Identifier: ( ) IP Addresses [✓ Custom ID]

Topic:

How to use Machine Group Topics?

* Custom ID: nginx-log-sidecar

Set a custom ID on the server:

Linux Logtail

Create a file to set a custom ID: **/etc/ilogtail/user_defined_id**

For example, if you set the file content to userdefined, the custom ID of the server is userdefined.

Windows Logtail

Create a file to set a custom ID: **C:\LogtailData\user_defined_id**

Example: If you set the file content to userdefined_windows, the custom ID of this server is userdefined_windows

Help

5. Configure the machine group.

   Select and move the machine group that you want to use from Source Server Groups to Applied Server Groups and then click **Next**.

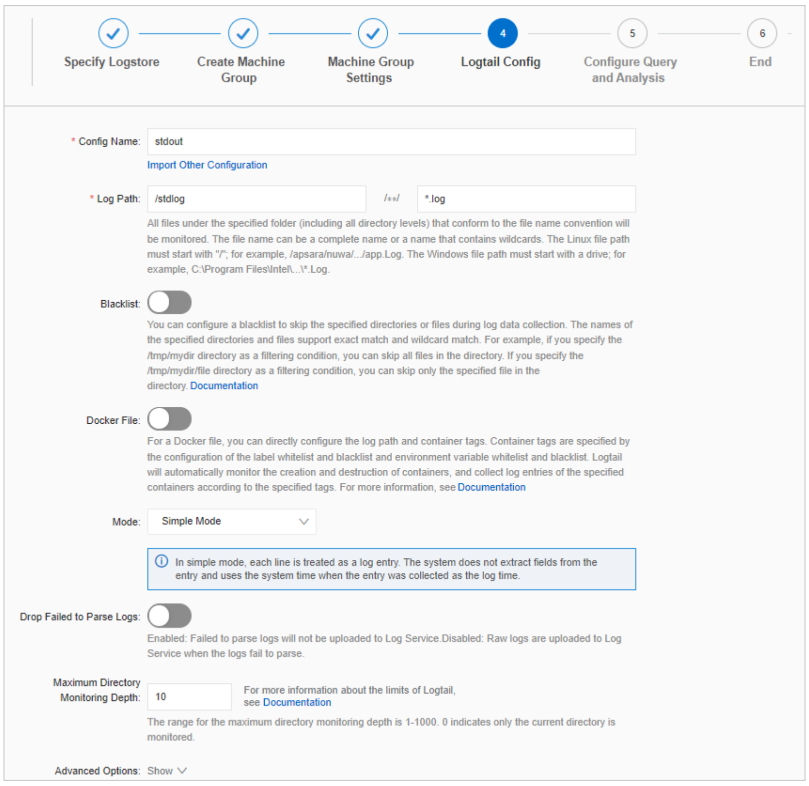
6. Configure Logtail.

   Logtail can collect text files in the following modes: Simple Mode, Full Regex Mode, Delimiter Mode, and JSON Mode. For more information, see Overview.

   > **Note**
   >
   > Turn off **Docker File**.

- Example on how to collect standard outputs

  If you want to collect standard outputs, the log path must be the same as the mount path of the stdlog volume. In this example, the log path is `/stdlog`.



* Config Name: stdout

Import Other Configuration

* Log Path: /stdlog /**/ *.log

All files under the specified folder (including all directory levels) that conform to the file name convention will be monitored. The file name can be a complete name or a name that contains wildcards. The Linux file path must start with "/"; for example, /apsara/nuwa/.../app.Log. The Windows file path must start with a drive; for example, C:\Program Files\Intel\...*.Log.

Blacklist:

You can configure a blacklist to skip the specified directories or files during log data collection. The names of the specified directories and files support exact match and wildcard match. For example, if you specify the /tmp/mydir directory as a filtering condition, you can skip all files in the directory. If you specify the /tmp/mydir/file directory as a filtering condition, you can skip only the specified file in the directory. Documentation

Docker File:

For a Docker file, you can directly configure the log path and container tags. Container tags are specified by the configuration of the label whitelist and blacklist and environment variable whitelist and blacklist. Logtail will automatically monitor the creation and destruction of containers, and collect log entries of the specified containers according to the specified tags. For more information, see Documentation

Mode: Simple Mode

In simple mode, each line is treated as a log entry. The system does not extract fields from the entry and uses the system time when the entry was collected as the log time.

Drop Failed to Parse Logs:

Enabled: Failed to parse logs will not be uploaded to Log Service.Disabled: Raw logs are uploaded to Log Service when the logs fail to parse.

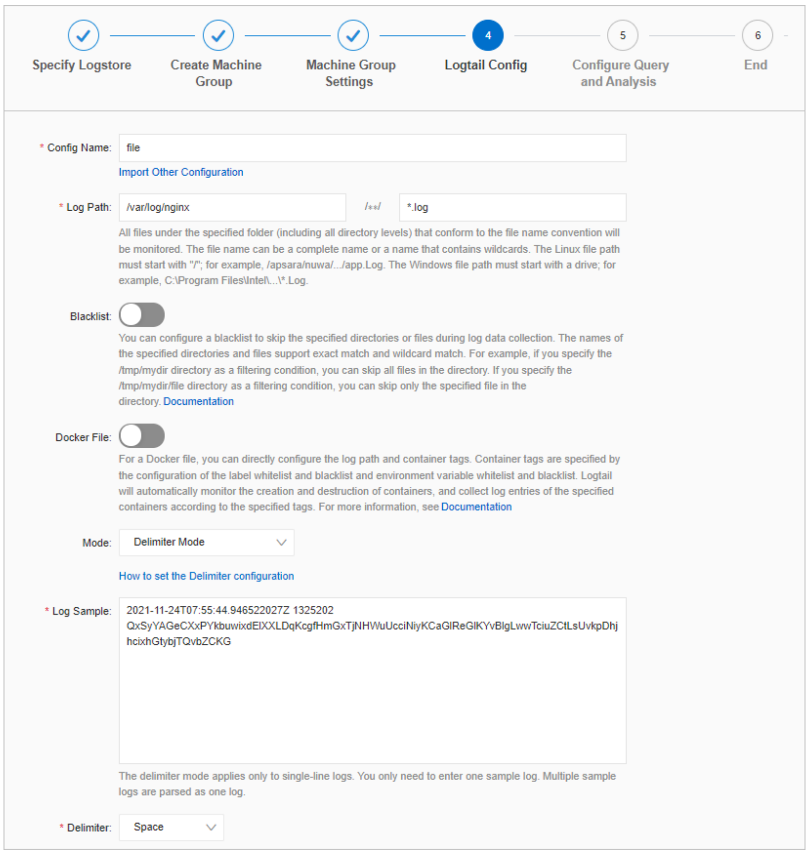
Maximum Directory Monitoring Depth: 10 For more information about the limits of Logtail, see Documentation

The range for the maximum directory monitoring depth is 1-1000. 0 indicates only the current directory is monitored.

Advanced Options: Show

- Example on how to collect text files

  If you want to collect text files, the log path must be the same as the mount path of the shared volume. In this example, the log path is `/var/log/nginx` .



* Config Name: file

Import Other Configuration

* Log Path: /var/log/nginx /**/ *.log

All files under the specified folder (including all directory levels) that conform to the file name convention will be monitored. The file name can be a complete name or a name that contains wildcards. The Linux file path must start with "/"; for example, /apsara/nuwa/.../app.Log. The Windows file path must start with a drive; for example, C:\Program Files\Intel\...*.Log.

Blacklist:

You can configure a blacklist to skip the specified directories or files during log data collection. The names of the specified directories and files support exact match and wildcard match. For example, if you specify the /tmp/mydir directory as a filtering condition, you can skip all files in the directory. If you specify the /tmp/mydir/file directory as a filtering condition, you can skip only the specified file in the directory. Documentation

Docker File:

For a Docker file, you can directly configure the log path and container tags. Container tags are specified by the configuration of the label whitelist and blacklist and environment variable whitelist and blacklist. Logtail will automatically monitor the creation and destruction of containers, and collect log entries of the specified containers according to the specified tags. For more information, see Documentation

Mode: Delimiter Mode

How to set the Delimiter configuration

* Log Sample:

```
2021-11-24T07:55:44.946522027Z 1325202
QxSyYAGeCXxPYkbuwixdElXXLDqKcgfHmGxTjNHWuUcciNiyKCaGlReGlKYvBlgLwwTciuZCtLsUvkpDhj
hcixhGtybjTQvbZCKG
```

The delimiter mode applies only to single-line logs. You only need to enter one sample log. Multiple sample logs are parsed as one log.

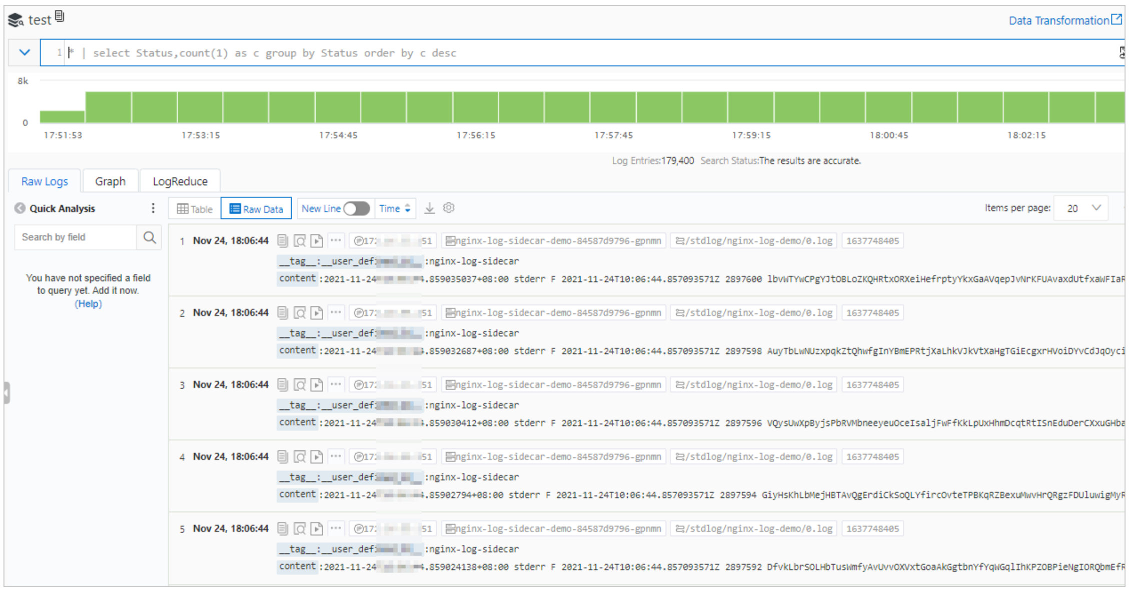
* Delimiter: Space

7. Configure log query and analysis.

   By default, indexes are configured. You can modify the indexes based on your business requirements. For more information, see Configure indexes.

8. View the logs that are collected from the elastic container instance.

   After you complete the preceding configurations, Log Service starts to collect logs of containers in the elastic container instance. The following figure shows an example on how to collect standard outputs to the Logstore of Log Service.



```
1 * | select Status,count(1) as c group by Status order by c desc
```

Log Entries:179,400 Search Status:The results are accurate.

```
1 Nov 24, 18:06:44  172[redacted]51  nginx-log-sidecar-demo-84587d9796-gpnmn  /stdlog/nginx-log-demo/0.log  1637748405
  __tag__:__user_def[redacted]:nginx-log-sidecar
  content:2021-11-24[redacted].859035037+08:00 stderr F 2021-11-24T10:06:44.857093571Z 2897600 lbvWTYwCPgYJtOBLoZKQHRtxORXeiHefrptyYkxGaAVqepJvNrKFUAvaxdUtfxaWFIaR
2 Nov 24, 18:06:44  172[redacted]51  nginx-log-sidecar-demo-84587d9796-gpnmn  /stdlog/nginx-log-demo/0.log  1637748405
  __tag__:__user_def[redacted]:nginx-log-sidecar
  content:2021-11-24[redacted].859032687+08:00 stderr F 2021-11-24T10:06:44.857093571Z 2897598 AuyTbLwNUzxpqkZtQhwfgInYBmEPRtjXaLhkVJkVtXaHgTGiEcgxrHVoiDYvCdJqOyci
3 Nov 24, 18:06:44  172[redacted]51  nginx-log-sidecar-demo-84587d9796-gpnmn  /stdlog/nginx-log-demo/0.log  1637748405
  __tag__:__user_def[redacted]:nginx-log-sidecar
  content:2021-11-24[redacted].859030412+08:00 stderr F 2021-11-24T10:06:44.857093571Z 2897596 VQysUwXpByjsPbRVMbneeyeuOceIsaljFwFfKkLpUxHhmDcqtRtISnEduDerCXxuGHba
4 Nov 24, 18:06:44  172[redacted]51  nginx-log-sidecar-demo-84587d9796-gpnmn  /stdlog/nginx-log-demo/0.log  1637748405
  __tag__:__user_def[redacted]:nginx-log-sidecar
  content:2021-11-24[redacted].85902794+08:00 stderr F 2021-11-24T10:06:44.857093571Z 2897594 GiyHsKhLbMejHBTAvQgErdiCkSoQLYfircOvteTPBKqRZBexuMwvHrQRgzFDUluwigMyR
5 Nov 24, 18:06:44  172[redacted]51  nginx-log-sidecar-demo-84587d9796-gpnmn  /stdlog/nginx-log-demo/0.log  1637748405
  __tag__:__user_def[redacted]:nginx-log-sidecar
  content:2021-11-24[redacted].859024138+08:00 stderr F 2021-11-24T10:06:44.857093571Z 2897592 DfvkLbrSOLHbTusWmfyAvUvvOXVxtGoaAkGgtbnYfYqWGqlIhKPZOBPieNgIORQbmEfR
```

Note The standard outputs in the stdlog volume of a pod are collected by the basic components of the elastic container instance. The format of the logs is the same as the format of Kubernetes logs. Kubernetes adds a prefix such as a timestamp to each entry of standard outputs. You must configure the log parser to remove the prefix. For more information, see Parse JSON logs.

14.5. Collect the logs of control plane components in an ASK cluster

Container Service for Kubernetes (ACK) allows you to collect the logs of control plane components in a serverless Kubernetes (ASK) cluster to a Log Service project that belongs to your Alibaba Cloud account. This topic describes how to enable log collection for control plane components in an ASK cluster and how to view the collected logs.

Prerequisites

Your Alibaba Cloud account has a sufficient quota of Logstores in Log Service.

Note The default Logstore quota of an Alibaba Cloud account is 50. To increase the quota, .

Context

You can analyze the logs of control plane components in a cluster to manage the cluster in a more secure and effective manner. You can enable log collection for control plane components in an ASK cluster after the cluster is created. The logs are shipped to the specified Log Service project that belongs to your account as log streams. You are charged for using Log Service on a pay-as-you-go basis. For more information, see Control plane components and Pay-as-you-go.

Enable log collection for control plane components

1.
2.
3.
4. In the left-side navigation pane, choose **Operations > Log Center**.
5. Click the **Logs of Control Plane Components** tab. Then, click **Enable Component Log Collection**.

Enable log collection for control plane components

When you create a cluster, select **Enable** for **Log Collection for Control Plane Components** on the **Component Configurations** wizard page.

Note

- By default, **Enable** is selected when you create an ASK Pro cluster. By default, this check box is not selected when you create an ASK standard cluster.
- You can select an existing Log Service project in the **Log Collection for Control Plane Components** section.

View the logs of control plane components

After you create an ASK cluster, you can use one of the following methods to view the logs of the control plane components:

Note You can view the logs of the following control plane components in an ASK cluster: **kube-apiserver**, **kube-controller-manager**, and **the cloud controller manager (CCM)**. For more information, see Log Service documentation.

Method 1: View the logs of control plane components in the Log Service console

1.
2. In the **Projects** section, click the name of the Log Service project that is used for the cluster.
3. On the left side of the **project details** page, click the **Logstores** tab. Then, click a Logstore to view the log of the component.

Method 2: View the logs of control plane components in the ACK console

1.
2.
3.

4. You can use the following entries to view the logs of the three control plane components.
    - View the logs of the three control plane components by using the **Cluster Information** menu.
        a. On the cluster details page, click the **Cluster Resources** tab and click the URL of the Log Service project.
        b. On the left side of the **project details** page, click the **Logstores** tab. Then, click a Logstore to view the log of the component.
    - You can also view the logs of the three control plane components in the **Operations** menu.
        a. In the left-side navigation pane of the cluster details page, choose **Operations > Log Center**.
        b. Click the **Logs of Control Plane Components** tab. You can select a component to view its log.

Logstores for control plane components

ASK allows you to collect the logs of the following control plane components. The log of each component is stored in a separate Logstore. For more information about the components, see Kubernetes components.

Component	Logstore	Description
kube-apiserver	apiserver	kube-apiserver is a component of the Kubernetes control plane that exposes the Kubernetes API. For more information, see kube-apiserver.
kube-controller-manager	kcm	kube-controller-manager is the control center of a Kubernetes cluster and runs controller processes. For more information, see kube-controller-manager.
Cloud Controller Manager	ccm	The CCM allows you to integrate Kubernetes with Alibaba Cloud services, such as Classic Load Balancer (CLB) and Virtual Private Cloud (VPC). CLB is formerly known as Server Load Balancer (SLB). The CCM manages the features, such as load balancing and cross-node communication, that are provided by these services. For more information, see Cloud Controller Manager.

Related information

- Pay-as-you-go
- Log search overview
- Create an ASK cluster

14.6. Customize log collection for an elastic container instance

Elastic Container Instance can export container logs to Log Service. This topic describes how to customize log collection for an elastic container instance, including how to specify a project, Logstore, and machine group.

Background information

If you enable log collection when you use an elastic container instance, the system generates the following log collection configurations by default:

- If you call API operations to use the elastic container instance, a project and a machine group are generated in each region.
    - The name of the project is `eci-log-default-project-{Region}-{The account ID}` .
    - The name of the machine group is `eci-log-default-machine-group-{Region}` .
- If you use the elastic container instance in a Kubernetes cluster, a project and a machine group are generated in each cluster.
    - The name of the project is `k8s-log-{The ID of the Kubernetes cluster}` .
    - The name of the machine group is `k8s-group-{The ID of the Kubernetes cluster}` .

You may need to customize Logstores and projects to collect logs of elastic container instances based on your business requirements. If you use elastic container instances together with different applications and services, you may need to add elastic container instances to different machine groups. You can use one of the following methods to customize log configurations, such as the project, Logstore, and machine group:

- Use the environment variables of elastic container instances

  After you use environment variables of elastic container instances to specify log configurations, the system automatically creates a project, Logstore, and machine group.
- Use the Log Service console or call API operations

  You can use the Log Service console or call API operations to create a project, Logstore, and machine group. After you create a project, Logstore, and machine group, you can customize a configuration file for the Logstore and apply the file to the machine group. Then, Log Service exports logs of the elastic container instances to the Logstore.

This topic describes how to use environment variables to customize log collection for an elastic container instance.

Configuration description

If you enable log collection by using the SlsEnable parameter when you call the CreateContainerGroup API operation to create an elastic container instance, you can use the environment variables of the elastic container instance to specify log configurations, including the project, Logstore, configuration, machine group, shard, and log retention period. The following section describes the format of and requirements on each configuration item when you specify a log configuration:

- Configuration (Logtail)

  The following example shows how to set a Logtail:

  ```
  -name: aliyun_logs_{The configuration name}
  -value: {The logging path}
  ```

  Notice

  If you want to collect stdout logs, set the logging path to stdout.

- Project

  The following example shows how to set a project:

  ```
  -name: aliyun_logs_{The configuration name}_project
  -value: {The project name}
  ```

  The project name must meet the following requirements:

  - The name must be 3 to 63 characters in length.
  - The name can contain only lowercase letters, digits, and hyphens (-). The name must start and end with a lowercase letter or a digit.

  Note

  If the name does not meet the preceding requirements, which causes the name format verification to fail, the project configuration is ignored and the default project is used.

- Logstore

  By default, the Logstore name is the same as the configuration name. The following example shows how to set a Logstore:

  ```
  -name: aliyun_logs_{The configuration name}_logstore
  -value: {The Logstore name}
  ```

  The Logstore name must meet the following requirements:

  - The name must be 3 to 63 characters in length.
  - The name can contain only lowercase letters, digits, hyphens (-), and underscores (_). The name must start and end with a lowercase letter or a digit.

  Note

  If the name does not meet the preceding requirements, which causes the name format verification to fail, the Logstore configuration is ignored and the default Logstore is used.

- Machine group

  The following snippet shows how to set a machine group:

```
  -name: aliyun_logs_{The configuration name}_machinegroup
  -value: {The machine group name}
```

The machine group name must meet the following requirements:

- The name must be 3 to 63 characters in length.
- The name can contain only lowercase letters, digits, hyphens (-), and underscores (_). The name must start and end with a lowercase letter or a digit.

> **Note**
>
> If the name does not meet the preceding requirements, which causes the name format verification to fail, the machine group configuration is ignored and the default machine group is used.

- Shard

  Read and write logs must be stored in shards of the Logstore. By default, each Logstore contains two shards. The following example shows how to set a shard:

  ```
  -name: aliyun_logs_{The configuration name}_shard
  -value: {The number of shards}    # Default value: 2. Valid values: 1 to 10.
  ```

- Log retention period

  By default, logs can be retained for 90 days in a Logstore. The following example shows how to set a log retention period:

  ```
  -name: aliyun_logs_{The configuration name}_ttl
  -value: {The number of days}    # Unit: days. Default value:90. Valid values: 1 to 3650.
  ```

- Tag

  The following example shows how to set a tag:

  ```
  --name: aliyun_logs_{The configuration name}_tags
  -value: {The tag key}={The tag value}
  ```

Configuration examples

If you need to customize the following log configurations:

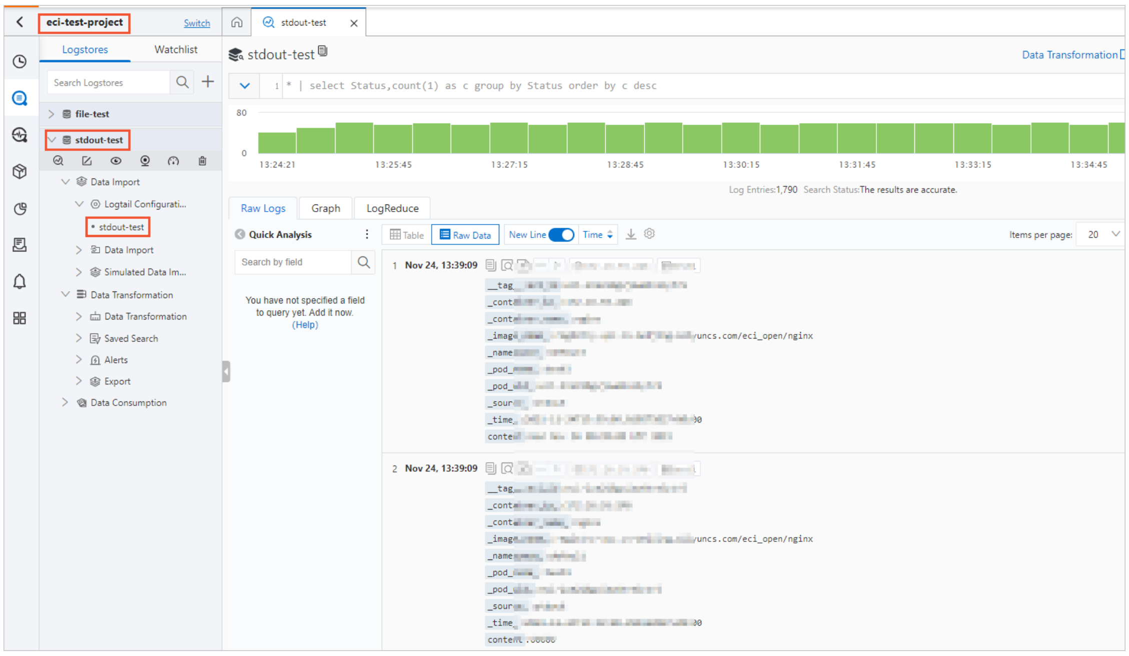
- Logstore and Logtail: Stdout logs are stored in the Logstore named stdout-test. Text logs are stored in the Logstore named file-test.
- Project: eci-test-project
- Machine group: eci-test-mg

The following example shows how to customize the log configurations:

```
# Enable log collection
'SlsEnable': 'true',
# Configure the collection of stdout logs.
'Container.1.EnvironmentVar.1.Key': 'aliyun_logs_stdout-test',
'Container.1.EnvironmentVar.1.Value': 'stdout',
'Container.1.EnvironmentVar.2.Key': 'aliyun_logs_stdout-test_project',
'Container.1.EnvironmentVar.2.Value': 'eci-test-project',
'Container.1.EnvironmentVar.3.Key': 'aliyun_logs_stdout-test_machinegroup',
'Container.1.EnvironmentVar.3.Value': 'eci-test-mg',
# Configure the collection of text logs.
'Container.1.EnvironmentVar.4.Key': 'aliyun_logs_file-test',
'Container.1.EnvironmentVar.4.Value': '/log/*.log',
'Container.1.EnvironmentVar.5.Key': 'aliyun_logs_file-test_project',
'Container.1.EnvironmentVar.5.Value': 'eci-test-project',
'Container.1.EnvironmentVar.6.Key': 'aliyun_logs_file-test_machinegroup',
'Container.1.EnvironmentVar.6.Value': 'eci-test-mg',
```

If you enable log collection and specify environment variables based on the preceding configurations when you create an elastic container instance, the logs of the instance can be collected to the corresponding Logstore of the project. The following figure shows the configuration effect.

- Collection of stdout logs



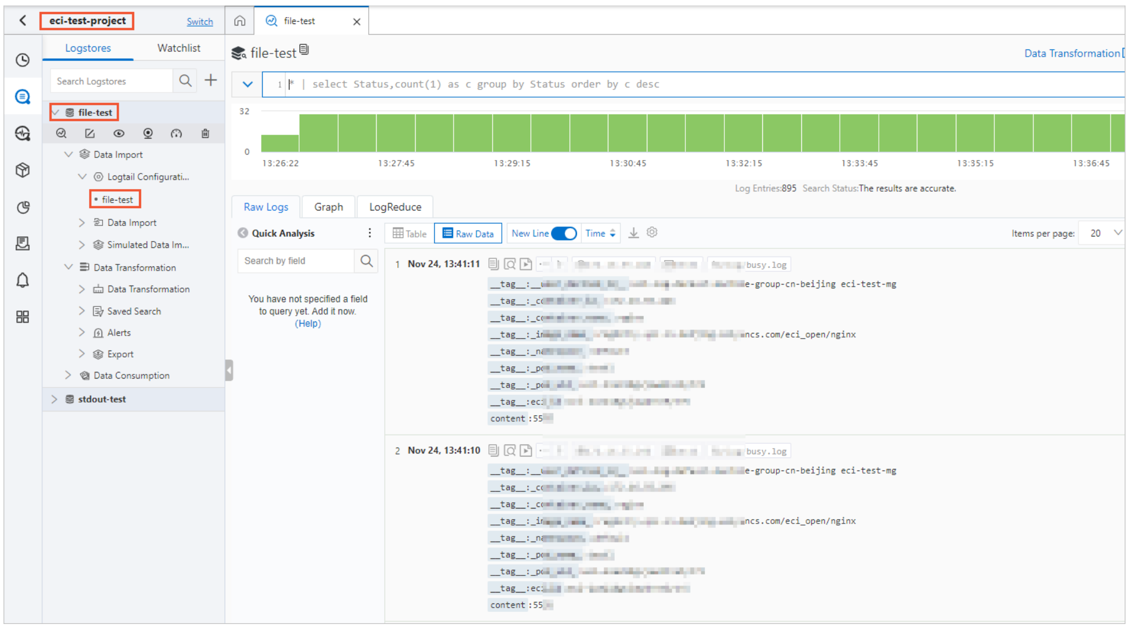
```
* | select Status,count(1) as c group by Status order by c desc
```

Log Entries:1,790 Search Status:The results are accurate.

You have not specified a field to query yet. Add it now. (Help)

```
1  Nov 24, 13:39:09
2  Nov 24, 13:39:09
```

- Collection of text logs



```
* | select Status,count(1) as c group by Status order by c desc
```

14.7. Parse JSON logs

If the stdout logs of containers in elastic container instances are Kubernetes logs, and Kubernetes automatically includes the timestamp and source of the logs in each log entry, the logs may fail to be parsed. For example, if Kubernetes automatically includes prefixes in JSON stdout logs, the logs fail to be parsed. This topic describes how to use the processor capability of Alibaba Cloud Log Service to parse JSON logs.

Problem description

The stdout logs that are generated by using the base components of elastic container instances use the same format as Kubernetes logs. Kubernetes automatically includes the timestamp and source of the logs in each entry of stdout logs. This may change the format of Kubernetes logs and cause the system to fail to parse JSON logs. Example:

```
2020-04-02T15:40:05.440500764+08:00 stdout F {"key1":"val1","key2":"val2"}
2020-04-02T15:40:07.442412564+08:00 stdout F {"key1":"val1","key2":"val2"}
2020-04-02T15:40:09.442774495+08:00 stdout F {"key1":"val1","key2":"val2"}
2020-04-02T15:40:11.443799303+08:00 stdout F {"key1":"val1","key2":"val2"}
2020-04-02T15:40:13.445099622+08:00 stdout F {"key1":"val1","key2":"val2"}
2020-04-02T15:40:15.445934358+08:00 stdout F {"key1":"val1","key2":"val2"}
2020-04-02T15:40:17.447064707+08:00 stdout F {"key1":"val1","key2":"val2"}
2020-04-02T15:40:19.448112987+08:00 stdout F {"key1":"val1","key2":"val2"}
2020-04-02T15:40:21.449393263+08:00 stdout F {"key1":"val1","key2":"val2"}
```

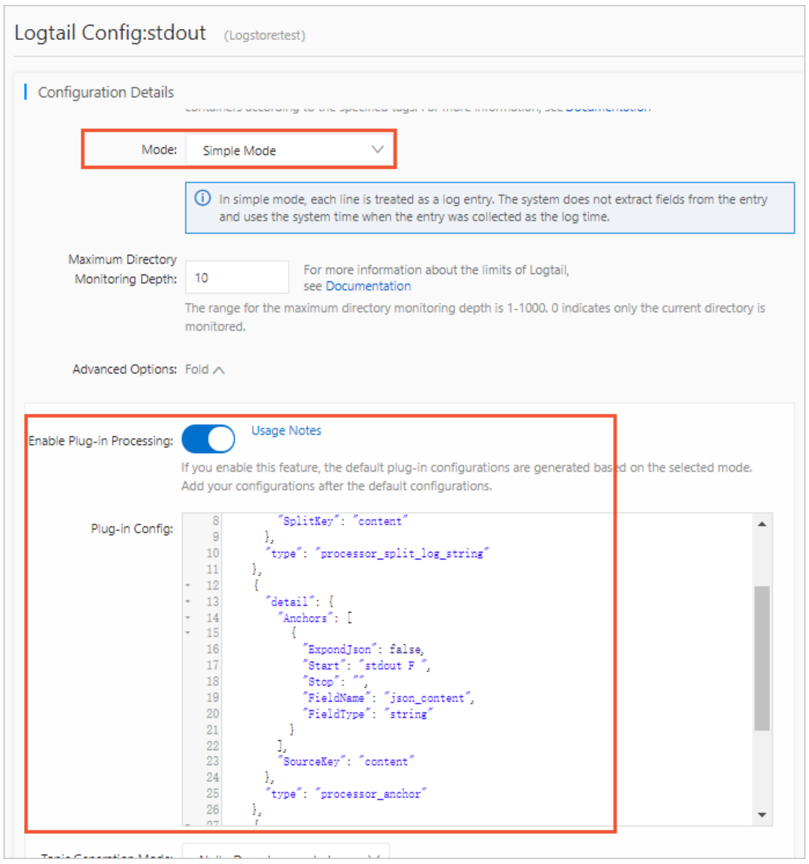
Solutions

You can use the processor capability of Log Service to parse JSON logs. Perform the following operations:

1. Log on to the Log Service console.
2. In the Projects section, click the name of the project in which the Logstore resides.

3. On the Logstores tab, find the Logstore in which the logs are stored, click the > icon to the left of the Logstore name, and then click the icon to the left of **Logtail Configurations**.
4. Click the Logtail configuration that you want to modify, and then click **Modify** in the upper-right corner of the page.
5. On the page that appears, set processors parameters in the Plug-in Config section and click **Save**.

   Select **Simple Mode** for Mode, turn on the **Enable Plug-in Processing** switch, and then set the processors parameters in the Plug-in Config section.

   Logtail Config:stdout (Logstore:test)

   Configuration Details

   Mode: Simple Mode

   In simple mode, each line is treated as a log entry. The system does not extract fields from the entry and uses the system time when the entry was collected as the log time.

   Maximum Directory Monitoring Depth: 10

   For more information about the limits of Logtail, see Documentation

   The range for the maximum directory monitoring depth is 1-1000. 0 indicates only the current directory is monitored.

   Advanced Options: Fold

   Enable Plug-in Processing: on

   Usage Notes

   If you enable this feature, the default plug-in configurations are generated based on the selected mode. Add your configurations after the default configurations.

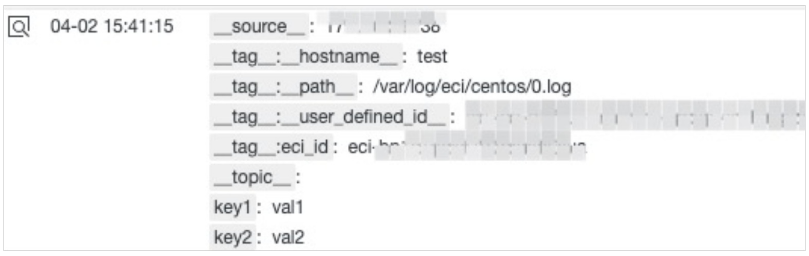
   Plug-in Config:

   ```
           "SplitKey": "content"
         },
         "type": "processor_split_log_string"
       },
       {
         "detail": {
           "Anchors": [
             {
               "ExpondJson": false,
               "Start": "stdout F ",
               "Stop": "",
               "FieldName": "json_content",
               "FieldType": "string"
             }
           ],
           "SourceKey": "content"
         },
         "type": "processor_anchor"
       },
   ```

   The following sample code shows how to configure processors:

```
{
    "processors": [
        {
            "type": "processor_anchor",
            "detail": {
                "SourceKey": "content",
                "Anchors": [
                    {
                        "Start": "stdout F ",
                        "Stop": "",
                        "FieldName": "json_content",
                        "FieldType": "string",
                        "ExpondJson": false
                    }
                ]
            }
        },
        {
            "type": "processor_json",
            "detail": {
                "SourceKey": "json_content",
                "KeepSource": false,
                "ExpandConnector": ""
            }
        }
    ]
}
```

Save the Logtail configurations and wait until the system displays the logs that are parsed as expected.



```
04-02 15:41:15   __source__ : [redacted]
                 __tag__:__hostname__ : test
                 __tag__:__path__ : /var/log/eci/centos/0.log
                 __tag__:__user_defined_id__ : [redacted]
                 __tag__:eci_id : eci-[redacted]
                 __topic__ :
                 key1 : val1
                 key2 : val2
```

15.Monitoring management

15.1. Enable ARMS for a serverless Kubernetes cluster

You can connect serverless Kubernetes clusters in different regions to Application Real-Time Monitoring Service (ARMS). This allows you to manage the clusters in a unified manner. This topic describes how to enable ARMS for a standard serverless Kubernetes cluster.

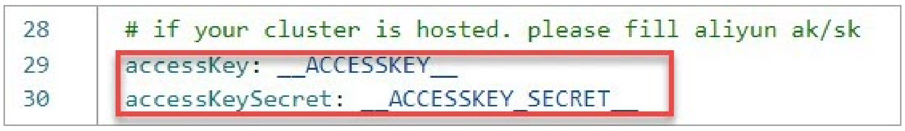
Prerequisites

Create an ASK cluster

Procedure

1.
2.
3. On the **Marketplace** page, click the **App Catalog** tab. On the App Catalog tab, find and click **ack-onepilot**.

   A large number of applications are displayed on the **App Catalog** tab. You can enter **ack-onepilot** or a keyword in the search box to search for the application.
4. On the **ack-onepilot** page, click **Deploy**.
5. In the **Deploy** wizard, select a cluster and a namespace, and then click **Next**.
6. On the **Parameters** wizard page, set the parameters and click **OK**.



```
# if your cluster is hosted. please fill aliyun ak/sk
accessKey: __ACCESSKEY__
accessKeySecret: __ACCESSKEY_SECRET__
```

Parameter	Description
accessKey	The AccessKey ID of your Alibaba Cloud account. Your account must be authorized to access ARMS.
accessKeySecret	The AccessKey secret of your Alibaba Cloud account.

> Notice
> - If an Express Connect circuit is deployed between the cluster and the virtual private cloud (VPC) in which the cluster resides, the Express Connect circuit is automatically used.
> - If the external cluster is registered over the Internet, you must delete `vpc` in the address of the image registry.

What's next

Verify that ack-arms-pilot is successfully deployed. For more information, see Install the ARMS agent for Java applications deployed in ACK.

Related information

For information about how to use ARMS, see ARMS application overview.

Overview

15.2. Enable ARMS Prometheus

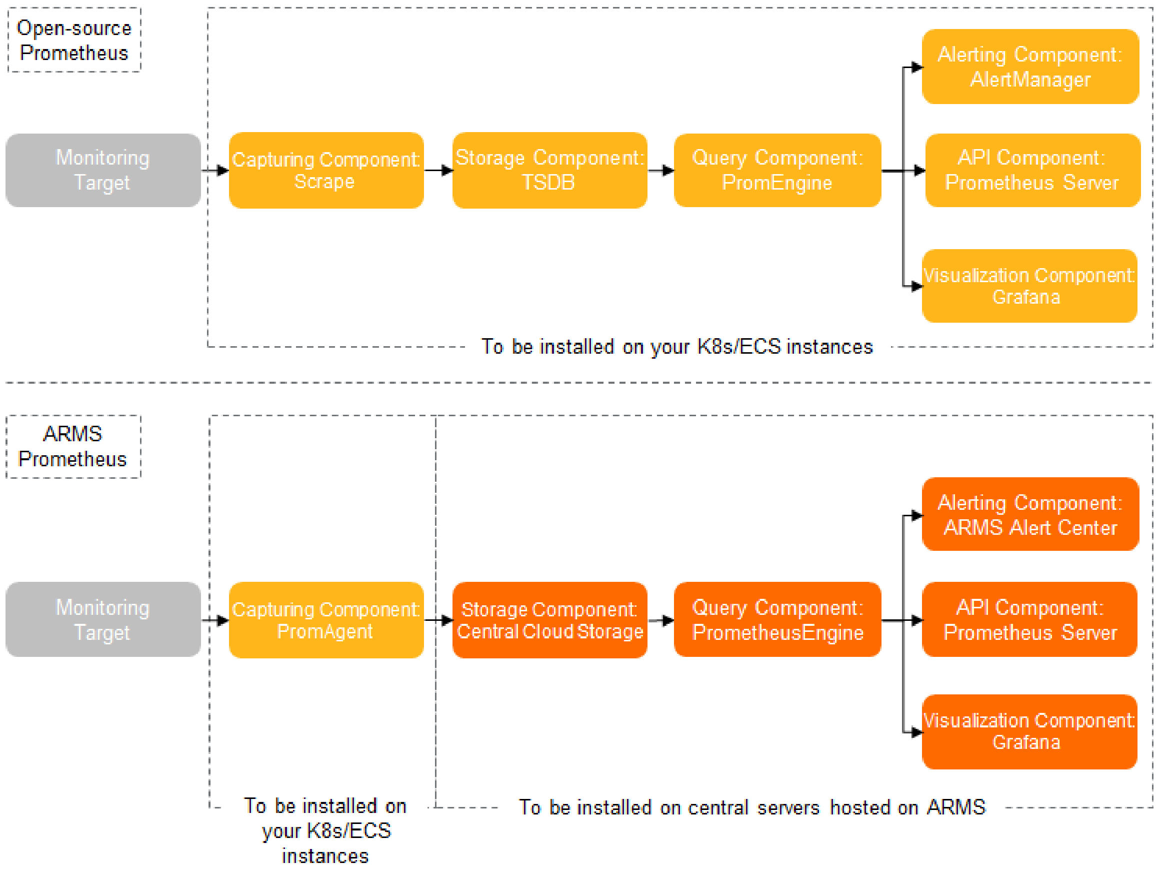
You can view metrics for serverless Kubernetes (ASK) clusters on predefined dashboards that are provided by Application Real-Time Monitoring Service (ARMS) Prometheus. This topic describes how to enable ARMS Prometheus for ASK.

Context

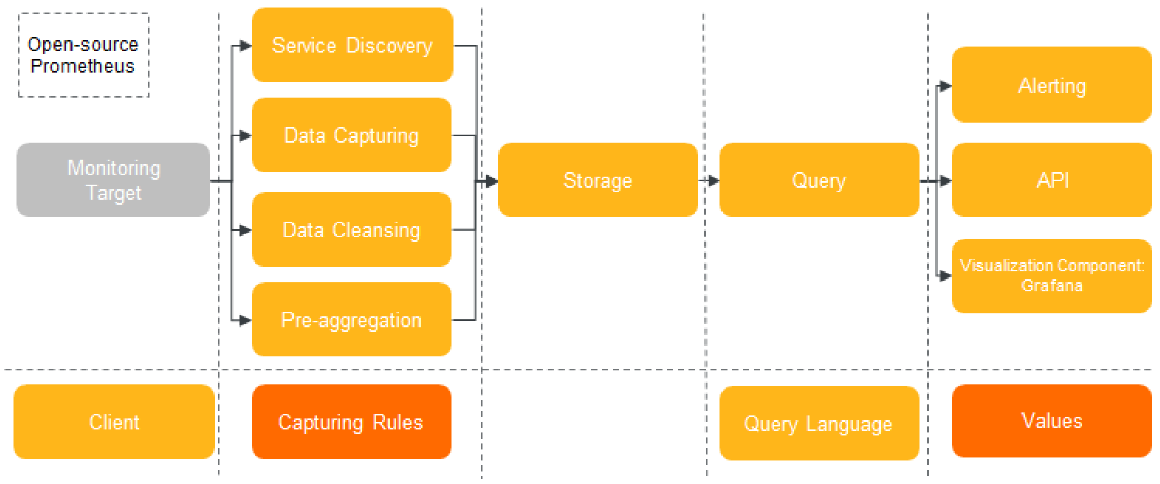
ARMS Prometheus is a managed monitoring service that is fully interfaced with the open source Prometheus ecosystem. ARMS Prometheus monitors a wide array of components and provides multiple ready-to-use dashboards. ARMS Prometheus saves you the efforts to manage underlying services, such as data storage, data presentation, and system maintenance.

Compared with open source Prometheus, ARMS Prometheus has the following benefits:

- Lightweight, stable, and accurate retry mechanism
    - Compared with open source Prometheus, the deployment of ARMS Prometheus is more lightweight. Instead of building a Prometheus monitoring system, you can install the Prometheus agent (PromAgent) to monitor your business.


Open-source Prometheus
Monitoring Target
Capturing Component: Scrape
Storage Component: TSDB
Query Component: PromEngine
Alerting Component: AlertManager
API Component: Prometheus Server
Visualization Component: Grafana
To be installed on your K8s/ECS instances
ARMS Prometheus
Monitoring Target
Capturing Component: PromAgent
Storage Component: Central Cloud Storage
Query Component: PrometheusEngine
Alerting Component: ARMS Alert Center
API Component: Prometheus Server
Visualization Component: Grafana
To be installed on your K8s/ECS instances
To be installed on central servers hosted on ARMS

  - Open source Prometheus requires 16 GB to 128 GB of memory. ARMS Prometheus requires only 200 MB to 1 GB of memory and 1 CPU core. ARMS Prometheus provides higher system stability than open source Prometheus.
  - Open source Prometheus retrieves data only once, and open source Prometheus may discard data when it writes data to the storage component. ARMS Prometheus retries multiple times if it fails to retrieve data. This ensures high concurrency when it writes data to the storage component. No data is discarded.
- Unlimited storage capacity
  - Open source Prometheus can collect up to one million metrics. ARMS Prometheus can scale its data collection capability based on the number of Kubernetes pods. Collection tasks can be distributed across pods.
  - The maximum storage capacity of open source Prometheus is limited by the size of the local disk. ARMS Prometheus uses the central cloud storage service. The storage capacity is not limited.
- Compatibility with the open source Prometheus systems
  - ARMS Prometheus is compatible with the clients and query languages in the open source Prometheus ecosystem. ARMS Prometheus provides optimized collection rules and usage values.



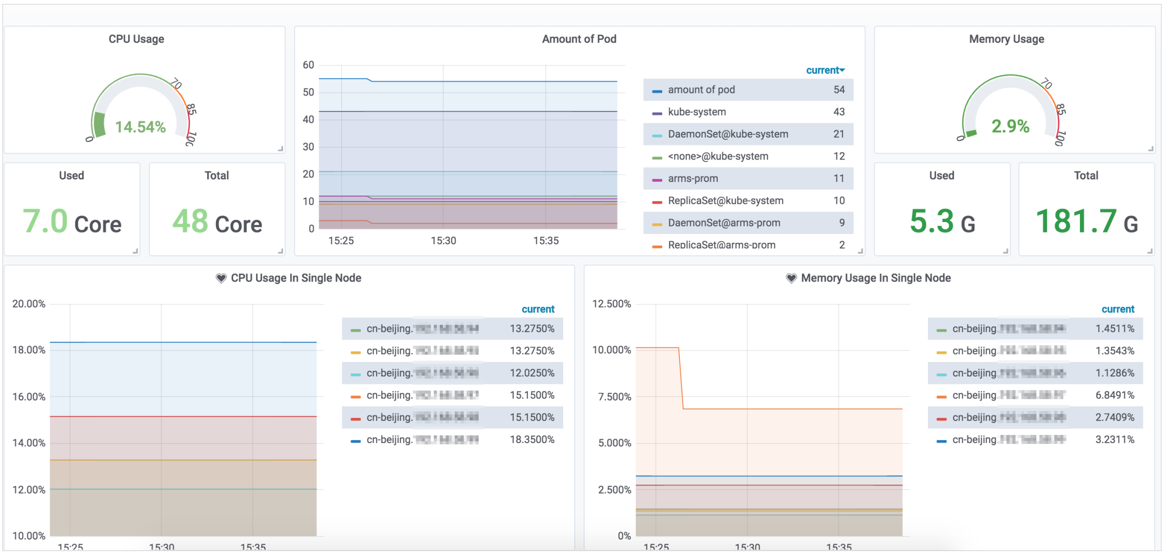
  - ARMS Prometheus is compatible with three mainstream methods to configure collection rules: the open source prometheus.yaml configuration file, the ServiceMonitor resource, and the annotation used to configure the default collection rule. ServiceMonitor is suitable for the monitoring of custom Kubernetes clusters. Compared with open source Prometheus, ARMS Prometheus allows you to update collection rules by using the prometheus.yaml configuration file. You need only to add the following three annotations to the Deployment file.

```
prometheus.io/scrape: "true"
prometheus.io/port: "9090"
prometheus.io/path: "/metrics"
```

  - ARMS Prometheus allows you to visualize data by using Grafana. By configuring the Prometheus HTTP API URL, you can implement multi-tenant isolation for the data source in Grafana and multi-tenant isolation for the Grafana dashboard. ARMS Prometheus is also compatible with the Explore data debugging module of Grafana.

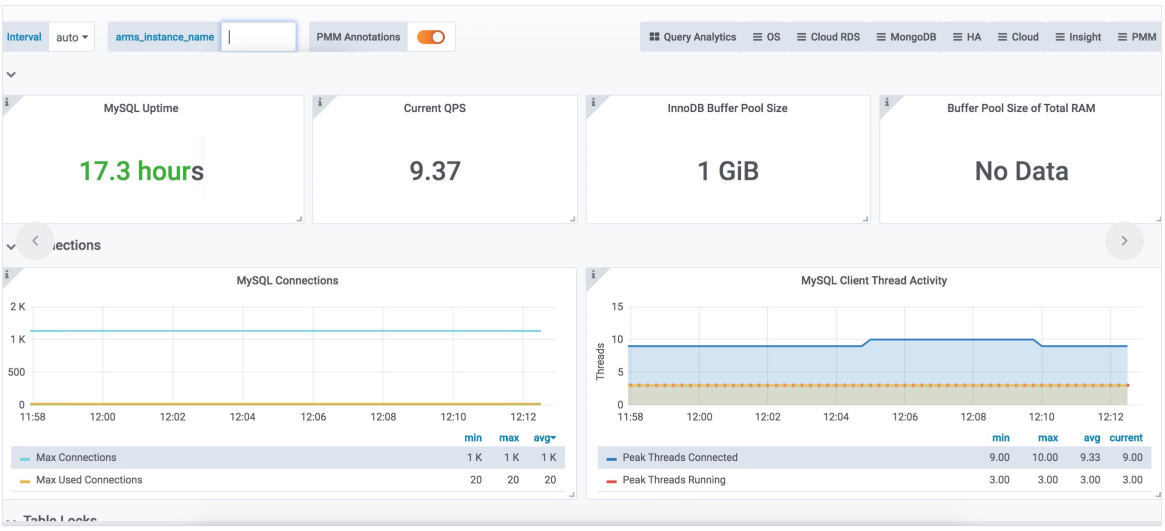
- ARMS Prometheus is compatible with the HTTP API module of open source Prometheus. ARMS Prometheus supports three standard API operations: query, query_range, and labelValues. In addition, you can add /userId/clusterId/regionId/ to the data URL to achieve multi-tenant isolation.
- ARMS Prometheus uses the built-in alerting system of ARMS and is compatible with the PromQL-based alert rules of open source Prometheus.

- Cost-effectiveness
  - ARMS Prometheus supports the default Kubernetes monitoring component. After you install the default Kubernetes monitoring component, ARMS Prometheus automatically creates exporters, collection rules, Grafana dashboards, and ARMS alerts. Compared with open source Prometheus, ARMS Prometheus reduces the time cost from about 3 days to about 10 minutes.



	current
amount of pod	54
kube-system	43
DaemonSet@kube-system	21
<none>@kube-system	12
arms-prom	11
ReplicaSet@kube-system	10
DaemonSet@arms-prom	9
ReplicaSet@arms-prom	2

  - ARMS Prometheus can monitor open source components. You can enter the accounts and passwords of the Relational Database Service (RDS) and Redis components. ARMS Prometheus automatically creates exporters and dashboards for these components. Compared with open source Prometheus, ARMS Prometheus reduces the time cost from about 7 days to about 3 minutes.



	min	max	avg
Max Connections	1 K	1 K	1 K
Max Used Connections	20	20	20

	min	max	avg	current
Peak Threads Connected	9.00	10.00	9.33	9.00
Peak Threads Running	3.00	3.00	3.00	3.00

- You can install or uninstall ARMS Prometheus with one click and perform health checks to debug Prometheus. Compared with open source Prometheus, ARMS Prometheus reduces your time cost from about 1 day to about 3 minutes.

Enable ARMS Prometheus

1.
2.
3.
4.
5. On the **Add-ons** page, click the **Logs and Monitoring** tab.
6. Find **ack-arms-prometheus** and click **Install**.
7. In the **Install ack-arms-prometheus** message, click **OK**.
   After you install the ack-arms-prometheus component, you can view the monitoring data in one of the following ways:
   - In the left-side navigation pane of the cluster details page, click **Cluster Information**. Then, click **Prometheus Monitoring** in the upper-right corner of the **Overview** tab.
   - In the left-side navigation pane of the cluster details page, choose **Operations > Prometheus Monitoring**.

16.Knative

16.1. Overview

Knative is an open source and Kubernetes-based platform for serverless applications. Knative helps you deploy and manage serverless workloads and build enterprise-class platforms for serverless workloads. Serverless Kubernetes (ASK) is integrated with Knative. To use cloud resources by calling the Knative API, you need only to create an ASK cluster and enable Knative for the cluster. In this case, you do not need to pay for the Knative controller.

Benefits of ASK Knative

Open source Knative	ASK Knative
By default, Istio gateways are used. Therefore, you must pay for the infrastructure resources that are used to install Istio controllers.	You do not need to pay for the Knative controller.
You are charged for the infrastructure resources that are used to install the Knative controller.	
A cold start occurs when you create a pod in an ASK cluster. Open source Knative uses the scale-to-zero mechanism to reduce costs. However, during the cold start, the cluster may fail to process requests due to session timeout.	ASK Knative does not scale the number of instances to zero during off-peak hours. Instead, ASK uses reserved instances. Reserved instances can be used to avoid cold starts at low costs. For more information about reserved instances, see Reserved instances.

Resource management in Knative

ASK hosts serverless applications and provides an easy way to use Kubernetes. You can directly deploy containerized applications in ASK clusters without the need to purchase nodes. The following list describes the benefits of using Knative to manage resources:

- You can use Knative to manage applications in ASK clusters.
- Knative automatically requests pod resources from ASK clusters.

The Knative Serving controller is integrated with ASK. To use cloud resources by calling Knative API operations, you need only to create an ASK cluster and enable Knative for the cluster. You are not charged for the Knative controller.

Knative Gateway

By default, open source Knative provides multiple Ingress gateway solutions, such as Istio, Gloo, Contour, Kourier, and Ambassador. Among these solutions, Istio is most frequently used. This is because Istio also functions as a service mesh. Each ASK cluster must contain at least two resident gateway instances. The two instances provide backup for each other to ensure high availability. The gateway controllers must be resident. You must pay infrastructure and O&M fees for these resident resources.

To improve user experience, Alibaba Cloud allows you to use Application Load Balancer (ALB) instances as Knative Ingress gateways. Knative Gateway provides gateway capabilities required by ASK clusters and is as stable and reliable as cloud services. Resident resources are not required. This reduces infrastructure costs and O&M workloads.

Reserved instances

By default, open source Knative scales the number of instances to zero during off-peak hours. This leads to a cold start the next time when the system starts the application. During a cold start, the system must allocate infrastructure resources, schedule pods, and pull the application image. A cold start also involves the startup time of the application. The size of the application image and the application startup time are based on the developer or service.

Unlike open source Knative, ASK Knative does not scale the number of instances to zero during off-peak hours. Instead, ASK Knative reserves one instance. The following content describes how reserved instances work:

- ASK Knative uses burstable instances to replace compute optimized instances during off-peak hours. When requests are received, ASK switches back to compute optimized instances. This mechanism reduces costs during off-peak hours.
- CPU credits that are accumulated during off-peak hours can be used during peak hours to reduce costs.

For more information about reserved instances, see Reserved instances.

Deploy Knative components in a Kubernetes cluster

You can deploy Knative in ASK clusters.

- If the version of your ASK cluster is 1.16 or later, you can deploy Knative in the Container Service for Kubernetes (ACK) console. For more information, see Enable Knative.
- Make sure that the version of the cluster is 1.16 or later.

Billing

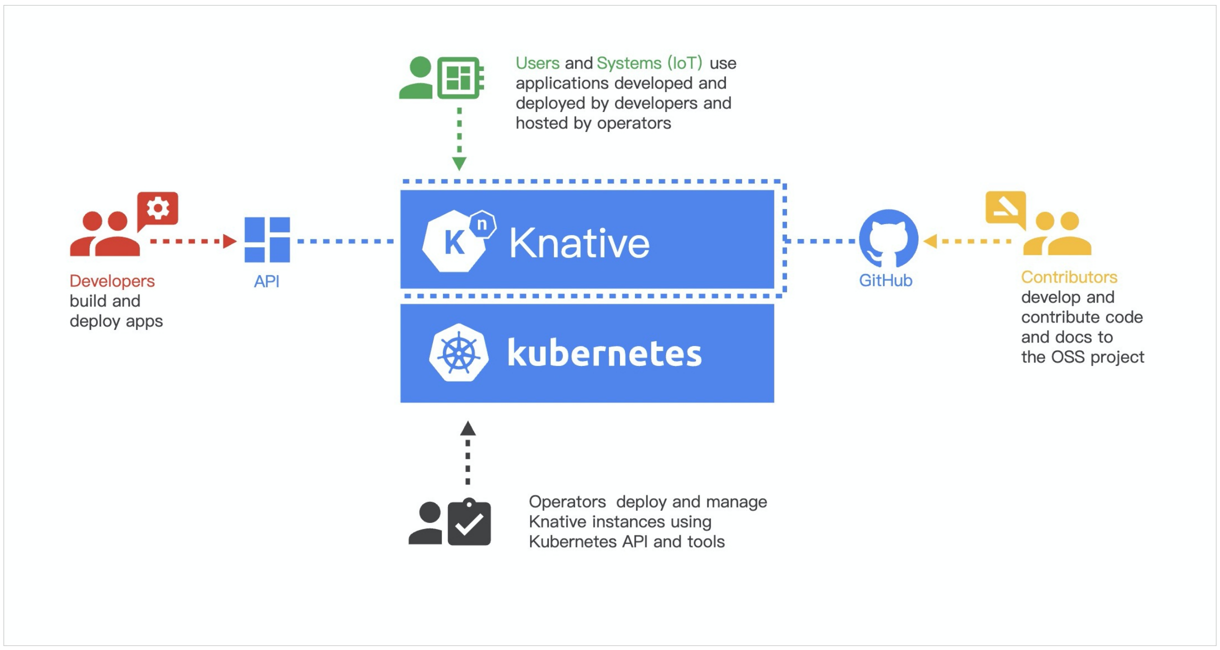
If you use ASK Knative, you are charged only for cloud resources that are used to manage your ASK cluster. The cloud resources include elastic container instances, SLB instances, and NAT gateways. These resources are charged based on corresponding billing rules. For more information, see Billing.

16.2. Knative community version

Knative is a Kubernetes-based serverless framework. Knative creates a cloud-native and cross-platform orchestration standard for serverless applications. Knative enforces this standard by streamlining the creation of container or function, workload management and auto scaling, and event models.

Knative architecture

The following figure shows how different roles interact with each other in the Knative architecture.


Users and Systems (IoT) use applications developed and deployed by developers and hosted by operators
Developers build and deploy apps
API
Knative
kubernetes
GitHub
Contributors develop and contribute code and docs to the OSS project
Operators deploy and manage Knative instances using Kubernetes API and tools

- Developers

  Developers of serverless applications can call the Kubernetes-native API to deploy serverless applications based on Knative.

- Contributors

  Contributors in the preceding figure mainly refer to the contributors to the Knative community.

- Operators

  Knative can be integrated with environments that are supported by Knative, such as environments of cloud service providers or internal environments of enterprises. Knative is based on Kubernetes. Therefore, it can be deployed in environments where Kubernetes is implemented.

- Users

  Users access services through Istio gateways, or run serverless applications by triggering Knative events.

Core Knative components

Knative consists of three core components. These components form a general-purpose serverless framework:

- Tekton: provides the capability to create images from source code.

  Tekton is used to retrieve source code from the code repository, compile the code into images, and push the images to the image repository. All these operations are performed in Kubernetes pods.

- Knative Eventing: provides event management capabilities, such as producing and consuming events.

  Knative Eventing has designed an all-in-one eventing system for event-driven serverless applications. The eventing system provides features that allow you to install external event sources, register and subscribe to events, and filter events. The event system decouples event producers and event consumers. An event producer can generate events before active event consumers listen to events. An event consumer can listen to events before active producers produce events.

- Knative Serving: manages the workloads of serverless applications. Knative Serving enables automatic scaling for pods where serverless applications are deployed based on Knative events and user requests. If no workload is processed, the number of pods is scaled to zero.

  Knative Serving is used to manage the workloads of serverless applications that provide services to users. The most important feature of Knative Serving is automatic scaling. The scaling capacity is not limited. Knative Serving also supports canary release.

16.3. Knative version release notes

16.3.1. Knative 0.18.3

Container Service for Kubernetes (ACK) supports Knative 0.18.3. This topic describes the changes and features of Knative 0.18.3.

Note We recommend that you set the apiVersion parameter of Knative Services to V1. The V1alpha1 and V1beta1 versions will be deprecated after Knative 0.19.0 is released.

Features

- To use the features of Knative 0.18.3, the Kubernetes version must be 1.18 or later.
- Supports multiple containers in a pod. Knative Services allow you to deploy multiple containers in a pod.
- Supports header-based routing policies. You can specify the `Knative-Serving-Tag: {revision-tag}` header in a request. This allows the Kourier ingress to directly send requests to specific revisions. You can use this feature to implement header-based canary releases.
- Adds compatibility with the nodeSelector feature in Kubernetes. The following parameters are required to configure this feature: affinity, nodeSelector, and tolerations.
- Knative Services support the dry-run feature. This feature allows you to validate the configurations of the current revision template. You can use one of the following parameters to enable the dry-run feature in the template:
  - features.knative.dev/podspec-dryrun: enabled
  - features.knative.dev/podspec-dryrun: strict

  Note When you create a Knative Service:
  - If you set features.knative.dev/podspec-dryrun to `enabled`, a dry run is performed if Kubernetes supports the dry-run feature. If Kubernetes does not support the dry-run feature, the system still tries to create the Knative Service.
  - If you set features.knative.dev/podspec-dryrun to `strict`, a `failure` is returned if Kubernetes does not support the dry-run feature.

16.4. Knative version release notes

16.4.1. Knative 0.18.3

Container Service for Kubernetes (ACK) supports Knative 0.18.3. This topic describes the changes and features of Knative 0.18.3.

Note We recommend that you set the apiVersion parameter of Knative Services to V1. The V1alpha1 and V1beta1 versions will be deprecated after Knative 0.19.0 is released.

Features

- To use the features of Knative 0.18.3, the Kubernetes version must be 1.18 or later.
- Supports multiple containers in a pod. Knative Services allow you to deploy multiple containers in a pod.
- Supports header-based routing policies. You can specify the `Knative-Serving-Tag: {revision-tag}` header in a request. This allows the Kourier ingress to directly send requests to specific revisions. You can use this feature to implement header-based canary releases.
- Adds compatibility with the nodeSelector feature in Kubernetes. The following parameters are required to configure this feature: affinity, nodeSelector, and tolerations.
- Knative Services support the dry-run feature. This feature allows you to validate the configurations of the current revision template. You can use one of the following parameters to enable the dry-run feature in the template:
  - features.knative.dev/podspec-dryrun: enabled
  - features.knative.dev/podspec-dryrun: strict

  Note When you create a Knative Service:
  - If you set features.knative.dev/podspec-dryrun to `enabled`, a dry run is performed if Kubernetes supports the dry-run feature. If Kubernetes does not support the dry-run feature, the system still tries to create the Knative Service.
  - If you set features.knative.dev/podspec-dryrun to `strict`, a `failure` is returned if Kubernetes does not support the dry-run feature.

16.5. Component management

16.5.1. Enable Knative

Knative is a serverless framework for applications on Kubernetes. Knative is intended for developing a cloud-native and cross-platform orchestration standard for serverless deployment. This topic describes how to enable Knative for a serverless Kubernetes (ASK) cluster.

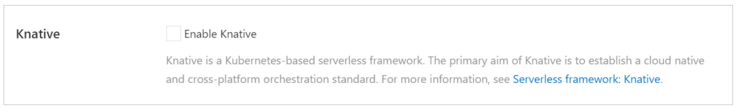
Context

Knative can be deployed in the following types of Container Service for Kubernetes (ACK) clusters: dedicated Kubernetes clusters, managed Kubernetes clusters, and ASK clusters. You can deploy only Knative Serving in ASK clusters.

Deploy Knative when you create an ASK cluster

1.
2.

3.
4. On the **Serverless Kubernetes** tab, configure the parameters. The following table describes the key parameters. For information about how to configure other parameters, see Create an ASK cluster.

Parameter	Description
Kubernetes Version	Select 1.15 or later.
NAT Gateway	Specifies whether to automatically create a NAT gateway and configure source network address translation (SNAT) rules for the virtual private cloud (VPC) where you want to deploy the ASK cluster. This parameter is required only when you set **VPC** to **Create VPC**. ◦ If you set VPC to Create VPC, you can select whether to automatically create a NAT gateway and configure SNAT rules for the VPC. ◦ If you clear this check box, you must manually create a NAT gateway and configure SNAT rules for the VPC. Otherwise, the cluster deployed in the VPC cannot access the Internet.
Access to the Internet	Enable or disable **Expose API Server with EIP**. ◦ If you select this check box, an elastic IP address (EIP) is created and port 6443 on master nodes is exposed. This port is used by the API server. You can connect to and manage the cluster by using kubeconfig over the Internet. ◦ If you clear this check box, no EIP is created. You can connect to and manage the cluster by using kubeconfig only within the VPC.
Knative	Select **Enable Knative** to deploy Knative in the ASK cluster.  Knative ☐ Enable Knative Knative is a Kubernetes-based serverless framework. The primary aim of Knative is to establish a cloud native and cross-platform orchestration standard. For more information, see Serverless framework: Knative.

5. Click **Create Cluster** in the upper-right corner of the page. In the **Confirm** message, click **OK** to start the deployment.

Deploy Knative in the ASK cluster

1.
2.
3.
4.
5. On the **Components** tab, click **Deploy Knative**.
6. After you select the Knative components that you want to install, click **Deploy**.

   You can use Knative Serving to manage serverless workloads. Pods can be automatically scaled for serverless applications upon Knative events and user requests. If no workload is processed, the number of pods can be scaled to zero.

Verify the result

After Knative is deployed, you can go to the **Knative Components** page and verify that the state of Knative is **Installed**.

16.5.2. Uninstall Knative components from an ACK cluster

This topic describes how to uninstall Knative components from a Container Service for Kubernetes (ACK) cluster.

Prerequisites

Enable Knative

Procedure

1.
2.
3.
4.
5. In the upper-right corner of the **Components** tab, click **Uninstall**.
6. In the **Confirm** dialog box, select **I confirm that I have read the above information and want to uninstall Knative** and click **OK**.

Result

You can find that Knative components are uninstalled on the **Knative Components** page.

16.6. Service management

16.6.1. Use Knative to deploy serverless applications

You can use Knative to deploy serverless applications with a few clicks. In the following example, a "Hello, World!" application is deployed.

Prerequisites

- Create an ASK cluster.
- Deploy Knative.

Procedure

1.
2.
3.

4.
5. In the upper-right corner of the **Services** tab, click **Create Service**.
6. On the Create Service page, set the **Cluster**, **Namespace**, and **Service Name** parameters. Then, select an image and image version.

Parameter	Description
Service Name	Enter a name for the Service. *helloworld-go* is used in this example.
Image Name	To select an image, click Select Image. In the dialog box that appears, select an image and click **OK**. You can also enter the address of an image in a private image registry. The image address must be in the following format: *domainname/namespace/imagename:tag*. In this example, *registry.cn-hangzhou.aliyuncs.com/knative-sample/helloworld-go* is used.
Image Version	Click Select Image Version to select an image version. If you do not set this parameter, the latest version is used. In this example, *73fbdd56* is selected.
Environment Variables	Set environment variables in key-value pairs. In this example, `TARGET=Knative` is entered.

For more information about how to set other parameters, see Parameter descriptions.

7. Click **Create**.

After you create the Service, the Service is displayed on the **Services** tab.

Access the Knative Service

After the Knative Service is deployed, you can point its domain name to the IP address of the gateway that is associated with the Service. This allows you to access the Knative Service through its URL.

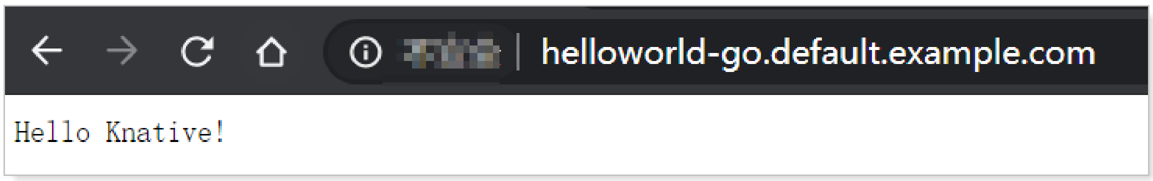
1. In the left-side navigation pane of the ACK console, choose **Knative > Components** to go to the **Knative Components** page. On the Knative Components page, you can view the gateway of the Knative Service.
2. Add the following information to the hosts file to point the domain name of the Service to the IP address of the gateway:

```
Gateway IP address + Domain name
```

Example:

```
47.95.XX.XX helloworld-go.default.example.com
```

3. After you modify the hosts file, enter *http://helloworld-go.default.example.com* into the address bar of the browser to access the Service.



```
Hello Knative!
```

16.6.2. Use ALB Ingresses in Knative

Application Load Balancer (ALB) runs at the application layer and supports protocols such as HTTP, HTTPS, and Quick UDP Internet Connections (QUIC). ALB offers high elasticity and can process a large amount of network traffic at Layer 7. In addition, ALB supports canary releases based on headers and cookies. This topic describes how to use ALB Ingresses in Knative.

Prerequisites

- A serverless Kubernetes (ASK) cluster is created. For more information, see ASK quick start.
- The ALB Ingress controller is installed. For more information, see Manage the ALB Ingress controller.

Step 1: Configure an ALB Ingress

You can configure an ALB Ingress by using one of the following methods:

Method 1: Select the ALB Ingress when you deploy Knative

1.
2.
3.
4.
5. On the **Components** tab, click **Deploy Knative**.
6. On the Deploy Knative page, select **ALB** and select at least two vSwitches. For more information about how to create vSwitches, see Create a vSwitch.
7. After you complete the configurations, click **Deploy**.

Method 2: Modify the configuration file of Knative

If Knative is deployed, you can modify the configuration file of Knative to specify an ALB Ingress.

1. Run the following command to modify the configuration file named *config-network.yaml*:

   ```
   kubectl -n knative-serving edit configmap config-network
   ```

2. Refer to the following template to modify `vswitch-ids` in the *config-network.yaml* file. Then, save the change and exit. `vswitch-ids` specifies the IDs of the vSwitches that are used by the ALB Ingress. You must specify at least two vSwitches IDs and the vSwitches must be deployed in different zones. For more information about the regions and zones that are supported by ALB, see Supported regions and zones.

   ```
   apiVersion: v1
   data:
     ...
     vswitch-ids: 'vsw-uf6kbvc7mccqia2pi****,vsw-uf66scyuw2fncpn38****'
     ...
   kind: ConfigMap
   metadata:
     name: config-network
     namespace: knative-serving
   ```

Step 2: Access a service by using the ALB Ingress

1. Run the following command to query the domain name of the ALB instance:

```
kubectl get albconfig knative-internet -ojson  jq .status.loadBalancer.dnsname
```

   Expected output:

```
"alb-ijvf32ubve64wz****.cn-shanghai.alb.aliyuncs.com"
```

2. Run the following command to access a Service:

```
curl -H  "Host: helloworld-go.default.example.com" http://alb-ijvf32ubve64wz****.cn-sha
nghai.alb.aliyuncs.com
```

Configure a CNAME record

ALB allows you to add a CNAME record to map your domain names to the publicly accessible domain name of the ALB instance. This allows you to access network resources in a convenient manner. For more information, see Configure a CNAME record.

16.6.3. Knative Gateway

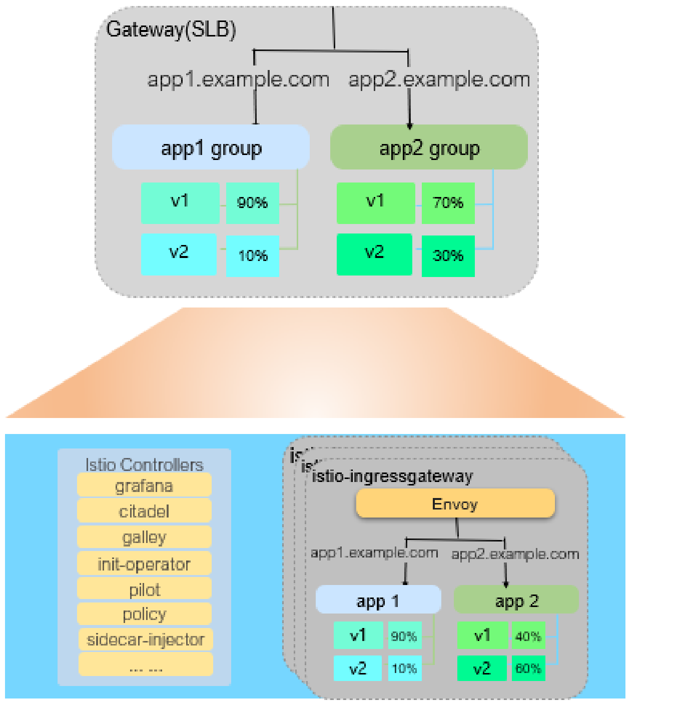
Knative provides multiple Ingress gateway solutions for you to create gateways for cloud resources. These gateways are stable and reliable because they do not require resident resources. Knative Ingress gateways help reduce infrastructure costs and simplify your maintenance work. This topic describes the benefits of Knative Ingress gateways and how to use Knative Ingress gateways.

Benefits

By default, open source Knative provides multiple Ingress gateway solutions, such as Istio, Gloo, Contour, Kourier, and Ambassador. Among these solutions, Istio is most frequently used. This is because Istio can also be used as a service mesh. At least two resident gateway instances are required for each application that is deployed within Knative. The two gateway instances provide backup for each other to ensure high availability. The Kubernetes controllers of these gateways must be resident. You must pay infrastructure and maintenance fees for these resident instances.

A serverless Kubernetes (ASK) cluster that has Knative enabled uses Server Load Balancer (SLB) instances as gateways. Internet-facing SLB instances function as external gateways and internal-facing SLB instances function as internal gateways. Knative Ingress gateways support both HTTP and HTTPS. By default, Knative generates a self-signed certificate for HTTPS connections. This certificate can secure all domain names. Therefore, you can use the certificate to test applications. Before you use Knative to deploy applications, configure an SSL certificate and specify the certificate ID in Kubernetes annotations. For more information, see Configure a certificate for an SLB instance that serves as a gateway to establish HTTPS connections.

To improve user experience, Alibaba Cloud allows you to use SLB instances as Knative Ingress gateways.



Use Knative Ingress gateways

1. Run the following command to query the IP addresses of the SLB instances:

```
kubectl -n knative-serving get svc
```

Expected output:

```
NAME                        TYPE            CLUSTER-IP      EXTERNAL-IP       PORT(S)
AGE
ingress-gateway             LoadBalancer    172.21.XX.XX    8.131.XX.XX       80:32701/TCP,443
:30561/TCP    2d20h
ingress-local-gateway       LoadBalancer    172.21.XX.XX    192.168.XX.XX     80:32537/TCP
2d20h
```

Note

- ingress-gateway indicates an external gateway that exposes applications to the Internet.
- ingress-local-gateway indicates an internal gateway that exposes applications only within a virtual private cloud (VPC).

2. Access the application.

   - Access the application over the Internet

- Access the application over HTTP.

  In the CLI, run the following command:

  ```
  curl -H  "Host: helloworld-go.default.example.com" http://8.131.XX.XX
  ```

  Expected output:

  ```
  Hello Knative!
  ```

- Access the application over HTTPS.

  In the CLI, run the following command:

  ```
  curl -H  "Host: helloworld-go.default.example.com" https://8.131.XX.XX -k
  ```

  Expected output:

  ```
  Hello Knative!
  ```

- Access the application within a VPC

  Note You must first enable Alibaba Cloud DNS PrivateZone.

  a. Run the following command to modify the *eci-profile* file:

  ```
  kubectl -n kube-system edit configmap eci-profile
  ```

  b. Set `enablePrivateZone` to `true`. Save the modification to *eci-profile* and exit.

  ```
  apiVersion: v1
  data:
    ...
    enablePrivateZone: "true"
    ...
  kind: ConfigMap
  metadata:
    name: eci-profile
    namespace: kube-system
  ```

  c. Access the application by using `Application name.namespace.svc.cluster.local`.

  The helloworld-go application in the default namespace is used as an example:

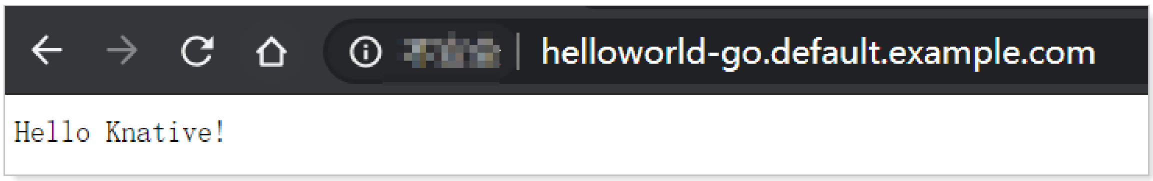
  ```
  http://helloworld-go.default.svc.cluster.local
  ```

3. Bind the IP address of the SLB instance to the domain name of Knative by modifying the hosts file. The following example shows how to bind the IP address to the domain name:

   Note The default root domain name of Knative is example.com. You can use a custom domain name. For more information, see Set a custom domain name for Knative Serving.

   ```
   106.15.2**.** helloworld-go.default.example.com
   ```

   If you can use the domain name to access the application after you modify the hosts file, the Knative Ingress gateway functions as normal.



Hello Knative!

16.6.4. Add a custom route

If you want to configure multiple domain names for a Knative Service, you can specify custom domain names and paths in the `knative.aliyun.com/serving-ingress` annotation. This way, you can use the custom domain names and paths to access the Knative Service. This topic describes how to add a custom domain name and a path for a Knative Service, and use the custom domain name and path to access the Knative Service.

Context

Knative has a default primary domain name. Each Knative Service has a unique domain name that is generated based on the primary domain name of Knative, the namespace, and the Service name. The format of a domain name is `{ksvc-name}.{namespace}.{knative-default-domain}`. The default primary domain name of Knative is `example.com`. If a Knative Service named coffee is deployed in the default namespace, the default domain name of the Knative Service is `coffee.default.example.com`. You can modify the default primary domain name `example.com` of Knative. For more information, see Set a custom domain name for Knative Serving.

If you want to configure multiple domain names for a Knative Service, specify custom domain names and paths in the `knative.aliyun.com/serving-ingress` annotation. For example, `knative.aliyun.com/serving-ingress: cafe.mydomain.com/coffee` specifies the `cafe.mydomain.com` domain name and the */coffee* path.

Note The custom domain name and path do not overwrite the default domain name. Both the default and custom domain names are available.

Procedure

1. Copy the following content to the *ingress-domain.yaml* file:

```
apiVersion: serving.knative.dev/v1
kind: Service
metadata:
  name: coffee-mydomain
  annotations:
    knative.aliyun.com/serving-ingress: cafe.mydomain.com/coffee
spec:
  template:
      spec:
      containers:
        - image: registry.cn-hangzhou.aliyuncs.com/knative-sample/helloworld-go:160e4dc
8
```

2. Create a Knative Service.

```
kubectl apply -f ingress-domain.yaml
```

3. View information about the Knative Service.

   The output shows that the default domain name of the Knative Service is `coffee-mydomain.default.example.com`.

   ```
   kubectl get ksvc
   ```

   Expected output:

   ```
   NAME              URL                                        LATESTCREATED
   LATESTREADY             READY   REASON
   coffee-mydomain   http://coffee-mydomain.default.example.com   coffee-mydomain-sbwhz
   coffee-mydomain-sbwhz   True
   ```

4. Use the default domain name to access the Knative Service.

   ```
   curl -H "Host: coffee-mydomain.default.example.com" http://alb-ijvf32ubve64wz****.cn-sh
   anghai.alb.aliyuncs.com
   ```

   Expected output:

   ```
   Hello World!
   ```

5. Use the custom domain name and path to access the Knative Service.

   ```
   curl -H "Host: cafe.mydomain.com" http://alb-ijvf32ubve64wz****.cn-shanghai.alb.aliyunc
   s.com/coffee
   ```

   Expected output:

   ```
   Hello World!
   ```

16.6.5. Configure access over HTTPS

The Application Load Balancer (ALB) Ingress controller supports automatic certificate discovery. ALB instances can discover certificates based on the domain name of Knative Services. This topic describes how to create a Transport Layer Security (TLS) certificate and use the certificate to configure access over HTTPS.

Prerequisites

An ALB Ingress is configured in Knative. For more information, see Use ALB Ingresses in Knative.

Procedure

You must create a TLS certificate, upload the certificate to the Certificate Management Service console, and then create a Knative Service that has TLS enabled. Then, the related ALB instance can automatically discover the TLS certificate based on the domain name of the Knative Service. Perform the following steps:

1. Run the following openssl commands to create a certificate:

```
openssl genrsa -out albtop-key.pem 4096
openssl req -subj "/CN=helloworld.default.knative.top" -sha256  -new -key albtop-key.pe
m -out albtop.csr
echo subjectAltName = DNS:helloworld.default.knative.top > extfile.cnf
openssl x509 -req -days 3650 -sha256 -in albtop.csr -CA ca.pem -CAkey ca-key.pem -CAcre
ateserial -out albtop-cert.pem -extfile extfile.cnf
```

2. Upload the certificate that you created to the Certificate Management Service console. For more information, see Upload a certificate.
3. Create a Knative Service that has TLS enabled with the following YAML template.

   Set `knative.k8s.alibabacloud/tls` to `true` to enable access over HTTPS.

```
apiVersion: serving.knative.dev/v1
kind: Service
metadata:
  name: helloworld
  namespace: default
  annotations:
    knative.k8s.alibabacloud/tls: "true"
spec:
  template:
    spec:
      containers:
      - image: registry-vpc.cn-shenzhen.aliyuncs.com/knative-sample/helloworld-go:73fbd
d56
        env:
        - name: TARGET
          value: "Knative"
```

4. Run the following command to access the Knative Service over HTTPS:

```
curl -H "host: helloworld.default.knative.top" https://alb-ppcate4ox6ge9m1wik.cn-shenzh
en.alb.aliyuncs.com -k
```

   Expected output:

```
Hello Knative!
```

   The output shows that the Knative Service can be accessed over HTTPS.

16.6.6. Configure a certificate for an SLB instance that serves as a gateway to establish HTTPS connections

Serverless Kubernetes (ASK) Knative uses Server Load Balancer (SLB) instances as gateways. The gateways support both HTTP and HTTPS. By default, Knative generates a self-signed certificate for HTTPS connections. This certificate can secure all domain names. Therefore, you can use the certificate to test applications. Before you use Knative to deploy applications, configure a certificate for HTTPS connections and specify the certificate ID for the Service of the Knative gateway by using annotations. This topic describes how to view, create, and use a certificate for HTTPS connections.

Context

For more information about Knative gateways, see Knative Gateway.

View the default certificate

1.
2. In the left-side navigation pane, choose **CLB (Formerly Known as SLB) > Certificates**.
3. On the **Certificates** page, you can view the certificate named knative-default-gateway-cert.

   The knative-default-gateway-cert certificate is the default certificate of Knative. The default certificate is automatically generated by Knative. You can use this certificate to test applications.

Create a certificate

You can use a certificate provided by Alibaba Cloud or upload a third-party certificate. For more information, see Use a certificate from Alibaba Cloud SSL Certificates Service or Upload a third-party certificate.

Use a certificate that you create

1.
2. In the left-side navigation pane, choose **CLB (Formerly Known as SLB) > Certificates**.
3. On the **Certificates** page, find the certificate that you want to use, move the pointer over the certificate ID, and then click the ❐ icon in the **Certificate Name/ID** column. The certificate ID is copied.
4. Change the value of `service.beta.kubernetes.io/alibaba-cloud-loadbalancer-cert-id` for the Service of the Knative gateway to the certificate ID that you obtained in step 3. After you specify the certificate ID by using the annotation, you can use the certificate for HTTPS connections.

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-protocol-port: "https:443"
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-cert-id: "${YOUR_CERT_ID}"
  name: nginx
spec:
  ports:
  - port: 443
    protocol: TCP
    targetPort: 80
  selector:
    run: nginx
  type: LoadBalancer
```

Manage multi-domain certificates

SLB supports multi-domain certificates. This allows you to secure multiple domain names by using a single certificate. You can configure multiple certificates for an SLB instance. For more information about how to manage domain names, see Add an additional certificate.

1.

2. In the left-side navigation pane, choose **CLB (FKA SLB) > Instances**.
3.
4. On the **Listener** tab, find the HTTPS listener that you create, and choose **> Manage Additional Certificate** in the Actions column.

   > **Note** On the **Listener** tab, if HTTPS is displayed in the **Frontend Protocol/Port** column, an HTTPS listener is configured.

5. In the **Manage Additional Certificate** panel, click **Add Additional Certificate**, enter an additional domain name, and then select a server certificate.

   The domain name can contain only letters, digits, hyphens (-), and periods (.). It must start with a letter or digit. To check whether the domain name is valid, use the domain detection tool.
6. After the configuration is completed, click **OK**.

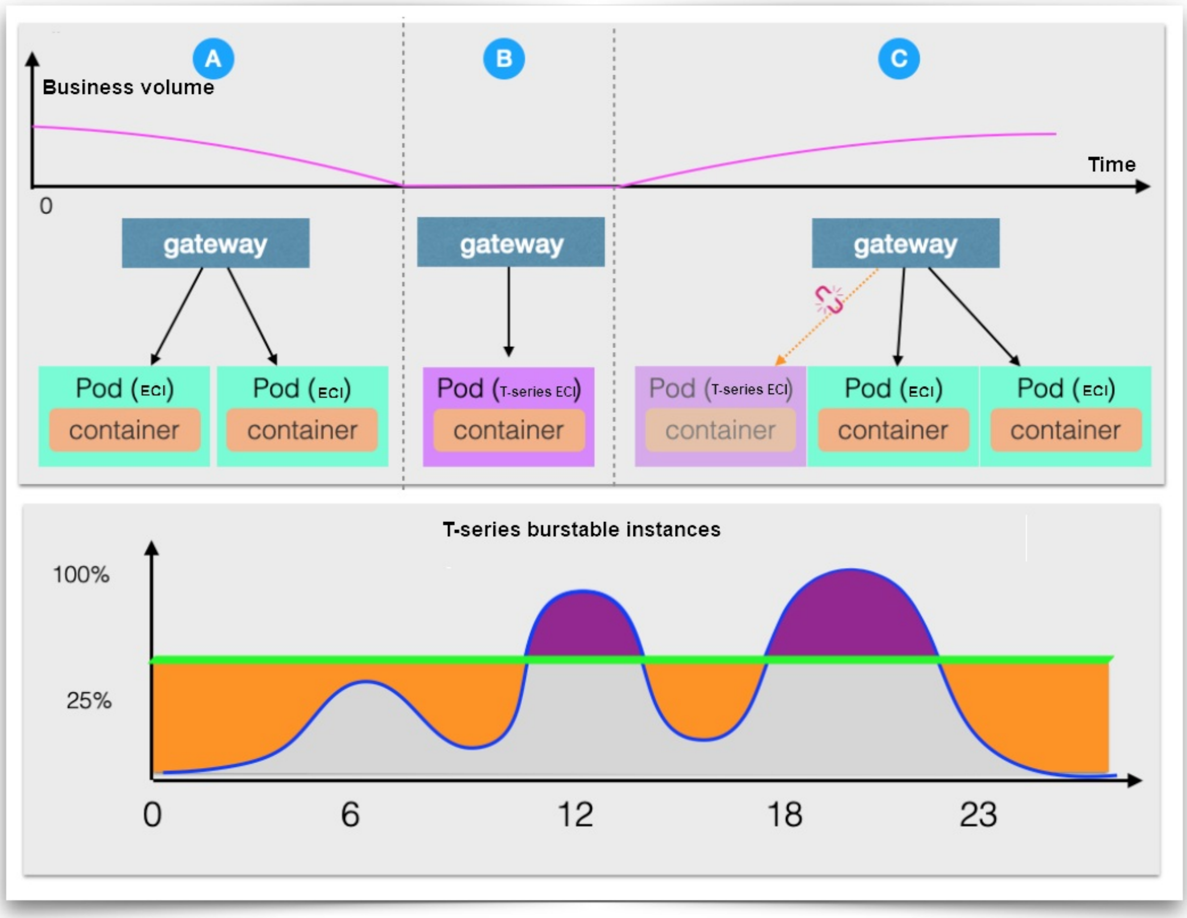
16.6.7. Reserved instances

The reserved instances feature is an exclusive feature of ASK Knative. The reserved instances can be used to avoid cold starts at low costs.

Benefits of reserved instances

By default, the Knative community version scales pods to zero during off-peak hours. This reduces the costs of running resident instances. However, after the pods are scaled to zero, a cold start occurs the next time when you start the application. To create a new instance, the system must first allocate infrastructures by using the Kubernetes scheduler. The system must also pull the application image and start the application. The image size and the time required to start an application vary based on the application developer and the service.

We recommend that you use reserved instances to avoid cold starts at low costs. ASK Knative does not scale pods to zero during off-peak hours. Instead, it reserves one pod. The type of the reserved instance can be different from the default instance type. For example, you can select an instance type that costs less than the default instance type.



How reserved instances work

ASK Knative uses reserved instances to replace compute optimized instances during off-peak hours. When ASK Knative receives a request, it uses the reserved instances to process the request and creates standard compute optimized instances in the ASK cluster. After the compute optimized instances are created, all new requests are forwarded to these instances. Reserved instances are released after they process all the requests that are sent to them. This seamless switchover mechanism reduces the costs of resident instances and avoids cold starts.

Reserved instance types

Alibaba Cloud Elastic Container Instance (ECI) provides multiple types of instances, such as burstable instances. Burstable instances use CPU credits to ensure computing performance and apply to scenarios where the CPU usage is low in general but bursts on occasion. A burstable instance can accumulate CPU credits. If a burstable instance fails to handle a heavy workload, it consumes CPU credits to improve computing performance. This mechanism does not affect the environment or applications that are running on the instance. Compared with other types of instances, a burstable instance provides a more flexible and cost-effective way to use CPU resources. The CPU credit mechanism allows you to minimize the consumption of resources during off-peak hours and scale out resources during peak hours. This reduces the cost of resources.

The cost-effectiveness and CPU credit features make burstable instances appropriate reserved instances. The default types of reserved instances in ASK Knative include ecs.t6-c1m1.large, ecs.t5-lc1m2.small, ecs.s6-c1m2.small, ecs.t6-c1m2.large, and ecs.n1.small. They are the lowest-priced instance types among all instance types that have at least one CPU core and 2 GB memory. For more information about the ECI instance types and prices, see ECI pricing.

Note You can configure up to five instance types for an ECI.

You can manually configure the types of reserved instances by using the `knative.aliyun.com/reserve-instance-eci-use-specs` annotation. The following list describes how to configure the instance types:

- Specify the types of ECIs. In the following example, ecs.t6-c1m1.large and ecs.t5-lc1m2.small are specified.

```
apiVersion: serving.knative.dev/v1
kind: Service
metadata:
  name: hello-spec-1
spec:
  template:
    metadata:
      annotations:
        knative.aliyun.com/reserve-instance-eci-use-specs: "ecs.t6-c1m1.large,ecs.t5-lc1m
2.small"
    spec:
      containers:
        - image: registry.cn-hangzhou.aliyuncs.com/knative-sample/helloworld-go:160e4dc8
```

- Specify the minimum CPU cores and memory. ASK Knative automatically searches for instance types that meet the requirements and uses the five lowest-priced instance types to create instances. In the following example, `1-2Gi` indicates 1-core 2 GB.

```
apiVersion: serving.knative.dev/v1
kind: Service
metadata:
  name: hello-spec-2
spec:
  template:
    metadata:
      annotations:
        knative.aliyun.com/reserve-instance-eci-use-specs: "1-2Gi"
    spec:
      containers:
        - image: registry.cn-hangzhou.aliyuncs.com/knative-sample/helloworld-go:160e4dc8
```

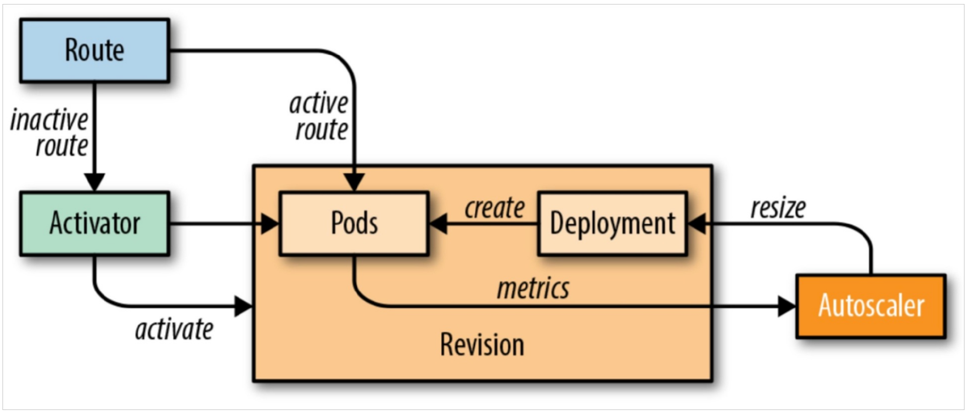
Knative creates instances based on a list of ECI instance types. The instance types are listed in an sequential order. If an instance type is unavailable, the next instance type in the list is used.

16.6.8. Enable auto scaling to withstand traffic fluctuations

Knative Pod Autoscaler (KPA) is an out-of-the-box feature that can scale pods for your application to withstand traffic fluctuations. This topic describes how to enable auto scaling of pods based on the number of requests.

Context

Knative Serving adds the Queue-Proxy container to each pod. The Queue-Proxy container sends concurrency metrics of the application containers to KPA. After KPA receives the metrics, KPA automatically adjusts the number of pods provisioned for a Deployment based on the number of concurrent requests and related algorithms.



Concurrency and QPS

- The number of concurrent requests refers to the number of requests received by a pod at the same time.
- Queries per second (QPS) refers to the number of requests that can be handled by a pod per second. It also shows the maximum throughput of a pod.

The increase of concurrent requests does not necessarily increase the QPS. In scenarios where the access load is heavy, if the concurrency is increased, the QPS may be decreased. This is because the system is overloaded and the CPU and memory usage is high, which downgrades the performance and increases the response latency.

Algorithms

KPA performs auto scaling based on the average number of concurrent requests per pod. This value is specified by the concurrency target. The default concurrency target is 100. The number of pods required to handle requests is determined based on the following formula: Number of pods = Total number of concurrent requests to the application/Concurrency target. For example, if you set the concurrency target to 10 and send 50 concurrent requests to an application, KPA creates five pods.

KPA provides two auto scaling modes: stable and panic.

- Stable

  In stable mode, KPA adjusts the number of pods provisioned for a Deployment to match the specified concurrency target. The concurrency target indicates the average number of requests received by a pod within a stable window of 60 seconds.

- Panic

KPA calculates the average number of concurrent requests per pod within a stable window of 60 seconds. Therefore, the number of concurrent requests must remain at a specific level for 60 seconds. KPA also calculates the number of concurrent requests per pod within a panic window of 6 seconds. If the number of concurrent requests reaches twice the concurrency target, KAP switches to the panic mode. In panic mode, KPA scales pods within a shorter time window than in stable mode. After the burst of concurrent requests lasts for 60 seconds, KPA automatically switches back to the stable mode.

```
                                                       |
                                  Panic Target--->  +--| 20
                                                    |  |
                                                    | <------Panic Window
                                                    |  |
       Stable Target--->  +-------------------------|--| 10   CONCURRENCY
                          |                         |  |
                          |                      <-----------Stable Window
                          |                         |  |
--------------------------+-------------------------+--+ 0
120                       60                           0
                     TIME
```

KPA configurations

- To configure KPA, you must configure config-autoscaler. By default, config-autoscaler is configured. The following content describes the key parameters.

  Run the following command to query config-autoscaler:

  ```
  kubectl -n knative-serving get cm config-autoscaler
  ```

  Expected output:

  ```
  apiVersion: v1
  kind: ConfigMap
  metadata:
   name: config-autoscaler
   namespace: knative-serving
  data:
   container-concurrency-target-default: "100"
   container-concurrency-target-percentage: "0.7"
   enable-scale-to-zero: "true"
   max-scale-up-rate: "1000"
   max-scale-down-rate: "2"
   panic-window-percentage: "10"
   panic-threshold-percentage: "200"
   scale-to-zero-grace-period: "30s"
   scale-to-zero-Pod-retention-period: "0s"
   stable-window: "60s"
   target-burst-capacity: "200"
   requests-per-second-target-default: "200"
  ```

- Configure parameters related to the scale-to-zero feature.
  - scale-to-zero-grace-period: The time period for which `inactive revison` keeps running before KPA scales the number of pods to zero. The minimum period is 30 seconds.

    ```
    scale-to-zero-grace-period: 30s
    ```

  - stable-window: In stable mode, KPA scales pods based on the average number of concurrent requests per pod within the stable window.

    ```
    stable-window: 60s
    ```

    You can also specify the stable window by using an annotation in the Deployment revision.

    ```
    autoscaling.knative.dev/window: 60s
    ```

  - enable-scale-to-zero: Set `enable-scale-to-zero` to `true` .

    ```
    enable-scale-to-zero: "true"
    ```

- Set a concurrency target for KPA.
  - `target`

    The `target` parameter sets a soft limit on the number of concurrent requests handled by each pod within a specified time period. We recommend that you use this parameter to control the concurrency. By default, the concurrency target in the ConfigMap is set to 100.

    ```
    `container-concurrency-target-default: 100`
    ```

    You can change the value by using the `autoscaling.knative.dev/target` per-revision annotation.

    ```
    autoscaling.knative.dev/target: 50
    ```

  - `containerConcurrency`

    The `containerConcurrency` parameter sets a hard limit on the number of concurrent requests handled by each pod within a specified time period. You can configure this parameter in the template section of the revision configuration file.

    Note

    To use `containerConcurrency`, make sure that the following conditions are met:

    - Use `containerConcurrency` only when you want to limit the number of concurrent requests handled by an application within a specified time period.
    - Use `containerConcurrency` only when you want to forcibly control the concurrency of an application.

    ```
    containerConcurrency: 0 | 1 | 2-N
    ```

    - 1: Only one request is handled at a time by each pod created based the current revision.
    - 2-N: Two or more concurrent requests are handled by each pod at a time.
    - 0: The number of concurrent requests handled by each pod is not limited. The number depends on the system.

  - container-concurrency-target-percentage

    This parameter is also known as a target utilization value or a concurrency factor. It is used in the calculation of the concurrency target for auto scaling. For example, the concurrency target or the `containerConcurrency` parameter is set to 100 and the target utilization value is set to 0.7. KPA adds pods to the application when the average number of concurrent requests per pod reaches 70 (100 × 0.7).

    The following formula applies: Concurrency value that triggers auto scaling = Concurrency target (or the value of `containerConcurrency`) × The value of container-concurrency-target-percentage.

- Set the scale bounds. You can use the `minScale` and `maxScale` parameters to set the scale bounds of pods that are provisioned for an application. You can use these two parameters to reduce cold starts and computing costs.

Note

- If you do not set the `minScale` annotation, all pods are removed when no traffic arrives. If you set `enable-scale-to-zero` to `false` in the config-autoscaler ConfigMap, KPA scales the number of pods to one when no traffic arrives.
- If you do not set the `maxScale` annotation, the number of pods that can be provisioned for the application is unlimited.

You can configure the `minScale` and `maxScale` parameters in the template section of the revision configuration file.

```
spec:
  template:
    metadata:
      autoscaling.knative.dev/minScale: "2"
      autoscaling.knative.dev/maxScale: "10"
```

Scenario 1: Enable auto scaling by setting a concurrency target

This example shows how to enable KPA to perform auto scaling by setting a concurrency target.

1. Enable Knative.
2. Create an *autoscale-go.yaml* file.

   Set the maximum number of concurrent requests per pod to 10.

   ```
   apiVersion: serving.knative.dev/v1
   kind: Service
   metadata:
     name: autoscale-go
     namespace: default
   spec:
     template:
       metadata:
         labels:
           app: autoscale-go
         annotations:
           autoscaling.knative.dev/target: "10"
       spec:
         containers:
           - image: registry.cn-hangzhou.aliyuncs.com/knative-sample/autoscale-go:0.1
   ```
3. Run the following command to query the Ingress gateway:

   ```
   kubectl -n knative-serving get svc
   ```

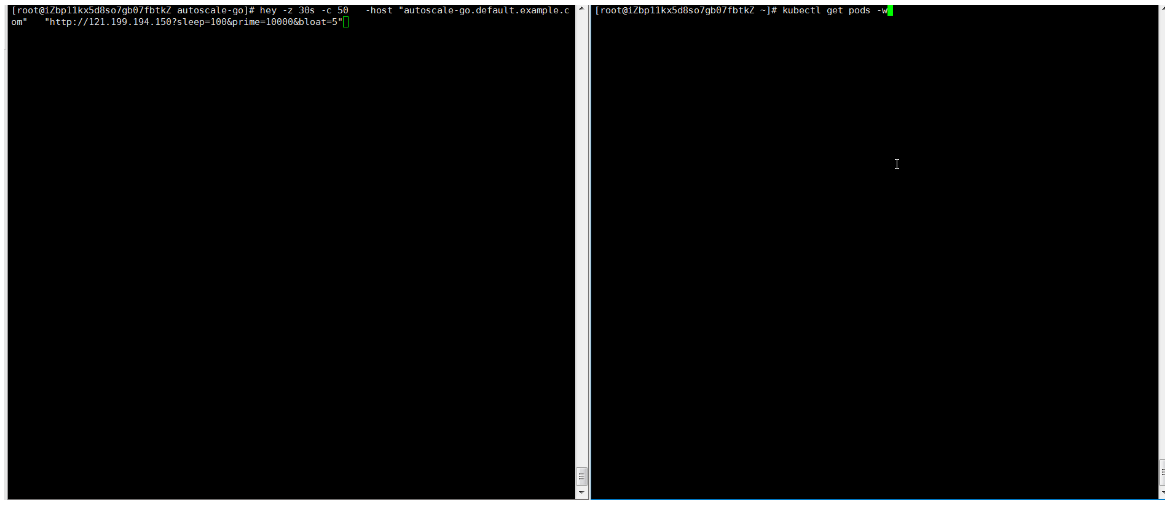
   Expected output:

   ```
   NAME               TYPE           CLUSTER-IP      EXTERNAL-IP      PORT(S)
   AGE
   ingress-gateway    LoadBalancer   172.19.1*.***   121.199.19*.***    80:32185/TCP,443:311
   37/TCP    69d
   ```
4. Use the load testing tool hey to send 50 concurrent requests to the application within 30 seconds.

Note For more information about hey, see hey.

```
hey -z 30s -c 50   -host "autoscale-go.default.example.com"   "http://121.199.194.150?s
leep=100&prime=10000&bloat=5"
```

Expected output:



```
[root@iZbp11kx5d8so7gb07fbtkZ autoscale-go]# hey -z 30s -c 50   -host "autoscale-go.default.example.c
om"   "http://121.199.194.150?sleep=100&prime=10000&bloat=5"
```

```
[root@iZbp11kx5d8so7gb07fbtkZ ~]# kubectl get pods -w
```

The output indicates that five pods are added.

Scenario 2: Enable auto scaling by setting scale bounds

Scale bounds control the minimum and maximum numbers of pods that can be provisioned for an application. This example shows how to enable auto scaling by setting scale bounds.

1. Enable Knative.
2. Create an *autoscale-go.yaml* file.

   Set the concurrency target to 10, `minScale` to 1, and `maxScale` to 3.

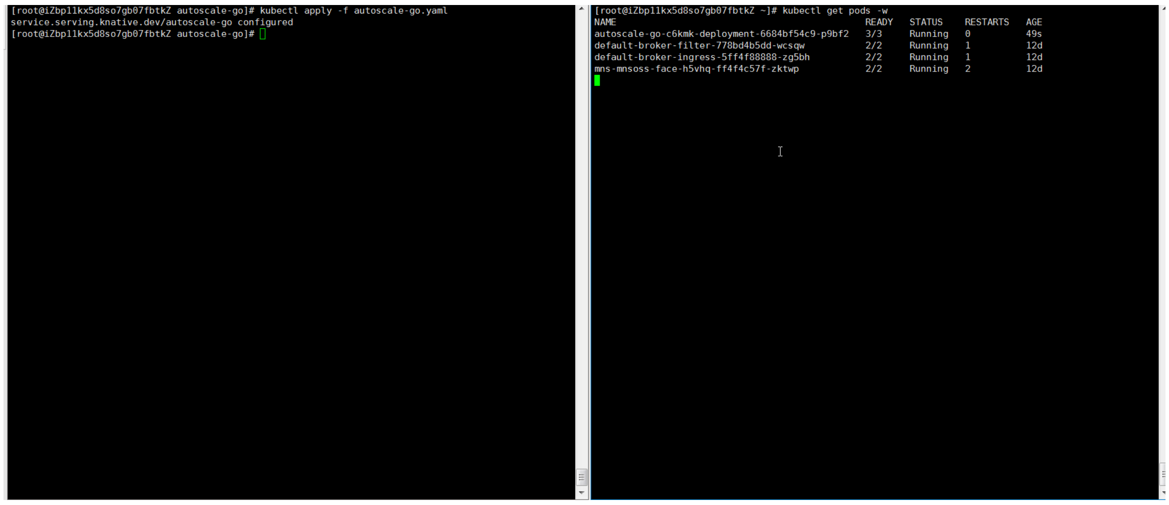
```
apiVersion: serving.knative.dev/v1alpha1
kind: Service
metadata:
  name: autoscale-go
  namespace: default
spec:
  template:
    metadata:
      labels:
        app: autoscale-go
      annotations:
        autoscaling.knative.dev/target: "10"
        autoscaling.knative.dev/minScale: "1"
        autoscaling.knative.dev/maxScale: "3"
    spec:
      containers:
        - image: registry.cn-hangzhou.aliyuncs.com/knative-sample/autoscale-go:0.1
```

3. Use hey to send 50 concurrent requests to the application within 30 seconds.

Note For more information about hey, see hey.

```
hey -z 30s -c 50   -host "autoscale-go.default.example.com"   "http://121.199.194.150?sleep=100&prime=10000&bloat=5"
```

Expected output:



```
[root@iZbp11kx5d8so7gb07fbtkZ autoscale-go]# kubectl apply -f autoscale-go.yaml
service.serving.knative.dev/autoscale-go configured
[root@iZbp11kx5d8so7gb07fbtkZ autoscale-go]#
```

```
[root@iZbp11kx5d8so7gb07fbtkZ ~]# kubectl get pods -w
NAME                                           READY   STATUS    RESTARTS   AGE
autoscale-go-c6kmk-deployment-6684bf54c9-p9bf2   3/3     Running   0          49s
default-broker-filter-778bd4b5dd-wcsqw           2/2     Running   1          12d
default-broker-ingress-5ff4f88888-zg5bh          2/2     Running   1          12d
mns-mnsoss-face-h5vhq-ff4f4c57f-zktwp            2/2     Running   2          12d
```

A maximum of three pods are added. One pod is reserved when no traffic flows to the application.

16.6.9. Use HPA in Knative

You can use Horizontal Pod Autoscaler (HPA) in Knative to automatically scale the number of resources. You can set the threshold of the CPU metric for a Knative service. This ensures that resources are automatically scaled to handle heavy workloads. This topic describes how to use HPA in Knative.

Prerequisites

Enable Knative

Procedure

1. Create the *ksvc-hpa.yaml* file.

   Configure an HPA scaling policy for a Knative service. The following code block is an example.

```
apiVersion: serving.knative.dev/v1
kind: Service
metadata:
  name: helloworld-go-hpa
spec:
  template:
    metadata:
      labels:
        app: helloworld-go-hpa
      annotations:
        autoscaling.knative.dev/class: "hpa.autoscaling.knative.dev"
        autoscaling.knative.dev/metric: "cpu"
        autoscaling.knative.dev/target: "75"
        autoscaling.knative.dev/minScale: "1"
        autoscaling.knative.dev/maxScale: "10"
    spec:
      containers:
        - image: registry.cn-hangzhou.aliyuncs.com/knative-samples/helloworld-go:160e4d
c8
          resources:
            requests:
              cpu: '200m'
```

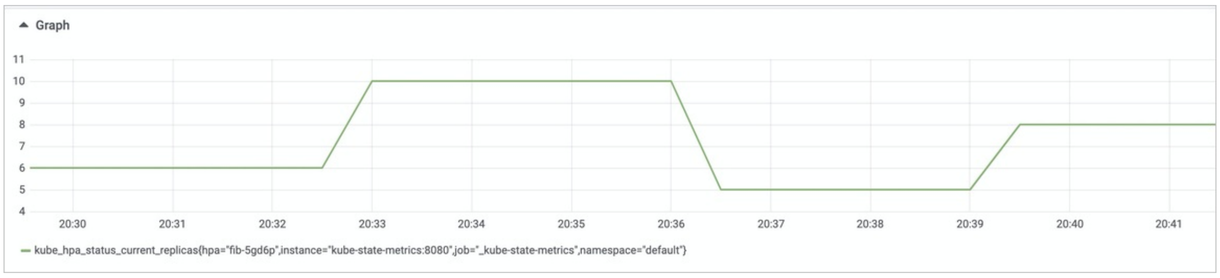
- `autoscaling.knative.dev/class: "hpa.autoscaling.knative.dev"` specifies the HPA plug-in.
- `autoscaling.knative.dev/metric` specifies the CPU metric.
- `autoscaling.knative.dev/target` specifies the threshold of the CPU metric.
- `autoscaling.knative.dev/minScale: "1"` specifies the minimum number of instances in the scaling policy.
- `autoscaling.knative.dev/maxScale: "10"` specifies the maximum number of instances in the scaling policy.

2. Apply the scaling policy.

```
kubectl apply -f ksvc-hpa.yaml
```

Result

After HPA is enabled for the Knative service, the number of instances is automatically scaled, as shown in the following figure.



16.6.10. Use AHPA to enable predictive scaling in Knative

Knative allows you to use Advanced Horizontal Pod Autoscaler (AHPA) in serverless Kubernetes (ASK) clusters. If your application requests resources in a periodic pattern, you can use AHPA to predict changes in resource requests and prefetch resources for scaling activities. This reduces the impact of cold starts when your application is scaled. This topic describes how to use AHPA to enable predictive scaling in Knative.

Prerequisites

- AHPA is in invitational preview. to apply to be added to a whitelist.
- Knative is deployed in your cluster. For more information, see Enable Knative.
- Application Real-Time Monitoring Service (ARMS) Prometheus is enabled. For more information, see Enable ARMS Prometheus.

Step 1: Install Application Intelligence Controller

Install Application Intelligence Controller from the Add-ons page of the Container Service for Kubernetes (ACK) console.

1.
2.
3.
4.
5. On the **Add-ons** page, click the **Others** tab. Find Application Intelligence Controller and click **Install**.
6. In the **Install Application Intelligence Controller** message, click **OK**.

Step 2: Configure Prometheus to collect metrics

Configure Prometheus to collect the response time (RT) and requests per second (RPS) metrics of your Knative Service.

a. Configure metric collection rules

1.
2. In the left-side navigation pane, choose **Prometheus Monitoring > Prometheus Instances**.
3. In the top navigation bar, select the region in which your cluster resides and click the name of the Prometheus instance that is used to monitor your cluster in the **Instance Name** column. The details page of the Prometheus instance appears.
4. In the left-side navigation pane, click **Service Discovery**. Then, click the **Configure** tab.
5. On the **Configure** tab, click **Custom Service Discovery**. Then, click **Add**.
6. In the **Add custom service discovery** dialog box, configure metric collection rules.

   Example:

```
job_name: queue-proxy
scrape_interval: 3s
scrape_timeout: 3s
kubernetes_sd_configs:
- role: pod
relabel_configs:
- source_labels:
  - __meta_kubernetes_pod_label_serving_knative_dev_revision
  - __meta_kubernetes_pod_container_port_name
  action: keep
  regex: .+;http-usermetric
- source_labels:
  - __meta_kubernetes_namespace
  target_label: namespace
- source_labels:
  - __meta_kubernetes_pod_name
  target_label: pod
- source_labels:
  - __meta_kubernetes_service_name
  target_label: service
```

b. Add ARMS Prometheus as a data source

1.
2. In the left-side navigation pane, choose **Prometheus Monitoring > Prometheus Instances**.
3. In the upper-left corner of the **Prometheus Monitoring** page, select the region in which your Prometheus instances are deployed. Then, click the name of a Prometheus instance whose **Instance Type** is **Prometheus for Container Service**. The details page of the Prometheus instance appears.
4. In the left-side navigation pane of the instance details page, click **Settings** and copy the public endpoint in the HTTP API Address section.
5. In the HTTP API Address section, click **Generate Token** to generate a token. The token is used to pass the authentication when you access the Prometheus instance.
6. Create an *application-intelligence.yaml* file with the following content. The file specifies the public endpoint of the Prometheus instance.

   ```
   apiVersion: v1
   kind: ConfigMap
   metadata:
     name: application-intelligence
     namespace: kube-system
   data:
     armsUrl: "https://cn-hangzhou.arms.aliyuncs.com:9443/api/v1/prometheus/da9d7dece901db
   4c9fc7f5b9c40****/158120454317****/cc6df477a982145d986e3f79c985a****/cn-hangzhou"
     token: "****"
   ```

   Descriptions of parameters:

   - `armsUrl` : Specify the public endpoint of the Prometheus instance that you copied in Step 4.
   - `token` : Specify the token that was generated in Step 5.
7. Run the following command to create a ConfigMap:

```
kubectl apply -f application-intelligence.yaml
```

Step 3: Create a Knative Service

Create a Knative Service that has AHPA enabled.

1. Create an *autoscale-go.yaml* with the following content:

```
apiVersion: serving.knative.dev/v1
kind: Service
metadata:
  name: autoscale-go
  namespace: default
spec:
  template:
    metadata:
      labels:
        app: autoscale-go
      annotations:
        autoscaling.knative.dev/class: ahpa.autoscaling.knative.dev
        autoscaling.knative.dev/target: "500"
        autoscaling.knative.dev/metric: "rt"
        autoscaling.knative.dev/minScale: "1"
        autoscaling.knative.dev/maxScale: "30"
        autoscaling.alibabacloud.com/scaleStrategy: "observer"
    spec:
      containers:
        - image: registry.cn-hangzhou.aliyuncs.com/knative-sample/autoscale-go:0.1
```

Parameter	Description
`autoscaling.knative.dev/class: ahpa.autoscaling.knative.dev`	Specifies that AHPA is enabled.
`autoscaling.knative.dev/metric: "rt"`	The metric based on which AHPA scales your application. Only the RT metric is supported.
`autoscaling.knative.dev/target: "500`	The scaling threshold for the metric. In this example, the threshold is set to 500 milliseconds.
`autoscaling.knative.dev/minScale: "1"`	The minimum number of pods to which AHPA can scale your application. In this example, the value is set to 1.
`autoscaling.knative.dev/maxScale: "30"`	The maximum number of pods to which AHPA can scale your application. In this example, the value is set to 30.

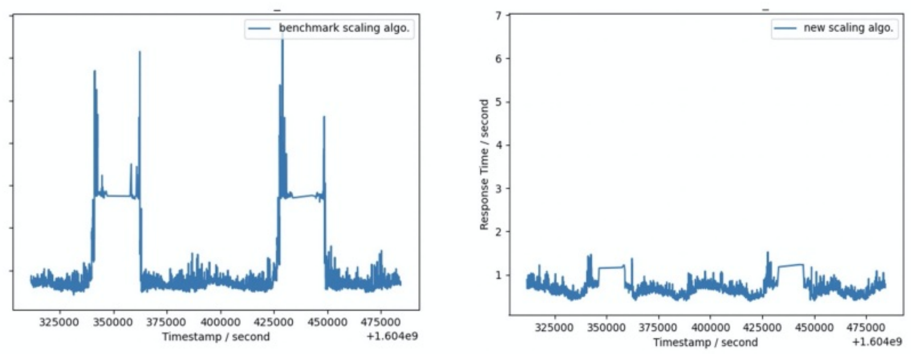
Parameter	Description
`autoscaling.alibabacloud.com/scaleStrategy: "observer"`	The scaling mode of AHPA. Default value: `observer`. ◦ `observer`: AHPA observes but does not scale your application. You can use this mode to check whether AHPA works as expected. The default mode is `observer` because AHPA performs predictive scaling based on the historical data with the last seven days. ◦ `auto`: AHPA scales your application based on the collected metric and the specified threshold.

2. Run the following command to enable AHPA:

```
kubectl apply -f autoscale-go.yaml
```

Verify the AHPA policy

You can compare the values of the RT metric of your Knative Service before and after AHPA predictive scaling is enabled, as shown in the following figure.



16.6.11. Configure schedule-based autoscaling in Knative

Serverless Kubernetes (ASK) clusters support schedule-based autoscaling. This feature allows you to create schedule-based autoscaling policies for the system to allocate resources. This feature is integrated with the resource scaling capabilities of Horizontal Pod Autoscaler (HPA). This ensures that the system can automatically allocate sufficient resources to cope with spikes in workloads. This topic describes how to configure and apply a schedule-based autoscaling policy in Knative.

Prerequisites

Enable Knative

Procedure

1. Configure a schedule-based autoscaling policy.

You can use a ConfigMap to define a schedule-based autoscaling policy. Each Knative Service is associated with one policy. The following code block shows a configuration example:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cron-autoscaler
  namespace: default
data:
  jobs: |
   - name: "workday"
     schedule: "* * 1-5"
     timeseries:
     - timeSlice: 01:30:30
       replicas: 1
     - timeSlice: 05:30:30
       replicas: 2
     - timeSlice: 07:30:30
       replicas: 5
     - timeSlice: 10:24:30
       replicas: 8
     - timeSlice: 11:47:30
       replicas: 10
     - timeSlice: 13:17:30
       replicas: 6
     - timeSlice: 16:50:30
       replicas: 9
     - timeSlice: 20:17:30
       replicas: 5
     - timeSlice: 23:30:30
       replicas: 1
   - name: "holiday"
     schedule: "* * 0,6"
     timeseries:
     - timeSlice: 08:24:30
       replicas: 4
     - timeSlice: 11:47:30
       replicas: 3
     - timeSlice: 13:17:30
       replicas: 2
```

- In the `schedule` field, the first number indicates the day of a month. The second number indicates the month of a year. The remaining indicates the days of a week. The format of the value is `* * 1-5`. In the preceding example, 1-5 indicates Monday to Friday, and 0,6 indicates Saturday and Sunday.
- The format of the value in the `timeseries` field is `hour: minute: second`.

Note

- Policy updates apply to all Knative Services that are associated with the policy.
- If you delete the policy, the change does not apply to the Knative Services that are associated with the policy. The Knative Services do not scale the number of resources to zero.

2. Apply the schedule-based autoscaling policy.

   After you configure a schedule-based autoscaling policy, you can apply the policy to Knative Services.

```
apiVersion: serving.knative.dev/v1
kind: Service
metadata:
  name: helloworld-go-ppa
  annotations:
    alicloud.autoscaling.service.knative.dev/timeseries: "cron-autoscaler"
spec:
  template:
    metadata:
      labels:
        app: helloworld-go-ppa
      annotations:
        autoscaling.knative.dev/class: "ppa.autoscaling.knative.dev"
        alicloud.autoscaling.knative.dev/metric-target: "cpu:50"
    spec:
      containers:
        - image: registry.cn-hangzhou.aliyuncs.com/knative-samples/helloworld-go:160e4d
c8
          resources:
            requests:
              cpu: '200m'
```

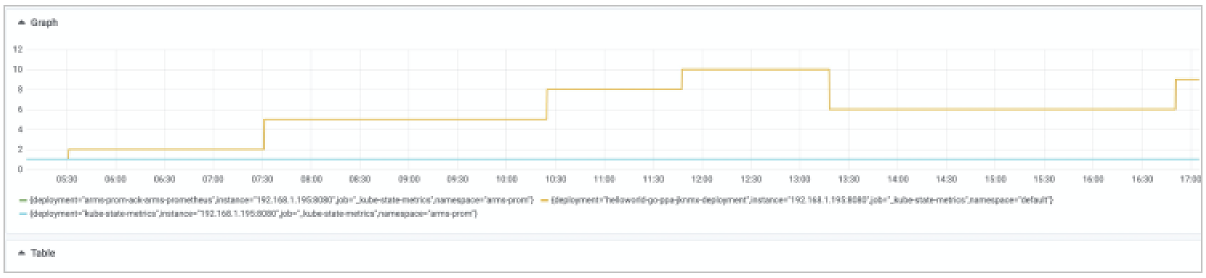
   - Set `alicloud.autoscaling.service.knative.dev/timeserie` to specify the policy name.
   - Set `autoscaling.knative.dev/class: "ppa.autoscaling.knative.dev"` to specify the schedule-based autoscaling component.
   - Specify multiple HPA metrics. Open source Knative allows you to specify only one HPA metric. In ASK Knative, you can specify multiple HPA metrics by setting `alicloud.autoscaling.knative.dev/metric-target`. For example, you can specify the number of CPU cores and the amount of memory resources by setting the parameter to `cpu:50,memory:50`.
   - Set `autoscaling.knative.dev/minScale: "1"` to specify the minimum number of pods that must be guaranteed.
   - Set `autoscaling.knative.dev/maxScale: "10"` to specify the maximum number of pods that are allowed.

ⓘ **Note** The following formulas are used to determine the maximum (`maxScale`) and minimum (`minScale`) numbers of pods for schedule-based autoscaling:

- Minimum number of pods = min(autoscaling.knative.dev/minScale,Number of pods specified in the schedule-based autoscaling policy)
- Maximum number of pods = max(autoscaling.knative.dev/maxScale,Number of a resource specified in the schedule-based autoscaling policy)

Verify the result

Check the changes in the number of pods after you apply the schedule-based autoscaling policy. The following trend chart shows an example.



16.6.12. Use Knative to deploy a function as a Service

Knative is integrated with Function Compute. You can use Knative to deploy a function as a Service by using the following methods: create a deployment file that contains the function code, pull the function code file from Object Storage Service (OSS), and use a container image. This topic describes how to use Knative to deploy a function as a Service.

Prerequisites

- ASK quick start
- Enable Knative

Create a deployment file that contains the function code

1. Create a file named *coffee.yaml*.

```
apiVersion: serving.knative.dev/v1
kind: Service
metadata:
  name: coffee
  annotations:
    workload.serving.knative.aliyun.com/class: "fc"
spec:
  template:
    metadata:
      annotations:
        fc.revision.serving.knative.aliyun.com/code-space: "inline"
    spec:
      containers:
        - image: nodejs8
          command:
          - |
            var getRawBody = require('raw-body')
            var version = 'coffee-default'
            module.exports.handler = function (request, response, context) {
                var respBody = new Buffer('Hello ' + version + '\n')
                response.setStatusCode(200)
                response.setHeader('content-type', 'application/json')
                response.send(respBody)
            };
```

The `image` parameter specifies that Node.js is used to deploy the function and the `command` parameter specifies the code of the function. Modify these parameters based on the function that you want to deploy.

2. Deploy the function as a Service.

```
kubectl apply -f coffee.yaml
```

Pull the function code file that is archived by Function Compute from OSS

You can deploy a function as a Service by downloading the archive of the function code file from OSS. The function code file is archived by Function Compute.

1. Create a file named *coffee-oss.yaml*.

```
apiVersion: serving.knative.dev/v1
kind: Service
metadata:
  name: coffee
  annotations:
    workload.serving.knative.aliyun.com/class: "fc"
spec:
  template:
    metadata:
      annotations:
        fc.revision.serving.knative.aliyun.com/code-space: "oss"
    spec:
      containers:
        - image: nodejs8
          command:
          - fc-zip
          - knative-demo/node-demo.zip
```

The `image` and `command` parameters specify that a function code file that is archived by Function Compute is pulled from OSS. The following content describes the `image` and `command` parameters:

- image: the runtime of the function, for example, nodejs8.
- command: an array that consists of BucketName and ObjectName.

2. Deploy the function as a Service.

```
kubectl apply -f coffee-oss.yaml
```

Use a container image

Function Compute provides a custom container runtime to simplify developer experience and improve the efficiency of development and delivery. Developers can interact with Function Compute over HTTP and deliver functions as container images. For more information about the custom container runtime, see Overview.

Limits

Configuration item	Limit
Image size	• If the memory required to run a function is less than 1 GB, the size of the container image cannot exceed 256 MB before it is decompressed. • If the memory required to run a function is 1 GB or greater, the size of the container image cannot exceed 1,024 MB before it is decompressed.
Image repository	Only image repositories of Container Registry Personal Edition are supported. Other image repositories will soon be supported. For more information about Container Registry Personal Edition, see What is Container Registry.

Configuration item	Limit
Supported images	Only images in a private image repository that belongs to the same account and region as the Service are supported. Public images will soon be supported.
Read and write permissions of the container	The run-as-user UID of the container is randomly selected from 10000 to 10999. By default, the container has the write permissions on the */tmp* directory. The read and write permissions on other directories depend on the file system of the container image. If a UID does not have permissions to read or write a file, you can modify the Dockerfile.
Storage space limit of the writable container layer	The data that is generated by the writable container layer cannot exceed 512 MB.
Permission policies	When you create a Service, you must attach the AliyunContainerRegistryReadOnlyAccess or AliyunContainerRegistryFullAccess permission policy to the service linked role. You must create a service linked role in Resource Access Management (RAM) and then attach the AliyunContainerRegistryReadOnlyAccess or AliyunContainerRegistryFullAccess permission policy to the service linked role. In the **Basic Information** section on the RAM Roles page, you can obtain the Alibaba Cloud Resource Name (ARN). For more information, see Grant permissions to a RAM role and RAM role overview.

1. Create a file named *coffee-image.yaml*.

```
apiVersion: serving.knative.dev/v1
kind: Service
metadata:
  name: coffee
  annotations:
    workload.serving.knative.aliyun.com/class: "fc"
spec:
  template:
    metadata:
      annotations:
        fc.revision.serving.knative.aliyun.com/code-space: "image"
        fc.revision.serving.knative.aliyun.com/role-arm: "acs:ram::1041208914252405:rol
e/fc-yuanyi-test"
    spec:
      containers:
        - image: registry.cn-shenzhen.aliyuncs.com/aliknative/nodejs-express:v3.0
```

The `annotations` and `image` parameters specify that a container image is used to deploy the function as a Service. The following describes the parameters:

- fc.revision.serving.knative.aliyun.com/code-space: image: the type of container image.
- fc.revision.serving.knative.aliyun.com/role-arm: the ARN that is used to attach the AliyunContainerRegistryReadOnlyAccess or AliyunContainerRegistryFullAccess permission policy.
- image: the name of the container image.

2. Deploy the function as a Service.

```
kubectl apply -f coffee-oss.yaml
```

Verify the result

You can check whether a function is deployed as a Service. In the following example, the function is deployed by using a deployment file that contains the function code.

1. Query the status of the Service.

```
kubectl get ksvc
```

Expected output:

```
NAME      URL
LATESTCREATED    LATESTREADY     READY    REASON
coffee    https://198639303048****.cn-beijing.fc.aliyuncs.com/2016-08-15/proxy/kn_defaul
t_coffee.http-prd/kn_default_coffee/    coffee-5bqdr     coffee-5bqdr    True
```

2. Run the curl command to access the Service.

```
curl https://198639303048****.cn-beijing.fc.aliyuncs.com/2016-08-15/proxy/kn_default_co
ffee.http-prd/kn_default_coffee/
```

Expected output:

```
Hello coffee-default
```

3. Check the Service and function in the Function Compute console.
   i.
   ii.
   iii. In the left-side navigation pane, click **Services and Functions**. On the **Services and Functions** page, the kn_default_coffee Service appears.
   iv. Click kn_default_coffee. On the **Functions** tab, the kn_default_coffee function appears.

16.6.13. Deploy a canary release for a Knative Service

Knative supports canary releases. You can deploy a canary release to split traffic between two application versions based on a specified ratio. This topic describes how to deploy a canary release for a Knative Service.

Prerequisites

Deploy Knative:

- If you want to deploy Knative in a Container Service for Kubernetes (ACK) cluster, see Deploy Knative.
- If you want to deploy Knative in a serverless Kubernetes (ASK) cluster, see Enable Knative.

Step 1: Create a Knative Service

1.

2.
3.
4.
5. On the **Services** tab, click **Create Service** in the upper-right corner.
6. On the Create Service page, set the **Namespace** and **Service Name** parameters. Then, select an image and an image version.

Parameter	Description
Namespace	Select the namespace to which the Service belongs.
Service Name	Enter a name for the Service. In this example, *helloworld-go* is used.
Image Name	To select an image, click **Select Image**. In the dialog box that appears, select an image and click **OK**. You can also enter the address of a private image **registry**. The registry address must be in the *domainname/namespace/imagename:tag* format. In this example, *registry.cn-hangzhou.aliyuncs.com/knative-sample/helloworld-go* is used.
Image Version	Click **Select Image Version** and select an image version. By default, the **latest** version is used. In this example, *73fbdd56* is selected.
Access Protocol	**HTTP** and **gRPC** are supported. **Note** gRPC is developed based on the HTTP/2 standard and Protocol Buffers (protobuf) serialization protocol, and supports various programming languages. Compared with HTTP/1.1, HTTP/2 allows you to send and receive packets more efficiently.
Container Port	The container port that you want to expose. The port number must be from 1 to 65535.

For more information about the other parameters, see Create a Knative Service.

7. Click **Create**.

After the Service is created, you can find the Service on the **Services** tab.

8. Run the following command to access the Service:

```
curl -H "<Default domain>" http://<Gateway address>
```

- Default domain: In this example, "host: helloworld-go.default.example.com" is used.
- Gateway address: In this example, "39.106.XX.XX" is used.

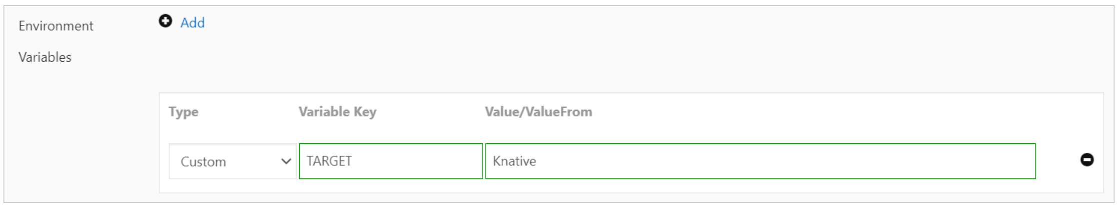
Expected output:

```
Hello World!
```

Step 2: Create a revision to deploy a canary release

1.

2.
3.
4.
5. Create a revision.
   i. On the **Services** tab, select the Service that you created and click **Details** in the **Actions** column.
   ii. Click **Create Revision**.
   iii. On the **Basic Information** wizard page, click **Advanced** and add the following environment variable setting: `TARGET=Knative`.



Environment Variables | Add

Type	Variable Key	Value/ValueFrom
Custom	TARGET	Knative

   iv. Click **Next Step**.
   v. On the **Traffic Splitting Settings** wizard page, set the **Percent %** parameter of the revision to 0. Then, click **Create**.

   Note The sum of traffic ratios of all revisions must be 100.

   vi. After the revision is created, you can find details about the new version on the **Services** tab.
   vii. Run the following command to access the Service:

   Note The **Percent** parameter is set to 0 for the new version. This means that all requests to access the helloworld-go Service are sent to the earlier version.

```
curl -H "host: helloworld-go.default.example.com" http://39.106.XX.XX
```

   Expected output:

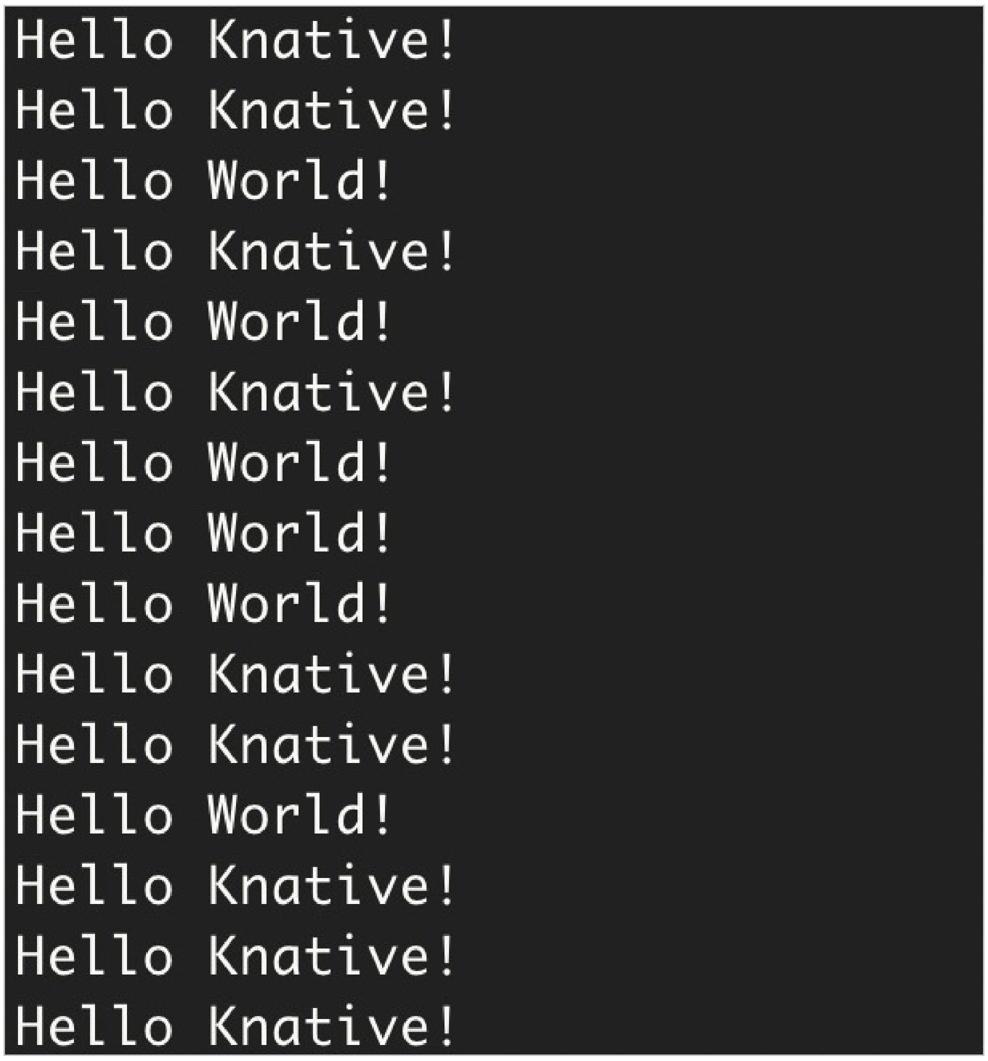
```
Hello World!
```

6. Modify the traffic splitting ratio to deploy a canary release.
   i. On the **Services** tab, select the Service that you created and click **Details** in the **Actions** column.
   ii. Click **Split Traffic**.
   iii. In the **Split Traffic** dialog box, set the **Percent** parameter to 50% for both the earlier and new versions. Then, click **OK**.
   iv. After you change the values of the **Percent** parameters, check the details about the old and new versions on the **Services** tab.

v. Run the following command to access the Service:

```
curl -H "host: helloworld-go.default.example.com" http://39.106.XX.XX
```

Expected output:



```
Hello Knative!
Hello Knative!
Hello World!
Hello Knative!
Hello World!
Hello Knative!
Hello World!
Hello World!
Hello World!
Hello Knative!
Hello Knative!
Hello World!
Hello Knative!
Hello Knative!
Hello Knative!
```

The **Percent** parameter is set to 50% for both the earlier and new versions. This means that requests to access the helloworld-go Service are evenly distributed to the earlier and new version.

You can change the value of the **Percent** parameter to implement canary releases. The new version is completely released if you set the **Percent** parameter to 100% for the new version. During the process, you can change the value of the **Percent** parameter to roll back if issues are found in the new version.

16.6.14. Configure GPU resources for a Knative Service

You can specify GPU-accelerated Elastic Compute Service (ECS) instance types to create GPU-accelerated elastic container instances. You can use GPU-accelerated elastic container instances to deploy Docker images without the need to install software such as TensorFlow and CUDA Toolkit. This topic describes how to create a GPU-accelerated elastic container instance and configure GPU resources for a Knative Service.

Prerequisites

- Create an ASK cluster
- Enable Knative

Configure GPU resources for a Knative Service

You can add the annotation `k8s.aliyun.com/eci-use-specs` to the `spec.template.metadata.annotation` section of the configurations of a Knative Service to specify a GPU-accelerated ECS instance type. You can add the `nvidia.com/gpu` field to the `spec.containers.resources.limits` section to specify the amount of GPU resources that are required by the Knative Service.

The following code block is an example:

```
apiVersion: serving.knative.dev/v1
kind: Service
metadata:
  name: helloworld-go
spec:
  template:
    metadata:
      labels:
        app: helloworld-go
      annotations:
        k8s.aliyun.com/eci-use-specs: ecs.gn5i-c4g1.xlarge  # Specify a GPU-accelerated ECS
instance type that is supported by Knative.
    spec:
      containers:
        - image: registry.cn-hangzhou.aliyuncs.com/knative-sample/helloworld-go:73fbdd56
          ports:
          - containerPort: 8080
          resources:
            limits:
              nvidia.com/gpu: '1'    # Specify the number of GPUs that are required by the
container. This field is required. If you do not specify this field, an error is returned w
hen the pod is launched.
```

The following GPU-accelerated ECS instance families are supported:

- gn7i, a GPU-accelerated compute-optimized instance family that uses NVIDIA A10 GPUs. This instance family includes a variety of instance types, such as ecs.gn7i-c8g1.2xlarge.
- gn7, a GPU-accelerated compute-optimized instance family that uses NVIDIA A100 GPUs. This instance family includes a variety of instance types, such as ecs.gn7-c12g1.3xlarge.
- gn6v, a GPU-accelerated compute-optimized instance family that uses NVIDIA V100 GPUs. This instance family includes a variety of instance types, such as ecs.gn6v-c8g1.2xlarge.
- gn6e, a GPU-accelerated compute-optimized instance family that uses NVIDIA V100 GPUs. This

instance family includes a variety of instance types, such as ecs.gn6e-c12g1.3xlarge.

- gn6i, a GPU-accelerated compute-optimized instance family that uses NVIDIA T4 GPUs. This instance family includes a variety of instance types, such as ecs.gn6i-c4g1.xlarge.
- gn5i, GPU-accelerated compute-optimized instance family that uses NVIDIA P4 GPUs. This instance family includes a variety of instance types, such as ecs.gn5i-c2g1.large.
- gn5, GPU-accelerated compute-optimized instance family that uses NVIDIA P100 GPUs. This instance family includes a variety of instance types, such as ecs.gn5-c4g1.xlarge.

Note

- The gn5 instance family is equipped with local disks. You can mount local disks to elastic container instances. For more information, see Create an elastic container instance that has local disks attached.
- The GPU driver version supported by GPU-accelerated elastic container instances is NVIDIA 460.73.01. The CUDA Toolkit version supported by GPU-accelerated elastic container instances is 11.2.
- For more information about GPU-accelerated ECS instance families, see ECS instance types available for each region and Instance family.

16.6.15. Associate an EIP with the elastic container instance on which a Knative Service runs

You can enable Internet access for an elastic container instance by associating it with an elastic IP address (EIP). Knative allows you to use an annotation to associate an existing EIP or create an EIP and associate the EIP with the elastic container instance on which a Knative Service is deployed. This topic describes how to use an annotation to associate an EIP with the elastic container instance on which a Knative Service is deployed.

Prerequisites

- Create an ASK cluster
- Enable Knative

Context

You can use one of the following methods to enable Internet access for an elastic container instance:

- Associate an EIP with the elastic container instance: EIPs are public IP addresses that can be separately purchased and managed. You can enable Internet access for an elastic container instance by associating an EIP with the instance.
- Associate an EIP with a NAT gateway: NAT gateways are Internet gateways that can be separately purchased. After you associate an EIP with a NAT gateway, the NAT gateway can provide Internet services for all elastic container instances that reside in the virtual private cloud (VPC) to which the NAT gateway belongs.

Knative allows you to enable Internet access for an elastic container instance only by associating an EIP with the elastic container instance. For more information about how to enable Internet access for an elastic container instance, see Enable Internet access.

Use an annotation to associate an EIP with an elastic container instance

You can use an annotation to associate an existing EIP or create an EIP and associate the EIP with the elastic container instance on which a Knative Service is deployed. The following table describes the annotations that you can use to configure the EIP.

Annotation	Description
`k8s.aliyun.com/eci-eip-instanceid`	Specifies an existing EIP that you want to associate.
`k8s.aliyun.com/eci-with-eip`	Specifies whether to automatically create an EIP and associate the EIP with the elastic container instance.
`k8s.aliyun.com/eip-bandwidth`	Specifies the maximum bandwidth for the EIP. Unit: Mbit/s. Default value: 5.
`k8s.aliyun.com/eip-common-bandwidth-package-id`	Specifies the EIP bandwidth plan that you want to use.
`k8s.aliyun.com/eip-isp`	Specifies the line type of the EIP. Valid values: • *BGP*: BGP (Multi-ISP) line • *BGP_PRO*: BGP (Multi-ISP) Pro line

- **Example 1: Associate an existing EIP**

```
apiVersion: serving.knative.dev/v1
kind: Service
metadata:
  name: helloworld-go
spec:
  template:
    metadata:
      labels:
        app: helloworld-go
      annotations:
        k8s.aliyun.com/eci-eip-instanceid: "eip-bp1q5n8cq4p7f6dzu****"    # Specify an ex
isting EIP that you want to associate.
    spec:
      containers:
        - image: registry.cn-hangzhou.aliyuncs.com/knative-sample/helloworld-go:73fbdd56
```

- **Example 2: Create and associate an EIP and specify the bandwidth limit of the EIP**

```
apiVersion: serving.knative.dev/v1
kind: Service
metadata:
  name: helloworld-go
spec:
  template:
    metadata:
      labels:
        app: helloworld-go
      annotations:
        k8s.aliyun.com/eci-with-eip: "true"   # Create and associate an EIP.
        k8s.aliyun.com/eip-bandwidth: "10"   # Specify the bandwidth limit of the EIP.
    spec:
      containers:
        - image: registry.cn-hangzhou.aliyuncs.com/knative-sample/helloworld-go:73fbdd56
```

- **Example 3: Create and associate an EIP and associate an EIP bandwidth plan with the EIP**

```
apiVersion: serving.knative.dev/v1
kind: Service
metadata:
  name: helloworld-go
spec:
  template:
    metadata:
      labels:
        app: helloworld-go
      annotations:
        k8s.aliyun.com/eci-with-eip: "true"   # Create and associate an EIP.
        k8s.aliyun.com/eip-common-bandwith-package-id: "cbwp-2zeukbj916scmj51m****"   # As
sociate an EIP bandwidth plan with the EIP.
    spec:
      containers:
        - image: registry.cn-hangzhou.aliyuncs.com/knative-sample/helloworld-go:73fbdd56
```

16.6.16. Use image caches to accelerate pod creation for Knative Services

Alibaba Cloud provides a type of Custom Resource Definition (CRD) object that you can use to reference image caches when you create pods on elastic container instances. This accelerates pod creation. This topic describes how to use image caches to accelerate pod creation for a Knative Service.

Context

It requires a large amount of time to download an image during the pod creation process. To accelerate pod creation, Elastic Container Instance provides image caches. You can create a cache snapshot from an image and then use the cache snapshot to deploy a pod on an elastic container instance. This reduces the image layers that you need to download and therefore accelerates pod creation. For example, if you want to create a pod by using a Docker image for Apache Flink of about 386.26 MB, 50 seconds is required to pull the image from Docker Hub. If you create the pod by using an image cache, only 5 seconds is required to pull the image.

Note

- The amount of time that is reduced varies based on the number and size of images that are required and the quality of the network connection to the image repository. Alibaba Cloud provides a type of CRD object named ImageCache that you can use to reference image caches when you create pods on elastic container instances.
- An ImageCache is a cluster-level resource in a Kubernetes cluster, and can be shared by all namespaces in the cluster.

Step 1: Create an ImageCache

1. Create an *imagecache-secrets-test.yaml* file based on the following code block:

```
apiVersion: eci.alibabacloud.com/v1
kind: ImageCache
metadata:
  name: imagecache-sample-test
  annotations:
    k8s.aliyun.com/eci-image-cache: "true" # Enable the reuse of ImageCaches.
spec:
  images:
  - centos:latest
  - busybox:latest
  imagePullSecrets:
  - default:secret1
  - default:secret2
  - kube-system:secret3
  imageCacheSize:
   25
  retentionDays:
   7
```

Note If the image that you want to cache contains image layers that can be found in existing ImageCaches, you can enable the reuse of ImageCaches to accelerate image caching.

2. Run the following command to create an ImageCache:

```
kubectl create -f imagecache-secrets-test.yaml
```

3. Run the following command to query an ImageCache:

```
kubectl get imagecache imagecache-sample-test
```

Expected output:

```
NAME                          AGE   CACHEID                              PHASE    P
ROGRESS
imagecache-sample-test        20h   imc-2zeditzeoemfhqor****             Ready
100%
```

Step 2: Use an ImageCache to accelerate pod creation

An ImageCache is a cluster-level resource that you can use to accelerate the creation of pods in different namespaces.

When you create a pod from an ImageCache, you can add annotations to the pod `metadata` to specify the ImageCache that you want to use or set the pod to automatically select an ImageCache. The following list describes the annotations:

- `k8s.aliyun.com/eci-image-snapshot-id`: specifies the ImageCache that you want to use.
- `k8s.aliyun.com/eci-image-cache`: automatically selects an optimal ImageCache.

Note If you specify `k8s.aliyun.com/eci-image-snapshot-id` and `k8s.aliyun.com/eci-image-cache` at the same time, only k8s.aliyun.com/eci-image-snapshot-id takes effect.

- **Method 1: Specify an ImageCache**

  When you create a Knative Service, you can use the `k8s.aliyun.com/eci-image-snapshot-id` annotation to specify the ImageCache that you want to use.

  Notice Make sure that the specified ImageCache is in the Ready state. Otherwise, the pod fails to be created.

  Example:

```
apiVersion: serving.knative.dev/v1
kind: Service
metadata:
  name: helloworld-go
spec:
  template:
    metadata:
      labels:
        app: helloworld-go
      annotations:
        k8s.aliyun.com/eci-image-snapshot-id: imc-2ze5tm5gehgtiiga****  # Specify the Ima
geCache that you want to use.
    spec:
      containers:
        - image: registry.cn-hangzhou.aliyuncs.com/knative-sample/helloworld-go:73fbdd56
```

- **Method 2: Automatically select an ImageCache**

  When you create a Knative Service, you can use the `k8s.aliyun.com/eci-image-cache` annotation to set the pod to automatically select an ImageCache.

  After this annotation is specified, Elastic Container Instance selects an ImageCache based on the following rules:

  i. Elastic Container Instance filters all ImageCaches in the region. The size of each ImageCache is not greater than the size of the temporary storage space of an elastic container instance.

  ii. Elastic Container Instance selects an optimal ImageCache based on the following attributes in descending order of priority: the match degree, the cache size, and the creation time.

     - Match degree: the match degree between an ImageCache and the specified image registry and image version. The ImageCache that has the highest match degree is used to create the pod.

- Cache size: the size of the ImageCache. The ImageCache whose size is closest to the size of the specified image is used to create the pod.
- Creation time: the time when the ImageCache is created. The most recently created ImageCache is used to create the pod.

Note If no ImageCache is matched, the specified image is automatically cached when the pod is created. In this case, the system pulls the image from the specified registry. We recommend that you set the image pull policy to IfNotPresent. This prevents repetitive downloads of the same image layer and improves caching efficiency.

Example:

```
apiVersion: serving.knative.dev/v1
kind: Service
metadata:
  name: helloworld-go
spec:
  template:
    metadata:
      labels:
        app: helloworld-go
      annotations:
        k8s.aliyun.com/eci-image-cache: "true"    # Automatically select an ImageCache.
    spec:
      containers:
        - image: registry.cn-hangzhou.aliyuncs.com/knative-sample/helloworld-go:73fbdd56
```

What to do next

- **Run the following command to query all ImageCaches in a cluster:**

```
kubectl get imagecache
```

- **Run the following command to query information about an ImageCache:**

```
kubectl get imagecache/<imagecache_name> -o yaml
```

Replace `<imagecache_name>` with the name of the ImageCache that you want to query.

- **Run the following command to delete an ImageCache:**

```
kubectl delete imagecache/<imagecache_name>
```

Replace `<imagecache_name>` with the name of the ImageCache that you want to delete.

Related information

- Overview of image caches

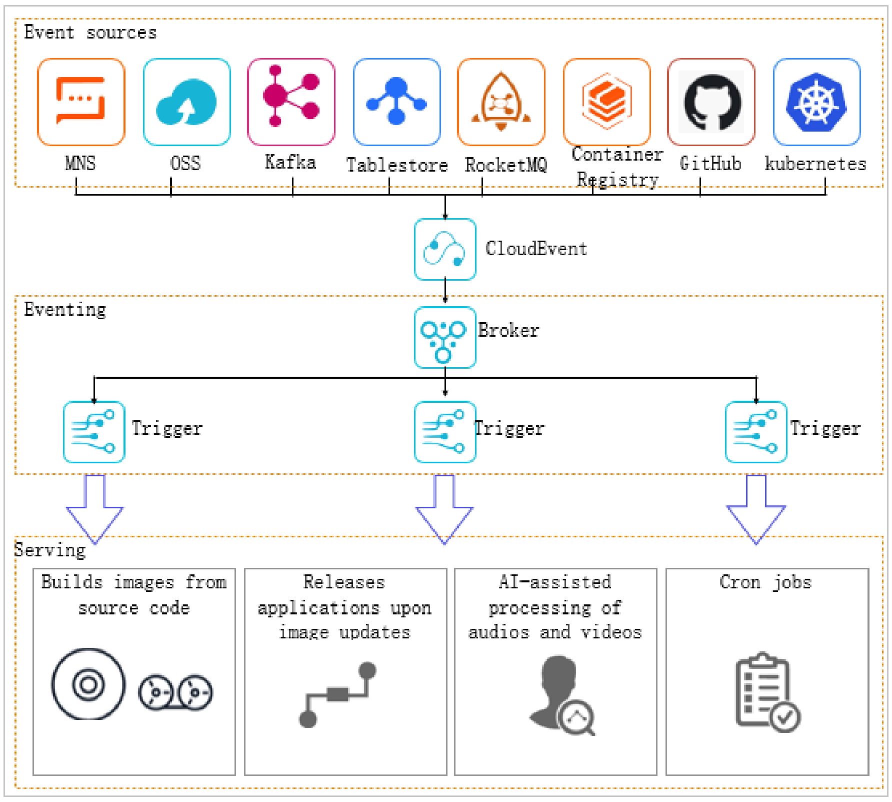
16.7. Knative Eventing

16.7.1. Overview

Knative Eventing is designed to address common demands for cloud-native development. It allows you to bind event sources with event consumers to handle events. This topic describes event sources, event handling, and event consumption of Knative Eventing.

Features

Knative Eventing meets the common needs in cloud-native development. In addition, Knative Eventing provides an architecture for serverless event-driven mode. The architecture contains event sources, event ingesting and subscription, and event filtering. The following figure shows the event-driven architecture.



- Event sources
  - Open source Knative provides various event sources such as Kafka and GitHub.
  - Open source Knative also allows you to use cloud services as event sources. The cloud services include Message Service (MNS) and RocketMQ.
- Event handling
  - Knative Eventing routes events from brokers to event sinks or consumers. You can create one or more triggers to filter or subscribe to specific events.
  - Events can be consumed by serverless applications that are managed by Knative.
- Event consumption
  - Automatically releases applications when images in Container Registry are updated.
  - Automatically creates images upon code submission.
  - Supports cron jobs and AI-assisted processing of audios and videos.

How to start

For more information about how to use Knative Eventing to handle events, see Deploy Knative Eventing.

Related information

- Deploy Knative Eventing

16.7.2. Deploy Knative Eventing

ASK Knative provides the Knative Eventing component to build an event-driven framework for serverless applications. Knative Eventing has designed an all-in-one eventing system for event-driven serverless applications. Knative Eventing provides event management capabilities, such as producing and consuming events. This topic describes how to deploy Knative Eventing in a serverless Kubernetes (ASK) cluster.

Prerequisites

- An ASK cluster is created. For more information, see ASK quick start.
- Knative is enabled. For more information, see Enable Knative.
- A kubectl client is connected to the cluster. For more information, see Connect to ACK clusters by using kubectl.

Procedure

1.
2.
3.
4.
5. In the **Core Component** section of the **Components** tab, find Eventing and click **Deploy** in the **Actions** column.

   After you click **Deploy**, if the **PrivateZone is not enabled for the cluster. For more information, see Enable PrivateZone when you deploy Knative Eventing** error appears in the **Deploy Eventing** dialog box, you must perform the following steps to enable Alibaba Cloud DNS PrivateZone.

   i. Run the following command to modify the *eci-profile* file:

   ```
   kubectl -n kube-system edit configmap eci-profile
   ```

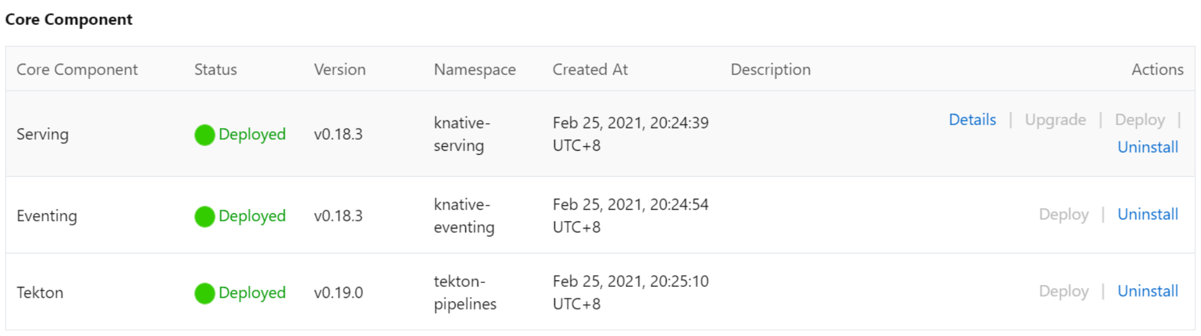
   ii. Set `enablePrivateZone` to `true`. Save and close the *eci-profile* file.

   ```
   apiVersion: v1
   data:
     #...
     enablePrivateZone: "true"
     #...
   kind: ConfigMap
   metadata:
     name: eci-profile
     namespace: kube-system
   ```

6. In the **Deploy Eventing** dialog box, click **Confirm**.

Verify the AHPA policy

After Knative Eventing is deployed, the component is in the **Deployed** state on the **Components** tab.



Core Component

Core Component	Status	Version	Namespace	Created At	Description	Actions
Serving	● Deployed	v0.18.3	knative-serving	Feb 25, 2021, 20:24:39 UTC+8		Details | Upgrade | Deploy | Uninstall
Eventing	● Deployed	v0.18.3	knative-eventing	Feb 25, 2021, 20:24:54 UTC+8		Deploy | Uninstall
Tekton	● Deployed	v0.19.0	tekton-pipelines	Feb 25, 2021, 20:25:10 UTC+8		Deploy | Uninstall

16.8. Best practices

16.8.1. Monitor the QPS, RT, and pod scaling trends of a Knative application

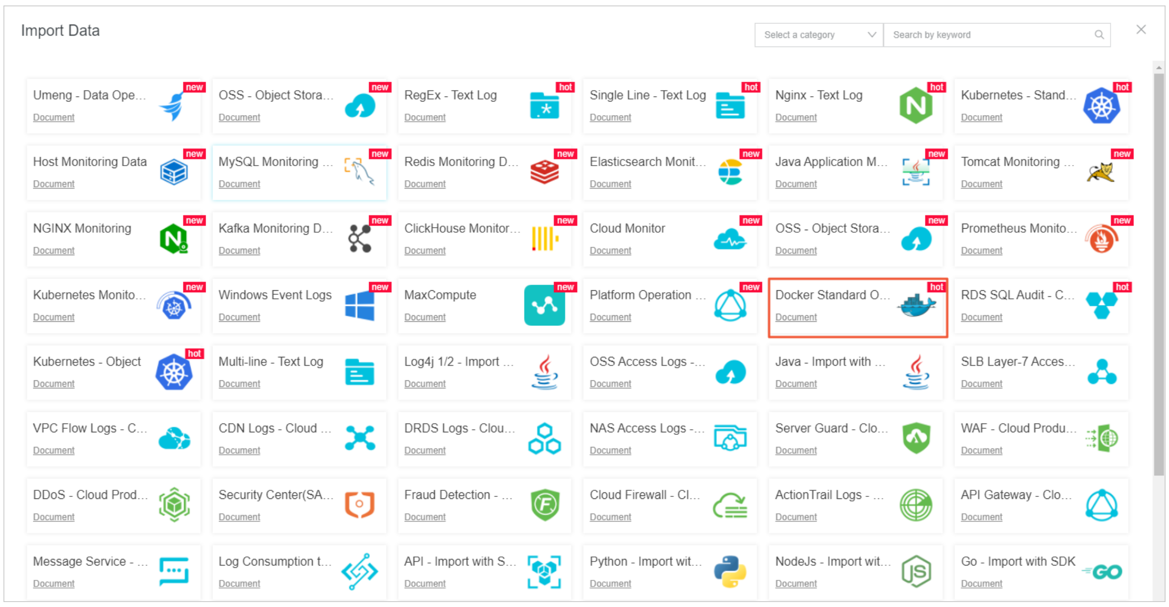
You can monitor the queries per second (QPS), response time (RT), and pod scaling trends of a Knative application. This topic describes how to collect these metrics and view the collected statistics.

Prerequisites

Log Service and Knative are enabled for a serverless Kubernetes (ASK) cluster. For more information, see Enable Knative.

Create and configure a Logstore

1.
2. In the **Projects** section, click the Log Service project that is created for the ASK cluster.
3. On the **Logstores** tab of the **Log Storage** page, click the **+** icon.
4. In the **Create Logstore** panel, set the parameters and click **OK**. For more information about the parameters that are required to **create a Logstore**, see Manage a Logstore.
5. In the **Created.** dialog box, click **Data Import Wizard**.
6. In the **Data Import** section, click **Docker Standard Output**.

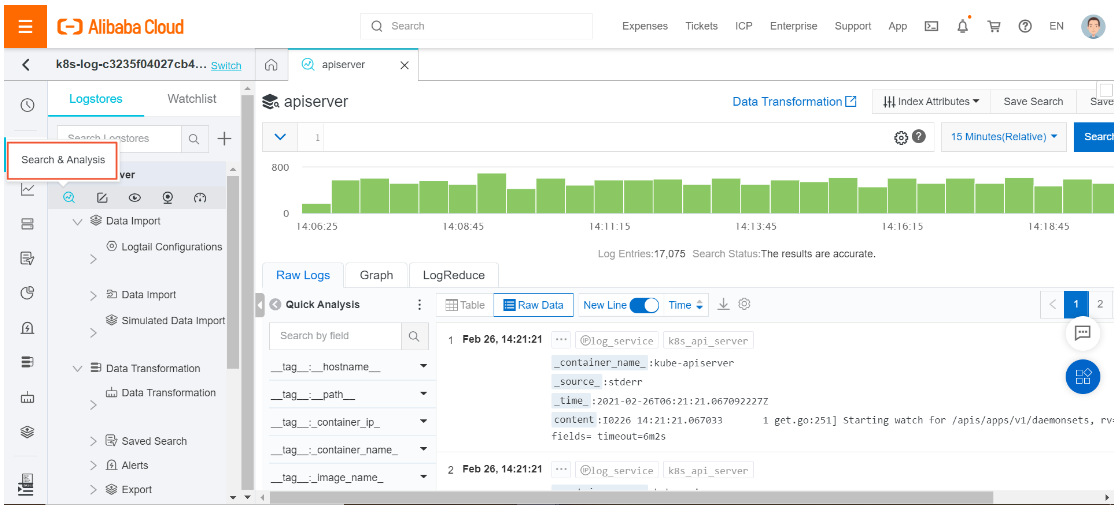

Import Data
Select a category
Search by keyword
Umeng - Data Ope... new Document
OSS - Object Stora... new Document
RegEx - Text Log hot Document
Single Line - Text Log hot Document
Nginx - Text Log hot Document
Kubernetes - Stand... hot Document
Host Monitoring Data new Document
MySQL Monitoring ... new Document
Redis Monitoring D... new Document
Elasticsearch Monit... new Document
Java Application M... new Document
Tomcat Monitoring ... new Document
NGINX Monitoring new Document
Kafka Monitoring D... new Document
ClickHouse Monitor... new Document
Cloud Monitor new Document
OSS - Object Stora... new Document
Prometheus Monito... new Document
Kubernetes Monito... new Document
Windows Event Logs new Document
MaxCompute new Document
Platform Operation ... new Document
Docker Standard O... hot Document
RDS SQL Audit - C... hot Document
Kubernetes - Object hot Document
Multi-line - Text Log Document
Log4j 1/2 - Import ... Document
OSS Access Logs -... Document
Java - Import with ... Document
SLB Layer-7 Acces... Document
VPC Flow Logs - C... Document
CDN Logs - Cloud ... Document
DRDS Logs - Clou... Document
NAS Access Logs -... Document
Server Guard - Clo... Document
WAF - Cloud Produ... Document
DDoS - Cloud Prod... Document
Security Center(SA... Document
Fraud Detection - ... Document
Cloud Firewall - Cl... Document
ActionTrail Logs - ... Document
API Gateway - Clo... Document
Message Service - ... Document
Log Consumption t... Document
API - Import with S... Document
Python - Import wit... Document
NodeJs - Import wit... Document
Go - Import with SDK Document

7. In the **Create Machine Group** step, click **Use Existing Machine Group**.
8. In the **Machine Group Settings** step, select an existing machine group and click **Next**.
9. In the **Specify Data Source** step, set log collection rules by using the following template and click **Next**.

```
{
    "inputs": [
        {
            "detail": {
                "IncludeEnv": {
                    "QUEUE_SERVING_PORT": "8012"
                },
                "ExcludeLabel": {}
            },
            "type": "service_docker_stdout"
        }
    ]
}
```

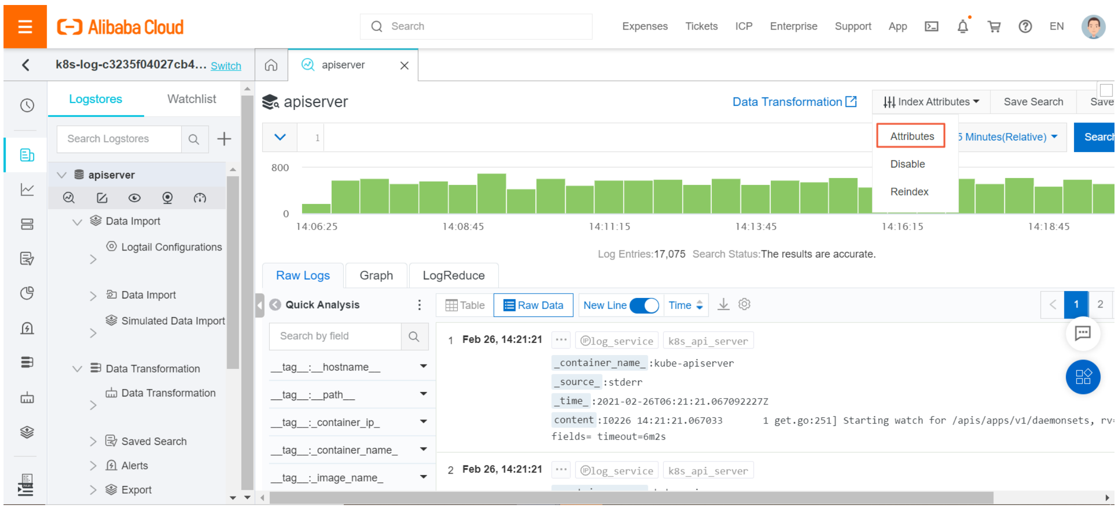
10. In the **Configure Query and Analysis** step, keep the default settings and click **Next**.
11. In the **End** step, click **Search** in the **Log Query** section.

i. Click the **Search & Analysis** icon to view the collected log data.



```
1  Feb 26, 14:21:21  @log_service  k8s_api_server
   _container_name_:kube-apiserver
   _source_:stderr
   _time_:2021-02-26T06:21:21.067092227Z
   content:I0226 14:21:21.067033      1 get.go:251] Starting watch for /apis/apps/v1/daemonsets, rv
   fields= timeout=6m2s
2  Feb 26, 14:21:21  @log_service  k8s_api_server
```

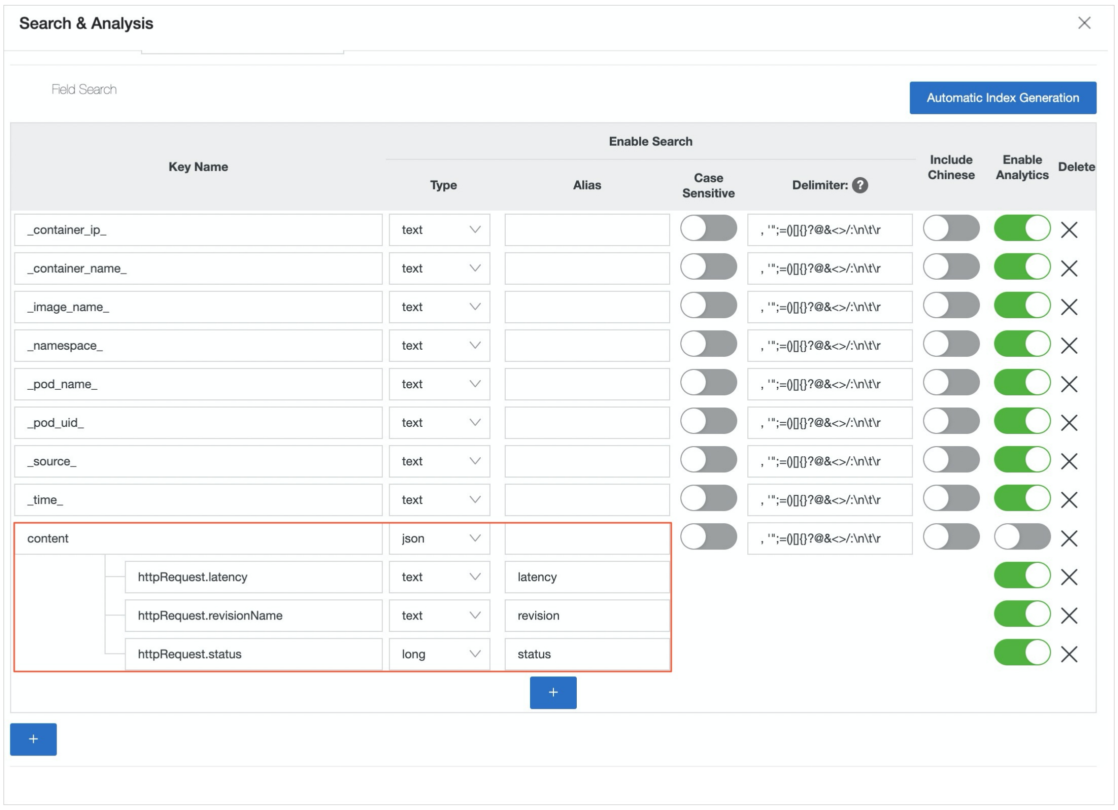
ii. Click **Index Attributes** and select **Attributes** from the drop-down list.



```
1  Feb 26, 14:21:21  @log_service  k8s_api_server
   _container_name_:kube-apiserver
   _source_:stderr
   _time_:2021-02-26T06:21:21.067092227Z
   content:I0226 14:21:21.067033      1 get.go:251] Starting watch for /apis/apps/v1/daemonsets, rv
   fields= timeout=6m2s
2  Feb 26, 14:21:21  @log_service  k8s_api_server
```

iii. In the **Search & Analysis** panel, click **Automatic Index Generation**. In the **Automatically Generate Index Attributes** dialog box, click **Append** and add the following parameters to the **content** field.

Parameter	Type	Alias
httpRequest.latency	text	latency
httpRequest.status	long	status
httpRequest.revisionName	text	revision


Search & Analysis

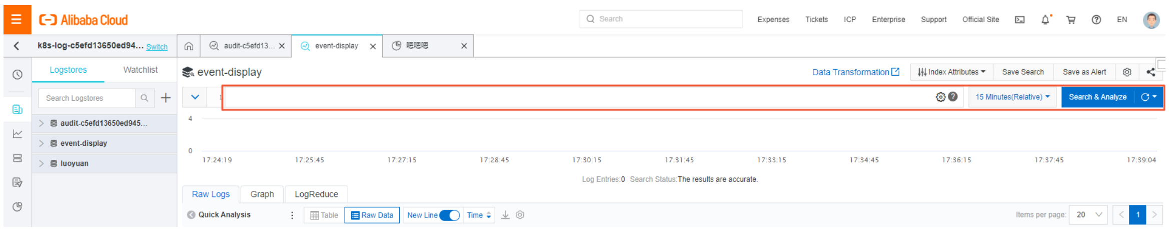
Field Search

Automatic Index Generation

Key Name	Type	Alias	Case Sensitive	Delimiter:	Include Chinese	Enable Analytics	Delete
_container_ip_	text		○	, '";=()[]{}?@&<>/:\n\t\r	○	●	✕
_container_name_	text		○	, '";=()[]{}?@&<>/:\n\t\r	○	●	✕
_image_name_	text		○	, '";=()[]{}?@&<>/:\n\t\r	○	●	✕
namespace	text		○	, '";=()[]{}?@&<>/:\n\t\r	○	●	✕
_pod_name_	text		○	, '";=()[]{}?@&<>/:\n\t\r	○	●	✕
_pod_uid_	text		○	, '";=()[]{}?@&<>/:\n\t\r	○	●	✕
source	text		○	, '";=()[]{}?@&<>/:\n\t\r	○	●	✕
time	text		○	, '";=()[]{}?@&<>/:\n\t\r	○	●	✕
content	json		○	, '";=()[]{}?@&<>/:\n\t\r	○	○	✕
httpRequest.latency	text	latency				●	✕
httpRequest.revisionName	text	revision				●	✕
httpRequest.status	long	status				●	✕

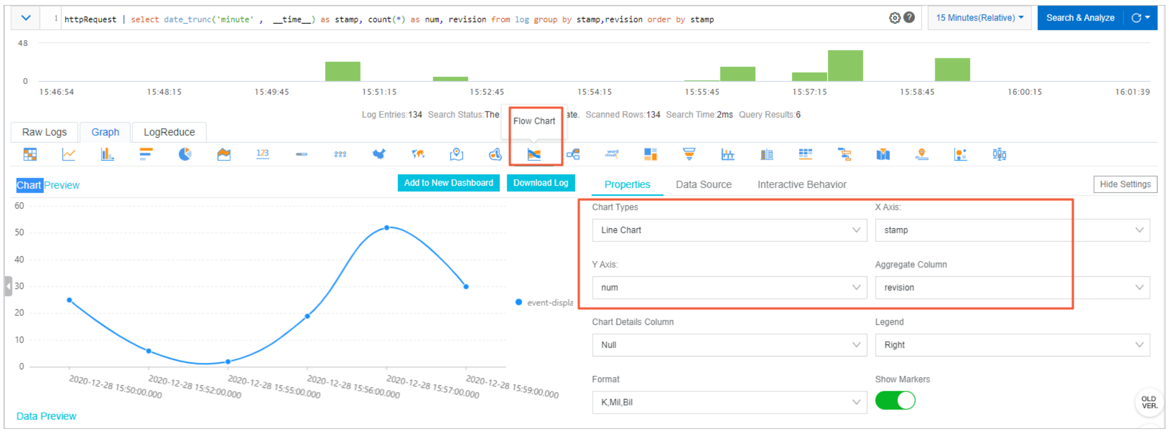
Create a QPS chart

1.
2. In the **Projects** section, click the Log Service project that is created for the ASK cluster.
3. On the **Logstores** tab of the **Log Storage** page, click the Logstore that you want to manage.
4. Enter the following SQL statement into the **Search & Analyze** field.



```
httpRequest | select date_trunc('minute' ,  __time__) as stamp, count(*) as num, revisi
on from log group by stamp,revision order by stamp
```

5. Click the **Flow Chart** icon. Set the parameters on the **Properties** tab and click **Add to New Dashboard**.



```
httpRequest | select date_trunc('minute' , __time__) as stamp, count(*) as num, revision from log group by stamp,revision order by stamp
```

Log Entries:134 Search Status:The ...ate. Scanned Rows:134 Search Time:2ms Query Results:6

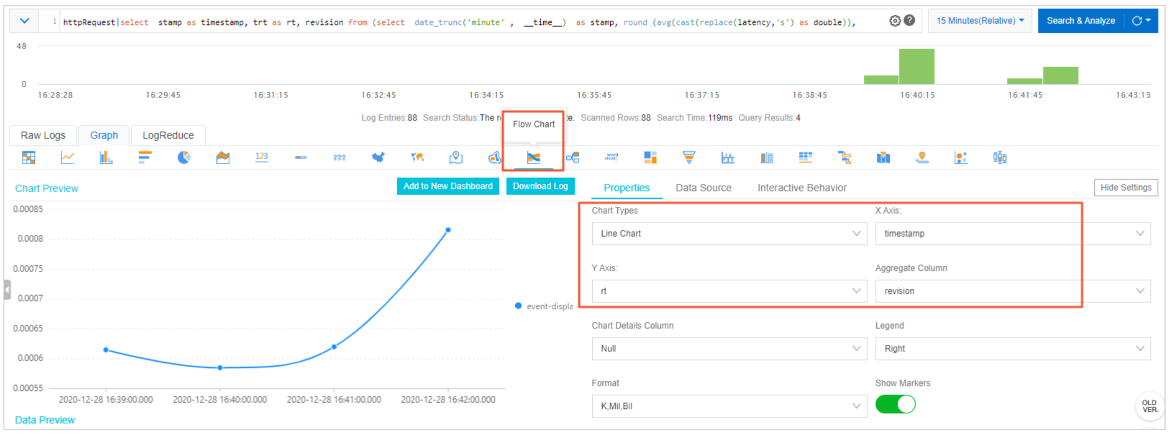
Chart Types: Line Chart
X Axis: stamp
Y Axis: num
Aggregate Column: revision
Chart Details Column: Null
Legend: Right
Format: K,Mil,Bil

Create an RT chart

1.
2. In the **Projects** section, click the Log Service project that is created for the ASK cluster.
3. On the **Logstores** tab of the **Log Storage** page, click the Logstore that you want to manage.
4. Enter the following SQL statement into the **Search & Analyze** field.

```
httpRequest|select  stamp as timestamp, trt as rt, revision from (select  date_trunc('m
inute' ,  __time__)  as stamp, round (avg(cast(replace(latency,'s') as double)), 6) as
trt, revision   from log   group by   stamp, revision
       order by stamp
       limit 100000)
```

5. Click the **Flow Chart** icon. Set the parameters on the **Properties** tab and click **Add to New Dashboard**.



```
httpRequest|select  stamp as timestamp, trt as rt, revision from (select  date_trunc('minute' ,  __time__)  as stamp, round (avg(cast(replace(latency,'s') as double)),
```

Log Entries:88 Search Status:The r...e. Scanned Rows:88 Search Time:119ms Query Results:4

Chart Types: Line Chart
X Axis: timestamp
Y Axis: rt
Aggregate Column: revision
Chart Details Column: Null
Legend: Right
Format: K,Mil,Bil

Create a pod scaling trend chart

1.
2. In the **Projects** section, click the Log Service project that is created for the ASK cluster.
3. On the **Logstores** tab of the **Log Storage** page, click the Logstore whose name starts with audit.
4. Click **Index Attributes** and select **Attributes** from the drop-down list.
5. In the **Search & Analysis** panel, add the following parameters to the **responseObject** field in the

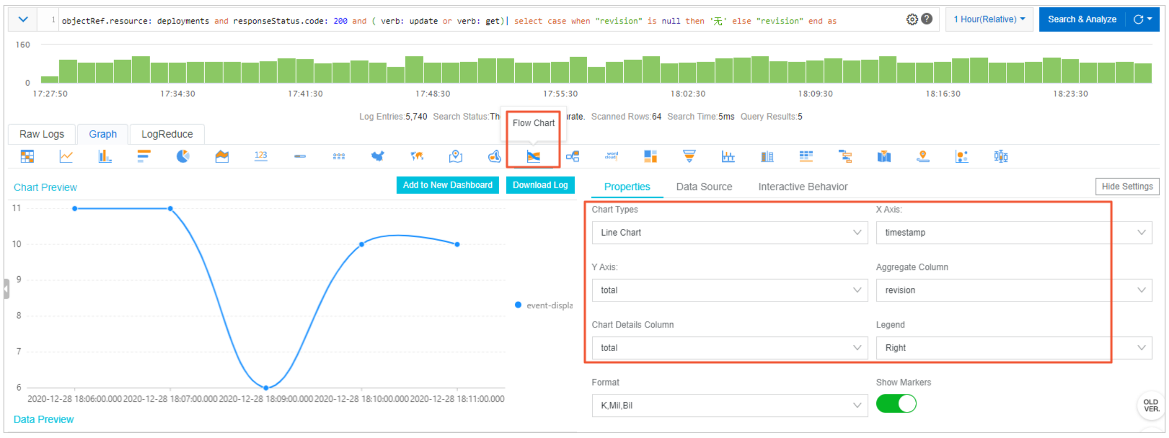
Field Search section.

Parameter	Type	Alias
metadata.labels.serving.knative.dev/revision	text	revision
metadata.labels.serving.knative.dev/service	text	service
status.readyReplicas	long	readyReplicas
status.replicas	long	replicas

6. Enter the following SQL statement into the **Search & Analyze** field.

```
objectRef.resource: deployments and responseStatus.code: 200 and ( verb: update or verb
: get)| select case when "revision" is null then 'none' else "revision" end as "revisio
n", "t" as timestamp, total,concat('total:', cast(total AS varchar), ', ready: ', cast(
ready AS varchar), ', notReady',cast((total-ready) AS varchar)) as detail from (select
date_trunc('minute' , __time__) as t, max(replicas) as total, max(readyReplicas) as re
ady, revision from log where service='event-display-common' group by t, revision order
by t)
```

7. Click the **Flow Chart** icon. Set the parameters on the **Properties** tab and click **Add to New Dashboard**.



```
objectRef.resource: deployments and responseStatus.code: 200 and ( verb: update or verb: get)| select case when "revision" is null then '无' else "revision" end as
```

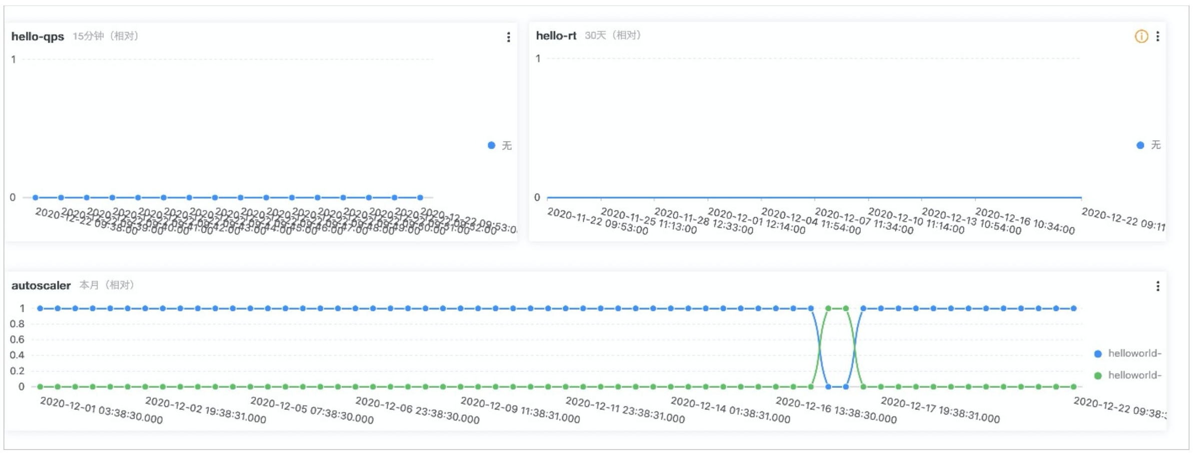
Log Entries:5,740 Search Status:Th… …rate. Scanned Rows:64 Search Time:5ms Query Results:5

Chart Types: Line Chart
X Axis: timestamp
Y Axis: total
Aggregate Column: revision
Chart Details Column: total
Legend: Right
Format: K,Mil,Bil
Show Markers: on

Monitor the application status

To monitor the application status, you can add the preceding charts to one dashboard by using the following methods:

- Select the same dashboard for each chart.
- Create a global dashboard and add the preceding charts to the dashboard. For more information, see Create a dashboard and Add charts to a dashboard.



16.8.2. Use Knative for live commenting

Knative is an open source and Kubernetes-based orchestration platform for serverless applications. The main objective of Knative is to develop a cloud-native and cross-platform orchestration standard for serverless applications. Knative provided by Alibaba Cloud is developed based on serverless Kubernetes (ASK) and is compatible with open source Knative. Knative can orchestrate applications in a unified manner for Function Compute-based and Elastic Container Instance-based workloads. Knative also supports event-driven scaling and auto scaling. This topic describes how to use Knative to deploy a demo service that consists of an event sink, an event source, and HomePage. The demo service is for live commenting.

Prerequisites

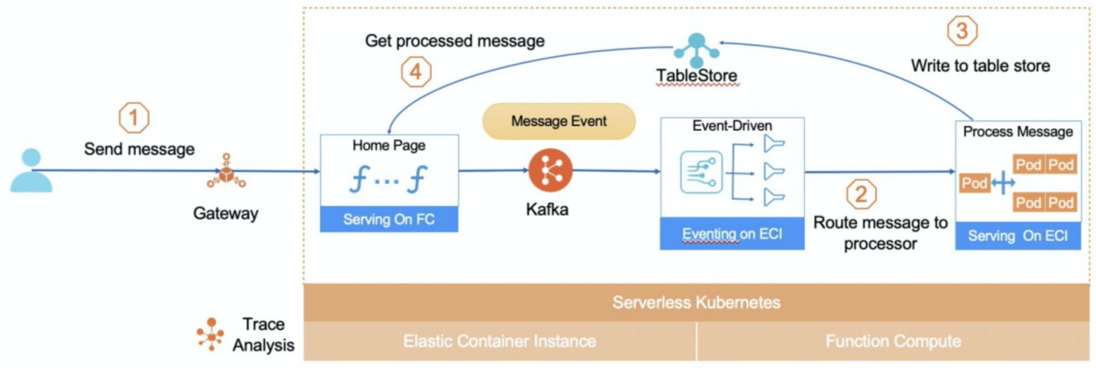
- Create an ASK cluster
- Enable Knative
- Deploy Knative Eventing

Introduction to the demo service

The demo service consists of the following components:

- HomePage: HomePage is used to receive and send live comments. HomePage is deployed to Function Compute by using Knative Serving.
- Event source: The event source is used to receive, filter, and transfer events. The event source is deployed on an elastic container instance by using Knative Serving.
- Event sink: The event sink is used to process live comments. The event sink is deployed on an elastic container instance by using Knative Serving.

The following figure shows how to use the demo service for live commenting.



1. Users send live comments to HomePage.
2. HomePage sends the live comments to Kafka. The event source receives the live comments and then transfers the live comments to the event sink for processing.
3. After the live comments are processed, the processed live comments are sent to Tablestore.
4. The processed live comments are displayed on the homepage.

Step 1: Deploy an event sink

You can deploy an event sink to receive live comments that are sent from event sources. You can configure auto scaling for the event source based on the number of live comments. After the live comments are processed, the event sink sends the live comments to Tablestore. Before you deploy the event sink, make sure that no workload exists. This helps you view the deployment result.

1. Log on to the and check whether workloads exist.
   i.
   ii.
   iii.
   iv.
   v. In the top navigation bar, select **default** from the **Namespace** drop-down list and make sure that no workload exists in the default namespace.
2. In the , deploy the event sink on an Elastic Container Instance-based workload by using Knative.
   i. In the left-side navigation pane, choose **Applications > Knative**.
   ii. On the page that appears, click the **Services** tab and then click **Create from Template**.
   iii. Select **default** from the **Namespace** drop-down list. Select **Custom** from the **Sample Template** drop-down list. Copy the following YAML template to the Template section and click **Create**.

   The YAML template contains the following content:

```
apiVersion: serving.knative.dev/v1
kind: Service
metadata:
  name: test-barrage-process
spec:
  template:
    metadata:
      annotations:
        autoscaling.knative.dev/maxScale: "100"   #Specify the maximum number of po
```

```
ds for the Service.
        autoscaling.knative.dev/minScale: "0"     #Specify the minimum number of po
ds for the Service.
        k8s.aliyun.com/eci-image-snapshot-id: imc-uf636kjjx8xr4e75****
      labels:
        danmu.role: "manager"
    spec:
      containerConcurrency: 2           #Specify the maximum number of concurrent re
quests that the pod can handle.
      serviceAccountName: barrage-install-sa
      containers:
        - args:
            - /manager
          env:
            - name: OTS_ENDPOINT      #Specify the endpoint of the Tablestore insta
nce.
              value: https://barrage.cn-hangzhou.tablestore.aliyuncs.com
            - name: TABLE_NAME
              value: barrage
            - name: OTS_INSTANCENAME
              value: barrage
            - name: OTS_KEYID
              value: xxx
            - name: OTS_SECRET
              value: xxx
            - name: POD_NAME
              valueFrom:
                fieldRef:
                  fieldPath: metadata.name
            - name: ROLE
              value: manager
            - name: POD_NAMESPACE
              valueFrom:
                fieldRef:
                  fieldPath: metadata.namespace
            - name: TRACE_NAME
              value: "process"
            - name: PARENT_SPAN
              value: "barrage-sender"
            - name: SUB_SPAN
              value: "process"
            - name: TRACING       #Specify the endpoint of the trace.
              value: "http://tracing-analysis-dc-sh.aliyuncs.com/adapt_g2it2kg78n@5
cf06035aec2eb9_g2it2kg78n@53df7ad2afe****/api/traces"
          image: registry-vpc.cn-shanghai.aliyuncs.com/knative-sample/barrage-manag
er:forrester-****_4cd77c84-20210618215458
          name: user-container
          ports:
            - containerPort: 8000
              name: http1
```

Step 2: Deploy an event source

You can deploy an event source to receive, filter, and transfer events. In this example, a KafkaSource event source is deployed. The KafkaSource event source receives live comments from Kafka and sends the live comments to the event sink.

1.
2. On the page that appears, click the **Services** tab and then click **Create from Template**.
3. Select **default** from the **Namespace** drop-down list. Select **Custom** from the **Sample Template** drop-down list. Copy the following YAML template to the Template section and click **Create**.

   The YAML template contains the following content:

   ```
   apiVersion: sources.knative.dev/v1alpha1
   kind: KafkaSource
   metadata:
     annotations:
       k8s.aliyun.com/req-timeout: "60"
       k8s.aliyun.com/retry-count: "5"
       k8s.aliyun.com/retry-interval: "2"
     name: barrage
     namespace: default
   spec:
     bootstrapServers: 192.168.42.205:9092,192.168.42.204:9092,192.168.42.203:9092   #Spec
   ify the default address of the Message Queue for Apache Kafka instance.
     consumerGroup: barrage-info-consumer    #Specify a consumer group for the Message Que
   ue for Apache Kafka instance.
     sink:
       ref:
         apiVersion: serving.knative.dev/v1
         kind: Service
         name: test-barrage-process  #Specify an event sink to which the live comments are
   sent.
         namespace: default
     topics: barrage-info     #Specify a topic for the Message Queue for Apache Kafka inst
   ance.
   ```

   After you deploy the event sink and the event source, choose **Workloads > Pods** in the left-side navigation pane of the cluster details page. On the **Pods** page, you can find that the pods of the event sink and the event source are in the **Running** state.

Step 3: Deploy HomePage

HomePage receives live comments from the front end and then sends the live comments to Kafka. HomePage receives the processed live comments from Tablestore. After you deploy HomePage to a Function Compute-based workload by using Knative, a service, function, and custom domain name are automatically created in the Function Compute console. Before you deploy HomePage, make sure that no service for live commenting, function, or custom domain name exists in the Function Compute console.

1. Log on to the console and check whether services for live commenting, functions, or custom domain names exist.

   i.

   ii. In the left-side navigation pane, click **Services & Functions**.

iii. In the top navigation bar, select a region from the drop-down list.

  In this example, China (Shanghai) is selected.

iv. On the **Services** page, find and click the service that you want to manage. On the Functions page, make sure that no service for live commenting or function exists.

v. In the left-side navigation pane of the console, choose **Advanced Features > Domain Names**. On the **Domain Names** page, make sure that no custom domain name exists.

2. In the Container Service for Kubernetes (ACK) console, deploy HomePage to a Function Compute-based workload by using Knative.

  i.

  ii.

  iii.

  iv. In the left-side navigation pane, choose **Applications > Knative**.

  v. On the page that appears, click the **Services** tab and then click **Create from Template**.

  vi. Select **default** from the **Namespace** drop-down list. Select **Custom** from the **Sample Template** drop-down list. Copy the following YAML template to the Template section and click **Create**.

  The YAML template contains the following content:

```
apiVersion: serving.knative.dev/v1
kind: Service
metadata:
  name: demo-barrage
  annotations:
    workload.serving.knative.aliyun.com/class: "fc"   #Deploy a Function Compute-ba
sed workload.
spec:
  template:
    metadata:
      annotations:
        fc.revision.serving.knative.aliyun.com/code-space: "image"
#Specify the type of the container image.
        fc.revision.serving.knative.aliyun.com/role-arm: "acs:ram::xxxx:role/knativ
e-fc"    #Enter the policy that is associated with the Alibaba Cloud Resource Name
(ARN).
        fc.revision.serving.knative.aliyun.com/domain: '{"domain":"barrage.demo.kna
tive.top","path":"/*"}'  #Specify the domain name that is accessed. The barrage.dem
o.knative.top domain name is used.
    spec:
      containers:
        - image: registry.cn-shanghai.aliyuncs.com/knative-sample/barrage-main:forr
ester-****_4cd77c84-20210618214527
          env:
            - name: OTS_ENDPOINT     #Specify the endpoint of the Tablestore instan
ce.
              value: https://barrage.cn-hangzhou.ots.aliyuncs.com
            - name: TABLE_NAME
              value: barrage
            - name: OTS_INSTANCENAME
              value: barrage
            - name: OTS_KEYID
              value: xxx
            - name: OTS_SECRET
              value: xxx
            - name: KAFKA_SERVER     #Specify the address of the Message Queue for
Apache Kafka instance.
              value: "106.15.XX.XX:9093,47.100.XX.XX:9093,47.102.XX.XX:9093"
            - name: KAFKA_USER
              value: "alikafka_pre-cn-xxx"
            - name: KAFKA_PWD
              value: "xxx"
            - name: KAFKA_TOPIC     ##Specify a topic for the Message Queue for Apa
che Kafka instance.
              value: "barrage-info"
            - name: TRACING         #Specify the endpoint of the trace.
              value: "http://tracing-analysis-dc-sh.aliyuncs.com/adapt_g2it2kg78n@5
cf06035aec****_g2it2kg78n@53df7ad2afe8301/api/traces"
            - name: TRACE_NAME1
              value: "sender"
            - name: TRACE_NAME2
              value: "receiver"
            - name: TRACE_NAME3
              value: "result"
            - name: PARENT_SPAN
              value: "barrage-sender"
            - name: SUB_SPAN1
              value: "sender"
            - name: SUB_SPAN2
              value: "result"
```

Verify the result

Log on to the console and check whether the service is deployed.

i.

ii. In the left-side navigation pane, choose **Advanced Features > Domain Names**. In the top navigation bar, select a region. In this example, China (Shanghai) is selected. On the **Domain Names** page, you can find the domain name that you configured.

iii. In the left-side navigation pane, click **Services & Functions**. On the **Services** page, click the service that you want to manage. On the Functions page, you can find that HomePage is deployed.

Step 4: Access the service

1. On the **Domain Names** page of the console, access the service by using the custom domain name.

   In this example, the custom domain name is *http://barrage.demo.knative.top*.

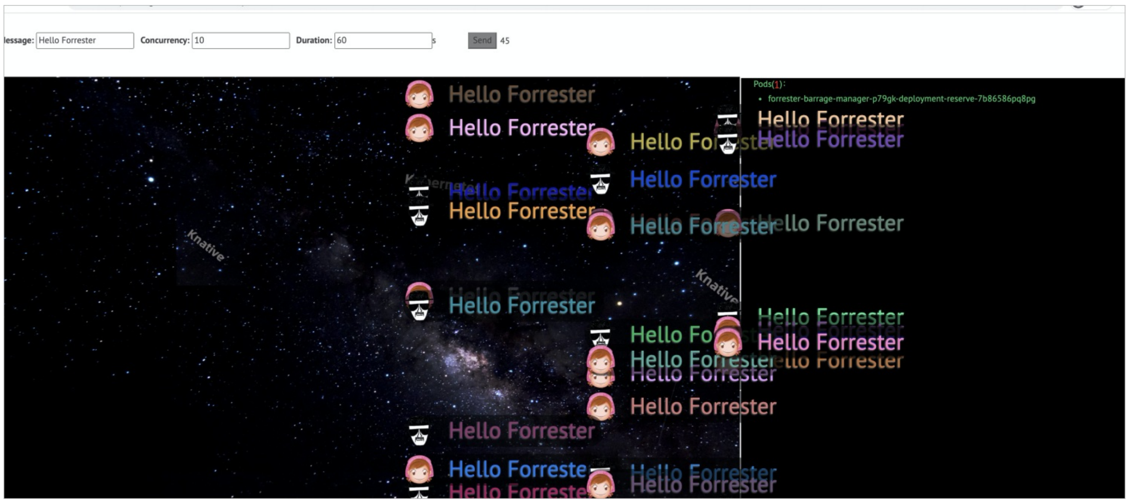
2. Set the following parameters and click **Send**.

   - Message: The live comments that you want to send.
   - Concurrency: The number of concurrent live comments.
   - Duration: The duration of the live comments.

   In this example, the default configurations are used.

Execution results

The following figure shows that the live comments continuously pop up.



16.8.3. Monitor the CPU and memory usage of a Knative application

You can monitor the CPU and memory usage of a Knative application. This topic describes how to collect these metrics.

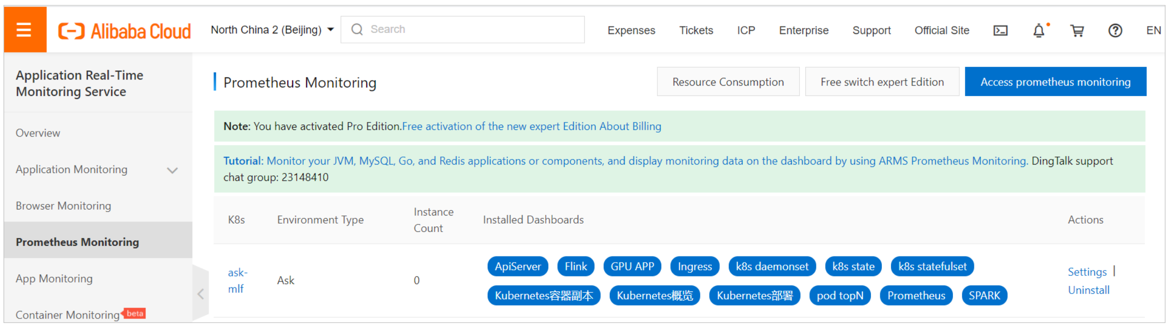
Prerequisites

- Knative is enabled for a serverless Kubernetes (ASK) cluster. For more information, see Enable Knative.
- Prometheus Monitoring is enabled for the ASK cluster. For more information, see Enable ARMS Prometheus.

Procedure

1.
2. In the left-side navigation pane, click **Prometheus Monitoring** and click the ASK cluster that you want to manage.

   Note If you have not installed Prometheus, click **Install** on the **Prometheus Monitoring** page.



Prometheus Monitoring

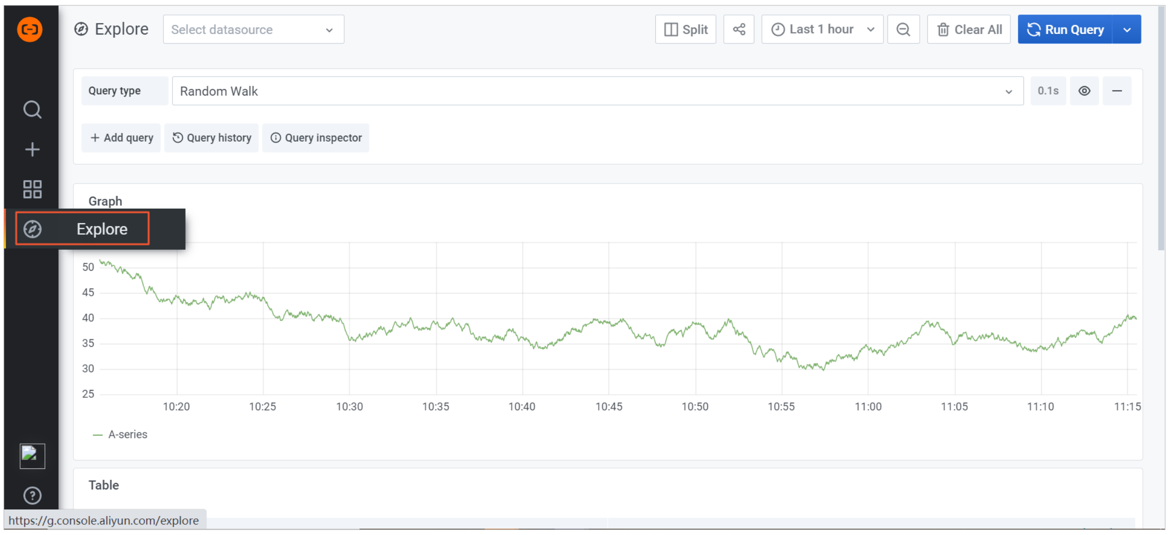
Note: You have activated Pro Edition.Free activation of the new expert Edition About Billing

Tutorial: Monitor your JVM, MySQL, Go, and Redis applications or components, and display monitoring data on the dashboard by using ARMS Prometheus Monitoring. DingTalk support chat group: 23148410

K8s	Environment Type	Instance Count	Installed Dashboards	Actions
ask-mlf	Ask	0	ApiServer, Flink, GPU APP, Ingress, k8s daemonset, k8s state, k8s statefulset, Kubernetes容器副本, Kubernetes概览, Kubernetes部署, pod topN, Prometheus, SPARK	Settings | Uninstall

3. On the **Dashboards** page, click **Prometheus** in the **Name** column.

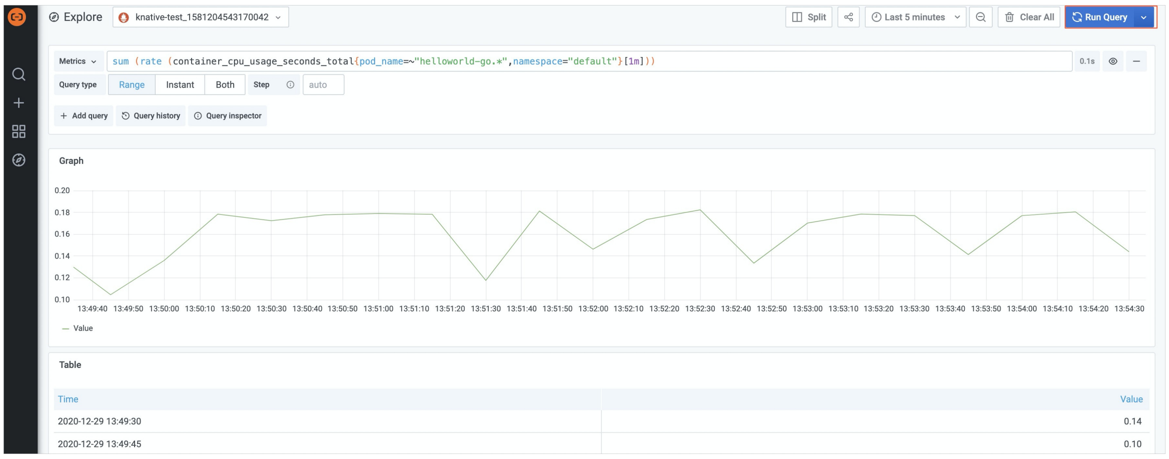
4. On the left side of the page that appears, click the **Explore** icon and select the ASK cluster from the **Select datasource** drop-down list.



5. Query the CPU usage of a Knative application.

Enter the following PromQL statement into the **Metrics** field. In this example, the **helloworld-go** application is queried. Click **Run Query** to query the CPU usage of the application.

```
sum (rate (container_cpu_usage_seconds_total{pod_name=~"helloworld-go.*",namespace="def
ault"}[1m]))
```



```
sum (rate (container_cpu_usage_seconds_total{pod_name=~"helloworld-go.*",namespace="default"}[1m]))
```

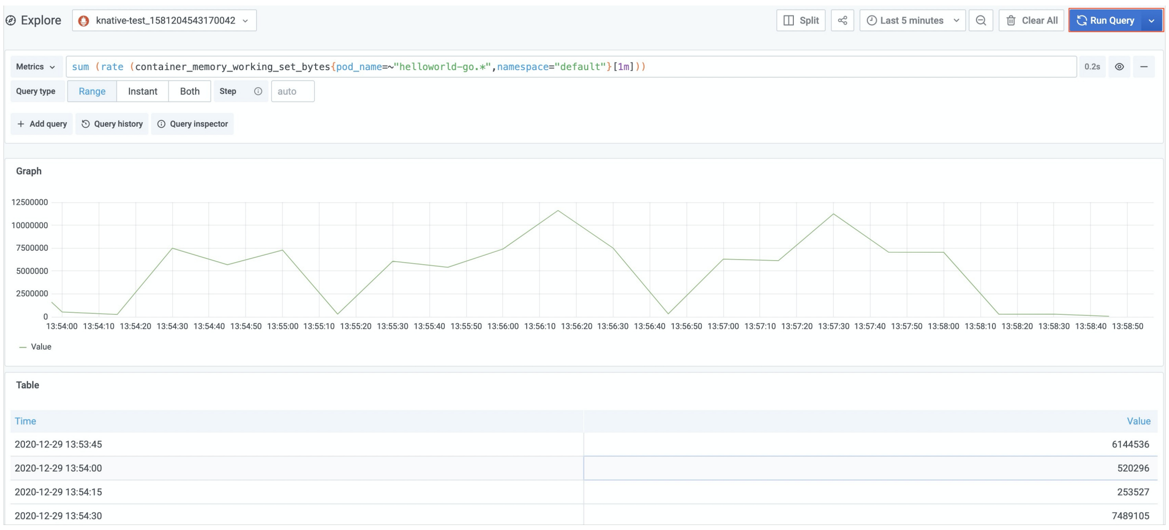
Table

Time	Value
2020-12-29 13:49:30	0.14
2020-12-29 13:49:45	0.10

6. Query the memory usage of a Knative application.

Enter the following PromQL statement into the **Metrics** field. In this example, the **helloworld-go** application is queried. Click **Run Query** to query the memory usage of the application.

```
sum (rate (container_memory_working_set_bytes{pod_name=~"helloworld-go.*",namespace="de
fault"}[1m]))
```



```
sum (rate (container_memory_working_set_bytes{pod_name=~"helloworld-go.*",namespace="default"}[1m]))
```

Time	Value
2020-12-29 13:53:45	6144536
2020-12-29 13:54:00	520296
2020-12-29 13:54:15	253527
2020-12-29 13:54:30	7489105

17.Auto scaling

17.1. Auto scaling overview

Auto scaling is a service that can dynamically scale computing resources to meet your business requirements. Auto scaling provides a more cost-effective method to manage your resources. This topic introduces auto scaling and the related components.

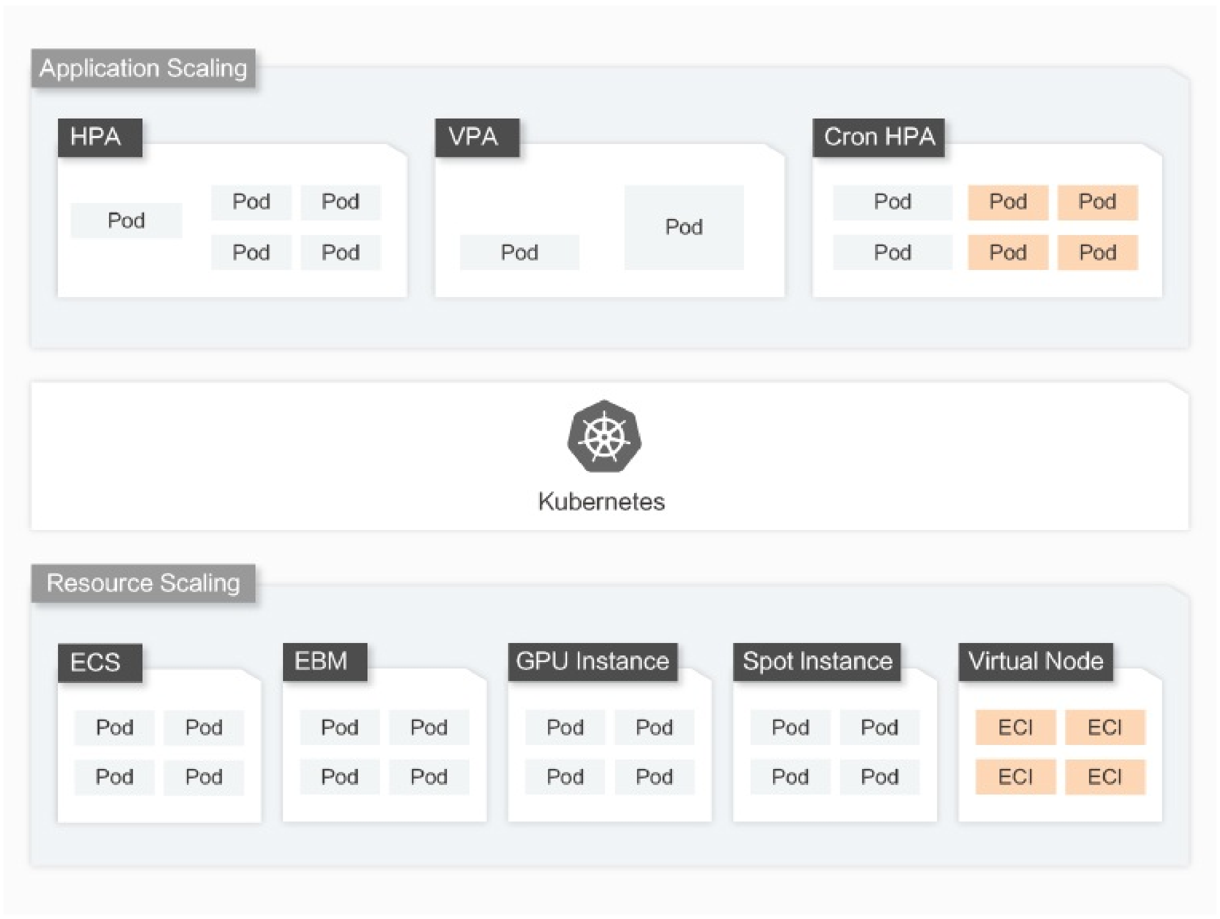
Background information

Auto scaling is widely used in Container Service for Kubernetes (ACK) clusters. Typically, auto scaling is used in scenarios such as online workload scaling, large-scale computing and training, GPU-accelerated deep learning, inference and training based on shared GPU resources, and periodical workload scheduling. Auto scaling enables elasticity from the following aspects:

- Workload scaling. Auto scaling can scale workloads, such as pods. For example, Horizontal Pod Autoscaler (HPA) is a typical workload scaling component that can change the number of replicated pods to scale the workload.
- Resource scaling. If the resources of a cluster cannot meet the scaling requirements of workloads, Elastic Compute Service (ECS) instances or elastic container instances are added to the cluster.

The components for workload scaling and resource scaling can be used separately or in combination. If you want to decouple the components, you must scale the workload within the resource limit of the cluster.

Introduction to auto scaling



Component s for workload scaling

Component	Description	Scenario	Limits	References
HPA	A built-in component of Kubernetes. HPA is used for online applications.	Online applications	HPA uses Deployments and StatefulSets to scale workloads.	Horizont al Pod Aut oscaling
Vertical Pod Autoscaler (VPA)	An open source component. VPA is used for monolithic applications.	Monolit hic applications	VPA is used for applications that cannot be horizont ally scaled. Typically, VPA is used when pods are recovered from anomalies.	Vertical pod aut oscaling

Component	Description	Scenario	Limits	References
CronHPA	An open source component provided by ACK. CronHPA is used for applications whose resource usage periodically changes.	Periodically fluctuating workloads	CronHPA uses Deployments and StatefulSets to scale workloads. CronHPA is compatible with HPA. You can use CronHPA and HPA in combination to scale workloads.	CronHPA
Elastic-Workload	A component provided by ACK. Elastic-Workload is used in scenarios where fine-grained scaling is required. For example, you can use Elastic-Workload if you want to distribute a workload across different zones.	Scenarios where fine-grained scaling is required	Elastic-Workload is applicable to online workloads that require fine-grained scaling. For example, some pods of a Deployment are scheduled to an ECS instance, and the rest of the pods are scheduled to elastic container instances.	Install ack-kubernetes-elastic-workload

Components for resource scaling

Component	Description	Scenario	Time cost for delivery	References
cluster-autoscaler	cluster-autoscaler is an open source component provided by Kubernetes that can scale nodes in a cluster. cluster-autoscaler is integrated with auto scaling to provide more elastic and cost-effective scaling services.	cluster-autoscaler is applicable to all scenarios, especially online workloads, deep learning, and large-scale computing.	The amount of time that is required to add 1,000 nodes to a cluster: • Standard mode: 120 seconds. • Fast mode: 60 seconds. • Standard mode with the qboot firmware: 90 seconds. • Fast mode with the qboot firmware: 45 seconds.	Auto scaling of nodes

17.2. Horizontal Pod Autoscaling

You can create an application that has Horizontal Pod Autoscaling (HPA) enabled in the Container Service console. HPA can automatically scale container resources for your application. You can also use a YAML file to describe HPA settings.

Prerequisites

- ASK quick start
- Connect to an ACK cluster by using kubectl

Create an application that has HPA enabled in the ACK console

1.
2.
3.
4.
5. On the **Deployments** tab, click **Create from Image**.
6. On the **Basic Information** wizard page, enter an application name, select a cluster and a namespace to deploy the application, and then click **Next**.

Parameter	Description
Namespace	Select the namespace to which the application belongs. The default namespace is automatically selected.
Name	Enter a name for the application.
Replicas	The number of pods that you want to provision for the application. Default value: 2.
Type	The type of the application. You can select **Deployment**, **StatefulSet**, **Job**, **CronJob**, or **DaemonSet**.
Label	Add labels to the application. The labels are used to identify the application.
Annotations	Add annotations to the application.

7. On the **Container** wizard page, set the container parameters, select an image, and then configure the required computing resources. Click **Next**. For more information, see Configure containers for an application.

   Note You must configure the required computing resources for the Deployment. Otherwise, you cannot enable HPA.

8. On the **Advanced** wizard page, find the **Access Control** section, click **Create** on the right side of Services, and then set the parameters. For more information, see Configure advanced settings for an

application.

9. On the **Advanced** wizard page, select **Enable** for **HPA** and configure the scaling threshold and related settings.
    - **Metric**: Select CPU Usage or Memory Usage. The selected resource type must be the same as the one that you have specified in the Required Resources field.
    - **Condition**: Specify the resource usage threshold. HPA triggers scaling activities when the threshold is exceeded.
    - **Max. Replicas**: Specify the maximum number of pods to which the Deployment can be scaled.
    - **Min. Replicas**: Specify the minimum number of pods that must run for the Deployment.
10. In the lower-right corner of the Advanced wizard page, click **Create**. The application is created with HPA enabled.

Verify the result

1. Click **View Details** or choose **Workloads > Deployments**. On the page that appears, click the **name of the created application** or click **Details** in the **Actions** column. Then, click the **Pod Scaling** tab to view information about the scaling group of the application.
2. After the application starts to run, container resources are automatically scaled based on the CPU utilization. You can check whether HPA is enabled in the staging environment by performing a CPU stress test on the pods of the application. Verify that the pods are automatically scaled within 30 seconds.

Use kubectl to enable HPA

You can also create a Horizontal Pod Autoscaler by using an orchestration template and associate the Horizontal Pod Autoscaler with the Deployment for which you want to enable HPA. Then, you can use **kubectl** to enable HPA.

In the following example, HPA is enabled for an NGINX application.

1. Create a file named *nginx.yml* and copy the following content into the file.

    The following code block is a YAML template that is used to create a Deployment:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9 # replace it with your exactly <image_name:tags>
        ports:
        - containerPort: 80
        resources:
          requests:                         ##To enable HPA, you must set this paramete
r.
            cpu: 500m
```

2. Run the following command to create an NGINX application:

```
kubectl create -f nginx.yml
```

3. Create a Horizontal Pod Autoscaler.

   Use scaleTargetRef to associate the Horizontal Pod Autoscaler with the Deployment named **nginx**.

```
apiVersion: autoscaling/v2beta2
kind: HorizontalPodAutoscaler
metadata:
  name: nginx-hpa
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: nginx
  minReplicas: 1
  maxReplicas: 10
  metrics:
  - type: Resource
    resource:
      name: cpu
      target:
        type: Utilization
        averageUtilization: 50
```

Note You must configure the requested resources for the pods of the application. Otherwise, the Horizontal Pod Autoscaler cannot be started.

4. Run the `kubectl describe hpa name` command. The following output is an example of a warning that is returned:

```
Warning  FailedGetResourceMetric       2m (x6 over 4m)  horizontal-pod-autoscaler  miss
ing request for cpu on container nginx in pod default/nginx-deployment-basic-75675f5897
-mqzs7
Warning  FailedComputeMetricsReplicas  2m (x6 over 4m)  horizontal-pod-autoscaler  fail
ed to get cpu utilization: missing request for cpu on container nginx in pod default/ng
inx-deployment-basic-75675f5
```

5. After the Horizontal Pod Autoscaler is created, run the `kubectl describe hpa name` command.

   If the following output is returned, it indicates that the Horizontal Pod Autoscaler runs as expected:

```
Normal SuccessfulRescale 39s horizontal-pod-autoscaler New size: 1; reason: All metrics
below target
```

   If the pod usage of the NGINX application exceeds 50% as specified in the HPA settings, the Horizontal Pod Autoscaler automatically adds pods. If the pod usage of the NGINX application drops below 50%, the Horizontal Pod Autoscaler automatically removes pods.

17.3. CronHPA

Prerequisites

- A serverless Kubernetes (ASK) cluster is created. For more information, see Create an ASK cluster.
- A kubectl client is connected to your cluster. For more information, see Connect to an ACK cluster by using kubectl.
- Helm 2.11.0 or a later version is installed on your on-premises machine. For more information, see Install Helm.

Context

Install the CronHPA controller

You can install the CronHPA controller ack-kubernetes-cronhpa-controller by using one of the following methods.

Method 1: Install the CronHPA controller on the Add-ons page in the ACK console

1.
2.
3. On the **Clusters** page, find the cluster that you want to manage and click its name or click **Details** in the **Actions** column.
4. In the left-side navigation pane of the details page, choose **Operations > Add-ons**.
5. On the **Add-ons** page, click the **Manage Applications** tab, find **ack-kubernetes-cronhpa-**

controller, and then click **Install**.

Method 2: Install the CronHPA controller from App Catalog

1.
2.
3. On the **Marketplace** page, click the **App Catalog** tab. Find and click **ack-kubernetes-cronhpa-controller**.
4. On the **ack-kubernetes-cronhpa-controller** page, click **Deploy**.
5. In the **Deploy** wizard, select a cluster and a namespace, and then click **Next**.
6. On the **Parameters** wizard page, configure the required parameters and click **OK**.

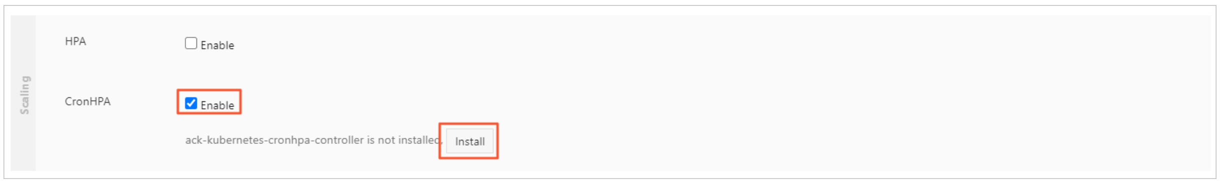
You can uninstall the CronHPA controller if CronHPA is no longer used. For more information about how to uninstall ack-kubernetes-cronhpa-controller, see Manage system components or Delete a release.

Create CronHPA jobs

Before you create and run CronHPA jobs for your application, make sure that the CronHPA controller runs as normal in your cluster and only one HPA task is created for your application. The following section describes how to enable CronHPA and HPA to interact without conflicts. You can create CronHPA jobs in the following scenarios.

Scenario 1: Create CronHPA jobs when you create an application

In the **Scaling** section on the **Advanced** wizard page, select **Enable** on the right side of **CronHPA** to create CronHPA jobs for the application. For more information about how to create an application, see Create a stateless application by using a Deployment or Use a StatefulSet to create a stateful application.



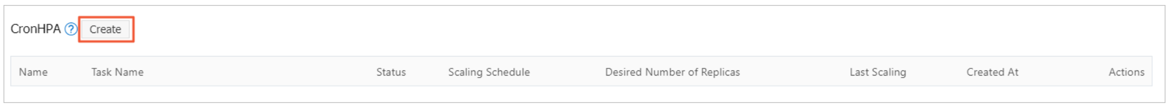
The ACK console automatically checks whether the CronHPA controller is installed in the cluster. If the CronHPA controller is not installed, the **Install** button appears on the page. After the CronHPA controller is installed, the CronHPA parameters appear on the page. The following table describes the parameters.

Parameter	Description
Job Name	Enter a name for the CronHPA job. The name of each CronHPA job must be unique.
Desired Number of Replicas	Replicated pods are scaled to the desired number at the scheduled time.
Scaling Schedule	Set the scaling schedule. For more information about how to set the scaling schedule for a CronHPA job, see AliyunContainerService/kubernetes-cronhpa-controller.

Scenario 2: Create CronHPA jobs for an existing application

The following example demonstrates how to create CronHPA jobs for an existing application. A stateless application is used in this example.

1.
2.
3.
4.
5. On the **Deployments** page, find the application that you want to manage and click **Details** in the **Actions** column.
6. Click the **Pod Scaling** tab and configure CronHPA jobs.
   - If the CronHPA controller is not installed, the **Install** button appears on the page. Click **Install** and perform the following steps.
   - If the CronHPA controller is installed, perform the following steps.
7. Click **Create** to the right side of **CronHPA**. In the **Create** dialog box, configure CronHPA jobs.

   CronHPA Create

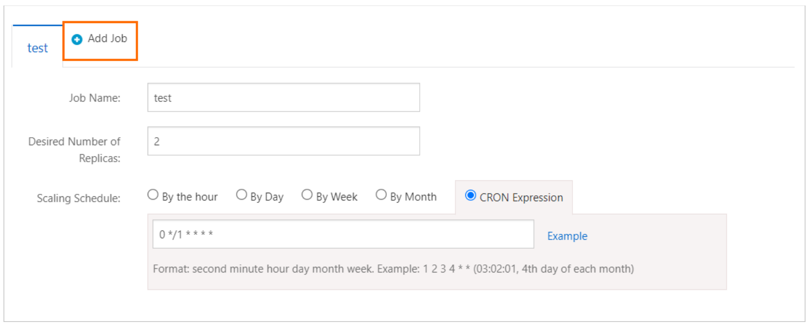
Name	Task Name	Status	Scaling Schedule	Desired Number of Replicas	Last Scaling	Created At	Actions

   The following table describes the parameters.

Parameter	Description
Job Name	Enter a name for the CronHPA job. The name of each CronHPA job must be unique.
Desired Number of Replicas	Replicated pods are scaled to the desired number at the scheduled time.
Scaling Schedule	Set the scaling schedule. For more information about how to set the scaling schedule for a CronHPA job, see AliyunContainerService/kubernetes-cronhpa-controller.

Create or modify CronHPA jobs

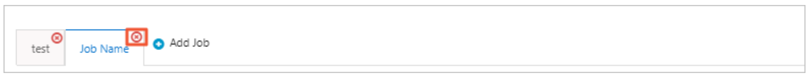
1. Go to the **Pod Scaling** tab by performing the steps described in the preceding Create CronHPA jobs section.
2. On the **Pod Scaling** tab, find the created CronHPA job in the **CronHPA** section and click **Add or Modify Job** in the **Actions** column.
3. In the **Edit** dialog box, click **Add Job** to create CronHPA jobs. You can also modify existing CronHPA jobs. Click **OK**.



Job Name: test
Desired Number of Replicas: 2
Scaling Schedule: By the hour By Day By Week By Month [✓ CRON Expression]
0 */1 * * * *
Format: second minute hour day month week. Example: 1 2 3 4 * * (03:02:01, 4th day of each month)

To delete CronHPA jobs, perform the following steps:

In the **Edit** dialog box, click the delete icon in the upper-right corner of the job that you want to delete. Click **OK**.



Enable CronHPA and HPA to interact without conflicts

CronHPA triggers horizontal scaling for containers based on schedules. HPA is used to scale pods to ensure the availability of your applications when network traffic spikes. If ACK detects that both CronHPA and HPA are deployed, ACK sets HPA as the scaling object of CronHPA. CronHPA triggers HPA to scale pods at the scheduled time.

The following YAML template shows the configurations of CronHPA and HPA:

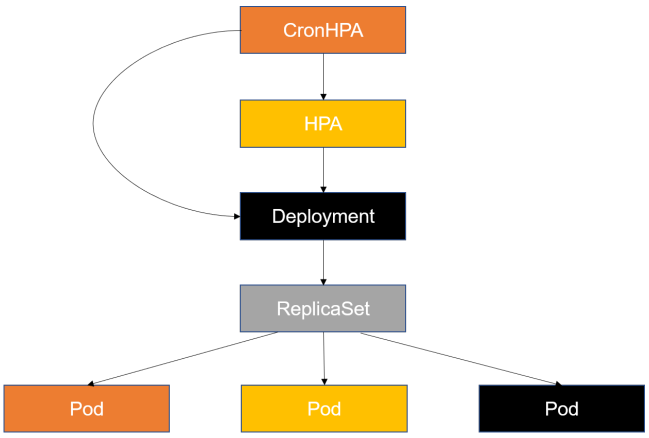
After you compare the configurations of CronHPA and HPA, you can find that:

- The scaleTargetRef field is used in the configurations of both CronHPA and HPA to specify the object to be scaled.
- The **crontab** rules in the jobs section of the CronHPA configuration specify the number to which pods are scaled at the scheduled time.
- HPA triggers scaling activities based on resource usage.

If both CronHPA and HPA are deployed, CronHPA and HPA may scale pods for the same application that is specified by scaleTargetRef. CronHPA and HPA are independent and unaware of each other. As a result, the CronHPA controller and the HPA controller scale pods for the application separately. The later scaling activity overwrites the earlier one.

Solution

The reason for the conflict between CronHPA and HPA is that the CronHPA controller and the HPA controller are unaware of each other. To resolve the conflict, CronHPA needs only to detect the status of HPA. In the HPA configuration, scaleTargetRef specifies the Deployment that is managed by HPA. The Deployment is related to a ReplicaSet. When HPA scales the Deployment, the ReplicaSet scales pods to the desired number. ACK modifies the CronHPA configuration by setting scaleTargetRef to HPA. CronHPA can find the application that is specified by scaleTargetRef in the HPA configuration. This enables CronHPA to detect the status of HPA.



The following YAML template shows the configurations that enable CronHPA and HPA to interact without conflicts:

```
apiVersion: autoscaling.alibabacloud.com/v1beta1
kind: CronHorizontalPodAutoscaler
metadata:
  labels:
    controller-tools.k8s.io: "1.0"
  name: cronhpa-sample
spec:
   scaleTargetRef:
      apiVersion: autoscaling/v1
      kind: HorizontalPodAutoscaler
      name:  nginx-deployment-basic-hpa
   jobs:
   - name: "scale-down"
     schedule: "30 */1 * * * *"
     targetSize: 1
     runOnce: true
   - name: "scale-up"
     schedule: "0 */1 * * * *"
     targetSize: 3
     runOnce: true
```

After you deploy the preceding YAML template, CronHPA is aware of the values of minReplicas, maxReplicas, and desiredReplicas in the HPA configuration. CronHPA is also aware of the current number of pods provisioned for the application that is specified by scaleTargetRef. CronHPA can detect the status of HPA by modifying the HPA configurations. CronHPA compares the desired number of pods with the current number of pods and then determines whether to trigger scaling activities and change the maximum number of pods in the HPA configuration. CronHPA also compares the desired number of pods with the maximum and minimum numbers of pods that are specified in the HPA configuration and then determines whether to change the minimum numbers of pods in the HPA configuration.

The following table describes the rules that enable CronHPA and HPA to interact without conflicts.

HPA (min/max)	Cronhpa	Deployment	Scaling result	Description

HPA (min/max)	Cronhpa	Deployment	Scaling result	Description
1/10	5	5	• HPA (min/max): 1/10 • Deployment: 5	If the number of pods desired by CronHPA equals the current number of pods, CronHPA does not change the maximum and minimum numbers of pods in the HPA configuration. In addition, no scaling activity is triggered.
1/10	4	5	• HPA (min/max): 1/10 • Deployment: 5	If the number of pods desired by CronHPA is smaller than the current number of pods, no scaling activity is triggered.
1/10	6	5	• HPA (min/max): 6/10 • Deployment: 6	• If the number of pods desired by CronHPA is greater than the current number of pods, CronHPA adds pods to reach the desired number. • If the number of pods desired by CronHPA is greater than the value of minReplicas in the HPA configuration, CronHPA changes the value of minReplicas.
5/10	4	5	• HPA (min/max): 4/10 • Deployment: 5	• If the number of pods desired by CronHPA is smaller than the current number of pods, no scaling activity is triggered. • If the number of pods desired by CronHPA is smaller than the value of minReplicas in the HPA configuration, CronHPA changes the value of minReplicas.

HPA (min/max)	Cronhpa	Deployment	Scaling result	Description
5/10	11	5	• HPA (min/max): 11/11 • Deployment: 11	• If the number of pods desired by CronHPA is greater than the current number of pods, CronHPA adds pods to reach the desired number. • If the number of pods desired by CronHPA is greater than the value of maxReplicas in the HPA configuration, CronHPA changes the value of maxReplicas.

The following list describes the parameters in the table:

- HPA (min/max): the minimum and maximum numbers of pods that are specified in the HPA configuration.
- CronHPA: the desired number of pods.
- Deployment: the current number of pods that are provisioned for the application.

CronHPA does not directly change the number of pods for the Deployment. It triggers HPA to scale the pods. This resolves the conflict between CronHPA and HPA.

[✓ CronHPA] HPA

```
apiVersion: autoscaling.alibabacloud.com/v1beta1
kind: CronHorizontalPodAutoscaler
metadata:
  labels:
    controller-tools.k8s.io: "1.0"
  name: cronhpa-sample
spec:
   scaleTargetRef:
      apiVersion: apps/v1
      kind: Deployment
      name: nginx-deployment-basic
   jobs:
   - name: "scale-down"
     schedule: "30 */1 * * * *"
     targetSize: 1
   - name: "scale-up"
     schedule: "0 */1 * * * *"
     targetSize: 11
```

```
apiVersion: autoscaling/v2beta2
kind: HorizontalPodAutoscaler
metadata:
  name: php-apache
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: nginx-deployment-basic
  minReplicas: 4
  maxReplicas: 10
  metrics:
  - type: Resource
    resource:
      name: cpu
      target:
        type: Utilization
        averageUtilization: 50
```

Related information

- Horizontal Pod Autoscaling

17.4. Implement horizontal auto scaling based on Alibaba Cloud metrics

Prerequisites

ASK quick start

Context

Examples

The following example shows how to configure HPA by creating a Deployment and a Service that are both named Nginx.

1.
2.
3. On the **Clusters** page, find the cluster that you want to manage, and click the name of the cluster or click **Applications** in the **Actions** column.
4. On the **Deployments** page, click **Create from YAML** in the upper-right corner.
5. Use the following template to create a Deployment and a ClusterIP type Service. Then, click **Create**.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment-basic
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
        ports:
        - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  name: nginx
  namespace: default
spec:
  ports:
    - port: 80
      protocol: TCP
      targetPort: 80
  selector:
    app: nginx
  type: ClusterIP
```

6. In the left-side navigation pane of the cluster details page, click **Ingresses**. On the **Ingresses** page, click **Create** in the upper-right corner.
7. In the **Create** dialog box, set the parameters and click **Create**. After you create an Ingress, the **Ingresses** page appears.
8. On the Ingresses page, find the newly created Ingress and click **Details** in the Actions column to view information about the Ingress.
9. Configure HPA.
   i. In the left-side navigation pane of the ACK console, choose **Marketplace** > **Orchestration Templates**.
   ii. On the **Templates** page, search for **hpa** and click **Create Application** to deploy an application with the following YAML template:

```
apiVersion: autoscaling/v2beta2
kind: HorizontalPodAutoscaler
metadata:
  name: ingress-hpa
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: nginx-deployment-basic
  minReplicas: 2
  maxReplicas: 10
  metrics:
    - type: External
      external:
        metric:
          name: sls_ingress_qps
          selector:
            matchLabels:
              sls.project: "***"
              sls.logstore: "nginx-ingress"
              sls.ingress.route: "default-nginx-80"
        target:
          type: AverageValue
          averageValue: 10
    - type: External
      external:
        metric:
          name: sls_ingress_latency_p9999
          selector:
            matchLabels:
              # default ingress log project is k8s-log-clusterId
              sls.project: "***"
              # default ingress logstre is nginx-ingress
              sls.logstore: "nginx-ingress"
              # namespace-svc-port
              sls.ingress.route: "default-nginx-80"
              # sls vpc endpoint, default true
              # sls.internal.endpoint: true
        target:
          type: Value
          # sls_ingress_latency_p9999 > 10ms
          value: 10
```

The following table describes the parameters that are used to configure HPA.

Parameter	Description
sls.ingress.route	-
sls.logstore	-
sls.project	-

Parameter	Description
sls.internal.endpoint	Specifies whether to access Log Service over the internal network. Default value: true. If you set the value to true, you access Log Service over the internal network. If you set the value to false, you access Log Service over the Internet.

Note

In this example, HPA triggers scaling activities based on the sls_ingress_qps and sls_ingress_latency_p9999 metrics. In the target sections, each metric has a different type value:

- The type value of the sls_ingress_qps metric is set to AverageValue. This indicates that the metric value is the result of dividing the total QPS by the number of pods.
- The type value of the sls_ingress_latency_p9999 metric is set to Value. This indicates that the latency is not divided by the number of pods.

The two type values are commonly used in HPA configurations.

iii. On the **hpa** page, click **Create** on the right side of the edtor.

10. After HPA is configured, run the following script to perform a stress test:

```
#!/bin/bash
##Use Apache Benchmark to send requests to the Service exposed by the Ingress. The test lasts 300 seconds and 10 concurrent requests are sent per second.
ab -t 300 -c 10 <The domain name of the Ingress>
```

11. Check whether HPA works as expected.

i. In the left-side navigation pane of the ACK console, click **Clusters**. On the **Clusters** page, find the cluster that you want to manage and choose **More > Open Cloud Shell** in the **Actions** column.

ii. Run the `kubectl get hpa ingress-hpa` command to check whether a scale-out activity is triggered.

If the value of REPLICAS is the same as the value of MAXPODS, it indicates that HPA scaled out the number of pods as expected.

```
$ kubectl get hpa ingress-hpa
NAME          REFERENCE                           TARGETS      MINPODS    MAXPODS    REPLICAS    AGE
ingress-hpa   Deployment/nginx-deployment-basic   27/10 (avg)  2          10         10          7m49s
```

17.5. AHPA for predictive scaling

Serverless Kubernetes (ASK) clusters provide the Advanced Horizontal Pod Autoscaler (AHPA) component that supports predictive scaling. This topic describes how to use AHPA to configure predictive scaling.

Prerequisites

- AHPA is in invitational preview. To use AHPA, to apply to be added to a whitelist.
- An ASK cluster is created. For more information, see Create an ASK cluster.
- Application Real-Time Monitoring Service (ARMS) Prometheus is enabled, and application statistics within at least seven days are collected by ARMS Prometheus. The statistics include details about the CPU and memory resources that are used by an application. For more information about how to enable ARMS Prometheus, see Enable ARMS Prometheus.

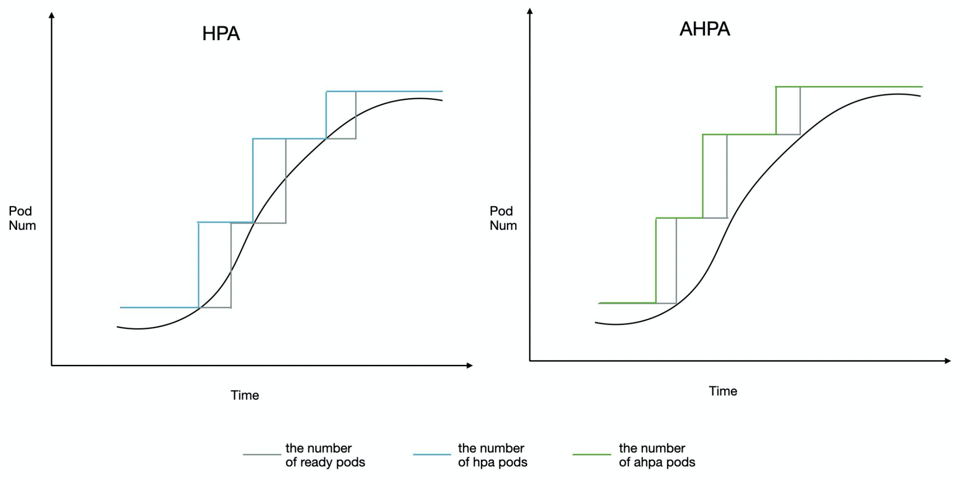
Context

The following traditional methods are used to manage the pods of an application: manually specify the number of pods, use Horizontal Pod Autoscaler (HPA), and use CronHPA. The following table describes the disadvantages of the preceding methods.

Method	Disadvantage
Manually specify the number of pods	Resources are wasted during off-peak hours. Idle resources are still billed.
HPA	• Scaling activities are performed after a scaling delay. Scale-out activities are triggered only if the resource usage exceeds the threshold and scale-in activities are triggered only if the resource usage drops below the threshold. • New pods require a period of time to enter the Ready state. This increases the response time and even causes request timeouts.
CronHPA	• The precision of scheduled pod scaling is low. If a schedule is not properly specified, resources may be wasted. • You must modify the scaling policy to adapt to fluctuating workloads.

ASK clusters provide the AHPA component that supports predictive scaling. You can use AHPA to increase resource utilization and improve the efficiency of resource management. AHPA can analyze historical data and predict the number of pods that are required per minute within the next 24 hours. If you use CronHPA, you must manually create 1,440 (24 hours × 60 minutes) schedules instead. The following figure shows the difference between traditional horizontal pod scaling and predictive horizontal pod scaling.

- Traditional horizontal pod scaling: Scale-out activities are triggered after the amount of workloads increases. The system cannot provision pods in a timely manner to handle fluctuating workloads due to the scaling delay.
- Predictive horizontal pod scaling: AHPA learns the pattern of workload fluctuations based on the historical values of specific metrics and the amount of time that a pod spent before the pod entered the Ready state. This way, AHPA can provision pods that are ready to be scheduled before a traffic spike occurs. This ensures that resources are allocated in a timely manner.



Step 1: Install Application Intelligence Controller

1.
2.
3.
4.
5. On the **Add-ons** page, click the **Others** tab, find Application Intelligence Controller, and then click **Install**.
6. In the **Install Application Intelligence Controller** message, click **OK**.

Step 2: Add ARMS Prometheus as a data source

1.
2. In the left-side navigation pane, choose **Prometheus Monitoring** > **Prometheus Instances**.
3. In the upper-left corner of the **Prometheus Monitoring** page, select the region in which your Prometheus instances are deployed. Then, click the name of a Prometheus instance whose **Instance Type** is **Prometheus for Container Service**. The details page of the Prometheus instance appears.
4. In the left-side navigation pane of the instance details page, click **Settings** and copy the internal endpoint in the HTTP API Address section.
5. In the HTTP API Address section, click **Generate Token** to generate a token. The token is used to pass the authentication when you access the Prometheus instance.
6. Create a ConfigMap named *application-intelligence.yaml* based on the following content and specify the internal endpoint of the Prometheus instance:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: application-intelligence
  namespace: kube-system
data:
  armsUrl: "https://cn-hangzhou.arms.aliyuncs.com:9443/api/v1/prometheus/da9d7dece901db
4c9fc7f5b9c40****/158120454317****/cc6df477a982145d986e3f79c985a****/cn-hangzhou"
  token: "****"
```

Descriptions of parameters:

- `armsUrl` : Specify the internal endpoint of the Prometheus instance that you copied in Step 4.
- `token` : Specify the token that was generated in Step 5.

7. Run the following command to deploy the ConfigMap:

```
kubectl apply -f application-intelligence.yaml
```

Step 3: Configure AHPA

1. Create an AHPA policy with the following content:

```
apiVersion: autoscaling.alibabacloud.com/v1beta1
kind: AdvancedHorizontalPodAutoscaler
metadata:
  name: ahpa-demo
spec:
  scaleStrategy: observer
  metrics:
  - type: Resource
    resource:
      name: cpu
      target:
        type: Utilization
        averageUtilization: 40
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: php-apache
  maxReplicas: 100
  minReplicas: 2
  prediction:
    quantile: 95
    scaleUpForward: 180
  instanceBounds:
  - startTime: "2021-12-16 00:00:00"
    endTime: "2022-12-16 24:00:00"
    bounds:
    - cron: "* 0-8 ? * MON-FRI"
      maxReplicas: 4
      minReplicas: 15
    - cron: "* 8-15 ? * MON-FRI"
      maxReplicas: 10
      minReplicas: 15
    - cron: "* 15-24 ? * MON-FRI"
      maxReplicas: 15
      minReplicas: 20
```

The following table describes some parameters that are specified in the preceding code block.

Parameter	Description

Parameter	Description
`scaleTargetRef`	Required. The Deployment for which you want to configure predictive scaling.
`metrics`	Required. The metrics based on which predictive scaling is performed. You can specify CPU and memory metrics.
`averageUtilization`	Required. The threshold that is used to trigger scaling activities. For example, a value of `40` indicates that the CPU or memory usage threshold is set to 40%.
`scaleStrategy`	Optional. The scaling mode. Valid values: `auto` and `observer`. Default value: `observer`. ◦ `auto`: AHPA automatically performs scaling activities. ◦ `observer`: AHPA observes the resource usage but does not perform scaling activities. You can use the observer mode to check whether AHPA works as expected.
`maxReplicas`	Required. The maximum number of pods that can be provisioned for the application.
`minReplicas`	Required. The minimum number of pods that must be provisioned for the application.
`instanceBounds`	Optional. The duration of a scaling activity. ◦ `startTime`: the start time of a scaling activity. ◦ `endTime`: the end time of a scaling activity.
`cron`	Optional. This parameter is used to create a scheduled scaling job. The `cron` expression contains fields that are separated by five space characters. These fields describe the details about a schedule. For example, `- cron: "* 0-8 ? * MON-FRI"` specifies the time period from 00:00:00 to 08:00:00 on Monday to Friday each month.

The following table describes the fields that are contained in a `cron` expression. For more information, see Cron expression.

Field	Required	Valid value	Valid special character
Minutes	Yes	0~59	* / , -

Field	Required	Valid value	Valid special character
Hours	Yes	0~23	* / , -
Day of Month	Yes	1~31	* / , - ?
Month	Yes	1 to 12 or JAN to DEC	* / , -
Day of Week	No	0 to 6 or SUN to SAT	* / , - ?

Note

- The Month and Day of Week fields are not case-sensitive. For example, you can specify `SUN`, `Sun`, or `sun`.
- The default value of the Day of Week field is `*`.
- Descriptions of special characters:
    - `*`: specifies an arbitrary value.
    - `/`: specifies an increment.
    - `,`: separates a list of values.
    - `-`: specifies a range.
    - `?`: indicates a placeholder.

2. Run the following command to apply the AHPA policy:

```
kubectl apply -f ahpa-demo.yaml
```

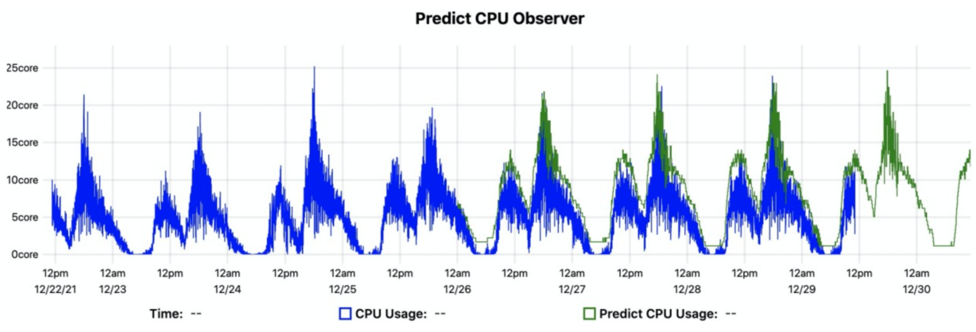
Verify the AHPA policy

In this section, an AHPA policy that uses the `observer` scaling mode is used as an example to check whether AHPA works as expected.

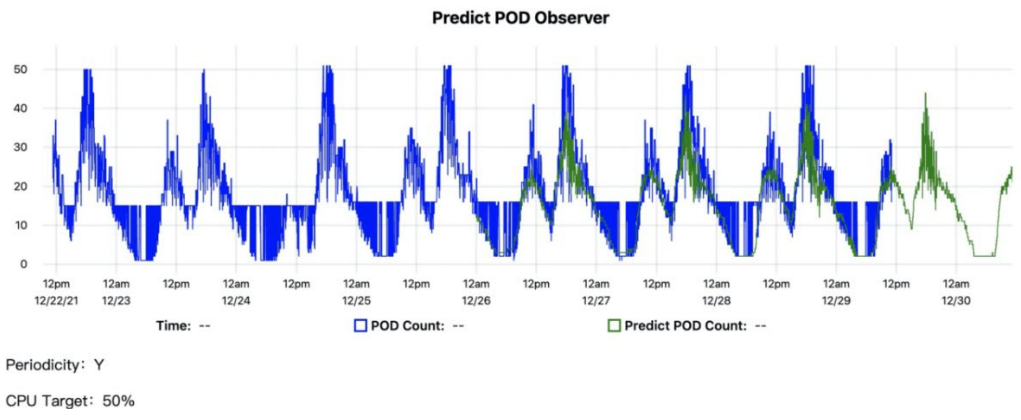
1. Run the following command to obtain the observer.html file. The file contains the AHPA prediction results that are compared with the HPA scaling results.

```
kubectl get --raw '/apis/metrics.alibabacloud.com/v1beta1/namespaces/default/prediction
sdeploymentobserver/fib-deployment'|jq -r '.content' |base64 -d > observer.html
```

2. Open the *observer.html* file and check the details. The following figures show the AHPA prediction results that are compared with the HPA scaling results based on the CPU usage.
    - Predict CPU Observer: The actual CPU usage based on HPA is represented by a blue line. The CPU usage predicted by AHPA is represented by a green line. The predicted CPU usage is higher than the actual CPU usage.



- Predict POD Observer: The actual number of pods that are provisioned by HPA is represented by a blue line. The number of pods that are predicted by AHPA is represented by a green line. The predicted number of pods is less than the actual number of pods. You can set the scaling mode to `auto` and configure other settings based on the predicted number of pods. This way, AHPA can save pod resources.



The results show that AHPA can use predictive scaling to handle fluctuating workloads as expected. After you confirm the prediction results, you can set the scaling mode to `auto`, which allows AHPA to automatically scale pods.

17.6. HPA FAQ

Issue 1: What do I do if HPA fails to collect resource metrics?

If a **FailedGetResourceMetric** warning is returned in the Events section, as shown in the following returned HPA conditions, Cloud Controller Manager (CCM) cannot collect resource metrics from the monitored resources.

```
Name:                                                  kubernetes-tutorial-deployment
Namespace:                                             default
Labels:                                                <none>
Annotations:                                           <none>
CreationTimestamp:                                     Mon, 10 Jun 2019 11:46:48 +0530
Reference:                                             Deployment/kubernetes-tutorial-
deployment
Metrics:                                               ( current / target )
  resource cpu on pods  (as a percentage of request):  <unknown> / 2%
Min replicas:                                          1
Max replicas:                                          4
Deployment pods:                                       1 current / 0 desired
Conditions:
  Type           Status  Reason                   Message
  ----           ------  ------                   -------
  AbleToScale    True    SucceededGetScale        the HPA controller was able to get the
target's current scale
  ScalingActive  False   FailedGetResourceMetric  the HPA was unable to compute the replica
count: unable to get metrics for resource cpu: unable to fetch metrics from resource
metrics API: the server is currently unable to handle the request (get pods.metrics.k8s.io)
Events:
  Type     Reason                   Age                     From                        
Message
  ----     ------                   ----                    ----                        ---
----
  Warning  FailedGetResourceMetric  3m3s (x1009 over 4h18m)  horizontal-pod-autoscaler
unable to get metrics for resource cpu: unable to fetch metrics from resource metrics API:
the server is currently unable to handle the request (get pods.metrics.k8s.io)
```

Possible causes:

- Cause 1: The data sources from which resource metrics are collected are unavailable.

  Run the `kubectl top pod` command to check whether the metric data of the monitored pods is returned. If no metric data is returned, run the `kubectl get apiservice` command to check whether the metrics-server component is available.

  The following output shows an example of the returned data:

```
NAME                                   SERVICE                      AVAILABLE   AGE
v1.                                    Local                        True        29h
v1.admissionregistration.k8s.io        Local                        True        29h
v1.apiextensions.k8s.io                Local                        True        29h
v1.apps                                Local                        True        29h
v1.authentication.k8s.io               Local                        True        29h
v1.authorization.k8s.io                Local                        True        29h
v1.autoscaling                         Local                        True        29h
v1.batch                               Local                        True        29h
v1.coordination.k8s.io                 Local                        True        29h
v1.monitoring.coreos.com               Local                        True        29h
v1.networking.k8s.io                   Local                        True        29h
v1.rbac.authorization.k8s.io           Local                        True        29h
v1.scheduling.k8s.io                   Local                        True        29h
v1.storage.k8s.io                      Local                        True        29h
v1alpha1.argoproj.io                   Local                        True        29h
v1alpha1.fedlearner.k8s.io             Local                        True        5h11m
v1beta1.admissionregistration.k8s.io   Local                        True        29h
v1beta1.alicloud.com                   Local                        True        29h
v1beta1.apiextensions.k8s.io           Local                        True        29h
v1beta1.apps                           Local                        True        29h
v1beta1.authentication.k8s.io          Local                        True        29h
v1beta1.authorization.k8s.io           Local                        True        29h
v1beta1.batch                          Local                        True        29h
v1beta1.certificates.k8s.io            Local                        True        29h
v1beta1.coordination.k8s.io            Local                        True        29h
v1beta1.events.k8s.io                  Local                        True        29h
v1beta1.extensions                     Local                        True        29h
...
[v1beta1.metrics.k8s.io                 kube-system/metrics-server   True         29h]
...
v1beta1.networking.k8s.io              Local                        True        29h
v1beta1.node.k8s.io                    Local                        True        29h
v1beta1.policy                         Local                        True        29h
v1beta1.rbac.authorization.k8s.io      Local                        True        29h
v1beta1.scheduling.k8s.io              Local                        True        29h
v1beta1.storage.k8s.io                 Local                        True        29h
v1beta2.apps                           Local                        True        29h
v2beta1.autoscaling                    Local                        True        29h
v2beta2.autoscaling                    Local                        True        29h
```

If the *apiservice* for **v1beta1.metrics.k8s.io** is not **kube-system/metrics-server**, check whether metrics-server is overwritten by Prometheus Operator. If metrics-server is overwritten by Prometheus Operator, use the following YAML template to redeploy metrics-server:

```
apiVersion: apiregistration.k8s.io/v1beta1
kind: APIService
metadata:
  name: v1beta1.metrics.k8s.io
spec:
  service:
    name: metrics-server
    namespace: kube-system
  group: metrics.k8s.io
  version: v1beta1
  insecureSkipTLSVerify: true
  groupPriorityMinimum: 100
  versionPriority: 100
```

If no error is found after you perform the preceding checks, see the troubleshooting section in the related topic of *metrics-server*.

- Cause 2: Metrics cannot be collected during a rolling update or scale-out activity.

  By default, metrics-server collects metrics at intervals of 1 second. However, metrics-server must wait a few seconds before it can collect metrics after it performs a rolling update or scale-out activity. We recommend that you update metrics 2 seconds after a rolling update or scale-out activity.

- Cause 3: The request field is missing.

  HPA automatically obtains the CPU or memory usage by calculating the value of `used resource/requested resource` of the pod. If the requested resource is not specified in the pod configurations, HPA cannot calculate the resource usage. Therefore, you must make sure that the requests field is specified in the pod configurations.

Issue 2: An excessive number of pods are added by HPA during a rolling update
During a rolling update, kube-controller-manager performs zero filling on pods whose monitoring data cannot be collected. This may cause HPA to add an excessive number of pods. Check whether metrics-server is updated to the latest version and configure the following startup settings for the pod on which metrics-server is deployed:

```
Add the following configuration to the startup settings.
--enable-hpa-rolling-update-skipped=true
```

Issue 3:What do I do if HPA does not scale pods when the scaling threshold is reached?
HPA may not scale the number of pods even if the CPU or memory usage drops below the scale-in threshold or exceeds the scale-out threshold. HPA also takes other factors into consideration when it scales pods. For example, it checks whether the current scale-out event triggers a scale-in activity or the scale-in event triggers a scale-out activity. This avoids repetitive scaling and prevents unnecessary resource consumption.
Issue 4: How do I set the data collection interval for HPA?
For metrics-server versions later than v0.2.1-b46d98c-aliyun, specify the `--metric_resolution` parameter in the startup settings. Example: `--metric_resolution=15s` .
Issue 4: I configured a CPU HPA and a memory HPA. Can the memory HPA trigger a scale-in activity after the CPU HPA triggers a scale-out activity?
Yes, after CPU Horizontal Pod Autoscaler (HPA) triggers a scale-out activity, the memory HPA triggers a scale-in activity. We recommend that you use the same HPA to monitor CPU and memory usage.
Issue 5: Must both the CPU usage and memory usage exceed the thresholds to trigger a scale-out

activity if I use the same HPA to monitor CPU and memory usage?

The numbers of pods to be scaled out based on the CPU usage and memory usage may be different. To ensure the stability of the workloads, the system preferably performs the scale-in activity that adds more pods when both the CPU usage and memory usage trigger scale-out activities. The same rule applies to scale-in activities. The system preferably performs the scale-in activity that reserves more pods when both the CPU usage and memory usage trigger scale-in activities.

18.Best practices for ASK clusters

18.1. Use ASK to run jobs

In a serverless Kubernetes (ASK) cluster, you can create pods as needed. The system stops billing after a pod is stopped and you do not need to reserve computing resources to handle jobs. This solves the issues of insufficient computing resources and saves you the need to expand the cluster. In addition, you can reduce the computing cost by using preemptible instances. This topic describes how to use ASK to create jobs to meet your business requirements.

Prerequisites

- Create an ASK cluster
- Connect to an ACK cluster by using kubectl

Procedure

1. Use the kubectl client to create the *job.yaml* file and copy the following content into the file:

```
apiVersion: batch/v1
kind: Job
metadata:
  name: pi
spec:
  template:
    spec:
      containers:
      - name: pi
        image: perl
        command: ["perl",  "-Mbignum=bpi", "-wle", "print bpi(2000)"]
        resources:
          requests:
            cpu: 16
            memory: 32Gi
      restartPolicy: Never
  backoffLimit: 4
```

2. Run the following command to deploy a job:

```
kubectl apply -f job.yaml
```

3. Run the following commands to check the state of the pod.

   Run the following command to check the state of the pod:

```
kubectl get pod
```

```
NAME       READY   STATUS      RESTARTS   AGE
pi-4f7w5   0/1     Completed   0          80s
```

   Run the following command to view detailed information about the state of the pod:

```
kubectl describe pod
```

```
Name:               pi-4f7w5
Namespace:          default
Priority:           0
PriorityClassName:  <none>
Node:               virtual-kubelet-cn-hongkong-b/10.10.66.169
...
Events:
  Type    Reason                 Age   From          Message
  ----    ------                 ----  ----          -------
  Normal  SuccessfulMountVolume  114s  kubelet, eci  MountVolume.SetUp succeeded for vo
lume "default-token-8k4jz"
  Normal  Pulling                113s  kubelet, eci  pulling image "perl"
  Normal  Pulled                 64s   kubelet, eci  Successfully pulled image "perl"
  Normal  Created                64s   kubelet, eci  Created container
  Normal  Started                64s   kubelet, eci  Started container
```

4. (Optional)To use a preemptible instance, add a preemptible instance annotation to the pod.

   For more information about how to a preemptible instance annotation, see Use preemptible instances.

```
apiVersion: batch/v1
kind: Job
metadata:
  name: pi
spec:
  template:
    metadata:
      annotations:
        k8s.aliyun.com/eci-spot-strategy: SpotAsPriceGo
    spec:
      containers:
      - name: pi
        image: perl
        command: ["perl",  "-Mbignum=bpi", "-wle", "print bpi(2000)"]
        resources:
          requests:
            cpu: 16
            memory: 32Gi
      restartPolicy: Never
  backoffLimit: 4
```

18.2. Use ASK to create Spark tasks

In a serverless Kubernetes (ASK) cluster, you can create pods to meet your business requirements. The system stops billing a pod after the pod lifecycle is terminated. You do not need to reserve computing resources for Spark tasks. This resolves the issues of insufficient computing resources and saves you the need to expand the cluster. In addition, you can reduce the computing costs by using preemptible instances. This topic describes how to use ASK to create Spark tasks to meet your business requirements.

Prerequisites

- An ASK cluster is created. For more information, see Create an ASK cluster.
- A kubectl client is connected to the cluster. For more information, see Connect to an ACK cluster by using kubectl.

Procedure

1. Deploy the ack-spark-operator chart by using one of the following methods:
   - Log on to the Container Service for Kubernetes (ACK) console. In the left-side navigation pane, choose **Marketplace > App Catalog** and select ack-spark-operator to deploy the chart.
   - Run the **helm** command to manually deploy the chart.

     Note The Helm version must be V3 or later.

     ```
     # Create a service account.
     kubectl create serviceaccount spark
     # Grant permissions.
     kubectl create clusterrolebinding spark-role --clusterrole=edit --serviceaccount=defa
     ult:spark --namespace=default
     # Install the operator.
     helm repo add incubator http://storage.googleapis.com/kubernetes-charts-incubator
     helm install incubator/sparkoperator --namespace default  --set operatorImageName=reg
     istry.cn-hangzhou.aliyuncs.com/acs/spark-operator  --set operatorVersion=ack-2.4.5-la
     test  --generate-name
     ```

   After you deploy the chart, run the following command to check whether spark-operator is started:

   ```
   kubectl -n spark-operator get pod
   ```

   Expected output:

   ```
   NAME                                  READY   STATUS      RESTARTS   AGE
   ack-spark-operator-7698586d7b-pvwln   1/1     Running     0          5m9s
   ack-spark-operator-init-26tvh         0/1     Completed   0          5m9s
   ```

2. Create a file named *spark-pi.yaml* and copy the following content into the file:

```
apiVersion: "sparkoperator.k8s.io/v1beta2"
kind: SparkApplication
metadata:
  name: spark-pi
  namespace: default
spec:
  arguments:
  - "1000"
  sparkConf:
    "spark.scheduler.maxRegisteredResourcesWaitingTime": "3000s"
    "spark.kubernetes.allocation.batch.size": "1"
    "spark.rpc.askTimeout": "36000s"
    "spark.network.timeout": "36000s"
    "spark.rpc.lookupTimeout": "36000s"
    "spark.core.connection.ack.wait.timeout": "36000s"
    "spark.executor.heartbeatInterval": "10000s"
  type: Scala
  mode: cluster
  image: "registry.aliyuncs.com/acs/spark:ack-2.4.5-latest"
  imagePullPolicy: Always
  mainClass: org.apache.spark.examples.SparkPi
  mainApplicationFile: "local:///opt/spark/examples/jars/spark-examples_2.11-2.4.5.jar"
  sparkVersion: "2.4.5"
  restartPolicy:
    type: Never
  args:
  driver:
    cores: 4
    coreLimit: "4"
    annotations:
      k8s.aliyun.com/eci-image-cache: "true"
    memory: "6g"
    memoryOverhead: "2g"
    labels:
      version: 2.4.5
    serviceAccount: spark
  executor:
    annotations:
      k8s.aliyun.com/eci-image-cache: "true"
    cores: 2
    instances: 1
    memory: "3g"
    memoryOverhead: "1g"
    labels:
      version: 2.4.5
```

3. Deploy a Spark task.
   i. Run the following command to deploy a Spark task:

   ```
   kubectl apply -f spark-pi.yaml
   ```

   Expected output:

```
sparkapplication.sparkoperator.k8s.io/spark-pi created
```

ii. Run the following command to view the deployment status of the Spark task:

```
kubectl get pod
```

Expected output:

```
NAME                READY   STATUS     RESTARTS    AGE
spark-pi-driver     1/1     Running    0           2m12s
```

The output shows that the pod is in the Running state, which indicates that the Spark task is being deployed.

iii. Run the following command to view the deployment status of the Spark task again:

```
kubectl get pod
```

Expected output:

```
NAME                READY   STATUS       RESTARTS    AGE
spark-pi-driver     0/1     Completed    0           2m54s
```

The output shows that the pod is in the Completed state, which indicates that the Spark task is deployed.

4. Run the following command to view the computing result of the Spark task:

```
kubectl logs spark-pi-driver|grep Pi
```

Expected output:

```
20/04/30 07:27:51 INFO DAGScheduler: ResultStage 0 (reduce at SparkPi.scala:38) finishe
d in 11.031 s
20/04/30 07:27:51 INFO DAGScheduler: Job 0 finished: reduce at SparkPi.scala:38, took 1
1.137920 s
Pi is roughly 3.1414371514143715
```

5. (Optional)To use a preemptible instance, add annotations for preemptible instances to the pod.

   For more information about how to add annotations for preemptible instances, see Use preemptible instances.

18.3. Associate an EIP with a pod

This topic describes how to associate an elastic IP address (EIP) with a pod in a serverless Kubernetes (ASK) cluster or a pod on a virtual node.

Context

Container Service for Kubernetes (ACK) provides ASK clusters and virtual nodes. You can associate an EIP with a pod in an ASK cluster or a pod on a virtual node. This facilitates how you deploy applications to an ASK cluster and access these applications. This feature has the following benefits:

- A pod that is associated with an EIP can access the Internet. You do not need to configure a NAT gateway for the virtual private cloud (VPC).
- A pod that is associated with an EIP can be accessed from the Internet. You do not need to deploy a

Service to expose the pod.

- You can dynamically associate an EIP with a pod.

Prerequisites

- An ASK cluster is created or a virtual node is deployed in a Kubernetes cluster. For more information, see Create an ASK cluster and Virtual nodes.
- Required ports are exposed in the security group rules for the cluster. In the following example, port 80 is exposed.

> **Note**
>
> - Upgrade Virtual Kubelet to a version that is supported by v1.0.0.7-aliyun.
> - You can associate an EIP with a pod only when you create the pod. If you associate an EIP with a pod when you modify the configuration of the pod, the association does not take effect.

Procedure

You can use the following methods to associate an EIP with a pod:

Method 1: Automatically associate an EIP with a pod

1.
2.
3.
4.
5. On the **Deployments** page, click **Create from YAML**, select a sample template or customize a template, and then click **Create**.

   You can use the following YAML template to create a pod. In this example, *k8s.aliyun.com/eci-with-eip* is set to *true*. This indicates that the ASK cluster or a virtual node automatically assigns and associates an EIP with the pod.

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  annotations:
    k8s.aliyun.com/eci-with-eip: "true"
  # k8s.aliyun.com/eip-bandwidth: '5' #Note: Do not set a unit for the specified bandwi
dth
spec:
  containers:
  - image: nginx:alpine
    imagePullPolicy: Always
    name: nginx
    ports:
    - containerPort: 80
      name: http
      protocol: TCP
  restartPolicy: OnFailure
```

Note

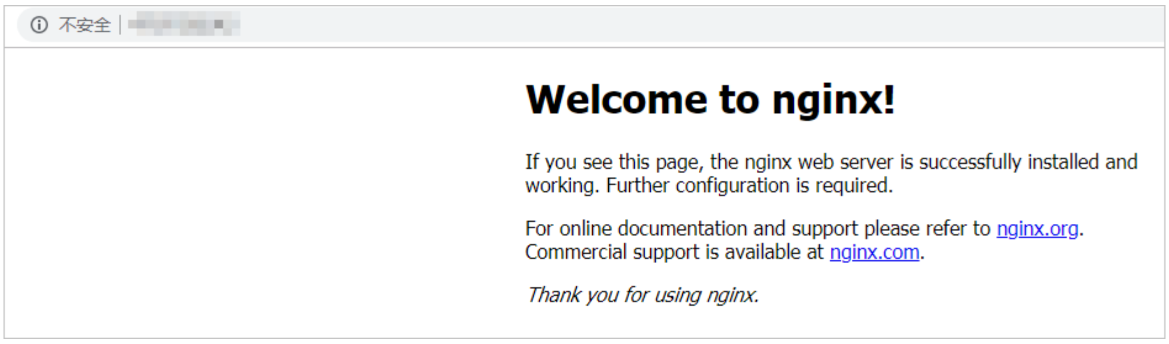
- You can use the *k8s.aliyun.com/eip-bandwidth* annotation to specify the bandwidth limit of the EIP. Default value: 5. Unit: Mbit/s.
- You can also use the *k8s.aliyun.com/eip-common-bandwith-package-id* annotation to associate the EIP with an EIP bandwidth plan.
- If the YAML template creates a Deployment, the system assigns an EIP to each pod in the Deployment. Proceed with caution.

6. In the left-side navigation pane, choose **Workloads > Pods** to view the states of pods.
7. Find the pod that you want to manage and click **Edit** in the Actions column. The **Edit YAML** dialog box appears.

Note In the YAML file of the pod, the IP address in the `k8s.aliyun.com/allocated-eipAddress: 47.110.XX.XX` field refers to the public IP address of the EIP.

8. Enter *http://IP address* into the address bar of a browser to visit the NGINX welcome page.

http://IP address refers to the IP address in `k8s.aliyun.com/allocated-eipAddress: 47.110.XX.XX` of the YAML file.



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

> Note This method dynamically assigns an EIP to a pod. The lifecycle of the EIP is the same as that of the pod. If you delete the pod, the EIP assigned to the pod is automatically deleted.

9. (Optional)If you want to specify a line for the EIP that is associated with an Elastic Container Instance-based pod, you must add the `k8s.aliyun.com/eip-isp` annotation and specify a value.

   ISP indicates the Internet connection type of the EIP. By default, this parameter is set to `BGP`. For more information, see AllocateEipAddressPro.

   The following YAML template provides an example:

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  annotations:
    k8s.aliyun.com/eci-with-eip: "true"
    k8s.aliyun.com/eip-isp: "BGP"
spec:
  containers:
  - image: nginx:alpine
    name: nginx
    ports:
    - containerPort: 80
      name: http
      protocol: TCP
  restartPolicy: OnFailure
```

Method 2: Specify an EIP ID

1. Log on to the VPC console and apply for an EIP. For more information, see Apply for an EIP.

   > Note The EIP that you apply for and the cluster must be deployed in the same region.

2.
3.
4.
5.
6. On the **Deployments** page, click **Create from YAML**, select a sample template or customize a template, and then click **Create**.

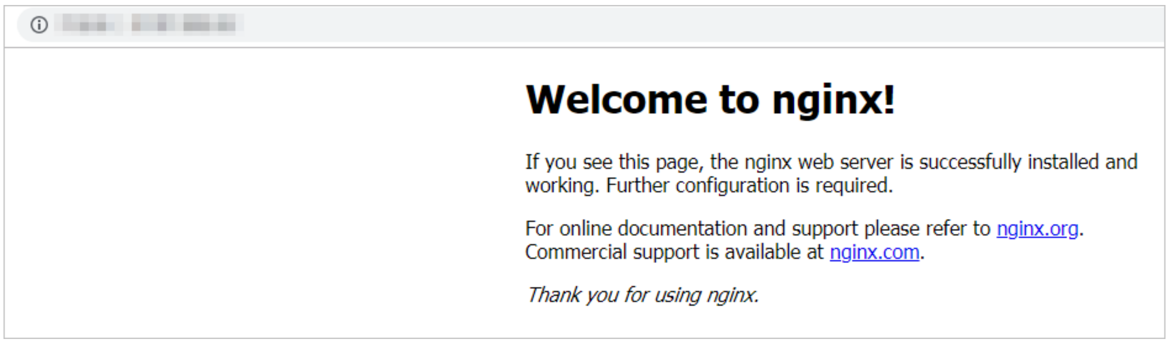
   You can use the following YAML template to create a pod. In this example, you can specify the ID of the EIP in the *k8s.aliyun.com/eci-eip-instanceid* annotation.

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  annotations:
    "k8s.aliyun.com/eci-eip-instanceid": "<youreipInstanceId>"
spec:
  containers:
  - image: nginx:alpine
    imagePullPolicy: Always
    name: nginx
    ports:
    - containerPort: 80
      name: http
      protocol: TCP
  restartPolicy: OnFailure
```

Note

- Replace `<youreipInstanceId>` with the ID of the EIP obtained in Step 1.
- If the system automatically assigns an EIP to a pod and you also specify an EIP for the pod, the EIP that you specify is not used.

7. In the left-side navigation pane, choose **Workloads > Pods** to view the states of pods.
8. Enter *http://IP address* into the address bar of a browser to visit the NGINX welcome page.



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

Note *http://IP address* refers to the IP address of the EIP that you applied for in Step 1.

18.4. Deploy Jenkins in an ASK cluster and build an application delivery pipeline

This topic describes how to deploy a Jenkins continuous integration environment in a serverless Kubernetes (ASK) cluster. This topic also provides step-by-step instructions on how to build an application delivery pipeline that includes source code compilation, image building and pushing, and application deployment.

Prerequisites

- An ASK cluster is created. For more information, see Create an ASK cluster.
- A kubectl client is connected to the cluster. For more information, see Connect to an ACK cluster by using kubectl.

Deploy Jenkins

1. Run the following command to download the Jenkins package:

```
git clone https://github.com/AliyunContainerService/jenkins-on-serverless.git
cd jenkins-on-serverless
```

2. Store the `jenkins_home` directory in persistent storage.

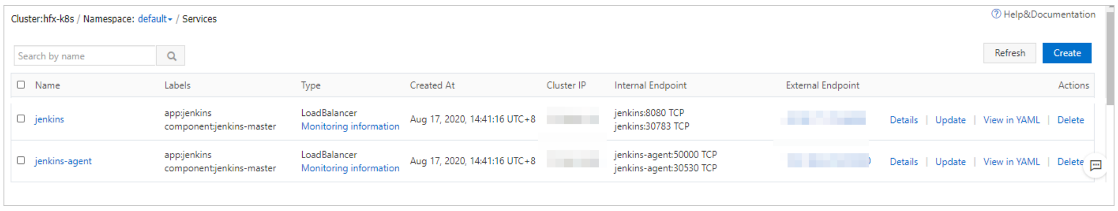
   If you require persistent storage, you can mount an `nfs volume` to the `jenkins_home` directory. You must modify the *serverless-k8s-jenkins-deploy.yaml* file to uncomment the following lines and set Network File System (NFS) parameters:

```
#volumeMounts:
        #  - mountPath: /var/jenkins_home
        #    name: jenkins-home
      .....
      #volumes:
      #  - name: jenkins-home
      #    nfs:
      #      path: /
      #      server:
```

3. Run the following command to deploy Jenkins:

```
kubectl apply -f serverless-k8s-jenkins-deploy.yaml
```

4. Log on to Jenkins.
   i.
   ii.
   iii.
   iv.
   v. Select Jenkins and click its external endpoint to log on to Jenkins.



Name	Labels	Type	Created At	Cluster IP	Internal Endpoint	External Endpoint	Actions
jenkins	app:jenkins component:jenkins-master	LoadBalancer Monitoring information	Aug 17, 2020, 14:41:16 UTC+8	[redacted]	jenkins:8080 TCP jenkins:30783 TCP	[redacted]	Details | Update | View in YAML | Delete
jenkins-agent	app:jenkins component:jenkins-master	LoadBalancer Monitoring information	Aug 17, 2020, 14:41:16 UTC+8	[redacted]	jenkins-agent:50000 TCP jenkins-agent:30530 TCP	[redacted]	Details | Update | View in YAML | Delete

   vi. On the Jenkins logon page, enter the username and password. The default username and password are both **admin**.

   Notice To ensure system security, change the password after you log on to Jenkins.

5. Configure the settings on the **Getting Started** page.

   i. On the **Getting Started** page, click **Install suggested plugins**.
   ii. After the plug-ins are installed, click **Save and Finish** in the **Instance Configuration** section of the **Getting Started** page.
   iii. Click **Start using Jenkins**.
6. Obtain the Secret that contains a token.
   i. Run the following command to query the token:

```
kubectl get secret
```

   Expected output:

```
NAME                         TYPE                                  DATA   AGE
ack-jenkins-sa-token-q****   kubernetes.io/service-account-token   3      28m
default-token-b****          kubernetes.io/service-account-token   3      27h
```

   ii. Run the following command to obtain the Secret named **ack-jenkins-sa-token-q******:

```
kubectl get secret ack-jenkins-sa-token-q**** -o jsonpath={.data.token} |base64 -d
```

   Expected output:

```
sdgdrh****
```

7. Create a credential of the Secret text type.
   i. In the left-side navigation pane of the Jenkins dashboard, click **Manage Jenkins**.
   ii. On the **Manage Jenkins** page, click **Manage Nodes and Clouds** in the **System Configuration** section.
   iii. In the left-side navigation pane of the **Nodes** page, click **Configure Clouds**.
   iv. On the **Configure Clouds** page, click **Kubernetes Cloud details...**.
   v. In the **Credentials** field, choose **Add > Jenkins**.

vi. In the **Jenkins Credentials Provider: Jenkins** dialog box, add a credential of the Secret text type.

The following table describes the credential parameters.

Parameter	Description
Domain	The domain of the credential. Default value: **Global credentials (unrestricted)**.
Kind	The kind of the credential. In this example, **Secret text** is selected.
Scope	The scope of the credential. Valid values: Global and System. In this example, **Global (Jenkins, nodes, items, all child items, etc)** is selected.
Secret	The Secret of the credential. In this example, the Secret obtained from the previous step is entered.
ID	The name of the credential. In this example, **ask-jenkins-token** is entered.
Description	The description of the credential.

vii. Click **Add**.

8. On the **Configure Clouds** page, set the required parameters. For more information, see Configure Kubernetes Cloud.

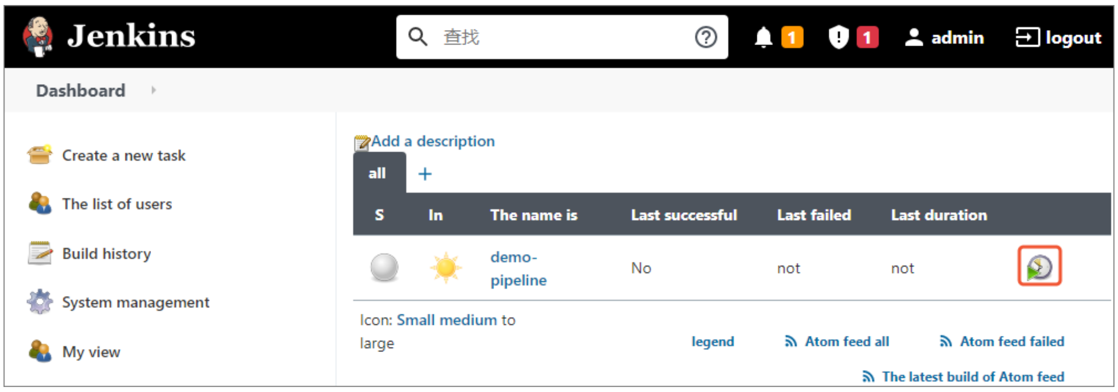
i. Set the **Kubernetes address**, set **Credentials** to **ask-jenkins-token**, and then click **Connection test** to check the connectivity between Jenkins and the cluster.

ii. Set the **Jenkins address** and **Jenkins channel** parameters.

iii. Click **Save**.

9. Build a pipeline named **demo-pipeline** and access the pipeline.

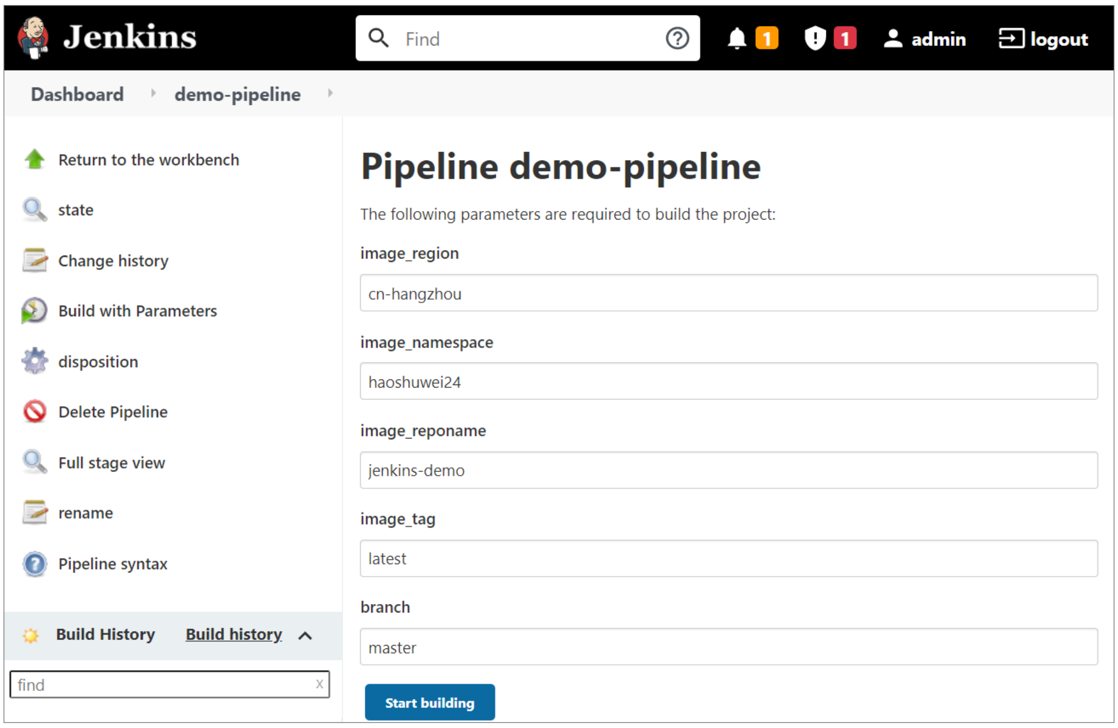
i. On the Jenkins homepage, click the icon to the right side of **demo-pipeline**.



S	In	The name is	Last successful	Last failed	Last duration
●	☀	demo-pipeline	No	not	not

Icon: Small medium to large   legend   Atom feed all   Atom feed failed   The latest build of Atom feed

ii. Modify the build parameters based on information about your image repository. In this example, the branch of the source code repository is `master` and the image is `registry.cn-beijing.aliyuncs.com/ack-cicd/ack-jenkins-demo:latest`.



Pipeline demo-pipeline

The following parameters are required to build the project:

image_region: cn-hangzhou

image_namespace: haoshuwei24

image_reponame: jenkins-demo

image_tag: latest

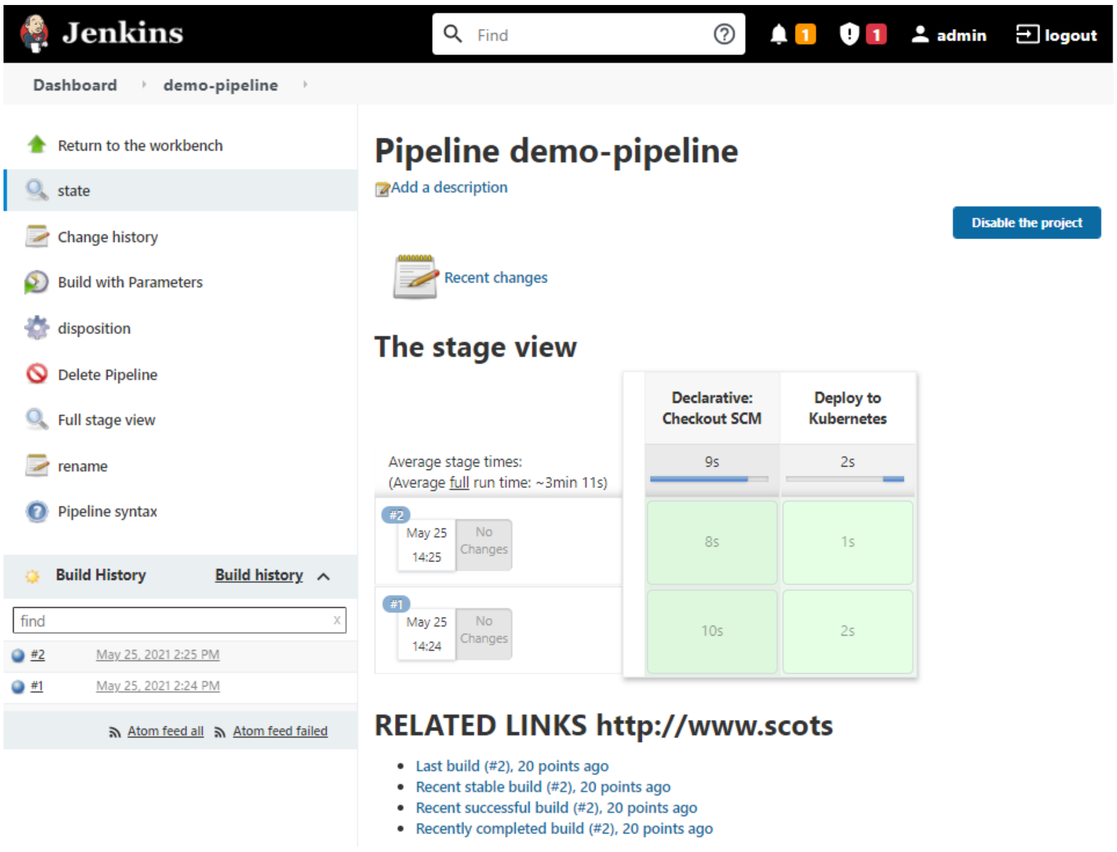
branch: master

iii. Click **Start building**.

Test the connectivity between the Jenkins master and the Jenkins slave pod that is dynamically allocated by the Kubernetes cluster.

After you click Build, Jenkins dynamically creates a slave pod in the Kubernetes cluster to run the build job. For more information about the sample application code, see jenkins-demo-GitHub or jenkins-demo-haoshuwei.

iv. In the left-side navigation pane, click **Status**. If the build job is successful, Jenkins runs as normal on Kubernetes.



What's next

- For more information about how to configure the Maven cache for slave pods, see Configure the Maven cache for slave pods.
- For more information about how to use kaniko to build and push container images, see Use kaniko to build and push container images.

18.5. Run a GPU-based TensorFlow training job

This topic describes how to use a Serverless Kubernetes (ASK) cluster and an elastic container instance to run a GPU-based TensorFlow training job.

Background information

In recent years, artificial intelligence (AI) and machine learning have been widely applied in a large number of fields, and various training models have been developed. An increasing number of training jobs are run on the cloud. However, it is not easy to continuously run training jobs in a cloud environment. You may encounter the following difficulties:

- Difficulties in deploying environments: You must purchase a GPU-accelerated instance and install a GPU driver on the instance. After you prepare images for training jobs, you must install the GPU runtime hook.

- Lack of scalability: After you deploy an environment and run the training jobs, you may need to release idle resources to save costs. The next time you want to run training jobs, you must deploy an environment and create instances again. If compute nodes are insufficient, you must scale out the compute nodes. In this case, you must create instances and deploy the environment again.

To resolve the preceding difficulties, we recommend that you use ASK clusters and elastic container instances to run training jobs. This solution has the following benefits:

- You are charged on a pay-as-you-go basis and do not need to manage resources.
- You need only to prepare the configurations once. Then, you can reuse the configurations without limits.
- The image cache feature allows you to create pods and start training jobs in a more efficient manner.
- Training data is decoupled from training models. Training data can be persisted.

Preparations

1. Prepare a container image and training data for the training model.

   In this example, a TensorFlow training job on GitHub is used. You can obtain the sample image eci/tensorflow from Alibaba Cloud Container Registry. For more information, see TensorFlow Model Garden.

2. Create an ASK cluster.

   Create an ASK cluster in the Container Service for Kubernetes console. For more information, see Create an ASK cluster.

   > **Note**
   >
   > If you want to pull an image from the Internet or if your training jobs need to access the Internet, you must configure a network address translation (NAT) gateway.

   You can use kubectl to manage and access the ASK cluster. You can use one of the following methods:

   - If you want to manage the cluster from your computer, install and configure the kubectl client. For more information, see Use kubectl to connect to an ACK cluster.
   - Use kubectl on Cloud Shell to manage the cluster. For more information, see Use kubectl on Cloud Shell to manage ACK clusters.

3. Create a network-attached storage (NAS) file system and add a mount target.

   Create a NAS file system and add a mount target in the NAS File System console. For more information, see Manage file systems and Manage mount targets.

   > **Note**
   >
   > The NAS file system and the ASK cluster must be in the same VPC.

Create an image cache

The image cache feature has been integrated into Kubernetes CRD to accelerate the pulling of images. For more information, see Use an image cache CRD to accelerate pod creation.

Perform the following operations:

1. Prepare the YAML file.

   The following code shows an example of the imagecache.yaml file:

   ```
   apiVersion: eci.alibabacloud.com/v1
   kind: ImageCache
   metadata:
     name: tensorflow
   spec:
     images:
     - registry-vpc.cn-beijing.aliyuncs.com/eci/tensorflow:1.0 # The image of the training
   job. We recommend that you upload the image to Alibaba Cloud Container Registry. The en
   dpoint of the VPC is used. Make sure that the VPC is the same as the one to which the c
   luster belongs.
   ```

2. Create an image cache.

   ```
   kubectl create -f imagecache.yaml
   ```

   You must pull an image when you create an image cache. The amount of time required to pull an image is related to the image size. You can run the following command to view the creation progress of the image cache:

   ```
   Kubectl get imagecache tensorflow
   ```

   A command output similar to the following one indicates that the image cache is created.

   ```
   NAME         AGE   CACHEID                     PHASE   PROGRESS
   tensorflow   11m   imc-2ze1xczztv7tgesg****    Ready   100%
   ```

Create a training job

You can use the image cache to create a training job.

1. Prepare the YAML file.

   The following code shows an example of the gpu_pod.yaml file:

```
apiVersion: v1
kind: Pod
metadata:
  name: tensorflow
  annotations:
    k8s.aliyun.com/eci-use-specs: "ecs.gn6i-c4g1.xlarge"  # Specify the GPU-accelerated instance type that is used to create an elastic container instance. Example: k8s.aliyun.com/eci-instance-type: "ecs.gn5i-c2g1.large"
    k8s.aliyun.com/eci-image-cache: "true"             # Enable Automatically Match Image Cache.
spec:
  containers:
  - name: tensorflow
    image: registry-vpc.cn-beijing.aliyuncs.com/eci/tensorflow:1.0 # The image of the training job.
    command:
      - "sh"
      - "-c"
      - "python models/tutorials/image/imagenet/classify_image.py" # The script used to start the training job.
    resources:
      limits:
        nvidia.com/gpu: "1"   # The number of GPUs required by the container.
    volumeMounts:
    - name: nfs-pv
      mountPath: /tmp/imagenet
  volumes:
  - name: nfs-pv   #Persist the training results to NAS files.
    flexVolume:
        driver:  alicloud/nas
        fsType: nfs
        options:
          server: 16cde4****-ijv**.cn-beijing.nas.aliyuncs.com     #The mount target of the NAS file system.
          path: /          #The mount directory.
  restartPolicy: OnFailure
```

2. Run the following command to create a pod:

```
kubectl create -f gpu_pod.yaml
```

3. View the execution results.

   You can view events or logs.

   - View events

   ```
   kubectl describe pod tensorflow
   ```

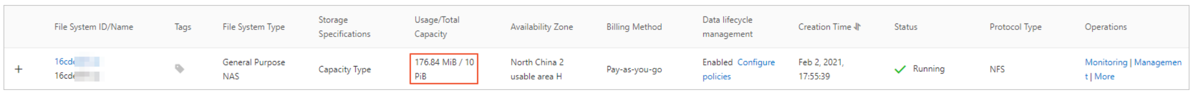
   - View logs

   ```
   kubectl logs tensorflow
   ```

View the results

You can view the results of the training job in the console.

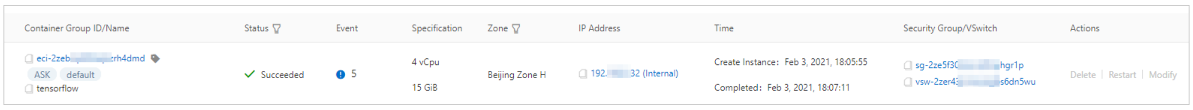
- You can view the storage capacity occupied by the training data in the NAS File System console.



	File System ID/Name	Tags	File System Type	Storage Specifications	Usage/Total Capacity	Availability Zone	Billing Method	Data lifecycle management	Creation Time	Status	Protocol Type	Operations
+	16cd[redacted] 16cd[redacted]		General Purpose NAS	Capacity Type	176.84 MiB / 10 PiB	North China 2 usable area H	Pay-as-you-go	Enabled Configure policies	Feb 2, 2021, 17:55:39	✓ Running	NFS	Monitoring | Management | More

> **Note**
> Training results are stored in the configured NAS file system. After the NAS file system is mounted, you can obtain the results in the corresponding path.

- You can view the elastic container instance that is successfully run in the Elastic Container Instance console.



Container Group ID/Name	Status	Event	Specification	Zone	IP Address	Time	Security Group/VSwitch	Actions
eci-2zeb[redacted]rh4dmd ASK default tensorflow	✓ Succeeded	5	4 vCpu 15 GiB	Beijing Zone H	192.[redacted]32 (Internal)	Create Instance: Feb 3, 2021, 18:05:55 Completed: Feb 3, 2021, 18:07:11	sg-2ze5f3[redacted]hgr1p vsw-2zer4[redacted]s6dn5wu	Delete | Restart | Modify

> **Note**
> If an elastic container instance is successfully run, containers in the instance are stopped. Then, the system reclaims the underlying computing resources, and the billing of pods is stopped.

18.6. Deploy a Spark application

This topic describes how to use Serverless Kubernetes(ASK) and Elastic Container Instance to deploy a Spark application.

Background information

Apache Spark is an open source program that is widely used to analyze workloads in scenarios such as big data computing and machine learning. You can use Kubernetes to run and manage resources on Apache Spark 2.3.0 and later.

Kubernetes Operator for Apache Spark is designed to run Spark jobs in Kubernetes clusters. It allows you to submit Spark tasks that are defined in custom resource definitions (CRDs) to Kubernetes clusters. Kubernetes Operator for Apache Spark provides the following benefits:

- Compared with open source Apache Spark, Kubernetes Operator for Apache Spark provides more features.
- Kubernetes Operator for Apache Spark can be integrated with the storage, monitoring, and logging components of Kubernetes.
- Kubernetes Operator for Apache Spark supports advanced Kubernetes features such as disaster recovery and auto scaling. In addition, Kubernetes Operator for Apache Spark can optimize resource scheduling.

Preparations

1. Create an ASK cluster.

   Create an ASK cluster in the Container Service for Kubernetes (ACK) console. For more information, see Create an ASK cluster.

> **Note**
>
> If you want to pull an image from the Internet or if your training jobs need to access the Internet, you must configure a NAT gateway.

To manage and access the ASK cluster by using kubectl, perform the following operations:

- If you want to manage the cluster from an on-premises machine, install and configure the kubectl client. For more information, see Connect to ACK clusters by using kubectl.
- You can also use kubectl to manage the ASK cluster on Cloud Shell. For more information, see Use kubectl to manage ACK clusters on Cloud Shell.

2. Create an OSS bucket.

   You must create an Object Storage Service (OSS) bucket to store data, including the test data, test results, and logs. For more information, see Create buckets.

Install Kubernetes Operator for Apache Spark

1. Install Kubernetes Operator for Apache Spark.
   i. In the left-side navigation pane of the ACK console, choose **Marketplace** > **Marketplace**.
   ii. On the **App Catalog** tab, find and click **ack-spark-operator**.
   iii. Click **Deploy** and configure the parameters in the panel.

      On the Parameters wizard page, set the `sparkJobNamespace` parameter to the namespace where you want to deploy the Spark job. The default value is `default`. An empty string indicates that the Spark job monitors all namespaces.

2. Create a ServiceAccount, Role, and RoleBinding.

   A Spark job requires a ServiceAccount to acquire the permissions to create pods. Therefore, you must create a ServiceAccount, Role, and RoleBinding. The following YAML example shows how to create a ServiceAccount, Role, and RoleBinding. Replace the namespaces with the actual values.

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: spark
  namespace: default
---
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  namespace: default
  name: spark-role
rules:
- apiGroups: [""]
  resources: ["pods"]
  verbs: ["*"]
- apiGroups: [""]
  resources: ["services"]
  verbs: ["*"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: spark-role-binding
  namespace: default
subjects:
- kind: ServiceAccount
  name: spark
  namespace: default
roleRef:
  kind: Role
  name: spark-role
  apiGroup: rbac.authorization.k8s.io
```

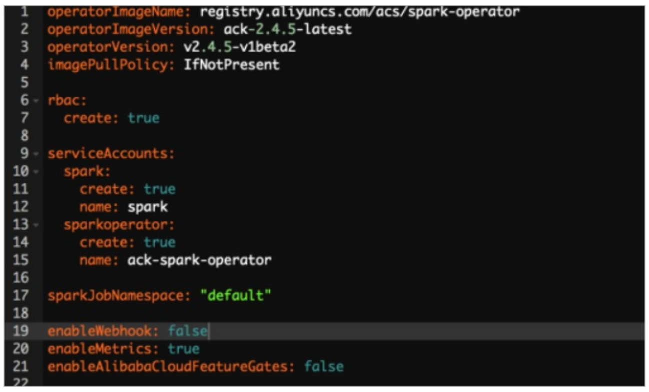
Build an image of the Spark job

You need to compile the Java Archive (JAR) package of the Spark job and use Dockerfile to package the image.

The following example shows how to configure Dockerfile when a Spark base image of ACK is used.

```
FROM registry.aliyuncs.com/acs/spark:ack-2.4.5-latest
RUN mkdir -p /opt/spark/jars
# If you want to read data from OSS or sink events to OSS, add the following JAR packages t
o the image.
ADD https://repo1.maven.org/maven2/com/aliyun/odps/hadoop-fs-oss/3.3.8-public/hadoop-fs-oss
-3.3.8-public.jar $SPARK_HOME/jars
ADD https://repo1.maven.org/maven2/com/aliyun/oss/aliyun-sdk-oss/3.8.1/aliyun-sdk-oss-3.8.1
.jar $SPARK_HOME/jars
ADD https://repo1.maven.org/maven2/org/aspectj/aspectjweaver/1.9.5/aspectjweaver-1.9.5.jar
$SPARK_HOME/jars
ADD https://repo1.maven.org/maven2/org/jdom/jdom/1.1.3/jdom-1.1.3.jar $SPARK_HOME/jars
COPY SparkExampleScala-assembly-0.1.jar /opt/spark/jars
```

We recommend that you use a Spark base image provided by Alibaba Cloud. Alibaba Cloud provides the Spark 2.4.5 base image, which is optimized for resource scheduling and auto scaling in Kubernetes clusters. This base image accelerates resource scheduling and application startups. You can enable the scheduling optimization feature by setting the `enableAlibabaCloudFeatureGates` variable in the Helm chart to true. If you want to start up the application faster, you can set `enableWebhook` to false.



```yaml
operatorImageName: registry.aliyuncs.com/acs/spark-operator
operatorImageVersion: ack-2.4.5-latest
operatorVersion: v2.4.5-v1beta2
imagePullPolicy: IfNotPresent

rbac:
  create: true

serviceAccounts:
  spark:
    create: true
    name: spark
  sparkoperator:
    create: true
    name: ack-spark-operator

sparkJobNamespace: "default"

enableWebhook: false
enableMetrics: true
enableAlibabaCloudFeatureGates: false
```

Build ImageCache

It may be time-consuming to pull a large Spark image. You can use ImageCache to accelerate image pulling. For more information, see Manage ImageCache and Use ImageCache to accelerate the creation of pods.

Create a Spark job template and submit the job

Create a YAML file for a Spark job and deploy the Spark job.

1. Create a file named spark-pi.yaml file.

   The following code provides a sample template of a typical Spark job. For more information, see spark-on-k8s-operator.

```
apiVersion: "sparkoperator.k8s.io/v1beta2"
kind: SparkApplication
metadata:
  name: spark-pi
  namespace: default
spec:
  type: Scala
  mode: cluster
  image: "registry.aliyuncs.com/acs/spark:ack-2.4.5-latest"
  imagePullPolicy: Always
  mainClass: org.apache.spark.examples.SparkPi
  mainApplicationFile: "local:///opt/spark/examples/jars/spark-examples_2.11-2.4.5.jar"
  sparkVersion: "2.4.5"
  restartPolicy:
    type: Never
  driver:
    cores: 2
    coreLimit: "2"
    memory: "3g"
    memoryOverhead: "1g"
    labels:
      version: 2.4.5
    serviceAccount: spark
    annotations:
      k8s.aliyun.com/eci-kube-proxy-enabled: 'true'
      k8s.aliyun.com/eci-image-cache: "true"
    tolerations:
    - key: "virtual-kubelet.io/provider"
      operator: "Exists"
  executor:
    cores: 2
    instances: 1
    memory: "3g"
    memoryOverhead: "1g"
    labels:
      version: 2.4.5
    annotations:
      k8s.aliyun.com/eci-kube-proxy-enabled: 'true'
      k8s.aliyun.com/eci-image-cache: "true"
    tolerations:
    - key: "virtual-kubelet.io/provider"
      operator: "Exists"
```

2. Deploy a Spark job.

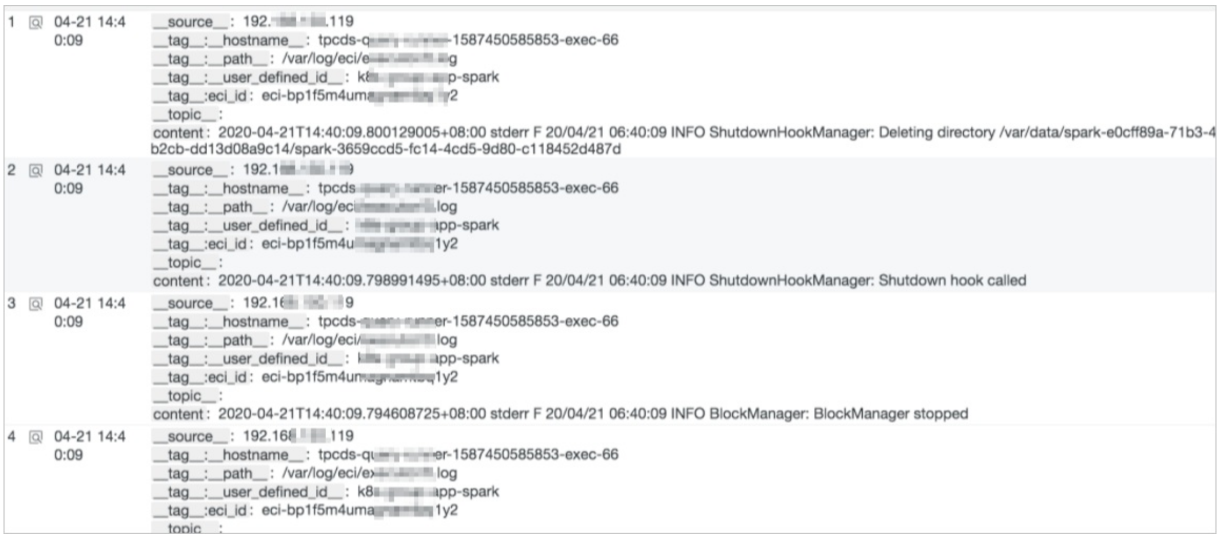
```
kubectl apply -f spark-pi.yaml
```

Configure log collection

If you want to collect the stdout logs of a Spark job, you can configure environment variables in the envVars field of the Spark driver and Spark executor. Then, logs are automatically collected. For more information, see Customize log collection for an elastic container instance.

```
envVars:
   aliyun_logs_test-stdout_project: test-k8s-spark
   aliyun_logs_test-stdout_machinegroup: k8s-group-app-spark
   aliyun_logs_test-stdout: stdout
```

Before you submit a Spark job, you can configure the environment variables of the Spark driver and Spark executor as shown in the preceding code to implement automatic log collection.



```
1  04-21 14:4   __source__: 192.[redacted].119
   0:09         __tag__:__hostname__: tpcds-q[redacted]-1587450585853-exec-66
                __tag__:__path__: /var/log/eci/e[redacted]g
                __tag__:__user_defined_id__: k[redacted]p-spark
                __tag__:eci_id: eci-bp1f5m4uma[redacted]y2
                __topic__:
                content: 2020-04-21T14:40:09.800129005+08:00 stderr F 20/04/21 06:40:09 INFO ShutdownHookManager: Deleting directory /var/data/spark-e0cff89a-71b3-4
                b2cb-dd13d08a9c14/spark-3659ccd5-fc14-4cd5-9d80-c118452d487d
2  04-21 14:4   __source__: 192.1[redacted]9
   0:09         __tag__:__hostname__: tpcds[redacted]er-1587450585853-exec-66
                __tag__:__path__: /var/log/ec[redacted].log
                __tag__:__user_defined_id__: [redacted]app-spark
                __tag__:eci_id: eci-bp1f5m4u[redacted]1y2
                __topic__:
                content: 2020-04-21T14:40:09.798991495+08:00 stderr F 20/04/21 06:40:09 INFO ShutdownHookManager: Shutdown hook called
3  04-21 14:4   __source__: 192.16[redacted]9
   0:09         __tag__:__hostname__: tpcds-[redacted]er-1587450585853-exec-66
                __tag__:__path__: /var/log/eci/[redacted]log
                __tag__:__user_defined_id__: k[redacted]app-spark
                __tag__:eci_id: eci-bp1f5m4u[redacted]1y2
                __topic__:
                content: 2020-04-21T14:40:09.794608725+08:00 stderr F 20/04/21 06:40:09 INFO BlockManager: BlockManager stopped
4  04-21 14:4   __source__: 192.16[redacted]119
   0:09         __tag__:__hostname__: tpcds-q[redacted]er-1587450585853-exec-66
                __tag__:__path__: /var/log/eci/e[redacted]log
                __tag__:__user_defined_id__: k8[redacted]app-spark
                __tag__:eci_id: eci-bp1f5m4uma[redacted]1y2
                __topic__:
```

Configure a history server

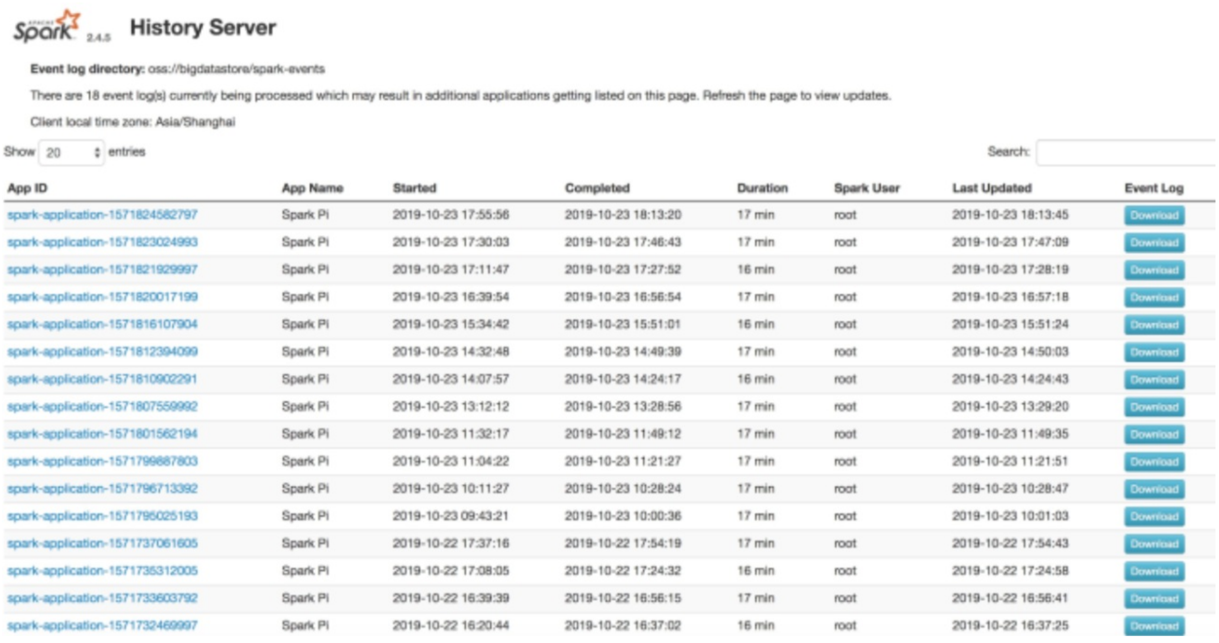
A Spark history server allows you to review Spark jobs. You can set the SparkConf field in the CRD of the Spark application to allow the application to sink events to OSS. Then, you can use the history server to retrieve the data from OSS. The following code shows an example:

```
sparkConf:
   "spark.eventLog.enabled": "true"
   "spark.eventLog.dir": "oss://bigdatastore/spark-events"
   "spark.hadoop.fs.oss.impl": "org.apache.hadoop.fs.aliyun.oss.AliyunOSSFileSystem"
   # oss bucket endpoint such as oss-cn-beijing.aliyuncs.com
   "spark.hadoop.fs.oss.endpoint": "oss-cn-beijing.aliyuncs.com"
   "spark.hadoop.fs.oss.accessKeySecret": ""
   "spark.hadoop.fs.oss.accessKeyId": ""
```

Alibaba Cloud provides a Helm chart for you to deploy Spark history servers. To deploy a Spark history server, log on to the ACK console, choose **Marketplace > Marketplace** in the left-side navigation pane. On the App Catalog tab, search for and deploy ack-spark-history-server. When you deploy the Spark history server, you must specify the OSS information in the **Parameters** section. The following code shows an example:

```
oss:
  enableOSS: true
  # Please input your accessKeyId
  alibabaCloudAccessKeyId: ""
  # Please input your accessKeySecret
  alibabaCloudAccessKeySecret: ""
  # oss bucket endpoint such as oss-cn-beijing.aliyuncs.com
  alibabaCloudOSSEndpoint: "oss-cn-beijing.aliyuncs.com"
  # oss file path such as oss://bucket-name/path
  eventsDir: "oss://bigdatastore/spark-events"
```

After you deploy the Spark history server, you can view its external endpoint on the Services page. You can access the external endpoint of the Spark history server to view historical Spark jobs.



Event log directory: oss://bigdatastore/spark-events

There are 18 event log(s) currently being processed which may result in additional applications getting listed on this page. Refresh the page to view updates.

Client local time zone: Asia/Shanghai

Show 20 entries

Search:

App ID	App Name	Started	Completed	Duration	Spark User	Last Updated	Event Log
spark-application-1571824582797	Spark Pi	2019-10-23 17:55:56	2019-10-23 18:13:20	17 min	root	2019-10-23 18:13:45	Download
spark-application-1571823024993	Spark Pi	2019-10-23 17:30:03	2019-10-23 17:46:43	17 min	root	2019-10-23 17:47:09	Download
spark-application-1571821929997	Spark Pi	2019-10-23 17:11:47	2019-10-23 17:27:52	16 min	root	2019-10-23 17:28:19	Download
spark-application-1571820017199	Spark Pi	2019-10-23 16:39:54	2019-10-23 16:56:54	17 min	root	2019-10-23 16:57:18	Download
spark-application-1571816107904	Spark Pi	2019-10-23 15:34:42	2019-10-23 15:51:01	16 min	root	2019-10-23 15:51:24	Download
spark-application-1571812394099	Spark Pi	2019-10-23 14:32:48	2019-10-23 14:49:39	17 min	root	2019-10-23 14:50:03	Download
spark-application-1571810902291	Spark Pi	2019-10-23 14:07:57	2019-10-23 14:24:17	16 min	root	2019-10-23 14:24:43	Download
spark-application-1571807559992	Spark Pi	2019-10-23 13:12:12	2019-10-23 13:28:56	17 min	root	2019-10-23 13:29:20	Download
spark-application-1571801562194	Spark Pi	2019-10-23 11:32:17	2019-10-23 11:49:12	17 min	root	2019-10-23 11:49:35	Download
spark-application-1571799887803	Spark Pi	2019-10-23 11:04:22	2019-10-23 11:21:27	17 min	root	2019-10-23 11:21:51	Download
spark-application-1571796713392	Spark Pi	2019-10-23 10:11:27	2019-10-23 10:28:24	17 min	root	2019-10-23 10:28:47	Download
spark-application-1571795025193	Spark Pi	2019-10-23 09:43:21	2019-10-23 10:00:36	17 min	root	2019-10-23 10:01:03	Download
spark-application-1571737061605	Spark Pi	2019-10-22 17:37:16	2019-10-22 17:54:19	17 min	root	2019-10-22 17:54:43	Download
spark-application-1571735312005	Spark Pi	2019-10-22 17:08:05	2019-10-22 17:24:32	16 min	root	2019-10-22 17:24:58	Download
spark-application-1571733603792	Spark Pi	2019-10-22 16:39:39	2019-10-22 16:56:15	17 min	root	2019-10-22 16:56:41	Download
spark-application-1571732469997	Spark Pi	2019-10-22 16:20:44	2019-10-22 16:37:02	16 min	root	2019-10-22 16:37:25	Download

View the result of the Spark job

1. Query the status of the pods.

   ```
   kubectl get pods
   ```

   Expected output:

   ```
   NAME                              READY      STATUS     RESTARTS    AGE
   spark-pi-1547981232122-driver     1/1       Running     0           12s
   spark-pi-1547981232122-exec-1     1/1       Running     0           3s
   ```

2. Query the real-time Spark user interface.

   ```
   kubectl port-forward spark-pi-1547981232122-driver 4040:4040
   ```

3. Query the status of the Spark application.

   ```
   kubectl describe sparkapplication spark-pi
   ```

   Expected output:

```
Name:         spark-pi
Namespace:    default
Labels:       <none>
Annotations:  kubectl.kubernetes.io/last-applied-configuration:
                {"apiVersion":"sparkoperator.k8s.io/v1alpha1","kind":"SparkApplication"
,"metadata":{"annotations":{},"name":"spark-pi","namespace":"defaul...
API Version:  sparkoperator.k8s.io/v1alpha1
Kind:         SparkApplication
Metadata:
  Creation Timestamp:  2019-01-20T10:47:08Z
  Generation:          1
  Resource Version:    4923532
  Self Link:           /apis/sparkoperator.k8s.io/v1alpha1/namespaces/default/sparkappl
ications/spark-pi
  UID:                 bbe7445c-1ca0-11e9-9ad4-062fd7c19a7b
Spec:
  Deps:
  Driver:
    Core Limit:  200m
    Cores:       0.1
    Labels:
      Version:        2.4.0
    Memory:           512m
    Service Account:  spark
    Volume Mounts:
      Mount Path:  /tmp
      Name:        test-volume
  Executor:
    Cores:      1
    Instances:  1
    Labels:
      Version:  2.4.0
    Memory:     512m
    Volume Mounts:
      Mount Path:         /tmp
      Name:               test-volume
  Image:                  gcr.io/spark-operator/spark:v2.4.0
  Image Pull Policy:      Always
  Main Application File:  local:///opt/spark/examples/jars/spark-examples_2.11-2.4.0.ja
r
  Main Class:             org.apache.spark.examples.SparkPi
  Mode:                   cluster
  Restart Policy:
    Type:  Never
  Type:    Scala
  Volumes:
    Host Path:
      Path:  /tmp
      Type:  Directory
    Name:    test-volume
Status:
  Application State:
    Error Message:
    State:          COMPLETED
  Driver Info:
    Pod Name:              spark-pi-driver
    Web UI Port:           31182
    Web UI Service Name:   spark-pi-ui-svc
  Execution Attempts:      1
  Executor State:
    Spark - Pi - 1547981232122 - Exec - 1:  COMPLETED
  Last Submission Attempt Time:             2019-01-20T10:47:14Z
  Spark Application Id:                     spark-application-1547981285779
  Submission Attempts:                      1
  Termination Time:                         2019-01-20T10:48:56Z
Events:
  Type    Reason                     Age                 From            Message
  ----    ------                     ----                ----            -------
  Normal  SparkApplicationAdded      55m                 spark-operator  SparkApplicati
on spark-pi was added, Enqueuing it for submission
  Normal  SparkApplicationSubmitted  55m                 spark-operator  SparkApplicati
on spark-pi was submitted successfully
  Normal  SparkDriverPending         55m (x2 over 55m)   spark-operator  Driver spark-p
i-driver is pending
  Normal  SparkExecutorPending       54m (x3 over 54m)   spark-operator  Executor spark
-pi-1547981232122-exec-1 is pending
  Normal  SparkExecutorRunning       53m (x4 over 54m)   spark-operator  Executor spark
-pi-1547981232122-exec-1 is running
  Normal  SparkDriverRunning         53m (x12 over 55m)  spark-operator  Driver spark-p
i-driver is running
  Normal  SparkExecutorCompleted     53m (x2 over 53m)   spark-operator  Executor spark
-pi-1547981232122-exec-1 completed
```

4. View the result of the Spark job in the log.

```
NAME                                       READY     STATUS      RESTARTS    AGE
spark-pi-1547981232122-driver    0/1       Completed    0           1m
```

If the Spark application is in the Succeed state, or the Spark driver pod is in the Completed state, the result of the Spark job is available. You can print the log and view the result of the Spark job in the log.

```
kubectl logs spark-pi-1547981232122-driver
Pi is roughly 3.152155760778804
```

18.7. Install and use WordPress

Cloud Shell

Note You can clone the sample code for this topic to Cloud Shell and learn how to manage an ECI in Cloud Shell. Click the preceding link to open Cloud Shell.

Create a serverless Kubernetes cluster

For more information, see Create a serverless Kubernetes cluster.

Install WordPress

Notice Make sure that the serverless Kubernetes cluster created in the preceding step is initialized before you install WordPress. Typically, the initialization takes about 3 to 5 minutes.

Access the serverless Kubernetes cluster in Cloud Shell.

```
source use-k8s-cluster ${Cluster ID}
```

Install WordPress by using a YAML file.

```
kubectl apply -f wordpress-all-in-one-pod.yaml
```

Check the installation progress. Wait until the status of the pod where WordPress is deployed turns to Running.

```
kubectl get pods
```

Query the Elastic IP Address (EIP) of the pod.

```
kubectl get -o json pod wordpress |grep "k8s.aliyun.com/allocated-eipAddress"
```

By default, a security group does not allow access through port 80. You must enable access through port 80 for the security group automatically created for the serverless Kubernetes cluster.

Find the ID of the recently created security group whose name is prefixed with `alicloud-cs-auto-created`.

```
aliyun ecs DescribeSecurityGroups|grep -B 1 'alicloud-cs-auto-created'|head -1
```

Enable access through port 80 for the security group.

```
aliyun ecs AuthorizeSecurityGroup --RegionId cn-chengdu --SecurityGroupId ${Security group
ID} --IpProtocol tcp --PortRange 80/80 --SourceCidrIp 0.0.0.0/0 --Priority 100
```

Use WordPress

Enter the EIP that is retrieved in the preceding step in the address bar of a browser and press Enter to use WordPress.

18.8. Run jobs on a preemptible instance

You can run jobs or stateless applications on preemptible elastic container instances at reduced costs. This topic describes how to run jobs on a preemptible elastic container instance.

Background information

Before you use a preemptible elastic container instance, take note of the following information:

- The market price of a preemptible instance fluctuates based on the supply and demand for its instance type. When you create a preemptible instance, you must specify a maximum hourly price to bid for a specific instance type. If your bid price is higher than the current market price and the resource inventory of the instance type is sufficient, the requested preemptible instance is created and billed at the current market price.
- A preemptible instance has a protection period of 1 hour after it is created. After the one-hour protection period of a preemptible instance expires, the instance is automatically released if the current market price of the preemptible instance exceeds the specified maximum hourly price or if resources are insufficient. A Kubernetes event occurs 3 minutes before a preemptible instance is released. During these 3 minutes, you can perform specific operations to prevent your services from being interrupted. For example, you can configure to deny inbound traffic to the instance.
- If a preemptible instance is released because the market price exceeds your bid price, the data on the instance is cleared, but the instance information is retained. In addition, the status of the instance changes to Failed and the cause of the failure is BidFailed. We recommend that you use persistent storage services such as cloud disks or Apsara File Storage NAS to store important data.

For more information, see Overview and Create a preemptible instance.

Configuration methods

You can configure annotations in the pod declaration to specify to create a preemptible elastic container instance. Take note of the following items:

- k8s.aliyun.com/eci-spot-strategy: specifies the bidding policy for the preemptible instance. Valid values:
  - SpotAsPriceGo: The current market price is automatically used as the bid price.
  - SpotWithPriceLimit: You must set the maximum hourly price that you are willing to pay for the preemptible instance.
- k8s.aliyun.com/eci-spot-price-limit: The maximum hourly price of the preemptible instance. This value can be accurate to three decimal places. You must specify thk8s.aliyun.com/eci-spot-price-limit when k8s.aliyun.com/eci-spot-strategy is set to SpotWithPriceLimit.

Note

When you create a preemptible elastic container instance, we recommend that you set k8s.aliyun.com/eci-use-specs to specify multiple ECS instance types to ensure that resources are sufficient to create the instance.

Example:

- SpotAsPriceGo

  In this mode, the system places bids based on the current market price.

```
apiVersion: v1
kind: Pod
metadata:
  name: spot-as-price-go
  annotations:
    k8s.aliyun.com/eci-use-specs: ecs.sn1ne.large,2-4Gi
    # If you set the k8s.aliyun.com/eci-spot-strategy annotation to SpotAsPriceGo, you do not need to specify a maximum hourly price. The instance is priced at the market price at the time of purchase.
    k8s.aliyun.com/eci-spot-strategy: SpotAsPriceGo
```

- SpotWithPriceLimit

  In this mode, the system places bids based on the specified maximum hourly price. The following situations may occur:

  - If your bid price is lower than the market price, the preemptible instance enters the Pending state. The system offers a bid price every 5 minutes. When your bid price is equal to or higher than the market price, the preemptible instance is created.
  - If your bid price is higher than or equal to the market price and resources are sufficient, your preemptible instance is created.

```
apiVersion: v1
kind: Pod
metadata:
  name: spot-with-price-limit
  annotations:
    k8s.aliyun.com/eci-use-specs: ecs.sn1ne.large
    k8s.aliyun.com/eci-spot-strategy: SpotWithPriceLimit
    # If you set the k8s.aliyun.com/eci-spot-strategy annotation to SpotWithPriceLimit, you must specify a maximum hourly price. If your bid price is higher than the current pay-as-you-go price of the elastic container instance, the preemptible instance is created and billed on a pay-as-you-go basis.
    k8s.aliyun.com/eci-spot-price-limit: "0.05"
```

Configuration examples

In Kubernetes, jobs are used to batch process short-lived and one-off tasks at a time. The following section provides an example of how to run jobs on a preemptible instance:

1. Prepare the configuration file of a job and name it spot_job.yaml.

```
apiVersion: batch/v1
kind: Job
metadata:
  name: pi
spec:
  template:
    metadata:
      annotations:
        k8s.aliyun.com/eci-use-specs: ecs.t5-c1m2.large,2-4Gi
        k8s.aliyun.com/eci-spot-strategy: SpotAsPriceGo    #The current market price is
automatically used as the bid price.
    spec:
      containers:
      - name: pi
        image: registry-vpc.cn-beijing.aliyuncs.com/ostest/perl
        command: ["perl",  "-Mbignum=bpi", "-wle", "print bpi(2000)"]
      restartPolicy: Never
```

2. Create a job.

```
kubectl create -f spot_job.yaml
```

3. View the running status.

```
kubectl get pod
```

The command output shows that the corresponding pod is created. Example output:

```
NAME        READY    STATUS      RESTARTS    AGE
pi-frmr8    1/1      Running     0           5s
```

A preemptible instance has a protection period of 1 hour after it is created. During this period, the instance is not released even if your bid price is lower than the current market price. This ensures that business can run properly on the instance. After the one-hour protection period of a preemptible instance expires, the instance is automatically released if the current market price of the preemptible instance exceeds the specified maximum hourly price or if resources are insufficient. You are notified of the release in the following ways:

- Event notifications before release

  Elastic Container Instance sends a release event to the list of Kubernetes events 3 minutes before the instance is released. You can run the following commands to view the event:

  - View pod details

    ```
    kubectl describe pod pi-frmr8
    ```

    Information about the release event is displayed in the Events section of the command output. Example output:

    ```
    Events:
      Type     Reason            Age    From          Message
      ----     ------            ----   ----          -------
      Warning  SpotToBeReleased  3m32s  kubelet, eci  Spot ECI will be released in 3 minute
    s
    ```

- View event details

  ```
  kukubectl get events
  ```

  Information about the release event is displayed in the command output. Example output:

  ```
  LAST SEEN   TYPE      REASON              OBJECT         MESSAGE
  3m39s       Warning   SpotToBeReleased    pod/pi-frmr8   Spot ECI will be released in 3
  minutes
  ```

- Status displayed after release

  After a preemptible instance is released, the instance information is still retained. The status of the instance is changed to Failed and the cause of the failure is BidFailed. You can run the following commands to view the instance status:

  - View pod information

    ```
    kubectl get pod
    ```

    The status of the pod is displayed as BidFailed in the command output. Example output:

    ```
    NAME       READY   STATUS      RESTARTS   AGE
    pi-frmr8   1/1     BidFailed   0          3h5m
    ```

  - View pod details

    ```
    kubectl describe pod pi-frmr8
    ```

    The status of the pod is displayed as Failed, and the reason is displayed as BidFailed in the command output. Example output:

    ```
    Status:             Failed
    Reason:             BidFailed
    Message:            The pod is spot instance, and have been released at 2020-04-08T12:3
    6Z
    ```

After the instance is released, Job Controller in Kubernetes creates new instances to run jobs. You can run the `kubectl get pod` command to view the pod. Example output:

```
NAME       READY   STATUS      RESTARTS   AGE
pi-frmr8   1/1     BidFailed   0          4h53m
pi-kp5zx   1/1     Running     0          3h45m
```