Alibaba Cloud Container Service

Best Practices

Issue: 20190815

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 Comparison between Swarm and Kubernetes cluster functions

1.1 Overview

This topic describes the prerequisites and limits for function comparisons between a Swarm cluster and a Kubernetes cluster that run in Container Service.

Prerequisites

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

Note:

·

- Alibaba Cloud Container Service for Kubernetes supports the following clusters: the dedicated Kubernetes cluster, the managed Kubernetes cluster, the multi-zone Kubernetes cluster, and the serverless Kubernetes cluster (in beta).
- The topic uses creating a Kubernetes cluster as an example to compare the functions between a Swarm and a Kubernetes cluster that run on Container Service.

Limits

- The applications used for the function comparison are as follows:
  - Stateless applications
  - Applications that use a data base or a storage device to store data

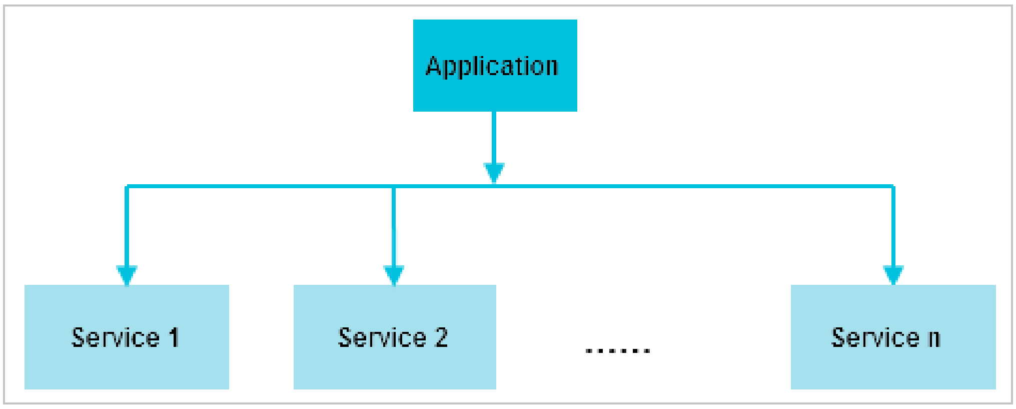
1.2 Basic terms

This topic compares the basic terms that are used for both Swarm clusters and Kubernetes clusters.

Application

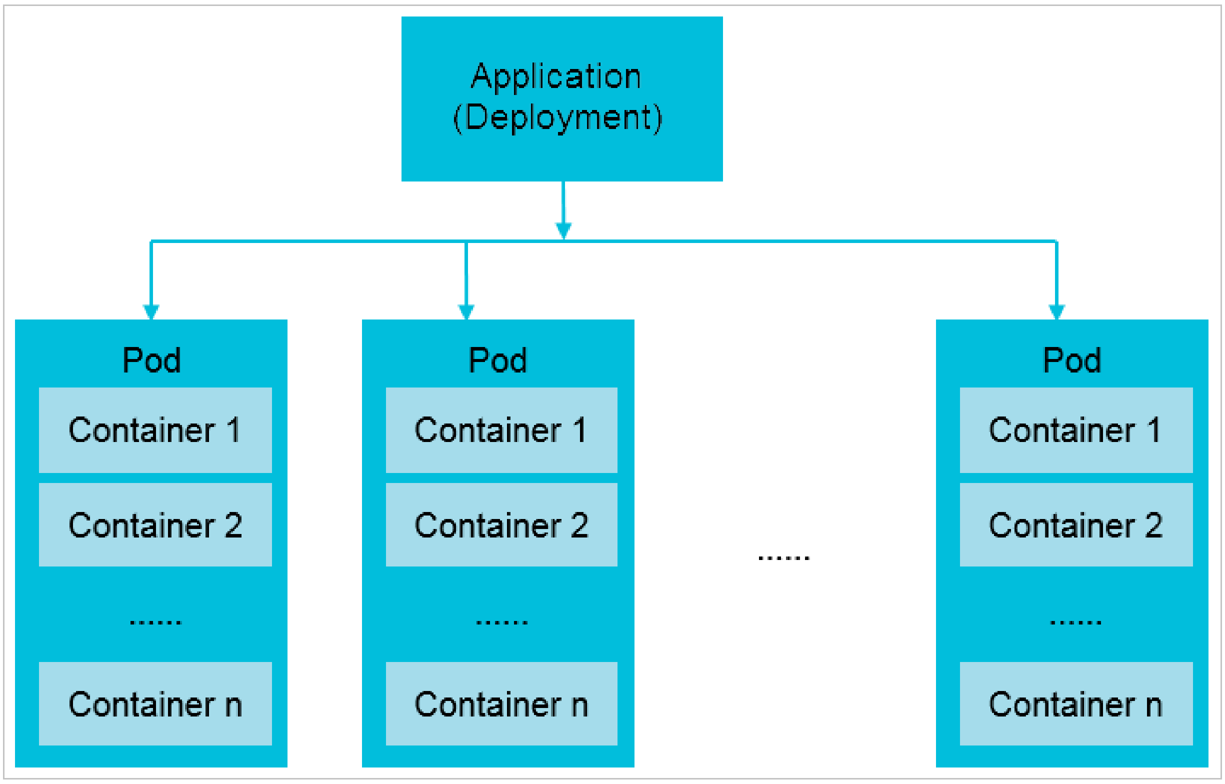
Container Service Swarm clusters

In a Container Service Swarm cluster, applications can be viewed as projects. Each application can include multiple services. Each service is an instance that provides the specific function. Services can be horizontally expanded.



Container Service Kubernetes clusters

In a Container Service Kubernetes cluster, an application, also known as a deployment, is used to provide functions. A deployment contains pods and containers. A pod is the minimum resource unit that can be scheduled in Kubernetes and each pod can contain multiple containers. A pod can be viewed as an instance of the application to which the pod belongs. Multiple pods can be scheduled to different nodes. This means that pods can be horizontally expanded.



Note:

The preceding figure in which each pod has multiple containers is used to show the expansion capability of pods. However, we recommend that you set only one container for each pod.

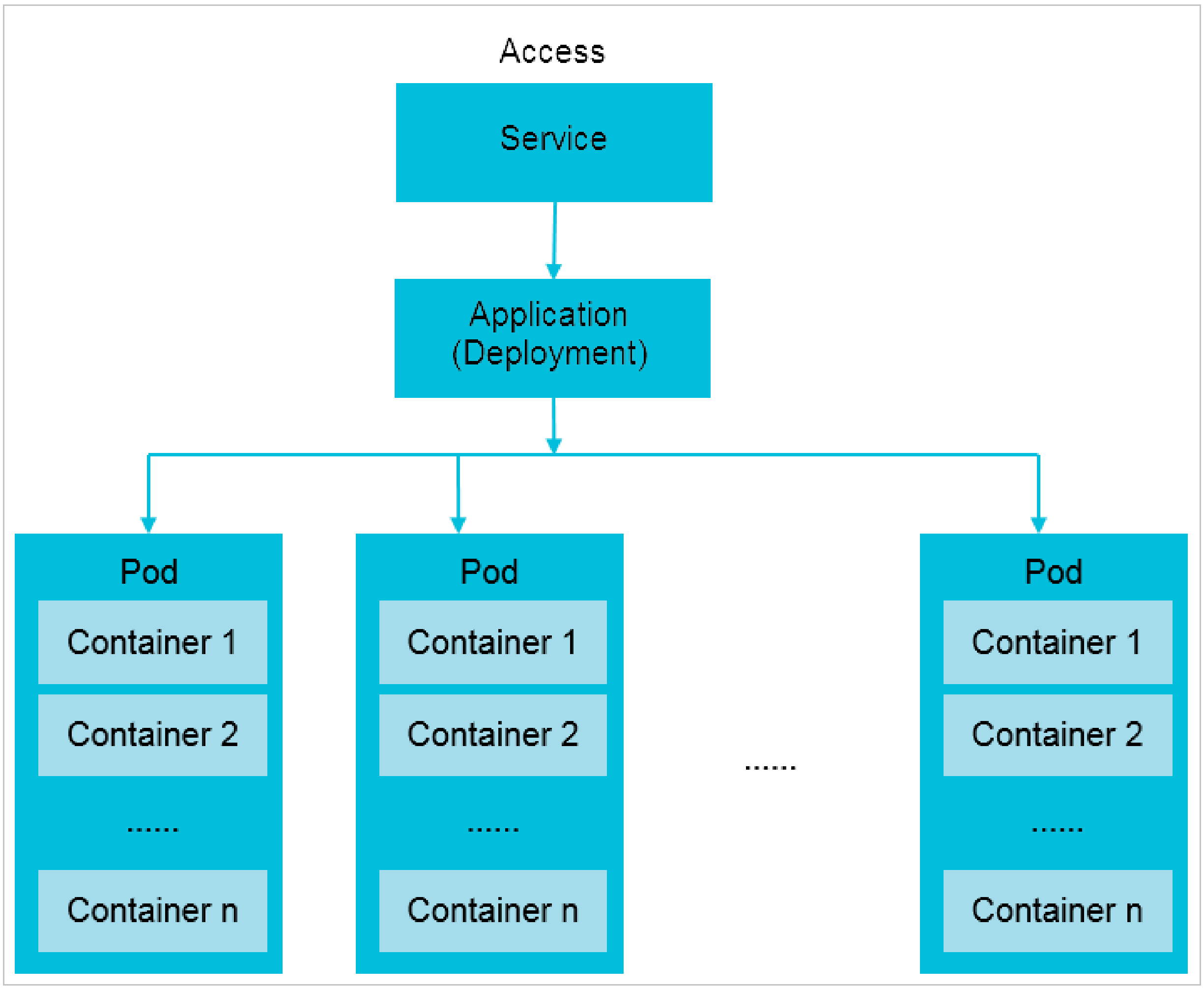
Service

Container Service Swarm clusters

Each service in a Container Service Swarm cluster is an instance that provides a specific function. When you create an application in a Swarm cluster, the access method of the service is exposed directly outside the cluster.

Container Service Kubernetes clusters

The service term in Container Service Kubernetes clusters is an abstract concept. A service can expose the access method of its application (or deployment) outside the cluster.



Application access

Container Service Swarm clusters

When you deploy an application in a Container Service Swarm cluster, you can select one from three types of application access methods that can directly expose the application. The three types of application access methods are:

- <HostIP>:<port>
- Simple routing
- Server Load Balancer (SLB)

Container Service Kubernetes clusters

After you create an application in a Container Service Kubernetes cluster, you must create a service to expose the access method of the application. Then the application becomes accessible. Applications within a Container Service Kubernetes cluster can then access each other through their service names. Service names are only applicable to the access within the cluster. To access the application from outside the cluster, you need to create a service of the NodePort type or a service of the LoadBalancer type to expose the application.

- ClusterIP (It has the same function as a service name. That is, it is applicable to accesses within a cluster.)
- NodePort (It can be viewed as <HostIP>:<port> of Swarm clusters.)
- LoadBalancer (It can be viewed as the SLB of Swarm clusters.)
- Domain name implemented by creating an Ingress (It can be viewed as the simple routing of Swarm clusters.)

1.3 General settings for creating an application through an image

This topic compares the general settings used in a Swarm cluster and those used in a Kubernetes cluster for creating an application through an image.

Create an application by using an image

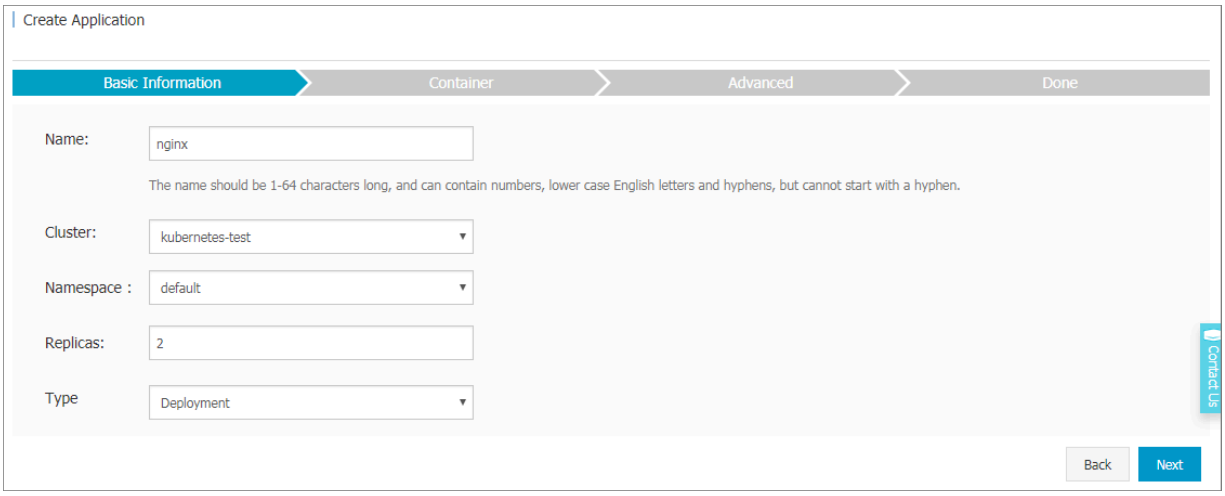
If you create an application in the Container Service console by using an image, the Swarm cluster Web interface is different from the Kubernetes cluster Web interface.

- For more information about the Web interface of a Swarm cluster, see Create an application.
- For more information about the Web interface of a Kubernetes cluster, see Create a deployment application by using an image.

Basic information

Container Service Swarm clusters

The basic information for creating an application in a Swarm cluster includes the application name, application version, deployment cluster, default update policy, and application description.



Create Application

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: kubernetes-test

Namespace : default

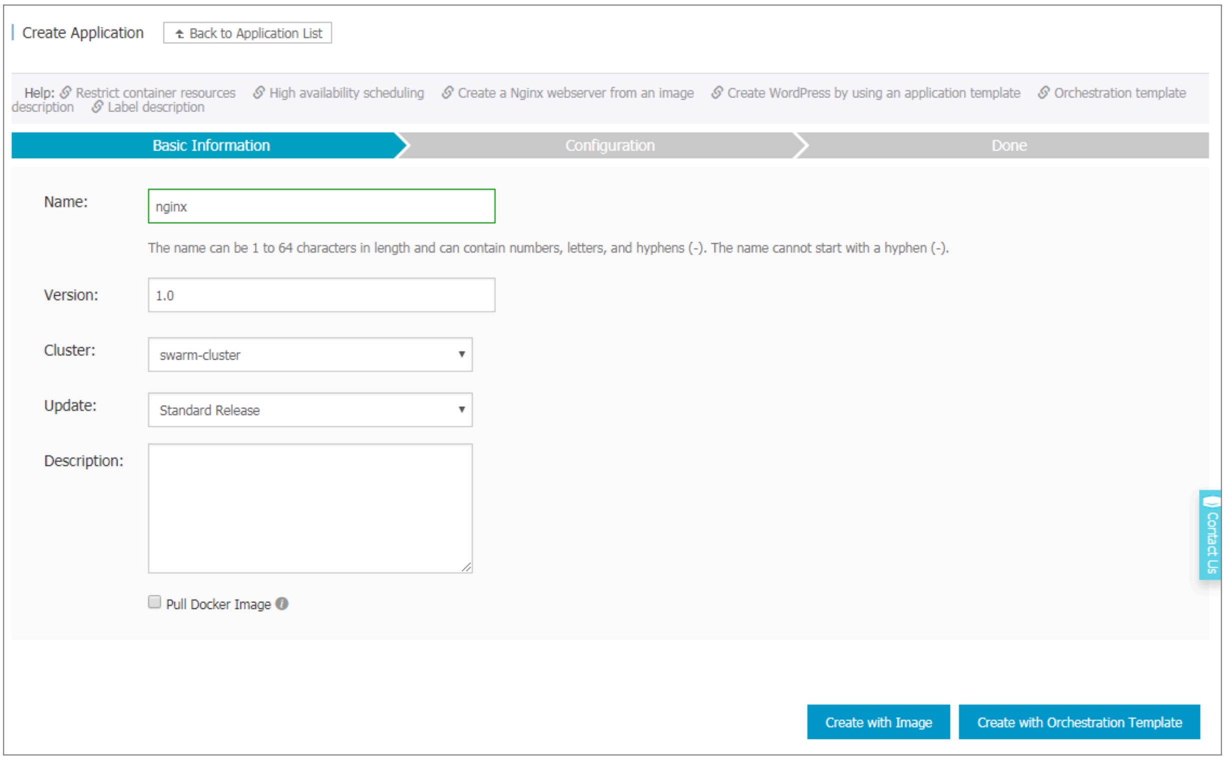
Replicas: 2

Type Deployment

Container Service Kubernetes clusters

The basic information for creating an application in a Kubernetes cluster includes the application name, application version, deployment cluster, namespace, number of replicas, and application type.

The namespace term is exclusive to Kubernetes clusters. Kubernetes uses namespaces to isolate resources such as CPU and memory. In addition, namespaces can be used to separate different environments such as test and development environments. We recommend that you use clusters to isolate production environments. For information about the namespace term, see Terms.



Create Application

Help: Restrict container resources High availability scheduling Create a Nginx webserver from an image Create WordPress by using an application template Orchestration template description Label description

Name: nginx

The name can be 1 to 64 characters in length and can contain numbers, letters, and hyphens (-). The name cannot start with a hyphen (-).

Version: 1.0

Cluster: swarm-cluster

Update: Standard Release

Description:

□ Pull Docker Image

General settings

The image name and image version settings are the most important.

Container Service Swarm clusters

The Network Mode supportsDefault and host.

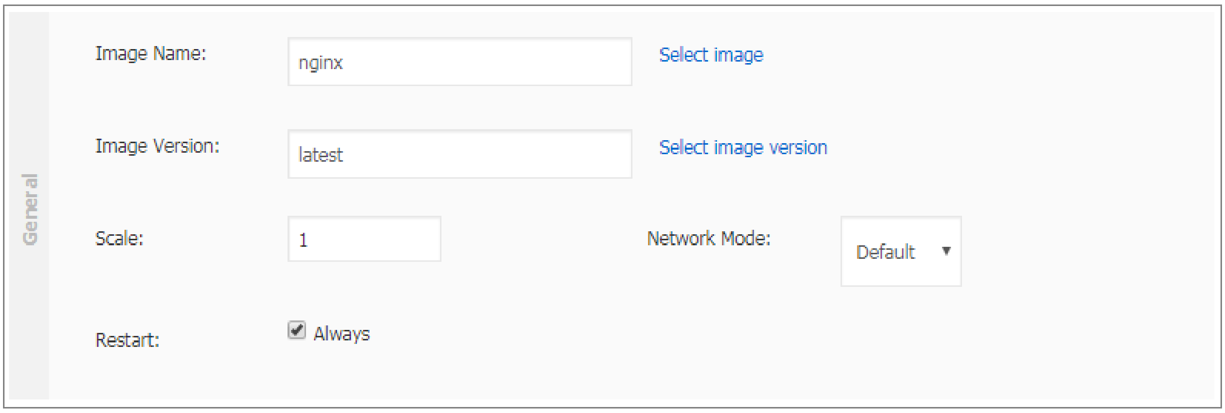


Image Name: nginx

Image Version: latest

Scale: 1

Network Mode: Default

Restart: ☑ Always

Container Service Kubernetes clusters

- The network mode of the application has been specified when you create the cluster. Available network plugins include Flannel and Terway. For more information, see Do I select the Terway or Flannel plugin for my Kubernetes cluster network?.
- Required resources include the CPU and memory resources required by the application. The resource limits are the upper thresholds of the resources quota.

You can compare the settings with the CPU Limit and Memory Limit settings of the Container settings in a Swarm cluster.

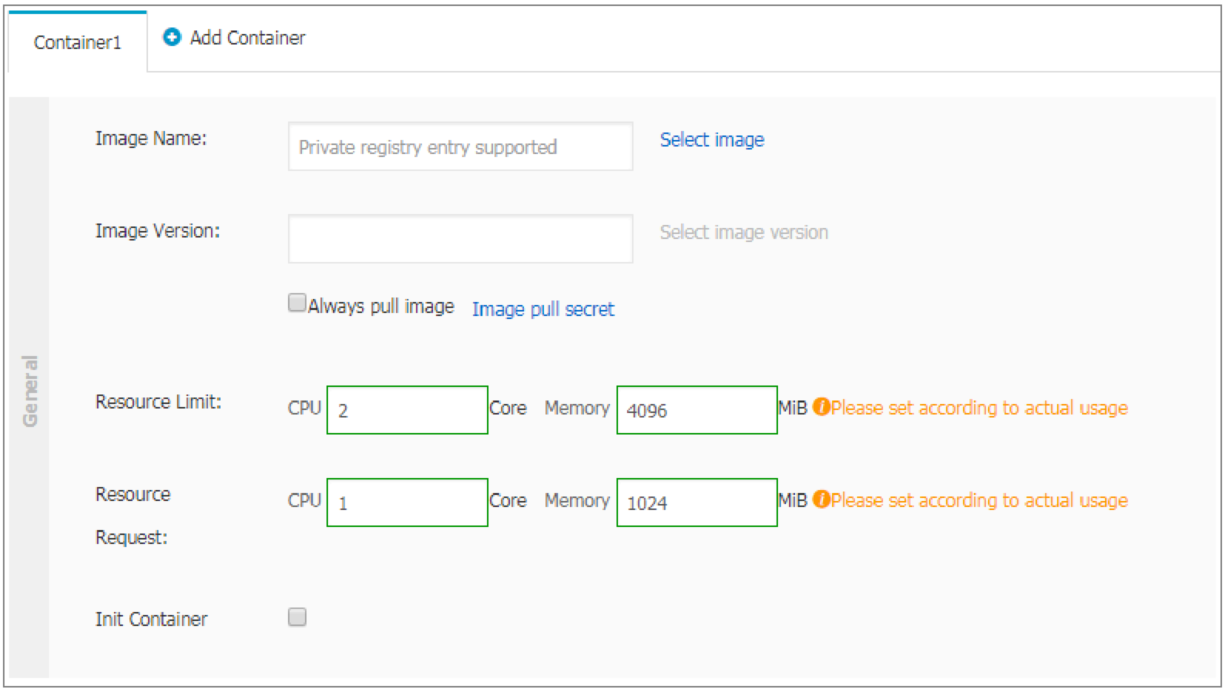


Image Name: Private registry entry supported

Image Version:

□ Always pull image

Resource Limit: CPU 2 Core Memory 4096 MiB Please set according to actual usage

Resource Request: CPU 1 Core Memory 1024 MiB Please set according to actual usage

Init Container □

1.4 Network settings used for creating an application through an image

This topic compares the network settings used in a Swarm cluster with those used in a Kubernetes cluster for creating an application through an image.

Create an application by using an image

If you create an application in the Container Service console by using an image, the Swarm cluster Web interface is different from the Kubernetes cluster Web interface.

- For more information about the Web interface of a Swarm cluster, see Create an application.
- For more information about the Web interface of a Kubernetes cluster, see Create a deployment application by using an image.

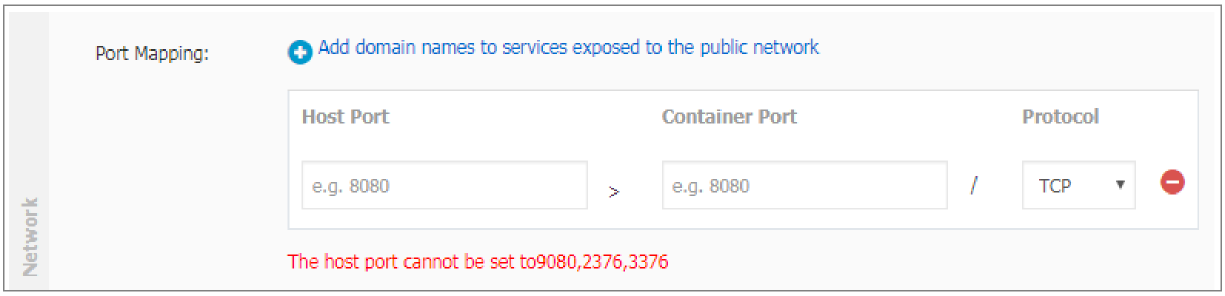
Network configuration

The Network Configuration of a Swarm cluster is used to expose the access methods outside the cluster for an application.

Configure port mapping

Container Service Swarm clusters

With the Port Mapping function of a Swarm cluster, you can map the application port to a host so that each host actives the same port. Then the application can be accessed through `< HostIP >:< Port >`.



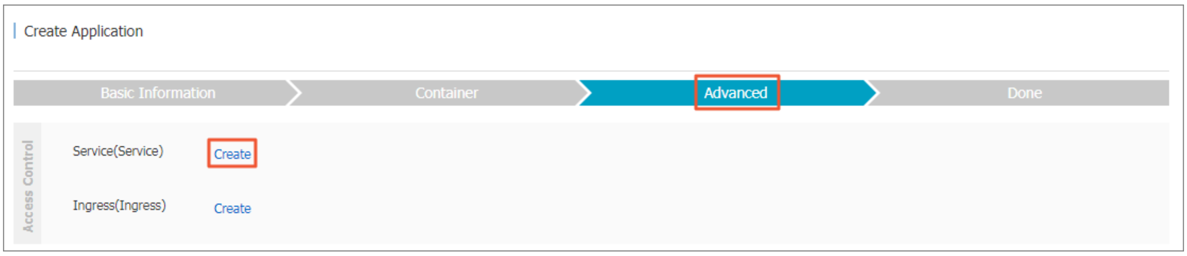
The host port cannot be set to9080,2376,3376

Container Service Kubernetes clusters

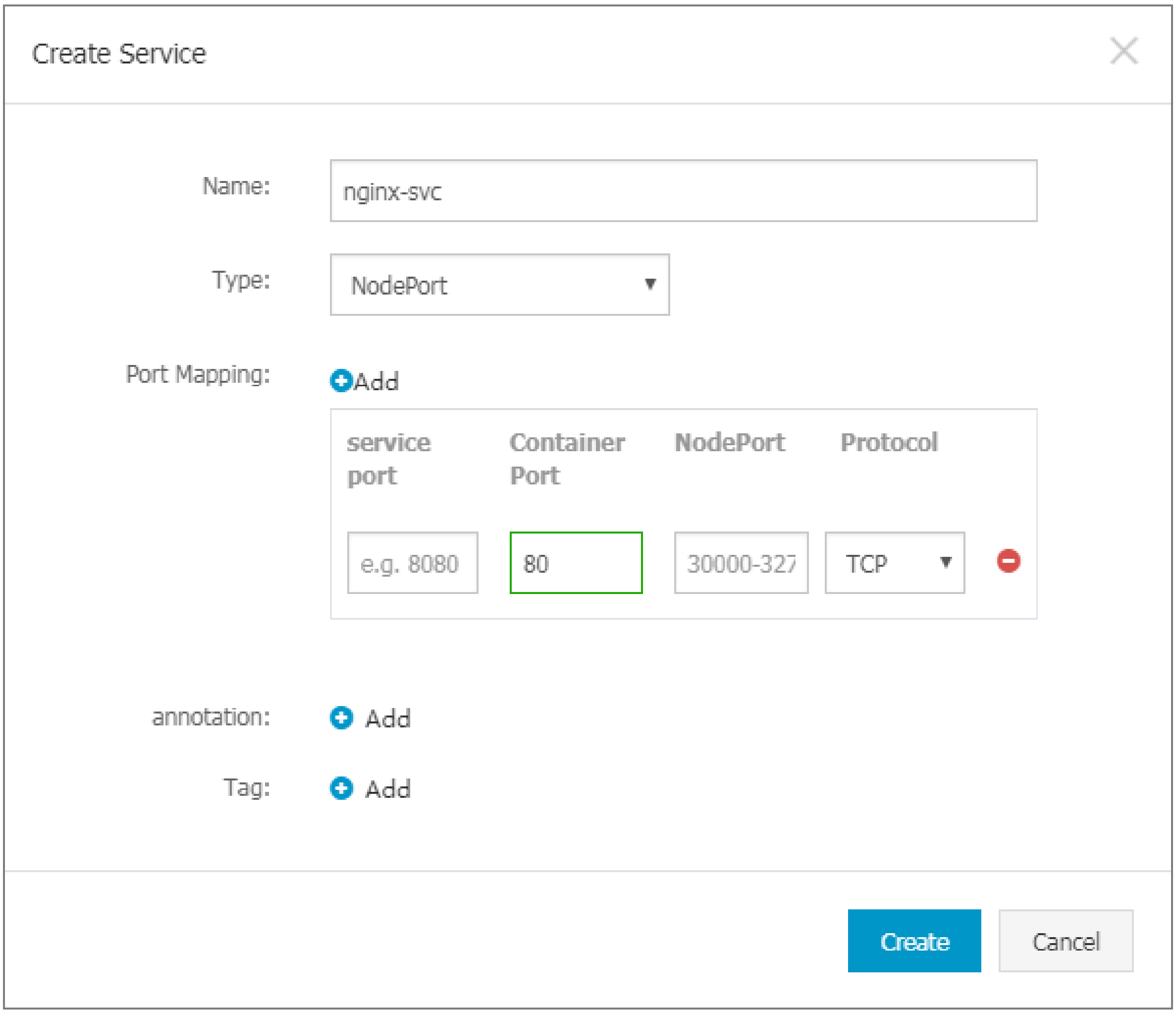
To implement the port mapping function in a Kubernetes cluster, you can create a NodePort type service by using either of the following two methods:

Method 1: Configure port mapping when creating an application

1. After you complete the Container setting, configure the Advanced setting. Specifically, click Create on the right of Service in the Access Control area.



2. Select the NodePort Type. For more information, see Create a deployment application by using an image.



Create Service

Name: nginx-svc

Type: NodePort

Port Mapping: Add

service port	Container Port	NodePort	Protocol
e.g. 8080	80	30000-327	TCP

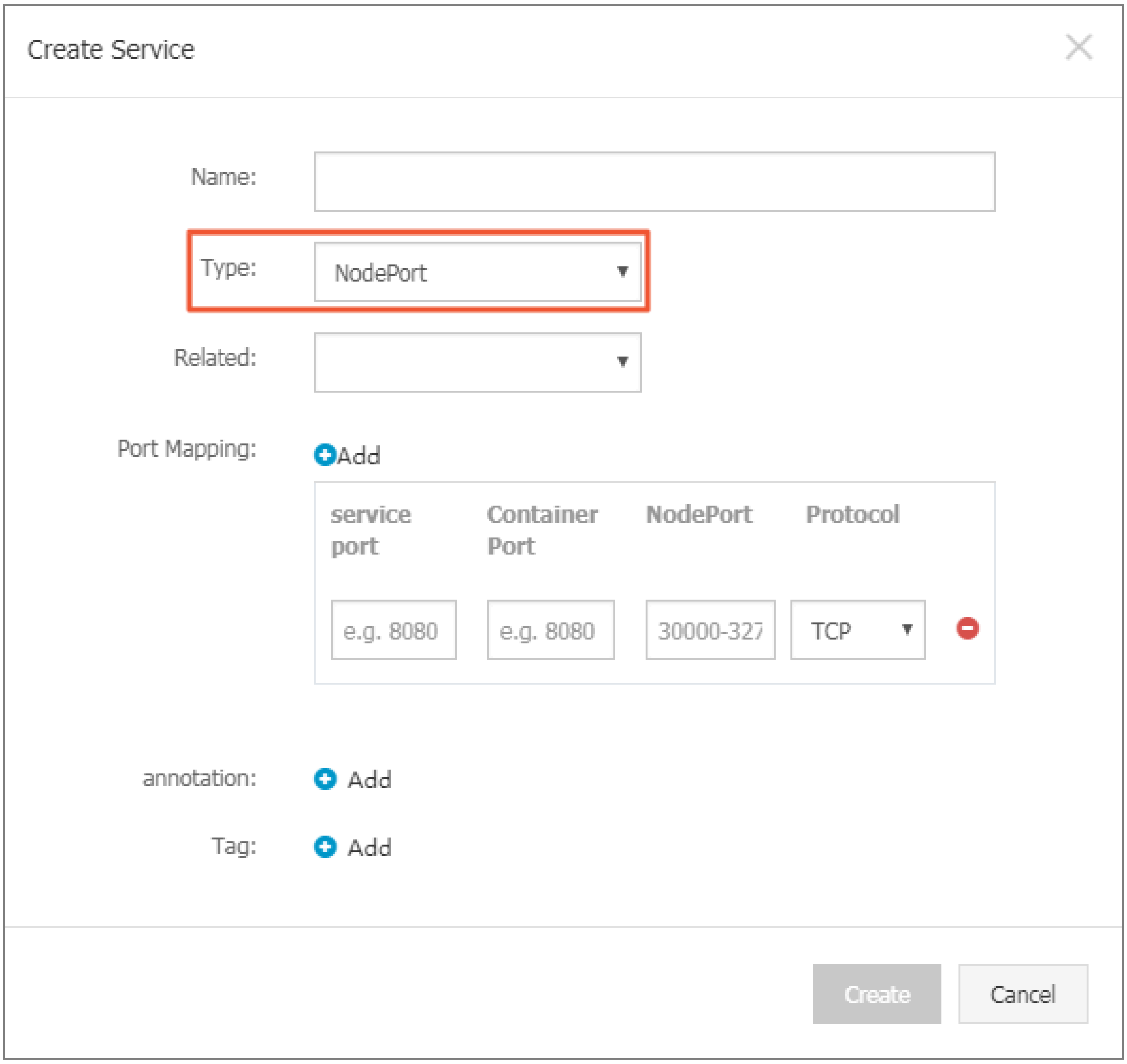
annotation: Add

Tag: Add

Method 2: Configure port mapping when creating a service

1. In the left-side navigation pane in the Container Service console, choose Discovery and Load Balancing > Service.

2. Select the target cluster and namespace, and click Create. In the Create Service dialog box, select the NodePort Type. For more information, see Create a service.



Create Service

Name:

Type: NodePort

Related:

Port Mapping: Add

service port	Container Port	NodePort	Protocol
e.g. 8080	e.g. 8080	30000-327	TCP

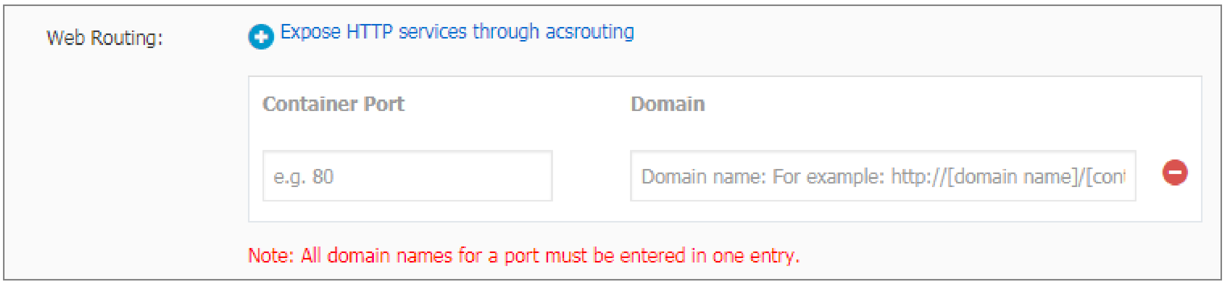
annotation: Add

Tag: Add

Configure simple routing

Container Service Swarm clusters

With the Simple Routing function of a Swarm cluster, you can access an application through a domain name. You can use the domain name provided by Container Service or customize the domain name.



Web Routing: Expose HTTP services through acsrouting

Container Port	Domain
e.g. 80	Domain name: For example: http://[domain name]/[cont

Note: All domain names for a port must be entered in one entry.

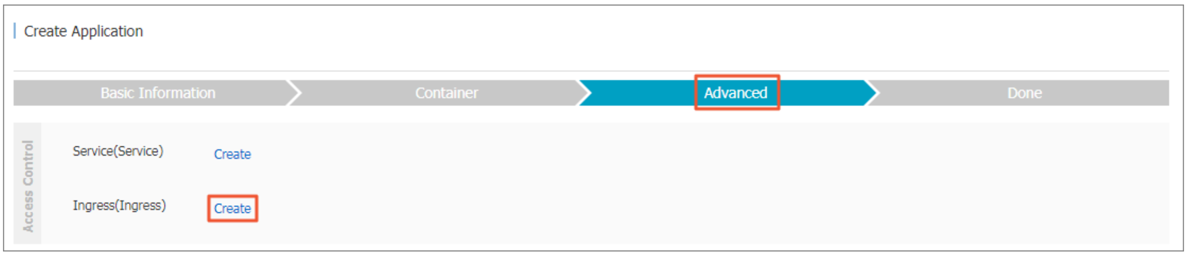
Container Service Kubernetes clusters

In a Kubernetes cluster, you can create an Ingress to implement simple routing. In addition, the Ingress function of Container Service for Kubernetes provides blue/green deployment and gray releases. For more information, see Gray releases and blue/green deployment.

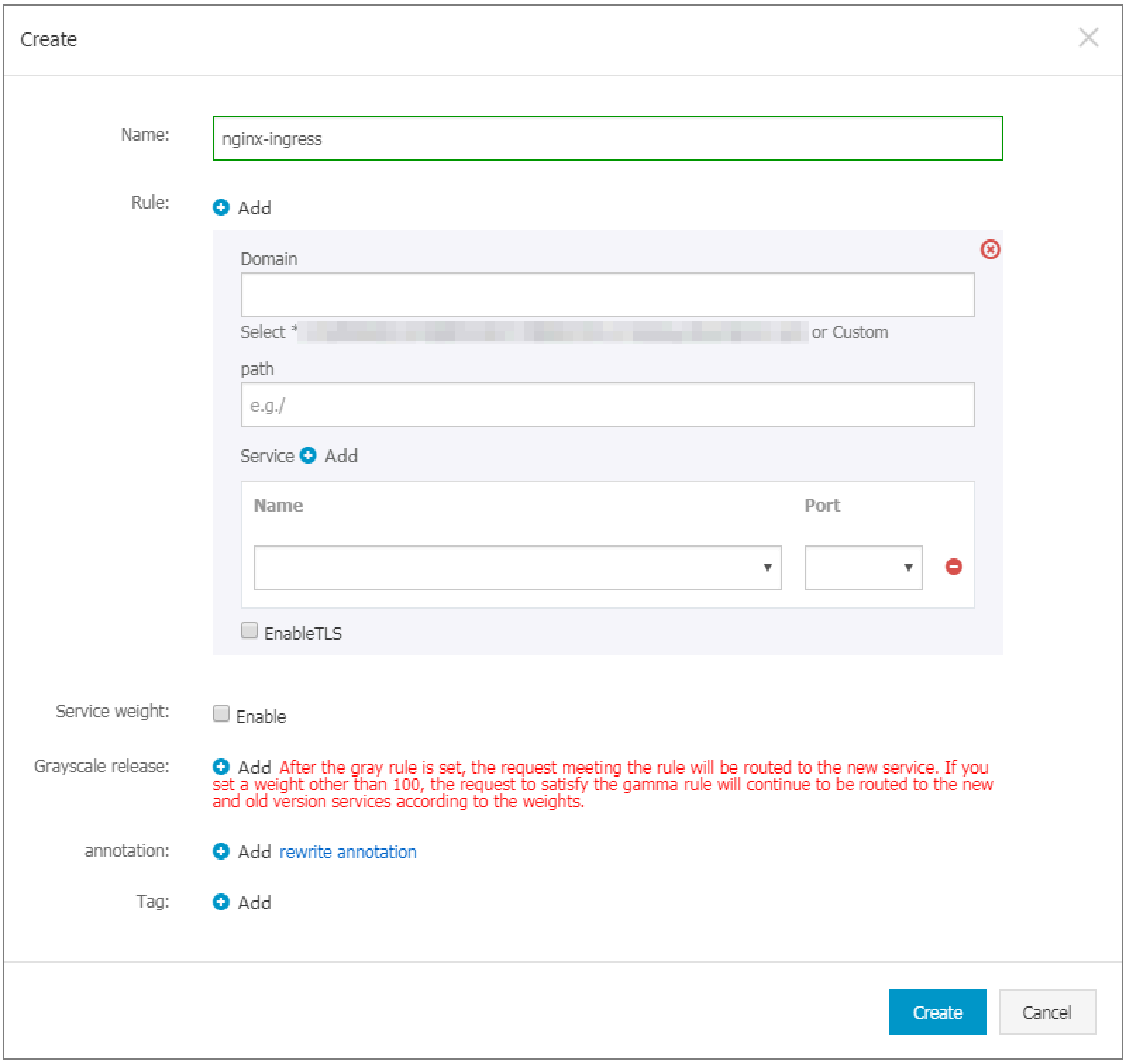
Two methods are available to implement the Ingress function in a Kubernetes cluster.

Method 1: Configure an Ingress when creating an application

1. After you complete the Container setting, configure the Advanced setting. Specifically, click Create on the right of Ingress in the Access Control area.



2. For more information, see Create a deployment application by using an image.



Create

Name: nginx-ingress

Rule: Add

Domain

Select * [redacted] or Custom

path

e.g./

Service Add

Name	Port

□ EnableTLS

Service weight: □ Enable

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

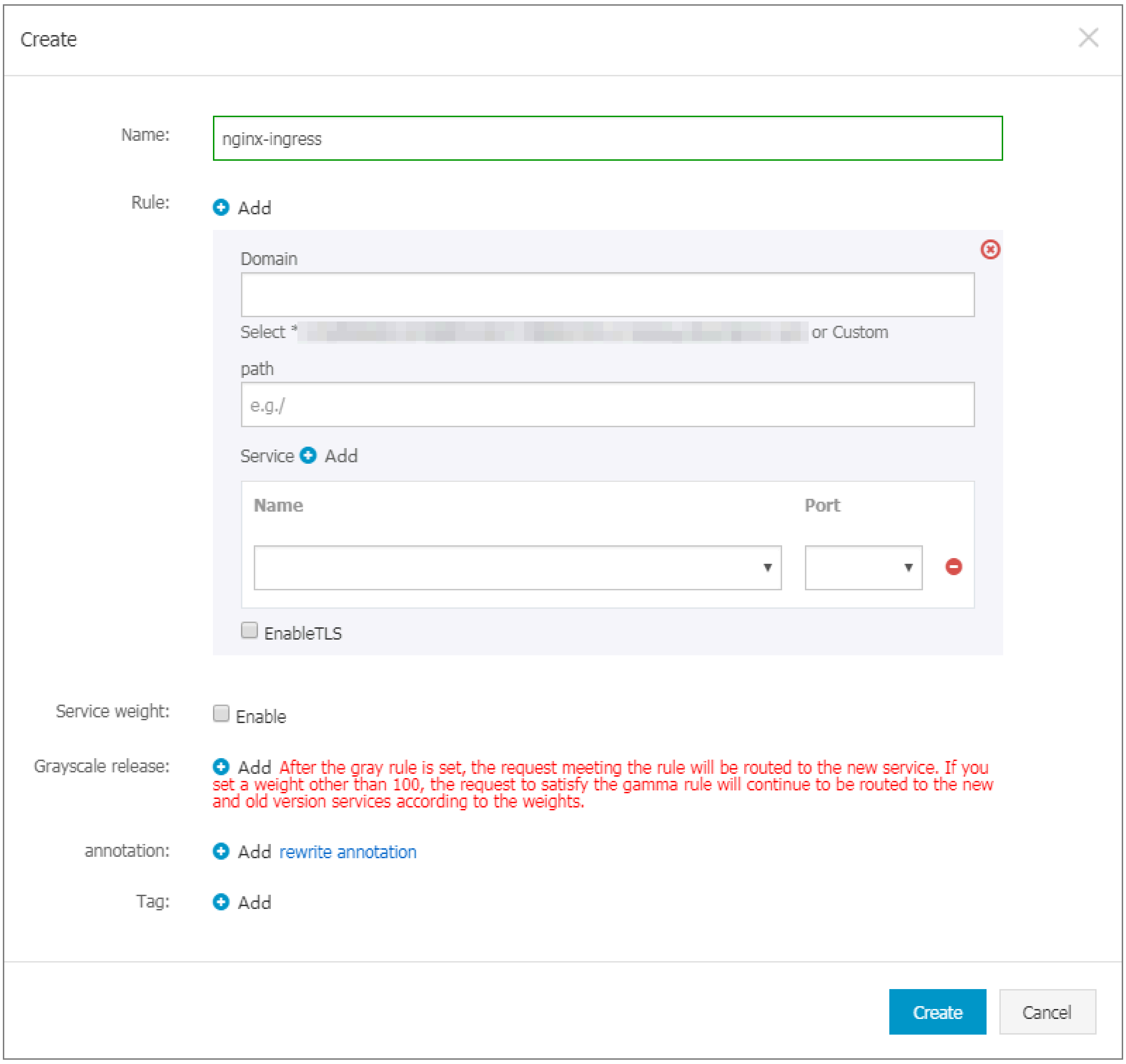
annotation: Add rewrite annotation

Tag: Add

Method 2: Configure an Ingress directly

1. In the left-side navigation pane in the Container Service console, choose Discovery and Load Balancing > Ingress.

2. Select the target cluster and namespace, and click Create. For more information, see Create an Ingress in the Container Service console.



Create

Name: nginx-ingress

Rule: Add

Domain

Select * [redacted] or Custom

path

e.g./

Service Add

Name	Port
▾	▾

□ EnableTLS

Service weight: □ Enable

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

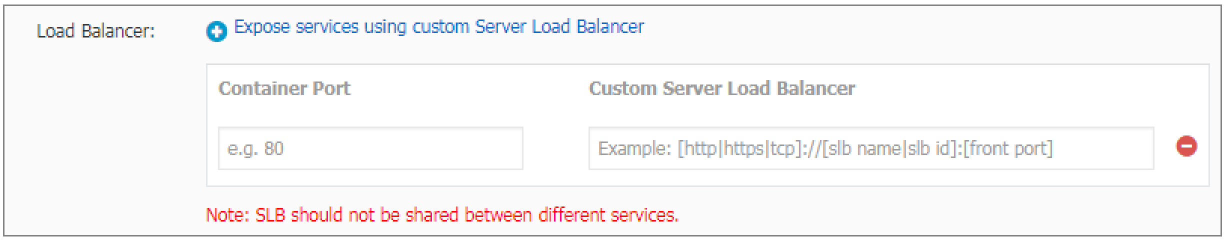
annotation: Add rewrite annotation

Tag: Add

Configure Server Load Balancer

Container Service Swarm clusters

With the Load Balancer function of a Swarm cluster, you can use Alibaba Cloud Server Load Balancer to expose the access method of an application. You must create an SLB and then associate the ID and the port number of the created SLB with the application so that you can access the application through <SLB_IP>:<Port>.



Load Balancer: Expose services using custom Server Load Balancer

Container Port	Custom Server Load Balancer
e.g. 80	Example: [http|https|tcp]://[slb name|slb id]:[front port]

Note: SLB should not be shared between different services.

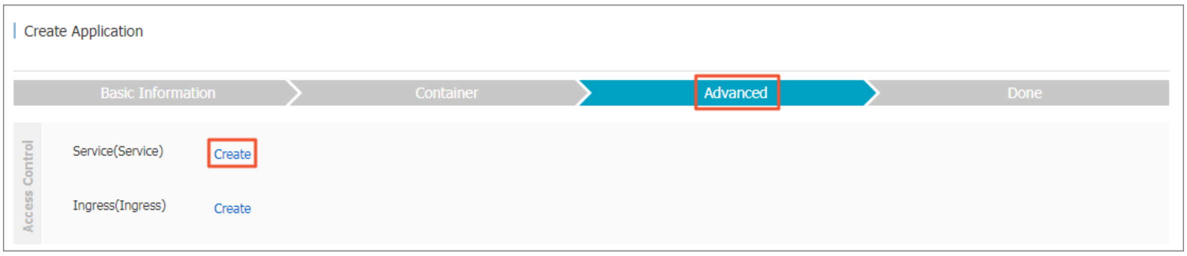
Container Service Kubernetes clusters

In a Kubernetes cluster, you can also expose the access method of an application by associating an SLB with the application. An SLB can be automatically created in a Kubernetes cluster through an SLB service. For SLB access, you can select either Internet access method or internal cluster access method. If you use a YAML file to create an application, you can specify an existing SLB and set session persistence. For more information, see Create a service.

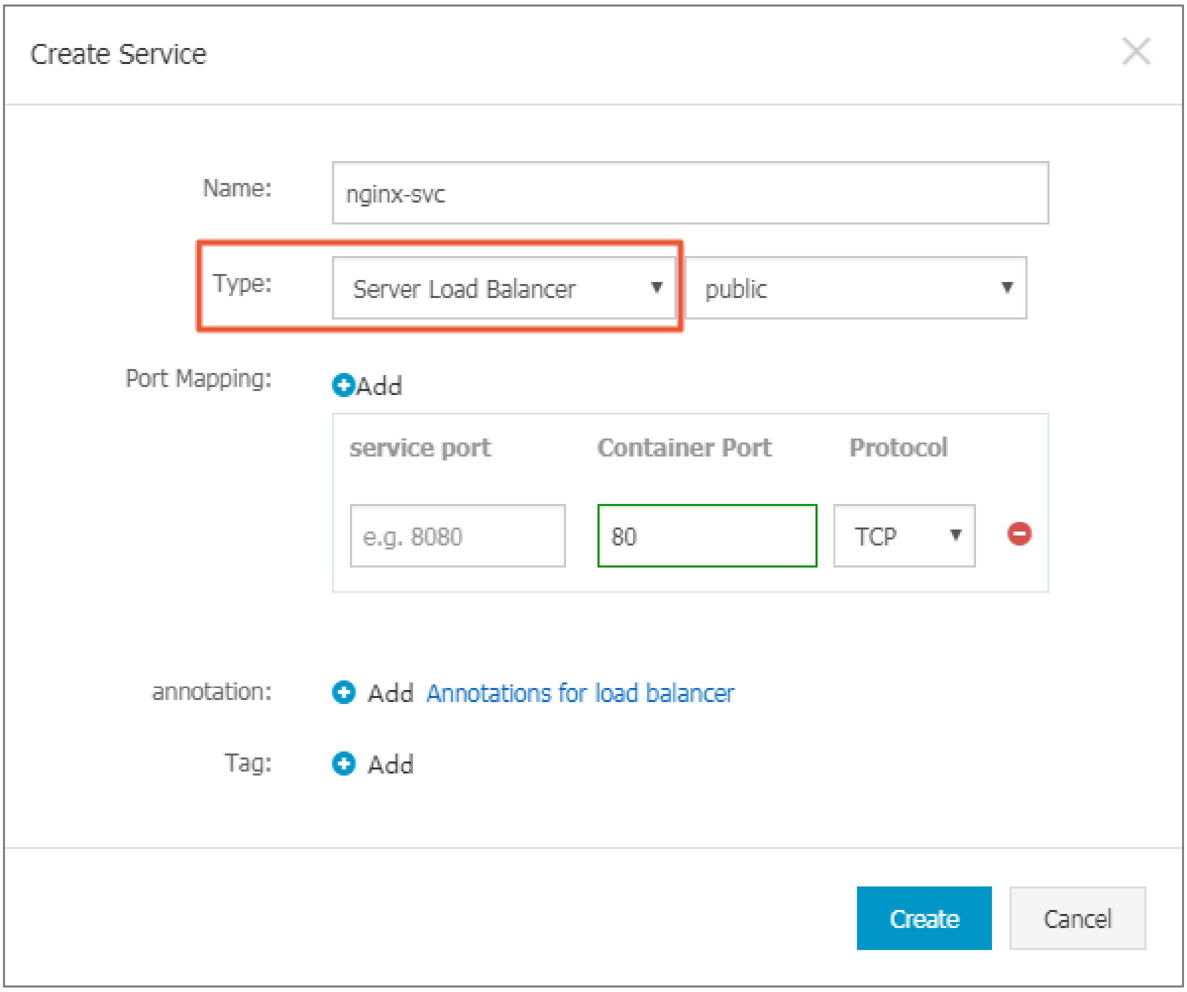
Two methods are available to create an SLB service in a Kubernetes cluster.

Method 1: Configure an SLB service when creating an application

1. After you complete the Container setting, configure the Advanced setting. Specifically, click Create on the right of Service in the Access Control area.



2. Select the Server Load Balancer Type. For more information, see Create a deployment application by using an image.



Create Service

Name: nginx-svc

Type: Server Load Balancer ▾ public ▾

Port Mapping:

service port	Container Port	Protocol
e.g. 8080	80	TCP ▾

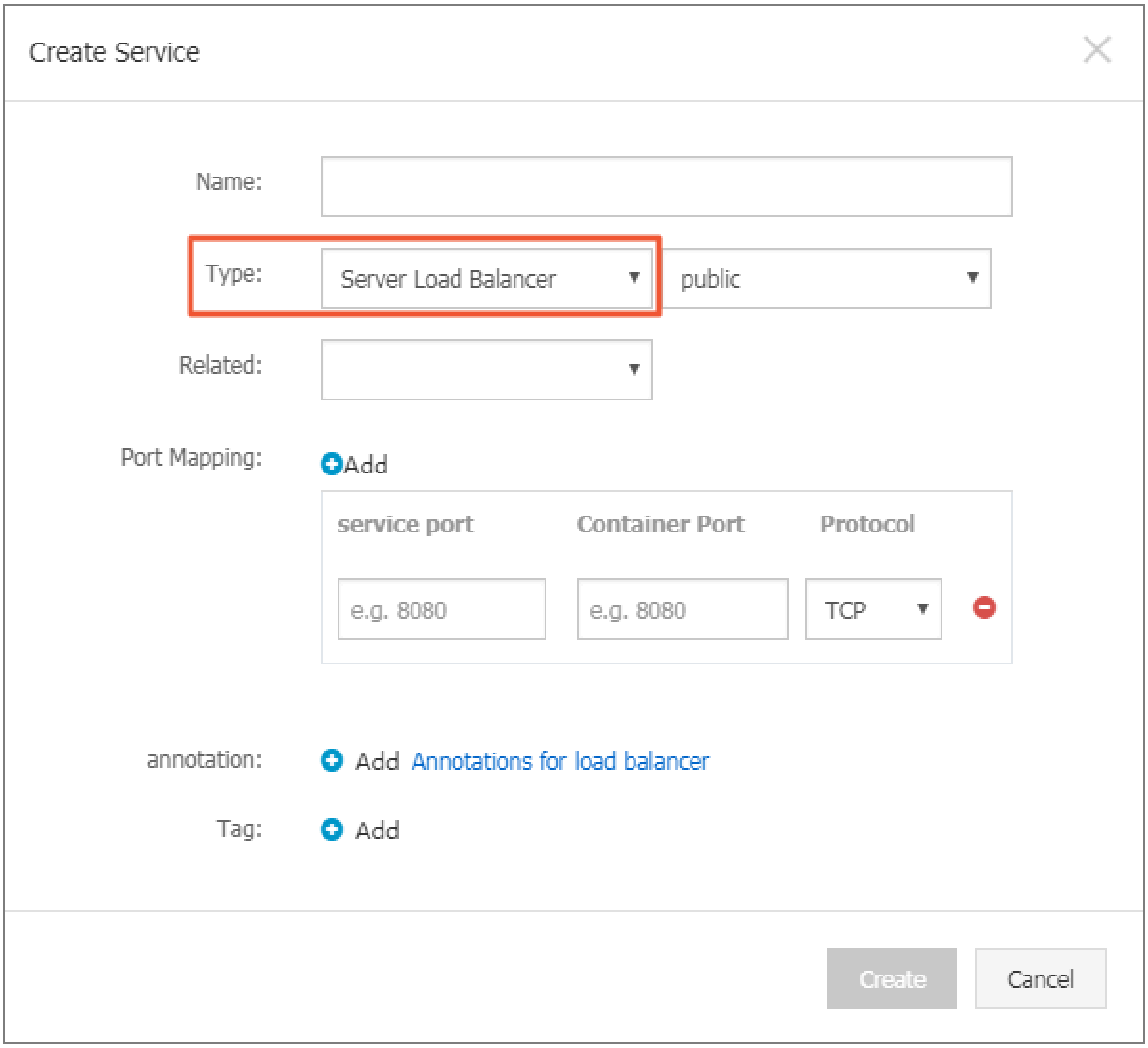
annotation:

Tag:

Method 2: Create an SLB service directly

1. In the left-side navigation pane in the Container Service console, chooseDiscovery and Load Balancing > Service.

2. Select the target cluster and namespace, and click Create. In the Create Service dialog box, select the Server Load Balancer Type. For more information, see Create a service.



Create Service

Name:

Type: Server Load Balancer public

Related:

Port Mapping:

service port	Container Port	Protocol
e.g. 8080	e.g. 8080	TCP

annotation: Add Annotations for load balancer

Tag: Add

1.5 Volume settings and environment variable settings used for creating an application through an image

This topic compares the volume settings and the environment variable settings used in a Swarm cluster with those used in a Kubernetes cluster for creating an application through an image.

Create an application by using an image

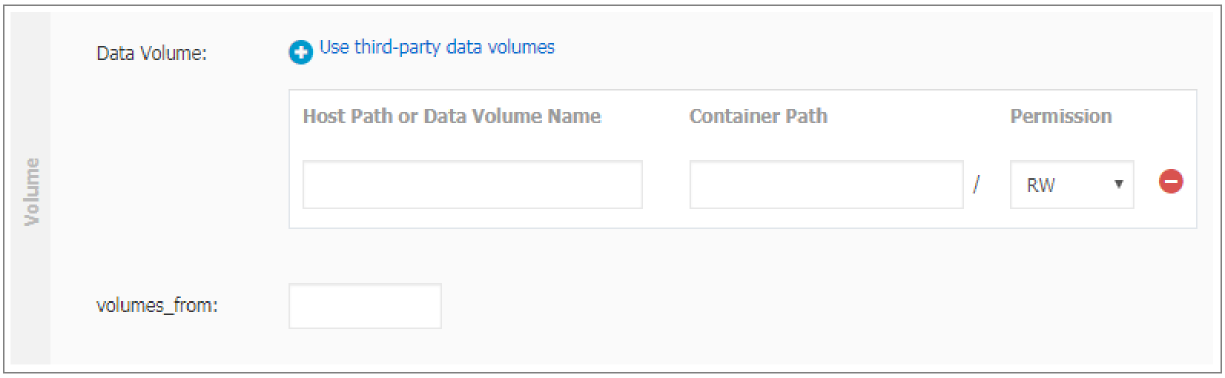
If you create an application in the Container Service console by using an image, the Swarm cluster Web interface is different from the Kubernetes cluster Web interface.

- For more information about the Web interface of a Swarm cluster, see Create an application.
- For more information about the Web interface of a Kubernetes cluster, see Create a deployment application by using an image.

Set a volume

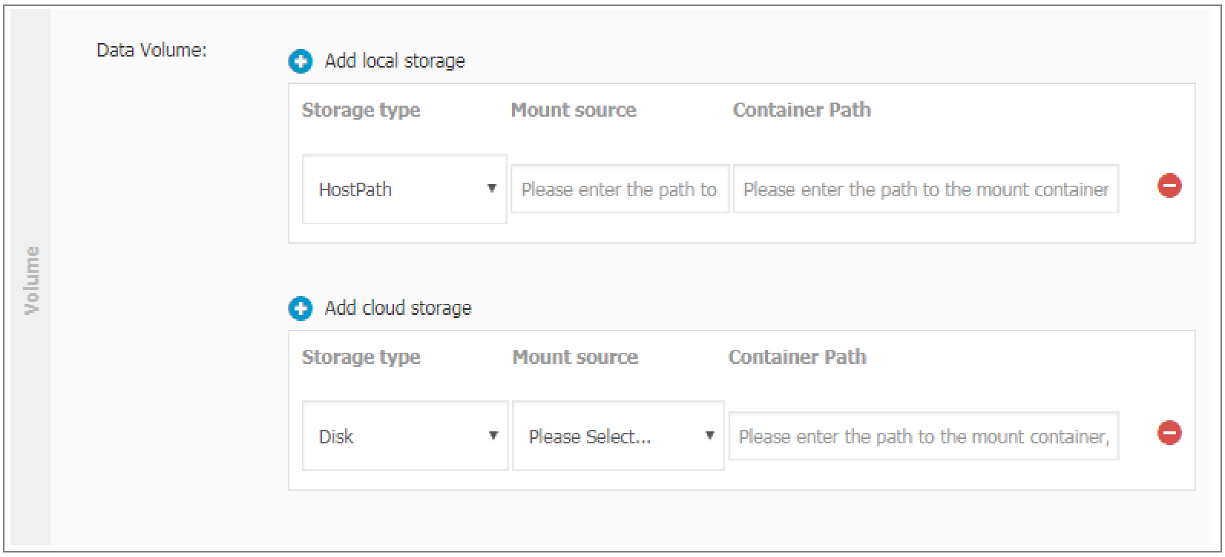
Container Service Swarm clusters

Specify your cloud or local storage path.



Container Service Kubernetes clusters

In Container Service, storage devices can be used in the same way in both Kubernetes and Swarm clusters, which have basically the same cluster console interface settings. However, the storage devices are mounted with different methods in these two types of clusters.

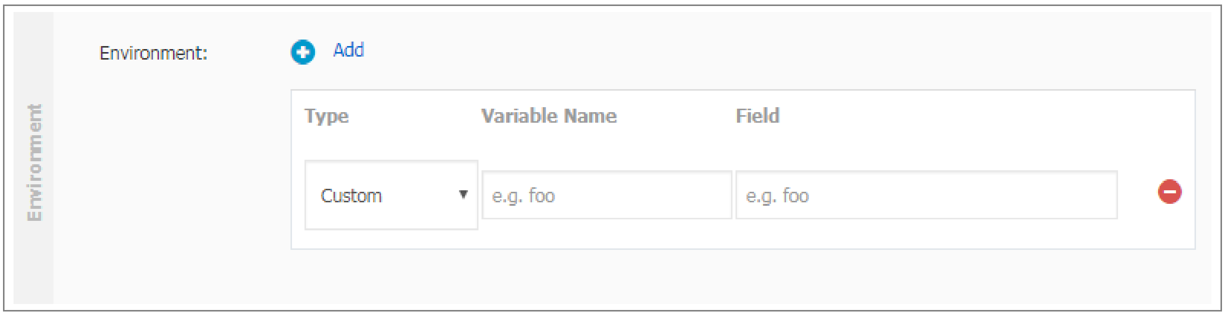


You can use either a local storage device or a cloud storage device.

- Available local storage types include HostPath, ConfigMap, Secret, and EmptyDir.
- Available cloud storage types include cloud disk, NAS, and OSS.

Set environment variables

The Environment parameter can be set with the same method for Swarm clusters and Kubernetes clusters. You only need to specify keys and their corresponding values.



1.6 Container settings and label settings used for creating an application through an image

This topic compares the container and label settings used in a Swarm cluster with those used in a Kubernetes cluster for creating an application through an image.

Create an application by using an image

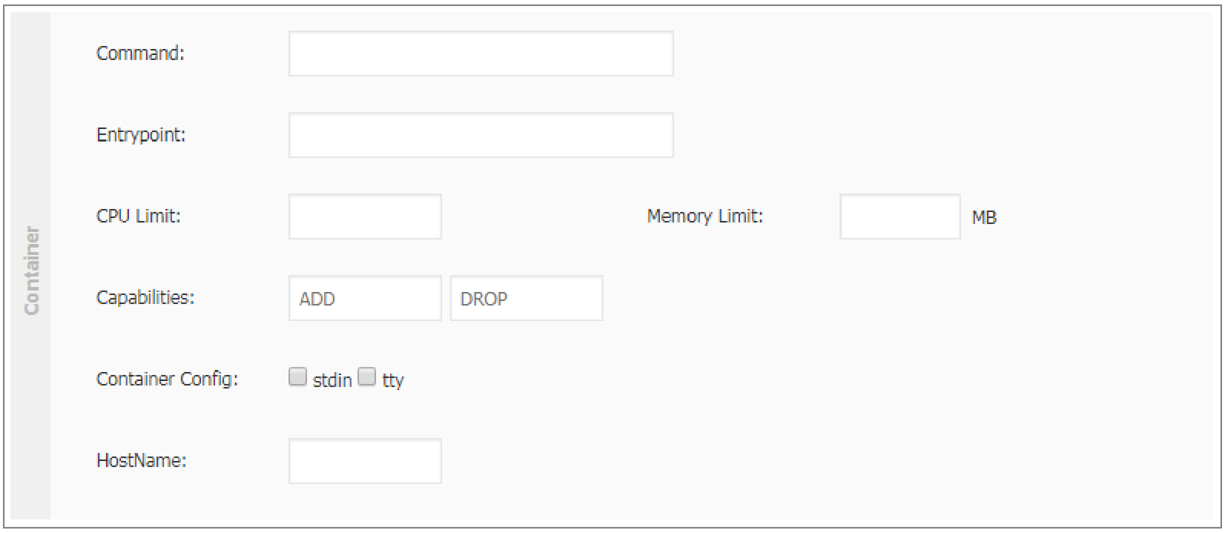
When you create an application in the Container Service console by using an image, you will see that the Web interfaces are different in a Swarm cluster and a Kubernetes cluster.

- For more information about the Web interface of a Swarm cluster, see Create an application.
- For more information about the Web interface of a Kubernetes cluster, see Create a deployment application by using an image.

Container settings

Container Service Swarm clusters

You can set container startup commands (through the Command parameter and the Entrypoint parameter), resource limits (including CPU Limit and Memory Limit), Container Config, and other parameters.



Container

Command:

Entrypoint:

CPU Limit: Memory Limit: MB

Capabilities: ADD DROP

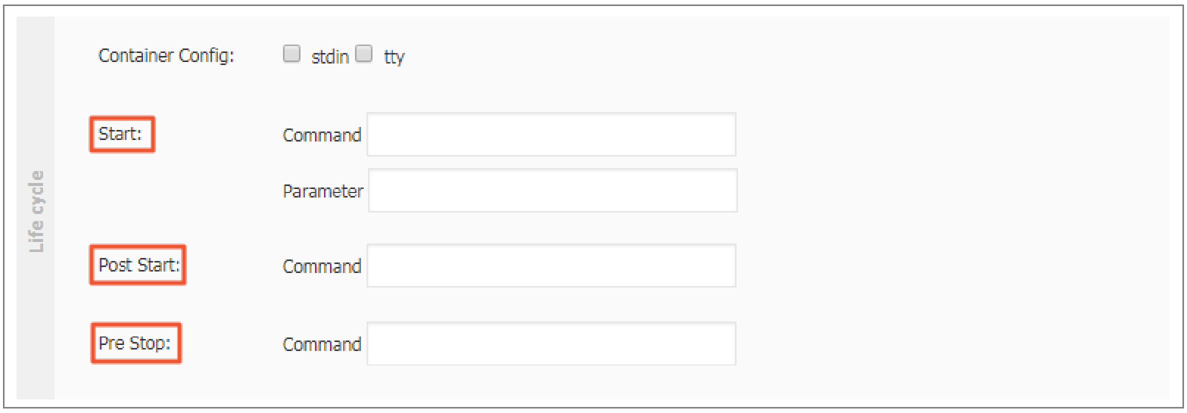
Container Config: □ stdin □ tty

HostName:

Container Service Kubernetes clusters

The Container settings of the Swarm cluster are similar to the life cycle settings and some general settings of the Kubernetes cluster.

- Life Cycle settings include the following parameters. For more information about the parameter description, see Create a deployment application by using an image.
  - Start
  - Post Start
  - Pre Stop



Life cycle

Container Config: □ stdin □ tty

Start: Command

Parameter

Post Start: Command

Pre Stop: Command

- General settings include the following parameters. For more information about the parameter description, see Create a deployment application by using an image. For more information about setting parameters, see Recommended Kubernetes cluster configurations to run highly reliable applications.
  - Resource Limit
  - Resource Request

Label

Container Service Swarm clusters

With labels, you can set health checks, access domain names, logs, and other functions.

Container Service Kubernetes clusters

A label can only mark an application in a Kubernetes cluster. Different methods are used in a Kubernetes cluster to implement the functions that are implemented through labels in a Swarm cluster, such as health checks and access domain names.

When you create an application in a Kubernetes cluster by using an image, a label of the same name as the application is created. The label is not displayed on the application configuration page. You can use labels in YAML files.

1.7 Health check settings and auto scaling settings used for creating an application through an image

This topic compares the health check settings and the auto scaling settings used in a Swarm cluster and those used in a Kubernetes cluster for creating an application through an image.

Create an application by using an image

When you create an application in the Container Service console by using an image, you will see that the Web interfaces are different in a Swarm cluster and a Kubernetes cluster.

- For more information about the Web interface of a Swarm cluster, see Create an application.
- For more information about the Web interface of a Kubernetes cluster, see Create a deployment application by using an image.

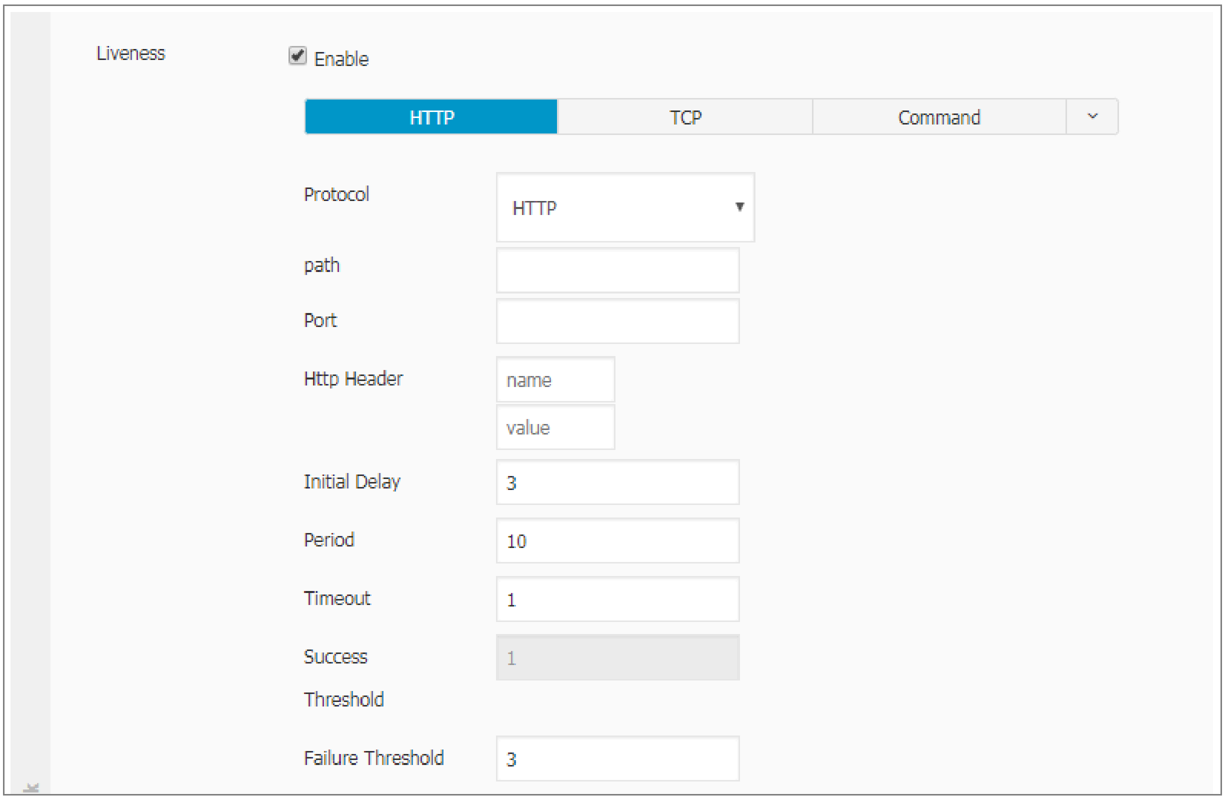
Set health checks

Container Service Swarm clusters

Health checks are implemented through labels.

Container Service Kubernetes clusters

If you use an image to create an application, you can set health checks on the Container tab page. You can set a Liveness probe and a Readiness probe.



Liveness ☑ Enable

Protocol: HTTP

path:

Port:

Http Header: name / value

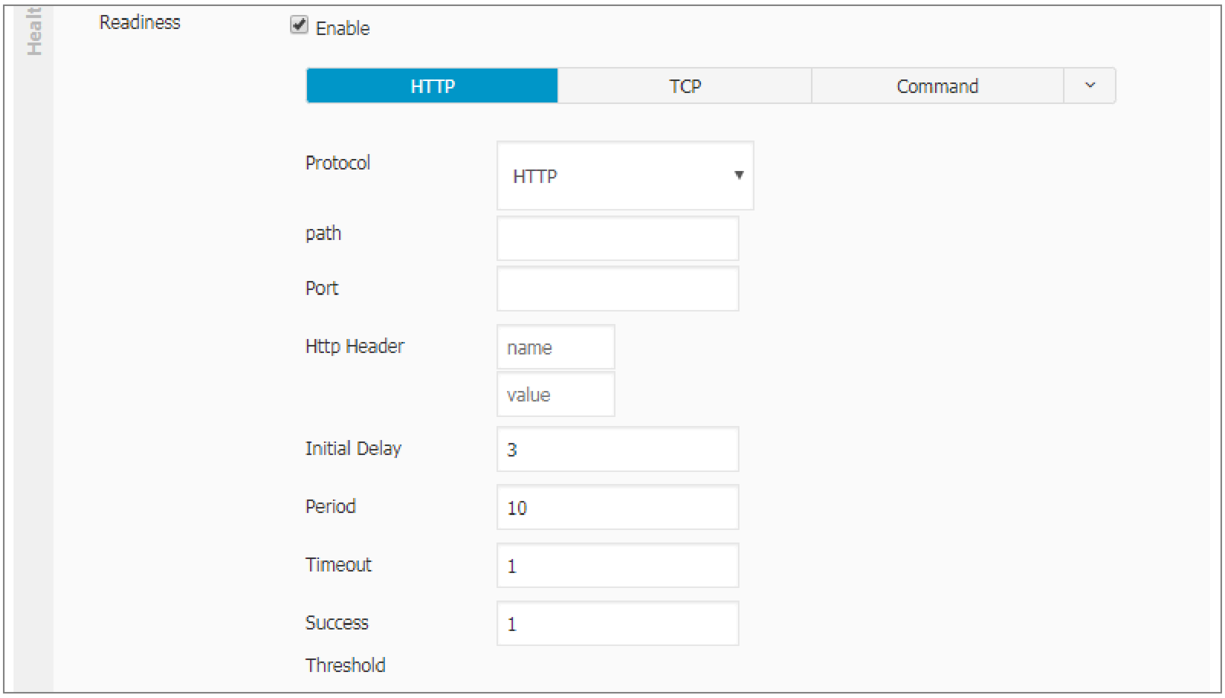
Initial Delay: 3

Period: 10

Timeout: 1

Success Threshold: 1

Failure Threshold: 3



Readiness ☑ Enable

Protocol: HTTP

path:

Port:

Http Header: name / value

Initial Delay: 3

Period: 10

Timeout: 1

Success Threshold: 1

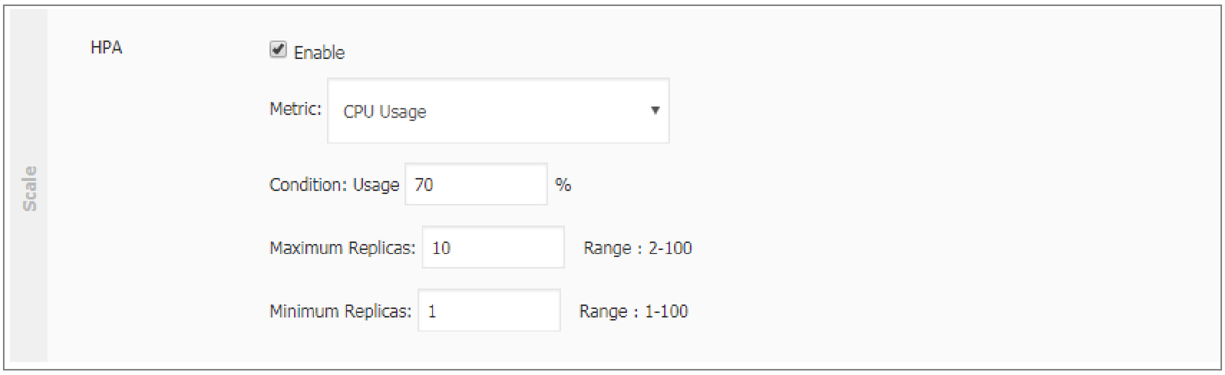
Set auto scaling

Container Service Swarm clusters

You can set auto scaling according to CPU usage and memory usage.

Container Service Kubernetes clusters

You can set auto scaling according to CPU usage and memory usage by enabling Horizontal Pod Autoscaling (HPA).



HPA ☑ Enable

Metric: CPU Usage ▾

Condition: Usage 70 %

Maximum Replicas: 10 Range : 2-100

Minimum Replicas: 1 Range : 1-100

1.8 YAML files used for creating applications

This topic describes the relation between the YAML files used in a Swarm cluster and those used in Kubernetes cluster for creating applications.

Background

The formats of the YAML files used to create applications in a Swarm cluster and a Kubernetes cluster are different.

- You can use Kompose to convert a Swarm cluster YAML file to a Kubernetes cluster YAML. But you still need to check the converted YAML file.

  To obtain Kompose, see https://github.com/AliyunContainerService/kompose.

  You can download Kompose at one of the following URLs:

  - The Kompose download URL for the Mac operating system is http://acs-public-mirror.oss-cn-hangzhou.aliyuncs.com/swarm/kompose-darwin-amd64
  - The Kompose download URL for the Linux operating system is http://acs-public-mirror.oss-cn-hangzhou.aliyuncs.com/swarm/kompose-linux-amd64
  - The Kompose download URL for the Windows operating system is http://acs-public-mirror.oss-cn-hangzhou.aliyuncs.com/swarm/kompose-windows-amd64.exe

  Note:

Kompose does not support certain customized labels in Alibaba Cloud. The Alibaba Cloud Container Service Team is developing solutions so that Kompose can support all customized labels.

Table 1-1: Kompose does not support the following tags.

Tag	Related link
external	External
dns_options	dns_options
oom_kill_disable	oom_kill_disable
affinity:service	Service deployment constraints (affinity:service)

- You can also manually modify a Swarm cluster YAML file to make it compatible with a Kubernetes cluster.

This topic describes the relation between the YAML files used in the two types of cluster. You must orchestrate YAML files according to conditions required by the application deployment. The YAML files in this topic are used only as examples.

Comparison between YAML files used in a Swarm and those used in a Kubernetes cluster for creating applications

Container Service Swarm cluster

The following is a `wordpress - swarm . yaml` file used in the Swarm cluster. Note each parameter marked by a number in the following YAML file corresponds to the parameter marked by the same number in the YAML file used in the Kubernetes cluster.

```
web :  #--- 1
  image :  registry . aliyuncs . com / acs - sample / wordpress : 4 .
5   #--- 2
  ports :   #--- 3
   - ' 80 '
  environmen  t :  #--- 4
    WORDPRESS_  AUTH_KEY :  changeme    #--- 5
    WORDPRESS_  SECURE_AUT  H_KEY :  changeme   #--- 5
    WORDPRESS_  LOGGED_IN_  KEY :  changeme   #--- 5
    WORDPRESS_  NONCE_KEY :  changeme    #--- 5
    WORDPRESS_  AUTH_SALT :  changeme    #--- 5
    WORDPRESS_  SECURE_AUT  H_SALT :  changeme   #--- 5
    WORDPRESS_  LOGGED_IN_  SALT :  changeme   #--- 5
    WORDPRESS_  NONCE_SALT :  changeme    #--- 5
    WORDPRESS_  NONCE_AA :  changeme    #--- 5
  restart :  always   #--- 6
  links :   #--- 7
   - ' db : mysql '
  labels :   #--- 8
    aliyun . logs : / var / log / mysql
    aliyun . probe . url :  http :// container / license . txt   #---
10
    aliyun . probe . initial_de  lay_second  s : ' 10 '   #--- 10
    aliyun . routing . port_80 :  http :// wordpress    #--- 11
    aliyun . scale : ' 3 '       #--- 12
db :    #--- 1
  image :  registry . aliyuncs . com / acs - sample / mysql : 5 . 7
#--- 2
  environmen  t :  #--- 4
    MYSQL_ROOT  _PASSWORD :  password   #--- 5
  restart :  always   #--- 6
  labels :  #--- 8
    aliyun . logs : / var / log / mysql   #--- 9
```

Container Service Kubernetes cluster

The WordPress application deployed through the *wordpress - swarm . yaml* file in the Swarm cluster corresponds to two services in the Kubernetes cluster, that is, the Web service and the db service.

A Kubernetes cluster requires two deployments and two services. You must create one service for each deployment. The two services are used to expose the access methods for the two applications.

In the Kubernetes cluster, the deployment and the service that correspond to the Web application of the Swarm cluster are created by using the following YAML files:

Note:
The following YAML files are used only as examples to describe their relation with the *wordpress - swarm . yaml* file. We recommend that you do not use these files to deploy your applications.

- *wordpress - kubernetes - web - deployment . yaml* file

```
apiVersion :  apps / v1      #  API   version
kind :  Deployment   #  type   of   the   resource   that   you
want   to   create
metadata :
  name :  wordpress      #--- 1
  labels :   #--- 8   This   label   is   only   used   to   mark
the   resource .
    app :  wordpress
spec : # resource   details
  replicas :  2        #--- 12   Indicates   the   number   of
replicas .
  selector :
    matchLabel  s :
      app :  wordpress
      tier :  frontend
strategy :
type :  Recreate
```

```
  template :   # Defines   the   pod   details .
    metadata :
      labels :  # Keeps   settings   consistent   with   the
preceding   labels   parameter .
        app :  wordpress
        tier :  frontend
    spec :    # Defines   the   container   details   in   the
pod .
      containers :    #
     -  image :  wordpress : 4    #--- 2    Correspond  s   to
the   image   name   and   version .
        name :  wordpress
        env :    #--- 4   Indicates   environmen  t   variable
  settings ,  including   config   maps   and   secrets   in
Kubernetes .
       -  name :  WORDPRESS_  DB_HOST
          value :  wordpress - mysql   #--- 7   Indicates   the
MySQL   that   you   want   to   access .
       -  name :  WORDPRESS_  DB_PASSWOR  D     #--- 5   Indicates
  a   password .  Note   Kubernetes   provides   a   secret   to
encrypt   the   password .
          valueFrom :
            secretKeyR  ef :
              name :  mysql - pass
              key :  password - wordpress
        ports :   #--- 3   Indicates   the   exposed   port   of
the   applicatio  n   within   the   container .
       -  containerP  ort :  80
          name :  wordpress
livenessPr  obe :        # Add   a   health   check   setting
--- 10   health   check
          httpGet :
            path : /
            port :  8080
          initialDel  aySeconds :  30
          timeoutSec  onds :  5
          periodSeco  nds :  5
        readinessP  robe :       # Add   a   health   check
setting   --- 10   health   check
          httpGet :
            path : /
            port :  8080
          initialDel  aySeconds :  5
          timeoutSec  onds :  1
          periodSeco  nds :  5
        volumeMoun  ts :  # Mount   the   volume   to   the
container .
       -  name :  wordpress - pvc
          mountPath : / var / www / html
      volumes :   # Indicates   to   obtain   the   volume .  You
need   to   first   create   a   PV   and   a   PVC .
     -  name :  wordpress - pvc
        persistent  VolumeClai  m :
          claimName :  wordpress - pv - claim
```

- *wordpress - kubernetes - web - service . yaml* **file**

```
apiVersion :  v1    # version   number
kind :  Service     # Indicates   the   type   of   the   resource
  that   you   want   to   create .  It   is   Service   in   this
  YAML   file .
metadata :
  name :  wordpress
  labels :
    app :  wordpress
spec :
  ports :
   - port :  80    # service   port
  selector :  # Indicates   to   associate   the   service   with
the   applicatio  n   through   the   label .
    app :  wordpress
    tier :  frontend
  type :  LoadBalanc  er   #--- 11   Defines   the   access
method .  This   YAML   file   specifies   an   SLB   service   and
  an   SLB   instance   will   be   created   automatica  lly .
```

In the Kubernetes cluster, the deployment and the service that correspond to the Web application of the Swarm cluster are created by using the following YAML files:

Note:

The following YAML files are only used as examples to describe their relation with the *wordpress - swarm . yaml* file. We recommend that you do not use these files for application deployment.

- *wordpress - kubernetes - db - deployment . yaml* file

```
apiVersion :  apps / v1
kind :  Deployment
metadata :
  name :  wordpress - mysql
  labels :
    app :  wordpress
spec :
  selector :
    matchLabel  s :
      app :  wordpress
      tier :  mysql
  strategy :
    type :  Recreate
  template :
    metadata :
      labels :
        app :  wordpress
        tier :  mysql
    spec :
      containers :
     - image :  mysql : 5 . 6
        name :  mysql
        env :
       - name :  MYSQL_ROOT  _PASSWORD
          valueFrom :
            secretKeyR  ef :
              name :  mysql - pass
              key :  password - mysql
        ports :
       - containerP  ort :  3306
          name :  mysql
        volumeMoun  ts :
       - name :  wordpress - mysql - pvc
          mountPath : / var / lib / mysql
      volumes :
```

```
      -  name :  wordpress - mysql - pvc
         persistent  VolumeClai  m :
           claimName :  wordpress - mysql - pv - claim
```

- *wordpress - kubernetes - db - service . yaml* file

```
apiVersion :  v1
kind :  Service
metadata :
  name :  wordpress - mysql
  labels :
    app :  wordpress
spec :
  ports :
   -  port :  3306
  selector :
    app :  wordpress
    tier :  mysql
      clusterIP :  None
```

1.9 Network

This topic compares the networks used by Swarm clusters and Kubernetes clusters.

Swarm cluster

A Swarm cluster can use either of the following two networks:

- A VPC
- A classic network

Kubernetes cluster

A Kubernetes cluster can only use a VPC. For more information, see Plan Kubernetes CIDR blocks under VPC.

- To guarantee that a Kubernetes cluster and a Swarm cluster can be connected with a VPC, you must select the same VPC when creating the Kubernetes cluster.
- To guarantee that a Kubernetes cluster can be connected with a Swarm cluster that uses a classic network, you must migrate the Swarm cluster to a VPC. For more information, see Migration overview.

  After a Kubernetes cluster and a Swarm cluster are connected through a network, storage devices (such as OSS, NAS, or RDS) or databases in the Swarm cluster will obtain IP addresses in the VPC. That is, Kubernetes cluster applications can use these IP addresses to access corresponding storage devices or databases in the Swarm cluster over the VPC.

1.10 Logging and monitoring

This topic compares logging and monitoring functions of a Swarm cluster with those of a Kubernetes cluster.

Logging

Swarm cluster

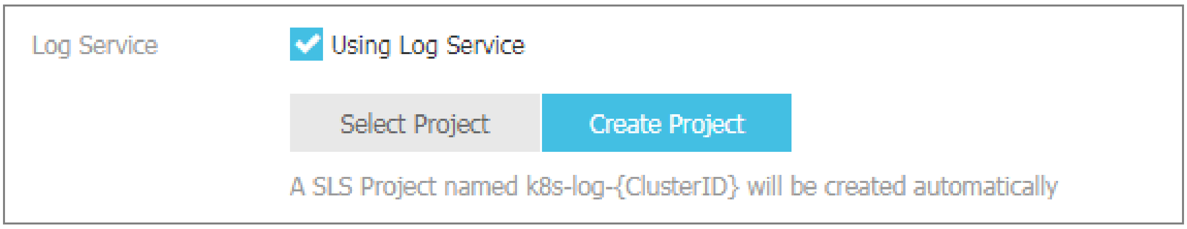
For a Swarm cluster, the logging function is implemented through labels.

Kubernetes cluster

For a Kubernetes cluster, the logging function is configured and used in the following scenarios:

- Create a Kubernetes cluster.

  On the Create Kubernetes Cluster page, select the Using Log Service check box. Then the Log Service plugin is automatically installed in the cluster. You can use an existing project or create a new project.

  Log Service ☑ Using Log Service

  A SLS Project named k8s-log-{ClusterID} will be created automatically

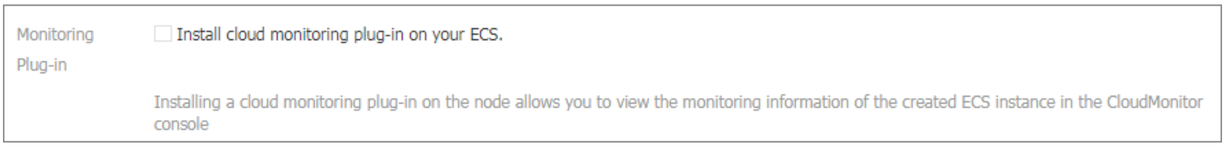
  You can also manually install Log Service components in the created cluster. For more information, see Manually install Log Service components in a created Kubernetes cluster.
- Configure Log Service when creating an application. For more information, see Configure Log Service when creating an application.
- Use Log Service after creating an application. For more information, see Container text logs and Container stdout.

Monitoring

For both Swarm and Kubernetes clusters, select the Install cloud monitoring plug-in on your ECS check box on the Create Cluster page. You can then monitor the ECS instances through the CloudMonitor console.

Swarm cluster

By default, the monitoring function is disabled.



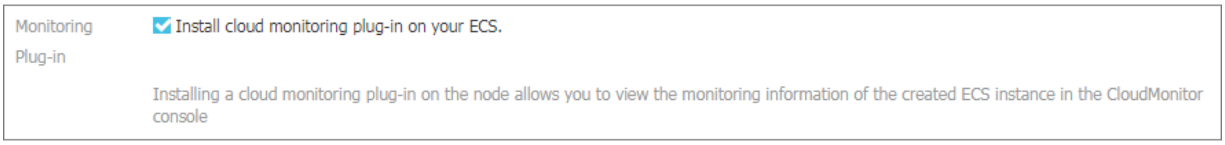
Monitoring Plug-in

□ 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

Kubernetes cluster

By default, the monitoring function is enabled.



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

For more information, see Integration and usage with CloudMonitor.

1.11 Application access methods

This topic compares the application access methods used in a Swarm cluster with those used in a Kubernetes cluster. Specifically, these methods are used for access between applications within a cluster, and access between applications outside the cluster and application within the cluster.

Access applications within a cluster

Container Service Swarm clusters

For a service name that is to be accessed in a Swarm cluster, you can use the `links` label to set the service name in the container environment variables.

For example, in YAML files used for creating applications, the Web service of the WordPress application is associated with *mysql* . Therefore, the MySQL service can be accessed through the *mysql* service name after the container is started.

```
links :    #--- 7
   - ' db : mysql '
```

Container Service Kubernetes clusters

In a Kubernetes cluster, an application can be accessed through the service cluster IP address or the application service name. We recommend that you use service names for access between applications within a Kubernetes cluster.

When creating an application, you can specify the service name that needs to be accessed as an environment variable.

For example, in YAML files used for creating applications, WordPress calls the *mysql* service through the environmental variable specified in the YAML file of the application.

```
spec :
      containers :
     -  image :  wordpress : 4
        name :  wordpress
        env :
       -  name :  WORDPRESS_  DB_HOST
          value :  wordpress - mysql     #--- 7   Use   the   mysql
service    name    to    specify    the    MySQL    that    needs    to    be
  accessed .
       -  name :  WORDPRESS_  DB_PASSWOR  D
```

Access applications from outside a cluster

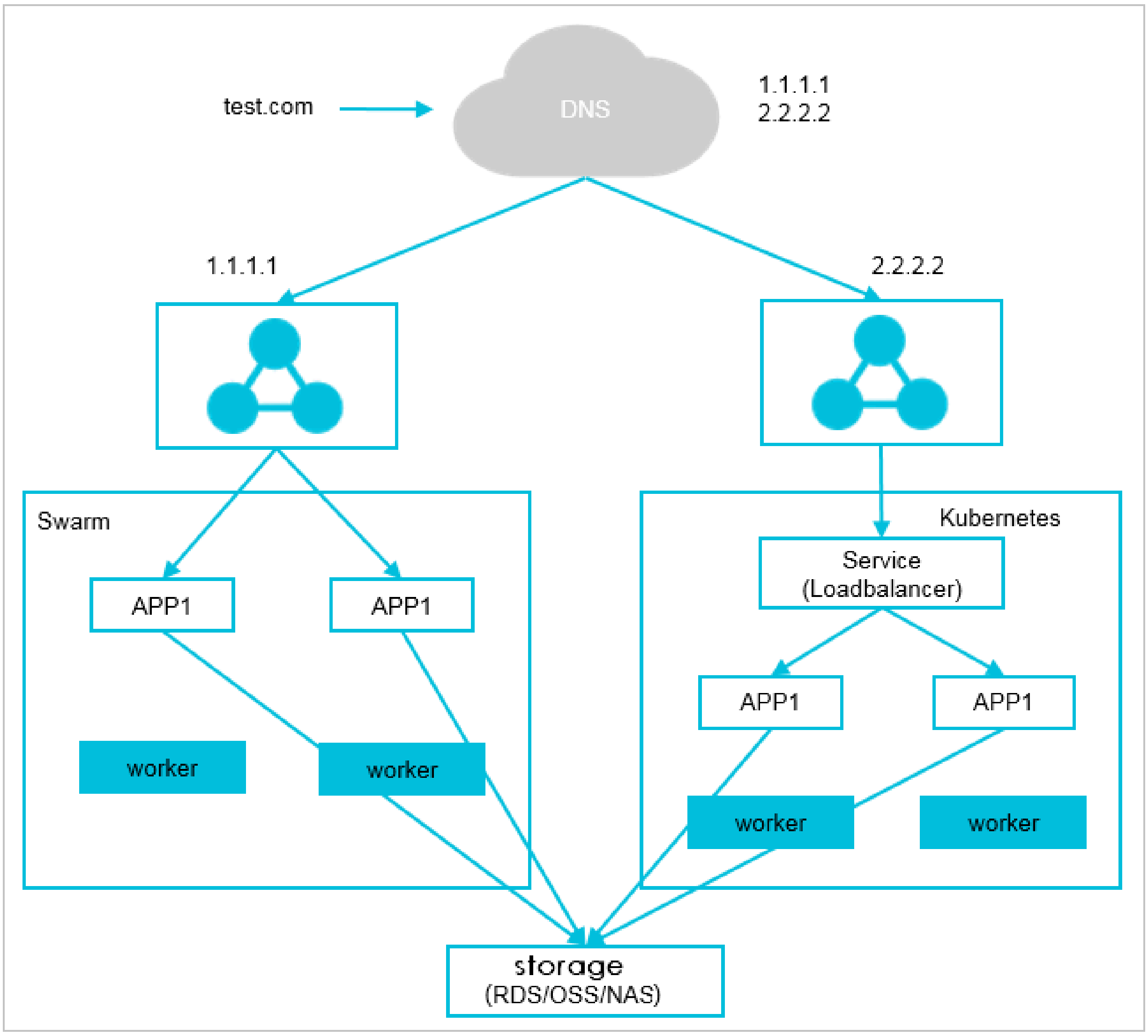
A Swarm cluster application is accessed through a domain name

Note:

- You must ensure the network connection status is normal for either a classic network or a VPC.
- DNS can forward traffic to different backend IP addresses through its load balancing capacity.
- If a Swarm cluster application is accessed through a domain name, you can migrate the application services from the Swarm cluster to a Kubernetes cluster without downtime.

Simple routing (a domain name bound to the default SLB of a Swarm cluster)

Create an application in a Kubernetes cluster and verify the application availability is available before migrating a Swarm cluster application to the Kubernetes cluster.



Migration method

- Follow these steps to create an application in a Kubernetes cluster:
  - In the Kubernetes cluster, create an application of the same type as the application that you want to migrate from a Swarm cluster.
  - In the Kubernetes cluster, create an SLB service for the application.
  - The SLB service creates an SLB instance. In this example, the IP address of the SLB instance is 2.2.2.2.
  - Add 2.2.2.2 to the backend IP addresses of the `test . com` domain name in DNS.
- Verify that the created application in the Kubernetes cluster is available

  Access the created application through 2.2.2.2 to verify the created application in the Kubernetes cluster is available.

- Migrate the application

  Remove 1.1.1.1 from the backend IP addresses of the `test . com` domain name in DNS.

After you complete the preceding steps, all traffic destined for the application in the Swarm cluster is all forwarded by DNS to the Kubernetes cluster application.

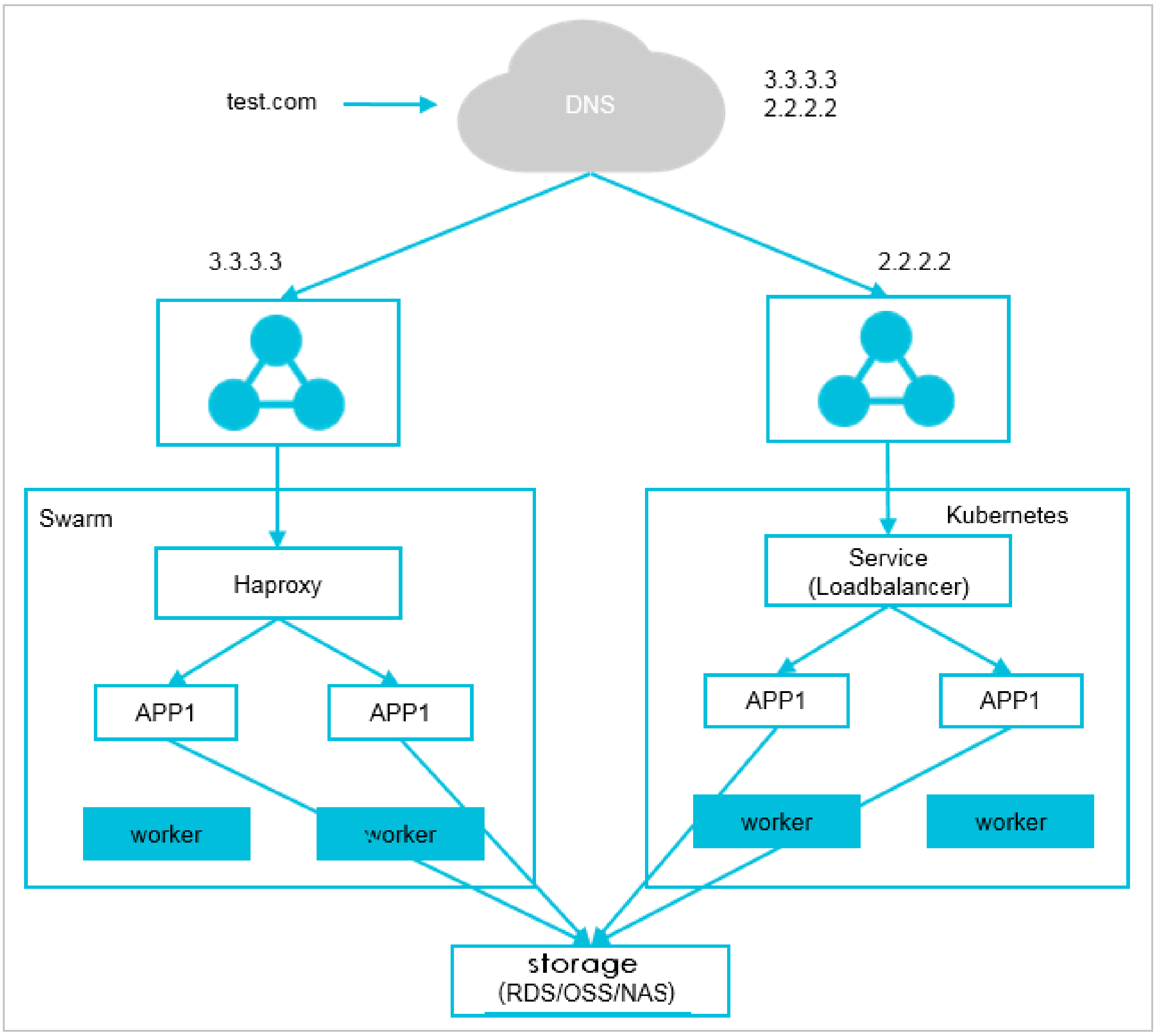
Simple routing (a domain name specified for an application is bound to an on-premise SLB of a Swarm cluster)

In a Swarm cluster, you can bind an application domain name to the default SLB or an on-premise SLB. The differences between these two methods are as follows:

- The SLB is on-premise and not the default one.
- By default, the DNS is Alibaba Cloud DNS. If you use your own domain name, you need to manually resolve it.

Migration method

You can use the same migration method as that used for the scenario in which the domain name is bound to the default SLB of a Swarm cluster. That is, create an application in a Kubernetes cluster and then verify if the application is available before migrating.



A Swarm cluster application is accessed through <HostIP>:<port>

If a Swarm cluster application is accessed through <HostIP>:<port>, the applicatio n service migration will encounter downtime. Therefore, we recommend that you migrate the application service when the application has the minimum access traffic.

Migration method

1. Create an application in a Kubernetes cluster and use a NodePort service to expose the access method of the application outside the cluster. For more information, see Configure port mapping.
2. Replace the <port> value of the Swarm cluster with the <NodePort> value specified for the Kubernetes cluster.

Note:

You need to disable and modify the applications in the Swarm cluster one by one.

3. Mount the Worker nodes in the Kubernetes cluster to the SLB instance in the Swarm cluster.
4. After you verify that the application in the Kubernetes cluster is available, remove the nodes of the Swarm cluster from the SLB instance in the Kubernetes cluster . Then the application services are migrated from the Swarm cluster to the Kubernetes cluster. Note that before you perform this step, some traffic destined for the application of the Swarm cluster will be forwarded to the application of the Kubernetes cluster.

An application is accessed through an SLB instance

If a Swarm cluster application is accessed through an SLB instance, the applicatio n service migration will encounter downtime. Therefore, we recommend that you migrate the application services when there is the minimum service traffic.

Migration method

In a Kubernetes cluster, you can use an SLB instance in the same way as in a Swarm cluster. For more information, see Configure Server Load Balancer.

2 Run TensorFlow-based AlexNet in Alibaba Cloud Container Service

AlexNet is a CNN network developed in 2012 by Alex Krizhevsky using five-layer convolution and three-layer ReLU layer, and won the ImageNet competition (ILSVRC). AlexNet proves the effectiveness in classification (15.3% error rate) of CNN, against the 25% error rate by previous image recognition tools. The emergence of this network marks a milestone for deep learning applications in the computer vision field.

AlexNet is also a common performance indicator tool for deep learning framework. TensorFlow provides the alexnet_benchmark.py tool to test GPU and CPU performance. This document uses AlexNet as an example to illustrate how to run a GPU application in Alibaba Cloud Container Service easily and quickly.

Prerequisite

Create a GN5 GPU cluster in Container Service console.

#unique_37

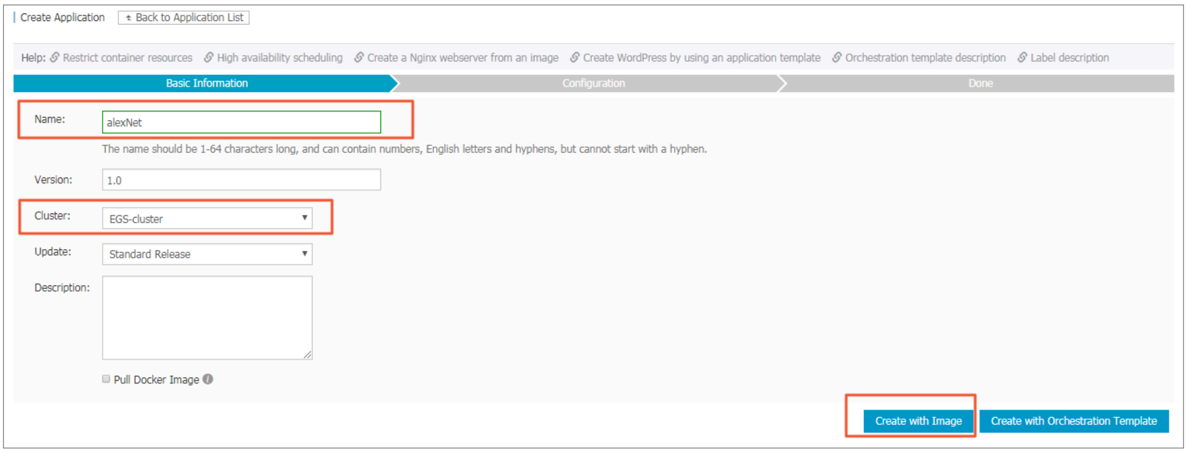
Prerequisite

This operation is based on the Container Service Beijing HPC or GN4 type GPU ECS instance.

Procedure

1. Log on to the Container Service console.
2. ClickImages and Templates > > Imagein the left-side navigation pane.

3. Enter the application name (alexNet in the example) and select the Beijing HPC or GN4 ECS cluster, and click Next step.


Create Application
Back to Application List
Help: Restrict container resources
High availability scheduling
Create a Nginx webserver from an image
Create WordPress by using an application template
Orchestration template description
Label description
Basic Information
Configuration
Done
Create with Image
Create with Orchestration Template

Name: alexNet

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

Version: 1.0

Cluster: EGS-cluster

Update: Standard Release

Description:

□ Pull Docker Image

4. Configure the application.

   a. Enter `registry.cn-beijing.aliyuncs.com/tensorflow-samples/alexnet_benchmark:1.0.0-devel-gpu` in the Image Name field.

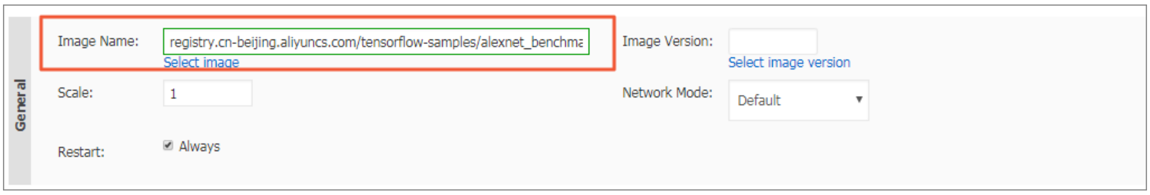


Image Name: registry.cn-beijing.aliyuncs.com/tensorflow-samples/alexnet_benchma

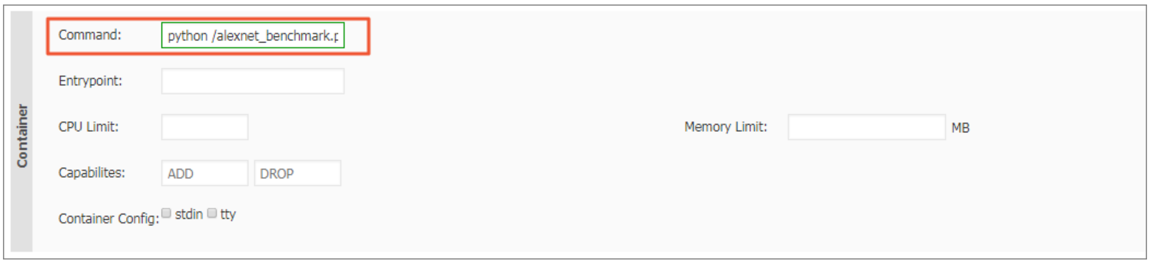
Image Version:

Scale: 1

Network Mode: Default

Restart: ☑ Always

   b. In the Container section, enter the command in the Command field. For example, enter `python /alexnet_benchmark.py --batch_size 128 -num_batches 100`.



Command: python /alexnet_benchmark.p

Entrypoint:

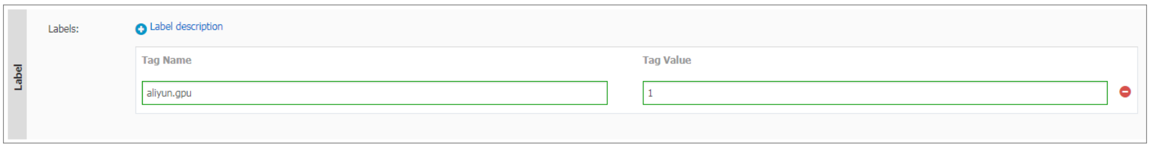
CPU Limit:

Memory Limit: MB

Capabilites: ADD DROP

Container Config: □ stdin □ tty

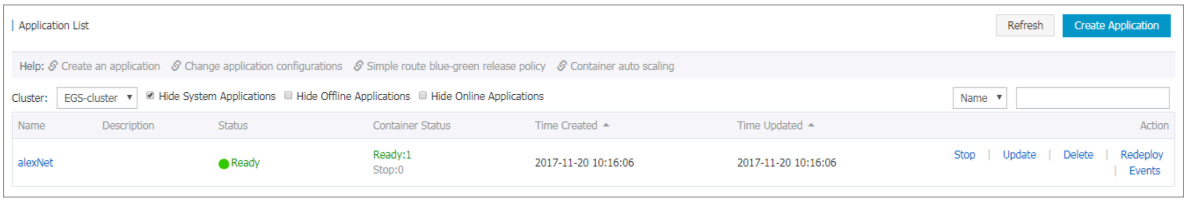
   c. Click the button in the Label section. Enter the Alibaba Cloud `gpu` extension label. Enter `aliyun.gpu` in the Tag Name field, and the number of scheduling GPUs (`1` in this example) in the Tag Value field.



Tag Name	Tag Value
aliyun.gpu	1

5. Click Create after completing the settings.

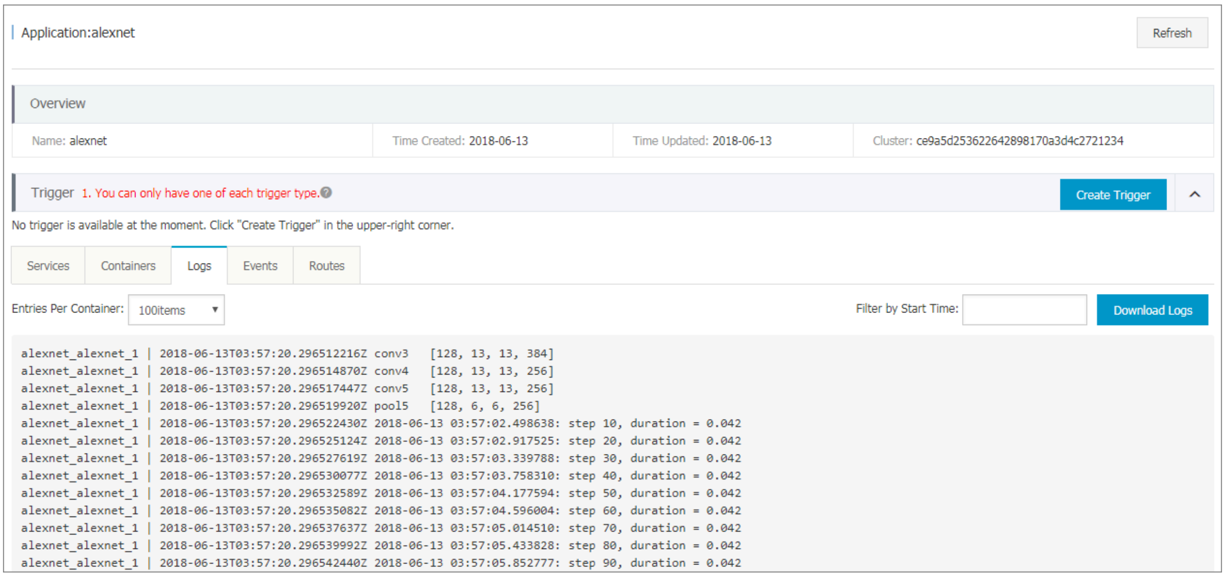
   You can view the created alexNet application on the Application List page.


Application List
Refresh
Create Application
Help: Create an application
Change application configurations
Simple route blue-green release policy
Container auto scaling
Cluster: EGS-cluster ▾
☑ Hide System Applications
□ Hide Offline Applications
□ Hide Online Applications
Name ▾

Name	Description	Status	Container Status	Time Created	Time Updated	Action
alexNet		● Ready	Ready:1 Stop:0	2017-11-20 10:16:06	2017-11-20 10:16:06	Stop | Update | Delete | Redeploy | Events

In this way, you can check the performance of AlexNet on EGS or HPC by means of the container Log Service in Container Service console.

On the Application List page, click the application name alexNet. Then, click the Container List, and click Logs on the right.



Name: alexnet | Time Created: 2018-06-13 | Time Updated: 2018-06-13 | Cluster: ce9a5d253622642898170a3d4c2721234

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.

```
alexnet_alexnet_1 | 2018-06-13T03:57:20.296512216Z conv3   [128, 13, 13, 384]
alexnet_alexnet_1 | 2018-06-13T03:57:20.296514870Z conv4   [128, 13, 13, 256]
alexnet_alexnet_1 | 2018-06-13T03:57:20.296517447Z conv5   [128, 13, 13, 256]
alexnet_alexnet_1 | 2018-06-13T03:57:20.296519920Z pool5   [128, 6, 6, 256]
alexnet_alexnet_1 | 2018-06-13T03:57:20.296522430Z 2018-06-13 03:57:02.498638: step 10, duration = 0.042
alexnet_alexnet_1 | 2018-06-13T03:57:20.296525124Z 2018-06-13 03:57:02.917525: step 20, duration = 0.042
alexnet_alexnet_1 | 2018-06-13T03:57:20.296527619Z 2018-06-13 03:57:03.339788: step 30, duration = 0.042
alexnet_alexnet_1 | 2018-06-13T03:57:20.296530077Z 2018-06-13 03:57:03.758310: step 40, duration = 0.042
alexnet_alexnet_1 | 2018-06-13T03:57:20.296532589Z 2018-06-13 03:57:04.177594: step 50, duration = 0.042
alexnet_alexnet_1 | 2018-06-13T03:57:20.296535082Z 2018-06-13 03:57:04.596004: step 60, duration = 0.042
alexnet_alexnet_1 | 2018-06-13T03:57:20.296537637Z 2018-06-13 03:57:05.014510: step 70, duration = 0.042
alexnet_alexnet_1 | 2018-06-13T03:57:20.296539992Z 2018-06-13 03:57:05.433828: step 80, duration = 0.042
alexnet_alexnet_1 | 2018-06-13T03:57:20.296542440Z 2018-06-13 03:57:05.852777: step 90, duration = 0.042
```

3 Best practices for restarting nodes

Restarting nodes directly may cause an exception in clusters. In the context of Alibaba Cloud use cases, this document introduces the best practices for restarting nodes in the situations such as performing active Operation & Maintenance (O&M) on Container Service.

Check the high availability configurations of business

Before restarting Container Service nodes, we recommend that you check or modify the following business configurations. In this way, restarting nodes cannot cause the exception of a single node and the business availability cannot be impaired.

- Data persistence policy of configurations

  We recommend the data persistence for external volumes of important data configurations such as configurations of logs and business. In this way, after the container is restructured, deleting the former container cannot cause the data loss.

  For how to use the Container Service data volumes, see Manage data volumes.
- Restart policy of configurations

  We recommend that you configure the `restart : always` restart policy for the corresponding business services so that containers can be automatically pulled up after the nodes are restarted.
- High availability policy of configurations

  We recommend that you integrate with the product architecture to configure the affinity and mutual exclusion policies, such as high availability scheduling (availability:az propery), specified node scheduling (affinity and constraint properties) , and specified nodes scheduling (constraint property), for the corresponding business. In this way, restarting nodes cannot cause the exception of a single node. For example, for the database business, we recommend the active-standby or multi-instance deployment, and integrating with the preceding characteristics to make sure that different instances are on different nodes and related nodes are not restarted at the same time.

Best practices

We recommend that you check the high availability configurations of business by reading the preceding instructions. Then, follow these steps in sequence on each node. Do not perform operations on multiple nodes at the same time.

1. Back up snapshots

   We recommend that you create the latest snapshots for all the related disks of the nodes and then back up the snapshots. When starting the shut-down nodes , an exception occurs because the server is not restarted for a long time and the business availability is impaired. However, by backing up the snapshots, this can be avoided.
2. Verify the container configuration availability of business

   For a swarm cluster, restarting the corresponding business containers on nodes makes sure that the containers can be pulled up again normally.
3. Verify the running availability of Docker Engine

   Try to restart Docker daemon and make sure that the Docker Engine can be restarted normally.
4. Perform related O&M

   Perform the related O&M in the plan, such as updating business codes, installing system patches, and adjusting system configurations.
5. Restart nodes

   Restart nodes normally in the console or system.
6. Check the status after the restart

   Check the health status of the nodes and the running status of the business containers in the Container Service console after restarting the nodes.

4 Use OSSFS data volumes to share WordPress attachments

This document introduces how to share WordPress attachments across different containers by creating OSSFS data volumes in Alibaba Cloud Container Service.

Scenarios

Docker containers simplify WordPress deployment. With Alibaba Cloud Container Service, you can use an orchestration template to deploy WordPress with one click.

Note:

For more information, see Create WordPress with an orchestration template.

In this example, the following orchestration template is used to create an application named wordpress.

```
web :
  image :  registry . aliyuncs . com / acs - sample / wordpress : 4 .
3
  ports :
   - ' 80 '
  environmen  t :
    WORDPRESS_  AUTH_KEY :  changeme
    WORDPRESS_  SECURE_AUT  H_KEY :  changeme
    WORDPRESS_  LOGGED_IN_  KEY :  changeme
    WORDPRESS_  NONCE_KEY :  changeme
    WORDPRESS_  AUTH_SALT :  changeme
    WORDPRESS_  SECURE_AUT  H_SALT :  changeme
    WORDPRESS_  LOGGED_IN_  SALT :  changeme
    WORDPRESS_  NONCE_SALT :  changeme
    WORDPRESS_  NONCE_AA :  changeme
  restart :  always
  links :
   - ' db : mysql '
  labels :
    aliyun . logs : / var / log
    aliyun . probe . url :  http :// container / license . txt
    aliyun . probe . initial_de  lay_second  s : ' 10 '
    aliyun . routing . port_80 :  http :// wordpress
    aliyun . scale : ' 3 '
db :
  image :  registry . aliyuncs . com / acs - sample / mysql : 5 . 7
  environmen  t :
    MYSQL_ROOT  _PASSWORD :  password
  restart :  always
  labels :
    aliyun . logs : / var / log / mysql
```

This application contains a MySQL container and three WordPress containers (`aliyun . scale : ' 3 '` is the extension label of Alibaba Cloud Container Service, and specifies the number of containers. For more information about the labels supported by Alibaba Cloud Container Service, see Label description). The WordPress containers access MySQL by using a link. The `aliyun . routing . port_80 : http :// wordpress` label defines the load balancing among the three WordPress containers (for more information, see Simple routing - Supports HTTP and HTTPS).

In this example, the application deployment is simple and the deployed application is of complete features. However, the attachments uploaded by WordPress are stored in the local disk, which means they cannot be shared across different containers or opened when requests are routed to other containers.

Solutions

This document introduces how to use OSSFS data volumes of Alibaba Cloud Container Service to share WordPress attachments across different containers, without any code modifications.

OSSFS data volume, a third-party data volume provided by Alibaba Cloud Container Service, packages various cloud storages (such as Object Storage Service (OSS)) as data volumes and then directly mounts them to the containers. This means the data volumes can be shared across different containers and automatically re-mounted to the containers when the containers are restarted or migrated.

Procedure

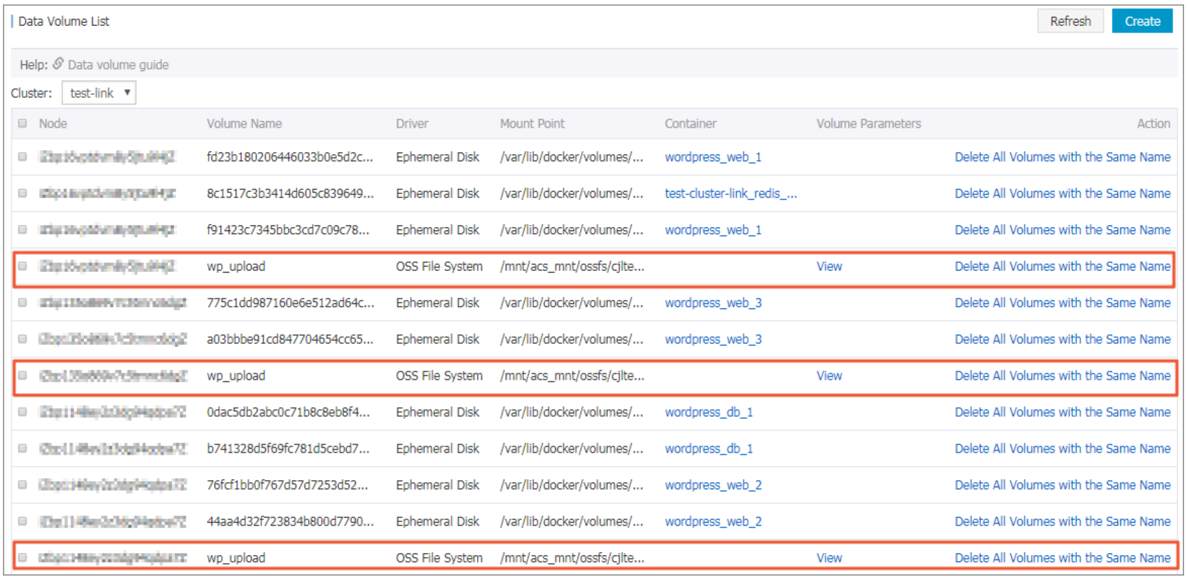
1. Create OSSFS data volumes.

   a. Log on to the Container Service console. Under Swarm, click Data Volumes in the left-side navigation pane.

   b. Select the cluster in which you want to create data volumes from the Cluster drop-down list. Click Create in the upper-right corner to create the OSSFS data volumes.

      For how to create OSSFS data volumes, see Create an OSSFS data volume.

   In this example, the created OSSFS data volumes are named wp_upload. Container Service uses the same name to create data volumes on each node of a cluster. As shown in the following figure.



Data Volume List

Help: Data volume guide

Cluster: test-link

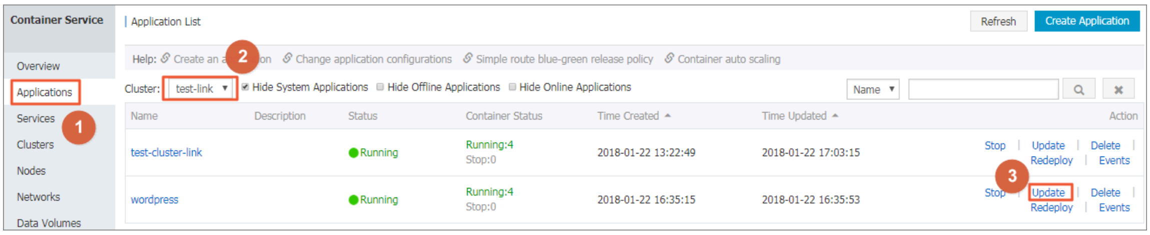
□ Node	Volume Name	Driver	Mount Point	Container	Volume Parameters	Action
□ [redacted]	fd23b180206446033b0e5d2c...	Ephemeral Disk	/var/lib/docker/volumes/...	wordpress_web_1		Delete All Volumes with the Same Name
□ [redacted]	8c1517c3b3414d605c839649...	Ephemeral Disk	/var/lib/docker/volumes/...	test-cluster-link_redis_...		Delete All Volumes with the Same Name
□ [redacted]	f91423c7345bbc3cd7c09c78...	Ephemeral Disk	/var/lib/docker/volumes/...	wordpress_web_1		Delete All Volumes with the Same Name
□ [redacted]	wp_upload	OSS File System	/mnt/acs_mnt/ossfs/cjlte...		View	Delete All Volumes with the Same Name
□ [redacted]	775c1dd987160e6e512ad64c...	Ephemeral Disk	/var/lib/docker/volumes/...	wordpress_web_3		Delete All Volumes with the Same Name
□ [redacted]	a03bbbe91cd847704654cc65...	Ephemeral Disk	/var/lib/docker/volumes/...	wordpress_web_3		Delete All Volumes with the Same Name
□ [redacted]	wp_upload	OSS File System	/mnt/acs_mnt/ossfs/cjlte...		View	Delete All Volumes with the Same Name
□ [redacted]	0dac5db2abc0c71b8c8eb8f4...	Ephemeral Disk	/var/lib/docker/volumes/...	wordpress_db_1		Delete All Volumes with the Same Name
□ [redacted]	b741328d5f69fc781d5cebd7...	Ephemeral Disk	/var/lib/docker/volumes/...	wordpress_db_1		Delete All Volumes with the Same Name
□ [redacted]	76fcf1bb0f767d57d7253d52...	Ephemeral Disk	/var/lib/docker/volumes/...	wordpress_web_2		Delete All Volumes with the Same Name
□ [redacted]	44aa4d32f723834b800d7790...	Ephemeral Disk	/var/lib/docker/volumes/...	wordpress_web_2		Delete All Volumes with the Same Name
□ [redacted]	wp_upload	OSS File System	/mnt/acs_mnt/ossfs/cjlte...		View	Delete All Volumes with the Same Name

2. Use the OSSFS data volumes.

   The WordPress attachments are stored in the `/var/www/html/wp-content/uploads` directory by default. In this example, map OSSFS data volumes to this directory and then an OSS bucket can be shared across different WordPress containers.

a. Log on to the Container Service console. Under Swarm, Click Applications in the left-side navigation pane.

b. Select the cluster used in this example from the Cluster drop-down list. Click Update at the right of the application wordpress created in this example.

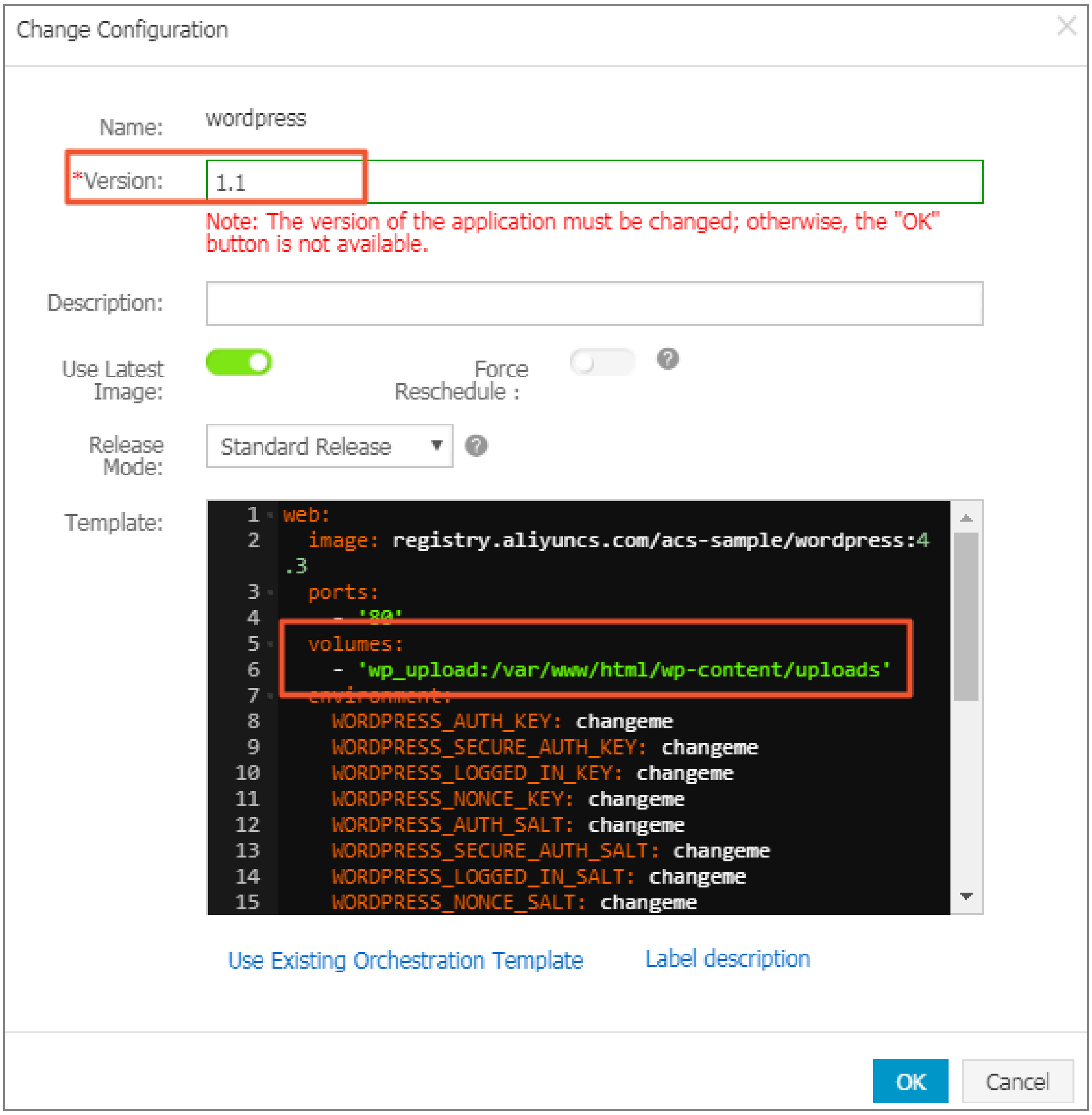


Name	Description	Status	Container Status	Time Created	Time Updated	Action
test-cluster-link		● Running	Running:4 Stop:0	2018-01-22 13:22:49	2018-01-22 17:03:15	Stop | Update | Delete | Redeploy | Events
wordpress		● Running	Running:4 Stop:0	2018-01-22 16:35:15	2018-01-22 16:35:53	Stop | Update | Delete | Redeploy | Events

c. In the Template field, add the mapping from OSSFS data volumes to the WordPress directory.

Note:

You must modify the Version. Otherwise, the application cannot be redeployed.



Name: wordpress

*Version: 1.1

Note: The version of the application must be changed; otherwise, the "OK" button is not available.

Description:

Use Latest Image: on

Force Reschedule : off

Release Mode: Standard Release

Template:

```
web:
  image: registry.aliyuncs.com/acs-sample/wordpress:4.3
  ports:
    - '80'
  volumes:
    - 'wp_upload:/var/www/html/wp-content/uploads'
  environment:
    WORDPRESS_AUTH_KEY: changeme
    WORDPRESS_SECURE_AUTH_KEY: changeme
    WORDPRESS_LOGGED_IN_KEY: changeme
    WORDPRESS_NONCE_KEY: changeme
    WORDPRESS_AUTH_SALT: changeme
    WORDPRESS_SECURE_AUTH_SALT: changeme
    WORDPRESS_LOGGED_IN_SALT: changeme
    WORDPRESS_NONCE_SALT: changeme
```

d. Click OK to redeploy the application.

3. Open WordPress and upload attachments. Then, you can see the uploaded attachments in the OSS bucket.

5 Use Docker Compose to test cluster network connectivity

This document provides a simple Compose file used to realize one-click deployment and you can test the container network connectivity by visiting the service access endpoint.

Scenarios

When deploying interdependent applications in a Docker cluster, you must make sure that the applications can access each other to realize cross-host container network connectivity. However, sometimes containers on different hosts cannot access each other due to network problems. If this happens, it is difficult to troubleshoot the problem. Therefore, an easy-to-use Compose file can be used to test the connectivity among cross-host containers within a cluster.

Solutions

Use the provided image and orchestration template to test the connectivity among containers.

```
web :
  image :  registry . aliyuncs . com / xianlu / test - link
  command :  python   test - link . py
  restart :  always
  ports :
     -  5000
  links :
     -  redis
  labels :
      aliyun . scale : ' 3 '
      aliyun . routing . port_5000 :  test - link ;
redis :
  image :  redis
  restart :  always
```

This example uses Flask to test the container connectivity.

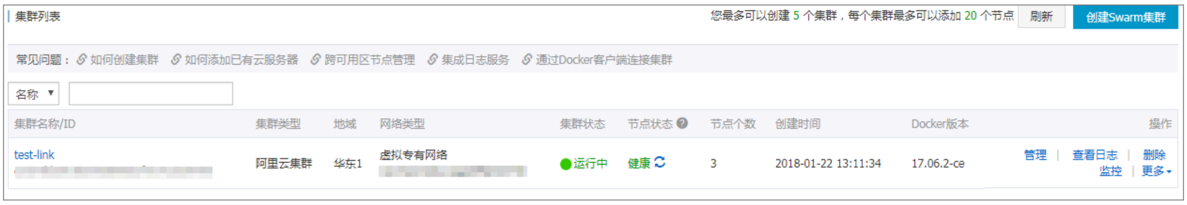
The preceding orchestration template deploys a Web service and a Redis service. The Web service contains three Flask containers and these three containers will be evenly distributed to three nodes when started. The three containers are on different hosts and the current network can realize cross-host container connectivity if the containers can ping each other. The Redis service runs on one of the three nodes . When started, each Flask container registers to the Redis service and reports the container IP address. The Redis service has the IP addresses of all the containers in the cluster after the three Flask containers are all started. When you access any of the three Flask containers, the container will send ping command to the other two containers and you can check the network connectivity of the cluster according to the ping command response.

Procedure

1. Create a cluster which contains three nodes.

   In this example, the cluster name is test-link. For how to create a cluster, see Create a cluster.

   > Note:
   >
   > Select to create a Server Load Balancer instance when creating the cluster.

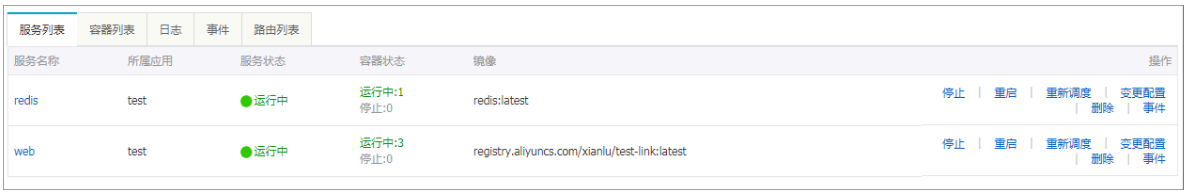
   

集群名称/ID	集群类型	地域	网络类型	集群状态	节点状态	节点个数	创建时间	Docker版本	操作
test-link [redacted]	阿里云集群	华东1	虚拟专有网络 [redacted]	● 运行中	健康	3	2018-01-22 13:11:34	17.06.2-ce	管理 | 查看日志 | 删除 监控 | 更多

2. Use the preceding template to create an application (in this example, the application name is test-cluster-link) to deploy the web service and redis service.

   For how to create an application, see Create an application.

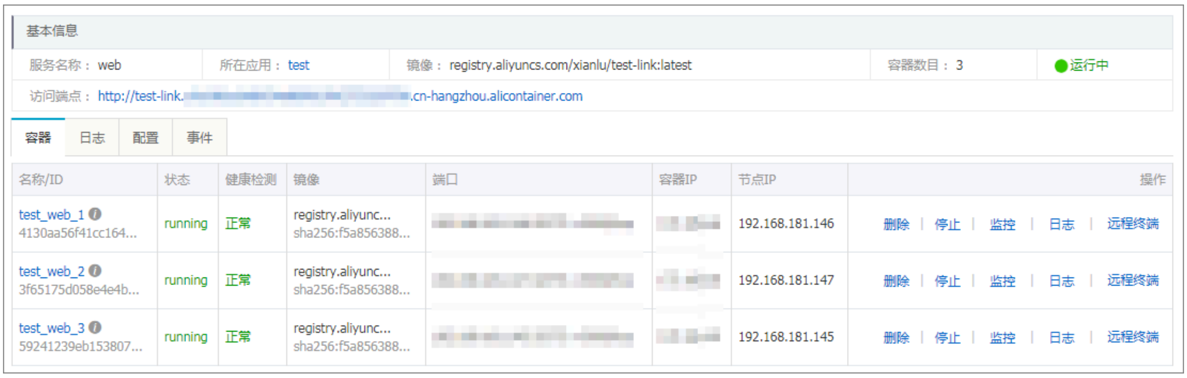
3. On the Application List page, click the application name to view the created services.

服务名称	所属应用	服务状态	容器状态	镜像	操作
redis	test	● 运行中	运行中:1 停止:0	redis:latest	停止 | 重启 | 重新调度 | 变更配置 | 删除 | 事件
web	test	● 运行中	运行中:3 停止:0	registry.aliyuncs.com/xianlu/test-link:latest	停止 | 重启 | 重新调度 | 变更配置 | 删除 | 事件

4. Click the name of the web service to enter the service details page.

   You can see that the three containers (test-cluster-link_web_1, test-cluster-link_web_2, and test-cluster-link_web_3) are all started and distributed on different nodes.



基本信息

服务名称： web | 所在应用： test | 镜像： registry.aliyuncs.com/xianlu/test-link:latest | 容器数目： 3 | ●运行中

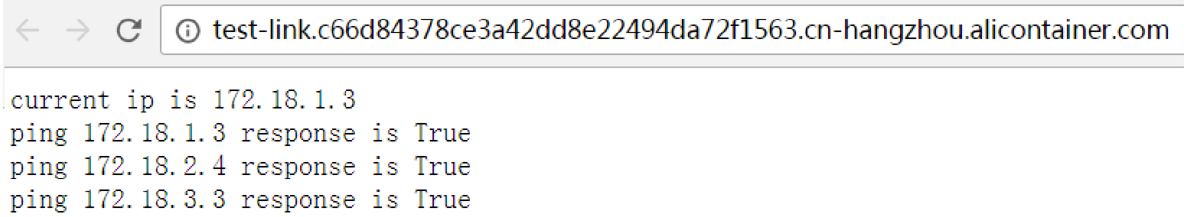
访问端点： http://test-link.[redacted].cn-hangzhou.alicontainer.com

容器 | 日志 | 配置 | 事件

名称/ID	状态	健康检测	镜像	端口	容器IP	节点IP	操作
test_web_1 4130aa56f41cc164...	running	正常	registry.aliyunc... sha256:f5a856388...	[redacted]	[redacted]	192.168.181.146	删除 | 停止 | 监控 | 日志 | 远程终端
test_web_2 3f65175d058e4e4b...	running	正常	registry.aliyunc... sha256:f5a856388...	[redacted]	[redacted]	192.168.181.147	删除 | 停止 | 监控 | 日志 | 远程终端
test_web_3 59241239eb153807...	running	正常	registry.aliyunc... sha256:f5a856388...	[redacted]	[redacted]	192.168.181.145	删除 | 停止 | 监控 | 日志 | 远程终端

5. Visit the access endpoint of the web service.

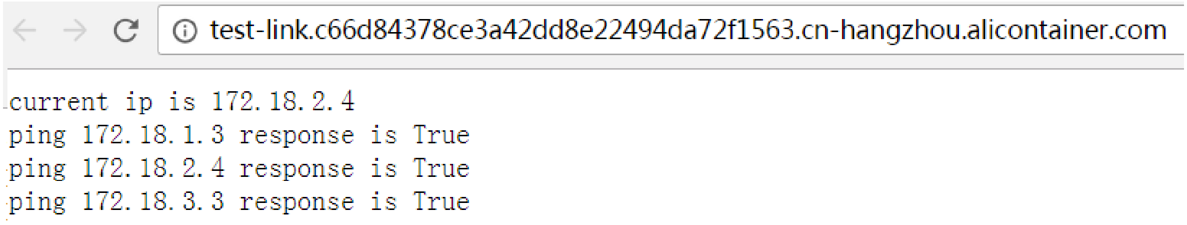
   As shown in the following figure, the container test-cluster-link_web_1 can access the container test-cluster-link_web_2 and container test-cluster-link_web_3.



test-link.c66d84378ce3a42dd8e22494da72f1563.cn-hangzhou.alicontainer.com

```
current ip is 172.18.1.3
ping 172.18.1.3 response is True
ping 172.18.2.4 response is True
ping 172.18.3.3 response is True
```

   Refresh the page. As shown in the following figure, the container test-cluster-link_web_2 can access the container test-cluster-link_web_1 and container test-cluster-link_web_3.



test-link.c66d84378ce3a42dd8e22494da72f1563.cn-hangzhou.alicontainer.com

```
current ip is 172.18.2.4
ping 172.18.1.3 response is True
ping 172.18.2.4 response is True
ping 172.18.3.3 response is True
```

   As the preceding results show, the containers in the cluster can access each other.

6 Log

6.1 Use ELK in Container Service

Background

Logs are an important component of the IT system.

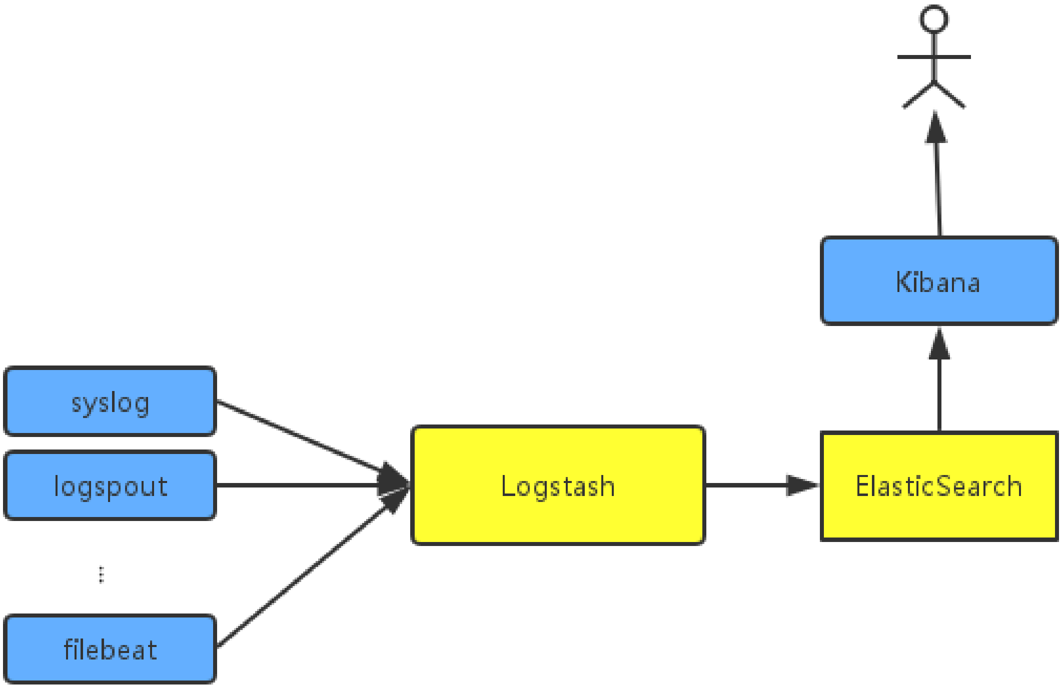
They record system events and the time when the events occur. We can troubleshoot system faults according to the logs and make statistical analysis.

Logs are usually stored in the local log files. To view logs, log on to the machine and filter keywords by using grep or other tools. However, when the application is deployed on multiple machines, viewing logs in this way is inconvenient. To locate the logs for a specific error, you have to log on to all the machines and filter files one after another. That is why concentrated log storage has emerged. All the logs are collected in Log Service and you can view and search for logs in Log Service.

In the Docker environment, concentrated log storage is even more important. Compared with the traditional operation and maintenance mode, Docker usually uses the orchestration system to manage containers. The mapping between container and host is not fixed and containers might be constantly migrated between hosts. You cannot view the logs by logging on to the machine and the concentrated log becomes the only choice.

Container Service integrates with Alibaba Cloud Log Service and automatically collects container logs to Log Service by using declarations. However, some users might prefer the This document introduces how to use ELK in Container Service. ELK (Elasticsearch+ Logstash+ Kibana) combination. This document introduces how to use ELK in Container Service.

Overall structure



An independent Logstash cluster must be deployed. Logsteins are heavy and resource -intensive, so they don't run logstroudsburg on every machine, not to mention every docker. To collect the container logs, syslog, Logspout, and filebeat are used. You might also use other collection methods.

To try to fit the actual scenario, two clusters are created here: one is the testelk cluster for deploying ELK, and the other is the app cluster for deploying applications.

Procedure

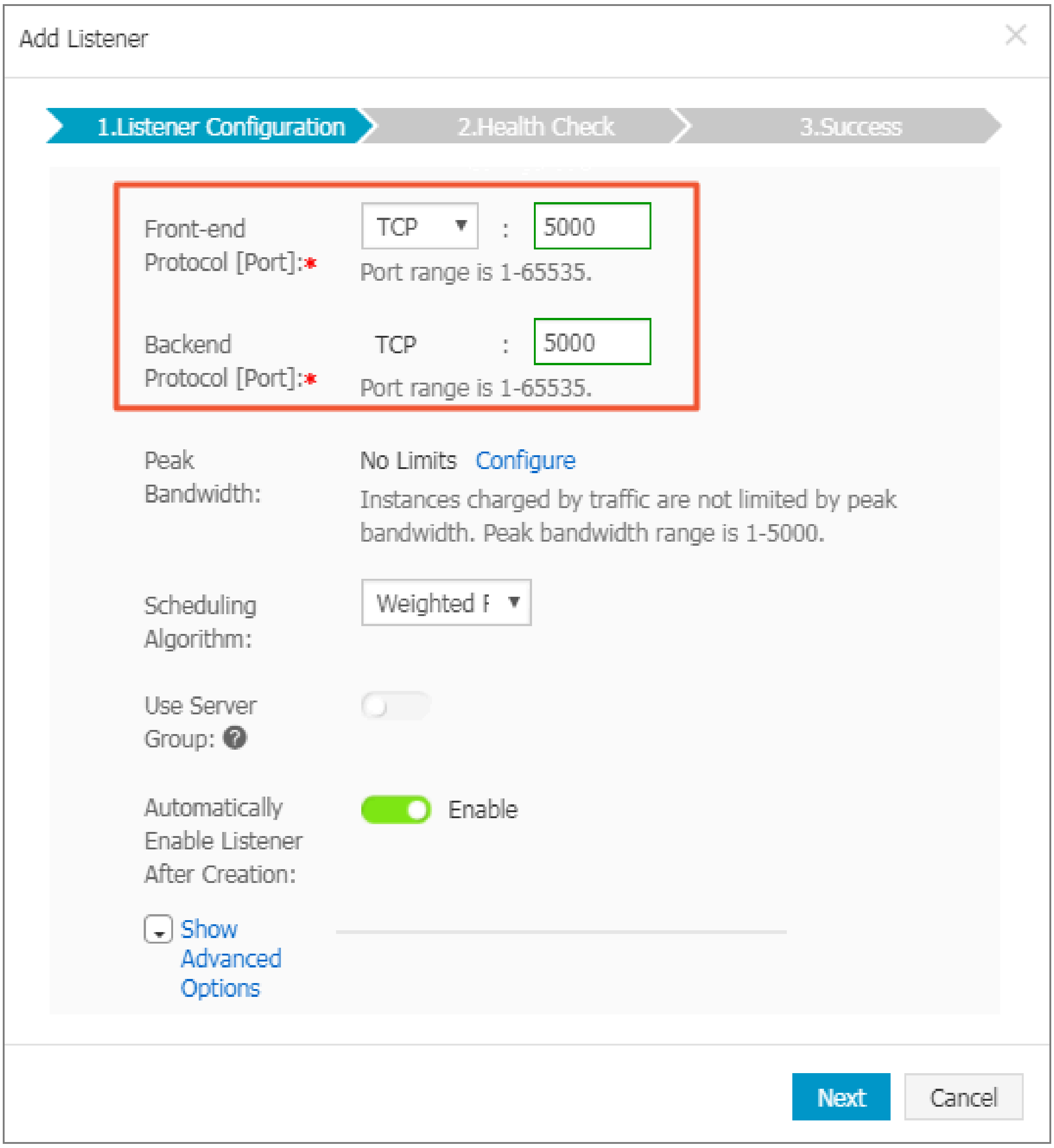
Note:
The clusters and Server Load Balancer instance created in this document must be in the same region.

Step 1. Create a Server Load Balancer instance

To enable other services to send logs to Logstash, create and configure a Server Load Balancer instance before configuring Logstash.

1. Log on to the Server Load Balancer console before creating an application.

2. Create a Server Load Balancer instance whose Instance type is Internet.
3. Add 2 listeners for the created Server Load Balancer instance. The frontend and backend port mappings of the 2 listeners are 5000: 5000 and 5044: 5044 respectively, with no backend server added.



Add Listener

1.Listener Configuration 2.Health Check 3.Success

Front-end Protocol [Port]:* TCP : 5000
Port range is 1-65535.

Backend Protocol [Port]:* TCP : 5000
Port range is 1-65535.

Peak Bandwidth: No Limits Configure
Instances charged by traffic are not limited by peak bandwidth. Peak bandwidth range is 1-5000.

Scheduling Algorithm: Weighted F

Use Server Group:

Automatically Enable Listener After Creation: Enable

Step 2. Deploy ELK

1. Log on to the Container Service console. Create a cluster named testelk.

   For how to create a cluster, see Create a cluster.

Note:
The cluster and the Server Load Balancer instance created in step 1 must be in the same region.

2. Bind the Server Load Balancer instance created in step 1 to this cluster.

   On the Cluster List page, Click Bind Server Load Balancer. Select the created Server Load Balancer instance from the Server Load Balancer ID list and then click OK. click Manage at the right of testelk. Click Load Balancer Settings in the left-side navigation pane. > Click Bind Server Load Balancer. Select the created Server Load Balancer instance from the Server Load Balancer ID list and then click OK.

3. Deploy ELK by using the following orchestration template. In this example, an application named elk is created.

   For how to create an application by using an orchestration template, see Create an application.

   Note:

   Replace `${ SLB_ID }` in the orchestration file with the ID of the Server Load Balancer instance created in step 1.

```
version : ' 2 '
 services :
   elasticsea  rch :
     image :  elasticsea  rch
   kibana :
     image :  kibana
     environmen  t :
       ELASTICSEA  RCH_URL :  http :// elasticsea  rch : 9200 /
     labels :
       aliyun . routing . port_5601 :  kibana
     links :
      -  elasticsea  rch
   logstash :
     image :  registry . cn - hangzhou . aliyuncs . com / acs -
sample / logstash
     hostname :  logstash
     ports :
      -  5044 : 5044
      -  5000 : 5000
     labels :
       aliyun . lb . port_5044 : ' tcp ://${ SLB_ID }: 5044 ' #
Create   a   Server   Load   Balancer   instance   first .
       aliyun . lb . port_5000 : ' tcp ://${ SLB_ID }: 5000 '
     links :
      -  elasticsea  rch
```

   In this orchestration file, the official images are used for Elasticsearch and Kibana, with no changes made. Logstash needs a configuration file, so make an image on your own to include the configuration file. The image source codes can be found in demo-logstash.

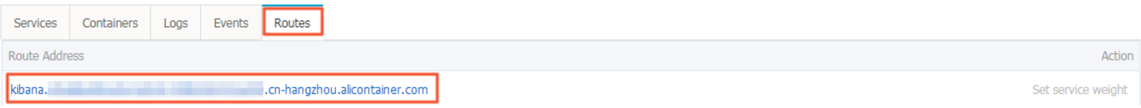
The Logstash configuration file is as follows. This is a simple Logstash configuration. Two input formats, syslog and filebeats, are provided and their external ports are 5044 and 5000 respectively.

```
input  {
     beats  {
         port  =>  5044
         type  =>  beats

     tcp  {
         port  =>  5000
         type  =>  syslog


 filter  {

 output  {
     elasticsea  rch  {
         hosts  => [" elasticsea  rch : 9200 "]

     stdout  {  codec  =>  rubydebug  }
```

4. Configure the Kibana index.

a. Access Kibana.

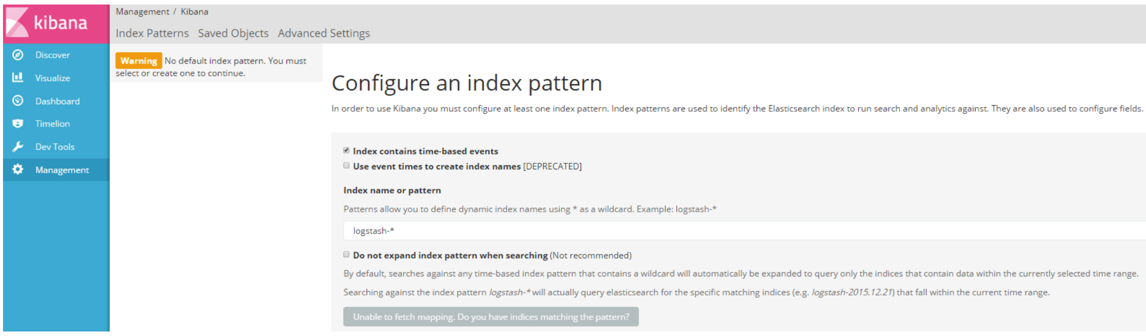
The URL can be found under the Routes tab of the application. On the Application List page, click the application name elk. Click the Routes tab and then click the route address to access Kibana.



kibana.[redacted].cn-hangzhou.alicontainer.com

b. Create an index.

Configure the settings as per your needs and then click Create.



Warning No default index pattern. You must select or create one to continue.

Configure an index pattern

In order to use Kibana you must configure at least one index pattern. Index patterns are used to identify the Elasticsearch index to run search and analytics against. They are also used to configure fields.

☑ Index contains time-based events

□ Use event times to create index names [DEPRECATED]

Index name or pattern

Patterns allow you to define dynamic index names using * as a wildcard. Example: logstash-*

logstash-*

□ Do not expand index pattern when searching (Not recommended)

By default, searches against any time-based index pattern that contains a wildcard will automatically be expanded to query only the indices that contain data within the currently selected time range.

Searching against the index pattern *logstash-** will actually query elasticsearch for the specific matching indices (e.g. *logstash-2015.12.21*) that fall within the current time range.

Unable to fetch mapping. Do you have indices matching the pattern?

Step 3. Collect logs

In Docker, the standard logs adopt Stdout file pointer. The following example first demonstrates how to collect Stdout to ELK. If you are using file logs, you can use filebeat directly. WordPress is used for the demonstration. The following is the orchestration template of WordPress. An application wordpress is created in another cluster.

1. Log on to the Container Service console. Create a cluster named app.

   For how to create a cluster, see Create a cluster.

   > Note:
   >
   > The cluster and the Server Load Balancer instance created in step 1 must be in the same region.

2. Create the application wordpress by using the following orchestration template.

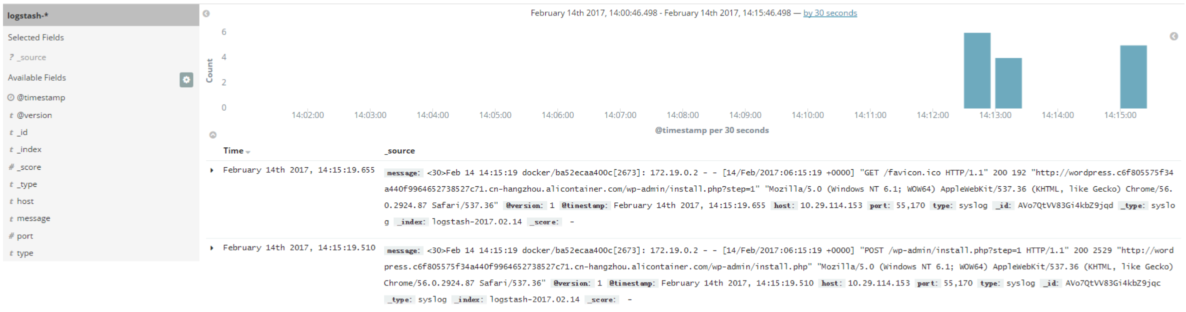
   > Note:
   >
   > Replace `${ SLB_IP }` in the orchestration file with the IP address of the Server Load Balancer instance created in step 1.

```
version : ' 2 '
 services :
   mysql :
     image :  mysql
     environmen  t :
      -  MYSQL_ROOT  _PASSWORD = password
   wordpress :
     image :  wordpress
     labels :
       aliyun . routing . port_80 :  wordpress
     links :
      - MySQL :  MySQL
     environmen  t :
      -  WORDPRESS_  DB_PASSWOR  D = password
     logging :
       driver :  syslog
       options :
         syslog - address : ' tcp ://${ SLB_IP }: 5000 '
```

   After the application is deployed successfully, click the application name wordpress on the Application List page. Click the Routes tab and then click the route address to access the WordPress application. click the application name wordpress on the Application List page. Click the Routes tab and then click the route address to access the WordPress application.

3. On the Application List page, click the application name elk. Click the Routes tab and then click the route address

   to access Kibana and view the collected logs.



```
Time    _source
February 14th 2017, 14:15:19.655    message: <30>Feb 14 14:15:19 docker/ba52ecaa400c[2673]: 172.19.0.2 - - [14/Feb/2017:06:15:19 +0000] "GET /favicon.ico HTTP/1.1" 200 192 "http://wordpress.c6f805575f34a440f9964652738527c71.cn-hangzhou.alicontainer.com/wp-admin/install.php?step=1" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/56.0.2924.87 Safari/537.36" @version: 1 @timestamp: February 14th 2017, 14:15:19.655 host: 10.29.114.153 port: 55,170 type: syslog _id: AVo7QtVV83Gi4kbZ9jqd _type: syslog _index: logstash-2017.02.14 _score: -
February 14th 2017, 14:15:19.510    message: <30>Feb 14 14:15:19 docker/ba52ecaa400c[2673]: 172.19.0.2 - - [14/Feb/2017:06:15:19 +0000] "POST /wp-admin/install.php?step=1 HTTP/1.1" 200 2529 "http://wordpress.c6f805575f34a440f9964652738527c71.cn-hangzhou.alicontainer.com/wp-admin/install.php" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/56.0.2924.87 Safari/537.36" @version: 1 @timestamp: February 14th 2017, 14:15:19.510 host: 10.29.114.153 port: 55,170 type: syslog _id: AVo7QtVV83Gi4kbZ9jqc _type: syslog _index: logstash-2017.02.14 _score: -
```

6.2 A new Docker log collection scheme: log-pilot

This document introduces a new log collection tool for Docker: log-pilot. Log-pilot is a log collection image we provide for you. You can deploy a log-pilot instance on each machine to collect all the Docker application logs. Docker of Linux version is supported, while Docker of Windows or Mac version is not supported.

Log-pilot has the following features:

- A separate log process collects the logs of all the containers on the machine. No need to start a log process for each container.
- Log-pilot supports file logs and stdout logs. Docker log driver or Logspout can only process stdout, while log-pilot supports collecting the stdout logs and the file logs.
- Declarative configuration. When your container has logs to collect, log-pilot will automatically collect logs of the new container if the path of the log file to be collected is declared by using the label. No other configurations need to be changed.
- Log-pilot supports multiple log storage methods and can deliver the logs to the correct location for powerful Alibaba Cloud Log Service, popular ElasticSearch combination, or even Graylog.
- Open-source. Log-pilot is fully open-sourced. You can download the codes from log-pilot GitHub project. If the current features cannot meet your requirements, welcome to raise an issue.

Quick start

See a simple scenario as follows: start a log-pilot and then start a Tomcat container, letting log-pilot collect Tomcat logs. For simplicity, here Alibaba Cloud Log Service or ELK is not involved. To run locally, you only need a machine that runs Docker.

First, start log-pilot.

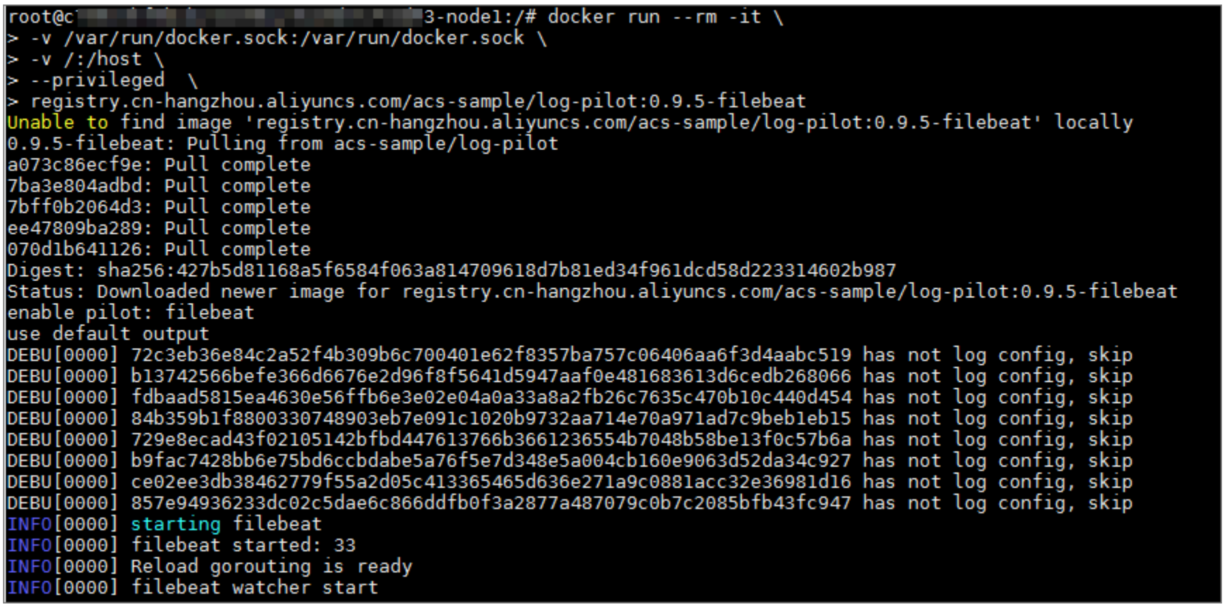
Note:

When log-pilot is started in this way, all the collected logs will be directly output to the console because no log storage is configured for backend use. Therefore, this method is mainly for debugging.

Open the terminal and enter the following commands:

```
docker   run  -- rm  - it  \
   - v  / var / run / docker . sock :/ var / run / docker . sock  \
   - v  /:/ host  \
   -- privileged  \
    registry . cn - hangzhou . aliyuncs . com / acs - sample / log -
pilot : 0 . 1
```

You will see the startup logs of log-pilot.



```
root@c[redacted]3-node1:/# docker run --rm -it \
> -v /var/run/docker.sock:/var/run/docker.sock \
> -v /:/host \
> --privileged  \
> registry.cn-hangzhou.aliyuncs.com/acs-sample/log-pilot:0.9.5-filebeat
Unable to find image 'registry.cn-hangzhou.aliyuncs.com/acs-sample/log-pilot:0.9.5-filebeat' locally
0.9.5-filebeat: Pulling from acs-sample/log-pilot
a073c86ecf9e: Pull complete
7ba3e804adbd: Pull complete
7bff0b2064d3: Pull complete
ee47809ba289: Pull complete
070d1b641126: Pull complete
Digest: sha256:427b5d81168a5f6584f063a814709618d7b81ed34f961dcd58d223314602b987
Status: Downloaded newer image for registry.cn-hangzhou.aliyuncs.com/acs-sample/log-pilot:0.9.5-filebeat
enable pilot: filebeat
use default output
DEBU[0000] 72c3eb36e84c2a52f4b309b6c700401e62f8357ba757c06406aa6f3d4aabc519 has not log config, skip
DEBU[0000] b13742566befe366d6676e2d96f8f5641d5947aaf0e481683613d6cedb268066 has not log config, skip
DEBU[0000] fdbaad5815ea4630e56ffb6e3e02e04a0a33a8a2fb26c7635c470b10c440d454 has not log config, skip
DEBU[0000] 84b359b1f8800330748903eb7e091c1020b9732aa714e70a971ad7c9beb1eb15 has not log config, skip
DEBU[0000] 729e8ecad43f02105142bfbd447613766b3661236554b7048b58be13f0c57b6a has not log config, skip
DEBU[0000] b9fac7428bb6e75bd6ccbdabe5a76f5e7d348e5a004cb160e9063d52da34c927 has not log config, skip
DEBU[0000] ce02ee3db38462779f55a2d05c413365465d636e271a9c0881acc32e36981d16 has not log config, skip
DEBU[0000] 857e94936233dc02c5dae6c866ddfb0f3a2877a487079c0b7c2085bfb43fc947 has not log config, skip
INFO[0000] starting filebeat
INFO[0000] filebeat started: 33
INFO[0000] Reload gorouting is ready
INFO[0000] filebeat watcher start
```

Do not close the terminal. Open a new terminal to start Tomcat. The Tomcat image is among the few Docker images that use stdout and file logs at the same time, and is suitable for the demonstration here.

```
 docker   run  - it  -- rm  - p   10080 : 8080  \
- v  / usr / local / tomcat / logs  \
-- label   aliyun . logs . catalina = stdout  \
-- label aliyun.logs.access=/usr/local/tomcat/logs/
 localhost_access_log.*.txt \
 tomcat
```

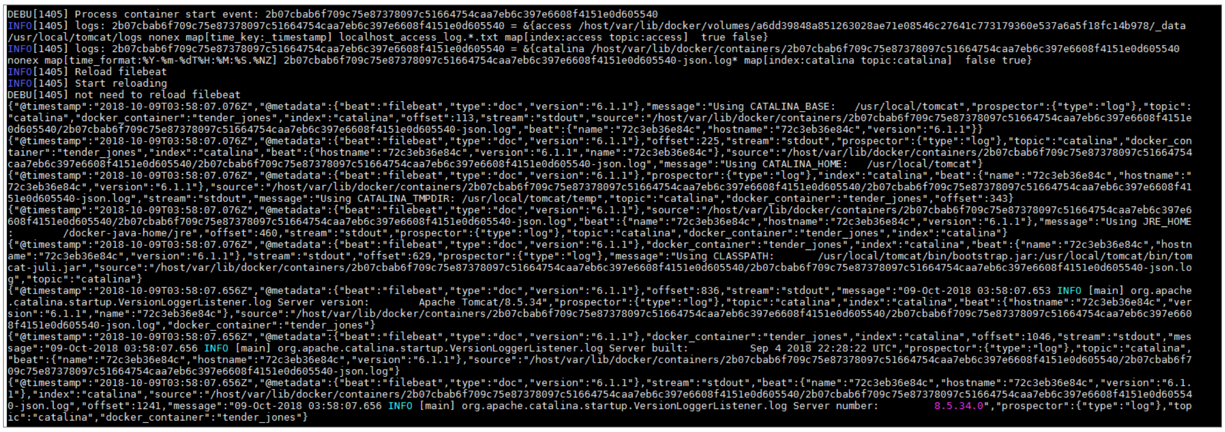
Note:

- `aliyun.logs.catalina=stdout` tells log-pilot that this container wants to collect stdout logs.
- `aliyun.logs.access=/usr/local/tomcat/logs/localhost_access_log.*.txt` indicates to collect all log files whose names comply with the *localhost_access_log.*.txt* format under the */usr/local/tomcat/logs/* directory in the container. The label usage will be introduced in details later.

Note:

If you deploy Tomcat locally, instead of in the Alibaba Cloud Container Service, specify `-v /usr/local/tomcat/logs`. Otherwise, log-pilot cannot read log files. Container Service has implemented the optimization and you do not need to specify `-v` on your own.

Log-pilot will monitor the events in the Docker container. When it finds any container with `aliyun.logs.xxx`, it will automatically parse the container configuration and start to collect the corresponding logs. After you start Tomcat, you will find many contents are output immediately by the log-pilot terminal, including the stdout logs output at the Tomcat startup, and some debugging information output by log-pilot itself.



```
DEBU[1405] Process container start event: 2b07cbab6f709c75e87378097c51664754caa7eb6c397e6608f4151e0d605540
INFO[1405] logs: 2b07cbab6f709c75e87378097c51664754caa7eb6c397e6608f4151e0d605540 = &{access /host/var/lib/docker/volumes/a6dd39848a851263028ae71e08546c27641c773179360e537a6a5f18fc14b978/_data
/usr/local/tomcat/logs nonex map[time_key:_timestamp] localhost_access_log.*.txt map[index:access topic:access]  true false}
INFO[1405] logs: 2b07cbab6f709c75e87378097c51664754caa7eb6c397e6608f4151e0d605540 = &{catalina /host/var/lib/docker/containers/2b07cbab6f709c75e87378097c51664754caa7eb6c397e6608f4151e0d605540
nonex map[time_format:%Y-%m-%dT%H:%M:%S.%NZ] 2b07cbab6f709c75e87378097c51664754caa7eb6c397e6608f4151e0d605540-json.log* map[index:catalina topic:catalina]  false true}
INFO[1405] Reload filebeat
INFO[1405] Start reloading
DEBU[1405] not need to reload filebeat
{"@timestamp":"2018-10-09T03:58:07.076Z","@metadata":{"beat":"filebeat","type":"doc","version":"6.1.1"},"message":"Using CATALINA_BASE:   /usr/local/tomcat","prospector":{"type":"log"},"topic":
"catalina","docker_container":"tender_jones","index":"catalina","offset":113,"stream":"stdout","source":"/host/var/lib/docker/containers/2b07cbab6f709c75e87378097c51664754caa7eb6c397e6608f4151e
0d605540/2b07cbab6f709c75e87378097c51664754caa7eb6c397e6608f4151e0d605540-json.log","beat":{"name":"72c3eb36e84c","hostname":"72c3eb36e84c","version":"6.1.1"}}
{"@timestamp":"2018-10-09T03:58:07.076Z","@metadata":{"beat":"filebeat","type":"doc","version":"6.1.1"},"offset":225,"stream":"stdout","prospector":{"type":"log"},"topic":"catalina","docker_con
tainer":"tender_jones","index":"catalina","beat":{"hostname":"72c3eb36e84c","version":"6.1.1","name":"72c3eb36e84c"},"source":"/host/var/lib/docker/containers/2b07cbab6f709c75e87378097c51664754
caa7eb6c397e6608f4151e0d605540/2b07cbab6f709c75e87378097c51664754caa7eb6c397e6608f4151e0d605540-json.log","message":"Using CATALINA_HOME:   /usr/local/tomcat"}
{"@timestamp":"2018-10-09T03:58:07.076Z","@metadata":{"beat":"filebeat","type":"doc","version":"6.1.1"},"prospector":{"type":"log"},"index":"catalina","beat":{"name":"72c3eb36e84c","hostname":"
72c3eb36e84c","version":"6.1.1"},"source":"/host/var/lib/docker/containers/2b07cbab6f709c75e87378097c51664754caa7eb6c397e6608f4151e0d605540/2b07cbab6f709c75e87378097c51664754caa7eb6c397e6608f41
51e0d605540-json.log","stream":"stdout","message":"Using CATALINA_TMPDIR: /usr/local/tomcat/temp","topic":"catalina","docker_container":"tender_jones","offset":343}
{"@timestamp":"2018-10-09T03:58:07.076Z","@metadata":{"beat":"filebeat","type":"doc","version":"6.1.1"},"source":"/host/var/lib/docker/containers/2b07cbab6f709c75e87378097c51664754caa7eb6c397e6
608f4151e0d605540/2b07cbab6f709c75e87378097c51664754caa7eb6c397e6608f4151e0d605540-json.log","beat":{"name":"72c3eb36e84c","hostname":"72c3eb36e84c","version":"6.1.1"},"message":"Using JRE_HOME
:        /docker-java-home/jre","offset":460,"stream":"stdout","prospector":{"type":"log"},"topic":"catalina","docker_container":"tender_jones","index":"catalina"}
{"@timestamp":"2018-10-09T03:58:07.076Z","@metadata":{"beat":"filebeat","type":"doc","version":"6.1.1"},"docker_container":"tender_jones","index":"catalina","beat":{"name":"72c3eb36e84c","hostn
ame":"72c3eb36e84c","version":"6.1.1"},"stream":"stdout","offset":629,"prospector":{"type":"log"},"message":"Using CLASSPATH:       /usr/local/tomcat/bin/bootstrap.jar:/usr/local/tomcat/bin/tom
cat-juli.jar","source":"/host/var/lib/docker/containers/2b07cbab6f709c75e87378097c51664754caa7eb6c397e6608f4151e0d605540/2b07cbab6f709c75e87378097c51664754caa7eb6c397e6608f4151e0d605540-json.lo
g","topic":"catalina"}
{"@timestamp":"2018-10-09T03:58:07.656Z","@metadata":{"beat":"filebeat","type":"doc","version":"6.1.1"},"offset":836,"stream":"stdout","message":"09-Oct-2018 03:58:07.653 INFO [main] org.apache
.catalina.startup.VersionLoggerListener.log Server version:        Apache Tomcat/8.5.34","prospector":{"type":"log"},"topic":"catalina","index":"catalina","beat":{"hostname":"72c3eb36e84c","ver
sion":"6.1.1","name":"72c3eb36e84c"},"source":"/host/var/lib/docker/containers/2b07cbab6f709c75e87378097c51664754caa7eb6c397e6608f4151e0d605540/2b07cbab6f709c75e87378097c51664754caa7eb6c397e660
8f4151e0d605540-json.log","docker_container":"tender_jones"}
{"@timestamp":"2018-10-09T03:58:07.656Z","@metadata":{"beat":"filebeat","type":"doc","version":"6.1.1"},"docker_container":"tender_jones","index":"catalina","offset":1046,"stream":"stdout","mes
sage":"09-Oct-2018 03:58:07.656 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Server built:          Sep 4 2018 22:28:22 UTC","prospector":{"type":"log"},"topic":"catalina",
"beat":{"name":"72c3eb36e84c","hostname":"72c3eb36e84c","version":"6.1.1"},"source":"/host/var/lib/docker/containers/2b07cbab6f709c75e87378097c51664754caa7eb6c397e6608f4151e0d605540/2b07cbab6f7
09c75e87378097c51664754caa7eb6c397e6608f4151e0d605540-json.log"}
{"@timestamp":"2018-10-09T03:58:07.656Z","@metadata":{"beat":"filebeat","type":"doc","version":"6.1.1"},"stream":"stdout","beat":{"name":"72c3eb36e84c","hostname":"72c3eb36e84c","version":"6.1.
1"},"index":"catalina","source":"/host/var/lib/docker/containers/2b07cbab6f709c75e87378097c51664754caa7eb6c397e6608f4151e0d605540/2b07cbab6f709c75e87378097c51664754caa7eb6c397e6608f4151e0d60554
0-json.log","offset":1241,"message":"09-Oct-2018 03:58:07.656 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Server number:         8.5.34.0","prospector":{"type":"log"},"top
ic":"catalina","docker_container":"tender_jones"}
```

You can access the deployed Tomcat in the browser, and find that similar records are displayed on the log-pilot terminal every time you refresh the browser. Wherein, the contents after `message` are the logs collected from `/ usr / local / tomcat / logs / localhost_access_log . XXX . txt` .

Use ElasticSearch + Kibana

Deploy ElastichSearch + Kibana. See Use ELK in Container Service to deploy ELK in Alibaba Cloud Container Service, or deploy them directly on your machine by following the ElasticSearch/Kibana documents. This document assumes that you have deployed the two components.

If you are still running the log-pilot, close it first, and then start it again by using the following commands:

Note:
Before running the following commands, replace the two variables `ELASTICSEARCH_HOST` and `ELASTICSEARCH_PORT` with the actual values you are using. `ELASTICSEARCH_PORT` is generally 9200.

```
docker  run  -- rm  - it  \
   - v  / var / run / docker . sock :/ var / run / docker . sock  \
   - v  /:/ host  \
   -- privileged  \
   - e   FLUENTD_OUTPUT = elasticsearch  \
   - e   ELASTICSEARCH_HOST =${ ELASTICSEARCH_HOST } \
   - e   ELASTICSEARCH_PORT =${ ELASTICSEARCH_PORT }
    registry . cn - hangzhou . aliyuncs . com / acs - sample / log -
pilot : 0 . 1
```

Compared with the previous log-pilot startup method, here three environment variables are added:

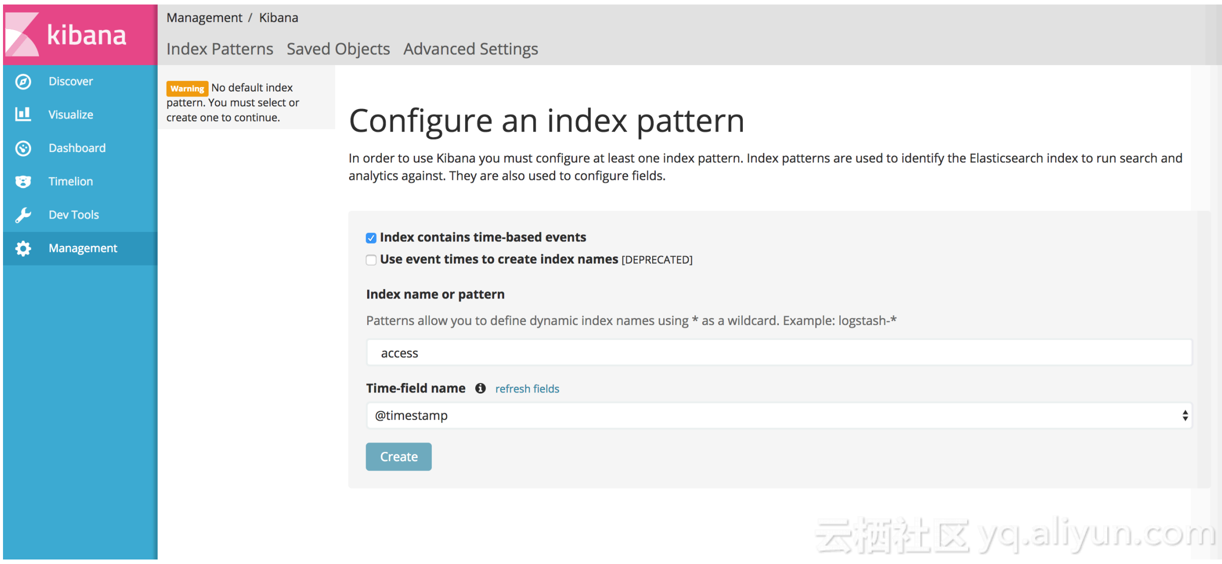
- `FLUENTD_OUTPUT = elasticsearch` : Send the logs to ElasticSearch.
- `ELASTICSEARCH_HOST =${ ELASTICSEARCH_HOST }`: The domain name of ElasticSearch.
- `ELASTICSEARCH_PORT =${ ELASTICSEARCH_PORT }`: The port number of ElasticSearch.

Continue to run the Tomcat started previously, and access it again to make Tomcat generate some logs. All these newly generated logs will be sent to ElasticSearch.

Open Kibana, and no new logs are visible yet. Create an index first. Log-pilot will write logs to the specific index of ElasticSearch. The rules are as follows:

If label `aliyun . logs . tags` is used in the application, and `tags` contains `target` , use `target` as the index of ElasticSearch. Otherwise, use `XXX` in the label `aliyun . logs . XXX` as the index.

In the previous example about Tomcat, the label `aliyun . logs . tags` is not used, so `access` and `catalina` are used by default as the index. First create the index `access` .



Configure an index pattern

In order to use Kibana you must configure at least one index pattern. Index patterns are used to identify the Elasticsearch index to run search and analytics against. They are also used to configure fields.

☑ Index contains time-based events

□ Use event times to create index names [DEPRECATED]

Index name or pattern

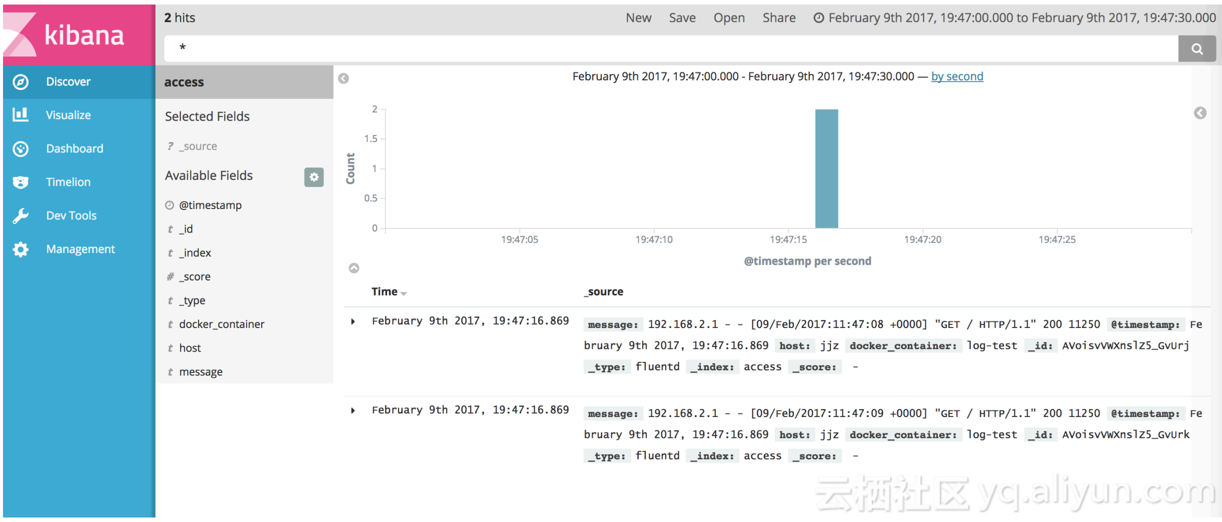
Patterns allow you to define dynamic index names using * as a wildcard. Example: logstash-*

access

Time-field name refresh fields

@timestamp

After the index is created, you can view the logs.



```
Time          _source
February 9th 2017, 19:47:16.869   message: 192.168.2.1 - - [09/Feb/2017:11:47:08 +0000] "GET / HTTP/1.1" 200 11250 @timestamp: Fe
                                  bruary 9th 2017, 19:47:16.869 host: jjz docker_container: log-test _id: AVoisvVWXnslZ5_GvUrj
                                  _type: fluentd _index: access _score: -
February 9th 2017, 19:47:16.869   message: 192.168.2.1 - - [09/Feb/2017:11:47:09 +0000] "GET / HTTP/1.1" 200 11250 @timestamp: Fe
                                  bruary 9th 2017, 19:47:16.869 host: jjz docker_container: log-test _id: AVoisvVWXnslZ5_GvUrk
                                  _type: fluentd _index: access _score: -
```

Use log-pilot in Alibaba Cloud Container Service

Container Service makes some special optimization for log-pilot, which adapts to running log-pilot best.

To run log-pilot in Container Service, create an application by using the following orchestration file. For how to create an application, see Create an application.

```
pilot :
  image :  registry . cn - hangzhou . aliyuncs . com / acs - sample /
log - pilot : 0 . 1
  volumes :
   - / var / run / docker . sock :/ var / run / docker . sock
   - /:/ host
  privileged :  true
  environmen  t :
    FLUENTD_OU  TPUT :  elasticsea  rch  # Replace   based   on
your   requiremen  ts
    ELASTICSEA  RCH_HOST : ${ elasticsea  rch } # Replace   based
on   your   requiremen  ts
    ELASTICSEA  RCH_PORT :  9200
  labels :
    aliyun . global :  true
```

Then, you can use the `aliyun . logs . xxx` label on the application that you want to collect logs.

Label description

When Tomcat is started, the following two labels are declared to tell log-pilot the location of the container logs.

```
-- label   aliyun . logs . catalina = stdout
-- label   aliyun . logs . access =/ usr / local / tomcat / logs /
 localhost_  access_log . *. txt
```

You can also add more labels on the application container.

- `aliyun . logs .$ name  = $ path`
  - The variable `name` is the log name and can only contain 0–9, a–z, A–Z, and hyphens (-).
  - The variable `path` is the path of the logs to be collected. The path must specify the file, and cannot only be a directory. Wildcards are supported as part of the file name, for example, `/ var / log / he . log` and `/ var / log /*. log` are both correct. However, `/ var / log` is not valid because the path cannot be only a directory. `stdout` is a special value, indicating standard output.

- `aliyun . logs .$ name . format` : The log format. Currently, the following formats are supported.
  - none: Unformatted plain text.
  - json: JSON format. One complete JSON string in each line.
  - csv: CSV format.
- `aliyun . logs .$ name . tags` : The additional field added when the logs are reported. The format is `k1 = v1 , k2 = v2` . The key-value pairs are separated by commas, for example, `aliyun . logs . access . tags =" name = hello , stage = test "`. Then, the logs reported to the storage will contain the `name` field and the `stage` field.

  If ElasticSearch is used for log storage, the `target` tag will have a special meaning, indicating the corresponding index in ElasticSearch.

Log-pilot extension

For most users, the existing features of log-pilot can meet their requirements. If log-pilot cannot meet your requirements, you can:

- Submit an issue at https://github.com/AliyunContainerService/log-pilot.
- Directly change the codes and then raise the PR.

7 Health check of Docker containers

In a distributed system, the service availability is frequently checked by using the health check to avoid exceptions when being called by other services. Docker introduced native health check implementation after version 1.12. This document introduces the health check of Docker containers.

Process-level health check checks whether or not the process is alive and is the simplest health check for containers. Docker daemon automatically monitors the PID1 process in the container. If the `docker run` command specifies the restart policy, closed containers can be restarted automatically according to the restart policy. In many real scenarios, process-level health check alone is far from enough. For example, if a container process is still alive, but is locked by an app deadlock and fails to respond to user requests, such problems won't be discovered by process monitoring.

Kubernetes provides Liveness and Readness probes to check the container and its service health respectively. Alibaba Cloud Container Service also provides a similar Service health check.

Docker native health check capability

Docker introduced the native health check implementation after version 1.12. The health check configurations of an application can be declared in the Dockerfile. `The HEALTHCHECK` instruction declares the health check command that can be used to determine whether or not the service status of the container master process is normal. This can reflect the real status of the container.

`HEALTHCHECK` instruction format:

- `HEALTHCHECK [ option ] CMD < command >`: The command that sets the container health check.
- `HEALTHCHECK NONE`: If the basic image has a health check instruction, this line can be used to block it.

Note:

The `HEALTHCHECK` can only appear once in the Dockerfile. If multiple HEALTHCHECK instructions exist, only the last one takes effect.

Images built by using Dockerfiles that contain `HEALTHCHEC K` instructions can check the health status when instantiating Docker containers. Health check is started automatically after the container is started.

`HEALTHCHEC K` supports the following options:

- `-- interval =< interval >`: The time interval between two health checks. The default value is 30 seconds.
- `-- timeout =< interval >`: The timeout for running the health check command. The health check fails if the timeout is exceeded. The default value is 30 seconds.
- `-- retries =< number of times >`: The container status is regarded as unhealthy if the health check fails continuously for a specified number of times. The default value is 3.
- `-- start - period =< interval >`: The initialization time of application startup. Failed health check during the startup is not counted. The default value is 0 second (introduced since version 17.05).

The command after `HEALTHCHEC K [ option ] CMD` follows the same format as `ENTRYPOINT`, in either the shell or the exec format. The returned value of the command determines the success or failure of the health check:

- 0: Success.
- 1: Failure.
- 2: Reserved value. Do not use.

After a container is started, the initial status is `starting`. Docker Engine waits for a period of `interval` to regularly run the health check command. If the returned value of a single check is not 0 or the running lasts longer than the specified `timeout` time, the health check is considered as failed. If the health check fails continuously for `retries` times, the health status changes to `unhealthy`.

- If the health check succeeds once, Docker changes the container status back to Healthy.
- Docker Engine issues a health_status event if the container health status changes.

Assume that an image is a simple Web service. To enable health check to determine whether or not its Web service is working normally, `curl` can be used to help with the determination and the `HEALTHCHEC  K` instruction in its Dockerfile can be written as follows:

```
FROM   elasticsea  rch : 5 . 5
HEALTHCHEC  K  -- interval = 5s  -- timeout = 2s  -- retries = 12  \
  CMD   curl  -- silent  -- fail   localhost : 9200 / _cluster /
health  ||  exit   1
```

```
docker   build  - t   test / elasticsea  rch : 5 . 5  .
docker   run  -- rm  - d  \
   -- name = elasticsea  rch  \
    test / elasticsea  rch : 5 . 5
```

You can use `docker   ps`. After several seconds, the Elasticsearch container changes from the Starting status to Healthy status.

```
$  docker   ps
 CONTAINER   ID   IMAGE   COMMAND   CREATED   STATUS   PORTS   NAMES
 c9a6e68d4a  7f   test / elasticsea  rch : 5 . 5  "/ docker -
 entrypoin ..."  2   seconds   ago   Up   2   seconds   ( health :
 starting )  9200 / tcp ,  9300 / tcp   elasticsea  rch
$  docker   ps
 CONTAINER   ID   IMAGE   COMMAND   CREATED   STATUS   PORTS   NAMES
 c9a6e68d4a  7f   test / elasticsea  rch : 5 . 5  "/ docker -
 entrypoin ..."  14   seconds   ago   Up   13   seconds  ( healthy )
 9200 / tcp ,  9300 / tcp   elasticsea  rch
```

Another method is to directly specify the health check policy in the `docker   run` command.

```
$  docker   run  -- rm  - d  \
    -- name = elasticsea  rch  \
    -- health - cmd =" curl  -- silent  -- fail   localhost : 9200 /
 _cluster / health  ||  exit   1 " \
    -- health - interval = 5s  \
    -- health - retries = 12  \
    -- health - timeout = 2s  \
     elasticsea  rch : 5 . 5
```

To help troubleshoot the issue, all output results of health check commands (including stdout and stderr) are stored in health status and you can view them with the docker inspect command. Use the following commands to retrieve the health check results of the past five containers.

```
docker   inspect  -- format ='{{ json  .  State . Health }}'
elasticsea  rch
```

Or

```
docker   inspect   elasticsea  rch  |  jq  ".[].  State . Health "
```

The sample result is as follows:

```
{
  " Status ": " healthy ",
  " FailingStr  eak ":  0 ,
  " Log ": [
    {
      " Start ": " 2017 - 08 - 19T09 : 12 : 53 . 393598805Z ",
      " End ": " 2017 - 08 - 19T09 : 12 : 53 . 452931792Z ",
      " ExitCode ":  0 ,
      " Output ": "..."
    },
    ...
}
```

Generally, we recommend that you declare the corresponding health check policy in the Dockerfile to facilitate the use of images because application developers know better about the application SLA. The application deployment and Operation & Maintenance personnel can adjust the health check policies as needed for deployment scenarios by using the command line parameters and REST API.

The Docker community provides some instance images that contain health check. Obtain them in the following project: https://github.com/docker-library/healthcheck.

Note:

- Alibaba Cloud Container Service supports Docker native health check and Alibaba Cloud extension health check.
- Currently, Kubernetes does not support Docker native health check.

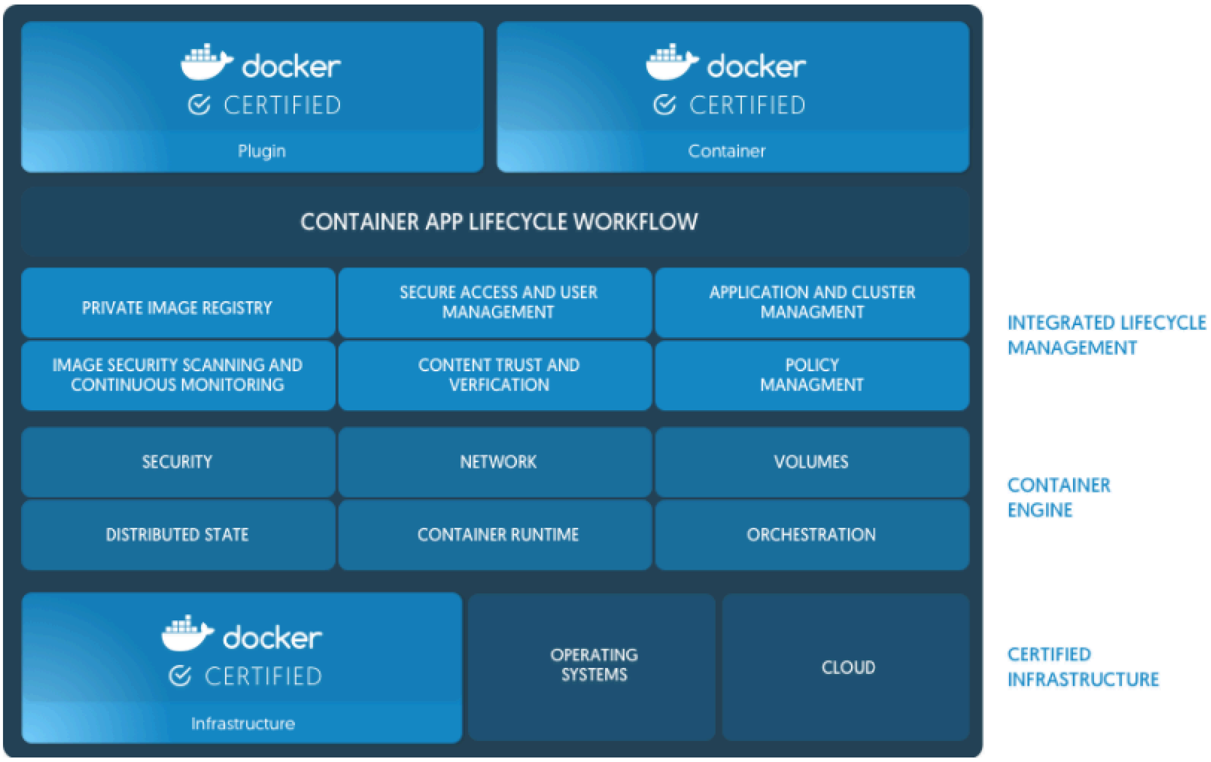
8 One-click deployment of Docker Datacenter

About DDC

Docker Datacenter (DDC) is an enterprise-level container management and service deployment package solution platform released by Docker. DDC is composed of the following three components:

- Docker Universal Control Plane (Docker UCP): A set of graphical management interfaces.
- Docker Trusted Registry (DTR): A trusted Docker image repository.
- Docker Engine Enterprise Edition: The Docker Engine providing technical support.

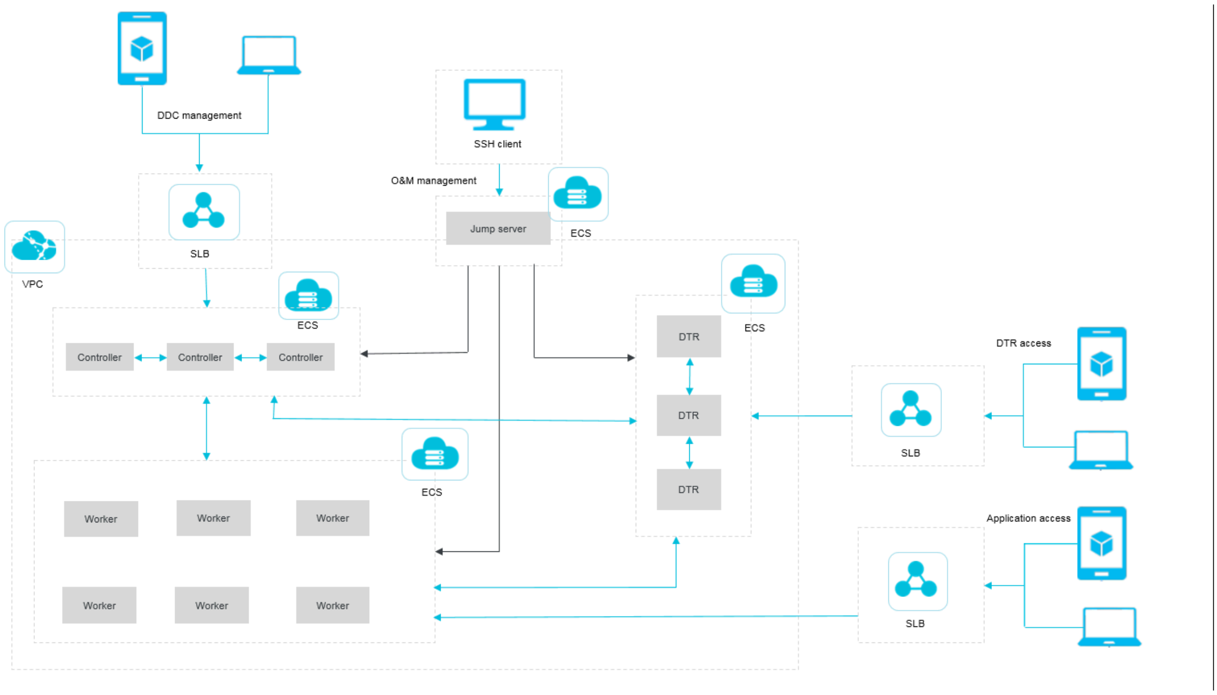
DDC is available on the Docker official website https://www.docker.com/products/docker-datacenter.


docker CERTIFIED Plugin
docker CERTIFIED Container
CONTAINER APP LIFECYCLE WORKFLOW
PRIVATE IMAGE REGISTRY
SECURE ACCESS AND USER MANAGEMENT
APPLICATION AND CLUSTER MANAGMENT
IMAGE SECURITY SCANNING AND CONTINUOUS MONITORING
CONTENT TRUST AND VERFICATION
POLICY MANAGMENT
SECURITY
NETWORK
VOLUMES
DISTRIBUTED STATE
CONTAINER RUNTIME
ORCHESTRATION
docker CERTIFIED Infrastructure
OPERATING SYSTEMS
CLOUD
INTEGRATED LIFECYCLE MANAGEMENT
CONTAINER ENGINE
CERTIFIED INFRASTRUCTURE

DDC is a counterpart of Docker Cloud, another online product of the Docker company . However, DDC primarily targets enterprise users for internal deployment. You can register your own Docker image to DTR and use UCP to manage the entire Docker cluster. Both components provide web interfaces.

You must purchase a license to use DDC, but the Docker company provides a free license for a one-month trial. You can download the trial license from the Docker official website after signing up.

DDC deployment architecture



In the preceding basic architecture figure, Controller primarily runs the UCP component, DTR runs the DTR component, and Worker primarily runs your own Docker service. The entire DDC environment is deployed on the Virtual Private Cloud (VPC) and all Elastic Compute Service (ECS) instances are in the same security group. Every component provides a Server Load Balancer instance for extranet access. Operations and maintenance are implemented by using the jump server. To enhance the availability, the entire DDC environment is deployed for high availability , meaning at least two Controllers and two DTRs exist.

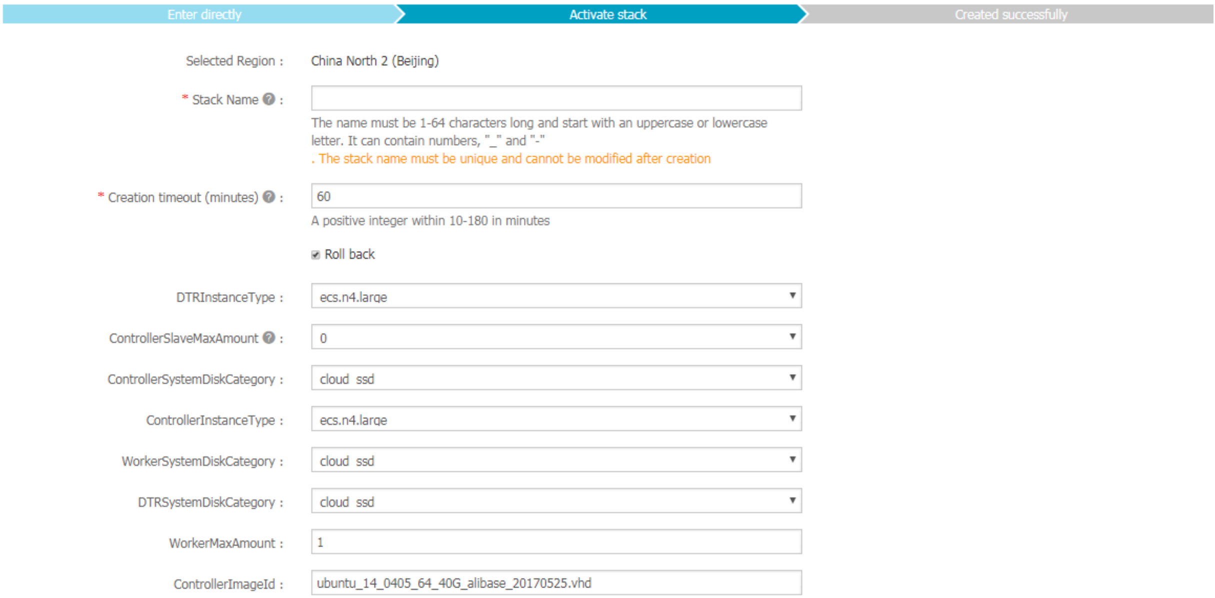
One-click deployment of DDC

You can use Alibaba Cloud Resource Orchestration Service (ROS) to deploy DDC in one click at the following link.

One-click deployment of DDC

In the preceding orchestration template, DDC is deployed in the region China North 2 (Beijing) by default. To change the region for deployment, click Back in the lower-right corner of the page. Select your region and then click Next.

Complete the configurations. Click Create to deploy a set of DDC.



Selected Region : China North 2 (Beijing)

* Stack Name ❓ :

The name must be 1-64 characters long and start with an uppercase or lowercase letter. It can contain numbers, "_" and "-"
. The stack name must be unique and cannot be modified after creation

* Creation timeout (minutes) ❓ : 60

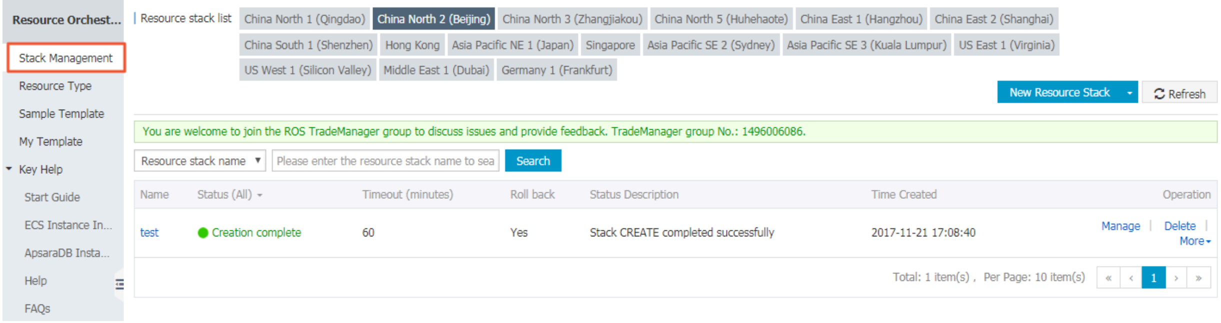
A positive integer within 10-180 in minutes

☑ Roll back

DTRInstanceType :	ecs.n4.large
ControllerSlaveMaxAmount ❓ :	0
ControllerSystemDiskCategory :	cloud_ssd
ControllerInstanceType :	ecs.n4.large
WorkerSystemDiskCategory :	cloud_ssd
DTRSystemDiskCategory :	cloud_ssd
WorkerMaxAmount :	1
ControllerImageId :	ubuntu_14_0405_64_40G_alibase_20170525.vhd

DDC access

After creating DDC successfully by using ROS, you can enter the ROS stack management page by clicking Stack Management in the left-side navigation pane. Find the created stack, and then click the stack name or Manage at the right of the stack. The Stack Overview page appears.



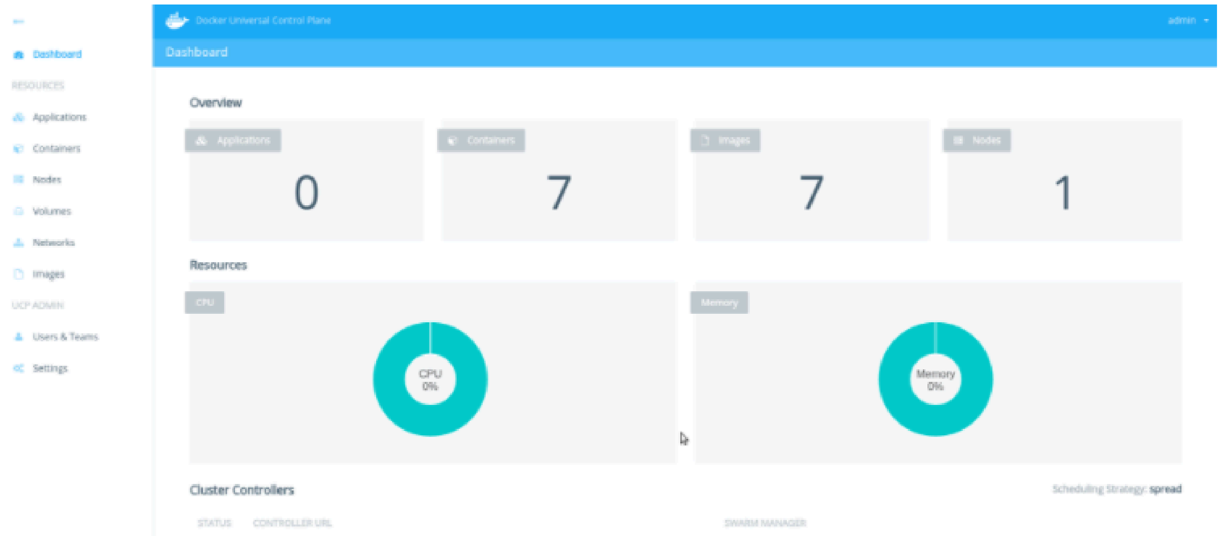
You are welcome to join the ROS TradeManager group to discuss issues and provide feedback. TradeManager group No.: 1496006086.

Name	Status (All)	Timeout (minutes)	Roll back	Status Description	Time Created	Operation
test	● Creation complete	60	Yes	Stack CREATE completed successfully	2017-11-21 17:08:40	Manage | Delete | More

Total: 1 item(s) , Per Page: 10 item(s) « ‹ 1 › »

You can view the addresses used to log on to UCP and DTR in the Output section.

Enter the UCP address in the browser and the UCP access page appears. Enter the administrator account and password created when installing UCP and the system prompts you to import the license file. Import the license file and then enter the UCP control interface.



Applications 0

Containers 7

Images 7

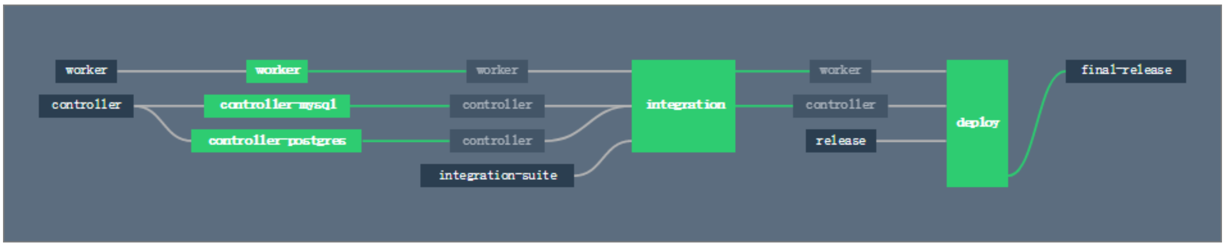
Nodes 1

Scheduling Strategy: spread

9 Build Concourse CI in Container Service in an easy way

Concourse CI, a CI/CD tool whose charm lies in the minimalist design, is widely applied to the CI/CD of each Cloud Foundry module. Concourse CI officially provides the standard Docker images and you can use Alibaba Cloud Container Service to deploy a set of Concourse CI applications rapidly.

Get to know the principle of Concourse if you are not familiar with the Concourse CI tool. For more information, see Concourse official website.

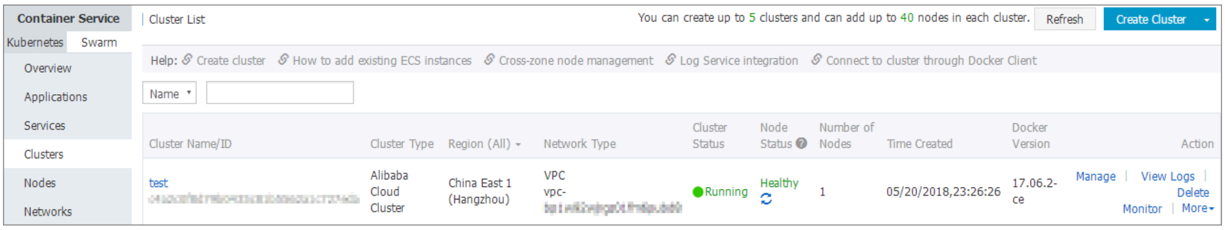


Create a swarm cluster

Log on to the Container Service console to create a cluster. In this example, create a swarm cluster with one node.

For how to create a cluster, see Create a cluster.

Note:
You must configure the external URL for Concourse, which allows you to access the Web service of Concourse from the current machine. Therefore, retain the Elastic IP (EIP) when creating a cluster.



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

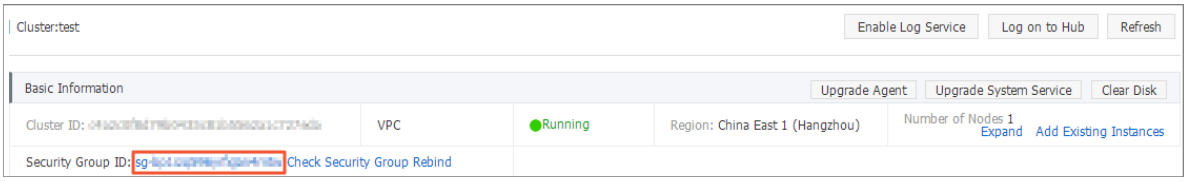
Help: Create cluster How to add existing ECS instances Cross-zone node management Log Service integration Connect to cluster through Docker Client

Cluster Name/ID	Cluster Type	Region (All)	Network Type	Cluster Status	Node Status	Number of Nodes	Time Created	Docker Version	Action
test [redacted]	Alibaba Cloud Cluster	China East 1 (Hangzhou)	VPC vpc-[redacted]	● Running	Healthy	1	05/20/2018,23:26:26	17.06.2-ce	Manage | View Logs | Delete Monitor | More

Configure security group rules

The Concourse component ATC listens to the port 8080 by default. Therefore, you must configure the inbound permissions of port 8080 for the cluster security group.

1. In the Container Service console, click Swarm > Clusters in the left-side navigation pane. Click Manage at the right of the created cluster.
2. On the Basic Information page, click the security group ID.

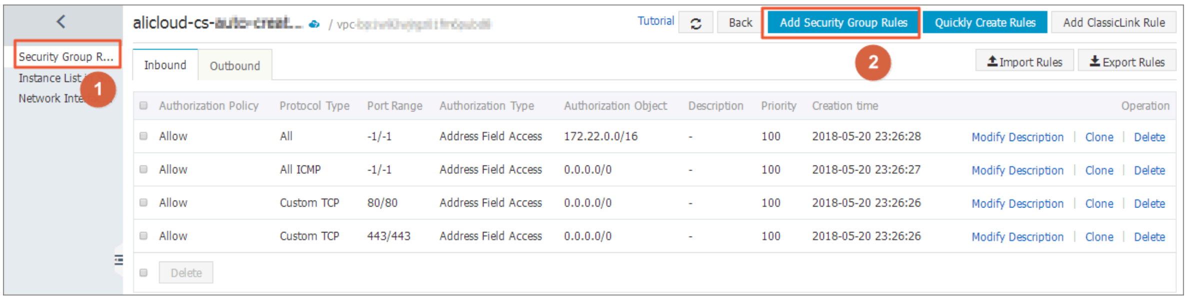


Cluster:test

Basic Information

Cluster ID: [redacted]	VPC	●Running	Region: China East 1 (Hangzhou)	Number of Nodes 1 Expand Add Existing Instances
Security Group ID: sg-[redacted] Check Security Group Rebind				

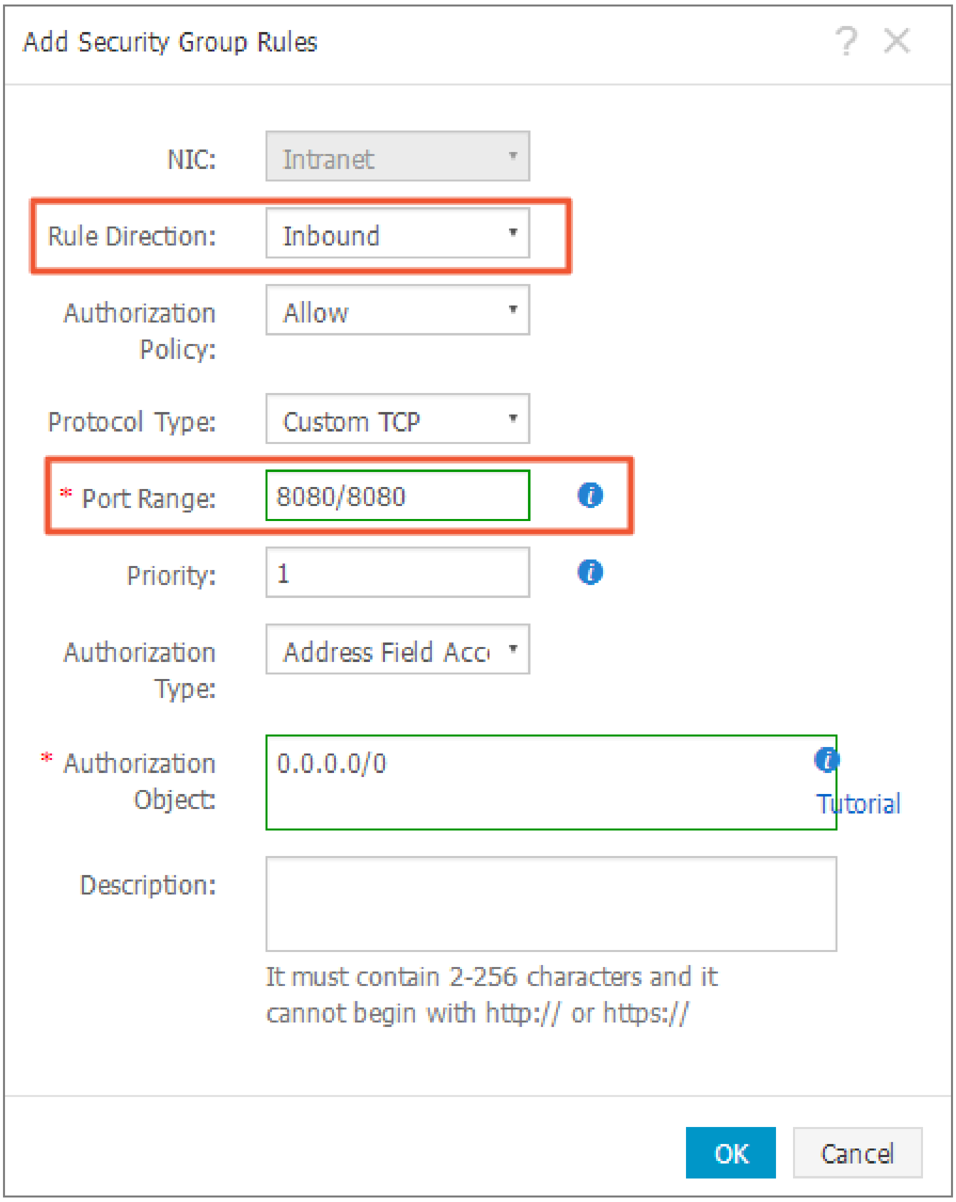
3. Click Security Group Rules in the left-side navigation pane. Click Add Security Group Rules in the upper-right corner.



□	Authorization Policy	Protocol Type	Port Range	Authorization Type	Authorization Object	Description	Priority	Creation time	Operation
□	Allow	All	-1/-1	Address Field Access	172.22.0.0/16	-	100	2018-05-20 23:26:28	Modify Description | Clone | Delete
□	Allow	All ICMP	-1/-1	Address Field Access	0.0.0.0/0	-	100	2018-05-20 23:26:27	Modify Description | Clone | Delete
□	Allow	Custom TCP	80/80	Address Field Access	0.0.0.0/0	-	100	2018-05-20 23:26:26	Modify Description | Clone | Delete
□	Allow	Custom TCP	443/443	Address Field Access	0.0.0.0/0	-	100	2018-05-20 23:26:26	Modify Description | Clone | Delete

□ Delete

4. Configure the inbound permissions of port 8080 for the security group and then click OK.



Add Security Group Rules

NIC: Intranet

Rule Direction: Inbound

Authorization Policy: Allow

Protocol Type: Custom TCP

* Port Range: 8080/8080

Priority: 1

Authorization Type: Address Field Acc

* Authorization Object: 0.0.0.0/0

Description:

It must contain 2-256 characters and it cannot begin with http:// or https://

Create keys in the ECS instance

You must generate three private keys for running Concourse safely.

1. Log on to the Elastic Compute Service (ECS) instance. In the root directory, create the directories *keys / web* and *keys / worker* . You can run the following command to create these two directories rapidly.

```
mkdir  - p   keys / web   keys / worker
```

2. Run the following commands to generate three private keys.

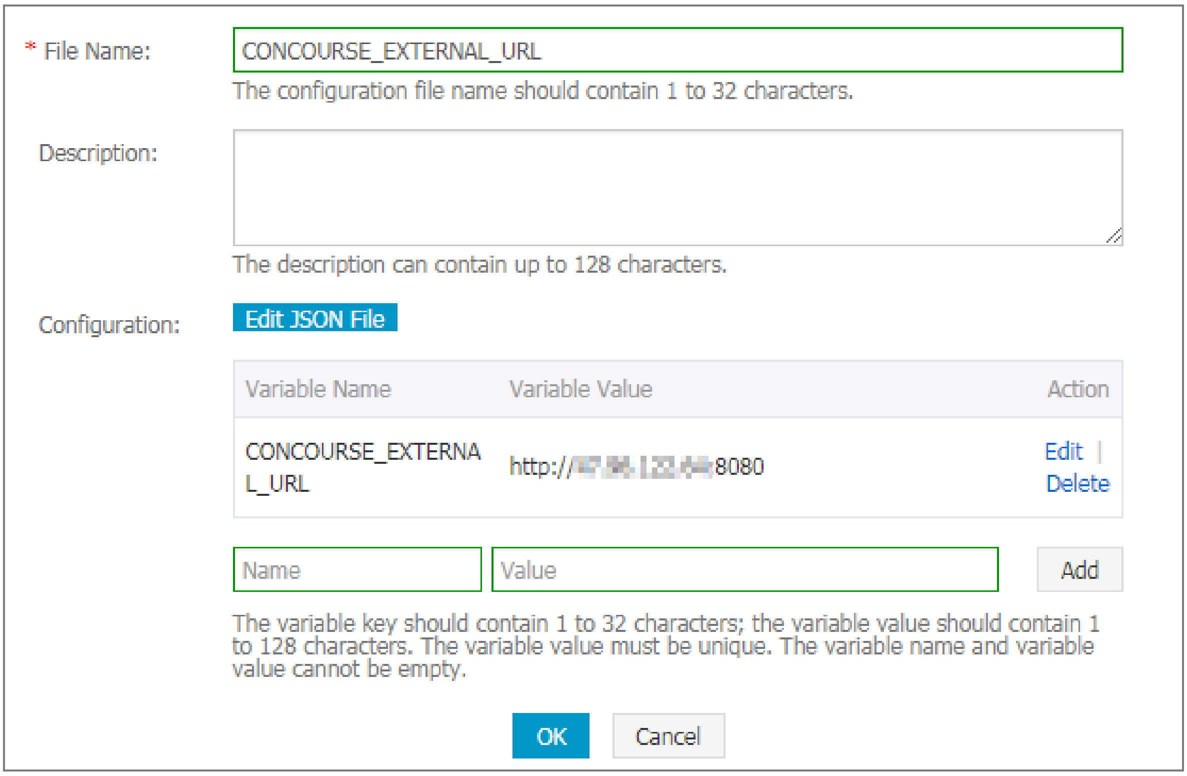
```
ssh - keygen  - t  rsa  - f  tsa_host_k ey  - N  ''
ssh - keygen  - t  rsa  - f  worker_key  - N  ''
ssh - keygen  - t   rsa  - f   session_si  gning_key  - N  ''
```

3. Copy the certificate to the corresponding directory.

```
cp  ./ keys / worker / worker_key . pub  ./ keys / web /
authorized  _worker_ke  ys
cp  ./ keys / web / tsa_host_k  ey . pub  ./ keys / worker
```

Deploy Concourse CI

1. Log on to the Container Service console.
2. Click Swarm > Configurations in the left-side navigation pane. Click Create in the upper-right corner. Enter CONCOURSE_EXTERNAL_URL as the Variable Name and `http :// your - ecs - public - ip : 8080` as the Variable Value.



* File Name: CONCOURSE_EXTERNAL_URL

The configuration file name should contain 1 to 32 characters.

Description:

The description can contain up to 128 characters.

Configuration: Edit JSON File

Variable Name	Variable Value	Action
CONCOURSE_EXTERNAL_URL	http://[redacted]8080	Edit | Delete

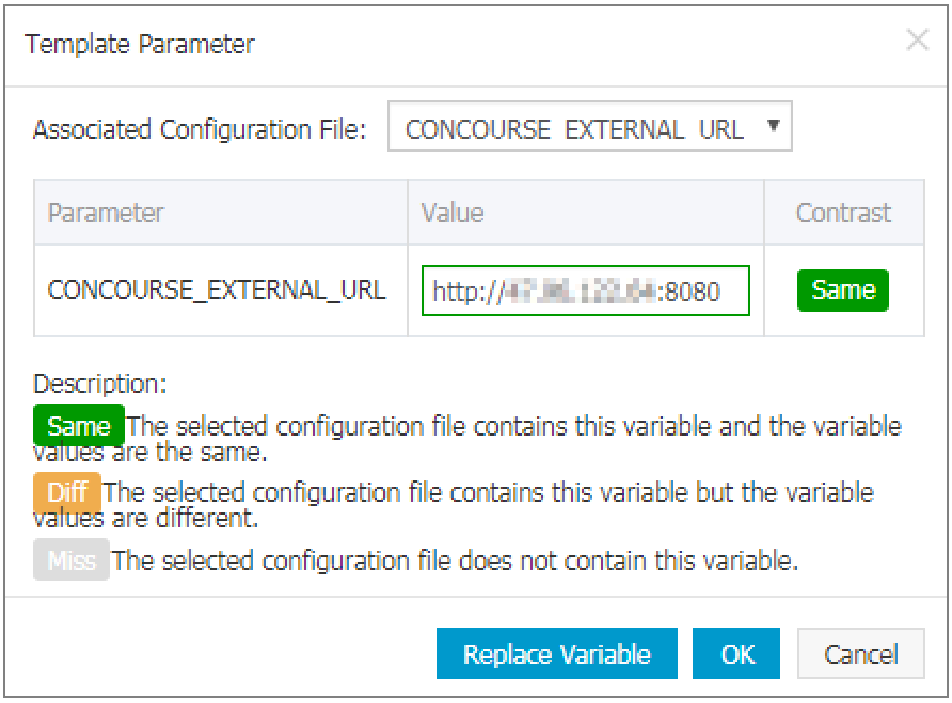
The variable key should contain 1 to 32 characters; the variable value should contain 1 to 128 characters. The variable value must be unique. The variable name and variable value cannot be empty.

3. Click Applications in the left-side navigation pane. Select the cluster used in this example from the Cluster drop-down list. Click Create Application in the upper-right corner.
4. Enter the basic information for the application you are about to create. Select Create with Orchestration Template. Use the following template:

```
version : ' 2 '
 services :
   concourse - db :
     image :  postgres : 9 . 5
     privileged :  true
     environmen  t :
       POSTGRES_D  B :  concourse
       POSTGRES_U  SER :  concourse
       POSTGRES_P  ASSWORD :  changeme
       PGDATA : / database
   concourse - web :
     image :  concourse / concourse
     links : [ concourse - db ]
     command :  web
     privileged :  true
     depends_on : [ concourse - db ]
     ports : [" 8080 : 8080 "]
     volumes : ["/ root / keys / web :/ concourse - keys "]
     restart :  unless - stopped  #  required   so   that   it
retries   until   conocourse - db   comes   up
     environmen  t :
       CONCOURSE_  BASIC_AUTH  _USERNAME :  concourse
       CONCOURSE_  BASIC_AUTH  _PASSWORD :  changeme
       CONCOURSE_  EXTERNAL_U  RL : "${ CONCOURSE_  EXTERNAL_U  RL
}"
       CONCOURSE_  POSTGRES_H  OST :  concourse - db
       CONCOURSE_  POSTGRES_U  SER :  concourse
       CONCOURSE_  POSTGRES_P  ASSWORD :  changeme
       CONCOURSE_  POSTGRES_D  ATABASE :  concourse
   concourse - worker :
     image :  concourse / concourse
     privileged :  true
     links : [ concourse - web ]
     depends_on : [ concourse - web ]
     command :  worker
     volumes : ["/ keys / worker :/ concourse - keys "]
     environmen  t :
       CONCOURSE_  TSA_HOST :  concourse - web
```

```
        dns: 8.8.8.8
```

5. Click Create and Deploy. The Template Parameter dialog box appears. Select the configuration file to be associated with from the `Associated Configuration File` drop-down list. Click Replace Variable and then click OK.



Template Parameter

Associated Configuration File: CONCOURSE_EXTERNAL_URL

Parameter	Value	Contrast
CONCOURSE_EXTERNAL_URL	http://[redacted]:8080	Same

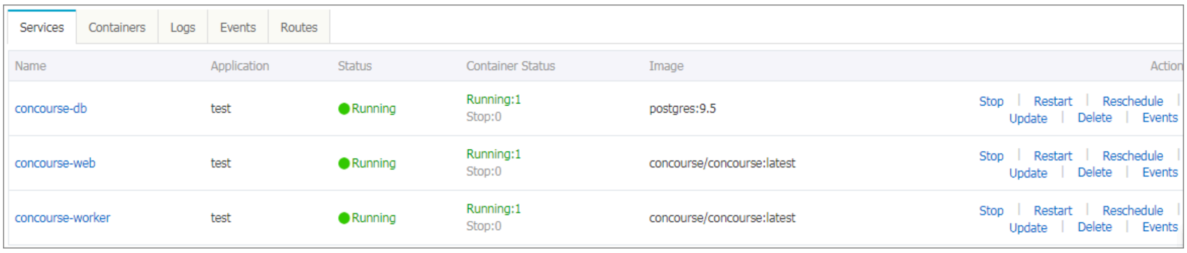
Description:

Same The selected configuration file contains this variable and the variable values are the same.

Diff The selected configuration file contains this variable but the variable values are different.

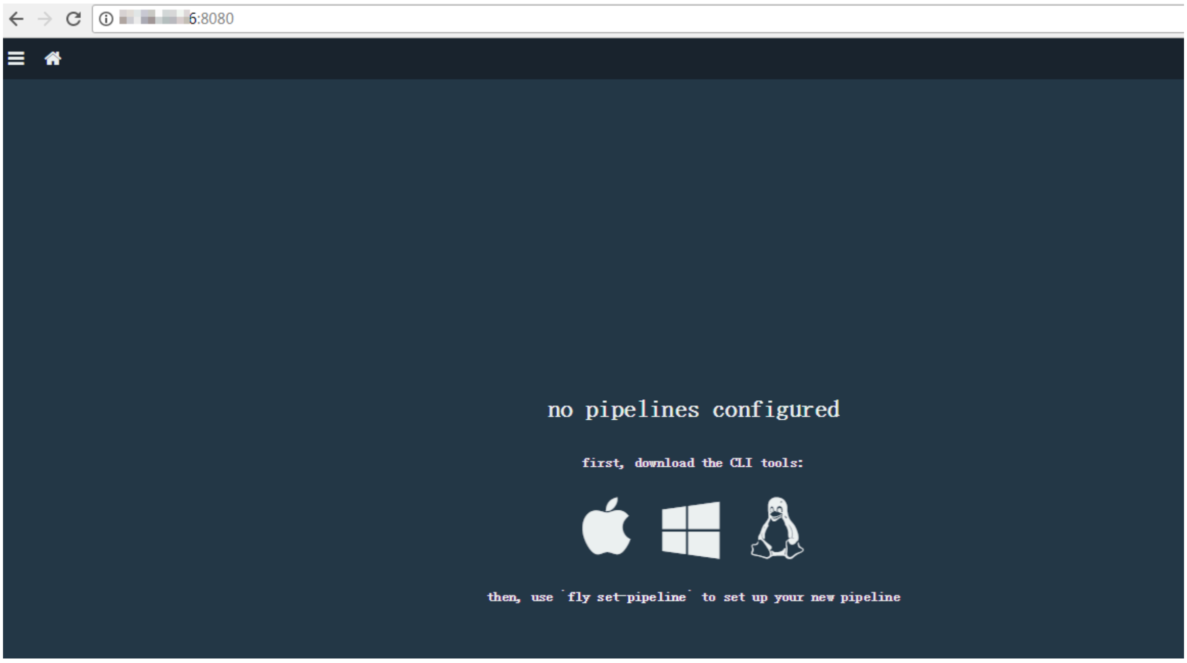
Miss The selected configuration file does not contain this variable.

After the application is created, the following three services are started.



Name	Application	Status	Container Status	Image	Action
concourse-db	test	● Running	Running:1 Stop:0	postgres:9.5	Stop | Restart | Reschedule | Update | Delete | Events
concourse-web	test	● Running	Running:1 Stop:0	concourse/concourse:latest	Stop | Restart | Reschedule | Update | Delete | Events
concourse-worker	test	● Running	Running:1 Stop:0	concourse/concourse:latest	Stop | Restart | Reschedule | Update | Delete | Events

Then, the Concourse CI deployment is finished. Enter `http://your-ecs-public-ip:8080` in the browser to access the Concourse CI.



no pipelines configured

first, download the CLI tools:

then, use `fly set-pipeline` to set up your new pipeline

Run a CI task (Hello world)

1. In the browser opened in the last section, download the CLI corresponding to your operating system and install the CLI client. Use ECS (Ubuntu 16.04) as an example.
2. For Linux and Mac OS X systems, you must add the execution permissions to the downloaded FLY CLI file first. Then, install the CLI to the system and add it to `$ PATH` .

```
chmod  + x  fly
 install  fly  / usr / local / bin / fly
```

3. After the installation, you can check the version.

```
$ fly  - v
  3 . 4 . 0
```

4. Connect to the target. The username and password are concourse and changeme by default.

```
$  fly  - t  lite  login  - c  http :// your - ecs - public - ip
 : 8080
  in  to  team  ' main '
  username :  concourse
  password :
  saved
```

5. Save the following configuration template as `hello . yml` .

```
jobs :
-  name :  hello - world
   plan :
  -  task :  say - hello
     config :
       platform :  linux
       image_reso  urce :
         type :  docker - image
         source : { repository :  ubuntu }
       run :
         path :  echo
         args : [" Hello ,  world !"]
```

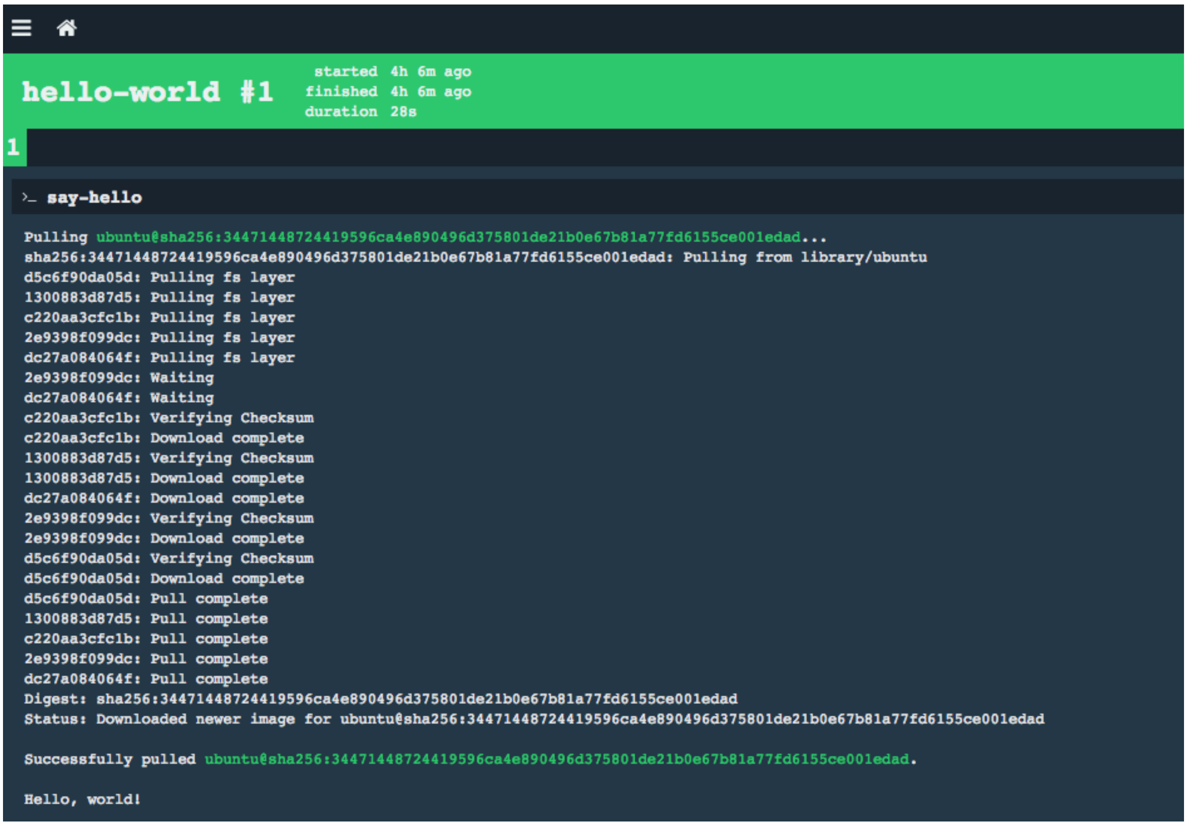
6. Register the task.

```
fly  - t   lite   set - pipeline  - p   hello - world  - c   hello
. yml
```

7. Start the migration task.

```
fly  - t   lite   unpause - pipeline  - p   hello - world
```

The page indicating the successful execution is as follows.



```
Pulling ubuntu@sha256:34471448724419596ca4e890496d375801de21b0e67b81a77fd6155ce001edad...
sha256:34471448724419596ca4e890496d375801de21b0e67b81a77fd6155ce001edad: Pulling from library/ubuntu
d5c6f90da05d: Pulling fs layer
1300883d87d5: Pulling fs layer
c220aa3cfc1b: Pulling fs layer
2e9398f099dc: Pulling fs layer
dc27a084064f: Pulling fs layer
2e9398f099dc: Waiting
dc27a084064f: Waiting
c220aa3cfc1b: Verifying Checksum
c220aa3cfc1b: Download complete
1300883d87d5: Verifying Checksum
1300883d87d5: Download complete
dc27a084064f: Download complete
2e9398f099dc: Verifying Checksum
2e9398f099dc: Download complete
d5c6f90da05d: Verifying Checksum
d5c6f90da05d: Download complete
d5c6f90da05d: Pull complete
1300883d87d5: Pull complete
c220aa3cfc1b: Pull complete
2e9398f099dc: Pull complete
dc27a084064f: Pull complete
Digest: sha256:34471448724419596ca4e890496d375801de21b0e67b81a77fd6155ce001edad
Status: Downloaded newer image for ubuntu@sha256:34471448724419596ca4e890496d375801de21b0e67b81a77fd6155ce001edad

Successfully pulled ubuntu@sha256:34471448724419596ca4e890496d375801de21b0e67b81a77fd6155ce001edad.

Hello, world!
```

For more information about the characteristics of Concourse CI, see Concourse CI project.

10 Deploy Container Service clusters by using Terraform

This document introduces how to use Terraform to deploy Alibaba Cloud Container Service cluster in the Virtual Private Cloud (VPC) environment and deploy a sample WordPress application in the cluster. In this document, a solution used to build Alibaba Cloud infrastructures is provided for you to use codes to automatically create, orchestrate, and manage services in Container Service.

Prerequisite

- You must activate Alibaba Cloud Container Service.
- You must activate Alibaba Cloud Container Service and create an AccessKey for your account. Keep your AccessKey ID and AccessKey Secret properly.

Step 1. Install Terraform

Download Terraform

Download Terraform from the official website. Select the corresponding version and platform. In this document, install the Terraform on Linux (the procedure is similar to that of installing the Terraform on Mac OS X).

1. Under Linux, click to download the `terraform_0.11.3_linux_amd64.zip` file.
2. Copy the *.zip* file to an appropriate path (*/usr/local/terraform* in this example).
3. Extract the .zip file and then get a binary file terraform.
4. Create the following entries in the */etc/profile* directory and add the path where the binary file resides (*/usr/local/terraform* in this example) to the PATH environment variable.

```
export TERRAFORM_HOME=/usr/local/terraform
export PATH=$PATH:$TERRAFORM_HOME
```

Install Alibaba Cloud Terraform package

Before using Terraform, an initialization operation is required to load Alibaba Cloud Provider. Run the following command in the template file directory:

```
terraform  init
```

After the download is successful, the corresponding plugin is downloaded to the *. terraform* hidden directory in the current folder. If you encounter a network timeout problem during the loading process, follow the instructions to complete the manual installation of the plugin.

- Download the corresponding version and platform Provider from Alibaba Cloud Terraform Provider official download address. In this example, the Linux type is selected.
- Copy the downloaded file *terraform - provider - alicloud_1 . 9 . 3_linux_am d64 . zip* to the Terraform installation directory */ usr / local / terraform* and extract it. The current directory gets Alibaba Cloud Provider *terraform - provider - alicloud_v 1 . 9 . 3_x4 .*

Run the following command to test the working of Terraform. If Terraform is successfully installed, the following contents are displayed:

```
$  terraform
 Usage :  terraform  [-- version ] [-- help ] [ args ]

 The   available   commands   for   execution   are   listed   below .
 The   most   common ,  useful   commands   are   shown   first ,
 followed   by
 less   common   or   more   advanced   commands .  If   you ' re
 just   getting
 started   with   Terraform ,  stick   with   the   common   commands
 .  For   the
 other   commands ,  please   read   the   help   and   docs   before
   usage .

 Common   commands :
....

 All   other   commands :
 debug   Debug   output   management  ( experiment  al )
 force - unlock   Manually   unlock   the   terraform   state
 state   Advanced   state   management
```

Step 2. Download Container Service Terraform scripts

You can download the Terraform template (the template download address) to create the swarm cluster and deploy the WordPress application . This template file defines the resources for creating a swarm cluster and the files that deploy Wordpess on the swarm cluster to help you quickly create and deploy swarm clusters. The template contains the following files after being extracted.

main.tf

The main file of Terraform that defines the resources to be deployed.

- Region

  Defines the region where resources are to be created.

```
 provider  " alicloud " {
 access_key  = "${ var . alicloud_a  ccess_key }"
 secret_key  = "${ var . alicloud_s  ecret_key }"
 region  = "${ var . region }"
}
```

- VPC

```
 resource  " alicloud_v  pc " " vpc " {
 name  = "${ var . vpc_name }"
 cidr_block  = "${ var . vpc_cidr }"
}
```

- VSwitch

```
 resource  " alicloud_v  switch " " vswitch " {
 availabili  ty_zone  = "${ data . alicloud_z  ones . default .
 zones .  0 . id }"
 name  = "${ var . vswitch_na  me }"
 cidr_block  = "${ var . vswitch_ci  dr }"
 vpc_id  = "${ alicloud_v  pc . vpc . id }"
}
```

- Container Service cluster

```
 resource  " alicloud_c  s_swarm " " cs_vpc " {
 password  = "${ var . password }"
 instance_t  ype  = "${ data . alicloud_i  nstance_ty  pes . main .
 instance_t  ypes .  0 . id }"
 name  = "${ var . cluster_na  me }"
 node_numbe  r  = "${ var . node_numbe  r }"
 disk_categ  ory  = "${ var . disk_categ  ory }"
 disk_size  = "${ var . disk_size }"
 cidr_block  = "${ var . cidr_block }"
 image_id  = "${ data . alicloud_i  mages . main . images .  0 . id
 }"
 vswitch_id  = "${ alicloud_v  switch . main . id }"
}
```

- WordPress application

```
 resource  " alicloud_c  s_applicat  ion " " wordpress " {
 cluster_na  me  = "${ alicloud_c  s_swarm . cs_vpc . name }"
 name  = "${ var . app_name  == "" ?  var . resource_g  roup_name
 :  var . app_name }"
 version  = "${ var . app_versio  n }"
 template  = "${ file (" wordpress . yml ")}"
```

```
  descriptio  n  = " terraform   deploy   consource "
  latest_ima  ge  = "${ var . latest_ima  ge }"
  blue_green  = "${ var . blue_green }"
  blue_green  _confirm  = "${ var . confirm_bl  ue_green }"
 }
```

outputs.tf

This file defines the output parameters. Resources created as part of the execution generate these output parameters. This is similar to the output parameters specified in a Resource Orchestration Service (ROS) template. For example, the template deploys a swarm cluster and Wordpress application instance. The following output parameters provide the cluster ID and the default domain name for the application.

```
 output  " cluster_id " {
   value  = "${ alicloud_c  s_swarm . cs_vpc . id }"
}
```

```
 output  " default_do  main " {
   value  = "${ alicloud_c  s_applicat  ion . wordpress . default_do
 main }"
}
```

variables.tf

This file contains the variables that can be passed to main.tf and helps you customize the environment.

```
 variable  " alicloud_a  ccess_key " {
   descriptio  n  = " The   Alicloud   Access   Key   ID   to   launch
   resources .  Support   to   environmen  t  ' ALICLOUD_A  CCESS_KEY
 '."
}
```

```
 variable  " alicloud_s  ecret_key " {
   descriptio  n  = " The   Alicloud   Access   Secret   Key   to
 launch   resources .   Support   to   environmen  t  ' ALICLOUD_S
 ECRET_KEY '."
}
```

```
 variable  " region " {
   descriptio  n  = " The   region   to   launch   resources ."
   default  = " cn - hongkong "
}
```

```
 variable  " vpc_cidr " {
   descriptio  n  = " The   cidr   block   used   to   launch   a
 new   vpc ."
   default  = " 172 . 16 . 0 . 0 / 12 "
}
```

```
 variable  " app_name " {
   descriptio  n  = " The   app   resource   name .  Default   to
 variable  ` resource_g  roup_name `"
```

```
    default  = " wordpress "
 }
```

wordpress.yml

Deploy the Compose template of the WordPress application from the orchestration templates provided in the console. Log on to the Container Service console, click Application in the left-side navigation pane, select Create Application > Create by template > Use an existing template.

Step 3. Run Terraform scripts

To run the script, first locate the directory where you stored the preceding files, such as */ root / terraform / wordpress* . You can use the following terraform related commands to run scripts, build container clusters, and deploy applications. For more information, see Terraform Commands (CLI).

Run `terraform  init` to initialize the environment.

```
$  terraform   init
   Initializi  ng   provider   plugins ...
  ...
   -  Checking   for   available   provider   plugins   on   https ://
 releases . hashicorp . com ...
   -  Downloadin  g   plugin   for   provider   " alicloud " ( 1 . 7 .
 2 )...
   *  provider . alicloud :  version  = "~>  1 . 7 "
    Terraform   has   been   successful  ly   initialize  d !
   ...
```

Run the `terraform  providers` command to list the installed providers.

```
 terraform   providers
.
└──  provider . alicloud
```

Before running `terraform  plan` , you must first enter the AccessKey ID and AccessKey Secret for authorization.

```
$  export   ALICLOUD_A  CCESS_KEY =" AccessKey   ID "
$  export   ALICLOUD_S  ECRET_KEY =" AccessKey   Secret "
```

Run `terraform  plan` to create an execution plan and help you understand the resources that are going to be created or changed.

```
$  terraform   plan
 Refreshing   Terraform   state   in - memory   prior   to   plan ...
 The   refreshed   state   will   be   used   to   calculate   this
 plan ,  but   will   not   be
 persisted   to   local   or   remote   state   storage .
 data . alicloud_i  mages . main :  Refreshing   state ...
 data . alicloud_i  nstance_ty  pes . default :  Refreshing   state
 ...
 data . alicloud_z  ones . default :  Refreshing   state ...
------------------------------------------------------------------------
 An   execution   plan   has   been   generated   and   is   shown
 below .
 Resource   actions   are   indicated   with   the   following
 symbols :
  +  create
 Terraform   will   perform   the   following   actions :
...
 Plan :  9   to   add ,  0   to   change ,  0   to   destroy .
------------------------------------------------------------------------
 Note :  You   didn ' t   specify   an   "- out "  parameter   to
 save   this   plan ,  so   Terraform
 can ' t   guarantee   that   exactly   these   actions   will   be
 performed   if
" terraform   apply "  is   subsequent  ly   run .
```

After the resources are created or updated as expected, run the `terraform   apply` command to start the execution of the Terraform module.

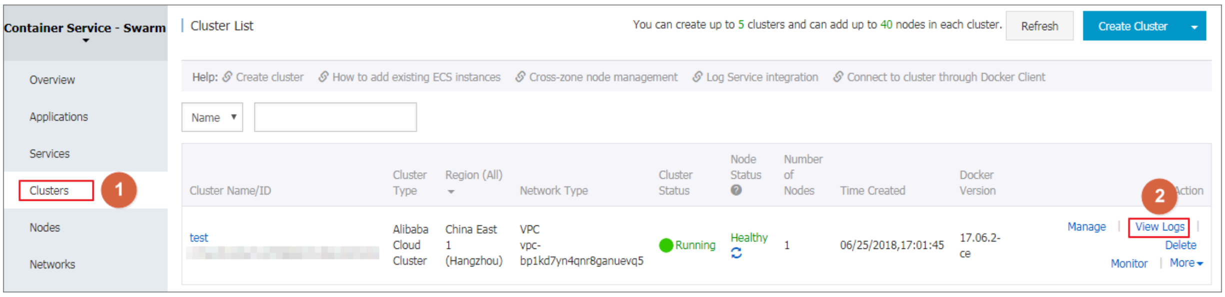
```
$  terraform   apply
 data . alicloud_i  nstance_ty  pes . default :  Refreshing   state
 ...
 data . alicloud_i  mages . main :  Refreshing   state ...
 data . alicloud_z  ones . default :  Refreshing   state ...
 An   execution   plan   has   been   generated   and   is   shown
 below .
 Resource   actions   are   indicated   with   the   following
 symbols :
  +  create
 Terraform   will   perform   the   following   actions :
...
 Plan :  9   to   add ,  0   to   change ,  0   to   destroy .
 Do   you   want   to   perform   these   actions ?
   Terraform   will   perform   the   actions   described   above .
   Only   ' yes '  will   be   accepted   to   approve .
   Enter   a   value :  yes
 alicloud_v  pc . vpc :  Creating ...
...
 Apply   complete !  Resources :  9   added ,  0   changed ,  0
 destroyed .
 Outputs : ## Note
 availabili  ty_zone  =  cn - hongkong - a
 cluster_id  =  c95537435b ********
 default_do  main  =  c95537435b ********. cn - hongkong . alicontain
 er . com
 vpc_id  =  vpc - 2zeaudqan6  uzt5lzry48  a
 vswitch_id  =  vsw - 2ze2x92n9b  5neor7fcjm  r
```

After running the `terraform   apply` command, the output parameters requested in the `outputs . tf` are displayed. In the preceding example, the output parameters are the cs_cluster cluster ID, available zone, VPC ID, VSwitch ID name, and the default_domain of the application instance.

The output values can be listed at any time by running the `terraform output` command to help you configure the WordPress application.

```
terraform   output
availabili  ty_zone  =  cn - hongkong - a
cluster_id  =  c95537435b ********
default_do  main  =  c95537435b ********. cn - hongkong . alicontain
er . com
vpc_id  =  vpc - 2zeaudqan6  uzt5lzry48  a
vswitch_id  =  vsw - 2ze2x92n9b  5neor7fcjm  r
```

You can view the cluster created by using Terraform in the Container Service console. View the cluster, node, container, and logs.

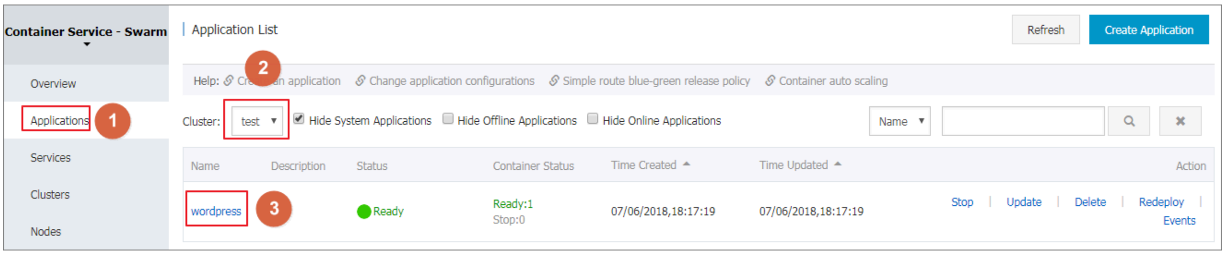


You can create up to 5 clusters and can add up to 40 nodes in each cluster.

Help: Create cluster How to add existing ECS instances Cross-zone node management Log Service integration Connect to cluster through Docker Client

Cluster Name/ID	Cluster Type	Region (All)	Network Type	Cluster Status	Node Status	Number of Nodes	Time Created	Docker Version	Action
test [redacted]	Alibaba Cloud Cluster	China East 1 (Hangzhou)	VPC vpc-bp1kd7yn4qnr8ganuevq5	● Running	Healthy	1	06/25/2018,17:01:45	17.06.2-ce	Manage | View Logs | Delete Monitor | More

At the same time, you can view the WordPress application information on the Application page.

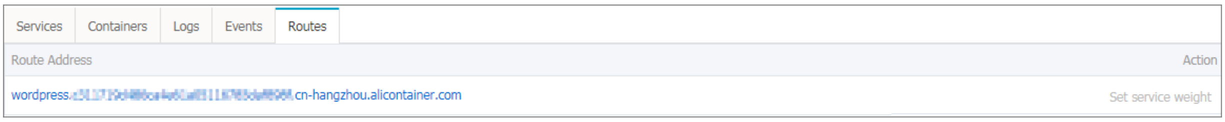


Help: Cr an application Change application configurations Simple route blue-green release policy Container auto scaling

Cluster: test ☑ Hide System Applications □ Hide Offline Applications □ Hide Online Applications

Name	Description	Status	Container Status	Time Created	Time Updated	Action
wordpress		● Ready	Ready:1 Stop:0	07/06/2018,18:17:19	07/06/2018,18:17:19	Stop | Update | Delete | Redeploy | Events

Click the application name, and then click Routes to view the route address.

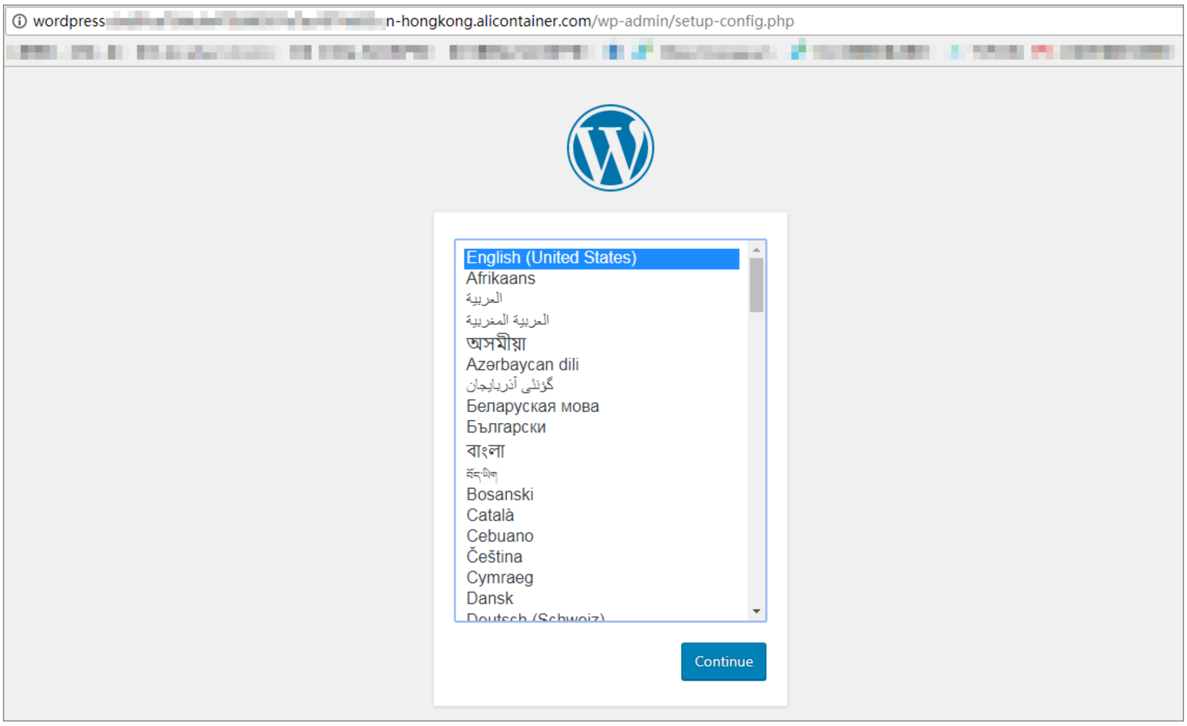


Route Address	Action
wordpress.[redacted].cn-hangzhou.alicontainer.com	Set service weight

Step 4. Access WordPress

1. Open the Wordpress Compose template `wordpress . yml` and find the application domain prefix `aliyun . routing . port_80 :  http :// wordpress` .
2. The value of the domain name prefix `http :// wordpress` and application `default_do  main` spliced with the `http : // wordpress . c95537435b ********. cn - hongkong . alicontain  er . com` . Enter the browser

to access the WordPress welcome page, select the language, and set other configurations.

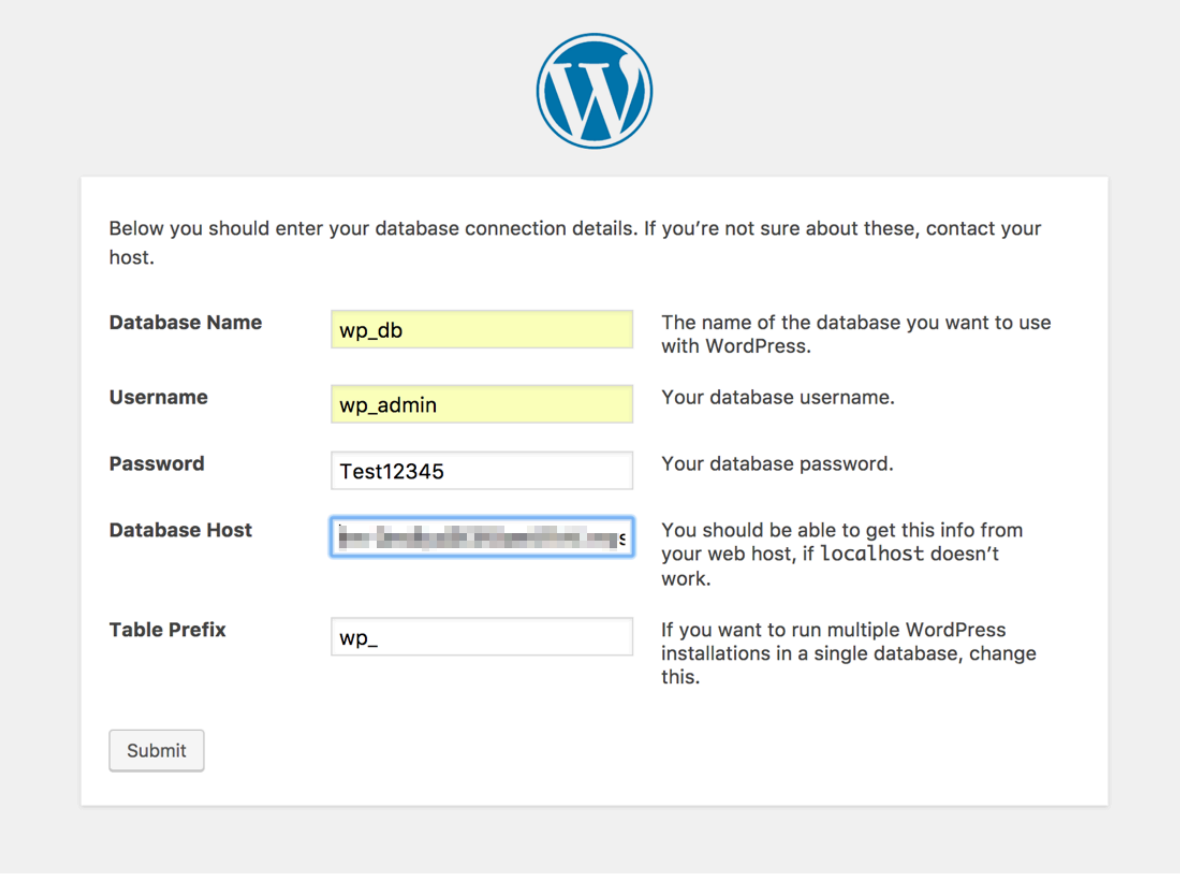


wordpress[redacted]n-hongkong.alicontainer.com/wp-admin/setup-config.php

[redacted]

English (United States)
Afrikaans
العربية
العربية المغربية
অসমীয়া
Azərbaycan dili
گؤنئی آذربایجان
Беларуская мова
Български
বাংলা
བོད་ཡིག
Bosanski
Català
Cebuano
Čeština
Cymraeg
Dansk
Deutsch (Schweiz)

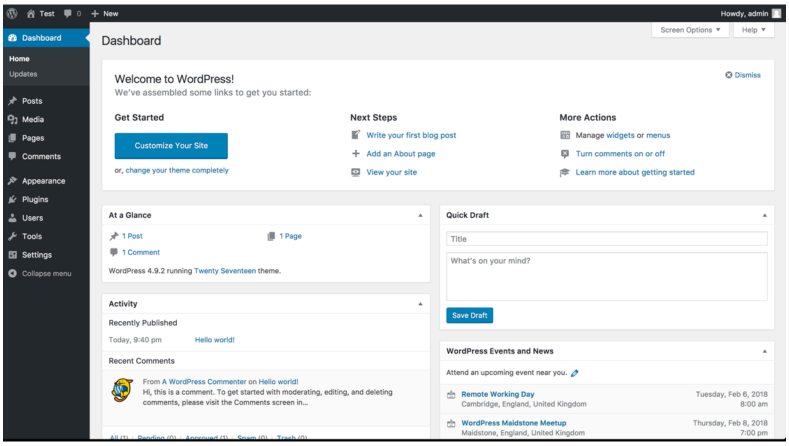
3. Enter the Site Title, username, and password of the administrator. Click Install WordPress.



Below you should enter your database connection details. If you're not sure about these, contact your host.

Database Name	wp_db	The name of the database you want to use with WordPress.
Username	wp_admin	Your database username.
Password	Test12345	Your database password.
Database Host	[redacted]	You should be able to get this info from your web host, if `localhost` doesn't work.
Table Prefix	wp_	If you want to run multiple WordPress installations in a single database, change this.

4. After the installation, click Log In. Enter the username and password of the administrator, and then click Log In on the WordPress logon page to log on to the WordPress application.



Welcome to WordPress!
We've assembled some links to get you started:

WordPress 4.9.2 running Twenty Seventeen theme.

Today, 9:40 pm Hello world!

From A WordPress Commenter on Hello world!
Hi, this is a comment. To get started with moderating, editing, and deleting comments, please visit the Comments screen in…

Attend an upcoming event near you.

Remote Working Day — Cambridge, England, United Kingdom — Tuesday, Feb 6, 2018 8:00 am

WordPress Maidstone Meetup — Maidstone, England, United Kingdom — Thursday, Feb 8, 2018 7:00 pm

Further information

Currently, Alibaba Cloud is the official major cloud provider of Terraform. To use Terraform to flexibly build Alibaba Cloud infrastructures, see Alibaba Cloud Provider for more information and customize the resource description files to quickly build your cloud infrastructures.

11 Use Chef to automatically deploy Docker and WebServer

Chef is an automated deployment framework. Combined with Alibaba Cloud Container Service, Chef can help you achieve customization and automation in your deployment. Log on to the Chef official website first to learn about basic terms for quick start, such as cookbook, recipe, chef workstation, chef server, and chef nodes.

Prerequisites

- You have created a swarm cluster that retains the EIP.
- Prepare a local Linux environment. This example uses Ubuntu 16.04. According to your local environment, download a ChefDK at https://downloads.chef.io/chefdk/.
- Log on to the Chef official website to register an account and create an organization . In this example, the created organization is called example.

Install the chef workstation on Linux

You need to go to the Chef official website to download a ChefDK which is compatible with your local Linux environment. This example uses a ChefDK corresponding to Ubuntu 16.04.

First create a `chef - repo` directory in the `/ home` directory.

```
mkdir  / home / chef - repo
```

Enter the `chef - repo` directory and use the `curl` command to download a ChefDK package to install.

```
cd  / home / chef - repo
curl  - O   https :// packages . chef . io / files / stable / chefdk
/ 3 . 0 . 36 / ubuntu / 16 . 04 / chefdk_3 . 0 . 36 - 1_amd64 . deb
dpkg  - i   chefdk_3 . 0 . 36 - 1_amd64 . deb
```

Then you need to perform a large number of Chef installation configurations. If you encounter problems during installation, see Chef official documents to troubleshoot the problems.

Verify Chef

```
chef   verify  # Verify   if   the   ChefDK   components   are
normal
```

```
chef  -- version  # View   the   Chef   version .
```

Set Chef environment variables

Set environment variables related to Chef, such as GEM_ROOT, GEM_HOME, and GEM_PATH.

```
export GEM_ROOT="/opt/chefdk/embedded/lib/ruby/gems/2.1.0"
export GEM_HOME="/root/.chefdk/gem/ruby/2.1.0"
export GEM_PATH="/root/.chefdk/gem/ruby/2.1.0:/opt/chefdk/embedded/lib/
ruby/gems/2.1.0"
```

In addition, if Ruby is already installed on your system, update the PATH variable related to Ruby.

```
export   PATH ="/ opt / chefdk / bin :/ root /. chefdk / gem / ruby
/ 2 . 1 . 0 / bin :/ opt / chefdk / embedded / bin :/ opt / chefdk /
bin :/ root /. chefdk / gem / ruby / 2 . 1 . 0 / bin :/ opt / chefdk
/ embedded / bin :/ opt / chefdk / bin :/ root /. chefdk / gem / ruby
/ 2 . 1 . 0 / bin :/ opt / chefdk / embedded / bin :/ usr / local /
sbin :/ usr / local / bin :/ usr / sbin :/ usr / bin :/ root / bin "
```

Configure firewalld rules for accessing Chef

To access the Chef Manage GUI on the Chef server, add the following firewalld rules and open corresponding ports on the Chef server.

```
firewall - cmd  -- direct  -- add - rule   ipv4  \
filter   INPUT_dire  ct   0  - i   eth0  - p   tcp  \
-- dport   443  - j   ACCEPT

firewall - cmd  -- direct  -- add - rule   ipv4  \
filter   INPUT_dire  ct   0  - i   eth0  - p   tcp  \
-- dport   80  - j   ACCEPT

firewall - cmd  -- direct  -- add - rule   ipv4  \
filter   INPUT_dire  ct   0  - i   eth0  - p   tcp  \
-- dport   9683  - j   ACCEPT

firewall - cmd  -- reload
```

Download Starter Kit from the Chef Manage Gui

Log on to Chef Manage GUI, click Administration, and select the organization in the drop-down list. In this example, the organization is example. After the organization is selected, click the Starter Kit in the left-side navigation pane to download the chef-starter.zip file to your local host.

Transfer the *chef - starter . zip* file to the Chef workstation in your local Linux , and extract it to the *home / chef - repo* directory.

```
#  cd  / home / chef - repo
```

```
unzip chef-starter.zip
```

Download the SSL Certificate for the Chef server

The certificate is downloaded to the *chef-repo/.chef/trusted_certs* directory.

```
# cd ~/chef-repo
# knife ssl fetch

 WARNING: Certificates from api.chef.io will be
 fetched and placed in your trusted_cert
 directory (/root/chef-repo/.chef/trusted_certs).

 Knife has no means to verify these are the correct
   certificates. You should
 verify the authenticity of these certificates after
 downloading.

 Adding certificate for wildcard_opscode_com in /root/
 chef-repo/.chef/trusted_certs/wildcard_opscode_com.crt

 Adding certificate for DigiCert_SHA2_Secure_Server_CA
   in /root/chef-repo/.chef/trusted_certs/DigiCert_S
 HA2_Secure_Server_CA.crt
```

Verify if the Chef workstation is installed successfully

After completing configuration, execute the following commands. If the created organization is displayed, you have successfully connected to the workstation.

```
# cd ~/chef-repo

# knife client list
 example-validator
```

Create a cookbook that implements Docker automatic initialization

1. Create a cookbook on the Chef workstation.

   - In the chef-repo/cookbooks directory, execute the following command to create a cookebook named docker_init.

     ```
     chef generate cookbook docker_init
     ```

   - Go to the *chef-repo/cookbooks/docker_init/recipe/* directory to find the default.rb file and configure the file. This example is used to start the latest version of Docker in Ubuntu.

     ```
     apt_update


     package 'apt-transport-https'
     ```

```
package  ' ca - certificat  es '
package  ' curl '
package  ' software - properties - common '


execute  ' apt - key '  do
command  ' apt - key   fingerprin  t   0EBFCD88 '
end


execute  ' apt - repo '  do
command  ' add - apt - repository  " deb  [ arch = amd64 ]  https
:// download . docker . com / linux / ubuntu / dists / xenial /
stable /"'


end


execute  ' apt - repo '  do
command  ' apt - get   update '
end


execute  ' apt - repo '  do
command  ' apt - get   install   docker - ce  - y  -- allow -
unauthenti  cated '
end


service  ' docker '  do
action  [: start , : enable ]
end
```

2. Verify if the cookbook named docker_init works locally.

```
#  chef - client  -- local - mode  -- runlist  ' recipe [
 docker_ini  t ]'

[ 2018 - 06 - 27T15 : 54 : 30 + 08 : 00 ]  INFO :  Started   chef -
 zero   at   chefzero :// localhost : 1   with   repository   at  /
 root / chef - repo
 One   version   per   cookbook


 Starting   Chef   Client ,  version   14 . 1 . 12
[ 2018 - 06 - 27T15 : 54 : 30 + 08 : 00 ]  INFO : ***  Chef   14 . 1
 . 12  ***
[ 2018 - 06 - 27T15 : 54 : 30 + 08 : 00 ]  INFO :  Platform :
 x86_64 - linux
[ 2018 - 06 - 27T15 : 54 : 30 + 08 : 00 ]  INFO :  Chef - client
 pid :  2010
[ 2018 - 06 - 27T15 : 54 : 30 + 08 : 00 ]  INFO :  The   plugin
   path  / etc / chef / ohai / plugins   does   not   exist .
 Skipping ...
[ 2018 - 06 - 27T15 : 54 : 31 + 08 : 00 ]  INFO :  Setting   the
 run_list   to  [#]  from   CLI   options
[ 2018 - 06 - 27T15 : 54 : 32 + 08 : 00 ]  INFO :  Run   List   is
 [ recipe [ docker_ini  t ]]
[ 2018 - 06 - 27T15 : 54 : 32 + 08 : 00 ]  INFO :  Run   List
 expands   to  [ docker_ini  t ]
[ 2018 - 06 - 27T15 : 54 : 32 + 08 : 00 ]  INFO :  Starting   Chef
 Run   for   yxm
```

```
[2018-06-27T15:54:32+08:00] INFO: Running start
 handlers
[2018-06-27T15:54:32+08:00] INFO: Start handlers
   complete.
 resolving cookbooks for run list: ["docker_init"]
[2018-06-27T15:54:32+08:00] INFO: Loading
 cookbooks [docker_init@0.1.0]
 Synchronizing Cookbooks:
- docker_init (0.1.0)
 Installing Cookbook Gems:
 Compiling Cookbooks...
 Converging 10 resources
 Recipe: docker_init::default
* apt_update[] action periodic [2018-06-27T15:54:32+
 08:00] INFO: Processing apt_update[] action periodic (
 docker_init::default line 9)
....
---- End output of add-apt-repository "deb [arch=
 amd64] https://download.docker.com/linux/ubuntu/dists
 /xenial/stable/" ----
 Ran add-apt-repository "deb [arch=amd64] https://
 download.docker.com/linux/ubuntu/dists/xenial/stable
 /" returned 1
```

Execute the following command to check if the locally installed docker is upgraded to the latest version.

```
# docker --version
 Docker version 17.06.2-ce, build 2e0fd6f
```

3. Upload the cookbook to the Chef server.

   - On the Chef workstation, upload the cookbook named docker_init to the Chef server by executing the following command.

     ```
     knife cookbook upload docker_init
     ```

   - Execute the following command to verify that the cookbook is uploaded successfully.

     ```
     # knife cookbook list
      docker_init 0.1.0
     ```

4. Import the cookbook into the node of the Alibaba Cloud swarm cluster.

   - On the Chef workstation, execute the following command to import docker_init into the node of the swarm cluster that act as a Chef node.

     Note:

     Replace ADDRESS with the EIP of the ECS node of the swarm cluster. USER is the logon user of the ECS node, typically root. PASSWORD is the ECS node logon password. If the swarm cluster has multiple nodes, execute this command for each ECS node.

```
#  knife   bootstrap   ADDRESS  -- ssh - user   USER  -- ssh -
 password  ' PASSWORD ' -- sudo  -- use - sudo - password  -- node
 - name   node1 - ubuntu  -- run - list  ' recipe [ docker_ini  t
 ]'

 Creating   new   client   for   node1 - ubuntu
 Creating   new   node   for   node1 - ubuntu
 Connecting   to   121 . 196 . 219 . 18
...
 https :// download . docker . com / linux / ubuntu / dists /
 xenial / stable /" ----
 121 . 196 . 219 . 18   Ran   add - apt - repository  " deb
 [ arch = amd64 ]  https :// download . docker . com / linux /
 ubuntu / dists / xenial / stable /"  returned   1
```

- Log on to each ECS node to check if the docker installed on each node has been updated to the latest version. Execute the `docker  --  version` command to verify.

Now you have updated the version of Alibaba Cloud container cluster Docker through the Chef automated deployment system.

Create a cookbook that automates the deployment of Web Server

1. Create a new cookbook on the Chef workstation.

   - In the chef-repo/cookbooks directory, execute the following command to create a cookbook named web_init.

   ```
    chef   generate   cookbook   web_init
   ```

   - Go to the *chef - repo / cookbooks / web_init / recipe /*directory to find the default.rb file and configure the file.

   ```

    execute  ' apt - repo '  do
    command  ' apt - get  - y   install   apache2  -- allow -
    unauthenti  cated '
    end


    service  ' apache2 '  do
    action  [: start , : enable ]
    end


    file  '/ var / www / html / index . html '  do
    content  '
    hello , world
   '
    end
   ```

```
service  ' iptables '  do
action  : stop
end
```

2. Verify that the cookbook works locally.

   - Execute the `curl  http :// localhost : 80` command to check if the web_init works on the local host.
   - On the Chef workstation, upload the cookbook named web_init to the Chef server.

   ```
   knife  cookbook  upload  web_init
   ```

3. Import the cookbook into the node of the Alibaba Cloud swarm cluster.

   On the Chef workstation, execute the following command to import web_init into the node of the swarm cluster that acts as a chef node.

   > Note:
   >
   > Replace ADDRESS with the EIP of the ECS node of the swarm cluster. USER is the logon user of the ECS node, typically root. PASSWORD is the ECS node logon password. If the swarm cluster has multiple nodes, execute this command for each ECS node.

   ```
   knife  bootstrap  ADDRESS  -- ssh - user  USER  -- ssh -
   password  ' PASSWORD ' -- sudo  -- use - sudo - password  -- node -
   name  node1 - ubuntu  -- run - list  ' recipe [ web_init ]'
   ```

4. Check if the Web Server starts successfully in the Alibaba Cloud swarm cluster. Log on to the node of the Alibaba Cloud swarm cluster.

   - Execute the `systemctl  status  apache2 . service` command to check if apache2 operates normally.
   - Visit `http :// ADDRESS : 80` in the browser to see if `hello  world` is displayed.

     > Note:
     >
     > ADDRESS is the EIP of the node.