Alibaba Cloud
Container Service for Kubernetes

quickstart

Issue: 20181113

MORE THAN JUST CLOUD | Alibaba Cloud

Legal disclaimer

Generic conventions

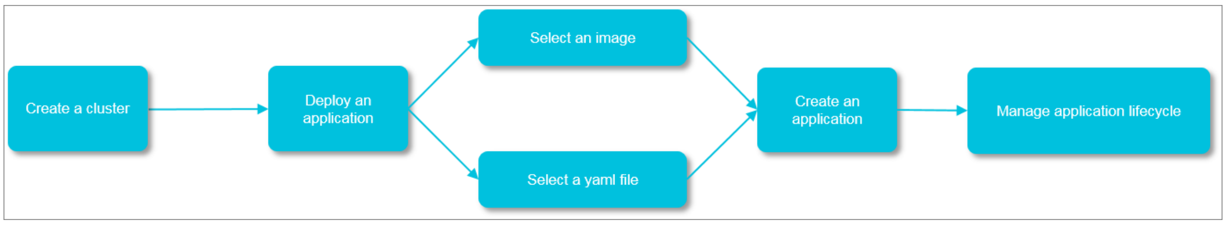
Table -1: Style conventions

Style	Description	Example
⛔	This warning information indicates a situation that will cause major system changes, faults, physical injuries, and other adverse results.	⛔ **Danger:** Resetting will result in the loss of user configuration data.
⚠	This warning information indicates a situation that may cause major system changes, faults, physical injuries, and other adverse results.	⚠ **Warning:** Restarting will cause business interruption. About 10 minutes are required to restore business.
📋	This indicates warning information, supplementary instructions, and other content that the user must understand.	📋 **Note:** 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]`*
{} 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 quickly

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.

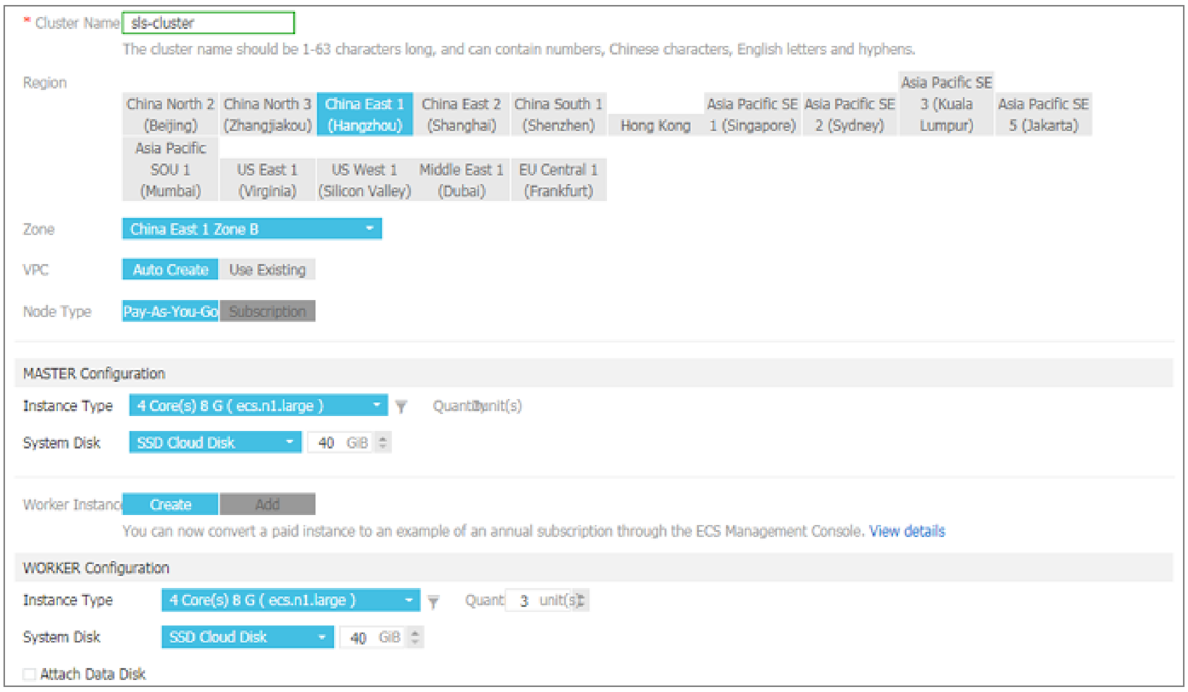
Context

This example shows how to quickly 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.



* Cluster Name: sls-cluster

The cluster name should be 1-63 characters long, and can contain numbers, Chinese characters, English letters and hyphens.

Region: China North 2 (Beijing) | China North 3 (Zhangjiakou) | [✓ China East 1 (Hangzhou)] | China East 2 (Shanghai) | China South 1 (Shenzhen) | Hong Kong | Asia Pacific SE 1 (Singapore) | Asia Pacific SE 2 (Sydney) | Asia Pacific SE 3 (Kuala Lumpur) | Asia Pacific SE 5 (Jakarta) | Asia Pacific SOU 1 (Mumbai) | US East 1 (Virginia) | US West 1 (Silicon Valley) | Middle East 1 (Dubai) | EU Central 1 (Frankfurt)

Zone: China East 1 Zone B

VPC: [✓ Auto Create] Use Existing

Node Type: [✓ Pay-As-You-Go] Subscription

MASTER Configuration

Instance Type: 4 Core(s) 8 G (ecs.n1.large) Quantity 3 unit(s)

System Disk: SSD Cloud Disk 40 GiB

Worker Instance: [✓ Create] Add

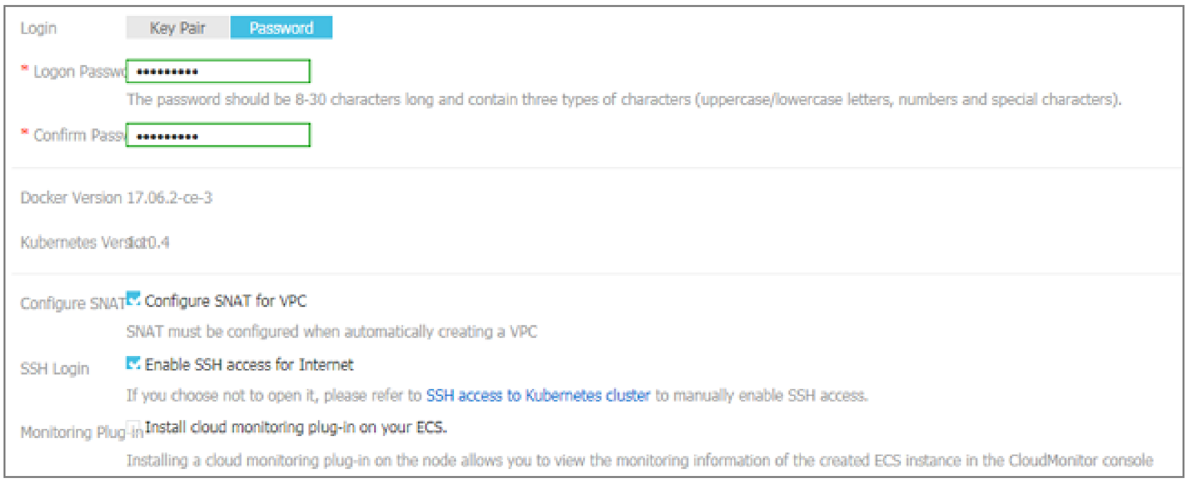
You can now convert a paid instance to an example of an annual subscription through the ECS Management Console. View details

WORKER Configuration

Instance Type: 4 Core(s) 8 G (ecs.n1.large) Quant 3 unit(s)

System Disk: SSD Cloud Disk 40 GiB

Attach Data Disk



Login: Key Pair [✓ Password]

* Logon Password: ••••••••••

The password should be 8-30 characters long and contain three types of characters (uppercase/lowercase letters, numbers and special characters).

* Confirm Password: ••••••••••

Docker Version 17.06.2-ce-3

Kubernetes Version 1.10.4

Configure SNAT: ✓ Configure SNAT for VPC

SNAT must be configured when automatically creating a VPC

SSH Login: ✓ Enable SSH access for Internet

If you choose not to open it, please refer to SSH access to Kubernetes cluster to manually enable SSH access.

Monitoring Plug-in: Install cloud monitoring plug-in on your ECS.

Installing a cloud monitoring plug-in on the node allows you to view the monitoring information of the created ECS instance in the CloudMonitor console

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.
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

Configuration	Description
	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.
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*

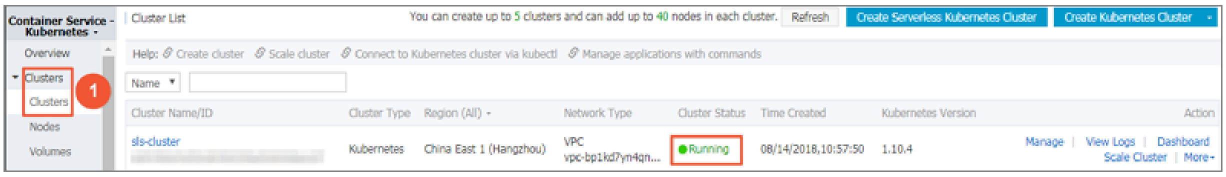
Configuration	Description
Pod Network CIDR and Service CIDR (optional)	For more information about the specific plan, see *Plan Kubernetes CIDR blocks under VPC*. **Note:** This option is available when you select **Use Existing** VPC.
Configure SNAT	SNAT must be configured if you select **Auto Create** 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*.
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 plug-in for my Kubernetes cluster network?*.

Configuration	Description
	• 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.


Container Service - Kubernetes
Overview
Clusters
Clusters
Nodes
Volumes
1
Refresh
Create Serverless Kubernetes Cluster
Create Kubernetes Cluster
Name
Manage | View Logs | Dashboard
Scale Cluster | More

Cluster List

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

Help: Create cluster Scale cluster Connect to Kubernetes cluster via kubectl Manage applications with commands

Cluster Name/ID	Cluster Type	Region (All)	Network Type	Cluster Status	Time Created	Kubernetes Version	Action
sls-cluster	Kubernetes	China East 1 (Hangzhou)	VPC vpc-bp1kd7yn4qn...	● Running	08/14/2018,10:57:50	1.10.4	Manage | View Logs | Dashboard Scale Cluster | More

Now you have quickly created a Kubernetes cluster.

2.2 Create a deployment application by using an image

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

Prerequisites

Create a Kubernetes cluster. For more information, see *Create a Kubernetes cluster*.

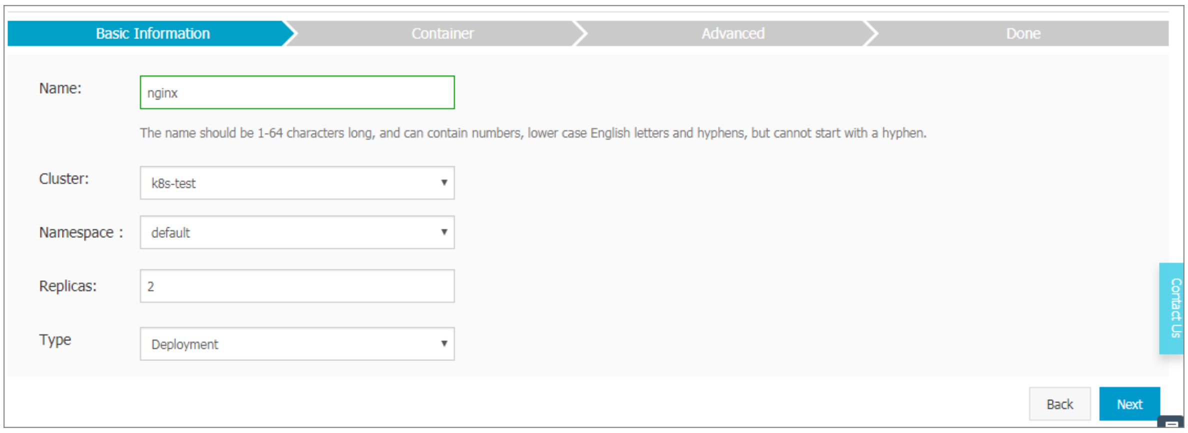
Procedure

1. Log on to the *Container Service console*.
2. Under Kubernetes, click **Application** > **Deployment** in the left-side navigation pane, and then click **Create by image** in the upper-right corner.
3. Configure **Name**, **Cluster**, **Namespace**, **Replicas**, and **Type**. The configured value of the replicas parameter specifies the number of pods contained in the application. Click **Next**.

   Note:

   In this example, select the **Deployment** type.

   If you do not configure **Namespace**, the system uses the default namespace by default.



Name: nginx

The name should be 1-64 characters long, and can contain numbers, lower case English letters and hyphens, but cannot start with a hyphen.

Cluster: k8s-test

Namespace : default

Replicas: 2

Type Deployment

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.

  Besides, you can 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 configured image is consistent with that of the cached image, Container Service reuses the cached image rather than pull the same image 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 an image when deploying an application so as to make sure the latest image and codes are 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/*.

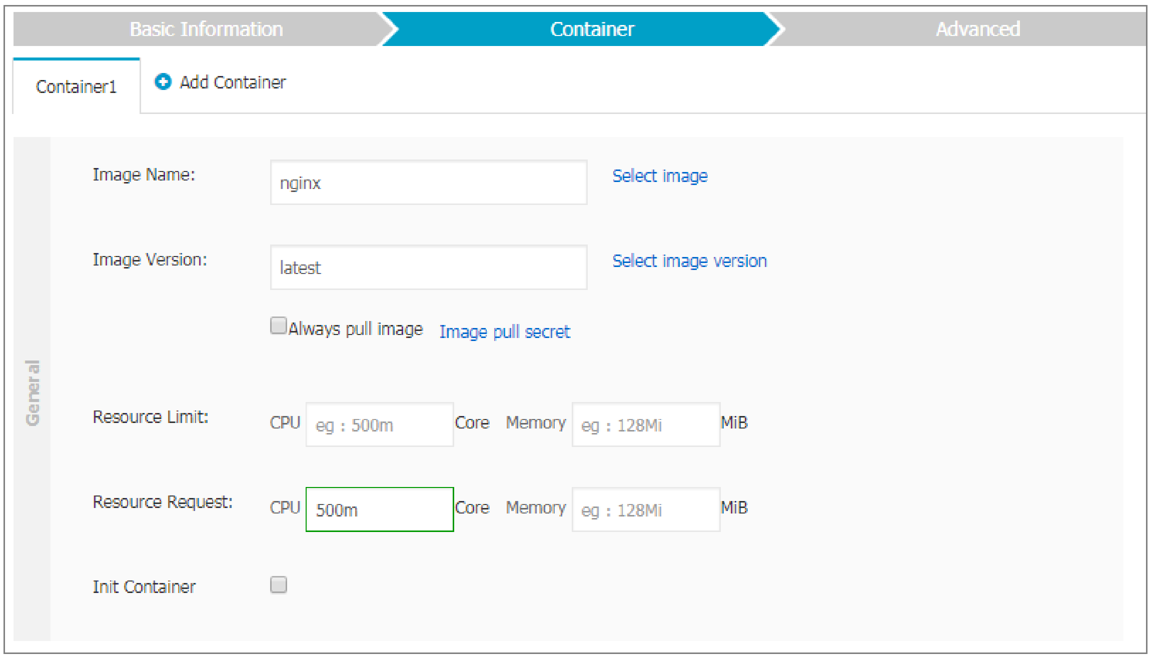


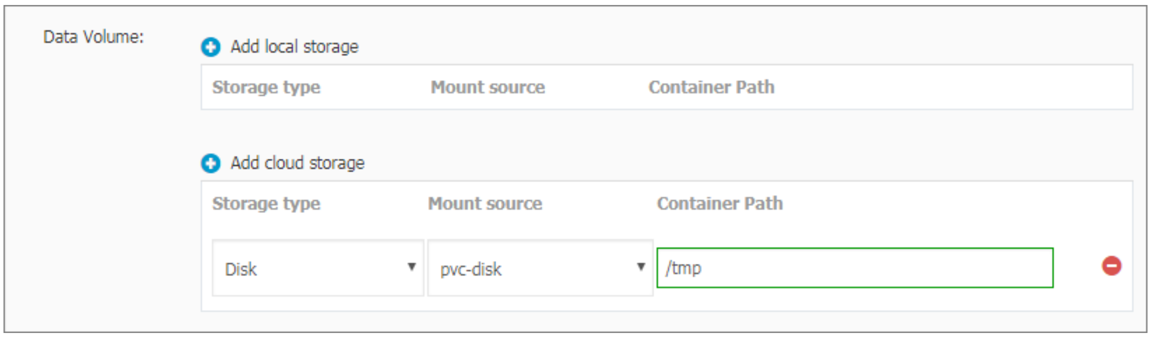
Image Name: nginx
Image Version: latest
Always pull image
Resource Limit: CPU eg : 500m Core Memory eg : 128Mi MiB
Resource Request: CPU 500m Core Memory eg : 128Mi MiB
Init Container

b) Optional: 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 cloud disk as the data volume and mount the cloud disk to the `/tmp` container path. Then container data generated in this path are stored to the cloud disk.



Storage type	Mount source	Container Path

Storage type	Mount source	Container Path
Disk	pvc-disk	/tmp

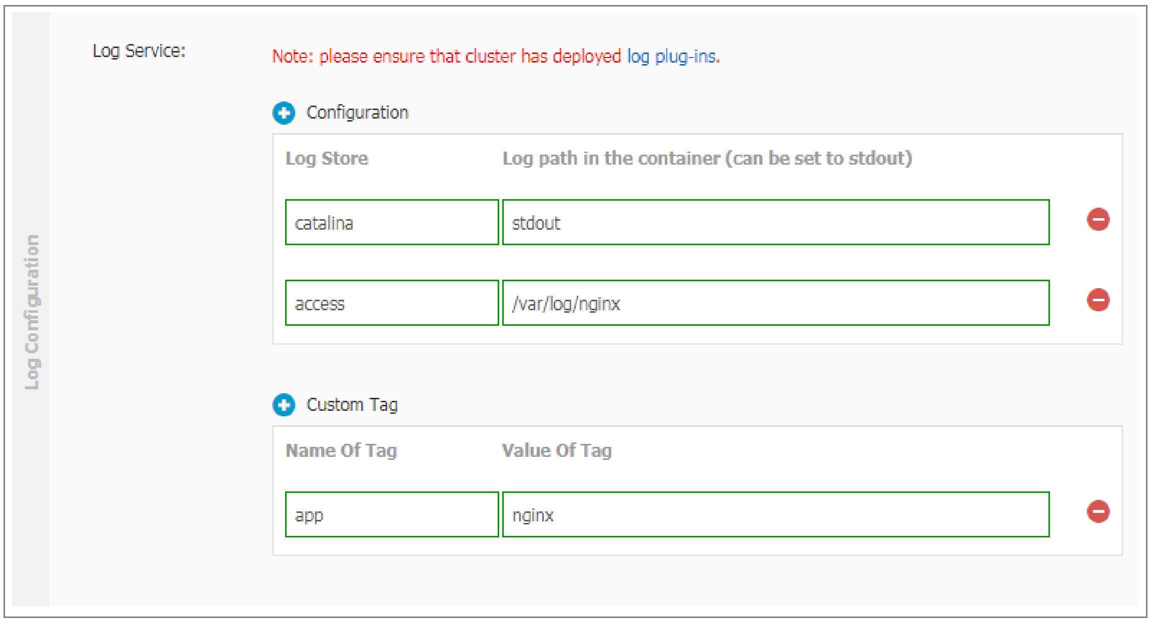
c) Optional: Configure **Log Service**. You can configure collection methods and customize tags for this service.

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 configured custom tags. The customized tags are collected to the container output logs. A custom tag can help you tag container logs, providing convenience to log analysis such as log statistics and filter.



Log Service: Note: please ensure that cluster has deployed log plug-ins.

Log Store	Log path in the container (can be set to stdout)
catalina	stdout
access	/var/log/nginx

Name Of Tag	Value Of Tag
app	nginx

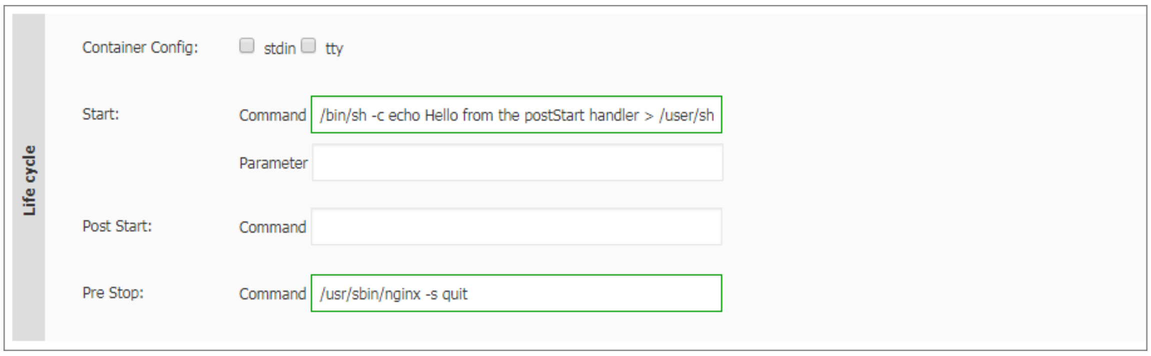
d) Optional: Configure environment variables.

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*.

e) Configure the lifecycle rule.

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

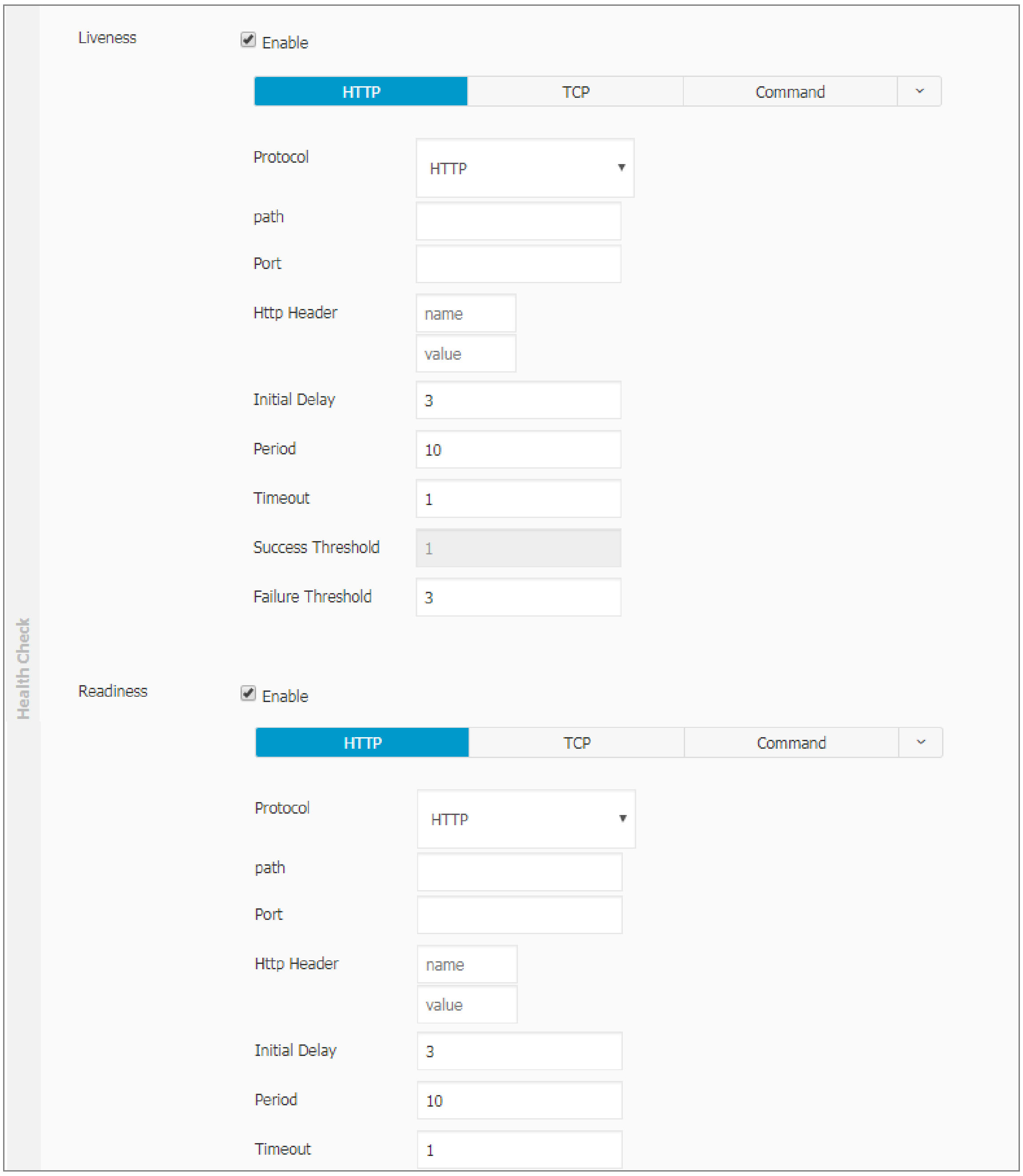
- **Container Config**: Select the stdin check box to enable standard input for the container. Select the tty check box to assign an virtual terminal to for the container to send signals to the container. These two options are usually used together, which indicates to bind the terminal (tty) to the container standard input (stdin). For example, an interactive program obtains standard input from you and then displays the obtained standard input in the terminal.
- **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.



Container Config:	☐ stdin ☐ tty	
Start:	Command	/bin/sh -c echo Hello from the postStart handler > /user/sh
	Parameter	
Post Start:	Command	
Pre Stop:	Command	/usr/sbin/nginx -s quit

f) 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*.



Health Check

Liveness ☑ Enable

HTTP | TCP | Command

Protocol: HTTP

path:

Port:

Http Header: name / value

Initial Delay: 3

Period: 10

Timeout: 1

Success Threshold: 1

Failure Threshold: 3

Readiness ☑ Enable

HTTP | TCP | Command

Protocol: HTTP

path:

Port:

Http Header: name / value

Initial Delay: 3

Period: 10

Timeout: 1

Request method	Configuration 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.

Request method	Configuration description
	• HTTP Header: Custom headers in the HTTP request. HTTP allows repeated headers. Supports key-value configurat ion. • 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.
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

Request method	Configuration description
	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.
Command	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.

Request method	Configuration description
	• 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.

5. Click **Next** after completing the configurations.
6. Configure advanced settings.

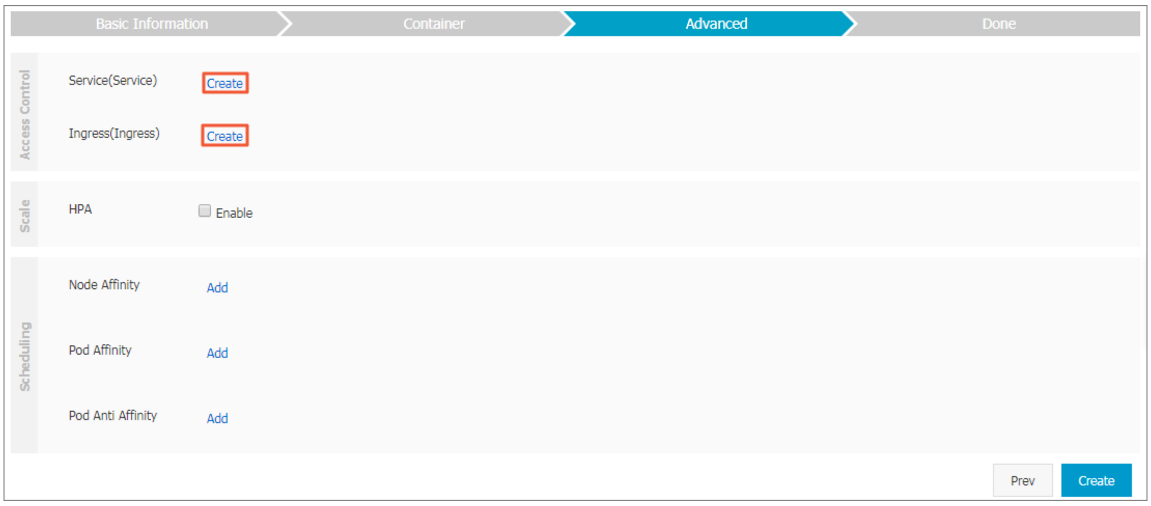
    a) Configure **Access Control**.

    You can configure how to expose the backend pod and click **Create**. In this example, select Cluster IP Service and Ingress to create an nginx application that is accessible for Internet.

    Note:

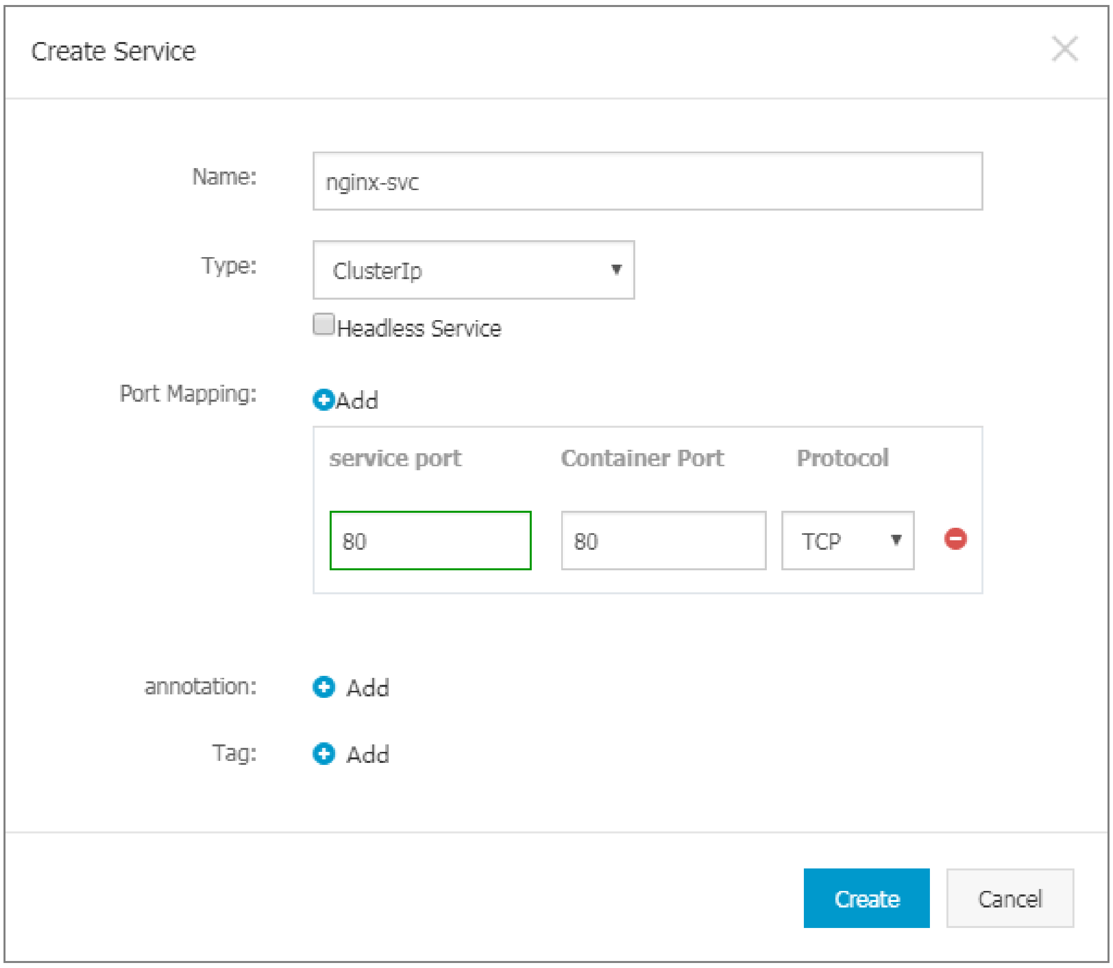
    To meet communication requirements of the application, you can configure access control based on your needs:

    - Internal applications: For applications that work only inside a cluster, you can create services of Cluster IP or Node Port for internal communication as needed.
    - External applications: For applications that need to be exposed to Internet, you can configure access control by using one of the following methods:
        - Create a service of Server Load Balancer: Use the Server Load Balancer (SLB) service provided by Alibaba Cloud, which provides Internet accessibility for the application.
        - Create a service of ClusterIP or NodePort, and create Ingress: This method provides Internet accessibility through ingress. For more information, see *https://kubernetes.io/docs/concepts/services-networking/ingress/*.



Service(Service) Create
Ingress(Ingress) Create
HPA ☐ Enable
Node Affinity Add
Pod Affinity Add
Pod Anti Affinity Add

1. Click **Create** at right of Service. Configure a service in the displayed dialog box, and then click **Create**.



Name: nginx-svc
Type: ClusterIp
☐ Headless Service
Port Mapping:

service port	Container Port	Protocol
80	80	TCP

annotation: Add
Tag: Add

- **Name**: You can enter your custom name. The default is `applicationname-svc`.
- **Type**: Select one from the following three service types.

- ClusterIP: Expose the service by using the internal IP address of your cluster. With this type selected, the service is accessible only within the cluster.
- NodePort: Expose the service by using the IP address and static port (NodePort) on each node. A NodePort service routes to a ClusterIP service, which is automatically created. You can access the NodePort service outside the cluster by requesting`<NodeIP>:<NodePort>`.
- Server Load Balancer: The Server Load Balancer service, which is provided by Alibaba Cloud. You can configure Internet access or intranet access by using this type of service. Server Load Balancer can route to the NodePort service and ClusterIP service.

• **Port Mapping**: Add a service port and a container port. If you select NodePort for **Type**, you must configure a node port to avoid port conflicts. TCP and UDP protocols are supported.

• **annotation**: Add an annotation to the service. Server Load Balancer configuration parameters are supported, see *Access services by using Server Load Balancer*.

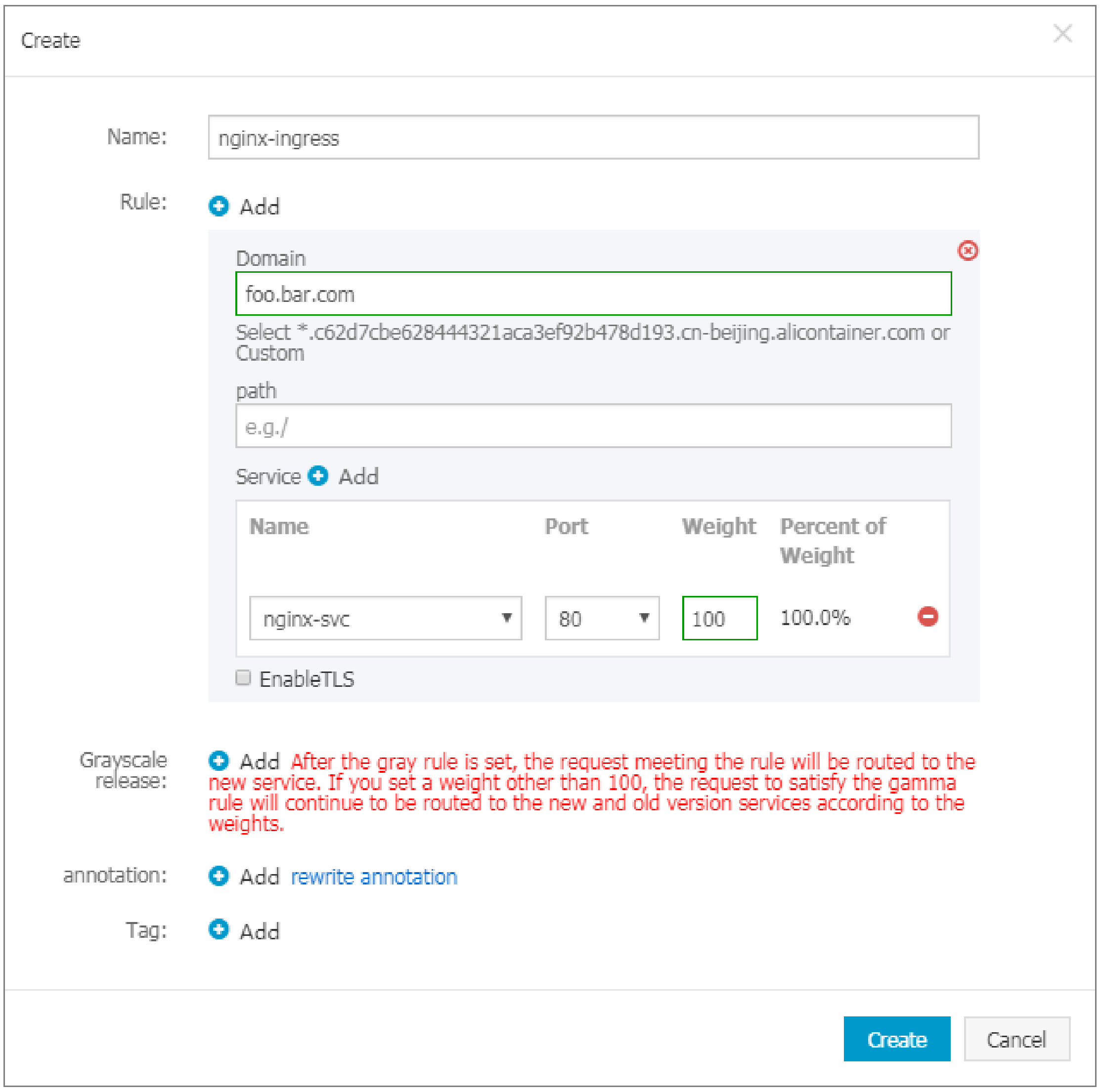
• **Label**: You can add a label to the service to identify the service.

2. Click **Create** at the right of Ingress. Configure rout rules for the backend pod in the displayed dialog box, and then click **Create**. For more information about route configuration, see *Ingress configurations*.

Note:

When you create an application by using an image, you can create ingress for only one service. In this example, use a virtual host name as the testing domain name. You need to add a record to the hosts. In actual work scenarios, use a filing domain name.

```
101.37.224.146   foo.bar.com    #the IP address of ingress
```



Create

Name: nginx-ingress

Rule: Add

Domain: foo.bar.com

Select *.c62d7cbe628444321aca3ef92b478d193.cn-beijing.alicontainer.com or Custom

path: e.g./

Service Add

Name	Port	Weight	Percent of Weight
nginx-svc	80	100	100.0%

EnableTLS

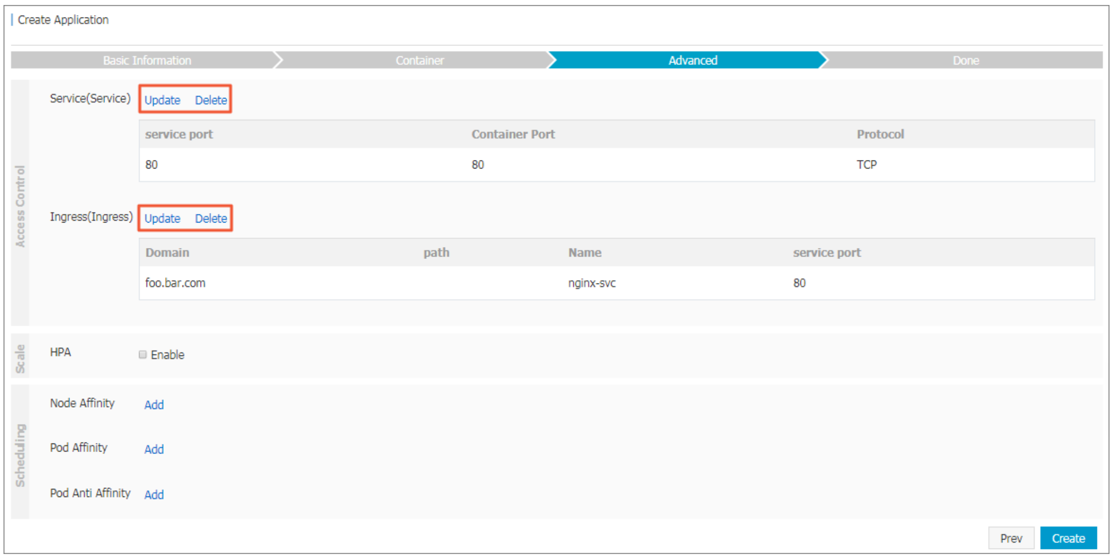
Grayscale release: Add After the gray rule is set, the request meeting the rule will be routed to the new service. If you set a weight other than 100, the request to satisfy the gamma rule will continue to be routed to the new and old version services according to the weights.

annotation: Add rewrite annotation

Tag: Add

Create Cancel

3. The created service and ingress are displayed in the access control section. You can reconfigure the service and ingress by clicking **Update** and **Delete**.



Create Application

Basic Information > Container > Advanced > Done

Access Control

Service(Service) Update Delete

service port	Container Port	Protocol
80	80	TCP

Ingress(Ingress) Update Delete

Domain	path	Name	service port
foo.bar.com		nginx-svc	80

Scale

HPA ☐ Enable

Scheduling

Node Affinity Add

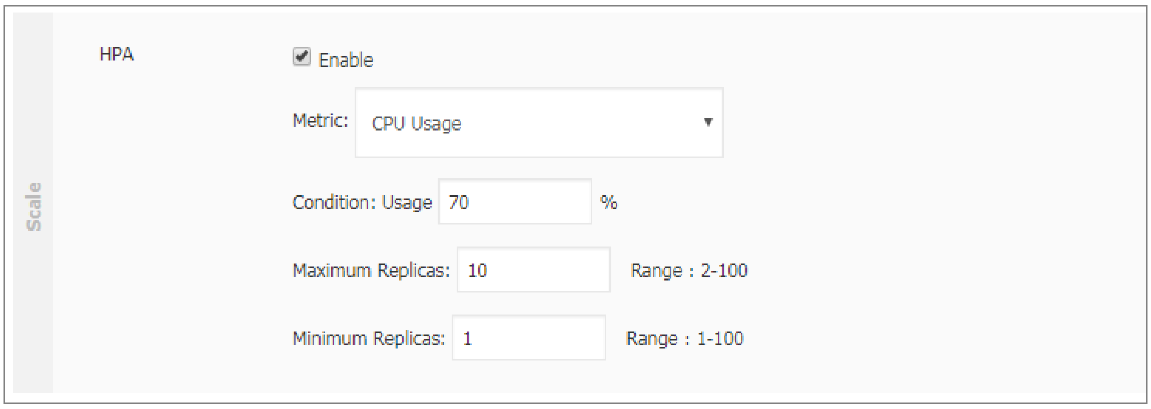
Pod Affinity Add

Pod Anti Affinity Add

Prev Create

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

You can choose whether to enable **HPA**. To meet the demands of applications under different loads, Container Service supports the container auto scaling, which automatically adjusts the number of containers according to the container CPU and memory usage.



Scale

HPA ☑ Enable

Metric: CPU Usage

Condition: Usage 70 %

Maximum Replicas: 10 Range : 2-100

Minimum Replicas: 1 Range : 1-100

Note:
To enable auto scaling, you must configure required resources for the deployment. Otherwise, the container auto scaling cannot take effect. See the basic configuration of containers.

- **Metric**: CPU and memory. Configure a resource type as needed.
- **Condition**: The percentage value of resource usage. The container begins to expand when the resource usage exceeds this value.
- **Maximum Replicas**: The maximum number of replicas that the deployment can expand to.

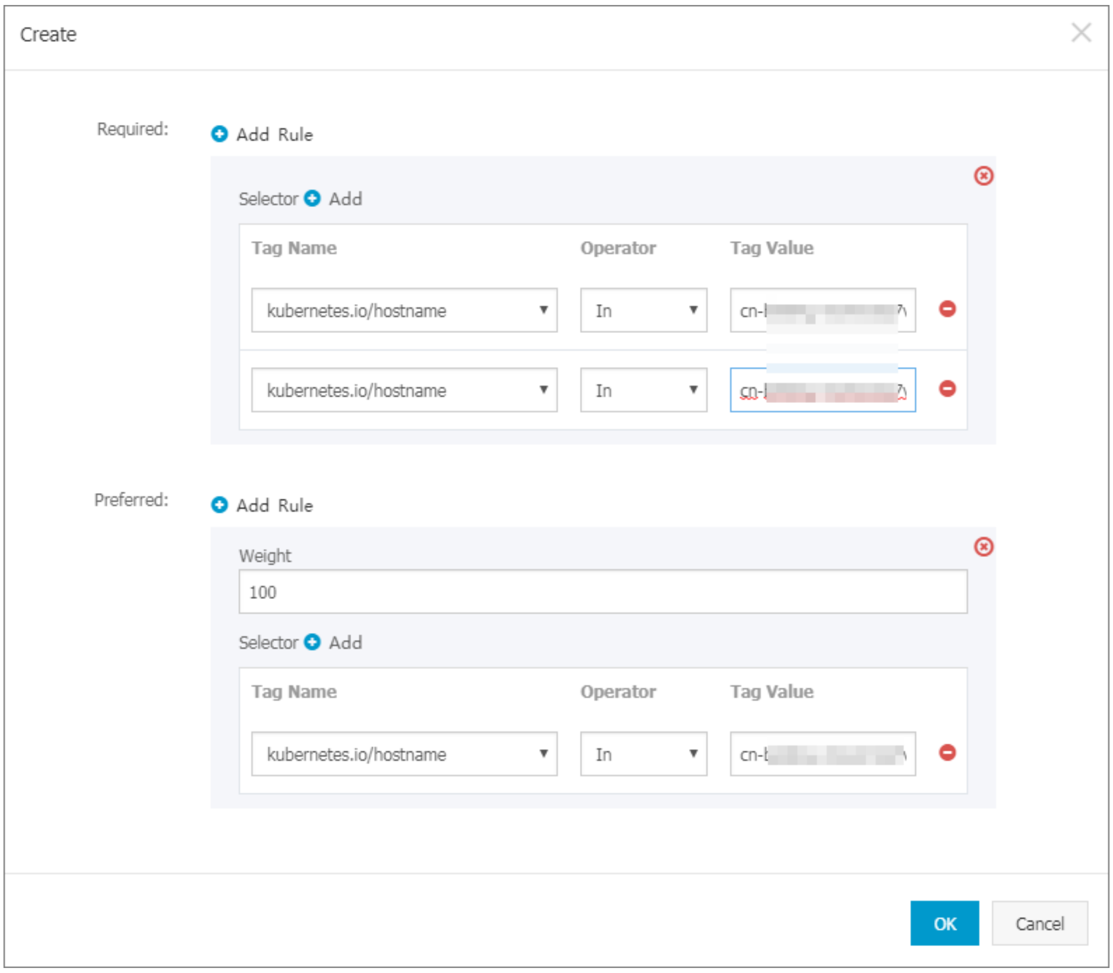
- **Minimum Replicas**: The minimum number of replicas that the deployment can contract to.

c) Optional: Configure **Scheduling Affinity**.

You can configure 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:
Affinity scheduling depends on node tags and pod tags. You can use built-in tads to schedule as well as configure tags for nodes and pods in advance.

1. Set **Node Affinity** by configuring node tags.



Required:

Tag Name	Operator	Tag Value
kubernetes.io/hostname	In	cn-[redacted]
kubernetes.io/hostname	In	cn-[redacted]

Preferred:

Weight
100

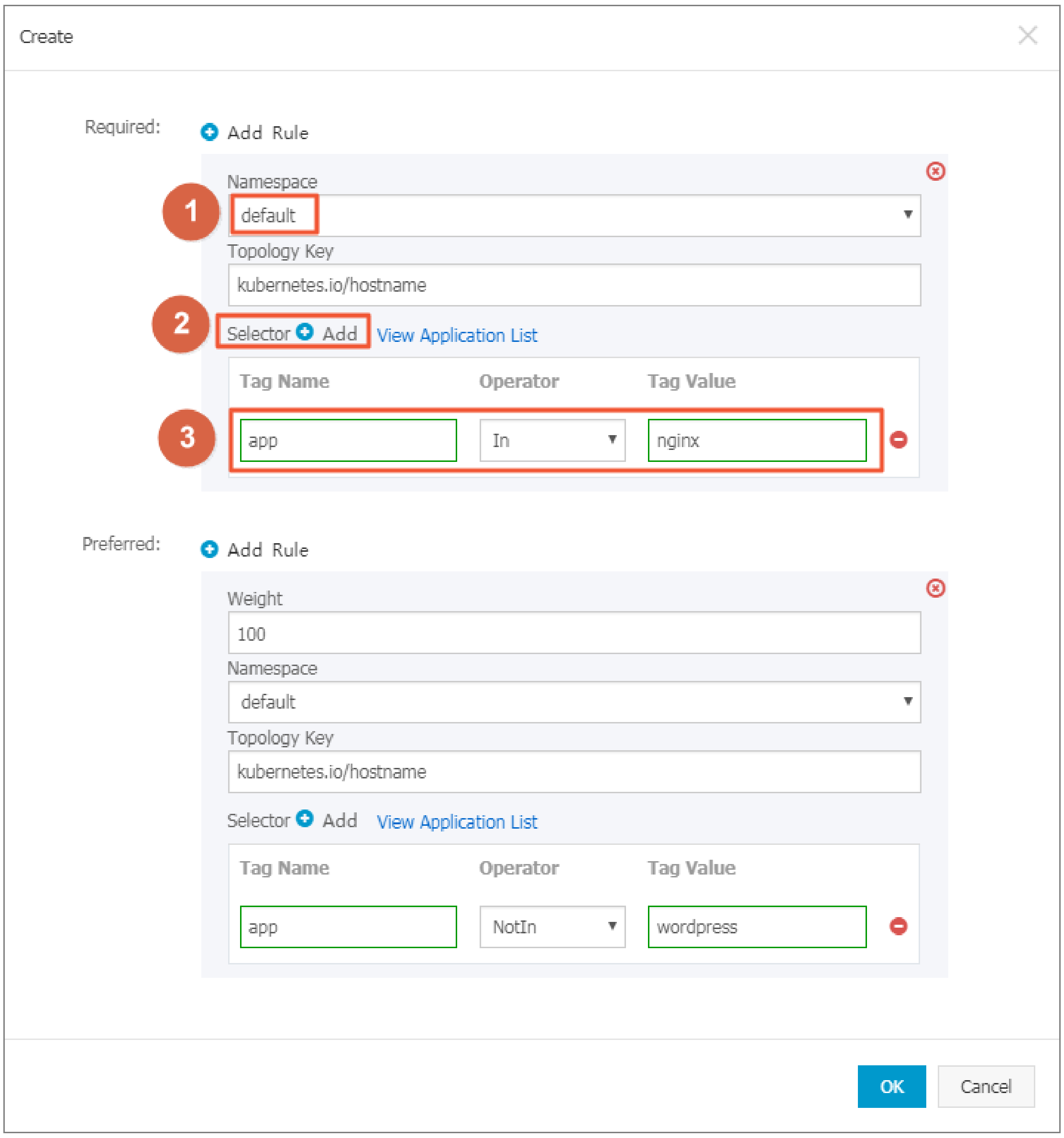
Tag Name	Operator	Tag Value
kubernetes.io/hostname	In	cn-[redacted]

Node scheduling supports both required and preferred rules, and various operators such as In, NotIn, Exists, DoesNotExist, GT, and LT.

- **Required** rules must be satisfied and correspond to **`requiredDuringSchedulingIgnoredDuringExecution`**. The required rules have the same effect as `NodeSelector`. In this example, the pod can be scheduled to only nodes with corresponding tags. You can add multiple required rules, but you only need to meet one of them.

- **Preferred** rules are not necessary satisfied and correspond to **`preferredDuringSchedulingIgnoredDuringExecution`**. In this example, the schedule tries not to schedule the pod to the node with the corresponding tag. You can also set weights for preferred rules. If multiple nodes that match the criteria exist, the node with the highest weight is scheduled as a priority. You can define multiple preferred rules, but all rules must be satisfied before scheduling.

2. Configure **Pod Affinity** to deploy the pod of the application in a topology domain together with other pods. For example, services that communicate with each other can be deployed to the same topology domain (such as a host) by configuring pod affinity scheduling to reduce network latency between them.



Required:

Namespace: default

Topology Key: kubernetes.io/hostname

Tag Name	Operator	Tag Value
app	In	nginx

Preferred:

Weight: 100

Namespace: default

Topology Key: kubernetes.io/hostname

Tag Name	Operator	Tag Value
app	NotIn	wordpress

Schedule pods according to tags of pods running on nodes. Available expressions are `In, NotIn, Exists, DoesNotExist`.

- **Required** rules must be satisfied and correspond to **`requiredDuringSchedulingIgnoredDuringExecution`**. The pod affinity scheduling must meet configured rules.
  - **Namespace**: The scheduling policy is based on pod tags so it is constrained by namespaces.
  - **Topology Key**: Specifies the domain to be scheduled through tags of nodes. For example, if you set `kubernetes.io/hostname` as the topology key, nodes are used to identify topologies. If you specify`beta.kubernetes.io/os` as the topology key, operating systems of nodes are used to identify topologies.
  - **Selector**: By clicking the Add button at the right of Selector, you can add hard constraint rules.
  - **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 this affinity configuration dialog box.
  - Hard constraints: Configure tags of existing applications, operators, and tag values. In this example, schedule the application to be created to this host that runs applications with the `app: nginx` tag.
- **Preferred** rules, that is, soft constraints, corresponding to **`preferredDuringSchedulingIgnoredDuringExecution`**. The pod affinity scheduling meet configured rules as soon as possible. For soft constraint rules, you can configure the weight of each rule. Other configuration requirements are the same as hard constraint rules.

  Note:

  Weight: Specifies the weight of one soft constraint rule in the range of 1 to 100. Weights of nodes that satisfies configured soft constraint rules are calculated through algorithm and then the pod is scheduled to the node with the greatest weight.

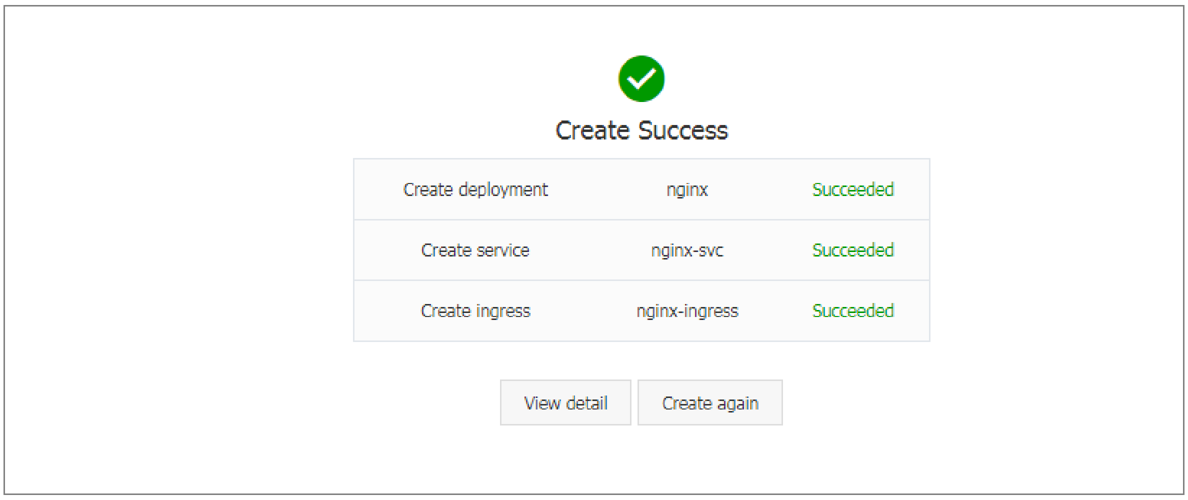
3. Configure **Pod Anti Affinity** to deploy the pod of the application in a topology domain excluding other pods. Scenarios that use pod anti affinity scheduling include:

   - Distribute pods of a service to different topology domains (such as hosts) to improve the stability of the service.

- Grant a pot the exclusive access to a node so as to guarantee no other pods use resources of the node.
- Distribute pods of services that may affect each other to different hosts.

Note:
Configuration methods of pod anti affinity scheduling are the same as that of pod affinity. But the same scheduling rules have different meanings for pod anti affinity scheduling. Select an appropriate scheduling rule based on scenarios.

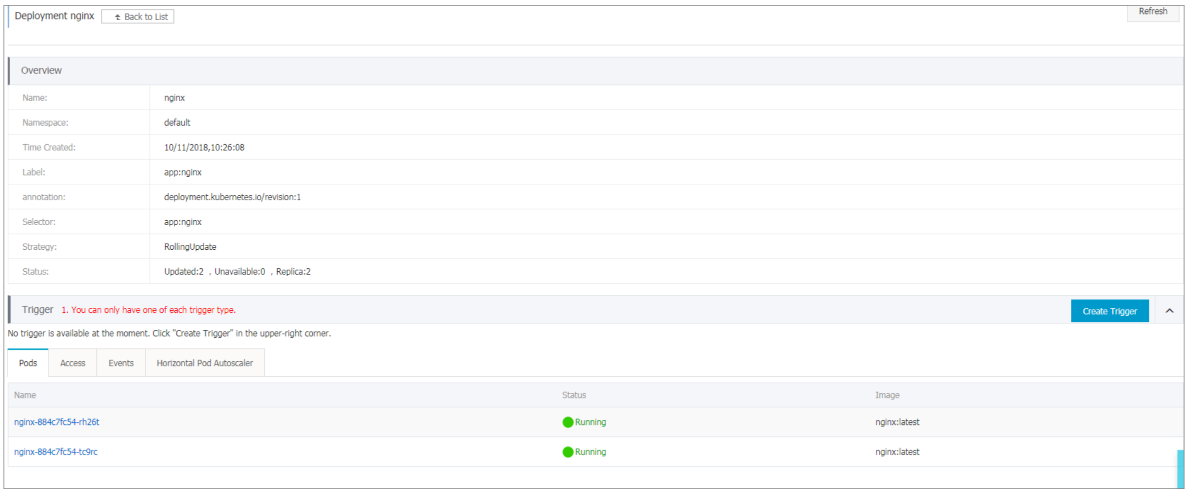
7. Click **Create**.
8. After you create the application, the create success page is displayed and objects contained in the application are listed by default. You can click **View detail** to view the deployment details.



Create Success

Create deployment	nginx	Succeeded
Create service	nginx-svc	Succeeded
Create ingress	nginx-ingress	Succeeded

The nginx-deployment page is displayed by default.



Deployment nginx

Overview

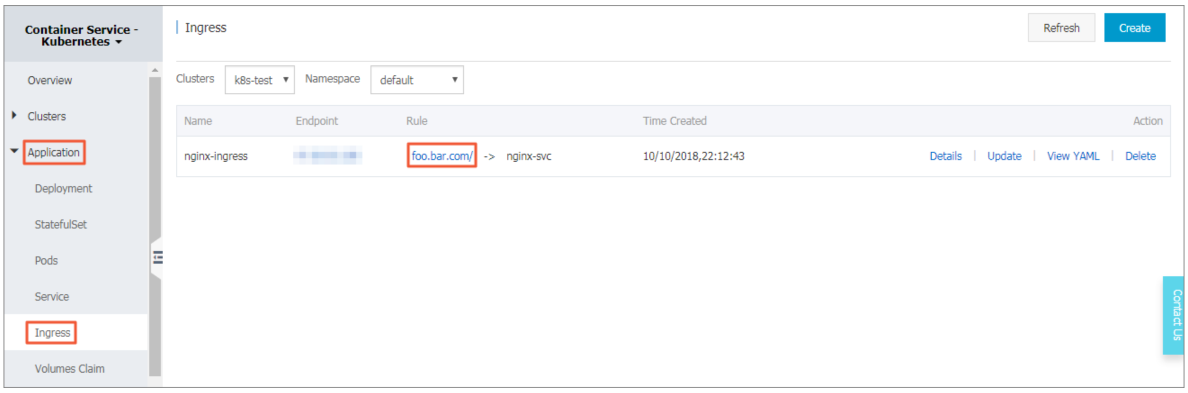
Name:	nginx
Namespace:	default
Time Created:	10/11/2018,10:26:08
Label:	app:nginx
annotation:	deployment.kubernetes.io/revision:1
Selector:	app:nginx
Strategy:	RollingUpdate
Status:	Updated:2 , Unavailable:0 , Replica:2

Trigger 1. You can only have one of each trigger type.

No trigger is available at the moment. Click "Create Trigger" in the upper-right corner.

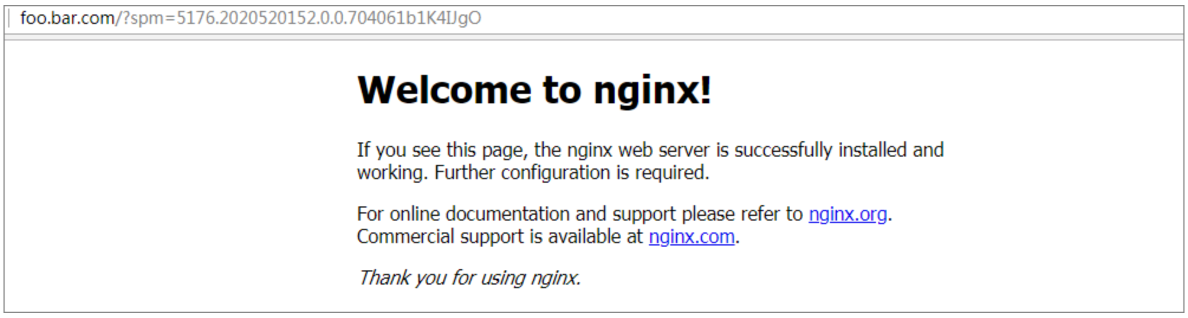
Name	Status	Image
nginx-884c7fc54-rh26t	● Running	nginx:latest
nginx-884c7fc54-tc9rc	● Running	nginx:latest

9. Click **Application** > **Ingress** in the left-side navigation pane, a rule is displayed under the Ingress list.



Name	Endpoint	Rule	Time Created	Action
nginx-ingress		foo.bar.com/ -> nginx-svc	10/10/2018,22:12:43	Details | Update | View YAML | Delete

10.Access the Ingress testing domain in a browser and you can see that the Nginx welcome page is displayed.


foo.bar.com/?spm=5176.2020520152.0.0.704061b1K4IJgO

Welcome to nginx!

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

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

Thank you for using nginx.

2.3 Use Yaml to create a statefull tomcat application

Prerequisites

- Create a Kubernetes cluster. For more information, see *Create a Kubernetes cluster quickly*.
- 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.

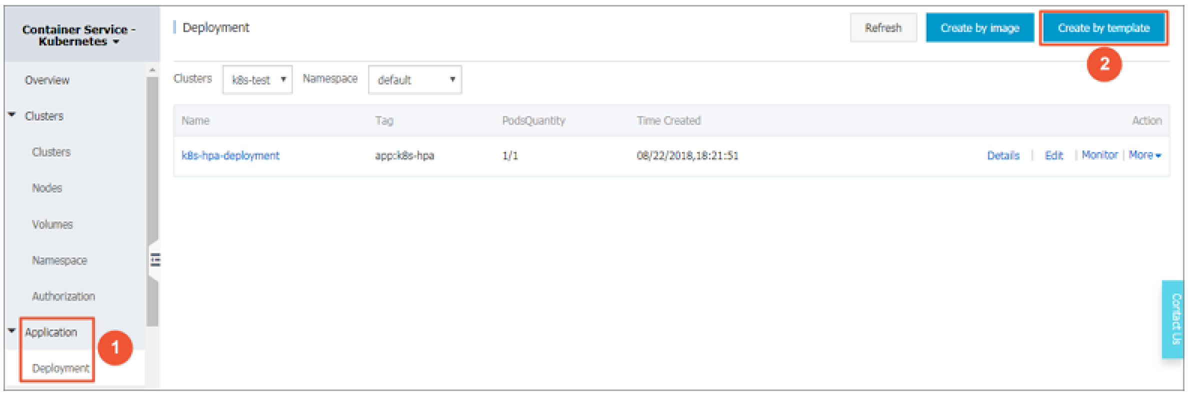
This example shows how to create a tomcat application by customizing a template in an orchestrat ion template. The resource objects involved are as follows:

1. Storage volumes
2. Config maps
3. Secrets
4. Nodes specified by labels

5. Health check
6. Server/Load Balancer

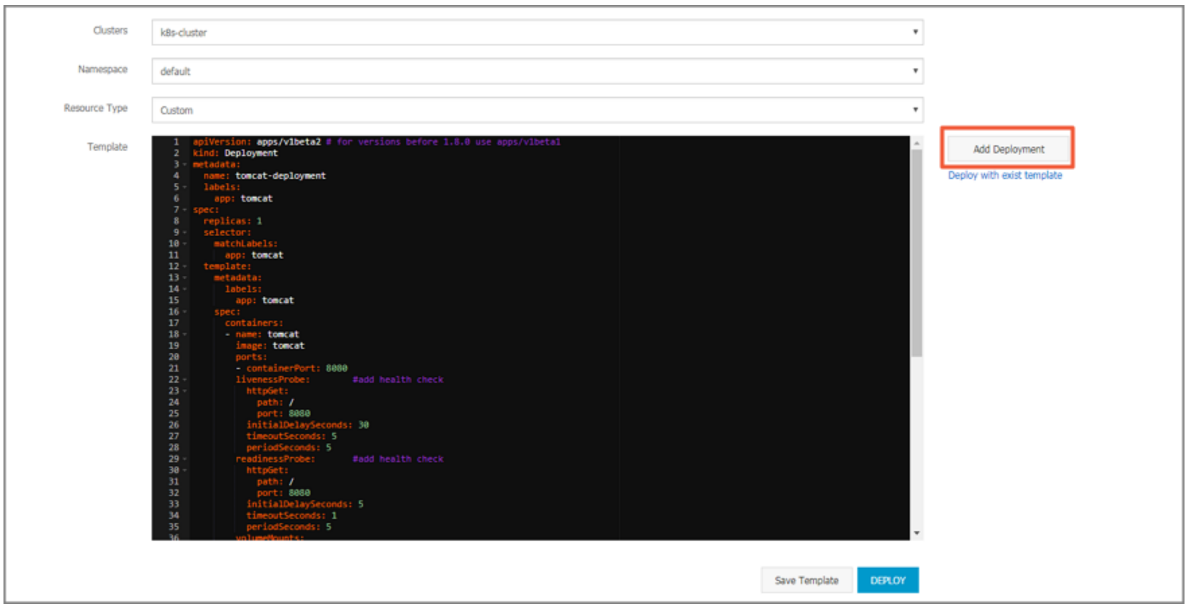
Procedure

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.



Name	Tag	PodsQuantity	Time Created	Action
k8s-hpa-deployment	app:k8s-hpa	1/1	08/22/2018,18:21:51	Details | Edit | Monitor | More

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.



Clusters: k8s-cluster

Namespace: default

Resource Type: Custom

Template:

```
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:
    matchLabels:
      app: tomcat
  template:
    metadata:
      labels:
        app: tomcat
    spec:
      containers:
      - name: tomcat
        image: tomcat
        ports:
        - containerPort: 8080
        livenessProbe:        #add health check
          httpGet:
            path: /
            port: 8080
          initialDelaySeconds: 30
          timeoutSeconds: 5
          periodSeconds: 5
        readinessProbe:       #add health check
          httpGet:
            path: /
            port: 8080
          initialDelaySeconds: 5
          timeoutSeconds: 1
          periodSeconds: 5
        volumeMounts:
```

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:
    matchLabels:
      app: tomcat
  template:
    metadata:
      labels:
        app: tomcat
    spec:
      containers:
      - name: tomcat
        image: tomcat # replace it with your exactly &lt;
image_name:tags>
        ports:
        - containerPort: 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:
    matchLabels:
      app: tomcat
  template:
    metadata:
      labels:
        app: tomcat
    spec:
      containers:
      - name: tomcat
        image: tomcat # replace it with your exactly &lt;
image_name:tags>
        ports:
        - containerPort: 8080
        volumeMounts:                 #add volume
        - name: pvc-yunpan-test
          mountPath: /data
      volumes:                        #add volume
      - name: pvc-yunpan-test
        persistentVolumeClaim:
          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_LEVEL:very` and`SPECIAL_TYPE: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:
    matchLabels:
      app: tomcat
  template:
    metadata:
      labels:
        app: tomcat
    spec:
      containers:
      - name: tomcat
        image: tomcat # replace it with your exactly &lt;
image_name:tags>
        ports:
        - containerPort: 8080
        volumeMounts:
        - name: pvc-yunpan-test
          mountPath: /data
        env:
        - name: SPECIAL_LEVEL_KEY #add configmap
          valueFrom:
            configMapKeyRef:
              name: special-config
              key: SPECIAL_LEVEL
        - name: SPECIAL_TYPE_KEY #add configmap
          valueFrom:
            configMapKeyRef:
              name: special-config
              key: SPECIAL_TYPE
      volumes:
      - name: pvc-yunpan-test
        persistentVolumeClaim:
          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:
    matchLabels:
      app: tomcat
  template:
    metadata:
      labels:
        app: tomcat
    spec:
      containers:
      - name: tomcat
        image: tomcat # replace it with your exactly &lt;
image_name:tags>
        ports:
        - containerPort: 8080
        volumeMounts:
        - name: pvc-yunpan-test
          mountPath: /data
        env:
        - name: SPECIAL_LEVEL_KEY
          valueFrom:
            configMapKeyRef:
              name: special-config
              key: SPECIAL_LEVEL
        - name: SPECIAL_TYPE_KEY
          valueFrom:
            configMapKeyRef:
              name: special-config
              key: SPECIAL_TYPE
        - name: SECRET_USERNAME #add secret
          valueFrom:
            secretKeyRef:
              name: account
              key: username
        - name: SECRET_PASSWORD #add secret
          valueFrom:
            secretKeyRef:
              name: account
              key: password
      volumes:
      - name: pvc-yunpan-test
        persistentVolumeClaim:
          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:
    matchLabels:
      app: tomcat
  template:
    metadata:
      labels:
        app: tomcat
    spec:
      containers:
      - name: tomcat
        image: tomcat
        ports:
        - containerPort: 8080
        volumeMounts:
        - name: pvc-yunpan-test
          mountPath: /data
        env:
        - name: SPECIAL_LEVEL_KEY
          valueFrom:
            configMapKeyRef:
              name: special-config
              key: SPECIAL_LEVEL
        - name: SPECIAL_TYPE_KEY
          valueFrom:
            configMapKeyRef:
              name: special-config
              key: SPECIAL_TYPE
        - name: SECRET_USERNAME
          valueFrom:
            secretKeyRef:
              name: account
              key: username
        - name: SECRET_PASSWORD
          valueFrom:
            secretKeyRef:
              name: account
              key: password
      volumes:
      - name: pvc-yunpan-test
        persistentVolumeClaim:
          claimName: pvc-yunpan-test
      nodeSelector:       #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:
    matchLabels:
      app: tomcat
  template:
    metadata:
      labels:
        app: tomcat
    spec:
      containers:
      - name: tomcat
        image: tomcat
        ports:
        - containerPort: 8080
        livenessProbe:        #add health check
          httpGet:
            path: /
            port: 8080
          initialDelaySeconds: 30
          timeoutSeconds: 5
          periodSeconds: 5
        readinessProbe:       #add health check
```

```
            httpGet:
              path: /
              - port: 8080
            initialDelaySeconds: 5
            timeoutSeconds: 1
            periodSeconds: 5
          volumeMounts:
          - name: pvc-yunpan-test
            mountPath: /data
          env:
          - name: SPECIAL_LEVEL_KEY
            valueFrom:
              configMapKeyRef:
                name: special-config
                key: SPECIAL_LEVEL
          - name: SPECIAL_TYPE_KEY
            valueFrom:
              configMapKeyRef:
                name: special-config
                key: SPECIAL_TYPE
          - name: SECRET_USERNAME
            valueFrom:
              secretKeyRef:
                name: account
                key: username
          - name: SECRET_PASSWORD
            valueFrom:
              secretKeyRef:
                name: account
                key: password
        volumes:
        - name: pvc-yunpan-test
          persistentVolumeClaim:
            claimName: pvc-yunpan-test
        nodeSelector:
          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.

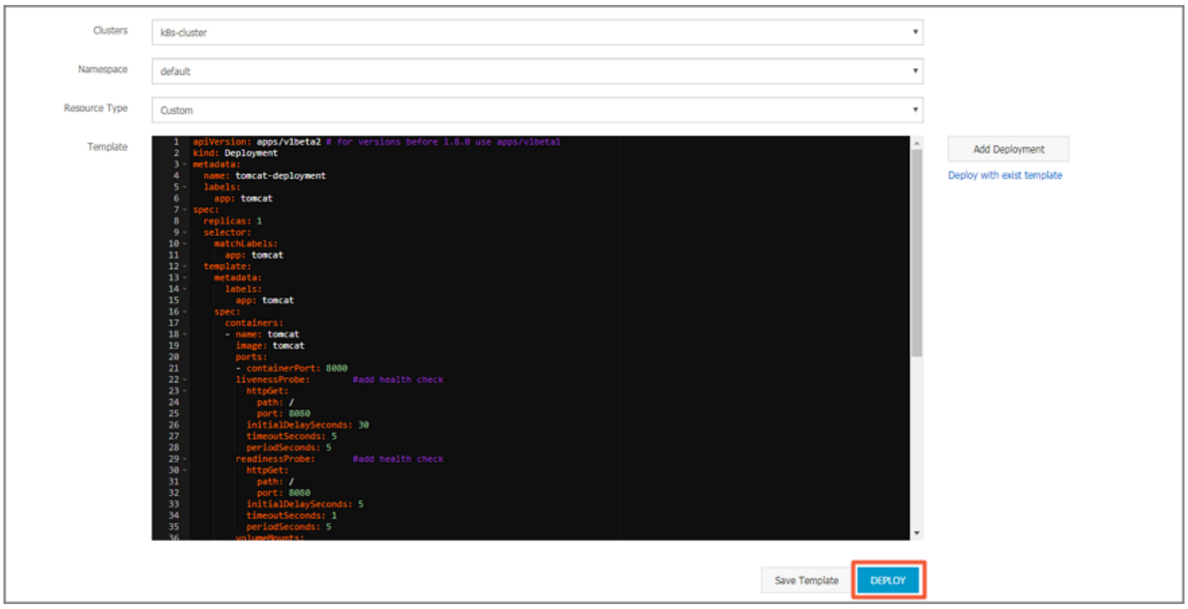
For information about creating a service, see *Create a service*.

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:
```

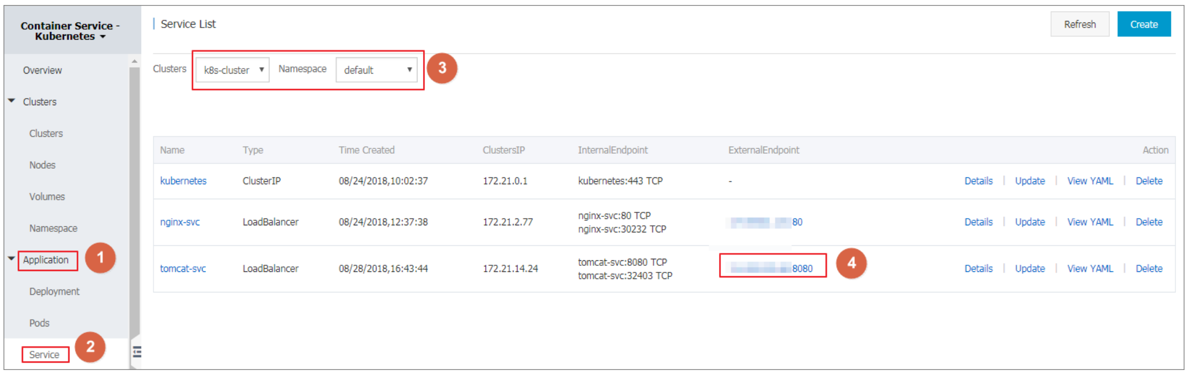
```
  - protocol: TCP
    port: 8080
    targetPort: 8080
  type: LoadBalancer
```

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



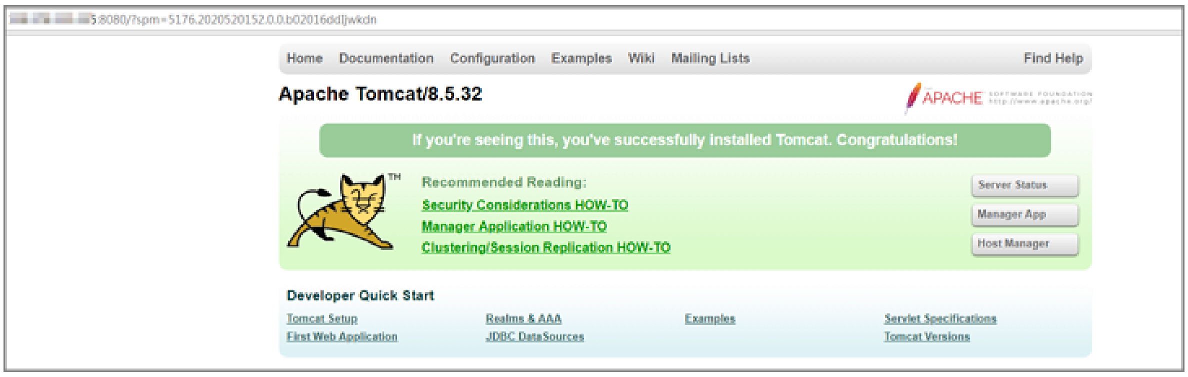
```
apiVersion: apps/v1beta2 # for versions before 1.8.0 use apps/v1beta1
kind: Deployment
metadata:
  name: tomcat-deployment
  labels:
    app: tomcat
spec:
  replicas: 1
  selector:
    matchLabels:
      app: tomcat
  template:
    metadata:
      labels:
        app: tomcat
    spec:
      containers:
      - name: tomcat
        image: tomcat
        ports:
        - containerPort: 8080
        livenessProbe:        #add health check
          httpGet:
            path: /
            port: 8080
          initialDelaySeconds: 30
          timeoutSeconds: 5
          periodSeconds: 5
        readinessProbe:        #add health check
          httpGet:
            path: /
            port: 8080
          initialDelaySeconds: 5
          timeoutSeconds: 1
          periodSeconds: 5
        volumeMounts:
```

6. After the deployment succeeds, click **Service** in the left navigation pane, and select the tomcat-svc service to view its external endpoint.



Name	Type	Time Created	ClustersIP	InternalEndpoint	ExternalEndpoint	Action
kubernetes	ClusterIP	08/24/2018,10:02:37	172.21.0.1	kubernetes:443 TCP	-	Details | Update | View YAML | Delete
nginx-svc	LoadBalancer	08/24/2018,12:37:38	172.21.2.77	nginx-svc:80 TCP nginx-svc:30232 TCP	[redacted]80	Details | Update | View YAML | Delete
tomcat-svc	LoadBalancer	08/28/2018,16:43:44	172.21.14.24	tomcat-svc:8080 TCP tomcat-svc:32403 TCP	[redacted]8080	Details | Update | View YAML | Delete

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



[redacted]5:8080/?spm=5176.2020520152.0.0.b02016ddljwkdn

Home Documentation Configuration Examples Wiki Mailing Lists Find Help

Apache Tomcat/8.5.32

If you're seeing this, you've successfully installed Tomcat. Congratulations!

Recommended Reading:
Security Considerations HOW-TO
Manager Application HOW-TO
Clustering/Session Replication HOW-TO

Server Status
Manager App
Host Manager

Developer Quick Start
Tomcat Setup
First Web Application
Realms & AAA
JDBC DataSources
Examples
Servlet Specifications
Tomcat Versions

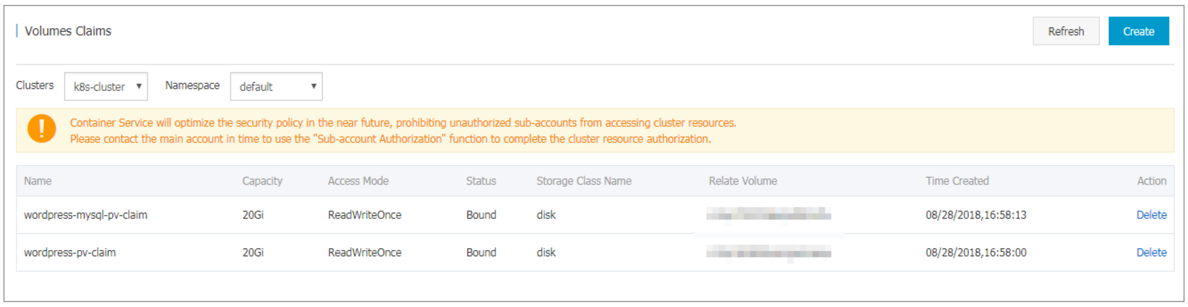
What's next

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

- Create a Kubernetes cluster. For more information, see *Create a Kubernetes cluster quickly*.
- Create storage volumes and storage volume claims. For how to create a storage volume, see *Use Alibaba Cloud cloud disks*, *Use Alibaba Cloud NAS*, and *Use Alibaba Cloud OSS*. For how to create a storage volume claim, see *Create a persistent storage 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.



Volumes Claims

Refresh Create

Clusters k8s-cluster Namespace default

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

Name	Capacity	Access Mode	Status	Storage Class Name	Relate Volume	Time Created	Action
wordpress-mysql-pv-claim	20Gi	ReadWriteOnce	Bound	disk	[redacted]	08/28/2018,16:58:13	Delete
wordpress-pv-claim	20Gi	ReadWriteOnce	Bound	disk	[redacted]	08/28/2018,16:58:00	Delete

Context

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

The main components are:

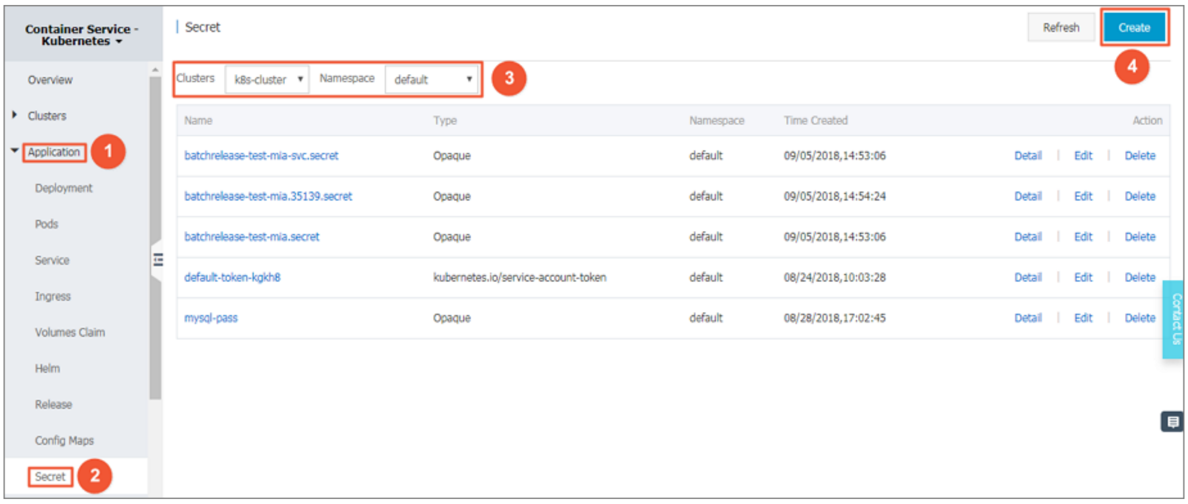
- wordpress
- mysql

Resources involved:

- Storage volume
- Secret
- Service

Procedure

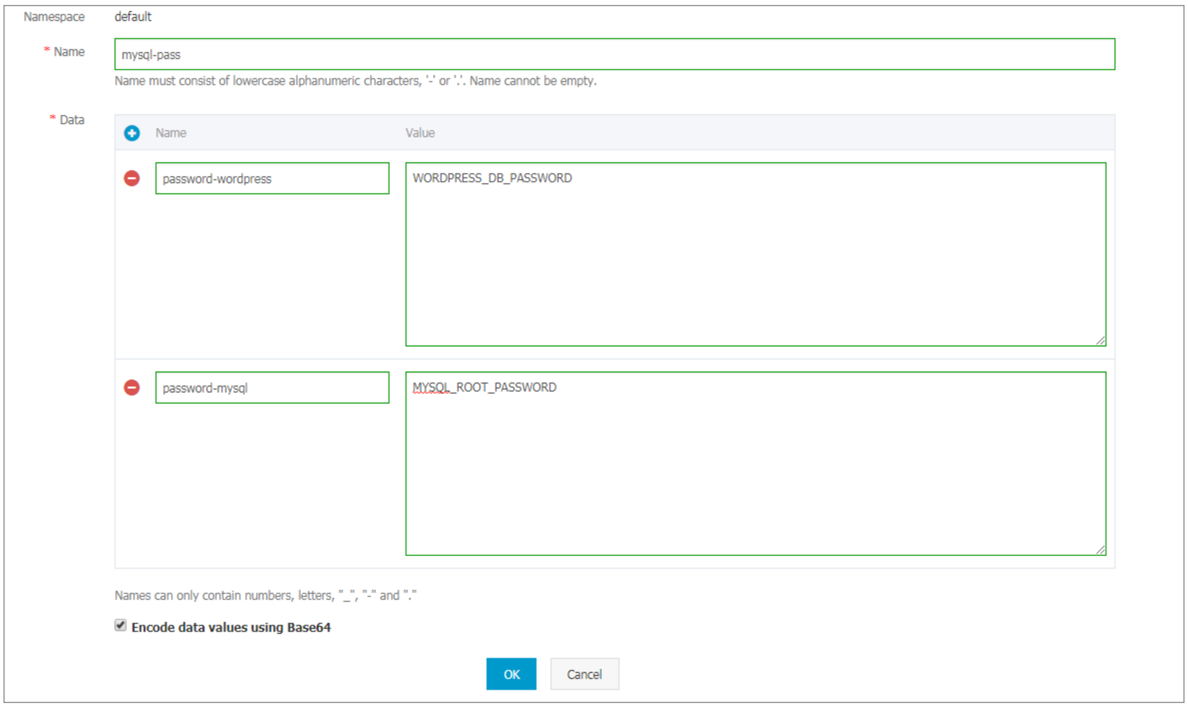
1. Log on to the *Container Service console*.
2. Use the prepared storage volume claims. Create two storage volume claims: wordpress-pv-claim and wordpress-mysql-pv-claim which are used in the wordpress yaml file and the wordpress-mysql yaml file respectively, to mount corresponding storage volumes.
3. Click **Application** > **Secret** in the left-side navigation pane, select a cluster and namespace, and click **Create** in the upper-right corner. For the creation process, see *Create a secret*.



Name	Type	Namespace	Time Created	Action
batchrelease-test-mia-svc.secret	Opaque	default	09/05/2018,14:53:06	Detail | Edit | Delete
batchrelease-test-mia.35139.secret	Opaque	default	09/05/2018,14:54:24	Detail | Edit | Delete
batchrelease-test-mia.secret	Opaque	default	09/05/2018,14:53:06	Detail | Edit | Delete
default-token-kgkh8	kubernetes.io/service-account-token	default	08/24/2018,10:03:28	Detail | Edit | Delete
mysql-pass	Opaque	default	08/28/2018,17:02:45	Detail | Edit | Delete

Since a user name and password is required to create and access the MySQL database, create a secret to manage the user name and password.

Before using a secret, create a secret that needs to be encrypted. In this example, the MySQL root password is created as the secret and the secret name is mysql-pass. This secret is used in the WordPress yaml file and wordpress-mysql yaml file.



Namespace default

* Name mysql-pass

Name must consist of lowercase alphanumeric characters, '-' or '.'. Name cannot be empty.

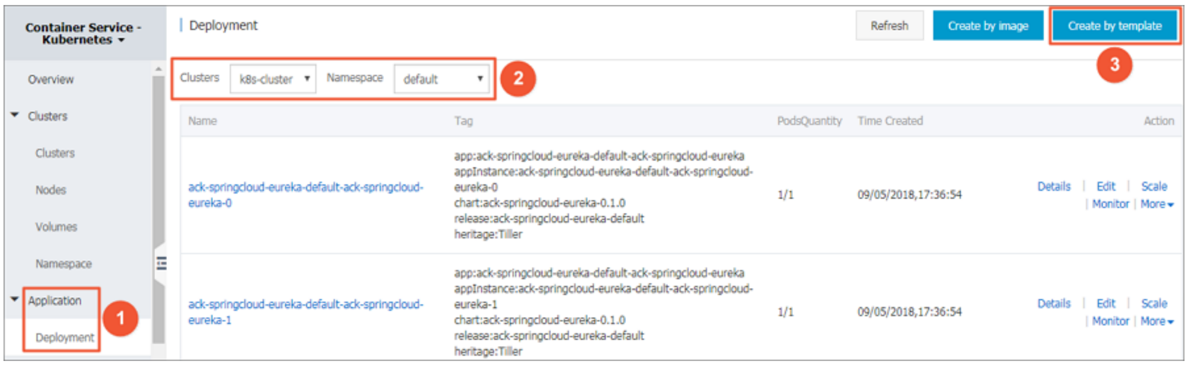
* Data

Name	Value
password-wordpress	WORDPRESS_DB_PASSWORD
password-mysql	MYSQL_ROOT_PASSWORD

Names can only contain numbers, letters, "_", "-" and "."

☑ Encode data values using Base64

4. Click **Application** > **Deployment** in the left-side navigation pane, and click **Create by template** in the upper-right corner.



Clusters k8s-cluster Namespace default

Name	Tag	PodsQuantity	Time Created	Action
ack-springcloud-eureka-default-ack-springcloud-eureka-0	app:ack-springcloud-eureka-default-ack-springcloud-eureka appInstance:ack-springcloud-eureka-default-ack-springcloud-eureka-0 chart:ack-springcloud-eureka-0.1.0 release:ack-springcloud-eureka-default heritage:Tiller	1/1	09/05/2018,17:36:54	Details | Edit | Scale | Monitor | More
ack-springcloud-eureka-default-ack-springcloud-eureka-1	app:ack-springcloud-eureka-default-ack-springcloud-eureka appInstance:ack-springcloud-eureka-default-ack-springcloud-eureka-1 chart:ack-springcloud-eureka-0.1.0 release:ack-springcloud-eureka-default heritage:Tiller	1/1	09/05/2018,17:36:54	Details | Edit | Scale | Monitor | More

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:
    matchLabels:
      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 corresponds to the mysql service name
.
        - name: WORDPRESS_DB_PASSWORD
          valueFrom:
            secretKeyRef:
              name: mysql-pass
              key: password-wordpress
        ports:
        - containerPort: 80
          name: wordpress
        volumeMounts:
        - name: wordpress-pvc
          mountPath: /var/www/html
      volumes:
      - name: wordpress-pvc
        persistentVolumeClaim:
          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:
    matchLabels:
      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:
            secretKeyRef:
              name: mysql-pass
              key: password-mysql
        ports:
        - containerPort: 3306
          name: mysql
        volumeMounts:
        - name: wordpress-mysql-pvc
          mountPath: /var/lib/mysql
      volumes:
      - name: wordpress-mysql-pvc
        persistentVolumeClaim:
          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.

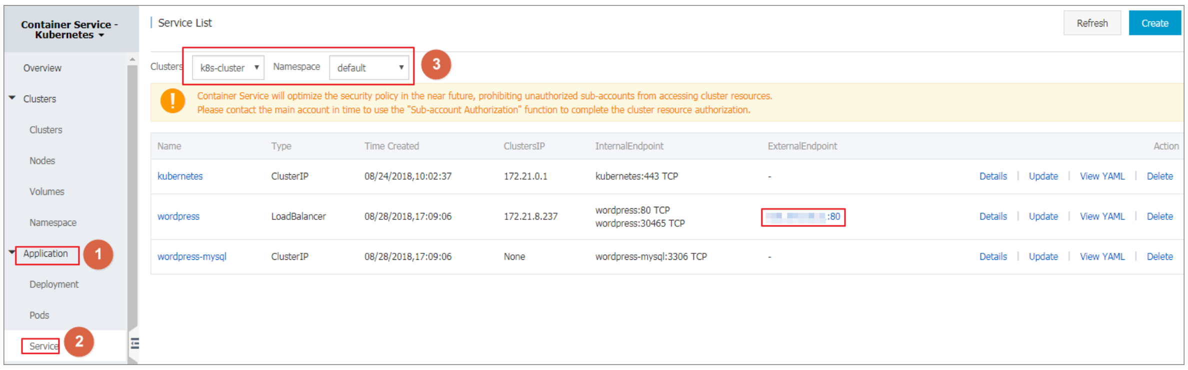
   Create a service named WordPress-mysql for the WordPress mysql so that the WordPress deploymet created on the WordPress mysql can be accessed. As the mysql is called only internally for the WordPress, you do not need to create a LoadBalancer type of service for it.

   For how to create a service, 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: LoadBalancer
---
apiVersion: v1
kind: Service
metadata:
  name: wordpress-mysql
  labels:
    app: wordpress
spec:
  ports:
    - port: 3306
  selector:
    app: wordpress
    tier: mysql
     clusterIP: None
```

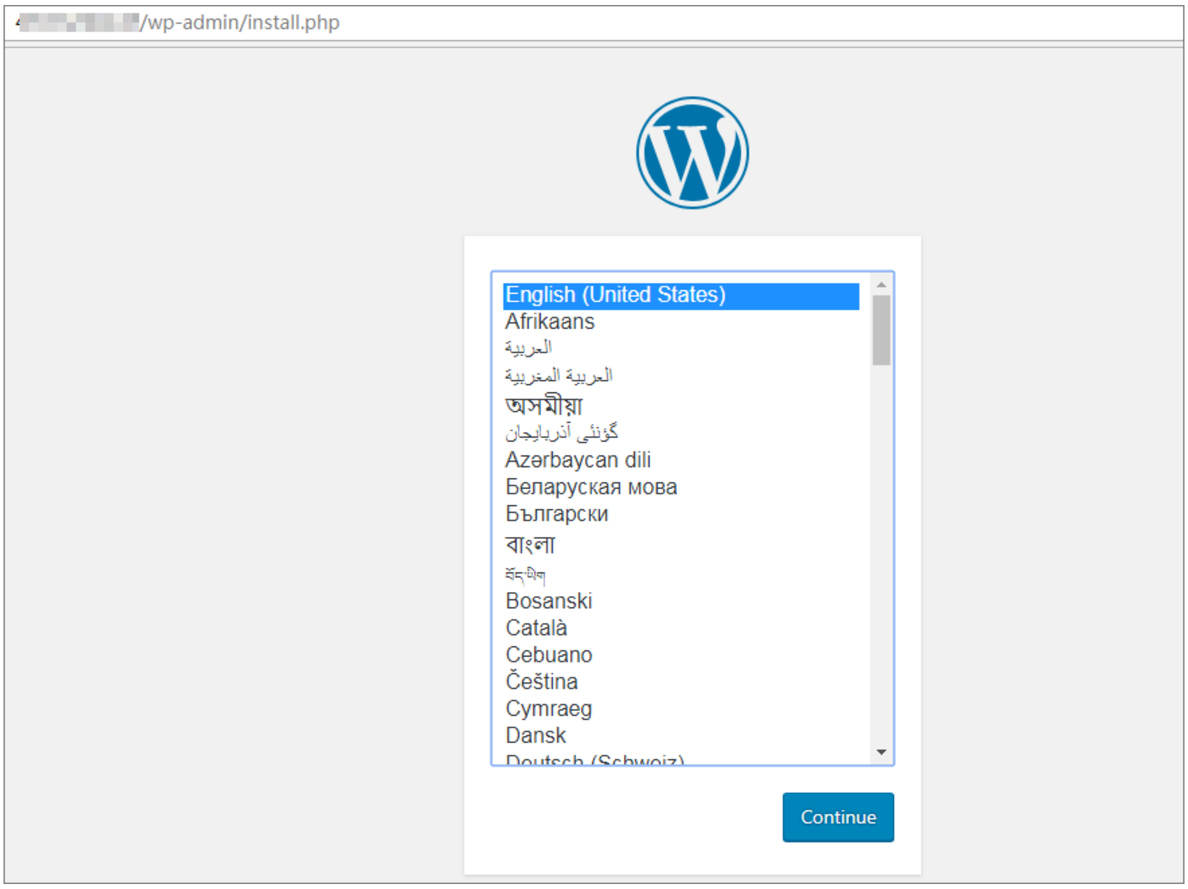
6. When the deployment is completed, click **Application** > **Service** in the left-side navigation pane. Locate the WordPress service and view its external endpoint.



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

Name	Type	Time Created	ClustersIP	InternalEndpoint	ExternalEndpoint	Action
kubernetes	ClusterIP	08/24/2018,10:02:37	172.21.0.1	kubernetes:443 TCP	-	Details | Update | View YAML | Delete
wordpress	LoadBalancer	08/28/2018,17:09:06	172.21.8.237	wordpress:80 TCP wordpress:30465 TCP	[redacted]:80	Details | Update | View YAML | Delete
wordpress-mysql	ClusterIP	08/28/2018,17:09:06	None	wordpress-mysql:3306 TCP	-	Details | Update | View YAML | Delete

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.

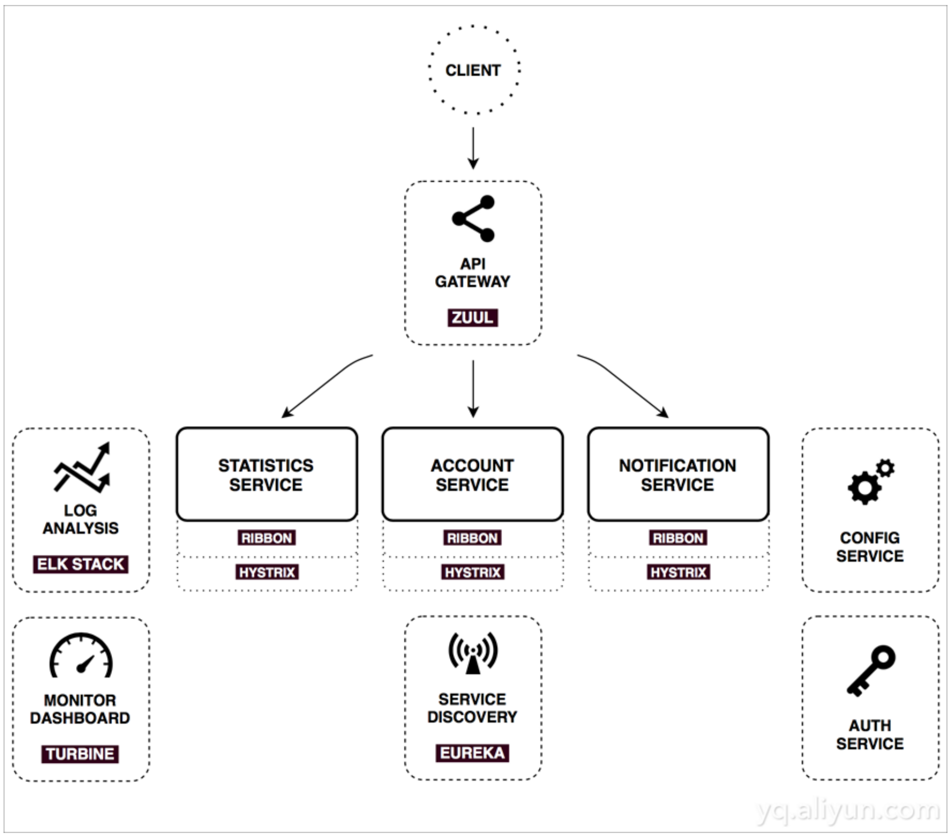
Deployment methods

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 DockerCompose 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. The Yunqi Community has tool named *kompose* that can convert the compose file to the Kubernetes deployment file in 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_healthy  #condition is not supported
```

Add Kubernetes server type annotation：

```
labels:
      kompose.service.type: loadbalancer
```

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.

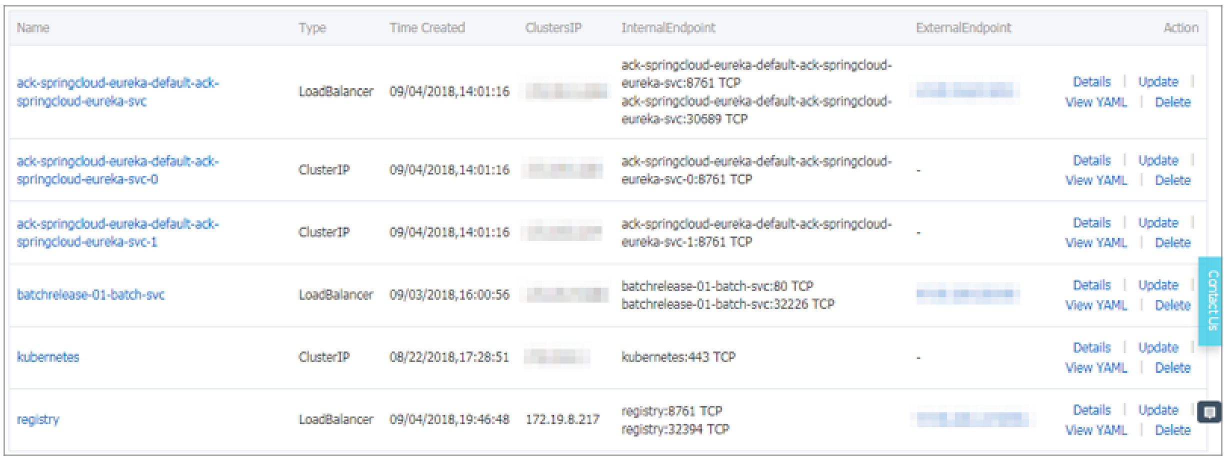
```
$ export NOTIFICATION_SERVICE_PASSWORD=passw0rd
$ export CONFIG_SERVICE_PASSWORD=passw0rd
$ export STATISTICS_SERVICE_PASSWORD=passw0rd
$ export ACCOUNT_SERVICE_PASSWORD=passw0rd
$ export MONGODB_PASSWORD=passw0rd
$ kompose convert -f docker-compose.yml -o piggymetrics -c
```

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

```
charts $ helm install -n piggymetrics piggymetrics/
```

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.

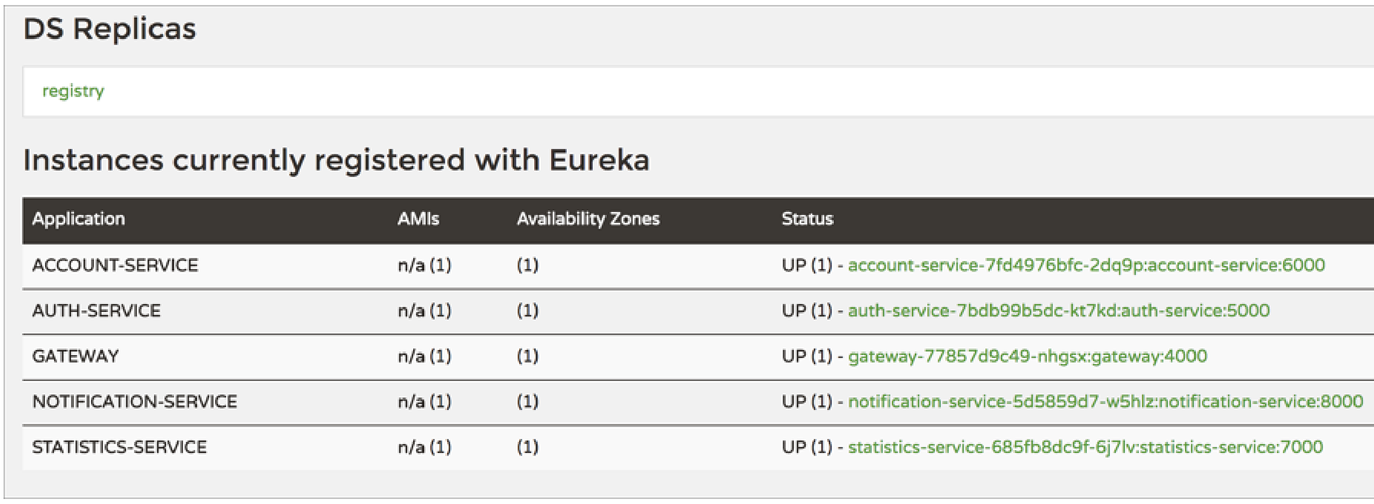


Name	Type	Time Created	ClustersIP	InternalEndpoint	ExternalEndpoint	Action
ack-springcloud-eureka-default-ack-springcloud-eureka-svc	LoadBalancer	09/04/2018,14:01:16	[redacted]	ack-springcloud-eureka-default-ack-springcloud-eureka-svc:8761 TCP ack-springcloud-eureka-default-ack-springcloud-eureka-svc:30689 TCP	[redacted]	Details | Update | View YAML | Delete
ack-springcloud-eureka-default-ack-springcloud-eureka-svc-0	ClusterIP	09/04/2018,14:01:16	[redacted]	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	[redacted]	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	[redacted]	batchrelease-01-batch-svc:80 TCP batchrelease-01-batch-svc:32226 TCP	[redacted]	Details | Update | View YAML | Delete
kubernetes	ClusterIP	08/22/2018,17:28:51	[redacted]	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	[redacted]	Details | Update | View YAML | Delete

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

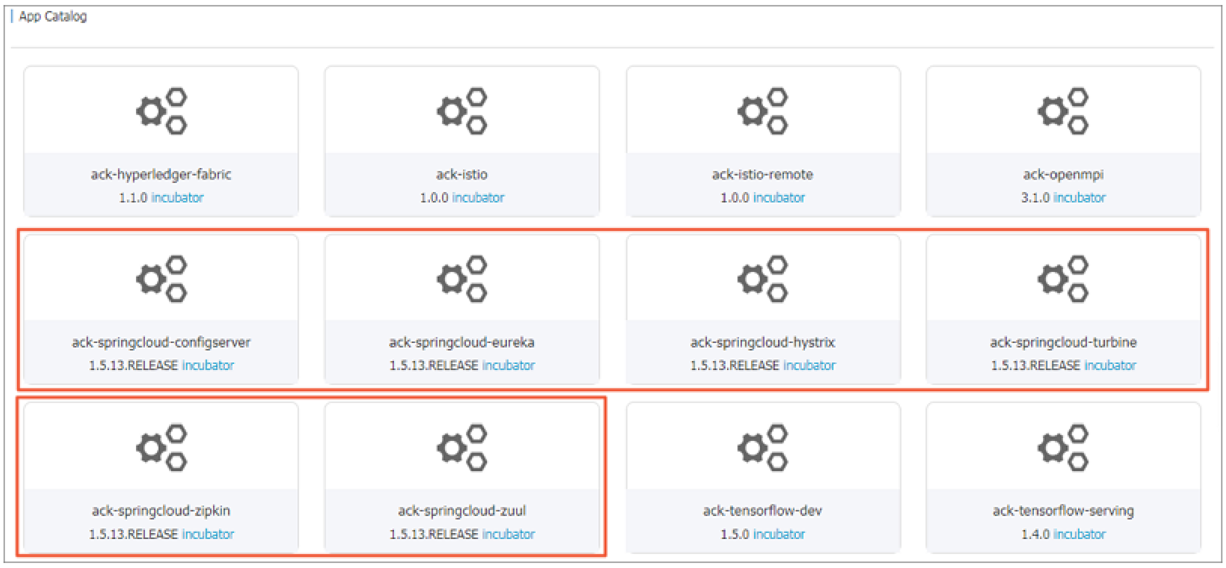
Remove PiggyMetrics to prepare for the next experiment:

```
charts $ helm delete --purge piggymetrics
release "piggymetrics" 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, ConfigServ er, 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.

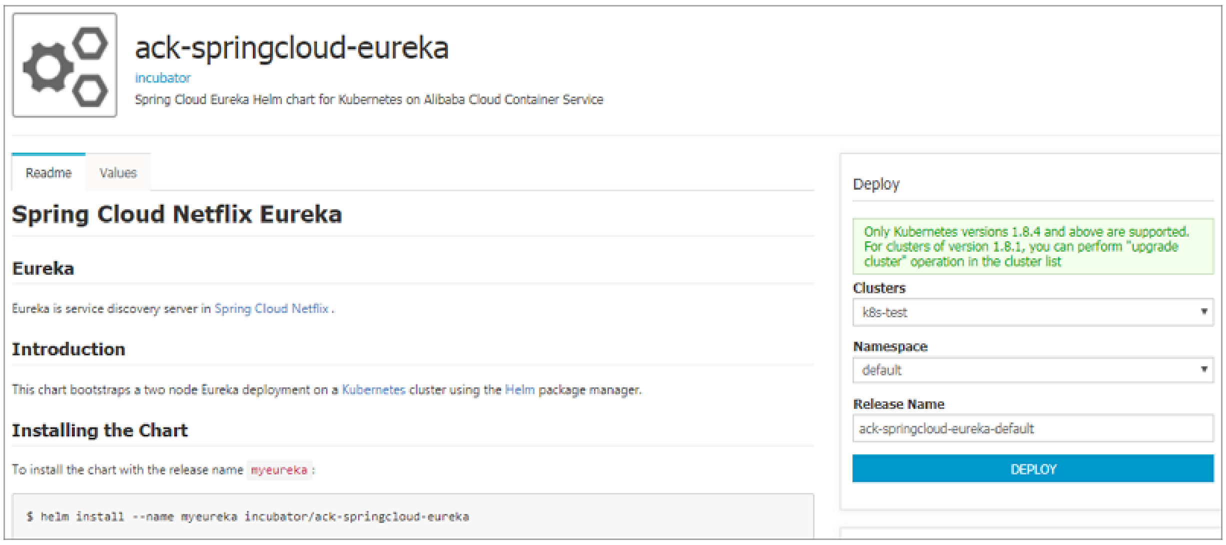
In Alibaba Cloud Container Service for Kubernetes, the App Catalog contains SpringCloud basic components.



App Catalog

ack-hyperledger-fabric 1.1.0 incubator	ack-istio 1.0.0 incubator	ack-istio-remote 1.0.0 incubator	ack-openmpi 3.1.0 incubator
ack-springcloud-configserver 1.5.13.RELEASE incubator	ack-springcloud-eureka 1.5.13.RELEASE incubator	ack-springcloud-hystrix 1.5.13.RELEASE incubator	ack-springcloud-turbine 1.5.13.RELEASE incubator
ack-springcloud-zipkin 1.5.13.RELEASE incubator	ack-springcloud-zuul 1.5.13.RELEASE incubator	ack-tensorflow-dev 1.5.0 incubator	ack-tensorflow-serving 1.4.0 incubator

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



ack-springcloud-eureka

incubator

Spring Cloud Eureka Helm chart for Kubernetes on Alibaba Cloud Container Service

Spring Cloud Netflix Eureka

Eureka

Eureka is service discovery server in Spring Cloud Netflix .

Introduction

This chart bootstraps a two node Eureka deployment on a Kubernetes cluster using the Helm package manager.

Installing the Chart

To install the chart with the release name `myeureka` :

```
$ helm install --name myeureka incubator/ack-springcloud-eureka
```

Deploy

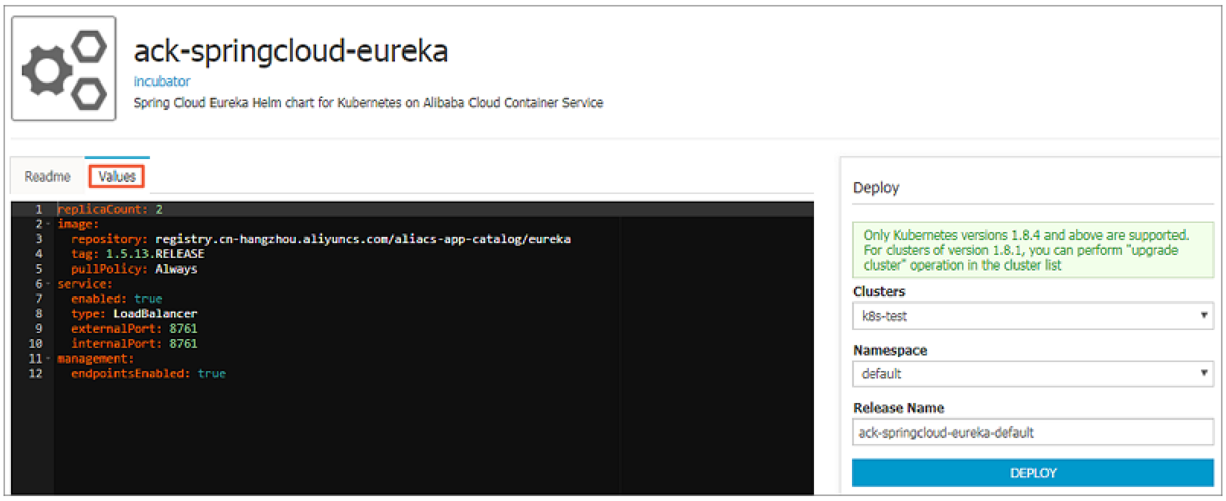
Only Kubernetes versions 1.8.4 and above are supported. For clusters of version 1.8.1, you can perform "upgrade cluster" operation in the cluster list

Clusters: k8s-test

Namespace: default

Release Name: ack-springcloud-eureka-default

To view or change the configuration, click the Values tab:



ack-springcloud-eureka

incubator

Spring Cloud Eureka Helm chart for Kubernetes on Alibaba Cloud Container Service

```
replicaCount: 2
image:
  repository: registry.cn-hangzhou.aliyuncs.com/aliacs-app-catalog/eureka
  tag: 1.5.13.RELEASE
  pullPolicy: Always
service:
  enabled: true
  type: LoadBalancer
  externalPort: 8761
  internalPort: 8761
management:
  endpointsEnabled: true
```

Deploy

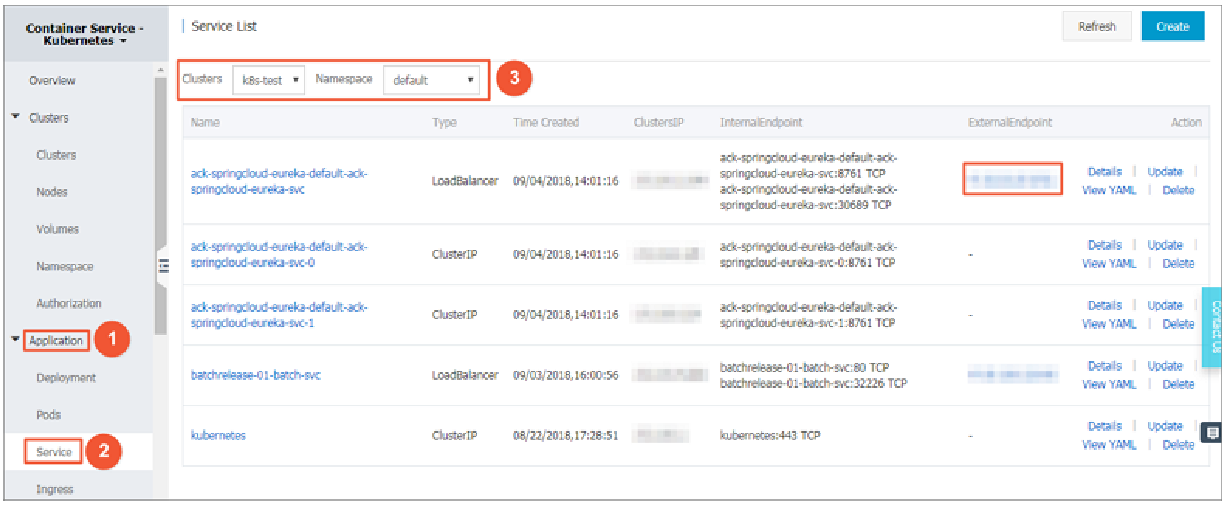
Only Kubernetes versions 1.8.4 and above are supported. For clusters of version 1.8.1, you can perform "upgrade cluster" operation in the cluster list

Clusters: k8s-test

Namespace: default

Release Name: ack-springcloud-eureka-default

Deploy directly without changing any parameters. After deployment, enter the Service List page, you can see that EurekaServer has two examples. The exposed service address is `ack-springcloud-eureka-default-ack-springcloud-eureka-svc`.



Clusters k8s-test Namespace default

Name	Type	Time Created	ClustersIP	InternalEndpoint	ExternalEndpoint	Action
ack-springcloud-eureka-default-ack-springcloud-eureka-svc	LoadBalancer	09/04/2018,14:01:16	[redacted]	ack-springcloud-eureka-default-ack-springcloud-eureka-svc:8761 TCP ack-springcloud-eureka-default-ack-springcloud-eureka-svc:30689 TCP	[redacted]	Details | Update | View YAML | Delete
ack-springcloud-eureka-default-ack-springcloud-eureka-svc-0	ClusterIP	09/04/2018,14:01:16	[redacted]	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	[redacted]	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	[redacted]	batchrelease-01-batch-svc:80 TCP batchrelease-01-batch-svc:32226 TCP	[redacted]	Details | Update | View YAML | Delete
kubernetes	ClusterIP	08/22/2018,17:28:51	[redacted]	kubernetes:443 TCP	-	Details | Update | View YAML | Delete

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:

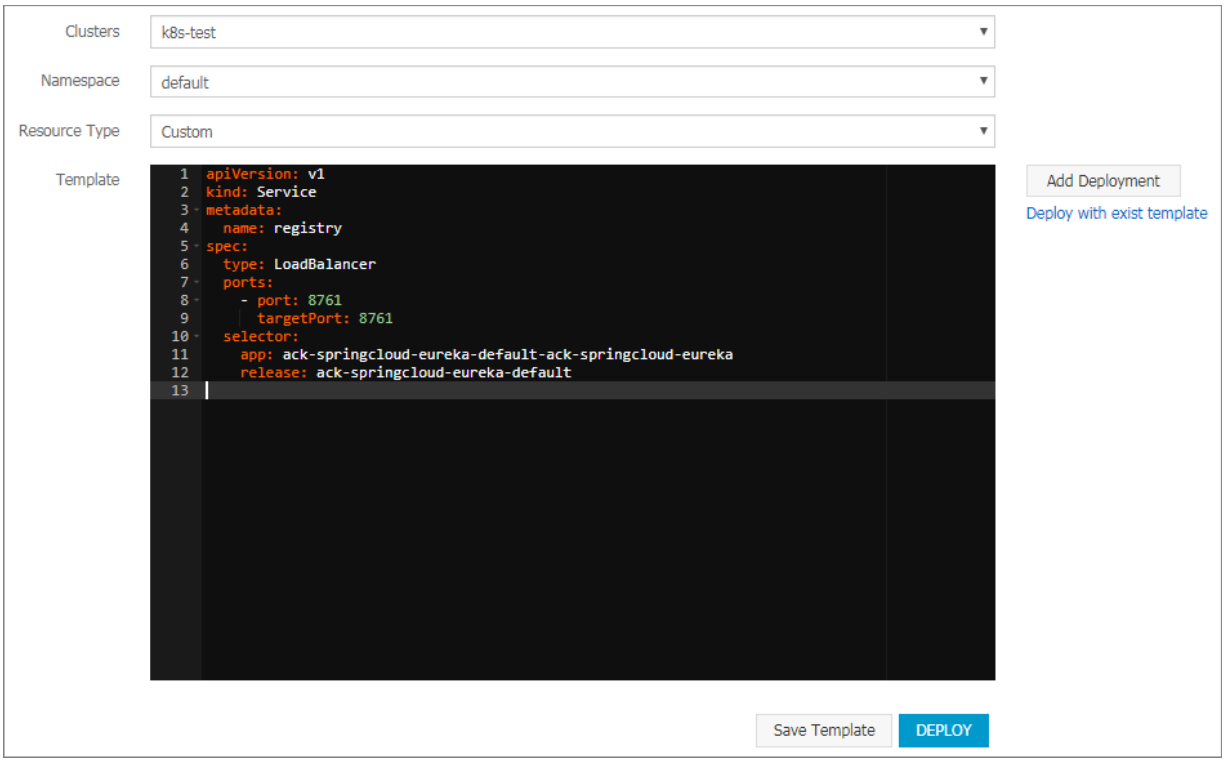
If you change the release name during App Catalog deployment, make the same changes to the followings.

```
apiVersion: v1
kind: Service
metadata:
  name: registry
spec:
  type: LoadBalancer
  ports:
    - port: 8761
      targetPort: 8761
  selector:
    app: ack-springcloud-eureka-default-ack-springcloud-eureka
    release: ack-springcloud-eureka-default
```

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

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

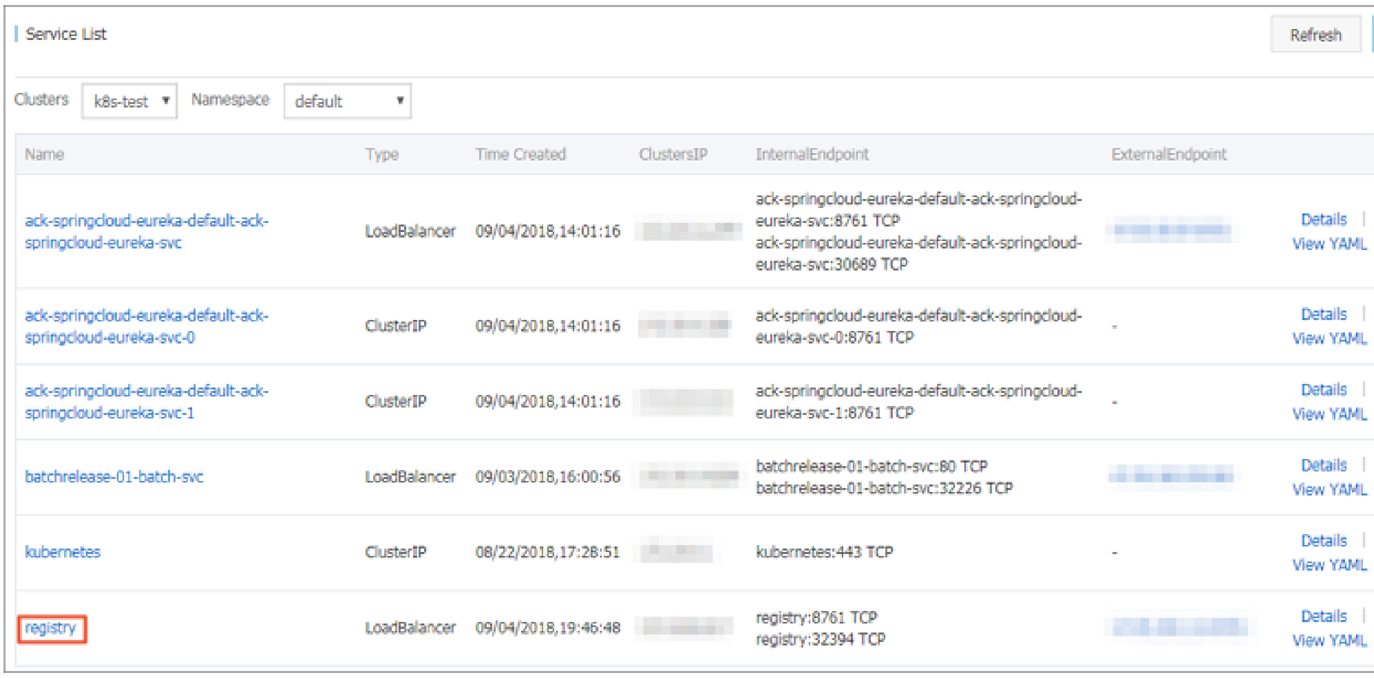
You can also do this through the console interface:



Clusters	k8s-test
Namespace	default
Resource Type	Custom
Template	

```yaml
apiVersion: v1
kind: Service
metadata:
  name: registry
spec:
  type: LoadBalancer
  ports:
    - port: 8761
      targetPort: 8761
  selector:
    app: ack-springcloud-eureka-default-ack-springcloud-eureka
    release: ack-springcloud-eureka-default
```

After deployment, enter the Service List page again, you can see the registry is created:



Service List

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	[redacted]	ack-springcloud-eureka-default-ack-springcloud-eureka-svc:8761 TCP ack-springcloud-eureka-default-ack-springcloud-eureka-svc:30689 TCP	[redacted]	Details | View YAML
ack-springcloud-eureka-default-ack-springcloud-eureka-svc-0	ClusterIP	09/04/2018,14:01:16	[redacted]	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	[redacted]	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	[redacted]	batchrelease-01-batch-svc:80 TCP batchrelease-01-batch-svc:32226 TCP	[redacted]	Details | View YAML
kubernetes	ClusterIP	08/22/2018,17:28:51	[redacted]	kubernetes:443 TCP	-	Details | View YAML
registry	LoadBalancer	09/04/2018,19:46:48	[redacted]	registry:8761 TCP registry:32394 TCP	[redacted]	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
```

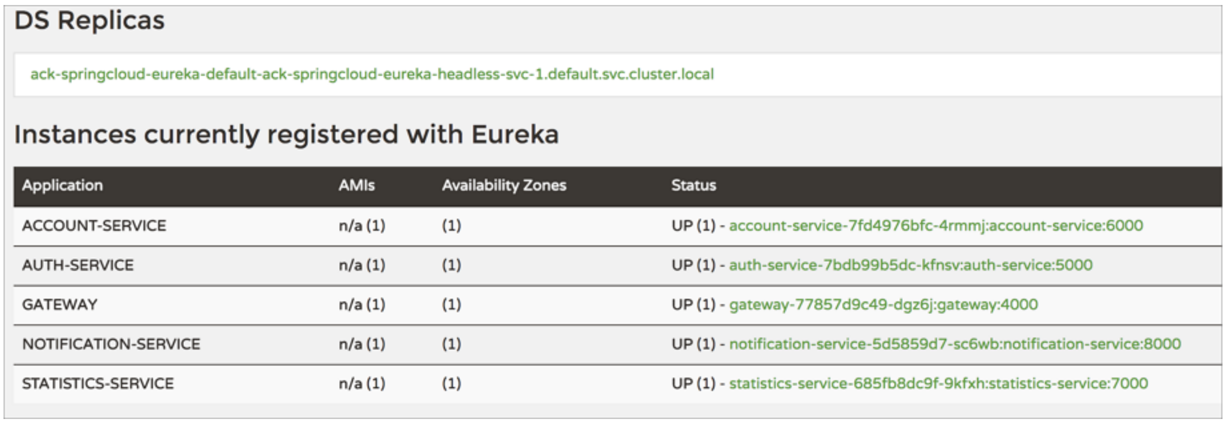
```
templates/registry-service.yaml
```

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 SpringClou d component, you do not have to repeat the deployment.

Execute the helm command to deploy PiggyMetrics again.

```
$ helm install -n piggymetrics piggymetrics-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

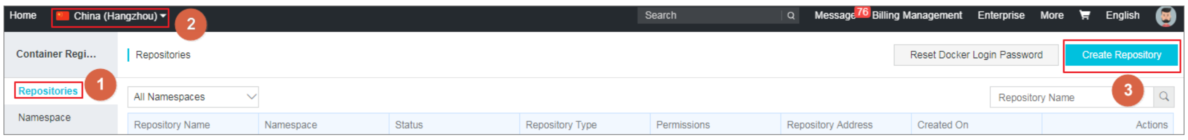
The PiggyMetrics application has been deployed to the environment with EurekaServer. Access `GATEWAY` to see the familiar login interface.

3.2 Use a private image repository to create an application

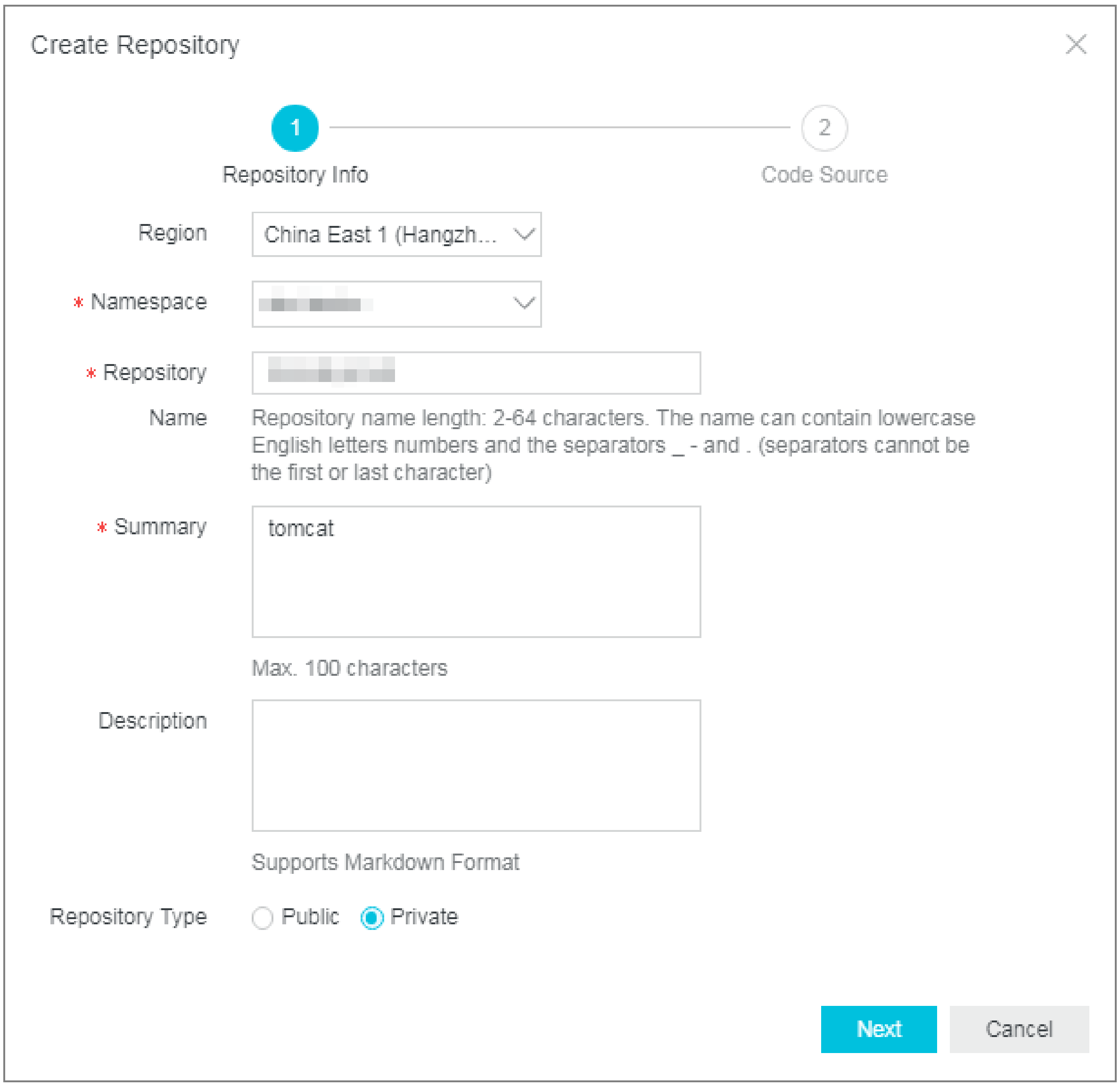
In many scenarios, an image in a private image repository is used for deploying an application. In this document, use Alibaba Cloud image repository service to create a private image repository, and create an application that uses this private image repository.

Step 1 Create a private image repository

1. Log on to the *Container Registry console*.
2. Click **Repositories** in the left-side navigation pane, select the target region, and lick **Create Repository**.
3. Configure the image repository in the dialog box, and then click **Create Repository**. In this example, select the private image repository type and set the code source as a local repository.



4. On the repositories page, select the target region, and you can see that the created image repository. Click **Manage**on the right.



Region: China East 1 (Hangzh...

* Namespace: [redacted]

* Repository Name: [redacted]

Repository name length: 2-64 characters. The name can contain lowercase English letters numbers and the separators _ - and . (separators cannot be the first or last character)

* Summary: tomcat

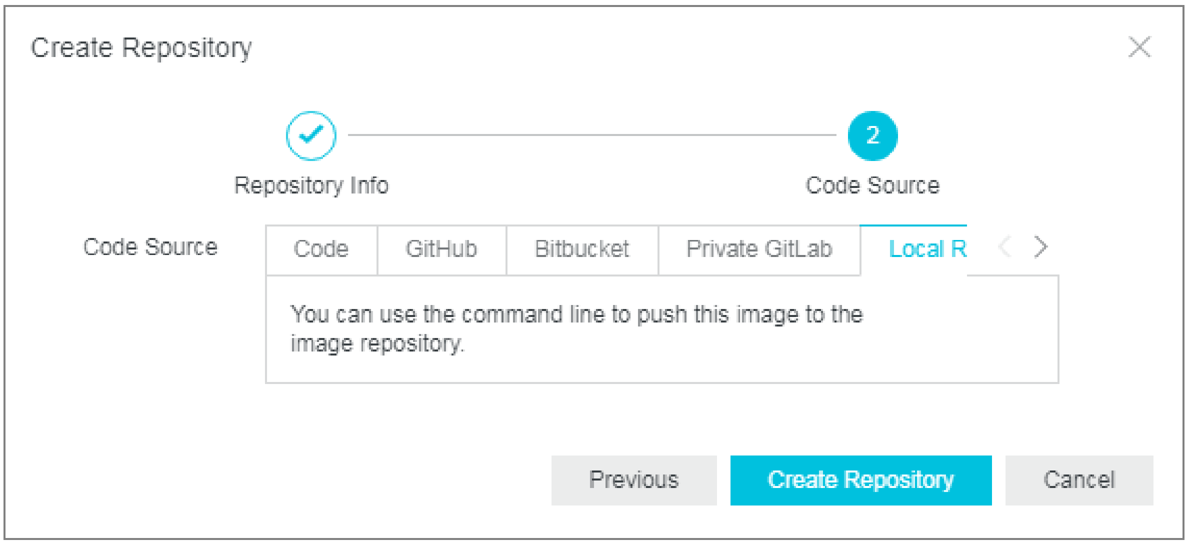
Max. 100 characters

Description:

Supports Markdown Format

Repository Type: Public [✓ Private]

5. On the repository management page, click **Details**, and you can follow the guide to use the private image repository.



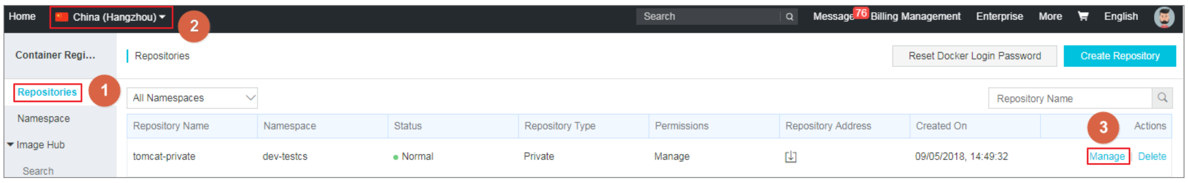
You can use the command line to push this image to the image repository.

6. Log on to the image repository in the Linux environment and upload the local image to the private image repository.

```
$ sudo docker login --username=abc@aliyun.com
 Password
            ## Image repository independent login password:
 Login Succe ed

$ dockeagesr im
           #This example is tomcat ages
REPOSITORY
        TAG                 IMAGE ID            CREATED
SIZE
tomcat
        latest              2d43521f2b1a        6 days ago
463MB

 $ sudo docker tag [ImageId] registry.cn-hangzhou.aliyuncs.com/
kubernetes-java/tomcat-private:[Image version number]            #V1
 in this example
 $ sudo docker push registry.cn-hangzhou.aliyuncs.com/kubernetes-
java/tomcat-private:[Inage version number]                       #V1
 in this example

 The push refers to a repository [registry.cn-hangzhou.aliyuncs.com/
kubernetes-java/tomcat-private]
9072c7b03a1b: Pushed
f9701cf47c58: Pushed
365c8156ff79: Pushed
2de08d97c2ed: Pushed
6b09c39b2b33: Pushed
4172ffa172a6: Pushed
1dccf0da88f3: Pushed
d2070b14033b: Pushed
63dcf81c7ca7: Pushed
ce6466f43b11: Pushed
719d45669b35: Pushed
3b10514a95be: Pushed
```

```
V1: digest: sha256:cded14cf64697961078aedfdf870e704a5227018
8c8194b6f70c778a8289d87e size: 2836
```

7. Return to the image repository detail page, and click **Image version** in the left navigation pane, you can see that the image has been uploaded successfully, and you can view the image version information.



Repository Name	Namespace	Status	Repository Type	Permissions	Repository Address	Created On	Actions
tomcat-private	dev-testcs	● Normal	Private	Manage		09/05/2018, 14:49:32	Manage Delete

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
imagePullSecrets:
    - 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:
    matchLabels:
      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:
        - containerPort: 8080
      imagePullSecrets:
      - name: regsecret
```

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