# 04_Deploy_Best Practice_Public Cloud-Copy

| | |
|---|---|
| **Issue** | 01 |
| **Date** | 2024-12-06 |

HUAWEI CLOUD COMPUTING TECHNOLOGIES CO., LTD.

## Huawei Cloud Computing Technologies Co., Ltd.

| | |
|---|---|
| Address: | Huawei Cloud Data Center Jiaoxinggong Road<br>Qianzhong Avenue<br>Gui'an New District<br>Gui Zhou 550029<br>People's Republic of China |
| Website: | https://www.huaweicloud.com/intl/en-us/ |

# Contents

**1 CodeArts Deploy Best Practices........1**

**2 Deploying an Application on an Intranet Host Using a Proxy Host........2**

**3 Using Nginx for Gray Release........8**

**4 Using Kubernetes Nginx-Ingress for Gray Release........33**

**5 HE2E DevOps Practice - Deploying an Application........39**

5.1 Overview........39

5.2 Deploying an Application on CCE........39

5.3 Deploying an Application on ECS........43

# 1 CodeArts Deploy Best Practices

**Table 1-1** Best practices

| Practice | | Description |
|---|---|---|
| Internal network deployment | **Deploying an Application on an Intranet Host Using a Proxy Host** | This practice provides a complete operation guide for deploying an application on a host or server on the internal network through a proxy host. |
| Gray deployment | **Using Nginx for Gray Release** | This practice implements blue-green and gray release of applications based on the Nginx load balancing mechanism. |
| | **Using Kubernetes Nginx-Ingress for Gray Release** | This practice implements gray release based on native Kubernetes features. |
| HE2E DevOps practice | **HE2E DevOps Practice - Deploying an Application** | This practice uses the DevOps process example project to describe how to deploy applications on CCE and ECS. |

# 2 Deploying an Application on an Intranet Host Using a Proxy Host

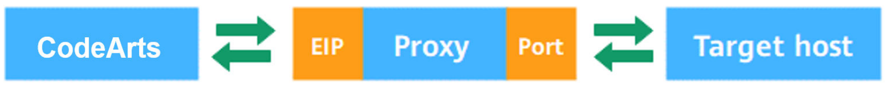
## Application Scenario

Deploy applications on the intranet through proxy hosts to effectively control intranet and extranet communication, enhance data security and network performance, and meet compliance requirements. This policy is widely used in various key scenarios, such as resource access control, secure communication between data centers, content cache acceleration, environment isolation, security audit, and sensitive data processing.

## Solution Architecture

The Internet forward proxy function of Squid is used to specify the IP address and port of the target host on the proxy, enabling the target host to access the public network.

For more information about Squid, go to **Squid official website**. The following procedure uses a Linux host as an example.



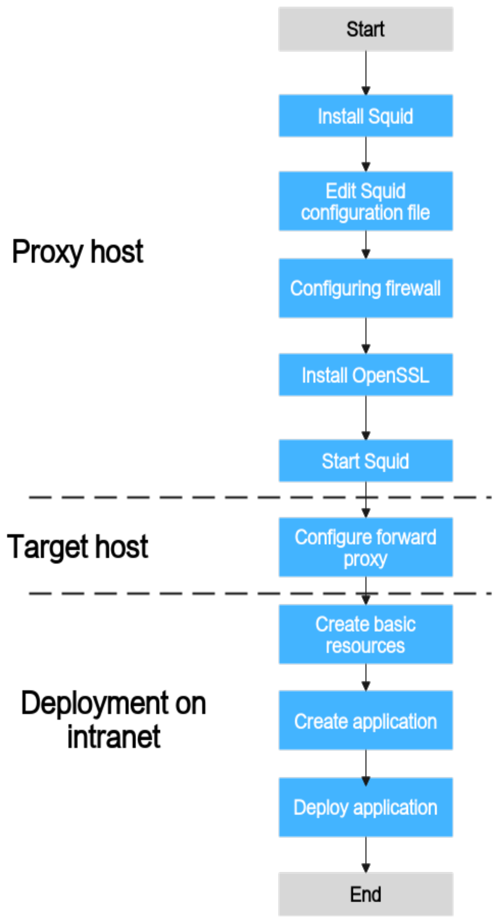
## Prerequisites

- A host (**Proxy-B**) bound to a public IP address is available. If no proxy host is available, see **Applying for an ECS**.
- A host (**Host-A**) not bound to a public IP address is available.
- **Proxy-B** and **Host-A** can access each other through the intranet.

## Process

This section describes how to deploy an application on an intranet host or server using a proxy host.

**Figure 2-1** Process flowchart



**Step 1** Install Squid.

Access the command line tool of **Proxy-B** and run the following command:

```
yum install squid -y
```

If **Complete** is displayed, run the following command:

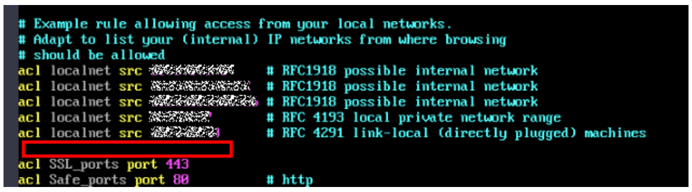
```
yum install iptables-services
```

Enter **Y**. If **Complete** is displayed, the installation is complete.

**Step 2** Edit the Squid configuration file.

1. Access the command line tool of **Proxy-B** and run the following command:

```
vim /etc/squid/squid.conf
```



```
# Example rule allowing access from your local networks.
# Adapt to list your (internal) IP networks from where browsing
# should be allowed
acl localnet src [redacted]    # RFC1918 possible internal network
acl localnet src [redacted]    # RFC1918 possible internal network
acl localnet src [redacted]    # RFC1918 possible internal network
acl localnet src [redacted]    # RFC 4193 local private network range
acl localnet src [redacted]    # RFC 4291 link-local (directly plugged) machines

acl SSL_ports port 443
acl Safe_ports port 80          # http
```

2. Add the following command to the position marked in the red box in the preceding figure:

```
acl local src Internal IP address of the host/24
```

3. Press **Esc** and enter **:wq** to save the file and exit.

**Step 3** Configure the firewall of **Proxy-B**.

Access the command line tool of **Proxy-B** and run the following commands in sequence:

```
systemctl stop firewalld.service
systemctl disable firewalld.service
yum install iptables-services iptables-devel -y
systemctl enable iptables.service
systemctl start iptables.service
iptables -I INPUT 1 -s Internal IP address of the host/24 -p tcp --dport 3128 -j ACCEPT
iptables -I INPUT 2 -p tcp --dport 3128 -j DROP
```

**NOTE**

The IP address in the last but one line must be set to the internal IP address segment or IP address of **Host-A**. **3128** is the proxy port of Squid.

**Step 4** Install OpenSSL.

Access the command line tool of **Proxy-B** and run the following command:

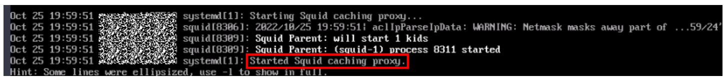
```
yum install openssl
```

If **Complete** is displayed, the installation is complete.

**Step 5** Start Squid.

Access the command line tool of **Proxy-B** and run the following command:

```
systemctl start squid      //Start Squid.
systemctl status squid    //Check the status of Squid.
```



```
Oct 25 19:59:51 [redacted] systemd[1]: Starting Squid caching proxy...
Oct 25 19:59:51            squid[8306]: 2022/10/25 19:59:51| aclIpParseIpData: WARNING: Netmask masks away part of ...59/24'
Oct 25 19:59:51            squid[8309]: Squid Parent: will start 1 kids
Oct 25 19:59:51            squid[8309]: Squid Parent: (squid-1) process 8311 started
Oct 25 19:59:51            systemd[1]: Started Squid caching proxy.
Hint: Some lines were ellipsized, use -l to show in full.
```

**Step 6** Configure the forward proxy.

Access the command line tool of **Host-A** and run the following command:

```
echo "export http_proxy=http://Internal IP address of the proxy host:3128" >>/etc/profile
echo "export https_proxy=http://Internal IP address of the proxy host:3128" >>/etc/profile
echo "export http_proxy=http://Internal IP address of the proxy host:3128" >>~/.bashrc
echo "export https_proxy=http://Internal IP address of the proxy host:3128" >>~/.bashrc
echo "export http_proxy=http://Internal IP address of the proxy host:3128" >>~/.bash_profile
echo "export https_proxy=http://Internal IP address of the proxy host:3128" >>~/.bash_profile
source /etc/profile
source ~/.bashrc
source ~/.bash_profile
```

**Step 7** Create basic resources.

1. Click **Homepage** to view all created projects, and then go to the target project.
2. Choose **Settings** > **General** > **Basic Resources**. The **Host Clusters** page is displayed by default.

   **NOTE**

   Choose **CICD** > **Deploy**. Click **Basic Resources**. The **Host Clusters** page is displayed by default.
3. Click **Create Host Cluster**, enter the following information, and click **Save**.

| Parameter | Mandatory | Description |
|---|---|---|
| Cluster Name | Yes | Enter a custom name. |
| OS | Yes | Select **Linux** based on the OS of the host to be added. |
| Host Connection Mode | Yes | Select **Proxy**. |
| Execution Resource Pool | Yes | A resource pool is a collection of physical environments where commands are executed during software package deployment. In this scenario, select **official**. |
| Description | No | Enter a description. |

4. Click **Add Host**, select **Adding IP** for **Add hosts by**, enter the following information, and click **OK**. The proxy host is created.

**Table 2-1** Parameters of a Linux proxy host

| Parameter | Mandatory | Description |
|---|---|---|
| Host Name | Yes | Enter a custom name, for example, **Proxy-B**. |
| IP | Yes | Enter the public IP address bound to **Proxy-B**. |
| OS | Yes | Keep the default value because it is the OS of your host cluster. |
| Authorization | Yes | In this scenario, the **Password** is used for authentication. Enter the username and password of **Proxy-B**. |
| SSH Port | Yes | Port **22** is recommended. |

5. Click **Add Host**, select **Adding IP** for **Add hosts by**, enter the following information, and click **OK**. The target host is created.

**Table 2-2** Parameters of a Linux target host

| Parameter | Mandatory | Description |
|---|---|---|
| Host Name | Yes | Enter a custom name, for example, **Host-A**. |

| Parameter | Mandatory | Description |
|---|---|---|
| Proxy Host | Yes | Select **Proxy-B** as the network proxy for the target host that cannot connect to the public network. |
| IP | Yes | Enter the private IP address of **Host-A**. |
| OS | Yes | Keep the default value because it is the OS of your host cluster. |
| Authorization | Yes | In this scenario, the **Password** is used for authentication. Enter the username and password of **Host-A**. |
| SSH Port | Yes | Port **22** is recommended. |

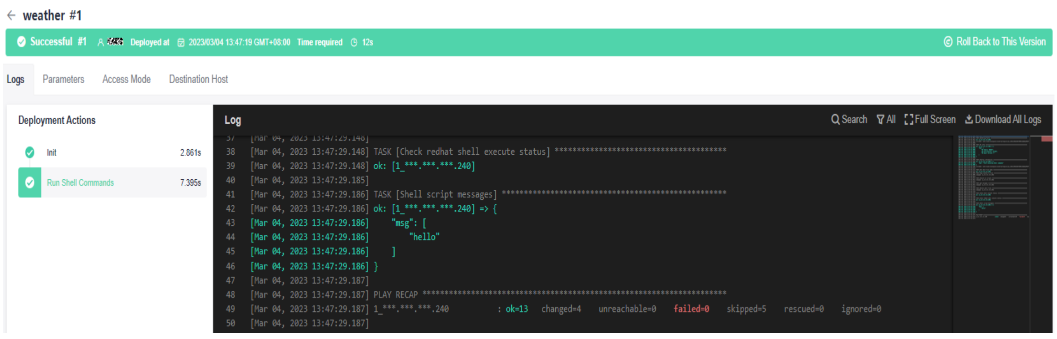
6. Click ▷ in the **Operation** column of a host to start the host for connectivity verification.

**Step 8** Create an application.

1. Log in to the CodeArts homepage and click the target project name to access the project.
2. Choose **CICD** > **Deploy**.
3. Click **Create Application**. On the **Set Basic Information** page, modify the basic information such as **Name**, **Description**, and **Execution Host** as required.
4. After editing the basic application information, click **Next**. On the deployment template selection page, select **Blank Template** and click **OK**.
5. On the **Deployment Actions** tab page, find the action list on the right, click **Add** to add an action to the orchestration area on the left.
6. On the **Environment Management** page, click **Create Environment**, enter the basic information, and click **Save**.
7. Click **Import Host**. The system automatically filters all clusters that meet the requirements of the current environment. In the dialog box that is displayed, select the target host cluster and import **Proxy-B** and **Host-A** to the environment.

**Step 9** Deploy the application. For details, see **Deploying an Application and Viewing the Result**.

1. Select the target application in the application list and click ▷ .
2. After the deployment is complete, click the application name and click the target deployment record. The application status bar changes to green and the message **Successful** is displayed, indicating that the application is successfully deployed.



```
[Mar 04, 2023 13:47:29.148]
[Mar 04, 2023 13:47:29.148] TASK [Check redhat shell execute status] ****************************************
[Mar 04, 2023 13:47:29.148] ok: [1_***.***.***.240]
[Mar 04, 2023 13:47:29.185]
[Mar 04, 2023 13:47:29.186] TASK [Shell script messages] ****************************************************
[Mar 04, 2023 13:47:29.186] ok: [1_***.***.***.240] => {
[Mar 04, 2023 13:47:29.186]     "msg": [
[Mar 04, 2023 13:47:29.186]         "hello"
[Mar 04, 2023 13:47:29.186]     ]
[Mar 04, 2023 13:47:29.186] }
[Mar 04, 2023 13:47:29.187]
[Mar 04, 2023 13:47:29.187] PLAY RECAP **********************************************************************
[Mar 04, 2023 13:47:29.187] 1_***.***.***.240          : ok=13    changed=4    unreachable=0    failed=0    skipped=5    rescued=0    ignored=0
[Mar 04, 2023 13:47:29.187]
```

**----End**

# 3 Using Nginx for Gray Release

## Application Scenario

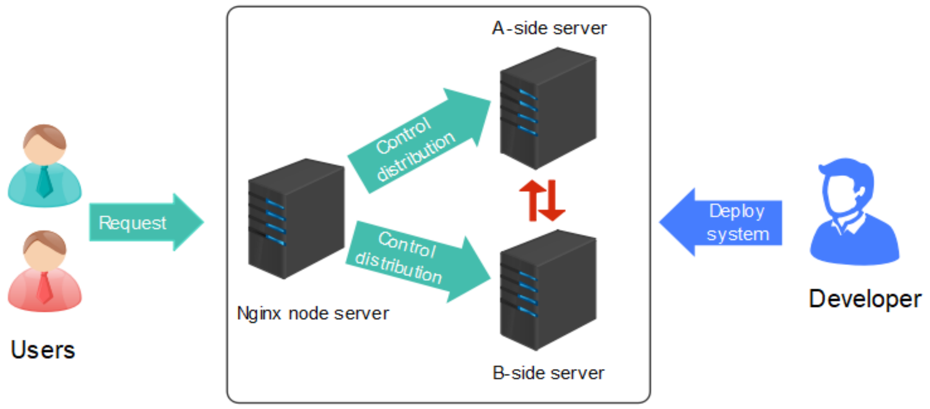
When you upgrade a new system, services may be stopped or gray verification may fail. In this practice, you can use the nginx load balancing mechanism for smooth system upgrade without affecting service running.

## Solution Architecture

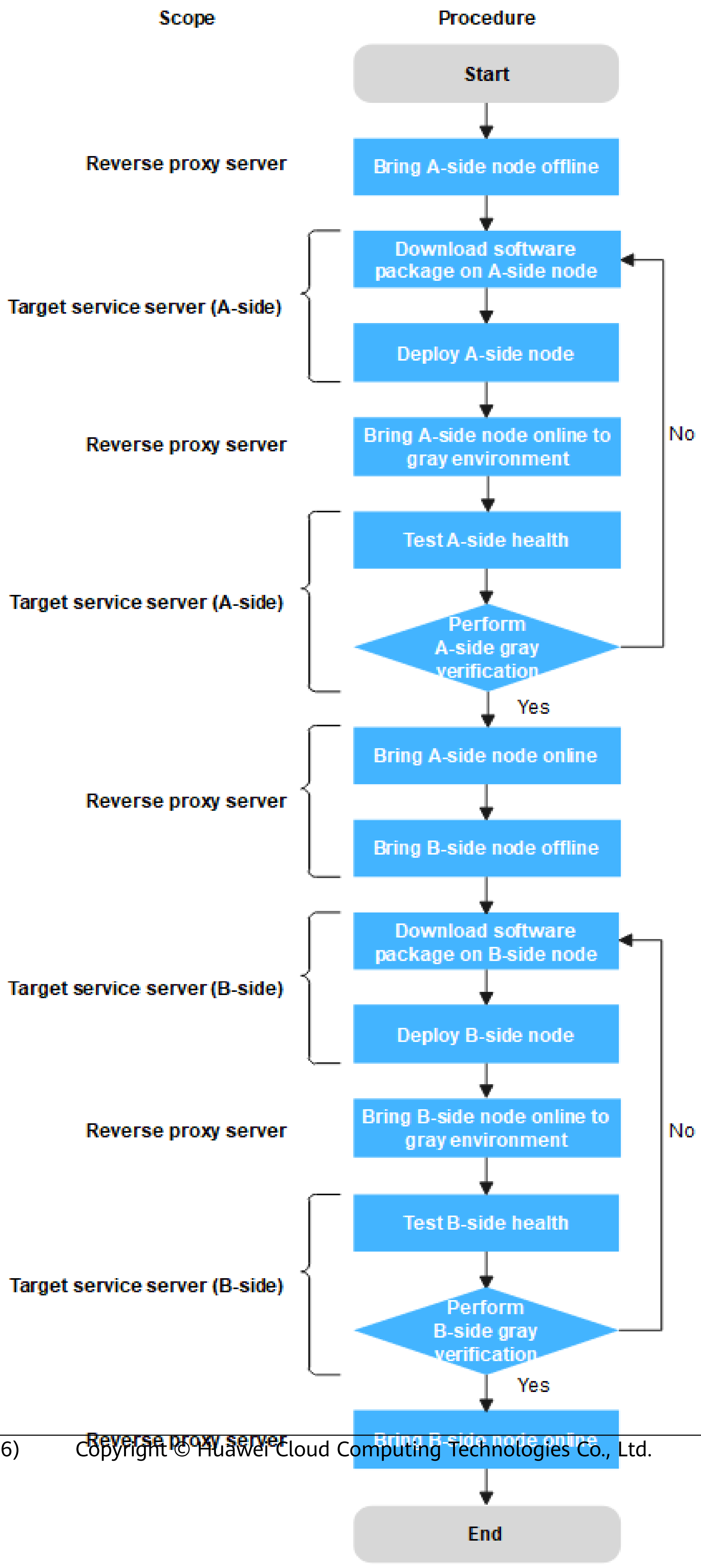
When upgrading the system using blue-green deployment, developers must carry out the following tasks:

1. Bring the A-side server (original blue environment) offline and redirect all access traffic to the B-side server. In this case, the server on A-side is upgraded.
2. After the A-side server is upgraded, set the server as the gray test environment. A tester performs gray verification on the A-side server.
3. After the gray verification is complete and the functions are normal, the A-side server (green environment) is officially released, and all traffic is distributed to the A-side server. In this case, the blue-green deployment is complete.
4. If an emergency occurs on the A-side server during service running, perform a blue-green switchover to quickly restore services.

**Figure 3-1** Gray release scheme



If you use canary release, repeat the preceding operations to upgrade the B-side server, complete the gray test, and release the system officially. In this case, the gray release of the new system is complete.


Scope
Procedure
Start
Reverse proxy server
Bring A-side node offline
Target service server (A-side)
Download software package on A-side node
Deploy A-side node
Reverse proxy server
Bring A-side node online to gray environment
No
Target service server (A-side)
Test A-side health
Perform A-side gray verification
Yes
Reverse proxy server
Bring A-side node online
Bring B-side node offline
Target service server (B-side)
Download software package on B-side node
Deploy B-side node
Reverse proxy server
Bring B-side node online to gray environment
No
Target service server (B-side)
Test B-side health
Perform B-side gray verification
Yes
Reverse proxy server
Bring B-side node online
End

## Prerequisites

- A project is available. If there is no project, **create one** first.
- You have the permission to create applications. For details, see **Editing Permissions**.
- Target service servers **A_test** and **B_test** are available, and application services are running on the servers.
- A reverse proxy server **Gray_release** is available.
- A gray verification host is available. This host represents a gray tester.

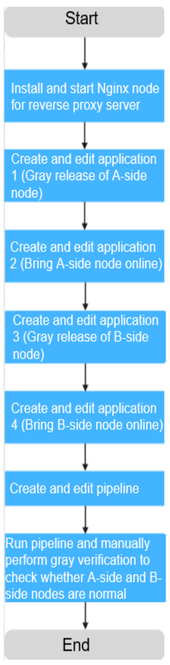
NOTE

Ensure that servers can communicate with each other. For example, add all servers to the same Virtual Private Cloud (VPC).

## Process

Based on the Nginx load balancing mechanism, this practice implements blue-green release and gray release in host deployment scenarios. For more information about Nginx, see **Nginx official website**.

**Figure 3-2** Process flowchart


Start
Install and start Nginx node for reverse proxy server
Create and edit application 1 (Gray release of A-side node)
Create and edit application 2 (Bring A-side node online)
Create and edit application 3 (Gray release of B-side node)
Create and edit application 4 (Bring B-side node online)
Create and edit pipeline
Run pipeline and manually perform gray verification to check whether A-side and B-side nodes are normal
End

**Step 1** **(Optional) Install and start an Nginx node for a reverse proxy server.**

NOTE

If the Nginx node has been installed and started on your reverse proxy server, skip this step.

1. Create basic resources.
   a. Go to the CodeArts homepage and click the target project name to access it.
   b. Choose **Settings** > **General** > **Basic Resources**. The **Host Clusters** page is displayed by default.

> **NOTE**
>
> Choose **CICD** > **Deploy**. Click **Basic Resources**. The **Host Clusters** page is displayed by default.

   c. Click **Create Host Cluster**, enter basic information such as the **Cluster Name**, **OS**, **Use Proxy**, **Execution Host**, and **Description**, and click **Save**.
   d. Click **Add Host**, select **Adding IP**, enter the **Host Name** (for example, **A_test**, **B_test**, or **Gray_release**), **IP**, **Username**, **Password** or **Key**, and **SSH Port**, and click **OK**. Repeat the preceding steps to create the three target hosts and verify the connectivity.

2. Create an application.
   a. Choose **CICD** > **Deploy**.
   b. Click **Create Application**. On the **Set Basic Information** page, modify the basic information such as **Name**, **Description**, and **Execution Host** as required.
   c. After editing the basic application information, click **Next**. The deployment template selection page is displayed.
   d. Select **Blank Template** and click **OK**. The **Deployment Actions** tab page is displayed.
3. Edit the application.
   a. Switch to the **Environment Management** tab page and add and edit an environment.
      - Click **Create Environment**, enter the environment name, for example, **Reverse_proxy_server_group**, select the OS corresponding to the server, and enter the description.
      - Click **Save**. The environment is created.
      - Click **Import Host**. The system automatically filters all clusters that meet the requirements of the current environment. In the dialog box that is displayed, select the target host cluster and click ↙ in the **Operation** column of the target host to import the host to the environment.
   b. Switch to the **Deployment Actions** tab page. Add and edit the following steps:
      - Add the **Install Nginx** action and modify the parameters in the following table (Linux is used as an example).

**Table 3-1** Parameter description

| Parameter | Description |
|---|---|
| **Environment** | Select **Reverse_proxy_server_group**. |
| **Nginx Version** | Select the target version. Example: **nginx-1.14.2**. |

| Parameter | Description |
|---|---|
| **Installation Path** | Enter the installation path of the Nginx service in the target environment. Example: **/usr/local/nginx**. |

- Add the **Start/Stop Nginx** action and modify the parameters in the following table (Linux is used as an example).

**Table 3-2** Parameter description

| Parameter | Description |
|---|---|
| **Environment** | Select **Reverse_proxy_server_group**. |
| **Operation** | Select **Start Nginx**. |
| **Installation Path** | Enter the installation path of the Nginx service in the target environment. Example: **/usr/local/nginx**. |

c. Click **Save & Deploy** to deploy the application.

4. Deploy the application.

   After the deployment is complete, the application status bar changes to green and the message **Successful** is displayed, indicating that the application is successfully deployed.

   If the application status bar turns red and displays **Failed**, the application fails to be deployed. In this case, click **View Solution**.

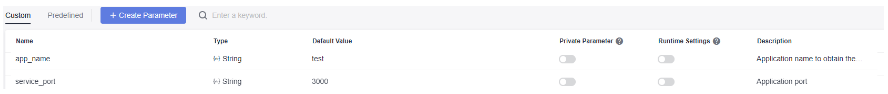
**Step 2 Create and edit application 1 (Gray release of A-side node).**

1. Create an application.
   a. Choose **CICD** > **Deploy**.
   b. Click **Create Application**. On the **Set Basic Information** page, modify the basic information such as **Name**, **Description**, and **Execution Host** as required.
   c. After editing the basic application information, click **Next**. The deployment template selection page is displayed.
   d. Select the **Deploy a General Application** template and click **OK**.
2. Edit the application.
   a. Switch to the **Environment Management** tab page and add and edit an environment.
      - Click **Create Environment**, enter the environment name, for example, **Reverse_proxy_server_group**, select the OS corresponding to the server, and enter the description.
      - Click **Save**. The environment is created.
      - Click **Import Host**. The system automatically filters all clusters that meet the requirements of the current environment. In the dialog box

that is displayed, select the target host cluster and click ↙ in the **Operation** column of the target host to import the host to the environment.

- Repeat the preceding steps to create target service server group **Target service server group_A-side node** and add the **A_test** server.

b. Switch to the **Parameters** tab page and add the following parameters:



Custom | Predefined | + Create Parameter | Enter a keyword.

| Name | Type | Default Value | Private Parameter | Runtime Settings | Description |
|---|---|---|---|---|---|
| app_name | String | test | | | Application name to obtain the... |
| service_port | String | 3000 | | | Application port |

c. Switch to the **Deployment Actions** tab page. Add and edit the following steps:

- Add the **Start/Stop Nginx** action and modify the parameters in the following table (Linux is used as an example).

**Table 3-3** Parameter description

| Parameter | Description |
|---|---|
| **Action Name** | Enter a name such as **Bring_A-side_node_offline**. |
| **Environment** | Select the target environment. Example: **Reverse_proxy_server_group**. |
| **Operation** | Specify the operation type Example: **Reload configuration file**. |
| **Installation Path** | Enter the installation path of the Nginx service in the target environment. Example: **/usr/local/nginx**. |
| **Modify configuration file before execution** | Select this parameter. |
| **Nginx Configuration File Path** | Enter the path of the Nginx configuration file on the target host. Example: **/usr/local/nginx/conf/nginx.conf**. |
| **Configuration File Backup Path** | Enter the target path for backing up the original Nginx configuration file on the target host. Example: **/usr/local/nginx/conf/nginx_bak.conf**. |
| **Configuration File Content** | Enter content of the new configuration file. See **Example code to bring A-side node offline** in the appendix. |

- Edit the **Download Software Package** action and change the parameter values to those listed in the following table (Linux is used as an example).

**Table 3-4** Parameter description

| Parameter | Description |
|---|---|
| **Action Name** | Enter a name **Download_software_package_on_A-side_node**. |
| **Source** | Select a source Example: **Artifact**. |
| **Environment** | Select the target environment. Example: **Target service server group_A-side node**. |
| **Software package** | Select a software package to be deployed in CodeArts Artifact. |
| **Download Path** | Enter the path for downloading the software package to the target host. Example: **/usr/local/**. |

- Edit the **Run Shell Commands** action and modify the parameters as follows (Linux is used as an example):

**Table 3-5** Parameter description

| Parameter | Description |
|---|---|
| **Action Name** | Enter the action name **Deploy A-side node**. |
| **Environment** | Select the target environment. Example: **Target service server group_A-side node**. |
| **Shell Commands** | Enter the commands to be executed. Example: See **Deployment node** in the appendix. |

- Add the **Start/Stop Nginx** action and modify the parameters in the following table (Linux is used as an example).

**Table 3-6** Parameter description

| Parameter | Description |
|---|---|
| **Action Name** | Enter a name such as **Bring A-side node online to gray environment**. |
| **Environment** | Select the target environment. Example: **Reverse_proxy_server_group**. |
| **Operation** | Specify the operation type Example: **Reload configuration file**. |
| **Installation Path** | Enter the installation path of the Nginx service in the target environment. Example: **/usr/local/nginx**. |

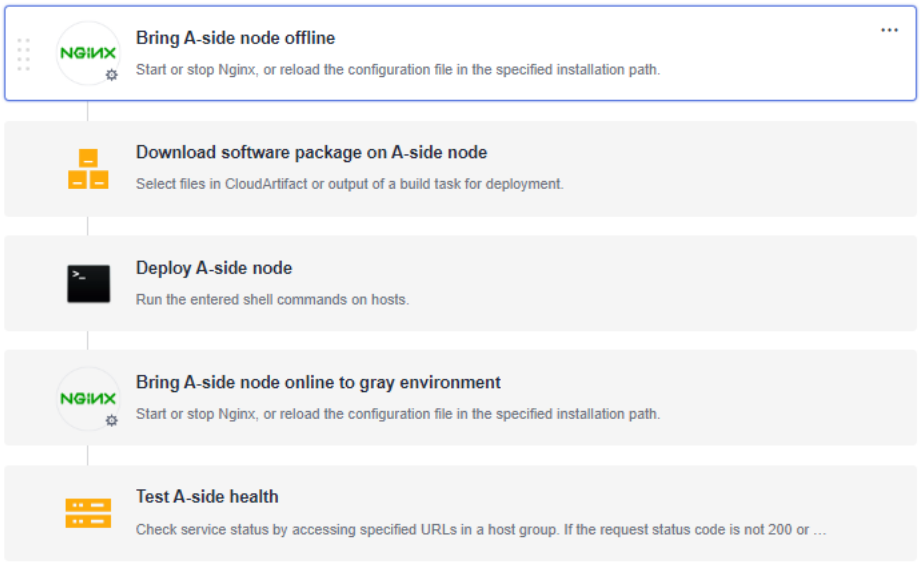
| Parameter | Description |
|---|---|
| **Modify configuration file before execution** | Select this parameter. |
| **Nginx Configuration File Path** | Enter the path of the Nginx configuration file on the target host. Example: **/usr/local/nginx/conf/nginx.conf**. |
| **Configuration File Backup Path** | Enter the target path for backing up the original Nginx configuration file on the target host. Example: **/usr/local/nginx/conf/nginx_bak.conf**. |
| **Configuration File Content** | Enter content of the new configuration file. See **Example code to bring A-side node online to the gray environment** in the appendix. |

- Edit the **Health Test via URLs** action and modify the parameters as follows (Linux is used as an example):

**Table 3-7** Parameter description

| Parameter | Description |
|---|---|
| **Action Name** | Enter a name such as **Test_A-side_health**. |
| **Environment** | Select the target environment. Example: **Target service server group_A-side node**. |
| **Retries** | If a service does not start up when the health test reaches the maximum retries, the service fails this test. Example: **1** |
| **Interval (s)** | Interval between two retries, in seconds. Example: **60** |
| **Test Path** | Used for the health test via URLs. You can add multiple URLs. Example: http://127.0.0.1:3000 (IP address and port number of the application service) |

3. Click **Save**. The application is created.


Bring A-side node offline
Start or stop Nginx, or reload the configuration file in the specified installation path.
Download software package on A-side node
Select files in CloudArtifact or output of a build task for deployment.
Deploy A-side node
Run the entered shell commands on hosts.
Bring A-side node online to gray environment
Start or stop Nginx, or reload the configuration file in the specified installation path.
Test A-side health
Check service status by accessing specified URLs in a host group. If the request status code is not 200 or ...

**Step 3 Create and edit application 2 (Bring A-side node online).**

1. Create an application.
   a. Click **Create Application**. On the **Set Basic Information** page, modify the basic information such as **Name**, **Description**, and **Execution Host** as required.
   b. After editing the basic application information, click **Next**. The deployment template selection page is displayed.
   c. Select **Blank Template** and click **OK**.
2. Edit the application.
   a. Switch to the **Environment Management** tab page and add and edit an environment.
      - Click **Create Environment**, enter the environment name, for example, **Reverse_proxy_server_group**, select the OS corresponding to the server, and enter the description.
      - Click **Save**. The environment is created.
      - Click **Import Host**. The system automatically filters all clusters that meet the requirements of the current environment. In the dialog box that is displayed, select the target host cluster and click in the **Operation** column of the target host to import the host to the environment.
   b. Switch to the **Deployment Actions** tab page. Add and edit the following steps:

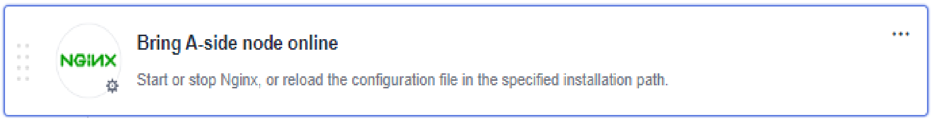
      Add the **Start/Stop Nginx** action and modify the parameters in the following table (Linux is used as an example).

04_Deploy_Best Practice_Public Cloud-Copy

**Table 3-8** Parameter description

| Parameter | Description |
|---|---|
| **Action Name** | Enter a name such as **Bring_A-side_node_online**. |
| **Environment** | Select the target environment. Example: **Reverse_proxy_server_group**. |
| **Operation** | Specify the operation type Example: **Reload configuration file**. |
| **Installation Path** | Enter the installation path of the Nginx service in the target environment. Example: **/usr/local/nginx**. |
| **Modify configuration file before execution** | Select this parameter. |
| **Nginx Configuration File Path** | Enter the path of the Nginx configuration file on the target host. Example: **/usr/local/nginx/conf/nginx.conf**. |
| **Configuration File Backup Path** | Enter the target path for backing up the original Nginx configuration file on the target host. Example: **/usr/local/nginx/conf/nginx_bak.conf**. |
| **Configuration File Content** | Enter content of the new configuration file. See **Example code to bring a node online** in the appendix. |

3. Click **Save**. The application is created.



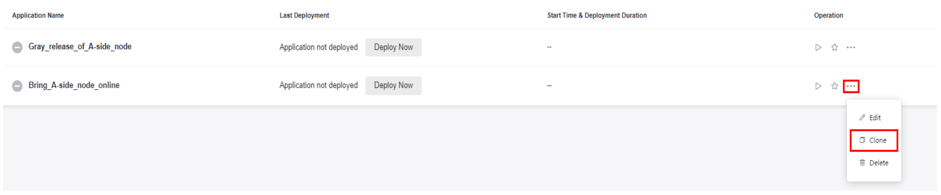
Bring A-side node online

Start or stop Nginx, or reload the configuration file in the specified installation path.

**Step 4 Clone and edit application 1. Create application 3 (gray release of B-side node).**

1. Clone an application.

   Click ••• and choose **Clone**.



| Application Name | Last Deployment | | Start Time & Deployment Duration | Operation |
|---|---|---|---|---|
| Gray_release_of_A-side_node | Application not deployed | Deploy Now | -- | |
| Bring_A-side_node_online | Application not deployed | Deploy Now | -- | |

2. Edit the application.

   a. Switch to the **Environment Management** tab page and add and edit an environment.

04_Deploy_Best Practice_Public Cloud-Copy

- Click **Create Environment**, enter the environment name, for example, **Reverse_proxy_server_group**, select the OS corresponding to the server, and enter the description.
- Click **Save**. The environment is created.
- Click **Import Host**. The system automatically filters all clusters that meet the requirements of the current environment. In the dialog box that is displayed, select the target host cluster and click ↙ in the **Operation** column of the target host to import the host to the environment.
- Repeat the preceding steps to create target service server group **Target service server group_B-side node** and add the **B_test** server.

b. Switch to the **Deployment Actions** tab page. Add and edit the following steps:

- Edit the **Bring A-side node offline** action and modify the parameters as follows (Linux is used as an example):

**Table 3-9** Parameter description

| Parameter | Description |
|---|---|
| **Action Name** | Enter a name such as **Bring_B-side_node_offline**. |
| **Environment** | Select the target environment. Example: **Reverse_proxy_server_group**. |
| **Operation** | Specify the operation type Example: **Reload configuration file**. |
| **Installation Path** | Enter the installation path of the Nginx service in the target environment. Example: **/usr/local/nginx**. |
| **Modify configuration file before execution** | Select this parameter. |
| **Nginx Configuration File Path** | Enter the path of the Nginx configuration file on the target host. Example: **/usr/local/nginx/conf/nginx.conf**. |
| **Configuration File Backup Path** | Enter the target path for backing up the original Nginx configuration file on the target host. Example: **/usr/local/nginx/conf/nginx_bak.conf**. |
| **Configuration File Content** | Enter content of the new configuration file. See **Example code to bring B-side node offline** in the appendix. |

- Edit the **Download software package on A-side node** action and change the parameter values to those listed in the following table (Linux is used as an example).

**Table 3-10** Parameter description

| Parameter | Description |
|---|---|
| **Action Name** | Enter a name such as **Download_software_package_on_B-side_node**. |
| **Source** | Select a source Example: **Artifact**. |
| **Environment** | Select the target environment. Example: **B_group**. |
| **Software package** | Select a software package to be deployed in CodeArts Artifact. |
| **Download Path** | Enter the path for downloading the software package to the target host. Example: **/usr/local/**. |

- Edit the **Deploy A-side node** action and modify the parameters as follows (Linux is used as an example):

**Table 3-11** Parameter description

| Parameter | Description |
|---|---|
| **Action Name** | Enter an action name such as **Deploy_B-side_node**. |
| **Environment** | Select the target environment. Example: **B_group**. |
| **Shell Commands** | Enter the commands to be executed. Example: See **Deployment node** in the appendix. |

- Edit the **Bring A-side node online to gray environment** action and modify the parameters as follows (Linux is used as an example):

**Table 3-12** Parameter description

| Parameter | Description |
|---|---|
| **Action Name** | Enter a name such as **Bring_B-side_node_online_to_gray_environment**. |
| **Environment** | Select the target environment. Example: **Reverse_proxy_server_group**. |

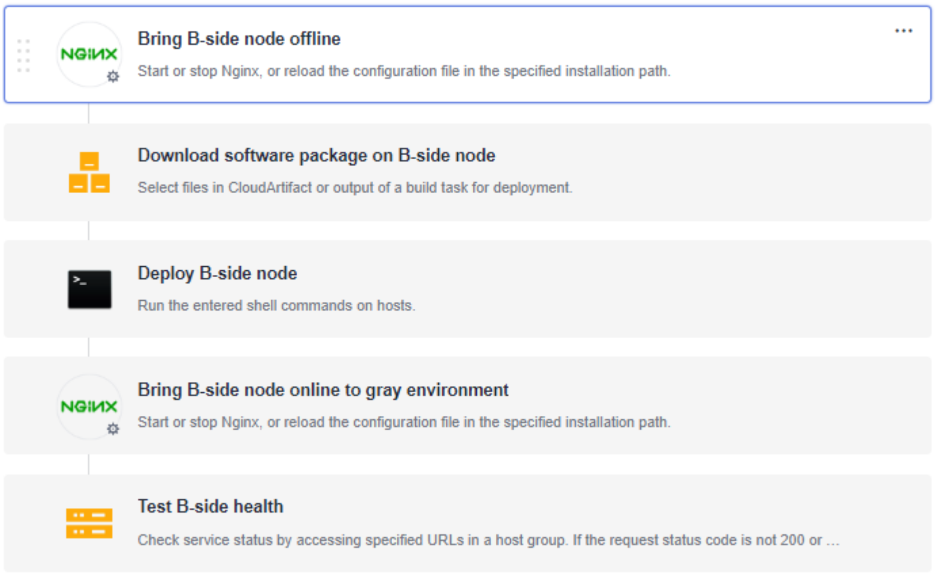
| Parameter | Description |
|---|---|
| **Operation** | Specify the operation type Example: **Reload configuration file**. |
| **Installation Path** | Enter the installation path of the Nginx service in the target environment. Example: **/usr/local/nginx**. |
| **Modify configuration file before execution** | Select this parameter. |
| **Nginx Configuration File Path** | Enter the path of the Nginx configuration file on the target host. Example: **/usr/local/nginx/conf/nginx.conf**. |
| **Configuration File Backup Path** | Enter the target path for backing up the original Nginx configuration file on the target host. Example: **/usr/local/nginx/conf/nginx_bak.conf**. |
| **Configuration File Content** | Enter content of the new configuration file. See **Example code to bring B-side node online to the gray environment** in the appendix. |

- Edit the **Test A-side health** action and modify the parameters as follows (Linux is used as an example):

**Table 3-13** Parameter description

| Parameter | Description |
|---|---|
| **Action Name** | Enter a name such as **Test_B-side_health**. |
| **Environment** | Select the target environment. Example: **B_group**. |
| **Retries** | If a service does not start up when the health test reaches the maximum retries, the service fails this test. Example: **1** |
| **Interval (s)** | Interval between two retries, in seconds. Example: **60** |
| **Test Path** | Used for the health test via URLs. You can add multiple URLs. Example: http://127.0.0.1:3000 (IP address and port number of the application service) |

3. Click **Save**. The application is created.


Bring B-side node offline
Start or stop Nginx, or reload the configuration file in the specified installation path.

Download software package on B-side node
Select files in CloudArtifact or output of a build task for deployment.

Deploy B-side node
Run the entered shell commands on hosts.

Bring B-side node online to gray environment
Start or stop Nginx, or reload the configuration file in the specified installation path.

Test B-side health
Check service status by accessing specified URLs in a host group. If the request status code is not 200 or ...

**Step 5** **Clone and edit application 2. Create application 4 (Bring B-side node online).**

1. Clone an application.

   Click ··· and choose **Clone**.

2. Edit the application.

   a. Switch to the **Environment Management** tab page and add and edit an environment.

      ▪ Click **Create Environment**, enter the environment name, for example, **Reverse_proxy_server_group**, select the OS corresponding to the server, and enter the description.

      ▪ Click **Save**. The environment is created.

      ▪ Click **Import Host**. The system automatically filters all clusters that meet the requirements of the current environment. In the dialog box that is displayed, select the target host cluster and click in the **Operation** column of the target host to import the host to the environment.

   b. Switch to the **Deployment Actions** tab page. Add and edit the following steps:

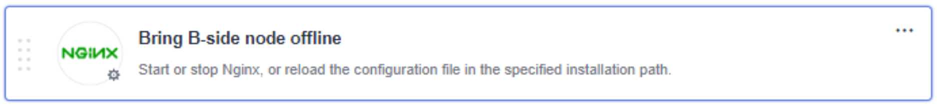
      Edit the **Bring A-side node online** action and modify the parameters as follows (Linux is used as an example):

**Table 3-14** Parameter description

| Parameter | Description |
|---|---|
| **Action Name** | Enter a name such as **Bring_B-side_node_online**. |
| **Environment** | Select the target environment. Example: **Reverse_proxy_server_group**. |

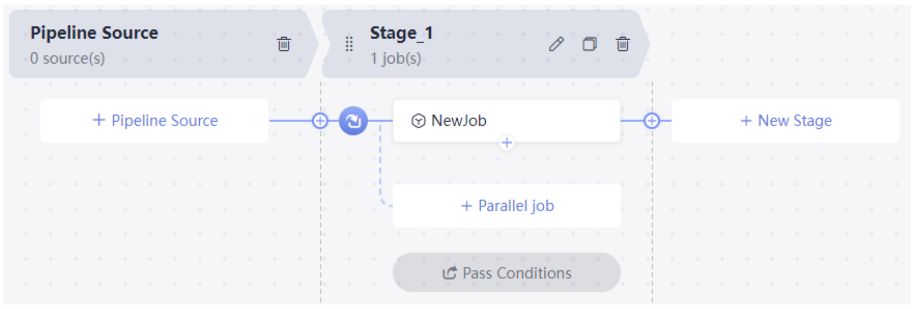
| Parameter | Description |
|---|---|
| **Operation** | Specify the operation type Example: **Reload configuration file**. |
| **Installation Path** | Enter the installation path of the Nginx service in the target environment. Example: **/usr/local/nginx**. |
| **Modify configuration file before execution** | Select this parameter. |
| **Nginx Configuration File Path** | Enter the path of the Nginx configuration file on the target host. Example: **/usr/local/nginx/conf/nginx.conf**. |
| **Configuration File Backup Path** | Enter the target path for backing up the original Nginx configuration file on the target host. Example: **/usr/local/nginx/conf/nginx_bak.conf**. |
| **Configuration File Content** | Enter content of the new configuration file. See **Example code to bring a node online** in the appendix. |

3. Click **Save**. The application is created.


Bring B-side node offline
Start or stop Nginx, or reload the configuration file in the specified installation path.

**Step 6 Create and edit a pipeline.**

1. Create a pipeline.
   - Choose **CICD** > **Pipeline**.
   - Click **Create Pipeline**, select a **Project**, enter a **Name**, set **Pipeline Source** to **None**, and click **Next**.
   - Select **Blank Template** and click **OK**.



2. Edit job 1 (**Gray release of A-side node**) in the pipeline stage.
   - Click . In the dialog box that is displayed, set the parameters as follows and click **Confirm**.

**Table 3-15** Parameter description

| Parameter | Description |
|---|---|
| **Stage Name** | Enter a name such as **Gray_release_of A-side_node**. |
| **Always Run** | Select **No**. |

- Click . In the displayed dialog box, set **Entry Type** to **Automatic** and click **OK**.
- Click **New Job**, click the **Deploy** tab, select **Deploy**, and click **Add**. In the dialog box that is displayed, set the parameters as follows and click **OK**.

**Table 3-16** Parameter description

| Parameter | Description |
|---|---|
| **Name** | Enter a name such as **Gray_release_of A-side_node**. |
| **Select Task** | Select **Gray_release_of A-side_node**. |
| **Build Task** | Leave it not configured. |

3. Create and edit job 2 (**Bring_A-side_node_online**) in the pipeline stage.
   - Click  and . In the dialog box that is displayed, set the parameters as follows and click **Confirm**.

**Table 3-17** Parameter description

| Parameter | Description |
|---|---|
| **Stage Name** | Enter a name such as **Bring_A-side_node_online**. |
| **Always Run** | Select **No**. |

- Click . In the displayed dialog box, set **Entry Type** to **Automatic** and click **OK**.
- Click **New Job**. In the window that is displayed, click the **Normal** tab, select **Manual Review** and click **Add**, set the parameters as follows, and click **OK**.

**Table 3-18** Parameter description

| Parameter | Description |
|---|---|
| **Name** | Enter a name such as **Gray_release_of A-side_node**. |

| Parameter | Description |
|---|---|
| **Reviewer** | Select the service verification personnel. |
| **Review Mode** | Select **Review by all**. |
| **Timeout Processing** | Select **Review failed and pipeline terminated**. |
| **Review Duration** | Example: 4 hours. |
| **Description** | This parameter is optional. |

– Click +, click the **Deploy** tab, select **Deploy**, and click **Add**. In the dialog box that is displayed, set the parameters as follows and click **OK**.

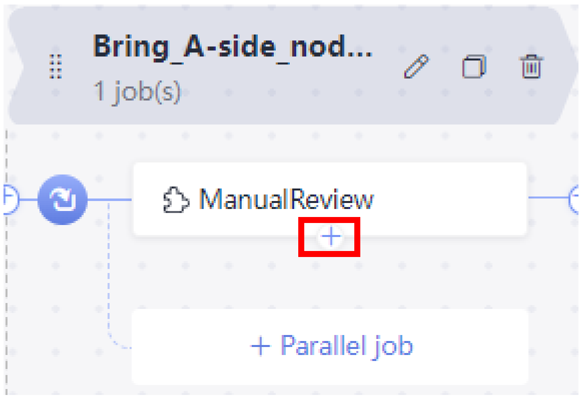


**Table 3-19** Parameter description

| Parameter | Description |
|---|---|
| **Name** | Enter a name such as **Bring_A-side_node_online**. |
| **Select Task** | Select **Bring_A-side_node_online**. |
| **Build Task** | Leave it not configured. |

4. Edit job 3 (**Gray_release_of_B-side_node**) in the pipeline stage.

– Click ⊕ and ✎. In the dialog box that is displayed, set the parameters as follows and click **Confirm**.

**Table 3-20** Parameter description

| Parameter | Description |
|---|---|
| **Stage Name** | Enter a name such as **Gray_release_of_B-side_node**. |

| Parameter | Description |
|---|---|
| **Always Run** | Select **No**. |

- Click . In the displayed dialog box, set **Entry Type** to **Automatic** and click **OK**.
- Click **New Job**, click the **Deploy** tab, select **Deploy**, and click **Add**. In the dialog box that is displayed, set the parameters as follows and click **OK**.

**Table 3-21** Parameter description

| Parameter | Description |
|---|---|
| **Name** | Enter a name such as **Gray_release_of_B-side_node**. |
| **Select Task** | Select **Gray_release_of_B-side_node**. |
| **Build Task** | Leave it not configured. |

5. Create and edit job 4 (**Bring_B-side_node_online**) in the pipeline stage.
   - Click  and . In the dialog box that is displayed, set the parameters as follows and click **Confirm**.

**Table 3-22** Parameter description

| Parameter | Description |
|---|---|
| **Stage Name** | Enter a name such as **Bring_B-side_node_online**. |
| **Always Run** | Select **No**. |

- Click . In the displayed dialog box, set **Entry Type** to **Automatic** and click **OK**.
- Click **New Job**. In the window that is displayed, click the **Normal** tab, select **Manual Review** and click **Add**, set the parameters as follows, and click **OK**.

**Table 3-23** Parameter description

| Parameter | Description |
|---|---|
| **Name** | Enter a name such as **Gray_verification_of_B-side_node**. |
| **Reviewer** | Select the service verification personnel. |

04_Deploy_Best Practice_Public Cloud-Copy

| Parameter | Description |
|---|---|
| **Review Mode** | Select **Review by all**. |
| **Timeout Processing** | Select **Review failed and pipeline terminated**. |
| **Review Duration** | Example: 4 hours. |
| **Description** | This parameter is optional. |

– Click , click the **Deploy** tab, select **Deploy**, and click **Add**. In the dialog box that is displayed, set the parameters as follows and click **OK**.

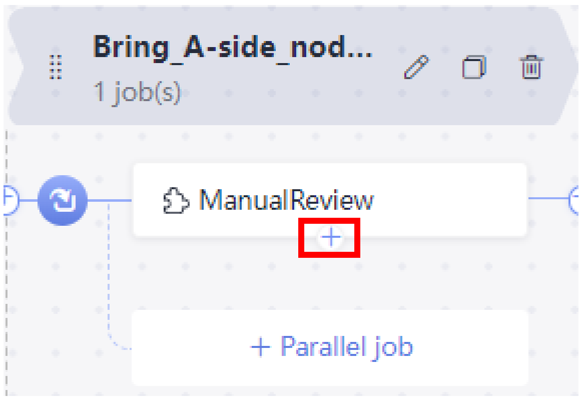
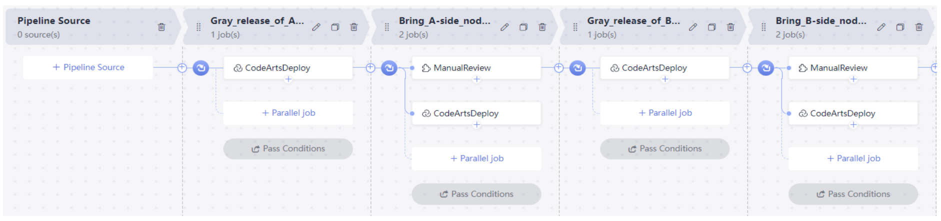


**Table 3-24** Parameter description

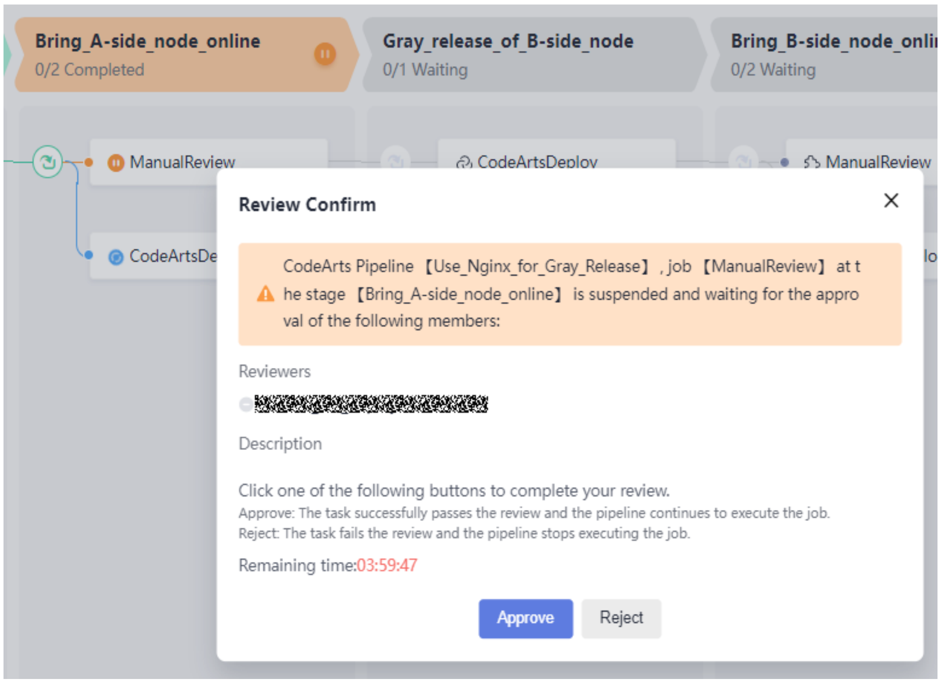
| Parameter | Description |
|---|---|
| **Name** | Enter a name such as **Bring_B-side_node_online**. |
| **Select Task** | Select **Bring_B-side_node_online**. |
| **Build Task** | Leave it not configured. |

6. After the preceding operations are complete, click **Save and Run** to run pipeline jobs.



**Step 7** **Run the pipeline and manually perform gray verification to check whether A-side and B-side nodes are normal.**

When CodeArts Pipeline is executed to bring node A or B online, pipeline execution is suspended. Gray users need to manually verify whether the servers on node A or B in the gray environment are working. Continue to run the pipeline if the servers are working.



Review Confirm

CodeArts Pipeline 【Use_Nginx_for_Gray_Release】, job 【ManualReview】 at the stage 【Bring_A-side_node_online】 is suspended and waiting for the approval of the following members:

Reviewers

[redacted]

Description

Click one of the following buttons to complete your review.
Approve: The task successfully passes the review and the pipeline continues to execute the job.
Reject: The task fails the review and the pipeline stops executing the job.

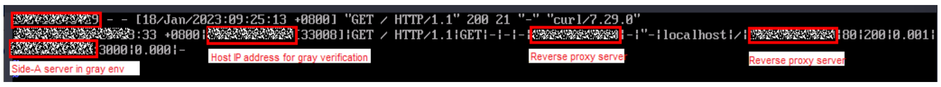
Remaining time:03:59:47

Gray users can run the **curl** command to check whether the gray environment is normal.

```
curl http://IP address of the reverse proxy server:Nginx port
```

**NOTE**

To check whether the gray user has accessed the target gray environment server, log in to the **reverse proxy server** and go to the path **logs/access.log** to view logs.



```
[redacted] - - [18/Jan/2023:09:25:13 +0800] "GET / HTTP/1.1" 200 21 "-" "curl/7.29.0"
[redacted]3:33 +0800|[redacted][33008]|GET / HTTP/1.1|GET|-|-|-|[redacted]|-|"-|localhost|/|[redacted]|80|200|0.001|
[redacted]3000|0.000|-
```

Side-A server in gray env
Host IP address for gray verification
Reverse proxy server
Reverse proxy server

**----End**

## Appendixes

- **Example code to bring A-side node offline**

```
worker_processes  1;
events {
    worker_connections  1024;
}
http {
    include       mime.types;
    default_type  application/octet-stream;
    log_format main  '$time_local|$remote_addr[$remote_port]|$request|$request_method|
$content_length|'
        '$content_type|$http_referer|$host|$http_x_forwarded_for|'
        '$http_true_client_ip|$server_name|$request_uri|$server_addr|$server_port|'
        '$status|$request_time|$upstream_addr|$upstream_response_time|$cookie_domain_tag';
    access_log logs/access.log main; #Access log: storage path and log level
    error_log logs/error.log; #Error log: storage path
    sendfile        on;
    keepalive_timeout  65;
    upstream portal {
        #Enter the IP address and application service port number of host A.
        #server X.X.X.X:X;      #Bring node A offline.
        #Enter the IP address and application service port number of host B.
        server X.X.X.X:X;
    }
    upstream portal_test {
        #Enter the IP address and application service port number of host A.
        server X.X.X.X:X;
        #Enter the IP address and application service port number of host B.
        server X.X.X.X:X;
    }

    server {
        listen       XXX;#Enter the Nginx port number.
        server_name  localhost;

        location / {
            set $backend portal;
            set $test portal_test;
            #Enter the IP address of the gray verification host.
            #if ( $remote_addr ~* "X.X.X.X") {
            #    set $backend $test;
            #}
            proxy_pass https://$backend;
        }
        error_page   500 502 503 504  /50x.html;
        location = /50x.html {
            root   html;
        }
   }
}
```

- **Deployment node**

```
#Obtain the application process ID.
pid=`ps -ef | grep app_name | grep -v grep | awk '{print $2}'`
if [ -z "$pid" ];
then
  echo "[app_name pid is not exist.]"
else
  echo "app_name pid: $pid "
  #End the process.
  kill -15 $pid
fi
#Restart the application. You can run the deployment script or command to start the application.
#Method 1: Run the deployment script to start the application.
# sh startup.sh
#Method 2: Run the command to start the application. nohup is recommended for backend startup.
# nohup java -jar /usr/local/app/SpringbootDemo.jar &
```

- **Example code to bring A-side node online to the gray environment**

```
worker_processes  1;
events {
    worker_connections  1024;
}
http {
    include       mime.types;
    default_type  application/octet-stream;
    log_format main  '$time_local|$remote_addr[$remote_port]|$request|$request_method|
$content_length|'
        '$content_type|$http_referer|$host|$http_x_forwarded_for|'
        '$http_true_client_ip|$server_name|$request_uri|$server_addr|$server_port|'
        '$status|$request_time|$upstream_addr|$upstream_response_time|$cookie_domain_tag';
    access_log logs/access.log main; #Access log: storage path and log level
    error_log logs/error.log; #Error log: storage path
    sendfile        on;
    keepalive_timeout  65;
    upstream portal {
        #Enter the IP address and application service port number of host A.
        #server X.X.X.X:X;      #Bring node A offline.
        #Enter the IP address and application service port number of host B.
        server X.X.X.X:X;
    }
    upstream portal_test {
        #Enter the IP address and application service port number of host A.
        server X.X.X.X:X;       #Gray release of node A
        #Enter the IP address and application service port number of host B.
        #server X.X.X.X:X;
    }

    server {
        listen       XXX;#Enter the Nginx port number.
        server_name  localhost;

        location / {
            set $backend portal;
            set $test portal_test;
            #Enter the IP address of the gray verification host.
            if ( $remote_addr ~* "X.X.X.X") {
                set $backend $test;
            }
            proxy_pass https://$backend;
        }
        error_page   500 502 503 504  /50x.html;
        location = /50x.html {
            root   html;
        }
    }
}
```

- **Example code to bring B-side node offline**

```
worker_processes  1;
events {
    worker_connections  1024;
}
http {
    include       mime.types;
    default_type  application/octet-stream;
    log_format main  '$time_local|$remote_addr[$remote_port]|$request|$request_method|
$content_length|'
        '$content_type|$http_referer|$host|$http_x_forwarded_for|'
        '$http_true_client_ip|$server_name|$request_uri|$server_addr|$server_port|'
        '$status|$request_time|$upstream_addr|$upstream_response_time|$cookie_domain_tag';
    access_log logs/access.log main; #Access log: storage path and log level
    error_log logs/error.log; #Error log: storage path
    sendfile        on;
    keepalive_timeout  65;
    upstream portal {
        #Enter the IP address and application service port number of host A.
        server X.X.X.X:X;
        #Enter the IP address and application service port number of host B.
        #server X.X.X.X:X;      #Bring node B offline.
    }
    upstream portal_test {
        #Enter the IP address and application service port number of host A.
        server X.X.X.X:X;
        #Enter the IP address and application service port number of host B.
        server X.X.X.X:X;
    }

    server {
        listen       XXX;#Enter the Nginx port number.
        server_name  localhost;

        location / {
            set $backend portal;
            set $test portal_test;
            #Enter the IP address of the gray verification host.
            #if ( $remote_addr ~* "X.X.X.X") {
            #    set $backend $test;
            #}
            proxy_pass https://$backend;
        }
        error_page   500 502 503 504  /50x.html;
        location = /50x.html {
            root   html;
        }
    }
}
```

- **Example code to bring B-side node online to the gray environment**

```
worker_processes  1;
events {
    worker_connections  1024;
}
http {
    include       mime.types;
    default_type  application/octet-stream;
    log_format main  '$time_local|$remote_addr[$remote_port]|$request|$request_method|
$content_length|'
        '$content_type|$http_referer|$host|$http_x_forwarded_for|'
        '$http_true_client_ip|$server_name|$request_uri|$server_addr|$server_port|'
        '$status|$request_time|$upstream_addr|$upstream_response_time|$cookie_domain_tag';
    access_log logs/access.log main; #Access log: storage path and log level
    error_log logs/error.log; #Error log: storage path
    sendfile        on;
    keepalive_timeout  65;
    upstream portal {
        #Enter the IP address and application service port number of host A.
        server X.X.X.X:X;
        #Enter the IP address and application service port number of host B.
        #server X.X.X.X:X;      #Bring node B offline.
    }
    upstream portal_test {
        #Enter the IP address and application service port number of host A.
        #server X.X.X.X:X;
        #Enter the IP address and application service port number of host B.
        server X.X.X.X:X;       # Gray release of node B
    }

    server {
        listen       XXX;#Enter the Nginx port number.
        server_name  localhost;

        location / {
            set $backend portal;
            set $test portal_test;
            #Enter the IP address of the gray verification host.
            if ( $remote_addr ~* "X.X.X.X") {
                set $backend $test;
            }
            proxy_pass https://$backend;
        }
        error_page   500 502 503 504  /50x.html;
        location = /50x.html {
            root   html;
        }
    }
}
```

- **Example code to bring a node online**

```
worker_processes  1;
events {
    worker_connections  1024;
}
http {
    include       mime.types;
    default_type  application/octet-stream;
    log_format main  '$time_local|$remote_addr[$remote_port]|$request|$request_method|
$content_length|'
        '$content_type|$http_referer|$host|$http_x_forwarded_for|'
        '$http_true_client_ip|$server_name|$request_uri|$server_addr|$server_port|'
        '$status|$request_time|$upstream_addr|$upstream_response_time|$cookie_domain_tag';
    access_log logs/access.log main; #Access log: storage path and log level
    error_log logs/error.log; #Error log: storage path
    sendfile        on;
    keepalive_timeout  65;
    upstream portal {
        #Enter the IP address and application service port number of host A.
        server X.X.X.X:X;
        #Enter the IP address and application service port number of host B.
        server X.X.X.X:X;
    }
    upstream portal_test {
        #Enter the IP address and application service port number of host A.
        server X.X.X.X:X;
        #Enter the IP address and application service port number of host B.
        server X.X.X.X:X;
    }

    server {
        listen       XXX;#Enter the Nginx port number.
        server_name  localhost;

        location / {
            set $backend portal;
            set $test portal_test;
            #Enter the IP address of the gray verification host.
            #if ( $remote_addr ~* "X.X.X.X") {
            #    set $backend $test;
            #}
            proxy_pass https://$backend;
        }
        error_page   500 502 503 504  /50x.html;
        location = /50x.html {
            root   html;
        }
    }
}
```

# 4 Using Kubernetes Nginx-Ingress for Gray Release

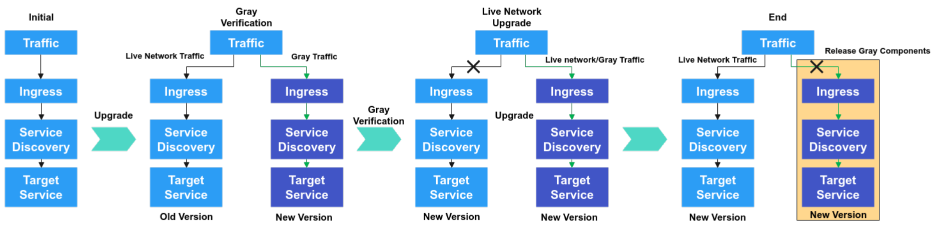
## Application Scenario

This practice implements gray release based on native Kubernetes features. When you upgrade a new system, services may be stopped or gray verification may fail. The native Kubernetes service features help you upgrade system smoothly without affecting services.

## Solution Architecture

During system upgrade, a group of gray loads are created when developers deploy applications for the first time. In this case, the system version in the gray loads is the new version. The Service forwards some traffic to the gray loads, and the testers verify the version in the gray loads. After the version verification is complete, the developers start to deploy the application for the second time to upgrade the services on the live network. In this case, the Service forwards all traffic to the gray loads and upgrades the services to the latest version on the live network. After the upgrade is complete, the Service forwards all traffic back to the live network load and releases the gray loads. Now, the new system is released.

**Figure 4-1** Gray release scheme



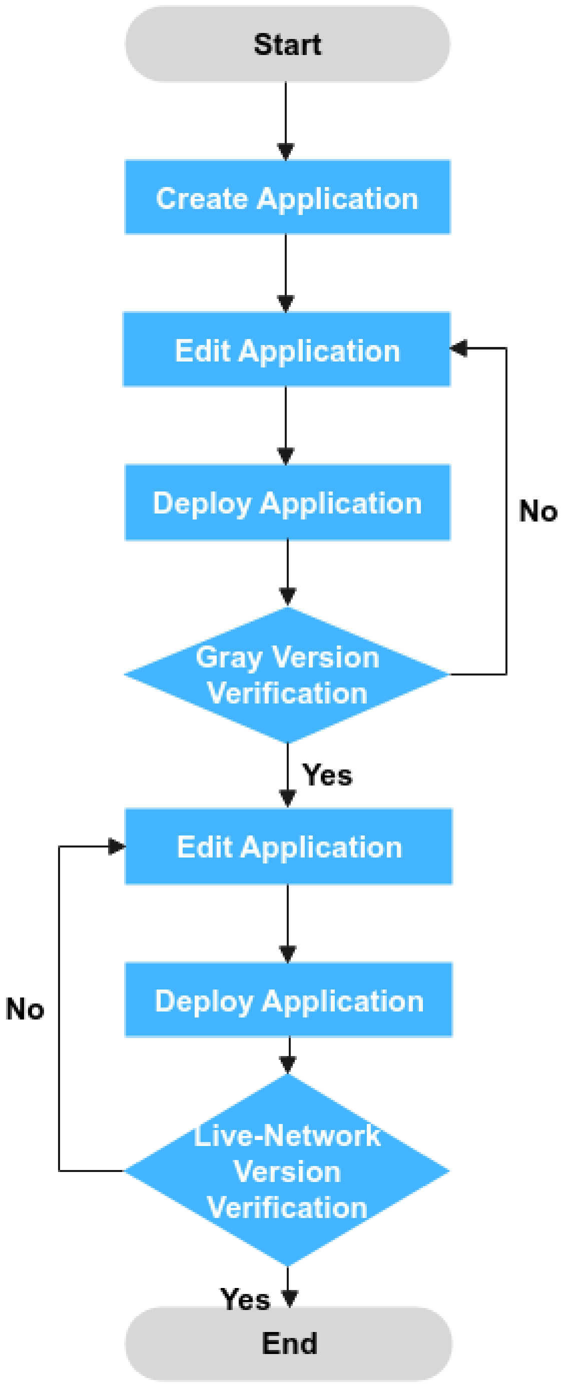
## Prerequisites

- A project is available. If there is no project, **create one** first.
- You have the permission to create applications. For details, see **Editing Permissions**.
- The service contains the following resources and they are defined as version 1:

- A CCE cluster, for example, **cce-demo**, is available.
- A deployment, for example, **deployment-doc**, has been created in the CCE cluster.
- A Service, for example, **service-doc**, has been created in the CCE cluster.
- A route, for example, **ingress-doc**, has been created in the CCE cluster.
- The nginx-ingress plug-in has been installed in the CCE cluster.

## Process

**Figure 4-2** Process flowchart



**Step 1** Create an application.

1. Go to the CodeArts homepage and click the target project name to access the project.
2. Choose **CICD** > **Deploy** and click **Create Application**. The **Set Basic Information** page is displayed.
3. You can modify the following basic information as required:

| Parameter | Description |
|---|---|
| Name | Mandatory. Name of an application. Example: **Kubernetes_Nginx-Ingress_Gray_Deployment** |
| Project | Retain the default value. Project to which an application belongs. |
| Description | Optional. Description of an application. Example: None |
| Execution Resource Pool | Optional. A resource pool is a collection of physical environments where commands are executed during software package deployment. You can use the **official resource pool** hosted by Huawei Cloud or host your own servers as a **self-hosted resource pool** on Huawei Cloud. For details about how to host your own servers, see **Self-hosted Resource Pool**. Example: **Official** |
| Deploy from pipeline | Optional. After this function is enabled, the app can be executed only by the pipeline driver and cannot be executed independently. |

4. After editing the basic application information, click **Next**. On the **Select Template** page, select **Blank Template** and click **OK**.

**Step 2** Edit the application.

On the **Deployment Actions** tab page, add **Kubernetes Nginx-Ingress Gray Deployment (CCE cluster)** and modify the parameters described in the following table.

**Table 4-1** Parameter description

| Parameter | Description | Example |
|---|---|---|
| Action Name | Name of an action displayed in **Deployment Actions** area. | Retain the default value. |

| Parameter | Description | Example |
|---|---|---|
| Tenant | ● **Current tenant**: The software package is deployed in your CCE cluster for release. Select **Current tenant**. You must have the CCE cluster operation permission. If you do not have it, select **Authorized User** for deployment.<br>● **Other tenant**: The software package is deployed in the CCE cluster of another tenant for release in IAM authorization mode.<br>If you select **Other tenant**, you must select an authorized tenant to deploy the CCE cluster. | Select **Current tenant**. |
| Authorized User | If you do not have the permission to execute an API, this parameter enables you to obtain the temporary AK/SK of the parent user to execute the CCE API through IAM. | Deselect it. |
| Region | Select the region for deployment. | Retain the default value. |
| Cluster Name | Select the Kubernetes cluster applied on CCE. | cce-ldf |
| Namespace | Select the namespace of the Kubernetes cluster on CCE. | Retain the default value. |
| Workload | Select the target deployment. | deployment-doc |
| Service | Name of the service bound to the target workload. | service-doc |
| Ingress | Select the name of the route bound to the target service. | ingress-doc |
| Container | Select the name of the CCE container to be deployed. | container-1 |
| Image | Select the image to be deployed. | Retain the default value. |
| Image Tag | Select the tag of the image to be deployed. | v2 |

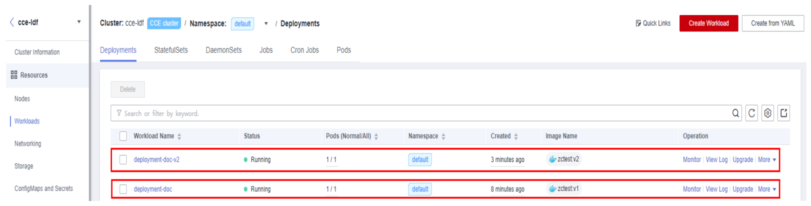
| Parameter | Description | Example |
|---|---|---|
| Enabling grayscale configuratio n | **Gray Strategy**:<br>• **Header**<br>**Header-Key**: You can enter the key of a custom header.<br>**Header-Value**: You can enter a custom header value. The value can be a character string or a regular expression. The regular expression format is **^....$**.<br>**Gray Traffic Weight (%)**: Traffic can be customized.<br>• **Cookie**<br>**Cookie**: Custom cookie content can be entered.<br>**Gray Traffic Weight (%)**: Traffic can be customized.<br>NOTE<br>The values of **Header** and **Cookie** can contain a maximum of 500 characters. | Selected<br>**Gray Strategy**: **Header**<br>**Header-Key**: **foo**<br>**Header-Value**: **bar**<br>**Gray Traffic Weight(%)**: **30** |

**Step 3** Deploy an application (create a gray version).

Click **Save & Deploy** to deploy the application. CodeArts Deploy has created the following gray version resources in the CCE cluster and diverts 30% of the live network traffic to the gray load.

- **Workload**: **deployment-doc-v2**. The image version is V2.

**Figure 4-3** Adding a workload whose image version is V2



| Workload Name | Status | Pods (Normal/All) | Namespace | Created | Image Name | Operation |
|---|---|---|---|---|---|---|
| deployment-doc-v2 | Running | 1 / 1 | default | 3 minutes ago | zctest:v2 | Monitor \| View Log \| Upgrade \| More |
| deployment-doc | Running | 1 / 1 | default | 8 minutes ago | zctest:v1 | Monitor \| View Log \| Upgrade \| More |

- **Service**: **service-doc-v2**
- **Route**: **ingress-doc-v2**

NOTE

In this case, you can add a data record (the value of **Key** is **foo** and the value of **Value** is **bar**) to the header to verify the latest version in the gray load.

**Step 4** Edit the application (deploy the latest version).

Go to the application created in **Step 1** and modify the following parameters.

**Table 4-2** Parameter description

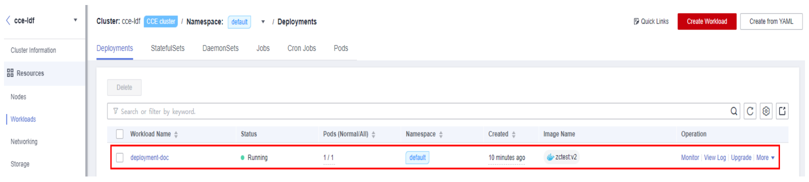
| Parameter | Example |
|---|---|
| Enabling grayscale configuration | Deselect it. |

**Step 5** Deploy the application (deploy the latest version).

Click **Save & Deploy** to deploy the application. CodeArts Deploy has deleted the following gray environment resources from the CCE cluster and replaced the V1 image with the V2 image:

- **Workload**: **deployment-doc-v2**
- **Service**: **service-doc-v2**
- **Route**: **ingress-doc-v2**

**Figure 4-4** The image version is upgraded to V2.



| Workload Name | Status | Pods (Normal/All) | Namespace | Created | Image Name | Operation |
|---|---|---|---|---|---|---|
| deployment-doc | Running | 1 / 1 | default | 10 minutes ago | zctest:v2 | Monitor \| View Log \| Upgrade \| More |

NOTE

You can check whether the system is the latest version on the live network.

**----End**

# 5 HE2E DevOps Practice - Deploying an Application

## 5.1 Overview

This section uses the DevOps process example project to describe how to deploy applications on CCE and ECS.

There are three deployment applications preset in the sample project.

The first application is used for CCE deployment, and the second and third applications are used for ECS deployment.

**Table 5-1** Preset applications

| Preset Application | Description |
|---|---|
| phoenix-cd-cce | Application deployed on CCE |
| phoenix-sample-standalone | Application deployed on ECS |
| phoenix-sample-predeploy | Application for which you install dependency tools on ECS |

## 5.2 Deploying an Application on CCE

### Buying and Configuring CCE

This section uses Cloud Container Engine (CCE) for deployment.

**Buy a CCE cluster** on the console.

For details about the mandatory configurations of clusters and nodes, see **Table 5-2** and **Table 5-3**. You can select the configurations that are not listed in the table based on the site requirements.

**Table 5-2** Buying a CCE cluster

| Category | Configuration Item | Suggestion |
|---|---|---|
| Basic Settings | Type | Select **CCE Standard Cluster**. |
| | Billing Mode | Select **Pay-per-use**. |
| | Cluster Name | Enter a name. |
| | Cluster Version | Select a version as required. You are advised to select the latest version. |
| Network Settings | Network Model | Select **Tunnel network**. |
| | VPC | Select an existing VPC. If no proper VPC is available in the list, click **Create VPC**. |
| | Default Node Subnet | Select an existing subnet. If no proper subnet is available in the list, click **Create Subnet**. |
| | Container CIDR Block | Click **Auto select**. |

**Table 5-3** Configuring a node

| Category | Configuration Item | Suggestion |
|---|---|---|
| Node Configuration | Billing Mode | Select **Pay-per-use**. |
| | Node Type | Select **Elastic Cloud Server (VM)**. |
| | Specifications | Select 2 vCPUs and 8 GiB memory or higher. |
| | OS | Click **Public image** and select an Euler image. |
| | Node Name | Enter a custom name. |

| Category | Configuration Item | Suggestion |
|---|---|---|
| | Login Mode | Select **Password**. |
| | Password | Enter a password. |
| Network Settings | Node IP | Select **Automatic**. |
| | EIP | Select **Auto create**. |

## Configuring and Executing an Application

**Step 1** Configure the application.

1. Go to the **Phoenix Mall** project and choose **CICD** > **Deploy**. The built-in deployment applications of the sample project are displayed on the page.
2. Find application **phoenix-cd-cce**. Click ··· and choose **Edit**.
3. On the **Deployment Actions** tab page, complete the following configurations in each action.

**Table 5-4** Configuring deployment actions

| Configuration Item | Suggestion |
|---|---|
| Cluster Name | Use the cluster name set when buying a CCE cluster. |
| Namespace | In this document, select **default**. |

4. Click the **Parameters** tab and set parameters.

**Table 5-5** Parameters

| Parameter | Example Value |
|---|---|
| ci_task_name | Enter **phoenix-sample-ci**. |
| version | Use the value of **version** of the **phoenix-sample-ci** task. |

5. Click **Save**.

**Step 2** Click **Console** on the navigation bar, and access CCE from the service list.

Locate the target cluster and click its name to go to the **Overview** page.

Choose **Workloads** from the navigation pane, click the **Deployments** tab, and verify that no record exists in the list.

If there are records in the list, select all records, click **Delete**, select all resource release options, and click **Yes** to clear the records in the list.

**Step 3** Return to the application list page, click ▷ in the row of the **phoenix-cd-cce** application, and click **OK** in the dialog box that is displayed to start deployment.

If ✓ is displayed on the page, the deployment is successful. If the deployment fails, rectify the fault based on the failed action and the error information in logs.

**Step 4** Verify the deployment result.

1. Go to the CCE console.
2. Locate the target cluster, click its name to go to the **Overview** page, and click the **Deployments** tab.

   Five records are displayed on the page. All of them are in the **Running** state.
3. Click **vote** to go to the details page. On the **Access Mode** tab page, choose **More** > **Update**.

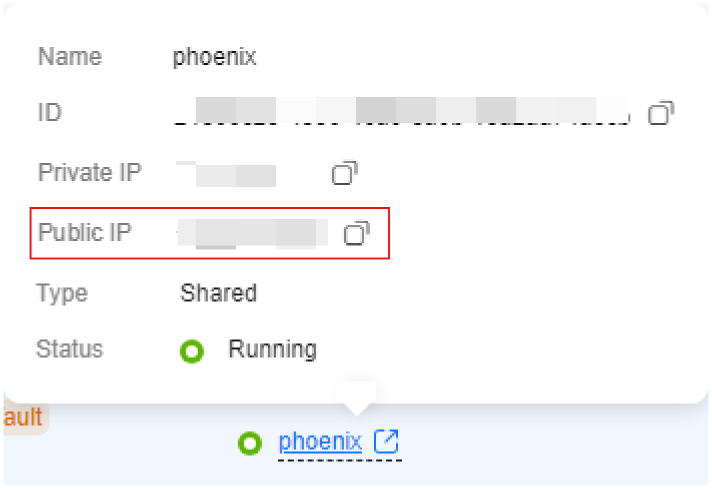
   Set the parameters by referring to **Table 5-6**, and click **OK**.

**Table 5-6** Updating a service

| Parameter | Example Value |
|---|---|
| Service Affinity | Select **Cluster-level**. |
| Load Balancer | – Choose **Shared** > **Auto create**.<br>– **Instance Name**: Enter **phoenix**.<br>– **EIP**: Select **Auto create**.<br>NOTE<br>If your account already has a load balancer, choose **Shared** > **Use existing** and select an existing load balancer. |
| Port | – **Container Port**: Set it to **80**.<br>– **Service Port**: Set it to **5000**. |

4. After the update is successful, return to the service list. Hover over the load balancer name of ● phoenix, and copy the public IP address.

**Figure 5-1** Copying the access address



| | |
|---|---|
| Name | phoenix |
| ID | |
| Private IP | |
| Public IP | |
| Type | Shared |
| Status | Running |

5. Open a new browser and enter **http://*IP*:5000** in the address box. *IP* is the public IP address recorded in **Step 4.4**. The Phoenix Mall homepage is displayed.
6. Return to the **Deployments** page and update **result** (select **phoenix** created in **Step 4.3** as the load balancer and enter service port **5001**) by referring to **Step 4.3**.

   After the creation is successful, enter **http://*IP*:5001** in a new browser. The dashboard of Phoenix Mall is displayed.

**----End**

## Releasing Resources

To avoid unnecessary fees, **delete the CCE cluster**.

If you need to create a pipeline after the deployment, you can delete the cluster after completing the pipeline practice.

## 5.3 Deploying an Application on ECS

### Purchasing and Configuring an ECS

This section uses Elastic Cloud Server (ECS). You can also use your own Linux host that runs Ubuntu 16.04 OS.

**Step 1** **Buy an ECS**.

The following table lists mandatory configurations. You can also select other configurations as necessary.

**Table 5-7** Configuring an ECS purchase

| Category | Configuration Item | Suggestion |
|---|---|---|
| Configure Basic Settings | Billing Mode | Select **Pay-per-use**. |
| Instance | CPU Architecture | Select **x86**. |
| | Specifications | Select 2 vCPUs and 8 GiB memory or higher. |
| OS | Image | Choose **Public image** > **Ubuntu** > **Ubuntu 16.04 server 64bit**. |
| Public Network Access | EIP | Select **Auto assign**. |
| | Billed By | Select **Bandwidth**. |
| Instance Management | Login Mode | Select **Password**. |
| | Password | Enter a custom password. |

**Step 2** Configure security group rules.

Use ports 5000 and 5001 to verify the sample project. Therefore, add an inbound rule that allows access over ports 5000 and 5001.

The procedure is as follows:

1. Log in to the ECS list page, locate the ECS purchased in step **Step 1**, and click the ECS name.
2. Click the **Security Groups** tab, and add an inbound rule with **Protocol** set to **TCP** and **Port Range** set to **5000-5001** by referring to **Configuring Security Group Rules**.

**----End**

## Adding a Target Host to the Project

Before deploying applications to ECSs, add the target hosts to the basic resources of the project.

**Step 1** Go to the **Phoenix Mall** project and choose **Settings** > **General** > **Basic Resources** from the navigation pane.

**Step 2** Click **Create Host Cluster**, configure the following information, and click **Save**.

04_Deploy_Best Practice_Public Cloud-Copy

**Table 5-8** Creating a host cluster

| Parameter | Suggestion |
|---|---|
| Cluster Name | Enter **host-group**. |
| OS | Select **Linux**. |
| Host Connection Mode | Select **Direct Connection**. |
| Execution Resource Pool | Select **Official**. |

**Step 3** Click **Add Host**, configure the following information, and click **OK**.

**Table 5-9** Adding a host

| Configuratio n Item | Suggestion |
|---|---|
| Add Hosts by | Select **Adding IP**. |
| Host Name | Keep this name same as the name of the purchased ECS. |
| IP | Enter the EIP generated when buying the ECS. |
| Authorization | Select **Password**. |
| Username | Enter **root**. |
| Password | Enter the password set when buying the ECS. |
| SSH Port | Enter **22**. |

**Step 4** A host record is displayed on the page. If **Succeed** is displayed in the **Verification Result** column, the host is added successfully.

If the host fails to be added, check the host configuration based on the failure details.

**----End**

## Installing Dependency Tools on ECS

The sample program depends on Docker and Docker-Compose, which must be installed on the target ECS.

**Step 1** Go to the **Phoenix Mall** project, choose **CICD** > **Deploy**, and find the **phoenix-sample-predeploy** application in the list.

**Step 2** Click ··· and choose **Edit** from the drop-down list.

**Step 3** Click the **Environment Management** tab and configure the host environment.

1. Click **Create Environment**, configure the following information, and click **Save**.

**Table 5-10** Creating an environment

| Parameter | Value |
|---|---|
| Environment | Enter **phoenix-env**. |
| Resource Type | Select **Host**. |
| OS | Select **Linux**. |

NOTE

If you do not have permission to create environments, contact the administrator to grant you permissions on the permission management page of the application.

2. Click **Import Host**. In the displayed dialog box, select the configured host cluster and host and click **Import**.
3. When a message indicating that the import is successful is displayed, close the window.

**Step 4** On the **Deployment Actions** tab page, edit the actions of the application.

In action **Install Docker**, select **phoenix-env** from the **Environment** drop-down list. If a dialog box is displayed, asking you to confirm whether you want to change the environment to **phoenix-env** for the subsequent actions, click **OK**.

**Step 5** Click **Save & Deploy** to start the deployment task.

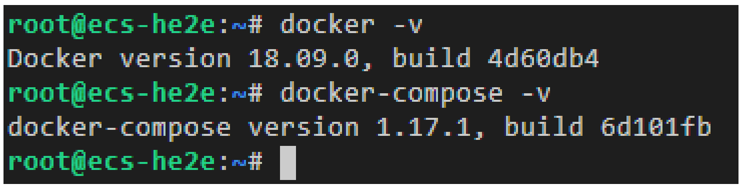
If a message is displayed indicating successful deployment, the task is successfully executed.

**Step 6** Log in to the ECS and check whether the dependency tools are successfully installed:

- Run the **docker -v** command to check the Docker image version.
- Run the **docker-compose -v** command to check the Docker-Compose version.

If the command output similar to **Figure 5-2** is displayed, the installation is successful.

**Figure 5-2** Checking the Docker and Docker-Compose versions



```
root@ecs-he2e:~# docker -v
Docker version 18.09.0, build 4d60db4
root@ecs-he2e:~# docker-compose -v
docker-compose version 1.17.1, build 6d101fb
root@ecs-he2e:~# 
```

**----End**

## Configuring and Executing an Application

During deployment, configure the ECS in the environment list of the application and set the build task **phoenix-sample-ci** as the deployment source.

**Step 1** Go to the **Phoenix Mall** project, choose **CICD** > **Deploy**, and find the **phoenix-sample-standalone** application in the list.

**Step 2** Click ··· and choose **Edit** from the drop-down list.

**Step 3** Click the **Environment Management** tab and configure the host environment.

1. Click **Create Environment**, configure the following information, and click **Save**.

**Table 5-11** Creating an environment

| Parameter | Value |
|---|---|
| Environment | Enter **phoenix-env**. |
| Resource Type | Select **Host**. |
| OS | Select **Linux**. |

2. Click **Import Host**. In the displayed dialog box, select the configured host cluster and host and click **Import**.
3. When a message indicating that the import is successful is displayed, close the window.

**Step 4** On the **Deployment Actions** tab page, edit the actions of the application.

1. Click **Select a Deployment Source**. Set the deployment source by referring to Table 5-12.

**Table 5-12** Configuring the deployment source

| Configuration Item | Suggestion |
|---|---|
| Source | Select **Build task**. |
| Build Task | Select **phoenix-sample-ci**. |
| Environment | Select **phoenix-env**.<br>If a dialog box is displayed, asking you to confirm whether you want to change the environment to **phoenix-env** for the subsequent actions, click **OK**. |

2. Retain the default settings in actions **Decompress Files** and **Run Shell Commands**.

**Step 5** Click the **Parameters** tab and set parameters based on the SWR login command.

Obtain the login command from the console.

**Step 6** Click **Save & Deploy** to start deployment.

If a message is displayed indicating that the deployment is successful, continue with the next step. If the deployment fails, rectify the fault based on the failed action and the error information in logs.

**Step 7** Verify the deployment result.

1. Open a browser, enter **http://*IP*:5000** in the address box, and press **Enter**. *IP* indicates the elastic IP address of the ECS. The Phoenix Mall homepage is displayed.
2. Enter **http://*IP*:5001** and press **Enter**. *IP* indicates the elastic IP address of the ECS. The Phoenix Mall dashboard is displayed.

**----End**

## Releasing Resources

To avoid unnecessary fees, **delete the ECS** after the practice.

If you need to create a pipeline after the deployment, you can delete the ECS after completing the pipeline practice.