Alibaba Cloud
Alibaba Cloud Container Service for Kubernetes

Quick Start

Issue: 20190921

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

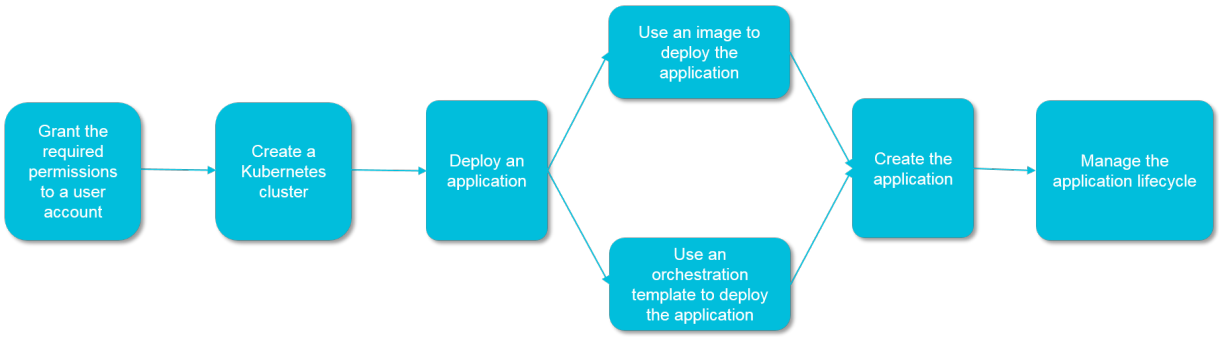
Style	Description	Example
	This warning information indicates a situation that will cause major system changes, faults, physical injuries, and other adverse results.	Danger: Resetting will result in the loss of user configuration data.
	This warning information indicates a situation that may cause major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

This topic describes the workflow of how to create an application in the Container Service console. This workflow involves the steps required to create an application.



To create an application, follow these steps:

1. Use a RAM role to grant the corresponding permissions to a user account

   For more information, see #unique_4.

2. Create a Kubernetes cluster

   Select a cluster type based on your needs. For more information, see #unique_5, #unique_6, and #unique_7.

3. Use an image or orchestration template to create an application

   Use an existing image or orchestration template, or create an image or orchestration template to create an application. For more information, see #unique_8 and #unique_9.

   > **Note:**
   > If you want to create an application that contains services supported by multiple images, we recommend that you use an orchestration template to create the application.

4. View the application status, and the services and pods of the application

   For more information, see #unique_10 and #unique_11.

2 Basic operations

2.1 Create a Kubernetes cluster

This topic describes how to use the Container Service console to quickly create a Kubernetes cluster. That is, you must retain the default settings for most of the parameters.

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 #unique_14.

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

Context

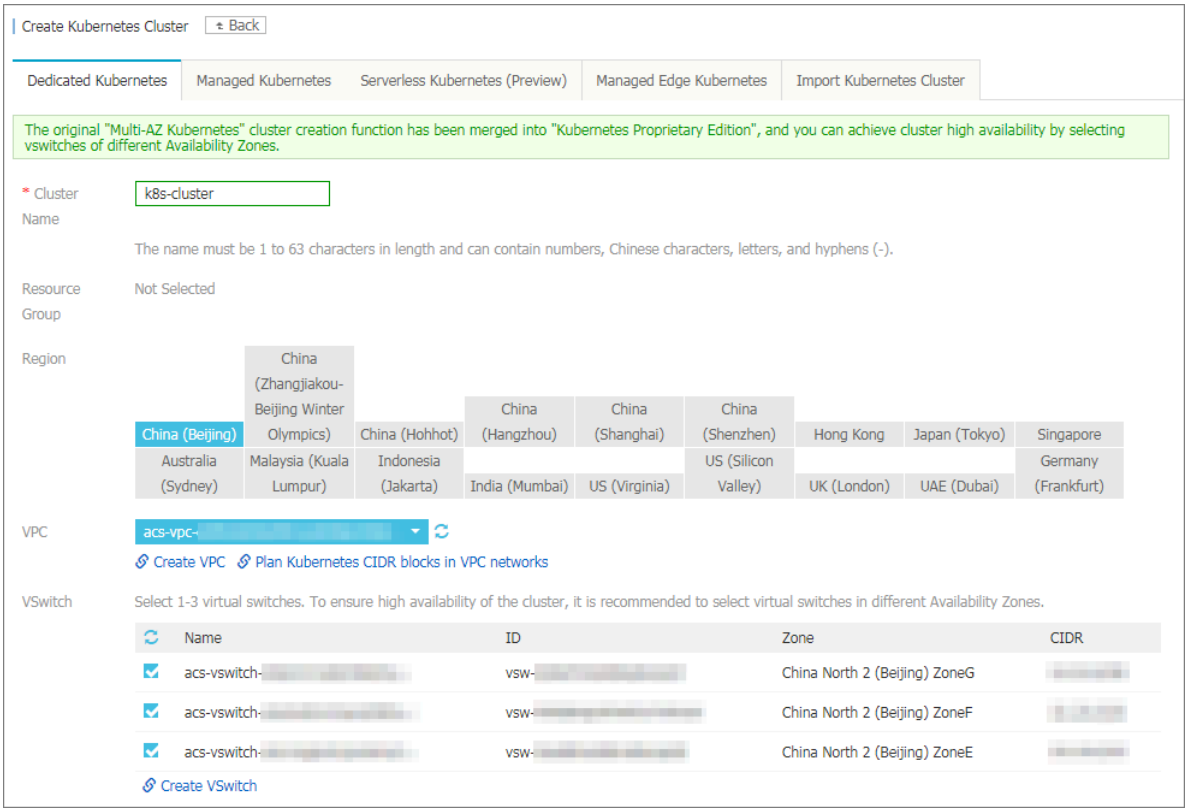
This example shows how to create a Kubernetes cluster. Some configurations use the default or the simplest configuration.

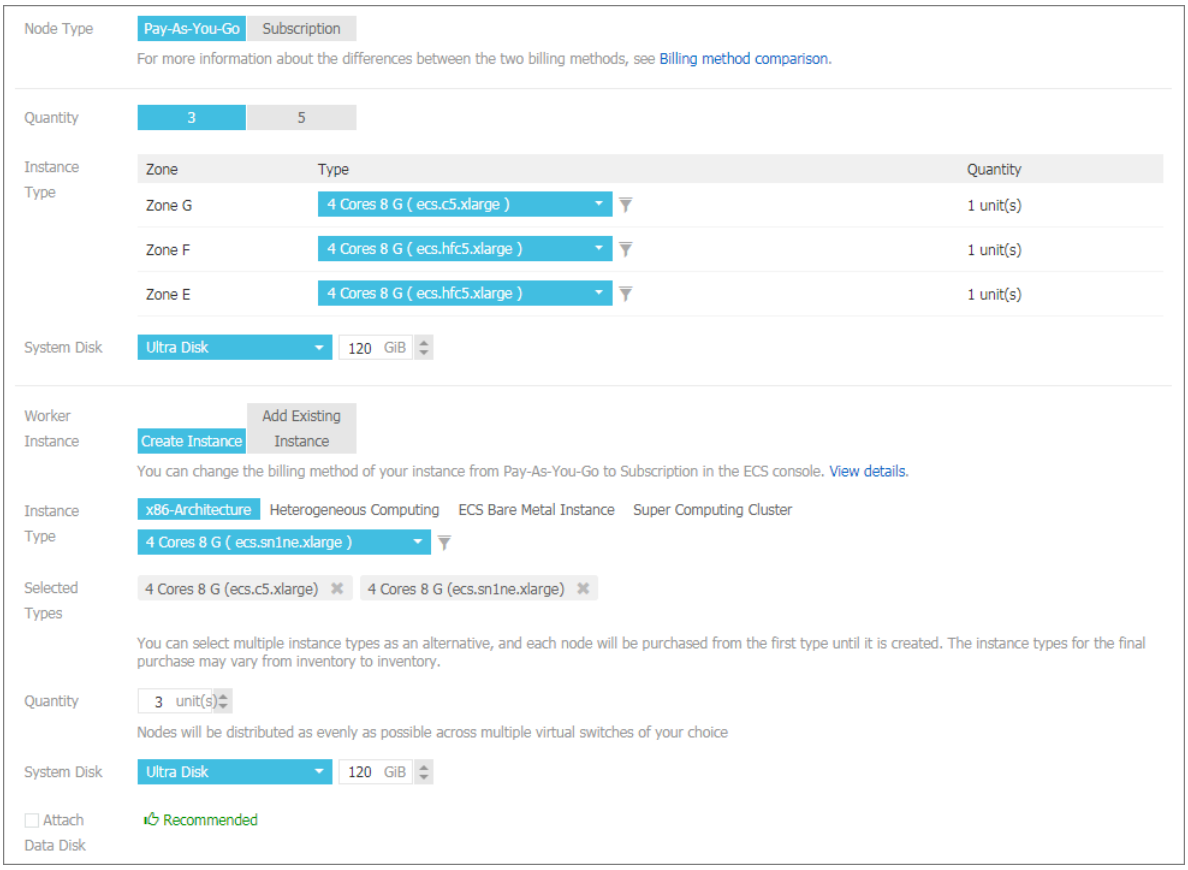
Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane under Container Service-Kubernetes, choose Clusters > Clusters.
3. In the upper-right corner, click Create Kubernetes Cluster. On the Select Cluster Template page, find the Standard Dedicated Cluster template, and click Create in the template.

4. Set cluster parameters.

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





Docker Version 18.09.2

Kubernetes Version 1.12.6-aliyun.1 1.11.5

Logon Type Key Pair Password

* Password ••••••••••

The password must be 8 to 30 characters in length and contain at least three of the following four types of characters: uppercase letters, lowercase letters, numbers, and special characters.

* Confirm Password ••••••••••

Network Plugin Flannel Terway

How to choose network plugins for a Kubernetes cluster

Pod CIDR Block 172.20.0.0/16

Specify a valid CIDR block that contains only internal IP addresses, namely one of the following CIDR blocks or their subnets: 10.0.0.0/8 172.16-31.0.0/12-16 and 192.168.0.0/16.
Make sure that the specified CIDR block does not overlap with the VPC CIDR block or the CIDR block used by an existing Kubernetes cluster in the VPC network.
You cannot modify the CIDR block after the cluster is created.
For more information about network segmentation of clusters, see Plan Kubernetes CIDR blocks in VPC networks .
In the current configuration, you can deploy up to 512 hosts in the cluster.

Service CIDR 172.21.0.0/20

Valid CIDR blocks include: 10.0.0.0/8 172.16-31.0.0/12-16 and 192.168.0.0/16.
Make sure that the specified CIDR block does not overlap with the VPC CIDR block or the CIDR block used by an existing Kubernetes cluster in the VPC network.
You cannot modify the CIDR block after the cluster is created.

Configure SNAT ☑ Configure SNAT for VPC

If the VPC network that you choose cannot access the Internet, NAT gateways and EIP instances will be used to configure SNAT for the VPC network.

Public Access ☑ Expose API Server with EIP

SSH Logon ☐ Allow Using SSH to Log on to Clusters from the Internet

If this function is disabled, you may need to manually enable SSH logon. For more information, see SSH access to Kubernetes clusters.

CloudMonitor Agent ☐ Install the CloudMonitor Agent on ECS Nodes Recommended

After the CloudMonitor agent is installed on ECS nodes, you can view monitoring information about the nodes in the CloudMonitor console.

Log Service ☐ Enable Log Service Note

The cluster auditing function is unavailable if Log Service is not activated.

RDS Whitelist Select RDS Instance

Master Node Protection ☑ Master Nodes Cannot Be Released by Using the Console or APIs

Labels : Add

Each label consists of a case-sensitive key value pair. You can add up to 20 labels.
The key must be unique and 1 to 64 characters in length. The value can be empty and up to 63 characters in length. Neither the key nor the value can start with any of the following strings: "aliyun", "acs:", "https://", and "http://". For more information, see Labels and Selectors.

Configuration	Description
Cluster Name	The cluster name must be 1 to 63 characters in length, and can contain letters, numbers, Chinese characters, letters, and hyphens (-).
Resource Group	The resource group for the cluster.

Configuration	Description
Region	The region in which the cluster is located.
VPC	The VPC that can be used by the cluster. · If the selected VPC has an existing NAT gateway, ACK then uses the already created NAT gateway. · If the selected VPC has an existing NAT gateway, ACK then uses the already created NAT gateway.undefined Note: If you do not want ACK to automatically create a NAT gateway for your selected VPC, clear Configure SNAT for VPC. Then, you must manually create a NAT gateway, or set an SNAT entry to ensure that your selected VPC is accessible to the Internet. Otherwise, instances in the VPC cannot access the Internet, which results in a cluster creation failure.
VSwitch	The VSwitches that can be used by the cluster. We recommend that you select three VSwitches that are located in different zones.
Node Type	Pay-As-You-Go and Subscription types are supported.
Duration	The period that you use the nodes. Note: If you select the Subscription node type, this parameter takes effect.

Configuration	Description
Auto Renewal	The period that nodes can be automatically renewed. Note: If you select the Subscription node type, this parameter takes effect.
Master node	Select the number of Master nodes, the instance type, and system disk. · Quantity: You can select 3 or 5. · Instance Type: For more information, see#unique_15 · System Disk: SSD disk and Ultra Disk are supported.
Worker node	You can create Worker node instances for the cluster or add existing Worker node instances to the cluster. To create Worker node instances, set the following parameters: · Instance Type: Indicates the ECS instance type. You can select multiple ECS instance types. For more information, see #unique_16. · System Disk: Indicates the type and capacity of the system disk. Available system disks are SSD disks and Ultra disks. · Attach Data Disk: Indicates the type and capacity of the data disk. Available data disks are SSD disks, Ultra disks, and basic disks.
Docker Version and Kubernetes Version	The Docker version and Kubernetes version.
Login	Key pair and password logon types are supported. For more information, see#unique_17.
Network Plugin	Available network plugins are Flannel and Terway. For more information, see #unique_18

Configuration	Description
Pod Network CIDR and Service CIDR	For more information, see #unique_19.
Configure SNAT	Optional. If you do not set this parameter, you must manually create a NAT gateway or an SNAT entry for the VPC.
Public Access	· If you enable this function, ACK then automatically creates an EIP and attaches it to an intranet SLB instance. In this case, the Master node port (namely, port 6443) is opened, and you can use the kubeconfig file to connect to and operate the cluster through the Internet. Note: The API server uses the Master node port. · If you do not enable this function, no EIP is created. In this case, you can only use the kubeconfig file to connect to and operate the cluster within your selected VPC.
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 more information, see #unique_20. Note: To enable this function, you must selectExpose API Server with EIP.

Configuration	Description
CloudMonitor Agent	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.
Log Service	Select Enable Log Service, and click Select Project or Create Project. Note: · A project is a space provided by Log Service to store log data. · If you select the check box, ACK then automatically installs the Log Service plugin for the cluster. This plugin allows you to collect logs of applications that run in the cluster and store the logs in the project that you set. For more information, see #unique_21.
RDS Whitelist	Add the IP addresses of the ECS instances to the RDS instance whitelist.
Master Node Protection	By default, this check box is selected to prevent users from releasing Master nodes by using the console or API.

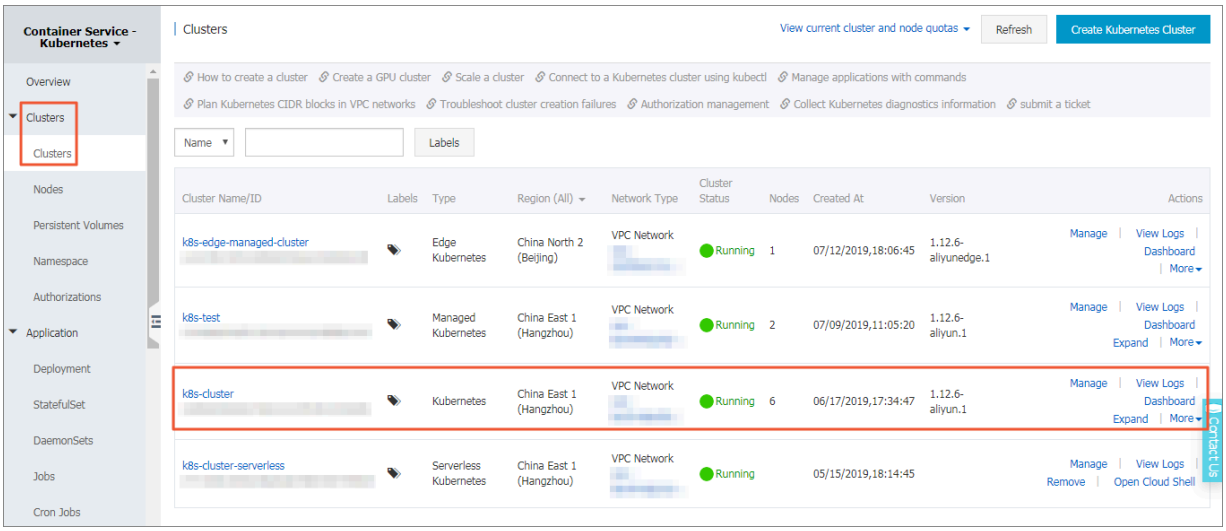
Configuration	Description
Label	Attach tags to the cluster. To attach tags to a cluster, the following rules apply: · For each tag, a key is required, but its value is optional. · A key can contain up to 64 characters but cannot start with `aliyun`, `http://`, or `https://`. It is not case sensitive. · The value must be 1 to 128 characters in length, and cannot start with `http://` or `https://`. It is not case sensitive. · For a cluster, each tag attached to it must be unique. If you attach a new tag that shares a key with an existing tag to a cluster, the new tag overwrites the existing tag. · A maximum of 20 tags can be attached to a cluster. If you want to continue to attach more tags, you must first remove the necessary number of tags to allow these tags to be added.

Configuration	Description
Advance Options	· Set the number of pods for each node. Note: This parameter specifies the maximum number of pods that run on a single node. We recommend that you retain the default setting. · Set the proxy mode for kube-proxy. - `iptables` : This is a mature and stable proxy mode that provides average performance. The service discovery and load balancing feature of a Kubernetes cluster uses `iptables` rules to configure services. This proxy mode is applicable only to a Kubernetes cluster that runs a few services. - `IPVS` : This is a high-performance kube-proxy mode. The service discovery and load balancing feature of a Kubernetes cluster uses the Linux IPVS module to configure services. This proxy mode can meet the requirement of the high-performance load balancing service. Therefore, this proxy mode is applicable to a Kubernetes cluster that runs a large number of services. · Set the range of node ports. Note: By default, the range of node ports is 30000 to 32767. · Set the CUP affinity policy. - none: Uses the default CPU affinity

5. In the upper-right corner, click Create. Then, in the displayed dialog box, click OK.

What's next

On the Cluster List page, verify that the Kubernetes cluster is created.



2.2 Create deployments by using images

You can use an image to create an Nginx application that is accessible to the Internet.

Prerequisites

Create a Kubernetes cluster. For more information, see #unique_23.

Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane, choose Applications > Deployments and then click Create from Image in the upper-right corner.
3. Set the following parameters: Name, Cluster, Namespace, Replicas, Type, Annotations, and Labels. The replicas parameter specifies the number of Pods. Click Next.

> **Note:**
> In this example, set the type parameter to Deployment.

If you do not set the Namespace parameter, the default namespace is used.

4. Configure containers.

> **Note:**
> You can configure multiple containers to run in the Pods.

a) Container general settings.

- Image Name: You can click Select Image to select the image in the dialog box that appears. In this example, select Nginx and click OK.

  You can also enter a private registry to specify the image. The format is as follows: `domainname / namespace / imagename : tag` .
- Image Version: You can click Select Image Version to select the version. If you do not specify the image version, the latest version is used by default.
- Always Pull Image: To improve efficiency, Container Service caches the image. During deployment, if the version of the newly specified image is the same as that of the cached image, Container Service will reuse the cached image rather than pull the image again. Therefore, if the image version is kept unchanged for some reason, for example, to make it easy to run upper-layer services, when the code or image is updated, the previously cached image will be used during deployment. When this option is selected, Container Service will always re-pull the image from the repository to deploy the application. This ensures that the latest image and code are used.
- Set Image Secret: Click Set Image Secret to set the image secret. You must set the secret if you need to access a private repository.#unique_24
- Resource Limit: The upper limits of CPU and memory resources that can be used by this application. This prevents the application from using excessive resources. The unit of CPU resources is Core. The unit of memory is MiB.
- Required Resources: The amount of CPU and memory resources that are reserved for this application. These resources are exclusive to the container.

This prevents the application from being unavailable when other services or processes compete for resources.

- Init Container: When this option is selected, the system creates an Init Container that contains useful tools. For more information, see https://kubernetes.io/docs/concepts/workloads/pods/init-containers/.

b) Optional: Configure environment variables.

You can configure environment variables for the Pods by using key-value pairs. Environment variables are used to add environment labels or pass configurations to the Pods. For more information, see Pod variable.

c) Optional: Configure Health Check settings.

Health check settings include liveness and readiness probes. Liveness probes are used to detect when to restart the container. Readiness probes determine if the container is ready to start accepting traffic. For more information about

health check, see https://kubernetes.io/docs/tasks/configure-pod-container/configure-liveness-readiness-probes.

Request type	Description
HTTP request	Sends an HTTP GET request to the container. Supported parameters are as follows: · Protocol: HTTP or HTTPS · Path: The requested path on the server. · Port: The port exposed by the container. The port number must be in the range of 1 to 65535. · HTTP Header: The custom headers in the HTTP request. Replicate headers are allowed. Supports key-value pairs. · Initial Delay (s): The `initialDelaySeconds` field. The time (in seconds) to wait before performing the first probe after the container is started. Default is 3. · Period (s): The `periodSeconds` field. How often (in seconds) to perform the probe. Default is 10. Minimum is 1. · Timeout (s): The `timeoutSeconds`. The time (in seconds) after which the probe times out. Default is 1. Minimum is 1. · Healthy Threshold: The minimum number of consecutive successes that must occur for the probe to be considered successful after having failed. Default is 1. Minimum is 1. For liveness probes, this parameter must be set to 1. · Failure Threshold: The minimum number of consecutive failed probes that are considered as failed after a successful probe. Default is 3. Minimum is 1.

Request type	Description
TCP connection	Sends a TCP socket to the container. The kubelet will attempt to open a socket to your container on the specified port. If a connection can be established, the container is considered healthy. Otherwise, it is considered unhealthy. Supported parameters are as follows: · Port: The port exposed by the container. The port number must be in the range of 1 to 65535. · Initial Delay (s): The `initialDelaySeconds` field. The time (in seconds) to wait before performing the first probe after the container is started. Default is 15. · Period (s): The `periodSeconds` field. How often (in seconds) to perform the probe. Default is 10. Minimum is 1. · Timeout (s): The `timeoutSeconds`. The time (in seconds) after which the probe times out. Default is 1. Minimum is 1. · Healthy Threshold: The minimum number of consecutive successes that must occur for the probe to be considered successful after having failed. Default is 1. Minimum is 1. For liveness probes, this parameter must be set to 1. · Unhealthy Threshold: The minimum number of consecutive failures that must occur for the probe to be considered failed after having succeeded. Default is 3. Minimum is 1.

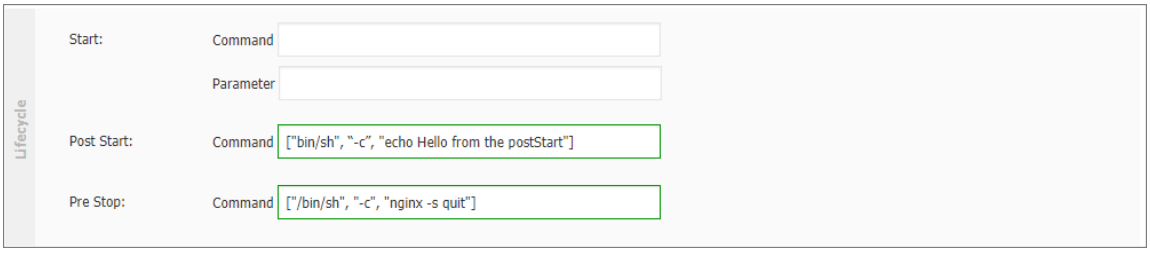
Request type	Description
Command line	Runs a probe command in the container to check its health. Supported parameters are as follows: · Command: The probe command that is used to check the health of the container. · Initial Delay (s): The `initialDelaySeconds` field. The time (in seconds) to wait before performing the first probe after the container is started. Default is 5. · Period (s): The `periodSeconds` field. How often (in seconds) to perform the probe. Default is 10. Minimum is 1. · Timeout (s): The `timeoutSeconds`. The time (in seconds) after which the probe times out. Default is 1. Minimum is 1. · Healthy Threshold: The minimum number of consecutive successes that must occur for the probe to be considered successful after having failed. Default is 1. Minimum is 1. For liveness probes, this parameter must be set to 1. · Unhealthy Threshold: The minimum number of consecutive failures that must occur for the probe to be considered failed after having succeeded. Default is 3. Minimum is 1.

d) Configure lifecycle events.

You can set the following parameters to configure the lifecycle of the container: start, postStart, and preStop. For more information, see https://kubernetes.io/docs/tasks/configure-pod-container/attach-handler-lifecycle-event/.

- Start: The pre-start command and parameter.
- Post Start: The post-start command.

- Pre Stop: The pre-stop command.



e) Optional: Configure volumes.

Local volumes and cloud volumes are supported.

- Local Volume: Supports hostPath, ConfigMaps, Secrets, and temporary directories. Local volumes mount the corresponding mount source to a path in the container. For more information, see Volumes.
- Cloud Volume: Supports three types of PVs: cloud disks, NAS, and OSS.

This example selects a PV created from a cloud disk and mounts the PV to the `/tmp` path in the container. Data generated in this path is stored to the cloud disk.

f) Optional: Configure Log Service. You can configure collection methods and custom labels.

> **Note:**
> Make sure that the Log Service agent has been installed on the cluster.

Configure log collection methods as follows:

- Logstore: Create a logstore to store log data in Log Service.
- Log Path: Supports stdout and text logs.
  - stdout: Collects the container's standard output.
  - Text Logs: Collects logs in the specific path of the container. This example collects text logs in the following path: /var/log/nginx. Wildcards are supported.

You can also add custom labels to logs. Once added, the labels will be collected and output along with logs. Custom labels make it easy to perform statistical analysis on log data.

5. Set other parameters based on your needs and then click Next.
6. Configure advanced settings.
   a) Configure Access Control settings.

   You can configure how to expose the Pods and click Create. This example creates a Service of the Cluster IP type and an Ingress to build an Nginx application that is accessible to the Internet.

   > Note:
   >
   > You can configure access control based on your needs:
   >
   > - Internal applications: For applications that run inside the cluster, you can create Services of the Cluster IP or Node Port type to enable internal communication as needed.
   > - External applications: For applications that need to be exposed to the Internet, you can configure access control by using one of the following methods:
   > - Create a Service of the Server Load Balancer type and expose your application to the Internet through the SLB instance.
   > - Create a Service of the Cluster IP or Node Port type, create an Ingress, and expose your application to the Internet through the Ingress. For more information, see https://kubernetes.io/docs/concepts/services-networking/ingress/.

   A. To create a Service, click Create in the Access Control section. Configure the Service in the dialog box that appears, and then click Create.

   - Name: The service name. Default is `applicatio nname - svc`.
   - Type: Select one from the following three types.
     - Cluster IP: Expose the Service through an internal IP address in the cluster. When selected, the Service is only accessible within the cluster.
     - Node Port: Expose the Service through the IP address and static port (NodePort) on each node. The Node Port Service can route requests to a Cluster IP Service, which is automatically created by the system. You

can access a Node Port Service from outside the cluster by requesting `< NodeIP >:< NodePort >`.

- Server Load Balancer: Expose the Service through Server Load Balancer , which supports Internet access and internal access. Server Load Balancer can route requests to Node Port and Cluster IP Services.

- Port Mapping: Set a service port and a container port. If the Type parameter is set to Node Port, you must set a node port to avoid port conflicts. TCP and UDP protocols are supported.
- Annotations: Add annotations to the Service. SLB parameters are supported. For more information, see #unique_25.
- Labels: Add labels to the Service.

B. To create an Ingress, click Create in the Access Control section. Configure Ingress rules in the dialog box that appears, and then click Create. For more information about Ingress configuration, see #unique_26.

> Note:
> When you create an application by using an image, you can create an Ingress for one Service only. This example uses a virtual host name as the test domain. You need to add a record to the hosts file. In actual scenarios, use a domain that has obtained an ICP filing.
>
> ```
> 101 . 37 . 224 . 146     foo . bar . com     # The   IP
> address   of   the   Ingress
> ```

C. You can find the newly created Service and Ingress in the Access Control section. You can click Update or Delete to make changes.

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

You can enable HPA to automatically scale the number of Pods based on the CPU and memory utilization. This enables the application to run smoothly at different load levels.

> Note:

To enable HPA, you must configure resource objects that can be scaled for the container. Otherwise, HPA would not work. For more information, see the general settings section.

- Metric: Supports CPU and memory. The resource type must be the same as the one you have specified in the Required Resources field.
- Condition: Specify the resource usage threshold. HPA starts scaling up the cluster when the threshold is exceeded.
- Maximum Replicas: The maximum number of replicas that the deployment can expand to.
- Minimum Replicas: The minimum number of replicas that the deployment keeps running.

c) Optional: Configure Scheduling settings.

You can set the following parameters: update method, node affinity, pod affinity, and pod anti affinity. For more information, see https://kubernetes.io/docs/concepts/configuration/assign-pod-node/#affinity-and-anti-affinity.

Note:
The affinity feature facilitates scheduling based on node labels and Pod labels. You can use built-in labels or add custom labels based on needs.

A. Set the Update Method.

You can select from RollingUpdate and Recreate. For more information, see https://kubernetes.io/zh/docs/concepts/workloads/controllers/deployment/.

B. Set Node Affinity rules.

Node scheduling supports required and preferred rules, and various operators such as In, NotIn, Exists, DoesNotExist, Gt, and Lt.

- Required rules must be met and are specified in the `requiredDuringSchedulingIgnoreDuringExecution` field of nodeAffinity. These rules have the same effect as `NodeSelector`. In this example, Pods can only be scheduled to nodes with specific labels. You can create multiple required rules in a way that only one of them must be met.
- Preferred rules may not be met and are specified in the `preferredDuringSchedulingIgnoreDuringExecution` field of nodeAffinity.

In this example, the scheduler tries not to schedule Pods to the nodes with specific labels. You can also set weights for preferred rules. If multiple nodes match the rule, the node with the highest weight is preferred. You can create multiple preferred rules in a way that all of them must be met before scheduling the Pods.

C. Set Pod Affinity rules. These rules specify how Pods are deployed relative to other Pods in the same topology domain. For example, you can use Pod affinity rules to deploy services that communicate with each other to the same topology domain, such as a host. This helps reduce the network latency between these services.

Pod affinity enables you to specify which nodes the Pods can be scheduled to based on the labels on other Pods. This feature supports required and preferred rules, and the following operators: `In` , `NotIn` , `Exists` , `DoesNotExist` .

- Required rules must be met and are specified in the `requiredDuringSchedulingIgnoreDuringExecution` field of podAffinity. Required rules must be met before a Pod can be scheduled to a node.
  - Namespace: The rule is defined based on the labels on Pods and therefore must be divided into namespaces.
  - Topology Domain: Specify the topologyKey, which is the key of the node label that the system uses to denote the topology domain. For example, if you set the parameter to `kubernetes.io/hostname` , nodes are used to identify topologies. If set to `beta.kubernetes.io/os` , the operating systems of nodes are used to identify topologies.
  - Selector: Click Add to add multiple required rules.
  - View Applicaitons: Click View Applicaitons and specify the namespace and application in the dialog box that appears. You can view the labels on the selected application and add them to the rule configuration page.
  - Required rule: Specify the label on the existing application, the operator, and the label value. This example schedules the application

to be created to the host that runs the application with label `app : nginx` .

- Preferred rules may not be met and are specified in the `preferredDuringSchedulingIgnoredDuringExecution` field of podAffinity. A preferred rule specifies that, if the rule is met, the scheduler tries to enforce the rule. You can set weights for preferred rules. The other parameters are the same as those of required rules.

> Note:
> Weight: Set the weight for a preferred rule to a value between 1 to 100. The scheduler calculates a weight for each node that meets the rule based on an algorithm and schedules the Pod to the node with the highest weight.

D. Set Pod Anti Affinity rules to prevent scheduling Pods to the same topology where other Pods with specific labels are already deployed. Pod anti affinity rules can be used in the following scenarios:

- Schedule the Pods of a service to different topology domains, such as multiple hosts, to enhance the stability of the service.
- Grant a Pod exclusive access to a node. This ensures resource isolation and guarantees that no other Pod can share the specified node.
- Schedule the Pods of a service to different hosts if these Pods may interfere each other.

> Note:
> The parameters of pod anti affinity rules are the same as those of pod affinity rules. Create the rules based on different scenarios.

7. Click Create.

8. After the application is created, you are redirected to the Complete page. You can find the resource objects under the application and click View Details to view application details.

   The nginx-deployment details page appears.

   > Note:
   > You can also create Ingresses and Services as follows: In the above figure, click the Access Method tab.
   > - Click Create next to Service. Follow the steps introduced in 6.i.a to create a Service.
   > - Click Create next to Ingress. Follow the steps introduced in 6.i.b to create an Ingress.

9. In the left-side navigation pane, choose Ingresses and Load Balancing > Ingresses. You can find the newly created Ingress on this page.

10. Visit the test domain through a browser and the Nginx welcome page appears.

2.3 Create a StatefulSet application by using an image

Kubernetes clusters of Alibaba Cloud Container Service allows you to quickly create applications of the StatefultSet type through the web interface. In this example, create a StatefultSet Nginx application and show features of a StatefultSet application.

Prerequisites

- You have created a Kubernetes cluster. For more information, see #unique_5.
- You have successfully created a cloud disk storage volume claim. For more information, see #unique_28.
- You have successfully connected to the master node of the Kubernetes cluster. For more information, see #unique_29.

Context

StatefulSet features are as follows:

Scenarios	Description
Pod consistency	Contains order (such as startup and stop order) and network consistency. This consistency is related to pods and has nothing to do with the node to which the pods are to be scheduled.
Stable persistent storage	Create a PV for each pod through VolumeClaimTemplate. Deleting or reducing replicas does not delete relevant volumes.
Stable network marker	The hostname mode for a pod is: `( statefulse t name )-( sequence number ).`
Stable order	For StatefulSet of N replicas, each pod is assigned a unique order number within the range of 0 to N.

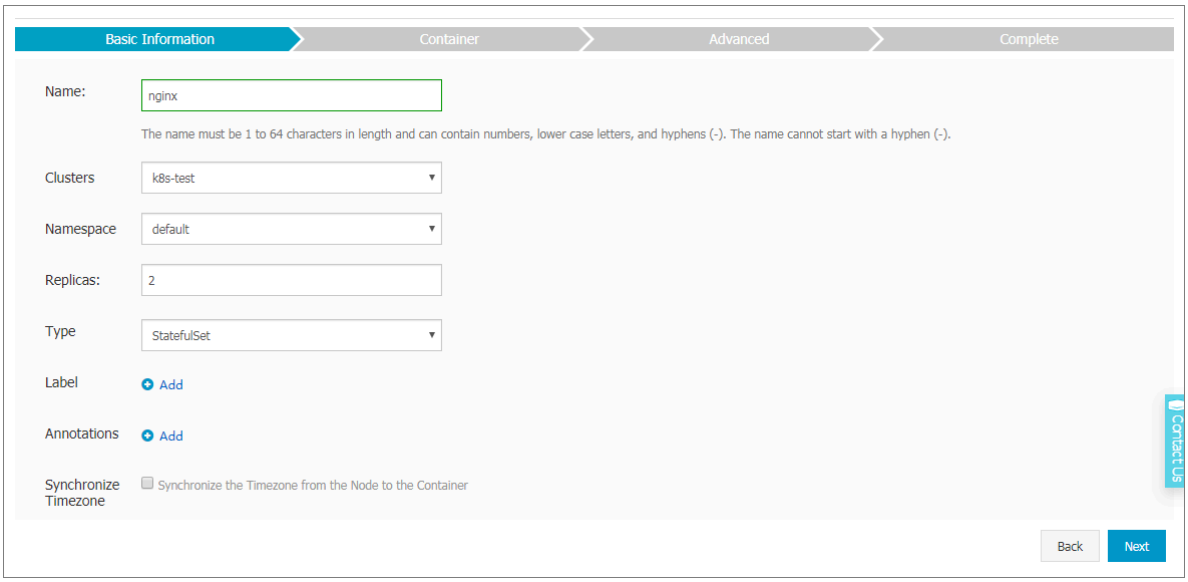
Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane under Container Service-Kubernetes, choose Application > StatefulSet. Then, click Create from Image in the upper-right corner.
3. Configure the basic parameters and then click Next.
   - Name: Enter the application name.
   - Cluster: Select a cluster to which the application is deployed.
   - Namespace: Select a namespace in which the application deployment is located. By default, the default namespace is used.
   - Replicas: Set the number of pods included in the application.
   - Type: Deployment type and StatefulSet type are available.

     Note:

In this example, select the StatefulSet type.

- Label: Add a label to the application.
- Annotations: Add a annotation to the application.



4. Configure containers.

Note:

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

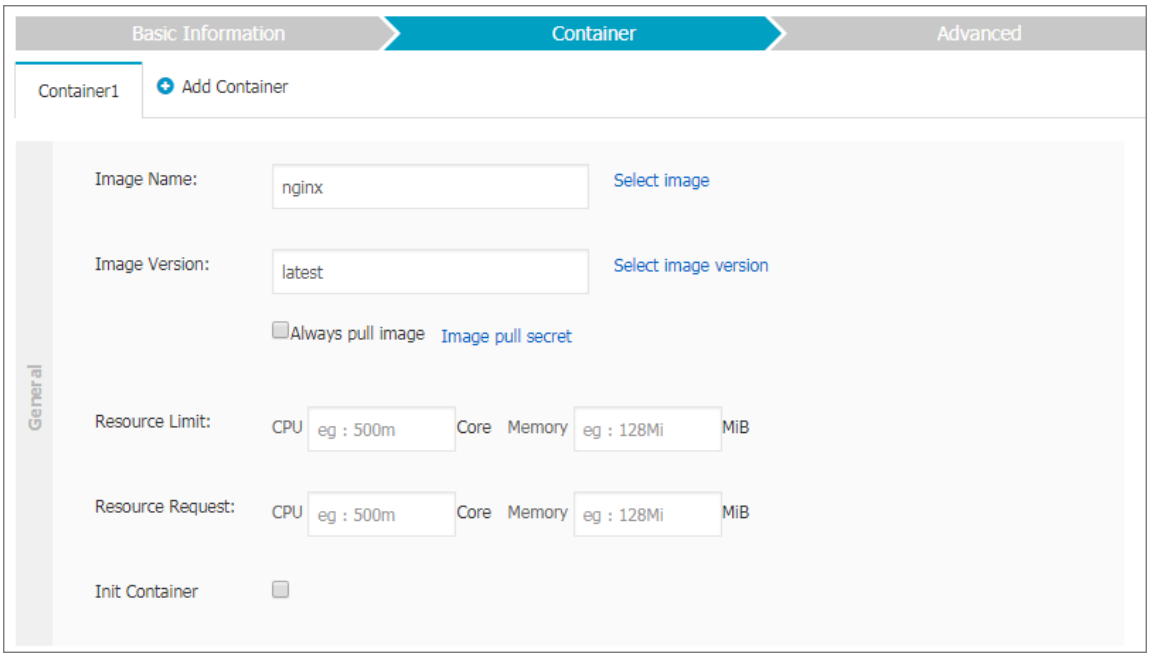
a) Configure the general settings for the application.

- 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 the 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 the image version is not specified, the system uses the latest version by default.
- Always pull image: Container Service caches the image to improve deployment efficiency. During deployment, if the image tag is found consistent with that on the local cache, the image on the local cache is reused and is not pulled again. Therefore, if you do not modify the image tag when changing your codes and image for convenience of upper-layer business, the early image on the local cache is used in the application deployment. With this check box selected, Container Service ignores the cached image and re-

pulls the image from the repository when deploying the application to make sure the latest image and codes are always used.

- 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, that is, these resources are exclusive to the container. Other services or processes will compete for resources when the resources are insufficient. By specifying the Resource Request, the application will not become unavailable because of insufficient resources.
- Init Container: Selecting this check box creates an Init Container which contains useful tools. For more information, see https://kubernetes.io/docs/concepts/workloads/pods/init-containers/.



b) Optional: Configure Environment .

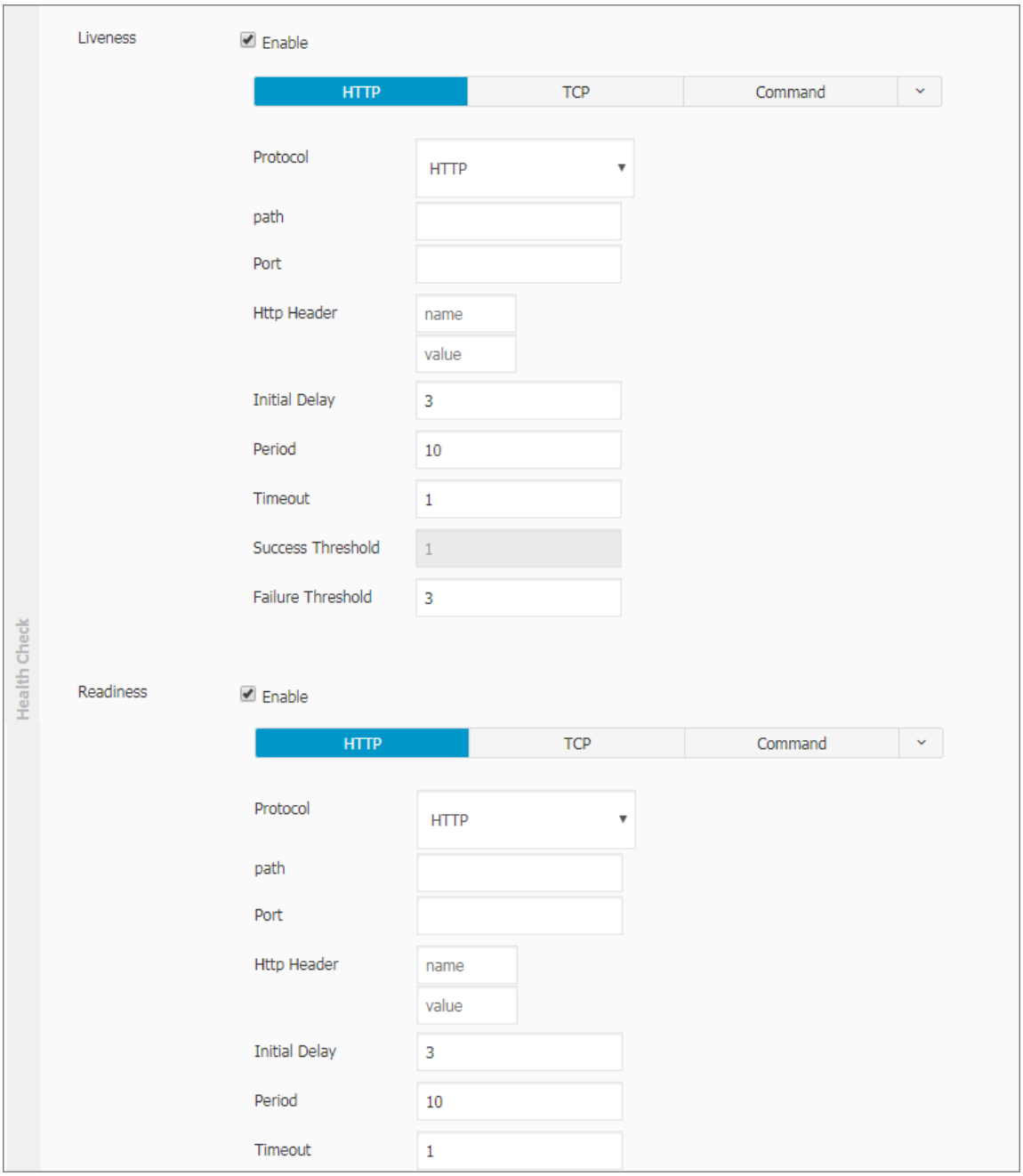
You can configure environment variables for the pod by using key-value pairs. Environment variables are used to add environment labels or pass configurations for the pod. For more information, see Pod variable.

c) Optional: Configure Health Check.

The health check function includes liveness probes and readiness probes. Liveness probes are used to detect when to restart the container. Readiness

probes determine if the container is ready for receiving traffic. For more

information about health check, see https://kubernetes.io/docs/tasks/configure-pod-container/configure-liveness-readiness-probes.



Request method	Description
HTTP request	An HTTP GET request is sent to the container. The following are supported parameters: · Protocol: HTTP/HTTPS · Path: Path to access the HTTP server · Port: Number or name of the 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. Supports the correct configuration of key values. · Initial Delay (in seconds): Namely, the initialDelaySeconds. Seconds for the first probe has to wait after the container is started. The default is 3. · Period (in seconds): Namely, the periodseconds. Intervals at which the probe is performed. The default value is 10. The minimum value is 1. · Timeout (in seconds): Namely, the timeoutSeconds. The time of probe timeout. The default value is 1 and the minimum value is 1. · Success Threshold: The minimum number of consecutive successful probes that are considered as successful after a failed probe. The default is 1 and the minimum is 1. It must be 1 for a liveness probe. · Failure Threshold: The minimum number of consecutive failed probes that are considered as failed after a successful probe. The default value is 3. The minimum value is 1.

Request method	Description
TCP connection	A TCP socket is send to the container. The kubelet attempts to open a socket to your container on the specified port. If a connection can be established, the container is considered healthy. If not, it is considered as a failure. The following are supported parameters: · Port: Number or name of the port exposed by the container. The port number must be in the range of 1 to 65535. · Initial Delay (in seconds): Namely, the initialDelaySeconds. Seconds for the first liveness or readiness probe has to wait after the container is started. The default is 15. · Period (in seconds): Namely, the periodseconds. Intervals at which the probe is performed. The default value is 10. The minimum value is 1. · Timeout (in seconds): Namely, the timeoutSeconds. The time of probe timeout. The default value is 1 and the minimum value is 1. · Success Threshold: The minimum number of consecutive successful probes that are considered as successful after a failed probe. The default is 1 and the minimum is 1. It must be 1 for a liveness probe. · Failure Threshold: The minimum number of consecutive failed probes that are considered as failed after a successful probe. The default value is 3. The minimum value is 1.

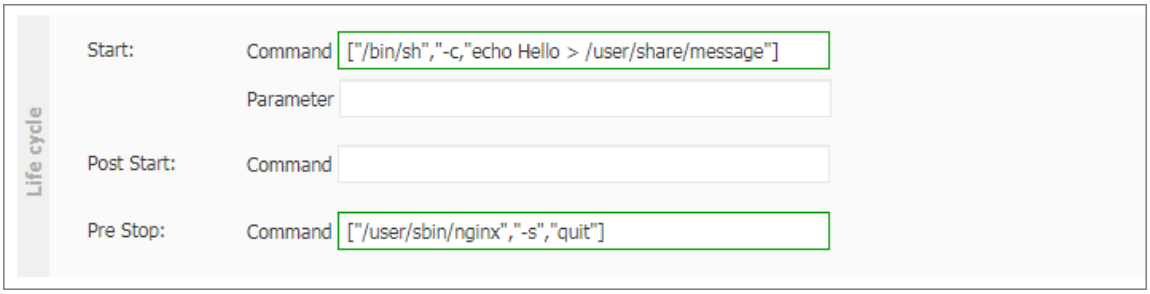
Request method	Description
Command line	Detect the health of the container by executing probe detection commands in the container. The following are supported parameters: · Command: A probe command used to detect the health of the container . · Initial Delay (in seconds): Namely, the initialDelaySeconds. Seconds for the first liveness or readiness probe has to wait after the container is started. The default is 5. · Period (in seconds): Namely, the periodseconds. Intervals at which the probe is performed. The default value is 10. The minimum value 1. · Timeout (in seconds): Namely, the timeoutSeconds. The time of probe timeout. The default value is 1 and the minimum value is 1. · Success Threshold: The minimum number of consecutive successful probes that are considered as successful after a failed probe. The default is 1 and the minimum is 1. It must be 1 for a liveness probe. · Failure Threshold: The minimum number of consecutive failed probes that are considered as failed after a successful probe. The default value is 3. The minimum value is 1.

d) Optional: Configure the lifecycle rule.

You can configure the following parameters for the container lifecycle: start, post start, and pre-stop. For more information, see https://kubernetes.io/docs/tasks/configure-pod-container/attach-handler-lifecycle-event/.

- Start: Configure a pre-start command and parameter for the container.
- Post Start: Configure a post-start command for the container.

- Pre Stop: Configure a pre-end command for the container.

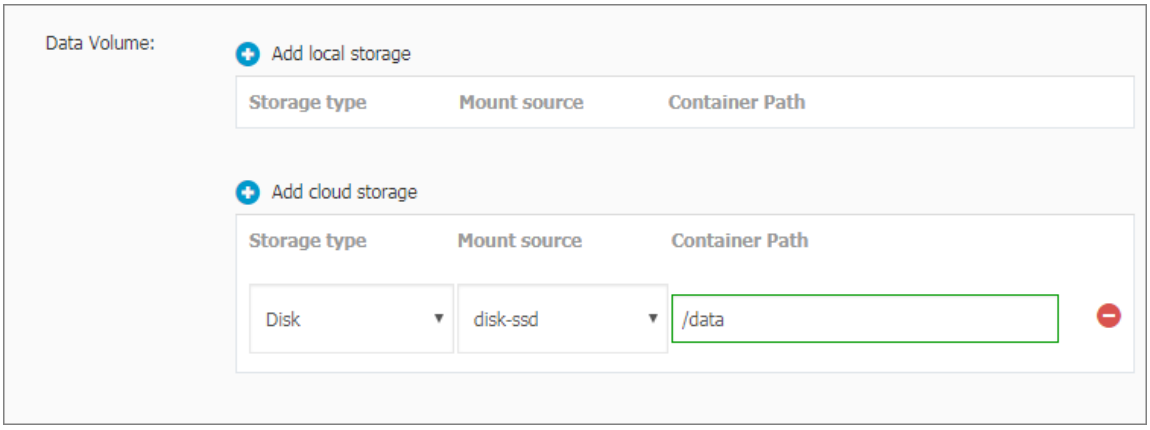


e) Configure data volumes.

Local storage and cloud storage can be configured.

- Local storage: Supports hostPath, configmap, secret, and temporary directory. The local data volumes mount the corresponding mount source to the container path. For more information, see Volumes.
- Cloud storage: Supports three types of cloud storage: cloud disks, Network Attached Storage (NAS), and Object Storage Service (OSS).

In this example, configure a data volume claim named disk-ssd of cloud disk type and mount it to the `/data` path.



f) Optional: Configure Log Service. You can configure collection methods and customize tags for this service.

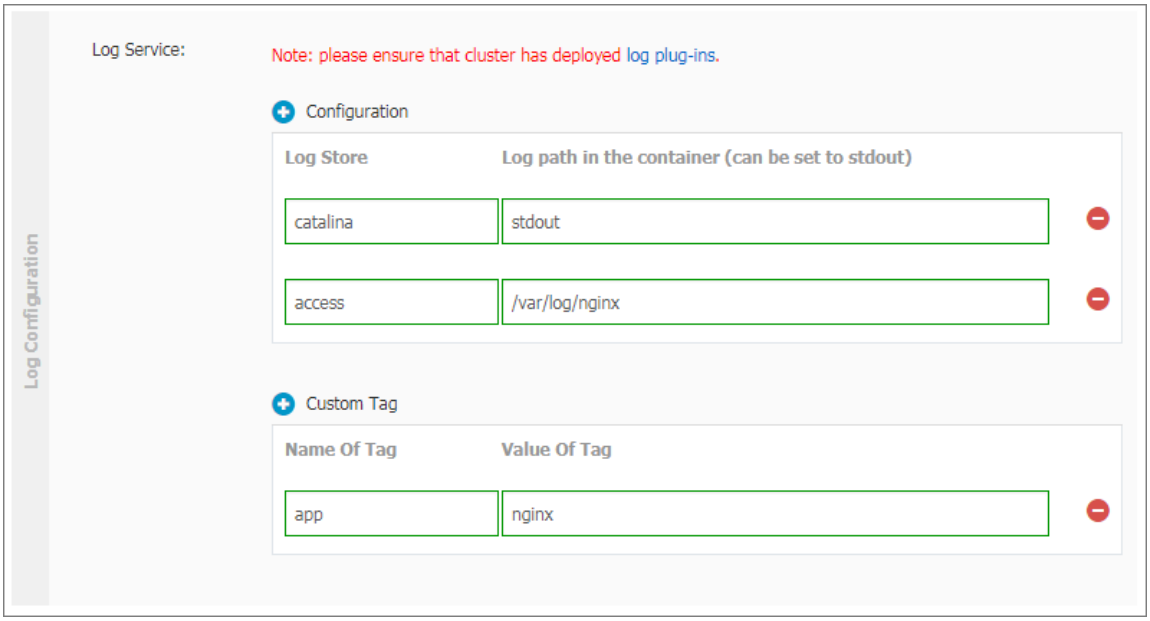
Note:

Make sure that a Kubernetes cluster is deployed and that the log plug-in is installed on the cluster.

Configure log collection methods as follows:

- Log Store: Configure a Logstore generated in Log Service which is used to store collected logs.
- Log path in the container: Supports stdout and text logs.
  - stdout: Collects standard output logs of containers.
  - text log: Collects logs in the specified path in the container. In this example, collect text logs in the path of /var/log/nginx. Wildcards are also supported.

You can also set custom tags. The customized tags are collected to the container output logs. A custom tag can help you tag container logs, providing convenienc e to log analysis such as log statistics and filter.



5. Click Next after completing the configurations.
6. Configure advanced settings. In this example, configure only access settings.
   a) Set Access Control.

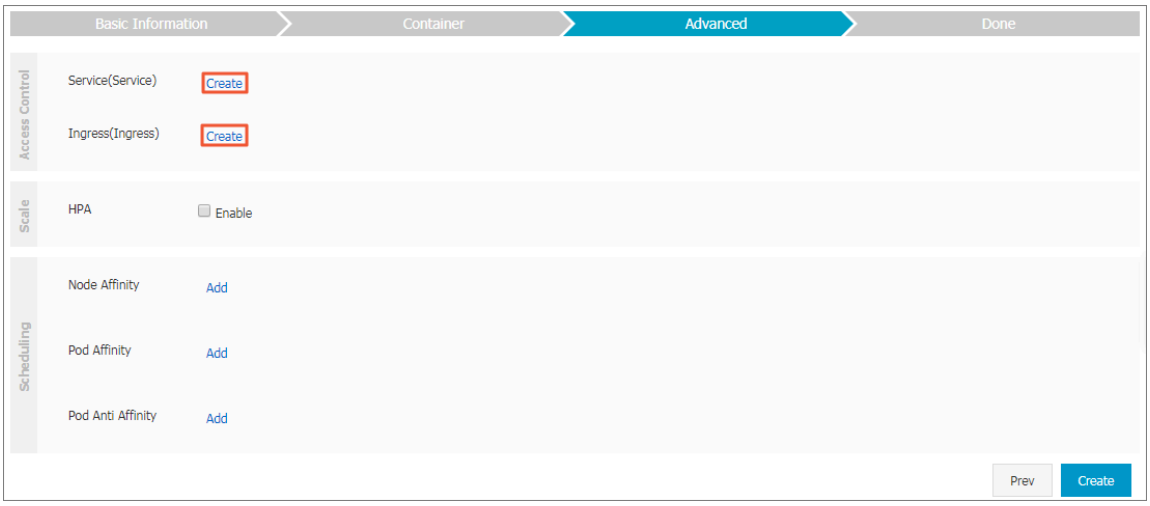
   You can set the methods to expose the application pod and then click Create. In this example, a cluster IP service and an Ingress are set to create an Nginx application that is accessible for the Internet.

   Note:

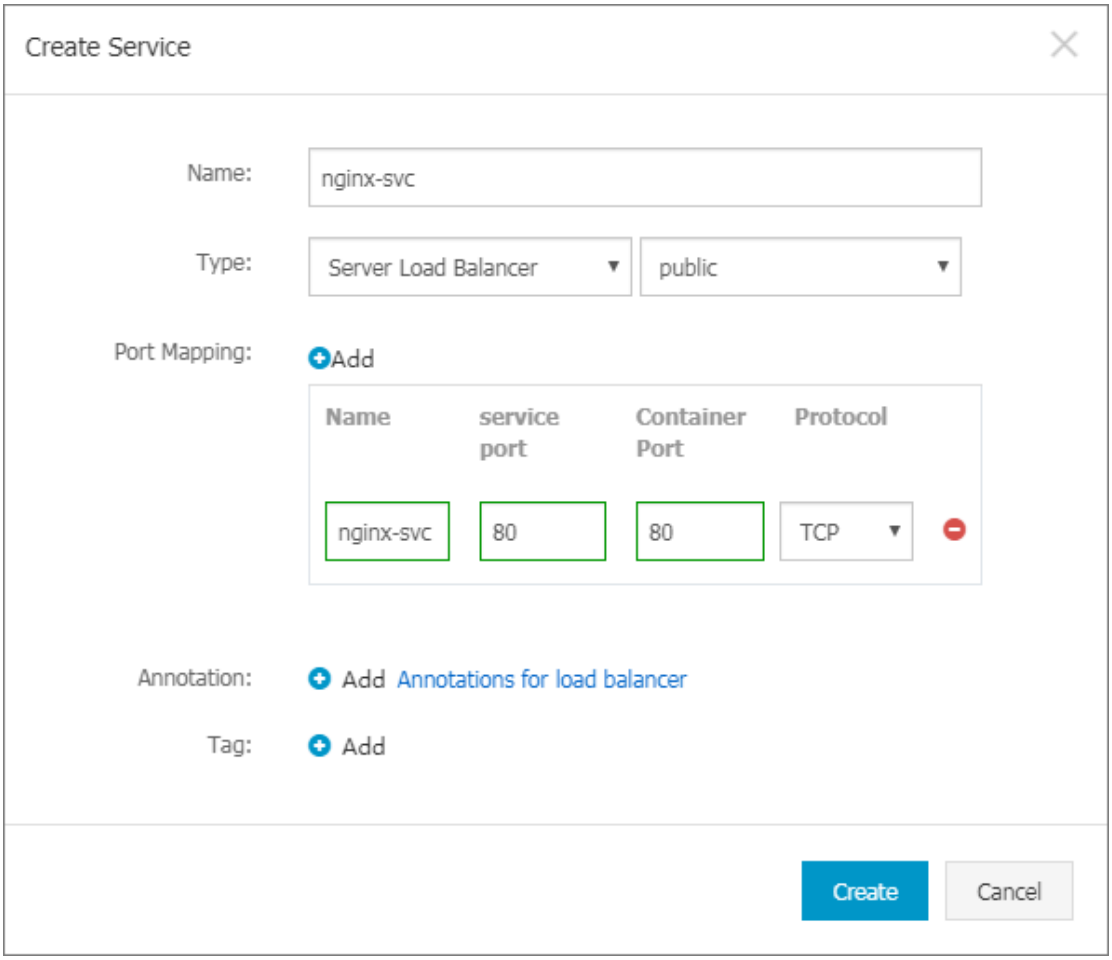
You can set access methods according to the communication requirements of your application.

- `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.

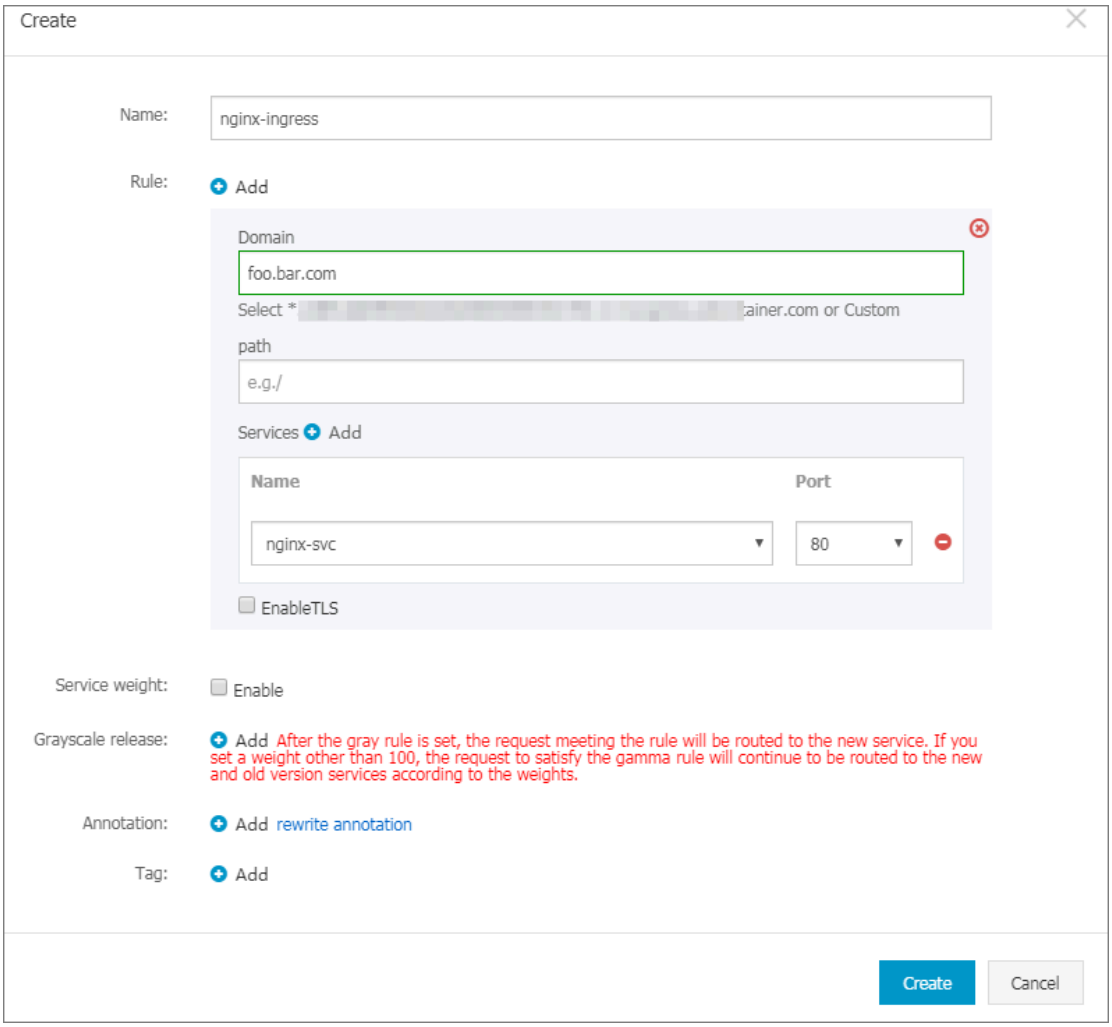
- Port Mapping: Add a service port and a container port, and select the TCP or UDP protocol. If you select the node port Type, you must add a node port to avoid port conflict.
- annotation: Add an annotation to the service. You can set SLB parameters. For more information, see #unique_30.
- Tag: Add 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 #unique_31.

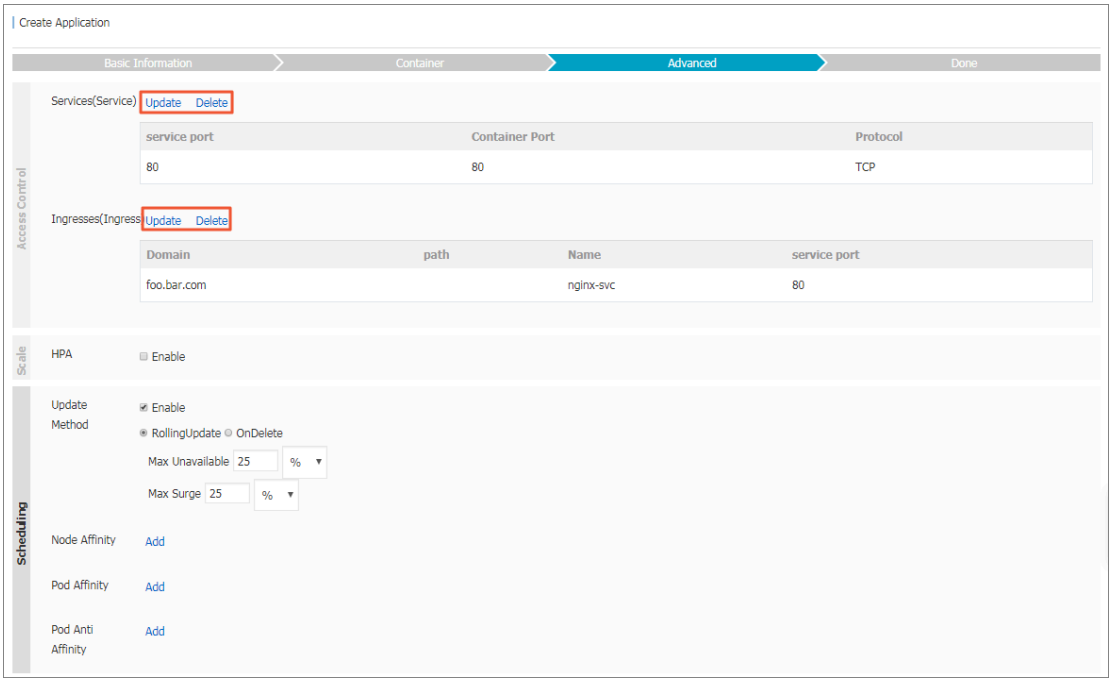
> Note:
> When you create an application by using an image, you can create an Ingress rule for only one service. In this example, a virtual host name is used as the

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 .
```

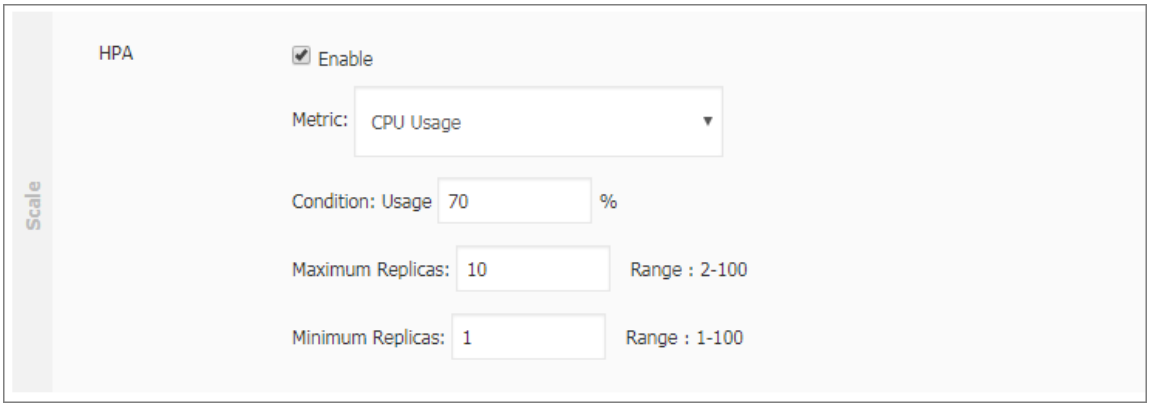


C. In the access control area, the created service and Ingress are displayed. You can perform further configurations by clicking Update or Delete.



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 StatefulSet application can contain.
- Minimum Replicas: the minimum number of the containers that the StatefulSet application can contain.

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.

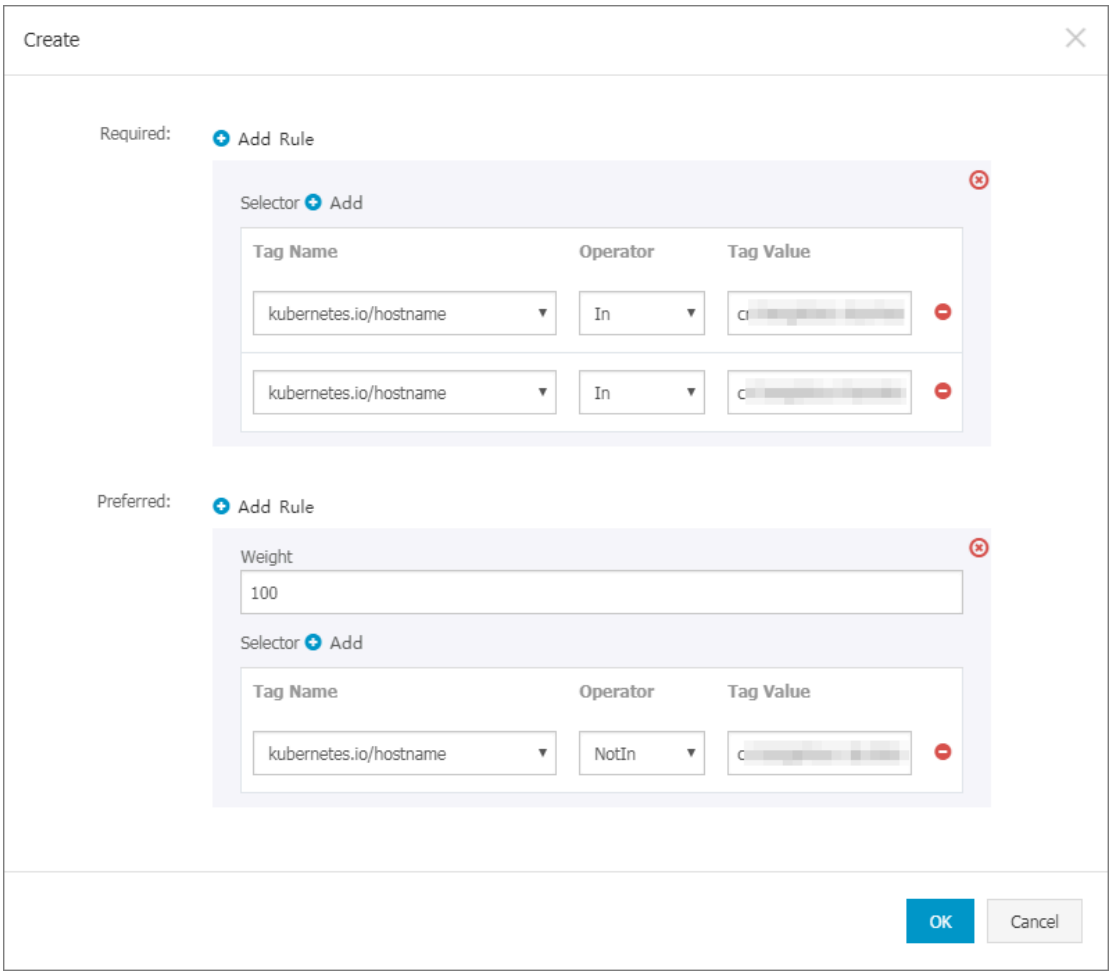
Note:

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 `requiredDuringSchedulingIgnoredDuringExecution`. 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.

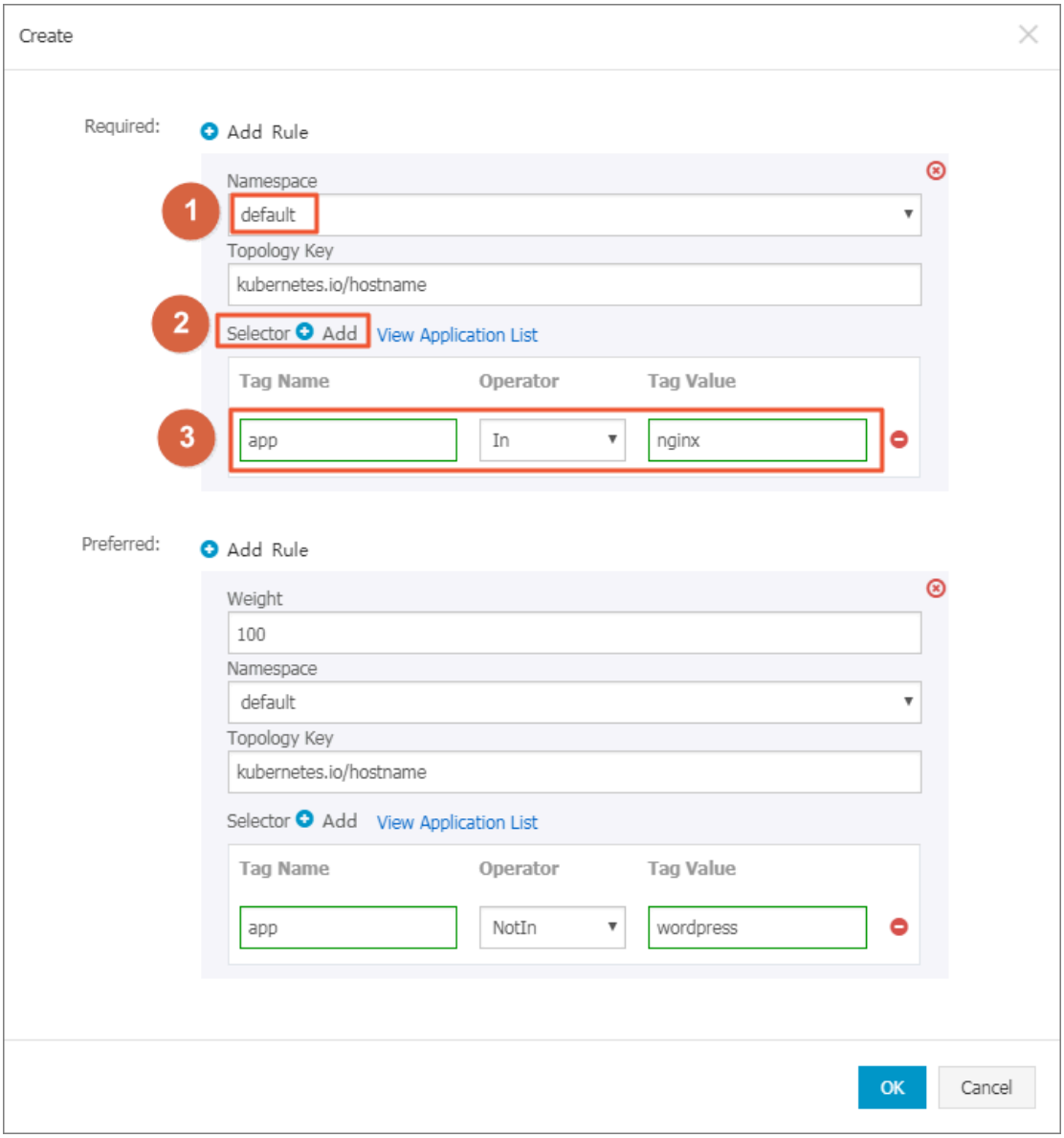
- Preferred rules can be unnecessarily satisfied and correspond to `preferredD uringSched ulingIgnor edDuringEx ecution`. With the scheduling setting in this example, the system tries not to schedule the pod to the nodes with the specified tag.

  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.

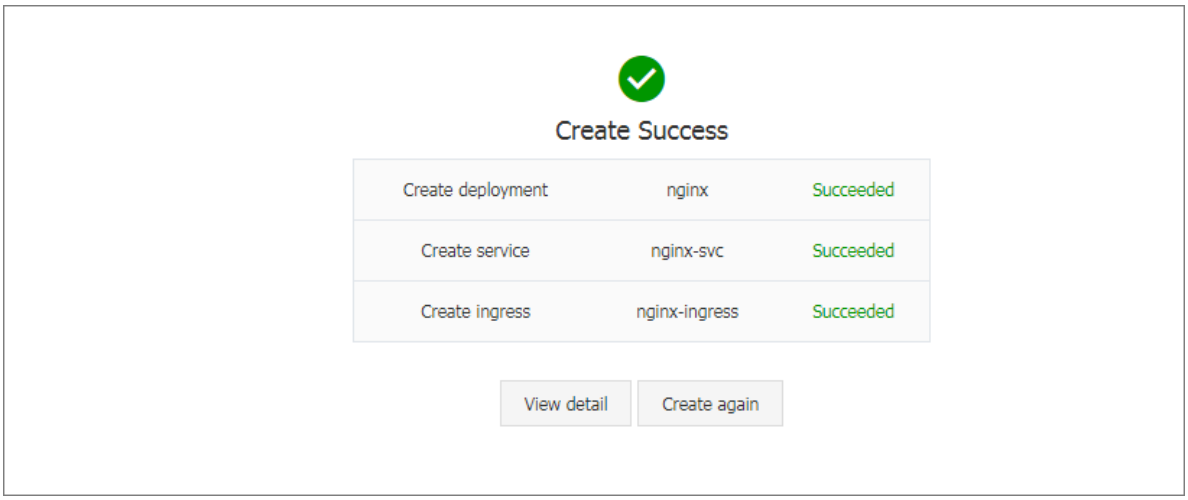
D. Set Pod Anti Affinity to deploy the application pods in a topology domain that excludes other pods. Scenarios that use pod anti affinity scheduling include:

· 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.

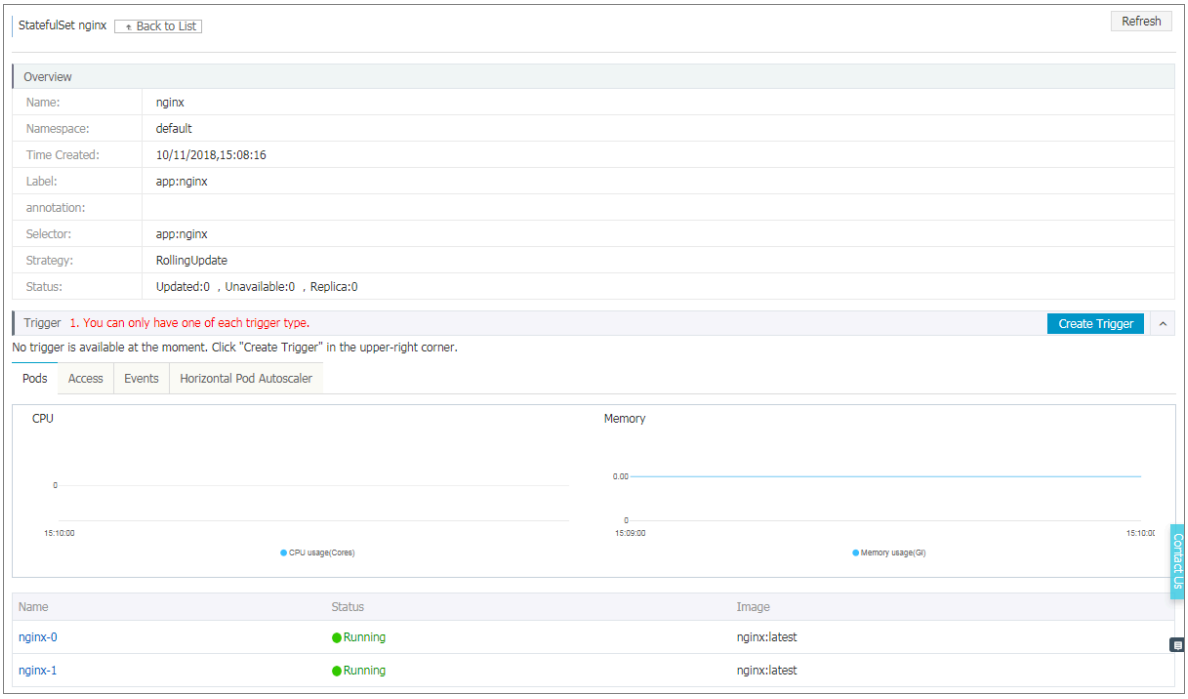
Note:
You can set pod anti affinity scheduling by using the same method as setting pod affinity scheduling. But the same scheduling rules have different

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

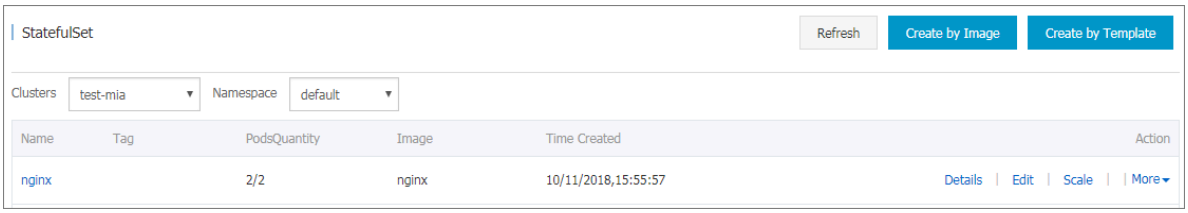
7. Click Create.
8. After you create the application, the create success page is displayed by default and objects contained in the application are listed. You can click View detail to view the deployment details.



The StatefulSet page is displayed by default.

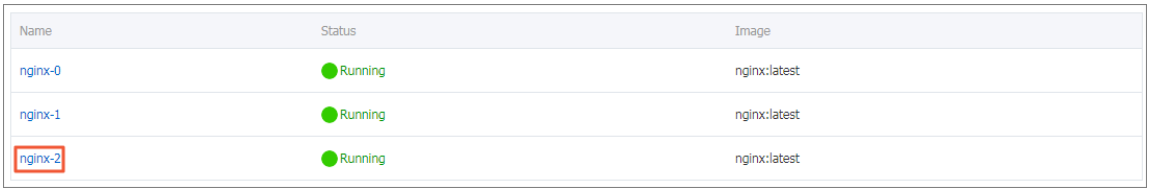


9. Then click Back to list in the upper-left corner to view the created StatefulSet application in the StatefulSet list page.

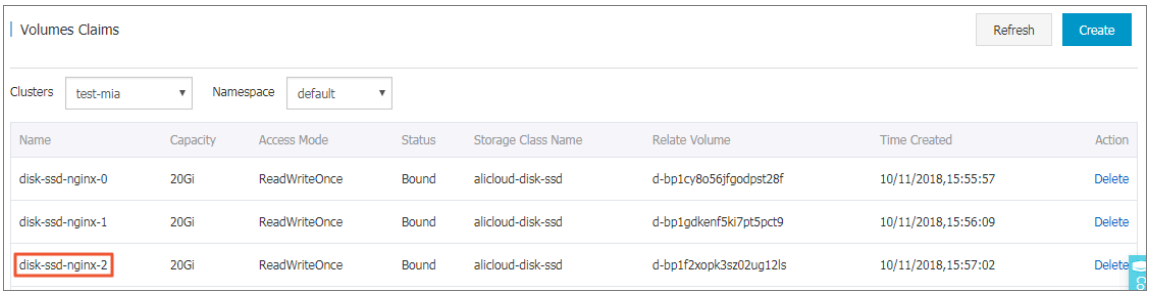


10. Optional: To verify service scalability, click Scale at the right of a target nginx application.

   a) In the displayed dialog box, set the number of pod to 3. You can see that when you expand pods, the pods are in the increment order; when you contract pods, the pods are in the descending order. This shows the order stability of pods in StatefulSet.



   b) Click Application > Volumes Claim in the left-side navigation pane, you can see that as the application expands, new cloud disk volumes are created with pods; if the application contracts, created PV/PVC will not be deleted.



What's next

Connect to the master node and run following commands to verify the persistent storage feature.

Create a temporary file on a cloud disk:

```
# kubectl exec nginx - 1 ls / tmp # list files
 under this directory
 lost + found

# kubectl exec nginx - 1 touch / tmp / statefulse t
 # add a temporty file named statefulse t
```

```
# kubectl exec nginx - 1 ls / tmp
 lost + found
 statefulse t
```

Remove the pod to verify the data persistence:

```
# kubectl delete pod nginx - 1
 pod " nginx - 1 " deleted

# kubectl exec nginx - 1 ls / tmp #
 data persistenc e storage
 lost + found
 statefulse t
```

In addition, you can also find that after you delete a pod, the pod automatically restarts after a period of time, which indicates the high availability of the StatefulSet application.

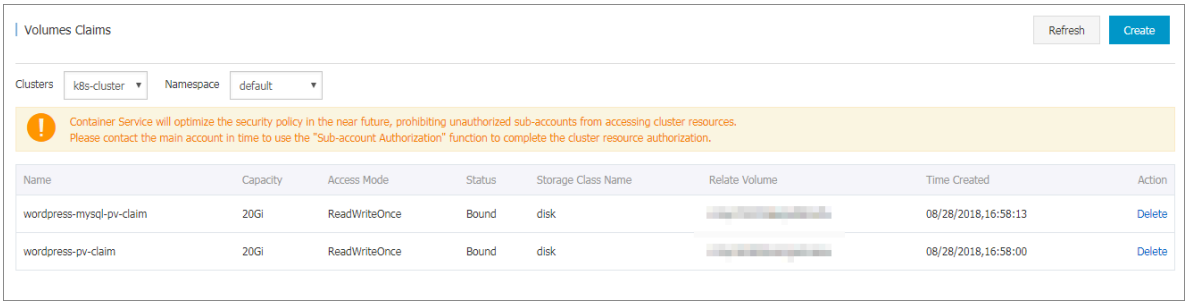
For more information, see Best practices of Kubernetes StatefulSet.

2.4 Deploy dependency-based WordPress applications

Prerequisites

- A Kubernetes cluster is created. For more information, see #unique_33.
- A Persistent Volume (PV) and Persistent Volume Claim (PVC) are created. For more information about how to create a PV, see #unique_34, #unique_35, and #unique_36. For more information about how to create a PVC, see #unique_28.

> **Note:**
> 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:

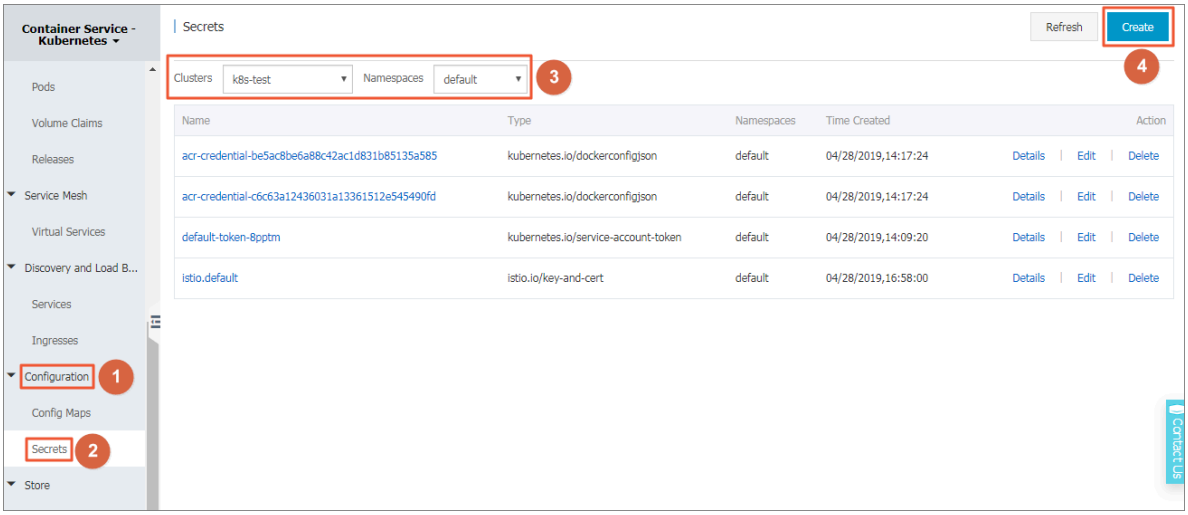
- wordpress
- mysql

Resources involved:

- Storage volume
- Secret
- Service

Procedure

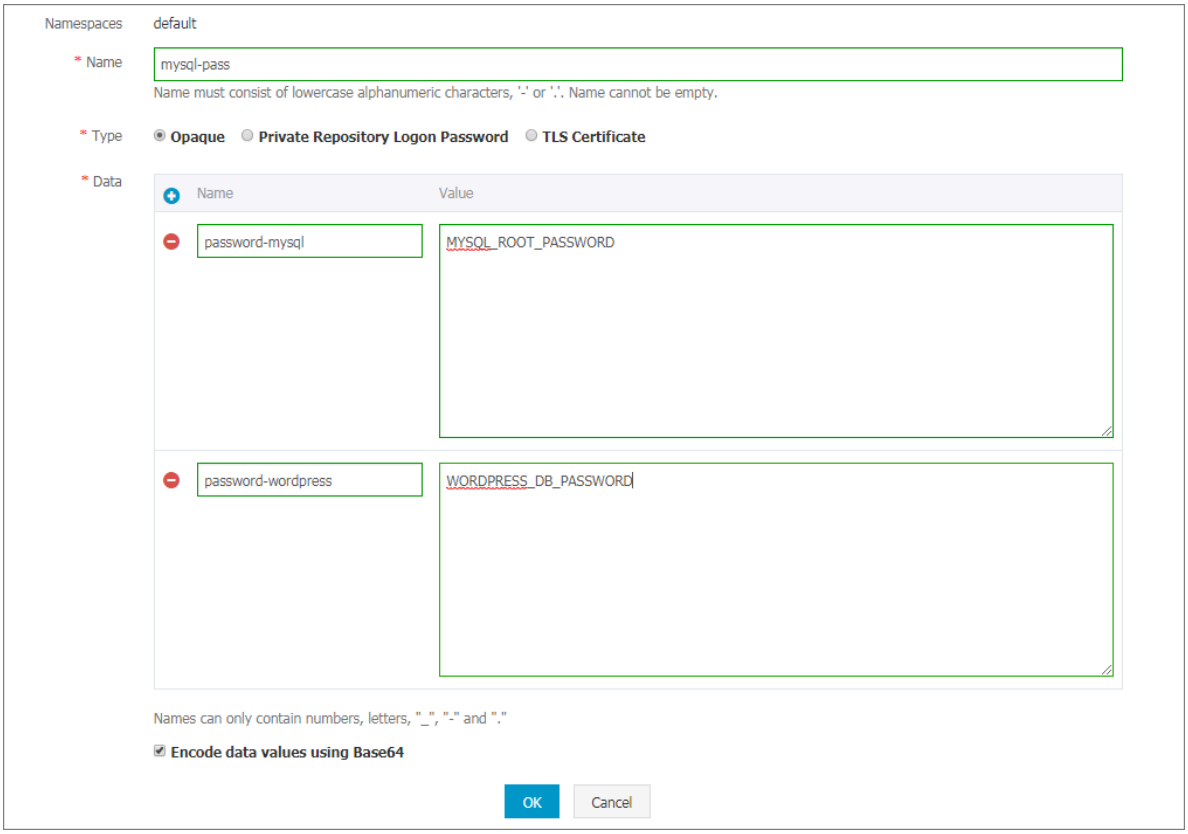
1. Log on to the Container Service console.
2. Use the prepared storage volume claims. Create two storage volume claims: wordpress-pv-claim and wordpress-mysql-pv-claim which are used in the wordpress yaml file and the wordpress-mysql yaml file respectively, to mount corresponding storage volumes.
3. 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 #unique_37.



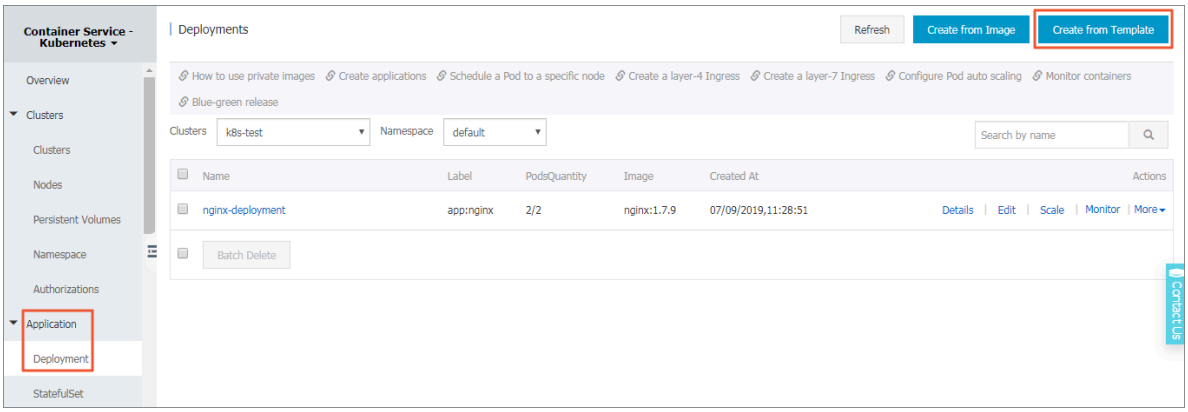
Note:

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 Application > Deployment, and click Create from 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

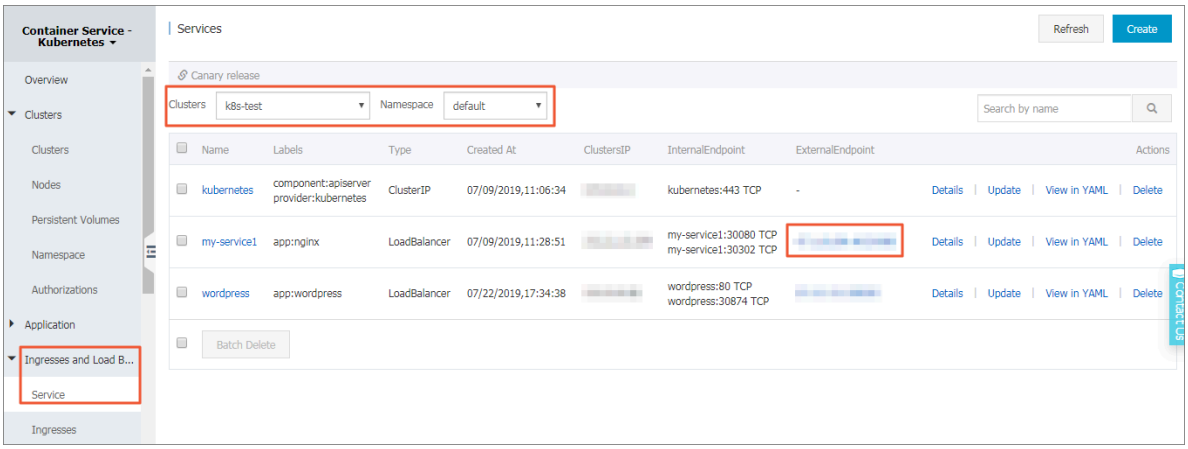
mysql is called only internally for the WordPress, you do not need to create a LoadBalancer type of service for it.

For more information, see #unique_38.

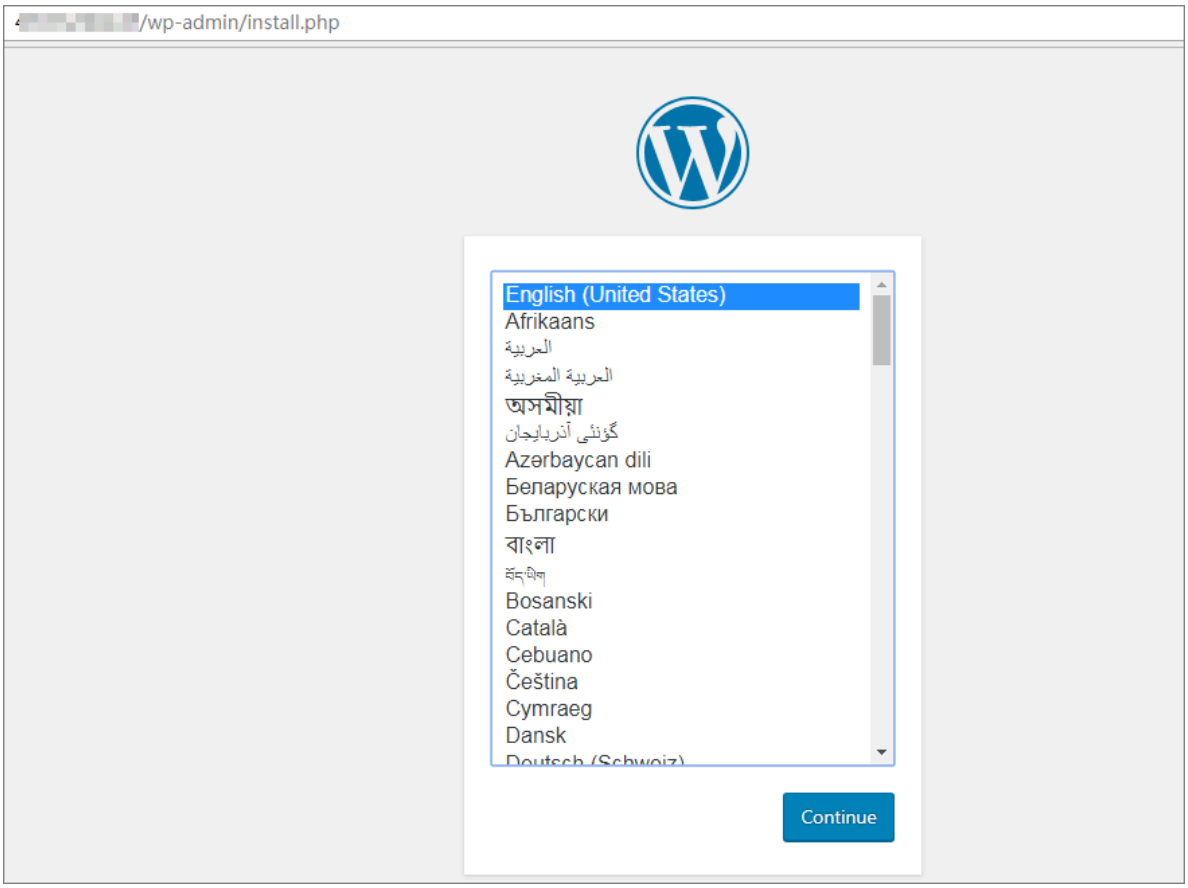
The yaml file used to create WordPress and mysql service is as follows:

```
 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 Ingresses and Load Balancing > Service 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 Deploy a microservice application with Helm

This topic describes two methods in which Helm can be used to deploy a microservice application in a Kubernetes cluster created by Alibaba Cloud Container Service for Kubernetes (ACK). In this topic, a PiggyMetrics application is deployed as the microservice application.

Deployment method overview

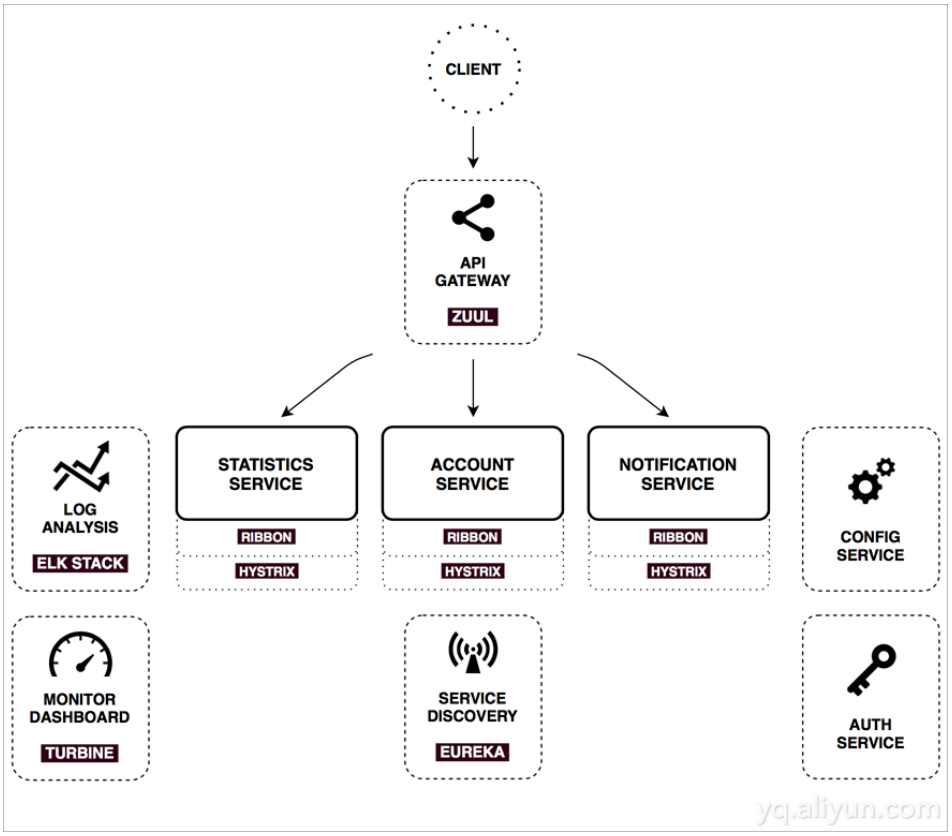
The following two methods can be used to deploy a microservice application with Helm:

- Method 1: Deploy a microservice application and the required basic components in a Kubernetes cluster
- Method 2: Deploy a microservice application in a Kubernetes cluster that contains the basic Spring Cloud components

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.

Method 1: Deploy a microservice application and the required basic components in a Kubernetes cluster

Prerequisites

- A Kubernetes cluster is created with ACK. For more information, see #unique_5.
- The Kubernetes cluster can be accessed by using kubectl. For more information, see #unique_29.
- Kompose and Helm are installed on your local host. For more information, see Kompose and Helm.

Overview

A PiggyMetrics application is defined by two Docker compose files: `docker-compose.yml` and `docker-compose.dev.yml`. Therefore, to deploy such an application in a Kubernetes cluster created by ACK, you must use Kompose to convert the two Docker compose files to a Kubernetes configuration file.

You can obtain the Docker compose files of a PiggyMetrics application at docker-compose and docker-compose.dev.

Procedure

1. Modify the Docker compose files of the PiggyMetrics application.

   a. Change the versions of these two Docker compose files from 2.1 to 2.

   > **Note:**
   > Kompose cannot convert Docker compose files of V2.1.

   b. In the file `docker-compose.yml`, find the following fields that is not supported by Kompose:

   ```
   depends_on:
     config:
       condition: service_healthy # condition is not supported
   ```

   c. In the file `docker-compose.yml`, replace the fields in the preceding step with the following:

   ```
   depends_on:
    - config
   labels:
     kompose.service.type: loadbalancer
   ```

   d. In the file `docker-compose.dev.yml`, change the external ports of the four MongoDB data bases used by the PiggyMetrics application to 27017.

   > **Note:**
   > The target PiggyMetrics application in this topic contains four MongoDB data bases, which are defined by four corresponding fields: `auth-mongodb`, `account-mongodb`, `statistics-mongodb`, and `notification-mongodb`.

   The following is a sample YAML file that is modified:

   ```
   version: '2'
   services:
     rabbitmq:
   ```

```
    image :  rabbitmq : 3 - management
    restart :  always
    labels :
      kompose . service . type :  nodeport
    ports :
     -  5672
     -  15672 : 15672
    logging :
      options :
        max - size : " 10m "
        max - file : " 10 "

  config :
    environmen  t :
      CONFIG_SER  VICE_PASSW  ORD : $ CONFIG_SER  VICE_PASSW  ORD
    image :  sqshq / piggymetri  cs - config
    restart :  always
    ports :
     -  8888
    logging :
      options :
        max - size : " 10m "
        max - file : " 10 "

  registry :
    environmen  t :
      CONFIG_SER  VICE_PASSW  ORD : $ CONFIG_SER  VICE_PASSW  ORD
    image :  sqshq / piggymetri  cs - registry
    restart :  always
    depends_on :
     -  config
    labels :
      kompose . service . type :  loadbalanc  er
    ports :
     -  8761 : 8761
    logging :
      options :
        max - size : " 10m "
        max - file : " 10 "

  gateway :
    environmen  t :
      CONFIG_SER  VICE_PASSW  ORD : $ CONFIG_SER  VICE_PASSW  ORD
    image :  sqshq / piggymetri  cs - gateway
    restart :  always
    depends_on :
     -  config
    labels :
      kompose . service . type :  loadbalanc  er
    ports :
     -  4000 : 4000
    logging :
      options :
        max - size : " 10m "
        max - file : " 10 "

  auth - service :
    environmen  t :
      CONFIG_SER  VICE_PASSW  ORD : $ CONFIG_SER  VICE_PASSW  ORD
      NOTIFICATI  ON_SERVICE  _PASSWORD : $ NOTIFICATI  ON_SERVICE
_PASSWORD
      STATISTICS  _SERVICE_P  ASSWORD : $ STATISTICS  _SERVICE_P
ASSWORD
```

```
      ACCOUNT_SE  RVICE_PASS  WORD : $ ACCOUNT_SE  RVICE_PASS
WORD
      MONGODB_PA  SSWORD : $ MONGODB_PA  SSWORD
    image :  sqshq / piggymetri  cs - auth - service
    restart :  always
    ports :
     -  5000
    depends_on :
     -  config
    logging :
      options :
        max - size : " 10m "
        max - file : " 10 "

  auth - mongodb :
    environmen  t :
      MONGODB_PA  SSWORD : $ MONGODB_PA  SSWORD
    image :  sqshq / piggymetri  cs - mongodb
    restart :  always
    ports :
     -  27017
    logging :
      options :
        max - size : " 10m "
        max - file : " 10 "

  account - service :
    environmen  t :
      CONFIG_SER  VICE_PASSW  ORD : $ CONFIG_SER  VICE_PASSW  ORD
      ACCOUNT_SE  RVICE_PASS  WORD : $ ACCOUNT_SE  RVICE_PASS
WORD
      MONGODB_PA  SSWORD : $ MONGODB_PA  SSWORD
    image :  sqshq / piggymetri  cs - account - service
    restart :  always
    ports :
     -  6000
    depends_on :
     -  config
    logging :
      options :
        max - size : " 10m "
        max - file : " 10 "

  account - mongodb :
    environmen  t :
      INIT_DUMP :  account - service - dump . js
      MONGODB_PA  SSWORD : $ MONGODB_PA  SSWORD
    image :  sqshq / piggymetri  cs - mongodb
    restart :  always
    ports :
     -  27017
    logging :
      options :
        max - size : " 10m "
        max - file : " 10 "

  statistics - service :
    environmen  t :
      CONFIG_SER  VICE_PASSW  ORD : $ CONFIG_SER  VICE_PASSW  ORD
      MONGODB_PA  SSWORD : $ MONGODB_PA  SSWORD
      STATISTICS  _SERVICE_P  ASSWORD : $ STATISTICS  _SERVICE_P
ASSWORD
    image :  sqshq / piggymetri  cs - statistics - service
    restart :  always
```

```
      ports :
       -  8888
      depends_on :
       -  config
      logging :
        options :
          max - size : " 10m "
          max - file : " 10 "

    statistics - mongodb :
      environmen  t :
        MONGODB_PA  SSWORD : $ MONGODB_PA  SSWORD
      image :  sqshq / piggymetri  cs - mongodb
      restart :  always
      ports :
       -  27017
      logging :
        options :
          max - size : " 10m "
          max - file : " 10 "

    notificati  on - service :
      environmen  t :
        CONFIG_SER  VICE_PASSW  ORD : $ CONFIG_SER  VICE_PASSW  ORD
        MONGODB_PA  SSWORD : $ MONGODB_PA  SSWORD
        NOTIFICATI  ON_SERVICE  _PASSWORD : $ NOTIFICATI  ON_SERVICE
_PASSWORD
      image :  sqshq / piggymetri  cs - notificati  on - service
      restart :  always
      ports :
       -  8000
      depends_on :
       -  config
      logging :
        options :
          max - size : " 10m "
          max - file : " 10 "

    notificati  on - mongodb :
      image :  sqshq / piggymetri  cs - mongodb
      restart :  always
      environmen  t :
        MONGODB_PA  SSWORD : $ MONGODB_PA  SSWORD
      ports :
       -  27017
      logging :
        options :
          max - size : " 10m "
          max - file : " 10 "

    monitoring :
      environmen  t :
        CONFIG_SER  VICE_PASSW  ORD : $ CONFIG_SER  VICE_PASSW  ORD
      image :  sqshq / piggymetri  cs - monitoring
      restart :  always
      depends_on :
       -  config
      labels :
        kompose . service . type :  loadbalanc  er
      ports :
       -  9000 : 8080
       -  8989 : 8989
      logging :
        options :
```

```
        max - size : " 10m "
        max - file : " 10 "
```

2. Use Kompose to generate the Kubernetes configuration file to deploy the PiggyMetrics application.

   a. Run the following commands to set the required environments to deploy the application:

```
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
```

   b. Run the following command to convert the Docker compose files to a Kubernetes configuration file:

```
kompose   convert  - f   docker - compose . yml  - f   docker -
compose . dev . yml  - o   piggymetri  cs  - c
```

3. Run the `helm install` command deploy the PiggyMetrics application in the target Kubernetes cluster.

   For example, you can run the following command to deploy an application named `piggy` in the `pm` namespace:

```
helm   install  -- namespace   pm  -- name   piggy   piggymetri  cs
/
```

4. Ensure that the version of your local Helm client is the same as the version of the remote Helm server.

   Note:
   If the versions of your local Helm client and the remote Helm server do not match with each, the following error messaged is displayed:

```
Error :  incompatib  le   versions   client [ v2 . 14 . 1 ]
server [ v2 . 11 . 0 ]
```

   If this error message is not displayed, you can directly ignore this step.

   - If you use the MAC OS, run the following commands to obtain the version 2.11.0 of the Helm client:

```
brew   unlink   kubernetes - helm
```

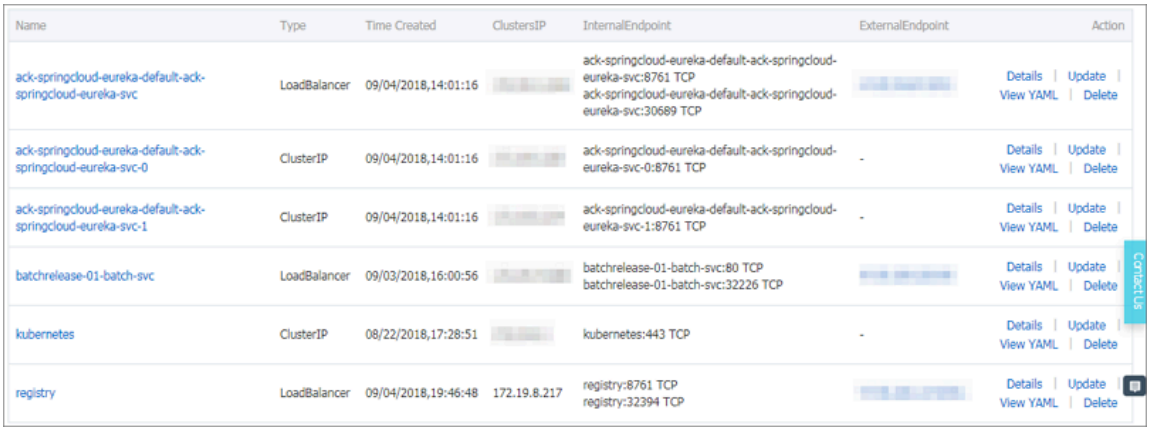
```
brew install https ://raw . githubuser content . com /
Homebrew / homebrew - core / ee94af7477 8e48ae103a 9fb080e26a
6a2f62d32c / Formula / kubernetes - helm . rb
```

- If you use the Window or Linux OS, install the corresponding version of the Helm client.

5. Use the IP address of the gateway service to access the deployed application.

> **Note:**
> The gateway service is one of the services provided by the target application.

a. Log on to the Container Service console.
b. In the left-side navigation pane under Container Service-Kubernetes, choose Ingresses and Load Balancing > Service.
c. Select the target cluster and namespace to find the the IP address of the gateway service.



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

6. Run the `helm delete -- purge` command to remove the deployed application.

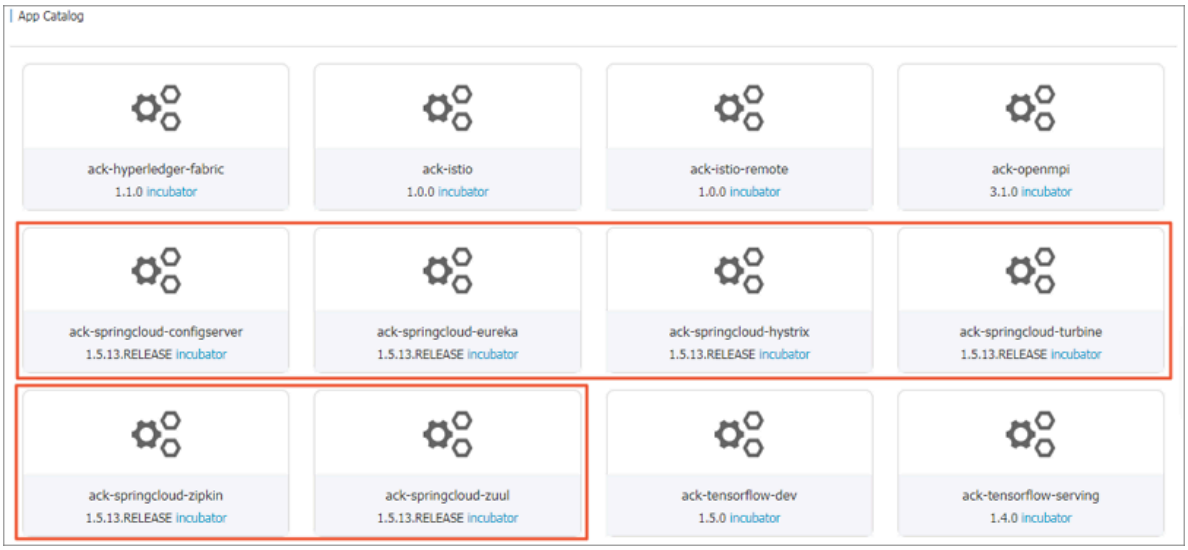
   In this topic, to remove the PiggyMetrics application named `piggy`, run the following command:

```
helm delete -- purge piggy
```

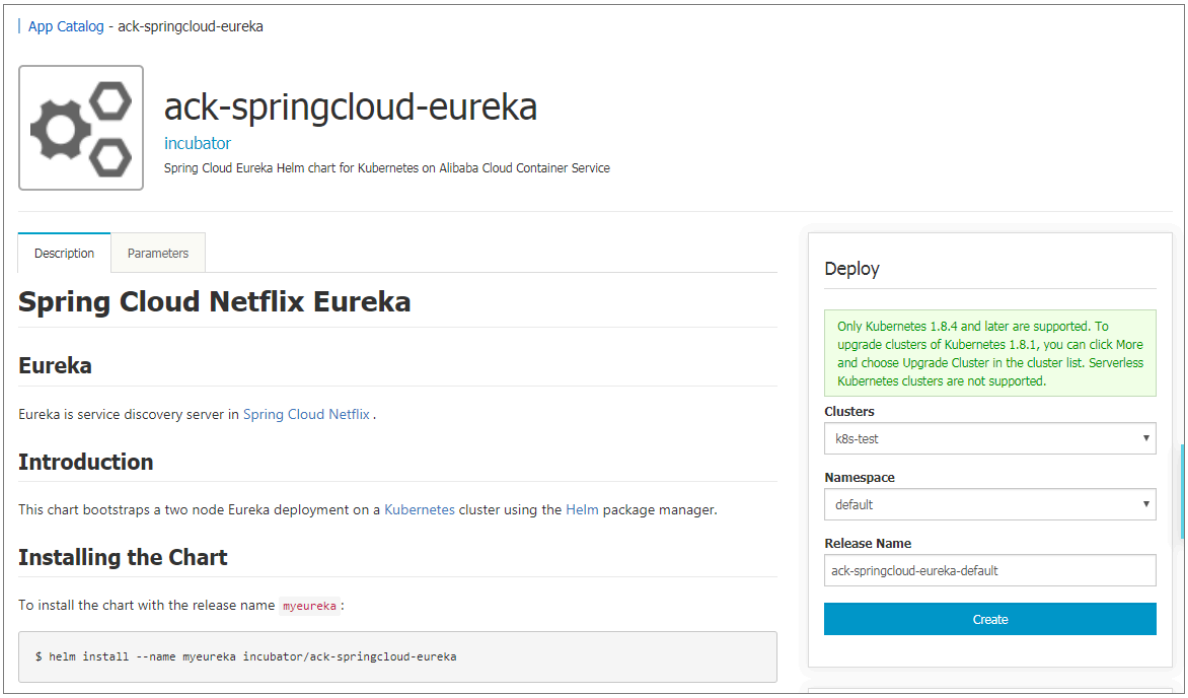
Method 2: Deploy the application in a Kubernetes cluster that contains the basic Spring Cloud components

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.

1. Log on to the Container Service console.
2. In the left-side navigation pane under Container Service-Kubernetes, choose Marketplace > App Catalog.



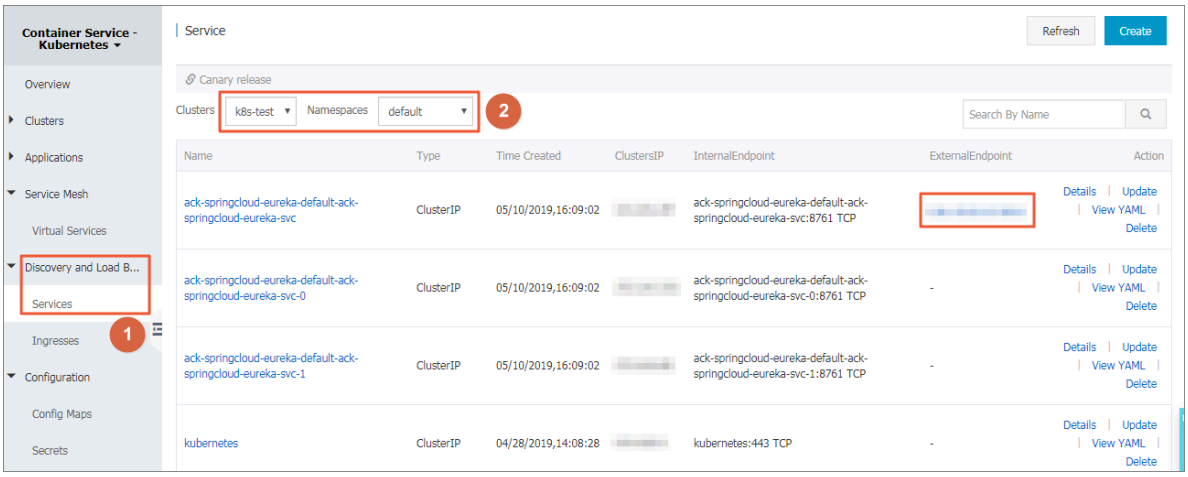
3. Click ack-springcloud-eureka.
4. In the Deploy area, select the target cluster. Then, click Create.



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

Note:

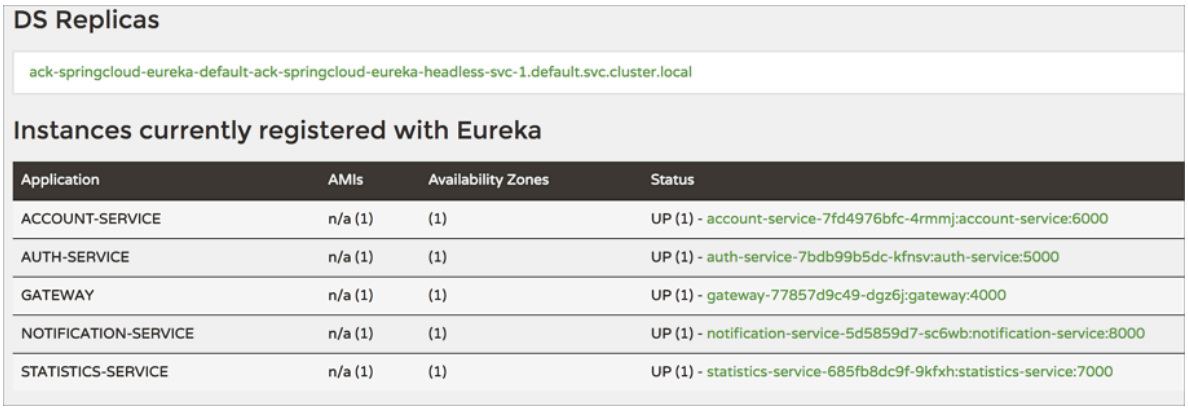
The service `ack-springcloud-eureka-default-ack-springcloud-eureka-svc` exposes Eureka by using its IP address.



6. Access the registry service to verify that all PiggyMetrics services are properly registered with EurekaServer.

Note:

- The PiggyMetrics application is deployed to the EurekaServer environment. Use `GATEWAY` to log on to the application.
- You can also deploy a PiggyMetrics application that does not contain Eureka by following the procedures described in Method 1.



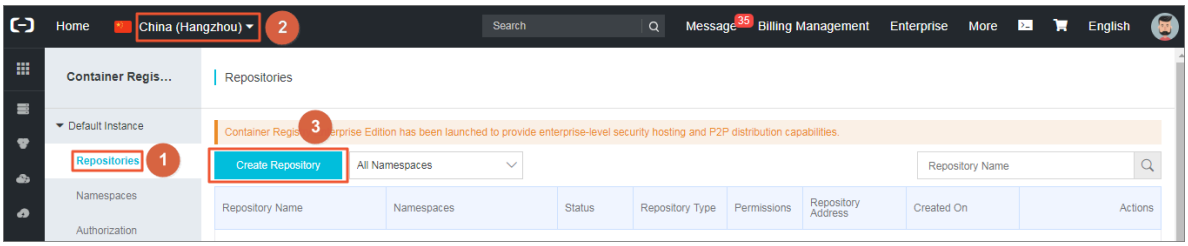
3.2 Use a private image repository to create an application

This topic describes how to create a private image repository in the Alibaba Cloud Container Registry console and how to use an image in this image repository to create an application in the Container Service console.

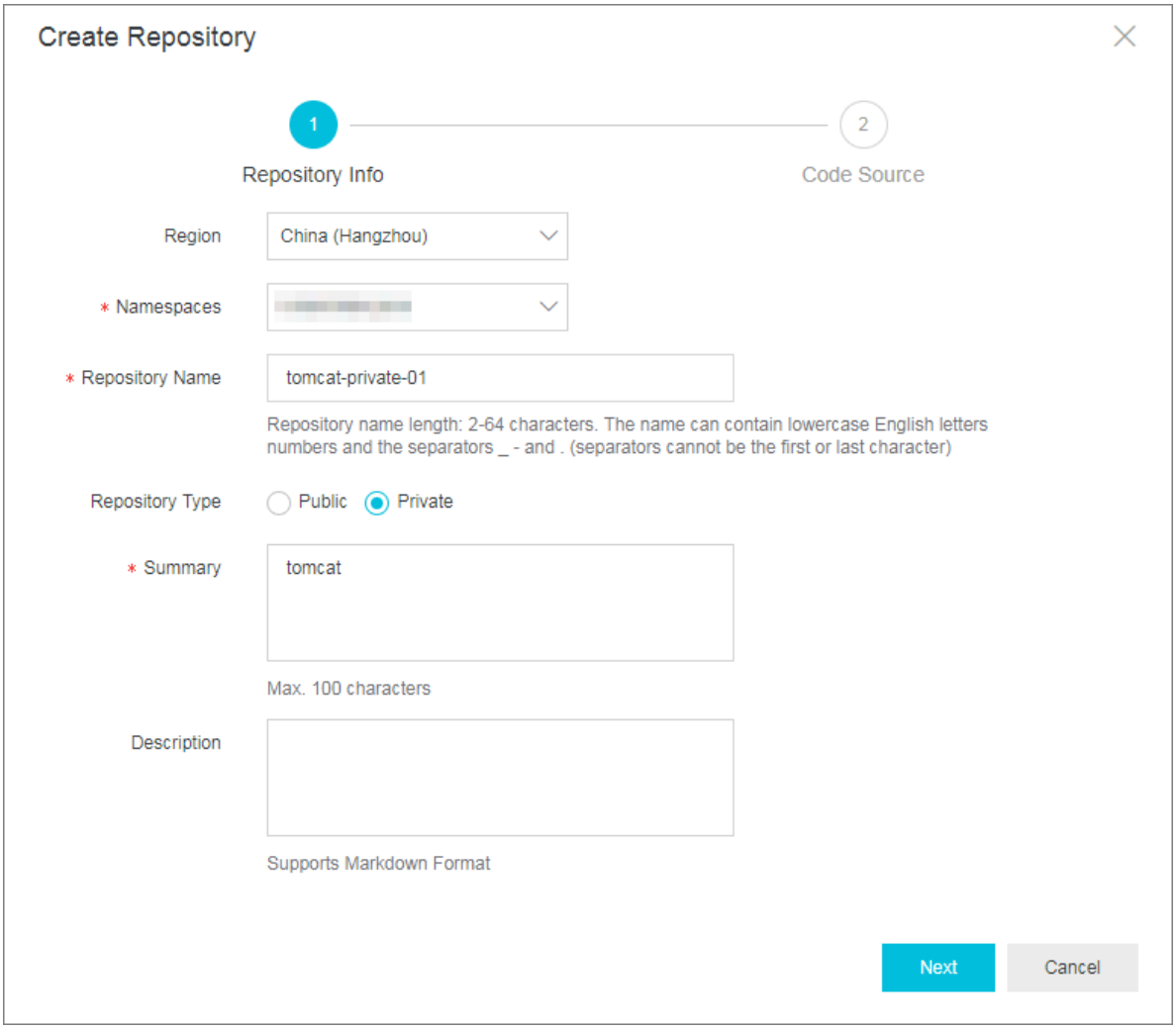
Step 1: Create a private image repository

> **Notice:**
> If you use Alibaba Cloud Container Registry, the system prompts you to set the logon password. You must click auto enable and then set the registry password.

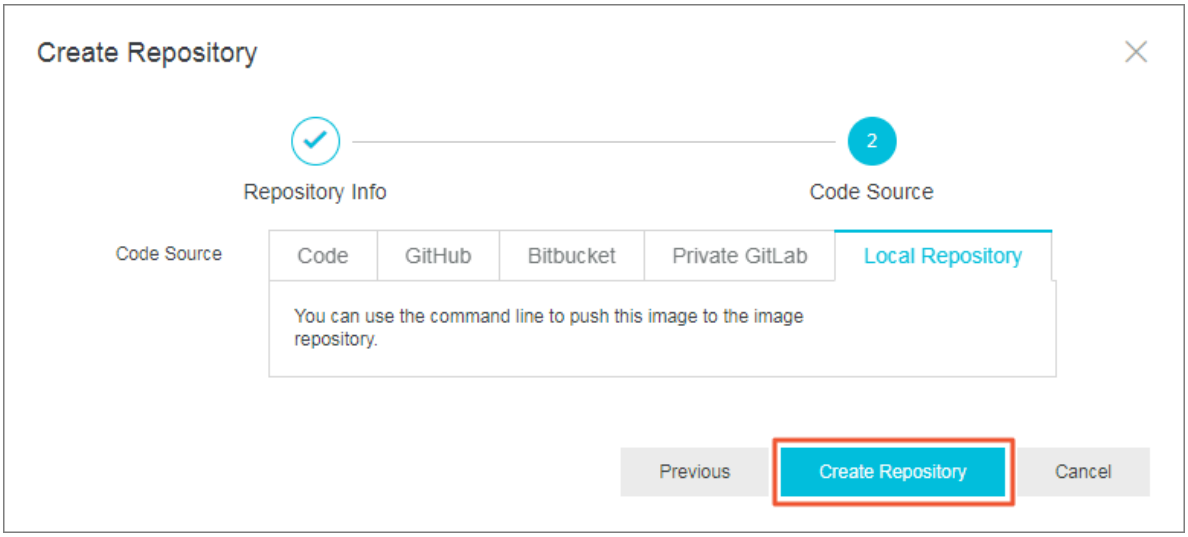
1. Log on to the Container Registry console.
2. In the left-side navigation pane, click Repositories. Select the target region, and then click Create Repository.



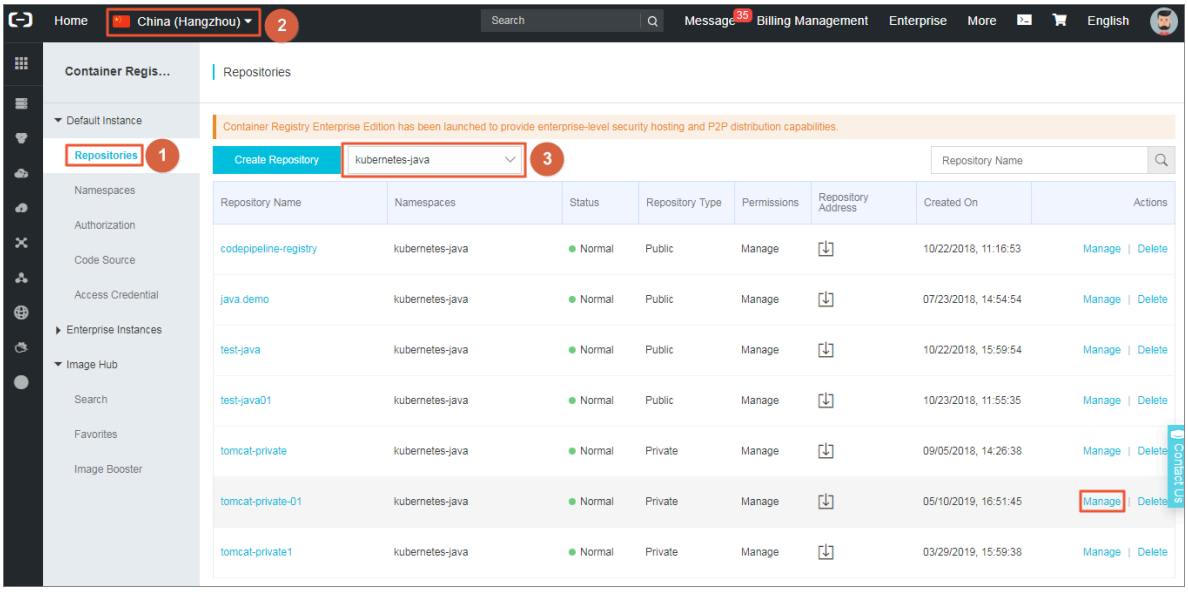
3. In the displayed dialog box, set the following parameters for the image repository: namespace, repository name, repository type, and summary. Then, click Next.



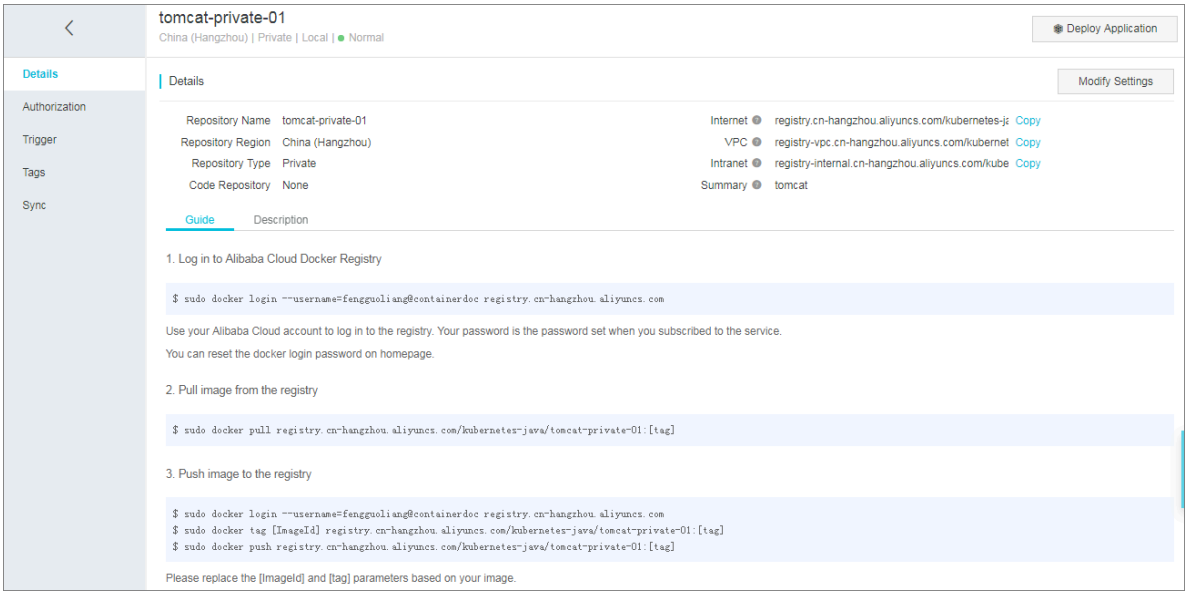
4. Select Local Repository, and then click Create Repository.



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 run the following commands to 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
```

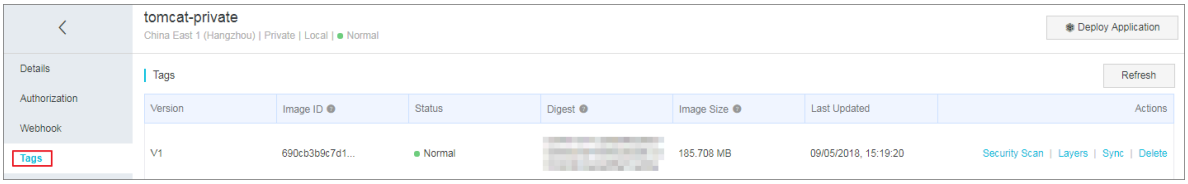
```
tomcat
          latest                2d43521f2b  1a          6   days
ago            463MB

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

The following is the output:

```
The   push   refers   to   a   repository  [ registry . cn -
hangzhou . aliyuncs . com / XXX / 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 ****  size :  2836
```

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

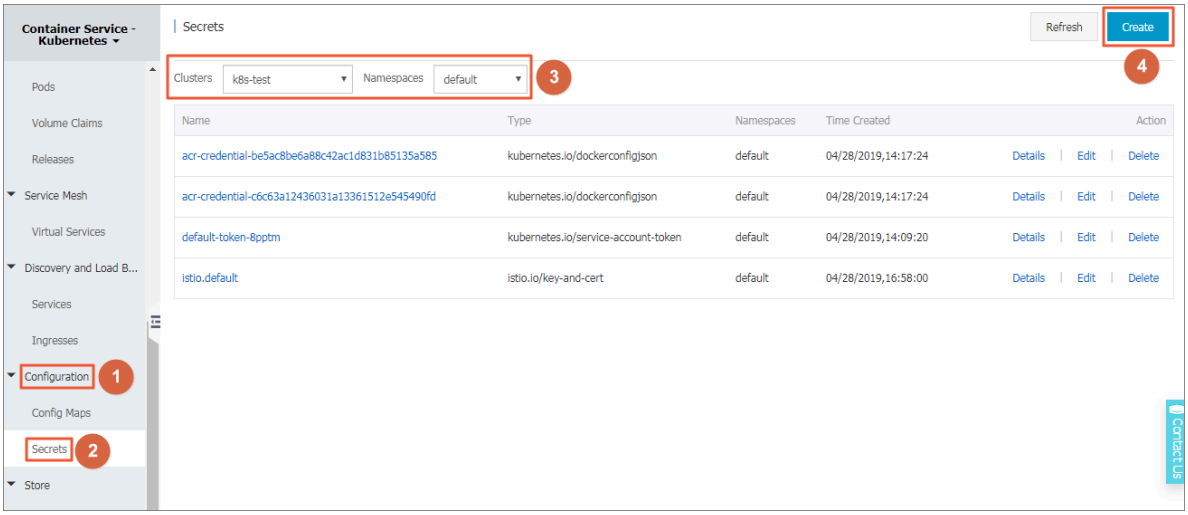
> **Note:**
> The information related to the image such as the image version number and image ID is displayed on the page.



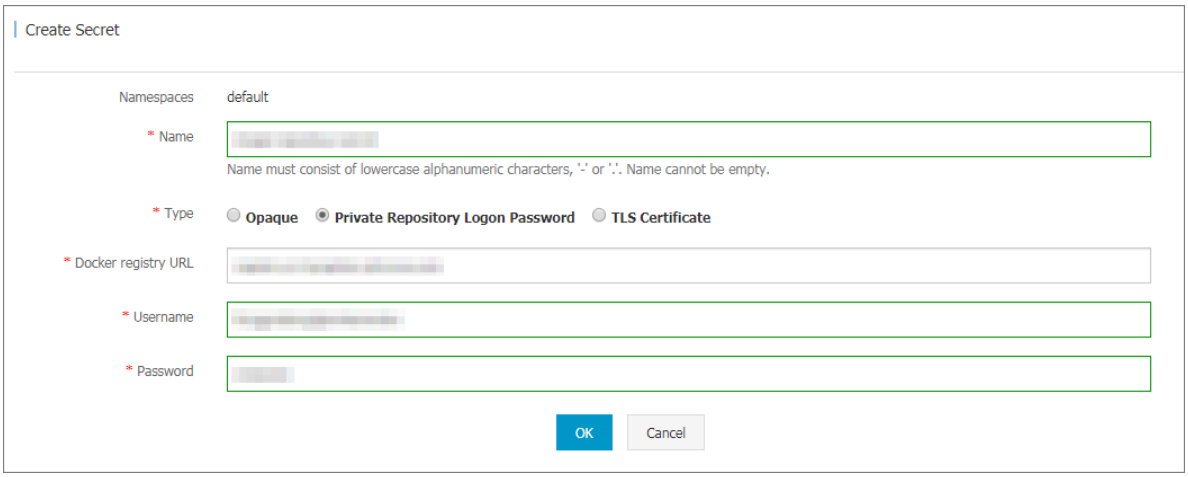
Step 2: Create a secret for logging on to a private image repository

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

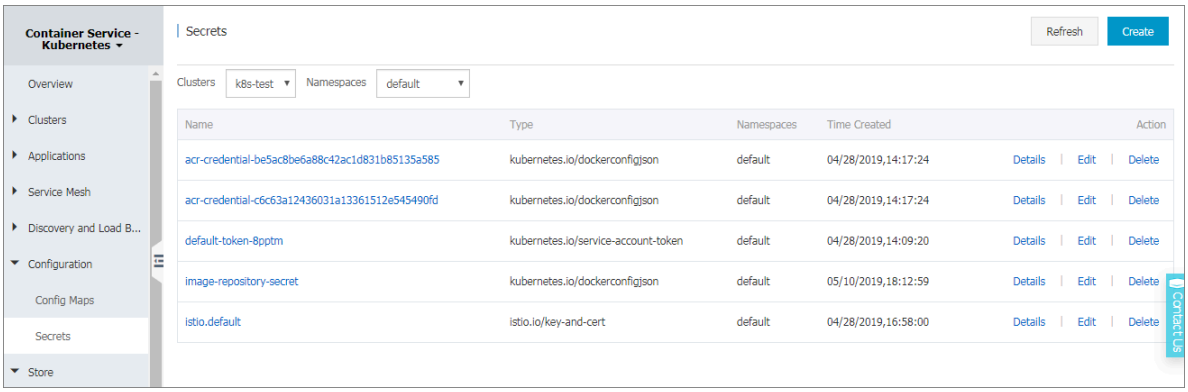
3. Select the target cluster and namespace, and then click Create in the upper-right corner.



4. Set a secret.



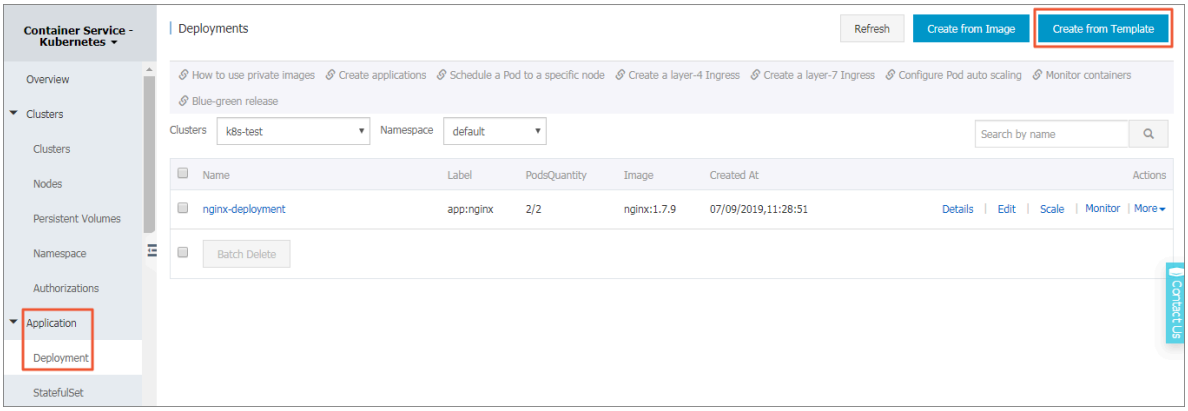
Then, the newly created secret is displayed in the secret list.



You can also use the kubectl CLI to create a secret for logging on to a private image repository. For more information, see #unique_29.

Step 3: Use a private image repository to create an application

1. Log on to the Container Service console.
2. In the left-side navigation pane under Container Service-Kubernetes, choose Applications > Deployments.
3. Select the target cluster and namespace, and then click Create from Template.



> **Note:**
> You can also create an application by clicking Create from Image. For more information, see #unique_42.

4. Select Custom from the Sample Template drop-down list, copy the following code and paste it to the Template area, and then click DEPLOY.

```
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:

You must replace the sample image URL in the preceding code with your private image URL.

For more information, see Use a private repository.