Alibaba Cloud
Aliyun Container for Kubernetes

Best Practices

Issue: 20190214

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 `cd /d C:/windows` command to enter the Windows system folder.
`Italics`	It is used for parameters and variables.	`bae log list --instanceid` *`Instance_ID`*
[] or [a|b]	It indicates that it is a optional value, and only one item can be selected.	`ipconfig` *`[-all|-t]`*

Style	Description	Example
{} or {a|b}	It indicates that it is a required value, and only one item can be selected.	`swich` *{stand | slave}*

Contents

1 Cluster

1.1 Plan Kubernetes CIDR blocks under 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 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.

1.2 ECS instance selection and cluster configurations

1.2.1 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 Mater 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*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*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.

1.2.2 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 *Do I select the Terway or Flannel plug-in for my Kubernetes cluster network?*.
- 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: `the remaining resources = the total resources - 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.

1.3 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.

1.4 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 *Create a Kubernetes cluster*.
- You have connected to the Kubernetes cluster through kubectl. For more information, see *Connect to a Kubernetes cluster by using kubectl*.

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 *Update the Kubernetes cluster certificates that are about to expire*.

Run a command to update all certificates

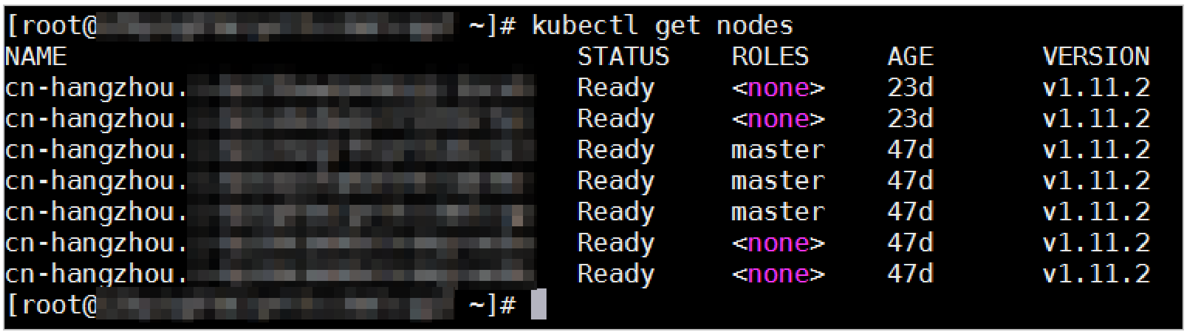
Log on to a Master node and run the following command:

```
$ 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:

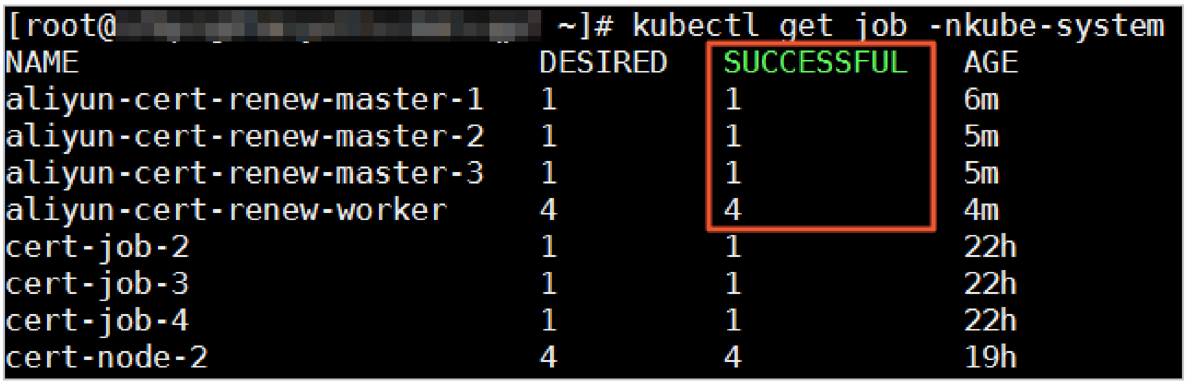
```
$ kubectl get nodes
```



```
[root@[redacted] ~]# kubectl get nodes
NAME                                   STATUS    ROLES     AGE       VERSION
cn-hangzhou.[redacted]                 Ready     <none>    23d       v1.11.2
cn-hangzhou.[redacted]                 Ready     <none>    23d       v1.11.2
cn-hangzhou.[redacted]                 Ready     master    47d       v1.11.2
cn-hangzhou.[redacted]                 Ready     master    47d       v1.11.2
cn-hangzhou.[redacted]                 Ready     master    47d       v1.11.2
cn-hangzhou.[redacted]                 Ready     <none>    47d       v1.11.2
cn-hangzhou.[redacted]                 Ready     <none>    47d       v1.11.2
[root@[redacted] ~]#
```

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@[redacted] ~]# 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.yml* file:

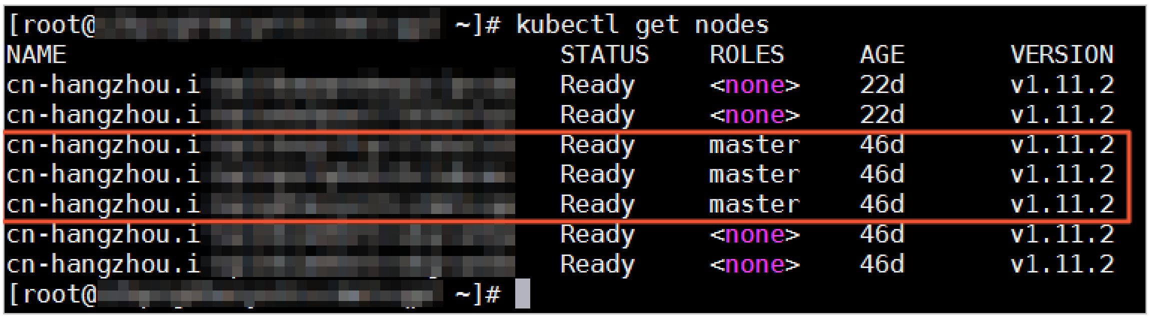
```
apiVersion: batch/v1
kind: Job
metadata:
  name: ${jobname}
  namespace: kube-system
spec:
  backoffLimit: 0
  completions: 1
  parallelism: 1
  template:
    spec:
      activeDeadlineSeconds: 3600
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
              - 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
        imagePullPolicy: Always
        name: ${jobname}
        securityContext:
          privileged: true
        volumeMounts:
        - mountPath: /alicoud-k8s-host
          name: ${jobname}
      hostNetwork: true
      hostPID: true
      restartPolicy: Never
      schedulerName: default-scheduler
      securityContext: {}
      tolerations:
      - 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
     ```

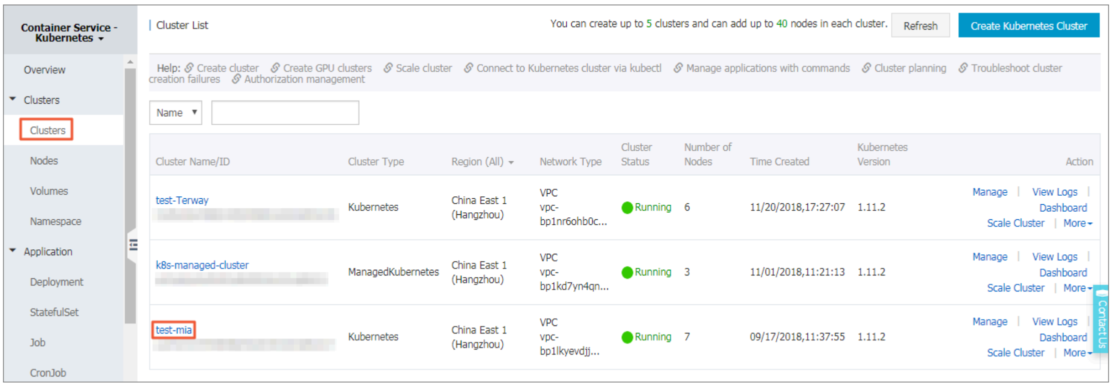


```
[root@[redacted] ~]# kubectl get nodes
NAME                                 STATUS   ROLES    AGE   VERSION
cn-hangzhou.i[redacted]              Ready    <none>   22d   v1.11.2
cn-hangzhou.i[redacted]              Ready    <none>   22d   v1.11.2
cn-hangzhou.i[redacted]              Ready    master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

   - Method 2

     a. Log on to the *Container Service console*.

     b. In the left-side navigation pane under Kubernetes, click Clusters.

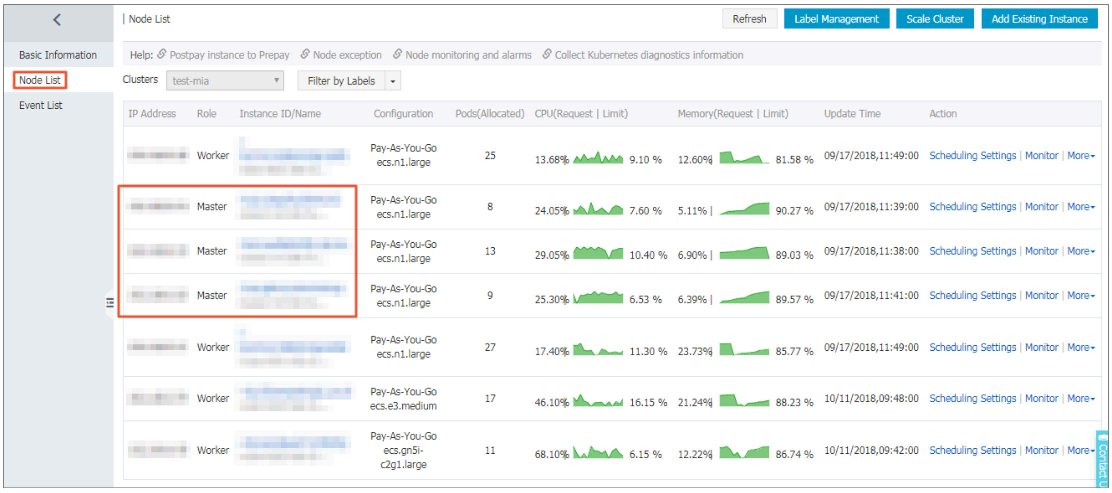


You can create up to 5 clusters and can add up to 40 nodes in each cluster.

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

Cluster Name/ID	Cluster Type	Region (All)	Network Type	Cluster Status	Number of Nodes	Time Created	Kubernetes Version	Action
test-Terway [redacted]	Kubernetes	China East 1 (Hangzhou)	VPC vpc-bp1nr6ohb0c...	● Running	6	11/20/2018,17:27:07	1.11.2	Manage | View Logs | Dashboard Scale Cluster | More
k8s-managed-cluster [redacted]	ManagedKubernetes	China East 1 (Hangzhou)	VPC vpc-bp1kd7yn4qn...	● Running	3	11/01/2018,11:21:13	1.11.2	Manage | View Logs | Dashboard Scale Cluster | More
test-mia [redacted]	Kubernetes	China East 1 (Hangzhou)	VPC vpc-bp1lkyevdjj...	● Running	7	09/17/2018,11:37:55	1.11.2	Manage | View Logs | Dashboard Scale Cluster | More

     c. 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.



Node List

Help: Postpay instance to Prepay Node exception Node monitoring and alarms Collect Kubernetes diagnostics information

Clusters test-mia Filter by Labels

IP Address	Role	Instance ID/Name	Configuration	Pods(Allocated)	CPU(Request | Limit)	Memory(Request | Limit)	Update Time	Action
[redacted]	Worker	[redacted]	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
[redacted]	Master	[redacted]	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
[redacted]	Master	[redacted]	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
[redacted]	Master	[redacted]	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
[redacted]	Worker	[redacted]	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
[redacted]	Worker	[redacted]	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
[redacted]	Worker	[redacted]	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.

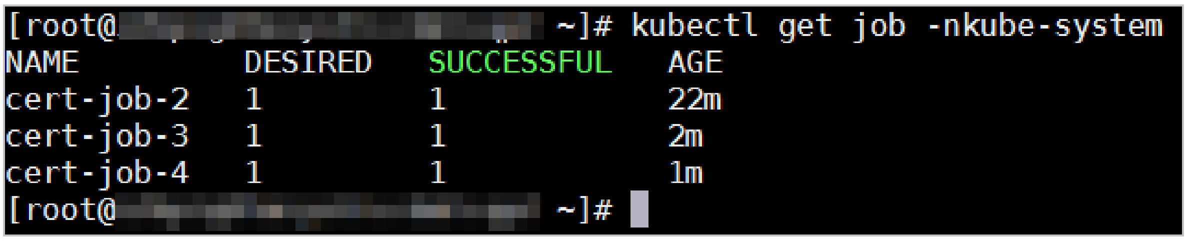
4. Run the following command to create a Job:

```
$ 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@[redacted] ~]# 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@[redacted] ~]#
```

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
  completions: ${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 specfcially, 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):

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

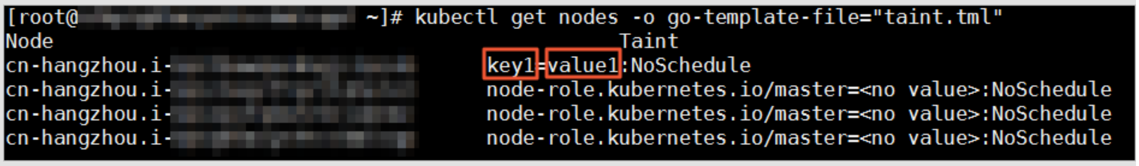
The method to obtain `${name}` and `${value}` is as follows:

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

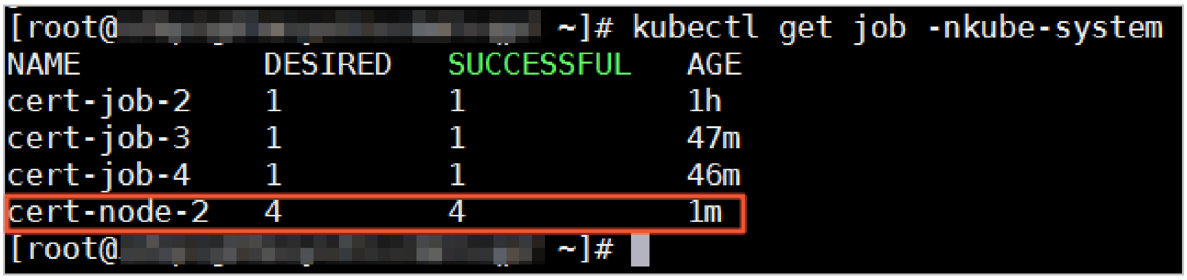
- `${key}` is the cluster CAKey. In this example, the `key` variable is replaced with the CAKey obtained in step 3 of *Manually update Worker node certificates*.

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@[redacted] ~]# 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@[redacted] ~]#
```

2 Network

2.1 Deploy a highly reliable Ingress controller

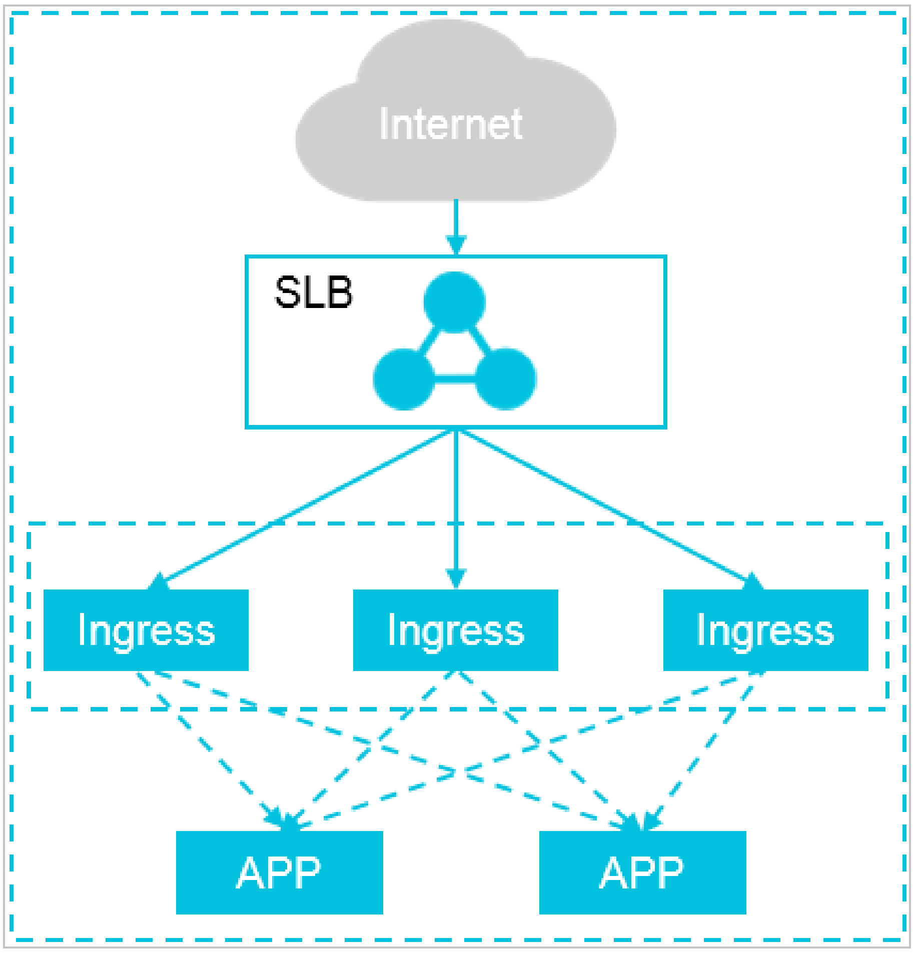
Ingress is a set of rules that authorize external access to the services in a Kubernetes cluster, providing Layer-7 Server Load Balancer capabilities. You can configure Ingress to provide externally accessible URLs, SLB, SSL, and name-based virtual hosts. Ingress requires high reliability because Ingress 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 *Create a Kubernetes cluster*.
- You have connected to the Master node by using SSH. For more information, see *Access Kubernetes clusters by using SSH*.

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.xx.x.xx    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-xqmfn                    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 configurat ions only, you can label the target nodes.

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

```
$ kubectl get node
NAME                                 STATUS   ROLES    AGE   VERSION
cn-hangzhou.i-bp11bcmsna8d4bpf17bc   Ready    master   21d   v1.11.5
cn-hangzhou.i-bp12h6biv9bg24lmdc2o   Ready    <none>   21d   v1.11.5
cn-hangzhou.i-bp12h6biv9bg24lmdc2p   Ready    <none>   21d   v1.11.5
cn-hangzhou.i-bp12h6biv9bg24lmdc2q   Ready    <none>   21d   v1.11.5
cn-hangzhou.i-bp181pofzyyksie2ow03   Ready    master   21d   v1.11.5
```

```
cn-hangzhou.i-bp1cbsg6rf3580z6uyo7   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-bp12h6biv9bg24lmdc2o` and the Ingress node `cn-hangzhou.i-bp12h6biv9bg24lmdc2p`:

   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-bp12h6biv9bg24lmdc2o node-role.
   kubernetes.io/ingress="true"
   node/cn-hangzhou.i-bp12h6biv9bg24lmdc2o labeled
   ```

   ```
   $ kubectl label nodes cn-hangzhou.i-bp12h6biv9bg24lmdc2p node-role.
   kubernetes.io/ingress="true"
   node/cn-hangzhou.i-bp12h6biv9bg24lmdc2p 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": {"nodeSelector": {"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 *Create a Kubernetes cluster*.
- You have created a cloud disk. For more information, see *Create a cloud disk*.
- You have connected to the Kubernetes cluster by using kubectl, see *Connect to a Kubernetes cluster by using kubectl*.

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: PersistentVolume
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
  accessModes:
    - ReadWriteOnce
  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`.

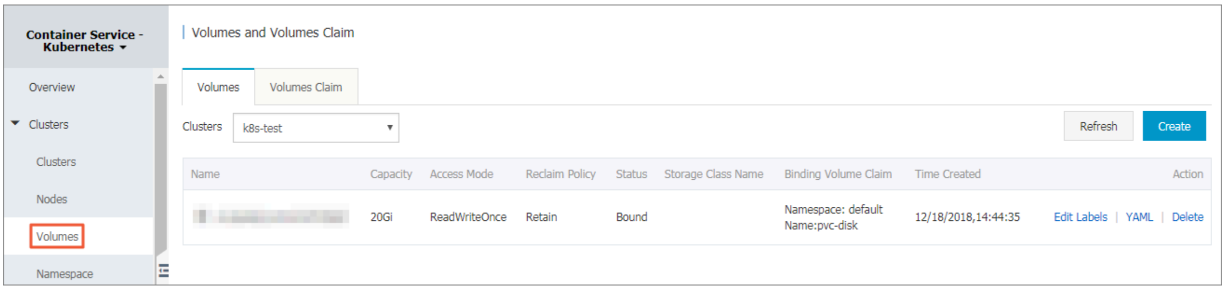
If you use a Kubernetes cluster that has multiple zones, you must set the `failure-domain.beta.kubernetes.io/zone` parameter and the `failure-domain.beta.kubernetes.io/region` parameter so that you can guarantee that your pod can be scheduled to the zone in which the cloud disk resides.

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.



Clusters: k8s-test

Name	Capacity	Access Mode	Reclaim Policy	Status	Storage Class Name	Binding Volume Claim	Time Created	Action
[redacted]	20Gi	ReadWriteOnce	Retain	Bound		Namespace: default Name:pvc-disk	12/18/2018,14:44:35	Edit Labels | YAML | Delete

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.

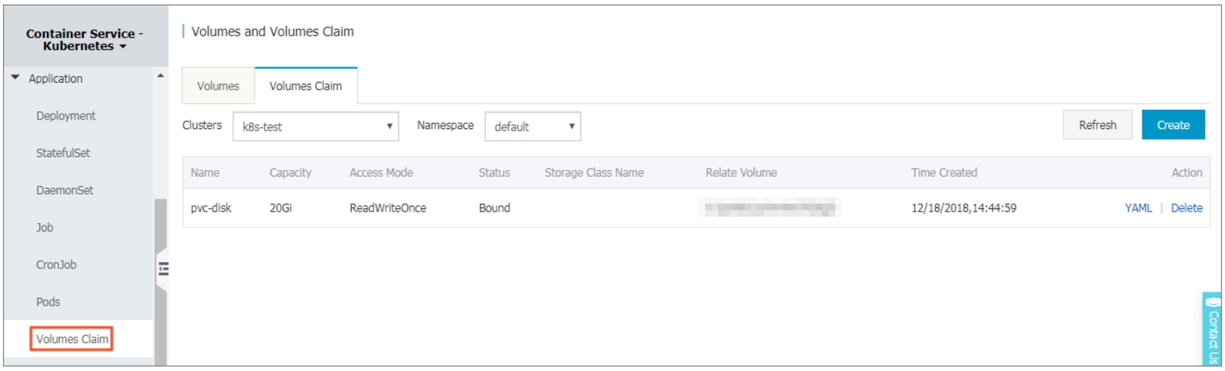
```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-disk
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 20Gi
  selector:
    matchLabels:
      alicloud-pvname: <your-disk-id>
```

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.



Clusters: k8s-test Namespace: default

Name	Capacity	Access Mode	Status	Storage Class Name	Relate Volume	Time Created	Action
pvc-disk	20Gi	ReadWriteOnce	Bound		[redacted]	12/18/2018,14:44:59	YAML | Delete

Create an application

1. Create a *static.yaml* file.

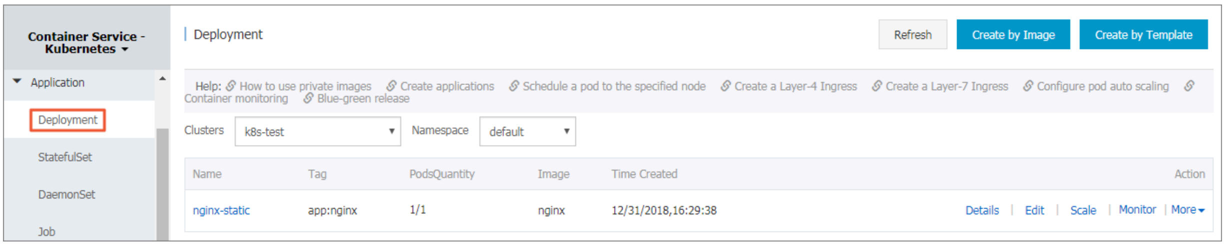
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-static
  labels:
    app: nginx
spec:
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx
        volumeMounts:
          - 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.



Clusters: k8s-test
Namespace: default

Name	Tag	PodsQuantity	Image	Time Created	Action
nginx-static	app:nginx	1/1	nginx	12/31/2018,16:29:38	Details | Edit | Scale | Monitor | More

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   Terminating   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   Terminating   0     73s
nginx-static-78c7dcb9d7-h6brd   0/2   Init:0/1   0     5s
nginx-static-78c7dcb9d7-g9lll   0/2   Terminating   0     78s
nginx-static-78c7dcb9d7-g9lll   0/2   Terminating   0     78s
nginx-static-78c7dcb9d7-h6brd   0/2   PodInitializing   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 *Create a Kubernetes cluster*.
- You have connected to the Kubernetes cluster by using kubectl, see *Connect to a Kubernetes cluster by using kubectl*.
- 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
```

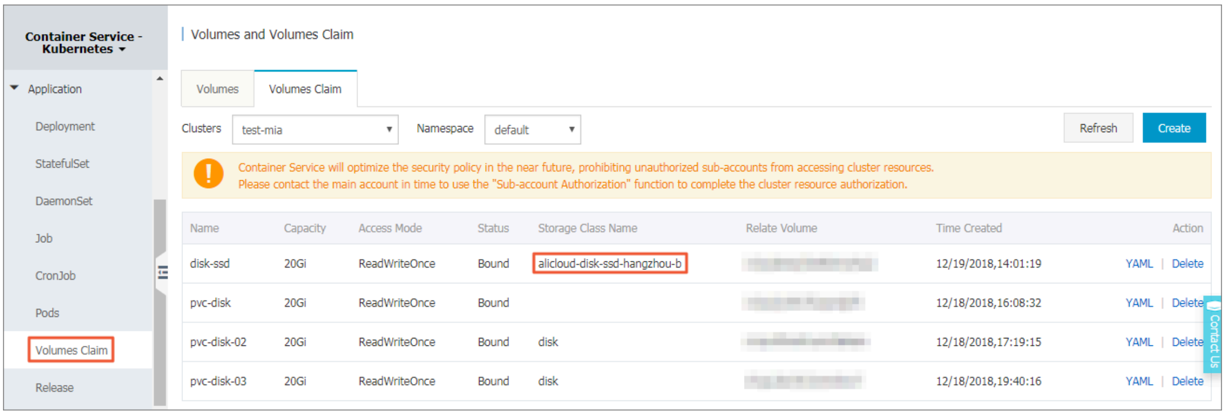
```
  resources:
    requests:
      storage: 20Gi
```

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 `StorageClass`, and the PVC is associated with the volume.



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-b	[redacted]	12/19/2018,14:01:19	YAML | Delete
pvc-disk	20Gi	ReadWriteOnce	Bound		[redacted]	12/18/2018,16:08:32	YAML | Delete
pvc-disk-02	20Gi	ReadWriteOnce	Bound	disk	[redacted]	12/18/2018,17:19:15	YAML | Delete
pvc-disk-03	20Gi	ReadWriteOnce	Bound	disk	[redacted]	12/18/2018,19:40:16	YAML | Delete

Create an application

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

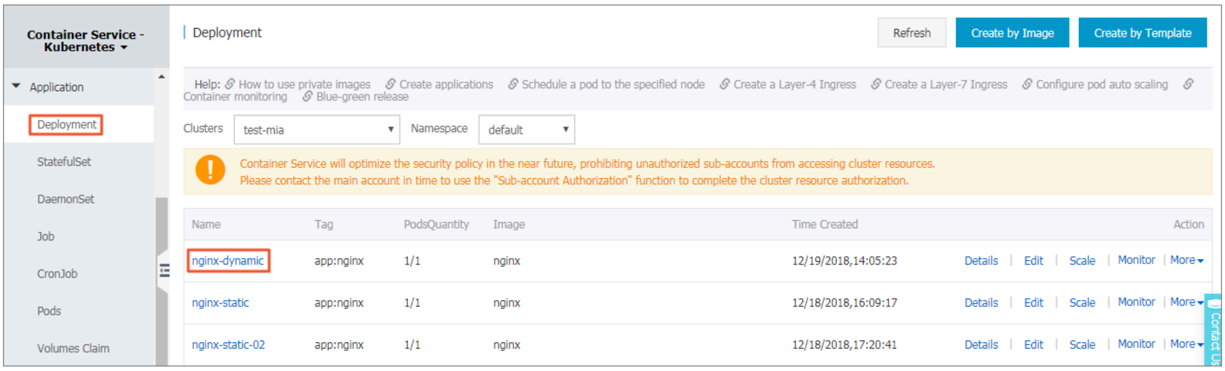
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-dynamic
  labels:
    app: nginx
spec:
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx
        volumeMounts:
          - 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.



Clusters: test-mia   Namespace: default

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	Tag	PodsQuantity	Image	Time Created	Action
nginx-dynamic	app:nginx	1/1	nginx	12/19/2018,14:05:23	Details | Edit | Scale | Monitor | More
nginx-static	app:nginx	1/1	nginx	12/18/2018,16:09:17	Details | Edit | Scale | Monitor | More
nginx-static-02	app:nginx	1/1	nginx	12/18/2018,17:20:41	Details | Edit | Scale | Monitor | More

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   Terminating   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   Terminating   0     7m32s
nginx-dynamic-5c74594ccb-zl9pf   0/2   Terminating   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 *Create a Kubernetes cluster*.
- You have connected to the Master node of the Kubernetes cluster. For more information, see *Connect to a Kubernetes cluster by using kubectl*.

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                         1/1     Running   0          6m
```

5. Run the following command to view the PVCs:

```
$ kubectl get pvc
NAME             STATUS   VOLUME                   CAPACITY   ACCESS
 MODES   STORAGECLASS        AGE
disk-ssd-web-0   Bound    d-2zegw7et6xc96nbojuoo   20Gi       RWO
         alicloud-disk-ssd   7m
disk-ssd-web-1   Bound    d-2zefbrqggvkd10xb523h   20Gi       RWO
         alicloud-disk-ssd   6m
```

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   AGE
web-0                         1/1     Running   0          34m
web-1                         1/1     Running   0          33m
web-2                         1/1     Running   0          26m
```

3. Run the following command to view the PVCs:

```
$ kubectl get pvc
NAME             STATUS   VOLUME                   CAPACITY   ACCESS
 MODES   STORAGECLASS        AGE
disk-ssd-web-0   Bound    d-2zegw7et6xc96nbojuoo   20Gi       RWO
         alicloud-disk-ssd   35m
disk-ssd-web-1   Bound    d-2zefbrqggvkd10xb523h   20Gi       RWO
         alicloud-disk-ssd   34m
disk-ssd-web-2   Bound    d-2ze4jx1zymn4n9j3pic2   20Gi       RWO
         alicloud-disk-ssd   27m
```

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                         1/1     Running   0          38m
web-1                         1/1     Running   0          38m
```

3. **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-2zegw7et6xc96nbojuoo   20Gi       RWO
         alicloud-disk-ssd   39m
disk-ssd-web-1   Bound    d-2zefbrqggvkd10xb523h   20Gi       RWO
         alicloud-disk-ssd   39m
disk-ssd-web-2   Bound    d-2ze4jx1zymn4n9j3pic2   20Gi       RWO
         alicloud-disk-ssd   31m
```

Rescale 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   AGE
web-0                         1/1     Running   0          1h
web-1                         1/1     Running   0          1h
web-2                         1/1     Running   0          8s
```

3. **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-2zegw7et6xc96nbojuoo   20Gi       RWO
         alicloud-disk-ssd   1h
disk-ssd-web-1   Bound    d-2zefbrqggvkd10xb523h   20Gi       RWO
         alicloud-disk-ssd   1h
```

```
disk-ssd-web-2   Bound    d-2ze4jx1zymn4n9j3pic2   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-2zegw7et6xc96nbojuoo   20Gi        RWO
         alicloud-disk-ssd   1h
disk-ssd-web-1   Bound    d-2zefbrqggvkd10xb523h   20Gi        RWO
         alicloud-disk-ssd   1h
disk-ssd-web-2   Bound    d-2ze4jx1zymn4n9j3pic2   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   Terminating   0     89s
web-1   0/1   Terminating   0     89s
web-1   0/1   Terminating   0     90s
web-1   0/1   Terminating   0     90s
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     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 *`statefulset`* file in the *`/data`* path:**

```
$ kubectl exec web-1 touch /data/statefulset
```

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

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

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 *`statefulset`* has not been removed, indicating that data in the cloud disk can be stored persistently:**

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

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 applicatio n 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 *Create a Kubernetes cluster*.
- You have connected to the Kubernetes cluster by using kubectl, see *Connect to a Kubernetes cluster by using kubectl*.
- You have created a NAS file system in the NAS console. For more information, see *Create a file system*. 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 *Add a mount point*. You must make sure that the NAS file system and your cluster are in the same VPC.

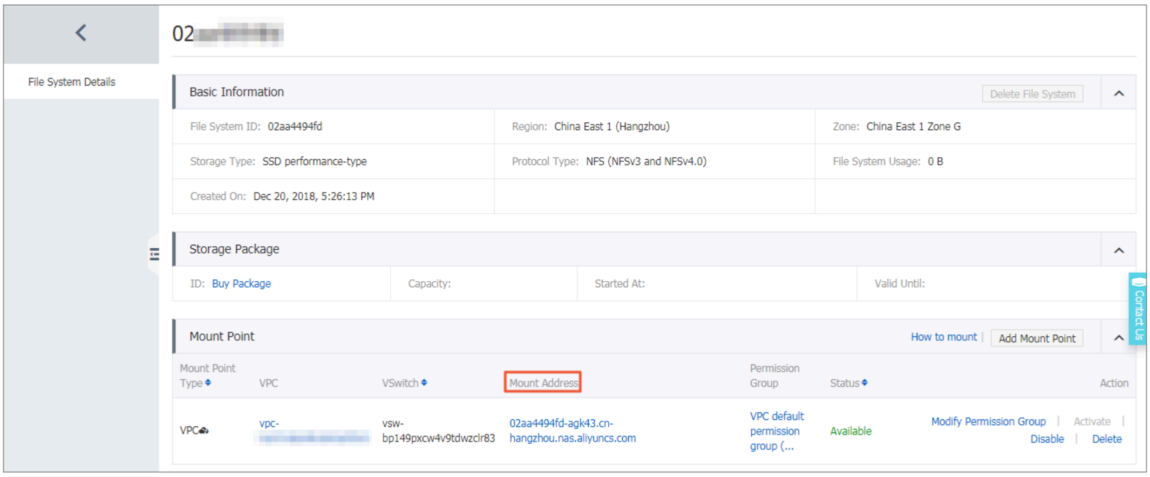
Create a PV

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

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-nas
  labels:
    alicloud-pvname: pv-nas
spec:
  capacity:
    storage: 5Gi
  accessModes:
    - ReadWriteMany
  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.



02[redacted]

Basic Information

File System ID: 02aa4494fd

Region: China East 1 (Hangzhou)

Zone: China East 1 Zone G

Storage Type: SSD performance-type

Protocol Type: NFS (NFSv3 and NFSv4.0)

File System Usage: 0 B

Created On: Dec 20, 2018, 5:26:13 PM

Storage Package

ID: Buy Package

Capacity:

Started At:

Valid Until:

Mount Point

Mount Point Type	VPC	VSwitch	Mount Address	Permission Group	Status	Action
VPC	vpc-[redacted]	vsw-bp149pxcw4v9tdwzclr83	02aa4494fd-agk43.cn-hangzhou.nas.aliyuncs.com	VPC default permission group (...	Available	Modify Permission Group | Activate | Disable | Delete

- `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.

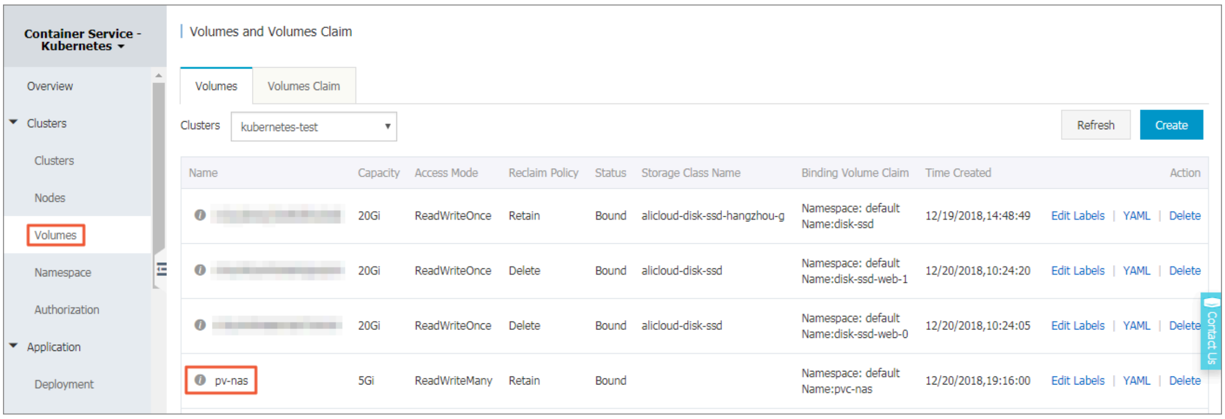
> Note:
>
> - Access permission to the root directory of the NAS file system cannot be set.
> - If you set the `mode` parameter for a NAS file system that stores a large amount of data, the process of mounting the NAS file system to a cluster may take an excessive amount of time or even fail. We recommend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



Clusters: kubernetes-test

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.

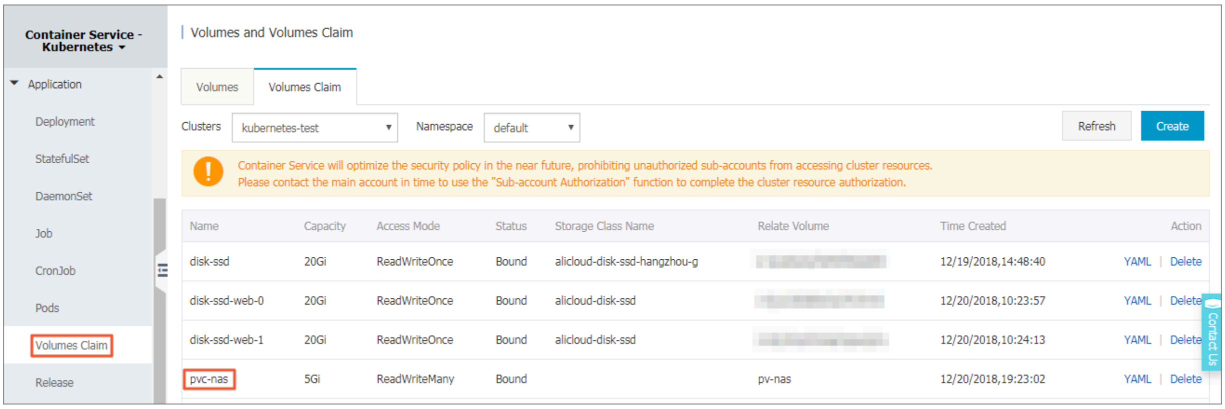
```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-nas
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 5Gi
  selector:
    matchLabels:
      alicloud-pvname: pv-nas
```

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.



Volumes and Volumes Claim

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	[redacted]	12/19/2018,14:48:40	YAML | Delete
disk-ssd-web-0	20Gi	ReadWriteOnce	Bound	alicloud-disk-ssd	[redacted]	12/20/2018,10:23:57	YAML | Delete
disk-ssd-web-1	20Gi	ReadWriteOnce	Bound	alicloud-disk-ssd	[redacted]	12/20/2018,10:24:13	YAML | Delete
pvc-nas	5Gi	ReadWriteMany	Bound		pv-nas	12/20/2018,19:23:02	YAML | Delete

Create an application

1. Create a *nas.yaml*.

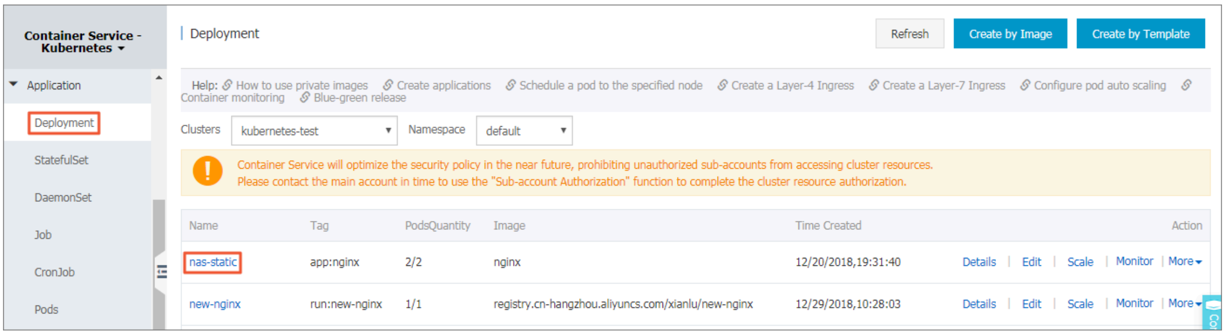
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nas-static
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx
        ports:
        - 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.



Clusters: kubernetes-test Namespace: default

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	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

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   AGE
nas-static-f96b6b5d7-rcb2f        1/1     Running   0          9m
nas-static-f96b6b5d7-wthmb        1/1     Running   0          9m
```

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
nas
```

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

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   Terminating   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   ContainerCreating   0     0s
nas-static-f96b6b5d7-wthmb   1/1   Terminating   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   ContainerCreating   0     0s
nas-static-f96b6b5d7-rcb2f   0/1   Terminating   0     28m
nas-static-f96b6b5d7-wthmb   0/1   Terminating   0     28m
nas-static-f96b6b5d7-rcb2f   0/1   Terminating   0     28m
nas-static-f96b6b5d7-rcb2f   0/1   Terminating   0     28m
nas-static-f96b6b5d7-wnqdj   1/1   Running   0     10s
nas-static-f96b6b5d7-wthmb   0/1   Terminating   0     28m
nas-static-f96b6b5d7-wthmb   0/1   Terminating   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-nwkds 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 *Create a Kubernetes cluster*.
- You have connected to the Kubernetes cluster by using kubectl, see *Connect to a Kubernetes cluster by using kubectl*.
- You have created a bucket in the OSS console, see *Create a bucket*.

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: PersistentVolume
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 *Regions and endpoints*. 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.

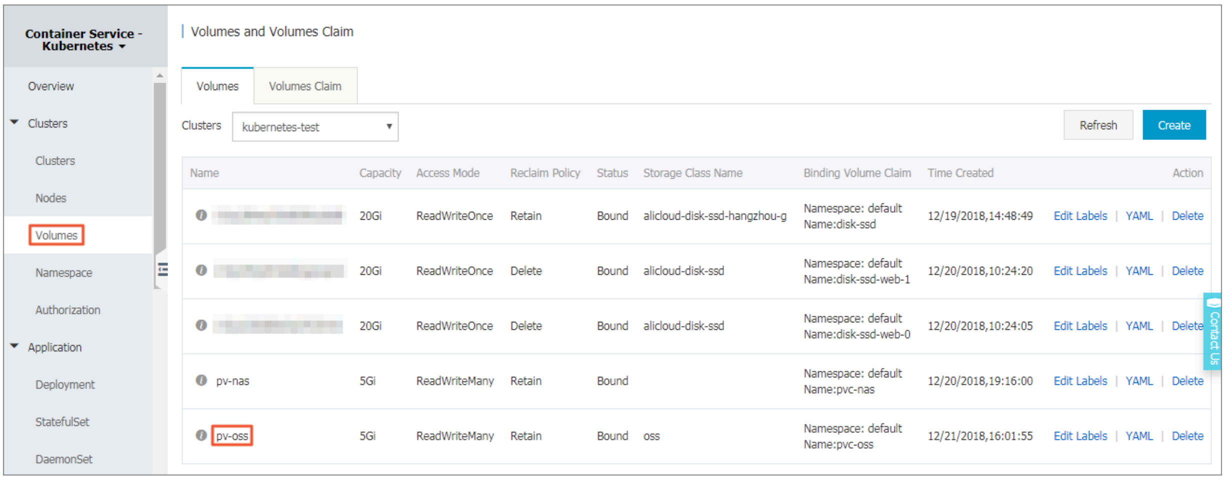
- `akSecret`: indicates your AccessKey Secret. Use the same method to obtain this parameter value as that to obtain the value of the `akId` parameter.
- `otherOpts`: indicates custom parameters for mounting the OSS bucket. Set this parameter in the format of `-o *** -o ***`. For more information, see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

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.



Clusters: kubernetes-test

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
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.

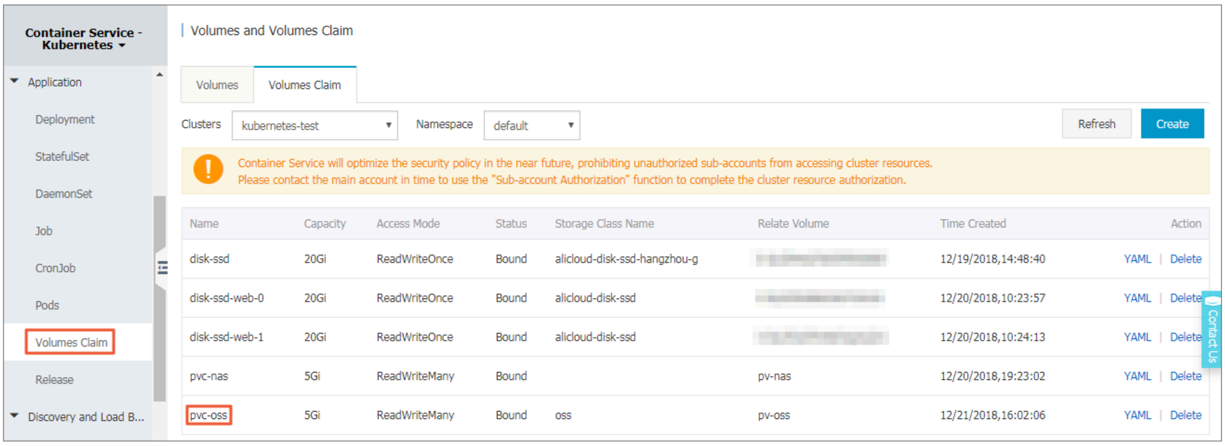
```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-oss
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: oss
  resources:
    requests:
      storage: 5Gi
  selector:
    matchLabels:
        alicloud-pvname: pv-oss
```

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.



Clusters: kubernetes-test
Namespace: default

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	[redacted]	12/19/2018,14:48:40	YAML | Delete
disk-ssd-web-0	20Gi	ReadWriteOnce	Bound	alicloud-disk-ssd	[redacted]	12/20/2018,10:23:57	YAML | Delete
disk-ssd-web-1	20Gi	ReadWriteOnce	Bound	alicloud-disk-ssd	[redacted]	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:

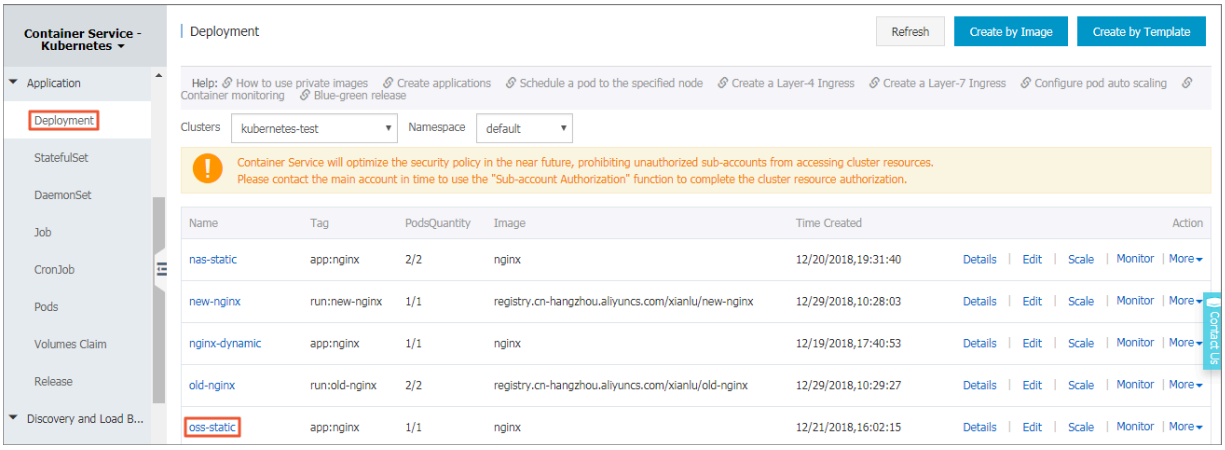
For more information about `livenessProbe`, see *Use Alibaba Cloud OSS volumes*.

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.



Clusters: kubernetes-test Namespace: default

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	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   Terminating   0     78m
oss-static-66fbb85b67-zlvmw  0/1   Pending   0     <invalid>
oss-static-66fbb85b67-zlvmw  0/1   Pending   0     <invalid>
oss-static-66fbb85b67-zlvmw  0/1   ContainerCreating   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-zlvmw  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-zlvmw     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-zlvmw 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
```

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 Application

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

4.3.1 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

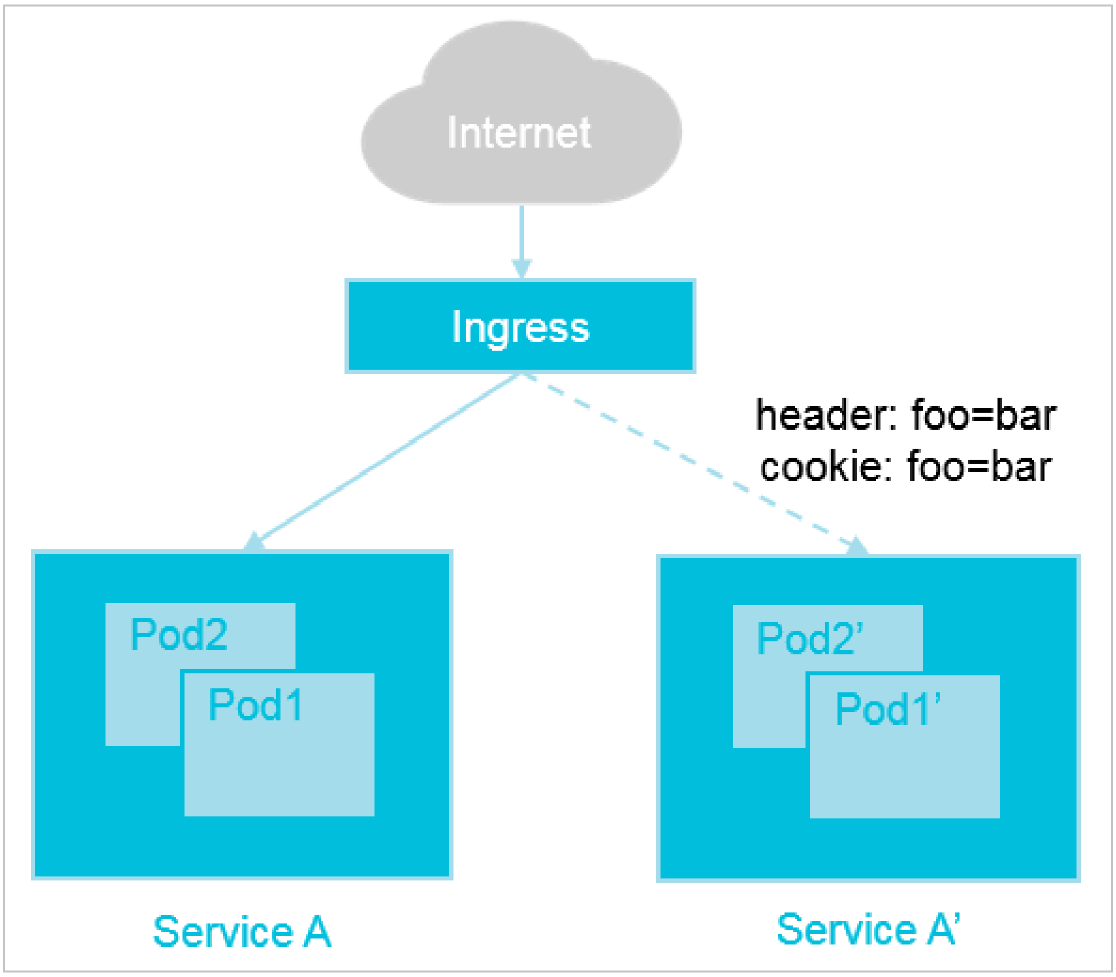
With a gray release or a blue/green deployment, you can create two identical production environments for the latest version of the target software and an earlier version. Then you can apply specific rules to reroute traffic from the earlier version to the latest version without affecting the software 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

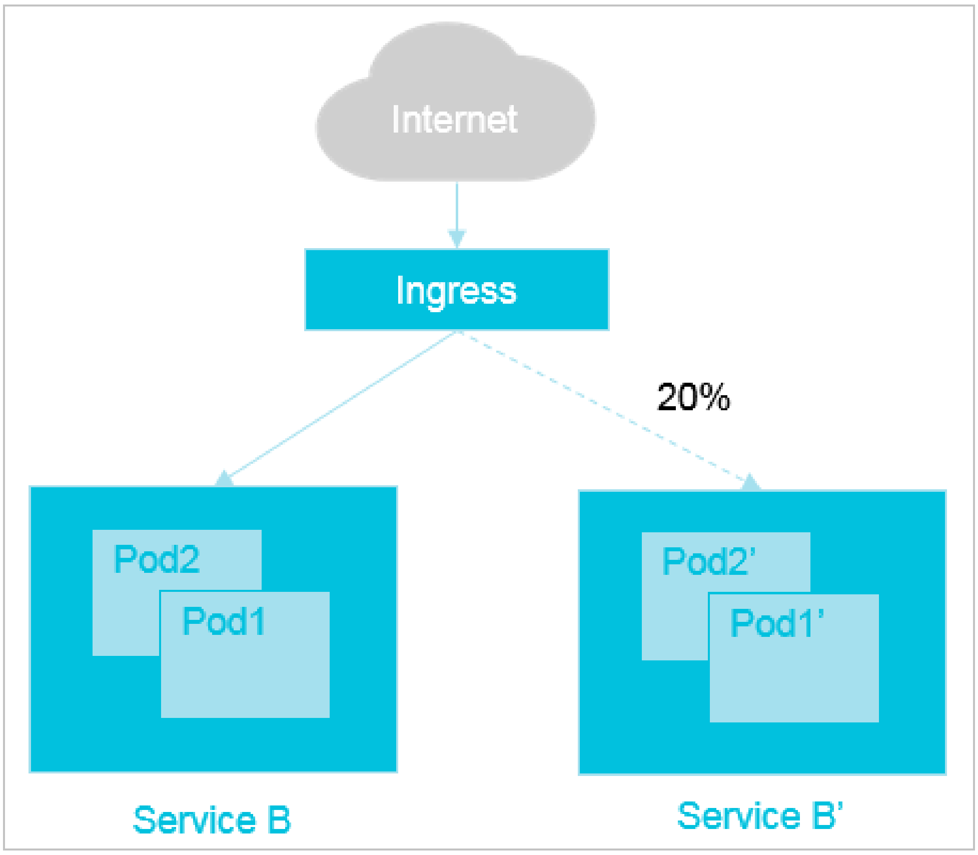
Scenario 1

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

4.3.2 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
```

4.3.3 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`, 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.

4.3.4 Step 1: Deploy a service

This topic describes how to deploy a service.

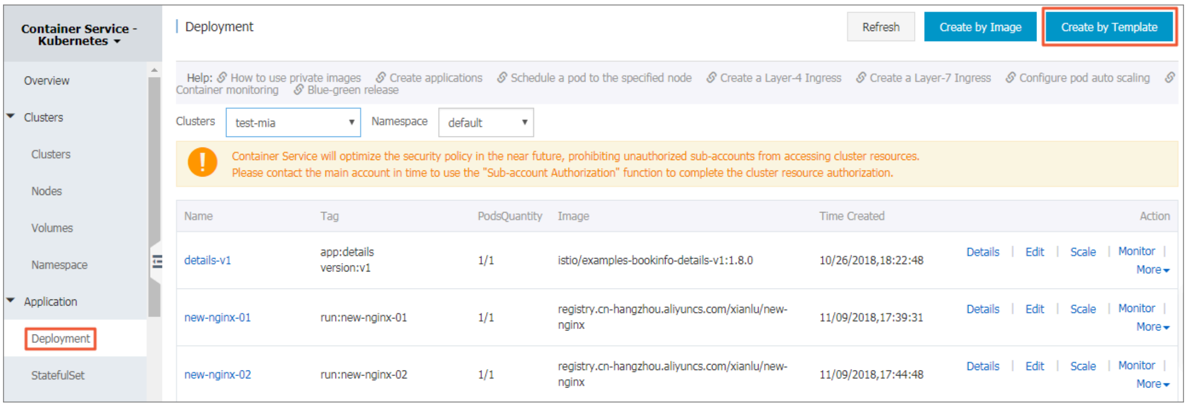
Prerequisites

- You have created a Kubernetes cluster. For more information, see *Create a Kubernetes cluster*.

- You have connected to the Kubernetes cluster by using kubectl. For more information, see *Connect to a Kubernetes cluster by using kubectl*.

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.

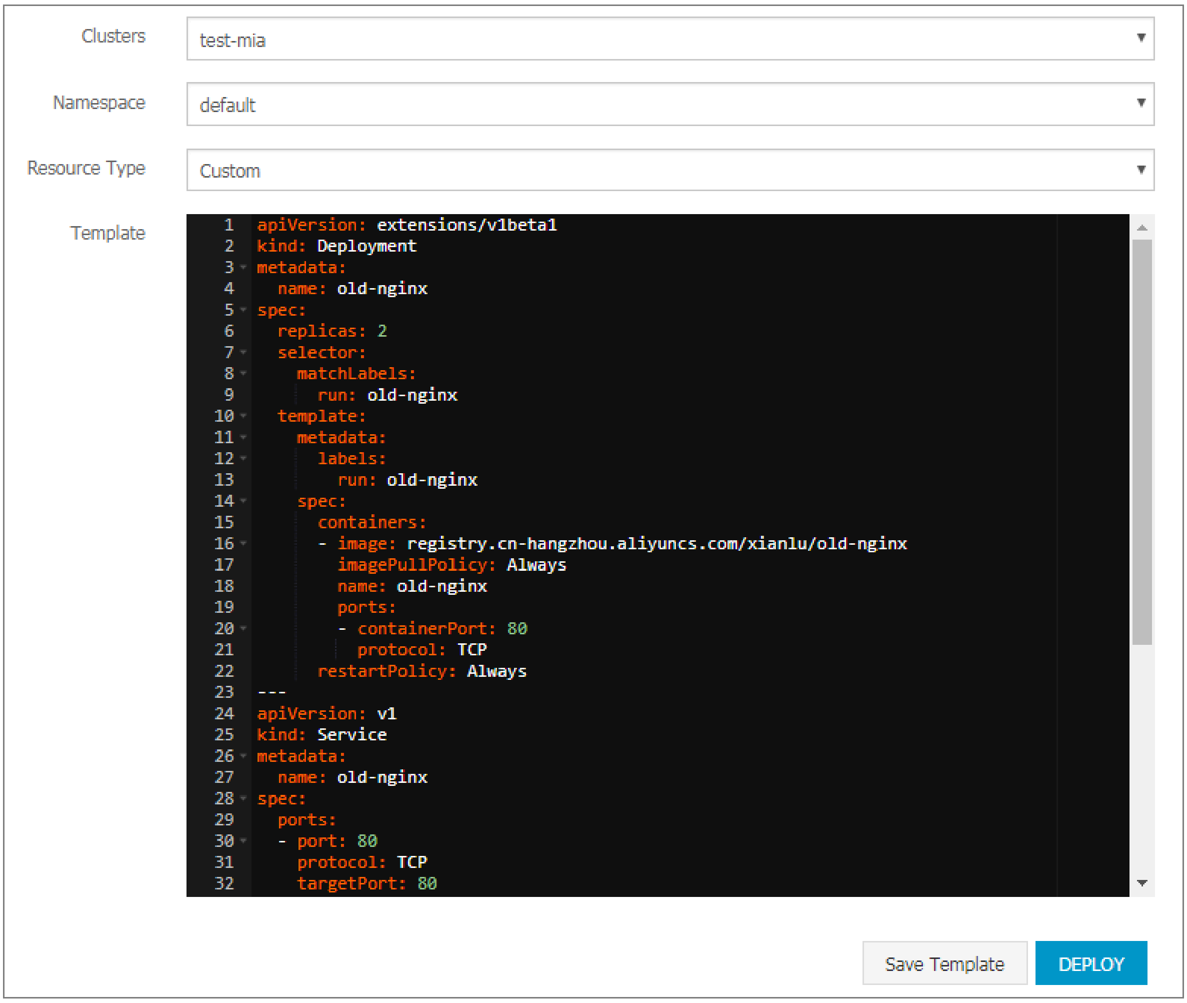


Clusters: test-mia Namespace: default

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	Tag	PodsQuantity	Image	Time Created	Action
details-v1	app:details version:v1	1/1	istio/examples-bookinfo-details-v1:1.8.0	10/26/2018,18:22:48	Details | Edit | Scale | Monitor | More
new-nginx-01	run:new-nginx-01	1/1	registry.cn-hangzhou.aliyuncs.com/xianlu/new-nginx	11/09/2018,17:39:31	Details | Edit | Scale | Monitor | More
new-nginx-02	run:new-nginx-02	1/1	registry.cn-hangzhou.aliyuncs.com/xianlu/new-nginx	11/09/2018,17:44:48	Details | Edit | Scale | Monitor | More

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



Clusters: test-mia

Namespace: default

Resource Type: Custom

Template:

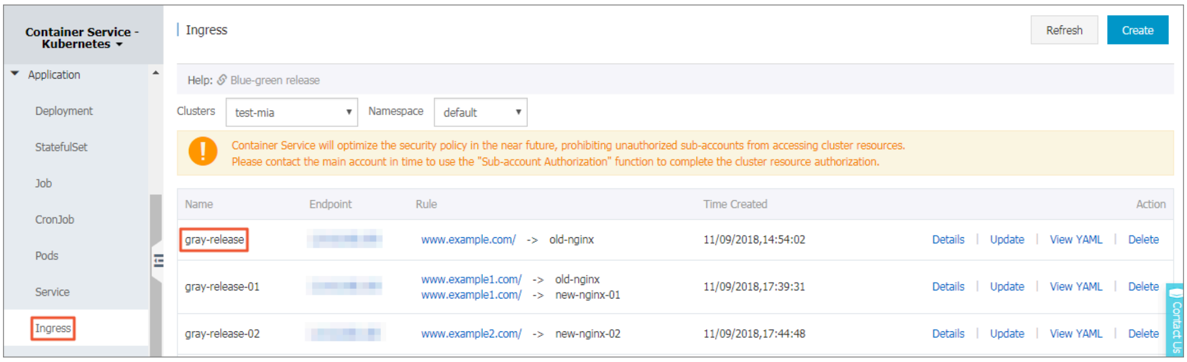
```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: old-nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      run: old-nginx
  template:
    metadata:
      labels:
        run: old-nginx
    spec:
      containers:
      - image: registry.cn-hangzhou.aliyuncs.com/xianlu/old-nginx
        imagePullPolicy: Always
        name: old-nginx
        ports:
        - containerPort: 80
          protocol: TCP
      restartPolicy: Always
---
apiVersion: v1
kind: Service
metadata:
  name: old-nginx
spec:
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
```

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

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

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



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	Endpoint	Rule	Time Created	Action
gray-release	[redacted]	www.example.com/ -> old-nginx	11/09/2018,14:54:02	Details | Update | View YAML | Delete
gray-release-01	[redacted]	www.example1.com/ -> old-nginx www.example1.com/ -> new-nginx-01	11/09/2018,17:39:31	Details | Update | View YAML | Delete
gray-release-02	[redacted]	www.example2.com/ -> new-nginx-02	11/09/2018,17:44:48	Details | Update | View YAML | Delete

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>
```

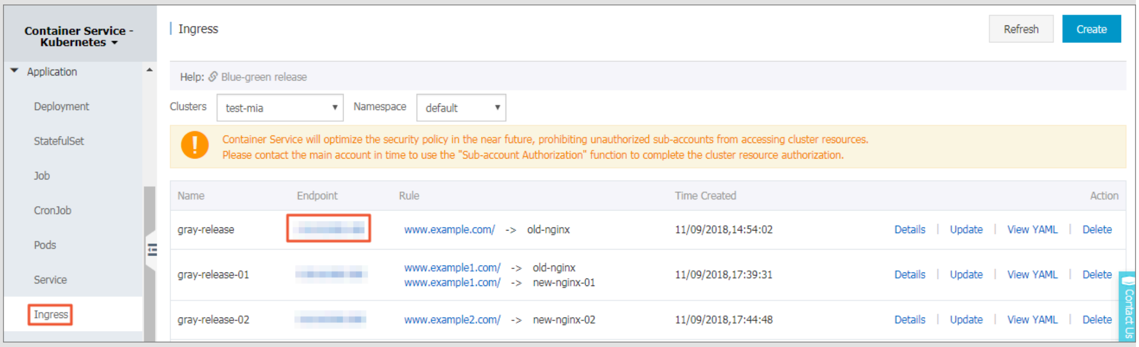
Note:

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

- Run the following command:

```
kubectl get ingress
```

- Under the Kubernetes menu, choose Application > Ingress, and view the endpoint information of the target Ingress.



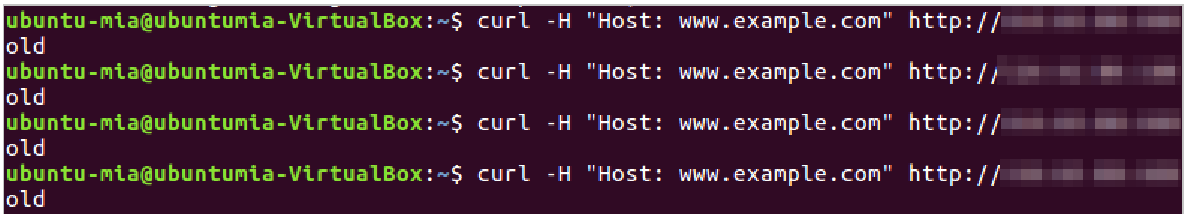
Ingress

Help: Blue-green release

Clusters: test-mia Namespace: default

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	Endpoint	Rule	Time Created	Action
gray-release	[redacted]	www.example.com/ -> old-nginx	11/09/2018,14:54:02	Details | Update | View YAML | Delete
gray-release-01	[redacted]	www.example1.com/ -> old-nginx www.example1.com/ -> new-nginx-01	11/09/2018,17:39:31	Details | Update | View YAML | Delete
gray-release-02	[redacted]	www.example2.com/ -> new-nginx-02	11/09/2018,17:44:48	Details | Update | View YAML | Delete



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

4.3.5 Step 2: Release the latest version of a service

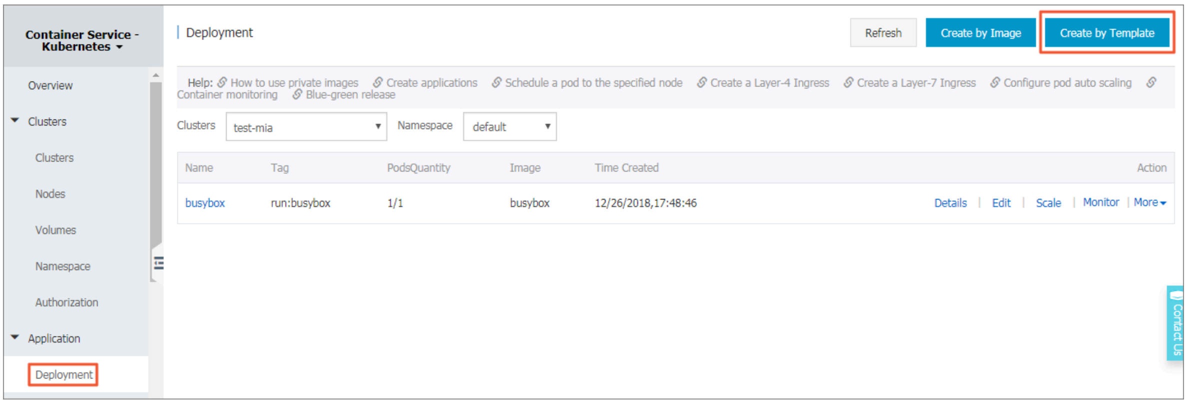
This topic describes how to release the latest version of a service by using the Ingress function provided by Alibaba Cloud Container Service for Kubernetes.

Prerequisites

- You have created a Kubernetes cluster. For more information, see *Create a Kubernetes cluster*.
- You have connected to the Kubernetes cluster by using kubectl. For more information, see *Connect to a Kubernetes cluster by using kubectl*.

Procedure

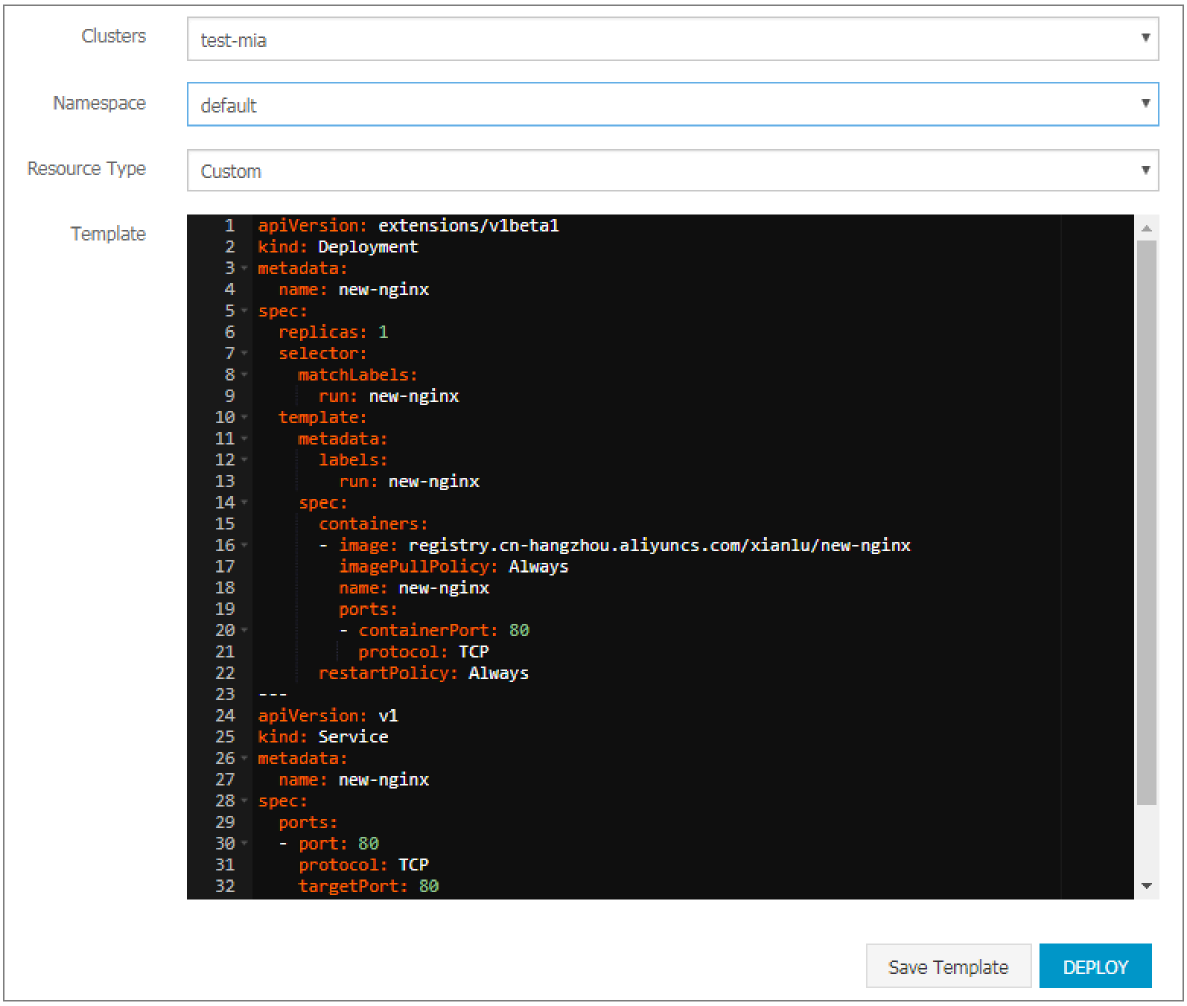
1. Log on to the *Container Service console*.
2. In the left-side navigation pane under Kubernetes, choose Application > Deployment.
3. In the upper-right corner, click Create by Template.


Container Service - Kubernetes
Overview
Clusters
Clusters
Nodes
Volumes
Namespace
Authorization
Application
Deployment
Deployment
Refresh
Create by Image
Create by Template
Help: How to use private images Create applications Schedule a pod to the specified node Create a Layer-4 Ingress Create a Layer-7 Ingress Configure pod auto scaling Container monitoring Blue-green release
Contact Us

Clusters: test-mia Namespace: default

Name	Tag	PodsQuantity	Image	Time Created	Action
busybox	run:busybox	1/1	busybox	12/26/2018,17:48:46	Details | Edit | Scale | Monitor | More

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



Clusters: test-mia

Namespace: default

Resource Type: Custom

Template:

```
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
```

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:
          serviceName: old-nginx
          servicePort: 80
      # Latest version of the service
      - path: /
        backend:
          serviceName: new-nginx
          servicePort: 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
  annotations:
      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:
          serviceName: old-nginx
          servicePort: 80
      # Latest version of the service
      - path: /
        backend:
          serviceName: new-nginx
          servicePort: 80
```

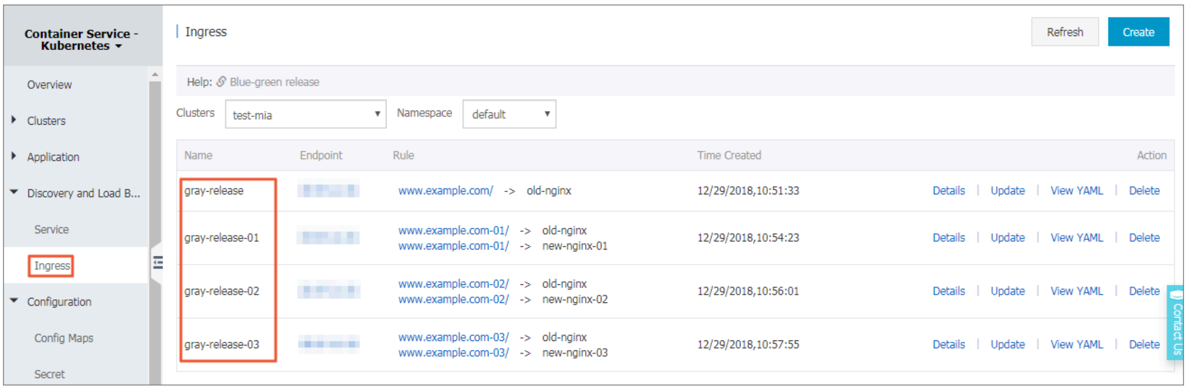
- **Ingress template used to specify that only 50% of the client request traffic that meets the requirements of foo=bar will be routed the latest version of the service**

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: gray-release
  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$/)
    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:
          serviceName: old-nginx
          servicePort: 80
      # Latest version of the service
      - path: /
        backend:
          serviceName: new-nginx
          servicePort: 80
```

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

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



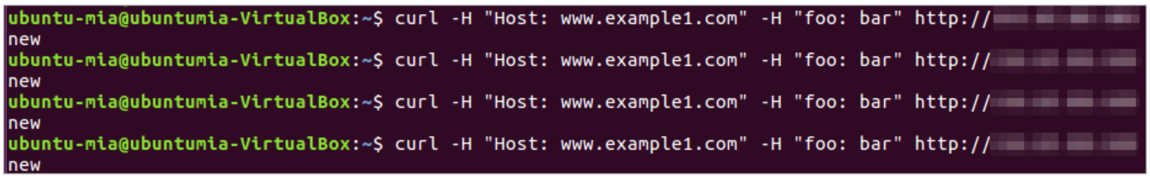
Clusters: test-mia Namespace: default

Name	Endpoint	Rule	Time Created	Action
gray-release	[redacted]	www.example.com/ -> old-nginx	12/29/2018,10:51:33	Details | Update | View YAML | Delete
gray-release-01	[redacted]	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	[redacted]	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	[redacted]	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.

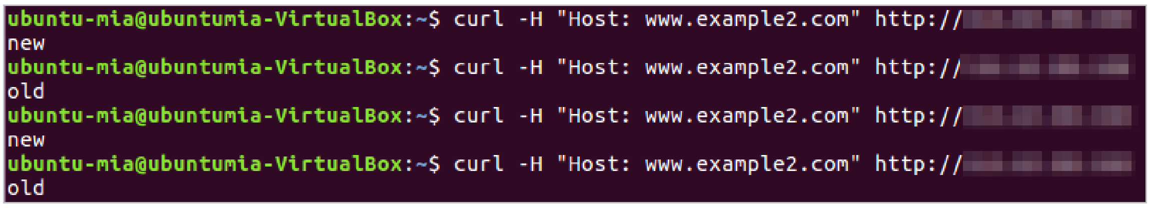
```
# curl -H "Host: www.example.com" -H "foo: bar" http://<EXTERNAL_I
P>
```



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

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

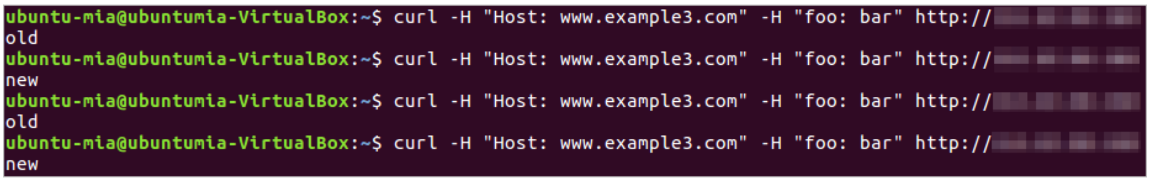
```
# curl -H "Host: www.example.com" http://<EXTERNAL_IP>
```



```
ubuntu-mia@ubuntumia-VirtualBox:~$ curl -H "Host: www.example2.com" http://[redacted]
new
ubuntu-mia@ubuntumia-VirtualBox:~$ curl -H "Host: www.example2.com" http://[redacted]
old
ubuntu-mia@ubuntumia-VirtualBox:~$ curl -H "Host: www.example2.com" http://[redacted]
new
ubuntu-mia@ubuntumia-VirtualBox:~$ curl -H "Host: www.example2.com" http://[redacted]
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@ubuntumia-VirtualBox:~$ curl -H "Host: www.example3.com" -H "foo: bar" http://[redacted]
old
ubuntu-mia@ubuntumia-VirtualBox:~$ curl -H "Host: www.example3.com" -H "foo: bar" http://[redacted]
new
ubuntu-mia@ubuntumia-VirtualBox:~$ curl -H "Host: www.example3.com" -H "foo: bar" http://[redacted]
old
ubuntu-mia@ubuntumia-VirtualBox:~$ curl -H "Host: www.example3.com" -H "foo: bar" http://[redacted]
new
```

4.3.6 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 *Create a Kubernetes cluster*.
- You have connected to the Kubernetes cluster by using kubectl, see *Connect to a Kubernetes cluster by using kubectl*.
- You have deployed an earlier version of the service. For more information, see *Step 1: Deploy a service*. You have also released a later version of the service through a gray release. For more information, see *Step 2: Release the latest version of a service*.

Run a command

1. Run the following command to edit the YAML file deployed by *Step 2: Release the latest version of a service* to remove the earlier version of the service:

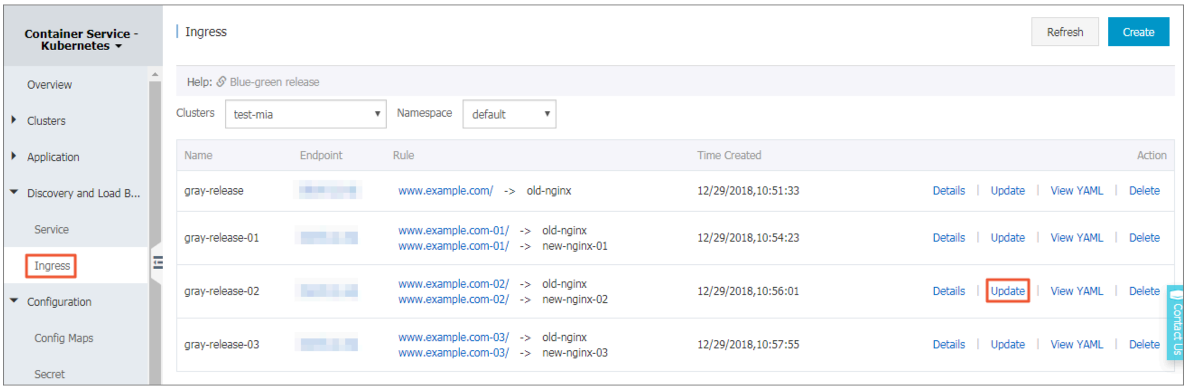
Note:

You need to remove the `annotations` 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.

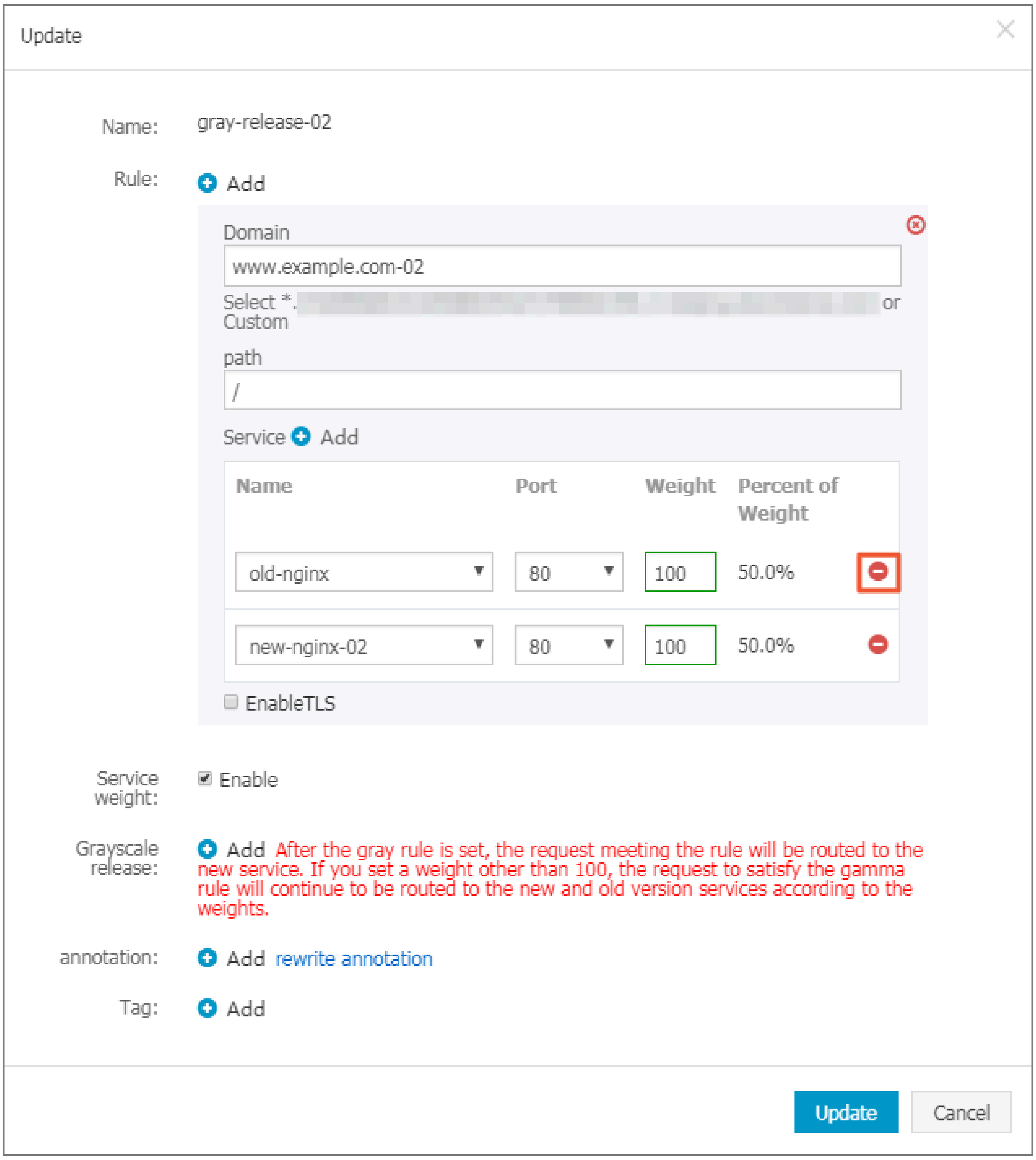


Clusters: test-mia Namespace: default

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

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 *.[redacted] or Custom

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.

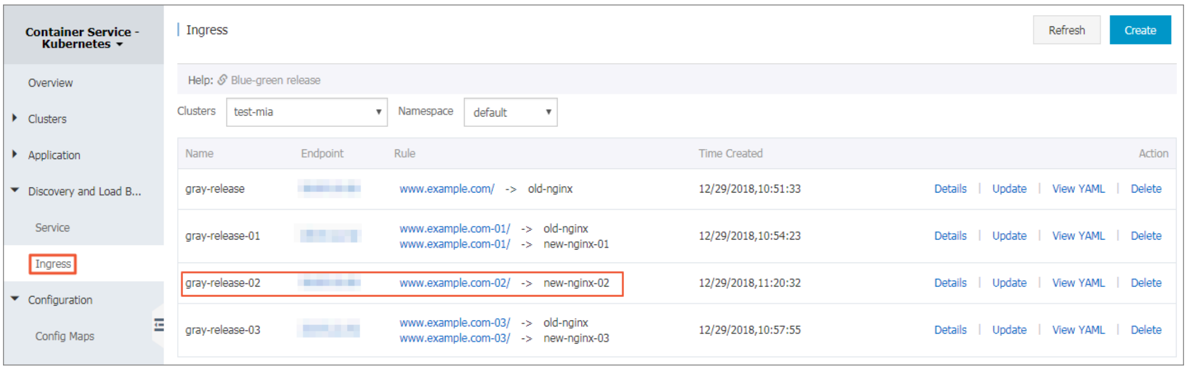
annotation: Add rewrite annotation

Tag: Add

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.

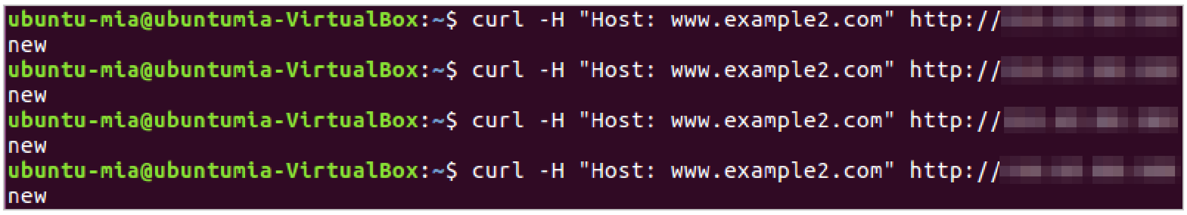


Clusters: test-mia Namespace: default

Name	Endpoint	Rule	Time Created	Action
gray-release	[redacted]	www.example.com/ -> old-nginx	12/29/2018,10:51:33	Details | Update | View YAML | Delete
gray-release-01	[redacted]	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	[redacted]	www.example.com-02/ -> new-nginx-02	12/29/2018,11:20:32	Details | Update | View YAML | Delete
gray-release-03	[redacted]	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_IP>
```



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

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 *Create a Kubernetes cluster*.
- You have deployed Istio. For more information, see *Deploy Istio*.

  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 *Connect to a Kubernetes cluster by using kubectl*.
- 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 vefiry 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.

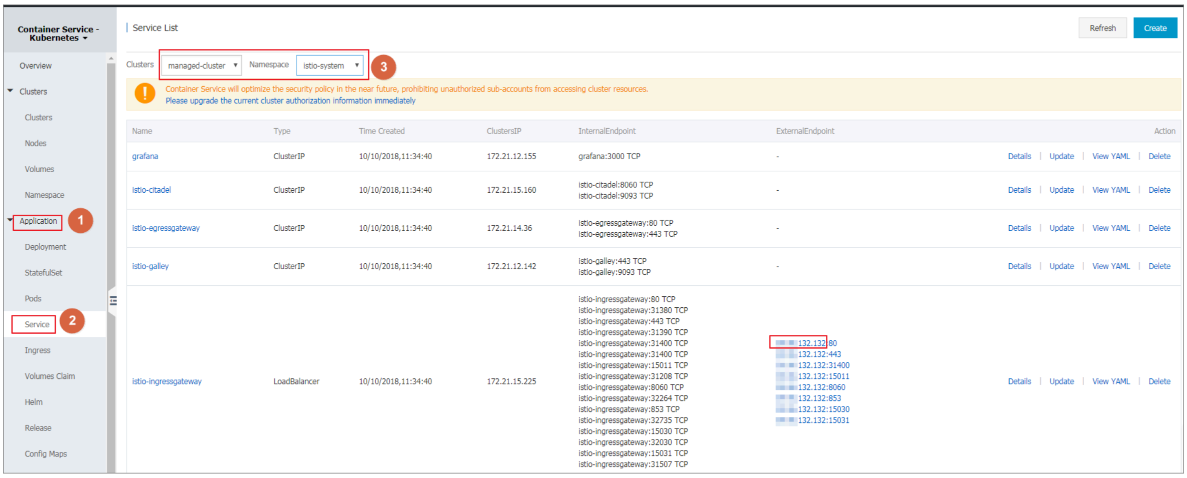
   ```
   $ kubectl apply -f samples/bookinfo/networking/bookinfo-gateway.yaml
   ```

   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.

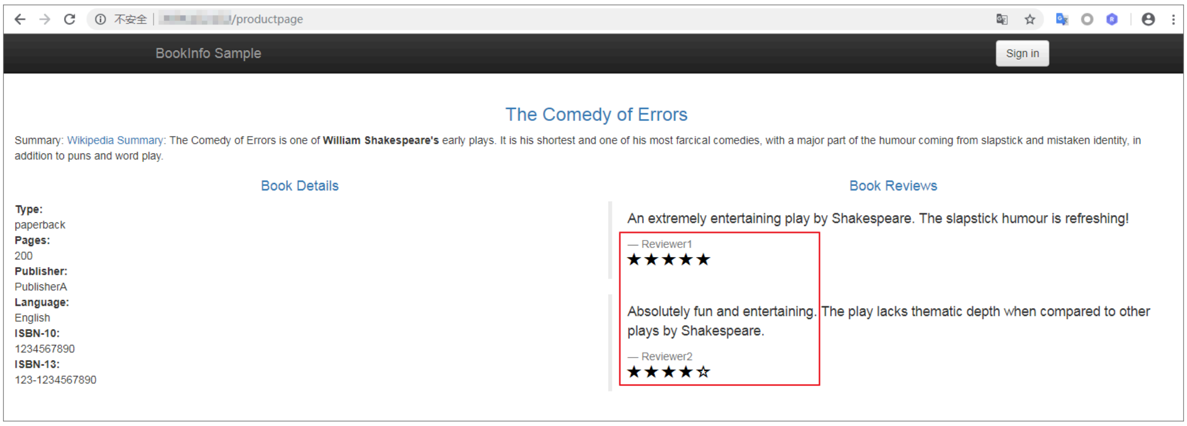


Clusters managed-cluster Namespace istio-system

Container Service will optimize the security policy in the near future, prohibiting unauthorized sub-accounts from accessing cluster resources.
Please upgrade the current cluster authorization information immediately

Name	Type	Time Created	ClustersIP	InternalEndpoint	ExternalEndpoint	Action
grafana	ClusterIP	10/10/2018,11:34:40	172.21.12.155	grafana:3000 TCP	-	Details | Update | View YAML | Delete
istio-citadel	ClusterIP	10/10/2018,11:34:40	172.21.15.160	istio-citadel:8060 TCP istio-citadel:9093 TCP	-	Details | Update | View YAML | Delete
istio-egressgateway	ClusterIP	10/10/2018,11:34:40	172.21.14.36	istio-egressgateway:80 TCP istio-egressgateway:443 TCP	-	Details | Update | View YAML | Delete
istio-galley	ClusterIP	10/10/2018,11:34:40	172.21.12.142	istio-galley:443 TCP istio-galley:9093 TCP	-	Details | Update | View YAML | Delete
istio-ingressgateway	LoadBalancer	10/10/2018,11:34:40	172.21.15.225	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:15011 TCP istio-ingressgateway:31208 TCP istio-ingressgateway:8060 TCP istio-ingressgateway:32264 TCP istio-ingressgateway:853 TCP istio-ingressgateway:32735 TCP istio-ingressgateway:15030 TCP istio-ingressgateway:32030 TCP istio-ingressgateway:15031 TCP istio-ingressgateway:31507 TCP	[redacted]132.132:80 [redacted]132.132:443 [redacted]132.132:31400 [redacted]132.132:15011 [redacted]132.132:8060 [redacted]132.132:853 [redacted]132.132:15030 [redacted]132.132:15031	Details | Update | View YAML | Delete

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



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
★★★★☆

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 destinationrules -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
```

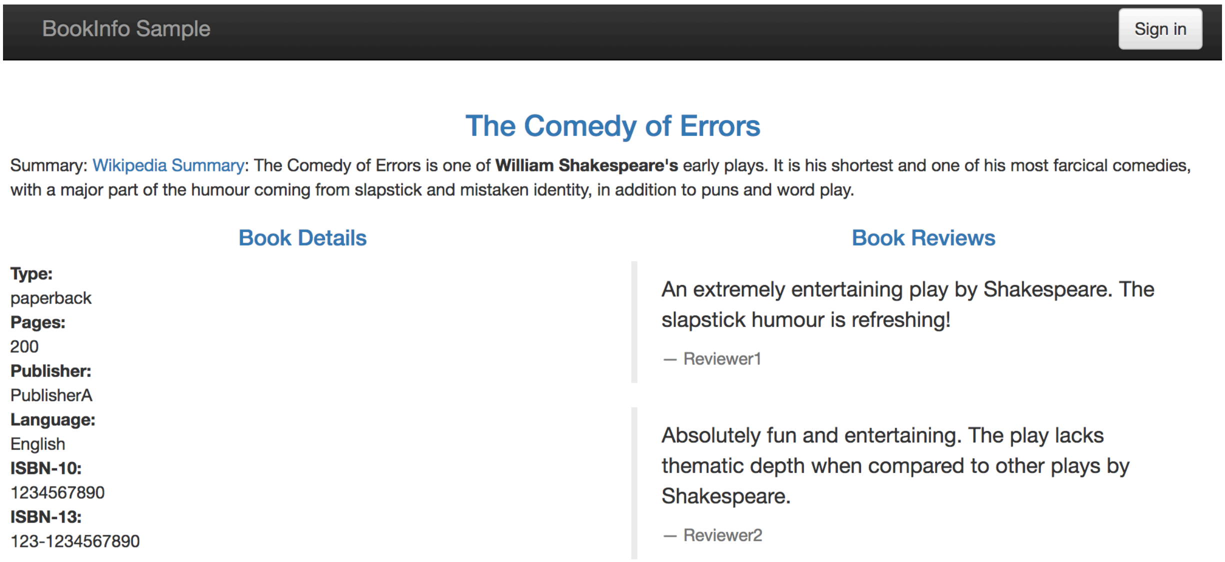
Run the following command to display all the created routing rules:

```
kubectl get virtualservices -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 starts. This is because the reviews:v1 service does not access the ratings service.



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:

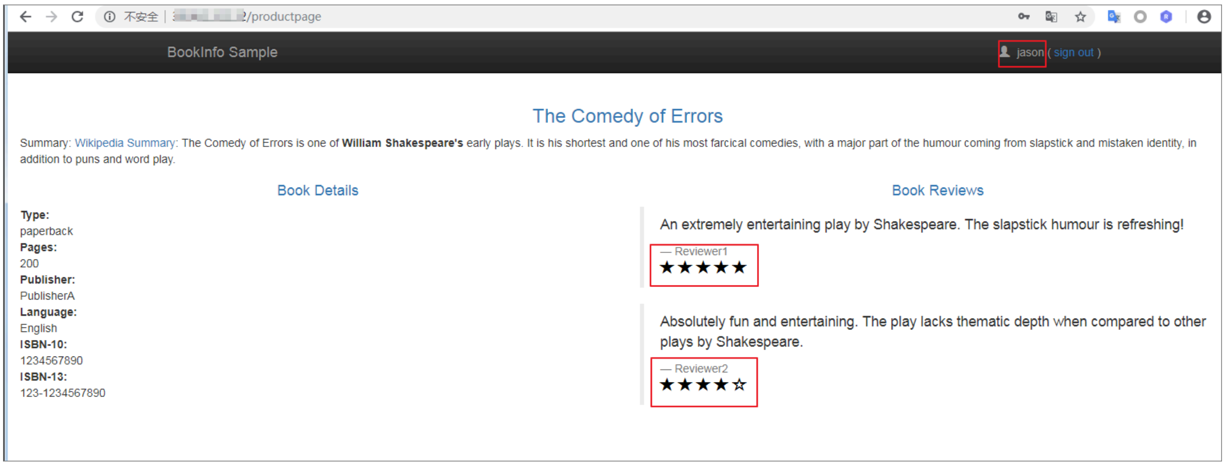
```
$ kubectl get virtualservice reviews -o yaml
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: reviews
  ...
spec:
  hosts:
  - reviews
  http:
  - match:
    - headers:
        end-user:
          exact: jason
    route:
    - destination:
        host: reviews
   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



不安全 | [redacted]/productpage

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
★★★★★

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

— Reviewer2
★★★★☆

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 .

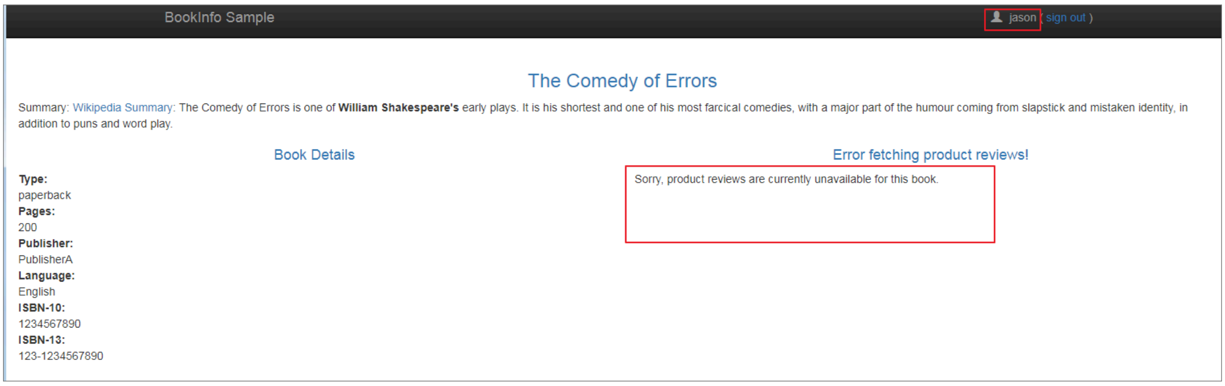
Inject an HTTP delay fault

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.



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

Error fetching product reviews!

Sorry, product reviews are currently unavailable for this book.

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.

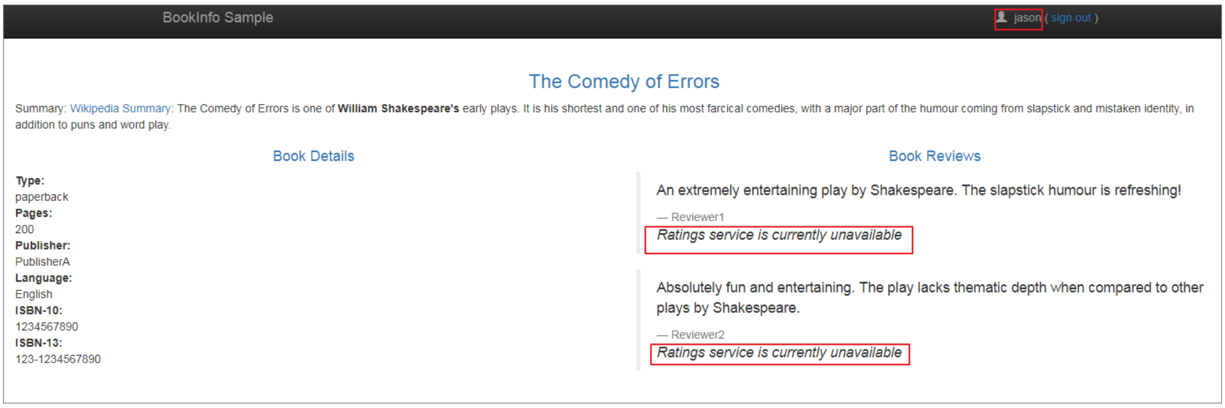
Inject an HTTP abort fault

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.



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 rouging, 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 tracking in Kubernetes

Background

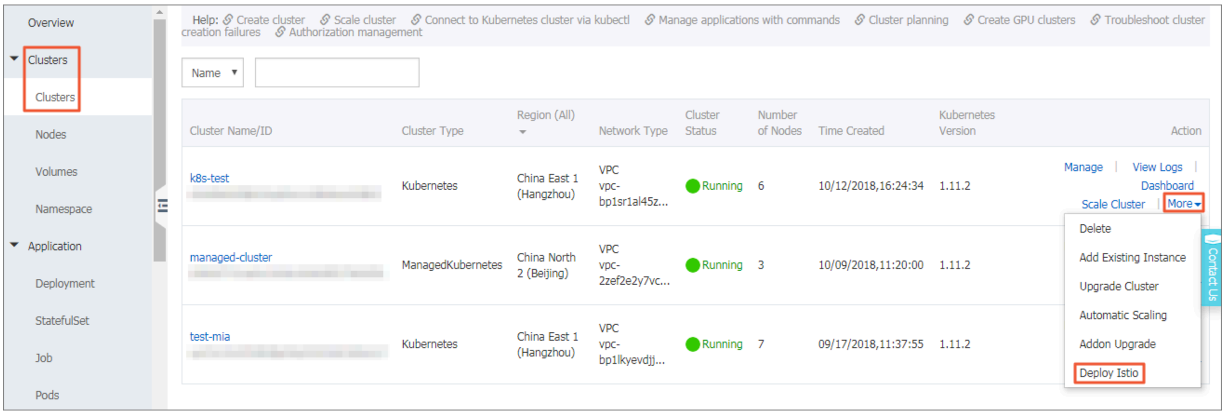
Microservice is a focus in the current era. More and more IT enterprises begin to embrace the microservices. The microservice architecture splits a complex system into several small services and each service can be developed, deployed, and scaled independently. As a heaven-made match, the microservice architecture and containers (Docker and Kubernetes) further simplify the microservice delivery and strengthen the flexibility and robustness of the entire system.

When monolithic applications are transformed to microservices, the distributed application architecture composed of a large number of microservices also increases the complexity of operation & maintenance, debugging, and security management . As microservices grow in scale and complexity, developers must be faced with complex challenges such as service discovery, Server Load Balancer, failure recovery , indicator collection, monitoring, A/B testing, throttling, access control, and end-to-end authentication, which are difficult to resolve.

In May 2017, Google, IBM, and Lyft published the open-source service network architecture Istio, which provides the connection, management, monitoring, and security protection of microservices. Istio provides an infrastructure layer for services to communicate with each other, decouples the issues such as version management, security protection, failover, monitoring, and telemetry in applicatio n logics and service access. Being unrelated to codes, Istio attracts enterprises to transform to microservices, which will make the microservice ecology develop fast.

Architecture principle of Istio

In Kubernetes, a pod is a collection of close-coupled containers, and these containers share the same network namespace. With the extension mechanism of Initializer in Kubernetes, an Envoy container is automatically created and started for each business pod, without modifying the deployment description of the business pod. The Envoy takes over the inbound and outbound traffic of business containers in the same pod. Therefore, the microservice governance functions, including the traffic management, microservice tracking, security authentication, access control, and strategy implementation, are realized by operating on the Envoy.



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

Cluster Name/ID	Cluster Type	Region (All)	Network Type	Cluster Status	Number of Nodes	Time Created	Kubernetes Version	Action
k8s-test [redacted]	Kubernetes	China East 1 (Hangzhou)	VPC vpc-bp1sr1al45z...	● Running	6	10/12/2018,16:24:34	1.11.2	Manage | View Logs | Dashboard Scale Cluster | More
managed-cluster [redacted]	ManagedKubernetes	China North 2 (Beijing)	VPC vpc-2zef2e2y7vc...	● Running	3	10/09/2018,11:20:00	1.11.2	
test-mia [redacted]	Kubernetes	China East 1 (Hangzhou)	VPC vpc-bp1lkyevdjj...	● Running	7	09/17/2018,11:37:55	1.11.2	

An Istio service mesh is logically split into a data plane and a control plane.

- The data plane is composed of a collection of intelligent proxies (Envoys) deployed as sidecars that mediate and control all network communication between microservices.
- The control plane is used to manage and configure the proxies to route traffic, and enforce polices at the runtime.

An Istio is mainly composed of the following components:

- Envoy: The Envoy is used to mediate all the inbound and outbound traffic for all the services in the service mesh. Functions such as dynamic service discovery, Server Load Balancer, fault injection, and traffic management are supported. The Envoy is deployed as a sidecar to the pods of related services.
- Pilot: The Pilot is used to collect and verify the configurations and distribute the configurations to all kinds of Istio components.
- Mixer: The Mixer is used to enforce the access control and usage policies in the service mesh, and collect telemetry data from Envoy proxies and other services.
- Istio-Auth: Istio-Auth provides strong service-to-service and end user authentication.

For more information about Istio, see the *Istio official document*.

Install Istio

Use an Alibaba Cloud Container Service Kubernetes cluster as an example.

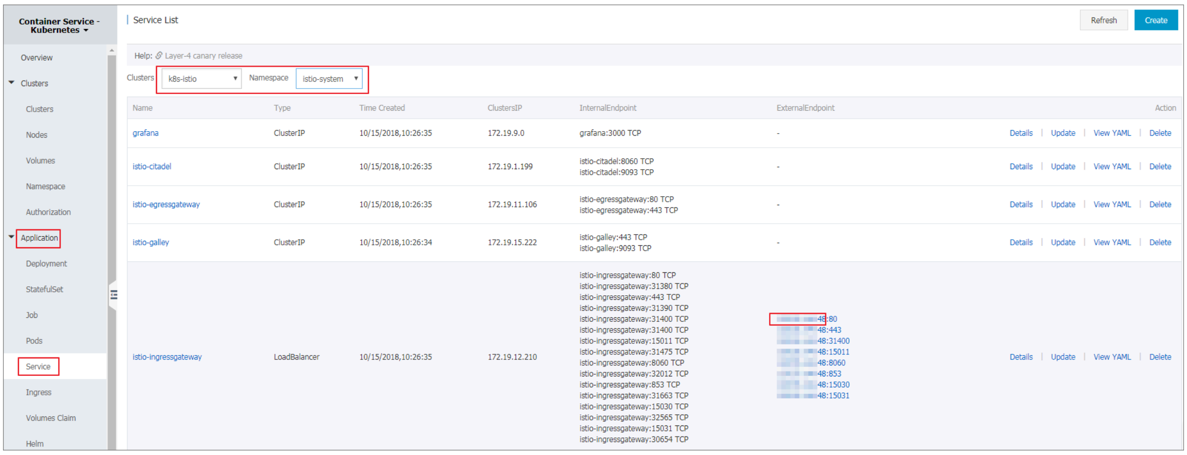
Alibaba Cloud Container Service has enabled the Initializers plug-in by default for Kubernetes clusters if the cluster version is later than 1.8. No other configurations are needed.

Note:

After you deploy the Istio, a sidecar is injected to each pod to take over the service communication. Therefore, we recommend that you verify this in the independent test environment.

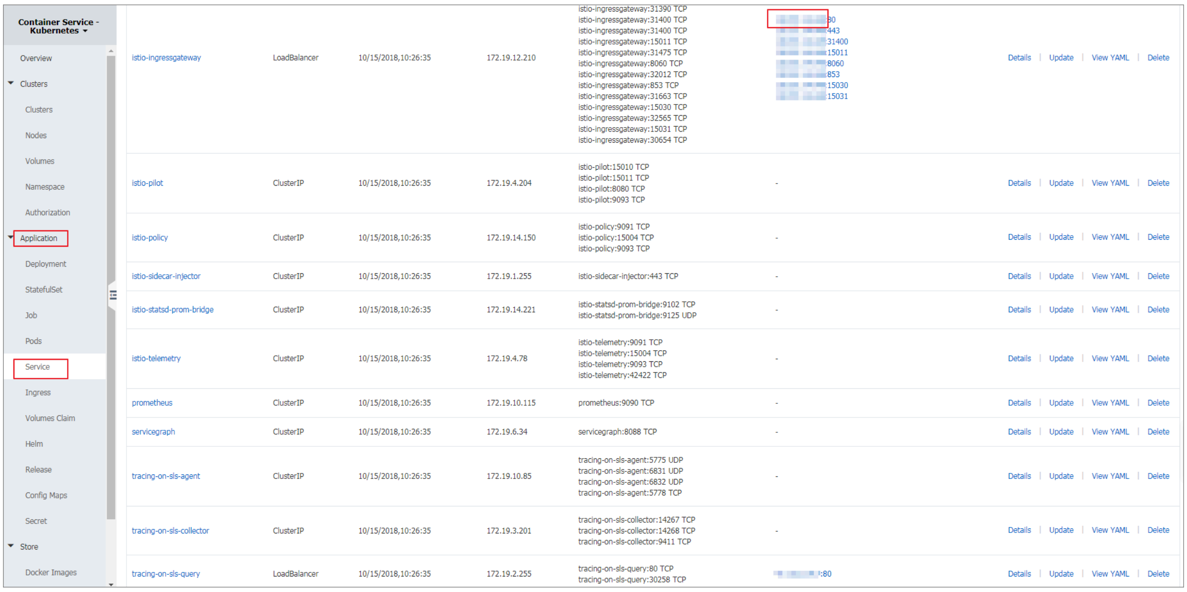
Create a Kubernetes cluster

1. Log on to the *Container Service console*.
2. Under Kubernetes, click Clusters in the left-side navigation pane, and click Create Kubernetes cluster in the upper-right corner.
3. Configure the parameters to create a cluster. For how to create a Kubernetes cluster, see *Create a Kubernetes cluster*.
4. After the cluster is created, click Manage at the right of the cluster when the cluster status is changed to Running.



Name	Type	Time Created	ClustersIP	InternalEndpoint	ExternalEndpoint	Action
grafana	ClusterIP	10/15/2018,10:26:35	172.19.9.0	grafana:3000 TCP	-	Details | Update | View YAML | Delete
istio-citadel	ClusterIP	10/15/2018,10:26:35	172.19.1.199	istio-citadel:8060 TCP istio-citadel:9093 TCP	-	Details | Update | View YAML | Delete
istio-egressgateway	ClusterIP	10/15/2018,10:26:35	172.19.11.106	istio-egressgateway:80 TCP istio-egressgateway:443 TCP	-	Details | Update | View YAML | Delete
istio-galley	ClusterIP	10/15/2018,10:26:34	172.19.15.222	istio-galley:443 TCP istio-galley:9093 TCP	-	Details | Update | View YAML | Delete
istio-ingressgateway	LoadBalancer	10/15/2018,10:26:35	172.19.12.210	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: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	[redacted]48:80 [redacted]48:443 [redacted]48:31400 [redacted]48:15011 [redacted]48:8060 [redacted]48:853 [redacted]48:15030 [redacted]48:15031	Details | Update | View YAML | Delete

5. On the cluster Basic Information page, you can configure the corresponding connection information based on the page information. You can connect to the cluster either by using *Connect to a Kubernetes cluster by using kubectl* or *Access Kubernetes clusters by using SSH*.



istio-ingressgateway	LoadBalancer	10/15/2018,10:26:35	172.19.12.210	istio-ingressgateway:31390 TCP 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	[redacted]:80 [redacted]:443 [redacted]:31400 [redacted]:15011 [redacted]:8060 [redacted]:853 [redacted]:15030 [redacted]:15031	Details | Update | View YAML | Delete
istio-pilot	ClusterIP	10/15/2018,10:26:35	172.19.4.204	istio-pilot:15010 TCP istio-pilot:15011 TCP istio-pilot:8080 TCP istio-pilot:9093 TCP	-	Details | Update | View YAML | Delete
istio-policy	ClusterIP	10/15/2018,10:26:35	172.19.14.150	istio-policy:9091 TCP istio-policy:15004 TCP istio-policy:9093 TCP	-	Details | Update | View YAML | Delete
istio-sidecar-injector	ClusterIP	10/15/2018,10:26:35	172.19.1.255	istio-sidecar-injector:443 TCP	-	Details | Update | View YAML | Delete
istio-statsd-prom-bridge	ClusterIP	10/15/2018,10:26:35	172.19.14.221	istio-statsd-prom-bridge:9102 TCP istio-statsd-prom-bridge:9125 UDP	-	Details | Update | View YAML | Delete
istio-telemetry	ClusterIP	10/15/2018,10:26:35	172.19.4.78	istio-telemetry:9091 TCP istio-telemetry:15004 TCP istio-telemetry:9093 TCP istio-telemetry:42422 TCP	-	Details | Update | View YAML | Delete
prometheus	ClusterIP	10/15/2018,10:26:35	172.19.10.115	prometheus:9090 TCP	-	Details | Update | View YAML | Delete
servicegraph	ClusterIP	10/15/2018,10:26:35	172.19.6.34	servicegraph:8088 TCP	-	Details | Update | View YAML | Delete
tracing-on-sls-agent	ClusterIP	10/15/2018,10:26:35	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 YAML | Delete
tracing-on-sls-collector	ClusterIP	10/15/2018,10:26:35	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 YAML | Delete
tracing-on-sls-query	LoadBalancer	10/15/2018,10:26:35	172.19.2.255	tracing-on-sls-query:80 TCP tracing-on-sls-query:30258 TCP	[redacted]:80	Details | Update | View YAML | Delete

Deploy Istio release version

Log on to the master node and run the following command to get the latest Istio installation package.

```
curl -L https://git.io/getLatestIstio | sh -
```

Run the following command:

```
cd istio-0.4.0                              ##Change the working directory
 to Istio
export PATH=$PWD/bin:$PATH                  ##Add the istioctl client to
PATH environment variable
```

Run the following command to deploy Istio.

```
kubectl apply -f install/kubernetes/istio.yaml             ## Deploy
Istio system components
kubectl apply -f install/kubernetes/istio-initializer.yaml     ##
Deploy Istio initializer plug-in
```

After the deployment, run the following command to verify if the Istio components are successfully deployed.

```
$ kubectl get svc,pod -n istio-systemNAME TYPE CLUSTER-IP EXTERNAL-
IP PORT(S) AGEsvc/istio-ingress LoadBalancer 172.21.10.18 101.37.113
.231 80:30511/TCP,443:31945/TCP 1msvc/istio-mixer ClusterIP 172.21.
14.221  9091/TCP,15004/TCP,9093/TCP,9094/TCP,9102/TCP,9125/UDP,42422/
TCP 1msvc/istio-pilot ClusterIP 172.21.4.20  15003/TCP,443/TCP 1mNAME
 READY STATUS RESTARTS AGEpo/istio-ca-55b954ff7-crsjq 1/1 Running 0
1mpo/istio-ingress-948b746cb-4t24c 1/1 Running 0 1mpo/istio-initialize
r-6c84859cd-8mvfj 1/1 Running 0 1mpo/istio-mixer-59cc756b48-tkx6c 3/3
Running 0 1mpo/istio-pilot-55bb7f5d9d-wc5xh 2/2 Running 0 1m
```

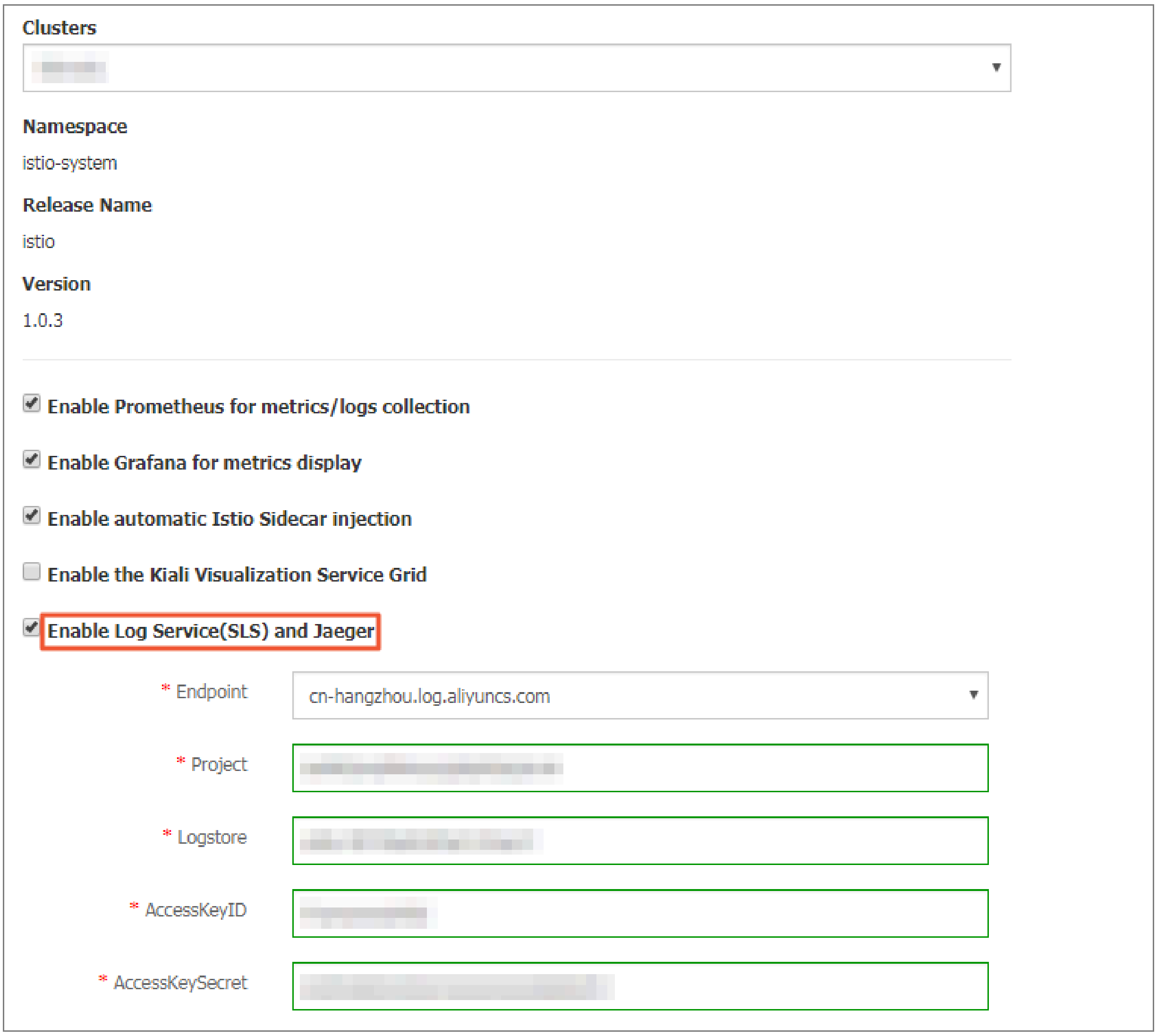
After all the pods are in the running status, the Istio deployment is finished.

Istio distributed service tracking case

Deploy and test the application BookInfo

BookInfo is an application similar to an online bookstore, which is composed of several independent microservices compiled by different languages. The application BookInfo is deployed in the container mode and does not have any dependencies on Istio. All the microservices are packaged together with an Envoy sidecar. The Envoy sidecar intercepts the inbound and outbound call requests of services to demonstrate the distributed tracking function of Istio service mesh.

For more information about BookInfo, see *Bookinfo guide*.



Clusters

[redacted] ▾

Namespace

istio-system

Release Name

istio

Version

1.0.3

☑ Enable Prometheus for metrics/logs collection

☑ Enable Grafana for metrics display

☑ Enable automatic Istio Sidecar injection

☐ Enable the Kiali Visualization Service Grid

☑ Enable Log Service(SLS) and Jaeger

* Endpoint: cn-hangzhou.log.aliyuncs.com ▾

* Project: [redacted]

* Logstore: [redacted]

* AccessKeyID: [redacted]

* AccessKeySecret: [redacted]

Run the following command to deploy and test the application Bookinfo.

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

In the Alibaba Cloud Kubernetes cluster environment, every cluster has been configured with the Server Load Balancer and Ingress. Run the following command to obtain the IP address of Ingress.

```
$ kubectl get ingress -o wide
NAME       HOSTS      ADDRESS          PORTS      AGE
gateway    *          101.37.xxx.xxx   80         2m
```

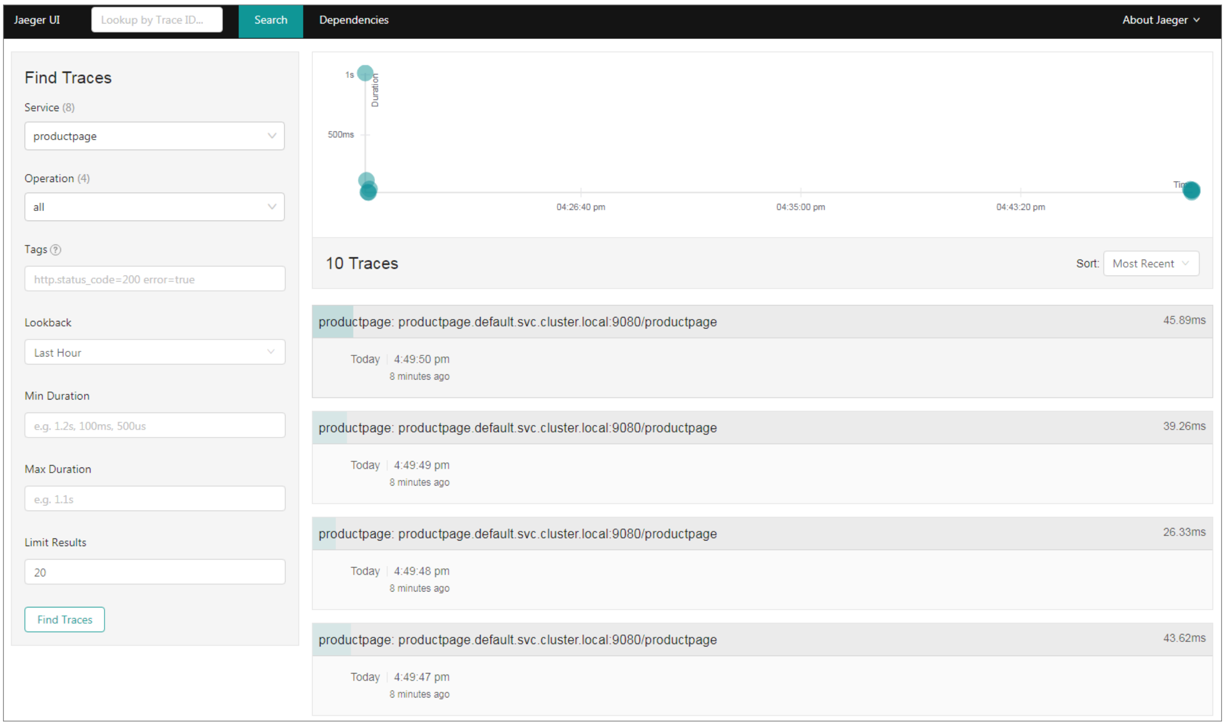
If the preceding command cannot obtain the external IP address, run the following command to obtain the corresponding address.

```
export GATEWAY_URL=$(kubectl get ingress -o wide -o jsonpath={.items[0
].status.loadBalancer.ingress[0].ip})
```

The application is successfully deployed if the following command returns 200.

```
curl -o /dev/null -s -w "%{http_code}\n" http://${GATEWAY_URL}/
productpage
```

You can open `http://${GATEWAY_URL}/productpage` in the browser to access the application. GATEWAY_URL is the IP address of Ingress.



Find Traces

Service (8): productpage

Operation (4): all

Tags: http.status_code=200 error=true

Lookback: Last Hour

Min Duration: e.g. 1.2s, 100ms, 500us

Max Duration: e.g. 1.1s

Limit Results: 20

10 Traces

Sort: Most Recent

productpage: productpage.default.svc.cluster.local:9080/productpage 45.89ms
Today | 4:49:50 pm
8 minutes ago

productpage: productpage.default.svc.cluster.local:9080/productpage 39.26ms
Today | 4:49:49 pm
8 minutes ago

productpage: productpage.default.svc.cluster.local:9080/productpage 26.33ms
Today | 4:49:48 pm
8 minutes ago

productpage: productpage.default.svc.cluster.local:9080/productpage 43.62ms
Today | 4:49:47 pm
8 minutes ago

Deploy Jaeger tracking system

Distributed tracking system helps you observe the call chains between services and is useful when diagnosing performance issues and analyzing system failures.

Istio ecology supports different distributed tracking systems, including *Zipkin* and *Jaeger*. Use the Jaeger as an example.

Istio version 0.4 supports Jaeger. The test method is as follows.

```
kubectl apply -n istio-system -f https://raw.githubusercontent.com/
jaegertracing/jaeger-kubernetes/master/all-in-one/jaeger-all-in-one-
template.yml
```

After the deployment is finished, if you connect to the Kubernetes cluster by using kubectl, run the following command to access the Jaeger control panel by using port mapping and open`http://localhost:16686` in the browser.

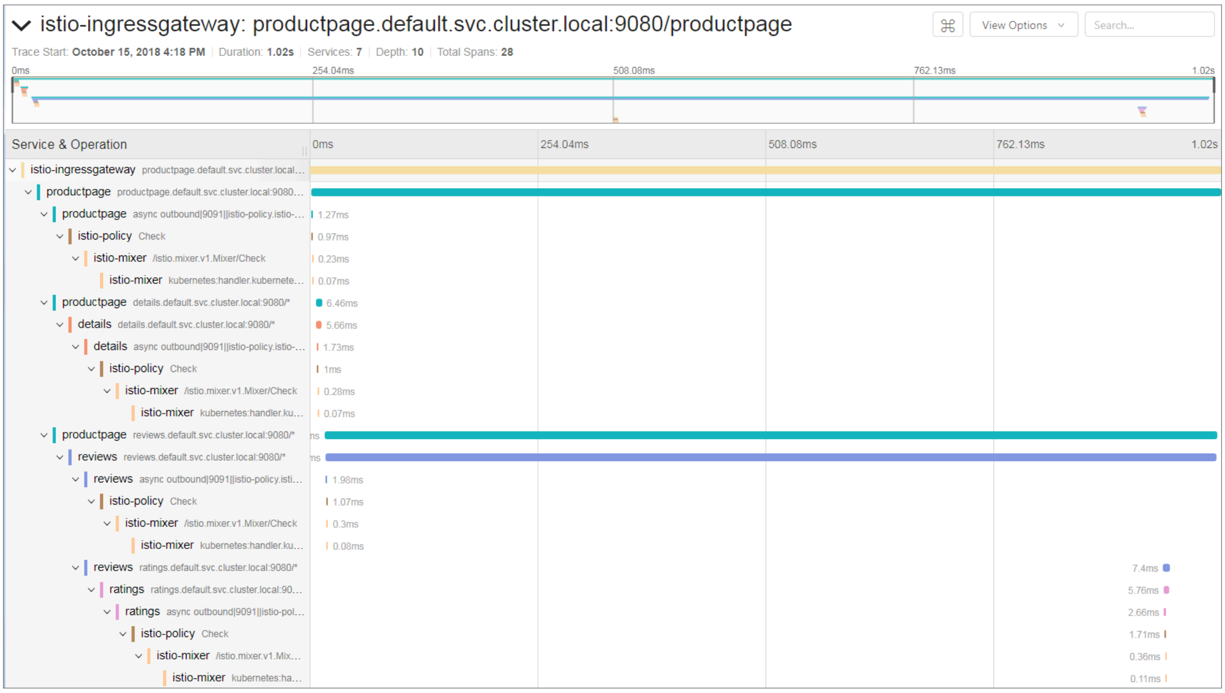
```
kubectl port-forward -n istio-system $(kubectl get pod -n istio-system
 -l app=jaeger -o jsonpath='{.items[0].metadata.name}') 16686:16686 &
```

If you connect to the Alibaba Cloud Kubernetes cluster by using SSH, run the following command to check the external access address of jaeger-query service.

```
$ kubectl get svc -n istio-system
NAME                TYPE           CLUSTER-IP      EXTERNAL-IP
PORT(S)                                                             AGE
jaeger-agent        ClusterIP      None            <none>
5775/UDP,6831/UDP,6832/UDP                                          1h
jaeger-collector    ClusterIP      172.21.10.187   <none>
14267/TCP,14268/TCP,9411/TCP                                        1h
jaeger-query        LoadBalancer   172.21.10.197   114.55.82.11     80:
31960/TCP     ##The external access address is 114.55.82.11:80.
zipkin              ClusterIP      None            <none>
9411/TCP
```

Record the external access IP address and port of jaeger-query and then open the application in the browser.

By accessing the application BookInfo for multiple times and generating the call chain information, we can view the call chain information of services clearly.

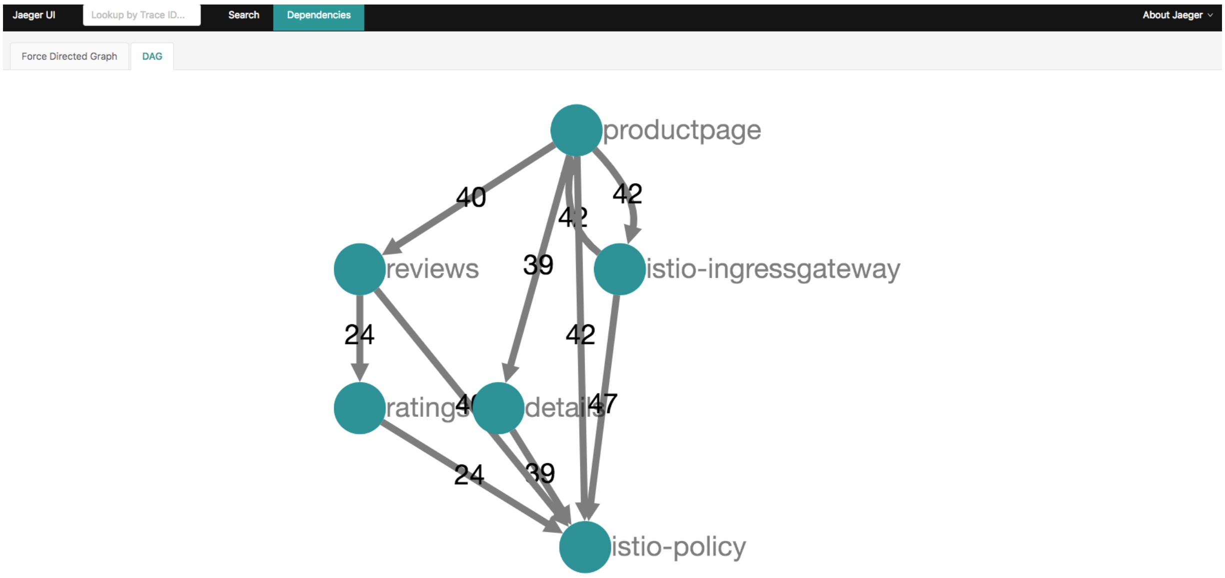


istio-ingressgateway: productpage.default.svc.cluster.local:9080/productpage

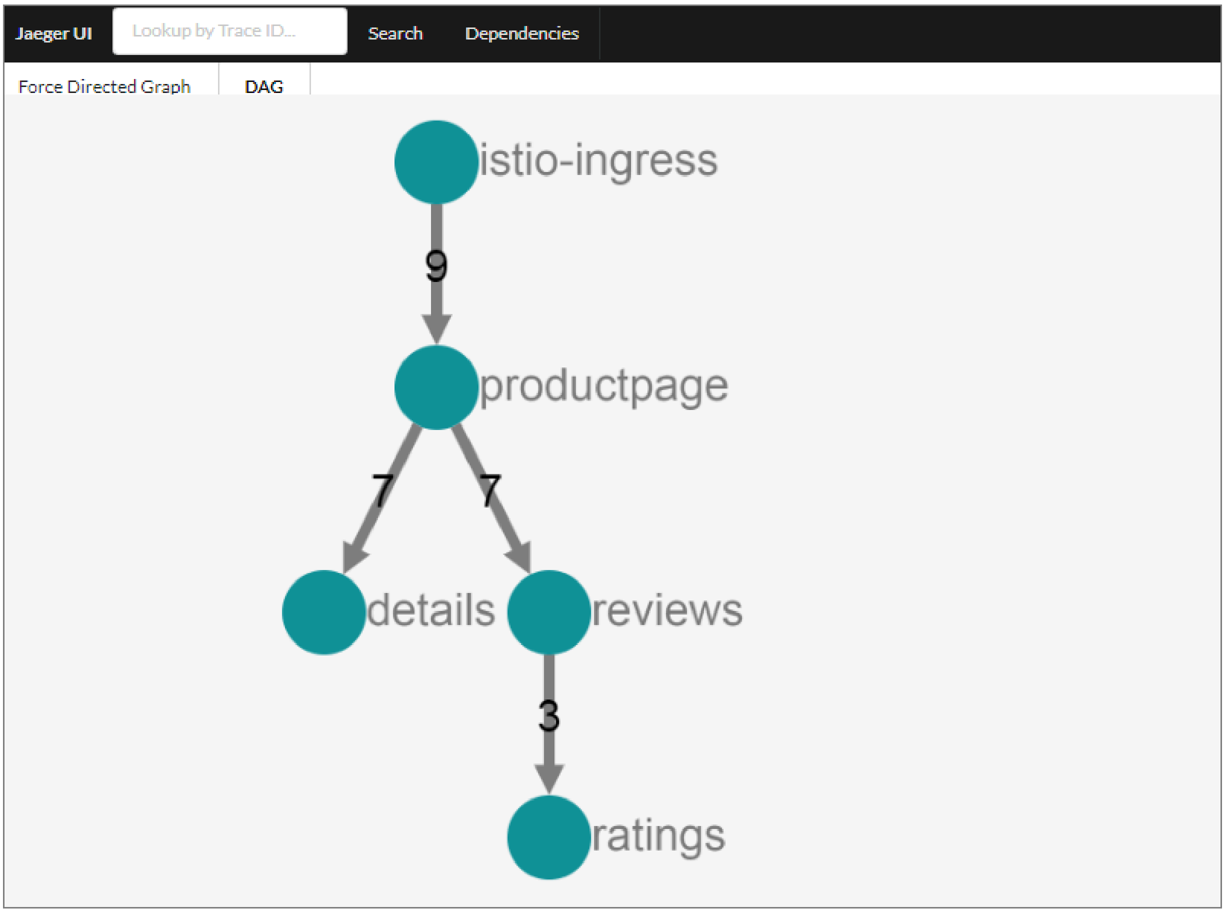
Trace Start: **October 15, 2018 4:18 PM** | Duration: **1.02s** | Services: **7** | Depth: **10** | Total Spans: **28**

```
Service & Operation                                                   0ms   254.04ms   508.08ms   762.13ms   1.02s
istio-ingressgateway  productpage.default.svc.cluster.local...
  productpage  productpage.default.svc.cluster.local:9080...
    productpage  async outbound|9091||istio-policy.istio-...          1.27ms
      istio-policy  Check                                             0.97ms
        istio-mixer  /istio.mixer.v1.Mixer/Check                      0.23ms
          istio-mixer  kubernetes:handler.kubernete...                0.07ms
    productpage  details.default.svc.cluster.local:9080/*             6.46ms
      details  details.default.svc.cluster.local:9080/*               5.66ms
        details  async outbound|9091||istio-policy.istio-...          1.73ms
          istio-policy  Check                                         1ms
            istio-mixer  /istio.mixer.v1.Mixer/Check                  0.28ms
              istio-mixer  kubernetes:handler.ku...                   0.07ms
    productpage  reviews.default.svc.cluster.local:9080/*             ns
      reviews  reviews.default.svc.cluster.local:9080/*               ns
        reviews  async outbound|9091||istio-policy.isti...            1.98ms
          istio-policy  Check                                         1.07ms
            istio-mixer  /istio.mixer.v1.Mixer/Check                  0.3ms
              istio-mixer  kubernetes:handler.ku...                   0.08ms
        reviews  ratings.default.svc.cluster.local:9080/*             7.4ms
          ratings  ratings.default.svc.cluster.local:90...            5.76ms
            ratings  async outbound|9091||istio-pol...                2.66ms
              istio-policy  Check                                     1.71ms
                istio-mixer  /istio.mixer.v1.Mix...                   0.36ms
                  istio-mixer  kubernetes:ha...                       0.11ms
```

Click a specific Trace to view the details.



You can also view DAG.



Implementation principle of Istio distributed tracking

The kernel of Istio service mesh is the Envoy, which is a high-performance and open-source Layer-7 proxy and communication bus. In Istio, each microservice is injected with an Envoy sidecar and this instance is responsible for processing all the inbound and outbound network traffic. Therefore, each Envoy sidecar can monitor all the API calls between services, record the time required by each service call, and record whether each service call is successful or not.

Whenever a microservice initiates an external call, the client Envoy will create a new span. A span represents the complete interaction process between a collection of microservices, starting from a caller (client) sending a request to receiving the response from the server.

In the service interaction process, clients record the request start time and response receipt time, and the Envoy on the server records the request receipt time and response return time.

Each Envoy distributes their own span view information to the distributed tracking system. When a microservice processes requests, other microservices may need to be called, which causes the creation of a causally related span and then forms the complete trace. Then, an application must be used to collect and forward the following Headers from the request message:

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

Envoys in the communication links can intercept, process, and forward the corresponding Headers.

```
Client Tracer                                           Server
Tracer
┌───────────────────┐
│ ┌───────────────────┐                                          │
│          │         │
│    TraceContext    │          Http Request Headers             │
TraceContext      │
│  ┌───────────────┐ │          ┌───────────────────┐            │
│ ┌───────────────┐ │ │
│ │ TraceId       │ │          │ X─B3─TraceId      │            │ │ TraceId
│ │       │ │
│ │               │ │          │                   │            │ │
│ │       │ │
│ │ ParentSpanId  │ │ Extract  │ X─B3─ParentSpanId │ Inject     │ │
ParentSpanId │ │
│ │               ├─┼─────────>│                   ├──────────┼>│
│ │       │ │
│ │ SpanId        │ │          │ X─B3─SpanId       │            │ │ SpanId
│ │       │ │
│ │               │ │          │                   │            │ │
│ │       │ │
│ │ Sampled       │ │          │ X─B3─Sampled      │            │ │ Sampled
│ │       │ │
│ └───────────────┘ │          └───────────────────┘            │
│ └───────────────┘ │ │
│                   │                                            │
│          │
└───────────────────┘
 └───────────────────┘
```

For specific codes, see the Istio document *https://istio.io/docs/tasks/telemetry/distributed-tracing.html*.

Conclusion

Istio is accelerating the application and popularization of service mesh by using the good expansion mechanism and strong ecology. In addition to those mentioned in the preceding sections, Weave Scope, Istio Dashboard, and Istio-Analytics projects provide abundant call link visualization and analysis capabilities.

5.3 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 *Create a Kubernetes cluster*.

Note:
You must make sure that you can connect to your Kubernetes cluster by using kubectl. For more information, see *Connect to a Kubernetes cluster by using kubectl*.

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.

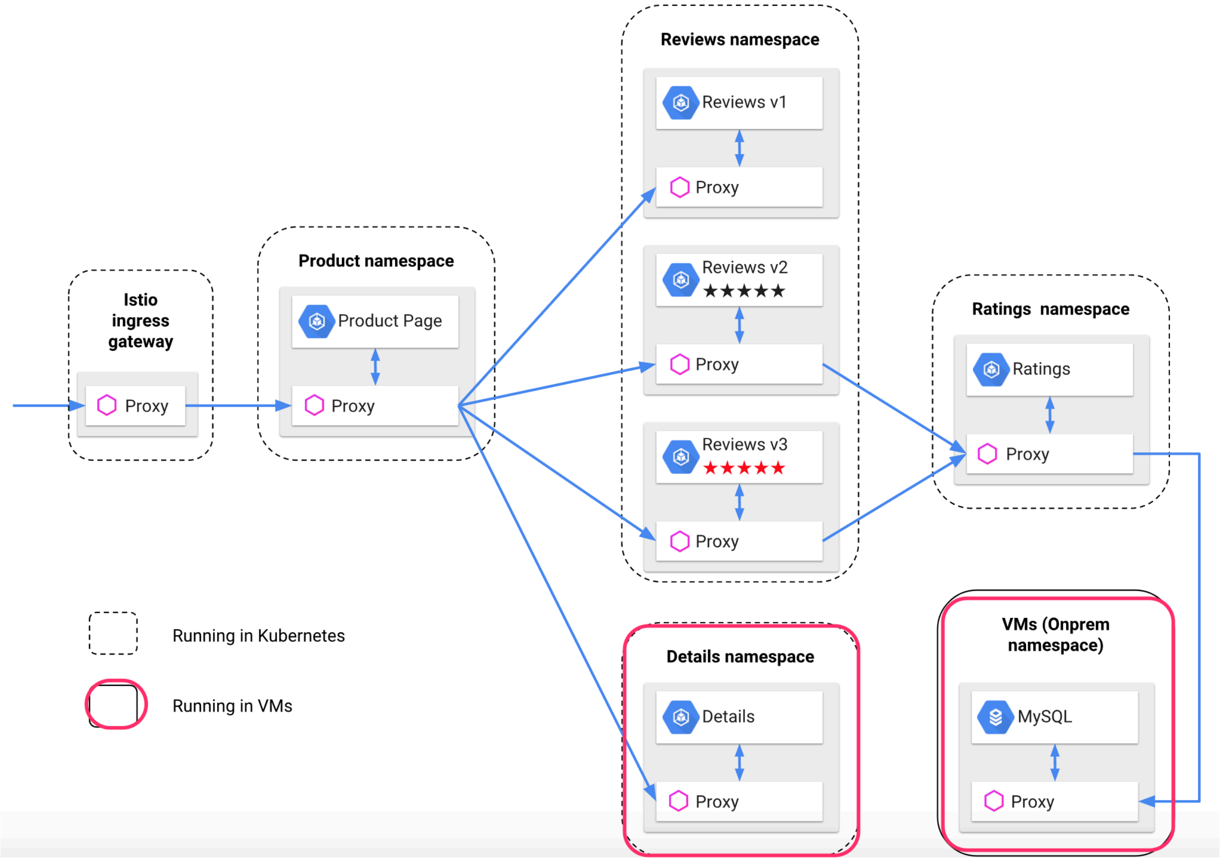
Install the sample application in your Kubernetes cluster

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.

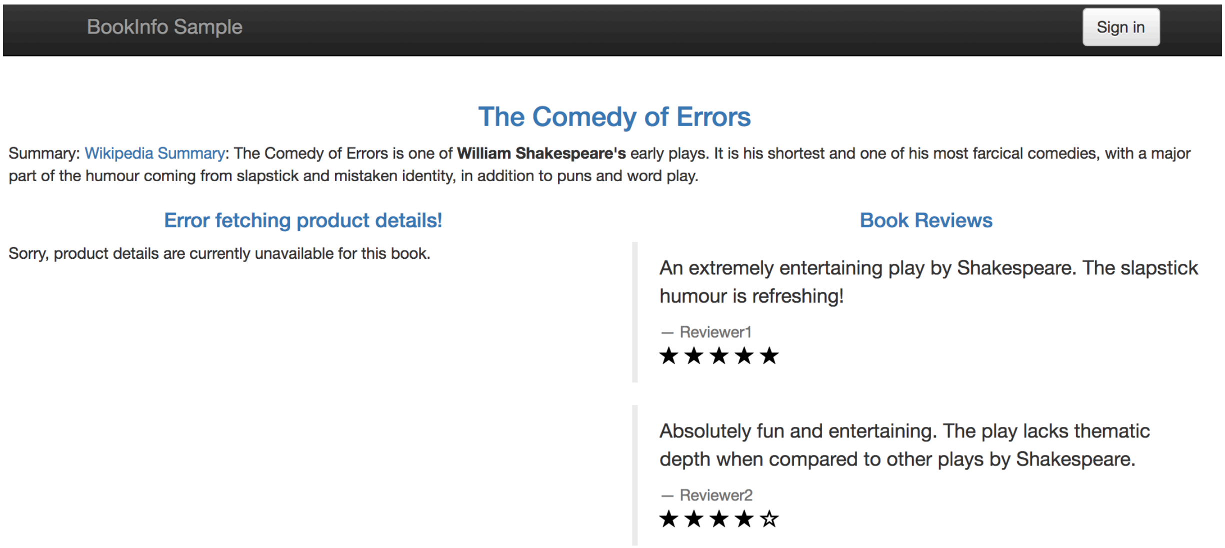
To obtain the files used in this example, see *Istio multi-cluster sample files*.

```
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

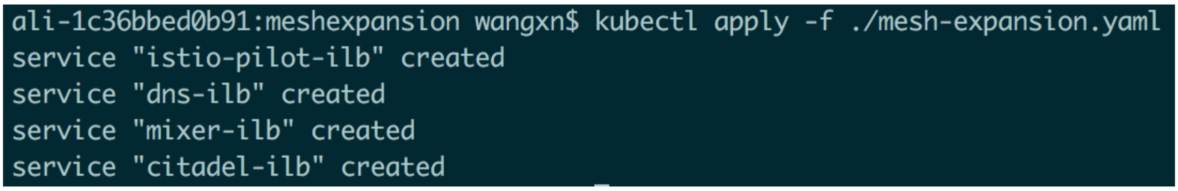
★★★★☆

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.

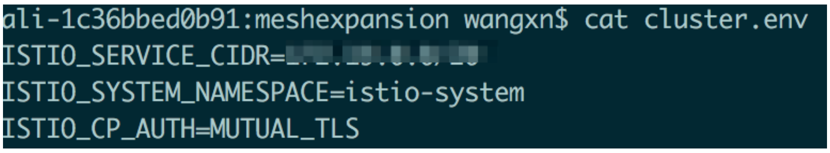
   ```
   ali-1c36bbed0b91:meshexpansion wangxn$ kubectl apply -f ./mesh-expansion.yaml
   service "istio-pilot-ilb" created
   service "dns-ilb" created
   service "mixer-ilb" created
   service "citadel-ilb" created
   ```

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:

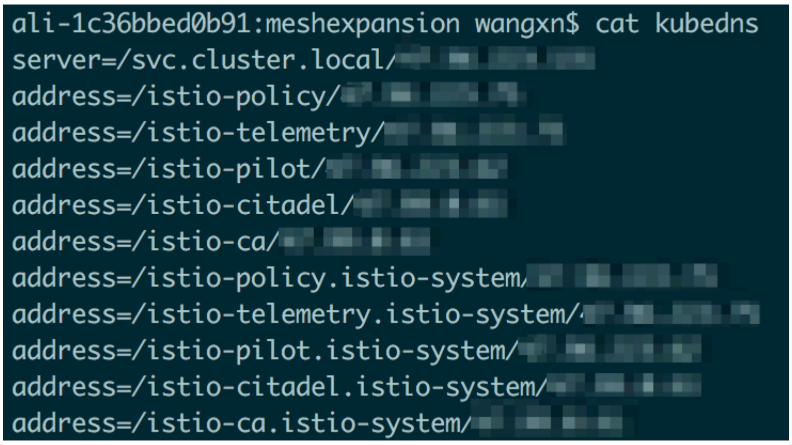
   ```
   ./setupMeshEx.sh generateClusterEnvAndDnsmasq
   ```

   Configuration file `cluster.env`

   ```
   ali-1c36bbed0b91:meshexpansion wangxn$ cat cluster.env
   ISTIO_SERVICE_CIDR=[redacted]
   ISTIO_SYSTEM_NAMESPACE=istio-system
   ISTIO_CP_AUTH=MUTUAL_TLS
   ```

   Configuration file `kubedns`

   ```
   ali-1c36bbed0b91:meshexpansion wangxn$ cat kubedns
   server=/svc.cluster.local/[redacted]
   address=/istio-policy/[redacted]
   address=/istio-telemetry/[redacted]
   address=/istio-pilot/[redacted]
   address=/istio-citadel/[redacted]
   address=/istio-ca/[redacted]
   address=/istio-policy.istio-system/[redacted]
   address=/istio-telemetry.istio-system/[redacted]
   address=/istio-pilot.istio-system/[redacted]
   address=/istio-citadel.istio-system/[redacted]
   address=/istio-ca.istio-system/[redacted]
   ```

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_IP>` 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 ""
```

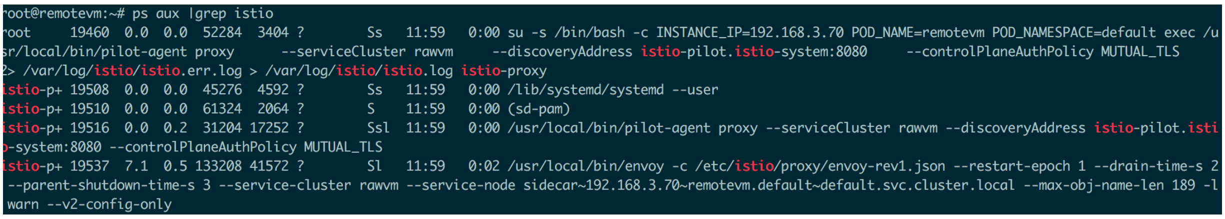
Note:
To ensure that the ECS instance and Kubernetes are mutually accessible over the Internet, you need to add them to the same security group.

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

```
export SERVICE_NAMESPACE=default
./setupMeshEx.sh machineSetup root@<ECS_HOST_IP>
```

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

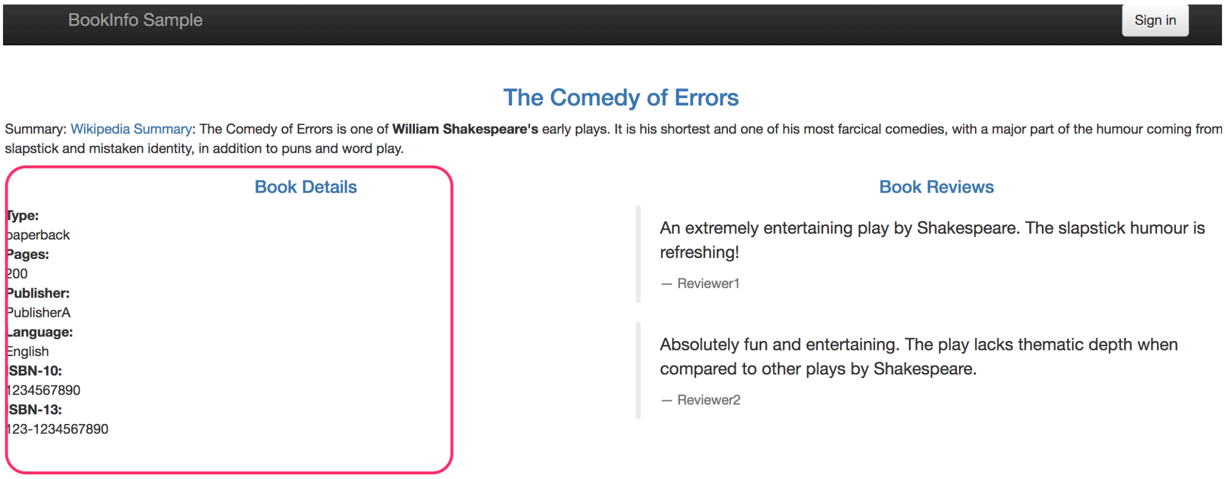
Run the following command to view the IP address of the VM so that you can add it to the service mesh:

```
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.



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

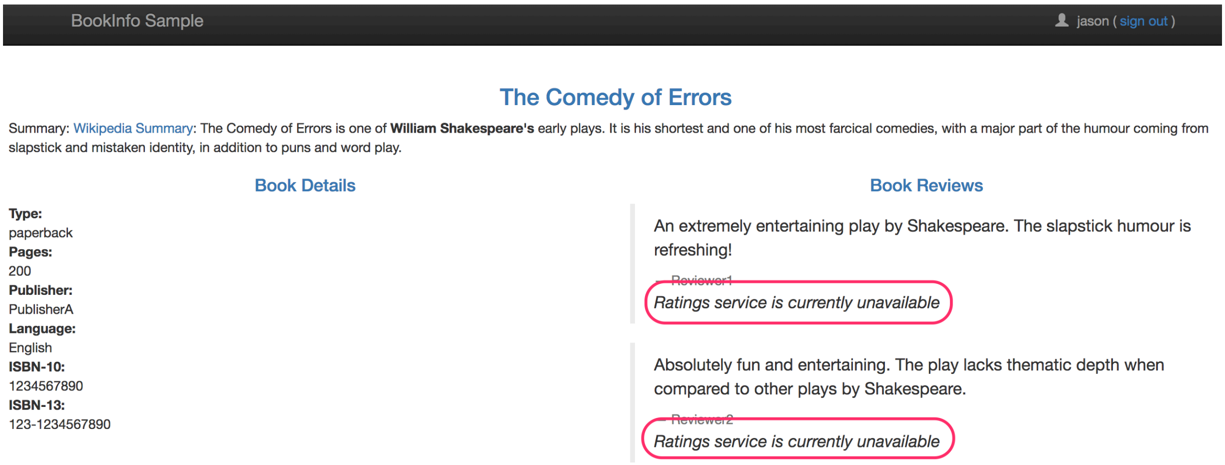
Absolutely fun and entertaining. The play lacks thematic depth when compared to other plays by Shakespeare.
— Reviewer2

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.



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 PRIVILEGESON *. * TO'root'@'localhost'IDENTIFIEDBY'password'
WITHGRANTOPTION;
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:

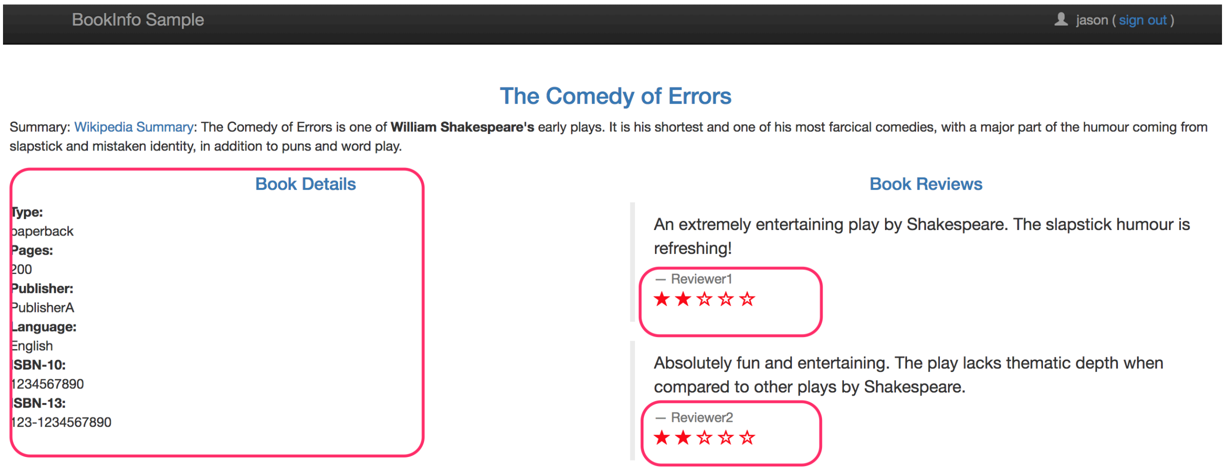
```
echo"ISTIO_INBOUND_PORTS=3306,9080,8080" > /var/lib/istio/envoy/
sidecar.env
systemctl restart istio
```

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-nbookinforegistermysqldb 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.



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
★★☆☆☆

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.

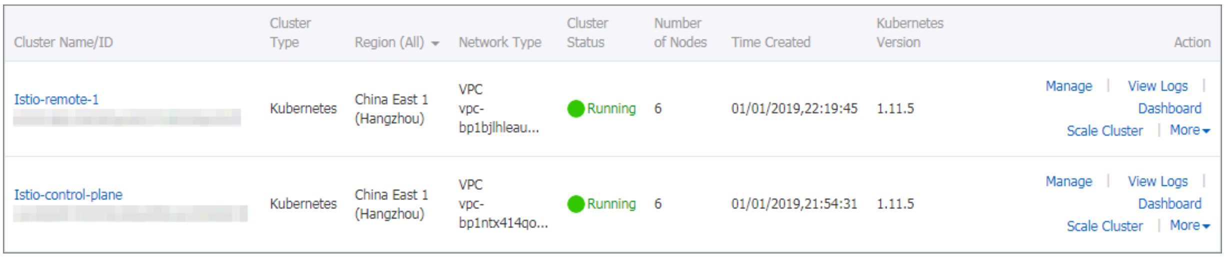
We recommend that you use Alibaba Cloud Container Service for Kubernetes to quickly build Istio, an open management platform for microservices, and integrate Istio with the microservice development of your project.

5.4 Use Istio to orchestrate application services on multiple Kubernetes clusters

Istio supports hybrid cloud development. For example, Istio can help you run applications on Alibaba Cloud Container Service, your local Kubernetes clusters, or other public cloud services. Istio provides a unified view for the entire service platform to help you manage connections between the different environments, and guarantee the security. Istio supports multiple clusters, which allows adding multiple Kubernetes clusters to a single service mesh and enables cross-cluster service discovery. According to its official introduction, Istio will support global cluster-level load balancing and support non-flat networks through gateway peering.

Use Istio to deploy application services across Kubernetes and ECS instances describes the Istio mesh expansion capability provided by Alibaba Cloud Container Service for Kubernetes.

This topic describes an example of how to use Istio to orchestrate application services on multiple Kubernetes clusters on Alibaba Cloud Container Service. After the Istio control plane is installed on a Kubernetes cluster and Istio connects this cluster and the other Kubernetes cluster, a mesh network across multiple Kubernetes clusters is generated.



Cluster Name/ID	Cluster Type	Region (All)	Network Type	Cluster Status	Number of Nodes	Time Created	Kubernetes Version	Action
Istio-remote-1 [redacted]	Kubernetes	China East 1 (Hangzhou)	VPC vpc-bp1bjlhleau...	● Running	6	01/01/2019,22:19:45	1.11.5	Manage | View Logs | Dashboard Scale Cluster | More
Istio-control-plane [redacted]	Kubernetes	China East 1 (Hangzhou)	VPC vpc-bp1ntx414qo...	● Running	6	01/01/2019,21:54:31	1.11.5	Manage | View Logs | Dashboard Scale Cluster | More

Prepare Kubernetes clusters

Alibaba Cloud Container Service for Kubernetes 1.10.4 is available. You can use the console to create a Kubernetes cluster. For more information, see *Create a Kubernetes cluster*.

Make sure that you have installed kubectl and you can connect to each Kubernetes cluster by using kubectl. Additionally, the following conditions must be met:

- Each cluster must have a unique pod CIDR block and service CIDR block.
- All pods in each cluster must be routable to each other.
- All Kubernetes control plane API servers must be routable to each other.

Cluster	ContainerCIDR	ServiceCIDR
Istio-control-plane	172.16.0.0/16	172.19.0.0/20
Istio-remote1	172.20.0.0/16	172.21.0.0/20

To obtain the files used in this example, see *Istio multi-cluster sample files*.

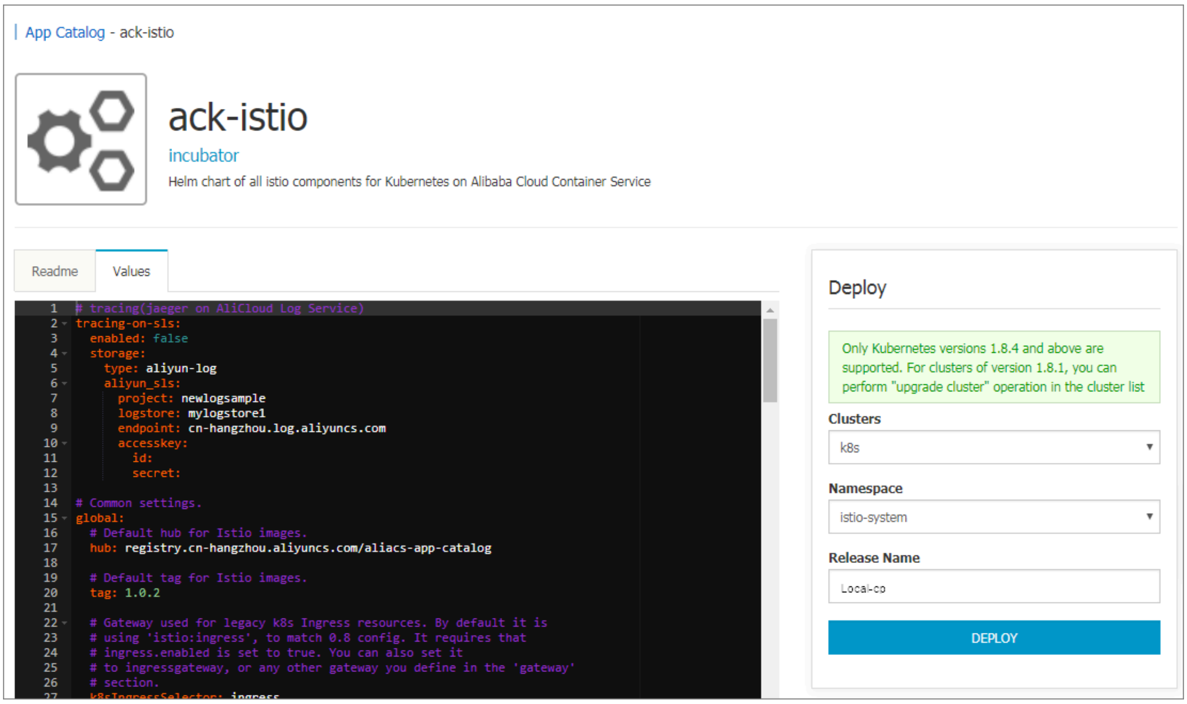
Install the Istio control plane

Deploy Istio by using App Catalog

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.

```
......
```

```
    istio-ilbgateway:
        enabled: true
......
```



App Catalog - ack-istio

ack-istio

incubator

Helm chart of all istio components for Kubernetes on Alibaba Cloud Container Service

```
# tracing(jaeger on AliCloud Log Service)
tracing-on-sls:
  enabled: false
  storage:
    type: aliyun-log
    aliyun_sls:
      project: newlogsample
      logstore: mylogstore1
      endpoint: cn-hangzhou.log.aliyuncs.com
      accesskey:
        id:
        secret:

# Common settings.
global:
  # Default hub for Istio images.
  hub: registry.cn-hangzhou.aliyuncs.com/aliacs-app-catalog

  # Default tag for Istio images.
  tag: 1.0.2

  # Gateway used for legacy k8s Ingress resources. By default it is
  # using 'istio:ingress', to match 0.8 config. It requires that
  # ingress.enabled is set to true. You can also set it
  # to ingressgateway, or any other gateway you define in the 'gateway'
  # section.
  k8sIngressSelector: ingress
```

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

Namespace: istio-system

Release Name: Local-co

Note:

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

Note:

If you have questions about CRD versions when you use Istio 1.0, see *FAQs for Istio practices on Alibaba Cloud Container Service for Kubernetes*. Questions with solutions are in the progress of updating.

Install the Istio remote component

Run the following scripts to obtain the connection information of the control plane:

```
export PILOT_POD_IP=$(kubectl -n istio-system get pod -l istio=pilot -
o jsonpath='{.items[0].status.podIP}')
export POLICY_POD_IP=$(kubectl -n istio-system get pod -l istio=mixer
 -o jsonpath='{.items[0].status.podIP}')
export STATSD_POD_IP=$(kubectl -n istio-system get pod -l istio=statsd
-prom-bridge -o jsonpath='{.items[0].status.podIP}')
export TELEMETRY_POD_IP=$(kubectl -n istio-system get pod -l istio-
mixer-type=telemetry -o jsonpath='{.items[0].status.podIP}')
export ZIPKIN_POD_IP=$(kubectl -n istio-system get pod -l app=jaeger -
o jsonpath='{.items[0].status.podI
P}')
echo "remotePilotAddress: $PILOT_POD_IP"
echo "remotePolicyAddress: $POLICY_POD_IP"
echo "remoteStatsdPromBridge: $STATSD_POD_IP"
echo "remoteTelemetryAddress: $TELEMETRY_POD_IP"
echo "remoteZipkinAddress: $ZIPKIN_POD_IP"
```

1. Log on to the *Container Service console*.
2. In the left-side navigation pane, choose Store > App Catalog, and click `ack-istio-remote` in the right pane.
3. On the displayed page, select istio-system from the namespace drop-down list, and click Values. Enter the following information in the values area to customize the Istio remote component.

   ```
     .....
     envoyStatsd:
       enabled: false
       host: "$remoteStatsdPromBridge"
   ```

   ```
   # Remote Istio endpoints. Can be hostnames or IP addresses# The
   Pilot address is required. The remaining parameters are optional.
   remotePilotAddress: "$remotePilotAddress"
   remotePolicyAddress: "$remotePolicyAddress"
   remoteTelemetryAddress: "$remoteTelemetryAddress"
   remoteZipkinAddress: "$remoteZipkinAddress"
   ```

Install the sample on the cluster Istio-control-plane

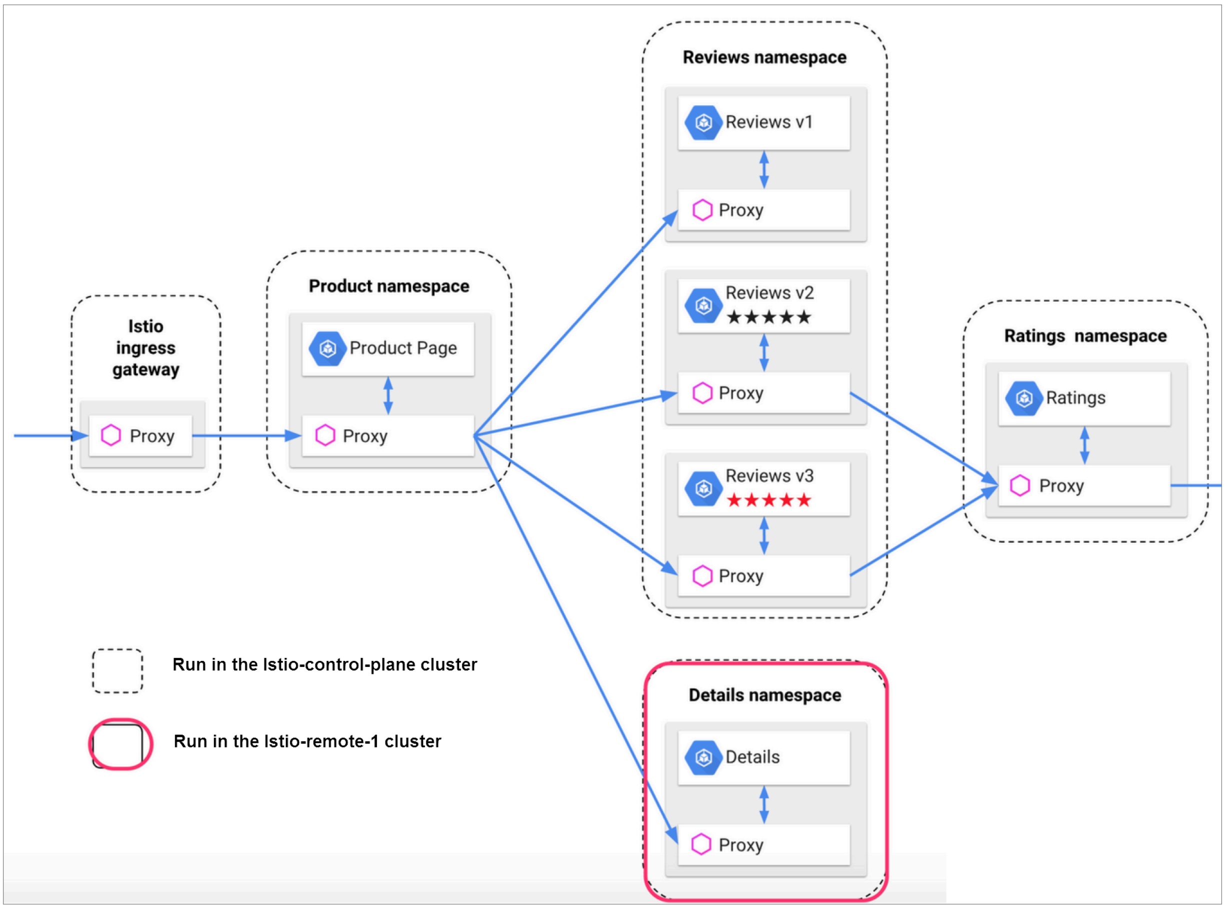
Run the following commands or use the console to create the `bookinfo` namespace, and then deploy the modified application. In the modified application, the `details` component is removed and `ingressgateway` is defined.

To obtain the files used in this example, see *Istio multi-cluster sample files*.

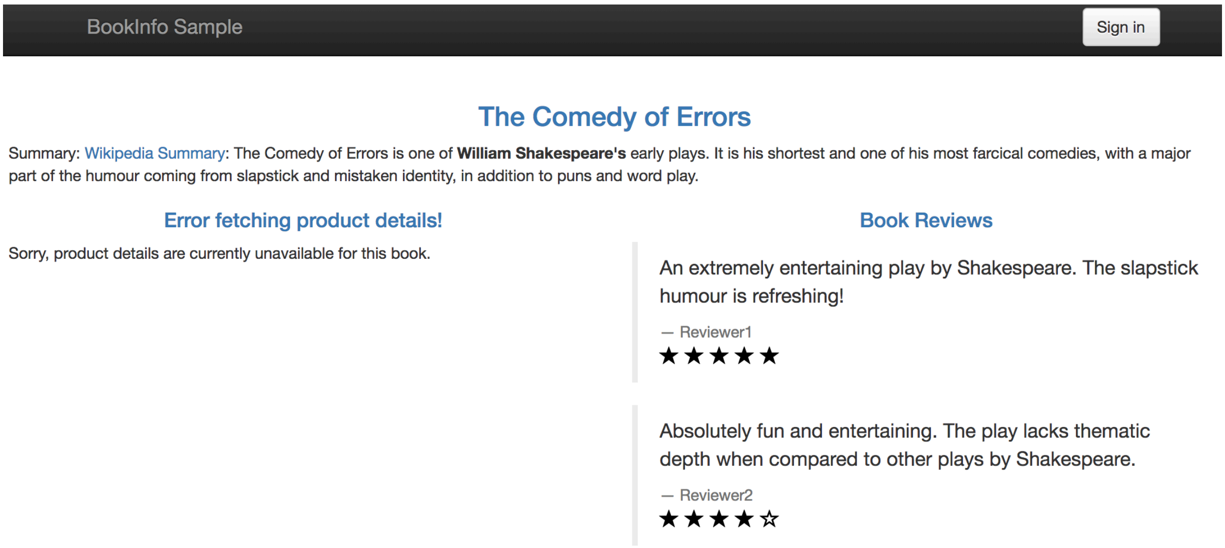
```
kubectl create ns bookinfo

\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
```

In the modified deployment based on the example, the `details` component runs on the Istio-control-plane Kubernetes cluster installed with the Istio control plane. Furthermore, the `details`component runs on the other Kubernetes cluster, that is, Istio-remote-1.



After the application runs normally, you can access the `/productpage` page through the access address exposed by `ingressgateway`. The following is a sample page. The `details` part may not be displayed because the `details` component has not been installed on the Istio-remote-1 cluster or joined to the same mesh.


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.

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 the remote cluster

The Istio control plane requires access to all the clusters in the mesh to complete service discovery. The following describes how to create a Service account in the remote cluster and grant the account RBAC permissions. The credentials of the Service account will be used to generate a kubeconfig file for the remote cluster, after which you can access the remote cluster.

The user with the root permission of each cluster must do the following on the cluster to add the cluster to the service mesh. Run the `generate-kubeconfig.sh` command in which a parameter is specified to identify each cluster.

```
./generate-kubeconfig.sh myremote1
```

After you complete the preceding steps, the kubeconfig file of the remote cluster is created in the current directory. The cluster file name and the original kubeconfig cluster name are the same.

Configure the control plane cluster

On the cluster that runs the Istio control plane, create a Secret for each remote cluster :

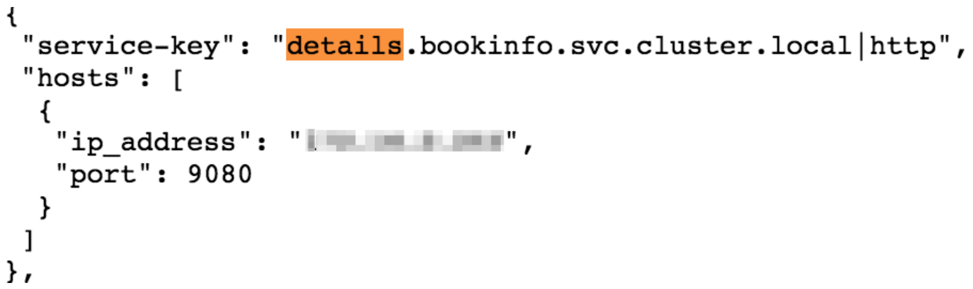
```
./create-secret.sh myremote1
```

Install the sample on the Istio-remote-1 cluster

Install the `details` part of the sample on the Istio-remote-1 cluster. To obtain the YAML file, see *Istio multi-cluster sample files*.

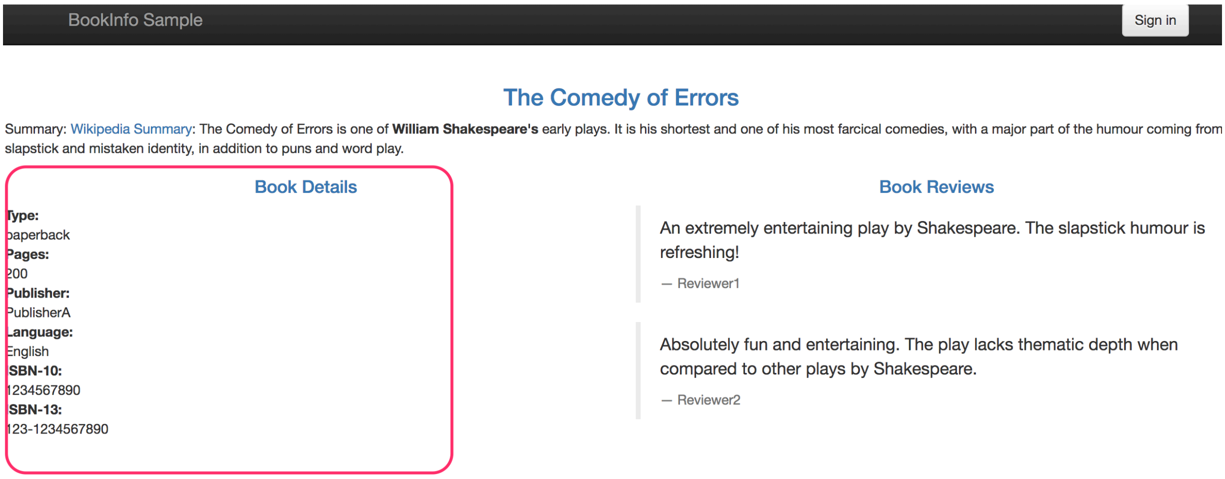
```
kubectl apply -n bookinfo -f ./bookinfo/bookinfo-details.yaml
```

The `details` service is then registered in the control plane. After that, you can check whether the `details` service is contained in the response result of `{pilot-ipAddress}:8080/v1/registration`.



```
{
 "service-key": "details.bookinfo.svc.cluster.local|http",
 "hosts": [
  {
   "ip_address": "[redacted]",
   "port": 9080
  }
 ]
},
```

Access the `/productpage` page again. The `details` part is displayed on the page as follows.



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

Conclusion

This topic describes an example of how to use Istio to orchestrate application services on multiple Kubernetes clusters on Alibaba Cloud Container Service. After the Istio control plane is installed on a Kubernetes cluster and Istio connects this cluster and the other Kubernetes cluster, a mesh network across multiple Kubernetes clusters is generated.

We recommend that you use Alibaba Cloud Container Service for Kubernetes to quickly build Istio so that you can easily integrate this platform with your microservices developments in your projects.

6 Monitoring

6.1 Use ARMS to monitor an application running in a Kubernetes cluster

This topic describes how to use Application Real-Time Monitoring Service (ARMS) to monitor an application running in Alibaba Cloud Container Service for Kubernetes.

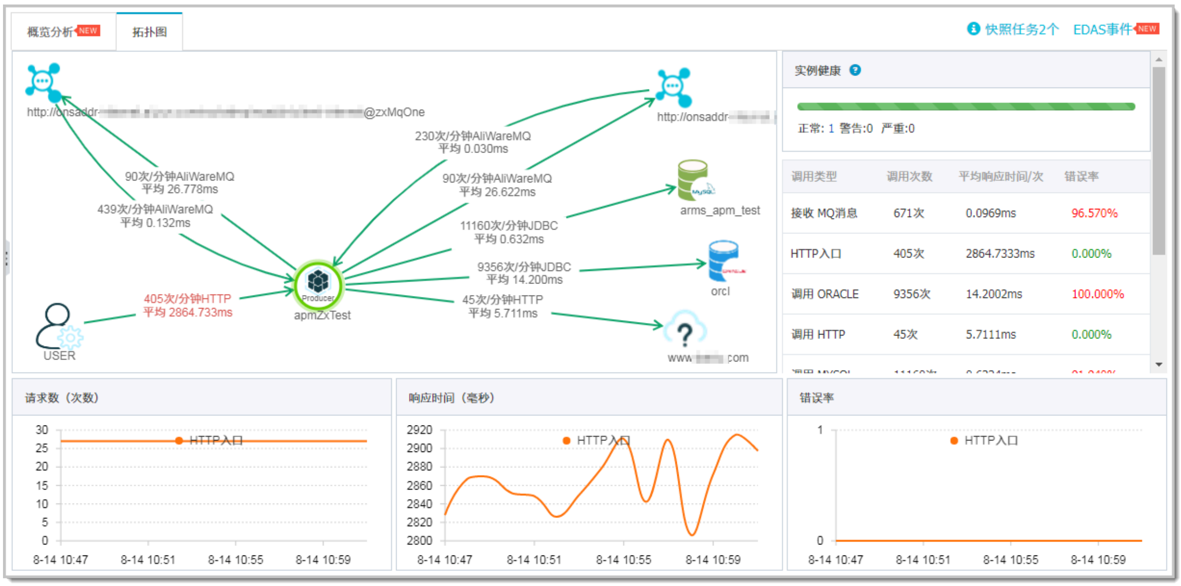
Overview about ARMS

ARMS is a Java application performance management (APM) monitoring product developed

by Alibaba Cloud. If you use ARMS to monitor a Java application, you only need to mount a probe in the application startup script without modifying any code. By comprehensively monitoring the application, the probe helps you quickly locate faulty and slow interfaces, reproduce parameter calling, detect memory leaks, and discover system bottlenecks, more efficiently diagnosing problems online. For more information, see *ARMS*.

ARMS monitors an application as follows:

- Automatically discovers the application topology



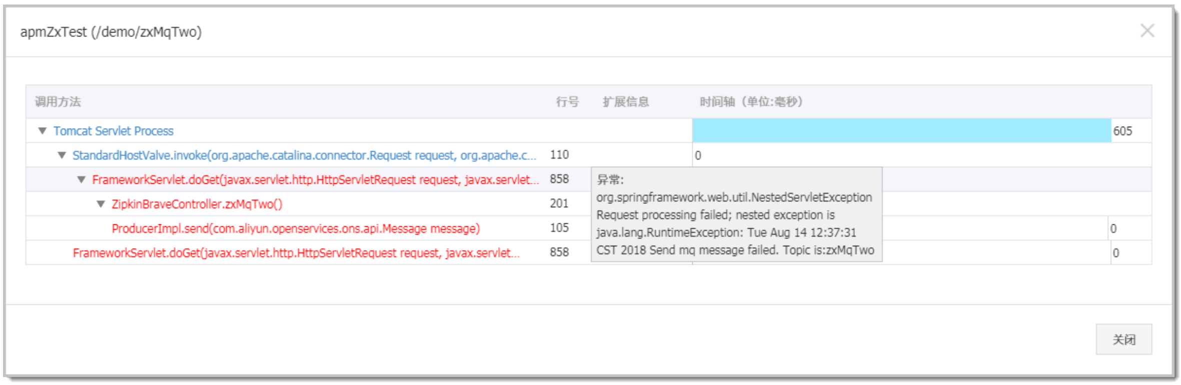
实例健康

正常: 1 警告:0 严重:0

调用类型	调用次数	平均响应时间/次	错误率
接收 MQ消息	671次	0.0969ms	96.570%
HTTP入口	405次	2864.7333ms	0.000%
调用 ORACLE	9356次	14.2002ms	100.000%
调用 HTTP	45次	5.7111ms	0.000%

- Automatically discovers and monitors interfaces
- Captures exception transactions and slow transactions, and provides SQL analysis
- Provides a Java exception report

- Provides trace-based transaction snapshot queries



apmZxTest (/demo/zxMqTwo)

调用方法	行号	扩展信息	时间轴（单位:毫秒）
▼ Tomcat Servlet Process			605
▼ StandardHostValve.invoke(org.apache.catalina.connector.Request request, org.apache.c...	110		0
▼ FrameworkServlet.doGet(javax.servlet.http.HttpServletRequest request, javax.servlet...	858		
▼ ZipkinBraveController.zxMqTwo()	201		
ProducerImpl.send(com.aliyun.openservices.ons.api.Message message)	105		0
FrameworkServlet.doGet(javax.servlet.http.HttpServletRequest request, javax.servlet...	858		0

异常:
org.springframework.web.util.NestedServletException Request processing failed; nested exception is java.lang.RuntimeException: Tue Aug 14 12:37:31 CST 2018 Send mq message failed. Topic is:zxMqTwo

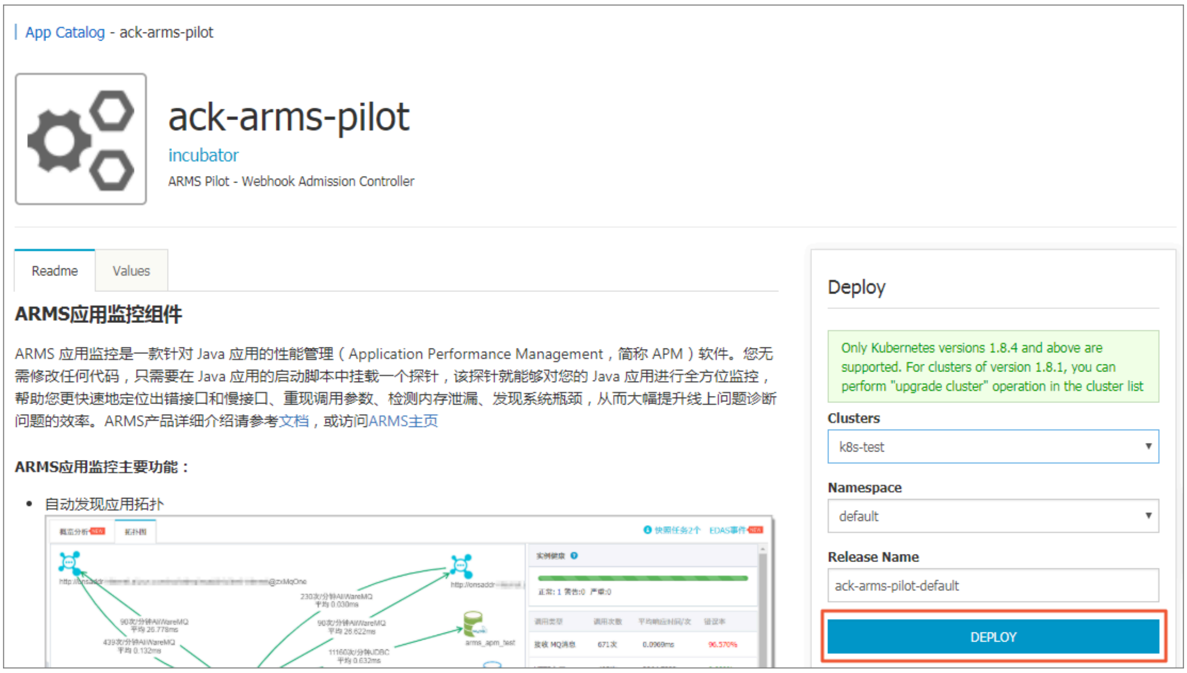
- Provides multidimensional ad-hoc troubleshooting, including multidimensional trace searches and exception trace searches
- Integrates PaaS platforms

Prerequisites

- You have created a Kubernetes cluster. For more information, see *Create a Kubernetes cluster*.
- You have activated ARMS, see *Activate ARMS*.

Install ARMS components

1. Log on to the *Container Service console*.
2. In the left-side navigation pane, choose Store > App Catalog. On the App Catalog page, select ack-arms-pilot.
3. On the App Catalog - ack-arms-pilot page, click DEPLOY.



App Catalog - ack-arms-pilot

ack-arms-pilot

incubator

ARMS Pilot - Webhook Admission Controller

ARMS应用监控组件

ARMS 应用监控是一款针对 Java 应用的性能管理（Application Performance Management，简称 APM）软件。您无需修改任何代码，只需要在 Java 应用的启动脚本中挂载一个探针，该探针就能够对您的 Java 应用进行全方位监控，帮助您更快速地定位出错接口和慢接口、重现调用参数、检测内存泄漏、发现系统瓶颈，从而大幅提升线上问题诊断问题的效率。ARMS产品详细介绍请参考文档，或访问ARMS主页

ARMS应用监控主要功能：

- 自动发现应用拓扑

Deploy

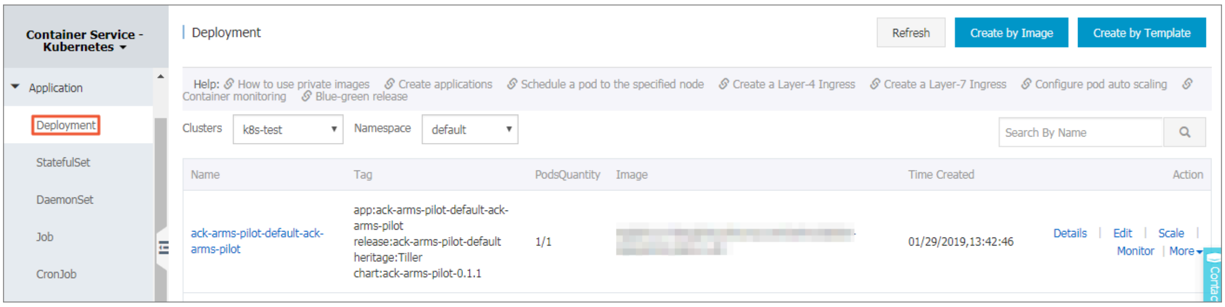
Only Kubernetes versions 1.8.4 and above are supported. For clusters of version 1.8.1, you can perform "upgrade cluster" operation in the cluster list

Clusters: k8s-test

Namespace: default

Release Name: ack-arms-pilot-default

In the left-side navigation pane, choose Application > Deployment, and then select the target cluster and namespace. An application named `ack-arms-pilot-default-ack-arms-pilot` is displayed on the page.


Container Service - Kubernetes
Application
Deployment
StatefulSet
DaemonSet
Job
CronJob
Deployment
Refresh
Create by Image
Create by Template
Help: How to use private images Create applications Schedule a pod to the specified node Create a Layer-4 Ingress Create a Layer-7 Ingress Configure pod auto scaling Container monitoring Blue-green release
Search By Name

Clusters: k8s-test Namespace: default

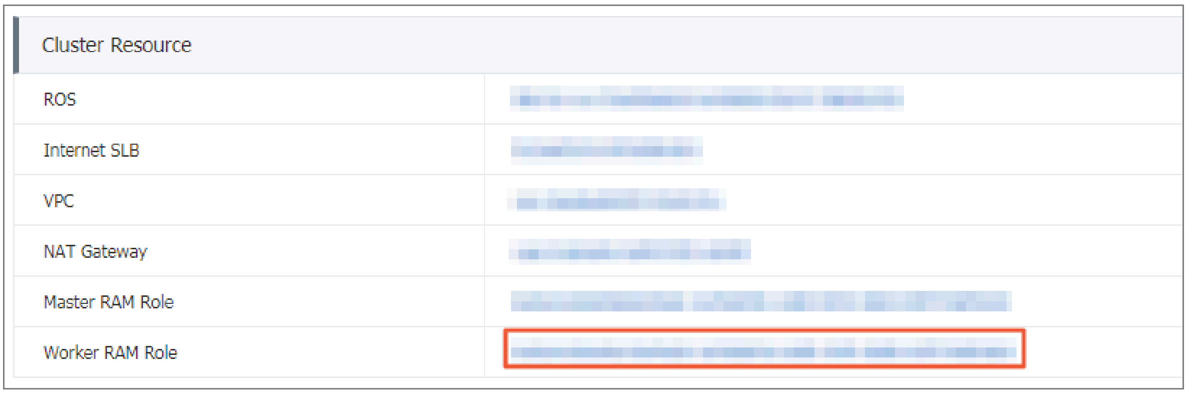
Name	Tag	PodsQuantity	Image	Time Created	Action
ack-arms-pilot-default-ack-arms-pilot	app:ack-arms-pilot-default-ack-arms-pilot release:ack-arms-pilot-default heritage:Tiller chart:ack-arms-pilot-0.1.1	1/1	[redacted]	01/29/2019,13:42:46	Details | Edit | Scale | Monitor | More

Grant permission to use ARMS

1. In the left-side navigation pane of Container Service console, click Clusters.

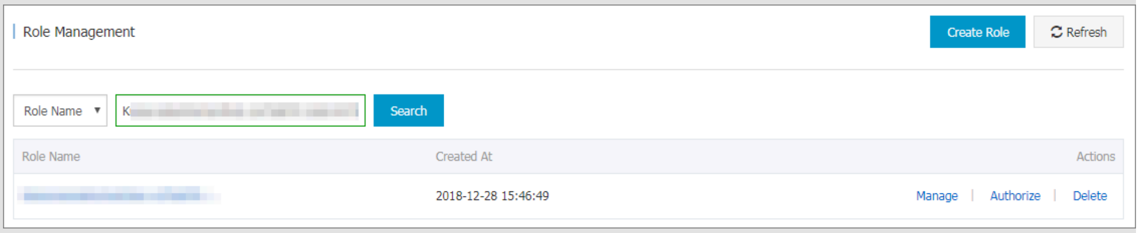
> **Note:**
> To perform the operations required in this section, you must use the primary account to log on to the Container Service console.

2. Click the target cluster name to view the cluster details.
3. In the Cluster Resources area, click Worker RAM Role.



Cluster Resource	
ROS	[redacted]
Internet SLB	[redacted]
VPC	[redacted]
NAT Gateway	[redacted]
Master RAM Role	[redacted]
Worker RAM Role	[redacted]

> **Note:**
>
> This topic uses the latest version of the RAM console.
>
> If you use an earlier version of the RAM console, you can modify the target policy document by using one of the following methods:
>
> Method 1
>
> a. In the left-side navigation pane, click Roles, and then enter the Worker RAM Role name in the Role Name box. Click the target Role Name.

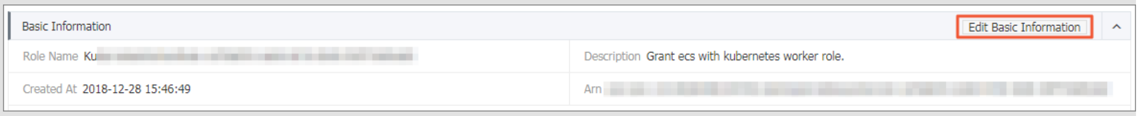


Role Management

K[redacted]

Role Name	Created At	Actions
[redacted]	2018-12-28 15:46:49	Manage | Authorize | Delete

b. In the Basic Information area, click Edit Basic Information in the upper-right corner.



Basic Information

Role Name Ku[redacted]

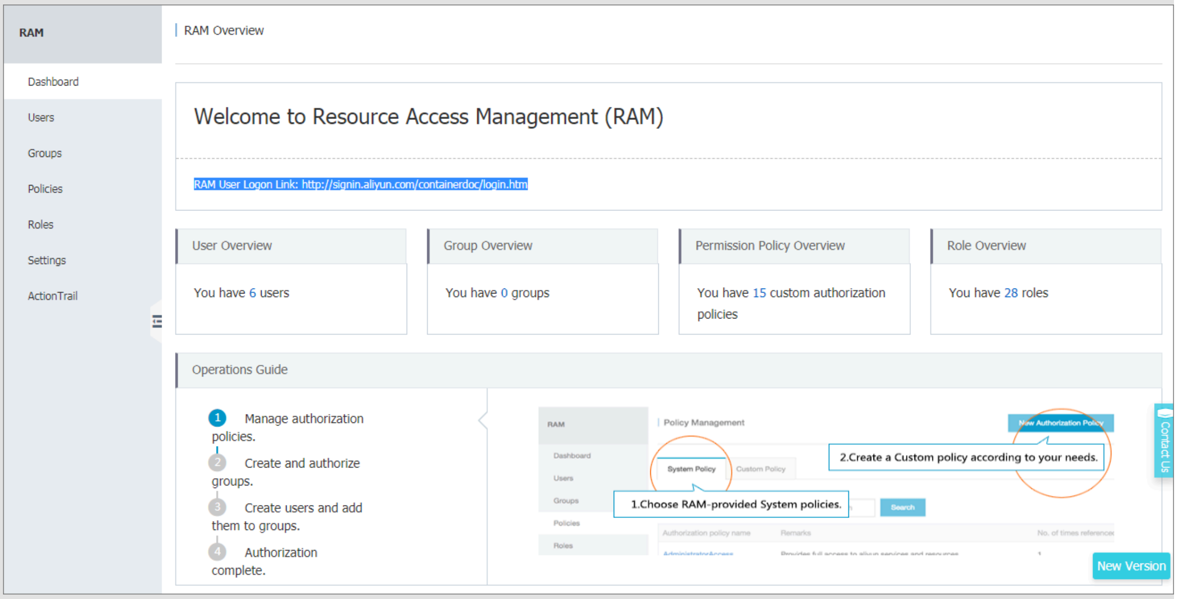
Description Grant ecs with kubernetes worker role.

Created At 2018-12-28 15:46:49

Arn [redacted]

Method 2

In the lower-right corner of the RAM dashboard page, click New Version to switch to the latest version of the RAM console. In the Container Service console, click Worker RAM Role to log on to the RAM console.



Welcome to Resource Access Management (RAM)

RAM User Logon Link: http://signin.aliyun.com/containerdoc/login.htm

User Overview: You have 6 users

Group Overview: You have 0 groups

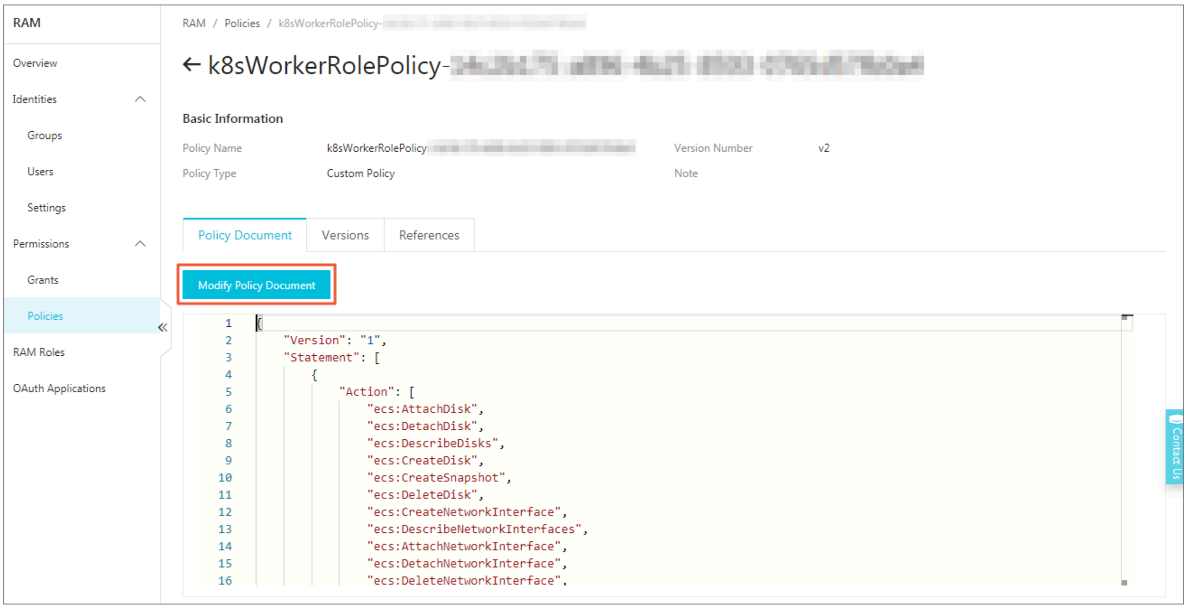
Permission Policy Overview: You have 15 custom authorization policies

Role Overview: You have 28 roles

Operations Guide

1. Manage authorization policies.
2. Create and authorize groups.
3. Create users and add them to groups.
4. Authorization complete.

4. On the RAM Roles page, click the policy name on the Permissions tab page.
5. On the Policies page, click Modify Policy Document on the Policy Document tab page.



RAM / Policies / k8sWorkerRolePolicy-[redacted]

← k8sWorkerRolePolicy-[redacted]

Basic Information

Policy Name k8sWorkerRolePolicy[redacted]

Version Number v2

Policy Type Custom Policy

Note

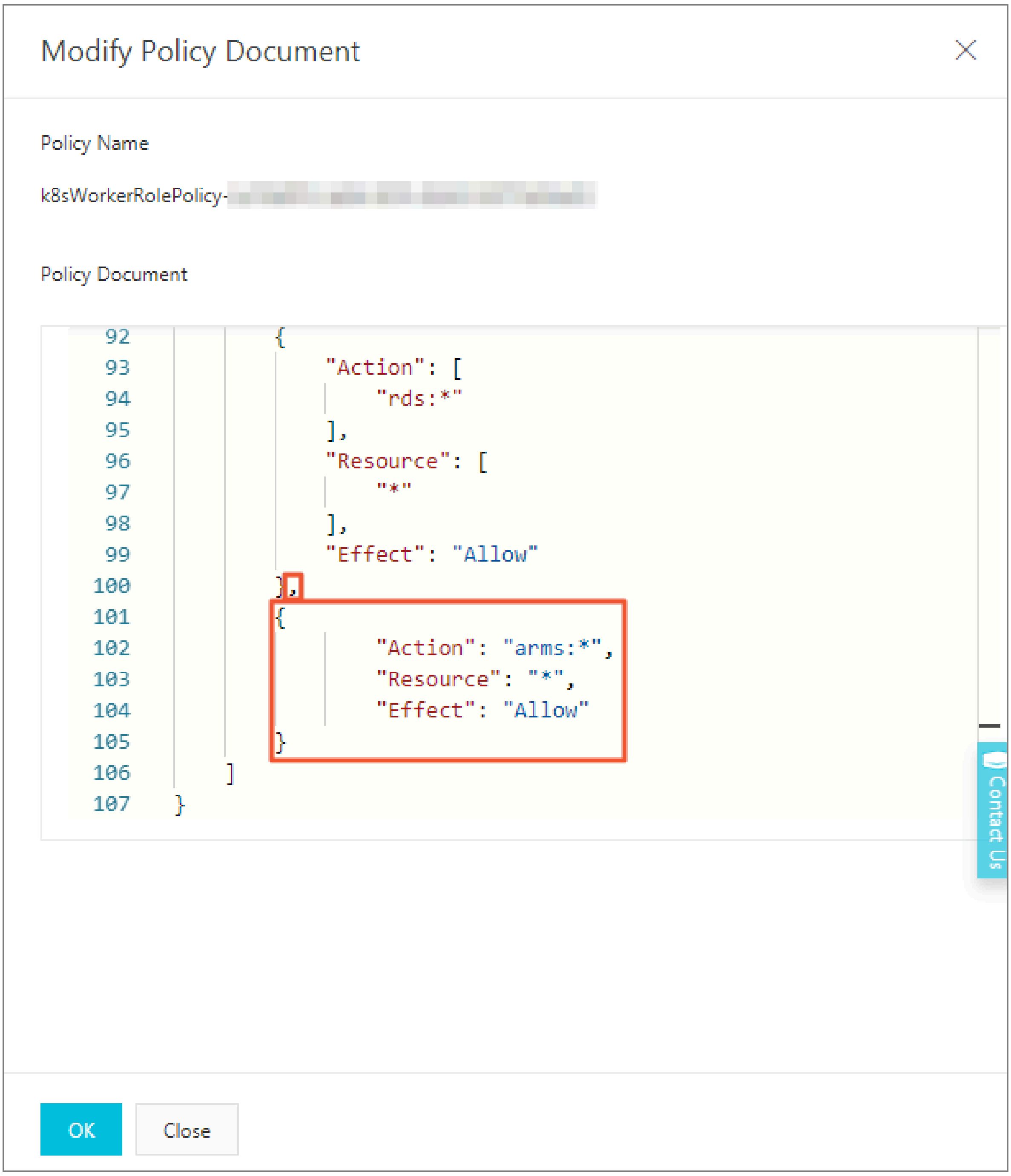
Policy Document | Versions | References

Modify Policy Document

```
{
    "Version": "1",
    "Statement": [
        {
            "Action": [
                "ecs:AttachDisk",
                "ecs:DetachDisk",
                "ecs:DescribeDisks",
                "ecs:CreateDisk",
                "ecs:CreateSnapshot",
                "ecs:DeleteDisk",
                "ecs:CreateNetworkInterface",
                "ecs:DescribeNetworkInterfaces",
                "ecs:AttachNetworkInterface",
                "ecs:DetachNetworkInterface",
                "ecs:DeleteNetworkInterface",
```

6. In the Policy Document area, add the following fields and then click OK.

```
{
    "Action": "arms:*",
    "Resource": "*",
    "Effect": "Allow"
}
```



Modify Policy Document

Policy Name

k8sWorkerRolePolicy-[redacted]

Policy Document

```
        {
            "Action": [
                "rds:*"
            ],
            "Resource": [
                "*"
            ],
            "Effect": "Allow"
        },
        {
                "Action": "arms:*",
                "Resource": "*",
                "Effect": "Allow"
        }
    ]
}
```

Deploy ARMS monitoring for an application

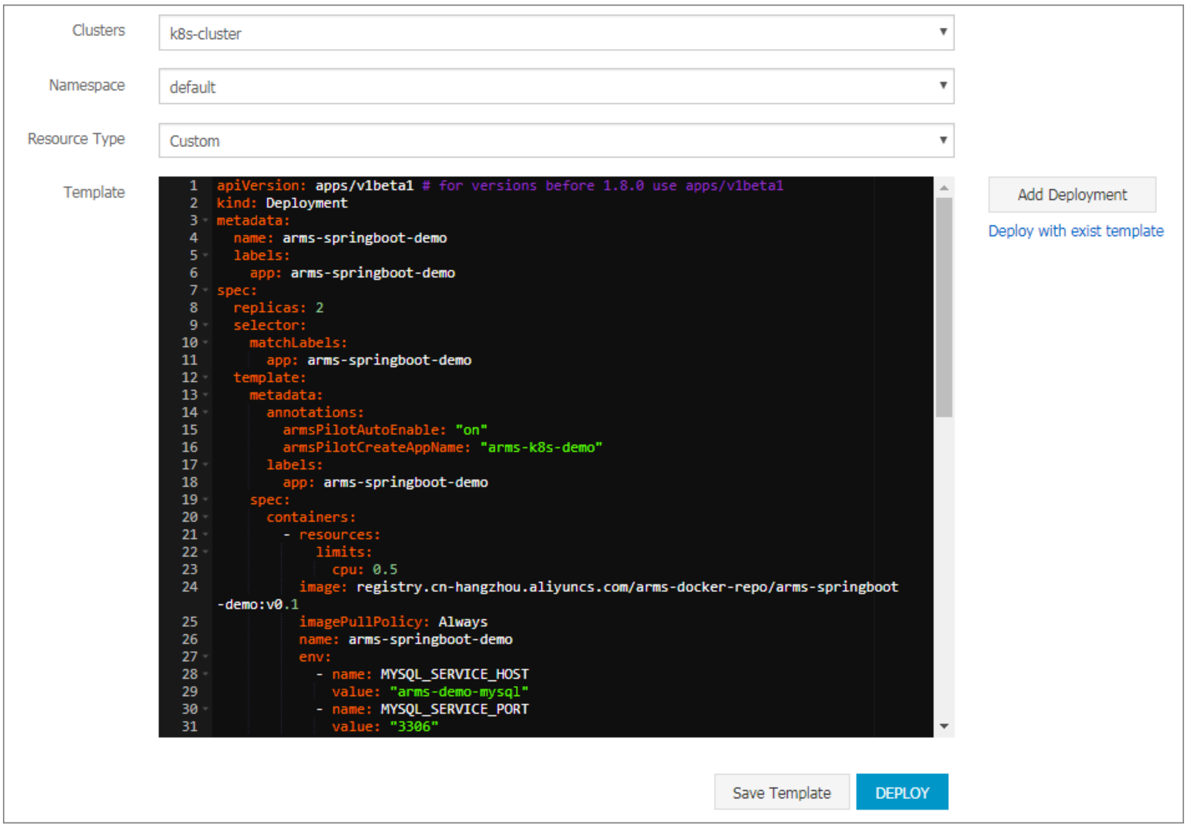
In a YAML file used to create the target deployment, add the following `annotations` to deploy ARMS application monitoring.

```
annotations:
  armsPilotAutoEnable: "on"
  armsPilotCreateAppName: "<your-deployment-name>"
```

Note:

- In the YAML file, you must add `annotations` under `metadata` of `template` in the `spec` field.
- The value of `armsPilotCreateAppName` is the name of the application monitored by ARMS.

1. In the left-side navigation pane of the Container Service console, choose Application > Deployment.
2. In the upper-right corner, click Create by Template.
3. Select the target cluster and namespace to create a deployment.



Clusters: k8s-cluster

Namespace: default

Resource Type: Custom

Template:

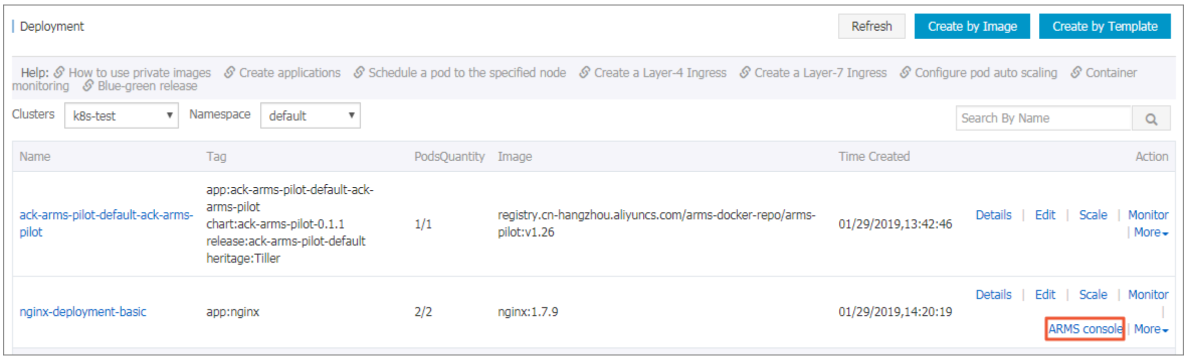
```
apiVersion: apps/v1beta1 # for versions before 1.8.0 use apps/v1beta1
kind: Deployment
metadata:
  name: arms-springboot-demo
  labels:
    app: arms-springboot-demo
spec:
  replicas: 2
  selector:
    matchLabels:
      app: arms-springboot-demo
  template:
    metadata:
      annotations:
        armsPilotAutoEnable: "on"
        armsPilotCreateAppName: "arms-k8s-demo"
      labels:
        app: arms-springboot-demo
    spec:
      containers:
        - resources:
            limits:
              cpu: 0.5
          image: registry.cn-hangzhou.aliyuncs.com/arms-docker-repo/arms-springboot
-demo:v0.1
          imagePullPolicy: Always
          name: arms-springboot-demo
          env:
            - name: MYSQL_SERVICE_HOST
              value: "arms-demo-mysql"
            - name: MYSQL_SERVICE_PORT
              value: "3306"
```

```
apiVersion: apps/v1beta1 # for versions before 1.8.0 use apps/
v1beta1
kind: Deployment
metadata:
  name: arms-springboot-demo
  labels:
    app: arms-springboot-demo
spec:
  replicas: 2
  selector:
    matchLabels:
      app: arms-springboot-demo
  template:
    metadata:
      annotations:
        armsPilotAutoEnable: "on"
        armsPilotCreateAppName: "arms-k8s-demo"
      labels:
        app: arms-springboot-demo
    spec:
      containers:
        - resources:
            limits:
              cpu: 0.5
          image: registry.cn-hangzhou.aliyuncs.com/arms-docker-repo/
arms-springboot-demo:v0.1
          imagePullPolicy: Always
          name: arms-springboot-demo
          env:
            - name: MYSQL_SERVICE_HOST
              value: "arms-demo-mysql"
            - name: MYSQL_SERVICE_PORT
              value: "3306"
---
apiVersion: apps/v1beta1 # for versions before 1.8.0 use apps/
v1beta1
kind: Deployment
metadata:
  name: arms-demo-mysql
  labels:
    app: mysql
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mysql
  template:
    metadata:
      labels:
        app: mysql
    spec:
      containers:
        - resources:
            limits:
              cpu: 0.5
          image: registry.cn-hangzhou.aliyuncs.com/arms-docker-repo/
arms-demo-mysql:v0.1
          name: mysql
          ports:
            - containerPort: 3306
              name: mysql
---
apiVersion: v1
kind: Service
metadata:
  labels:
    name: mysql
  name: arms-demo-mysql
spec:
  ports:
    # the port that this service should serve on
    - name: arms-mysql-svc
      port: 3306
      targetPort: 3306
  # label keys and values that must match in order to receive
traffic for this service
  selector:
    app: mysql
---
```

Verify the results

1. In the left-side navigation pane of the Container Service console, choose Application > Deployment, and then select the target cluster and namespace to view the created deployment.
2. In the Action column of the target deployment, click ARMS console to log on to the ARMS console to view application details such as Application Overview, Interface Invocation, and other information.

> **Note:**
> If ARMS console is not displayed in the Action column, check whether you have granted Container Service permission to use ARMS. For more information, see *Grant permission to use ARMS*.



Clusters: k8s-test
Namespace: default

Name	Tag	PodsQuantity	Image	Time Created	Action
ack-arms-pilot-default-ack-arms-pilot	app:ack-arms-pilot-default-ack-arms-pilot chart:ack-arms-pilot-0.1.1 release:ack-arms-pilot-default heritage:Tiller	1/1	registry.cn-hangzhou.aliyuncs.com/arms-docker-repo/arms-pilot:v1.26	01/29/2019,13:42:46	Details | Edit | Scale | Monitor | More
nginx-deployment-basic	app:nginx	2/2	nginx:1.7.9	01/29/2019,14:20:19	Details | Edit | Scale | Monitor | ARMS console | More

7 DevOps