Alibaba Cloud
Aliyun Container for Kubernetes

Quick Start

Issue: 20190604

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

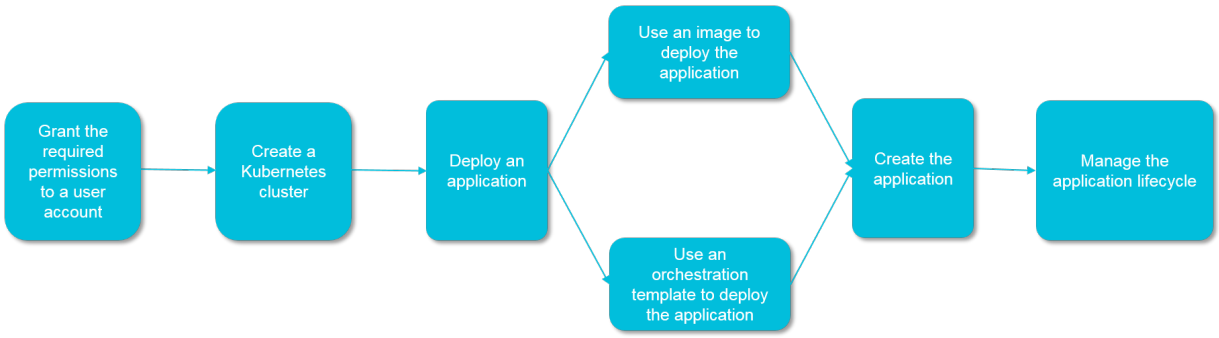
Style	Description	Example
	This warning information indicates a situation that will cause major system changes, faults, physical injuries, and other adverse results.	Danger: Resetting will result in the loss of user configuration data.
	This warning information indicates a situation that may cause major 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 informatio n, 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 Workflow

The complete workflow for Container Service is as follows.



Step 1: Create a cluster.

You can select the network environment of the cluster, and set the number of nodes and configurations for the cluster.

If you use a sub-account, grant an appropriate role to the sub-account. For more information, see *Role authorization*.

Step 2: Create an application by using an image or orchestration template.

Select an existing image or orchestration template, or create a new image or orchestration template.

If your application is composed of services supported by multiple images, create the application by using an orchestration template.

Step 3: Check the application status and the information of relevant services and containers after the deployment.

2 Basic operations

2.1 Create a Kubernetes cluster

Prerequisites

Activate the following services: Container Service, Resource Orchestration Service (ROS), and Resource Access Management (RAM). For more information about the limits and instructions, see *Create a Kubernetes cluster*.

Log on to the *Container Service console*, *ROS console*, and *RAM console* to activate the corresponding services.

Context

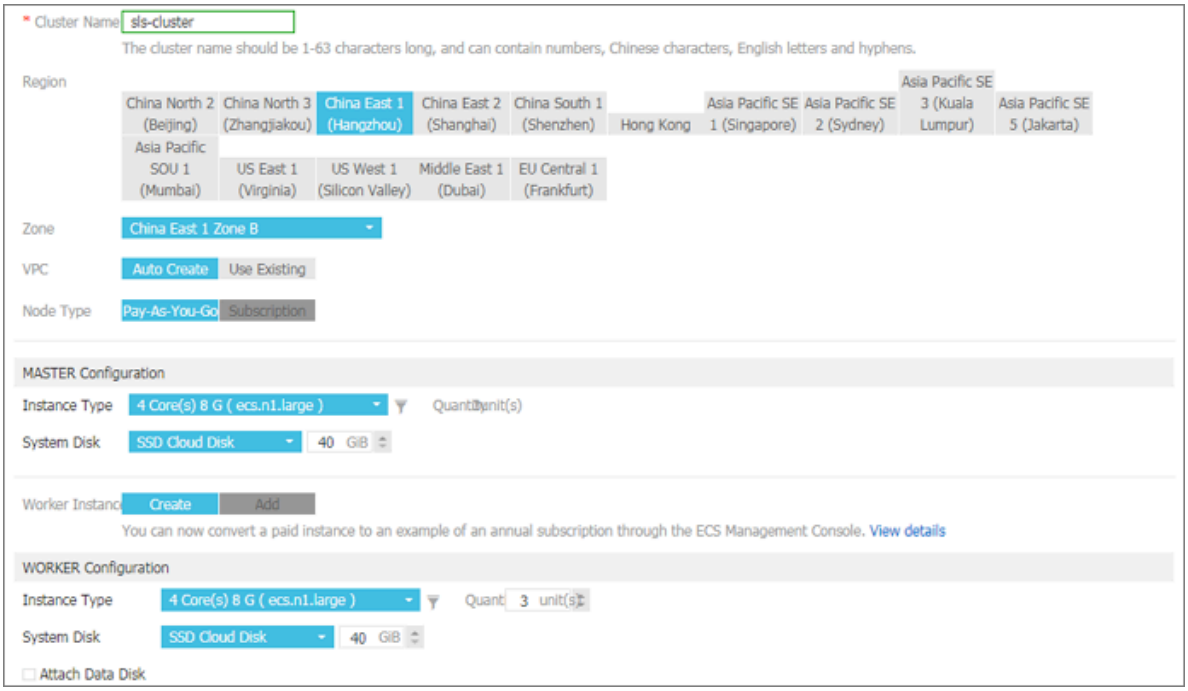
This example shows how to create a Kubernetes cluster. Some configurations use the default or the simplest configuration.

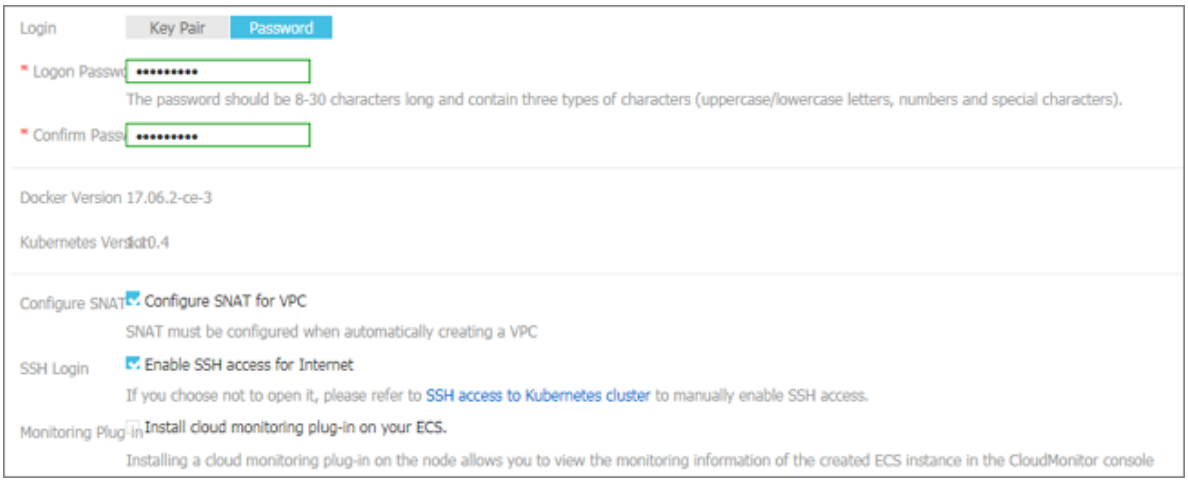
Procedure

1. Log on to the *Container Service console*.
2. Under Kubernetes, click Clusters in the left-side navigation pane. On the displayed page, click Create Kubernetes Cluster in the upper-right corner.

3. Set cluster parameters.

   Most of the configurations in this example retain the default values. The specific configuration is shown in the following figure.





Configuration	Description
Cluster Name	The cluster name can be 1–63 characters long and contain numbers, Chinese characters, English letters, and hyphens (-).
Region and Zone	The region and zone in which the cluster is located.

Configuration	Description
VPC	You can select Auto Create or Use Existing. · Auto Create: The system automatically creates a NAT Gateway for your VPC when a cluster is created. · Use Existing: If the selected VPC has a NAT Gateway, Container Service uses the NAT Gateway. Otherwise, the system automatically creates a NAT Gateway by default. If you do not want the system to automatically create a NAT Gateway, deselect the Configure SNAT for VPC check box. Note: If you deselect the check box, configure the NAT Gateway on your own to implement the VPC Internet environment with secure access, or manually configure the SNAT. Otherwise, instances in the VPC cannot access the Internet normally, which leads to cluster creation failure.
Node Type	Pay-As-You-Go and Subscription types are supported.
MASTER Configuration	Select an instance type and system disk. · Instance Type: For details, see*Instance type families* · System Disk: SSD disk and Ultra Disk are supported.

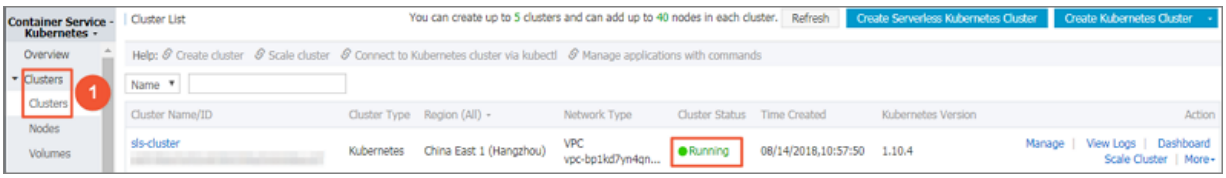
Configuration	Description
WORKER Configuration	You can select to create a Worker node or add existing ECS instances. If you select to add an instance, you can configure it as follows. · Instance Type: For details, see*Instance type families* · System Disk: SSD Disk and Ultra Disk are supported. · Attach Data Disk: SSD Disk, Ultra Disk, and Basic Disk are supported.
Login	Key Pair and Password are supported. For details, see*Access Kubernetes clusters by using SSH key pairs*
Pod Network CIDR and Service CIDR (optional)	For more information about the specific plan, see *Plan Kubernetes CIDR blocks under VPC*. Note: This option is available when you select Use Existing VPC.
Configure SNAT	SNAT must be configured if you select Auto Create a VPC. If you select Use Existing VPC, you can select whether to automatically configure SNAT Gateway. If you select not to configure SNAT automatically, configure the NAT Gateway or configure SNAT manually.
SSH Login	· If you select to enable SSH access for Internet, you can access a cluster by using SSH. · If you select not to enable SSH access for Internet, you cannot access a cluster by using SSH or connect to a cluster by using kubectl. You can manually enable SSH access. For details, see *Access Kubernetes clusters by using SSH*.

Configuration	Description
Monitoring Plug-in	You can install a cloud monitoring plug-in on the ECS node to view the monitoring information of the created ECS instances in the CloudMonitor console.
RDS Whitelist (optional)	Add the IP addresses of the ECS instances to the RDS instance whitelist. Note: This option is available when select to Use Existing VPC.
Show Advance Config	· Network Plugin: Flannel and Terway network plug-ins are supported. By default, Flannel is used. For details, see *Do I select the Terway or Flannel plugin for my Kubernetes cluster network?*. · Pod Number for Node: Maximum number of pods that can be run by a single node. · Custom Image: Indicates whether to install a custom image. The ECS instance installs the default CentOS version if no custom image is selected. · Cluster CA: Indicates whether to use a custom cluster CA.

4. Click Create Cluster in the upper-right corner.

What's next

After the cluster is successfully created, you can view the cluster in the Cluster List.



Now you have quickly created a Kubernetes cluster.

2.2 Create a deployment application by using an image

This topic describes how to use an image to create a deployment application. In this topic, an Nginx application that is accessible to the Internet is created.

Prerequisites

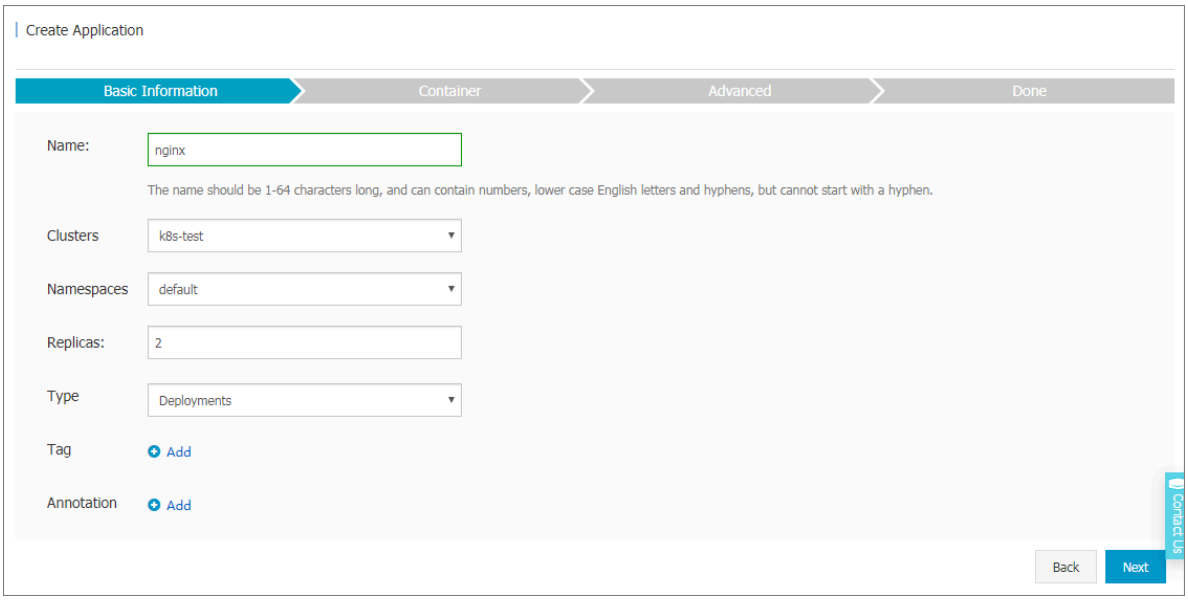
A Kubernetes cluster is created. For more information, see *Create a Kubernetes cluster*.

Procedure

1. Log on to the *Container Service console*.
2. In the left-side navigation pane under Kubernetes, choose Application > Deployment, and then click Create by Image in the upper-right corner.
3. Set Name, Cluster, Namespace, Replicas, Type, Tag, and Annotation. The replicas parameter indicates the number of pods contained in the application. Then click Next.

> **Note:**
> In this example, you need to select the Deployment type.

If you do not set Namespace, the system automatically uses the default namespace.



4. Configure a container.

> **Note:**

You can configure multiple containers for the pod of the application.

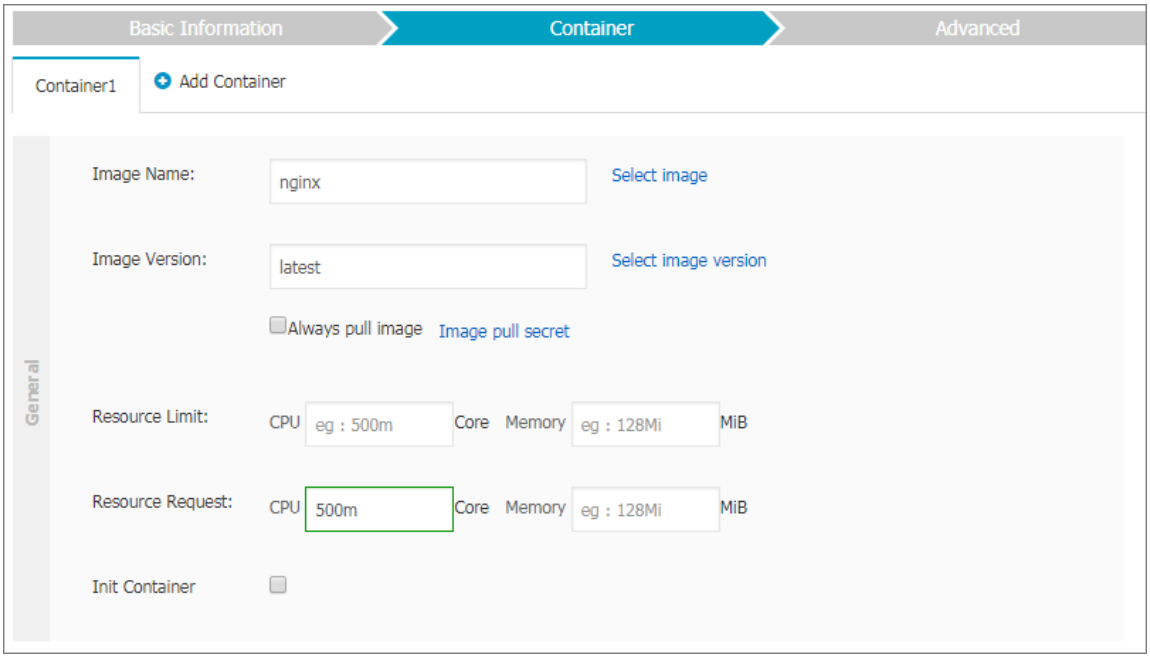
a) Set general container parameters.

- Image Name: Click Select image to select the image in the displayed dialog box and then click OK. In this example, select the Nginx image.

  You can also enter a private registry in the format of `domainname / namespace / imagename : tag` to specify an image.
- Image Version: Click Select image version to select a version. If you do not select an image version, the system uses the latest version by default.
- Always pull image: Container Service caches the image to improve deployment efficiency. During deployment, if the tag of the newly specified image is the same as that of the cached image, Container Service reuses the cached image, instead of re-pulling the same image. Therefore, if you do not modify the image tag when changing your code and image, the early image in the local cache is used in the application deployment. If you select this check box, Container Service ignores the cached image and re-pulls an image when deploying the application to make sure the latest image and code are always used.
- Image pull secret: Create a Secret for the image. A secret is required to pull a image from a private image repository. For more information, see *Use an image Secret*.
- Resource Limit: Specify the upper limit for the resources (CPU and memory) that can be used by this application to avoid occupying excessive resources. CPU is measured in millicores, that is, one thousandth of one core. Memory is measured in bytes, which can be Gi, Mi, or Ki.
- Resource Request: Specify how many resources (CPU and memory) are reserved for the application. These resources can be set to be exclusive to the container by using this parameter. If you do not set this parameter, other

services or processes will compete for resources. Then the application may become unavailable due to resource shortage.

- Init Container: Select this check box to create an Init Container that contains useful tools. For more information, see *Init containers*.



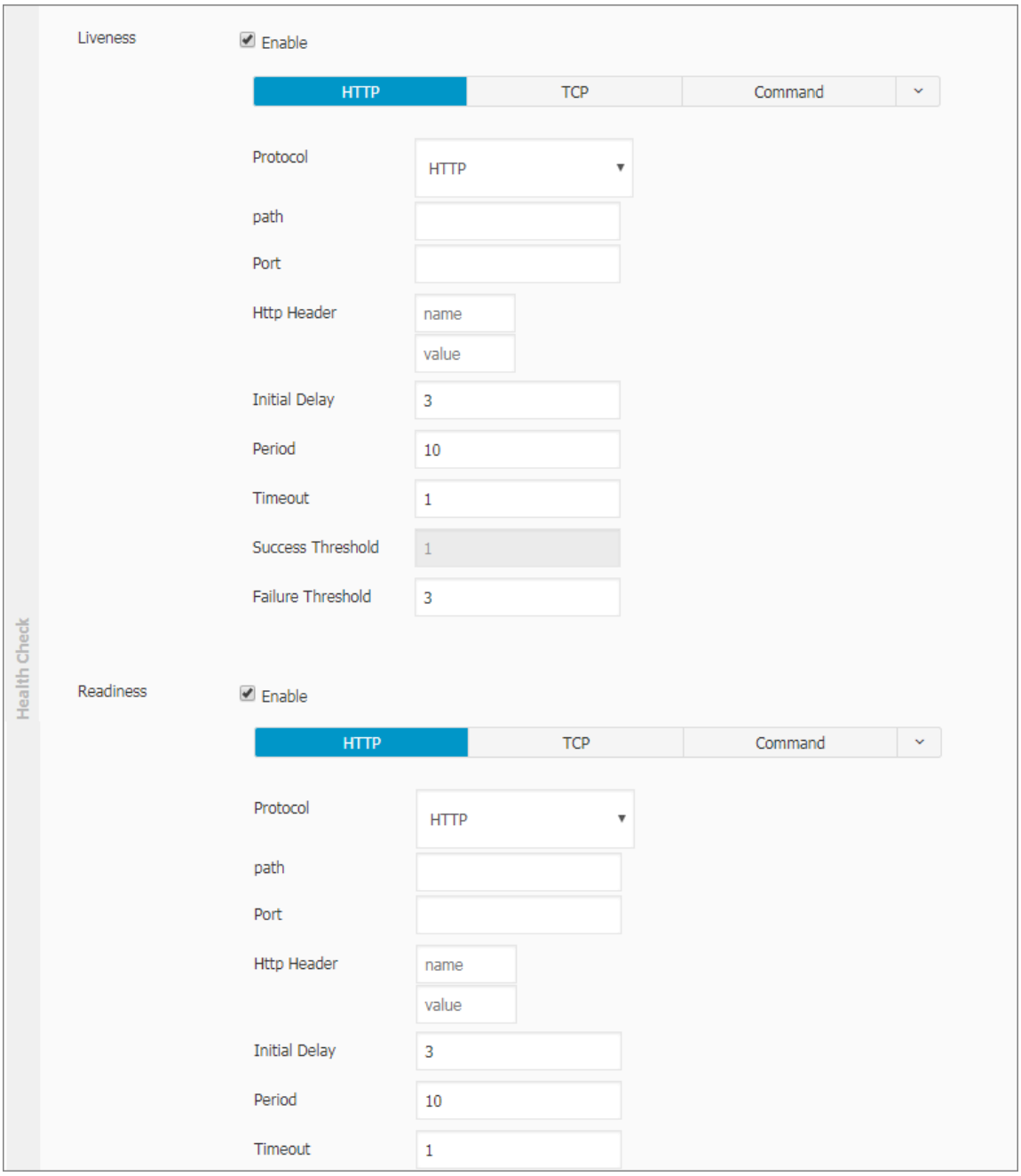
b) Optional: Set environment variables.

You can use key-value pairs to set environment variables for the pods. Environment variables are used to add environment labels or pass configurations for the pods. For more information, see *Pod variable*.

c) Optional: Set health checks.

You can set liveness probes and readiness probes. Liveness probes are used to detect when to restart the container. Readiness probes determine if the

container is ready to receive traffic. For more information about health checks, see *Configure liveness and readiness probes*.



Request method	Description
HTTP request	With this health check method, you can send an HTTP GET request to the container. The following parameters are supported: · Protocol: HTTP/HTTPS. · Path: the path to access the HTTP server. · Port: the number or name of the access port exposed by the container. The port number must be in the range of 1 to 65535. · HTTP Header: custom headers in the HTTP request. HTTP allows repeated headers. You can use a key-value pair to set an HTTP Header. · Initial Delay (in seconds): the initialDelaySeconds parameter, indicating the number of seconds for which the first probe must wait after the container is started. The default value is 3. · Period (in seconds): the periodseconds parameter, indicating the interval at which probes are performed. The default value is 10. The minimum value is 1. · Timeout (in seconds): the timeoutSeconds parameter, indicating the number of time that the probe has timed out. The default value is 1 and the minimum value is 1. · Success Threshold: The minimum number of consecutive successful probes needed for determining a probe success after a failed probe. The default value is 1 and the minimum value is 1. It must be set to 1 for a liveness probe. · Failure Threshold: The minimum number of consecutive failed probes needed for determining a probe failure after a successful probe. The default value is 3. The minimum value is 1.

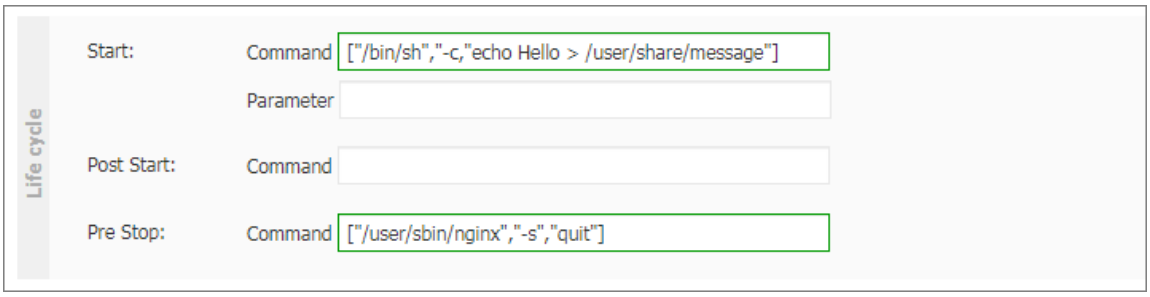
Request method	Description
TCP connection	If you use this health check method, a TCP socket is sent to the container. The kubelet then attempts to open the socket of the container on a specified port. If a connection can be established, the container is considered healthy. If not, it is considered unhealthy. The following parameters are supported: · Port: the number or name of the access port exposed by the container. The port number must be in the range of 1 to 65535. · Initial Delay (in seconds): the initialDelaySeconds parameter, indicating the seconds for the first liveness or readiness probe must wait for after the container is started. The default value is 15. · Period (in seconds): the periodseco nds parameter, indicating the interval at which probes are performed. The default value is 10. The minimum value is 1. · Timeout (in seconds): the timeoutSeconds parameter, indicating the number of time that the probe has timed out. The default value is 1 and the minimum value is 1. · Success Threshold: The minimum number of consecutive successful probes needed for determining a probe success after a failed probe . The default value is 1 and the minimum value is 1. It must be set to 1 for a liveness probe. · Failure Threshold: The minimum number of consecutive failed probes needed for determining a probe failure after a successful probe. The default value is 3. The minimum value is 1.

Request method	Description
Command line	With this heath check method, you can detect the container health by executing a probe detection command in the container. The following parameters are supported: · Command: a probe command used to detect the container health. · Initial Delay (in seconds): the initialDelaySeconds parameter, indicating the number of seconds for which the first liveness or readiness probe must wait after the container is started. The default value is 5. · Period (in seconds): the periodseco nds parameter, indicating the interval at which probes are performed. The default value is 10. The minimum value 1. · Timeout (in seconds): the timeoutSeconds parameter, indicating the number of time that the probe has timed out. The default value is 1 and the minimum value is 1. · Success Threshold: The minimum number of consecutive successful probes needed for determining a probe success after a failed probe . The default value is 1 and the minimum value is 1. It must be set to 1 for a liveness probe. · Failure Threshold: The minimum number of consecutive failed probes needed for determining a probe failure after a successful probe. The default value is 3. The minimum value is 1.

d) Set life cycle rules.

You can set the following parameters for the container life cycle: start, post start, and pre-stop. For more information, see *Attach handlers to container lifecycle events*.

- Start: Set a pre-start command and parameter for the container.
- Post Start: Set a post-start command for the container.
- Pre Stop: Set a pre-stop command for the container.

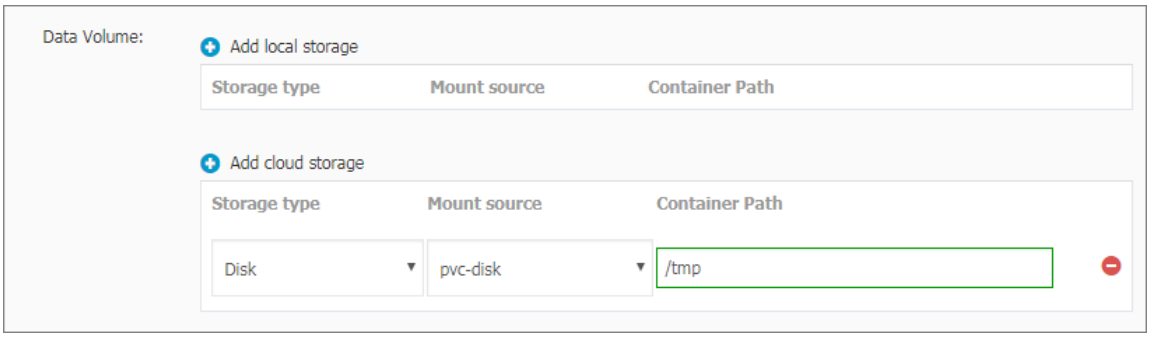


e) Optional: Set volumes.

You can configure local storage and cloud storage.

- local storage: Supported storage types include HostPath, ConfigMap, Secret, and EmptyDir. By setting a type of local storage, you can mount its mount source to the container path. For more information, see *Volumes*.
- cloud storage: Supported types of cloud storage include disks, Network Attached Storage (NAS), and Object Storage Service (OSS).

This example sets a disk as the volume and mounts the disk to the `/ tmp` container path. Then container data generated in this path is stored to the disk.



f) Optional: Set Log Service. You can set collection parameters and customize tags.

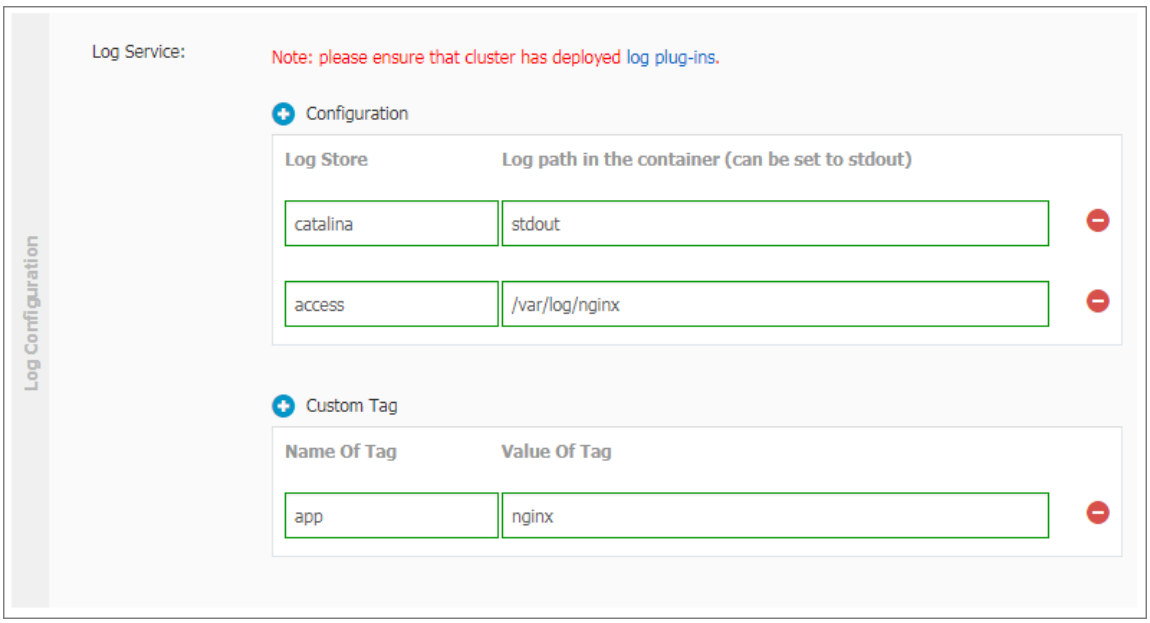
Note:

Make sure that you have deployed a Kubernetes cluster and installed the log plugin on the cluster.

Set the following log collection parameters:

- Log Store: Set a Logstore. After you specify the Logstore name, the Logstore is generated in Log Service to store collected logs.
- Log path in the container: Set this parameter to stdout or set a log path.
  - stdout: If you set the log path parameter to stdout, you can collect the standard output logs of the container.
  - text log: If you specify a container log path, you can collect the text logs of the path. Wildcards can be used in setting the log file name for a log path. In this example, text logs in the path of /var/log/nginx are collected.

You can also customize log tags. The customized log tags can be collected together with container output logs and can benefit log analysis actions such as collecting log statistics and filtering specific logs.



5. Click Next.

6. Configure advanced settings.
   a) Set Access Control.

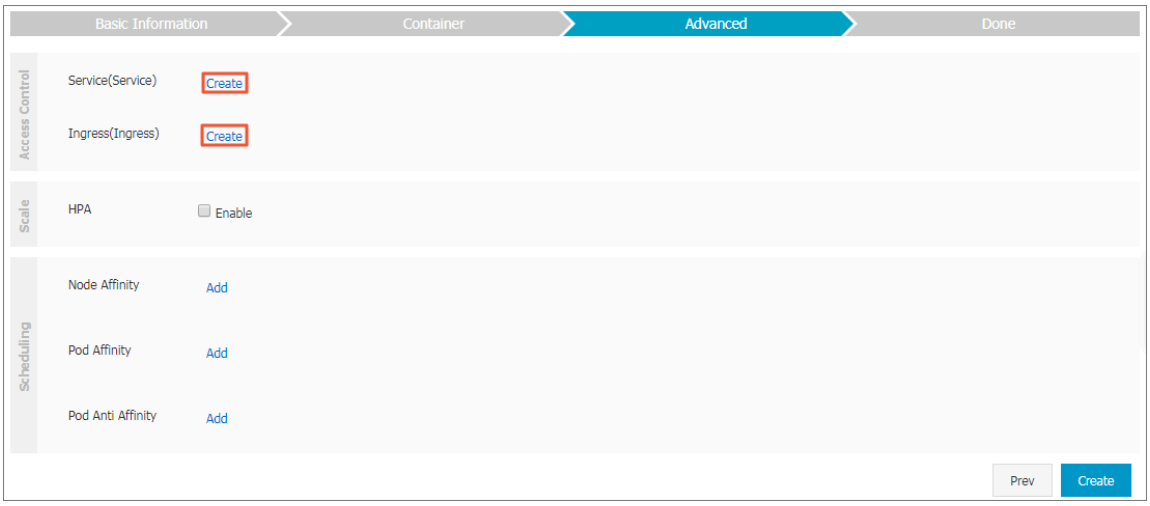
      You can set the methods to expose the application pod and then click Create. In this example, a cluster IP service and an Ingress are set to create an Nginx application that is accessible for the Internet.

      Note:

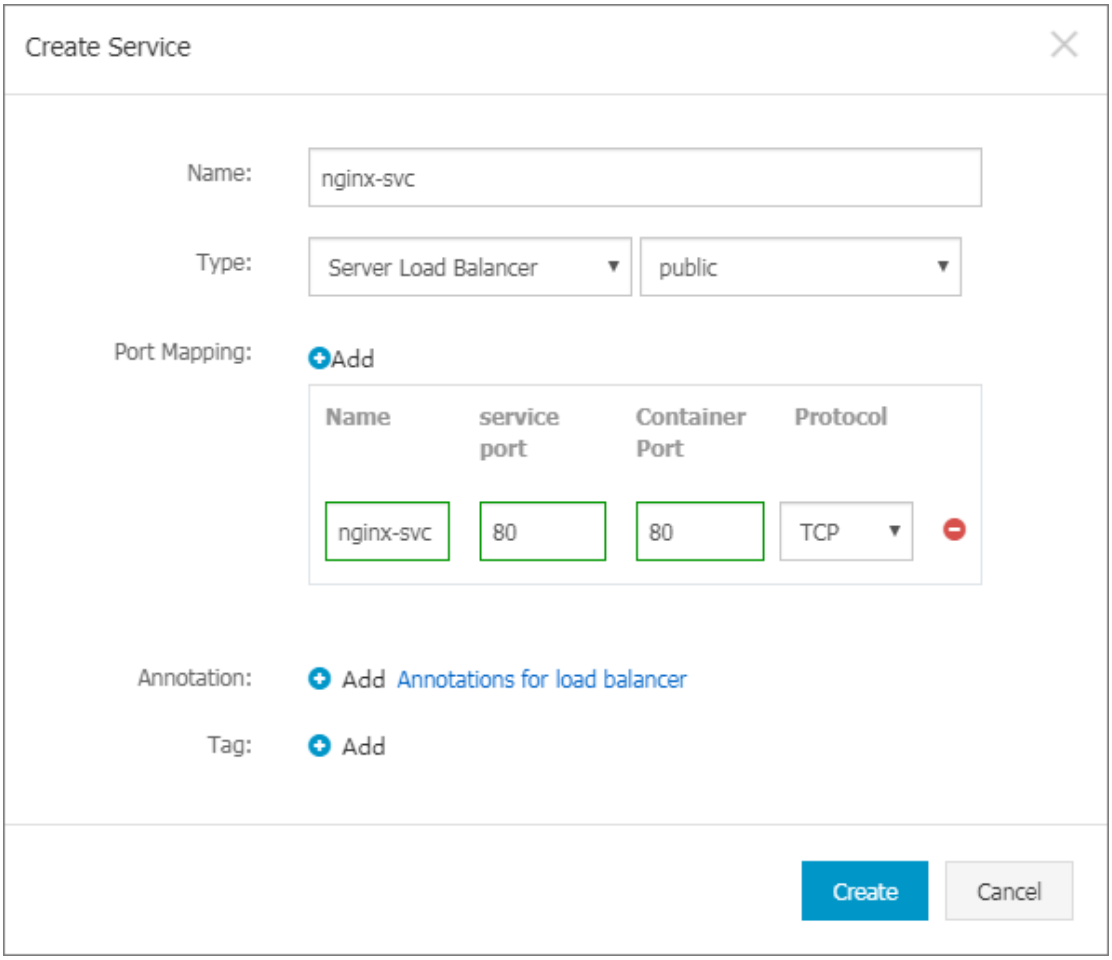
      You can set access methods according to the communication requirements of your application.

      - `Internal application`: an application that works only inside the cluster. You can create a cluster IP service or a node port service as needed for communication within the cluster.

- `External application`: an application that needs to be exposed to the Internet. You can set how the application is accessed by using either of the following two methods:
  - Create a Server Load Balancer service. This method uses Alibaba Cloud Server Load Balancer (SLB) to provide Internet accessibility for the application.
  - Create a cluster IP service or a node port service, and create an Ingress. This method provides Internet accessibility through the Ingress. For more information, see *Ingress*.



A. Click Create on the right of Service. Configure a service in the displayed dialog box, and then click Create.



- Name: Enter the service name. The default is `applicatio nname - svc`.
- Type: Select one service type.
  - Cluster IP: Exposes the service by using the internal IP address of your cluster. If you select this service type, the service is accessible only within the cluster.
  - Node port: Exposes the service by using the IP address and the static port (NodePort) of each node. A node port service routes to a cluster IP service that is automatically created. You can access the node port service from outside the cluster by requesting `< NodeIP >:< NodePort >`.
  - Server Load Balancer: Alibaba Cloud Server Load Balancer service. With this type of service, you can set an Internet or intranet access method

for your application. SLB can route to a node port service and a cluster IP service.

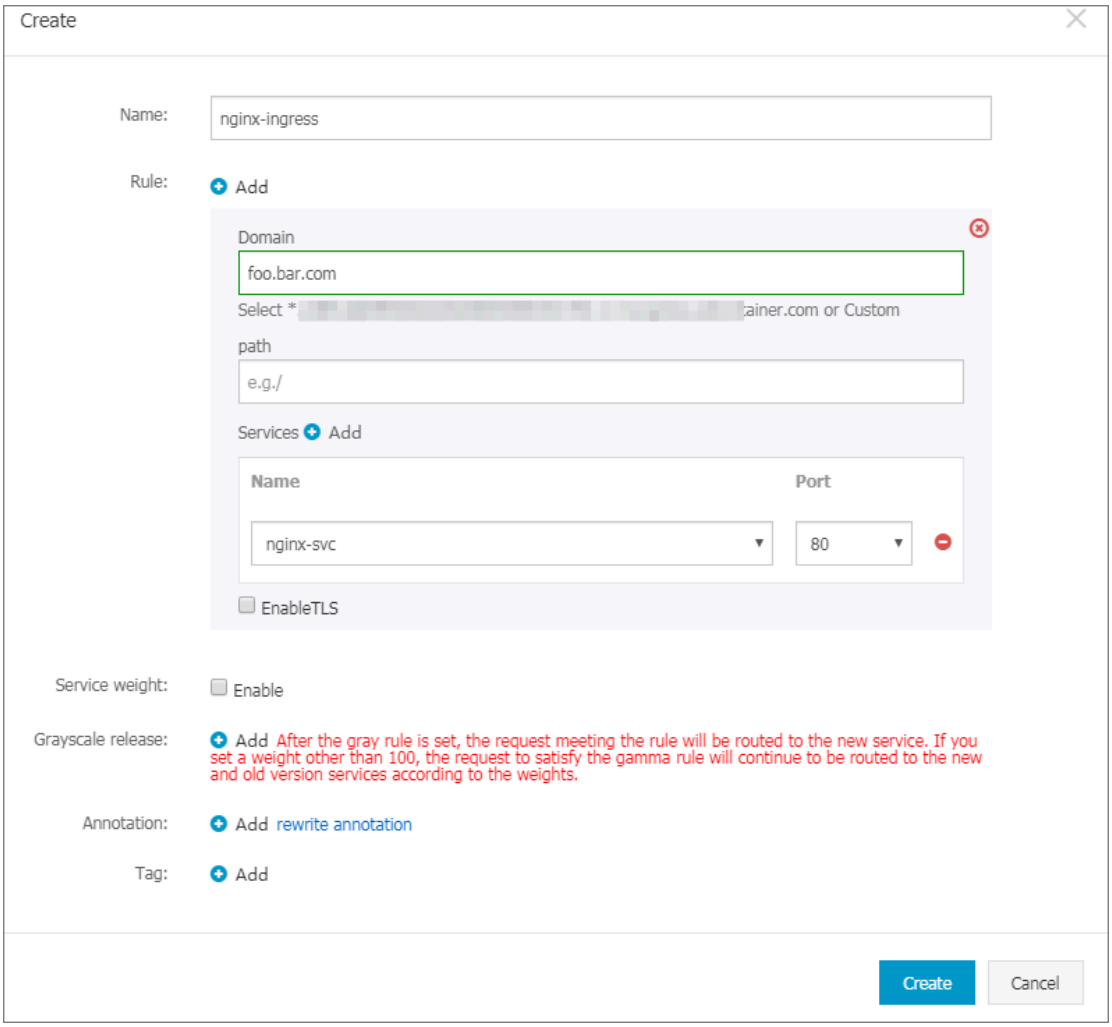
- Port Mapping: Add a service port and a container port, and select the TCP or UDP protocol. If you select the node port Type, you must add a node port to avoid port conflict.
- annotation: Add an annotation to the service. You can set SLB parameters. For more information, see *Access services by using Server Load Balancer*.
- Tag: Add a tag to the service to identify the service.

B. Click Create on the right of Ingress. In the displayed dialog box, configure an Ingress rule for the application pod, and then click Create. For more information, see *Ingress configurations*.

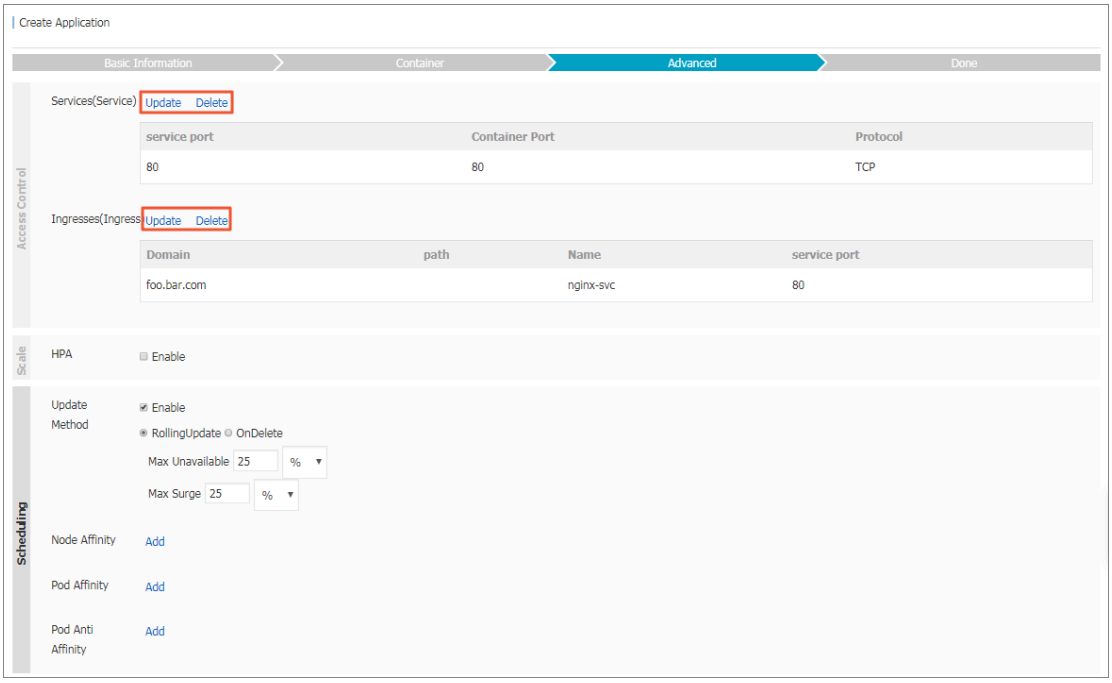
> Note:
> When you create an application by using an image, you can create an Ingress rule for only one service. In this example, a virtual host name is used as the

test domain name. You need to add a record to the host. You must use a filing domain name when you create your application.

```
101 . 37 . 224 . 146     foo . bar . com     # This    is    the
  IP    address    of    the    Ingress .
```

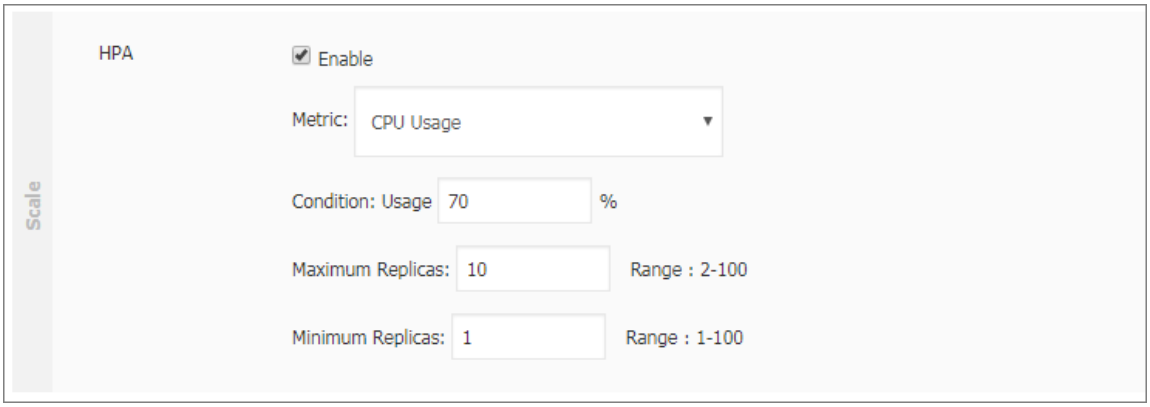


C. In the access control area, the created service and Ingress are displayed. You can perform further configurations by clicking Update or Delete.



b) Optional: Set Horizontal Pod Autoscaling (HPA).

You enable HPA by selecting the Enable check box. Alibaba Cloud Container Service for Kubernetes provides pod auto scaling to deal with different application workloads. That is, you can change the number of pods according to the container CPU and memory usage.



Note:

To use this function, you must set required resources for the pod. Otherwise, pod auto scaling cannot take effect. For more information, see general container settings.

- Metric: resource type. CPU or memory is available. This parameter must be specified with a resource type that is the same as the required resource type.
- Condition: the percentage value of resource usage. The number of containers increases when the resource usage exceeds this value.
- Maximum Replicas: the maximum number of the containers that the deployment can include.
- Minimum Replicas: the minimum number of the containers that the deployment can include.

c) Optional: Set Scheduling.

You can set an update method, node affinity, pod affinity, and pod anti affinity. For more information, see *Affinity and anti-affinity*.

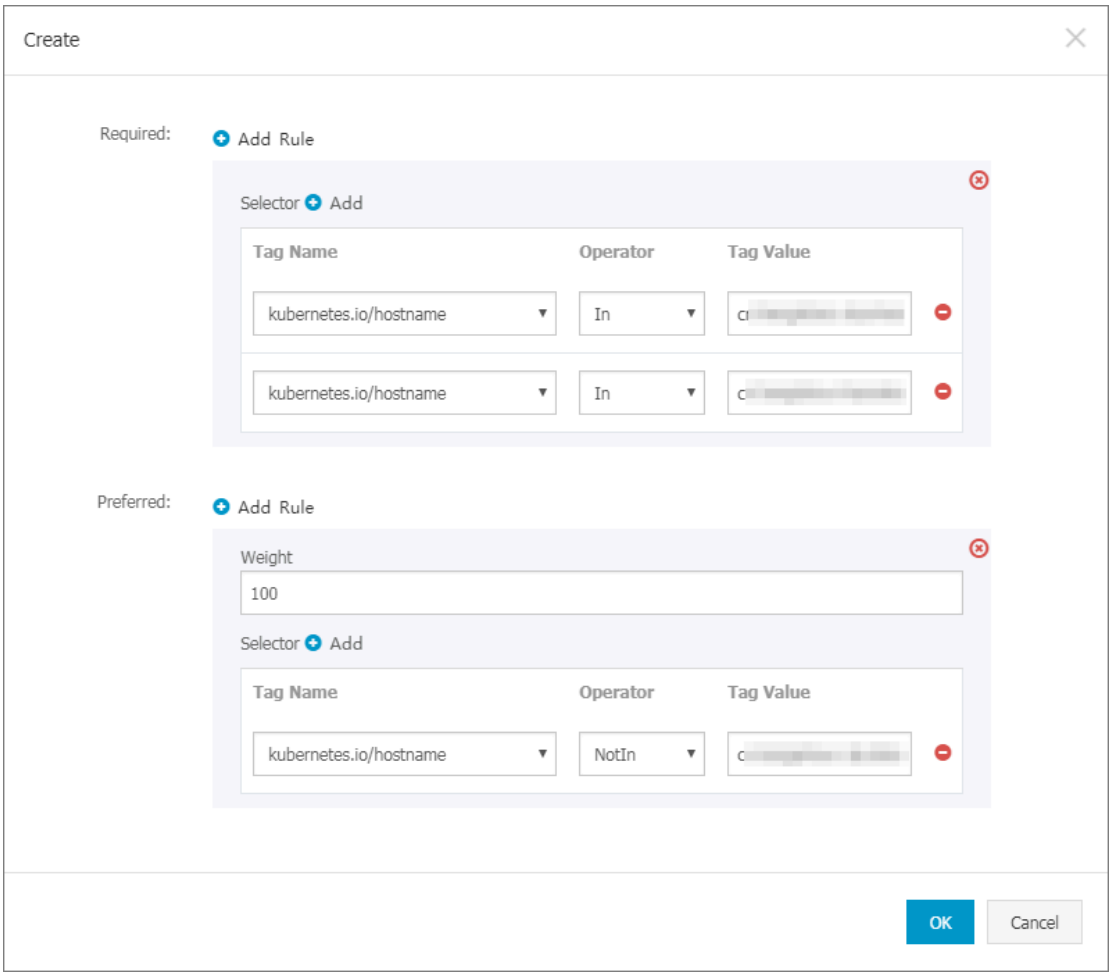
Note:

Affinity scheduling depends on node tags and pod tags. You can use built-in or customized tags to schedule nodes or pods.

A. Set Update Method.

You can select the `RollingUpdate` or `Recreate` (OnDelete) method to replace old pods with new ones. For more information, see *Deployments*.

B. Set Node Affinity by using node tags.



Required rules and preferred rules are supported, and available operators include `In`, `NotIn`, `Exists`, `DoesNotExist`, `Gt`, and `Lt`.

- Required rules must be satisfied and correspond to `requiredDuringSchedulingIgnoreDuringExecution`. The required rules

have the same effect as `NodeSelect or`. In this example, the pod can be scheduled to only a node with the specified tags.

You can add multiple required rules, but only one required rule needs to be satisfied for pod scheduling.

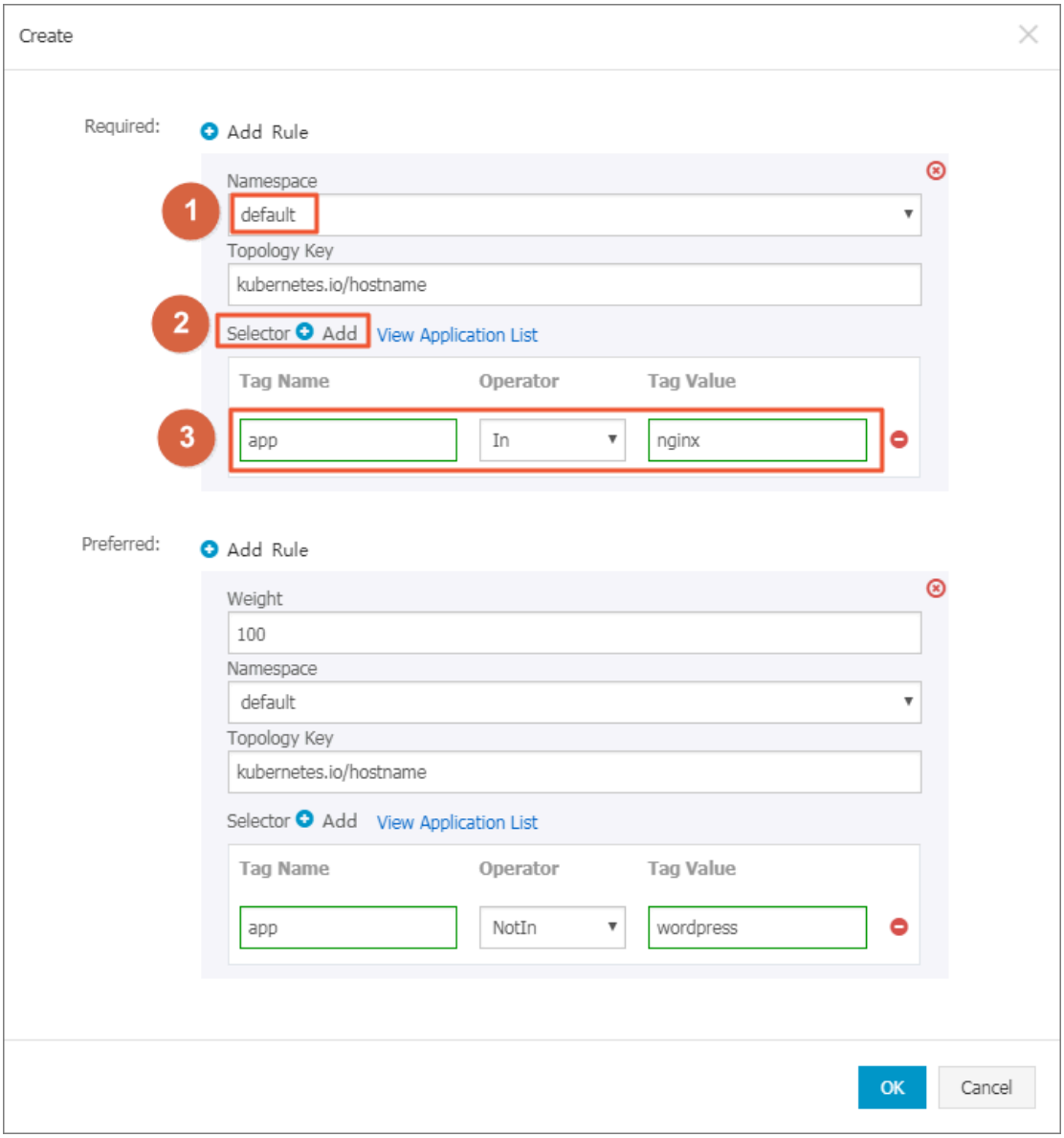
- Preferred rules can be unnecessarily satisfied and correspond to `preferredD uringSched ulingIgnor edDuringEx ecution`. With the scheduling setting in this example, the system tries not to schedule the pod to the nodes with the specified tag.

  You can also set Weight for each preferred rule. If multiple nodes satisfies the preferred rules, the system schedules the pod to a node with the highest weight.

  You can add multiple preferred rules, and all the rules must be satisfied for pod scheduling.

C. Set Pod Affinity to deploy the application pod in a topology domain together with other pods. For example, to reduce network latency between the services

that communicate with each other, you can deploy their pods to a topology domain (for example, a host).



You can schedule pods according to tags of pods running on nodes. Required rules and preferred rules are supported, and available operators include `In`, `NotIn`, `Exists`, `DoesNotExist`.

- Required rules must be satisfied and correspond to `requiredDuringSchedulingIgnoreDuringExecution`. All specified conditions of required rules must be met for pod affinity scheduling.
  - Namespace: Set a namespace. This parameter is required because the scheduling policy is based on pod tags.

- Topology Key: Set a topology domain to which pods are scheduled. This parameter takes effect through node tags. For example, if you set `kubernetes . io / hostname` as the topology key, a node is used to identify a topology. If you set `beta . kubernetes . io / os` as the topology key, a node operating system is used to identify a topology.
- Selector: Click this button to add a required rule.
- View Applicaiton List: Click View Applicaiton List, a dialog box is displayed. In the dialog box, you can view applications in each namespace and export application tags to the dialog box in which you set pod affinity.
- Required rule tag: Set a tag name, its operator, and the tag value for existing applications. This example schedules the application to be created to a host on which applications tagged with `app : nginx` run.

· Preferred rules can be unnecessarily satisfied and correspond to `preferredD uringSched ulingIgnor edDuringEx ecution .` Specified conditions of required rules will be met as many as possible for pod affinity scheduling.

  You can set Weight for each preferred rule. The weight value range is 1 to 100. If multiple nodes satisfies the preferred rules, the system schedules the pod to a node with the highest weight. Other parameters are the same with the required rule setting.

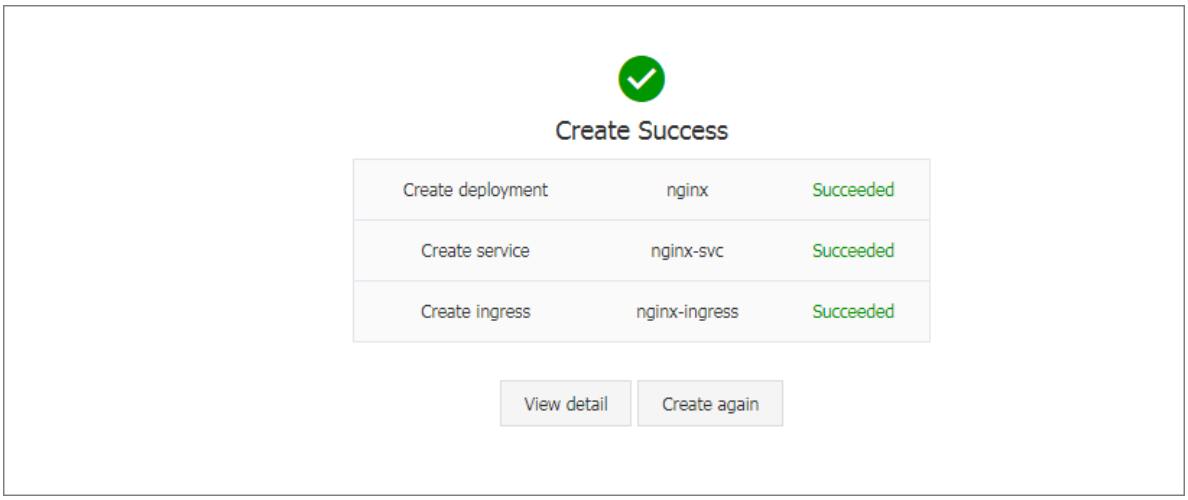
D. Set Pod Anti Affinity to deploy the application pods in a topology domain that excludes other pods. Scenarios that use pod anti affinity scheduling include:

· Distribute the pods of a service to different topology domains (for example , different hosts) to improve the service stability.
· Grant a pod the exclusive access to a node so as to guarantee that no other pods use the resources of the node.
· Distribute pods of the services that may affect each other to different hosts.

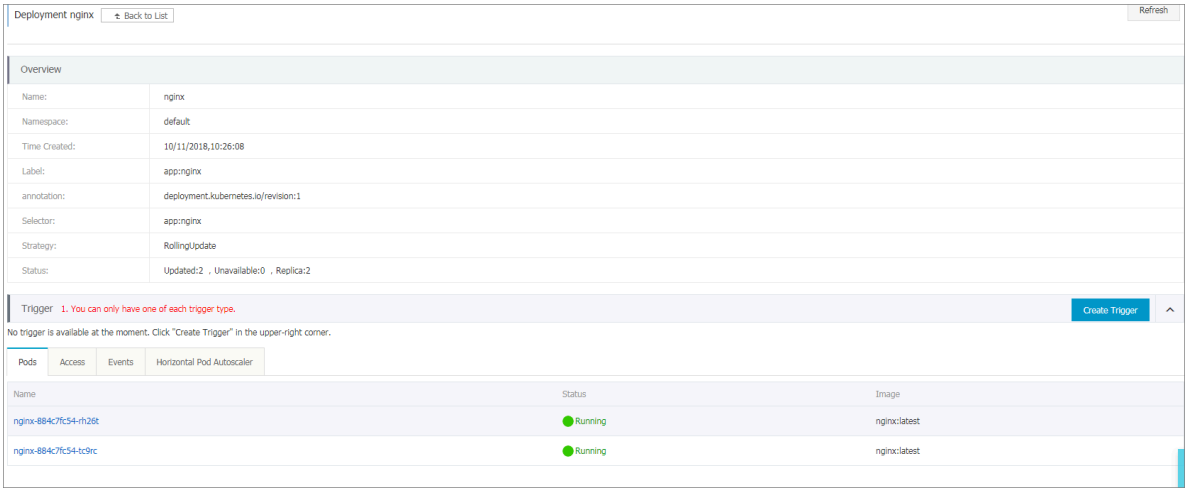
Note:
You can set pod anti affinity scheduling by using the same method as setting pod affinity scheduling. But the same scheduling rules have different

meanings for these two types of scheduling. You need to select appropriate scheduling rules as needed.

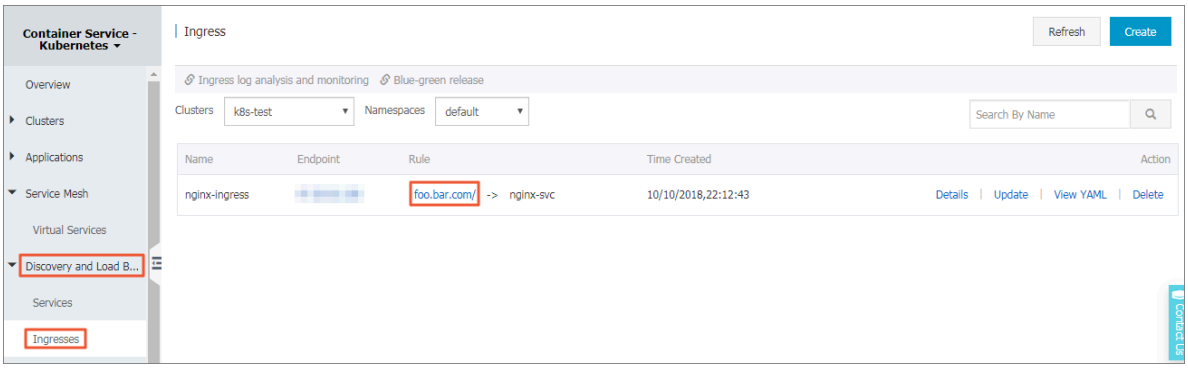
7. Click Create.
8. After you create the application, anew page is displayed by default to prompt that you have created the application and lists objects included in the application.You can click View detail to view the deployment details.



The nginx-deployment page is displayed by default.



9. Choose Discovery and Load Balancing > Ingress to verify that a rule is displayed in the Ingress list.



10. Access the test domain name in your browser to verify that you can visit the Nginx welcome page.

2.3 Use Yaml to create a statefull tomcat application

Prerequisites

- Create a Kubernetes cluster. For more information, see *Create a Kubernetes cluster*.
- You have created the resource objects involved in this example, such as storage volumes, config maps, secrets, node labels, and other resource objects.

Context

In a Container Service Kubernetes orchestration template, you must define resource objects required for running an application, and combine the resource objects into a complete application by using label selector.

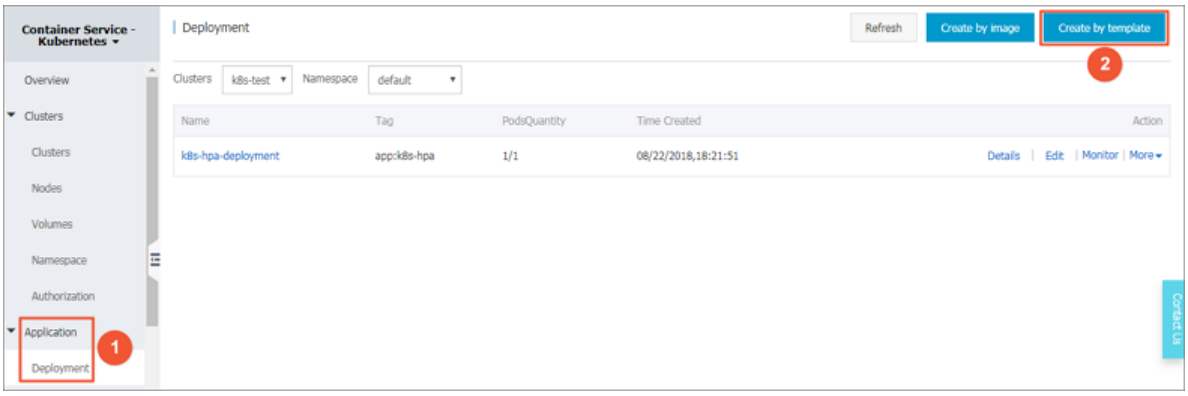
This example shows how to create a tomcat application by customizing a template in an orchestration template. The resource objects involved are as follows:

1. Storage volumes
2. Config maps
3. Secrets

4. Nodes specified by labels
5. Health check
6. Server/Load Balancer

Procedure

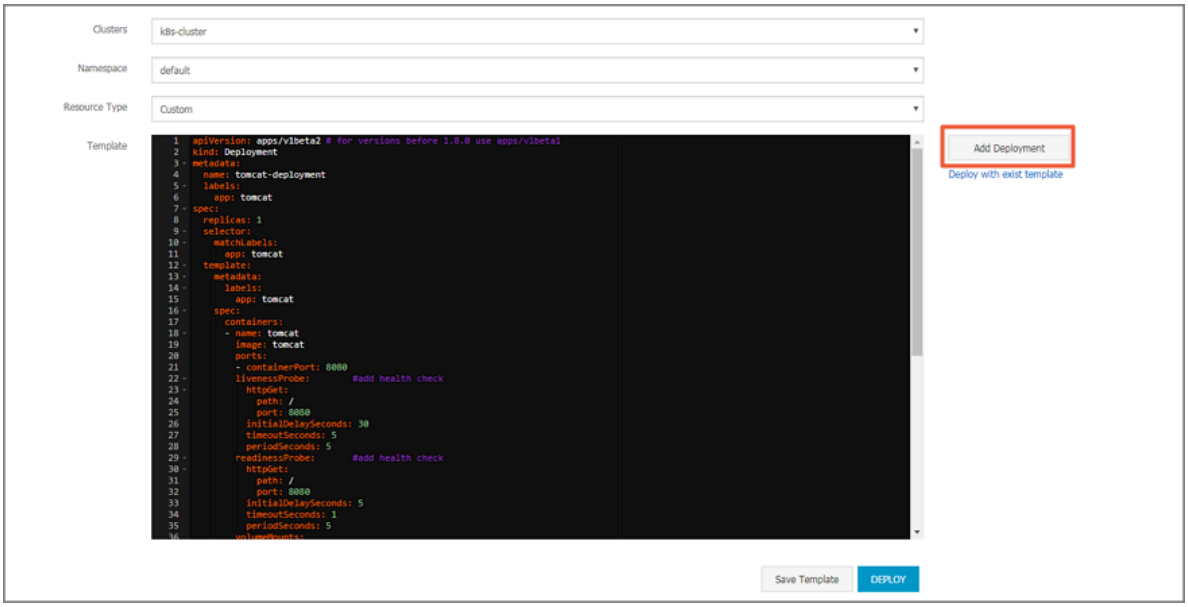
1. Log on to the *Container Service console*.
2. Under Kubernetes, click Application > Deployment in the left-side navigation pane.
3. On the Deployment page, click Create by Template in the upper-right corner.



4. Configure the template. Customize the template to create a tomcat application.
   - Clusters: Select a cluster. Resource objects are to be deployed in this cluster.
   - Namespace: Select a namespace to which resource objects belong. The default namespace is default. Except for the underlying computing resources such as nodes and persistent storage volumes, most of the resource objects must act on a namespace.
   - Resource Type: Alibaba Cloud Container Service provides multiple resource types of Kubernetes yaml sample templates, enabling you to quickly deploy

resource objects. You can write a template based on the format requirements of Kubernetes yaml orchestration to describe the resource type you want to define.

- Add Deployment: If you are not familiar with Kubernetes yaml orchestration, click Add Deployment to configure through the web interface.



a) First create a basic tomcat template on which this example shows how to configure resource objects in a yaml file.

```
apiVersion : apps / v1beta2 # for versions before 1 . 8
. 0 use apps / v1beta1
kind : Deployment
metadata :
  name : tomcat - deployment
  labels :
    app : tomcat
spec :
  replicas : 1
  selector :
    matchLabel s :
      app : tomcat
  template :
    metadata :
      labels :
        app : tomcat
    spec :
      containers :
     - name : tomcat
        image : tomcat # replace it with your exactly
& lt ; image_name : tags >
        ports :
       - containerP ort : 8080
```

b) Add a storage volume based on the basic template

Before adding a data volume, apply for a storage volume and create the storage volume claim. You can apply for a storage volume by using one of the following

methods: *Use Alibaba Cloud cloud disks*, *Use Alibaba Cloud NAS*, and *Use Alibaba Cloud OSS*.

Create the storage volume claim after applying for a storage volume. For more information, see *Create a persistent storage volume claim*. In this example, use Alibaba Cloud cloud disks as the storage volumes and use the cloud disk static storage volumes by using PV/PVC. The PVC name is pvc-yunpan-test.

```
apiVersion : apps / v1beta2 # for versions before 1 . 8
. 0 use apps / v1beta1
kind : Deployment
metadata :
  name : tomcat - deployment
  labels :
    app : tomcat
spec :
  replicas : 1
  selector :
    matchLabel s :
      app : tomcat
  template :
    metadata :
      labels :
        app : tomcat
    spec :
      containers :
     - name : tomcat
        image : tomcat # replace it with your exactly
& lt ; image_name : tags >
        ports :
       - containerP ort : 8080
        volumeMoun ts :                # add volume
       - name : pvc - yunpan - test
          mountPath : / data
      volumes :                        # add volume
     - name : pvc - yunpan - test
        persistent VolumeClai m :
          claimName : pvc - yunpan - test
```

c) Add a config map

Before using a config map, create a config map. For information about creating and using a config map, see *Use a config map in a pod*.

In this example, use the config map name and content in the following sample. The config map name is special-config. The config maps are `SPECIAL_LE VEL : very` and `SPECIAL_TY PE : charm`. Use config maps by means of environment variables.

```
apiVersion : apps / v1beta2 # for versions before 1 . 8
. 0 use apps / v1beta1
kind : Deployment
metadata :
```

```
  name : tomcat - deployment
  labels :
    app : tomcat
spec :
  replicas : 1
  selector :
    matchLabel s :
      app : tomcat
  template :
    metadata :
      labels :
        app : tomcat
    spec :
      containers :
     - name : tomcat
        image : tomcat # replace it with your exactly
& lt ; image_name : tags >
        ports :
       - containerP ort : 8080
        volumeMoun ts :
       - name : pvc - yunpan - test
         mountPath : / data
        env :
       - name : SPECIAL_LE VEL_KEY # add configmap
         valueFrom :
           configMapK eyRef :
             name : special - config
             key : SPECIAL_LE VEL
       - name : SPECIAL_TY PE_KEY # add configmap
         valueFrom :
           configMapK eyRef :
             name : special - config
             key : SPECIAL_TY PE
      volumes :
     - name : pvc - yunpan - test
        persistent VolumeClai m :
          claimName : pvc - yunpan - test
```

d) Add a secret

Create a secret first. For more information, see *Create a secret*.

```
apiVersion : apps / v1beta2 # for versions before 1 . 8
. 0 use apps / v1beta1
kind : Deployment
metadata :
  name : tomcat - deployment
  labels :
    app : tomcat
spec :
  replicas : 1
  selector :
    matchLabel s :
      app : tomcat
  template :
    metadata :
      labels :
        app : tomcat
    spec :
      containers :
     - name : tomcat
```

```
        image : tomcat # replace it with your exactly
& lt ; image_name : tags >
        ports :
      - containerP ort : 8080
        volumeMoun ts :
      - name : pvc - yunpan - test
        mountPath : / data
        env :
      - name : SPECIAL_LE VEL_KEY
        valueFrom :
          configMapK eyRef :
            name : special - config
            key : SPECIAL_LE VEL
      - name : SPECIAL_TY PE_KEY
        valueFrom :
          configMapK eyRef :
            name : special - config
            key : SPECIAL_TY PE
      - name : SECRET_USE RNAME # add secret
        valueFrom :
          secretKeyR ef :
            name : account
            key : username
      - name : SECRET_PAS SWORD # add secret
        valueFrom :
          secretKeyR ef :
            name : account
            key : password
      volumes :
     - name : pvc - yunpan - test
        persistent VolumeClai m :
          claimName : pvc - yunpan - test
```

e) Add a node

When you deploy an application, you can deploy the application on a node with the specific label. For instructions, see *Schedule a pod to a specified node*.

In this example, label a node with group:worker. When the application deployment succeeds, the application is deployed on the labeled node.

```
apiVersion : apps / v1beta2 # for versions before 1 . 8
. 0 use apps / v1beta1
kind : Deployment
metadata :
  name : tomcat - deployment
  labels :
    app : tomcat
spec :
  replicas : 1
  selector :
    matchLabel s :
      app : tomcat
  template :
    metadata :
      labels :
        app : tomcat
    spec :
      containers :
     - name : tomcat
```

```
        image :  tomcat
        ports :
      -  containerP  ort :  8080
        volumeMoun  ts :
      -  name :  pvc - yunpan - test
         mountPath : / data
        env :
      -  name :  SPECIAL_LE  VEL_KEY
         valueFrom :
           configMapK  eyRef :
             name :  special - config
             key :  SPECIAL_LE  VEL
      -  name :  SPECIAL_TY  PE_KEY
         valueFrom :
           configMapK  eyRef :
             name :  special - config
             key :  SPECIAL_TY  PE
      -  name :  SECRET_USE  RNAME
         valueFrom :
           secretKeyR  ef :
             name :  account
             key :  username
      -  name :  SECRET_PAS  SWORD
         valueFrom :
           secretKeyR  ef :
             name :  account
             key :  password
      volumes :
     -  name :  pvc - yunpan - test
        persistent  VolumeClai  m :
          claimName :  pvc - yunpan - test
      nodeSelect  or :       # add   node   selector
        group :  worker
```

f) **Add health check**

On Container Service platform, you can add health check for the application to check the health status of the application. Use liveness probes and readiness probes to detect the health status of a container in the application.

```
apiVersion :  apps / v1beta2  #  for   versions   before   1 . 8
. 0   use   apps / v1beta1
kind :  Deployment
metadata :
  name :  tomcat - deployment
  labels :
    app :  tomcat
spec :
  replicas :  1
  selector :
    matchLabel  s :
      app :  tomcat
  template :
    metadata :
      labels :
        app :  tomcat
    spec :
      containers :
     -  name :  tomcat
        image :  tomcat
```

```
        ports :
      -  containerP  ort :  8080
        livenessPr  obe :        # add   health   check
          httpGet :
            path : /
            port :  8080
          initialDel  aySeconds :  30
          timeoutSec  onds :  5
          periodSeco  nds :  5
        readinessP  robe :       # add   health   check
          httpGet :
            path : /
           -  port :  8080
          initialDel  aySeconds :  5
          timeoutSec  onds :  1
          periodSeco  nds :  5
        volumeMoun  ts :
      -  name :  pvc - yunpan - test
          mountPath : / data
        env :
      -  name :  SPECIAL_LE  VEL_KEY
          valueFrom :
            configMapK  eyRef :
              name :  special - config
              key :  SPECIAL_LE  VEL
      -  name :  SPECIAL_TY  PE_KEY
          valueFrom :
            configMapK  eyRef :
              name :  special - config
              key :  SPECIAL_TY  PE
      -  name :  SECRET_USE  RNAME
          valueFrom :
            secretKeyR  ef :
              name :  account
              key :  username
      -  name :  SECRET_PAS  SWORD
          valueFrom :
            secretKeyR  ef :
              name :  account
              key :  password
      volumes :
     -  name :  pvc - yunpan - test
        persistent  VolumeClai  m :
          claimName :  pvc - yunpan - test
      nodeSelect  or :
        group :  worker
```

g) Creates a LoadBalancer type service for the tomcat deployment.

To access applications deployed on Container Service from external networks such as the public network, you can expose the application by creating a LoadBalancer type service. A LoadBalancer type service creates Load Balancer

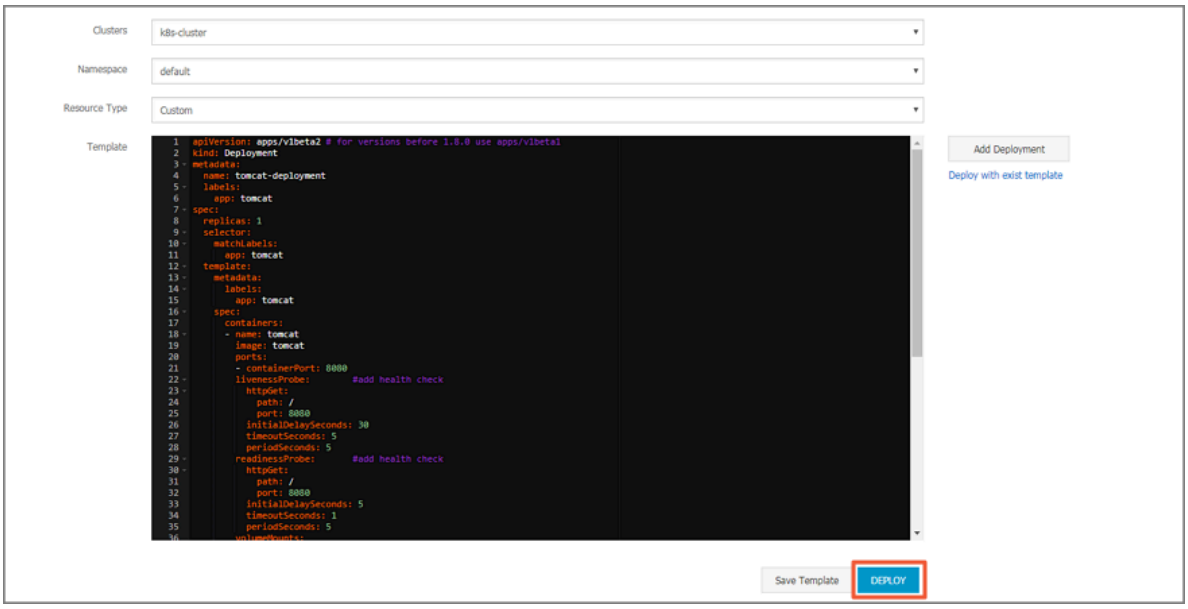
on Alibaba Cloud. You can access the application through the Load Balancer IP address.

For information about creating a service, see *Create a service*.

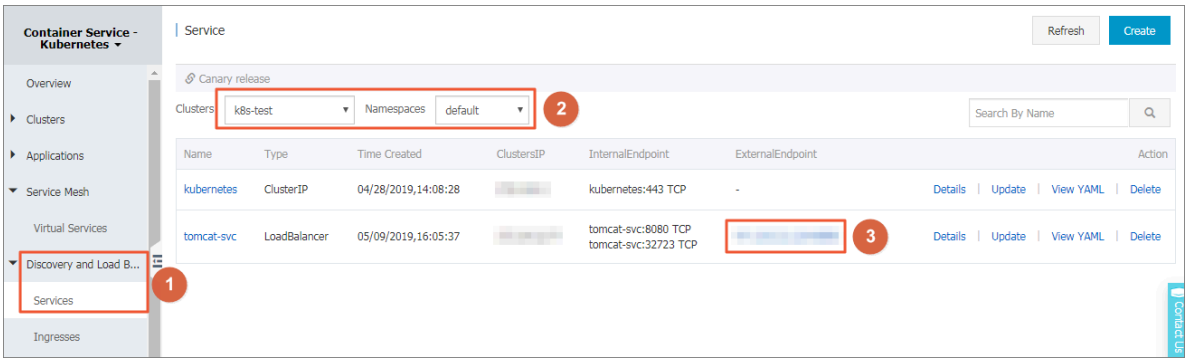
In this example, the orchestration template is as follows:

```
apiVersion :  v1
kind :  Service
metadata :
  name :  tomcat - svc
  labels :
    app :  tomcat - svc
spec :
  selector :
    app :  tomcat
  ports :
 -  protocol :  TCP
    port :  8080
    targetPort :  8080
  type :  LoadBalanc  er
```

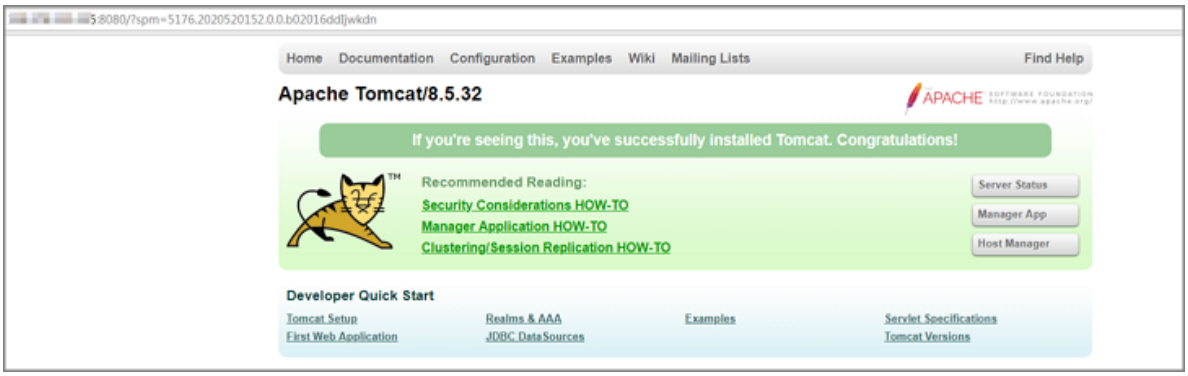
5. After the configuration is completed according to the application requirements, click Create.



6. After the deployment succeeds, choose Discovery and Load Balancing > Service in the left navigation pane, and select the tomcat-svc service to view its external endpoint.



7. Entering the external endpoint in the browser address bar, you can access the tomcat app welcome page .



What's next

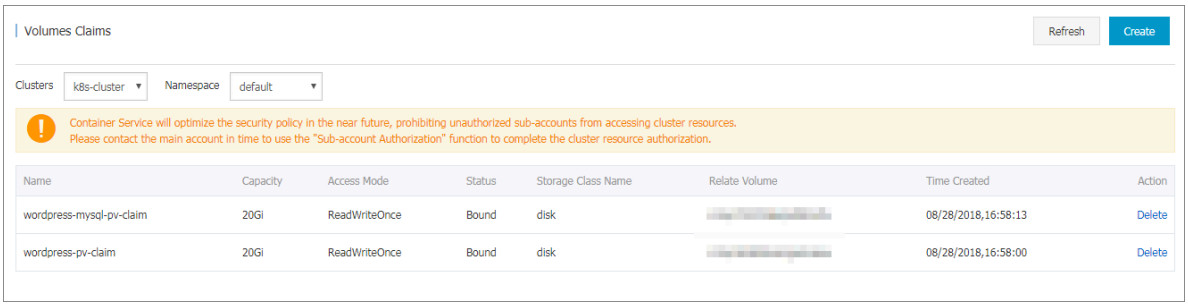
According to your orchestration template, you can explore features of the tomcat application in storage volumes, secrets, config maps, node scheduling, and health check.

2.4 Deploy dependency-based WordPress applications

Prerequisites

- A Kubernetes cluster is created. For more information, see *Create a Kubernetes cluster*.
- A Persistent Volume (PV) and Persistent Volume Claim (PVC) are created. For more information about how to create a PV, see *Use Alibaba Cloud cloud disk volumes*, *Use NAS file systems of Alibaba Cloud*, and *Use Alibaba Cloud OSS volumes*. For more information about how to create a PVC, see *Create a persistent volume claim*. Use

Alibaba Cloud disks as storage volumes. In the example, choose PV/PVC for the storage volume mount. Create two storage volume claims: wordpress-pv-claim and wordpress-mysql-pv-claim which are used in the wordpress yaml file and the wordpress-mysql yaml file respectively, to mount corresponding storage volumes.



Context

This example shows how to create dependency-based applications by customizing a template in a orchestration template.

The main components are:

- wordpress
- mysql

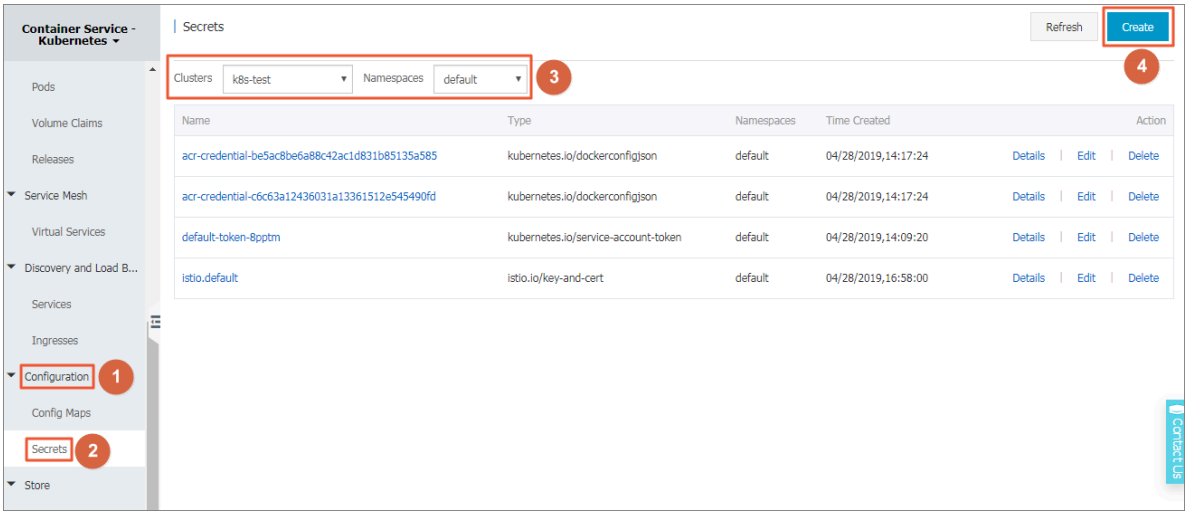
Resources involved:

- Storage volume
- Secret
- Service

Procedure

1. Log on to the *Container Service console*.
2. Use the prepared storage volume claims. Create two storage volume claims: wordpress-pv-claim and wordpress-mysql-pv-claim which are used in the wordpress yaml file and the wordpress-mysql yaml file respectively, to mount corresponding storage volumes.

3. In the left-side navigation pane, choose Configuration > Secrets, select the target cluster and namespace, and click Create in the upper-right corner. For more information, see *Create a Secret*.

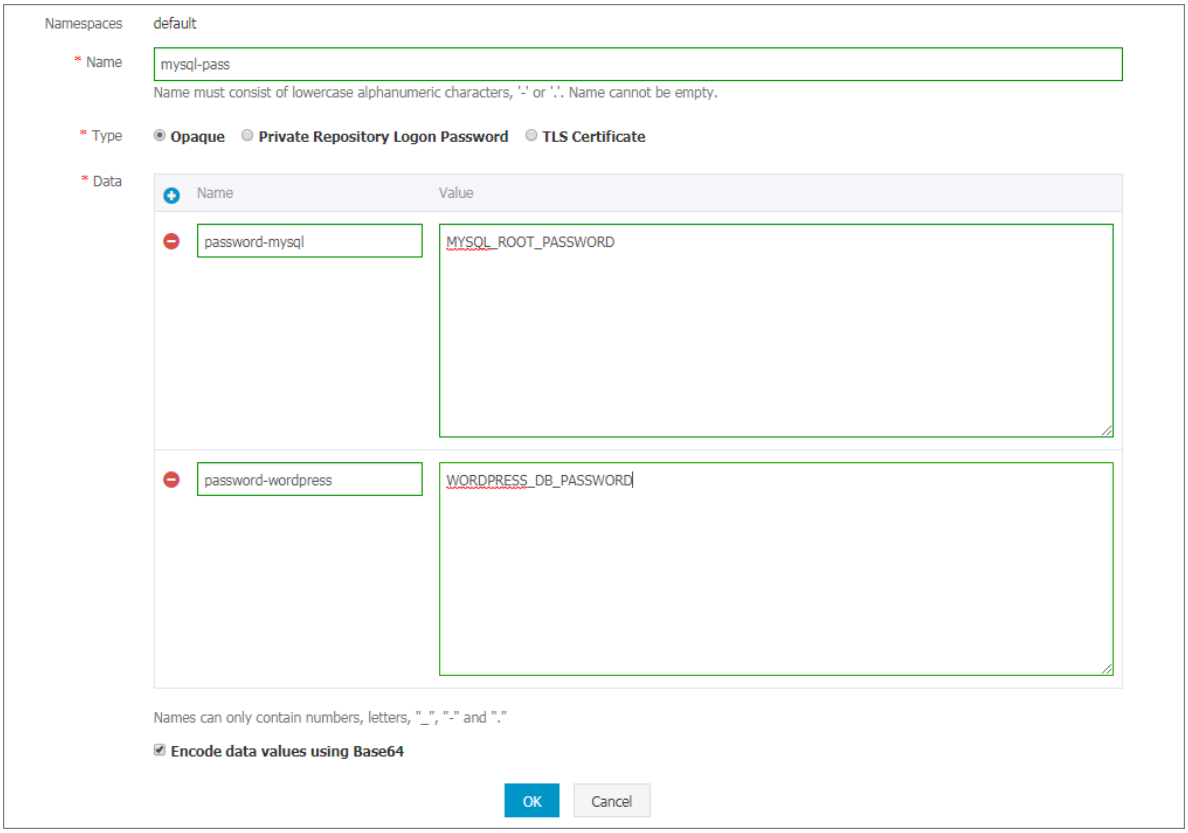


Note:

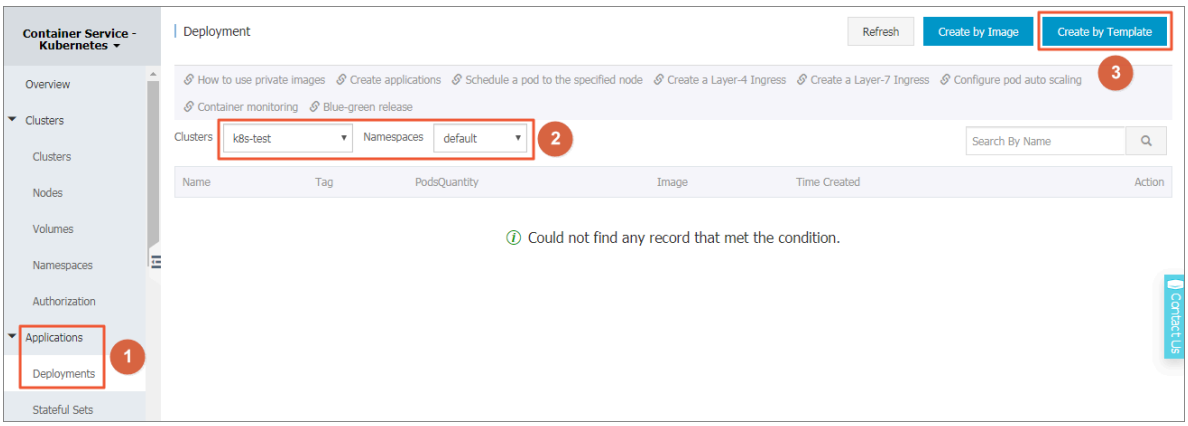
A user name and its password is required for creating and accessing a MySQL database. Therefore, you must create a secret to mange the user name and its password for a MySQL database.

Before using a secret, create a secret that needs to be encrypted. In this example, the MySQL root password is created as the secret, the secret name is set to `mysql`

`- pass` , and the Opaquesecret type is selected . This secret is used in the WordPress yaml file and wordpress-mysql yaml file.



4. In the left-side navigation pane, choose Applications > Deployments, and click Create by Template in the upper-right corner.



Select a cluster and namespace. The yaml file for creating WordPress deployment is as follows:

```
apiVersion :  apps / v1
kind :  Deployment
metadata :
  name :  wordpress
  labels :
    app :  wordpress
```

```
spec :
  selector :
    matchLabel  s :
      app :  wordpress
      tier :  frontend
  strategy :
    type :  Recreate
  template :
    metadata :
      labels :
        app :  wordpress
        tier :  frontend
    spec :
      containers :
     -  image :  wordpress : 4
        name :  wordpress
        env :
       -  name :  WORDPRESS_  DB_HOST
          value :  wordpress - mysql   # Use   the   name   to
  point   to   the   mysql   to   be   accessed .  The   name
correspond  s   to   the   mysql   service   name .
       -  name :  WORDPRESS_  DB_PASSWOR  D
          valueFrom :
            secretKeyR  ef :
              name :  mysql - pass
              key :  password - wordpress
        ports :
       -  containerP  ort :  80
          name :  wordpress
        volumeMoun  ts :
       -  name :  wordpress - pvc
          mountPath : / var / www / html
      volumes :
     -  name :  wordpress - pvc
        persistent  VolumeClai  m :
          claimName :  wordpress - pv - claim
```

The yaml file for creating mysql deployment is as follows:

```
apiVersion :  apps / v1
kind :  Deployment
metadata :
  name :  wordpress - mysql
  labels :
    app :  wordpress
spec :
  selector :
    matchLabel  s :
      app :  wordpress
      tier :  mysql
  strategy :
    type :  Recreate
  template :
    metadata :
      labels :
        app :  wordpress
        tier :  mysql
    spec :
      containers :
     -  image :  mysql : 5 . 6
        name :  mysql
        env :
       -  name :  MYSQL_ROOT  _PASSWORD
```

```
            valueFrom :
              secretKeyR  ef :
                name :  mysql - pass
                key :  password - mysql
          ports :
        -  containerP  ort :  3306
           name :  mysql
          volumeMoun  ts :
        -  name :  wordpress - mysql - pvc
           mountPath : / var / lib / mysql
       volumes :
      -  name :  wordpress - mysql - pvc
         persistent  VolumeClai  m :
           claimName :  wordpress - mysql - pv - claim
```

5. To enable external access for the WordPress, you need to create the access method exposed by the WordPress service. In this example, create the WordPress service of the LoadBalancer type so that Container Service automatically creates Alibaba Cloud Server Load Balancer to provide external access.

   Create a service named WordPress-mysql for the WordPress mysql so that the WordPress deploymet created on the WordPress mysql can be accessed. As the mysql is called only internally for the WordPress, you do not need to create a LoadBalancer type of service for it.

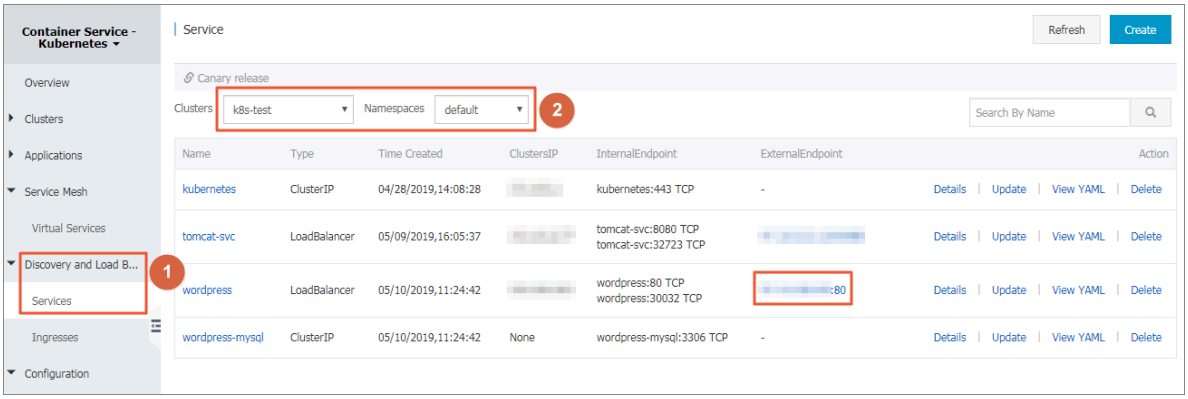
   For more information, see *Create a service*.

   The yaml file used to create WordPress and mysql service is as follows:

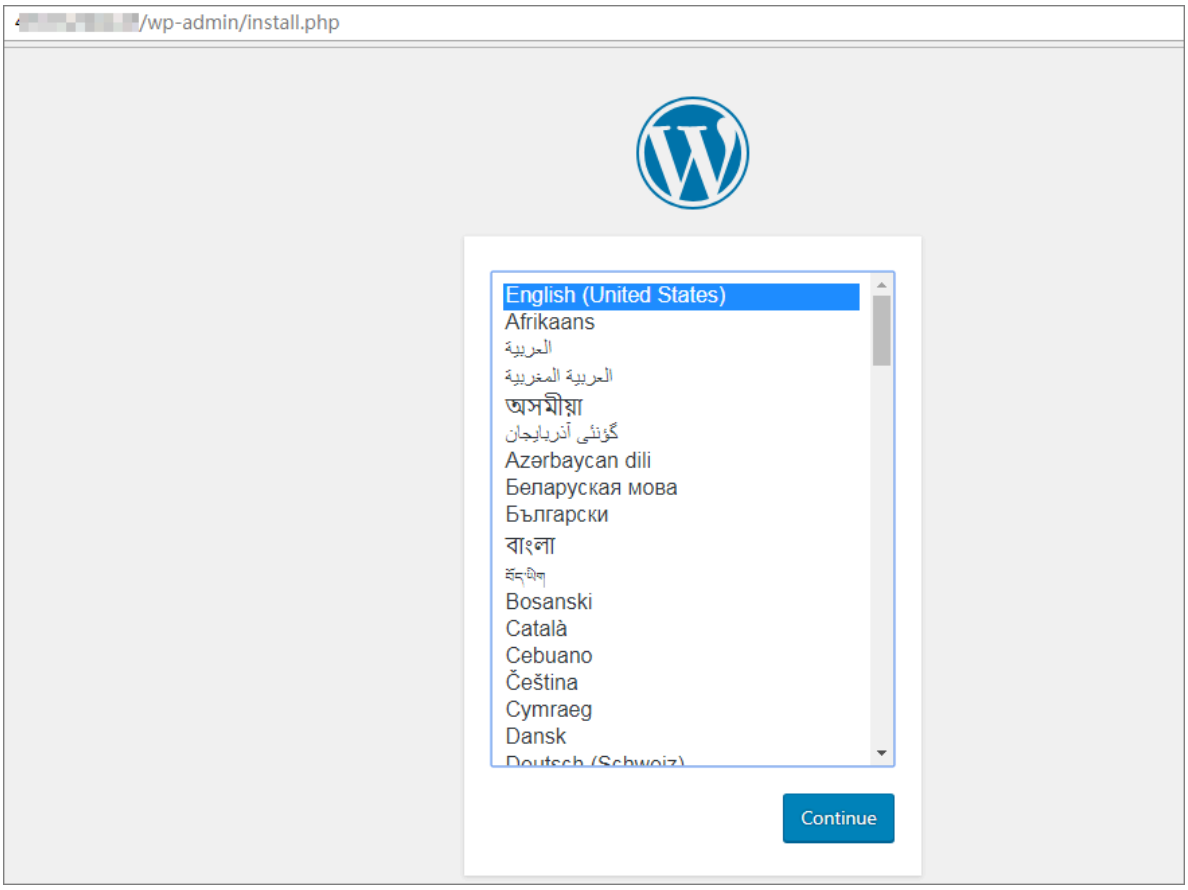
```
 apiVersion :  v1
 kind :  Service
 metadata :
   name :  wordpress
   labels :
     app :  wordpress
 spec :
   ports :
    - port :  80
   selector :
     app :  wordpress
     tier :  frontend
   type :  LoadBalanc  er
---
 apiVersion :  v1
 kind :  Service
 metadata :
   name :  wordpress - mysql
   labels :
     app :  wordpress
 spec :
   ports :
    -  port :  3306
   selector :
     app :  wordpress
     tier :  mysql
```

```
        clusterIP :  None
```

6. When the deployment is completed, choose Discovery and Load Balancing > Services in the left-side navigation pane. Locate the WordPress service and view its external endpoint.



7. Access the external endpoint of the WordPress service in a browser and you can access the WordPress application through the IP address provided by Server Load Balancer.



What's next

During the configuration of the WordPress application, you can log on to the application by using the password configured in the secret. In addition, the data

generated by the container to which the WordPress application belongs is saved in the data storage volume.

3 Advanced operations

3.1 Use Helm to deploy a microservice application

This document describes how to deploy a complex application to Alibaba Cloud Kubernetes Container Service. You can use different methods to deploy a SpringCloud application based on different combinations of infrastructure deployment and application deployment.

Deployment methods

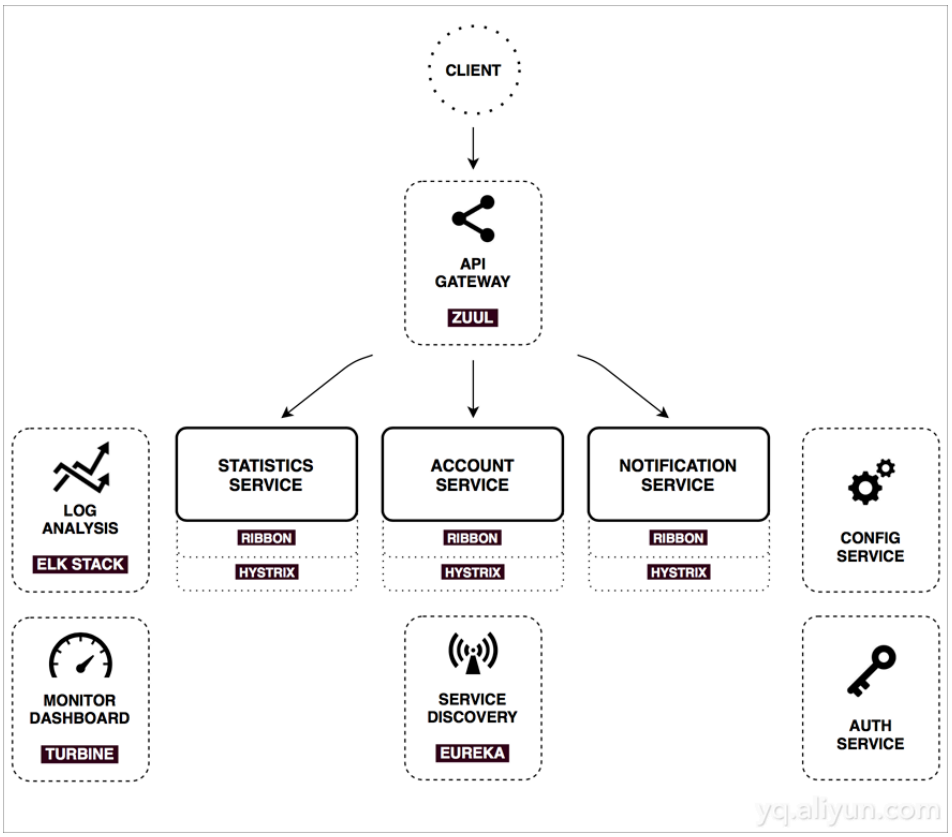
1. Deploy infrastructures such as Eureka and ConfigServer together with the application.
2. Deploy the application after building infrastructures on Container Service

Sample application PiggyMetrics

PiggyMetrics is a SpringCloud application project on GitHub with more than 3400 Stars. The project main body is deployed by using Docker Compose and contains complete source codes and well-built container images. It is a very good SpringCloud containerization example.

This project contains three business microservices: statistical service, account service, and notification service. Each service corresponds to a separate MongoDB

. The microservice architecture diagram(using the author's original diagram) is as follows:



SpringCloud basic components include the registry service (Eureka service registration), config service (configuration management), gateway (the API gateway, also the JavaScript Web Interface), monitor service (Hystrix Dashboard/Turbine) and more.

The deployment description file used in this article is on GitHub. If you are interested in the files, see the link: *https://github.com/binblee/PiggyMetrics/tree/master/charts*.

Scenario 1 Deploy all services with one-click deployment of helm

PiggyMetrics is deployed to a standalone device in the docker-compose YAML. To deploy PiggyMetrics to the Kubernetes environment, convert the docker-compose YAML to Kubernetes deployment YAML. *Kompose* can be used to convert a compose file to the Kubernetes deployment file through one click.

> Note:
> The *docker compose* template in PiggyMetrics is in version 2.1 that is not supported by kompose. Therefore, change the version of the docker compose file to version 2.

Additionally, remove the syntax that kompose does not support:

```
depends_on :
      config :
        condition :  service_he  althy   # condition   is   not
supported
```

Add Kubernetes server type annotation:

```
labels :
      kompose . service . type :  loadbalanc  er
```

For the changed compose file, see *https://github.com/binblee/PiggyMetrics/blob/master/charts/docker-compose.yml*.

Set the environmental variables required for PiggyMetrics deployment before executing kompose.

```
$  export   NOTIFICATI  ON_SERVICE  _PASSWORD = passw0rd
$  export   CONFIG_SER  VICE_PASSW  ORD = passw0rd
$  export   STATISTICS  _SERVICE_P  ASSWORD = passw0rd
$  export   ACCOUNT_SE  RVICE_PASS  WORD = passw0rd
$  export   MONGODB_PA  SSWORD = passw0rd
$  kompose   convert  - f   docker - compose . yml  - o   piggymetri
 cs  - c
```

The `- c` option of kompose can generate to the *helm chart* format directory result. Use a *helm* command to deploy all services.

```
charts  $  helm   install  - n   piggymetri  cs   piggymetri  cs /
```

You can see that the success message is output after deployment.

Try this configuration on your own Minikube or Alibaba Cloud Container Service for Kubernetes: *https://www.aliyun.com/product/kubernetes*. After the deployment is completed, go to the service list page, you can see all services, and the access addresses and port numbers exposed by the corresponding LoadBalancer services.

Name	Type	Time Created	ClustersIP	InternalEndpoint	ExternalEndpoint	Action
ack-springcloud-eureka-default-ack-springcloud-eureka-svc	LoadBalancer	09/04/2018,14:01:16		ack-springcloud-eureka-default-ack-springcloud-eureka-svc:8761 TCP ack-springcloud-eureka-default-ack-springcloud-eureka-svc:30689 TCP		Details | Update | View YAML | Delete
ack-springcloud-eureka-default-ack-springcloud-eureka-svc-0	ClusterIP	09/04/2018,14:01:16		ack-springcloud-eureka-default-ack-springcloud-eureka-svc-0:8761 TCP	-	Details | Update | View YAML | Delete
ack-springcloud-eureka-default-ack-springcloud-eureka-svc-1	ClusterIP	09/04/2018,14:01:16		ack-springcloud-eureka-default-ack-springcloud-eureka-svc-1:8761 TCP	-	Details | Update | View YAML | Delete
batchrelease-01-batch-svc	LoadBalancer	09/03/2018,16:00:56		batchrelease-01-batch-svc:80 TCP batchrelease-01-batch-svc:32226 TCP		Details | Update | View YAML | Delete
kubernetes	ClusterIP	08/22/2018,17:28:51		kubernetes:443 TCP	-	Details | Update | View YAML | Delete
registry	LoadBalancer	09/04/2018,19:46:48	172.19.8.217	registry:8761 TCP registry:32394 TCP		Details | Update | View YAML | Delete

Contact Us

You can access the PiggyMetrics interface by clicking registry service.

Piggymetrics is a personal financial service that allows you to express beautiful reports after entering your income and expenditure.

Visit the registry service to see all the services registered to the Eureka server.

DS Replicas

registry

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
ACCOUNT-SERVICE	n/a (1)	(1)	UP (1) - account-service-7fd4976bfc-2dq9p:account-service:6000
AUTH-SERVICE	n/a (1)	(1)	UP (1) - auth-service-7bdb99b5dc-kt7kd:auth-service:5000
GATEWAY	n/a (1)	(1)	UP (1) - gateway-77857d9c49-nhgsx:gateway:4000
NOTIFICATION-SERVICE	n/a (1)	(1)	UP (1) - notification-service-5d5859d7-w5hlz:notification-service:8000
STATISTICS-SERVICE	n/a (1)	(1)	UP (1) - statistics-service-685fb8dc9f-6j7lv:statistics-service:7000

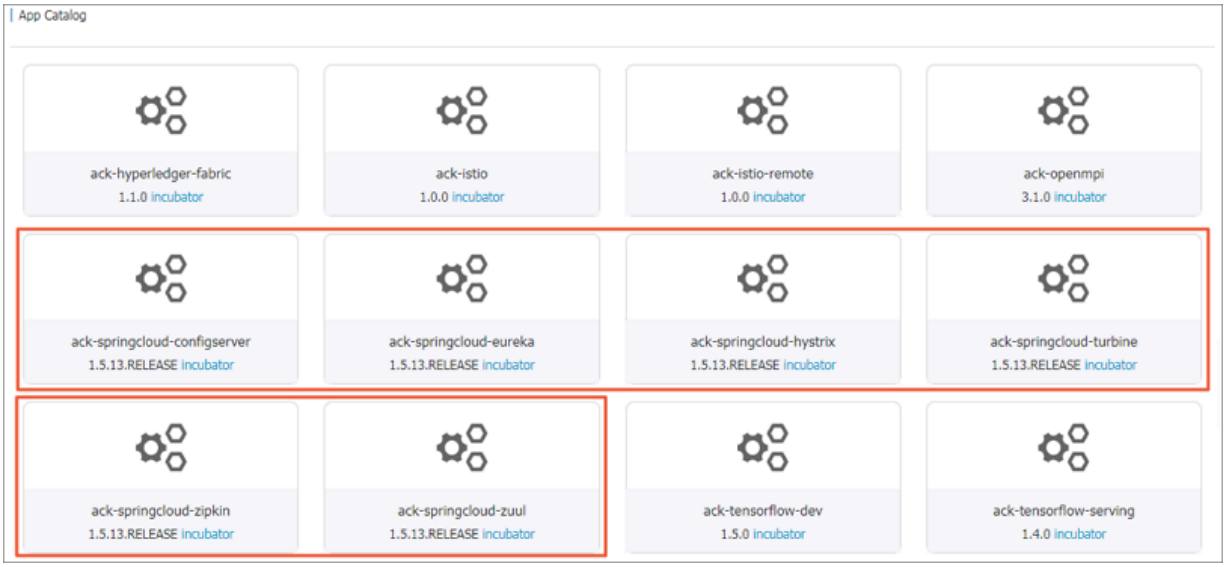
Remove PiggyMetrics to prepare for the next experiment:

```
charts $ helm delete -- purge piggymetri cs
release " piggymetri cs " deleted
```

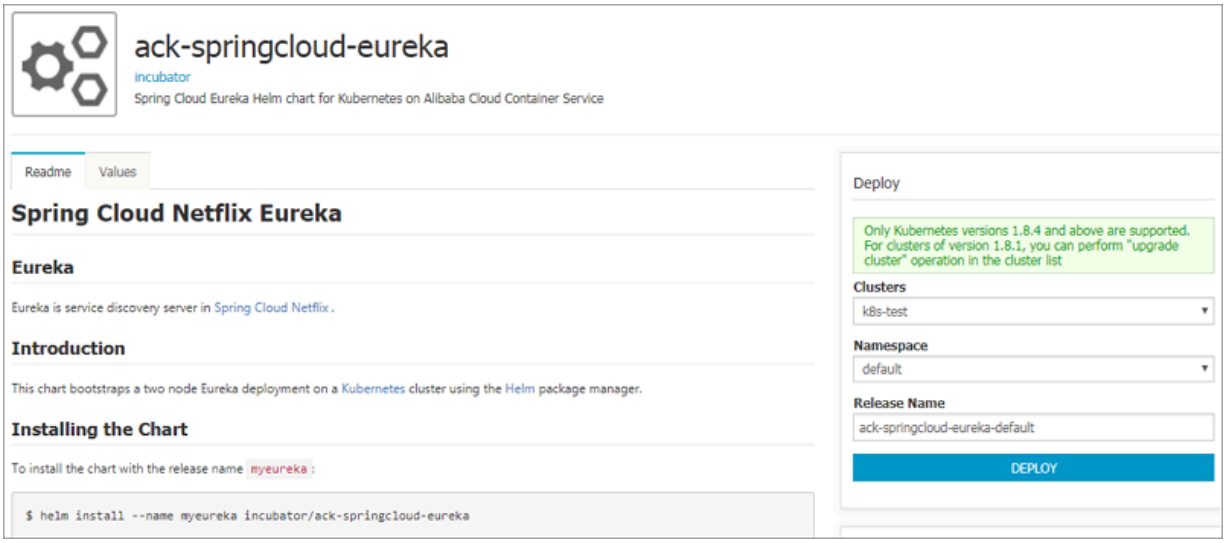
Scenario 2 Deploy the application to an existing SpringCloud basic component environment

The preceding Scenario 1 shows how to deploy all basic components (Eureka, Zuul, ConfigServer, and Hystrix Dashboard) and service applications (gateway, notification , and statistics) with one helm chart. In practice, the more common situation is that basic components such as Eureka already exist in the cluster. You only need to deploy , upgrade, and maintain your business applications.

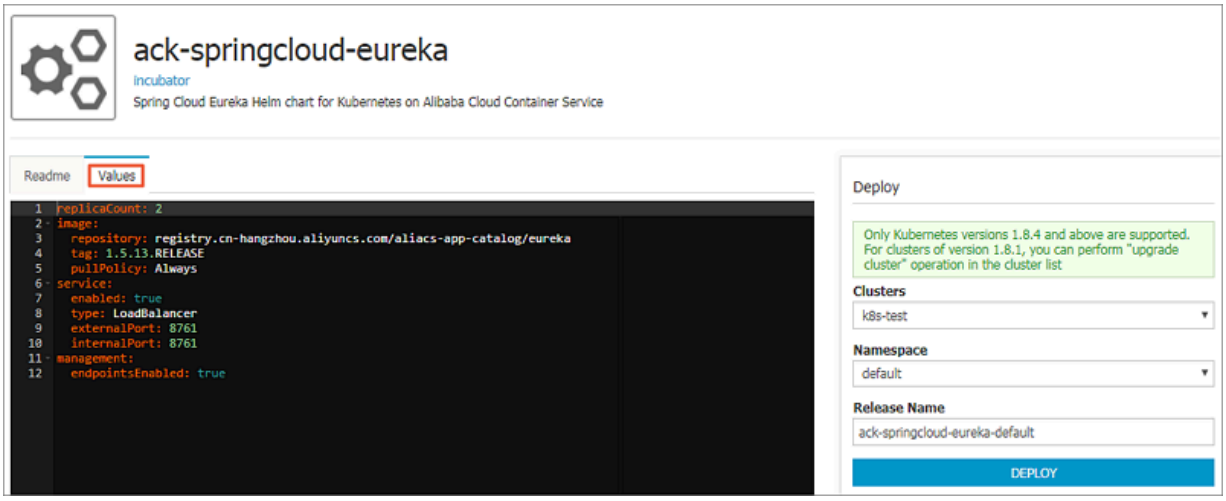
To use the Spring Cloud components in Alibaba Cloud Container Service for Kubernetes, choose Store > App Catalog.



Deploy Eureka services on the App Catalog. Click the ack-springcloud-eureka component.

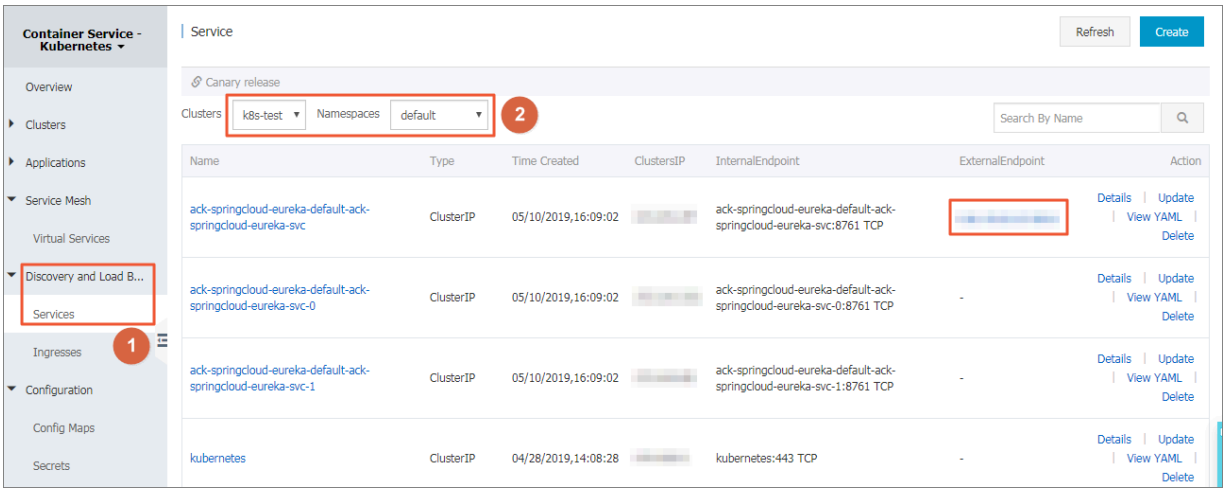


To view or change the configuration, click Values.



Click DEPLOY.

In the left-side navigation pane, choose Discovery and Load Balancing > Services. Then, select the target cluster and namespace. You can view that EurekaServer has two examples. The exposed service address is `ack-springcloud-eureka-default-ack-springcloud-eureka-svc`.



In PiggyMetrics, the EUREKA service that all containers automatically access on startup is called `registry`. In general, the EUREKA service name can be passed as a parameter in the image. In this experiment, do not change any codes or images. Therefore, take another measure, namely, expose Eureka to another service called registry.

Use Container Service to deploy the following yaml files.

Note:

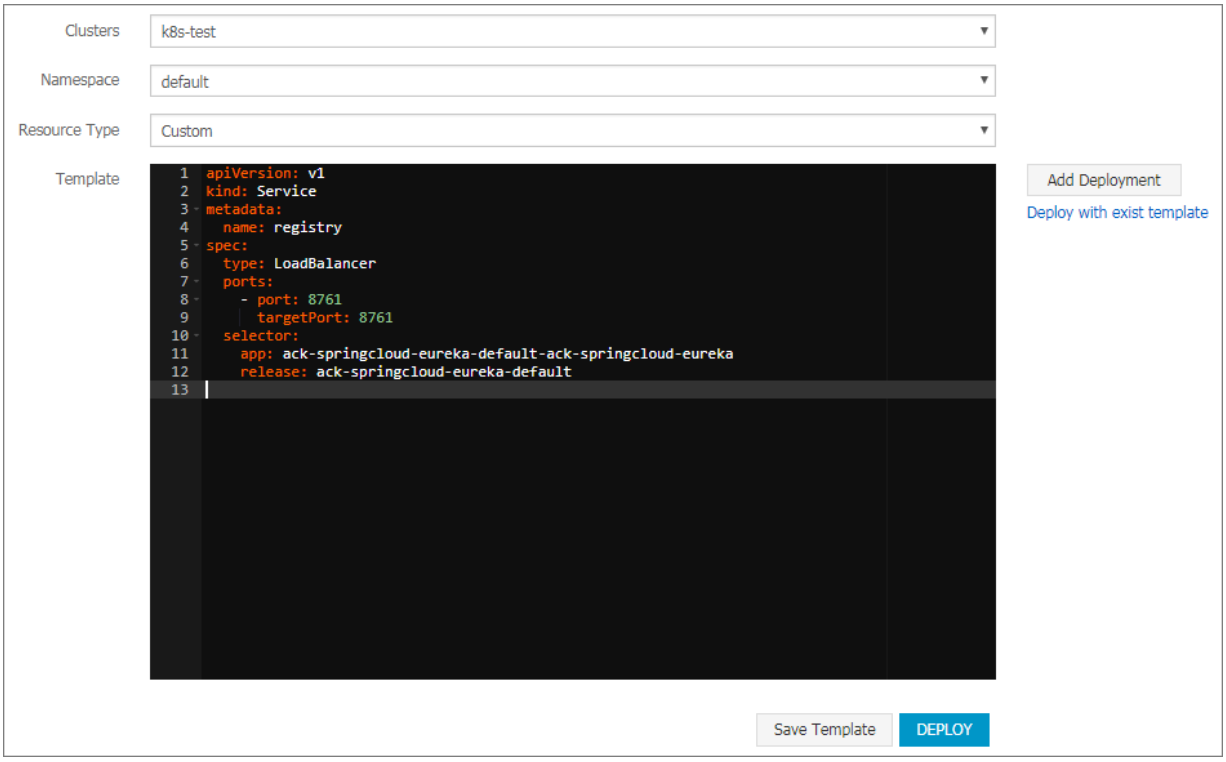
If you change the release name during App Catalog deployment, make the same changes to the followings.

```
apiVersion : v1
kind : Service
metadata :
  name : registry
spec :
  type : LoadBalanc er
  ports :
   - port : 8761
     targetPort : 8761
  selector :
    app : ack - springclou d - eureka - default - ack - springclou
d - eureka
    release : ack - springclou d - eureka - default
```

You can use the kubectl command line to create the service:

```
$ kubectl apply - f registry - svc . yml
```

You can also do this through the console interface:



In the left-side navigation pane, choose Discovery and Load Balancing > Services to verify that the registry is created.

Service List

Refresh

Clusters k8s-test Namespace default

Name	Type	Time Created	ClustersIP	InternalEndpoint	ExternalEndpoint	
ack-springcloud-eureka-default-ack-springcloud-eureka-svc	LoadBalancer	09/04/2018,14:01:16		ack-springcloud-eureka-default-ack-springcloud-eureka-svc:8761 TCP ack-springcloud-eureka-default-ack-springcloud-eureka-svc:30689 TCP		Details View YAML
ack-springcloud-eureka-default-ack-springcloud-eureka-svc-0	ClusterIP	09/04/2018,14:01:16		ack-springcloud-eureka-default-ack-springcloud-eureka-svc-0:8761 TCP	-	Details View YAML
ack-springcloud-eureka-default-ack-springcloud-eureka-svc-1	ClusterIP	09/04/2018,14:01:16		ack-springcloud-eureka-default-ack-springcloud-eureka-svc-1:8761 TCP	-	Details View YAML
batchrelease-01-batch-svc	LoadBalancer	09/03/2018,16:00:56		batchrelease-01-batch-svc:80 TCP batchrelease-01-batch-svc:32226 TCP		Details View YAML
kubernetes	ClusterIP	08/22/2018,17:28:51		kubernetes:443 TCP	-	Details View YAML
registry	LoadBalancer	09/04/2018,19:46:48		registry:8761 TCP registry:32394 TCP		Details View YAML

Copy the helm chart directory of PiggyMetrics to a new directory, piggymetrics-no-eureka. Delete the following two files:

```
templates / registry - deployment . yaml
templates / registry - service . yaml
```

These two files are the yaml files used to deploy Eureka deployment and svc, respectively. As you have used the App Catalog to successfully deploy a new registry service as the basic SpringCloud component, you do not have to repeat the deployment.

Execute the helm command to deploy PiggyMetrics again.

```
$ helm install - n piggymetri cs piggymetri cs - no - eureka
/
```

After all services start, access the registry service and you can see that all PiggyMetrics services are properly registered with EurekaServer.

DS Replicas

ack-springcloud-eureka-default-ack-springcloud-eureka-headless-svc-1.default.svc.cluster.local

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
ACCOUNT-SERVICE	n/a (1)	(1)	UP (1) - account-service-7fd4976bfc-4rmmj:account-service:6000
AUTH-SERVICE	n/a (1)	(1)	UP (1) - auth-service-7bdb99b5dc-kfnsv:auth-service:5000
GATEWAY	n/a (1)	(1)	UP (1) - gateway-77857d9c49-dgz6j:gateway:4000
NOTIFICATION-SERVICE	n/a (1)	(1)	UP (1) - notification-service-5d5859d7-sc6wb:notification-service:8000
STATISTICS-SERVICE	n/a (1)	(1)	UP (1) - statistics-service-685fb8dc9f-9kfxh:statistics-service:7000

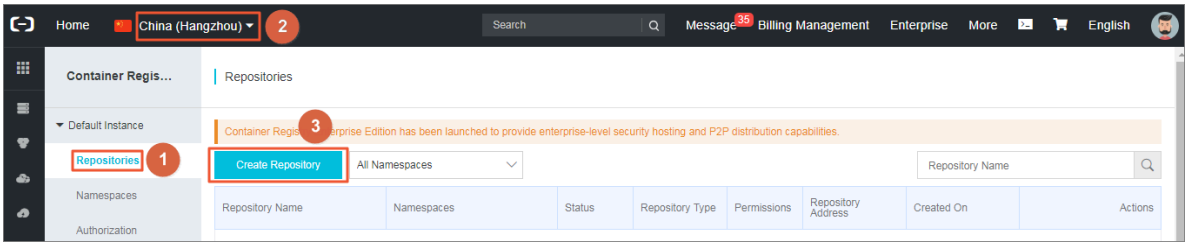
The PiggyMetrics application has been deployed to the environment with EurekaServer. Access `GATEWAY` to see the familiar login interface.

3.2 Use a private image repository to create an application

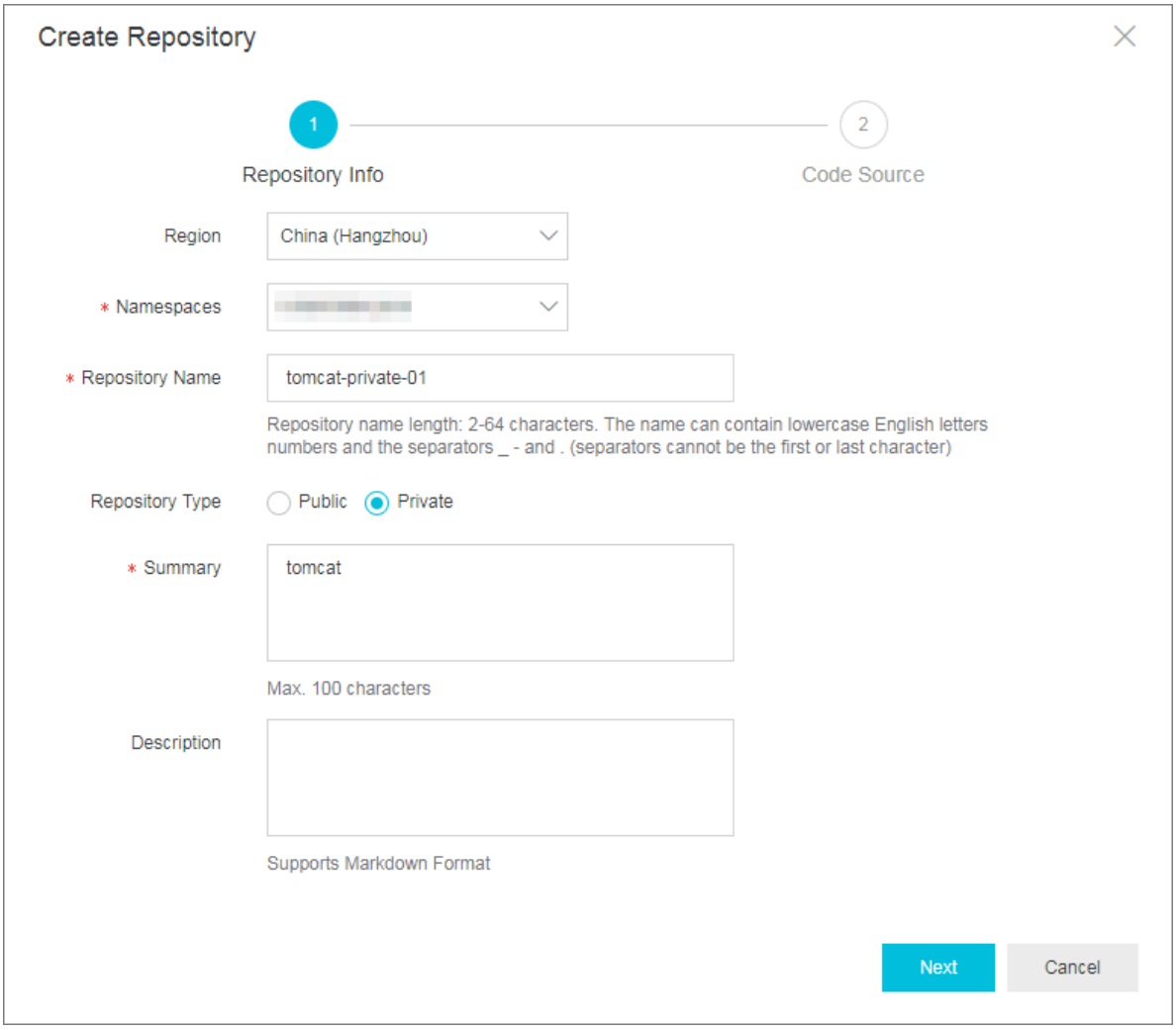
In many scenarios, an image in a private image repository is used for deploying an application. In this document, use Alibaba Cloud image repository service to create a private image repository, and create an application that uses this private image repository.

Step 1: Create a private image repository

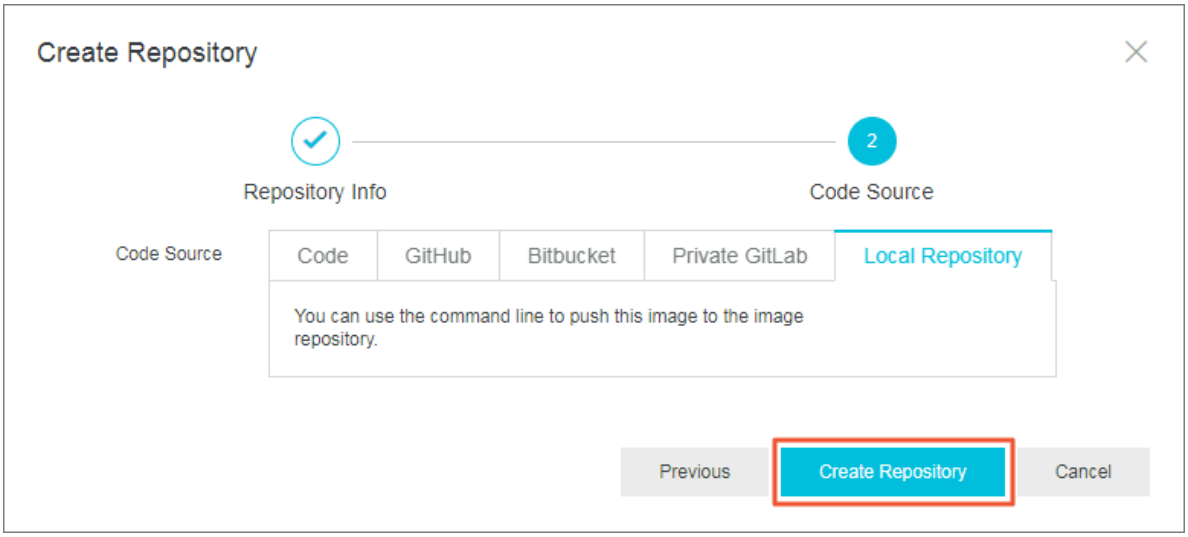
1. Log on to the *Container Registry console*.
2. In the left-side navigation pane, click Repositories. Select the target region, and then click Create Repository.



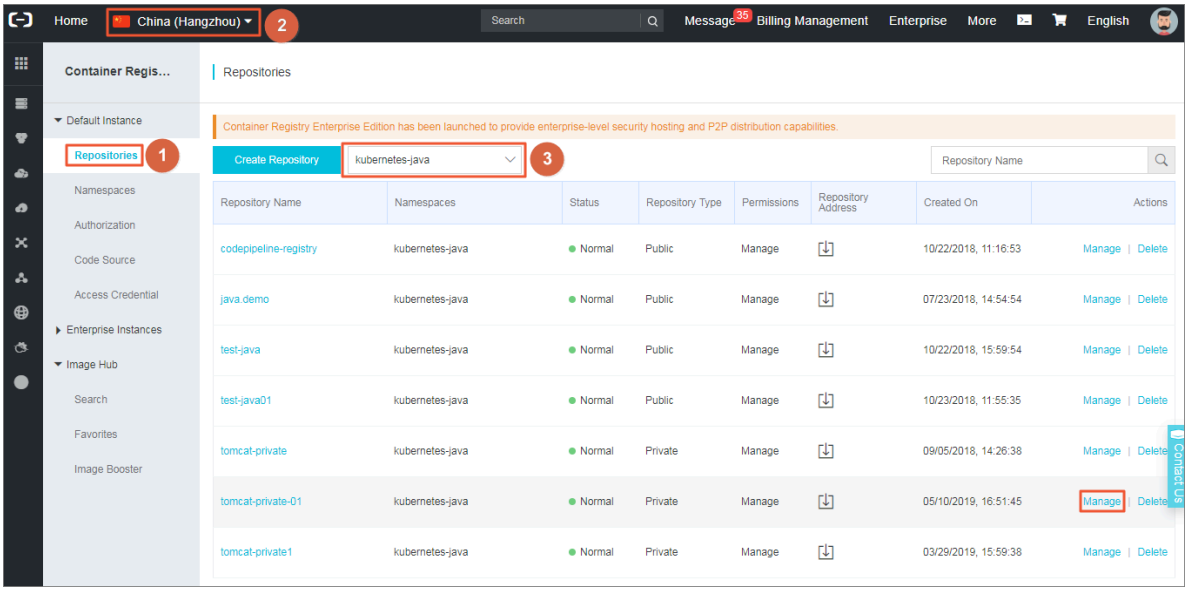
3. In the displayed dialog box, set the following parameters for the image repository: namespace, repository name, repository type, and summary. Then, click Next.



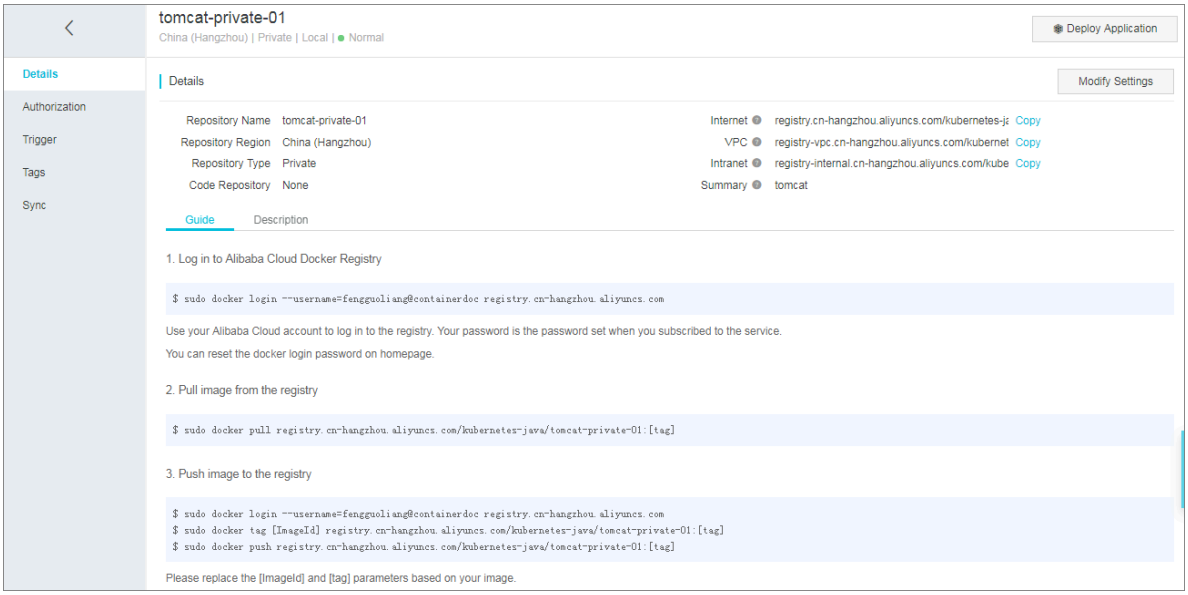
4. Select Local Repository, and then click Create Repository.



5. On the Repositories page, select the target region and namespace, find the image repository that you have created, and then click Manage in the Actions column.



6. On the Details page, view how to use this private image repository.



7. Log on to this image repository in the Linux operating system, and upload a local image to the private image repository.

```
$ sudo docker login --username=abc@aliyun.com
Password
## The password for logging on to the image repository.
Login Succeed

$ docker images
# A Tomcat image is used.
REPOSITORY
TAG IMAGE ID
CREATED SIZE
```

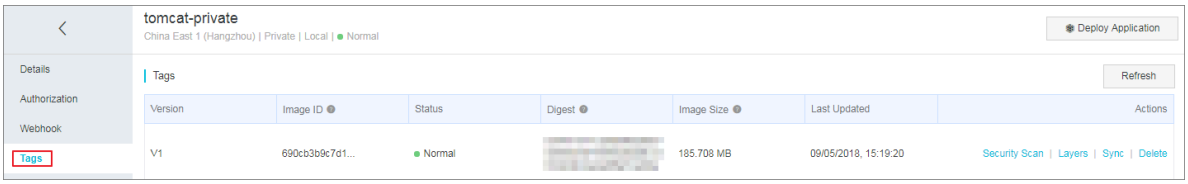
```
      tomcat
                 latest                2d43521f2b  1a          6
days   ago            463MB

     $  sudo   docker   tag  [ ImageId ]  registry . cn - hangzhou
. aliyuncs . com / kubernetes - java / tomcat - private :[ image
version   number ]
     $  sudo   docker   push   registry . cn - hangzhou . aliyuncs
. com / kubernetes - java / tomcat - private :[ image   version
number ]

      The   push   refers   to   a   repository  [ registry . cn -
hangzhou . aliyuncs . com / kubernetes - java / tomcat - private ]
      9072c7b03a  1b :  Pushed
      f9701cf47c  58 :  Pushed
      365c8156ff  79 :  Pushed
      2de08d97c2  ed :  Pushed
      6b09c39b2b  33 :  Pushed
      4172ffa172  a6 :  Pushed
      1dccf0da88  f3 :  Pushed
      d2070b1403  3b :  Pushed
      63dcf81c7c  a7 :  Pushed
      ce6466f43b  11 :  Pushed
      719d45669b  35 :  Pushed
      3b10514a95  be :  Pushed
      V1 :  digest :  sha256 : cded14cf64  697961078a  edfdf870e7
04a5227018  8c8194b6f7  0c778a8289  d87e   size :  2836
```

8. Return to the Details page of the image repository, and click Tags in the left navigation pane to verify that the local image has been uploaded.

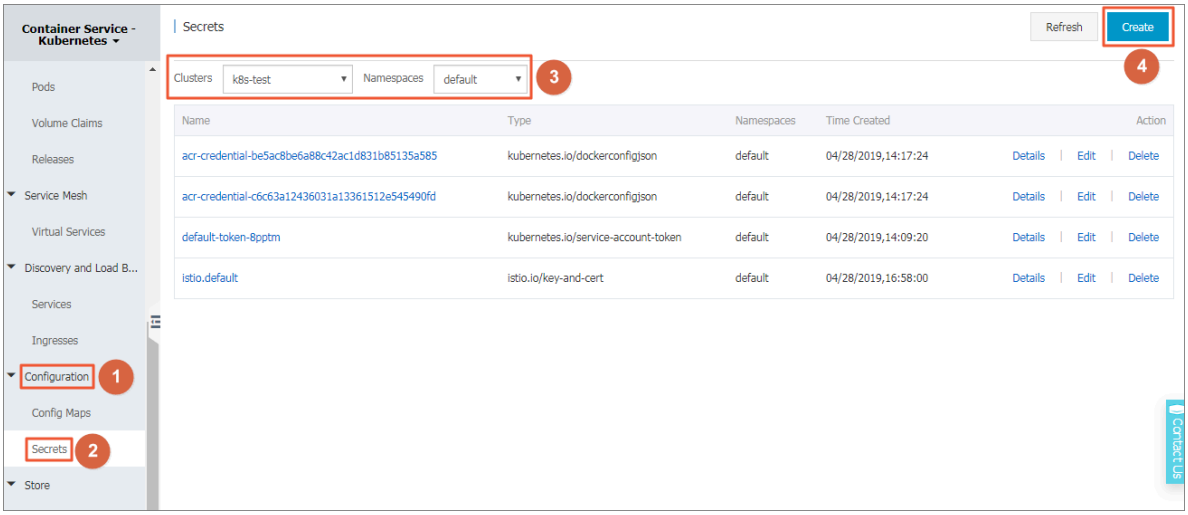
> **Note:**
> On the page displays the information related to the image, such as the image version number and image ID.



Step 2: Create a secret for logging on to a private image repository

1. Log on to the *Container Service console*.
2. In the left-side navigation pane under Container Service-Kubernetes, choose Configuration > Secrets.

3. Select the target cluster and namespace, and then click Create in the upper-right corner.

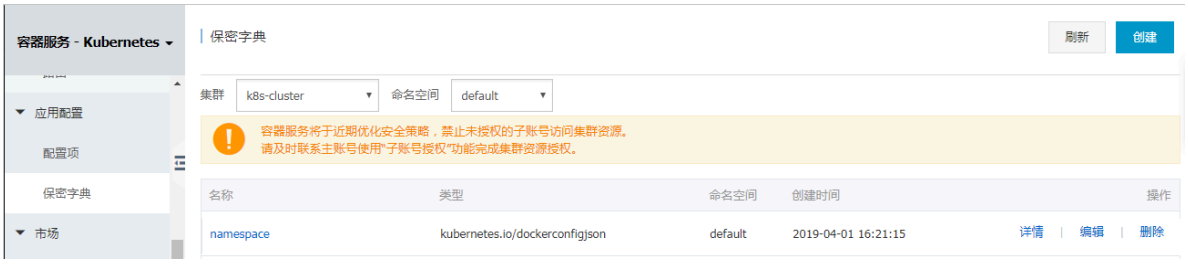


4. 配置新的保密字典。

Note:

在这里不能用容器服务控制台上的保密字典进行secret的创建。

5. 默认返回保密字典页面，您可看到新建的密钥出现在列表中。



您也可以## kubectl ## Kubernetes ##创建私有镜像仓库登录密钥类型的密钥。

步骤三 通过私有镜像仓库创建应用

1. 登录 #########。

2. 在 Kubernetes 菜单下，单击左侧导航栏中的应用 > 无状态，进入无状态（Deployment）页面。
3. 选择所需的集群和命名空间，单击右上角的使用模板创建。

4. 示例模板选择自定义，并将以下内容复制到模板中，单击创建。

```
apiVersion : apps / v1beta2 # for versions before 1 . 8 .
0 use apps / v1beta1
kind : Deployment
metadata :
  name : private - image
  nameSpace : default
  labels :
    app : private - image
spec :
  replicas : 1
  selector :
    matchLabel s :
      app : private - image
  template :
    metadata :
      labels :
        app : private - image
    spec :
      containers :
     - name : private - image
        image : registry . cn - hangzhou . aliyuncs . com / xxx /
tomcat - private : latest
        ports :
       - containerP ort : 8080
      imagePullS ecrets :
     - name : regsecret
```

> **Note:**
> 需要将镜像url更换成自己的私有镜像地址

更多内容请参考kubernetes官方文档######。

Step 2 Create a docker-registry secret

When using Kubernetes to create an application by pulling a private image, pass the identity authentication information of the private image repository to Kubernetes through a docker-registry secret.

Create a docker-registry secret as follows:

```
kubectl create secret docker - registry regsecret -- docker
- server = registry - internal . cn - hangzhou . aliyuncs . com --
```

```
docker - username = abc @ aliyun . com  -- docker - password = xxxxxx
 -- docker - email = abc @ aliyun . com
```

where:

- --regsecret: Specifies the secret key name and the name is customizable.
- --docker-server: Specifies the Docker repository address.
- --docker-username: Specifies the user name of the Docker repository.
- --docker-password: Specifies the Docker repository login password, namely, the independent login password of the container image registry.
- --docker-email: Specifies the email address.

Note:
You cannot use the secrets on the Container Service console to create secrets.

To pull an image successfully, add the secret parameter to the yml file.

```
containers :
    -  name :  foo
     image :  registry - internal . cn - hangzhou . aliyuncs . com /
abc / test : 1 . 0
     imagePullS  ecrets :
    -  name :  regsecret
```

where:

- imagePullSecrets declares that a secret key must be specified when you pull the image.
- regsecret must be the same as the preceding secret key name.
- The docker repository name in the image must be the same as that in the -- docker-server.

Step 3 Use a private image repository to create an application

The orchestration is as follows:

```
apiVersion :  apps / v1beta2  #  for   versions   before   1 . 8 . 0
  use   apps / v1beta1
    kind :  Deployment
    metadata :
    name :  private - image
    nameSpace :  default
    labels :
    app :  private - image
    spec :
    replicas :  1
    selector :
    matchLabel  s :
    app :  private - image
    template :
```

```
    metadata :
    labels :
    app :  private - image
    spec :
    containers :
   -  name :  private - image
    image :  registry . cn - hangzhou . aliyuncs . com / xxx / tomcat
- private : latest
    ports :
   -  containerP  ort :  8080
    imagePullS  ecrets :
   -  name :  regsecret
```

For more information, see the official Kubernetes documentation *Use a Private Registry*.