

Alibaba Cloud Alibaba Cloud Container Service for Kubernetes Best Practices

Issue: 20190819

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 due to product version upgrades, adjustments, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and the updated versions of this document will be occasionally released through Alibaba Cloud-authorized channels. You shall 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 the document in the context that Alibaba Cloud products and services are provided on an "as is", "with all faults" and "as available" basis. 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 bear any liability for any errors or financial losses incurred by any organizations, companies, or individuals arising from their download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, bear responsibility for any indirect, consequential, exemplary, incidental, 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 content of the Alibaba Cloud website, including but not limited to works, products, images, archives, information, materials, website architecture, website graphic layout, and webpage design, 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 the Alibaba Cloud website, product programs, or content 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 contact Alibaba Cloud directly if you discover any errors in this document.

Generic conventions

Table -1: Style conventions

Style	Description	Example
	This warning information 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.
	This warning information 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 restore business.
	This indicates warning information, supplementary instructions, and other content that the user must understand.	 Notice: Take the necessary precautions to save exported data containing sensitive information.
	This indicates supplemental instructions, best practices, tips, and other content that is good to know for the user.	 Note: You can use Ctrl + A to select all files.
>	Multi-level menu cascade.	Settings > Network > Set network type
Bold	It is used for buttons, menus, page names, and other UI elements.	Click OK.
Courier font	It is used for commands.	Run the <code>cd / d C :/ windows</code> command to enter the Windows system folder.
<i>Italics</i>	It is used for parameters and variables.	<code>bae log list --instanceid Instance_ID</code>
[] or [a b]	It indicates that it is an optional value, and only one item can be selected.	<code>ipconfig [-all -t]</code>

Style	Description	Example
<code>{}</code> or <code>{a b}</code>	It indicates that it is a required value, and only one item can be selected.	<code>swich {stand slave}</code>

Contents

Legal disclaimer.....	I
Generic conventions.....	I
1 Cluster.....	1
1.1 ECS instance selection and cluster configurations.....	1
1.1.1 Select ECS instances.....	1
1.1.2 Recommended Kubernetes cluster configurations to run highly reliable applications.....	3
1.2 Update expired certificates of a Kubernetes cluster.....	9
1.3 Update the Kubernetes cluster certificates that are about to expire.....	10
1.4 Plan Kubernetes CIDR blocks under a VPC.....	18
2 Network.....	21
2.1 Deploy a highly reliable Ingress controller.....	21
3 Storage.....	25
3.1 Use a static cloud disk when creating a stateful service.....	25
3.2 Use a dynamic cloud disk when creating a stateful service.....	30
3.3 Use a StatefulSet service.....	36
3.4 Use a NAS file system when creating a stateful service.....	42
3.5 Use an OSS bucket when creating a stateful service.....	49
4 Release.....	56
4.1 Implement Layer-4 canary release by using Alibaba Cloud Server Load Balancer in a Kubernetes cluster.....	56
4.2 Implement a gray release and a blue/green deployment through Ingress in a Kubernetes cluster.....	60
4.2.1 Gray releases and blue/green deployment.....	60
4.2.2 Gray release limits.....	63
4.2.3 Annotation.....	64
4.2.4 Step 1: Deploy a service.....	67
4.2.5 Step 2: Release the latest version of a service.....	70
4.2.6 Step 3: Remove the earlier version of a service.....	75
4.3 Application.....	78
5 Istio.....	79
5.1 Use Istio to implement intelligent routing in Kubernetes.....	79
5.2 Implement Istio distributed tracing on Kubernetes.....	87
5.3 Integrate Istio with Alibaba Cloud Log Service on Kubernetes.....	93
5.4 Use Istio to deploy application services across Kubernetes and ECS instances.....	102
5.5 Use Istio route rules to control ingress TCP traffic.....	110
5.6 Use the Canary method that uses Istio to deploy a service.....	120

5.7 Use a VirtualService and DestinationRule to complete blue/green and canary deployments.....	138
6 Operation and maintenance.....	142
6.1 Check a Kubernetes cluster to troubleshoot exceptions.....	142
7 Serverless.....	146
7.1 Use a GPU container instance.....	146
7.2 Associate an EIP address with a pod.....	149
8 Auto Scaling.....	153
8.1 Deploy an Ingress application on a virtual node.....	153
9 DevOps.....	158
9.1 Deploy Jenkins in a Kubernetes cluster and perform a pipeline build.....	158
9.2 Use GitLab CI to run a GitLab runner and activate a pipeline in a Kubernetes cluster.....	169
9.3 Deploy Jenkins in a serverless Kubernetes cluster and perform an application pipeline build.....	180
9.4 Use Bamboo to deploy a remote agent and run a build plan.....	188
10 Migrate applications from a Swarm cluster to a Kubernetes cluster.....	201
10.1 Swarm cluster to Kubernetes Cluster features comparison.....	201
10.1.1 Overview.....	201
10.1.2 Basic terms.....	201
10.2 Migration solution overview.....	204

1 Cluster

1.1 ECS instance selection and cluster configurations

1.1.1 Select ECS instances

This topic describes the recommend ECS instances for creating a Kubernetes cluster.

Overall cluster ECS instance selection

Low performance ECS instances have the following disadvantages:

- The Worker nodes that run on low performance ECS instances can use only a limited number of network resources.
- If one container consumes most of the resources provided by a low performance ECS instance, the remaining resources become idle because they are insufficient for operations such as creating new containers or restoring failed containers. If you set multiple low performance ECS instances, an excessive amount of resources will be wasted.

High performance ECS instances have the following advantages:

- Large network bandwidth is available. For applications that require large bandwidth, resource usage is high.
- More container communication occurs within one ECS instance, reducing data transmission over networks.
- Images can be more efficiently pulled. For a cluster that uses high performance ECS instances, it only requires one attempt to pull an image and the pulled image then can be used by multiple containers. By contrast, for a cluster that uses low performance ECS instances, multiple attempts must be made to pull an image. Furthermore, scaling a cluster that uses low performance ECS instances takes much longer to perform.

Select the Master node specification

For Kubernetes clusters created through Alibaba Cloud Container Service, core components such as etcd, kube-apiserver, and kube-controller run on Master nodes. These core components are critical for ensuring cluster stability. Generally, large clusters have higher requirements on the Master node specification.

**Note:**

You can determine your cluster size by considering the following factors: the number of nodes, the number of pods, deployment frequency, and the number of visits. In this topic, only the number of nodes is used to determine the size of a cluster.

To select the Master node specification of a cluster of the standard size, see the following table. However, you can select the lower performance Master nodes for clusters in a test environment. The specifications recommended in the following table are designed to keep Master node loads low.

Number of nodes	Master node specification
1 to 5	4 cores, 8 GiB (We recommend that you do not select 2 cores with 4 GiB.)
6 to 20	4 cores, 16 GiB
21 to 100	8 cores, 32 GiB
100 to 200	16 cores, 64 GiB

Select the Worker node specification

- Determine the number of cores required by the cluster and the allowed core failure ratio.

For example, assume a cluster has 160 cores in total. If the allowed core failure ratio is 10%, you must select at least ten 16-core ECS instances and ensure that the upper limit of the cluster load is $160 \times 90\% = 144$ cores. If the allowed core failure ratio is 20%, you must select at least five 32-core ECS instances and ensure that the upper limit of the cluster load is $160 \times 80\% = 128$ cores. In either of these two cases, if one ECS instance fails, the remaining ECS instances can still support the cluster services.

- Determine the CPU:memory ratio. If you run applications that consume large amount of memory resource, for example, Java applications, we recommend that you select an ECS instance with a CPU:memory ratio of 1:8.

Select the ECS Bare Metal Instance

We recommend that you select an ECS Bare Metal (EBM) Instance in the following two scenarios:

- You cluster requires 1000 cores for daily operation. In this case, you can use about ten or eleven EBM instances to build your cluster because one EBM instance has a minimum of 96 cores.
- You want to quickly scale out a large number of containers. For example, assume that you are prepared for a popular E-commerce product promotion. To handle the expected large amount of traffic, you can add EBM instances to your cluster because a single EBM instance can run multiple containers.

EBM instances provide the following benefits to your cluster:

- Ultra-high network performance. Remote Direct Memory Access (RDMA) technology is used. Furthermore, the Terway plugin is designed for you to get the most from your hardware and provides a container bandwidth higher than 9 Gbit/s across hosts.
- Zero jitter computing performance. EBM instances use chips developed by Alibaba Cloud to replace Hypervisor, meaning virtualization overhead or resource preemption concerns are no longer issues.
- High security. EBM instances use physical level encryption, support the Intel SGX encryption, provide a reliable computing environment, and support blockchain applications.

1.1.2 Recommended Kubernetes cluster configurations to run highly reliable applications

To help you guarantee that your applications stably and reliably run in Kubernetes, this topic describes the recommended Kubernetes cluster configurations.

Set the disk type and size

Select the disk type

- We recommend that you select the SSD disk type.
- For Worker nodes, we recommend that you select the Attach Data Disk check box when you create a cluster. This disk is provided exclusively for the `/var/lib/docker` file to store local images. It is designed to allow the root disk to store a massive number of images. After your cluster has run for a period, many images you no longer require remain stored. To quickly solve this, we recommend that you take the machine offline, rebuild this disk, and then bring the machine back online.

Set the disk size

Kubernetes nodes require a large disk space because the Docker images, system logs, and application logs are stored in the disk. When creating a Kubernetes cluster, you need to consider the number of pods on each node, the log size of each pod, the image size, the temporary data size, and the space required for system reserved values.

We recommend that you reserve a space of 8 GiB for the ECS instance operation system because the operation system requires a disk space of at least 3 GiB.

Kubernetes resource objects then use the remaining disk space.

Whether to build Worker nodes when creating your cluster

When you create a cluster, you can select either of the following Node Type:

- Pay-As-You-Go, indicates that you can build Worker nodes when creating a cluster.
- Subscription, indicates that you can purchase ECS instances as needed and add the instances to your cluster after you create you cluster.

Configure your cluster network settings

- If you want to connect your cluster with services outside Kubernetes, for example, Relational Database Service (RDS), we recommend that you use an existing VPC, rather than create a VPC. This is because VPCs are logically isolated. You can create a VSwitch and add the ECS instances that run Kubernetes to the VSwitch.
- You can select the Terway network plugin or the Flannel network plugin when creating a Kubernetes cluster. For more information, see [#unique_7](#).
- We recommend that you do not set a small CIDR block of the pod network that only supports a minimal number of nodes. The CIDR block setting of the pod network is associated with the Pod Number for Node setting in Advanced Config. For example, if you set the CIDR block of the pod network to X.X.X.X/16, it means that the number of IP addresses assigned to your cluster is 256*256. Additionally, if you set the number of pods on each node to 128, it means that the maximum number of nodes supported by your cluster is 512.

Use multiple zones

Alibaba Cloud supports multiple regions and each region supports multiple zones. Zones are physical areas that have independent power grids and networks within a region. Using multiple zones enables disaster recovery across areas, but increases

network latency. When creating a Kubernetes cluster, you can choose to create a multi-zone cluster. For more information, see [Create a multi-zone Kubernetes cluster](#).

Claim resources for each pod

When you use a Kubernetes cluster, a common problem is that too many pods are scheduled to one node. This scheduling of pods overloads the node, making it unable to provide services.

We recommend that you specify the resource request parameter and the resource limit parameter when configuring a pod in Kubernetes. This recommended configuration enables Kubernetes to select a node with sufficient resources according to the pod resource requirements during the pod deployment. The following example claims that the Nginx pod uses 1-core CPU and 1024 MiB memory, and the pod cannot use more than 2-core CPU or 4096 MiB memory.

```
apiVersion : v1
kind : Pod
metadata :
  name : nginx
spec :
  containers :
  - name : nginx
    image : nginx
    Resources : # Resource claim .
      requests :
        memory : " 1024Mi "
        cpu : " 1000m "
      limits :
        memory : " 4096Mi "
        cpu : " 2000m "
```

Kubernetes uses a static resource scheduling method, which means that instead of using the resources that have been used to calculate the remaining resources on each node, it uses allocated resources. Its calculation method is: $\text{the remaining resources} = \text{the total resources} - \text{the resources that have been allocated}$. If you manually run a resource-consuming program, Kubernetes is not aware of the resources that are being used by the program.

Therefore, you must claim resources for all pods. For the pods that do not have resource claims, after they are scheduled to a node, Kubernetes assumes that the resources used by them on the corresponding node are still available. Therefore, too many pods may be scheduled to this node.

Configure cluster operation and maintenance settings

- Enable Log Service

When creating a cluster, select the Using Log Service check box.

- Configure cluster monitoring

Alibaba Cloud Container Service is integrated with CloudMonitor. By configuring monitoring on nodes, you can implement real-time monitoring. By adding monitoring alarm rules, you can quickly locate the issues that cause abnormal resource usage.

When you create a Kubernetes cluster through Container Service, two application groups are automatically created in CloudMonitor: one for Master nodes and one for Worker nodes. You can add alarm rules under these two groups and these rules apply to all machines in the groups. When subsequent nodes are added to the corresponding group, the alarm rules in the group are automatically applied.

This means that you only need to configure alarm rules for the ECS resources.



Note:

- To monitor ECS instances, you need to set alarm rules for resources such as CPU, memory, and disk. We recommend that you set the `/var/lib/docker` file on an exclusive disk.

Set an application to wait for its dependent application after it starts

Some applications may have some external dependencies. For example, an application may need to read data from a database (DB) or access the interface of another service. However, when the application starts, the DB or the interface may not be available. In traditional manual O&M, if the external dependencies of an application are unavailable when the application starts, the application exits directly. This is known as `failfast`. This strategy is not applicable for Kubernetes, because O&M in Kubernetes is automated and does not require manual intervention. For example, when you deploy an application, you do not need to manually select a node or start the application on the node. If the application fails, Kubernetes

automatically restarts it. Additionally, automatic capacity increase is supported through HPA when large loads occur.

For example, assume that application A depends on application B, and these two applications run on the same node. After the node restarts, application A starts, but application B has not started. In this case, the dependency of application A is unavailable. According to the strategy of failfast, application A exists and will not start even after application B starts. In this case, application A must be started manually.

In Kubernetes, you can set the system to check the dependency of the application during startup, and to implement polling to wait until the dependency is available. This can be implemented through [Init Container](#).

Set the pod restart policy

When a bug in the code or excessive memory consumption causes application processes to fail, the pod in which the processes reside also fails. We recommend that you set a restart policy for the pod so that the pod can automatically restart after failure.

```
apiVersion : v1
kind : Pod
metadata :
  name : tomcat
spec :
  containers :
  - name : tomcat
    image : tomcat
    restartPolicy : OnFailure #
```

Available values of the restart policy parameter are:

- **Always** : indicates to always restart the pod automatically.
- **OnFailure** : indicates to automatically restart the pod when the pod fails (the exiting status of the process is not 0).
- **Never** : indicates to never restart the pod.

Configure the liveness probe and readiness probe

A running pod may not necessarily be able to provide services because processes in the pod may be locked. However, Kubernetes does not automatically restart the pod because the pod is still running. Therefore, you must configure the liveness probe in

each pod to determine whether the pod is alive, and whether it can provide services. Then, Kubernetes restarts the pod when the liveness probe detects any exception.

The readiness probe is used to detect whether the pod is ready to provide services. It takes some time for an application to initialize during startup. During the initialization, the application cannot provide services. The readiness probe can determine when the pod is ready to receive traffic from Ingress or Service. When the pod is faulty, the readiness probe stops new traffic being forwarded to the pod.

```
apiVersion : v1
kind : Pod
metadata :
  name : tomcat
spec :
  containers :
    - name : tomcat
      image : tomcat
      livenessProbe :
        httpGet :
          path : / index . jsp
          port : 8080
        initialDelaySeconds : 3
        periodSeconds : 3
      readinessProbe :
        httpGet :
          path : / index . jsp
          port : 8080
```

Set one process to run in each container

Users who are new to the container technology tend to use containers as virtual machines and put multiple processes into one container, such as monitoring process, log process, sshd process, and even the whole systemd. This causes the following two problems:

- It becomes complex to determine the resource usage of the pod as a whole, and it becomes difficult for the resource limit that you set to take effect.
- If only one process runs in a container, the container engine can detect process failures and it restarts the container upon each process failure. However, if multiple processes are put into a container, the container engine cannot determine the failure of any single process. Therefore, the engine does not restart the container when a single process fails even though the container does not work normally.

If you want to run multiple processes simultaneously, Kubernetes can help you easily implement that. For example, nginx and php-fpm communicate with each other

through a Unix domain socket. You can use a pod that contains two containers, and put the Unix socket into a shared volume of the two containers.

Avoid Single Point of Failure (SPOF)

If an application uses only one ECS instance, the application is unavailable during the period when Kubernetes restarts the instance upon an instance failure. This issue also occurs when you release an updated version of the application. Therefore, we recommend that you do not directly use pods in Kubernetes. Instead, deploy Deployment or StatefulSet applications and set more than two pods for each application.

1.2 Update expired certificates of a Kubernetes cluster

When cluster certificates expire, communication with the cluster API server by using `kubectl` or calling APIs is disabled, and the expired certificates on cluster nodes cannot be updated automatically through template deployment. To update the certificates, you can log on to each cluster node and run the container stating commands, `docker run`.

Update the expired certificates on a Master node

1. Log on to a Master node with the root permission.
2. Run the following command in any directory to update the expired certificates on the Master node:

```
$ docker run -it -- privileged = true - v /:/ alicoud - k8s  
- host -- pid host -- net host \  
registry . cn - hangzhou . aliyuncs . com / acs / cert - rotate :  
v1 . 0 . 0 / renew / upgrade - k8s . sh -- role master
```

3. Repeat the preceding steps on each cluster Master node to update all the expired certificates.

Update the expired certificates on a Worker node

1. Log on to a Master node with the root permission.

2. Run the following command to obtain the cluster rootCA private key:

```
$ cat / etc / kubernetes / pki / ca . key
```

3. Run either of the following commands to obtain the cluster root private key encoded through base64:

- If the cluster rootCA private key has a blank line, run the following command:

```
$ sed ' 1d ' / etc / kubernetes / pki / ca . key | base64 - w
0
```

- If the cluster rootCA private key does not have any blank line, run the following command:

```
$ cat / etc / kubernetes / pki / ca . key | base64 - w 0
```

4. Log on to a Worker node with the root permission.

5. Run the following command in any directory to update the expired certificates on the Worker node.

```
$ docker run - it -- privileged = true - v / : / alicoud - k8s
- host -- pid host -- net host \
registry . cn - hangzhou . aliyuncs . com / acs / cert - rotate
: v1 . 0 . 0 / renew / upgrade - k8s . sh -- role node --
rootkey ${ base64CAKey }
```



Note:

In step 3, you have obtained `${base64CAKey}`, which is the cluster root private key encoded through base64.

6. Repeat the preceding steps on each cluster Worker node to update all the expired certificates.

1.3 Update the Kubernetes cluster certificates that are about to expire

This topic describes how to update the Kubernetes cluster certificates that are about to expire. You can use one of three methods to update the cluster certificates. You can update the cluster certificates in the Container Service console, update all the certificates by running a single command, or update Master and Worker node certificates separately by running different commands.

Prerequisites

- You have created a Kubernetes cluster. For more information, see [#unique_11](#).

- You have connected to the Kubernetes cluster through kubectl. For more information, see [#unique_12](#).

Updates all certificates through the Container Service console

In the Container Service console, click the Update Certificate prompt of the target cluster. For more information, see [#unique_13](#).

Run a command to update all certificates

Log on to a Master node and run the following command:

```
$ curl http://aliacs-k8s-cn-hangzhou.oss-cn-hangzhou.aliyuncs.com/public/cert-update/renew.sh | bash
```

Verify the results

1. Run the following command to view the status of Master nodes and Worker nodes:

```
$ kubectl get nodes
```

```
[root@ ~]# kubectl get nodes
```

NAME	STATUS	ROLES	AGE	VERSION
cn-hangzhou.	Ready	<none>	23d	v1.11.2
cn-hangzhou.	Ready	<none>	23d	v1.11.2
cn-hangzhou.	Ready	master	47d	v1.11.2
cn-hangzhou.	Ready	master	47d	v1.11.2
cn-hangzhou.	Ready	master	47d	v1.11.2
cn-hangzhou.	Ready	<none>	47d	v1.11.2
cn-hangzhou.	Ready	<none>	47d	v1.11.2

```
[root@ ~]#
```

2. Run the following command. When the value of the `SUCCESSFUL` parameter of each Master node is `1`, and the value of the `SUCCESSFUL` parameter of each Worker node meets the number of cluster Worker nodes, all certificates are updated.

```
$ kubectl get job -nkube-system
```

```
[root@ ~]# kubectl get job -nkube-system
```

NAME	DESIRED	SUCCESSFUL	AGE
aliyun-cert-renew-master-1	1	1	6m
aliyun-cert-renew-master-2	1	1	5m
aliyun-cert-renew-master-3	1	1	5m
aliyun-cert-renew-worker	4	4	4m
cert-job-2	1	1	22h
cert-job-3	1	1	22h
cert-job-4	1	1	22h
cert-node-2	4	4	19h

Manually update the certificates of each Master node

1. Copy the following code and paste it into any path to create a `job - master .`

`yaml` file:

```
apiVersion : batch / v1
kind : Job
metadata :
  name : ${ jobname }
  namespace : kube - system
spec :
  backoffLim it : 0
  completion s : 1
  parallelis m : 1
  template :
    spec :
      activeDead lineSecond s : 3600
      affinity :
        nodeAffini ty :
          requiredDu ringSchedu lingIgnore dDuringExe cution :
            nodeSelect orTerms :
              - matchExpre ssions :
                - key : kubernetes . io / hostname
                  operator : In
                  values :
                    - ${ hostname }
      containers :
        - command :
            - / renew / upgrade - k8s . sh
            - -- role
            - master
          image : registry . cn - hangzhou . aliyuncs . com / acs /
cert - rotate : v1 . 0 . 0
          imagePullP olicy : Always
          name : ${ jobname }
          securityCo ntext :
            privileged : true
            volumeMoun ts :
              - mountPath : / alicoud - k8s - host
                name : ${ jobname }
          hostNetwor k : true
          hostPID : true
          restartPol icy : Never
          schedulerN ame : default - scheduler
          securityCo ntext : {}
          toleration s :
            - effect : NoSchedule
              key : node - role . kubernetes . io / master
          volumes :
            - hostPath :
                path : /
                type : Directory
```

```
name : ${ jobname }
```

2. Obtain the number of Master nodes in the cluster and the `hostname` of each Master node.

• Method 1

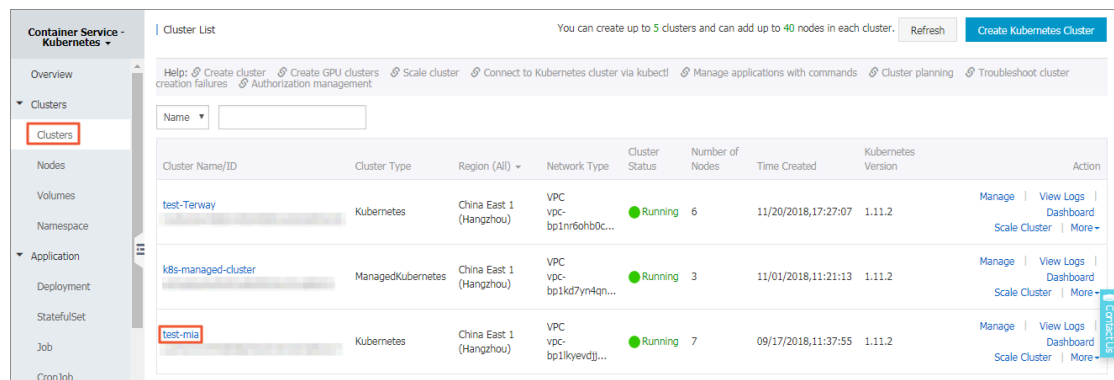
Run the following commands:

```
$ kubectl get nodes
```

```
[root@ ~]# kubectl get nodes
NAME                                STATUS    ROLES    AGE    VERSION
cn-hangzhou.i                      Ready    <none>   22d    v1.11.2
cn-hangzhou.i                      Ready    <none>   22d    v1.11.2
cn-hangzhou.i                      Ready    master   46d    v1.11.2
cn-hangzhou.i                      Ready    master   46d    v1.11.2
cn-hangzhou.i                      Ready    master   46d    v1.11.2
cn-hangzhou.i                      Ready    <none>   46d    v1.11.2
cn-hangzhou.i                      Ready    <none>   46d    v1.11.2
[root@ ~]#
```

• Method 2

- Log on to the [Container Service console](#).
- In the left-side navigation pane under Container Service-Kubernetes, choose **Clusters > Clusters**.



- Click the target cluster name, and then click **Node List** in the left-side navigation pane to view the number of Master nodes and the `hostname` of each Master node.

IP Address	Role	Instance ID/Name	Configuration	Pods(Allocated)	CPU(Request Limit)	Memory(Request Limit)	Update Time	Action
	Worker		Pay-As-You-Go ecs.n1.large	25	13.68% 9.10 %	12.60% 81.58 %	09/17/2018,11:49:00	Scheduling Settings Monitor More
	Master		Pay-As-You-Go ecs.n1.large	8	24.05% 7.60 %	5.11% 90.27 %	09/17/2018,11:39:00	Scheduling Settings Monitor More
	Master		Pay-As-You-Go ecs.n1.large	13	29.05% 10.40 %	6.90% 89.03 %	09/17/2018,11:38:00	Scheduling Settings Monitor More
	Master		Pay-As-You-Go ecs.n1.large	9	25.30% 6.53 %	6.39% 89.57 %	09/17/2018,11:41:00	Scheduling Settings Monitor More
	Worker		Pay-As-You-Go ecs.n1.large	27	17.40% 11.30 %	23.73% 85.77 %	09/17/2018,11:49:00	Scheduling Settings Monitor More
	Worker		Pay-As-You-Go ecs.e3.medium	17	46.10% 16.15 %	21.24% 88.23 %	10/11/2018,09:48:00	Scheduling Settings Monitor More
	Worker		Pay-As-You-Go ecs.gn5i-c2g1.large	11	68.10% 6.15 %	12.22% 86.74 %	10/11/2018,09:42:00	Scheduling Settings Monitor More

3. Run the following command to specify the `${ jobname }` and `${ hostname }` variables in the `job - master . yml` file:

```
$ sed 's /${ jobname }/ cert - job - 2 / g ; s /${ hostname }/ hostname / g' job - master . yml > job - master2 . yml
```

In this code line:

- `${ jobname }` is the Job and pod name. In this example, this variable is set to `cert - job - 2`.
- `${ hostname }` is the Master name. In this example, `hostname` is set to a Master name obtained in step 2.

4. Run the following command to create a Job:

```
$ kubectl create -f job - master2 . yml
```

5. Run the following command to view the Job status. When the value of the `SUCCESSFUL` parameter is `1`, the certificates of this Master node have been updated.

```
$ kubectl get job - nkube - system
```

6. Repeat step 3 to step 5 to update the certificates of the remaining Master nodes in the cluster.

```
[root@ ~]# kubectl get job -nkube-system
NAME          DESIRED  SUCCESSFUL  AGE
cert-job-2    1        1           22m
cert-job-3    1        1           2m
cert-job-4    1        1           1m
[root@ ~]#
```

Manually update Worker node certificates

1. Copy the following code and paste it into any path to create a `job - node . yml` file:

```
apiVersion : batch / v1
kind : Job
metadata :
  name : ${ jobname }
  namespace : kube - system
spec :
  backoffLimit : 0
  completionTime : ${ nodesize }
  parallelism : ${ nodesize }
  template :
    spec :
      activeDeadlineSeconds : 3600
      affinity :
        podAntiAffinity :
          requiredDuringSchedulingIgnoredDuringExecution :
            - labelSelector :
                matchExpressions :
                  - key : job - name
                    operator : In
                    values :
                      - ${ jobname }
              topologyKey : kubernetes . io / hostname
      containers :
        - command :
            - / renew / upgrade - k8s . sh
            - -- role
            - node
            - -- rootkey
            - ${ key }
          image : registry . cn - hangzhou . aliyuncs . com / acs /
cert - rotate : v1 . 0 . 0
          imagePullPolicy : Always
          name : ${ jobname }
          securityContext :
            privileged : true
          volumeMounts :
            - mountPath : / alicoud - k8s - host
              name : ${ jobname }
      hostNetwork : true
      hostPID : true
      restartPolicy : Never
      schedulerName : default - scheduler
      securityContext : {}
      volumes :
        - hostPath :
            path : /
            type : Directory
            name : ${ jobname }
```



Note:

If a Worker node has a taint, you need to add `tolerations` for the taint in the `job - node . yml` file. More specifically, you need to add the following code

between `securityContext : {}` and `volumes :` (If the number of Worker nodes that have taints is `n`, you need to add the following code `n` times):

```
toleration s :
- effect : NoSchedule
  key : ${ key }
  operator : Equal
  value : ${ value }
```

The method to obtain `${ name }` and `${ value }` is as follows:

a. Copy the following code and paste it into any path to create a `taint . tml` file:

```
{{ printf "%- 50s %- 12s \n " " Node " " Taint " }}
{{- range . items }}
  {{- if $ taint := ( index . spec " taints " ) }}
    {{- . metadata . name }}{{ "\ t " }}
    {{- range $ taint }}
      {{- . key }}={{ . value }}:{{ . effect }}{{ "\ t " }}
    {{- end }}
  {{- end }}
{{- end }}
```

b. Run the following command to view the values of `${ name }` and `${ value }` for the Worker nodes that have taints:

```
$ kubectl get nodes -o go-template-file="taint . tml "
```

```
[root@ ~]# kubectl get nodes -o go-template-file="taint.tml"
Node                               Taint
cn-hangzhou.i-                     key1=value1:NoSchedule
cn-hangzhou.i-                     node-role.kubernetes.io/master=<no value>:NoSchedule
cn-hangzhou.i-                     node-role.kubernetes.io/master=<no value>:NoSchedule
cn-hangzhou.i-                     node-role.kubernetes.io/master=<no value>:NoSchedule
```

2. Run the following command to obtain the cluster CAKey:

```
$ sed '1d' /etc/kubernetes/pki/ca.key | base64 -w 0
```

3. Run the following command to specify the `${ jobname }`, `${ nodesize }`, and `${ key }` variables in the `job - node . yml` file:

```
$ sed 's /${ jobname }/ cert - node - 2 / g ; s /${ nodesize }/ nodesize / g ; s /${ key }/ key / g ' job - node . yml > job - node2 . yml
```

In this code line:

- `${ jobname }` is the Job and pod name. In this example, this variable is set to `cert - node - 2`.
- `${ nodesize }` is the number of Worker nodes. For how to obtain this value, see step 2 in [Manually update the certificates of each Master node](#). In this example, the `nodesize` variable is replaced with the number of the Worker nodes in the cluster.
- `${ key }` is the cluster CAKey. In this example, the `key` variable is replaced with the CAKey obtained in step 3 of [#unique_14/unique_14_Connect_42_section_k4z_skb_rfb](#).

4. Run the following command to create a Job:

```
$ kubectl create -f job - node2 . yml
```

5. Run the following command to view the Job status. When the value of the `SUCCESSFUL` parameter is equal to the number of the cluster Worker nodes, all certificates have been updated.

```
$ kubectl get job - nkube - system
```

```
[root@ ~]# kubectl get job -nkube-system
NAME           DESIRED  SUCCESSFUL  AGE
cert-job-2     1        1           1h
cert-job-3     1        1           47m
cert-job-4     1        1           46m
cert-node-2    4        4           1m
[root@ ~]#
```

1.4 Plan Kubernetes CIDR blocks under a VPC

Generally, you can select to create a Virtual Private Cloud (VPC) automatically and use the default network address when creating a Kubernetes cluster in Alibaba Cloud. In some complicated scenarios, plan the Elastic Compute Service (ECS) address, Kubernetes pod address, and Kubernetes service address on your own. This document introduces what the addresses in Kubernetes under Alibaba Cloud VPC environment are used for and how to plan the CIDR blocks.

Basic concepts of Kubernetes CIDR block

The concepts related to IP address are as follows:

VPC CIDR block

The CIDR block selected when you create a VPC. Select the VPC CIDR block from 10.0.0.0/8, 172.16.0.0/12, and 192.168.0.0/16.

VSwitch CIDR Block

The CIDR block specified when you create a VSwitch in VPC. The VSwitch CIDR block must be the subset of the current VPC CIDR block, which can be the same as the VPC CIDR block but cannot go beyond that range. The address assigned to the ECS instance under the VSwitch is obtained from the VSwitch CIDR block. Multiple VSwitches can be created under one VPC, but the VSwitch CIDR blocks cannot overlap .

The VPC CIDR block structure is as follows.

Pod CIDR block

Pod is a concept in Kubernetes. Each pod has one IP address. You can specify the pod CIDR block when creating a Kubernetes cluster in Alibaba Cloud Container Service and the pod CIDR block cannot overlap with the VPC CIDR block. For example, if the VPC CIDR block is 172.16.0.0/12, then the pod CIDR block of Kubernetes cannot use 172.16.0.0/16, 172.17.0.0/16, or any address that is included in 172.16.0.0/12.

Service CIDR block

Service is a concept in Kubernetes. Each service has its own address. The service CIDR block cannot overlap with the VPC CIDR block or pod CIDR block. The service

address is only used in a Kubernetes cluster and cannot be used outside a Kubernetes cluster.

The relationship between Kubernetes CIDR block and VPC CIDR block is as follows.

How to select CIDR block

Scenario of one VPC and one Kubernetes cluster

This is the simplest scenario. The VPC address is determined when the VPC is created. Select a CIDR block different from that of the current VPC when creating a Kubernetes cluster.

Scenario of one VPC and multiple Kubernetes clusters

Create multiple Kubernetes clusters under one VPC. In the default network mode (Flannel), the pod message needs to be routed by using VPC, and Container Service automatically configures the route table to each pod CIDR block on the VPC route. The pod CIDR blocks of all the Kubernetes clusters cannot overlap, but the service CIDR blocks can overlap.

The VPC address is determined when the VPC is created. Select a CIDR block that does not overlap with the VPC address or other pod CIDR blocks for each Kubernetes cluster when creating a Kubernetes cluster.

In such a situation, parts of the Kubernetes clusters are interconnected. The pod of one Kubernetes cluster can directly access the pod and ECS instance of another Kubernetes cluster, but cannot access the

Scenario of VPC interconnection

You can configure what messages are to be sent to the opposite VPC by using route tables when two VPCs are interconnected. Take the following scenario as an example: VPC 1 uses the CIDR block 192.168.0.0/16 and VPC 2 uses the CIDR block 172.16.0.0/12. By using route tables, specify to send the messages of 172.16.0.0/12 in VPC 1 to VPC 2.

In such a situation, the CIDR block of the Kubernetes cluster created in VPC 1 cannot overlap with VPC 1 CIDR block or the CIDR block to be routed to VPC 2. applies to the scenario when you create a Kubernetes cluster in VPC 2. In this example, the pod CIDR block of the Kubernetes cluster can select a sub-segment under 10.0.0.0/8.

**Note:**

The CIDR block routing to VPC 2 can be considered as an occupied address. Kubernetes clusters cannot overlap with an occupied address.

To access the Kubernetes pod of VPC 1 in VPC 2, configure the route to the Kubernetes cluster in VPC 2.

Scenario of VPC to IDC

Similar to the scenario of VPC interconnection, if parts of the CIDR blocks in VPC route to IDC, the pod address of Kubernetes clusters cannot overlap with those addresses. pod address of Kubernetes clusters in IDC, configure the route table to leased line virtual border router (VBR) in IDC.

2 Network

2.1 Deploy a highly reliable Ingress controller

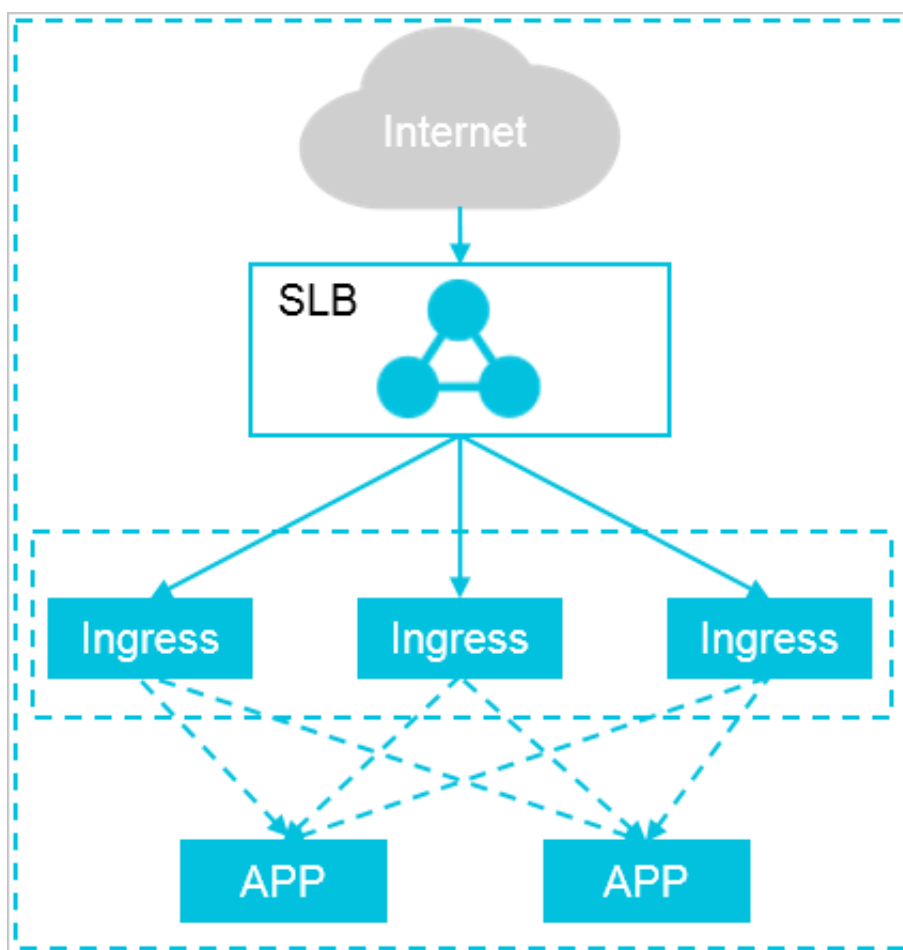
Ingress is a set of rules that authorize external access to the services in a Kubernetes cluster, providing Layer-7 Server Load Balancer capabilities. You can configure Ingress to provide externally accessible URLs, SLB, SSL, and name-based virtual hosts. Ingress requires high reliability because Ingress functions as the access layer through which external traffic goes into a cluster. This topic describes how to deploy a high-performance, highly reliable Ingress access layer.

Prerequisites

- You have created a Kubernetes cluster. For more information, see [#unique_11](#).
- You have connected to the Master node by using SSH. For more information, see [#unique_18](#).

Highly reliable deployment architecture

To achieve high reliability, you must first resolve any SPOFs. Deploying multiple replicas is the general solution for this problem. Specifically, you can use the multi-node deployment architecture to deploy a highly reliable Ingress access layer in a Kubernetes cluster. We also recommend that you configure exclusive Ingress nodes to prevent service applications from competing for resources with the Ingress service because Ingress functions as the traffic access port of a cluster.



As shown in the preceding figure, multiple exclusive Ingress instances constitute an access layer that processes the inbound traffic to the cluster. Furthermore, the number of Ingress nodes can be scaled according to the traffic amount required by the backend services. If your cluster is of a moderate size, you can also deploy the Ingress service and other service applications in a hybrid way. However, we recommend that you limit the number of resources and isolate them for the Ingress and corresponding applications.

View the cluster pod replicas deployed by default and the Internet SLB address

After you create a cluster, a set of Nginx Ingress controller services that have two pod replicas are deployed within the cluster by default. The frontend of this set of services is mounted to an Internet SLB instance.

Run the following command to view the pods on which the Nginx Ingress controller services are deployed:

```
$ kubectl -n kube-system get pod | grep nginx-ingress-controller
nginx-ingress-controller-8648ddc696-2bshk
1 / 1      Running    0          3h
```

```
nginx - ingress - controller - 8648ddc696 - jvbs9
1 / 1      Running      0      3h
```

Run the following command to view the Internet SLB address corresponding to the nginx-ingress-lb service:

```
$ kubectl -n kube-system get svc nginx-ingress-lb
NAME                                TYPE                CLUSTER-IP      EXTERNAL-IP      PORT(S)          AGE
nginx-ingress-lb                    LoadBalancer        172.17.0.118     .xxx.xxx.xx      80:32457/TCP,443:31370/TCP 21d
```

To guarantee the high performance and availability of the cluster access layer for a growing cluster, you need to expand the Ingress access layer. You can use either of the following two methods:

Method 1: Expand the number of replicas

You can quickly scale the Ingress access layer by changing the number of the replicas of the Nginx Ingress controller deployment.

Run the following command to scale out the number of pod replicas to three:

```
$ kubectl -n kube-system scale --replicas=3 deployment/nginx-ingress-controller
deployment.extensions/nginx-ingress-controller scaled
```

Run the following command to view the pods on which the Nginx Ingress controller services are deployed:

```
$ kubectl -n kube-system get pod | grep nginx-ingress-controller
nginx-ingress-controller-8648ddc696-2bshk
1 / 1      Running      0      3h
nginx-ingress-controller-8648ddc696-jvbs9
1 / 1      Running      0      3h
nginx-ingress-controller-8648ddc696-xqmf9
1 / 1      Running      0      33s
```

Method 2: Deploy the Ingress service on a specified node

If you want the Nginx Ingress controller to run on target nodes of advanced configurations only, you can label the target nodes.

1. Run the following command to view the cluster nodes:

```
$ kubectl get node
NAME                                STATUS    ROLES    AGE
cn-hangzhou-i-bp11bcmsna-8d4bpf17bc  Ready    master   21d
cn-hangzhou-i-bp12h6biv9-bg24lmdc2o    Ready    <none>   21d
```

```

cn - hangzhou . i - bp12h6biv9  bg24lmdc2p    Ready    < none >
21d      v1 . 11 . 5
cn - hangzhou . i - bp12h6biv9  bg24lmdc2q    Ready    < none >
21d      v1 . 11 . 5
cn - hangzhou . i - bp181pofzy  yksie2ow03    Ready    master
21d      v1 . 11 . 5
cn - hangzhou . i - bp1cbsg6rf  3580z6uyo7    Ready    master
21d      v1 . 11 . 5

```

2. Run the following commands to add the label `node - role . kubernetes . io`

```

/ ingress =" true " to the Ingress node cn - hangzhou . i - bp12h6biv9
bg24lmdc2o and the Ingress node cn - hangzhou . i - bp12h6biv9
bg24lmdc2p :

```



Note:

- The number of the labeled nodes must be greater than or equal to the number of the cluster pod replicas so that multiple pods do not run on one node.
- We recommend that you label Worker nodes only to deploy the Ingress service.

```

$ kubectl label nodes cn - hangzhou . i - bp12h6biv9
bg24lmdc2o node - role . kubernetes . io / ingress =" true "
node / cn - hangzhou . i - bp12h6biv9 bg24lmdc2o labeled

```

```

$ kubectl label nodes cn - hangzhou . i - bp12h6biv9
bg24lmdc2p node - role . kubernetes . io / ingress =" true "
node / cn - hangzhou . i - bp12h6biv9 bg24lmdc2p labeled

```

3. Run the following command to update your deployment and add the nodeSelector setting:

```

$ kubectl -n kube-system patch deployment nginx -
ingress - controller -p '{"spec": {"template": {"spec": {"
nodeSelect or ": {"node - role . kubernetes . io / ingress ": "
true }}}}}'
deployment.extensions/nginx-ingress-controller patched

```

Result:

Run the following command to verify that the Ingress pods are deployed on the cluster nodes that are labeled by `node - role . kubernetes . io / ingress ="`

`true ":`

```

$ kubectl -n kube-system get pod -o wide | grep
nginx-ingress-controller
nginx-ingress-controller-8648ddc696-2bshk
1 / 1 Running 0 3h 172.16.2.15
cn - hangzhou . i - bp12h6biv9 bg24lmdc2p < none >
nginx-ingress-controller-8648ddc696-jvbs9
1 / 1 Running 0 3h 172.16.2.145
cn - hangzhou . i - bp12h6biv9 bg24lmdc2o < none >

```

3 Storage

3.1 Use a static cloud disk when creating a stateful service

This topic describes typical scenarios in which a static cloud disk is needed for creating a stateful service, and the procedure for how to use one.

Scenarios and method

Scenarios for using cloud disks:

- You want to create applications that demand high disk I/O performance and do not require shared data. For example, MySQL, Redis, and other data storage services.
- You want logs to be written at high speed.
- You want your stored data to exist persistently. That is, the data still exist when the life cycle of the pod ends.

Scenario for using static cloud disks:

You have purchased a cloud disk.

Method of using static cloud disks:

Manually create a Persistent Volume (PV) and a Persistent Volume Claim (PVC).

Prerequisites

- You have created a Kubernetes cluster. For more information, see [#unique_11](#).
- You have created a cloud disk. For more information, see [#unique_21](#).
- You have connected to the Kubernetes cluster by using kubectl, see [#unique_12](#).

Limits

- Cloud disks are the non-shared storage devices provided by the Alibaba Cloud Storage Team. Each cloud disk can be mounted to only one pod.
- In a Kubernetes cluster, a cloud disk can be mounted only to a node that resides in the same zone as the cloud disk.

Create a PV

1. Create a `pv - static . yaml` file.

```
apiVersion : v1
kind : Persistent Volume
```

```

metadata :
  name : < your - disk - id >
  labels :
    alicloud - pvname : < your - disk - id >
    failure - domain . beta . kubernetes . io / zone : < your -
zone >
    failure - domain . beta . kubernetes . io / region : < your -
region >
spec :
  capacity :
    storage : 20Gi
  accessMode s :
    - ReadWrite0 nce
  flexVolume :
    driver : " alicloud / disk "
    fsType : " ext4 "
    options :
      volumeId : "< your - disk - id >"

```

**Note:**

- `alicloud - pvname : < your - disk - id >`: indicates the PV name. This parameter must be set to the same value as that of the `volumeID` parameter, namely, the cloud disk ID.
- `failure - domain . beta . kubernetes . io / zone : < your - zone >`: indicates the zone in which the cloud disk resides. For example, `cn - hangzhou - b`.
- `failure - domain . beta . kubernetes . io / region : < your - region >`: indicates the region in which the cloud disk resides. For example, `cn - hangzhou`.

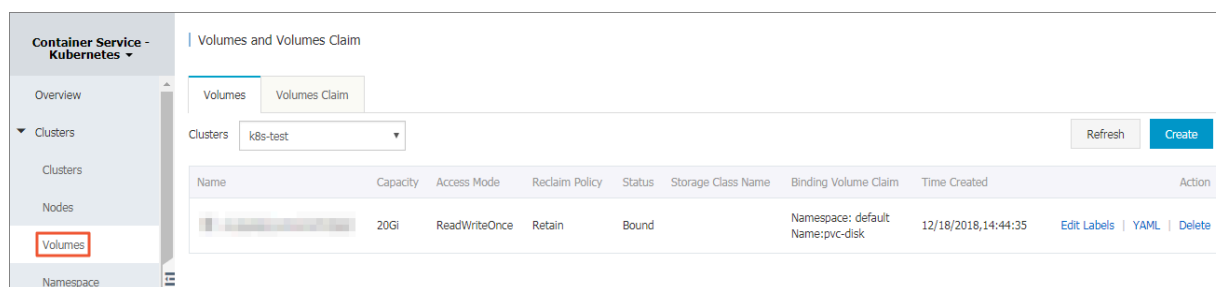
If you use a Kubernetes cluster that has multiple zones, you must set the `failure - domain . beta . kubernetes . io / zone` parameter and the `failure - domain . beta . kubernetes . io / region` parameter so that you can guarantee that your pod can be scheduled to the zone in which the cloud disk resides.

2. Run the following command to create a PV:

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

Result

In the left-side navigation pane under Kubernetes, choose Clusters > Volumes, and select the target cluster to see the created PV.



Create a PVC

Create a PVC for the cloud disk. Specifically, you need to set the `selector` field to filter for the created PV so that you can associate the PVC with the correct PV.

1. Create a `pvc - static . yaml` file.

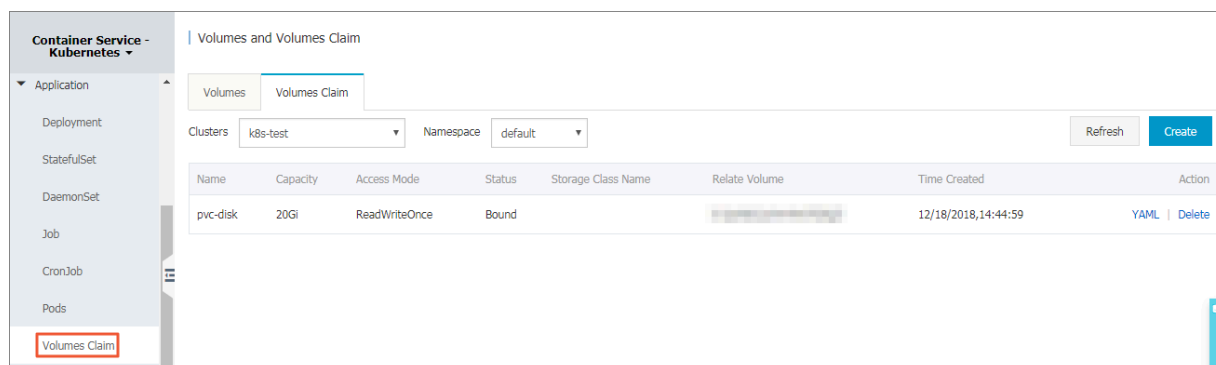
```
kind : Persistent VolumeClaim
apiVersion : v1
metadata :
  name : pvc - disk
spec :
  accessModes :
    - ReadWriteOnce
  resources :
    requests :
      storage : 20Gi
  selector :
    matchLabels :
      alicloud - pvname : < your - disk - id >
```

2. Run the following command to create a PVC:

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

Result

In the left-side navigation pane under Kubernetes, choose **Application > Volumes Claim**, and select the target cluster and namespace to see the created PVC.



Create an application

1. Create a `static.yaml` file.

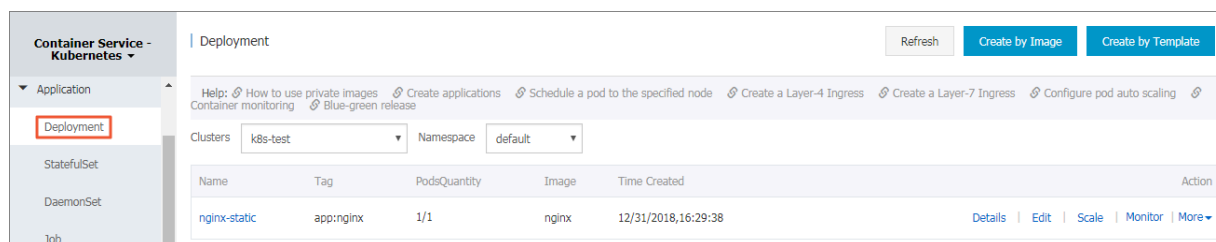
```
apiVersion : apps / v1
kind : Deployment
metadata :
  name : nginx - static
  labels :
    app : nginx
spec :
  selector :
    matchLabels :
      app : nginx
  template :
    metadata :
      labels :
        app : nginx
    spec :
      containers :
        - name : nginx
          image : nginx
          volumeMounts :
            - name : disk - pvc
              mountPath : "/ data "
      volumes :
        - name : disk - pvc
          persistentVolumeClaim :
            claimName : pvc - disk
```

2. Run the following command to create a deployment:

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

Result

In the left-side navigation pane under Kubernetes, choose **Application > Deployment**, and select the target cluster and namespace to see the created deployment.



Persistent data storage on the static cloud disk

1. Run the following command to view the pod in which the created deployment resides:

```
$ kubectl get pod | grep static
```

```
nginx - static - 78c7dcb9d7 - g9lll    2 / 2    Running    0
32s
```

2. Run the following command to check whether the new cloud disk is mounted to the `/ data` path:

```
$ kubectl exec nginx - static - 78c7dcb9d7 - g9lll df |
grep data
/ dev / vdf          20511312      45080      20449848      1 % / data
```

3. Run the following command to view the file in the `/ data` path:

```
$ kubectl exec nginx - static - 78c7dcb9d7 - g9lll ls /
data
lost + found
```

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

```
$ kubectl exec nginx - static - 78c7dcb9d7 - g9lll touch /
data / static
```

5. Run the following command to view the files in the `/ data` path:

```
$ kubectl exec nginx - static - 78c7dcb9d7 - g9lll ls /
data
static
lost + found
```

6. Run the following command to remove the pod named `nginx - static -`

```
78c7dcb9d7 - g9lll :
```

```
$ kubectl delete pod nginx - static - 78c7dcb9d7 - g9lll
pod " nginx - static - 78c7dcb9d7 - g9lll " deleted
```

7. Open another kubectl interface and run the following command to view the process in which the preceding pod is removed and a new pod is created by Kubernetes:

```
$ kubectl get pod - w - l app = nginx
NAME                                READY    STATUS    RESTARTS
AGE
nginx - static - 78c7dcb9d7 - g9lll  2 / 2    Running    0
50s
nginx - static - 78c7dcb9d7 - g9lll  2 / 2    Terminat  g
0 72s
nginx - static - 78c7dcb9d7 - h6brd   0 / 2    Pending    0
0s
nginx - static - 78c7dcb9d7 - h6brd   0 / 2    Pending    0
0s
nginx - static - 78c7dcb9d7 - h6brd   0 / 2    Init : 0 / 1
0 0s
nginx - static - 78c7dcb9d7 - g9lll   0 / 2    Terminat  g
0 73s
nginx - static - 78c7dcb9d7 - h6brd   0 / 2    Init : 0 / 1
0 5s
```

```

nginx - static - 78c7dcb9d7 - g9lll    0 / 2    Terminatin g
0      78s
nginx - static - 78c7dcb9d7 - g9lll    0 / 2    Terminatin g
0      78s
nginx - static - 78c7dcb9d7 - h6brd    0 / 2    PodInitial izing
0      6s
nginx - static - 78c7dcb9d7 - h6brd    2 / 2    Running      0
8sg    0      8s

```

8. Run the following command to view the new pod created by Kubernetes:

```

$ kubectl get pod
NAME                                READY   STATUS    RESTARTS
AGE
nginx - static - 78c7dcb9d7 - h6brd 2 / 2   Running   0
14s

```

9. Run the following command to verify that the created file named `static` in the `/data` path has not been removed, indicating that data in the static cloud disk can be stored persistently:

```

$ kubectl exec nginx - static - 78c7dcb9d7 - h6brd ls /
data
static
lost + found

```

3.2 Use a dynamic cloud disk when creating a stateful service

This topic describes typical scenarios in which a dynamic cloud disk is needed for creating a stateful service, and the procedure for how to use one.

Scenarios and method

Scenario for using dynamic cloud disks:

You want to configure the system to automatically purchase cloud disks when you deploy an application, rather than manually purchase cloud disks before deploying the application.

Method of using a dynamic cloud disk:

1. Manually create a PVC and claim a specific StorageClass in the PVC.
2. Use the StorageClass to enable the system to automatically create a PV when you deploy an application.

Prerequisites

- You have created a Kubernetes cluster. For more information, see [#unique_11](#).
- You have connected to the Kubernetes cluster by using kubectl, see [#unique_12](#).

- You have installed the provisioner plugin in the Kubernetes cluster. The plugin automatically creates a cloud disk according to a specific StorageClass.

Provisioner plugin

When you create a cluster through Alibaba Cloud Container Service for Kubernetes, the provisioner plugin is installed in the cluster by default.

Create a StorageClass

By default, Alibaba Cloud Container Service for Kubernetes creates four StorageClasses for a cluster during the cluster initialization, and the StorageClasses use the default settings. Furthermore, the four default StorageClasses are created only for a cluster that has a single zone. For a cluster that has multiple zones, you need to manually create a StorageClass. The following are the four StorageClasses created by default:

- `alicloud - disk - common` indicates to automatically create a basic cloud disk.
- `alicloud - disk - efficiency` indicates to automatically create an Ultra cloud disk.
- `alicloud - disk - ssd` indicates to automatically create an SSD cloud disk.
- `alicloud - disk - available` indicates a systematic method of disk selection.

Specifically, the system first attempts to create an Ultra cloud disk. If the Ultra cloud disks in the specified zone are sold out, the system tries to create an SSD cloud disk. If the SSD cloud disks are sold out, the system tries to create a basic cloud disk.

1. Create a `storageclass.yaml` file.

```
kind : StorageClass
apiVersion : storage.k8s.io / v1beta1
metadata :
  name : alicloud - disk - ssd - hangzhou - b
provisioner : alicloud / disk
reclaimPolicy : Retain
parameters :
  type : cloud_ssd
  regionid : cn - hangzhou
  zoneid : cn - hangzhou - b
  fstype : " ext4 "
```

```
readonly : " false "
```

Parameter setting

- `provisioner` : Set this parameter to `alicloud/disk` to specify that the StorageClass creates an Alibaba Cloud cloud disk by using the provisioner plugin.
- `reclaimPolicy` : Set a policy to reclaim the cloud disk. Available values of this parameter are `Delete` and `Retain` . The default setting is `Delete` .



Note:

If you maintain the default setting, namely, `Delete` , the data on the cloud disk cannot be restored after you remove the PVC because the cloud disk is also removed.

- `type` : Specify a cloud disk type by using one the following values: `cloud` , `cloud_efficiency` , `cloud_ssd` , and `available` .
- `regionid` : (optional) Set the region in which the cloud disk is automatically created. This region must be the same as the region in which your cluster resides.
- `zoneid` : (optional) Set the zone in which a cloud disk is automatically created.
 - If you set this parameter for a single-zone cluster, the value must be the same as the zone in which the cluster resides.
 - If you set this parameter for a multi-zone cluster, multiple values can be set. For example,

```
zoneid : cn - hangzhou - a , cn - hangzhou - b , cn -
hangzhou - c
```

- `fstype` : (optional) Set the type of the file system used for automatic cloud disk creation. The default setting is `ext4` .
- `readonly` : (optional) Set whether the automatically created cloud disk is read only. If you set this parameter to `true` , the cloud disk can only be read. If you set this parameter to `false` , the cloud disk can be read and written. The default setting is `false` .
- `encrypted` : (optional) Set whether to encrypt the automatically created cloud disk. If you set this parameter to `true` , the cloud disk is encrypted. If you set

this parameter to `false`, the cloud disk is not encrypted. The default setting is `false`.

2. Run the following command to create a StorageClass:

```
$ kubectl create -f storageclass.yaml
```

Create a PVC

1. Create a `pvc - ssd . yaml` file.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: disk-ssd
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: alicloud-disk-ssd-hangzhou-b
  resources:
    requests:
      storage: 20Gi
```

2. Run the following command to create a PVC:

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

Result

In the left-side navigation pane under Kubernetes, choose **Application > Volumes Claim**, and select the target cluster and namespace to see that the storage class name associated to the PVC is `alicloud - disk - ssd - hangzhou - b` specified in the `StorageClassName`, and the PVC is associated with the volume.

Name	Capacity	Access Mode	Status	Storage Class Name	Relate Volume	Time Created	Action
disk-ssd	20Gi	ReadWriteOnce	Bound	alicloud-disk-ssd-hangzhou-b		12/19/2018,14:01:19	YAML Delete
pvc-disk	20Gi	ReadWriteOnce	Bound			12/18/2018,16:08:32	YAML Delete
pvc-disk-02	20Gi	ReadWriteOnce	Bound	disk		12/18/2018,17:19:15	YAML Delete
pvc-disk-03	20Gi	ReadWriteOnce	Bound	disk		12/18/2018,19:40:16	YAML Delete

Create an application

1. Create a `pvc - dynamic . yaml` file.

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

```

metadata :
  name : nginx - dynamic
  labels :
    app : nginx
spec :
  selector :
    matchLabels :
      app : nginx
  template :
    metadata :
      labels :
        app : nginx
    spec :
      containers :
        - name : nginx
          image : nginx
          volumeMounts :
            - name : disk - pvc
              mountPath : "/ data "
      volumes :
        - name : disk - pvc
          persistentVolumeClaim :
            claimName : disk - ssd

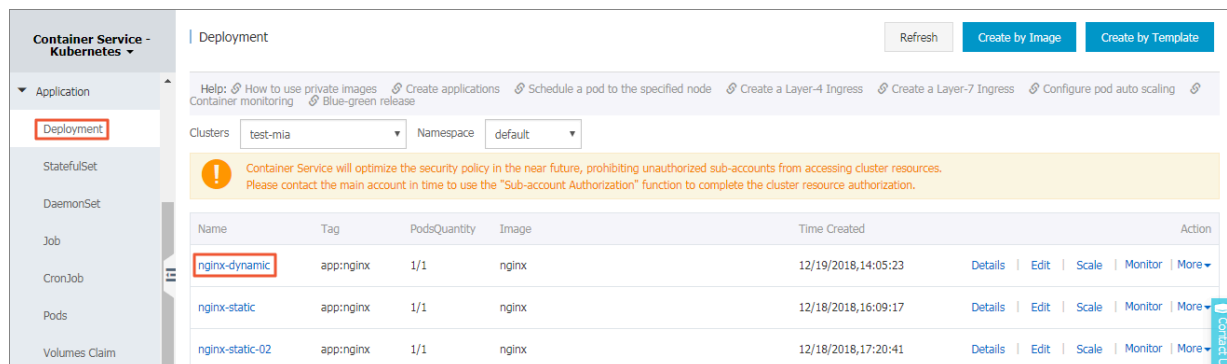
```

2. Run the following command to create a deployment:

```
$ kubectl create -f nginx - dynamic . yaml
```

Result

In the left-side navigation pane under Kubernetes, choose **Application > Deployment**, and select the target cluster and namespace to see the created deployment.



Persistent storage for a dynamic cloud disk

1. Run the following command to view the pod in which the created deployment resides:

```
$ kubectl get pod | grep dynamic
```



```
nginx - dynamic - 5c74594ccb - zl9pf      2 / 2      Running
0                               3m
```

2. Run the following command to check whether a new cloud disk is mounted to the `/data` path:

```
$ kubectl exec nginx - dynamic - 5c74594ccb - zl9pf df |
grep data
/ dev / vdh          20511312      45080      20449848      1 % / data
```

3. Run the following command to view the file in the `/data` path:

```
$ kubectl exec nginx - dynamic - 5c74594ccb - zl9pf ls /
data
lost + found
```

4. Run the following command to create a file named `dynamic` in the `/data` path:

```
$ kubectl exec nginx - dynamic - 5c74594ccb - zl9pf touch /
data / dynamic
```

5. Run the following command to view the files in the `/data` path:

```
$ kubectl exec nginx - dynamic - 5c74594ccb - zl9pf ls /
data
dynamic
lost + found
```

6. Run the following command to remove the pod named `nginx - dynamic - 78c7dcb9d7 - g9lll`:

```
$ kubectl delete pod nginx - dynamic - 5c74594ccb - zl9pf
pod "nginx - dynamic - 5c74594ccb - zl9pf" deleted
```

7. Open another kubectl interface and run the following command to view the process in which the preceding pod is removed and a new pod is created by Kubernetes:

```
$ kubectl get pod -w -l app=nginx
NAME                                READY    STATUS    RESTARTS
AGE
nginx - dynamic - 5c74594ccb - zl9pf 2 / 2    Running
0                               6m48s
nginx - dynamic - 5c74594ccb - zl9pf 2 / 2    Terminat g
0                               7m32s
nginx - dynamic - 5c74594ccb - 45sd4 0 / 2    Pending    0
0s
nginx - dynamic - 5c74594ccb - 45sd4 0 / 2    Pending    0
0s
nginx - dynamic - 5c74594ccb - 45sd4 0 / 2    Init : 0 / 1
0    0s
nginx - dynamic - 5c74594ccb - zl9pf 0 / 2    Terminat g
0    7m32s
nginx - dynamic - 5c74594ccb - zl9pf 0 / 2    Terminat g
0    7m33s
```

```

nginx - dynamic - 5c74594ccb - zl9pf      0 / 2      Terminating
0      7m33s
nginx - dynamic - 5c74594ccb - 45sd4      0 / 2      PodInitializing
0      5s
nginx - dynamic - 5c74594ccb - 45sd4      2 / 2      Running      0
22s

```

8. Run the following command to view the pod newly created by Kubernetes:

```

$ kubectl get pod
NAME                                READY   STATUS
RESTARTS   AGE
nginx - dynamic - 5c74594ccb - 45sd4  2 / 2   Running
0      2m

```

9. Run the following command to verify that the created file named `dynamic` in the `/data` path has not been removed, indicating that data in the dynamic cloud disk can be stored persistently:

```

$ kubectl exec nginx - dynamic - 5c74594ccb - 45sd4 ls /
data
dynamic
lost + found

```

3.3 Use a StatefulSet service

This topic describes the typical scenarios in which a StatefulSet is needed for creating a stateful service, and the procedure for how to use one.

Background information

A StatefulSet with N replicas is typically used for applications that require one or more of the following conditions:

- A stable deployment order. Pods are deployed or expanded sequentially. That is, pods are deployed in the defined order of 0 to N-1. Before a new pod is deployed, all its predecessors must have been in Running and Ready status.
- A stable scaling order. Pods are deleted in the defined order of N-1 to 0. Before a pod is deleted, all its predecessors must be all Running and Ready.
- Stable and unique network identifiers. After a pod is rescheduled to any other node, its PodName and HostName remain unchanged.
- Stable and persistent storage implemented through a PVC. After a pod is rescheduled, it can still access the same persistent data.

Method of using a StatefulSet service

Set `volumeClaimTemplates` to enable the system to automatically create a PVC and a PV.

This topic describes how to:

- Deploy a StatefulSet service
- Scale a StatefulSet service
- Remove a StatefulSet service
- Persistent storage of a StatefulSet service

Prerequisites

- You have created a Kubernetes cluster. For more information, see [#unique_11](#).
- You have connected to the Master node of the Kubernetes cluster. For more information, see [#unique_12](#).

Deploy a StatefulSet service



Note:

`volumeClaimTemplates` : indicates a template of PVCs of the same type. If you set this field, the system creates PVCs according to the number of the replicas that are set for the StatefulSet service. That is, the number of the PVCs and that of the replicas are the same. Furthermore, these PVCs share the same settings except for names.

1. Create a `statefulset.yaml` file.



Note:

You need to set the `storageClassName` parameter to `alicloud-disk-ssd`, indicating that an Alibaba Cloud SSD cloud disk is used.

```
apiVersion : v1
kind : Service
metadata :
  name : nginx
  labels :
    app : nginx
spec :
  ports :
    - port : 80
      name : web
    clusterIP : None
  selector :
    app : nginx
---
apiVersion : apps / v1beta2
```

```

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"
        resources :
          requests :
            storage : 20Gi

```

2. Run the following command to deploy a StatefulSet service:

```
$ kubectl create -f statefulset.yaml
```

3. Open another kubectl interface and run the following command to check that the pods are deployed in order:

```

$ kubectl get pod -w -l app=nginx
NAME          READY   STATUS    RESTARTS   AGE
web-0         0 / 1   Pending   0           0s
web-0         0 / 1   Pending   0           0s
web-0         0 / 1   ContainerCreating   0
0s
web-0         1 / 1   Running   0           20s
web-1         0 / 1   Pending   0           0s
web-1         0 / 1   Pending   0           0s
web-1         0 / 1   ContainerCreating   0
0s
web-1         1 / 1   Running   0           7s

```

4. Run the following command to view the deployed pod:

```

$ kubectl get pod
NAME          READY   STATUS    RESTARTS   AGE
web-0         1 / 1   Running   0           6m

```

web - 1 6m	1 / 1	Running	0
---------------	-------	---------	---

5. Run the following command to view the PVCs:

```
$ kubectl get pvc
```

NAME	ACCESS	MODES	STATUS	VOLUME	AGE	CAPACITY
disk - 0	ssd	web - 0	Bound	d - 2zegw7et6x	c96nbojuoo	7m
20Gi		RWO		alicloud - disk - ssd		
disk - 1	ssd	web - 1	Bound	d - 2zefbrqggv	kd10xb523h	6m
20Gi		RWO		alicloud - disk - ssd		

Scale a StatefulSet service

Scale out a StatefulSet service

1. Run the following command to scale out the StatefulSet service to three pods:

```
$ kubectl scale sts web -- replicas = 3
statefulset.apps/web scaled
```

2. Run the following command to view the pods:

```
$ kubectl get pod
```

NAME	READY	STATUS	RESTARTS
web - 0 34m	1 / 1	Running	0
web - 1 33m	1 / 1	Running	0
web - 2 26m	1 / 1	Running	0

3. Run the following command to view the PVCs:

```
$ kubectl get pvc
```

NAME	ACCESS	MODES	STATUS	VOLUME	AGE	CAPACITY
disk - 0	ssd	web - 0	Bound	d - 2zegw7et6x	c96nbojuoo	35m
20Gi		RWO		alicloud - disk - ssd		
disk - 1	ssd	web - 1	Bound	d - 2zefbrqggv	kd10xb523h	34m
20Gi		RWO		alicloud - disk - ssd		
disk - 2	ssd	web - 2	Bound	d - 2ze4jx1zym	n4n9j3pic2	27m
20Gi		RWO		alicloud - disk - ssd		

Scale in a StatefulSet service

1. Run the following command to scale in the StatefulSet service to two pods:

```
$ kubectl scale sts web -- replicas = 2
statefulset.apps/web scaled
```

2. Run the following command to view the pod and verify that the number of pods is reduced to two:

```
$ kubectl get pod
```

NAME	READY	STATUS	RESTARTS
AGE			
web - 0 38m	1 / 1	Running	0
web - 1 38m	1 / 1	Running	0

- Run the following command to view the PVCs and verify that the number of PVCs and PVs remains unchanged after the number of pods is changed:

```
$ kubectl get pvc
```

NAME	STATUS	VOLUME	CAPACITY
ACCESS MODES	STORAGECLASS	AGE	
disk - ssd - web - 0	Bound	d - 2zegw7et6x c96nbojuoo	39m
20Gi RWO		alicloud - disk - ssd	
disk - ssd - web - 1	Bound	d - 2zefbrqggv kd10xb523h	39m
20Gi RWO		alicloud - disk - ssd	
disk - ssd - web - 2	Bound	d - 2ze4jx1zym n4n9j3pic2	31m
20Gi RWO		alicloud - disk - ssd	

Rescale out a StatefulSet service

- Run the following command to scale out the StatefulSet service to three pods:

```
$ kubectl scale sts web -- replicas = 3
statefulset.apps/web scaled
```

- Run the following command to view the pods:

```
$ kubectl get pod
```

NAME	READY	STATUS	RESTARTS
AGE			
web - 0 1h	1 / 1	Running	0
web - 1 1h	1 / 1	Running	0
web - 2 8s	1 / 1	Running	0

- Run the following command to view the PVCs and verify that the newly created pods still use the original PVCs and PVs after the StatefulSet service is scaled out:

```
$ kubectl get pvc
```

NAME	STATUS	VOLUME	CAPACITY
ACCESS MODES	STORAGECLASS	AGE	
disk - ssd - web - 0	Bound	d - 2zegw7et6x c96nbojuoo	1h
20Gi RWO		alicloud - disk - ssd	
disk - ssd - web - 1	Bound	d - 2zefbrqggv kd10xb523h	1h
20Gi RWO		alicloud - disk - ssd	

```

disk - ssd - web - 2      Bound      d - 2ze4jx1zym  n4n9j3pic2
20Gi                    RWO          alicloud - disk - ssd  1h

```

Remove a StatefulSet service

1. Run the following command to view the PVC that is used by the pod named `web - 1` :

```

$ kubectl describe pod web - 1 | grep ClaimName
ClaimName : disk - ssd - web - 1

```

2. Run the following command to remove the pod named `web - 1` :

```

$ kubectl delete pod web - 1
pod "web - 1" deleted

```

3. Run the following command to view the pods and verify that the recreated pod shares the same name with the removed pod:

```

$ kubectl get pod
NAME                                READY    STATUS    RESTARTS
AGE
web - 0                            1 / 1    Running    0
1h
web - 1                            1 / 1    Running    0
25s
web - 2                            1 / 1    Running    0
9m

```

4. Run the following command to view the PVCs and verify that the recreated pod uses the same PVC as removed the pod:

```

$ kubectl get pvc
NAME                                STATUS    VOLUME                                     CAPACITY
ACCESS  MODES   STORAGECLASS  AGE
disk - ssd - web - 0      Bound    d - 2zegw7et6x  c96nbojuoo
20Gi    RWO    alicloud - disk - ssd  1h
disk - ssd - web - 1      Bound    d - 2zefbrqggv  kd10xb523h
20Gi    RWO    alicloud - disk - ssd  1h
disk - ssd - web - 2      Bound    d - 2ze4jx1zym  n4n9j3pic2
20Gi    RWO    alicloud - disk - ssd  1h

```

5. Open a new kubectl interface and run the following command to view the process of pod removal and pod recreation:

```

$ kubectl get pod -w -l app = nginx
NAME                                READY    STATUS    RESTARTS    AGE
web - 0                            1 / 1    Running    0            102m
web - 1                            1 / 1    Running    0            69s
web - 2                            1 / 1    Running    0            10m
web - 1                            1 / 1    Terminat  g            0            89s
web - 1                            0 / 1    Terminat  g            0            89s
web - 1                            0 / 1    Terminat  g            0            90s
web - 1                            0 / 1    Terminat  g            0            90s
web - 1                            0 / 1    Pending    0            0s
web - 1                            0 / 1    Pending    0            0s
web - 1                            0 / 1    ContainerC reating    0            0s

```

```
web - 1      1 / 1      Running      0      20s
```

Persistent storage of a StatefulSet service

1. Run the following command to view the file in the `/ data` path:

```
$ kubectl exec web - 1 ls / data
lost + found
```

2. Run the following command to create a `statefulse t` file in the `/ data` path:

```
$ kubectl exec web - 1 touch / data / statefulse t
```

3. Run the following command to view the files in the `/ data` path:

```
$ kubectl exec web - 1 ls / data
lost + found
statefulse t
```

4. Run the following command to remove the pod named `web - 1` :

```
$ kubectl delete pod web - 1
pod " web - 1 " deleted
```

5. Run the following command to view the files in the `/ data` path and verify that the created file named `statefulse t` has not been removed, indicating that data in the cloud disk can be stored persistently:

```
$ kubectl exec web - 1 ls / data
lost + found
statefulse t
```

3.4 Use a NAS file system when creating a stateful service

This topic describes typical scenarios in which a NAS file system is needed for creating a stateful service, and the procedure for how to use one.

Scenarios and method

If a NAS file system is mounted to multiple pods, the pods share the data in the NAS file system. After a pod modifies the data stored in the NAS file system, the application supported by the pods is required to automatically update the modified data for the other pods.

Scenarios for using a NAS file system

- You want to create or run applications that demand high disk I/O performance.
- You need a storage service that has higher read and write performance than OSS.

- You want to share files across hosts. For example, you want to use a NAS file system as a file server.

Method of using a NAS file system:

1. Manually create a NAS file system and add a mount point to it.
2. Manually create a PV and a PVC.

This topic describes how to use Alibaba Cloud NAS services in the PV/ PVC mode by using the *flexvolume* plugin provided by Alibaba Cloud.

Prerequisites

- You have created a Kubernetes cluster. For more information, see [#unique_11](#).
- You have connected to the Kubernetes cluster by using `kubectl`, see [#unique_12](#).
- You have created a NAS file system in the NAS console. For more information, see [#unique_25](#). You must make sure that the NAS file system and your Kubernetes cluster are in the same zone.
- You have added a mount point for your Kubernetes cluster in the created NAS file system. For information, see [#unique_26](#). You must make sure that the NAS file system and your cluster are in the same VPC.

Create a PV

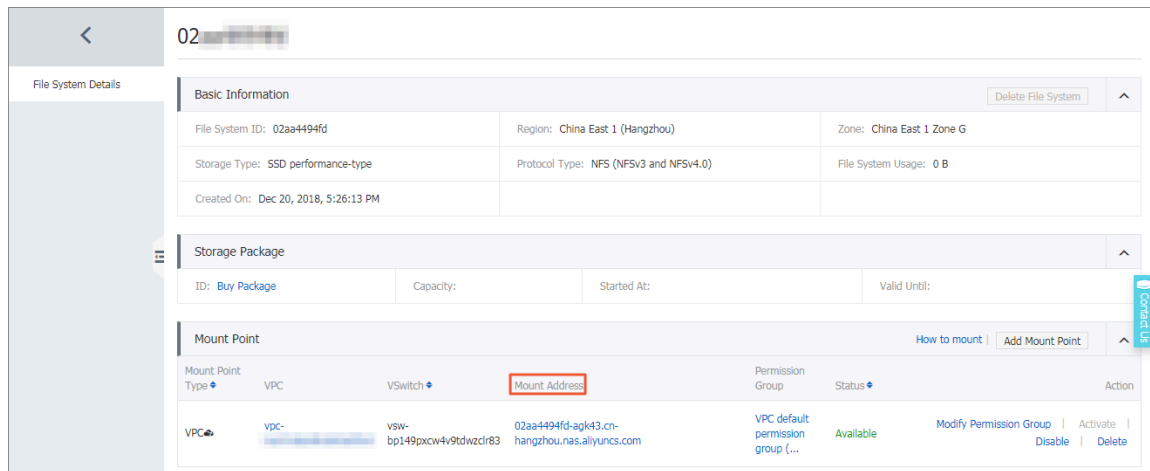
1. Create a `pv - nas . yml` file.

```
apiVersion : v1
kind : Persistent Volume
metadata :
  name : pv - nas
  labels :
    alicloud - pvname : pv - nas
spec :
  capacity :
    storage : 5Gi
  accessMode s :
    - ReadWriteM any
  flexVolume :
    driver : " alicloud / nas "
    options :
      server : "***-**. cn - hangzhou . nas . aliyuncs . com "
  //// Replace this value with your mount point .
  path : "/ k8s1 "
```

```
vers : " 4 . 0 "
```

Parameter description

- `alicloud - pvname` : indicates a PV name.
- `server` : indicates a NAS mount point. To view your mount point, log on to the NAS console, click File System List in the left-side navigation pane, select the target file system, and click Manage in the Action column to view the Mount Address in the Mount Point area. The mount address is the mount point of your NAS file system.



- `path` : indicates the NAS mount directory. You can mount a NAS sub-directory to your cluster. If the NAS sub-directory specified by you does not exist, the system automatically creates the NAS sub-directory and mounts it to your cluster.
- `vers` : (optional) indicates the version number of the NFS mount protocol. NFS file system V3.0 and V4.0 are available. The default is V4.0.
- `mode` : (optional) indicates the access permission to the mount directory. By default, this parameter is not set.



Note:

- Access permission to the root directory of the NAS file system cannot be set.
- If you set the `mode` parameter for a NAS file system that stores a large amount of data, the process of mounting the NAS file system to a cluster may take an excessive amount of time or even fail. We recommend that you do not set this parameter.

2. Run the following command to create a PV:

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

Result

In the left-side navigation pane under Kubernetes, choose Clusters > Volumes, and select the target cluster to view the created PV.

Name	Capacity	Access Mode	Reclaim Policy	Status	Storage Class Name	Binding Volume Claim	Time Created	Action
[Redacted]	20Gi	ReadWriteOnce	Retain	Bound	alicloud-disk-ssd-hangzhou-g	Namespace: default Name: disk-ssd	12/19/2018,14:48:49	Edit Labels YAML Delete
[Redacted]	20Gi	ReadWriteOnce	Delete	Bound	alicloud-disk-ssd	Namespace: default Name: disk-ssd-web-1	12/20/2018,10:24:20	Edit Labels YAML Delete
[Redacted]	20Gi	ReadWriteOnce	Delete	Bound	alicloud-disk-ssd	Namespace: default Name: disk-ssd-web-0	12/20/2018,10:24:05	Edit Labels YAML Delete
pv-nas	5Gi	ReadWriteMany	Retain	Bound		Namespace: default Name: pvc-nas	12/20/2018,19:16:00	Edit Labels YAML Delete

Create a PVC

Create a PVC for the NAS file system. Specifically, you need to set the `selector` field to filter for the created PV so that you can associate the PVC with the correct PV.

1. Create a `pvc - nas . yaml` file.

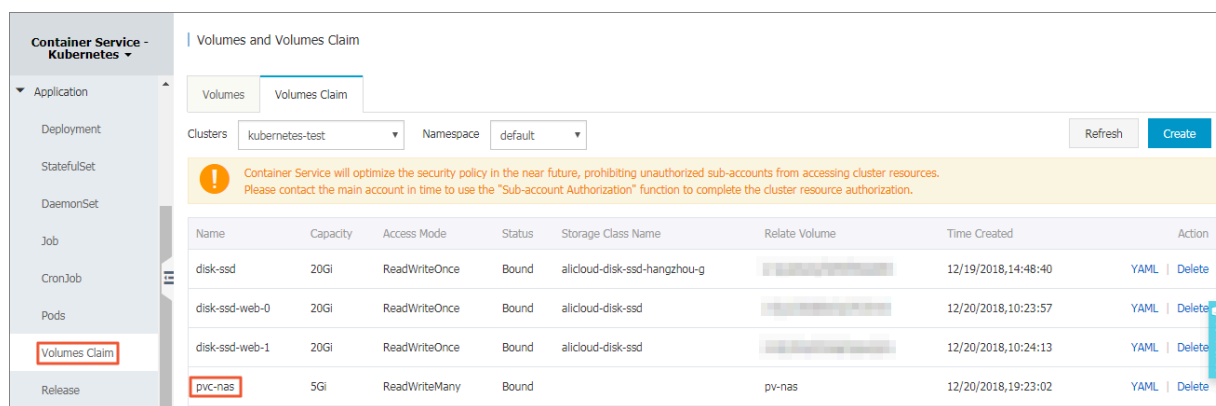
```
kind : Persistent VolumeClaim
apiVersion : v1
metadata :
  name : pvc - nas
spec :
  accessModes :
    - ReadWriteMany
  resources :
    requests :
      storage : 5Gi
  selector :
    matchLabels :
      alicloud - pvname : pv - nas
```

2. Run the following command to create a PVC:

```
$ kubectl create -f pvc - nas . yaml
```

Result

In the left-side navigation pane under Kubernetes, choose Application > Volumes Claim, and select the target cluster and namespace to view the created PVC.



Create an application

1. Create a `nas . yaml` .

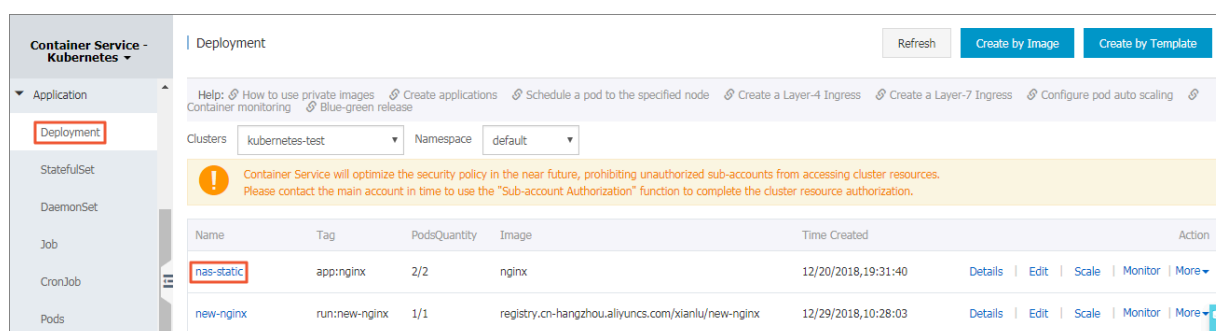
```
apiVersion : apps / v1
kind : Deployment
metadata :
  name : nas - static
  labels :
    app : nginx
spec :
  replicas : 2
  selector :
    matchLabels :
      app : nginx
  template :
    metadata :
      labels :
        app : nginx
    spec :
      containers :
        - name : nginx
          image : nginx
          ports :
            - containerPort : 80
          volumeMounts :
            - name : pvc - nas
              mountPath : "/ data "
      volumes :
        - name : pvc - nas
          persistentVolumeClaim :
            claimName : pvc - nas
```

2. Run the following command to create a deployment:

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

Result

In the left-side navigation pane under Kubernetes, choose **Application > Deployment**, and select the target cluster and namespace to view the created deployment.



Verify that the NAS file system is shared by pods

1. Run the following command to view the pods in which the created deployment resides:

```
$ kubectl get pod
```

NAME	READY	STATUS	RESTARTS
nas - static - f96b6b5d7 - rcb2f	1 / 1	Running	0
nas - static - f96b6b5d7 - wthmb	1 / 1	Running	0

2. Run the following commands to view the files in the `/ data` path of each pod:

```
$ kubectl exec nas - static - f96b6b5d7 - rcb2f ls / data
```

```
$ kubectl exec nas - static - f96b6b5d7 - wthmb ls / data
```



Note:

The two `/ data` paths are empty.

3. Run the following command to create file `nas` in the `/ data` path of one pod:

```
$ kubectl exec nas - static - f96b6b5d7 - rcb2f touch / data / nas
```

4. Run the following commands to view the files in the `/ data` path of each pod:

```
$ kubectl exec nas - static - f96b6b5d7 - rcb2f ls / data
```

```
$ kubectl exec nas - static - f96b6b5d7 - wthmb ls / data
```



Note:

After you create the file in the `/ data` path of one pod, the file then exists in both the `/ data` paths of the two pods. This means that the two pods share the NSA file system.

Verify that data on the NAS file system are stored persistently

1. Run the following command to remove all the pods of the created application:

```
$ kubectl delete pod nas - static - f96b6b5d7 - rcb2f nas -
static - f96b6b5d7 - wthmb
pod " nas - static - f96b6b5d7 - rcb2f " deleted
pod " nas - static - f96b6b5d7 - wthmb " deleted
```

2. Open another kubectl interface and run the following command to view the process in which the original pods are removed and new pods are created by Kubernetes:

```
$ kubectl get pod -w -l app=nginx
NAME                                READY    STATUS    RESTARTS
AGE
nas - static - f96b6b5d7 - rcb2f    1 / 1    Running    0
27m
nas - static - f96b6b5d7 - wthmb    1 / 1    Running    0
27m
nas - static - f96b6b5d7 - rcb2f    1 / 1    Terminat  g    0
28m
nas - static - f96b6b5d7 - wnqdj    0 / 1    Pending    0
0s
nas - static - f96b6b5d7 - wnqdj    0 / 1    Pending    0
0s
nas - static - f96b6b5d7 - wnqdj    0 / 1    ContainerC reating
0 0s
nas - static - f96b6b5d7 - wthmb    1 / 1    Terminat  g    0
28m
nas - static - f96b6b5d7 - nwkds    0 / 1    Pending    0
0s
nas - static - f96b6b5d7 - nwkds    0 / 1    Pending    0
0s
nas - static - f96b6b5d7 - nwkds    0 / 1    ContainerC reating
0 0s
nas - static - f96b6b5d7 - rcb2f    0 / 1    Terminat  g    0
28m
nas - static - f96b6b5d7 - wthmb    0 / 1    Terminat  g    0
28m
nas - static - f96b6b5d7 - rcb2f    0 / 1    Terminat  g    0
28m
nas - static - f96b6b5d7 - rcb2f    0 / 1    Terminat  g    0
28m
nas - static - f96b6b5d7 - wnqdj    1 / 1    Running    0
10s
nas - static - f96b6b5d7 - wthmb    0 / 1    Terminat  g    0
28m
nas - static - f96b6b5d7 - wthmb    0 / 1    Terminat  g    0
28m
nas - static - f96b6b5d7 - nwkds    1 / 1    Running    0
17s
```

3. Run the following command to view the new pods created by Kubernetes:

```
$ kubectl get pod
NAME                                READY    STATUS    RESTARTS
AGE
nas - static - f96b6b5d7 - nwkds    1 / 1    Running    0
21s
```

```
nas - static - f96b6b5d7 - wnqdj      1 / 1      Running      0
21s
```

4. Run the following commands to view the files in the `/ data` path of each pod:

```
$ kubectl exec nas - static - f96b6b5d7 - nwks ls / data
nas
```

```
$ kubectl exec nas - static - f96b6b5d7 - wnqdj ls / data
nas
```



Note:

The created file, namely, file `nas` has not been removed. This means that data in the NAS file system can be stored persistently.

3.5 Use an OSS bucket when creating a stateful service

This topic describes typical scenarios in which an Object Storage Service (OSS) bucket is needed for creating a stateful service, and the procedure for how to use the bucket.

Scenarios and method

Alibaba Cloud OSS provides massive, secure, low-cost, and highly reliable cloud storage services. An OSS bucket can be mounted to multiple pods.

Scenarios:

- Disk I/O performance requirements are low.
- Shared services such as files, figures, and short videos are to be configured.

Method:

1. Manually create a bucket.
2. Obtain the AccessKey ID and AccessKey Secret pair.
3. Manually create a Persistent Volume (PV) and a Persistent Volume Claim (PVC).

Prerequisites

- You have created a Kubernetes cluster. For more information, see [#unique_11](#).
- You have connected to the Kubernetes cluster by using `kubectl`, see [#unique_12](#).
- You have created a bucket in the OSS console, see [#unique_28](#).

Precautions

- Upgrading a Kubernetes cluster of Alibaba Cloud Container Service causes kubelet and the OSSFS driver to restart. As a result, the OSS directory becomes unavailabl

e. In this case, the pods that use the OSS bucket must be recreated. We recommend that you add health check settings in the YAML file of your application. If you add health check settings for your application, the pods will be automatically restarted to remount the OSS bucket when the OSS directory within your container becomes unavailable.

- If you use a Kubernetes cluster of the latest version, the preceding issue does not affect you.

Create a PV


1. Create a `pv - oss . yaml` file.

```
apiVersion : v1
kind : Persistent Volume
metadata :
  name : pv - oss
  labels :
    alicloud - pvname : pv - oss
spec :
  capacity :
    storage : 5Gi
  accessModes :
    - ReadWriteMany
  storageClassName : oss
  flexVolume :
    driver : "alicloud / oss"
    options :
      bucket : "docker"          //// Replace
this value with your bucket name .
      url : "oss - cn - hangzhou . aliyuncs . com"  ////
Replace this value with your URL .
      akId : "****"              //// Replace this
value with your AccessKey ID .
      akSecret : "****"          //// Replace this
value with your AccessKey Secret .
      otherOpts : "- o max_stat_cache_size = 0 - o
allow_other"  //// Replace this value with your
specified otherOpts value .
```

Parameter description

- `alicloud - pvname` : indicates a PV name. This parameter value must be used in the `selector` field of the PVC associated with the PV.
- `bucket` : indicates a bucket name. Only buckets can be mounted to a Kubernetes cluster. The sub-directories or files in a bucket cannot be mounted to any Kubernetes cluster.
- `url` : indicates a domain name used to access the OSS bucket, namely, an endpoint. For more information, see [#unique_29](#). You can also view the

endpoint of the created OSS bucket in the OSS console. That is, log on to the OSS console, select the target bucket, and view Endpoint in the Domain Names area.

- `akId` : indicates your AccessKey ID. In the Container Service console, click  in the upper-right corner. For a primary account, select accesskeys.

For a RAM user, select AccessKey. Then, you can create your AccessKey ID and AccessKey Secret.

- `akSecret` : indicates your AccessKey Secret. Use the same method to obtain this parameter value as that to obtain the value of the `akId` parameter.
- `otherOpts` : indicates custom parameters for mounting the OSS bucket. Set this parameter in the format of `- o *** - o ***`. For more information, see [#unique_30](#).

2. Run the following command to create a PV:

```
$ kubectl create -f pv - oss . yaml
```

Result

In the left-side navigation pane under Kubernetes, choose Clusters > Volumes, and select the target cluster to view the created PV.

Container Service - Kubernetes

Overview

Clusters

Clusters

Nodes

Volumes

Namespace

Authorization

Application

Deployment

StatefulSet

DaemonSet

Volumes and Volumes Claim

Volumes

Volumes Claim

Clusters

kubernetes-test

Refresh

Create

Name	Capacity	Access Mode	Reclaim Policy	Status	Storage Class Name	Binding Volume Claim	Time Created	Action
	20Gi	ReadWriteOnce	Retain	Bound	alicloud-disk-ssd-hangzhou-g	Namespace: default Name: disk-ssd	12/19/2018,14:48:49	Edit Labels YAML Delete
	20Gi	ReadWriteOnce	Delete	Bound	alicloud-disk-ssd	Namespace: default Name: disk-ssd-web-1	12/20/2018,10:24:20	Edit Labels YAML Delete
	20Gi	ReadWriteOnce	Delete	Bound	alicloud-disk-ssd	Namespace: default Name: disk-ssd-web-0	12/20/2018,10:24:05	Edit Labels YAML Delete
pv-nas	5Gi	ReadWriteMany	Retain	Bound		Namespace: default Name: pvc-nas	12/20/2018,19:16:00	Edit Labels YAML Delete
pv-oss	5Gi	ReadWriteMany	Retain	Bound	oss	Namespace: default Name: pvc-oss	12/21/2018,16:01:55	Edit Labels YAML Delete

Create a PVC

Create a PVC for the OSS bucket. Specifically, you need to set the `selector` field to filter for the created PV so that you can associate the PVC with the correct PV. Set the `storageClassName` parameter to associate the PVC with only the PV of the OSS type.

1. Create a `pvc - oss . yaml` file.

```
kind : Persistent VolumeClaim
apiVersion : v1
metadata :
  name : pvc - oss
spec :
  accessModes :
    - ReadWriteMany
  storageClassName : oss
resources :
  requests :
    storage : 5Gi
selector :
  matchLabels :
    alicloud - pvname : pv - oss
```

2. Run the following command to create a PVC:

```
$ kubectl create -f pvc - oss . yaml
```

Result

In the left-side navigation pane under Kubernetes, choose **Application > Volumes Claim**, and select the target cluster and namespace to view the created PVC.

Container Service will optimize the security policy in the near future, prohibiting unauthorized sub-accounts from accessing cluster resources. Please contact the main account in time to use the "Sub-account Authorization" function to complete the cluster resource authorization.

Name	Capacity	Access Mode	Status	Storage Class Name	Relate Volume	Time Created	Action
disk-ssd	20Gi	ReadWriteOnce	Bound	alicloud-disk-ssd-hangzhou-g		12/19/2018,14:48:40	YAML Delete
disk-ssd-web-0	20Gi	ReadWriteOnce	Bound	alicloud-disk-ssd		12/20/2018,10:23:57	YAML Delete
disk-ssd-web-1	20Gi	ReadWriteOnce	Bound	alicloud-disk-ssd		12/20/2018,10:24:13	YAML Delete
pvc-nas	5Gi	ReadWriteMany	Bound		pv-nas	12/20/2018,19:23:02	YAML Delete
pvc-oss	5Gi	ReadWriteMany	Bound	oss	pv-oss	12/21/2018,16:02:06	YAML Delete

Create an application

1. Create an `oss - static . yaml` file.

```
apiVersion : apps / v1
kind : Deployment
metadata :
  name : oss - 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 - oss
          mountPath : "/ data "
        - name : pvc - oss
          mountPath : "/ data1 "
      livenessProbe :
        exec :
          command :
            - sh
            - -c
            - cd / data
        initialDelaySeconds : 30
        periodSeconds : 30
  volumes :
    - name : pvc - oss
      persistentVolumeClaim :
        claimName : pvc - oss

```

**Note:**

For more information about `livenessProbe`, see [#unique_31](#).

2. Run the following command to create a deployment:

```
$ kubectl create -f oss - static . yaml d
```

Result

In the left-side navigation pane under Kubernetes, choose **Application > Deployment**, and select the target cluster and namespace to view the created deployment.

The screenshot shows the 'Deployment' page in the Alibaba Cloud Container Service for Kubernetes console. The left sidebar has 'Deployment' selected. The main area shows a table of deployments with columns: Name, Tag, PodsQuantity, Image, Time Created, and Action. The 'oss-static' deployment is highlighted with a red box.

Name	Tag	PodsQuantity	Image	Time Created	Action
nas-static	app:nginx	2/2	nginx	12/20/2018,19:31:40	Details Edit Scale Monitor More
new-nginx	run:new-nginx	1/1	registry.cn-hangzhou.aliyuncs.com/xianlu/new-nginx	12/29/2018,10:28:03	Details Edit Scale Monitor More
nginx-dynamic	app:nginx	1/1	nginx	12/19/2018,17:40:53	Details Edit Scale Monitor More
old-nginx	run:old-nginx	2/2	registry.cn-hangzhou.aliyuncs.com/xianlu/old-nginx	12/29/2018,10:29:27	Details Edit Scale Monitor More
oss-static	app:nginx	1/1	nginx	12/21/2018,16:02:15	Details Edit Scale Monitor More

Verify that data in the OSS bucket are stored persistently

1. Run the following command to view the pod in which the created deployment resides:

```
$ kubectl get pod
NAME                                READY    STATUS    RESTARTS
AGE
oss - static - 66fbb85b67 - dqbl2  1 / 1    Running    0
1h
```

2. Run the following command to view the files in the `/ data` path:

```
$ kubectl exec oss - static - 66fbb85b67 - dqbl2 ls / data
| grep tmpfile
```



Note:

The `/ data` path is empty.

3. Run the following command to create a file named `tmpfile` in the `/ data` path:

```
$ kubectl exec oss - static - 66fbb85b67 - dqbl2 touch /
data / tmpfile
```

4. Run the following command to view the file in the `/ data` path:

```
$ kubectl exec oss - static - 66fbb85b67 - dqbl2 ls / data
| grep tmpfile
tmpfile
```

5. Run the following command to remove the pod named `oss - static - 66fbb85b67 - dqbl2` :

```
$ kubectl delete pod oss - static - 66fbb85b67 - dqbl2
pod "oss - static - 66fbb85b67 - dqbl2" deleted
```

6. Open another kubectl interface and run the following command to view the process in which the preceding pod is removed and a new pod is created by Kubernetes:

```
$ kubectl get pod - w - l app = nginx
NAME                                READY    STATUS    RESTARTS
AGE
oss - static - 66fbb85b67 - dqbl2  1 / 1    Running    0
78m
oss - static - 66fbb85b67 - dqbl2  1 / 1    Terminat  g    0
78m
oss - static - 66fbb85b67 - zlvnw  0 / 1    Pending    0
< invalid >
oss - static - 66fbb85b67 - zlvnw  0 / 1    Pending    0
< invalid >
oss - static - 66fbb85b67 - zlvnw  0 / 1    ContainerC reating
0 < invalid >
```

```

oss - static - 66fbb85b67 - dqbl2    0 / 1    Terminating    0
78m
oss - static - 66fbb85b67 - dqbl2    0 / 1    Terminating    0
78m
oss - static - 66fbb85b67 - dqbl2    0 / 1    Terminating    0
78m
oss - static - 66fbb85b67 - zlvwm    1 / 1    Running          0
< invalid >

```

7. Run the following command to view the pod created by Kubernetes:

```

$ kubectl get pod
NAME                                READY   STATUS    RESTARTS
AGE
oss - static - 66fbb85b67 - zlvwm   1 / 1   Running   0
40s

```

8. Run the following command to verify that the created file named *tmpfile* in the */data* path has not been removed, indicating that data in the OSS bucket can be stored persistently:

```

$ kubectl exec oss - static - 66fbb85b67 - zlvwm ls / data
| grep tmpfile
tmpfile

```

4 Release

4.1 Implement Layer-4 canary release by using Alibaba Cloud Server Load Balancer in a Kubernetes cluster

In a Kubernetes cluster, Layer-7 Ingress cannot properly implement gray release for services accessed by using TCP/UDP. This document introduces how to implement Layer-4 canary release by using Server Load Balancer.

Prerequisites

- You have created a Kubernetes cluster. For more information, see [#unique_34](#).
- You have connected to the master node by using SSH. For more information, see [#unique_35](#).

Step 1 Deploy the old version of the service

1. Log on to the [Container Service console](#).
2. Click Application > Deployment in the left-side navigation pane.
3. Click Create by template in the upper-right corner.
4. Select the cluster and namespace from the Clusters and Namespace drop-down lists. Select a sample template or Custom from the Resource Type drop-down list. Click DEPLOY.

In this example, an nginx orchestration that exposes the service by using SLB.

```
apiVersion : extensions / v1beta1
kind : Deployment
metadata :
  labels :
    run : old - nginx
  name : old - nginx
spec :
  replicas : 1
  selector :
    matchLabels :
      run : old - nginx
  template :
    metadata :
      labels :
        run : old - nginx
        app : nginx
    spec :
```

```

        containers :
        - image : registry . cn - hangzhou . aliyuncs . com /
xianlu / old - nginx
        imagePullPolicy : Always
        name : old - nginx
        ports :
        - containerPort : 80
          protocol : TCP
        restartPolicy : Always
---
apiVersion : v1
kind : Service
metadata :
  labels :
    run : nginx
    name : nginx
spec :
  ports :
  - port : 80
    protocol : TCP
    targetPort : 80
  selector :
    app : nginx
  sessionAffinity : None
  type : LoadBalancer ## Expose the service by using
Alibaba Cloud SLB .

```

5. Click **Application > Deployment and Application > Service** in the left-side navigation pane to check the deployment and service.

6. Click the external endpoint at the right of the service to go to the Nginx default welcome page. In this example, old is displayed on the Nginx welcome page, which indicates that the currently accessed service corresponds to the backend old-nginx container.

To easily display the results of multiple releases , we recommend that you log on to the master node and execute the `curl` command to view the deployment results.

```

# bash
# for x in { 1 .. 10 } ; do curl EXTERNAL - IP ; done
## EXTERNAL - IP is the external endpoint of the
service .
old
old
old
old
old
old
old
old
old
old

```

old

Step 2 Bring new deployment version online

1. Log on to the [Container Service console](#).
2. Click **Application > Deployment** in the left-side navigation pane.
3. Click **Create by template** in the upper-right corner.
4. Select the cluster and namespace from the **Clusters** and **Namespace** drop-down lists. Select a sample template or **Custom** from the **Resource Type** drop-down list. Click **DEPLOY**.

In this example, create a new version of `nginx` deployment that contains the `app : nginx` label. The label is used to use the same `nginx` service as that of the old version of deployment to bring the corresponding traffic.

The orchestration template in this example is as follows:

```
apiVersion : extensions / v1beta1
kind : Deployment
metadata :
  labels :
    run : new - nginx
  name : new - nginx
spec :
  replicas : 1
  selector :
    matchLabels :
      run : new - nginx
  template :
    metadata :
      labels :
        run : new - nginx
        app : nginx
    spec :
      containers :
        - image : registry . cn - hangzhou . aliyuncs . com /
          xianlu / new - nginx
          imagePullPolicy : Always
          name : new - nginx
          ports :
            - containerPort : 80
              protocol : TCP
          restartPolicy : Always
```

5. Click **Deployment** in the left-side navigation pane. The deployment of `new-nginx` is displayed on the **Deployment** page.

6. Log on to the master node and execute the curl command to view the service access.

```
# bash
# for x in { 1 .. 10 } ; do curl EXTERNAL - IP ; done
## EXTERNAL - IP is the external endpoint of the
service .
new
new
new
old
new
old
new
new
old
old
```

You can see that the old service and new service are accessed for five times respectively. This is mainly because the service follows the Server Load Balancer policy of average traffic to process traffic requests, and the old deployment and new deployment are the same pod, which makes their traffic ratio as 1:1.

Step 3 Adjust traffic weight

You must adjust the number of pods in the backend to adjust the corresponding weight for the canary release based on Server Load Balancer. For example, to make the new service to have higher weight, you can adjust the number of new pods to four .



Note:

If the old application version and new application version coexist, the results returned after executing the curl command of a sample do not conform to the configured weight strictly. In this example, to obtain the approximate effect, execute the curl command for 10 times to observe more samples.

1. Log on to the [Container Service console](#).
2. Under Kubernetes, click Application > Deployment in the left-side navigation pane.
3. Select the cluster and namespace from the Clusters and Namespace drop-down lists. Click Update at the right of the deployment.

4. In the displayed dialog box, set the number of pods to four.

**Note:**

The default update method of Kubernetes deployment resources is `rollingUpdate`. Therefore, during the update process, the minimum number of containers that provide the service is guaranteed and this number can be adjusted in the template.

5. After the deployment, log on to the master node and execute the `curl` command to view the effect.

```
# bash
# for x in { 1 .. 10 } ; do curl EXTERNAL - IP ; done
## EXTERNAL - IP is the external endpoint of the
service .
new
new
new
new
new
old
new
new
new
old
```

You can see the new service is requested for eight times and the old service is requested twice among the 10 requests.

You can dynamically adjust the number of pods to adjust the weights of the new service and old service and implement the canary release.

4.2 Implement a gray release and a blue/green deployment through Ingress in a Kubernetes cluster

4.2.1 Gray releases and blue/green deployment

This topic describes how to implement a gray release and a blue/green deployment by using the Ingress function provided by Alibaba Cloud Container Service for Kubernetes.

Background information

With a gray release or a blue/green deployment, you can create two identical production environments for the latest version of the target software and an earlier

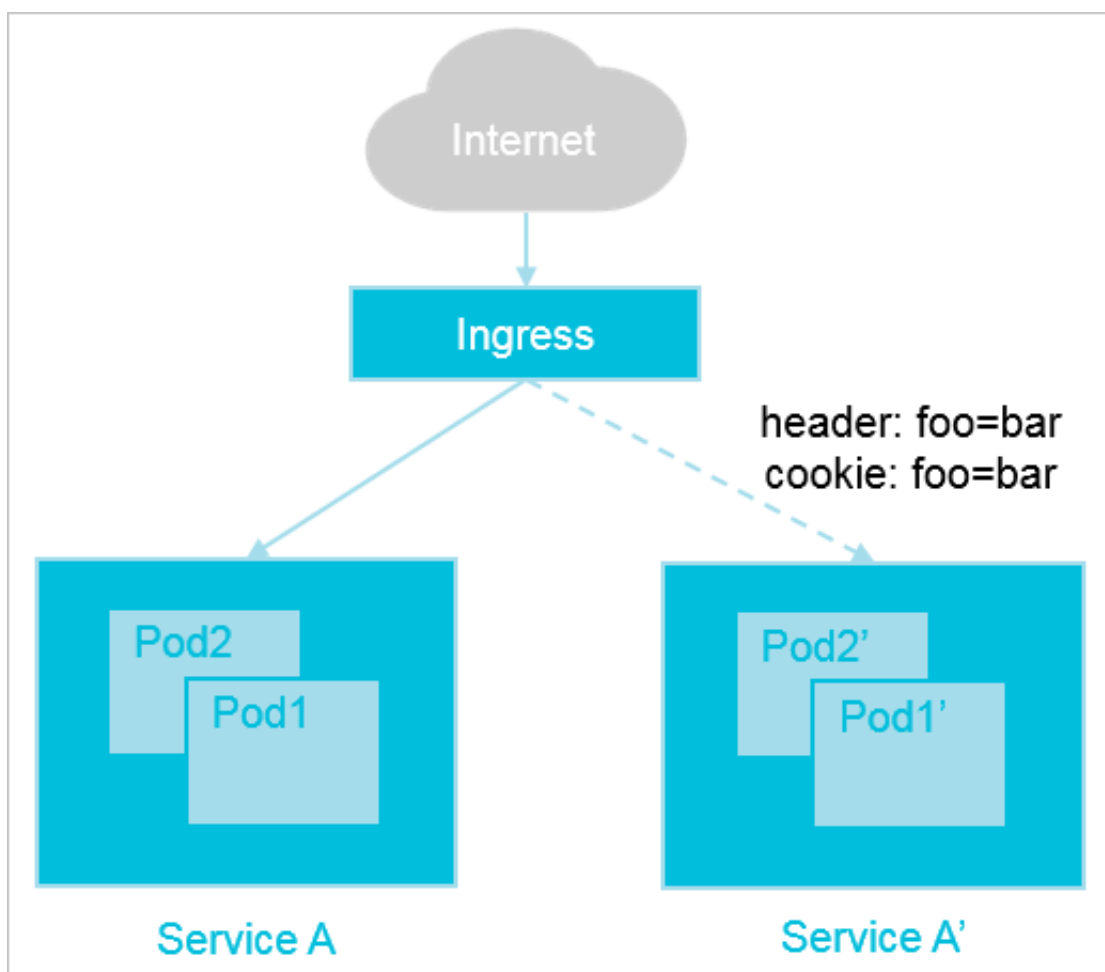
version. Then you can apply specific rules to reroute traffic from the earlier version to the latest version without affecting the software of the earlier version. After the software of the latest version has run without exceptions for a specified period, you can reroute all traffic from the earlier version to the latest version.

A/B testing is a type of comparative and incremental gray release. Specifically, with A/B testing, you can keep some users using the service of an earlier version, and reroute traffic of other users to the service of the latest version. If the service of the latest version runs without exceptions for the specified period of time, then you can gradually reroute all user traffic to the service of the latest version.

Scenarios

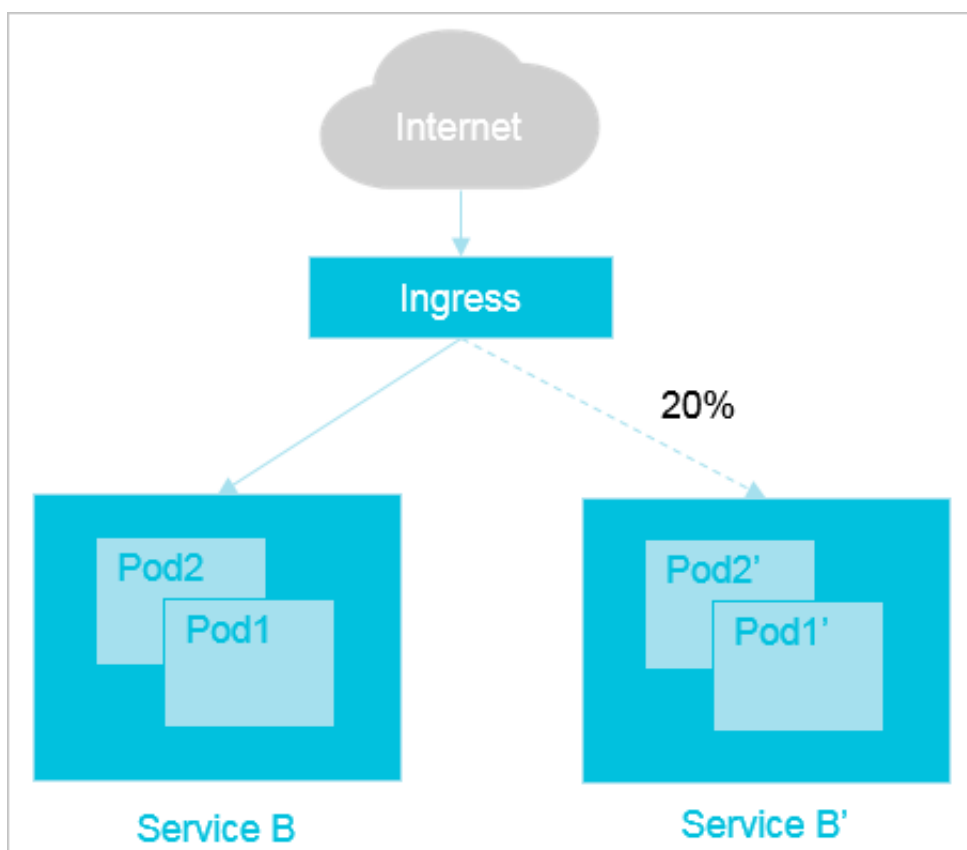
Scenario 1

For example, assume that Service A already runs online to provide an externally accessible Layer-7 service, and a new version of this service with new features, namely, Service A' , is developed. You want to release Service A' , but you do not want it to directly replace Service A at once. Additionally, you want the client requests of which the request headers contain `foo = bar` or the cookies contain `foo = bar` to be forwarded to Service A' . Then, after Service A' has run without exceptions for a specified period, you want to reroute all traffic from Service A to Service A' , and then smoothly bring Service A offline.



Scenario 2

For example, assume that an earlier version of a service, named Service B, is running online to provide an externally accessible Layer-7 service. However, it has known problems. A new version, namely Service B' is developed with the problems fixed and you want to release this latest version. However, you initially want to reroute only 20% of all client traffic to Service B' . Then, after Service B' has run without exceptions for a period, you want to reroute all traffic from Service B to Service B' , and then smoothly bring Service B offline.



To meet the preceding application release requirements, Alibaba Cloud Container Service for Kubernetes uses the Ingress function to provide the following four methods of traffic distribution:

In A/B testing

- Distribute traffic according to the request header
- Distribute traffic according to the cookie
- Distribute traffic according to the Query Param

In a blue/green deployment

- Distribute traffic according to the service weight

4.2.2 Gray release limits

This topic describes the limits for a gray release that is implemented by the Ingress function provided by Alibaba Cloud Container Service for Kubernetes.

The Ingress controller of Alibaba Cloud Container Service for Kubernetes must be V `0 . 12 . 0 - 5` or later.

To view the version number of the Ingress controller, run either of the following commands as required.

- For a cluster in which applications are deployed by using the Deployment method, run:

```
kubectl -n kube-system get deploy nginx - ingress -
controller -o yaml | grep -v 'apiVersion' | grep '
aliyun - ingress - controller'
```

- For a cluster in which applications are deployed by using the DaemonSet method, run:

```
kubectl -n kube-system get ds nginx - ingress -
controller -o yaml | grep -v 'apiVersion' | grep '
aliyun - ingress - controller'
```

If your Ingress Controller is earlier than `0.12.0-5`, you can upgrade it by running either of the following commands as required.

- For a cluster in which applications are deployed by using the Deployment method, run:

```
kubectl -n kube-system set image deploy / nginx -
ingress - controller nginx - ingress - controller = registry.cn -
hangzhou.aliyuncs.com / acs / aliyun - ingress - controller :
0.12.0-5
```

- For a cluster in which applications are deployed by using the DaemonSet method, run:

```
kubectl -n kube-system set image ds / nginx - ingress
- controller nginx - ingress - controller = registry.cn -
hangzhou.aliyuncs.com / acs / aliyun - ingress - controller : 0
.12.0-5
```

4.2.3 Annotation

This topic describes the annotation used when you implement a gray release by using the Ingress function provided by Alibaba Cloud Container Service for Kubernetes.

To support a gray release, the Ingress function of Alibaba Cloud Container Service for Kubernetes provides the following annotation: routing rules set by using `nginx`.

`ingress.kubernetes.io/service-match` and service weight set by using `nginx.ingress.kubernetes.io/service-weight`.



Note:

If you set routing rules by using `nginx.ingress.kubernetes.io/service-match` and service weight by using `nginx.ingress.kubernetes.io/service-weight`.

`service - weight` , the system first determines whether the routing rules set by using `nginx . ingress . kubernetes . io / service - match` are matched when receiving a request :

- If no routing rules are matched, the system forwards the request to the application of the earlier version.
- If the routing rules are matched, the system forwards the request according to the service weight that you set by using `nginx . ingress . kubernetes . io / service - weight` .

Routing rules set by using `nginx . ingress . kubernetes . io / service - match`

This annotation is used to set the routing rules for the service of the latest version.

The annotation format is as follows:

```
nginx . ingress . kubernetes . io / service - match : |
  < service - name >: < match - rule >
```

Parameter description

`service - name` : service name. The requests that meet the requirements of the route matching rules are routed to this service.

`match - rule` : matching rules of routes.

- Matching types:
 - `header` : based on the request header. This matching type supports regular expression matches and full expression matches.
 - `cookie` : based on the cookie. This matching type supports regular expression matches and full matches.
 - `query` : based on the queried parameter. This matching type supports regular expression matches and full matches.
- Matching methods:
 - The format of a regular expression match is `/ { Regular Expression } /`.
 - The format of a full match is `" { exact expression } "`

Configuration examples

```
# If the request header of a request meets the
# requirements of the regular expression of foo and
# ^ bar $, the request is forwarded to the new - nginx
# service .
new - nginx : header (" foo ", /^ bar $/)
```

```
# In the request header of a request, if foo fully
  matches bar, the request will be forwarded to the
  new - nginx service .
new - nginx : header (" foo ", " bar ")

# In the cookie of a request, if foo matches the
  regular expression ^ sticky -.+$, the request will be
  forwarded to the new - nginx service .
new - nginx : cookie (" foo ", /^ sticky -.+$/ )

# In the query param of a request, if foo fully
  matches bar, the request will be forwarded to the
  new - nginx service .
new - nginx : query (" foo ", " bar ")
```

Service weight set by using `nginx . ingress . kubernetes . io / service - weight`

This annotation is used to set the traffic weights for the service of the latest version and the service of the earlier version. The annotation format is as follows:

```
nginx . ingress . kubernetes . io / service - weight : |
  < new - svc - name >:< new - svc - weight >, < old - svc - name
  >:< old - svc - weight >
```

Parameter description

`new - svc - name` : name of the service of the latest version.

`new - svc - weight` : weight of the service of the latest version.

`old - svc - name` : name of the service of the earlier version.

`old - svc - weight` : weight of the service of the earlier version.

Configuration examples

```
nginx . ingress . kubernetes . io / service - weight : |
  new - nginx : 20 , old - nginx : 60
```



Note:

- Service weights are calculated by using relative values. In the preceding example, the service of the latest version is set to 20 weight and the service of the earlier version is set to 60 weight. Therefore, the weight percentage of the latest version service is 25% and the weight percentage of the earlier version service is 75%.
- In a service group that is composed of services that have the same host and path in an Ingress YAML, the default service weight is 100.

4.2.4 Step 1: Deploy a service

This topic describes how to deploy a service.

Prerequisites

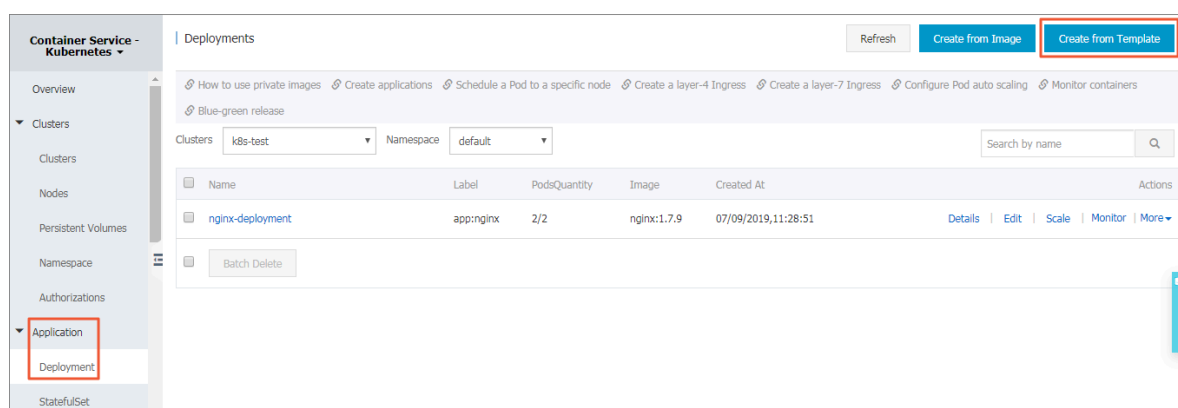
- You have created a Kubernetes cluster. For more information, see [#unique_11](#).
- You have connected to the Kubernetes cluster by using `kubectl`. For more information, see [#unique_12](#).

Context

With canary deployment or blue/green deployment, you can run the new version of a service in an environment identical to the environment where the old version of the service operates. Specifically, you can set a rule to direct part of the traffic that is destined for the old version to the new version of the service without affecting the old version. After the new version of the service operates normally for a period, you can direct the remaining traffic to the the new version of the service. In the following steps, a service named `lod - nginx` is created as the old version of the sample service.

Procedure

1. Log on to the [Container Service console](#).
2. In the left-side navigation pane under Container Service-Kubernetes, choose **Application > Deployment**.
3. In the upper-right corner, click **Create from Template**.



4. Select the target cluster and namespace, select a sample template or customize a template, and then click Create.

Clusters: k8s-test

Namespace: default

Sample Template: Custom

Template

```

1 apiVersion: extensions/v1beta1
2 kind: Deployment
3 metadata:
4   name: old-nginx
5 spec:
6   replicas: 2
7   selector:
8     matchLabels:
9       run: old-nginx
10  template:
11    metadata:
12      labels:
13        run: old-nginx
14    spec:
15      containers:
16        - image: registry.cn-hangzhou.aliyuncs.com/xianlu/old-nginx
17          imagePullPolicy: Always
18          name: old-nginx
19          ports:
20            - containerPort: 80
21              protocol: TCP
22          restartPolicy: Always
23 ---
24 apiVersion: v1
25 kind: Service
26 metadata:
27   name: old-nginx
28 spec:
29   type: NodePort
  
```

The creation process has started. Click here to check the progress: [Kubernetes Dashboard](#)

Save Template Create

In this example, a template is orchestrated to deploy an Nginx application that contains the required deployment, the target service, and an Ingress. The deployment exposes its port through NodePort. The Ingress provides externally accessible services. The orchestration template is as follows:

```

apiVersion : extensions / v1beta1
kind : Deployment
metadata :
  name : old - nginx
spec :
  replicas : 2
  selector :
    matchLabel s :
      run : old - nginx
  template :
    metadata :
      labels :
        run : old - nginx
    spec :
      containers :
        - image : registry . cn - hangzhou . aliyuncs . com / xianlu
          / old - nginx
          imagePullP olicy : Always
          name : old - nginx
          ports :
            - containerP ort : 80
              protocol : TCP
          restartPol icy : Always
  ---
apiVersion : v1
kind : Service
  
```

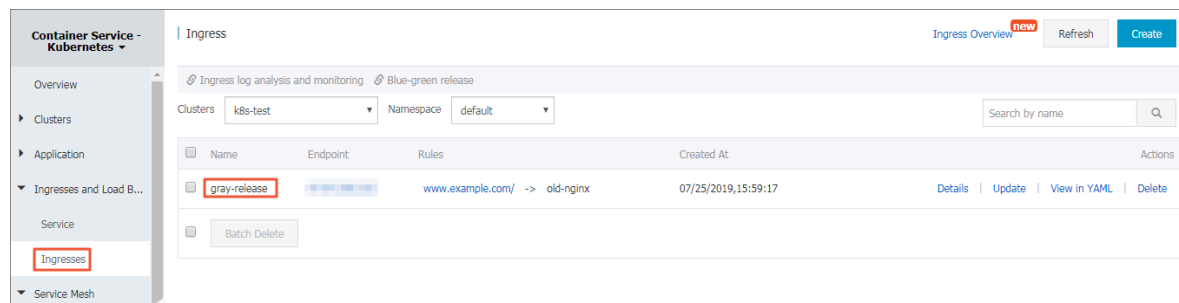
```

metadata :
  name : old - nginx
spec :
  ports :
    - port : 80
      protocol : TCP
      targetPort : 80
  selector :
    run : old - nginx
  sessionAffinity : None
  type : NodePort
---
apiVersion : extensions / v1beta1
kind : Ingress
metadata :
  name : gray - release
spec :
  rules :
    - host : www . example . com
      http :
        paths :
          # The service of an earlier version .
          - path : /
            backend :
              serviceName : old - nginx
              servicePort : 80

```

5. In the left-side navigation pane under Container Service-Kubernetes, choose Ingresses and Load Balancing > Ingresses.

You can see that the virtual host name points to old-nginx.



6. Log on to the Master node and run the curl command to view the Ingress.

```
curl -H "Host : www . example . com " http ://< EXTERNAL_IP >
```



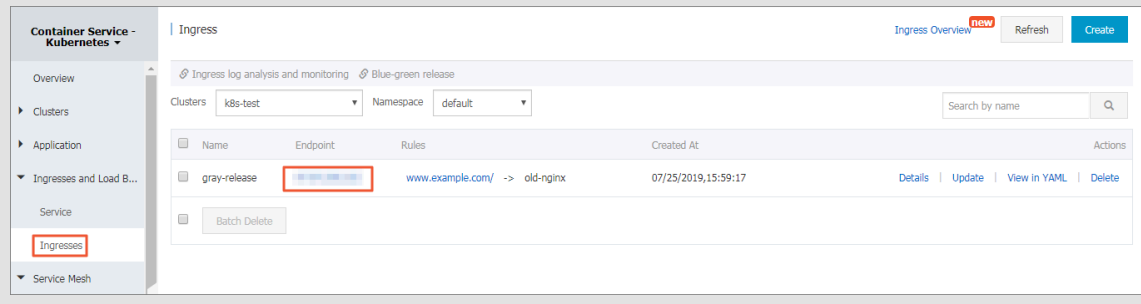
Note:

You can obtain the value of <EXTERNAL_IP> by using either of the following two methods:

- Run the following command:

```
kubectl get ingress
```

- In the left-side navigation pane under Container Service-Kubernetes, choose Ingresses and Load Balancing > Ingresses to view the endpoint information of the target Ingress.



```
ubuntu-mia@ubuntumia-VirtualBox:~$ curl -H "Host: www.example.com" http://
old
ubuntu-mia@ubuntumia-VirtualBox:~$ curl -H "Host: www.example.com" http://
old
ubuntu-mia@ubuntumia-VirtualBox:~$ curl -H "Host: www.example.com" http://
old
ubuntu-mia@ubuntumia-VirtualBox:~$ curl -H "Host: www.example.com" http://
old
```

4.2.5 Step 2: Release the latest version of a service

This topic describes how to release the latest version of a service by using the Ingress function provided by Alibaba Cloud Container Service for Kubernetes.

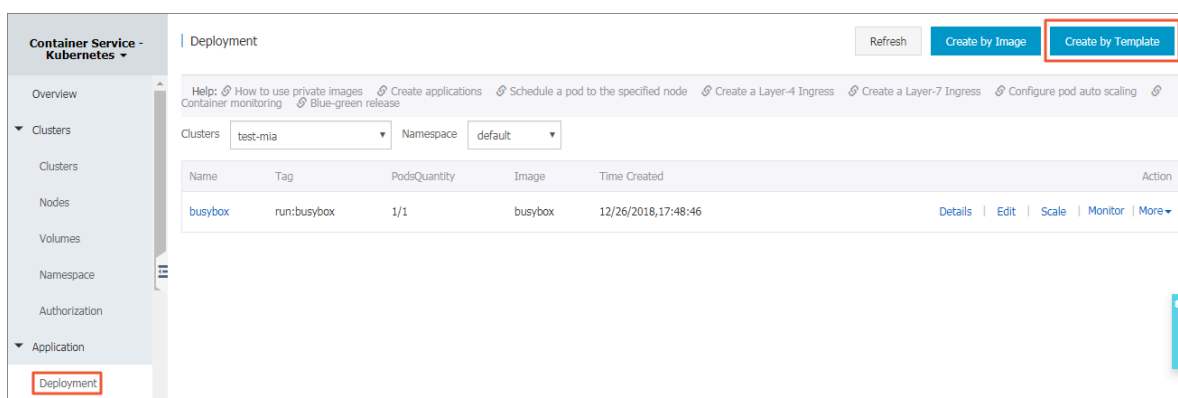
Prerequisites

- You have created a Kubernetes cluster. For more information, see [#unique_11](#).
- You have connected to the Kubernetes cluster by using kubectl. For more information, see [#unique_12](#).

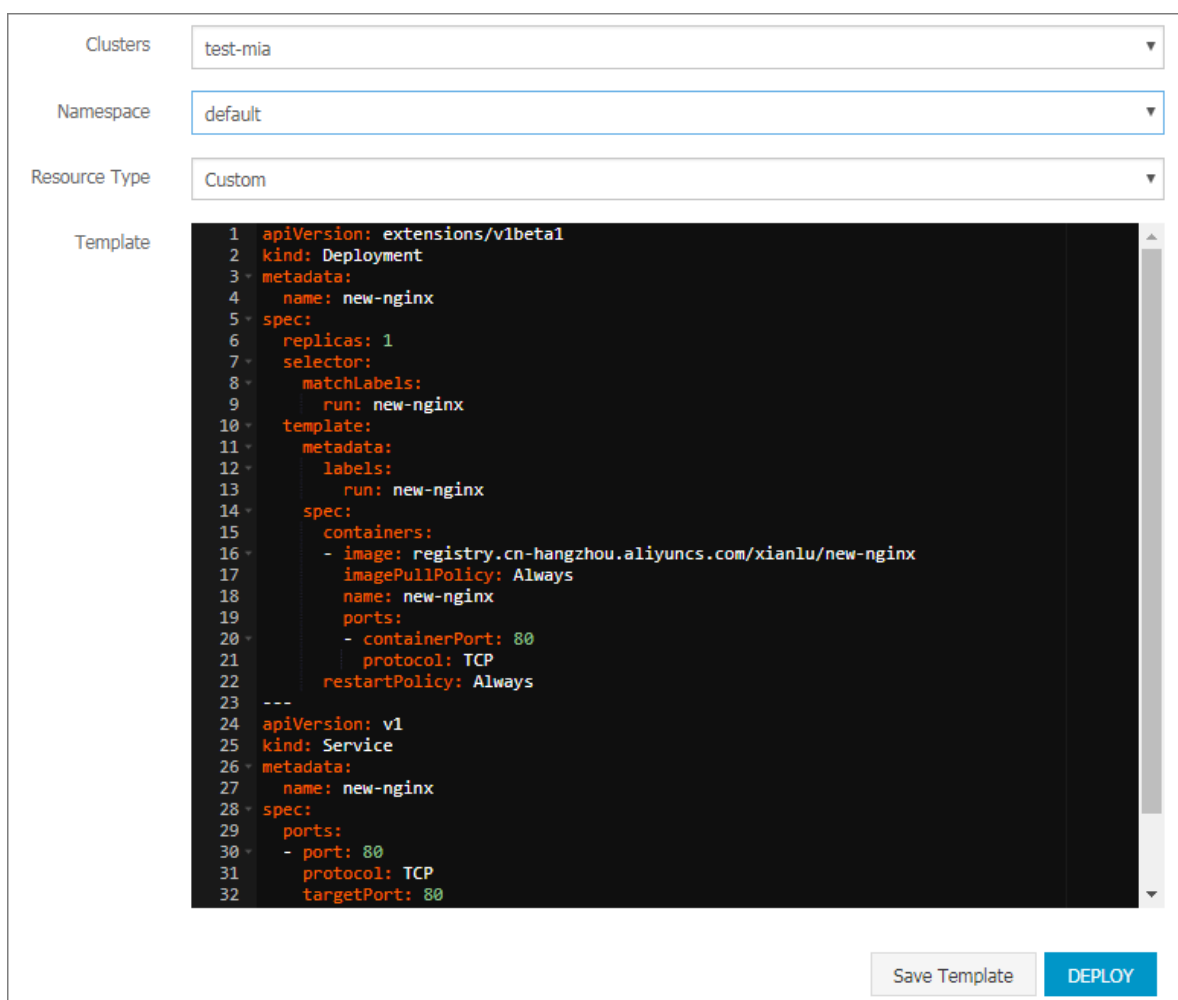
Procedure

1. Log on to the [Container Service console](#).
2. In the left-side navigation pane under Kubernetes, choose Application > Deployment.

3. In the upper-right corner, click Create by Template.



4. Select the target cluster and namespace, select a sample template or customize a template, and then click DEPLOY.



Deploy an Nginx application of the latest version that contains the required deployment, the target service, and an Ingress. The orchestration template that contains the deployment and service is as follows:

```

apiVersion : extensions / v1beta1
kind : Deployment
  
```

```

metadata :
  name : new - nginx
spec :
  replicas : 1
  selector :
    matchLabels :
      run : new - nginx
  template :
    metadata :
      labels :
        run : new - nginx
    spec :
      containers :
        - image : registry . cn - hangzhou . aliyuncs . com / xianlu
          / new - nginx
          imagePullPolicy : Always
          name : new - nginx
          ports :
            - containerPort : 80
              protocol : TCP
          restartPolicy : Always
---
apiVersion : v1
kind : Service
metadata :
  name : new - nginx
spec :
  ports :
    - port : 80
      protocol : TCP
      targetPort : 80
  selector :
    run : new - nginx
  sessionAffinity : None
  type : NodePort

```

The following are Ingress orchestration templates of different annotation settings:



Note:

If you do not set `service - match` or `service - weight` in the annotations field of an Ingress template, the Ingress controller forwards client requests evenly to the latest and earlier services in a random manner.

- Ingress template used to specify only the client requests that meet the requirement of the regular expression `foo=bar` to be routed to the latest version of the service

```

apiVersion : extensions / v1beta1
kind : Ingress
metadata :
  name : gray - release
  annotations :
    nginx . ingress . kubernetes . io / service - match : |
      # Only if the request header of a request meets
      the requirements of the regular expression foo =
      bar , can the request be routed to the new - nginx
      service .

```

```

        new - nginx : header (" foo ", /^ bar $/)
spec :
  rules :
  - host : www . example . com
    http :
      paths :
        # Earlier version of the service
        - path : /
          backend :
            serviceNam e : old - nginx
            servicePor t : 80
        # Latest version of the service
        - path : /
          backend :
            serviceNam e : new - nginx
            servicePor t : 80

```

- Ingress template used to specify the proportion of requests that can be routed to the latest version of the service



Note:

In this example, the latest version of the service and the earlier version version of the service are weighted at 50% each.

```

apiVersion : extensions / v1beta1
kind : Ingress
metadata :
  name : gray - release
  annotation s :
    nginx . ingress . kubernetes . io / service - weight : |
# Set 50 % of traffic to be routed to the new -
nginx service .
    new - nginx : 50 , old - nginx : 50
spec :
  rules :
  - host : www . example . com
    http :
      paths :
        # Earlier version of the service
        - path : /
          backend :
            serviceNam e : old - nginx
            servicePor t : 80
        # Latest version of the service
        - path : /
          backend :
            serviceNam e : new - nginx
            servicePor t : 80

```

- Ingress template used to specify that only 50% of the client request traffic that meets the requirements of foo=bar will be routed the latest version of the service

```

apiVersion : extensions / v1beta1
kind : Ingress
metadata :
  name : gray - release

```

```

    annotation s :
      nginx . ingress . kubernetes . io / service - match : | #
      Only if the request header of a request meets
      the requirements of the regular expression foo =
      bar , can the request be routed to the new - nginx
      service .
      new - nginx : header ( " foo " , / ^ bar $ / )
      nginx . ingress . kubernetes . io / service - weight : | #
      Only 50 % of the client request traffic that meets
      the requirements of the preceding matching rule
      can be routed to the new - nginx service .
      new - nginx : 50 , old - nginx : 50
  spec :
    rules :
      - host : www . example . com
      http :
        paths :
          # Earlier version of the service
          - path : /
            backend :
              serviceNam e : old - nginx
              servicePor t : 80
          # Latest version of the service
          - path : /
            backend :
              serviceNam e : new - nginx
              servicePor t : 80

```

5. In the left-side navigation pane, choose **Application > Ingress**.

You can see that the virtual host name points to old-nginx.

Name	Endpoint	Rule	Time Created	Action
gray-release		www.example.com/ -> old-nginx	12/29/2018,10:51:33	Details Update View YAML Delete
gray-release-01		www.example.com-01/ -> old-nginx www.example.com-01/ -> new-nginx-01	12/29/2018,10:54:23	Details Update View YAML Delete
gray-release-02		www.example.com-02/ -> old-nginx www.example.com-02/ -> new-nginx-02	12/29/2018,10:56:01	Details Update View YAML Delete
gray-release-03		www.example.com-03/ -> old-nginx www.example.com-03/ -> new-nginx-03	12/29/2018,10:57:55	Details Update View YAML Delete

6. Log on to the Master node and run the following curl commands to view the Ingress access of the following settings:

- Only the client requests that meet the requirements of the regular expression `foo=bar` can be routed to the latest version of the service.

```
# curl -H "Host : www . example . com " -H " foo : bar "
http ://< EXTERNAL_I P >
```

```
ubuntu-mia@ubuntu-mia-VirtualBox:~$ curl -H "Host: www.example1.com" -H "foo: bar" http://
new
ubuntu-mia@ubuntu-mia-VirtualBox:~$ curl -H "Host: www.example1.com" -H "foo: bar" http://
new
ubuntu-mia@ubuntu-mia-VirtualBox:~$ curl -H "Host: www.example1.com" -H "foo: bar" http://
new
ubuntu-mia@ubuntu-mia-VirtualBox:~$ curl -H "Host: www.example1.com" -H "foo: bar" http://
new
```

- Requests of a specified proportion can be routed to the latest version of the service.

```
# curl -H "Host : www . example . com " http ://<
EXTERNAL_I P >
```

```
ubuntu-mia@ubuntu-mia-VirtualBox:~$ curl -H "Host: www.example2.com" http://
new
ubuntu-mia@ubuntu-mia-VirtualBox:~$ curl -H "Host: www.example2.com" http://
old
ubuntu-mia@ubuntu-mia-VirtualBox:~$ curl -H "Host: www.example2.com" http://
new
ubuntu-mia@ubuntu-mia-VirtualBox:~$ curl -H "Host: www.example2.com" http://
old
```

- Only 50% of the client request traffic that meets the requirements of the regular express `foo=bar` can be routed to the latest version of the service.

```
# curl -H "Host : www . example . com " -H " foo : bar "
http ://< EXTERNAL_I P >
```

```
ubuntu-mia@ubuntu-mia-VirtualBox:~$ curl -H "Host: www.example3.com" -H "foo: bar" http://
old
ubuntu-mia@ubuntu-mia-VirtualBox:~$ curl -H "Host: www.example3.com" -H "foo: bar" http://
new
ubuntu-mia@ubuntu-mia-VirtualBox:~$ curl -H "Host: www.example3.com" -H "foo: bar" http://
old
ubuntu-mia@ubuntu-mia-VirtualBox:~$ curl -H "Host: www.example3.com" -H "foo: bar" http://
new
```

4.2.6 Step 3: Remove the earlier version of a service

This topic describes how to remove the earlier version of a service when the latest version of the service (which has been released through a gray release) has run without exceptions for a specified period of time.

Prerequisites

- You have created a Kubernetes cluster. For more information, see [#unique_11](#).
- You have connected to the Kubernetes cluster by using `kubectl`, see [#unique_12](#).

- You have deployed an earlier version of the service. For more information, see [#unique_43](#). You have also released a later version of the service through a gray release. For more information, see [#unique_44](#).

Run a command

1. Run the following command to edit the YAML file deployed by [#unique_44](#) to remove the earlier version of the service:



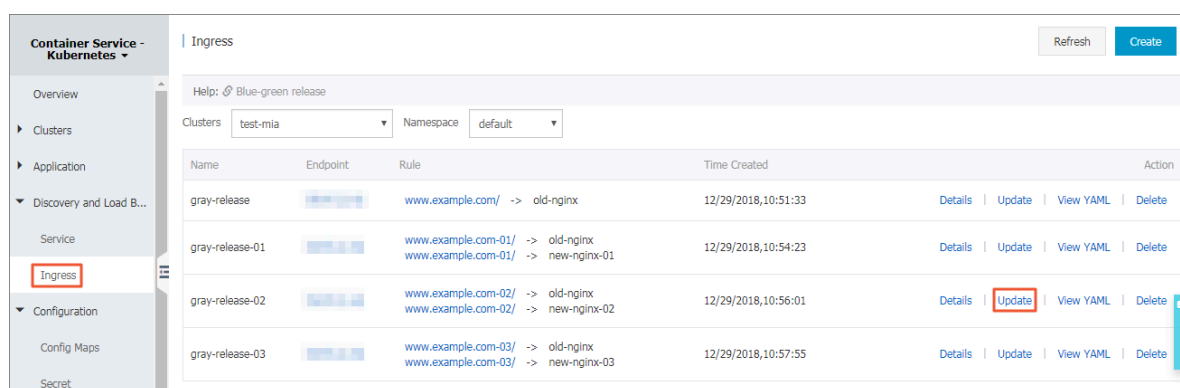
Note:

You need to remove the `annotation s` field.

```
$ kubectl get ingress gray - release - 02
```

Use the Container Service console

1. Log on to the [Container Service console](#).
2. In the left-side navigation pane under Kubernetes, choose **Application > Ingress**.
3. Select the target cluster and namespace, select the target Ingress, and click **Update** in the action column.



4. In the displayed dialog box, modify the Ingress as follows:

a. In the Rule > Service area, remove the earlier version of the service rule.

Update

Name: gray-release-02

Rule:

+ Add

Domain

www.example.com-02

Select *

Custom

or

path

/

Service

+ Add

Name	Port	Weight	Percent of Weight
old-nginx	80	100	50.0%
new-nginx-02	80	100	50.0%

☐ EnableTLS

Service weight:

☒ Enable

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.

annotation:

+ Add

rewrite annotation

Tag:

+ Add

Update

Cancel

b. Click Update.

Result

1. Return to the Ingress page. Here, you can see that only one Ingress rule points to the new-nginx service.

Name	Endpoint	Rule	Time Created	Action
gray-release		www.example.com/ -> old-nginx	12/29/2018,10:51:33	Details Update View YAML Delete
gray-release-01		www.example.com-01/ -> old-nginx www.example.com-01/ -> new-nginx-01	12/29/2018,10:54:23	Details Update View YAML Delete
gray-release-02		www.example.com-02/ -> new-nginx-02	12/29/2018,11:20:32	Details Update View YAML Delete
gray-release-03		www.example.com-03/ -> old-nginx www.example.com-03/ -> new-nginx-03	12/29/2018,10:57:55	Details Update View YAML Delete

2. Log on to the Master node and run the curl command to view the Ingress access.

```
$ curl -H "Host : www . example2 . com " http ://<
EXTERNAL_I P >
```

```
ubuntu-mia@ubuntumia-VirtualBox:~$ curl -H "Host: www.example2.com" http://
new
ubuntu-mia@ubuntumia-VirtualBox:~$ curl -H "Host: www.example2.com" http://
new
ubuntu-mia@ubuntumia-VirtualBox:~$ curl -H "Host: www.example2.com" http://
new
ubuntu-mia@ubuntumia-VirtualBox:~$ curl -H "Host: www.example2.com" http://
new
```

Now, all requests are routed to the latest version of the service, which means you have completed the gray release deployment cycle. You can also remove the deployment and service of the earlier version.

4.3 Application

5 Istio

5.1 Use Istio to implement intelligent routing in Kubernetes

Alibaba Cloud Container Service for Kubernetes supports one-click deployment of Istio and multiple functions expanded on Istio. This topic describes how to implement intelligent routing through Istio. For information about Istio official documents, see [Intelligent Routing](#).

Prerequisites

- You have created a Kubernetes cluster. For more information, see [#unique_11](#).
- You have deployed Istio. For more information, see [#unique_48](#).



Note:

Istio used in this topic is V 1.0.2.

- You have a local Linux environment in which you have configured the kubectl tool and used the tool to connect to the cluster. For more information, see [#unique_12](#).
- You have downloaded the project code of an Istio version and run the relevant commands in the Istio file directory. See <https://github.com/istio/istio/releases>.

Install the Istio official sample application

Install the Istio official sample application, Bookinfo. For more information, see <https://istio.io/docs/guides/bookinfo>.

Quickly deploy the Bookinfo sample application

1. Label the `default` namespace with the `istio - injection = enabled` tag.



Note:

Kubernetes clusters running on Alibaba Cloud Container Service support one-click deployment of Istio and automatic sidecar injection.

```
$ kubectl label namespace default istio-injection=enabled
```

2. Run the following `kubectl` command to deploy the Bookinfo sample application:

```
$ kubectl apply -f samples/bookinfo/platform/kube/bookinfo.yaml
```

The preceding command starts all four microservices. All three `Reviews` service versions (v1, v2, and v3) are also started.

3. Run the following command to verify that all services and pods are properly defined and started:

```
$ kubectl get svc, pods
```

4. You need to access the application from the outside of your Kubernetes cluster, for example, a browser. You need to create an [Istio Gateway](#). Define the ingress gateway for the application.

```
$ kubectl apply -f samples/bookinfo/networking/bookinfo-gateway.yaml
```

Run the following command to verify that the gateway has been created:

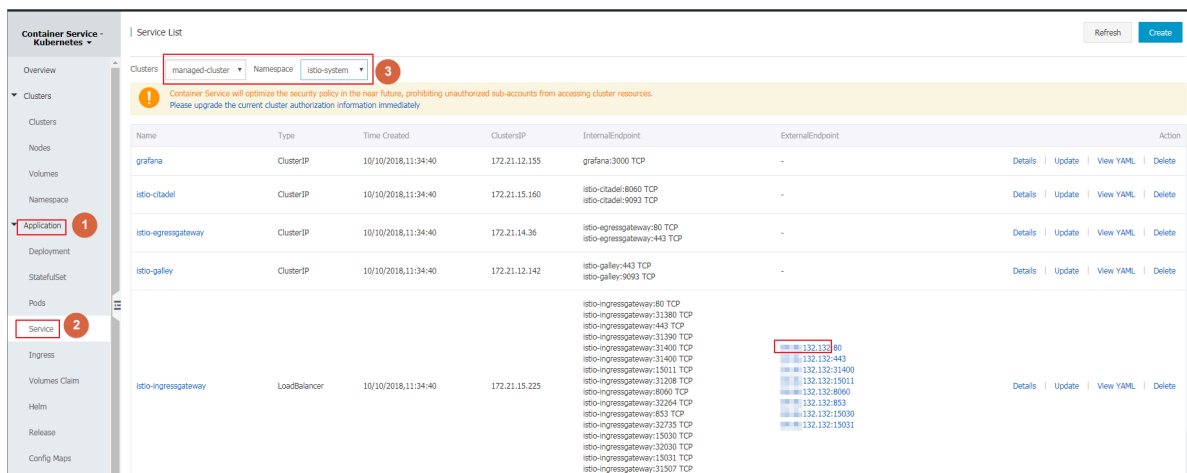
```
$ kubectl get gateway
NAME          AGE
```

```
bookinfo - gateway 32s
```

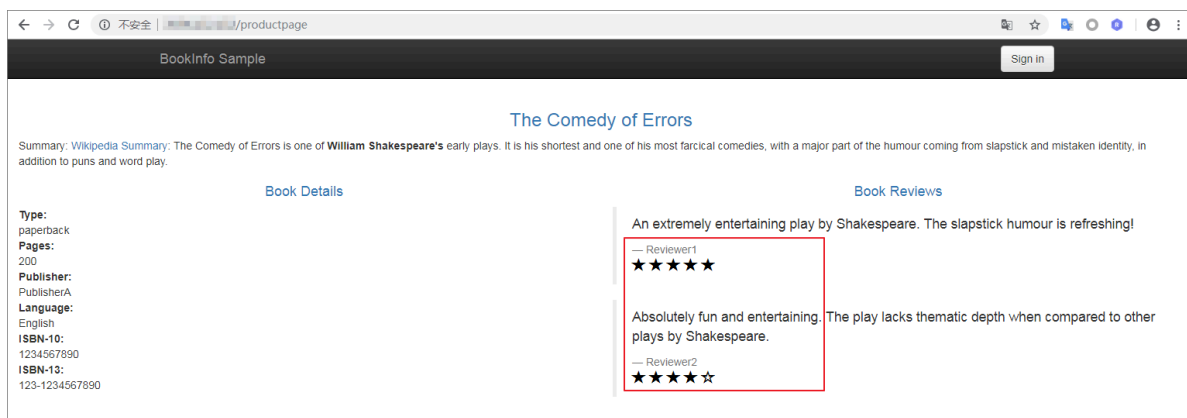
5. Run the following command to check the IP address of `istio - ingressgateway`.

```
$ kubectl get svc istio - ingressgateway -n istio - system
```

You can also log on to the Container Service console to view the IP address of `istio - ingressgateway`. Specifically, choose Application > Service in the left-side navigation pane, select the target cluster and the Istio-system namespace.



6. Access the BookInfo home page. The access address is `http://{EXTERNAL-IP}/productpage`.



If you refresh the browser, different versions of the reviews are displayed on the productpage in a round-robin manner (starting with a red star, to a black star, to no star). This indicates that Istio is currently not being used to control the version routing.

Set a route for requests

You need to set a default route because three Reviews service versions are deployed for the BookInfo sample application. Otherwise, if you access the application multiple times, you will notice that sometimes the book review output contains star ratings and other times it does not. This is because you have not set a default route for the rating service versions, and Istio then randomly routes requests to all available versions in a round robin fashion.

You need to define available versions in the destination routing rule before using Istio to control the route to the service versions of the BookInfo application.

Create the default destination routing rule for the BookInfo service.

- If you do not want to enable bidirectional TLS, run the following command:

```
$ kubectl apply -f samples / bookinfo / networking /  
destination - rule - all . yaml
```

- If you want to enable bidirectional TLS, run the following command:

```
$ kubectl apply -f samples / bookinfo / networking /  
destination - rule - all - mTLS . yaml
```

Wait for a few seconds until the destination routing rule takes effect. Run the following command to view the destination routing rule:

```
$ kubectl get destination nrules - o yaml
```

Set the default version of all microservices to v1

Run the following command to set the default version of all microservices to v1:

```
$ kubectl apply -f samples / bookinfo / networking / virtual -  
service - all - v1 . yaml
```

Run the following command to display all the created routing rules:

```
kubectl get virtualservice - o yaml
```

It takes a period of time for the routing rule to be synchronized to all pods because the routing rule is distributed to the proxy in an asynchronized manner. Therefore, we recommend that you wait for a few seconds before accessing the application.

Open the URL of the Bookinfo application in your browser: `http://{EXTERNAL - IP }/ productpage`.

On the product page of the BookInfo application, the displayed content does not contain the reviews with stars. This is because the reviews:v1 service does not access the ratings service.

BookInfo Sample

Sign in

The Comedy of Errors

Summary: [Wikipedia Summary](#): The Comedy of Errors is one of **William Shakespeare's** early plays. It is his shortest and one of his most farcical comedies, with a major part of the humour coming from slapstick and mistaken identity, in addition to puns and word play.

Book Details

Type:
paperback

Pages:
200

Publisher:
PublisherA

Language:
English

ISBN-10:
1234567890

ISBN-13:
123-1234567890

Book Reviews

An extremely entertaining play by Shakespeare. The slapstick humour is refreshing!

— Reviewer1

Absolutely fun and entertaining. The play lacks thematic depth when compared to other plays by Shakespeare.

— Reviewer2

Route the requests from a specific user to reviews:v2

Run the following command to route requests from the test user named jason to reviews:v2 to enable the ratings service:

```
$ kubectl apply -f samples/bookinfo/networking/virtual-service-reviews-test-v2.yaml
```

Run the following command to check whether routing rules are created:

```
$ kubectl get virtualservice reviews -o yaml
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: reviews
...
spec:
  hosts:
  - reviews
  http:
  - match:
    - headers:
        end-user:
          exact: jason
      route:
      - destination:
          host: reviews
          subset: v2
    - route:
      - destination:
          host: reviews
```

```
subset : v1
```

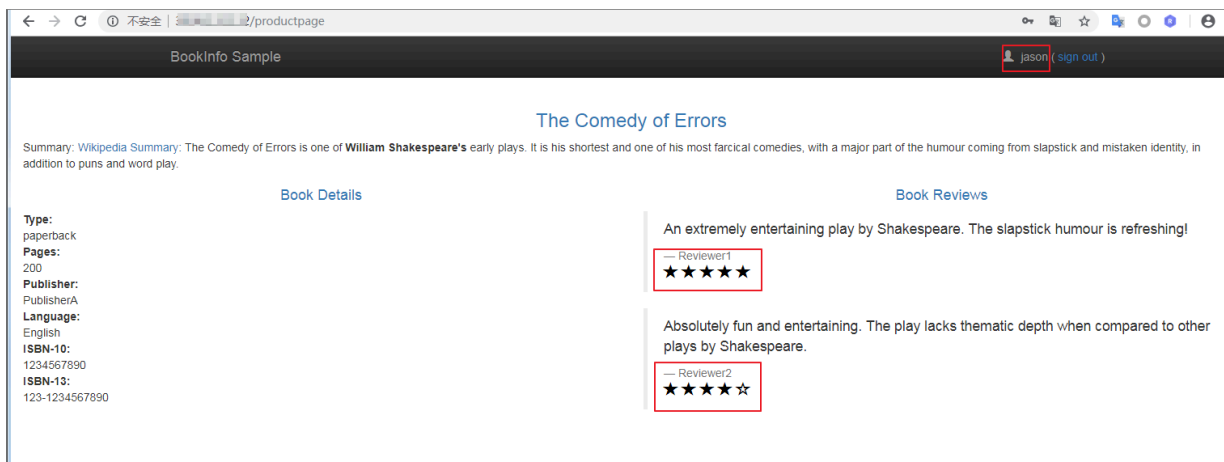
After you confirm that the routing rule is created, open the URL of the BookInfo application in your browser: `http://{EXTERNAL-IP}/productpage`.

Log on to the product page as the jason user to verify that the rating information is displayed under each review record.



Note:

Both the logon account name and password for are `jason` ..



Note:

In this example, two request routing rules have been changed. Firstly, all requests are routed to the v1 version of the Reviews service provided by the BookInfo application. Then, a new routing rule is set to route specific requests to the v2 version of the Reviews service according to the header of a request (for example, the user cookie).

Inject faults

To test the resiliency of the microservices application, namely, BookInfo, inject a 7-second delay between the `reviews : v2` microservices and the `ratings` microservices for the jason user. Note that the reviews:v2 service has a 10-second hard-coded connection timeout for calls to the ratings service. Therefore, you can still expect the end-to-end flow to continue without any errors even you have set the 7-second delay .

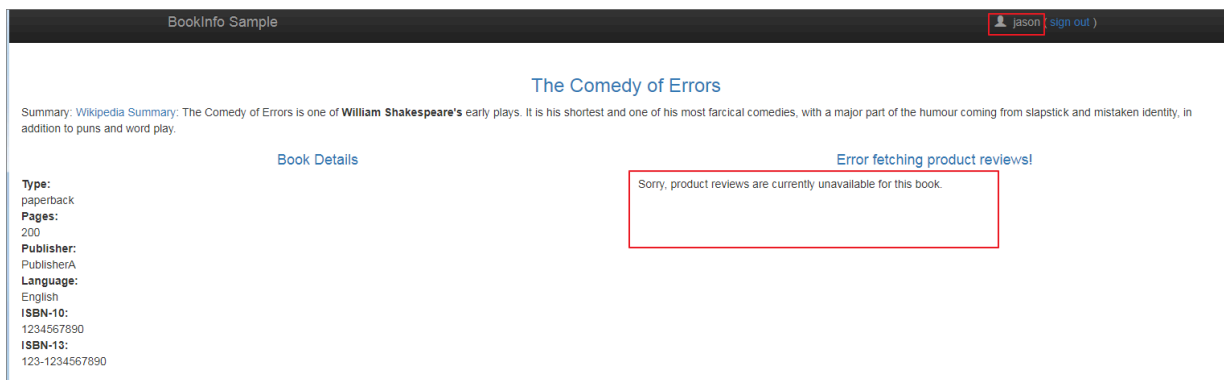
Inject an HTTP delay fault

Create a fault injection rule to delay traffic coming from the jason user.

```
$ kubectl apply -f samples / bookinfo / networking / virtual -  
service - ratings - test - delay . yaml
```

After you confirm that the rule is created, open the URL of the BookInfo application in your browser: `http://{EXTERNAL-IP}/productpage`.

Log on to the productpage as the jason user to view the following.



Note:

The reviews service fails because the timeout between the productpage and reviews services is shorter than the timeout between the reviews and ratings services, that is, (3 seconds + 1 retry = 6 seconds) is shorter than 10 seconds. Bugs like this can occur in typical enterprise applications where different teams develop different microservices independently. Istio's fault injection rules help you identify such anomalies without impacting end users.

Inject an HTTP abort fault

Create a fault injection rule to send an HTTP abort

```
$ kubectl apply -f samples / bookinfo / networking / virtual -  
service - ratings - test - abort . yaml
```

After you confirm that the rule is created, open the URL of the BookInfo application in your browser: `http://{EXTERNAL-IP}/productpage`.

Log on to the productpage as the jason user to view the following.

Bookinfo Sample jason (sign out)

The Comedy of Errors

Summary: [Wikipedia Summary](#): The Comedy of Errors is one of **William Shakespeare's** early plays. It is his shortest and one of his most farcical comedies, with a major part of the humour coming from slapstick and mistaken identity, in addition to puns and word play.

Book Details

Type:
paperback
Pages:
200
Publisher:
PublisherA
Language:
English
ISBN-10:
1234567890
ISBN-13:
123-1234567890

Book Reviews

An extremely entertaining play by Shakespeare. The slapstick humour is refreshing!

— Reviewer1

Ratings service is currently unavailable

Absolutely fun and entertaining. The play lacks thematic depth when compared to other plays by Shakespeare.

— Reviewer2

Ratings service is currently unavailable

Migrate traffic

In addition to the content-based routing rule, Istio also supports the weight-based routing rule.

Run the following command to route all traffic to the v1 version of all microservices.

```
$ kubectl replace -f samples / bookinfo / networking / virtual
- service - all - v1 . yaml
```

Run the following command to route 50% of traffic from the reviews v1 service to the reviews v3 service:

```
$ kubectl replace -f samples / bookinfo / networking / virtual
- service - reviews - 50 - v3 . yaml
```

Refresh the productpage for multiple times in the browser. You have a 50% probability to see the review content marked with red stars on the page.



Note:

Note that this method is completely different from using the deployment feature of the container orchestration platform for version migration. The container orchestration platform uses the instance scaling method to manage the traffic. With Istio, two versions of the reviews service can expand and shrink capacity independently, without affecting the distribution of traffic between the two versions of services.

Assuming you decide that the `reviews : v3` microservice is stable, you can route 100% of the traffic to `reviews : v3` to implement a gray release by running the following command:

```
$ kubectl replace -f samples / bookinfo / networking / virtual  
- service - reviews - v3 . yaml
```

Conclusion

You can use Alibaba Cloud Container Service for Kubernetes to quickly build the open platform, that is, Istio, to connect, manage, and secure microservices, and to introduce and configure multiple relevant services for applications. This topic uses a sample application from Istio to detail how to use Istio functions such as traffic routing, fault injection, and traffic migrating. We recommend that you use Alibaba Cloud Container Service for Kubernetes to quickly build Istio, an open management platform for microservices, and integrate Istio with the microservice development of your project.

5.2 Implement Istio distributed tracing on Kubernetes

This topic details an example about how to use the distributed tracing system Jaeger in an Istio-enabled application.

When applications are transformed from the monolithic architecture to the microservices architecture, traditional monitoring tools cannot provide visibility across different services. This means that a distributed tracing tool must be implemented.

The OpenTracing specification has been created to address API incompatibility between different distributed tracing systems. Designed to be a lightweight, standardized layer, OpenTracing is located between application/class libraries and tracing or log analysis programs. OpenTracing has joined the Cloud Native Computing Foundation (CNCF) to provide unified concepts and data standards for global distributed tracing systems. It provides platform-independent and vendor-independent APIs. With OpenTracing, developers can easily add or replace tracing systems.

Jaeger is an open source distributed tracing system released by [CNCF](#), and is compatible with OpenTracing APIs.

Prerequisites

- You have created a Kubernetes cluster. For more information, see [#unique_11](#).
- You have a local Linux environment in which you have configured the kubectl tool and used the tool to connect to the cluster. For more information, see [#unique_12](#).
- You have downloaded the project code of an Istio version and run the relevant commands in the Istio file directory. See [Istio releases](#).

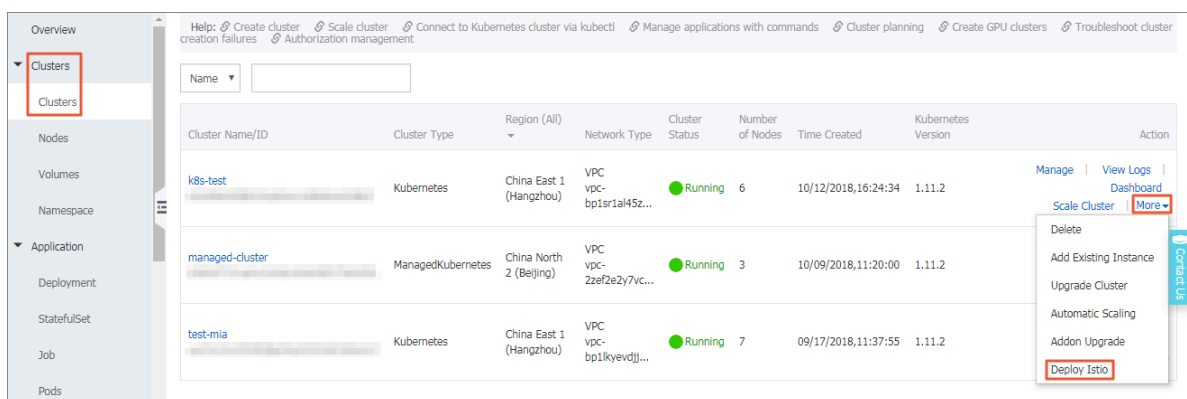
Step 1: Deploy the Jaeger tracing system in one click

1. Log on to the [Container Service console](#).
2. In the left-side navigation pane, choose Clusters. Select a cluster and choose More > Deploy Istio.



Note:

This examples uses Istio V1.1. For more information, see [#unique_48](#).



3. On the Istio deployment page, select Activate Tracing Analysis. With Alibaba Cloud Container Service, you can deploy the Jaeger tracing system in one click in the Web interface.
4. After you complete the deployment, choose Ingresses and Load Balancing > Service in the left-side navigation tree, and select the target cluster and namespace to find the tracing service.
5. If the service type of tracing service is set as LoadBalancer, click the external endpoint of the tracing service to display the Jaeger UI page. If the service type of

tracing service is set as ClusterIP, you can access the service through port-forward or Ingress.

You can use Jaeger to observe the traces between services, diagnose performance, and analyze system faults.

Step 2: Deploy a Bookinfo sample application of the distributed tracing service

1. Label the `default` namespace with `istio - injection = enabled`.



Note:

Kubernetes clusters of Alibaba Cloud Container Service support one-click Istio deployment and automatic sidecar injection. For more information, see [Bookinfo Application](#).

```
$ kubectl label namespace default istio - injection = enabled
```

2. Use the following `kubectl` command to deploy a Bookinfo sample application:

```
$ kubectl apply -f samples / bookinfo / platform / kube / bookinfo . yaml
```

The preceding command start all the four microservices. All three `reviews` service versions (v1, v2, and v3) are also started.

3. Confirm all services and pods are properly defined and started.

```
$ kubectl get svc , pods
```

4. If you need to access the application from outside of your Kubernetes cluster, for example, from a browser, you must create an [Istio Gateway](#) to define the ingress gateway for the application.

```
$ kubectl apply -f samples / bookinfo / networking / bookinfo - gateway . yaml
```

Verify that the gateway has been created:

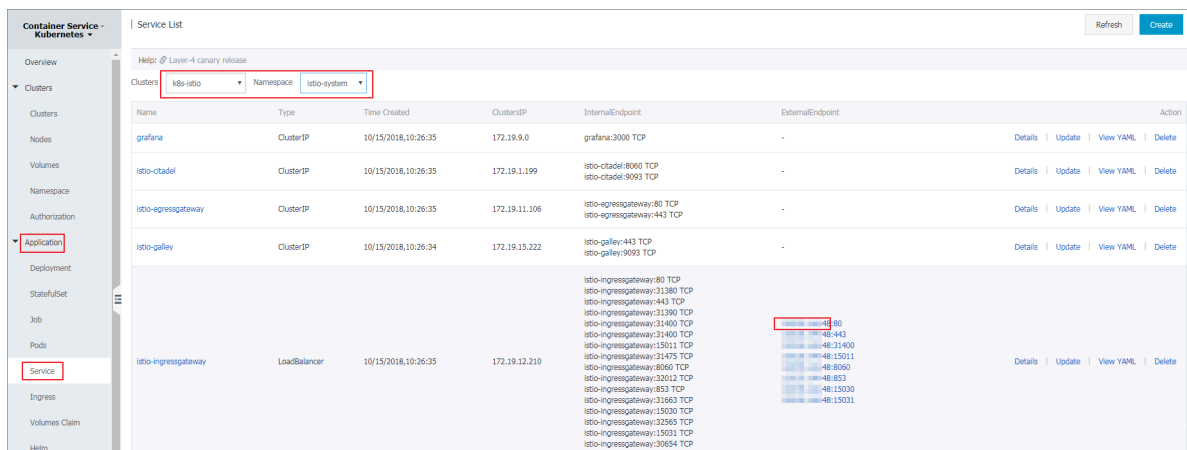
```
$ kubectl get gateway  
NAME AGE
```

```
bookinfo - gateway 32s
```

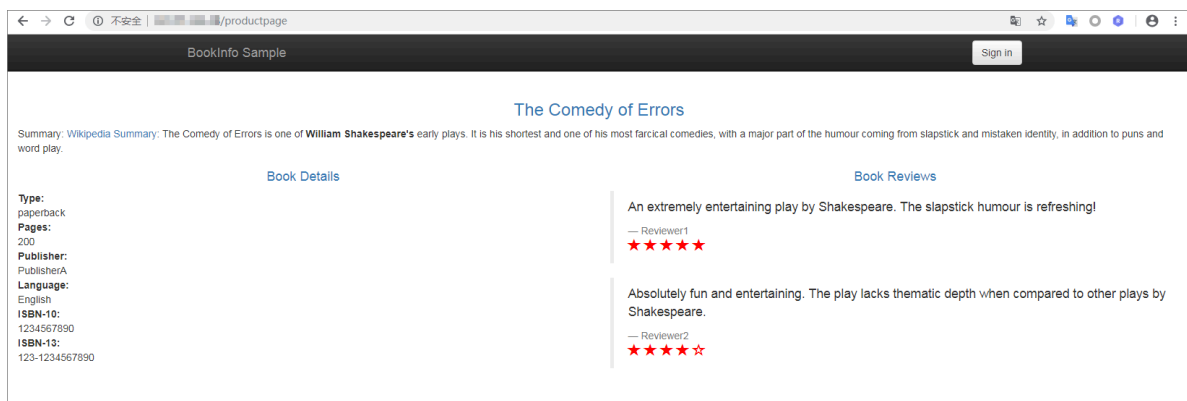
5. Confirm the IP address of `istio - ingressgateway`. You can quickly query the IP address by running the following command:

```
$ kubectl get svc istio - ingressgateway -n istio - system
```

You can also view the IP address of `istio - ingressgateway` through the Container Service console. In the left-side navigation pane, choose Ingresses and Load Balancing > Service, and select the cluster and the Istio-system namespace.



6. Access the Bookinfo home page. The access address is `http://{EXTERNAL-IP}/productpage`.



If you refresh the browser, different versions of the reviews are displayed on the productpage in a round-robin manner (starting with a red star, to a black star, to no star). This indicates that Istio is currently not being used to control the version routing.

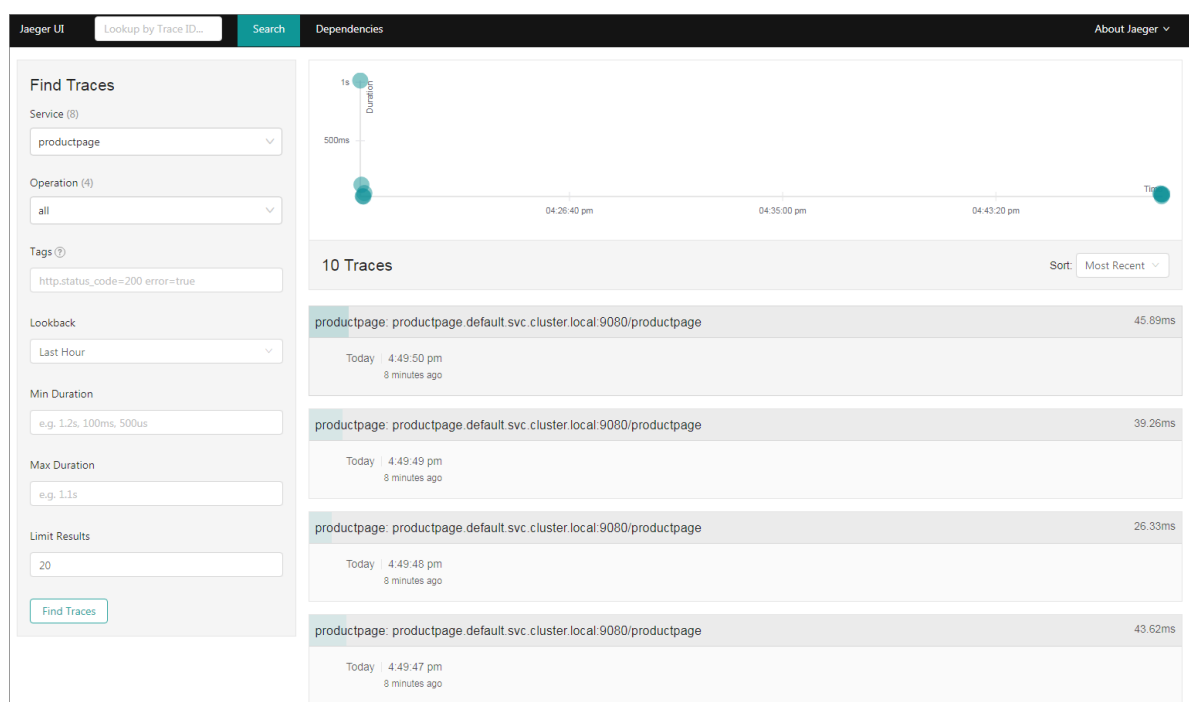
Step 3: View the tracing results displayed by Jaeger

The Jaeger UI displays the results of the distributed service tracing.

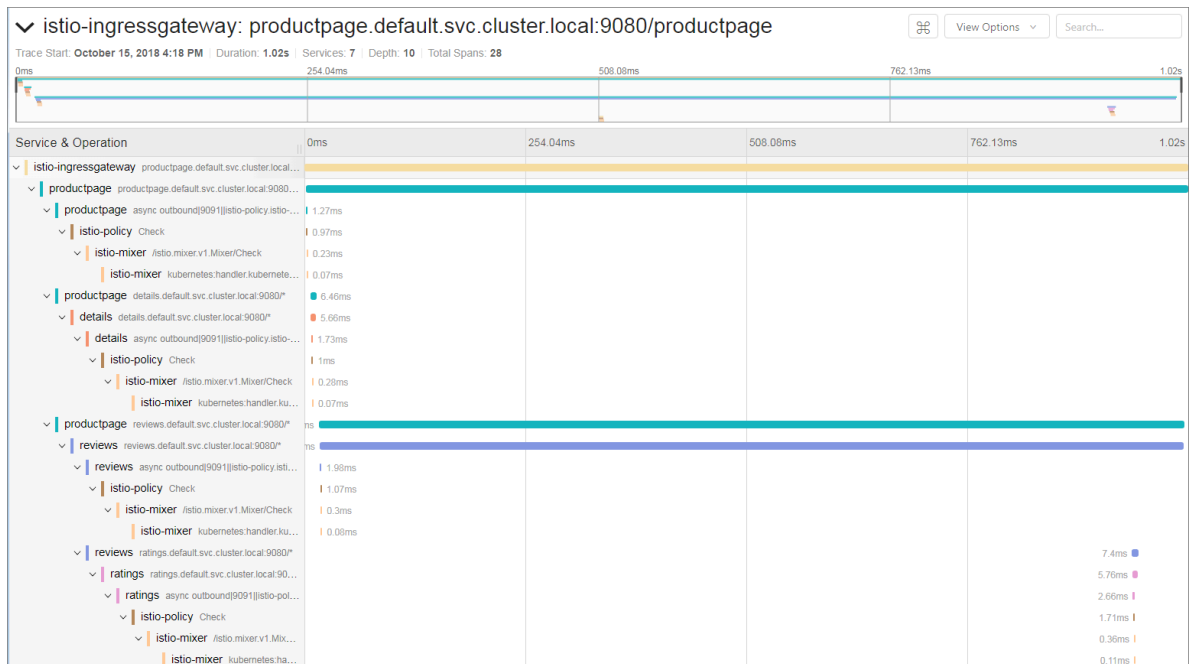
1. Click the external endpoint of the tracing-on-sls-query service to display the Jaeger UI page.
2. On the left side, select a service from the Service drop-down list, set other parameters, and click Find Traces. The scatter diagram of time and duration displayed in the upper-right corner displays the results and supports drill-down capability.

**Note:**

In this example, the Bookinfo sample application is used to view the productpage service calling.



3. You can select multiple views to visualize the tracing results, for example, a histogram within a tracing period, or the service duration within the tracing process.



Istio distributed tracing mechanism

Although Istio proxies can automatically send spans, they need identifiers to tie together the entire trace. Applications need to propagate the appropriate HTTP headers so that when the proxies send span information, the spans can be correlated correctly into a single trace.

Therefore, an application needs to collect the following headers from incoming requests and propagate the headers to any outgoing requests:

```
x - request - id
x - b3 - traceid
x - b3 - spanid
x - b3 - parentspan id
x - b3 - sampled
x - b3 - flags
x - ot - span - context
```

The productpage service (Python) extracts the required headers from an HTTP request as follows:

```
def get_forw_headers ( request ):
    headers = {}

    user_cookie = request.cookies.get("user")
    if user_cookie:
        headers['Cookie'] = 'user=' + user_cookie
```

```

incoming_headers = [ ' x - request - id ',
                      ' x - b3 - traceid ',
                      ' x - b3 - spanid ',
                      ' x - b3 - parentspan id ',
                      ' x - b3 - sampled ',
                      ' x - b3 - flags ',
                      ' x - ot - span - context '
]

for ihdr in incoming_headers :
    val = request . headers . get ( ihdr )
    if val isnotNone :
        headers [ ihdr ] = val
    # print " incoming : "+ ihdr +":"+ val
return headers

```

When you call other services in your application, these headers are propagated to form a trace.

For information about the code, see the Istio document [Distributed tracing](#).

Conclusion

You can use Alibaba Cloud Container Service for Kubernetes to quickly build an Istio open platform for connecting, managing, and securing microservices, and to introduce and configure multiple relevant services for applications. This topic uses an example to demonstrate how to use the distributed tracing system Jaeger in an Istio-enabled application. We recommend that you use Alibaba Cloud Container Service for Kubernetes to quickly build an Istio open platform of microservices and integrate Istio with microservices development in your project.

5.3 Integrate Istio with Alibaba Cloud Log Service on Kubernetes

With Alibaba Cloud Container Service for Kubernetes, you can integrate Istio with Log Service. This topic uses an official example to describe how to integrate Istio with the distributed tracing system based on Alibaba Cloud Log Service.

Prerequisites

- You have created a Kubernetes cluster. For more information, see [#unique_11](#).
- You have a local Linux environment in which you have configured the kubectl tool and connected to the cluster with kubectl. For more information, see [#unique_12](#).
- You have downloaded the project codes of an Istio version and run the relevant commands in the Istio file directory. For more information, see <https://github.com/istio/istio/releases>.

- You have deployed the test application, BookInfo. For more information, see [#unique_51/unique_51_Connect_42_section_csj_jln_cfb](#) or [Bookinfo Application](#).

Introduction to OpenTracing

The OpenTracing specification has been created to address API incompatibility between different distributed tracing systems. OpenTracing is a lightweight and standardized layer that is located between applications/class libraries and tracing or log analysis programs. OpenTracing has joined the Cloud Native Computing Foundation (CNCF) and provided unified concepts to provide data standards for global distributed tracing systems. It provides platform-independent and vendor-independent APIs. With OpenTracing, developers can easily add or replace tracing systems.

Jaeger is an open source distributed tracing system released by CNCF, and is compatible with OpenTracing APIs.

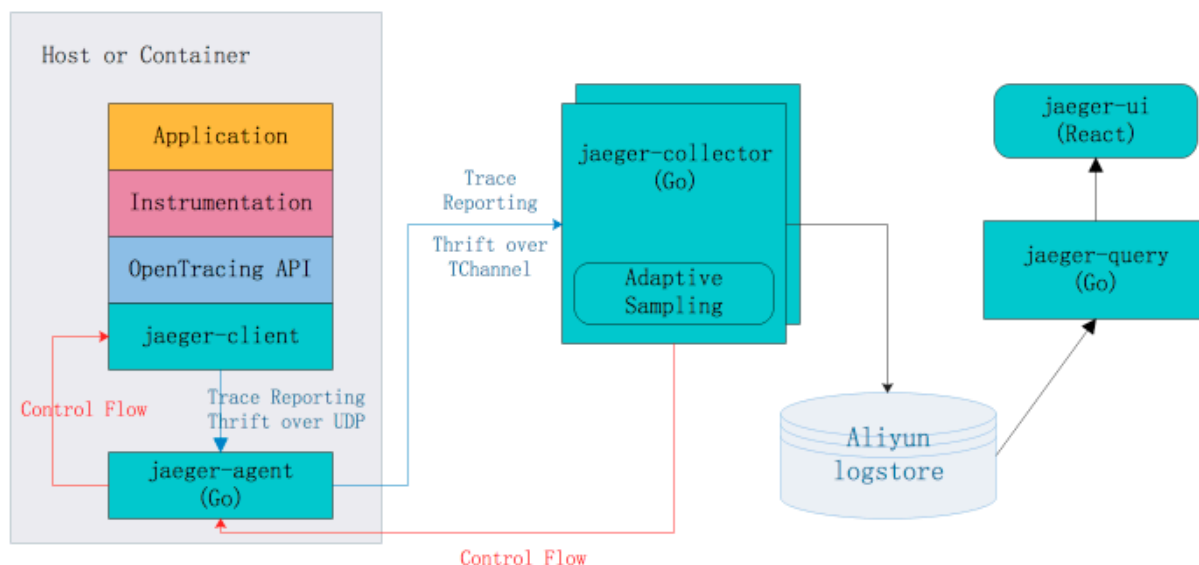
Alibaba Cloud Log Service and the distributed tracing system Jaeger

Log Service (LOG for short, formerly called SLS) is a one-stop service for real-time data. It has been used extensively by Alibaba Group in big data scenarios. Log service helps you collect, consume, ship, query, and analyze massive log data with improved processing capabilities.

Jaeger is an open source distributed tracing system released by Uber for microservice scenarios. It is used to analyze the calling process of multiple servers and display the service calling tracks with graphs. It is a powerful tool for diagnosing performance and analyzing system failures.

Jaeger on Aliyun Log Service is a distributed tracing system developed on Jaeger. With this system, you can store the collected tracing data to Alibaba Cloud Log Service permanently, and query and present the data through the native interfaces of Jaeger.

Jaeger on Aliyun Log Service



Component	Description
Jaeger client	The Jaeger client provides SDKs that conform to the OpenTracing standard for different languages. Applications write data to the client through APIs. The client library delivers the traces to the Jaeger agent according to the sampling policy specified by the applications. The tracing data is serialized by Thrift and transmitted through UDP.
Jaeger agent	The Jaeger agent is a network daemon that listens to spans sent over UDP and sends data in batches to the Jaeger collector. The Jaeger agent is designed as an infrastructure component that is deployed to all hosts. The Jaeger agent decouples the Jaeger client and Jaeger collector, and makes the collectors invisible to the client. Collectors are then no longer routed or discovered by the client.

Component	Description
Jaeger collector	The Jaeger collector receives traces from the Jaeger agent and writes the traces into the backend storage system . The backend storage is a pluggable component. Jaeger on Aliyun Log Service supports Alibaba Cloud Log Service.
Alibaba Cloud Log Service	The Jaeger collector stores the received spans to Log Service permanently. Query retrieves data from Log Service.
Query & UI	Query is a service that receives query requests, retrieves traces from the backend storage system, and hosts a UI to display traces.

Step 1: Deploy Istio with a single click through the Web interface

1. Log on to the [Container Service console](#).
2. In the left-side navigation pane, choose Clusters, select a cluster, and click More > Deploy Istio.



Note:

In this example, you need to enable Log Service (SLS) and Jaeger. For more information, see [#unique_48](#). This example uses Istio V1.0.

Container Service - Kubernetes

Cluster List

You can create up to 5 clusters and can add up to 40 nodes in each cluster. Refresh Create Kubernetes Cluster

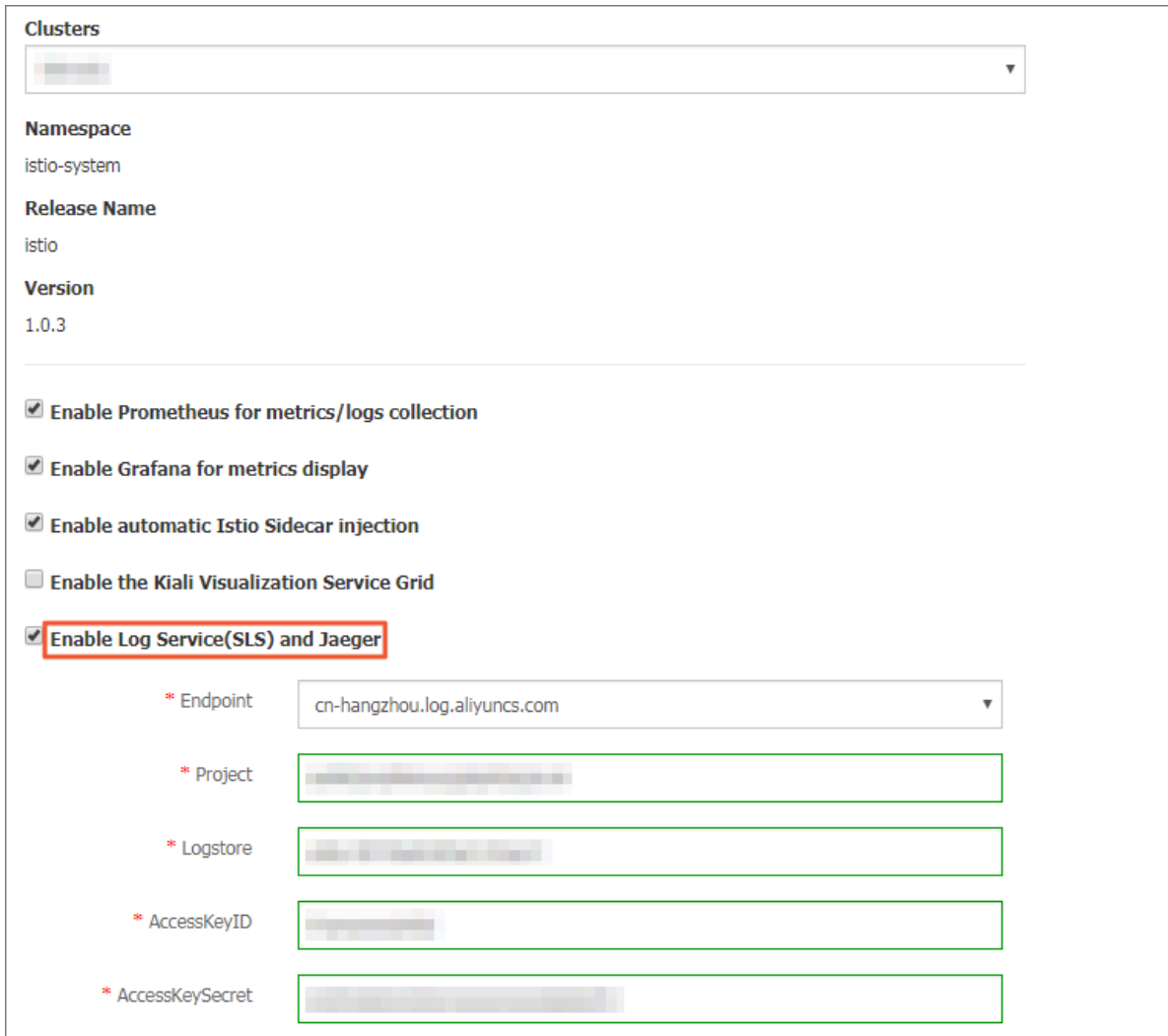
Help: Create cluster Scale cluster Connect to Kubernetes cluster via kubectl Manage applications with commands Cluster planning Create GPU clusters Troubleshoot cluster creation failures Authorization management

Name

Cluster Name/ID	Cluster Type	Region (All)	Network Type	Cluster Status	Number of Nodes	Time Created	Kubernetes Version	Action
k8s-test c9a8fedbf98d44aa6be1a46b8aa9cf862	Kubernetes	China East 1 (Hangzhou)	VPC vpc-bp1sr1ah45z...	Running	6	10/12/2018,16:24:34	1.11.2	Manage View Logs Dashboard Scale Cluster More
managed-cluster c8b9a7701ede14c0eb19e60bf37b63951	ManagedKubernetes	China North 2 (Beijing)	VPC vpc-zze2e2y7vc...	Running	3	10/09/2018,11:20:00	1.11.2	Delete Add Existing Instance Upgrade Cluster Automatic Scaling Addon Upgrade Deploy Istio
test-mia cad7e15c0784848a5be02443e9186ccb7	Kubernetes	China East 1 (Hangzhou)	VPC vpc-bp1lkyevdj...	Running	7	09/17/2018,11:37:55	1.11.2	

3. On the Istio deployment page, select Log Service(SLS) and Jaeger.

Container Service for Kubernetes is integrated with Log Service. Its distributed tracing data is saved to the Logstore of Log Service.



The screenshot shows the Istio deployment configuration interface. The 'Clusters' dropdown is set to a default value. The 'Namespace' is 'istio-system', 'Release Name' is 'istio', and 'Version' is '1.0.3'. Under the 'Enable' section, the checkbox for 'Enable Log Service(SLS) and Jaeger' is checked and highlighted with a red rectangle. Below this, several fields are visible, each with a red asterisk indicating they are required:

- * Endpoint: A dropdown menu showing 'cn-hangzhou.log.aliyuncs.com'.
- * Project: A text input field with a blurred value.
- * Logstore: A text input field with a blurred value.
- * AccessKeyID: A text input field with a blurred value.
- * AccessKeySecret: A text input field with a blurred value.

The descriptions of Log Service parameters are as follows.

Name	Parameter	Description
Endpoint	storage . aliyun_sls . endpoint	Specifies the endpoint in which the Project resides to store spans.

Name	Parameter	Description
Project	<code>storage . aliyun_sls . project</code>	Specifies the Project that stores spans. The Project name must be a string of 3 to 63 bytes and must contain only lowercase letters, numbers, and hyphens (-). In addition, the name must start and end with lowercase letters or numbers.
Logstore	<code>storage . aliyun_sls . logstore</code>	Specifies the Logstore that stores spans. The Logstore name must be a string of 3 to 63 bytes that must contain lowercase letters, numbers, hyphens (-), and underscores (_). The name must start and end with lowercase letters or numbers. A Logstore name must be unique in the Project to which it belongs.
AccessKeyID	<code>storage . aliyun_sls . accesskey . id</code>	Specify the user Access Key ID.
AccessKeySecret	<code>storage . aliyun_sls . accesskey . secret</code>	Specify the user Access Key Secret.

**Note:**

If you specify a Project that does not exist, the system automatically creates and initializes a new Project.

- Click Deploy Istio. After a few minutes, the Istio integrated with Alibaba Cloud Log Service is created.

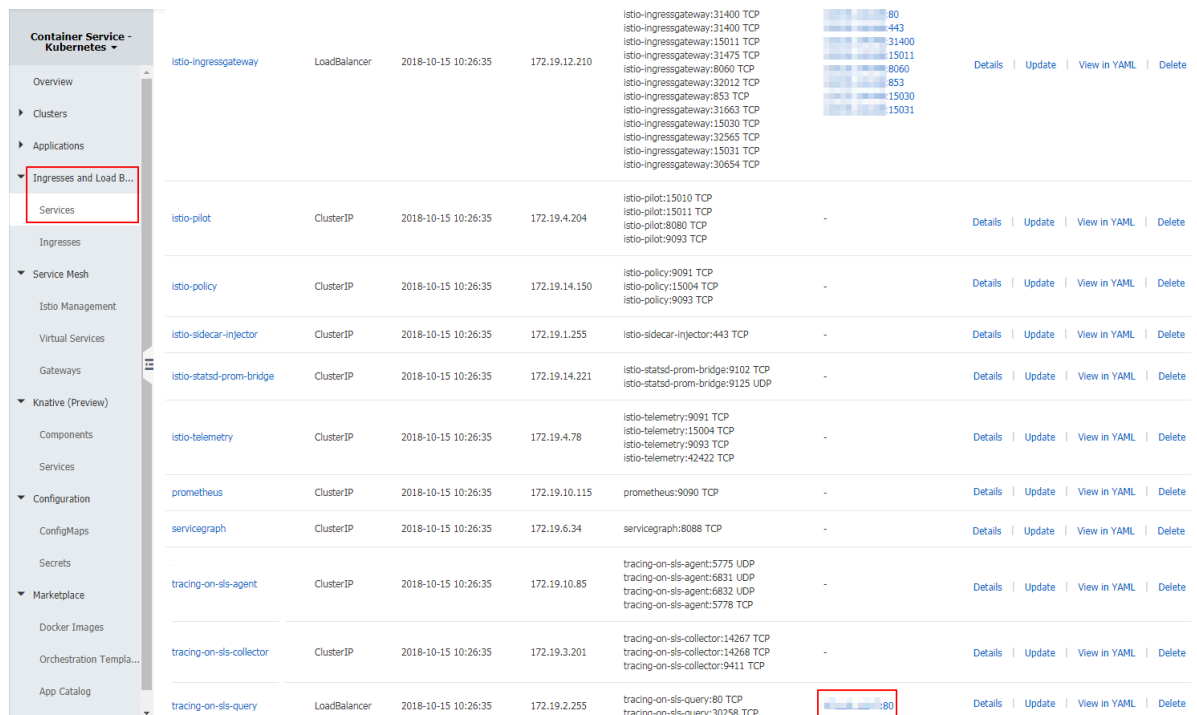
5. In the left-side navigation pane, when you choose Ingresses and Load Balancing > Service, the Istio services, including the integrated Log Service, are displayed on the right.

Container Service - Kubernetes Overview Clusters Applications Ingresses and Load Balancing Services Ingresses Service Mesh Istio Management Virtual Services Gateways Knative (Preview) Components Services Configuration ConfigMaps Secrets Marketplace Docker Images Orchestration Templates App Catalog	istio-ingressgateway	LoadBalancer	2018-10-15 10:26:35	172.19.12.210	istio-ingressgateway:31400 TCP istio-ingressgateway:31400 TCP istio-ingressgateway:15011 TCP istio-ingressgateway:31475 TCP istio-ingressgateway:8060 TCP istio-ingressgateway:32012 TCP istio-ingressgateway:853 TCP istio-ingressgateway:31663 TCP istio-ingressgateway:15030 TCP istio-ingressgateway:32565 TCP istio-ingressgateway:15031 TCP istio-ingressgateway:30654 TCP		Details Update View in YAML Delete
	istio-pilot	ClusterIP	2018-10-15 10:26:35	172.19.4.204	istio-pilot:15010 TCP istio-pilot:15011 TCP istio-pilot:8080 TCP istio-pilot:9093 TCP	-	Details Update View in YAML Delete
	istio-policy	ClusterIP	2018-10-15 10:26:35	172.19.14.150	istio-policy:9091 TCP istio-policy:15004 TCP istio-policy:9093 TCP	-	Details Update View in YAML Delete
	istio-sidecar-injector	ClusterIP	2018-10-15 10:26:35	172.19.1.255	istio-sidecar-injector:443 TCP	-	Details Update View in YAML Delete
	istio-statsd-prom-bridge	ClusterIP	2018-10-15 10:26:35	172.19.14.221	istio-statsd-prom-bridge:9102 TCP istio-statsd-prom-bridge:9125 UDP	-	Details Update View in YAML Delete
	istio-telemetry	ClusterIP	2018-10-15 10:26:35	172.19.4.78	istio-telemetry:9091 TCP istio-telemetry:15004 TCP istio-telemetry:9093 TCP istio-telemetry:42422 TCP	-	Details Update View in YAML Delete
	prometheus	ClusterIP	2018-10-15 10:26:35	172.19.10.115	prometheus:9090 TCP	-	Details Update View in YAML Delete
	servicegraph	ClusterIP	2018-10-15 10:26:35	172.19.6.34	servicegraph:8088 TCP	-	Details Update View in YAML Delete
	tracing-on-sls-agent	ClusterIP	2018-10-15 10:26:35	172.19.10.85	tracing-on-sls-agent:5775 UDP tracing-on-sls-agent:6831 UDP tracing-on-sls-agent:6832 UDP tracing-on-sls-agent:5778 TCP	-	Details Update View in YAML Delete
	tracing-on-sls-collector	ClusterIP	2018-10-15 10:26:35	172.19.3.201	tracing-on-sls-collector:14267 TCP tracing-on-sls-collector:14268 TCP tracing-on-sls-collector:9411 TCP	-	Details Update View in YAML Delete
	tracing-on-sls-query	LoadBalancer	2018-10-15 10:26:35	172.19.2.255	tracing-on-sls-query:80 TCP tracing-on-sls-query:30258 TCP		Details Update View in YAML Delete

Step 2: View Jaeger on Aliyun Log Service

1. Log on to the [Container Service console](#).

2. In the left-side navigation pane, click **Ingresses and Load Balancing > Service**, click a cluster and the **istio-system** namespace, and then click the external endpoint of the **tracing-on-sls-query** service.



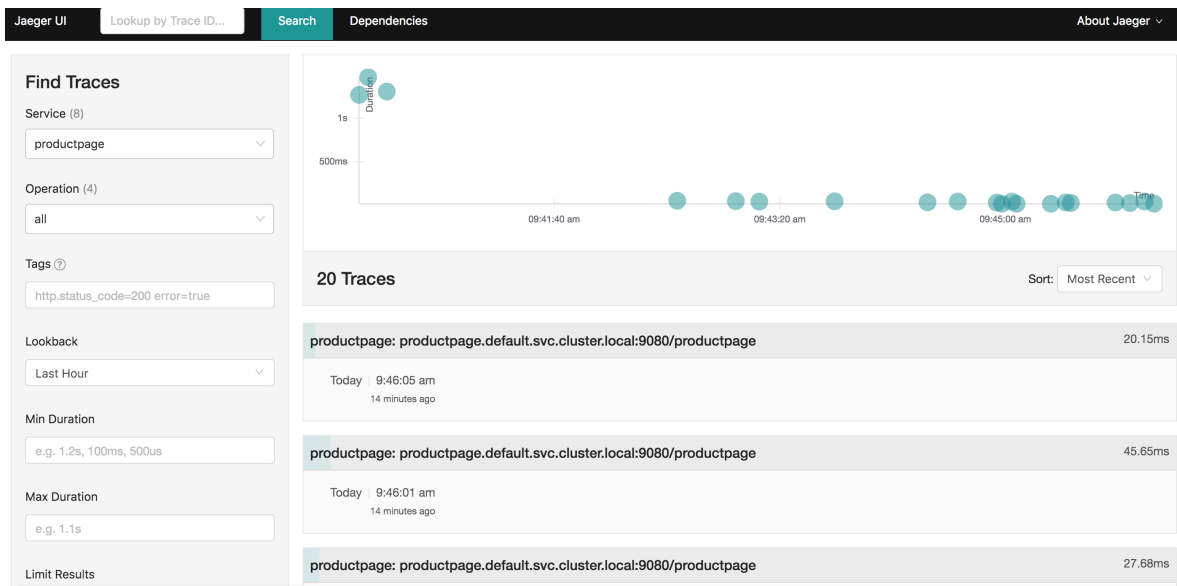
Service Name	Type	Creation Time	IP Address	Ports	Annotations	Actions
istio-ingressgateway	LoadBalancer	2018-10-15 10:26:35	172.19.12.210	istio-ingressgateway:31400 TCP istio-ingressgateway:31400 TCP istio-ingressgateway:15011 TCP istio-ingressgateway:31475 TCP istio-ingressgateway:8060 TCP istio-ingressgateway:32012 TCP istio-ingressgateway:853 TCP istio-ingressgateway:15030 TCP istio-ingressgateway:32565 TCP istio-ingressgateway:15031 TCP istio-ingressgateway:30654 TCP		Details Update View in YAML Delete
istio-pilot	ClusterIP	2018-10-15 10:26:35	172.19.4.204	istio-pilot:15010 TCP istio-pilot:15011 TCP istio-pilot:8080 TCP istio-pilot:9093 TCP	-	Details Update View in YAML Delete
istio-policy	ClusterIP	2018-10-15 10:26:35	172.19.14.150	istio-policy:9091 TCP istio-policy:15004 TCP istio-policy:9093 TCP	-	Details Update View in YAML Delete
istio-sidecar-injector	ClusterIP	2018-10-15 10:26:35	172.19.1.255	istio-sidecar-injector:443 TCP	-	Details Update View in YAML Delete
istio-statsd-prom-bridge	ClusterIP	2018-10-15 10:26:35	172.19.14.221	istio-statsd-prom-bridge:9102 TCP istio-statsd-prom-bridge:9125 UDP	-	Details Update View in YAML Delete
istio-telemetry	ClusterIP	2018-10-15 10:26:35	172.19.4.78	istio-telemetry:9091 TCP istio-telemetry:15004 TCP istio-telemetry:9093 TCP istio-telemetry:42422 TCP	-	Details Update View in YAML Delete
prometheus	ClusterIP	2018-10-15 10:26:35	172.19.10.115	prometheus:9090 TCP	-	Details Update View in YAML Delete
servicegraph	ClusterIP	2018-10-15 10:26:35	172.19.6.34	servicegraph:8088 TCP	-	Details Update View in YAML Delete
tracing-on-sls-agent	ClusterIP	2018-10-15 10:26:35	172.19.10.85	tracing-on-sls-agent:5775 UDP tracing-on-sls-agent:6831 UDP tracing-on-sls-agent:6832 UDP tracing-on-sls-agent:5778 TCP	-	Details Update View in YAML Delete
tracing-on-sls-collector	ClusterIP	2018-10-15 10:26:35	172.19.3.201	tracing-on-sls-collector:14267 TCP tracing-on-sls-collector:14268 TCP tracing-on-sls-collector:9411 TCP	-	Details Update View in YAML Delete
tracing-on-sls-query	LoadBalancer	2018-10-15 10:26:35	172.19.2.255	tracing-on-sls-query:80 TCP tracing-on-sls-query:30258 TCP		Details Update View in YAML Delete

3. On the left side, select a service from the Service drop-down list, set other parameters, and click **Find Traces**. The scatter diagram of time and duration displayed in the upper-right corner displays the results and provides drill-down capability.

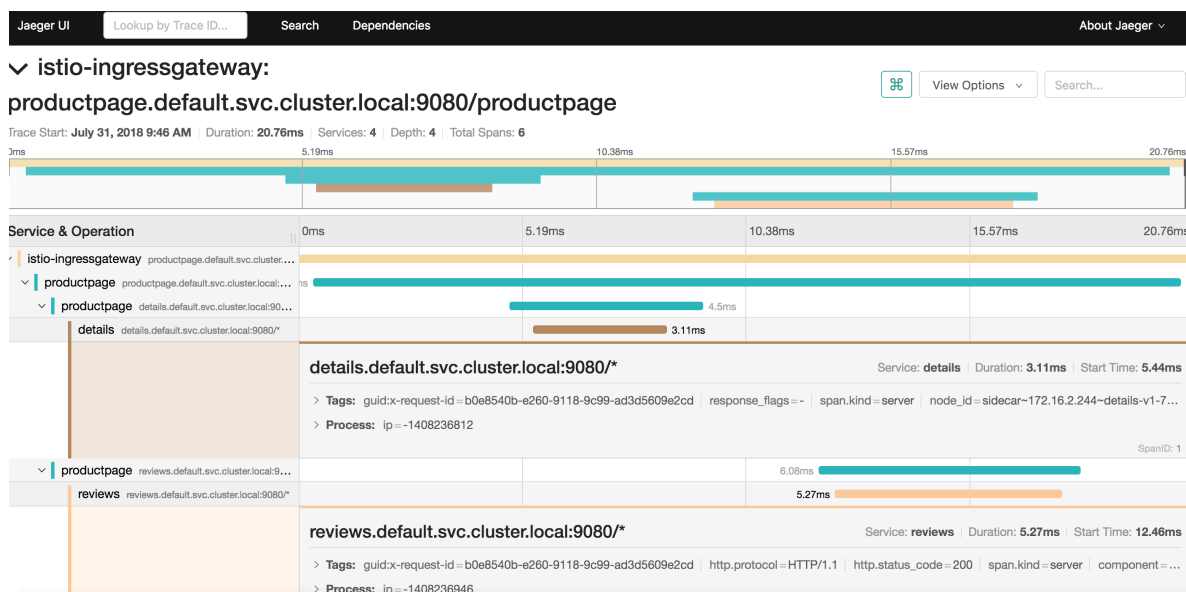


Note:

In this example, the Bookinfo sample application is used to view the productpage service calling.



- You can select from multiple views that display trace results in different ways. For example, you can select a histogram within a tracing period, or the service duration time of the tracing process.



Step 3: View Log Service

- Log on to the [Log Service console](#).
- Select a Project and click the Project name.
- Click Search on the right of a Logstore.

4. Select a log query time range, and then click Search & Analysis.

With Log Service, you can use query and analysis statements and other extended functions to optimize your query. For example, you can use analysis charts.

The following is a statistical chart.

Conclusion

Alibaba Cloud Container Service for Kubernetes can integrate Istio with Log Service . This topic uses an official example to describe how to integrate Istio with the distributed tracing system based on Alibaba Cloud Log Service.

We recommend that you use Alibaba Cloud Container Service for Kubernetes to quickly build Istio, an open management platform for microservices, and integrate Istio with the microservice development of your project.

5.4 Use Istio to deploy application services across Kubernetes and ECS instances

Starting from v0.2, Istio provides mesh expansion. With this feature, you can integrate non-Kubernetes services that typically run on VMs or bare metal hosts with the Istio service mesh that runs on your Kubernetes cluster.

Alibaba Cloud Container Service for Kubernetes supports the Istio mesh expansion capabilities. This topic uses an example from the Istio official website to details how to use Istio to deploy application services across Kubernetes and ECS instances.

Mesh expansion

Mesh expansion is a method based on the Istio service mesh deployed on Kubernetes. With this method, you can integrate VMs or bare metal hosts into the service mesh.

Mesh expansion is suitable for when you need to migrate your applications from your local system to cloud services. In a microservices system, not all workloads can run in Kubernetes. This means you may encounter scenarios in which you can only operate and maintain some services in Kubernetes, while other services run on VMs or bare metal hosts.

With the Istio control plane, you can manage services across Kubernetes and VMs or bare metal hosts, and ensure that all your services can continue to run normally.

Create a Kubernetes cluster and install Istio

Alibaba Cloud Container Service for Kubernetes 1.11.5 is now available. You can quickly create a Kubernetes cluster through the Container Service console. For more information, see [#unique_11](#).



Note:

You must make sure that you can connect to your Kubernetes cluster by using `kubectl`. For more information, see [#unique_12](#).

Deploy Istio through the app catalog. Create the `istio - system` namespace through a command or the console.

1. Log on to the [Container Service console](#).
2. In the left-side navigation pane, choose Store > App Catalog, and click `ack - istio` on the right side.
3. On the displayed page, select `istio - system` from the namespace drop-down list, and click Values. You can edit parameters to customize your Istio.



Note:

The readme document on the page provides the installation and removal information, including common questions about Custom Resource Definition (CRD) versions.

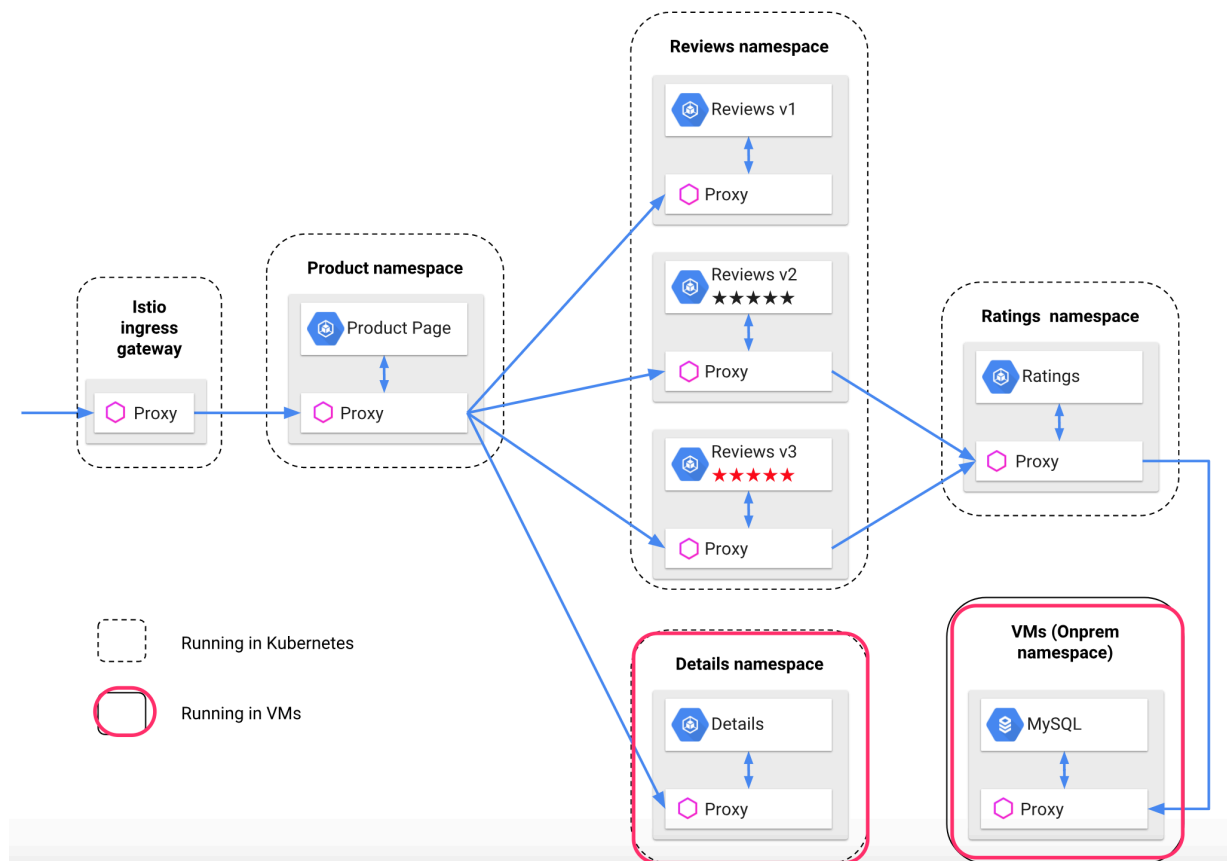
Install the sample application in your Kubernetes cluster

Run the following commands or use the console to create the `bookinfo` namespace, and then deploy the modified application. In the modified application, the `details` component is removed and `ingressgateway` is defined.

To obtain the files used in this example, see [Istio multi-cluster sample files](#).

```
kubectl create ns bookinfo
kubectl label namespace bookinfo istio-injection=enabled
kubectl apply -n bookinfo -f ./bookinfo/bookinfo-without-details.yaml
kubectl apply -n bookinfo -f ./bookinfo/bookinfo-gateway.yaml
```

Both the `details` and the database components of the application deployment run on the ECS instance that is outside the Kubernetes system.



Access the `/productpage` page through the address exposed by `ingressgateway` and verify that the `details` part cannot be displayed.



The Comedy of Errors

Summary: [Wikipedia Summary](#): The Comedy of Errors is one of **William Shakespeare's** early plays. It is his shortest and one of his most farcical comedies, with a major part of the humour coming from slapstick and mistaken identity, in addition to puns and word play.

Error fetching product details!

Sorry, product details are currently unavailable for this book.

Book Reviews

An extremely entertaining play by Shakespeare. The slapstick humour is refreshing!

— Reviewer1
★★★★★

Absolutely fun and entertaining. The play lacks thematic depth when compared to other plays by Shakespeare.

— Reviewer2
★★★★☆

Configure your Kubernetes

1. If you have not set internal load balancers for Kube DNS, Pilot, Mixer, and Citadel when you install Istio, you need to run the following command:

```
kubectl apply -f ./mesh-expansion.yaml
```

As shown in the following figure, the four services are created.

```
ali-1c36bbbed0b91:meshexpansion wangxn$ kubectl apply -f ./mesh-expansion.yaml
service "istio-pilot-ilb" created
service "dns-ilb" created
service "mixer-ilb" created
service "citadel-ilb" created
```

2. Generate the `cluster.env` Istio configuration file and the `kubedns` DNS configuration file both of which are to be deployed in the VMs. The `cluster.env` file contains the range of the cluster IP addresses that will be intercepted. The `kubedns` file contains the cluster service names that can be resolved by the applications on the VMs and then will be intercepted and forwarded by the sidecar.

To generate the configuration files, run the following command:

```
./setupMeshE x . sh generateCl usterEnvAn dDnsmaq
```

Configuration file `cluster.env`

```
ali-1c36bbbed0b91:meshexpansion wangxn$ cat cluster.env
ISTIO_SERVICE_CIDR=172.17.0.0/16
ISTIO_SYSTEM_NAMESPACE=istio-system
ISTIO_CP_AUTH=MUTUAL_TLS
```

Configuration file `kubedns`

```
ali-1c36bbbed0b91:meshexpansion wangxn$ cat kubedns
server=/svc.cluster.local/172.17.0.1:53
address=/istio-policy/172.17.0.1:15010
address=/istio-telemetry/172.17.0.1:15011
address=/istio-pilot/172.17.0.1:15012
address=/istio-citadel/172.17.0.1:15013
address=/istio-ca/172.17.0.1:15014
address=/istio-policy.istio-system/172.17.0.1:15010
address=/istio-telemetry.istio-system/172.17.0.1:15011
address=/istio-pilot.istio-system/172.17.0.1:15012
address=/istio-citadel.istio-system/172.17.0.1:15013
address=/istio-ca.istio-system/172.17.0.1:15014
```

Set the ECS instance

Configure your working environment to communicate with the ECS instance.

Generate an SSH key and assign it to the ECS instance. You can run the `ssh root@< ECS_HOST_I P >` command to check if you can connect to the ECS instance.

To generate a public key, run the following command:

```
ssh - keygen - b 4096 - f ~/.ssh / id_rsa - N ""
```



Note:

To ensure that the ECS instance and Kubernetes are mutually accessible over the Internet, you need to add them to the same security group.

With Alibaba Cloud Container Service for Kubernetes, you can quickly configure an ECS instance by running the following script:

```
export SERVICE_NAME_NAMESPACE = default
./ setupMeshE x . sh machineSet up root @< ECS_HOST_I P >
```

Run the following command to check the running process:

```
ps aux | grep istio
```

```
root@remotevm:~# ps aux | grep istio
root      19460  0.0  0.0 52284 3404 ?        Ss   11:59   0:00 su -s /bin/bash -c INSTANCE_IP=192.168.3.70 POD_NAME=remotevm POD_NAMESPACE=default exec /u
sr/local/bin/pilot-agent proxy --serviceCluster rawvm --discoveryAddress istio-pilot.istio-system:8080 --controlPlaneAuthPolicy MUTUAL_TLS
2> /var/log/istio/istio.err.log > /var/log/istio/istio.log istio-proxy
istio-p+ 19508  0.0  0.0 45276 4592 ?        Ss   11:59   0:00 /lib/systemd/systemd --user
istio-p+ 19510  0.0  0.0 61324 2064 ?        S    11:59   0:00 (sd-pam)
istio-p+ 19516  0.0  0.2 31204 17252 ?       Ssl  11:59   0:00 /usr/local/bin/pilot-agent proxy --serviceCluster rawvm --discoveryAddress istio-pilot.isti
o-system:8080 --controlPlaneAuthPolicy MUTUAL_TLS
istio-p+ 19537  7.1  0.5 133208 41572 ?       Sl   11:59   0:02 /usr/local/bin/envoy -c /etc/istio/proxy/envoy-rev1.json --restart-epoch 1 --drain-time-s 2
--parent-shutdown-time-s 3 --service-cluster rawvm --service-node sidecar~192.168.3.70~remotevm.default~default.svc.cluster.local --max-obj-name-len 189 -l
warn --v2-config-only
```

Run the following command to check if the node agent authenticated by Istio is running in a healthy status:

```
sudo systemctl status istio-auth - node - agent
```

Run services on the ECS instance

As shown in the preceding deployment figure, two services run on the ECS instance: one is the Details service, the other one is the Database service.

Run the Details service on the ECS instance

Run the following commands to simulate (by using Docker only) the `Details` service, run the service on the ECS instance, and expose port 9080 for the service.

```
docker pull istio / examples - bookinfo - details - v1 : 1 . 8 . 0
docker run -d -p 9080 : 9080 -- name details - on - vm
istio / examples - bookinfo - details - v1 : 1 . 8 . 0
```

Configure the sidecar to intercept the port. You need to configure this in the `/ var / lib / istio / envoy / sidecar . env` path and use the `ISTIO_INBOUND_PORTS` environment variable.

Run the following command on the VM in which the service runs:

```
echo " ISTIO_INBOUND_PORTS = 9080 , 8080 " > / var / lib / istio /
envoy / sidecar . env
systemctl restart istio
```

Register the Details service with Istio

Run the following command to view the IP address of the VM so that you can add it to the service mesh:

```
hostname - I
```

Manually configure a selector-less service and endpoints. The selector-less service is used to host services that are not backed by Kubernetes pods. For example, run the following command to register the `Details` service on a server that has the permissions to modify Kubernetes services and supports `istioctl` commands:

```
istioctl -n bookinfo register details 192 . 168 . 3 . 202
http : 9080
```

Access the `/ productpage` page again to verify that the `details` part is displayed as shown in following figure.

BookInfo Sample Sign in

The Comedy of Errors

Summary: [Wikipedia Summary](#): The Comedy of Errors is one of **William Shakespeare's** early plays. It is his shortest and one of his most farcical comedies, with a major part of the humour coming from slapstick and mistaken identity, in addition to puns and word play.

Book Details

Type:
paperback
Pages:
200
Publisher:
PublisherA
Language:
English
ISBN-10:
1234567890
ISBN-13:
123-1234567890

Book Reviews

An extremely entertaining play by Shakespeare. The slapstick humour is refreshing!
— Reviewer1

Absolutely fun and entertaining. The play lacks thematic depth when compared to other plays by Shakespeare.
— Reviewer2

Update the Ratings service to the version that can access a database

By default, the Ratings service cannot access any database. Run the following command to update the service version so that the service can access the database:

```
kubectl apply -f ./bookinfo / bookinfo - ratings - v2 - mysql -
vm . yaml
kubectl apply -f ./bookinfo / virtual - service - ratings -
mysql - vm . yaml
```

Access the `/productpage` page to verify that the `Ratings` part cannot be displayed as shown in the following figure. Then, you need to build a database service on the ECS instance and add the service to Istio.

BookInfo Sample 👤 jason (sign out)

The Comedy of Errors

Summary: [Wikipedia Summary](#): The Comedy of Errors is one of **William Shakespeare's** early plays. It is his shortest and one of his most farcical comedies, with a major part of the humour coming from slapstick and mistaken identity, in addition to puns and word play.

Book Details

Type:
paperback
Pages:
200
Publisher:
PublisherA
Language:
English
ISBN-10:
1234567890
ISBN-13:
123-1234567890

Book Reviews

An extremely entertaining play by Shakespeare. The slapstick humour is refreshing!
Reviewer1
Ratings service is currently unavailable

Absolutely fun and entertaining. The play lacks thematic depth when compared to other plays by Shakespeare.
Reviewer2
Ratings service is currently unavailable

Run a database service on the ECS instance

On the VM, run MariaDB as the backend for the Ratings service, and set MariaDB to be remotely accessible.

```
apt - get update && apt - getinstall - y mariadb - server
```

```
sed -i 's / 127 \. 0 \. 0 \. 1 / 0 \. 0 \. 0 \. 0 / g' /
etc / mysql / mariadb . conf . d / 50 - server . cnf
sudo mysql
# Grant the root permission .
GRANT ALL PRIVILEGES ON *.* TO 'root'@'localhost'
IDENTIFIED BY 'password' WITH GRANT OPTION ;
quit ;
sudo systemctl restart mysql
```

Run the following command to initialize the Ratings database on the VM:

```
curl -q https://raw.githubusercontent.com/istio/istio/master/samples/bookinfo/src/mysql/mysqldb-init.sql | mysql -u root -ppassword
```

To view different outputs of the Bookinfo application, run the following command to modify the rating records to generate different rating data that are displayed on the page:

```
mysql -u root -ppassword test -e "select * from ratings ;"
mysql -u root -ppassword test -e "update ratings set rating = 2 ; select * from ratings ;"
```

Register the database service into Istio

Configure the sidecar to intercept the port. You need to configure this in the `/var/lib/istio/envoy/sidecar.env` path and use the `ISTIO_INBOUND_PORTS` environment variable.

Run the following command on the VM in which the service runs:

```
echo "ISTIO_INBOUND_PORTS = 3306 , 9080 , 8080 " > /var/lib/istio/envoy/sidecar.env
systemctl restart istio
```

Run the following command to register the database service on a server that has the permissions to modify Kubernetes services and supports `istioctl` commands:

```
istioctl -n bookinfo register mysqldb 192.168.3.202 3306
```

Now Kubernetes pods and other servers included by mesh expansion can access the database service running on this server.

Access the `/productpage` page to verify that both the Details and Ratings parts can be displayed and these two services are provided by the ECS instance.

BookInfo Sample Jason (sign out)

The Comedy of Errors

Summary: [Wikipedia Summary](#): The Comedy of Errors is one of **William Shakespeare's** early plays. It is his shortest and one of his most farcical comedies, with a major part of the humour coming from slapstick and mistaken identity, in addition to puns and word play.

Book Details

Type:
paperback
Pages:
200
Publisher:
PublisherA
Language:
English
SBN-10:
1234567890
SBN-13:
123-1234567890

Book Reviews

An extremely entertaining play by Shakespeare. The slapstick humour is refreshing!

— Reviewer1
★ ★ ★ ★ ★

Absolutely fun and entertaining. The play lacks thematic depth when compared to other plays by Shakespeare.

— Reviewer2
★ ★ ★ ★ ★

Conclusion

Alibaba Cloud Container Service for Kubernetes provides the Istio mesh expansion capabilities. This topic uses a sample application from the Istio official website to details how to use Istio to deploy application services across Kubernetes and ECS instances.

We recommend that you use Alibaba Cloud Container Service for Kubernetes to quickly build Istio, an open management platform for microservices, and integrate Istio with the microservice development of your project.

5.5 Use Istio route rules to control ingress TCP traffic

This topic describes how to use standard Istio route rules to control ingress TCP traffic

Background information

The routing model provided by Istio for traffic management decouples traffic from infrastructure. This means that you can directly manage the traffic generated to access an application by setting route rules for the traffic through Istio Pilot. This capability is supported by deployed sidecar proxies.

In most cases, an Istio service mesh contains one or more load balancers (also referred to as gateways). The ingress gateways terminate TLS from external networks and allow traffic to come into the service mesh. Then, the traffic flows through internal services through sidecar gateways. For more information, see [Gateways in an Istio service mesh](#). An Istio gateway configures a load balancer that usually operates

at the edge of the service mesh to process incoming or outgoing HTTP or TCP traffic. Multiple gateways can coexist within the same service mesh.

Prerequisites

- A Kubernetes cluster of V1.11.2 or later is created with ACK. For more information, see [#unique_11](#).
- The Master node of the Kubernetes cluster is accessible. For more information, see [#unique_12](#).
- Netcat is installed in two local hosts.



Note:

- If you have not installed Netcat, use the tool (yum, apt-get, or other tools) according to your operating system to install it.
- Netcat is used in two local hosts as the terminals to access the target application (specifically, a TCP server in this topic). In this topic, the two terminals are referred to Terminal A and Terminal B.

Step 1: Deploy Istio



1. Log on to the [Container Service console](#).
2. In the left-side navigation pane under Container Service-Kubernetes, choose Clusters > Clusters.
3. Find the target cluster. Then, in the Action column, choose More > Deploy Istio.


Cluster Name/ID	Cluster Type	Region (All)	Network Type	Cluster Status	Number of Nodes	Time Created	Kubernetes Version	Action
	Kubernetes	China North 2 (Beijing)	VPC vpc-zze6clfm15d...	Running	6	10/22/2018,19:41:50	1.11.2	Manage View Logs Dashboard Scale Cluster More
	Kubernetes	China North 2 (Beijing)	VPC vpc-zze8sk2raka...	Failed to delete	4	09/18/2018,14:00:45	1.11.2	Delete Add Existing Instance Upgrade Cluster Automatic Scaling Addon Upgrade Deploy Istio

4. Set the following parameters.



Setting	Description
Clusters	The target cluster on which Istio is deployed.
Enable Prometheus for metrics/logs collection	Indicates whether to enable Prometheus for metrics and logs collection. This is enabled by default.

Setting	Description
Enable Grafana for metrics display	Indicates whether to allow Grafana to display metrics. This is enabled by default.
Enable the Kiali Visualization Service Mesh	<p>Indicates whether to enable the Kiali visualization service mesh. Disabled by default.</p> <p>To enable this feature, set the following parameters:</p> <ul style="list-style-type: none">• Username: The default is admin.• Password: The default is admin.

Setting	Description
Tracing Analysis Settings	<ul style="list-style-type: none"> • Enable Distributed Tracing with Jaeger : Indicates whether to enable Jaeger (the distributed tracing system). To use Jaeger, select this radio button, and activate Alibaba Cloud Log Service. <div>  Note: If you select this radio button, Log Service automatically creates a project named <code>istio - tracing - { ClusterID }</code> that is used to store the tracking data. </div> <ul style="list-style-type: none"> • Activate Tracing Analysis: Indicates whether to activate the Tracing Analysis service. To activate this service, select this radio button, and then click Activate now. You must enter an endpoint address in the format of <code>http :// tracing - analysis - dc - hz . aliyuncs . com /.../ api / v1 / spans .</code> <div>  Note: <ul style="list-style-type: none"> - An address of this format indicates an Internet or intranet endpoint that is used by a Zipkin client. This client is used to transmit collected data to the Tracing Analysis service, which then uses the API from v1 release. - If you use an intranet endpoint, you must ensure that your Kubernetes cluster and the Tracing Analysis instance are in the same region to maintain stable network performance. </div>
Pilot Settings	The trace sampling percentage. The value range is from 0 to 100. The default value is 1.

Setting	Description
Control Egress Traffic	<ul style="list-style-type: none"> • Permitted Addresses for External Access: range of IP addresses that can be used to directly access services in the Istio service mesh. By default, this field is left blank. Use commas (,) to separate multiple IP address ranges. • Blocked Addresses for External Access: range of IP addresses that are blocked against external accesses. By default, this IP address range contains the cluster pod CIDR block and service CIDR block. Use commas (,) to separate multiple IP address ranges. • ALL: Select this check box to block all the IP addresses used to access the Internet. <div>  Note: If the settings of these two parameters conflict with each other, the Permitted Addresses for External Access prevails. For example, if an IP address is listed in both IP address ranges that you set for these two parameters, the IP address can be still accessed. That is, the setting of Permitted Addresses for External Access prevails. </div>

5. Click Deploy Istio.

Step	Status
Create Istio Resource Definition	 Succeeded 53 / 53
Deploy Istio	 Running
Deploy Istio	

Step 2: Build an image of a TCP server

You can directly use a built image file (such as `registry.cn-hangzhou.aliyuncs.com/wangxining/tcptest:0.1`), or following these steps to build an image of a TCP server:

1. Clone a code repository from this address: `https://github.com/osswangxing/Istio-TCPRoute-Sample`.
2. Switch to the code directory to verify that a file named `Dockerfile` is created.

```
.
├── Dockerfile
├── Godeps
│   ├── Godeps.json
│   └── Readme
├── LICENSE
├── README.md
├── k8s
│   ├── deployment.yml
│   ├── destination-rule-all.yaml
│   ├── gateway.yaml
│   ├── service.yaml
│   └── virtualservice.yaml
├── tcp-echo.go
├── vendor
│   ├── github.com
│   │   └── nu7hatch
│   │       └── gouuid
│   │           ├── COPYING
│   │           ├── README.md
│   │           └── uuid.go
└── 6 directories, 14 files
```

3. Run the following command to build the image:

```
docker build -t { the address of the target image repository } .
```

Step 3: Deploy a TCP server

1. Run the following commands:

```
cd k8s
kubectl apply -f deployment.yml
kubectl apply -f service.yml
```



Note:

The preceding commands creates one service `tcp - echo` , and two deployments `tcp - echo - v1` and `tcp - echo - v2` that are labeled with `app : tcp - echo` . The service `tcp - echo` is associated with these two deployments by using the selector field.

```
selector :
  app : " tcp - echo "
```

2. Run the `kubectl get pods -- selector = app = tcp - echo` command to verify that the pods of the TCP server is in running status.

NAME	READY	STATUS	RESTARTS
AGE			
tcp - echo - v1 - 7c775f57c9 - frprp	2 / 2	Running	
0 1m			
tcp - echo - v2 - 6bcfd7dcf4 - 2sqhf	2 / 2	Running	
0 1m			

3. Run the `kubectl get service -- selector = app = tcp - echo` command to verify the `tcp - echo` service of the TCP server is created.

NAME	TYPE	CLUSTER - IP	EXTERNAL - IP
PORT (S)	AGE		
tcp - echo	ClusterIP	172 . 19 . 46 . 255	< none >
3333 / TCP	17h		

Step 4: Set gateways for the TCP server

1. Create the file `gateway . yaml` , and copy the following code to the file:

```
apiVersion : networking . istio . io / v1alpha3
kind : Gateway
metadata :
  name : tcp - echo - gateway
spec :
  selector :
    istio : ingressgateway # use istio default
  controller
  servers :
    - port :
        number : 31400
        name : tcp
        protocol : TCP
      hosts :
        - "*"
---
apiVersion : networking . istio . io / v1alpha3
kind : Gateway
metadata :
  name : tcp - echo - gateway - v2
spec :
  selector :
    istio : ingressgateway # use istio default
  controller
  servers :
    - port :
```

```

    number : 31401
    name : tcp
    protocol : TCP
    hosts :
    - "*"

```

2. Run the `kubectl apply -f gateway . yaml` command to create gateways.



Note:

- Two gateways are created. One listens to port 31400, and the other one listens to port 31401.
- The two gateways share one service `istio-ingressgateway` of the `LoadBalancer` type. This service provides an Internet IP address for the TCP server.

			istio-ingressgateway:80 TCP
			istio-ingressgateway:31380 TCP
			istio-ingressgateway:443 TCP
			istio-ingressgateway:31390 TCP
			istio-ingressgateway:31400 TCP
			istio-ingressgateway:31400 TCP
			istio-ingressgateway:31401 TCP
			istio-ingressgateway:31401 TCP
			istio-ingressgateway:15011 TCP
			istio-ingressgateway:31285 TCP
			istio-ingressgateway:8060 TCP
			istio-ingressgateway:32571 TCP
			istio-ingressgateway:853 TCP
			istio-ingressgateway:30849 TCP
			istio-ingressgateway:15030 TCP
			istio-ingressgateway:31258 TCP
			istio-ingressgateway:15031 TCP
			istio-ingressgateway:30671 TCP

Step 5: Create Istio route rules

Create the files `destination-rule-all . yaml` and `virtualservice . yaml`, copy the following code to these two files, and then run the `kubectl apply -f destination-rule-all . yaml` and `kubectl apply -f virtualservice . yaml` commands for these two files, respectively.

```

apiVersion : networking . istio . io / v1alpha3
kind : VirtualService
metadata :
  name : tcp - echo
spec :
  hosts :
  - "*"
  gateways :
  - tcp - echo - gateway
  - tcp - echo - gateway - v2
  tcp :

```

```

- match :
  - port : 31400
    gateways :
      - tcp - echo - gateway
    route :
      - destination n :
          host : tcp - echo . default . svc . cluster . local
          subset : v1
          port :
            number : 3333
- match :
  - port : 31401
    gateways :
      - tcp - echo - gateway - v2
    route :
      - destination n :
          host : tcp - echo . default . svc . cluster . local
          subset : v2
          port :
            number : 3333

```

Step 6: Verify the traffic is routed according to the rules

1. Obtain the IP address (namely, the the external endpoint displayed in the console) of the `istio - ingressgateway` service.
 - a. Log on to the [Container Service console](#).
 - b. In the left-side navigation pane under Container Service-Kubernetes, choose **Discovery and Load Balancing > Services**.
 - c. Select the target cluster and namespace, and then find the external endpoint of the `istio - ingressgateway` service.
2. Enable Terminal A to run the following command:

```
nc INGRESSGAT_EWAY_IP 31400
```

3. Enter `hello , appl` to verify that the traffic destined for port 31400 is forwarded to pod V1.

```

Welcome , you are connected to node cn - beijing . i -
2zeij4azns u1dvd4mj5c .
Running on Pod tcp - echo - v1 - 7c775f57c9 - frprp .
In namespace default .
With IP address 172 . 16 . 2 . 90 .
Service default .
hello , appl
hello , appl
continue ..
continue ..

```

4. View the logs of pod V1.

```
kubectl logs -f tcp - echo - v1 - 7c775f57c9 - frprp - c
tcp - echo - container | grep Received
```

```

2018 / 10 / 17 07 : 32 : 29 6c7f4971 - 40f1 - 4f72 - 54c4 -
e1462a8461 89 - Received Raw Data : [ 104 101 108 108
111 44 32 97 112 112 49 10 ]
2018 / 10 / 17 07 : 32 : 29 6c7f4971 - 40f1 - 4f72 - 54c4 -
e1462a8461 89 - Received Data ( converted to string ):
hello , app1
2018 / 10 / 17 07 : 34 : 40 6c7f4971 - 40f1 - 4f72 - 54c4 -
e1462a8461 89 - Received Raw Data : [ 99 111 110 116
105 110 117 101 46 46 10 ]
2018 / 10 / 17 07 : 34 : 40 6c7f4971 - 40f1 - 4f72 - 54c4 -
e1462a8461 89 - Received Data ( converted to string ):
continue ..

```

5. Enable Terminal B to run the following command:

```
nc INGRESSGAT EWAY_IP 31401
```

6. Enter `hello , app2` to verify that the traffic destined for port 31401 is forwarded to pod V2.

```

Welcome , you are connected to node cn - beijing . i -
2zeij4azns uldvd4mj5b .
Running on Pod tcp - echo - v2 - 6bcfd7dcf4 - 2sqhf .
In namespace default .
With IP address 172 . 16 . 1 . 95 .
Service default .
hello , app2
hello , app2
yes , this is app2
yes , this is app2

```

7. View the logs of pod V2.

```

kubectl logs -f tcp - echo - v2 - 6bcfd7dcf4 - 2sqhf - c
tcp - echo - container | grep Received
2018 / 10 / 17 07 : 36 : 29 1a70b9d4 - bbc7 - 471d - 4686 -
89b9234c8f 87 - Received Raw Data : [ 104 101 108 108
111 44 32 97 112 112 50 10 ]
2018 / 10 / 17 07 : 36 : 29 1a70b9d4 - bbc7 - 471d - 4686 -
89b9234c8f 87 - Received Data ( converted to string ):
hello , app2
2018 / 10 / 17 07 : 36 : 37 1a70b9d4 - bbc7 - 471d - 4686 -
89b9234c8f 87 - Received Raw Data : [ 121 101 115 44
116 104 105 115 32 105 115 32 97 112 112 50
10 ]

```

```
2018 / 10 / 17 07 : 36 : 37 1a70b9d4 - bbc7 - 471d - 4686 -  
89b9234c8f 87 - Received Data ( converted to string ) :  
yes , this is app2
```

5.6 Use the Canary method that uses Istio to deploy a service

This topic describes how to use the Canary method (that uses Istio) to deploy a service through Alibaba Cloud Container Service for Kubernetes (ACK).

Background information

Istio is a service mesh that can be used to meet the requirements of the distributed application architectures that involve microservices such as application O&M, debugging, and security management. You also can use Istio for microservice network scenarios such as load balancing, service-to-service authentication, and monitoring.

By using Istio, ACK allows you to manage services of an application. With ACK, you can create and manage multiple versions of one service for an application, manage the distribution of traffic destined for the service, and implement a Canary release for the service.

Prerequisites

- A Kubernetes cluster is created. For more information, see [#unique_11](#).
- Istio is deployed for this Kubernetes cluster. For more information, see [#unique_48](#).

Procedure

1. Enable automatic sidecar injection.

- a. Log on to the [Container Service console](#).
- b. In the left-side navigation pane under Container Service-Kubernetes, choose **Clusters > Namespaces**.
- c. Click **Create**.
- d. In the displayed dialog box, set **Name** to `demo`, and set **Tag** to follows:
 - The variable name of the tag is set to `istio - injection`.
 - The variable value of the tag is set to `enabled`.

Create Namespace

Name:

1-63 characters, can only contain numbers, lower case letters, and "-", and can only be letters or numbers at the beginning and end

Tags:

Variable Name	Variable Value	Action
<input type="text" value="istio-injection"/>	<input type="text" value="enabled"/>	<input type="button" value="Add"/>

2

- e. Click **Add**, and then click **OK**.

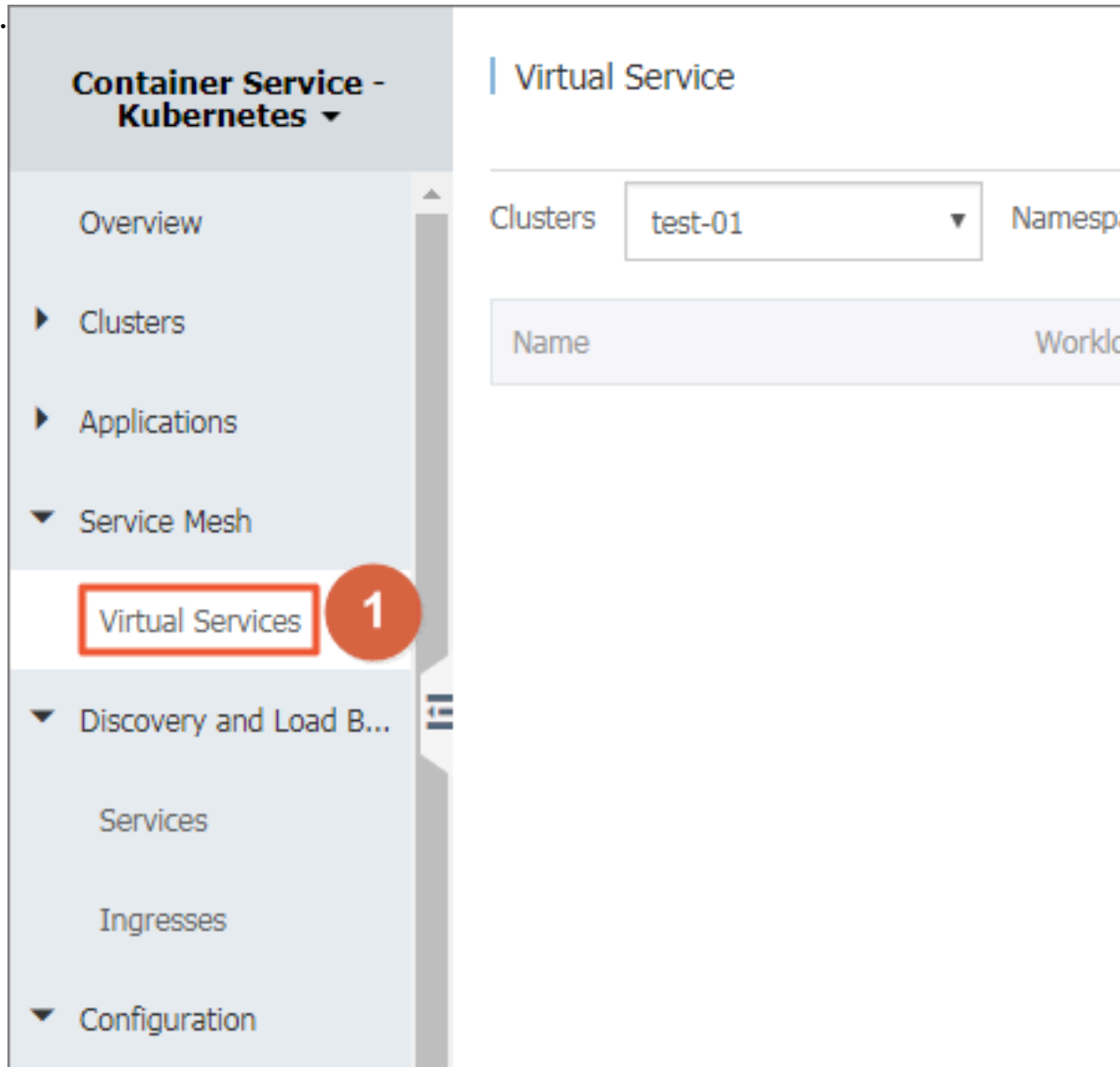
Then, the newly created namespace is displayed in the namespace list.

Name	Tag	Status	Time Created	Action
default		Ready	02/28/2019,14:50:52	ResourceQuota and LimitRange Edit Delete
demo	istio-injection: enabled	Ready	04/26/2019,16:39:07	ResourceQuota and LimitRange Edit Delete
istio-system	name: istio-system	Ready	04/01/2019,16:21:52	ResourceQuota and LimitRange Edit Delete
kube-public		Ready	02/28/2019,14:50:52	ResourceQuota and LimitRange Edit Delete
kube-system		Ready	02/28/2019,14:50:52	ResourceQuota and LimitRange Edit Delete

2. Create a server-end application and its virtual service.

- a. Log on to the [Container Service console](#).
- b. In the left-side navigation pane, choose Service Mesh > Virtual Service.
- c. In the upper-right corner of the page, click

Create.



d. On the Basic Information tab page, set the required parameters.

- **Name:** Set the application name.
- **Version:** Set the application version.
- **Clusters:** Select the cluster where you want to deploy the application.
- **Namespaces:** Select the namespace where you want to deploy the application. You must select the namespace created in step 1.
- **Replicas:** Set the number of replicas, that is, the number of pods.

Create Application

Basic Information Container Service Configuration Done

Name:

The name should be 1-64 characters long, and can contain numbers, lower case English letters and hyphens, but cannot start with a hyphen.

Version:

The version begins with a letter and can contain numbers, English letters, or "-", less than 64 characters in length.

Clusters:

Namespaces:

Replicas:

Back Next

- e. Configure the container. Set Image Name to nginx, and set Image Version to 1.14. For more information about other container settings, see [#unique_55](#). Then, click Next.

Create Application

Basic Information Container Service Configuration Done

Container1 Add Container

Image Name: [Select image](#)

Image Version: [Select image version](#)

☐ Always pull image [Image pull secret](#)

Resource Limit: CPU Core Memory MiB

Resource Request: CPU Core Memory MiB ⓘ Please set according to actual usage

Init Container ☐

Overview

- General
- Environment
- Health Check
- Life cycle
- Volume
- Log Configuration

Prev Next

- f. Set the Service parameters.

- A. In the upper-left corner, click Create. Then, in the displayed dialog box, set the required parameters.

Create Application

Basic Information Container Service Configuration Done

Services(Service) [Create](#)

Name	service port	Container Port	Protocol

Prev Create

- Name: Enter the service name.
- Type: Select the ClusterIP service type.

- **Port Mapping:** Set the service port name, service port number, container port number, and TCP protocol. The format of a service port name must be `<protocol>[-<suffix>-]`. The protocol part of a service port name can be one of the following: gRPC, HTTP, HTTP/2, HTTPS, Mongo, Redis, TCP, TLS, or UDP. Istio supports a routing service through one of these protocols.



Note:

The protocol part of a service port name is case insensitive.

- **Annotation:** Add an annotation to the service.
- **Tag:** Add a tag to the service to identify the service.

Create Service

Name:

Type:

☐ Headless Service

Port Mapping: [+ Add](#)

Name	service port	Container Port	Protocol
http	8080	80	TCP

Annotation: [+ Add](#)

Tag: [+ Add](#)

[Create](#) [Cancel](#)

B. Click Create.

Create Application

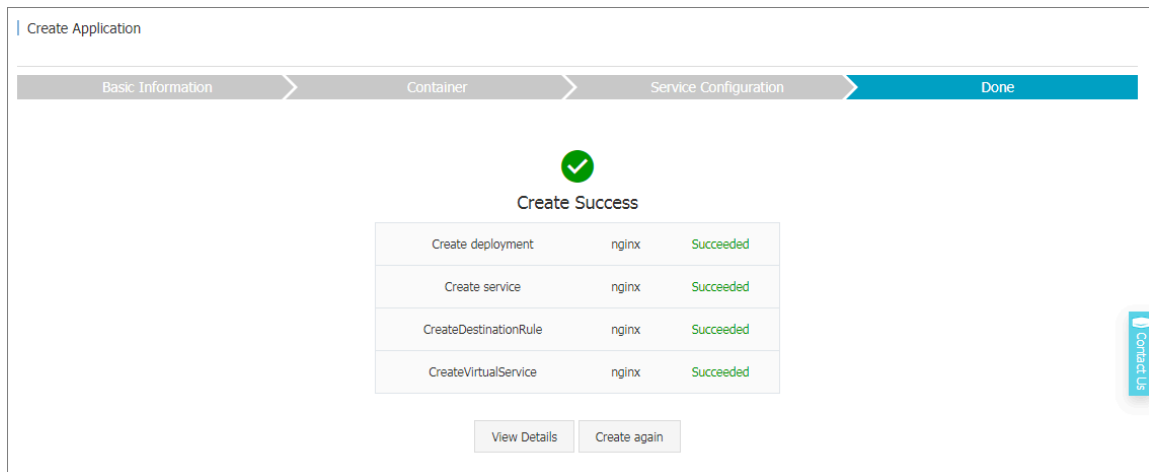
Basic Information > Container > **Service Configuration** > Done

Services(Service) [Update](#) [Delete](#)

Name	service port	Container Port	Protocol
http	8080	80	TCP

[Prev](#) [Create](#)

- g. In the lower-right corner, click **Create**. Then, the system displays all the created resources if no error occurs. You can click **View Details** to view the application details.



3. Create a client-end application named `sleep`. This application will be used to send access traffic to the created virtual service.

a. Save the following as a file named `sleep . yaml`.

```
apiVersion : v1
kind : Service
metadata :
  name : sleep
  labels :
    app : sleep
spec :
  ports :
    - port : 80
      name : http
  selector :
    app : sleep
---
apiVersion : extensions / v1beta1
kind : Deployment
metadata :
  name : sleep
spec :
  replicas : 1
  template :
    metadata :
      labels :
        app : sleep
    spec :
      containers :
        - name : sleep
          image : pstauffer / curl
          command : ["/ bin / sleep ", " 3650d "]
```

```
imagePullPolicy : IfNotPresent
```

- b. In the demo namespace, run the `kubectl apply -f sleep.yaml -n demo` command.
- c. Run the `kubectl exec -it -n demo <podName> bash` command to log on to the pod where the `sleep` application runs, and then run the following commands to call the `nginx` service:

```
for i in `seq 1000`
do
  curl -I http://nginx.demo:8080 ;
  echo '';
  sleep 1 ;
done ;
```

If the service is called, the content shown in the following figure is displayed.

```
HTTP/1.1 200 OK
content-type: text/html; charset=utf-8
content-length: 5719
server: istio-envoy
date: Tue, 02 Apr 2019 08:21:13 GMT
x-envoy-upstream-service-time: 24
```

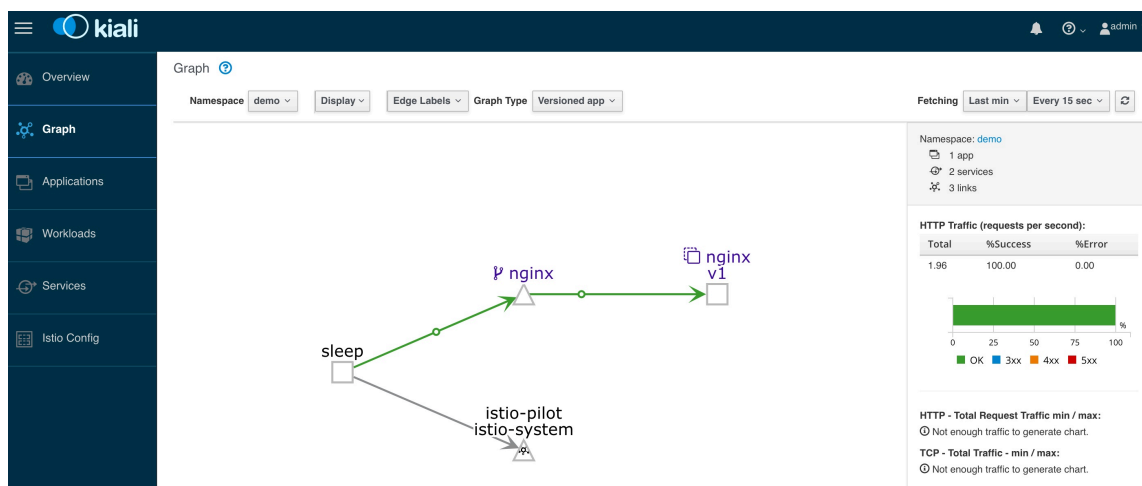
- d. Access the Kiali console to verify that the virtual service is called.



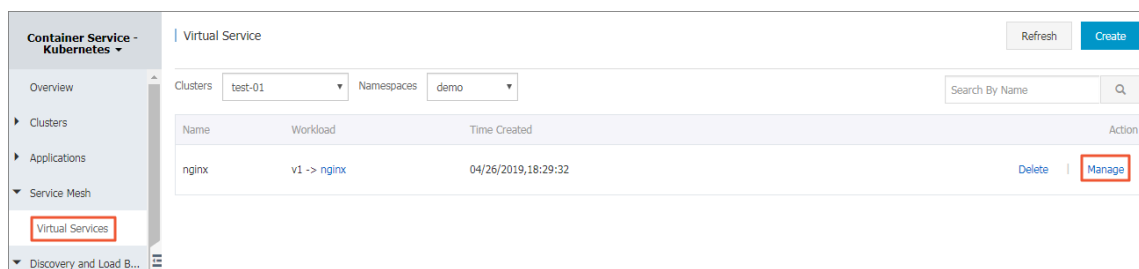
Note:

Before you perform this step, you must enable the Kiali visualized service mesh when deploying Istio for the target Kubernetes cluster.

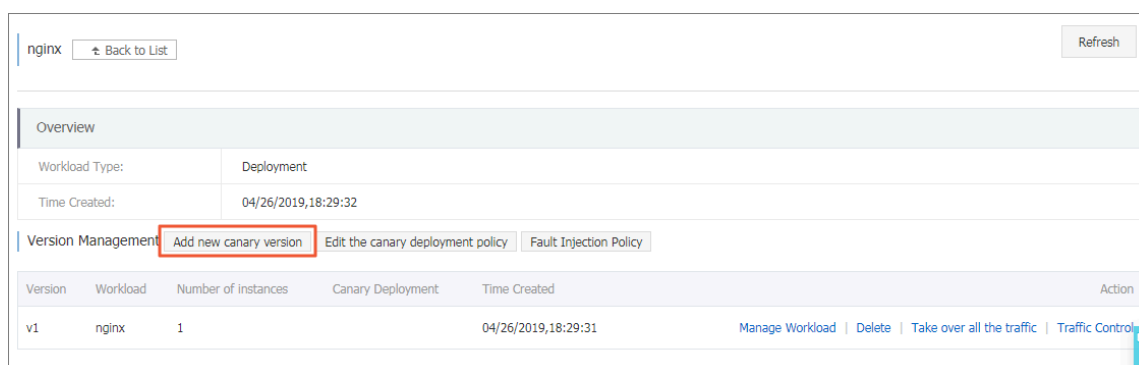
- A. Run the `kubectl port - forward svc / kiali - n istio - system 20001 : 20001` command to open port 20001 in your local host.
- B. Enter `http :// localhost : 20001` in your browser.



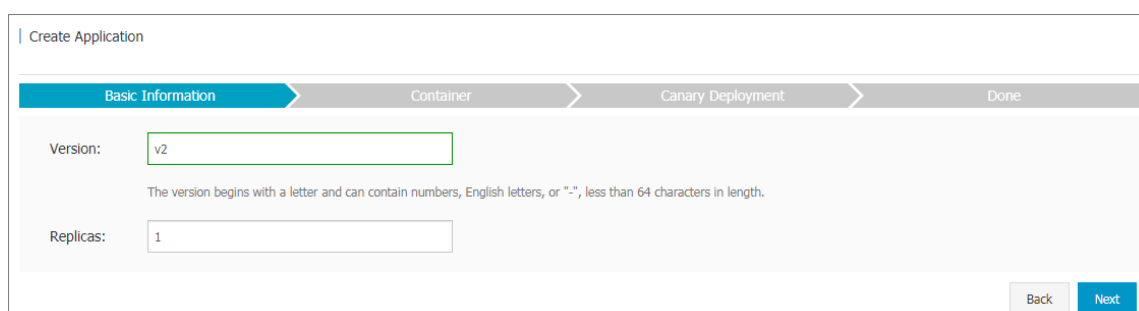
4. Create a version of the virtual service for a Canary release.
 - a. Log on to the [Container Service console](#).
 - b. In the left-side navigation pane under Container Service-Kubernetes, choose Service Mesh > Virtual Service.
 - c. On the right of the target virtual service, click Manage.



- d. Click Add new canary version.



- e. Set the Version and Replicas, then click Next.



- f. Configure a container. Set Image Name to nginx, and set Image Version to 1.15. For more information about other container configurations, see [#unique_55](#). Then, click Next.

The screenshot shows the 'Create Application' wizard with the 'Container' tab selected. The 'Image Name' is 'nginx' and 'Image Version' is '1.15'. The 'Resource Limit' section shows CPU 'eg : 500m' and Memory 'eg : 128Mi'. The 'Resource Request' section shows CPU '0.25' and Memory '512'. The 'Init Container' checkbox is unchecked. The 'Overview' sidebar on the right lists 'General', 'Environment', 'Health Check', 'Life cycle', 'Volume', and 'Log Configuration'. The 'Next' button is highlighted in orange.

- g. Set the Type parameter to select a Canary deployment policy. For example, you can select the Weight based Routing radio button, and then set v2Traffic Weight to 30 %.

**Note:**

- If you select the Weight based Routing radio button, then you can set the traffic distribution ratio for versions of the service that is to be deployed.

- You only need to set the v2Traffic Weight parameter. The v1Traffic Weight value is shown through automatic calculation.

Create Application

Basic Information > Container > **Canary Deployment** > Done

Deployment Config

Type ☒ Weight based Routing ☐ Request Content based Routing

Traffic Weight v2Traffic Weight % v1Traffic Weight %

Prev Create

- h. Click Create. After a creation success message is displayed by the system, choose Service Mesh > Virtual Services.

Container Service - Kubernetes

Virtual Service

Clusters test-01 Namespaces demo Search By Name

Name	Workload	Time Created	Action
nginx	v1 -> nginx v2 -> nginx-v2	04/26/2019,18:29:32	Delete Manage

- i. On the right of the target service, click Manage.

Container Service - Kubernetes

nginx Back to List Refresh

Overview

Workload Type: Deployment
Time Created: 04/26/2019,18:29:32

Version Management Add new canary version Edit the canary deployment policy Fault Injection Policy

Version	Workload	Number of Instances	Canary Deployment	Time Created	Action
v1	nginx	1	Traffic Weight 70%	04/26/2019,18:29:31	Manage Workload Delete Take over all the traffic Traffic Control
v2	nginx-v2	1	Traffic Weight 30%	04/28/2019,10:25:37	Manage Workload Delete Take over all the traffic Traffic Control

- j. Run the `kubectl exec -it -n demo < podName > bash` command to log on to the pod where the `sleep` application runs, and then run the following commands to call the `nginx` service:

```
for i in `seq 1000`
do
curl -I http://nginx.demo:8080 ;
echo " ";
sleep 1 ;
```

```
done ;
```

If the service is called, the content shown in the following figure is displayed.

```
HTTP/1.1 200 OK
content-type: text/html; charset=utf-8
content-length: 5719
server: istio-envoy
date: Tue, 02 Apr 2019 08:21:13 GMT
x-envoy-upstream-service-time: 24
```

- k. Access the Kiali console to verify that the access traffic of the virtual service are distributed to the two versions according to the distribution ratio that you set.

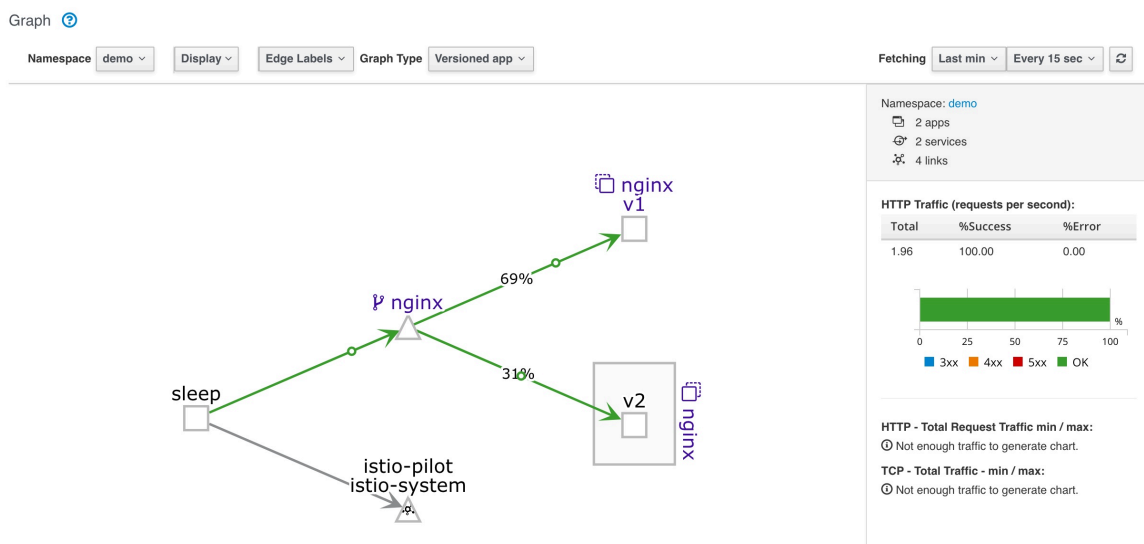


Note:

Before you perform this step, you must enable the Kiali visualized service mesh when deploying Istio for the target Kubernetes cluster.

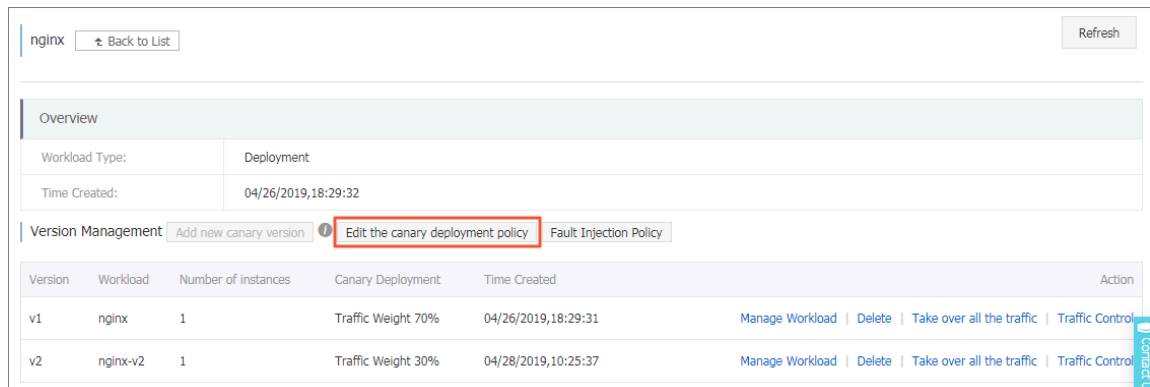
A. Run the `kubectl port - forward svc / kiali - n istio - system 20001 : 20001` command to open port 20001 in your local host.

B. Enter `http :// localhost : 20001` in your browser.



5. Edit the Canary release policy.

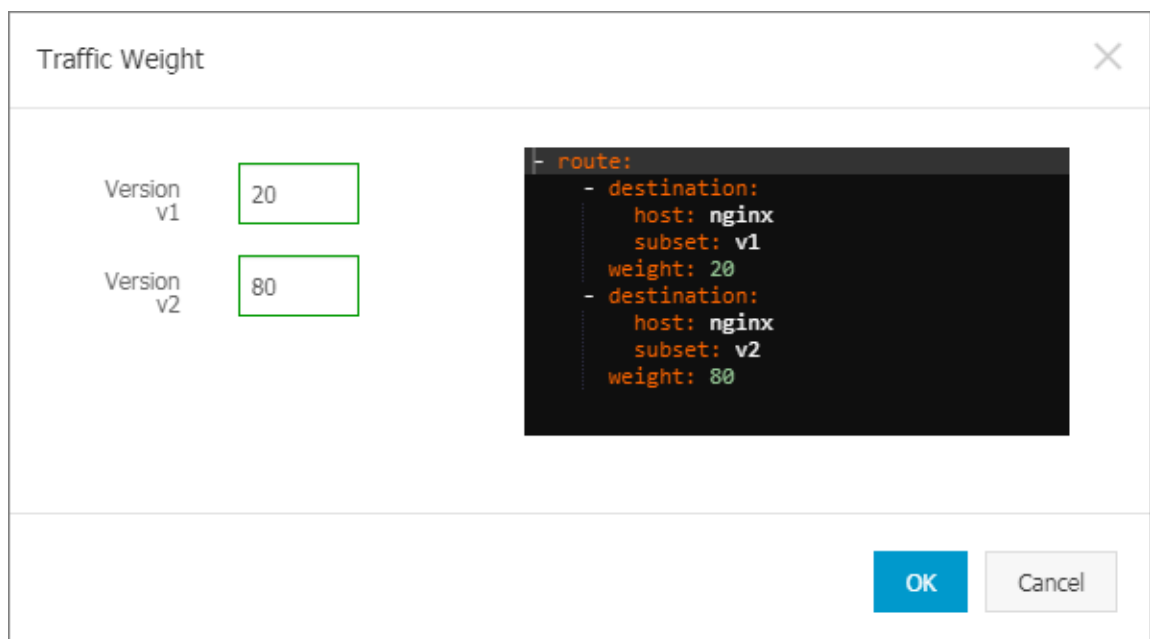
- On the page that displays the details of the target virtual service, click Edit the canary deployment policy.



The screenshot shows the 'nginx' Virtual Service details page. The 'Version Management' tab is active, displaying a table of canary versions. The 'Edit the canary deployment policy' button is highlighted with a red box.

Version	Workload	Number of instances	Canary Deployment	Time Created	Action
v1	nginx	1	Traffic Weight 70%	04/26/2019,18:29:31	Manage Workload Delete Take over all the traffic Traffic Control
v2	nginx-v2	1	Traffic Weight 30%	04/28/2019,10:25:37	Manage Workload Delete Take over all the traffic Traffic Control

- Change the traffic distribution ratio for the two versions of the target virtual services. For example, set Version v1 to 20 %, and set Version v2 to 80 %. Then, click OK.

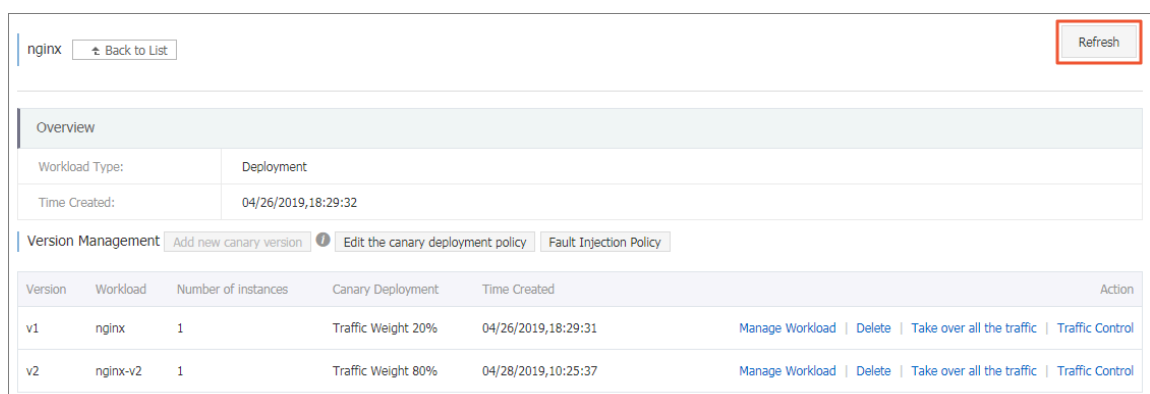


The 'Traffic Weight' dialog box shows the configuration for the two versions. Version v1 is set to 20 and Version v2 is set to 80. A code snippet shows the corresponding Istio route configuration:

```
route:
- destination:
  host: nginx
  subset: v1
  weight: 20
- destination:
  host: nginx
  subset: v2
  weight: 80
```

The OK button is highlighted with a blue box.

- In the upper-right corner, click Refresh.



The screenshot shows the 'nginx' Virtual Service details page after refreshing. The 'Refresh' button in the upper-right corner is highlighted with a red box.

Version	Workload	Number of instances	Canary Deployment	Time Created	Action
v1	nginx	1	Traffic Weight 20%	04/26/2019,18:29:31	Manage Workload Delete Take over all the traffic Traffic Control
v2	nginx-v2	1	Traffic Weight 80%	04/28/2019,10:25:37	Manage Workload Delete Take over all the traffic Traffic Control

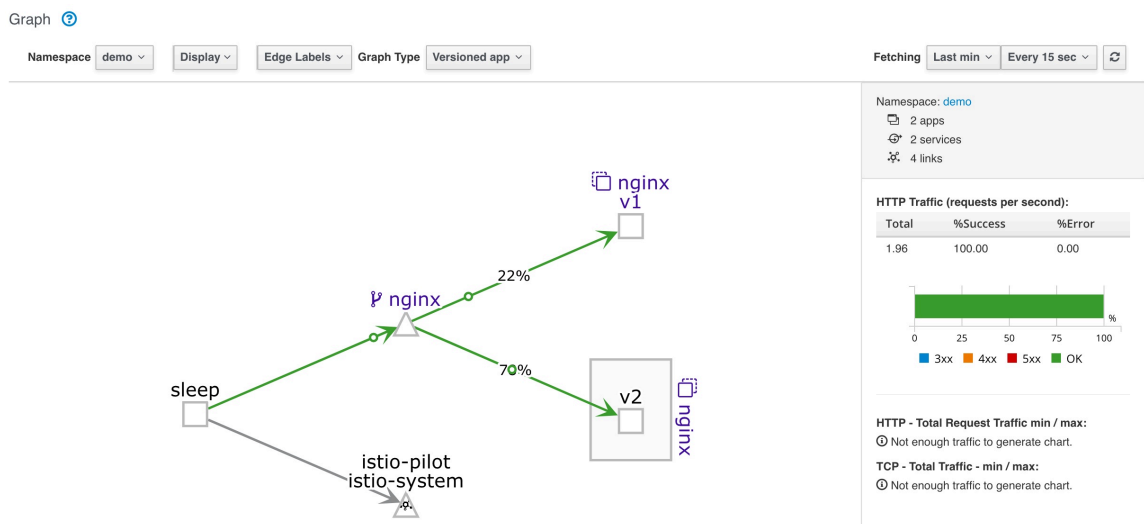
- d. Run the `kubectl exec -it -n demo <podName> bash` command to log on to the pod where the `sleep` application runs, and then run the following commands to call the `nginx` service:

```
for i in `seq 1000`
do
curl -I http://nginx.demo:8080;
echo '';
sleep 1;
done;
```

If the service is called, the content shown in the following figure is displayed.

```
HTTP/1.1 200 OK
content-type: text/html; charset=utf-8
content-length: 5719
server: istio-envoy
date: Tue, 02 Apr 2019 08:21:13 GMT
x-envoy-upstream-service-time: 24
```

- e. In your browser, enter `http://localhost:20001` to log on to the Kiali console to check how the target service is called.



6. Use the latest version of the target service to take over all the traffic that is destined for the earlier version.
 - a. Log on to the [Container Service console](#).
 - b. In the left-side navigation pane, choose Service Mesh > Virtual Service.
 - c. On the right of the target virtual service, click Manage.
 - d. In the Version Management area, find v2, and click Take over all the traffic in the Action column.

Version	Workload	Number of instances	Canary Deployment	Time Created	Action
v1	nginx	1	Traffic Weight 20%	04/26/2019,18:29:31	Manage Workload Delete Take over all the traffic Traffic Control
v2	nginx-v2	1	Traffic Weight 80%	04/28/2019,10:25:37	Manage Workload Delete Take over all the traffic Traffic Control

- e. In the displayed dialog box, click Confirm. Then, in the Canary Deployment column of v2, Traffic Weight 100% is displayed.

Version	Workload	Number of instances	Canary Deployment	Time Created	Action
v1	nginx	1	Traffic Weight 0%	04/26/2019,18:29:31	Manage Workload Delete Take over all the traffic Traffic Control
v2	nginx-v2	1	Traffic Weight 100%	04/28/2019,10:25:37	Manage Workload Delete Take over all the traffic Traffic Control

- f. Run the `kubectl exec -it -n demo < podName > bash` command to log on to the pod where the `sleep` application runs, and then run the following commands to call the `nginx` service:

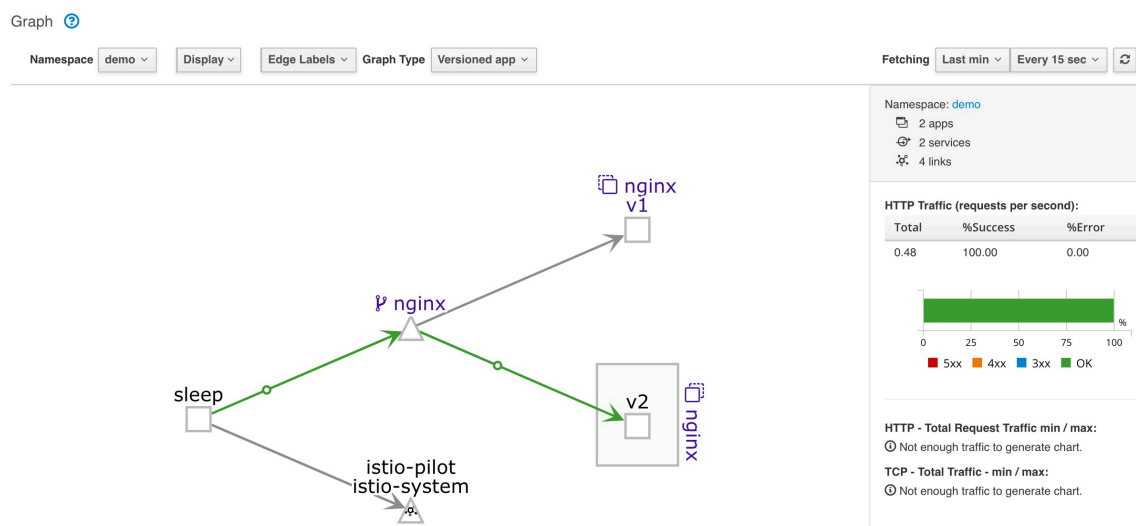
```
for i in `seq 1000`
do
  curl -I http://nginx.demo:8080 ;
  echo " ";
  sleep 1 ;
done
```

```
done ;
```

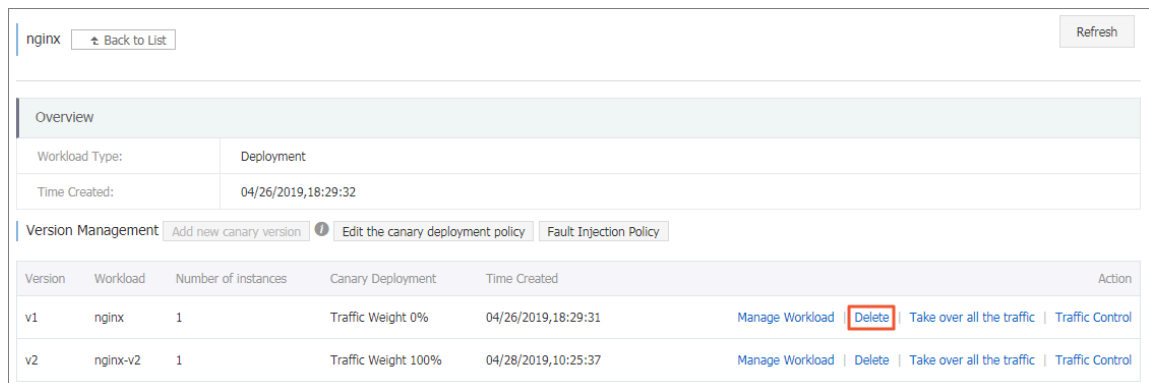
If the service is called, the content shown in the following figure is displayed.

```
HTTP/1.1 200 OK
content-type: text/html; charset=utf-8
content-length: 5719
server: istio-envoy
date: Tue, 02 Apr 2019 08:21:13 GMT
x-envoy-upstream-service-time: 24
```

g. In your browser, enter `http://localhost:20001` to log on to the Kiali console to check how the target service is called.

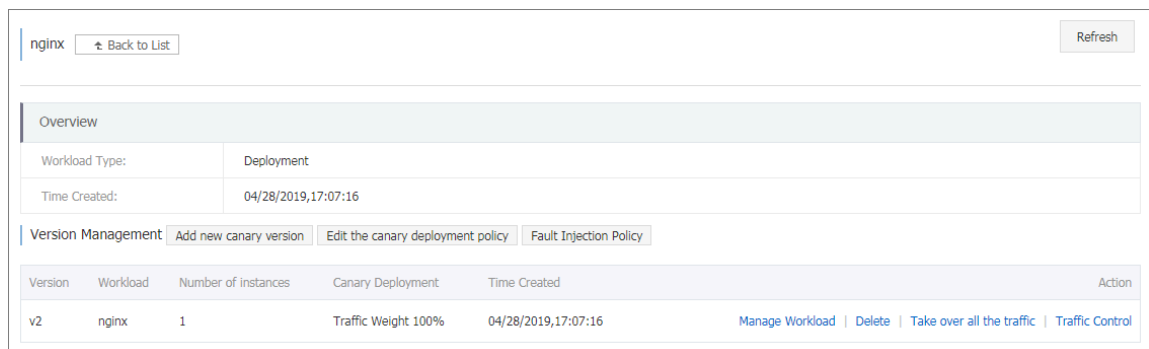


7. Delete the earlier version of the target service.
 - a. Log on to the [Container Service console](#).
 - b. In the left-side navigation pane under Container Service-Kubernetes, choose Service Mesh > Virtual Service.
 - c. On the right of the target virtual service, click Manage.
 - d. In the Version Management area, find v1, and click Delete in the Action column.



Version	Workload	Number of instances	Canary Deployment	Time Created	Action
v1	nginx	1	Traffic Weight 0%	04/26/2019,18:29:31	Manage Workload Delete Take over all the traffic Traffic Control
v2	nginx-v2	1	Traffic Weight 100%	04/28/2019,10:25:37	Manage Workload Delete Take over all the traffic Traffic Control

Then, only the latest version of the target service operates.



Version	Workload	Number of instances	Canary Deployment	Time Created	Action
v2	nginx	1	Traffic Weight 100%	04/28/2019,17:07:16	Manage Workload Delete Take over all the traffic Traffic Control



Note:

If you want to release one more version of the target service by using the Canary deployment method, you can repeat the preceding steps.

5.7 Use a VirtualService and DestinationRule to complete blue/green and canary deployments

This topic describes how to use the Istio resources `VirtualService` and `DestinationRule` to complete blue/green and canary deployments.

Before you begin

1. Deploy a `VirtualService` and `DestinationRule`.
 - a. Use [Cloud Shell](#) to connect to the target Kubernetes cluster.
 - b. Create the file `virtual-service.yaml`, copy the following code into the file, and then run the `kubectl apply -f virtual-service.yaml` command to deploy a `VirtualService`.

```

apiVersion : networking . istio . io / v1alpha3
kind : VirtualService
metadata :
  name : productpage
spec :
  hosts :
  - productpage
  http :
  - route :
    - destination :
        host : productpage
        subset : v1
---
apiVersion : networking . istio . io / v1alpha3
kind : VirtualService
metadata :
  name : reviews
spec :
  hosts :
  - reviews
  http :
  - route :
    - destination :
        host : reviews
        subset : v1
---
apiVersion : networking . istio . io / v1alpha3
kind : VirtualService
metadata :
  name : ratings
spec :
  hosts :
  - ratings
  http :
  - route :
    - destination :
        host : ratings
        subset : v1
---
apiVersion : networking . istio . io / v1alpha3
kind : VirtualService

```



```

metadata :
  name : details
spec :
  hosts :
  - details
  http :
  - route :
    - destination :
        host : details
        subset : v1

```

- c. Create the file `destination-rule.yaml`, copy the following code into the file, and then run the `kubectl apply -f destination-rule.yaml` command to deploy a `DestinationRule`.

```

apiVersion : networking.istio.io / v1alpha3
kind : DestinationRule
metadata :
  name : productpage
spec :
  host : productpage
  subsets :
  - name : v1
    labels :
      version : v1
---
apiVersion : networking.istio.io / v1alpha3
kind : DestinationRule
metadata :
  name : reviews
spec :
  host : reviews
  subsets :
  - name : v1
    labels :
      version : v1
  - name : v2
    labels :
      version : v2
  - name : v3
    labels :
      version : v3
---
apiVersion : networking.istio.io / v1alpha3
kind : DestinationRule
metadata :
  name : ratings
spec :
  host : ratings
  subsets :
  - name : v1
    labels :
      version : v1
  - name : v2
    labels :
      version : v2
  - name : v2 - mysql
    labels :
      version : v2 - mysql
  - name : v2 - mysql - vm
    labels :

```

```

      version : v2 - mysql - vm
---
apiVersion : networking . istio . io / v1alpha3
kind : DestinationRule
metadata :
  name : details
spec :
  host : details
  subsets :
  - name : v1
    labels :
      version : v1
  - name : v2
    labels :
      version : v2

```

2. Run the following command to set an Istio ingress gateway to forward traffic:

```

kubectl port - forward - n istio - system svc / istio -
ingressgateway 3000 : 80

```

3. In your browser, enter `localhost : 3000 / productpage` to access the Product page.

The Book Review area displays only comments without any ratings. This means that the Product page uses the version 1 of the review service.

Use the blue/green deployment to release the version 2 of the review service

After you complete this deployment, refresh the page. Then, the Book Review area displays ratings with black stars.

```

apiVersion : networking . istio . io / v1alpha3
kind : VirtualService
metadata :
  name : reviews
spec :
  hosts :
  - reviews
  http :
  - route :
    - destination :
        host : reviews
        subset : v2

```

Use the canary deployment to release the version 3 of the review service

The version 2 and version 3 of the review service can be provided at the same time to response to 50% traffic respectively.

Each time when you refresh the page, the Book Review area randomly displays V2 and V3 of the review service. Ratings of the V3 contains red stars.

```

apiVersion : networking . istio . io / v1alpha3
kind : VirtualService
metadata :

```

```

name : reviews
spec :
  hosts :
    - reviews
  http :
    - route :
        - destination :
            host : reviews
            subset : v2
            weight : 50
        - destination :
            host : reviews
            subset : v3
            weight : 50

```

Use the canary deployment to release the version 3 of the review service by HTTP headers

This method allows you to control the users to access to different versions of one page .

After you refresh the page for multiple times, the stars in the review area retain in black.

```

apiVersion : networking . istio . io / v1alpha3
kind : VirtualService
metadata :
  name : reviews
spec :
  hosts :
    - reviews
  http :
    - match :
        - headers :
            end-user :
              exact : jason
        route :
          - destination :
              host : reviews
              subset : v3
          - route :
              - destination :
                  host : reviews
                  subset : v2

```

In the upper-right corner, click Sign in, use jason to log on to the application (no password is required). The page displays that the stars in the review area are in red.



Note:

When you use the jason account to log on to the application to access the backend service, your requests are attached with HTTP headers of `end-user = XXX` . Then, your requests meet the rules defined by the preceding YAML file. Therefore, your requests are directed to the review service of V3.

6 Operation and maintenance

6.1 Check a Kubernetes cluster to troubleshoot exceptions

This topic describes how to quickly check a Kubernetes cluster in the Container Service console to troubleshoot exceptions.

Prerequisites

- A Kubernetes cluster is created. For more information, see [#unique_11](#).
- The Kubernetes cluster is in the running status.

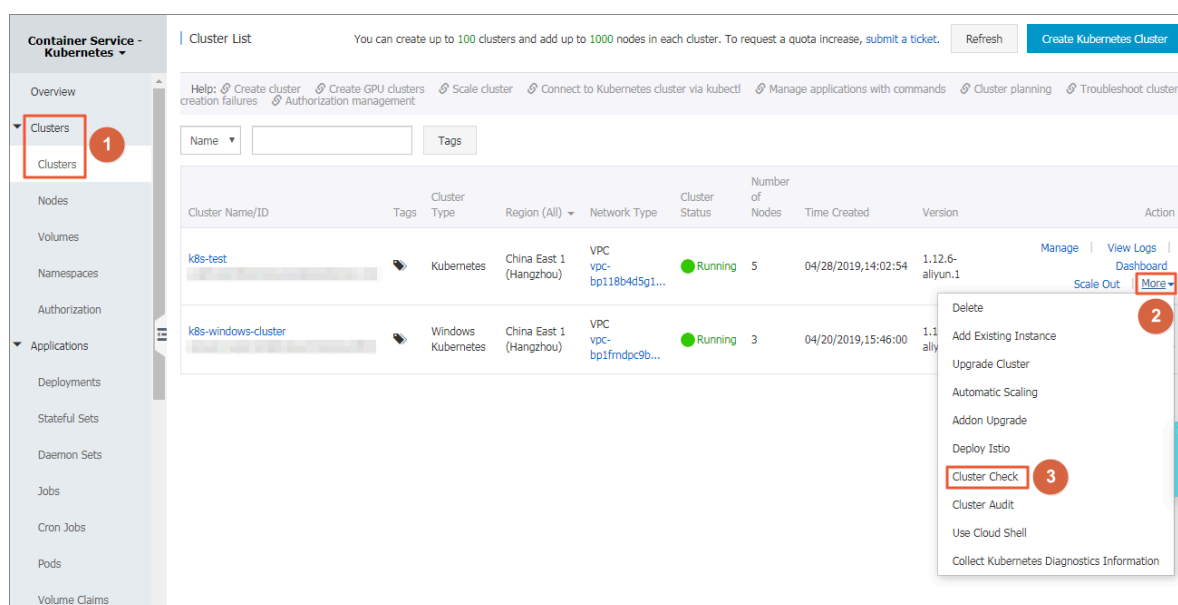


Note:

On the Cluster List page, you can check whether the Kubernetes cluster status is in the Running status.

Procedure

1. Log on to the [Container Service console](#).
2. In the left-side navigation pane under Container Service-Kubernetes, choose Clusters > Clusters. Find the target Kubernetes cluster, and choose More > Cluster Check in the action column.

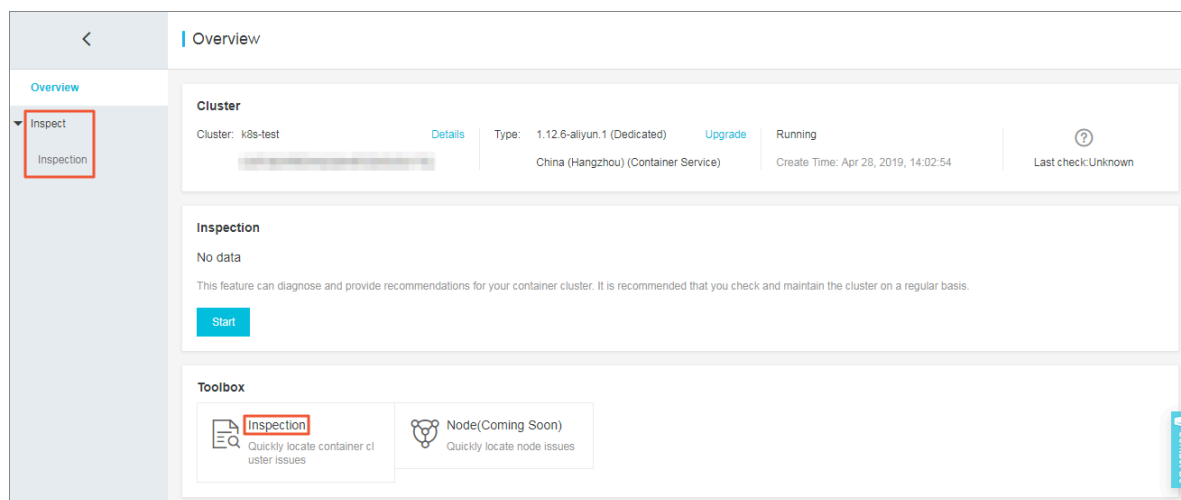


3. In the left-side navigation pane, choose Inspect > Inspection.



Note:

You can also click Inspection in the toolbox area.



- Cluster: displays the cluster name, type, status, and the result of the latest check
-
- Inspection: displays the corresponding time and results for the latest five cluster checks.



Note:

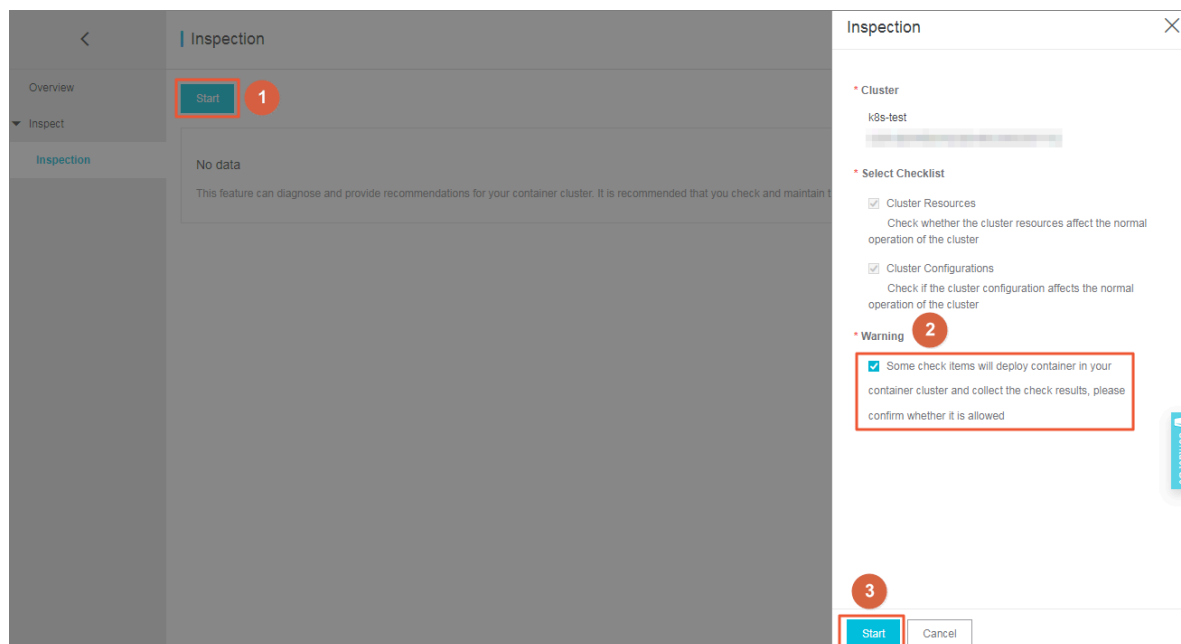
If you check a Kubernetes cluster for the first time, this area does not display any data except for the Start button. Clicking this button opens the the Inspection page where you can then enable cluster checking.

- Toolbox: displays the cluster check module.
4. Click Start. Then, in the displayed page, select the Warning check box, and click Start.

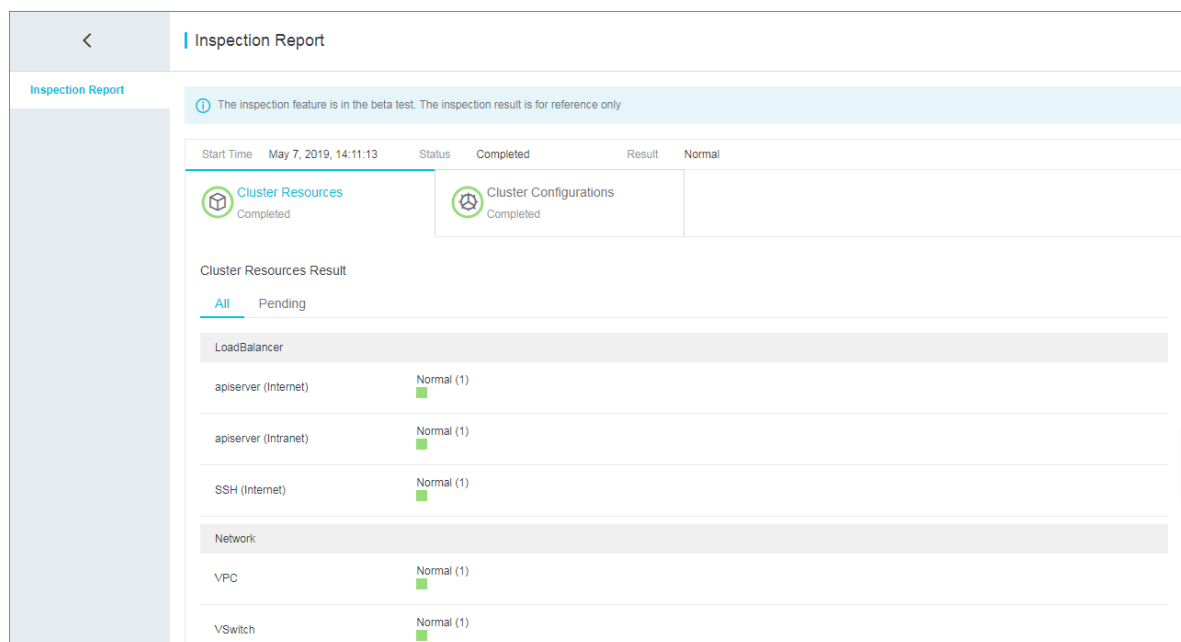


Note:

After you enable the cluster checks, the page displays the progress for the cluster checks in real time.

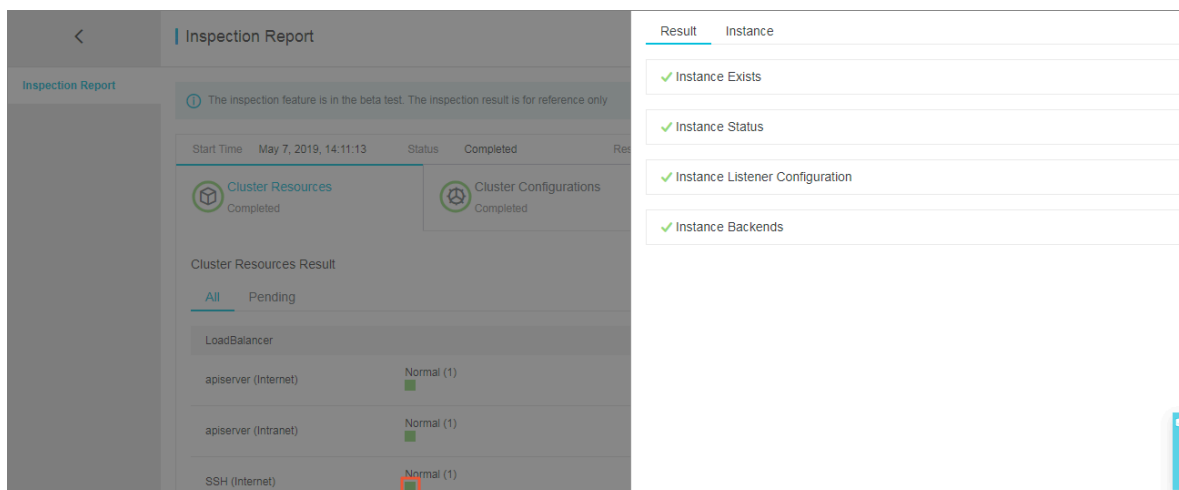


5. Click Details to view the results for the cluster resource and configuration checks.



6. Perform the action that corresponds to the result returned from the check.

- **Normal:** No action is required.
- **Warning:** Confirm the severity of the issue. If the issue may result in cluster exceptions, perform the appropriate actions to troubleshoot the issue.
- **Exception:** Perform the required actions to troubleshoot the issue immediately.



Note:

- For each resource or configuration (which contains result and instance information) in the result of a check, you can click the corresponding box icon on the right to view the instance and result information.
- If a resource or configuration is not in the normal state, the result displays the checked items, severity of exceptions, and recommended solutions. You can perform actions to troubleshoot as needed.

7 Serverless

7.1 Use a GPU container instance

This topic describes how to use a GPU container instance by using an example in which Tensorflow is used to identify a picture. ACK serverless Kubernetes provides GPU container instances by integrating with Elastic Container Instance (ECI). As a part of this service, GPU container instances can be used in a serverless Kubernetes cluster or on a virtual node of a dedicated or managed Kubernetes cluster.

Billing

GPU and CPU resources provided by ACK serverless Kubernetes are charged with the Pay-As-You-Go billing method.

GPU instances provided by ACK serverless Kubernetes are charged the same as the ECS GPU instances. You are not charged additional fees.

Overview

In an ACK serverless Kubernetes cluster, you can easily create a pod to which a GPU is attached. Specifically, for the pod configurations, you must set the `annotation s` parameter to specify the GPU type that you want, and set the `limits` parameter to specify the number of GPU.



Note:

A GPU is exclusive to one pod.

Prerequisites

Either a serverless Kubernetes cluster is created, or a virtual node is created for a Kubernetes cluster. For more information, see [#unique_62](#) or [#unique_63](#).

Procedure

In the following steps, Tensorflow is used in a serverless Kubernetes cluster to identify the following picture.



1. Log on to the [Container Service console](#).
2. In the left-side navigation pane under Container Service-Kubernetes, choose Applications > Deployments. Then, in the upper-right corner, click Create by Template.
3. Select the target cluster and namespace, select an example template or customize a template, and then click DEPLOY.

You can use the following YAML template to create a pod:



Note:

In this pod template, the GPU type is set to P4 , and the number of GPU is set to 1.

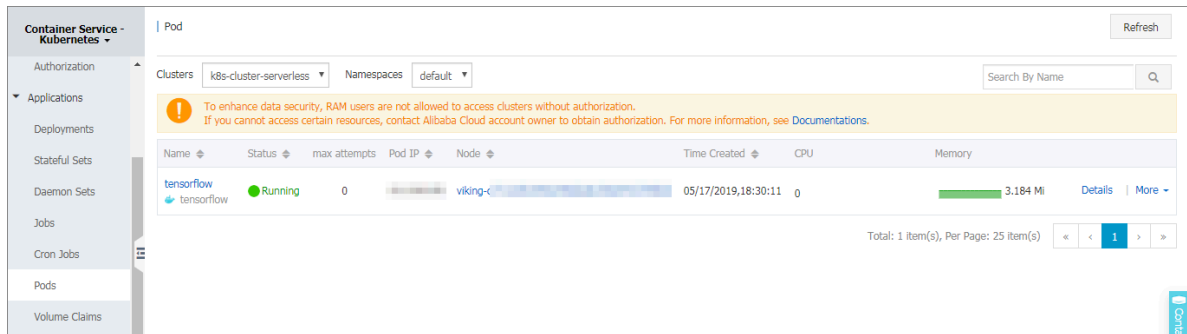
```
apiVersion : v1
kind : Pod
metadata :
  name : tensorflow
  annotations :
    k8s.aliyun.com/eci-gpu-type : "P4"
spec :
  containers :
    - image : registry-vpc.cn-hangzhou.aliyuncs.com/ack-serverless/tensorflow
      name : tensorflow
      command :
        - "sh"
        - "-c"
        - "python models/tutorials/image/imagenet/classify_image.py"
      resources :
        limits :
          nvidia.com/gpu : "1"
        restartPolicy : OnFailure
```

4. In the left-side navigation pane under Container Service-Kubernetes, choose Applications > Pods to verify that the pod is created.

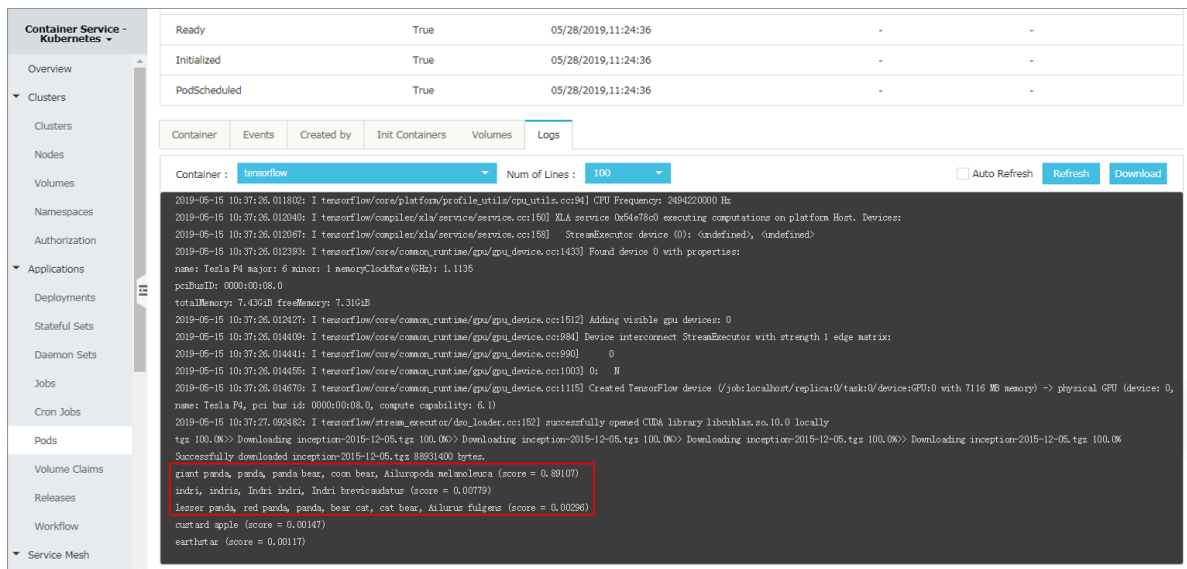


Note:

It may take a few minutes to create a pod. If you the pod is not displayed, you can refresh the page.



- Click the target pod, and click the Logs tab to verify that the picture is identified according to the logs marked in the red frame.



If you want to use a GPU container instance on a virtual node of a dedicated or managed Kubernetes cluster, follow these steps:



Note:

A GPU container instance used on a virtual node supports a larger variety of deep learning platforms such as Kubeflow, Arena, or other customized CRD.

- Create a pod on the virtual node or in a namespace to which the `virtual - node - affinity - injection = enabled` label tag is added. For more information, see [#unique_63](#).
- Replace the template used in step 3 with the following template:

```
apiVersion : v1
kind : Pod
metadata :
```

```
name : tensorflow
annotation s :
  k8s . aliyun . com / eci - gpu - type : " P4 "
spec :
  containers :
    - image : registry - vpc . cn - hangzhou . aliyuncs . com / ack
      - serverless / tensorflow
        name : tensorflow
        command :
          - " sh "
          - "- c "
          - " python  models / tutorials / image / imagenet / classify_i
            mage . py "
        resources :
          limits :
            nvidia . com / gpu : " 1 "
          restartPol icy : OnFailure
          nodeName : virtual - kubelet
```

7.2 Associate an EIP address with a pod

This topic describes how to associate an EIP address with a pod in a serverless Kubernetes cluster or with a pod on a virtual node of a dedicated or managed Kubernetes cluster.

Benefits

Associating an EIP address with a target pod offers the following benefits:

- You can easily deploy an application that uses serverless containers.
- Services provided by the application can be more quickly accessed.
- Your pod can access the Internet without the need of a VPC NAT gateway.
- Your target application can be accessed through the Internet without the need of an SLB instance.

Prerequisites

- Either a serverless Kubernetes cluster is created, or a virtual node is created in a dedicated or managed Kubernetes cluster. For more information, see [#unique_62](#) or [#unique_63](#).
- The corresponding port for the target application that you want to run is enabled in the security group of the target Kubernetes cluster.



Note:

The Nginx application is used as an example. Therefore, port 80 must be enabled in the security group of the target Kubernetes cluster.

Procedure

1. Log on to the VPC console to create an EIP address. For more information, see [#unique_65](#).



Note:

You must create an EIP address in the same region where the target Kubernetes cluster is located.

Instance ID/Name	IP Address	Monitor	Bandwidth	Connection Type	Charge Type(All)	Status(All)	Shared Bandwidth/... Acceleration	Bind Instance	Instance Type(All)	Resource Group	Actions
[Redacted]	[Redacted]	[Icon]	1 Mbps Pay By Traffic	BGP	Pay-As-You-Go 09/28/2019, 20:31:45 Created	Available	Add to Shared Bandwidth Package Add to Global Acceleration	-	-	-	Bind Unbind More

2. Log on to the [Container Service console](#).
3. In the left-side navigation pane under Container Service-Kubernetes, choose Applications > Deployments.
4. In the upper-right corner, click Create by Template.

5. Select the target cluster and namespace, select an example template or customize a template, and then click DEPLOY.

Create with Template

Only Kubernetes versions 1.8.4 and above are supported. For clusters of version 1.8.1, you can perform "upgrade cluster" operation in the cluster list

Clusters: managed-cluster

Namespaces: kube-system

Sample Template: Custom

Template

```

1 apiVersion: v1
2 kind: Pod
3 metadata:
4   name: nginx
5   annotations:
6     "k8s.aliyun.com/eci-eip-instanceid": "<yourEIPInstanceID>"
7 spec:
8   containers:
9     - image: registry-vpc.cn-hangzhou.aliyuncs.com/jovi/nginx:alpine
10     imagePullPolicy: Always
11     name: nginx
12     ports:
13       - containerPort: 80
14         name: http
15         protocol: TCP
16     restartPolicy: OnFailure

```

Add Deployment
Use Existing Template

Save Template DEPLOY

You can use the following YAML template to create a pod:



Note:

You must replace `< yourEIPInstanceID >` with the EIP ID obtained in step 1.

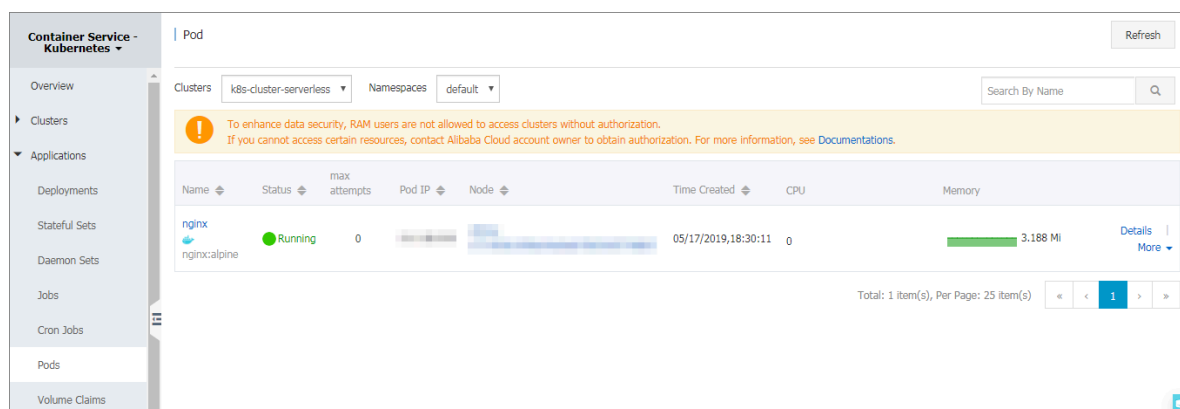
```

apiVersion : v1
kind : Pod
metadata :
  name : nginx
  annotations :
    " k8s . aliyun . com / eci - eip - instanceid " : "< yourEIPInstanceID >"
spec :
  containers :
    - image : registry - vpc . cn - hangzhou . aliyuncs . com / jovi / nginx : alpine
      imagePullPolicy : Always
      name : nginx
      ports :
        - containerPort : 80
          name : http
          protocol : TCP

```

```
restartPolicy : OnFailure
```

6. In the left-side navigation pane under Container Service-Kubernetes, choose **Applications > Pods** to view the pod status.

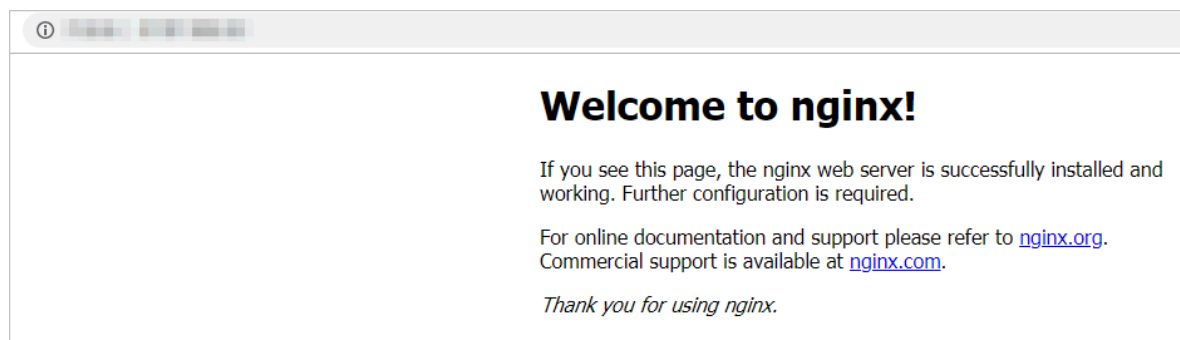


7. In your browser, enter the URL `http://ip address` to access the welcome page of Nginx.



Note:

For this URL, you must enter the IP address obtained in step 1.



8 Auto Scaling

8.1 Deploy an Ingress application on a virtual node

This topic describes how to deploy an Ingress application on a virtual node of a Kubernetes cluster. With the virtual node, the Kubernetes cluster can provide the application with greater computing capability without the need for a new node needs to be created for the cluster.

Prerequisites

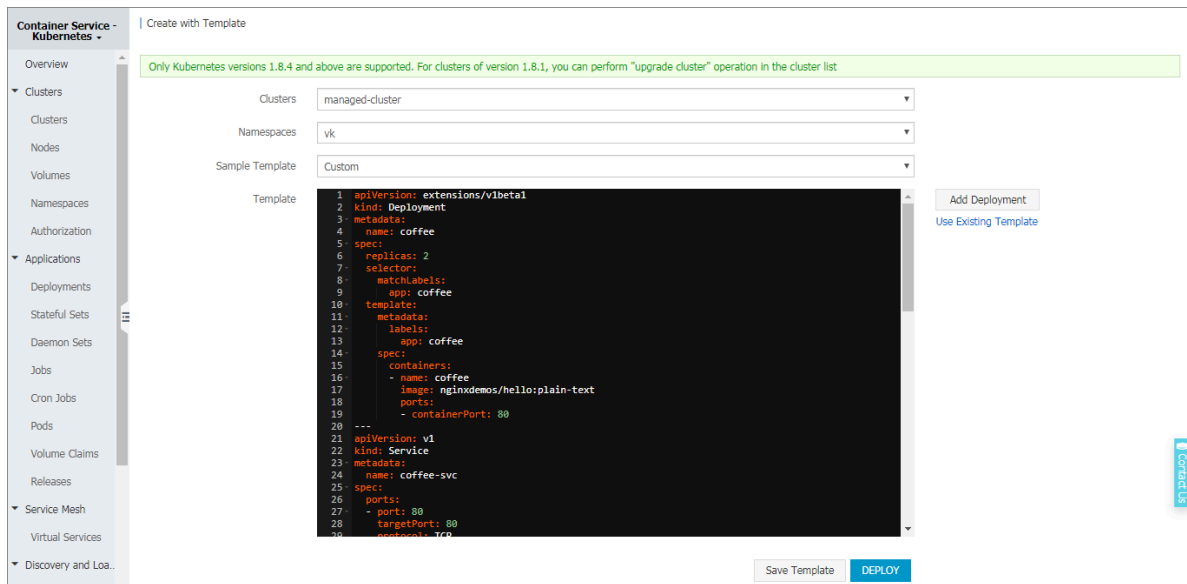
A virtual node is deployed in your target Kubernetes cluster. For more information, see [#unique_63](#).

The tag `virtual - node - affinity - injection: enabled` is added to the target namespace. For more information, see [Set a namespace tag to create a pod](#).

Procedure

1. Log on to the [Container Service console](#).
2. In the left-side navigation pane under Container Service-Kubernetes, choose Applications > Deployments.
3. In the upper-right corner, click Create by Template.

4. Select the target cluster and namespace, select an example template or customize a template, and then click **DEPLOY**.



You can use the following template to customize a YAML template to create an Ingress application:

```
apiVersion : extensions / v1beta1
kind : Deployment
metadata :
  name : coffee
spec :
  replicas : 2
  selector :
    matchLabels :
      app : coffee
  template :
    metadata :
      labels :
        app : coffee
    spec :
      containers :
        - name : coffee
          image : nginxdemos / hello : plain - text
          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 : extensions / v1beta1
```



```

kind : Deployment
metadata :
  name : tea
spec :
  replicas : 3
  selector :
    matchLabels :
      app : tea
  template :
    metadata :
      labels :
        app : tea
    spec :
      containers :
        - name : tea
          image : nginxdemos / hello : plain - text
          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
---
apiVersion : extensions / v1beta1
kind : Ingress
metadata :
  name : cafe - ingress
spec :
  rules :
    - host : cafe . example . com
      http :
        paths :
          - path : / tea
            backend :
              serviceName : tea - svc
              servicePort : 80
          - path : / coffee
            backend :
              serviceName : coffee - svc

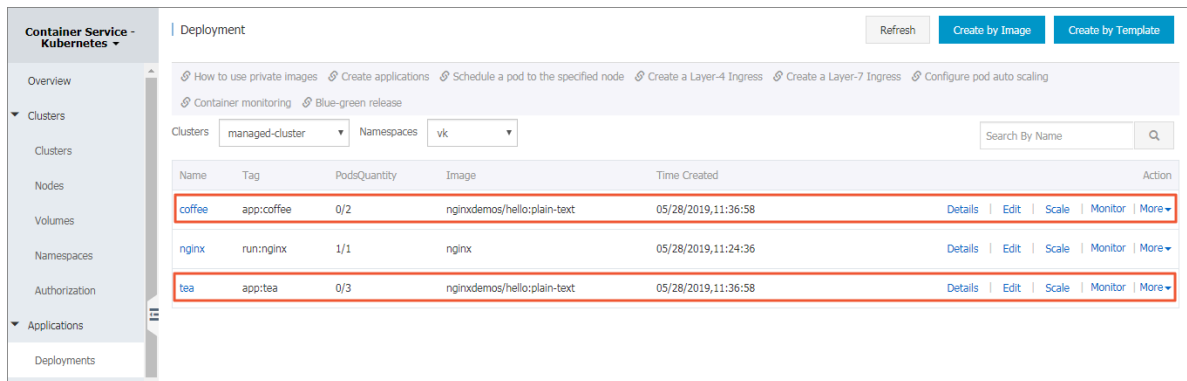
```

```
servicePort : 80
```

Verify the results

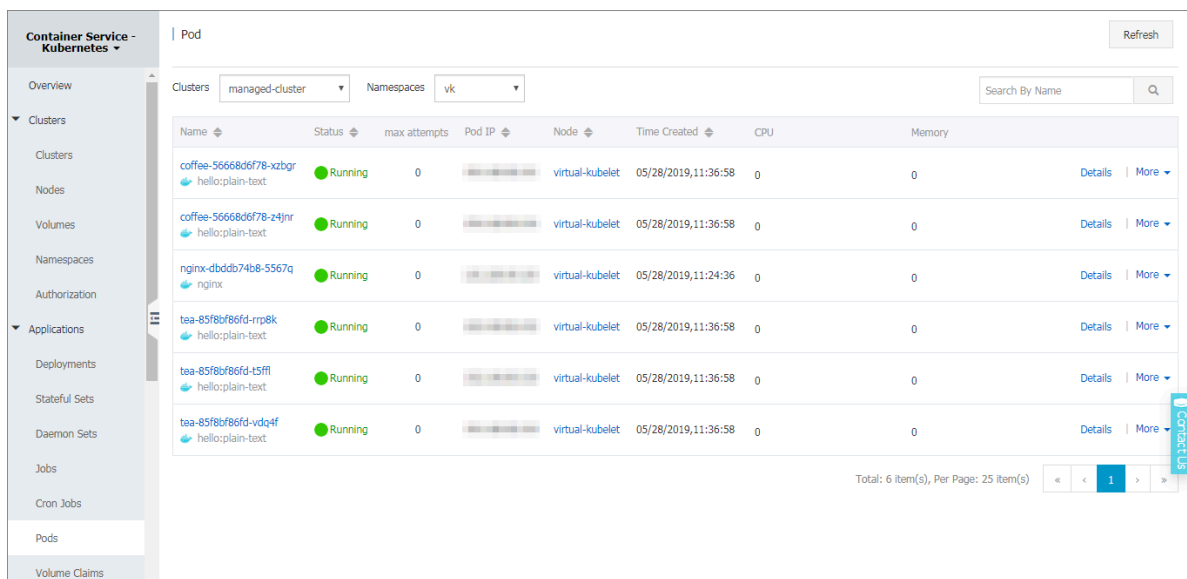
- To verify that the `coffee` and `tea` deployments are created, follow these steps:

- In the left-side navigation pane under Container Service-Kubernetes, choose **Applications > Deployments**.
- Select the target cluster and namespace.

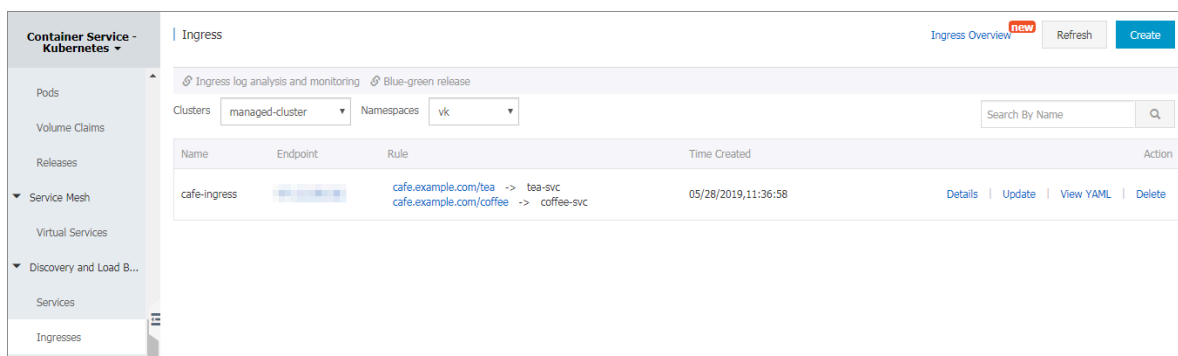


- To verify that all the pods of the Ingress application run on the virtual-kubelet node, follow these steps:

- In the left-side navigation pane under Container Service-Kubernetes, choose **Applications > Pods**.
- Select the target cluster and namespace.



- To verify that the target Ingress is created, follow these steps:
 1. In the left-side navigation pane under Container Service-Kubernetes, choose **Discovery and Load Balancing > Ingresses**.
 2. Select the target cluster and namespace.



- To verify that the created Ingress application can be accessed, run the following commands:

```
kubectl -n vk get ing
curl -H "Host: cafe.example.com" < EXTERNAL_IP > /tea
curl -H "Host: cafe.example.com" < EXTERNAL_IP > /coffee
```

```
shell@Alicloud:~$ kubectl -n vk get ing
NAME          HOSTS          ADDRESS          PORTS    AGE
cafe-ingress  cafe.example.com  [REDACTED]      80       6m33s
shell@Alicloud:~$ curl -H "Host:cafe.example.com" [REDACTED]/tea
Server address: [REDACTED]
Server name: vk-tea-[REDACTED]
Date: 28/May/2019:03:45:48 +0000
URI: /tea
Request ID: 0a78cffe790dd19ceea022255c7b8730
shell@Alicloud:~$ curl -H "Host:cafe.example.com" [REDACTED]/coffee
Server address: [REDACTED]
Server name: vk-coffee-[REDACTED]
Date: 28/May/2019:03:46:52 +0000
URI: /coffee
Request ID: 977d8d6e38869d5a9e5603b6fdd2b5ed
shell@Alicloud:~$
```

9 DevOps

9.1 Deploy Jenkins in a Kubernetes cluster and perform a pipeline build

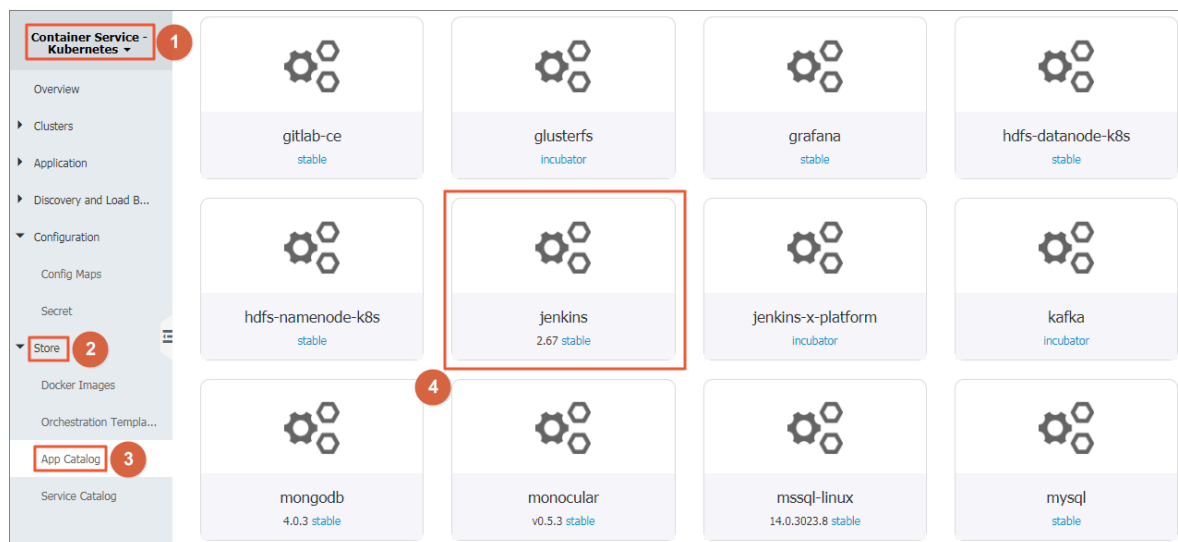
This topic describes how to deploy Jenkins, a continuous integration environment, in an Alibaba Cloud Kubernetes cluster, and how to perform an application pipeline build. The example in this topic details the pipeline build, including how to compile the source code of the application, build and push the application image, and deploy the application.

Prerequisites

You have created a Kubernetes cluster. For more information, see [Create a Kubernetes cluster](#).

Deploy Jenkins

1. Log on to the [Container Service console](#).
2. In the left-side navigation pane, choose Container Service-Kubernetes > Store > App Catalog. Then, click jenkins.



3. Click the Values tab.
4. Modify the `AdminPassword` field to set a password.



Note:

To ensure that your password takes effect, you must remove the pound sign (#) before `AdminPassword` ord

If you do not set any password, the system generates a password after Jenkins is deployed. You can run the following command to view the password:

```
$ printf $( kubectl get secret -- namespace ci jenkins -
ci - jenkins - o jsonpath="{. data . jenkins - admin - password
}" | base64 -- decode ); echo
```

5. Select the target Cluster and Namespace, enter a Release Name, then click DEPLOY.

App Catalog - jenkins

jenkins
stable
Open source continuous integration server. It supports multiple SCM tools including CVS, Subversion and Git. It can execute Apache Ant and Apache Maven-based projects as well as arbitrary scripts.

Readme Values

```
1- # Default values for jenkins.
2- # This is a YAML-formatted file.
3- # Declare name/value pairs to be passed into your templates.
4- # name: value
5-
6- Master:
7-   Name: jenkins-master
8-   Image: "registry.cn-beijing.aliyuncs.com/acs-sample/jenkins-master"
9-   ImageTag: "2.150.1"
10-  ImagePullPolicy: "Always"
11-  Component: "jenkins-master"
12-  UseSecurity: true
13-  AdminUser: admin
14-  #AdminPassword: <defaults to random>
15-  Cpu: "200m"
16-  Memory: "256Mi"
17-  # Set min/max heap here if needed with:
18-  # JavaOpts: "-Xms512m -Xmx512m"
19-  # JenkinsOpts: ""
20-  # JenkinsUriPrefix: "/jenkins"
21-  ServicePort: 8080
22-  # For minikube, set this to NodePort, elsewhere use LoadBalancer
23-  # Use ClusterIP if your setup includes ingress controller
24-  ServiceType: LoadBalancer
25-  # Master Service annotations
26-  ServiceAnnotations: {}
27-  # service.beta.kubernetes.io/aws-load-balancer-backend-protocol: https
28-  # Used to create Ingress record (should used with ServiceType: ClusterIP)
29-  # HostName: jenkins.cluster.local
30-  # NodePort: <to set explicitly, choose port between 30000-32767
31-  ContainerPort: 8080
32-  SlaveListenerPort: 50000
```

Deploy

Only Kubernetes versions 1.8.4 and above are supported. For clusters of version 1.8.1, you can perform "upgrade cluster" operation in the cluster list

Clusters
k8s-cluster

Namespace
test

Release Name
jenkins-demo

DEPLOY

Version
0.9.0



Note:

We recommend that you select a custom namespace or the default namespace. In this example, a custom namespace named test is selected.

6. In the left-side navigation pane, choose Discovery and Load Balancing > Service.

7. Select the cluster and the namespace used for deploying Jenkins.

8. Click the external endpoint of the Jenkins service to log on to Jenkins.

The screenshot shows the 'Service' page in the Alibaba Cloud Container Service for Kubernetes console. The left sidebar contains navigation options: CronJob, Pods, Volumes Claim, Release (1), Discovery and Load B..., Service (2), Ingress, Configuration, Config Maps, Secret, Store, and Docker Images. The main area displays a table of services. The table has columns: Name, Type, Time Created, ClustersIP, InternalEndpoint, ExternalEndpoint, and Action. The first row is for 'jenkins-demo-jenkins' (LoadBalancer type). Its ExternalEndpoint is 'jenkins-demo-jenkins:8080 TCP' and 'jenkins-demo-jenkins:30004 TCP'. A red box (4) highlights the 'jenkins-demo-jenkins:8080 TCP' endpoint. Above the table, there are filters for Clusters (k8s-cluster) and Namespace (test) (3), and a search bar. A 'Refresh' button and a 'Create' button are at the top right.

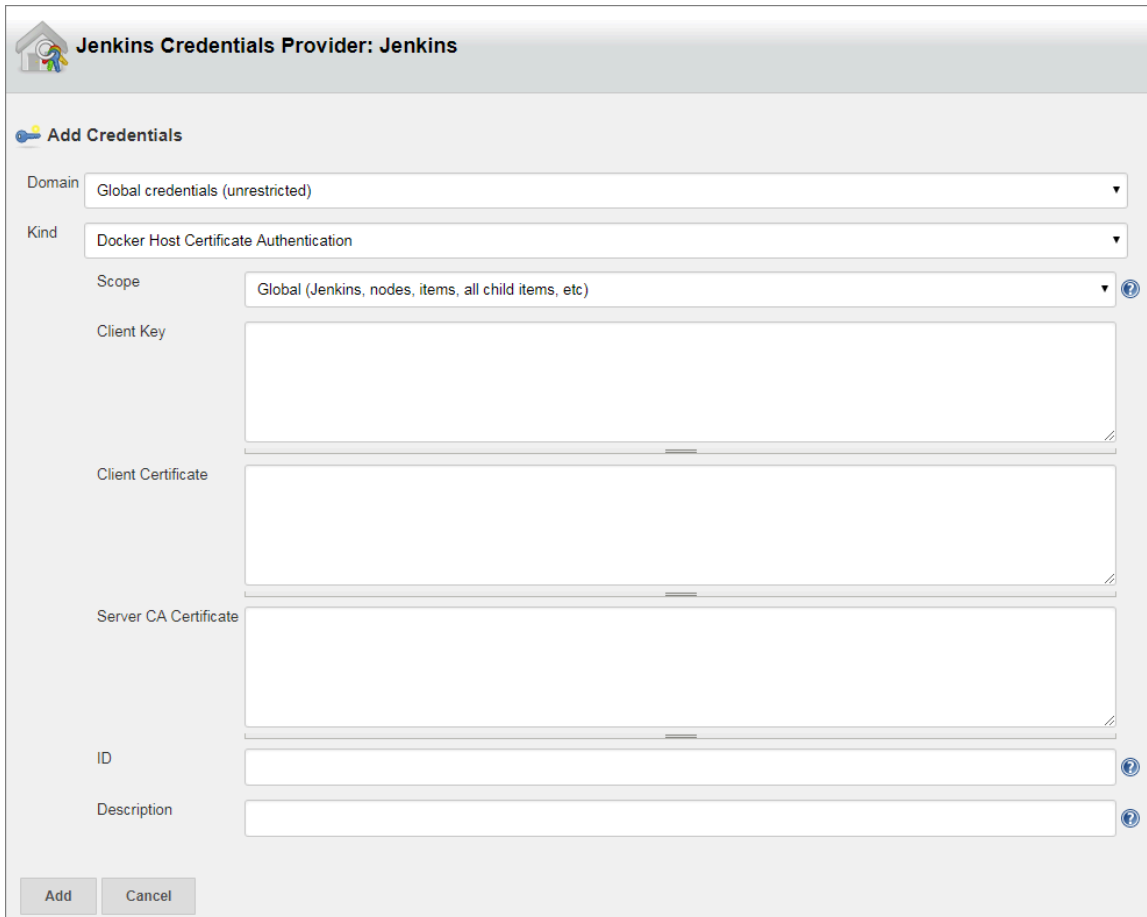
Name	Type	Time Created	ClustersIP	InternalEndpoint	ExternalEndpoint	Action
jenkins-demo-jenkins	LoadBalancer	02/18/2019,14:47:08	172.19.8.69	jenkins-demo-jenkins:8080 TCP jenkins-demo-jenkins:30004 TCP	jenkins-demo-jenkins:8080 TCP jenkins-demo-jenkins:30004 TCP	Details Update View YAML Delete
jenkins-demo-jenkins-agent	ClusterIP	02/18/2019,14:47:08	172.19.1.20	jenkins-demo-jenkins-agent:50000 TCP	-	Details Update View YAML Delete

Create a cluster certificate and an image repository certificate, and build and deploy an application

1. Create a Kubernetes cluster certificate.
 - a. In the left-side navigation pane, click Manage Jenkins.
 - b. On the Manage Jenkins page, click Configure System.
 - c. In the Cloud area, click Add on the right of Credentials.

Before adding a credential, you must obtain KubeConfig on the Basic Information tab page of the target Kubernetes cluster.

- d. In the displayed dialog box, set the following parameters:



Jenkins Credentials Provider: Jenkins

Add Credentials

Domain: Global credentials (unrestricted)

Kind: Docker Host Certificate Authentication

Scope: Global (Jenkins, nodes, items, all child items, etc)

Client Key

Client Certificate

Server CA Certificate

ID

Description

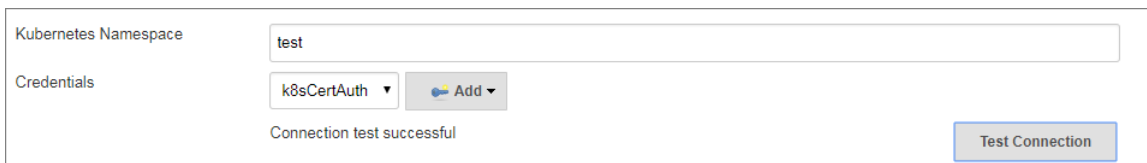
Add Cancel

- **Kind:** Select Docker Host Certificate Authentication.
- **Client Key:** Paste the copied client-key-data content in KubeConfig.
- **Client Certificate:** Paste the copied client-certificate-data content in KubeConfig.
- **ID:** Enter the certificate ID. In this example, k8sCertAuth is entered.
- **Description :** Enter description content.

e. Click Add.

f. Test connectivity.

Select the added credential in the preceding step from the Credentials drop-down list, and then click Test Connection.



Kubernetes Namespace: test

Credentials: k8sCertAuth Add

Connection test successful

Test Connection

g. Set the Kubernetes cluster to dynamically build pods as follows.

Cloud

Kubernetes

Name:

Kubernetes URL:

Kubernetes server certificate key:

Disable https certificate check: ☒

Kubernetes Namespace:

Credentials:

Connection test successful

Jenkins URL:

Jenkins tunnel:

Connection Timeout:

Read Timeout:

Container Cap:

h. Set the Kubernetes pod template.

The slave-pipeline uses four containers to create each corresponding stage of the pipeline.

- Set container jnlp as follows.

Kubernetes Pod Template

Name:

Namespace:

Labels:

Usage:

The name of the pod template to inherit from:

Containers

Container Template

Name:

Docker image:

Always pull image: ☐

Working directory:

Command to run:

Arguments to pass to the command:

Allocate pseudo-TTY: ☐

EnvVars:

List of environment variables to set in agent pod

Use `jenkins-slave-jnlp` as a Docker image. The `jenkins-slave-jnlp` image is used to create the `jnlp` node to connect the Jenkins master.

```
registry.cn-beijing.aliyuncs.com/acs-sample/jenkins-slave-jnlp:3.14-1
```

- Set container `kaniko` as follows.

Container Template

Name

kaniko

?

Docker image

registry.cn-beijing.aliyuncs.com/ac

?

Always pull image

☐

Working directory

/home/jenkins

?

Command to run

/bin/sh -c

?

Arguments to pass to the command

cat

?

Allocate pseudo-TTY

☒

EnvVars

Add Environment Variable

Variable


▼

List of environment variables to set in agent pod

Use `jenkins-slave-kaniko` as a Docker image. The `jenkins-slave-kaniko` image is used to create and push an image.

```
registry.cn-beijing.aliyuncs.com/acs-sample/jenkins-slave-kaniko:0.6.0
```

- Set container `kubectl` as follows.

 **Container Template**

Name	<input type="text" value="kubectl"/> ?
Docker image	<input type="text" value="registry.cn-beijing.aliyuncs.com/ac"/> ?
Always pull image	<input type="checkbox"/>
Working directory	<input type="text" value="/home/jenkins"/> ?
Command to run	<input type="text" value="/bin/sh -c"/> ?
Arguments to pass to the command	<input type="text" value="cat"/> ?
Allocate pseudo-TTY	<input checked="" type="checkbox"/>
EnvVars	<div><div>Add Environment Variable ▼</div><div>List of environment variables to set in agent pod</div></div>

Use `jenkins-slave-kubectl` as a Docker image. The `jenkins-slave-kubectl` image is used to deploy the application.

```
registry . cn - beijing . aliyuncs . com / acs - sample /  
jenkins - slave - kubectl : 1 . 11 . 5
```

- Set container maven as follows.

Container Template

Name

maven

?

Docker image

registry.cn-beijing.aliyuncs.com/ac

?

Always pull image

☐

Working directory

/home/jenkins

?

Command to run

/bin/sh -c

?

Arguments to pass to the command

cat

?

Allocate pseudo-TTY

☒

EnvVars

Add Environment Variable

▼

List of environment variables to set in agent pod

Use `jenkins-slave-maven` as a Docker image. The `jenkins-slave-maven` image is used by `mvn` to package and build the application.

```
registry.cn-beijing.aliyuncs.com/acs-sample/jenkins-slave-maven:3.3.9-jdk-8-alpine
```

- i. Set image repository permission for kaniko as follows.

Add Container ▼

List of container in the agent pod

Environment Variable

Key

DOCKER_CONFIG

?

Value

/home/jenkins/.docker

?

Delete Environment Variable

Add Environment Variable ▼

List of environment variables to set in all container of the pod

Secret Volume

Secret name

jenkins-docker-cfg

?

Mount path

/home/jenkins/.docker

?

Delete Volume

j. Click Save.

2. To set image repository permission, use `kubectl` to create `jenkins-docker-cfg` secret in the target Kubernetes cluster.

In this example, the Beijing image repository provided by Alibaba Cloud is used.

```
$ docker login -u xxx -p xxx registry.cn-beijing.aliyuncs.com
Login Succeeded
```

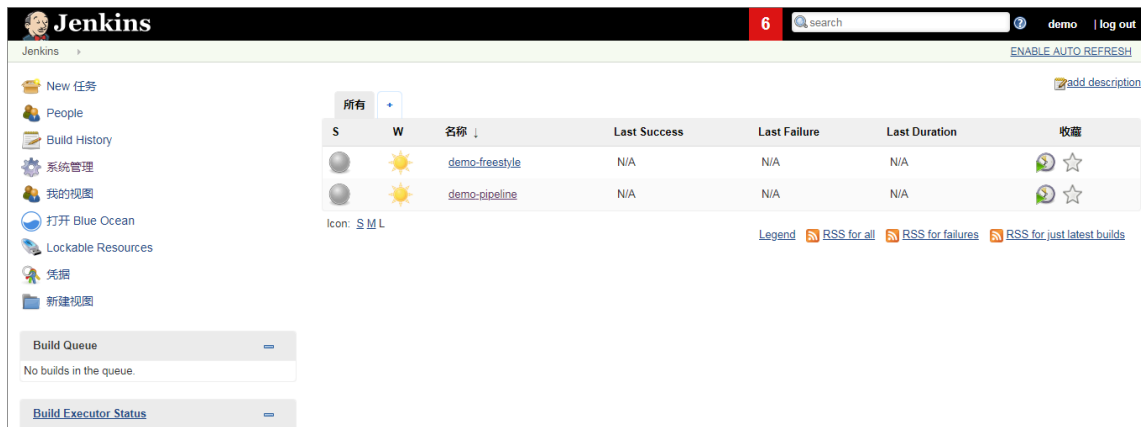
Issue: 20190819

167

```
$ kubectl create secret generic jenkins - docker - cfg - n
ci -- from - file =/ root /. docker / config . json
```

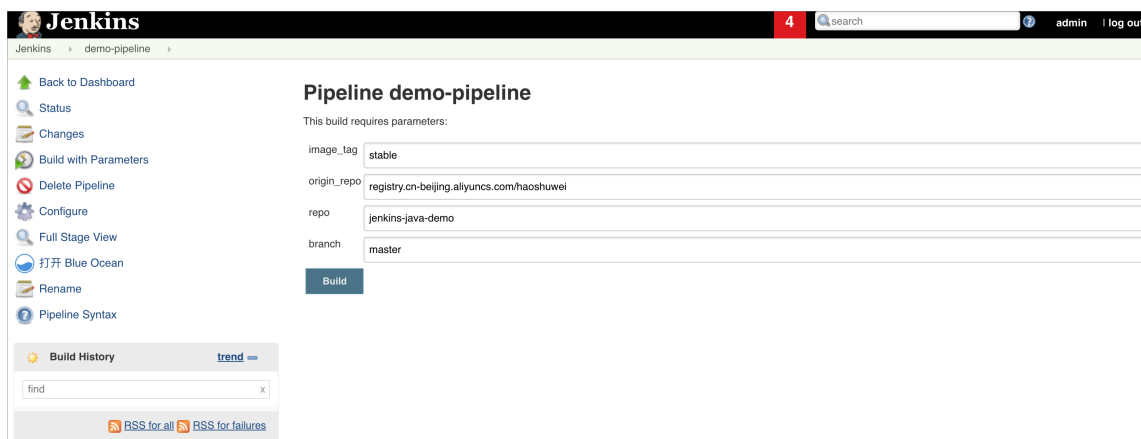
3. Create demo-pipeline and access the application service.

a. On the Jenkins home page, click demo-pipeline.



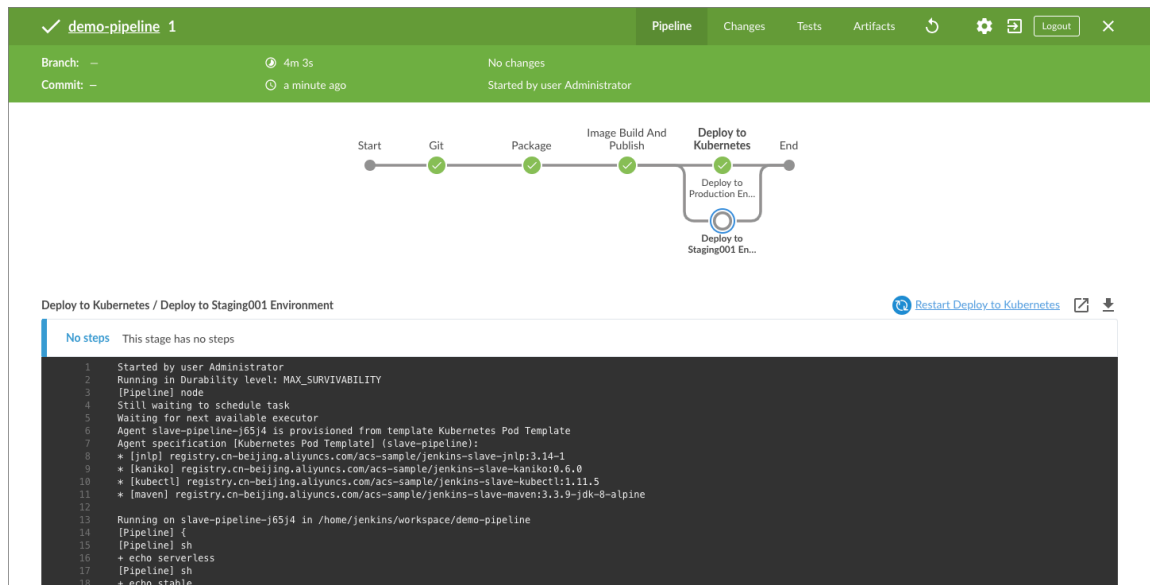
b. In the left-side navigation pane, click Build with Parameters.

c. Modify the parameters according to your image repository information. In this example, the source code repository is master, and the image is registry.cn-beijing.aliyuncs.com/haoshuwei:stable.



d. Click Build.

e. Click Build History to check the result. The following figure indicates a success.



- f. Log on to the [Container Service console](#) to view the IP addresses of the services provided by the application.

Click [Here](#) to obtain the source code repository used in the example.

For more information, see [Container Service](#).

For more information, see [kaniko](#).

9.2 Use GitLab CI to run a GitLab runner and activate a pipeline in a Kubernetes cluster

This topic describes how to install and register a GitLab runner in a Kubernetes cluster and add a Kubernetes executor to build an application. After completing the preceding operations, you can build a Java application source code project. The example in this topic shows a CI/CD pipeline that includes the following actions in order: compiling and building the code, packaging the application image, and deploying the application.

Background

In the following example, a Java software project is built and deployed in a Kubernetes cluster that runs on Alibaba Cloud Container Service for Kubernetes. The example shows how to use GitLab CI to run a GitLab runner, set a Kubernetes executor, and activate a pipeline in an Alibaba Cloud Kubernetes cluster.

Create the GitLab source code project and upload the example code

1. Create the GitLab source code project.

In this example, the address of the created GitLab source code project is as follows:

```
http://xx.xx.xx.xx/demo/gitlab-java-demo.git
```

2. Run the following commands to obtain the example code and upload it to GitLab:

```
$ git clone https://code.aliyun.com/CodePipeline/gitlabci-java-demo.git
$ git remote add gitlab http://xx.xx.xx.xx/demo/gitlab-java-demo.git
```



```
$ git push gitlab master
```

Install the GitLab runner in a Kubernetes cluster

1. Obtain the GitLab runner registration information.

a. Obtain the registration information of the runner specific to the project.

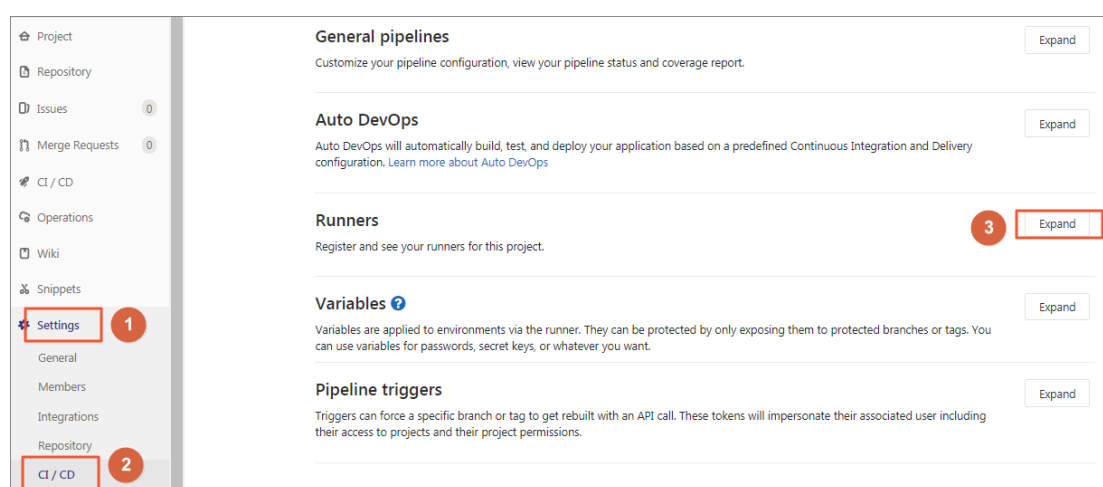
A. Log on to GitLab.

B. In the top navigation bar, choose Projects > Your projects.

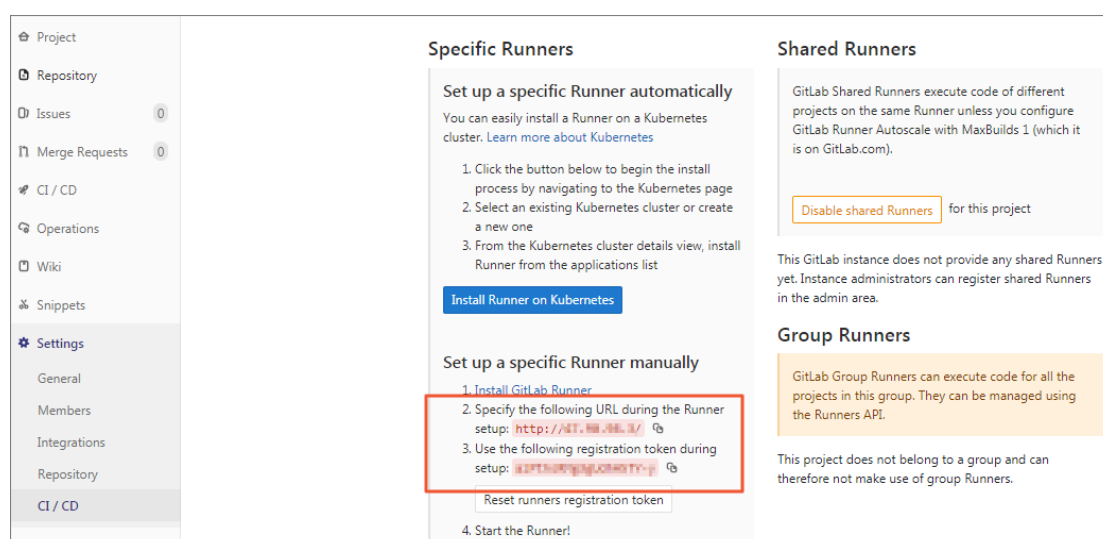
C. On the **Your projects** tab page, select the target project.

D. In the left-side navigation pane, choose Settings > CI / CD.

E. Click Expand on the right of Runners.



F. Obtain the URL and the registration token.



b. Obtain the group runner registration information.

A. In the top navigation bar, choose Groups > Your groups.

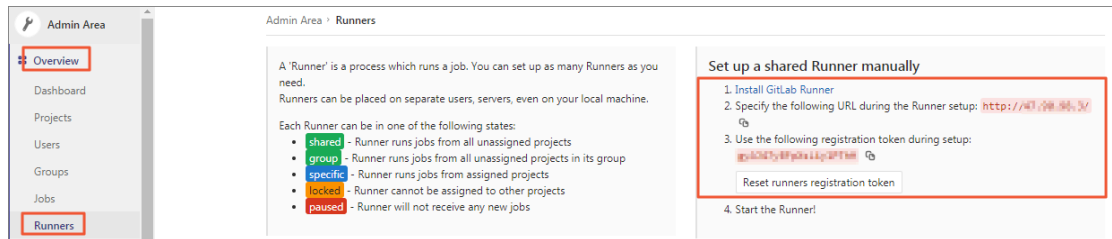
B. On the Your groups tab page, select the target group.

Only the administrator has permission to perform this action.

A. In the top navigation bar, click .

B. In the left-side navigation pane of the Admin Area page, choose Overview > Runners.

C. Obtain the URL and the registration token.



2. Run the following command to obtain and modify the Helm Chart of the GitLab runner:

```
$ git clone https://code.aliyun.com/CodePipeline/gitlab-runner.git
```

Modify the `values.yaml` file as follows:

```
## GitLab Runner Image
##
image: gitlab/gitlab-runner:alpine-v11.4.0

## Specify an imagePullPolicy
##
imagePullPolicy: IfNotPresent

## Default container image to use for initcontainer
init:
  image: busybox
  tag: latest

## The GitLab Server URL (with protocol) that want
## to register the runner against
gitlabUrl: http://xx.xx.xx.xx/

## The Registration Token for adding new Runners to
## the GitLab Server. This must
## be retrieved from your GitLab Instance.
##
runnerRegistrationToken: "AMvEWrBTBu-d8czEYyfy"
## Unregister all runners before termination
##
unregisterRunners: true

## Configure the maximum number of concurrent jobs
##
concurrent: 10

## Defines in seconds how often to check GitLab for
## a new builds
```

```

##
checkInter val : 30

## For RBAC support :
##
rbac :
  create : true
  clusterWideAccess : false

## Configure integrated Prometheus metrics exporter
##
metrics :
  enabled : true

## Configuration for the Pods that the runner
  launches for each new job
##
runners :
  ## Default container image to use for builds when
  none is specified
  ##
  image : ubuntu : 16 . 04

  ## Specify the tags associated with the runner .
  Comma - separated list of tags .
  ##
  tags : " k8s - runner "

  ## Run all containers with the privileged flag
  enabled
  ## This will allow the docker : dind image to run
  if you need to run Docker
  ## commands . Please read the docs before turning
  this on :
  ##
  privileged : true

  ## Namespace to run Kubernetes jobs in ( defaults to
  the same namespace of this release )
  ##
  namespace : gitlab

  cachePath : "/ opt / cache "

  cache : {}
  builds : {}
  services : {}
  helpers : {}

resources : {}

```

3. Run the following command to install the GitLab runner.

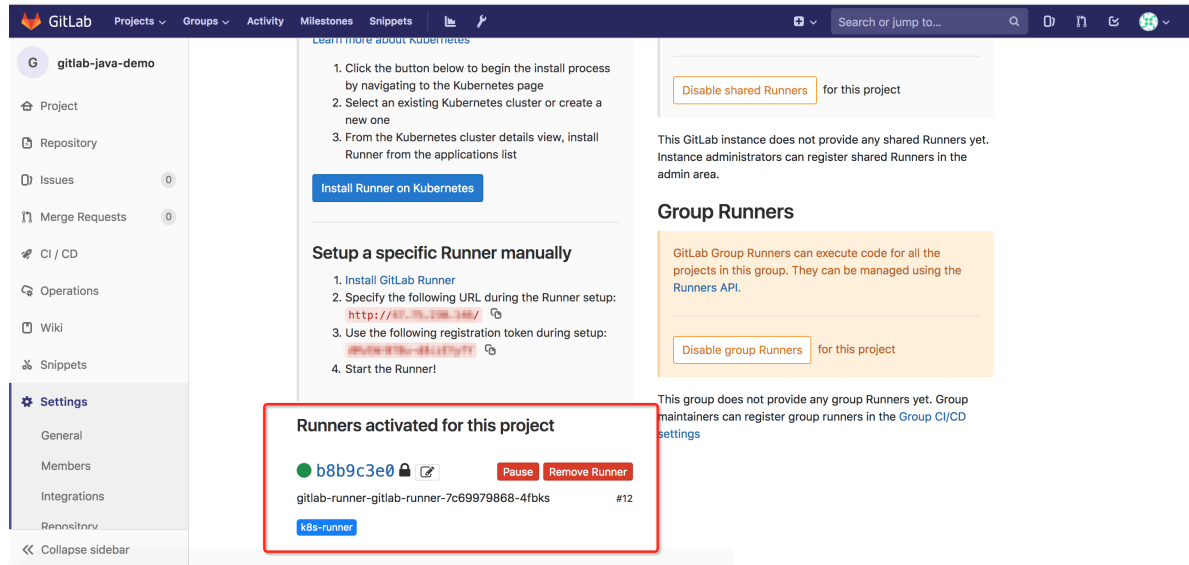
```

$ helm package .
Successful ly packaged chart and saved it to : / root
/ gitlab / gitlab - runner / gitlab - runner - 0 . 1 . 37 . tgz

```

```
$ helm install -- namespace gitlab -- name gitlab - runner
*. tgz
```

Check whether the related deployment/pod has been started. If the related deployment/pod has been started, the GitLab runner that has been registered in GitLab is displayed.



Set the GitLab runner cache

GitLab runners have a limited cache capacity. Therefore, you need to mount a volume to your GitLab runner so that the volume functions as the GitLab runner cache. In this example, the `/opt/cache` directory is used as the GitLab runner cache by default. You can modify the `runners.cachePath` field of the `values.yaml` file to change the default cache directory.

For example, to create a maven cache, add the `MAVEN_OPTS` variable to `variables` and specify a local cache directory as follows:

```
variables :
  KUBECONFIG : / etc / deploy / config
  MAVEN_OPTS : "- Dmaven.repo.local=/opt/cache/.m2/repository "
```

To mount a new volume, modify the following fields in the `templates/configmap.yaml` file:

```
cat >>/home/gitlab-runner/.gitlab-runner/config.toml
<< EOF

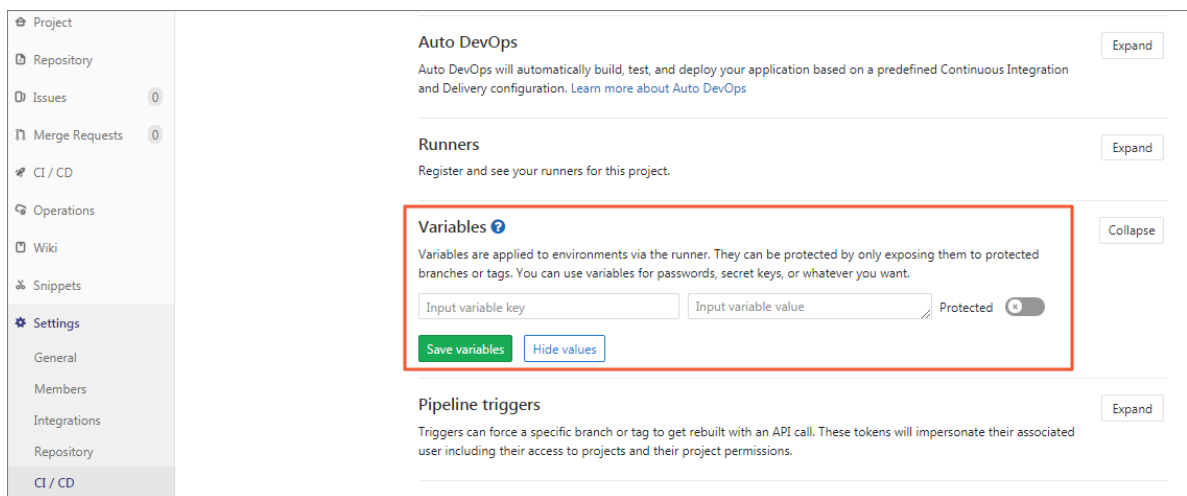
[[ runners.kubernetes.volumes.pvc ]]
  name = "gitlab-runner-cache"
  mount_path = "{{ .Values.runners.cachePath }}"
```

EOF

This means that you need to modify the settings of `config.toml` during the period between the time at which the GitLab runner was registered and the time at which the runner has not yet been started.

Set the global variables

1. In the top navigation bar, choose **Projects > Your projects**.
2. On the **Your projects** tab page, select the target project.
3. In the left-side navigation pane, choose **Settings > CI / CD**.
4. Click **Expand** on the right of **Runners** to add available to the GitLab runner.



In this example, add the following three variables:

- **REGISTRY_USERNAME**: indicates a registry username.
- **REGISTRY_PASSWORD**: indicates a registry password.
- **kube_config**: indicates a string of KubeConfig code characters.

Run the following command to generate a string of KubeConfig code characters:

```
echo $( cat ~/. kube / config | base64 ) | tr - d " "
```

Edit the `.gitlab-ci.yml` file

Edit the `.gitlab-ci.yml` file to compile and build the Java demo source code project, push the application image, and deploy the application. For more information, see the `.gitlab-ci.yml.example` of the `gitlabci-java-demo` source code project. This section first shows an example of an entire YAML file, and then describes key sections of the file in more detail.

The entire `.gitlab-ci.yml` file is as follows:

```
image : docker : stable
stages :
  - package
  - docker_bui ld
  - deploy_k8s
variables :
  KUBECONFIG : / etc / deploy / config
mvn_build_ job :
  image : registry . cn - beijing . aliyuncs . com / codepipeli
ne / public - blueocean - codepipeli ne - slave - java : 0 . 1 -
63b99a20
  stage : package
  tags :
    - k8s - test
  script :
    - mvn package - B - DskipTests
    - cp target / demo . war / opt / cache
docker_bui ld_ job :
  image : registry . cn - beijing . aliyuncs . com / codepipeli
ne / public - blueocean - codepipeli ne - slave - java : 0 . 1 -
63b99a20
  stage : docker_bui ld
  services :
    - docker : dind
  variables :
    DOCKER_DRI VER : overlay
    DOCKER_HOS T : tcp :// localhost : 2375
  tags :
    - k8s - test
  script :
    - docker login - u $ REGISTRY_U SERNAME - p $ REGISTRY_P
ASSWORD registry . cn - beijing . aliyuncs . com
    - mkdir target
    - cp / opt / cache / demo . war target / demo . war
    - docker build - t registry . cn - beijing . aliyuncs . com
/ gitlab - demo / java - demo :$ CI_PIPELIN E_ID .
    - docker push registry . cn - beijing . aliyuncs . com /
gitlab - demo / java - demo :$ CI_PIPELIN E_ID
deploy_k8s_ job :
  image : registry . cn - beijing . aliyuncs . com / codepipeli
ne / public - blueocean - codepipeli ne - slave - java : 0 . 1 -
63b99a20
  stage : deploy_k8s
  tags :
    - k8s - test
  script :
    - mkdir - p / etc / deploy
    - echo $ kube_conf_i g | base64 - d > $ KUBECONFIG
    - sed - i " s / IMAGE_TAG /$ CI_PIPELIN E_ID / g " deployment
. yaml
    - cat deployment . yaml
    - kubectl apply - f deployment . yaml
```

The `.gitlab-ci.yml` file defines a pipeline that is executed in three stages.

```
image : docker : stable # If no image is specified for
each step of the pipeline , the docker : stable image
is used by default .
stages :
  - package # package the soucre code
```

```
- docker_build # build , package , and push the
applicatio n image
- deploy_k8s # deploy the applicatio n
variables :
  KUBECONFIG : / etc / deploy / config # define the global
variable , namely , KUBECONFIG
```

- **Package the maven source code**

```
mvn_build_job : # the job name
  image : registry . cn - beijing . aliyuncs . com / codepipeli
ne / public - blueocean - codepipeli ne - slave - java : 0 . 1 -
63b99a20 # the image used in this stage
  stage : package # the stage name
  tags : # GitLab Runner tag
    - k8s - test
  script :
    - mvn package - B - DskipTests # the script to
execute the build
    - cp target / demo . war / opt / cache # save the
build to the cache
```

- **Build, package, and push the application image**

```
docker_build_job : # the job name
  image : registry . cn - beijing . aliyuncs . com / codepipeli
ne / public - blueocean - codepipeli ne - slave - java : 0 . 1 -
63b99a20 # the image used in this stage
  stage : docker_build # the stage name
  services : # Use the docker : dind
service . You must set the runners . privileged in
GitLab Runner Helm Chart to true .
    - docker : dind
  variables :
    DOCKER_DRIVER : overlay
    DOCKER_HOST : tcp :// localhost : 2375 # Connect the
Docker Daemon .
  tags : # GitLab Runner tag
    - k8s - test
  script :
    - docker login - u REGISTRY_USERNAME - p $ REGISTRY_P
ASSWORD registry . cn - beijing . aliyuncs . com # log on
to the registry
    - mkdir target
    - cp / opt / cache / demo . war target / demo . war
    - docker build - t registry . cn - beijing . aliyuncs .
com / gitlab - demo / java - demo :$ CI_PIPELINE_ID . #
Package the Docker image . The used tag is the
pipeline ID .
    - docker push registry . cn - beijing . aliyuncs . com /
gitlab - demo / java - demo :$ CI_PIPELINE_ID # Push the
Docker image .
```

- **Deploy the application**

```
deploy_k8s_job : # the job name
  image : registry . cn - beijing . aliyuncs . com / codepipeli
ne / public - blueocean - codepipeli ne - slave - java : 0 . 1 -
63b99a20 # the image used in this stage
  stage : deploy_k8s # the stage name
  tags : # GitLab Runner tag
    - k8s - test
```



```
script :
- mkdir -p /etc / deploy
- echo $ kube_config | base64 -d > $ KUBECONFIG #
Set the config file that is used to connect the
Kubernetes cluster .
- sed -i "s / IMAGE_TAG / $ CI_PIPELINE_ID / g "
deployment . yaml # Dynamically replace the image tag
of the deployment file .
- kubectl apply -f deployment . yaml
```

Activate the pipeline

After you submit the `.gitlab-ci.yml` file, the gitlab-java-demo project automatically detects this file and activate the pipeline.

Access the application service

If no namespace is specified in the deployment file, the application is deployed in the GitLab namespace by default.

```
$ kubectl -n gitlab get svc
NAME          TYPE          CLUSTER-IP      EXTERNAL-IP
PORT (S)      AGE
java-demo     LoadBalancer  172.19.9.252     xx.xx.xx.xx
xx.xx.xx      80 : 32349 / TCP 1m
```

Visit `xx.xx.xx.xx/demo` in your browser to check the result.

For more information, see [Container Service](#) and [GitLab CI](#).

9.3 Deploy Jenkins in a serverless Kubernetes cluster and perform an application pipeline build

This topic describes how to deploy Jenkins, a continuous integration environment, in an Alibaba Cloud serverless Kubernetes cluster, and how to perform an application pipeline build. The example in this topic details the pipeline build, including how to compile the source code of the application, build and push the application image, and deploy the application.

Prerequisites

You have created a serverless Kubernetes cluster. For more information, see [Create a serverless Kubernetes cluster](#).

Deploy Jenkins

1. Run the following command to download the Jenkins deployment file:

```
$ git clone https://github.com/AlibabaCloud/aliyun-container-service-jenkins-on-serverless.git
$ cd jenkins-on-serverless
```

2. Persist the `jenkins_home` directory.

Serverless Kubernetes clusters do not support cloud disks. To persist the `jenkins_home` directory, create the `nfs` volume, then modify the `serverless-k8s-jenkins-deploy.yaml` file to add the following field annotations and set `nfs` parameters:

```
# volumeMounts:
# - mountPath: /var/jenkins_home
#   name: jenkins-home
....
# volumes:
```

```
# - name : jenkins - home
#   nfs :
#     path : /
```

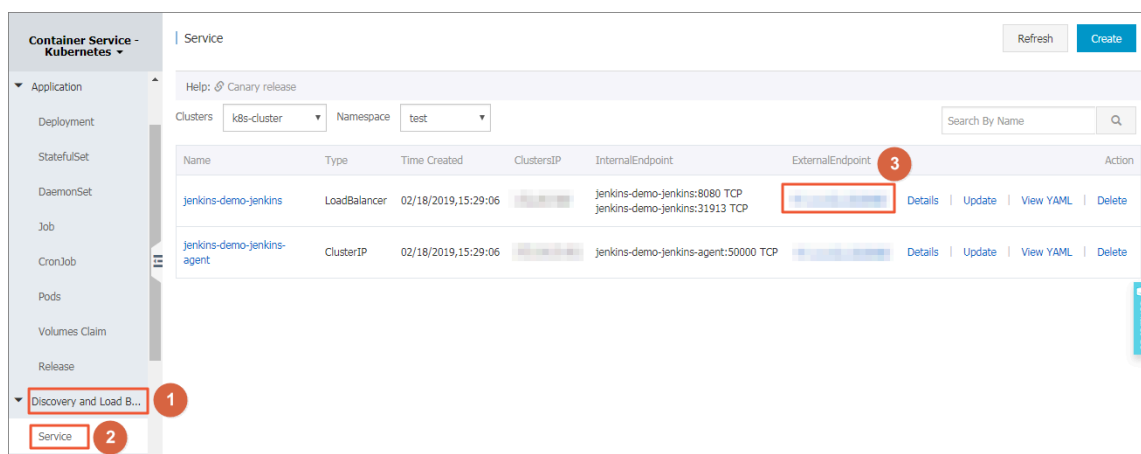
```
# server :
```

3. Run the following command to deploy Jenkins:

```
$ kubectl apply -f serverless - k8s - jenkins - deploy . yml
```

4. Log on to Jenkins.

- a. Log on to the [Container Service console](#).
- b. In the left-side navigation pane, choose **Discovery and Load Balancing > Service**.
- c. Click the external endpoint of the Jenkins service to log on to Jenkins.



- d. On the Jenkins logon page, enter the user name and the password. The default user name and password are admin. We recommend that you modify them after you log on to Jenkins.



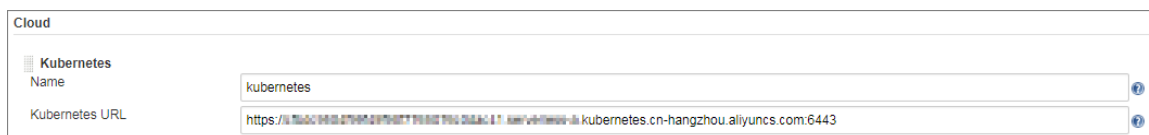
Welcome to Jenkins!

Sign in

☐ Keep me signed in

Create a cluster certificate and an image repository certificate, and build and deploy an application

1. Set Kubernetes Cloud parameters to dynamically create a slave pod.
 - a. In the left-side navigation pane, click Manage Jenkins.
 - b. Click Configure System.
 - c. In the Cloud area, enter the API server URL in KubeConfig as the Kubernetes URL.




Cloud

Kubernetes

Name: kubernetes

Kubernetes URL: https://kubernetes.cn-hangzhou.aliyuncs.com:6443

- d. Click Add on the right of Credentials.



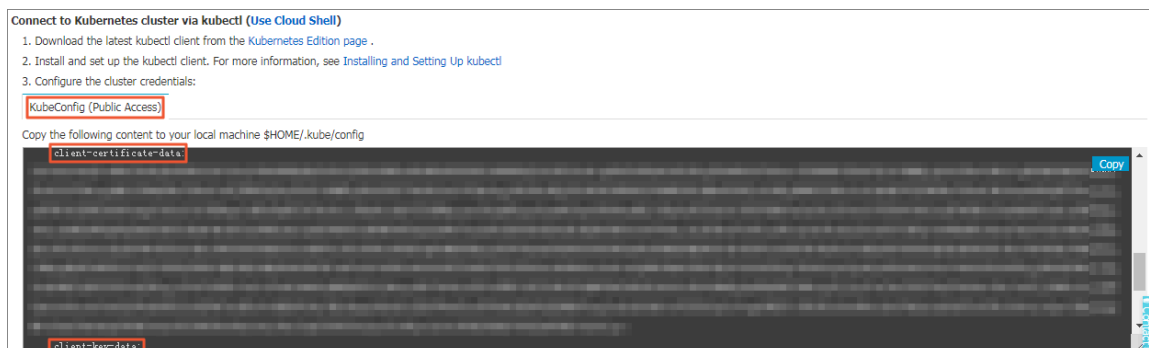
Disable https certificate check: ☒

Kubernetes Namespace: test

Credentials: - none - Add

Test Connection

Before adding a credential, you must obtain KubeConfig on the Basic Information tab page of the target Kubernetes cluster.



Connect to Kubernetes cluster via kubectl (Use Cloud Shell)

- Download the latest kubectl client from the [Kubernetes Edition](#) page.
- Install and set up the kubectl client. For more information, see [Installing and Setting Up kubectl](#).
- Configure the cluster credentials:

KubeConfig (Public Access)

Copy the following content to your local machine \$HOME/.kube/config

```
client-certificate-data:
client-key-data:
```

In the displayed dialog box, set the following parameters:

- **Kind:** Select Docker Host Certificate Authentication.
- **Client Key:** Paste the copied client-key-data content in KubeConfig.
- **Client Certificate:** Paste the copied client-certificate-data content in KubeConfig.
- **ID:** Enter the certificate ID. In this example, k8sCertAuth is entered.
- **Description:** Enter description content.

e. Click Add.

f. Test connectivity.

Select the added credential in the preceding step from the Credentials drop-down list, and then click Test Connection.

g. Enter the external endpoint of the jenkins service as Jenkins URL, and enter the external endpoint of the jenkins-agent service as Jenkins tunnel.

Jenkins URL	<input type="text" value="http://jenkins-test-jenkins 8080"/>	
Jenkins tunnel	<input type="text" value="10.1.10.1:50000"/>	
Connection Timeout	<input type="text" value="0"/>	
Read Timeout	<input type="text" value="0"/>	

h. Click Save.

- 2. To set image repository permission, use kubectl to create jenkins-docker-cfg secret in the target serverless Kubernetes cluster.**

In this example, the Beijing image repository provided by Alibaba Cloud is used.

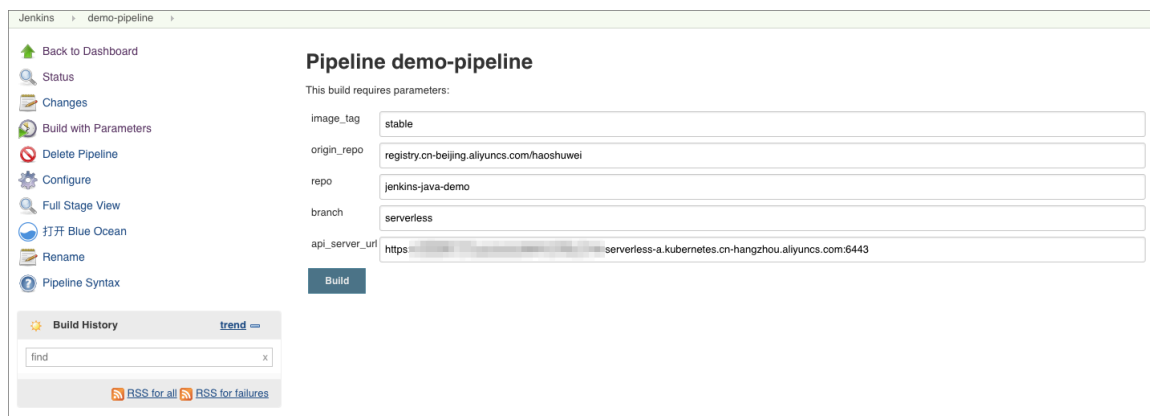
```
$ docker login -u xxx -p xxx registry.cn-beijing.aliyuncs.com
Login Succeeded
```



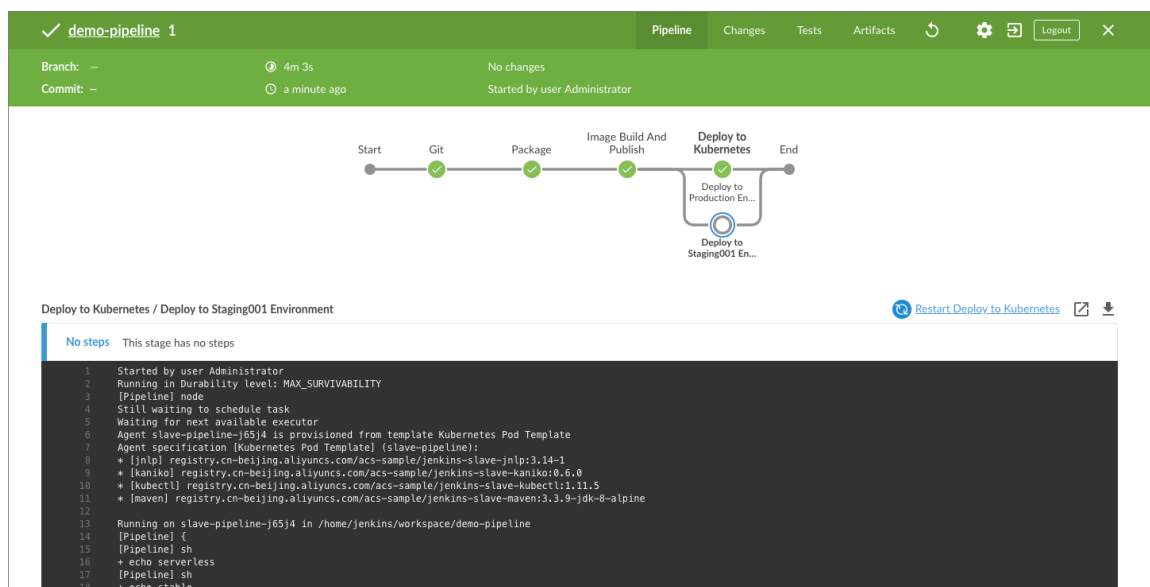
```
$ kubectl create secret generic jenkins - docker - cfg --
from - file =/ root /. docker / config . json
```

3. Create demo-pipeline and access the application service.

- On the Jenkins home page, click demo-pipeline.
- In the left-side navigation pane, click Build with Parameters.
- Modify the parameters according to your image repository information. In this example, the source code repository branch is serverless, and the image is registry.cn-beijing.aliyuncs.com/haoshuwei:stable.



- Click Build.
- Click Build History to check the result. The following figure indicates a success.



- Log on to the [Container Service console](#) to view the IP addresses of the services provided by the application.

The source code repository used in this topic can be found at <https://github.com/AliyunContainerService/jenkins-demo>.

9.4 Use Bamboo to deploy a remote agent and run a build plan

This topic describes how to use Bamboo to deploy a remote agent in a Kubernetes cluster implemented with Alibaba Cloud Container Service for Kubernetes (ACK) and how to use the agent to run a build plan for an application. In this topic, an example application compiled in Java is created and deployed to a Kubernetes cluster.

Prerequisites

- A Kubernetes cluster is created by using ACK. For more information, see [#unique_11](#).
- A Bamboo server is created.

Source code of the application to be deployed

The source code of the application created in this topic can be obtained with the following address:

```
https://github.com/AlibabaCloudContainerService/jenkins-demo.git
```

After you access the corresponding GitHub page, you can find the source code in the `bamboo` branch.

Deploy a remote agent in a Kubernetes cluster

1. Create a kaniko-docker-cfg secret.



Note:

This secret is used to set the permissions for accessing the target image repository required by building tasks in the remote agent and using kaniko to push a container image.

- a. Log on to your Linux server by using the root account to run the following command to create a `/root/.docker/config.json` file:

```
docker login registry.cn-hangzhou.aliyuncs.com
```

- b. Use [Cloud Shell](#) to connect to the target Kubernetes cluster, and then run the following command to create a kaniko-docker-cfg secret:

```
kubectl -n bamboo create secret generic kaniko-docker-cfg --from-file=/root/.docker/config.json
```

2. Create a Bamboo agent in the target Kubernetes cluster.

`ServiceAccount` and `ClusterRoleBinding` are created to set the permissions required for `kubectl` to deploy an application to the target Kubernetes cluster.

- a. Create a file `bamboo-agent.yaml` and copy the following code to the file:

```
apiVersion: v1
kind: ServiceAccount
metadata:
  namespace: bamboo
  name: bamboo
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: bamboo-cluster-admin
subjects:
- kind: ServiceAccount
  name: bamboo
  namespace: bamboo
roleRef:
  kind: ClusterRole
  name: cluster-admin
  apiGroup: rbac.authorization.k8s.io
---
apiVersion: apps/v1beta2
kind: Deployment
metadata:
  name: bamboo-agent
spec:
  replicas: 1
  selector:
    matchLabels:
      app: bamboo-agent
  template:
    metadata:
      labels:
```

```

    app : bamboo - agent
spec :
  serviceAccountName : bamboo
  containers :
  - name : bamboo - agent
    env :
    - name : BAMBOO_SERVER_URL
      value : http://xx.xx.xx.xx:8085
    image : registry.cn-hangzhou.aliyuncs.com/haoshuwei/docker-bamboo-agent:v1
    imagePullPolicy : Always
    volumeMounts :
    - mountPath : /root/.docker/
      name : kaniko-docker-cfg
  volumes :
  - name : kaniko-docker-cfg
    secret :
      secretName : kaniko-docker-cfg

```

b. Run the `kubectl -n bamboo apply -f bamboo-agent.yaml` command to create the Bamboo agent.

c. Run the following command to view logs of the agent:

```
kubectl -n bamboo logs -f <bamboo agent pod name>
```



Note:

You must replace `<bamboo agent pod name>` with the file name that you use.

d. Log on to the Bamboo server you created. Then, in the upper-right corner, click



, and select Agent to view the deployed agent.

Bamboo administration

Agents

Agents summary

An agent is a service that executes Bamboo builds and deployments. You can use this page to view, add and delete agents. You can also use this matrix to determine which agents can execute which build plans.

Local agents

Local agents run on the Bamboo server.

Select: All, None, Idle, Disabled Action: [Delete](#) [Disable](#) [Enable](#)

Agent	Status	Operations
<input type="checkbox"/> Default Agent	Idle	View Edit

Remote agents

Remote agents run on computers other than the Bamboo server.

[Online remote agents](#) [Offline remote agents](#)

There is currently 1 remote agent online.

Select: All, None, Idle, Disabled Action: [Delete](#) [Disable](#) [Enable](#)

Agent	Status	Operations
<input type="checkbox"/> bamboo-agent-755cf799bc-2hpm8	Idle	View Edit

Jun 5, 2019 10:25:24 AM A remote agent is loading on bamboo-agent-77b96dd97c-vcc9d (114.55.243.78).
 Jun 5, 2019 10:25:29 AM A remote agent is loading on bamboo-agent-77b96dd97c-vcc9d (114.55.243.78).
 Jun 5, 2019 10:25:29 AM Remote agent "bamboo-agent-77b96dd97c-vcc9d" has registered.
 Jun 5, 2019 11:02:06 AM A remote agent is loading on bamboo-agent-77b96dd97c-cctmz (114.55.243.78).
 Jun 5, 2019 11:02:11 AM A remote agent is loading on bamboo-agent-77b96dd97c-cctmz (114.55.243.78).
 Jun 5, 2019 11:02:53 AM Remote agent "bamboo-agent-77b96dd97c-cctmz" has registered.
 Jun 5, 2019 11:08:33 AM A remote agent is loading on bamboo-agent-7b469cb67-gmtbk (114.55.243.78).
 Jun 5, 2019 11:08:38 AM A remote agent is loading on bamboo-agent-7b469cb67-gmtbk (114.55.243.78).
 Jun 5, 2019 11:09:28 AM Remote agent "bamboo-agent-7b469cb67-gmtbk" has registered.
 Jun 5, 2019 11:54:06 AM A remote agent is loading on bamboo-agent-755cf799bc-rdxhl (114.55.243.78).

Powered by a free Atlassian Bamboo evaluation license. Please consider purchasing it today.

Configure a build plan

1. Create a build plan.

- a. Log on to the Bamboo server you created, and choose **Create > Create plan**.
- b. Select **bamboo - ack - demo** from the **Project** drop-down list, set **Plan name**, **Plan key**, and **Plan description**, select **java - demo** from the **Repository host** drop-down list, and then click **Configure plan**.

Create plan

Configure plan Configure job

Configure plan [How to create a build plan](#)

Your build plan defines everything about your build process. Each plan has a Default job when it is created. More advanced configuration options, including those for apps, and the ability to add more jobs will be available to you after creating this plan.

Project and build plan name

Project: **bamboo-ack-demo** (dropdown)
The project the new plan will be created in.

Plan name*: **bamboo-ack-demo** (text box)

Plan key*: **BAM11** (text box)
For example WEB (for a plan named Website)

Plan description: (text box)

Plan access: ☒ Allow all users to view this plan. Applies to new project as well.

Link repository to new build plan

Repository host*: ☒ Previously linked repository
java-demo (dropdown)

☐ Link new repository
☐ None

Configure plan Cancel

2. Configure a job that contains four required tasks for the build plan.

- a. Confirm and save the setting of the source code repository.



Note:

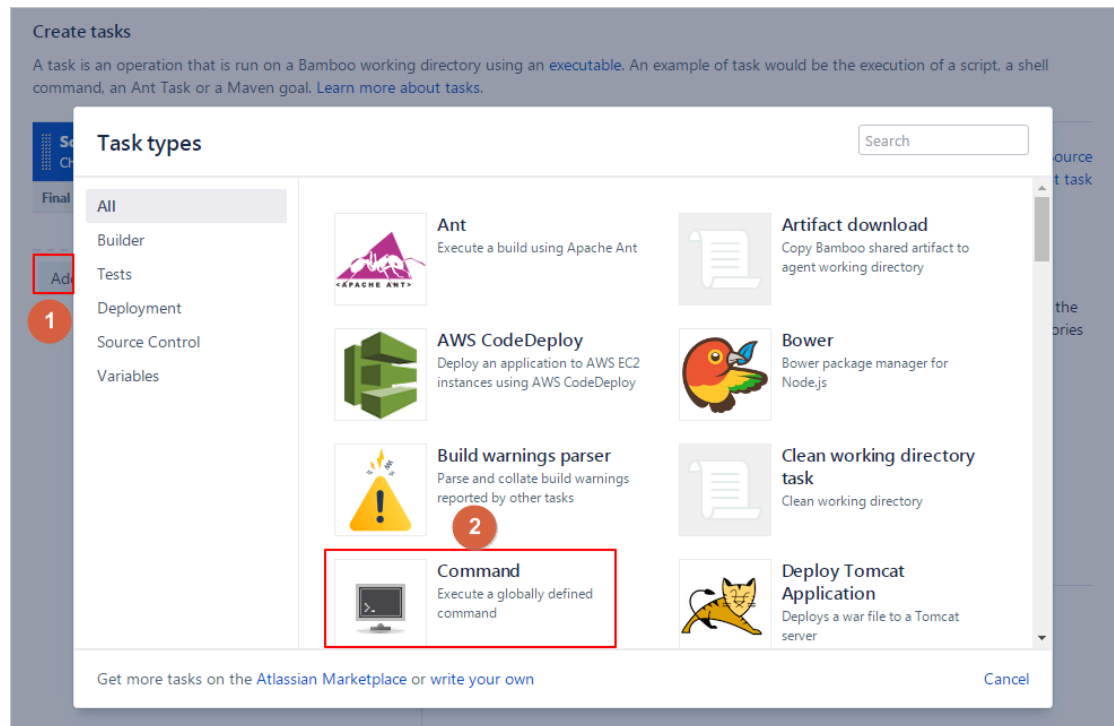
In the preceding step where you create the build plan, your setting for Repository host specifies the source code repository. You can retain or modify this setting.

A. In the Create tasks area, click Source Code Checkout.

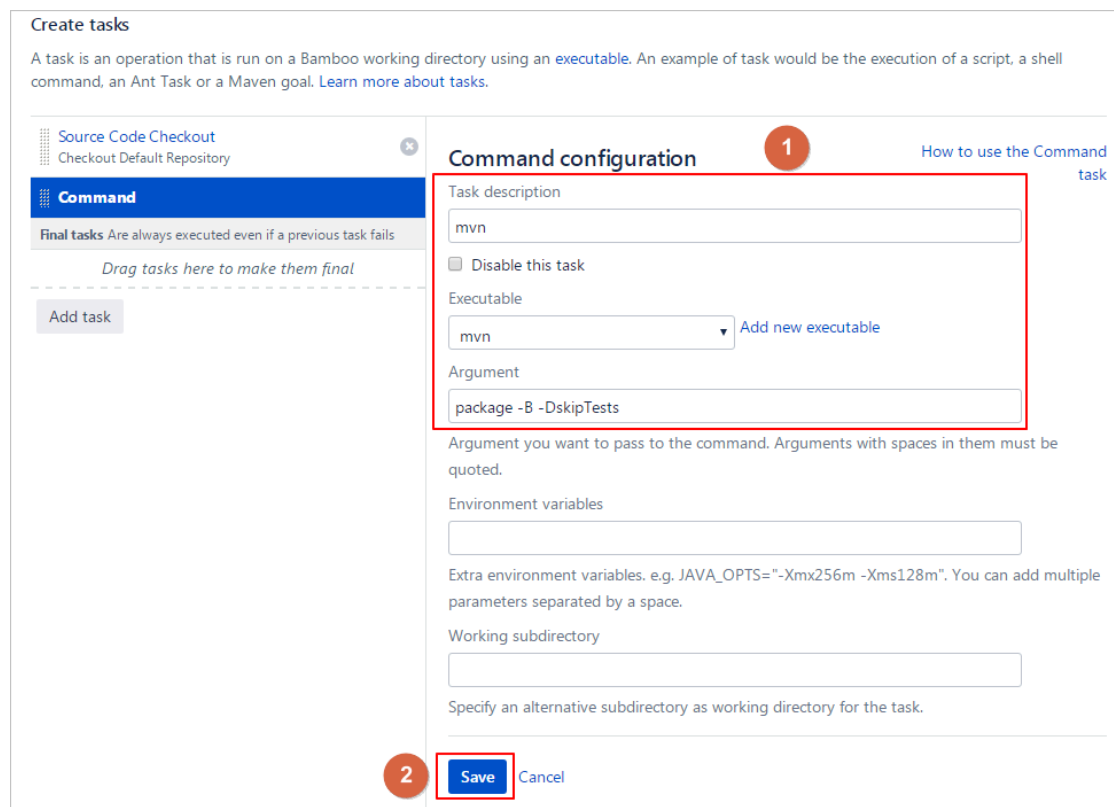
The screenshot shows the 'Create tasks' interface. On the left, under 'Source Code Checkout', there is a task named 'Checkout Default Repository'. Below it, a dashed line indicates where to drag tasks to make them final, with an 'Add task' button. On the right, the 'Source Code Checkout configuration' page is displayed. It includes a 'Task description' field with the text 'Checkout Default Repository'. There is a checkbox for 'Disable this task'. A paragraph explains that you can check out one or more repositories, choosing between the Plan's Default Repository or a Specific Repository. Below this is a 'Repository*' dropdown menu currently set to 'java-demo'. A note states 'Default always points to Plans default repository.' There is a 'Checkout Directory' text input field. Below that, an optional note says '(Optional) Specify an alternative sub-directory to which the code will be checked out.' There is a checkbox for 'Force Clean Build' with a description: 'Removes the source directory and checks it out again prior to each build. This may significantly increase build times.' At the bottom of the configuration area is an 'Add repository' link. At the very bottom of the page are 'Save' and 'Cancel' buttons.

- B. On the Source Code Checkout configuration page, select a new repository from the Repository drop-down list, and then click Save.
- b. Add the command type of mvn to use the mvn tool.

A. In the Create tasks area, click Add task. Then, on the displayed Task types page, click Command.

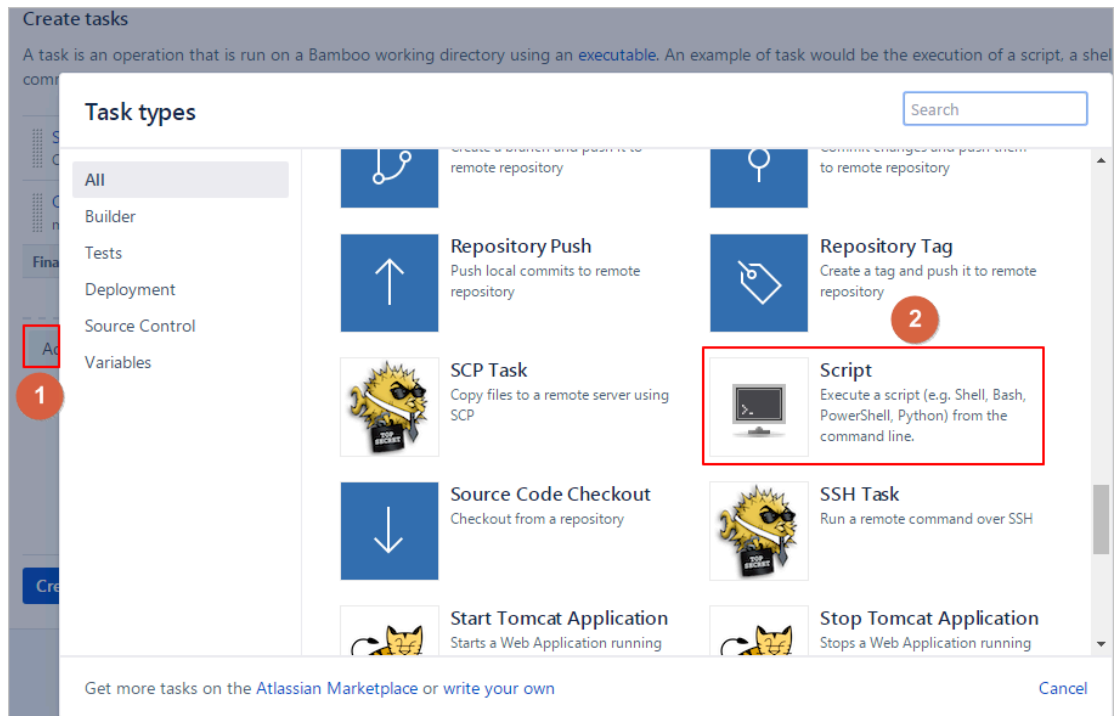


B. On the Command configuration page, set Task description, Executable, and Argument, and then click Save.



c. Use kaniko to package and push the required container image to the target image repository.

A. In the Create tasks area, click Add task. Then, on the displayed Task types page, click Script.



B. On the Script configuration page, set Task description, and Script location, then click Save.



Note:

For the remaining parameters of the script, you can retain the default settings.

Script configuration [How to use the Script task](#)

Task description
kaniko

☐ Disable this task

Interpreter
Shell

An interpreter is chosen based on the shebang line of your script.

Script location
Inline

Script body*

```
1 kaniko -f `pwd`/Dockerfile -c `pwd` --destination=registry.cn-hangzhou.aliyuncs.com/haoshuwei/bamboo-java-demo:latest
```

Argument

Environment variables

Working subdirectory

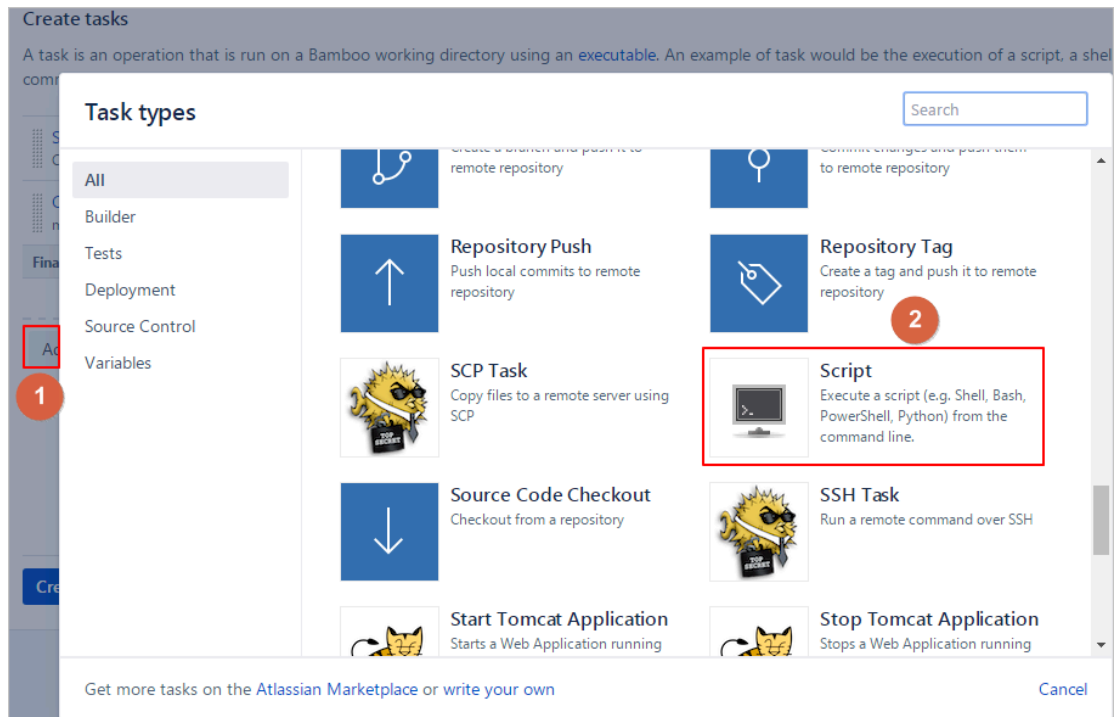
[Save](#) [Cancel](#)

In this example, the Script location is set as follows:

```
kaniko -f `pwd`/Dockerfile -c `pwd` --destination=registry.cn-hangzhou.aliyuncs.com/haoshuwei/bamboo-java-demo:latest
```

d. Use `kubectl` to deploy the application on the target Kubernetes cluster.

A. In the Create tasks area, click **Add task**. Then, on the displayed Task types page, click **Script**.



B. On the Script configuration page, set Task description and Script location, then click Save.



Note:

For the remaining parameters of the script, you can retain the default settings.

Script configuration

Task description
kube deploy

☐ Disable this task

Interpreter
Shell

An interpreter is chosen based on the shebang line of your script.

Script location
Inline

Script body*

```
1 sed -i 's#IMAGE_URL#registry.cn-hangzhou.aliyuncs.com/haoshuwei/b
2 kubectl apply -f ./
```

Argument

Environment variables

Working subdirectory

Save Cancel

In this example, the Script location is set as follows:

```
sed -i 's # IMAGE_URL # registry . cn - hangzhou .
aliyuncs . com / haoshuwei / bamboo - java - demo : latest
#' ./*. yaml
```

```
kubectl apply -f ./
```

3. Run the build plan.

a. On the Configure Job page, click Create.

Configure Job

Each plan has a default job when it is created. Here, you can configure Tasks for this plan's default job. You can add more jobs to this plan once the plan has been created.

Isolate build

Builds are normally run in the agent's native operating system. If you want to run your build in an isolated and controlled environment, you can do it with Docker.

Run this job in ☒ Agent environment ☐ Docker container

Create tasks

A task is an operation that is run on a Bamboo working directory using an [executable](#). An example of task would be the execution of a script, a shell command, an Ant Task or a Maven goal. [Learn more about tasks.](#)

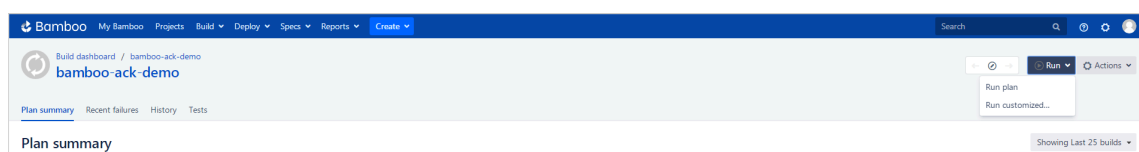
Source Code Checkout Checkout Default Repository	<input type="checkbox"/>
Command mvn	<input type="checkbox"/>
Script kaniko	<input type="checkbox"/>
Script kube deploy	<input type="checkbox"/>
Final tasks Are always executed even if a previous task fails	
Drag tasks here to make them final	
<input type="button" value="Add task"/>	

☒ Task created successfully.

No task selected

Select a task from the list on the left to configure it.

b. In the upper-right corner of the page, choose Run > Run plan.



You can click the Logs tab to view the logs.

Build dashboard / bamboo-ack-demo / bamboo-ack-demo

Build #15

bamboo-ack-demo

#15 was successful - Manual run by bamboo

Summary Tests Commits Artifacts **Logs** Metadata

Logs

The following logs have been generated by the jobs in this plan. [Expand all](#) [Collapse all](#)

Job	Logs
Default Job Default Stage	<p>05-Jun-2019 11:58:49 INFO [0008] Skipping paths under /proc, as it is a whitelisted directory</p> <p>05-Jun-2019 11:58:49 INFO [0008] Skipping paths under /root/.docker, as it is a whitelisted directory</p> <p>05-Jun-2019 11:58:49 INFO [0008] Skipping paths under /run/secrets/kubernetes.io/serviceaccount, as it is a whitelisted directory</p> <p>05-Jun-2019 11:58:49 INFO [0008] Skipping paths under /sys, as it is a whitelisted directory</p> <p>05-Jun-2019 11:58:49 INFO [0008] Skipping paths under /var/run, as it is a whitelisted directory</p> <p>05-Jun-2019 11:58:55 INFO [0014] Using files from context: [/root/bamboo-agent-home/xml-data/build-dir/BAM-BAM-JOB1/target/demo.war]</p> <p>05-Jun-2019 11:58:55 INFO [0014] ADD target/demo.war /usr/local/tomcat/webapps/demo.war</p> <p>05-Jun-2019 11:58:55 INFO [0014] Taking snapshot of files...</p> <p>05-Jun-2019 11:58:55 2019/06/05 11:58:55 existing blob: sha256:6bc6f18ca5a114be8c090bf0e3613fbc33393b7ed210e99368028cf1cbb7a7e</p> <p>05-Jun-2019 11:58:55 2019/06/05 11:58:55 existing blob: sha256:39aa78db61d45511d3498958869b9000e9b86842b7aab3c94ca988f72da378f4</p> <p>05-Jun-2019 11:58:55 2019/06/05 11:58:55 existing blob: sha256:3b7ca19181b24b87e24423c01b490633bc1e47d2fcd1987bf2e37949d6789b5</p> <p>05-Jun-2019 11:58:55 2019/06/05 11:58:55 existing blob: sha256:c5e155d5a1d138a7f8a3e24cee009e1349bf13f90ecc9a1478e558fde53c14</p> <p>05-Jun-2019 11:58:55 2019/06/05 11:58:55 existing blob: sha256:22108000ae9675aad2913aacbaefac1ce8b7084f9765a6c0813486002c5c69</p> <p>05-Jun-2019 11:58:55 2019/06/05 11:58:55 existing blob: sha256:4250b3117dca5e14edc32ebf1366cd54ed4da91f17610b76c504a86917f7fb95</p> <p>05-Jun-2019 11:58:55 2019/06/05 11:58:55 existing blob: sha256:5aff210feafa4e5af4354699a16353389c28e2675782148a201cb373b7b5cbd</p> <p>05-Jun-2019 11:58:55 2019/06/05 11:58:55 existing blob: sha256:2aa7bd7991994877c85197911e30e5a0f6eccc76c0684b9d5eaf095aba2671a</p> <p>05-Jun-2019 11:58:55 2019/06/05 11:58:55 existing blob: sha256:47ce9f19f5673fe0e07492108520a49400c647485f1759a08c5d3e79b37cfcf9</p> <p>05-Jun-2019 11:58:55 2019/06/05 11:58:55 existing blob: sha256:17095069ba1e31add36da6b2e69f9a502914ac903a1c7f58659a61286b9</p> <p>05-Jun-2019 11:58:55 2019/06/05 11:58:55 existing blob: sha256:acd9a83f588d8e2b2e3f146ed7adb7b6f8e373516efc24c78c8208f8812ad14</p> <p>05-Jun-2019 11:58:55 2019/06/05 11:58:55 existing blob: sha256:83d1e68279c414eb4d9e34a511e795f826ac6dd32bc65a92df48f7b533b2e5</p> <p>05-Jun-2019 11:58:56 2019/06/05 11:58:56 pushed blob sha256:d72440d042979e5917b13aef6d1e017300d8ea91d040d0bcc63fcaec8a3c5f</p> <p>05-Jun-2019 11:58:56 2019/06/05 11:58:56 pushed blob sha256:1e2873a9c63d5787dde9893f85d9eae763d24bb3c562fab17955a2915a58779</p> <p>05-Jun-2019 11:58:56 2019/06/05 11:58:56 registry.cn-hangzhou.aliyuncs.com/haoshuwei/bamboo-java-demo:latest: digest: sha256:e11d0935307d028a084b7ba61092cc1b992b8bf5eb29aa70b6054e4fc5a5e5c</p> <p>05-Jun-2019 11:58:57 deployment.extensions/jenkins-java-demo created</p> <p>05-Jun-2019 11:58:57 service/jenkins-java-demo created</p>

Powered by a free Atlassian Bamboo evaluation license. Please consider purchasing it today.

4. Access the deployed application.

- Run the `kubectl -n bamboo get svc` command to view the Internet IP address of the application.

```
[ root @ iZbp12i73k oztp1cz75s kaZ bamboo ]# kubectl -n
bamboo get svc
NAME                                TYPE                                CLUSTER - IP
EXTERNAL - IP                       PORT ( S )                         AGE
jenkins - java - demo               LoadBalancer                      xx . xx . xx . xx
xx . xx . xx . xx                  80 : 32668 / TCP                   39m
```

- In your browser, enter `http :// EXTERNAL - IP` to access the application.

← → 🔍 不安全 | [redacted]

Home Documentation Configuration Examples Wiki Mailing Lists Find Help

Apache Tomcat/8.5.41

If you're seeing this, you've successfully installed Tomcat. Congratulations!

Recommended Reading:

- [Security Considerations HOW-TO](#)
- [Manager Application HOW-TO](#)
- [Clustering/Session Replication HOW-TO](#)

[Server Status](#) [Manager App](#) [Host Manager](#)

Developer Quick Start

- [Tomcat Setup](#)
- [Realms & AAA](#)
- [Examples](#)
- [Servlet Specifications](#)
- [First Web Application](#)
- [JDBC DataSources](#)
- [Tomcat Versions](#)

Managing Tomcat

For security, access to the [manager webapp](#) is restricted. Users are defined in:

```
$CATALINA_HOME/conf/tomcat-users.xml
```

In Tomcat 8.5 access to the manager application is split between different users. [Read more...](#)

[Release Notes](#) [Changelog](#)

Documentation

[Tomcat 8.5 Documentation](#)

[Tomcat 8.5 Configuration](#)

[Tomcat Wiki](#)

Find additional important configuration information in:

```
$CATALINA_HOME/RUNNING.txt
```

Developers may be interested in:

[Tomcat 8.5 Release Notes](#)

Getting Help

[FAQ and Mailing Lists](#)

The following mailing lists are available:

- [tomcat-announce](#)
Important announcements, releases, security vulnerability notifications. (Low volume).
- [tomcat-users](#)
User support and discussion
- [taglibs-user](#)
User support and discussion for [Apache Taglibs](#)

Reference

For information about how to create the `registry.cn-hangzhou.aliyuncs.com/haoshuwei/docker-bamboo-agent:v1` image, see [Docker bamboo agent](#).

For more information, see [Bamboo](#).

10 Migrate applications from a Swarm cluster to a Kubernetes cluster

10.1 Swarm cluster to Kubernetes Cluster features comparison

10.1.1 Overview

This topic describes the prerequisites and limits for function comparisons between a Swarm cluster and a Kubernetes cluster that run in Container Service.

Prerequisites

You have created a Kubernetes cluster. For more information, see [#unique_11](#).



Note:

-
- Alibaba Cloud Container Service for Kubernetes supports the following clusters: the dedicated Kubernetes cluster, the managed Kubernetes cluster, the multi-zone Kubernetes cluster, and the serverless Kubernetes cluster (in beta).
- The topic uses creating a Kubernetes cluster as an example to compare the functions between a Swarm and a Kubernetes cluster that run on Container Service.

Limits

- The applications used for the function comparison are as follows:
 - Stateless applications
 - Applications that use a data base or a storage device to store data

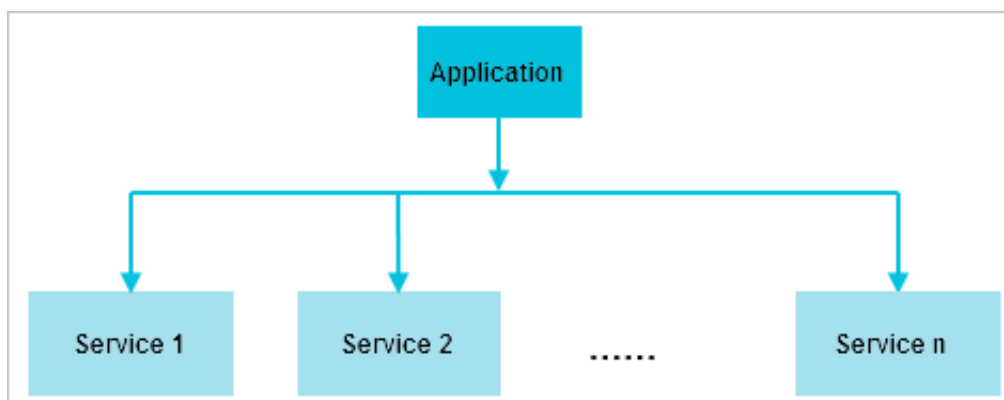
10.1.2 Basic terms

This topic compares the basic terms that are used for both Swarm clusters and Kubernetes clusters.

Application

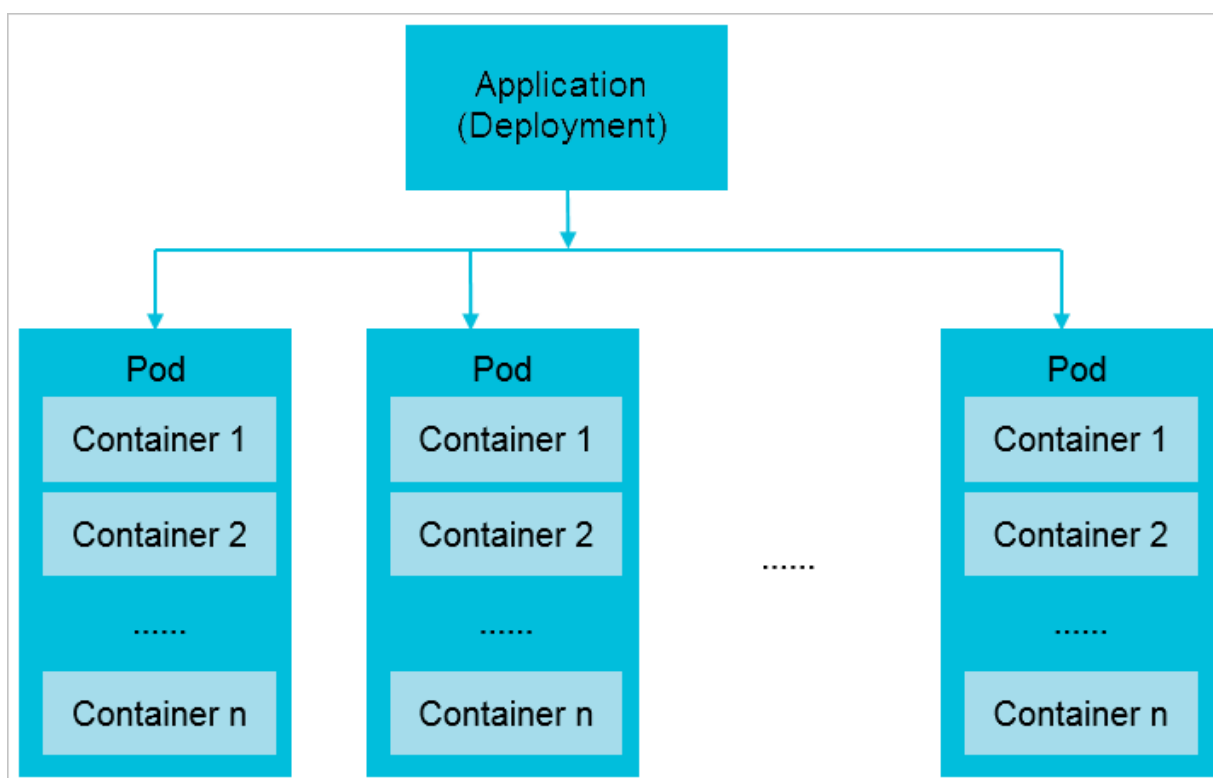
Container Service Swarm clusters

In a Container Service Swarm cluster, applications can be viewed as projects. Each application can include multiple services. Each service is an instance that provides the specific function. Services can be horizontally expanded.



Container Service Kubernetes clusters

In a Container Service Kubernetes cluster, an application, also known as a deployment, is used to provide functions. A deployment contains pods and containers. A pod is the minimum resource unit that can be scheduled in Kubernetes and each pod can contain multiple containers. A pod can be viewed as an instance of the application to which the pod belongs. Multiple pods can be scheduled to different nodes. This means that pods can be horizontally expanded.



Note:

The preceding figure in which each pod has multiple containers is used to show the expansion capability of pods. However, we recommend that you set only one container for each pod.

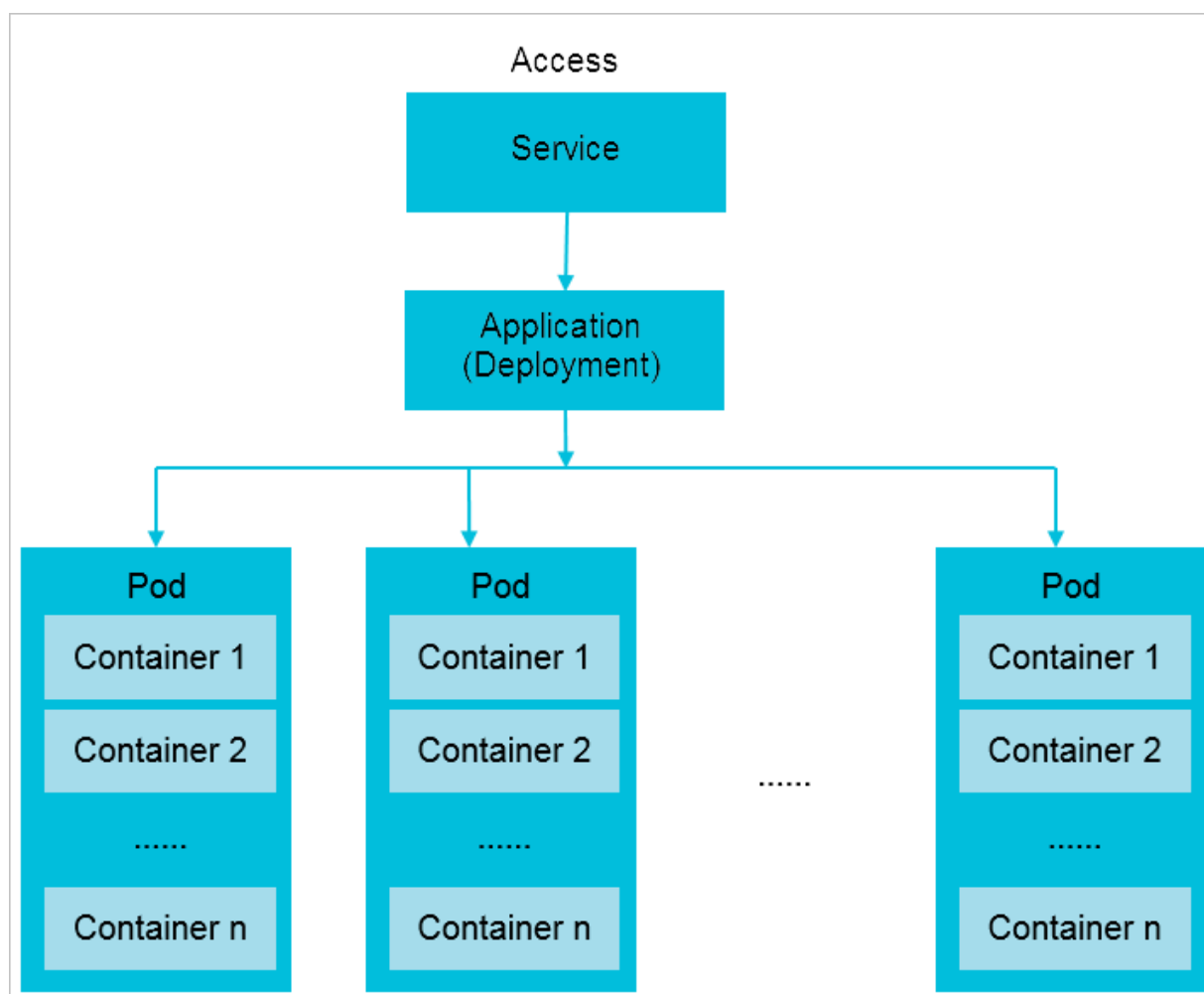
Service

Container Service Swarm clusters

Each service in a Container Service Swarm cluster is an instance that provides a specific function. When you create an application in a Swarm cluster, the access method of the service is exposed directly outside the cluster.

Container Service Kubernetes clusters

The service term in Container Service Kubernetes clusters is an abstract concept. A service can expose the access method of its application (or deployment) outside the cluster.



Application access

Container Service Swarm clusters

When you deploy an application in a Container Service Swarm cluster, you can select one from three types of application access methods that can directly expose the application. The three types of application access methods are:

- <HostIP>:<port>
- Simple routing
- Server Load Balancer (SLB)

Container Service Kubernetes clusters

After you create an application in a Container Service Kubernetes cluster, you must create a service to expose the access method of the application. Then the application becomes accessible. Applications within a Container Service Kubernetes cluster can then access each other through their service names. Service names are only applicable to the access within the cluster. To access the application from outside the cluster, you need to create a service of the NodePort type or a service of the LoadBalancer type to expose the application.

- ClusterIP (It has the same function as a service name. That is, it is applicable to accesses within a cluster.)
- NodePort (It can be viewed as <HostIP>:<port> of Swarm clusters.)
- LoadBalancer (It can be viewed as the SLB of Swarm clusters.)
- Domain name implemented by creating an Ingress (It can be viewed as the simple routing of Swarm clusters.)

10.2 Migration solution overview

This topic describes a solution used to migrate applications from a Swarm cluster to a Kubernetes cluster without causing any disruption to the applications. The solution contains seven steps in which different personnel are involved.

Procedure

1. Standardize the target Swarm cluster.



Note:

Operations described in this step must be performed by the O&M personnel.

- a. Configure the SLB instance for the target Swarm cluster and then use a client to access an application that runs on the target Swarm cluster.



Note:

This step ensures that you can use the canary deployment to direct the traffic destined for the Swarm cluster to the target Kubernetes cluster.

- If you use the SLB instance to access the application, you can check the traffic , or roll back the migrated application workloads when an exception occur.
- If you use any other method to access the application, any newly released features of the application within 48 hours cannot take effect in real time.

- b. Deploy CloudMonitor on the target Swarm cluster to monitor the ECS instances used by the Swarm cluster.

2. Create and configure a Kubernetes cluster.



Note:

Operations described in this step must be performed by the O&M personnel.

- a. Create a Kubernetes cluster.



Note:

We recommend that you create a managed Kubernetes cluster.

- b. Migrate ECS instances and network used by the Swam cluster.
- c. Migrate node tags.
- d. Verify that the VPC that you set for the Kubernetes cluster can provide reliable connectivity.
- e. Migrate volumes.
- f. Migrate ConfigMaps.

3. Use Kompose to migrate applications from the Swarm cluster to the Kubernetes cluster.



Note:

Operations described in this step must be performed by a developer.

- a. Install Kompose and kubectl to prepare an environment to implement the migration task.
 - b. Modify the Swarm orchestration file of the application.
 - c. Convert the Swarm orchestration file to a Kubernetes resource file.
 - d. Deploy the Kubernetes resource file.
 - e. Manually migrate Swarm tags.
 - f. Debug the application migrated to the Kubernetes cluster.
 - g. Migrate the log configurations of the application.
4. Implement regression testing for the migrated application.



Note:

Operations described in this step must be performed by the testing personnel.

- a. Set a testing domain name for the application.
 - b. Test the services provided by the application.
 - c. Verify that the application logs can be collected.
 - d. Verify that the application can be monitored by monitoring products of Alibaba Cloud.
5. Use the canary deployment method to direct traffic destined for the Swarm cluster to the Kubernetes cluster.



Note:

Operations described in this step must be performed by the O&M personnel.

- a. Create a service of the NodePort type in the Kubernetes cluster.
- b. Direct the traffic destined for the Kubernetes cluster back to the Swarm cluster.



Note:

If the Kubernetes cluster cannot work normally, you must perform this step.

6. Direct traffic to the Kubernetes cluster.



Note:

Operations described in this step must be performed by the O&M personnel.

To complete this task, perform one of the following operations:

- Modify the DNS configurations.
- Upgrade the code or configurations of the client.

7. Delete the Swarm cluster and release its resources.



Note:

Operations described in this step must be performed by the O&M personnel.

- a. Verify that no traffic is destined for the Swarm cluster.
- b. Delete the Swarm cluster and release its resources.