Alibaba Cloud
Alibaba Cloud Container Service for Kubernetes

User Guide for Serverless Kubernetes Clusters

Issue: 20190904

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 Overview

Alibaba Cloud Serverless Kubernetes allows you to quickly create Kubernetes container applications without having to manage and maintain clusters and servers . The Pay-As-You-Go billing method is applied, which is based on the amount of CPU and memory resources used by applications. With Serverless Kubernetes, you can focus on designing and building applications, rather than managing the infrastructure on which your applications run. Serverless Kubernetes is based on the Alibaba Cloud elastic computing architecture and is fully compatible with the Kuberentes API, combining the security, elasticity, and Kubernetes ecosystem of virtualized resources.

Benefits

- Easy to use: You can deploy an application in a serverless Kubernetes cluster in seconds without the need to manage the infrastructure of the serverless Kubernetes cluster. A serverless Kubernetes cluster is highly available and secure.
- Compatible with multiple tools and platforms: You can use the Kubernetes command line interface or API to deploy containerized applications. In addition, you can migrate your applications to Alibaba Cloud Container Service for Kubernetes (ACK).
- Secure isolation: Serverless Kubernetes clusters are developed on the basis of the elastic computing architecture of Alibaba Cloud. Containers on which different applications run are isolated from each other to prevent mutual interference.
- Scalable resources upon workload requirements: Resources required by your applications can be expanded according to the workload requirements.
- High interconnection: Containerized applications that run in a serverless Kubernetes cluster can use more basic services provided by Alibaba Cloud. The containerized applications in ACK can interconnect with the existing applications and databases in your VPC, and the applications that run on a virtual machine.

Regions available for public beta

Currently, Serverless Kubernetes clusters of Alibaba Cloud Container Service are in public beta stage. Now, only several regions are available for public beta. Other regions are going to be available soon.

Limits

Pods in a serverless Kubernetes cluster are created based on Elastic Container Instance (ECI). For more information about the pod specifications and pod usage limits, see Limits.

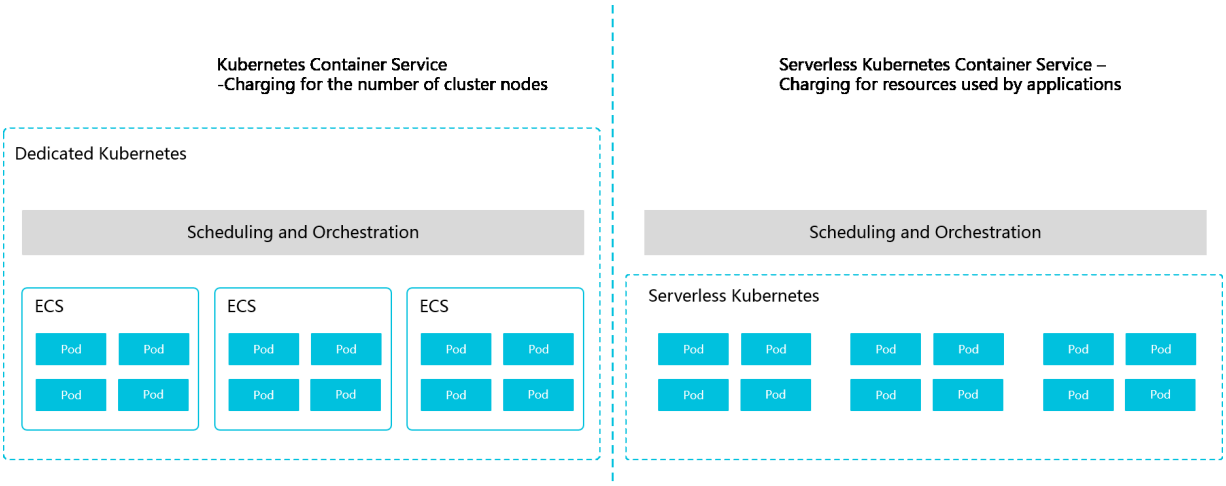
Pricing

Serverless Kubernetes clusters are free of charge.

For more information about ECI pricing, see Pricing.

Each type of resource (such as an SLB instance and a private zone) used in a serverless Kubernetes cluster are charged according to the price specified by the corresponding product.

Comparison with Container Service



Scenarios

The applications that run in a serverless Kubernetes cluster can be used in the following scenarios:

- Complete multi-media tasks
- Capture and modify data
- Process the sensor data generated by Internet of Things
- Perform flow computing
- Develop chat robots
- Perform batch computing
- Develop web applications
- Develop backend services of mobile applications

- **Complete business logic processing tasks**
- **Implement continuous integration**

2 Kubernetes supported functions

API version

Kubernetes 1.9 API is supported.

Application load

- Deployment, StatefulSet, Job/CronJob, Bare Pod are supported.
- DaemonSet is not supported.

Pod definition

Supports starting multiple containers, setting environment variables, RestartPolicy, health check commands, and mounting volumes.

Load balancing

- Supports creation of Load Balancer type applications.
- Ingress is supported.
- NodePort type is not supported.

Configuration

Secret and ConfigMap are supported.

Storage

- EmptyDir and NFS volume types are supported.
- PersistentVolume and PersistentVolumeClaim are not supported.

Namespace

Only the default namespace can be viewed, and no namespaces can be added.

Node

Node information of Kubernetes cannot be viewed.

Events

Default namespace events can be viewed.

Containers logs

View the container logs in real time by using `kubectl logs` .

Container exec/attach

Enter the container to run the commands by using `kubectl exec` .

3 Cluster management

3.1 Create a serverless Kubernetes cluster

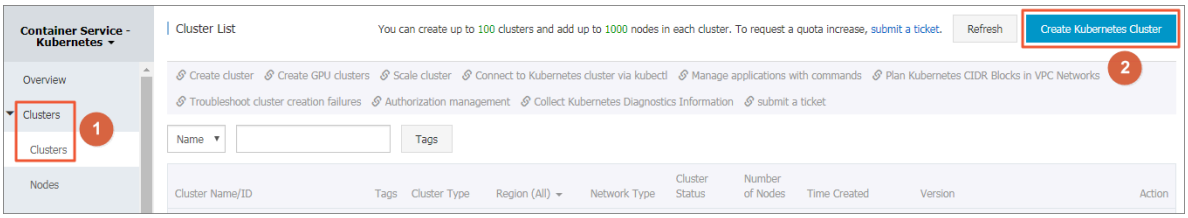
This topic describes how to use the Container Service console to create a serverless Kubernetes cluster.

Prerequisites

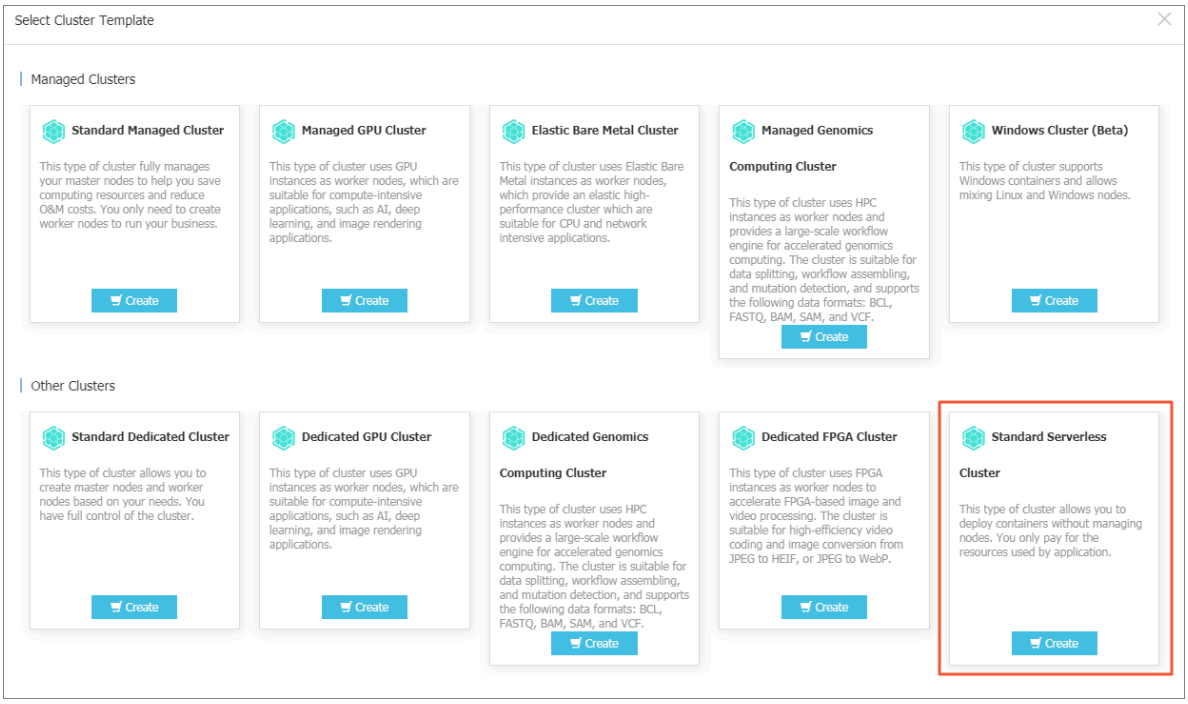
Container Service and RAM must be activated.

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.

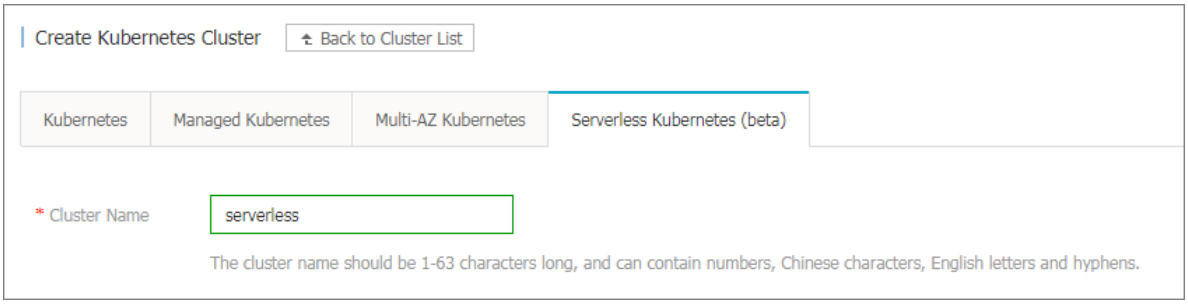


4. On the Select Cluster Template page, find the Standard Serverless Cluster template, and click Create in the template.

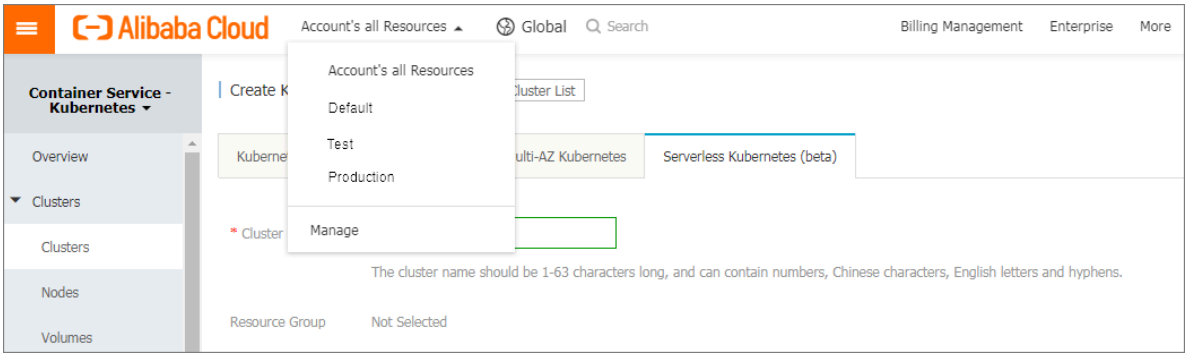


5. Enter the cluster name.

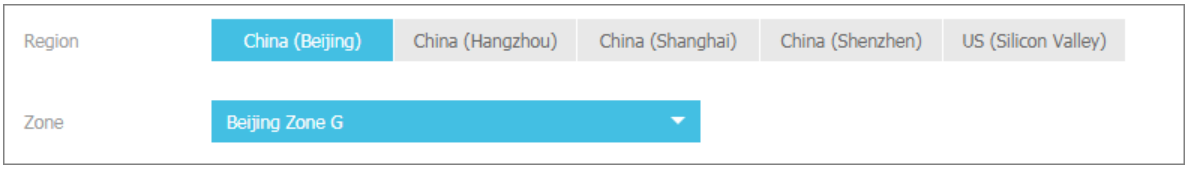
> **Note:**
> The cluster name must be 1 to 63 characters in length and can contain letters, numbers, Chinese characters, letters, and hyphens (-).



6. Select the resource group for the cluster.

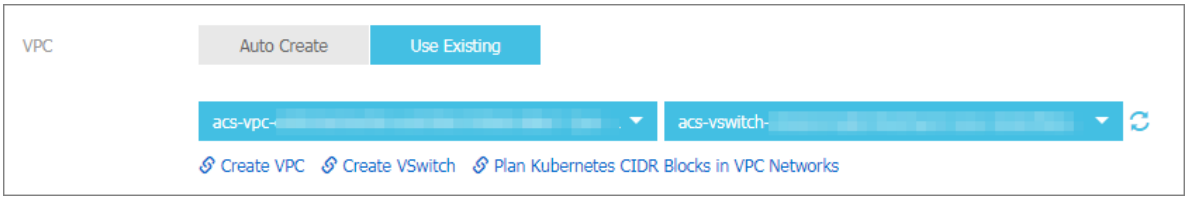


7. Select the region and zone where you want to locate the cluster.



8. Set a VPC for the serverless cluster.

> **Note:**
> Kubernetes clusters support only the VPC network type.



To set a VPC for the serverless cluster, you can click one of the following two options:

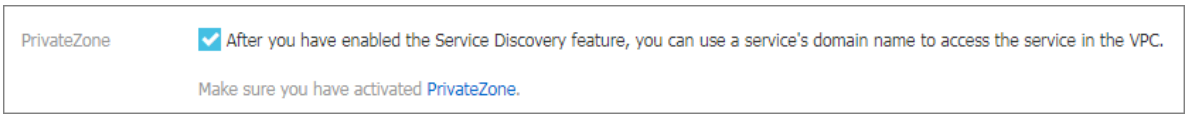
- Auto Create: Set the system to automatically create a VPC. Furthermore, in the VPC, the system also creates an NAT gateway and sets an SNAT rule.
- Use Existing: Select an existing VPC from the VPC drop-down list, and then select a VSwitch from the VSwitch drop-down list.

  If you want to enable the cluster to access Internet resources, you must set an NAT gateway for the cluster. For more information, see #unique_7.

9. Set a NAT gateway and a SNAT rule for the VPC.

> **Note:**
> - If you set the system to automatically create a VPC, you must select the Nat Gateway check box. This action sets the system to automatically set a NAT gateway and an SNAT rule for the VPC.
> - If you select an existing VPC, you must manually set an NAT gateway or set an SNAT rule. Otherwise, the serverless cluster in the VPC cannot access the Internet, which results in a cluster creation failure.

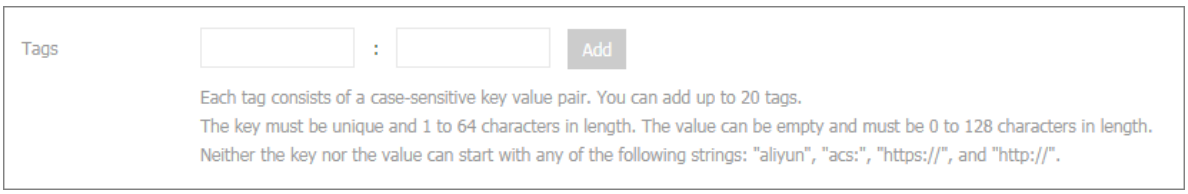
10. Set the PrivateZone-based service discovery feature for the cluster. This feature allows you to use a domain name to access the corresponding service in the VPC.



> **Note:**
> Before using this feature, you must activate the PrivateZone service, see Service discovery based on Alibaba Cloud DNS PrivateZone.

11. Attach a tag to the cluster.

    Enter a key and its value, and click Add.



12. Select the Terms of Service for Serverless Kubernetes check box.
13. In the upper-right corner, click Create

What's next

On the Cluster List page, view the serverless Kubernetes cluster you created.

In the Action column, click Manage to view the details of this cluster.

3.2 Connect to a Kubernetes cluster by using kubectl

To connect to a Kubernetes cluster from a client computer, use the Kubernetes command line client kubectl.

Procedure

1. Download the latest kubectl client from the Kubernetes version page.
2. Install and set the kubectl client.

   For more information, see Install and set kubectl.

3. Configure the cluster credentials.

   You can view the cluster credentials on the cluster information page.

   a) Log on to the Container Service console.
   b) Under Kubernetes, click Clusters in the left-side navigation pane.
   c) Click Manage at the right of the cluster.
   d) In the Connection Information section, view the master node SSH IP address.

   e) Copy the cluster credentials to a local file, and you can create and save the cluster credentials to `$ HOME /. kube / config` (location where kubectl credentials are to be stored). You can also name a new file, such as `/ tmp / kubeconfig` , and run the `export KUBECONFIG =/ tmp / kubeconfig` command.
   f) After the preceding operation is performed, you can confirm the cluster connectivity by running the following command.

```
# kubectl get pod
 No resources found .
```

What's next

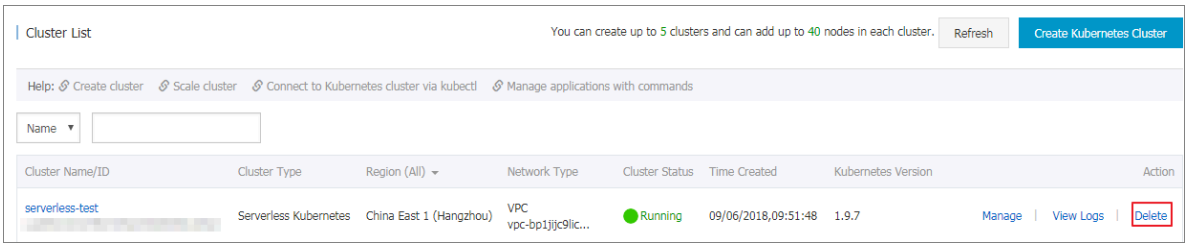
After the configuration is complete, you can use kubectl to access the Kubernetes cluster from a local computer.

3.3 Delete a cluster

You can delete clusters that are no longer in use in the Container Service console.

Procedure

1. Log on to the Container Service console.
2. Under Kubernetes, click Clusters in the left-side navigation pane.
3. Select the target cluster and click Delete on the right.



4. In the dialog box, click OK.

4 Application management

4.1 Manage applications by using commands

You can create applications or view containers in applications by using commands.

Prerequisites

Before using commands, configure #unique_13 first.

Create an application by using commands

Execute the following statements to run a simple container (a Nginx Web server in this example).

```
root @ master  #  kubectl   run   nginx  -- image = registry . cn -
hangzhou . aliyuncs . com / spacexnice / netdia : latest
```

This command creates a service portal for this container. Specify `-- type = LoadBalanc er` and Alibaba Cloud Server Load Balancer route will be created to the Nginx container.

```
root @ master  #  kubectl   expose   deployment   nginx  -- port = 80
 -- target - port = 80  -- type = LoadBalanc  er
```

View containers by using commands

Run the following command to list all the running containers in the default namespaces.

```
root @ master  #  kubectl   get   pods
NAME                                      READY       STATUS
RESTARTS     AGE
nginx - 2721357637 - dvwq3                   1 / 1         Running
  1            9h
```

4.2 Create an application by using an image

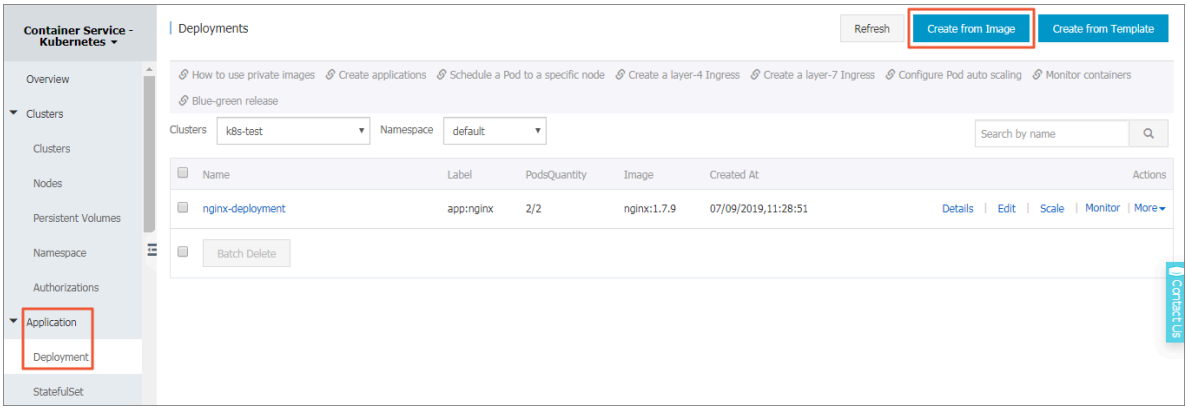
Prerequisites

Create a Serverless Kubernetes cluster. For more information, see #unique_15.

Procedure

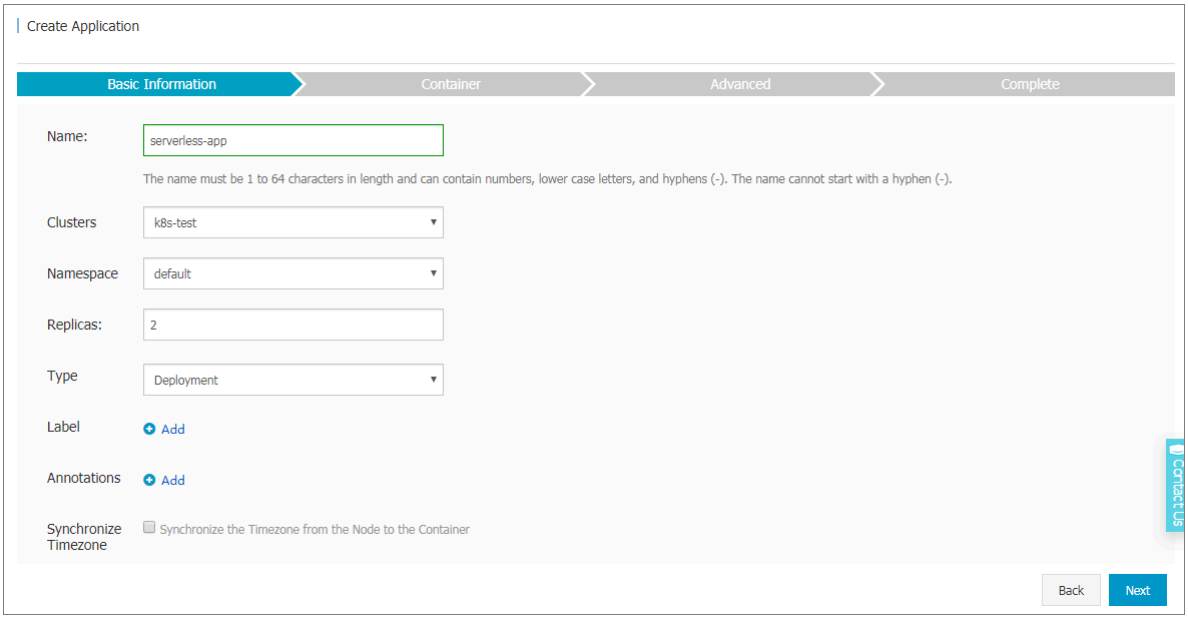
1. Log on to the Container Service console.

2. In the left-side navigation pane under Container Service-Kubernetes, choose Application > Deployment.
3. In the upper-right corner, click Create from Image



4. Set Name, Clusters, Namespace, Replicas, Type, Label, and Annotations. Then, click Next.

   If the namespace is not set, the default namespace is used.



5. Select the image you want to use and the version of the image.
   - Image name: In the displayed dialog box click Select Image, and then click OK. You can also enter the private registry. In the format of `domainname / namespace / imagename : tag`. In this example, the image name is nginx.
   - Image version: Click Select image version to select the version. If not specified, the latest version is used by default.

6. Configure the number of containers (Scale).

   This example is a single container pod, and if multiple containers are specified, the same number of pods is started.

7. Configure resource limits and required resources for the container.

   Serverless Kubernetes is currently in beta stage and only supports the 2C4G specification.

   - Resource limits: You can specify the maximum amount of resources that the application can use, including CPU and memory to prevent excessive use of resources.
   - Required resources: The amount of resources reserved for the application, including CPU and memory. That is, container monopolizes the resource, so as to prevent other services or processes from competing for the resource due to insufficiency, resulting the application to be unavailable.

   CPU is measured in millicores (one thousandth of one core). Memory is measured in bytes, which can be GB, MB, or KB.

8. Configure the environment variable.

   You can configure the environment variable for the pod in the format of key-value pairs to add the environment label or pass the configurations for the pod. For more information, see Pod variable.

9. Configure the container.

   You can configure the Command and Arguments for the container running in the pod.

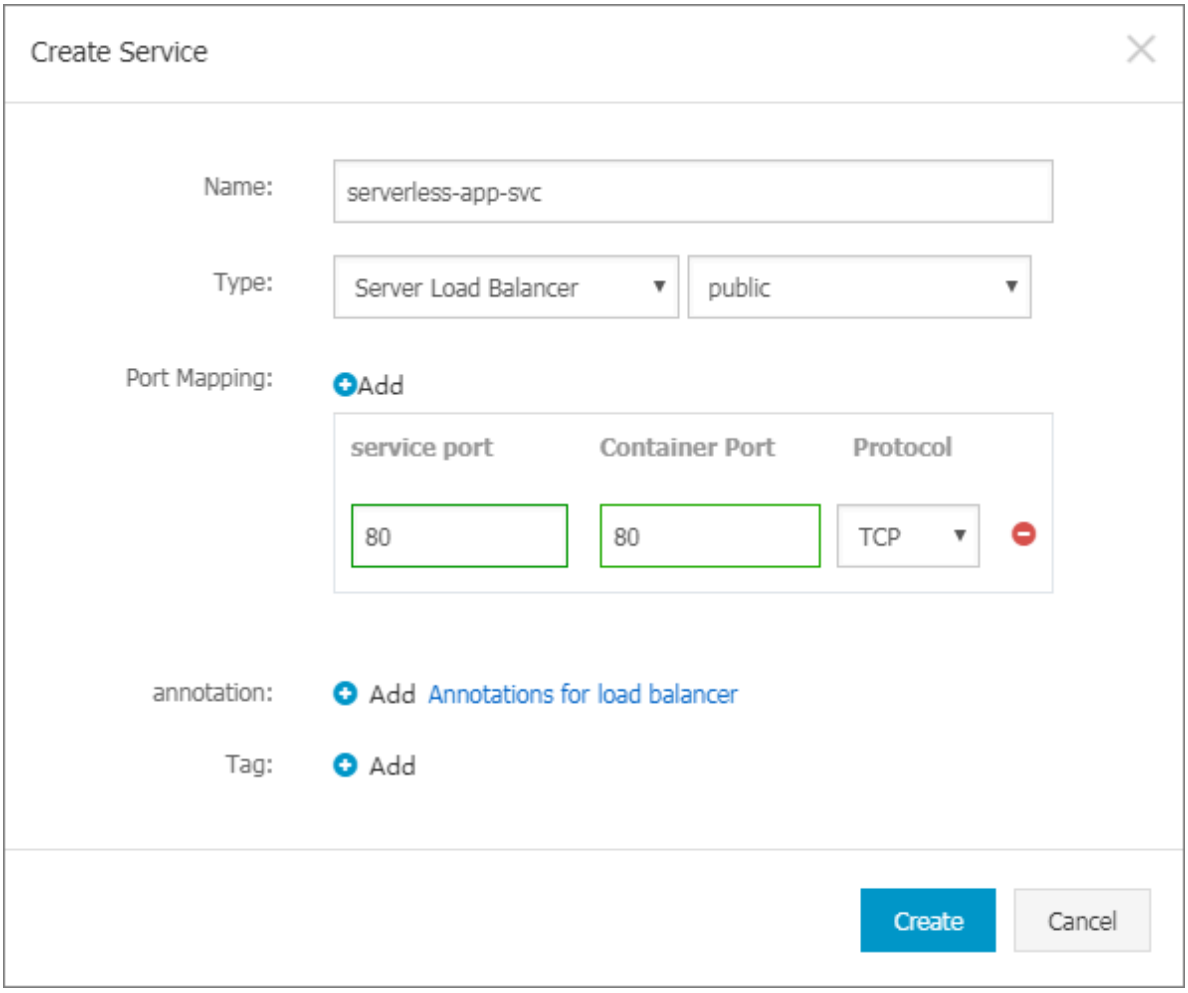
   Command and Args: If not configured, the default settings for the image is used. If configured, the default settings are overwritten. If only arguments are configured, when the container starts, the default command executes the new arguments.

   Command and arguments cannot be modified after pod is created.

10. After the application configuration is complete, click Next to enter the Access Settings page to set up a service that binds the backend pod.

    You can select not to create a service, or select a service type. Currently, only load balancing type is supported.

    - Load Balancing: Load Balancer is a load balancing service provided by Alibaba Cloud, public network access or intranet access can be used.
    - Name: By default, a service name with the application name suffix `svc` is generated, in this example serverless-app-svc. Name of the service can be modified.
    - Port mapping: Specify the port mapping between service and container, and select TCP or UDP as the protocol.

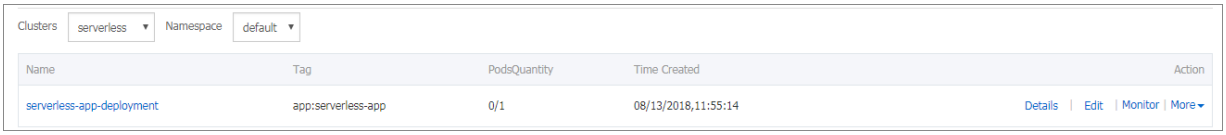


11. After the access configuration is complete, click Create.

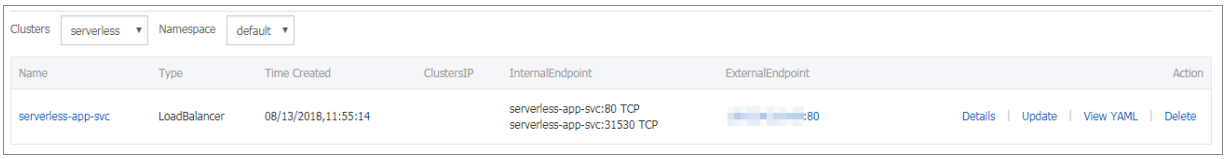
What's next

After the creation is successful, go to the creation completion page. The objects contained in the application are displayed. You can click View to view the deployment list.

You can view the new serverless-app-svc under the deployment list.



Click Ingresses and Load Balancing > Service in the left-side navigation pane to see the new service serverless-app-svc under the list of services.



Access the external endpoint in the browser to access the Nginx welcome page.

4.3 Create a service

Kubernetes service, which is generally called a microservice, is an abstraction which defines a logical set of pods and a policy by which to access them. This set of pods can be accessed by the service, typically by using the Label Selector.

Kubernetes pods are created and deleted in a short time even if they have their own IP addresses. Therefore, using pods directly to provide services externally is not a solution of high availability. The service abstraction decouples the relationship between the frontend and the backend. Therefore, the loose-coupling microservice allows the frontend to not care about the implementations of the backend.

For more information, see Kubernetes service.

Prerequisites

You have successfully created a Serverless Kubernetes cluster, see #unique_15.

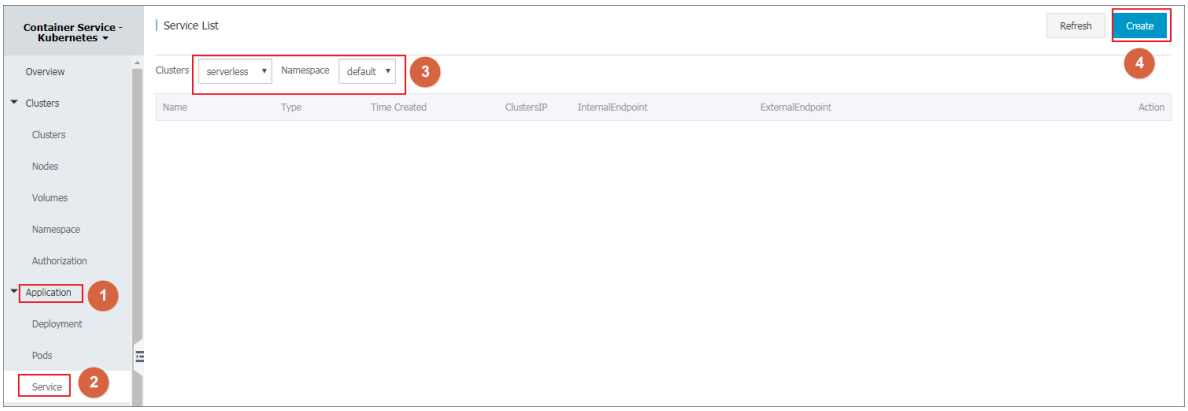
Step 1. Create a deployment

Create a deployment by using the image, in this example create serverless-app-deployment. For more information, see #unique_17.

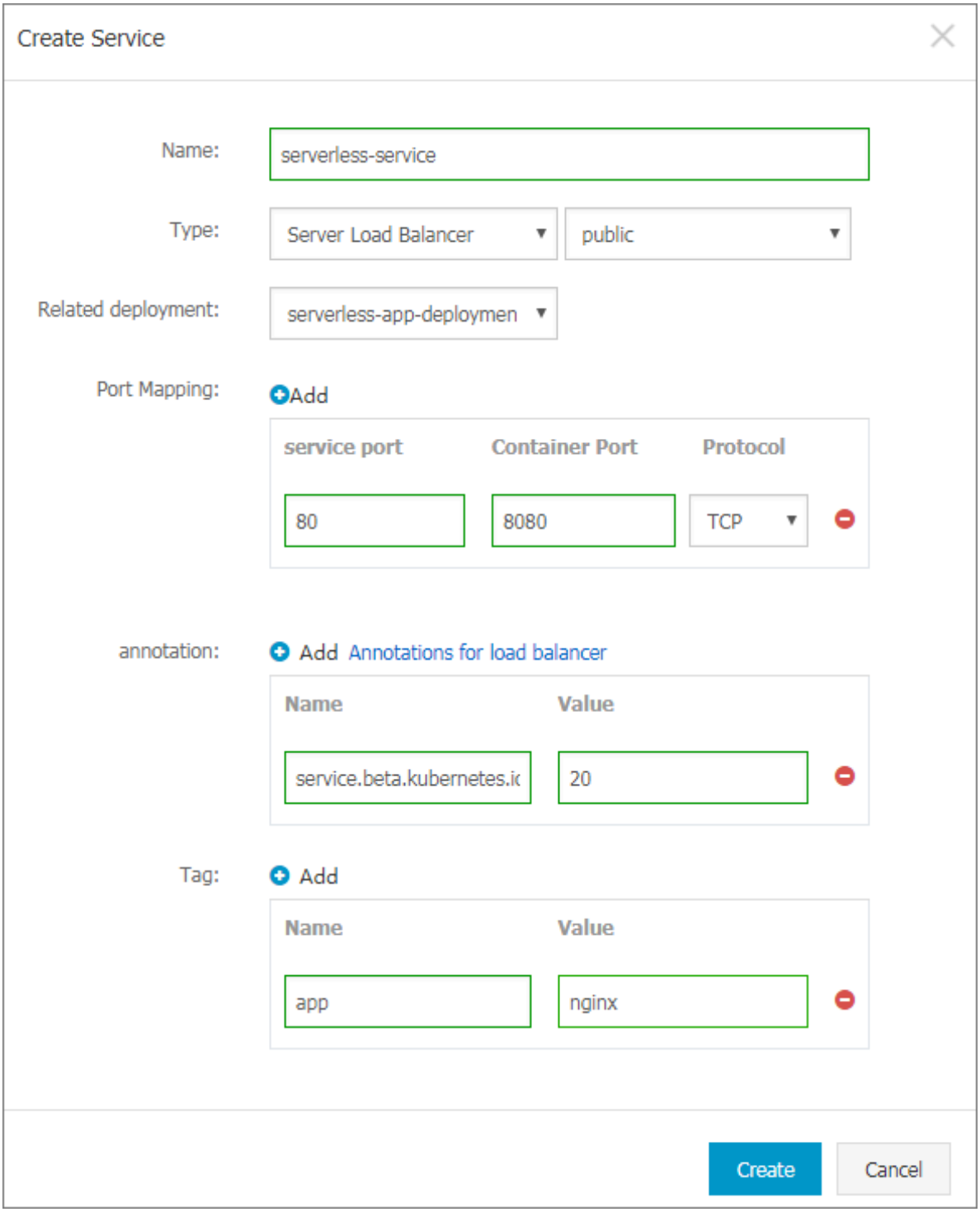
Step 2. Create a service

1. Log on to the Container Service console.

2. Under Kubernetes, click Application > Service in the left-side navigation pane to enter the Service List page.
3. Select the target cluster and namespace and click Create in the upper-right corner.



4. Complete the configurations in the displayed Create Service dialog box.

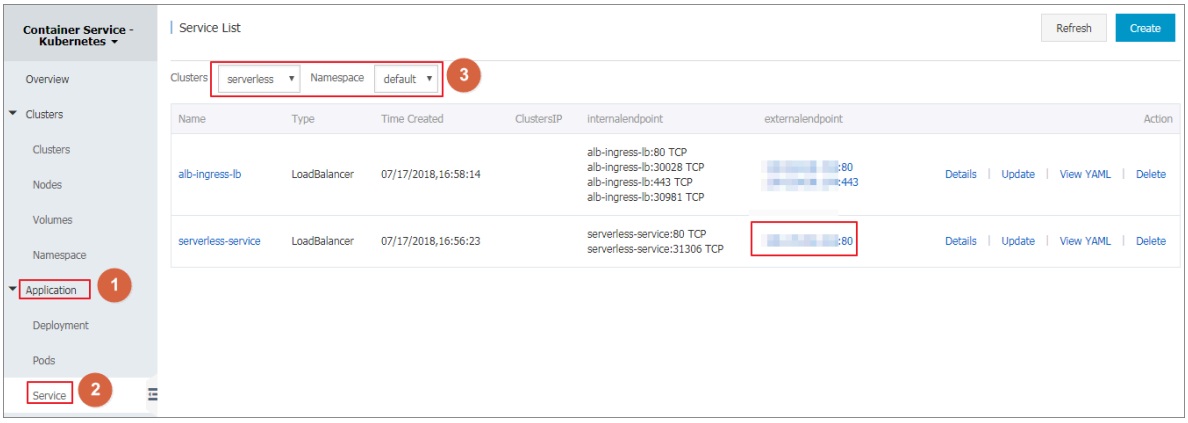


- Name: Enter the name of the service, in this example serverless-service.
- Type: Select the service type, namely, the access method of the service. Currently, only load balancing type is supported. Load Balancer is the load

balancing service provided by Alibaba Cloud, public network access or intranet access can be used.

- Related deployment: Select the backend object to bind with this service, in this example, select nginx-deployment-basic. The corresponding Endpoints object is not created if no deployment is selected here. You can manually map the service to your own endpoints. For more information, see services-without-selectors.
- Port Mapping: Add the service port and container port. The container port must be the same as the one exposed in the backend pod.
- Annotation: Add an annotation to the service and configure load balancing parameters such as `service.beta.kubernetes.io/alicloud-Loadbalancer-bandwidth:20` indicating that the bandwidth of the service is set to 20 Mbit/s to control the traffic of the service. For more information, see #unique_18.
- Label: You can add a label to the service to identify the service.

5. Click Create, and the serverless-service appears in the list of services.



6. You can view the basic information of the service and access the external endpoint of serverless-service in your browser.

Then, you have created a service that is related to a backend deployment and accessed the Nginx welcome page successfully.

4.4 Delete a service

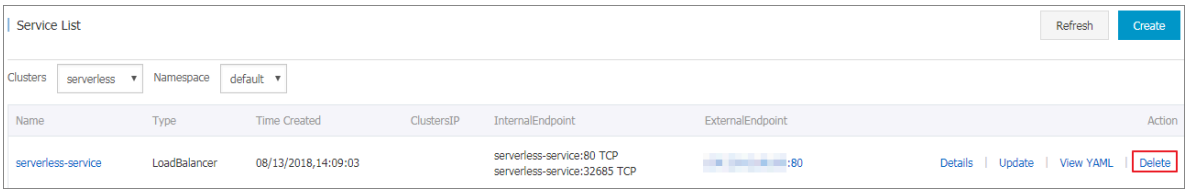
You can quickly delete a service in the Container Service console.

Prerequisites

- You have successfully created a Serverless Kubernetes cluster, see #unique_15.
- You have successfully created a service, see #unique_20.

Procedure

1. Log on to the Container Service console.
2. Under Kubernetes, click Application > Service in the left-side navigation pane to enter the Service List page.
3. Select the cluster and namespace, select the target service (serverless-service in this example), and click Delete on the right.



4. In the displayed window, click OK to confirm the deletion, and the service disappears from the list of services.

4.5 View pods

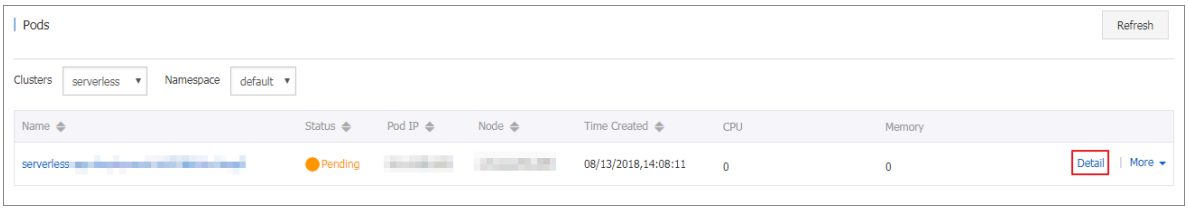
You can view the pods of the Serverless Kubernetes cluster in the Container Service console.

Procedure

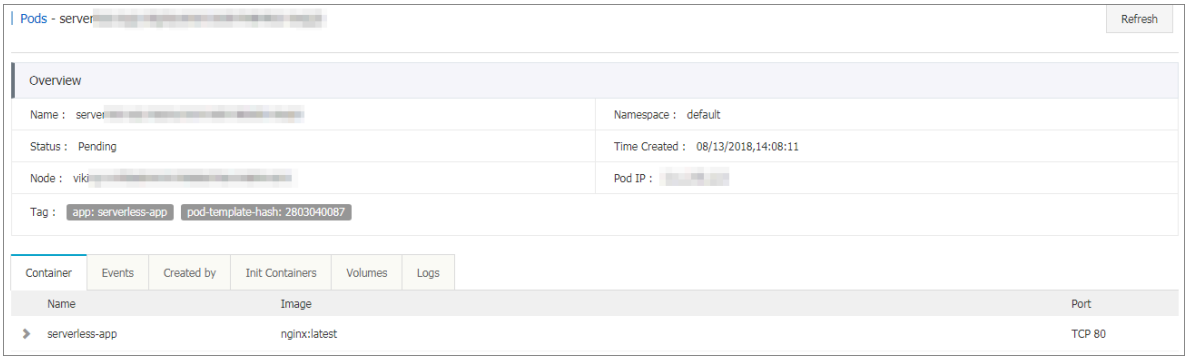
1. Log on to the Container Service console.
2. Under Kubernetes, click Application > Pods to enter the Pods page.
3. Select the target cluster and namespace, the target pod, and click Details on the right.

Note:

You can update or delete a pod. For pods created by using deployments, we recommend that you manage these pods by using deployments.



4. View the pod details.



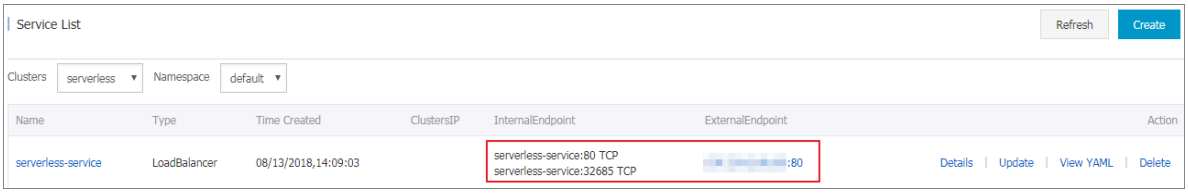
4.6 View services

If the external service is configured when you create the application, in addition to running containers, system creates the external services for pre-assigning the Server Load Balancer to bring traffic to the containers in the cluster.

Procedure

1. Log on to the Container Service console.
2. Under Kubernetes, click Application > Service in the left-side navigation pane to enter the Service List page.
3. You can view the deployed services by selecting the required clusters and namespaces.

   You can view information, such as the name, type, creation time, cluster IP, and external endpoints of the service. In this example, view the external endpoint (IP address) assigned to the service.



4.7 Service discovery based on Alibaba Cloud DNS Private Zone

Alibaba Cloud Serverless Kubernetes supports service discovery. Currently, service discovery for intranet SLB and headless service is supported.

Alibaba Cloud DNS Private Zone

Alibaba Cloud DNS Private Zone is a private domain name resolution and management service based on VPC (virtual private cloud) environment of Alibaba Cloud. You can map a private domain name to an IP resource address in one or more of the custom private networks, while your private domain names are not accessible in other network environments.

Prerequisites

1. You have activated Alibaba Cloud DNS Private Zone in the Alibaba Cloud DNS console.
2. You have successfully created a Serverless Kubernetes cluster, see #unique_15.
3. You have connected to a Serverless Kubernetes cluster, see #unique_13.

Procedure

1. Connect to the Kubernetes cluster by using kubectl and run the following command to confirm the connection to the cluster.

```
kubectl cluster - info
 Kubernetes master is running at https :// xxxxxx .
serverless - 1 . kubernetes . cn - shanghai . aliyuncs . com : 6443
```

2. Deploy and create a service. Currently, only intranet service and headless service are supported.

   Take the intranet service as an example. Create a sample file: `nginx - deployment - basic . yaml .`

```
vim nginx - deployment - basic . yaml
```

   The sample template is as follows, copy the following yaml code in the yaml file. Then run the `kubectl create - f nginx - deployment - basic . yaml` command to create it.

```
apiVersion : apps / v1beta2 # for versions before 1 . 8 .
0 use apps / v1beta1
 kind : Deployment
 metadata :
   name : nginx - deployment - basic
```

```
    labels :
      app :  nginx
  spec :
    replicas :  2
    selector :
      matchLabel  s :
        app :  nginx
    template :
      metadata :
        labels :
          app :  nginx
      spec :
        containers :
       -  name :  nginx
          image :  nginx : 1 . 7 . 9                #  replace   it
   with   your   exactly  < image_name : tags >
          ports :
         -  containerP  ort :  80
---
 apiVersion :  v1
 kind :  Service
 metadata :
   name :  nginx - service - intranet            # Access   by
service   name   as   short   domain   name
   annotation  s :                           ## Add   annotation
     service . beta . kubernetes . io / alicloud - loadbalanc  er -
address - type :  intranet
 spec :
   ports :
  -  port :  80
     protocol :  TCP
   selector :
     app :  nginx
   type :  LoadBalanc  er
```

You can also create services of the headless service type, as shown in the following sample template.

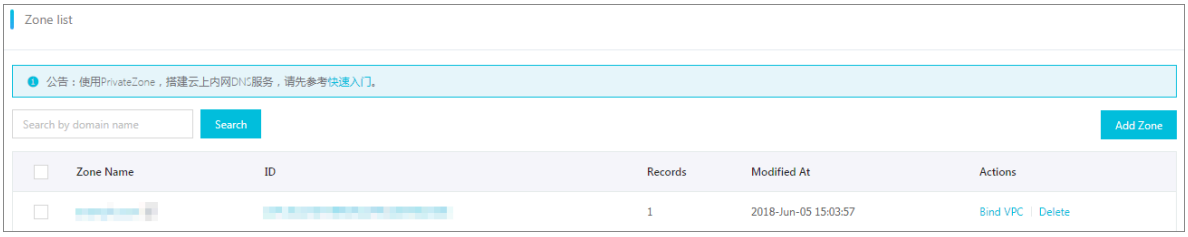
```
apiVersion :  v1
kind :  Service
metadata :
  name :  nginx - service - headless
spec :
  ports :
 -  port :  80
    protocol :  TCP
  selector :
    app :  nginx
  clusterIP :  None
```

3. Execute the following command to view the health status of the application.

```
kubectl   get   svc , pod , deployment
```

4. Log on to the Alibaba Cloud DNS console.

5. Click Private Zone > Zone List in the left-side navigation pane to see that the record is automatically generated under this list.



You can access Service (long domain or short domain) by the private domain name in VPC network environment.

- Long domain access: In this example, `nginx - service - intranet . $ NAMESPACE . svc . cluster . local` , where *$NAMESPACE* is the Serverless cluster ID, which can be seen in the console, or you can also use environmental variables in the yaml file of the pod.

```
env :
      -  name :  NAMESPACE
         valueFrom :
           fieldRef :
             fieldPath :  metadata . namespace
```

- Short domain access: nginx-service-intranet or nginx-service-headless, the name of the service defined in the yaml template.

For more information, see serverless-k8s-examples.

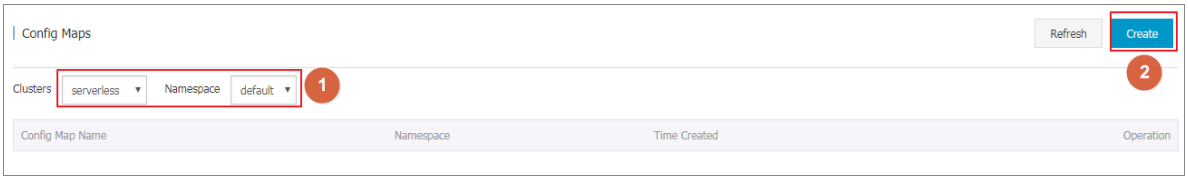
5 Config Map

5.1 Create a Config Map

This topic describes how to create a Config Map by using the Container Service console.

Procedure

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

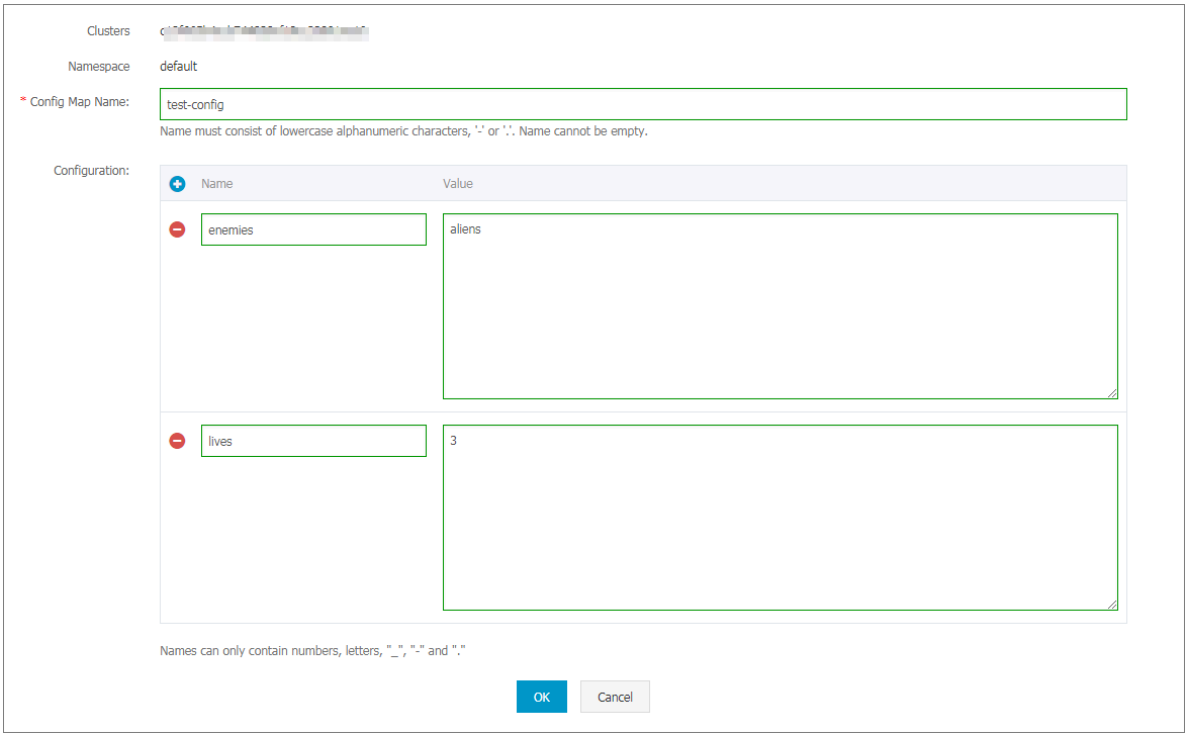


4. Complete the configuration and then click OK.
   - Namespace: Select the namespace to which the Config Map belongs. Config Map is a Kubernetes resource object that must be applied to the namespace.
   - Config Map Name: Enter the Config Map name, which can contain lowercase letters, numbers, hyphens (-), and periods (.). The name cannot be empty.

Other resource objects must reference the Config Map name to obtain the configuration information.

- Configuration: Enter the Variable Name and the Variable Value. Then, click Add on the right. You can also click Edit, complete the configuration in the displayed dialog box, and click OK.

In this example, configure the variables enemies and lives to pass the parameters aliens and 3 respectively.



5. You can view the Config Map test-config on the Config Maps page after clicking OK.

5.2 Delete a Config Map

This topic describes how to delete a Config Map.

Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane under Container Service-Kubernetes, choose Configuration > Config Maps.
3. Select the target cluster and namespace. Then, find the target Config Map, and click Delete on the right.

5.3 Modify a Config Map

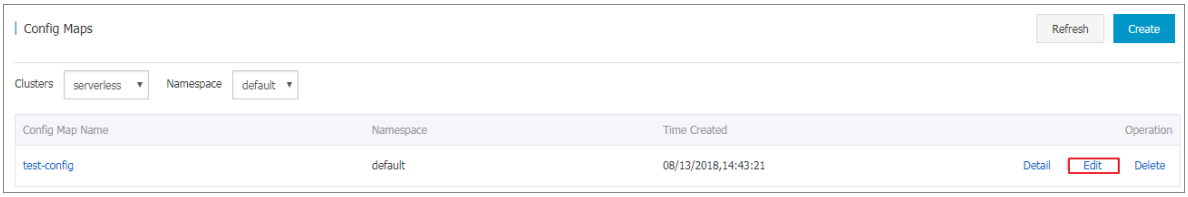
You can modify the configurations of a Config Map.

Context

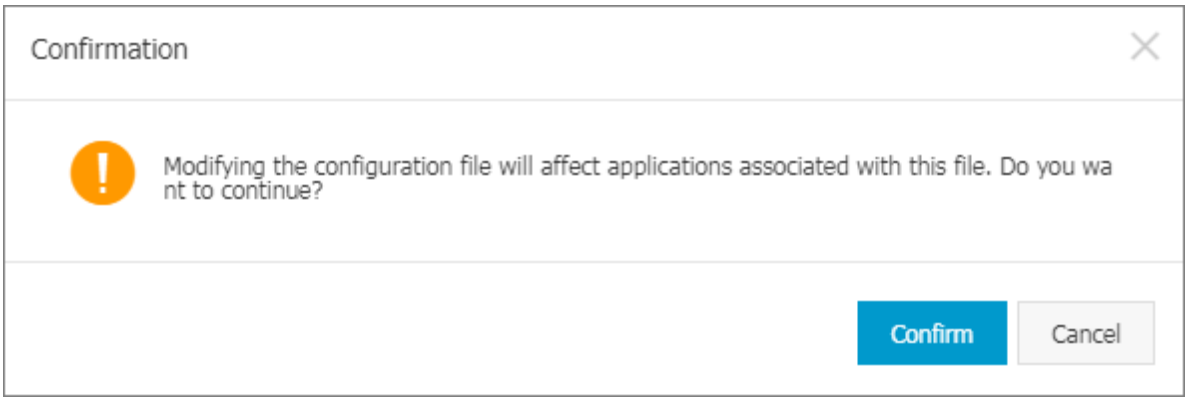
If you modify a Config Map, the applications that use the Config Map will be affected.

Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane under Container Service-Kubernetes, choose Configuration > Config Maps.
3. Select the target cluster, namespace, and Config Map you want to modify. Then, click Modify on the right.

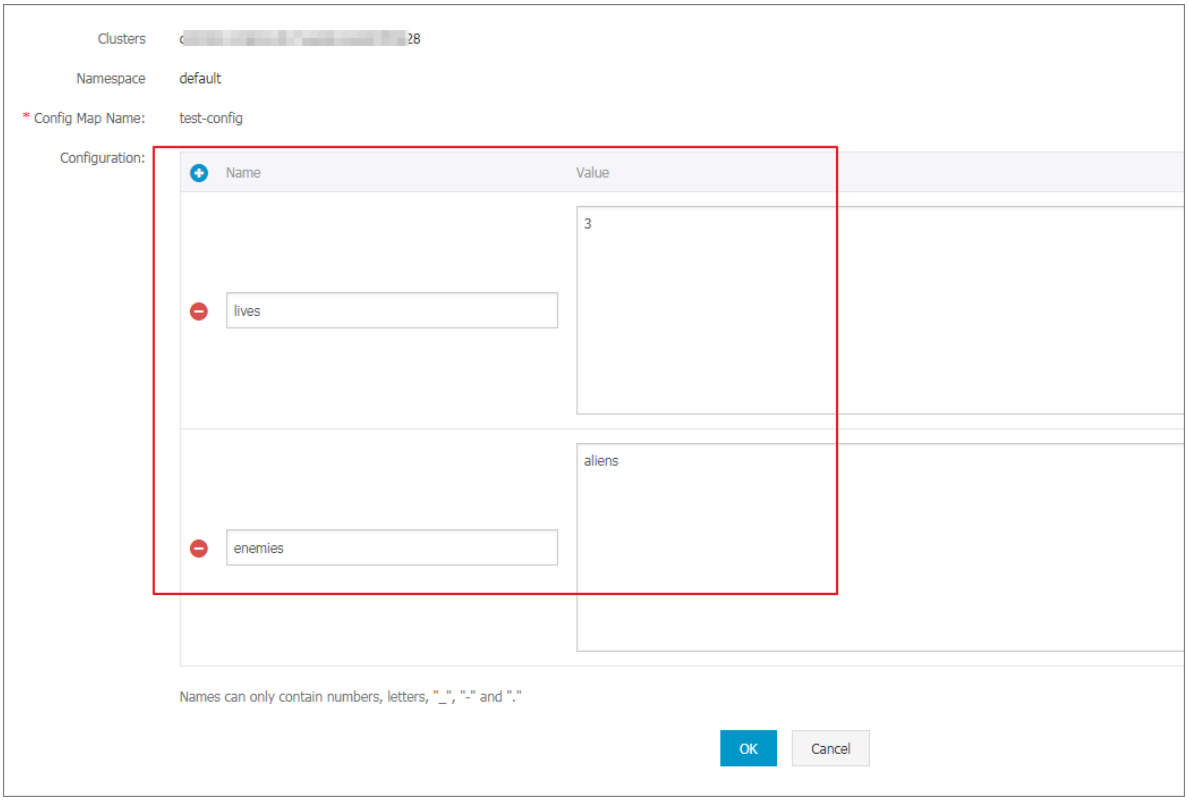


4. Click Confirm in the displayed dialog box.



5. Modify the configurations.

   - Click Edit on the right of the configuration you want to modify. Update the configuration and then click Save.
   - You can also click Edit configuration file. Finish editing, and click OK.



6. After modifying the configurations, click OK.

6 Server Load Balancer management

6.1 Use an SLB instance to access services

This topic describes how to access services by using an Alibaba Cloud Server Load Balancer (SLB) instance.

Operate by using command line

1. Create an Nginx application by using command line.

```
root @ master  #  kubectl   run   nginx  -- image = registry .
aliyuncs . com / acs / netdia : latest
root @ master  #  kubectl   get   po
NAME                                      READY       STATUS
RESTARTS     AGE
nginx - 2721357637 - d ****                   1 / 1         Running
   1            6s
```

2. Create Alibaba Cloud Server Load Balancer service for the Nginx application and specify `type = LoadBalanc  er` to expose the Nginx service to the Internet.

```
root @ master  #  kubectl   expose   deployment   nginx  -- port =
80  -- target - port = 80  -- type = LoadBalanc  er
root @ master  #  kubectl   get   svc
NAME                    CLUSTER - IP        EXTERNAL - IP
PORT ( S )                          AGE
nginx                   172 .**.**.***    101 .**.***.**    80 : 3
****/ TCP                     4s
```

3. Visit `http :// 101 .**.***.**` in the browser to access your Nginx service.

More information

Alibaba Cloud Server Load Balancer also supports parameter configurations such as health check, billing method, and load balancing. For more information, see Server Load Balancer configuration parameters.

Annotations

Alibaba Cloud supports a lot of Server Load Balancer features by using annotations.

Use existing intranet Server Load Balancer instance

1. Use Cloud Shell to access the target Kubernetes cluster.

2. Create a file `slb.svc`, copy the following code into the file, and run the `kubectl apply -f slb.svc` command.

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.beta.kubernetes.io/alicloud-loadbalancer-
address-type: intranet
    service.beta.kubernetes.io/alicloud-loadbalancer-
id: your-loadbalancer-id
    service.beta.kubernetes.io/alicloud-loadbalancer-
force-override-listeners: "true"
  labels:
    run: nginx
  name: nginx
  namespace: default
spec:
  ports:
  - name: web
    port: 80
    protocol: TCP
    targetPort: 80
  selector:
    run: nginx
  sessionAffinity: None
  type: LoadBalancer
```

Note:

You must set three annotations.

Create an HTTPS type Server Load Balancer instance

Create a certificate in the Alibaba Cloud console and record the cert-id. Then, use the following annotation to create an HTTPS type Server Load Balancer instance.

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.beta.kubernetes.io/alicloud-loadbalancer-
cert-id: your-cert-id
    service.beta.kubernetes.io/alicloud-loadbalancer-
protocol-port: "https:443"
  labels:
    run: nginx
  name: nginx
  namespace: default
spec:
  ports:
  - name: web
    port: 443
    protocol: TCP
    targetPort: 443
  selector:
    run: nginx
  sessionAffinity: None
```

```
    type :  LoadBalanc  er
```

Note:

Annotations are case sensitive.

Annotation	Description	Default value
service.beta.kubernetes.io/alicloud-loadbalancer-protocol-port	Use commas (,) to separate multiple values. For example, https:443,http:80.	None
service.beta.kubernetes.io/alicloud-loadbalancer-address-type	The value is Internet or intranet.	Internet
service.beta.kubernetes.io/alicloud-loadbalancer-slb-network-type	Server Load Balancer network type. The value is classic or VPC.	Classic
service.beta.kubernetes.io/alicloud-loadbalancer-charge-type	The value is paybytraffic or paybybandwidth.	paybybandwidth
service.beta.kubernetes.io/alicloud-loadbalancer-id	The Server Load Balancer instance ID. Specify an existing Server Load Balance with the loadbalancer-id, and the existing listener is overwritten. Server Load Balancer is not deleted when the service is deleted.	None
service.beta.kubernetes.io/alicloud-loadbalancer-backend-label	Use label to specify which nodes are mounted to the Server Load Balancer backend.	None
service.beta.kubernetes.io/alicloud-loadbalancer-region	The region in which Server Load Balancer resides.	None
service.beta.kubernetes.io/alicloud-loadbalancer-bandwidth	Server Load Balancer bandwidth.	50

Annotation	Description	Default value
service.beta.kubernetes.io/alicloud-loadbalancer-cert-id	ID of a certificate on Alibaba Cloud. You must have uploaded a certificate first.	“”
service.beta.kubernetes.io/alicloud-loadbalancer-health-check-flag	The value is on or off.	The default value is off. No need to modify the TCP parameters because TCP enables health check by default and you cannot configure it.
service.beta.kubernetes.io/alicloud-loadbalancer-health-check-type	See HealthCheck.	None
service.beta.kubernetes.io/alicloud-loadbalancer-health-check-uri	See HealthCheck.	None
service.beta.kubernetes.io/alicloud-loadbalancer-health-check-connect-port	See HealthCheck.	None
service.beta.kubernetes.io/alicloud-loadbalancer-healthy-threshold	See HealthCheck.	None
service.beta.kubernetes.io/alicloud-loadbalancer-unhealthy-threshold	See HealthCheck.	None
service.beta.kubernetes.io/alicloud-loadbalancer-health-check-interval	See HealthCheck.	None
service.beta.kubernetes.io/alicloud-loadbalancer-health-check-connect-timeout	See HealthCheck.	None
service.beta.kubernetes.io/alicloud-loadbalancer-health-check-timeout	See HealthCheck.	None

6.2 Use Ingress to provide Layer-7 service access

In the Alibaba Cloud serverless Kubernetes cluster, Server Load Balancer provides Layer-4 service access. You can also use Layer-7 service access provided by Ingress. This document describes how to provide Layer-7 domain name service access in the serverless Kubernetes cluster.

Prerequisites

- You have created a Serverless cluster. VPC cluster must be configured with a NAT gateway to access the external network and download the container image.
- You have connected to the cluster by using kubectl, see #unique_33.

Instructions

- If Server Load Balancer is not specified, system automatically generates a public network Server Load Balancer instance.
- The default front-end listening ports for SLB instances are 80 (HTTP Protocol) and 443 (HTTPS protocol).
- By default, the HTTPS certificate of the SLB instance is initialized for the first created Ingress-configured TLS certificate. Otherwise, the system default certificat e is initialized. You can modify it in the SLB console as needed.
- When you specify to use an existing SLB instance, SLB instance specification must be of performance guarantee type (supports ENI). Also, make sure that ports 80 and 443 are not currently used by other services.

Use the default generated SLB instance

If an SLB instance is not specified, the system automatically generates a performance guaranteed public network SLB instance when the first Ingress is created.

1. Deploy test services.

   a. Create a file `cafe - service . yaml` , copy the following code to the file, and then run the `kubectl apply - f cafe - service . yaml` command to deploy a coffee service and tea service.

```
apiVersion : extensions / v1beta1
kind : Deployment
metadata :
  name : coffee
spec :
  replicas : 2
  selector :
```

```
    matchLabel  s :
      app :  coffee
  template :
    metadata :
      labels :
        app :  coffee
    spec :
      containers :
     -  name :  coffee
        image :  registry . cn - hangzhou . aliyuncs . com / acs
 - sample / nginxdemos : latest
        ports :
       -  containerP  ort :  80
---
 apiVersion :  v1
 kind :  Service
 metadata :
   name :  coffee - svc
 spec :
   ports :
  -  port :  80
     targetPort :  80
     protocol :  TCP
   selector :
     app :  coffee
   clusterIP :  None
---
 apiVersion :  extensions / v1beta1
 kind :  Deployment
 metadata :
   name :  tea
 spec :
   replicas :  1
   selector :
     matchLabel  s :
       app :  tea
   template :
     metadata :
       labels :
         app :  tea
     spec :
       containers :
      -  name :  tea
         image :  registry . cn - hangzhou . aliyuncs . com / acs
 - sample / nginxdemos : latest
         ports :
       -  containerP  ort :  80
---
 apiVersion :  v1
 kind :  Service
 metadata :
   name :  tea - svc
   labels :
 spec :
   ports :
  -  port :  80
     targetPort :  80
     protocol :  TCP
   selector :
     app :  tea
```

```
  clusterIP : None
```

The following outputs indicate the services are deployed:

```
deployment " coffee " created
service " coffee - svc " created
deployment " tea " created
service " tea - svc " created
```

b. Run the `kubectl get svc , deploy` command to view the status of the services.

```
NAME                TYPE          CLUSTER - IP     EXTERNAL - IP
  PORT ( S )    AGE
svc / coffee - svc     ClusterIP    < none >       < none >
  80 / TCP      1m
svc / tea - svc        ClusterIP    < none >       < none >
  80 / TCP      1m

NAME               DESIRED     CURRENT     UP - TO - DATE
AVAILABLE     AGE
deploy / coffee     2           2           2              2
       1m
deploy / tea        1           1           1              1
       1m
```

2. Configure an Ingress.

a. Create a file `cafe - ingress . yaml` , copy the following code into the file, and run the `kubectl apply - f cafe - ingress . yaml` command.

```
apiVersion : extensions / v1beta1
kind : Ingress
metadata :
  name : cafe - ingress
spec :
  rules :
 # Set a Layer - 7 domain name .
 - host : foo . bar . com
   http :
     paths :
    # Set the context path .
    - path : / tea
      backend :
        serviceNam e : tea - svc
        servicePor t : 80
    # Set the context path .
    - path : / coffee
      backend :
        serviceNam e : coffee - svc
```

```
            servicePor  t :  80
```

The following output indicates that the Ingress is deployed.

```
ingress  " cafe - ingress "  created
```

b. Run the `kubectl  get  ing` command to obtain the IP address of the SLB instance.

```
NAME              HOSTS            ADDRESS             PORTS
AGE
cafe - ingress      foo . bar . com      139 .***.**.***     80
    1m
```

3. Test service access.

> **Note:**
> The domain name of the SLB instance IP must be resolved manually.

In this example, a DNS domain name resolution rule is added to *hosts* for testing service access. We recommend that you enter the domain name in your work environment.

```
139 .***.**.***      foo . bar . com
```

- Use your browser to access the coffee service.

- Run the following command to access the coffee service:

```
curl  - H  " Host :  foo . bar . com "  http :// 139 .***.**.***/
coffee
```

- Use your browser to access the coffee service.

- Run the following command to access the tea service:

```
curl  - H  " Host :  foo . bar . com "  http :// 139 .***.**.***/
tea
```

Use the specified SLB instance

You can specify to use of an existing SLB instance by using the `service . beta . kubernetes . io / alicloud - loadbalanc  er - id` annotation, but the SLB instance specification must be of performance guarantee type (supports ENI).

> **Note:**

System automatically initializes ports 80 and 443 of the SLB instance, make sure that ports are not currently used by other services.

1. Deploy test services.

   a. Create a file `tomcat-service.yml`, copy the following code into the file, and run the `kubectl apply -f tomcat-service.yml` command to deploy a Tomcat application.

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: tomcat
spec:
  replicas: 1
  selector:
    matchLabels:
      run: tomcat
  template:
    metadata:
      labels:
        run: tomcat
    spec:
      containers:
      - image: tomcat:7.0
        imagePullPolicy: Always
        name: tomcat
        ports:
        - containerPort: 8080
          protocol: TCP
      restartPolicy: Always
---
apiVersion: v1
kind: Service
metadata:
  name: tomcat
spec:
  ports:
  - port: 8080
    protocol: TCP
    targetPort: 8080
  selector:
    run: tomcat
  clusterIP: None
```

   The following outputs indicates the Tomcat application is deployed:

```
deployment "tomcat" created
service "tomcat" created
```

   b. Run the `kubectl get svc,deploy tomcat` command to view the application status.

```
NAME          TYPE        CLUSTER-IP     EXTERNAL-IP
PORT(S)       AGE
svc/tomcat    ClusterIP   <none>         <none>
8080/TCP      1m
```

```
NAME              DESIRED     CURRENT     UP - TO - DATE
AVAILABLE     AGE
deploy / tomcat     1           1           1              1
       1m
```

2. Apply for an SLB instance.

   You must apply for a performance guarantee type SLB instance (such as slb.s2.small) under the cluster and region. According to the specific needs, private or public network can be used. For more information, see #unique_34. In this example, apply for an Internet SLB instance and record the ID of the SLB instance.

3. Configure an TLS certificate.

   You must configure the TLS certificate for HTTPS access.

   a. Run the following commands to generate an TLS certificate:

```
$ openssl req - x509 - nodes - days 365 - newkey rsa
 : 2048 - keyout tls . key - out tls . crt - subj "/ CN =
 bar . foo . com / O = bar . foo . com "
$ kubectl create secret tls cert - example -- key tls
 . key -- cert tls . crt

 secret " cert - example " created
```

   b. Run the following command to view the created TLS certificate:

```
$ kubectl get secret cert - example

 NAME              TYPE                  DATA        AGE
 cert - example     kubernetes . io / tls     2           12s
```

> **Note:**
> System automatically initializes the SLB HTTPS default certificate according to the first created Ingress TLS certificate. If you want to modify the HTTPS default certificate, you can modify it in the SLB console. If you want to configure multiple certificates, you can manually add them in the SLB console HTTPS listener domain name extension.

4. Configure an Ingress.

   a. Create a file `tomcat - ingress . yml` , copy the following command to the file, and run the `kubectl apply - f tomcat - ingress . yml` command.

```
apiVersion : extensions / v1beta1
kind : Ingress
```

```
metadata :
  name :  tomcat - ingress
  annotation  s :
   #  Specify   an   SLB   instance .
    service . beta . kubernetes . io / alicloud - loadbalanc  er
- id :  lb - xxxxxxxxxx             ## Set   the   ID   of   the
  target   SLB   instance .
    service . beta . kubernetes . io / alicloud - loadbalanc  er
- force - override - listeners : " true "
spec :
  tls :
 -  hosts :
   -  bar . foo . com
   #  Set   a   TLS   certificat  e .
    secretName :  cert - example
  rules :
 #  Set   a   Layer - 7   domain   name .
 -  host :  bar . foo . com
    http :
      paths :
     #  Set   the   context   path .
     -  path : /
        backend :
          serviceNam  e :  tomcat
          servicePor  t :  8080
```

The following output indicates the tomcat-ingress is deployed:

```
ingress  " tomcat - ingress "  created
```

b. Run the `kubectl get ing tomcat - ingress` command to obtain the IP address of the SLB instance.

```
NAME                HOSTS           ADDRESS          PORTS
AGE
tomcat - ingress     bar . foo . com     47 .***.**.**    80 ,
443     1m
```

5. Test service access.

Note:

Currently, the domain name of the SLB instance IP must be resolved manually.

In this example, a DNS domain name resolution rule is added to `hosts` for testing service access. We recommend that you enter the domain name in your work environment.

```
47 .***.**.**    bar . foo . com
```

- Use you browser to access the Tomcat service.

- Run the following command to access the Tomcat service:

```
curl  - k  - H  " Host :  bar . foo . com "  https :// 47
.***.**.**
```

7 Log management

7.1 Overview

You can view the logs of the Serverless Kubernetes cluster in the following ways.

- View the container running logs by using kubectl logs command.

  For more information, see kubectl logs.

  > Note:
  > Before using the kubectl logs command to view the container running logs, see #unique_13.

- #unique_37

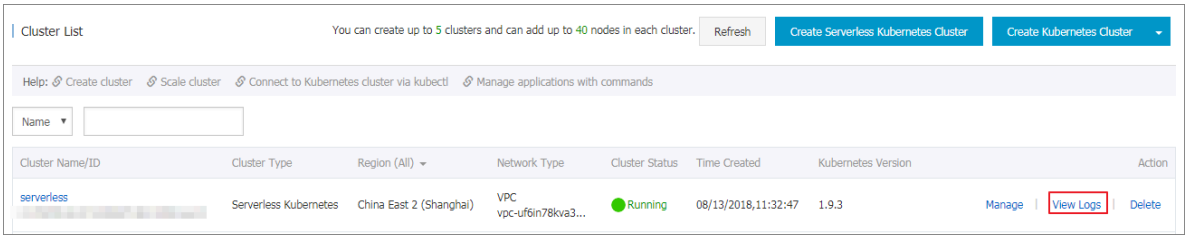
7.2 View cluster logs

You can view the cluster operation logs by using the simple log service of Container Service.

Procedure

1. Log on to the Container Service console.
2. Under Kubernetes, click Clusters in the left-side navigation pane.

3. Select the target cluster and click View Logs on the right.



View the cluster operation information.