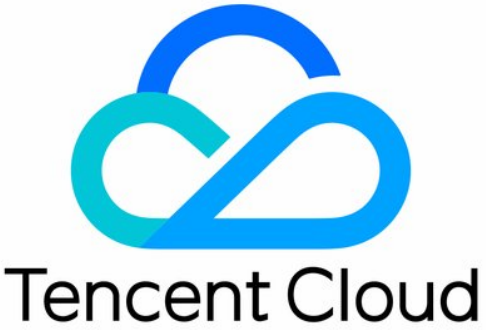
CODING DevOps

Continuous Deployment



Copyright Notice

Trademark Notice

Service Notice

This document provides an overview of the as-is details of Tencent Cloud's products and services in their entirety or part. The descriptions of certain products and services may be subject to adjustments from time to time.

The commercial contract concluded by you and Tencent Cloud will provide the specific types of Tencent Cloud products and services you purchase and the service standards. Unless otherwise agreed upon by both parties, Tencent Cloud does not make any explicit or implied commitments or warranties regarding the content of this document.

Contact Us

We are committed to providing personalized pre-sales consultation and technical after-sale support. Don't hesitate to contact us at 4009100100 or 95716 for any inquiries or concerns.

Contents

Continuous Deployment Quick Start

Last updated: 2024-09-05 16:43:19

Description of the Feature

Coding-CD is used to manage project release, deployment, and delivery processes after the build. It can seamlessly connect to upstream Git repositories and artifact repositories to achieve full automation deployment. Based on a stable technical architecture and operational tools, it supports blue-green deployment, grayscale release (canary release), rolling release, and rapid rollback.
The following will use a simple demo project as an example to demonstrate how to use the CODING-CD console to deploy applications to the Tencent Cloud cluster.

Preparation

- To enable Continuous Deployment and set permissions, refer to Permission details.
- Import Sample Code Library. Click to learn how to Import or Associate External Repository.
- Docker Artifact Repository, click to learn how to use Docker Artifact Repository in the project.
- A Kubernetes cluster accessible by CODING-CD, click to learn how to apply for Tencent Cloud Standard Cluster.

> **Note:**
> The recommended configuration for the cluster is 8 cores and 32 GB, with public network access enabled.

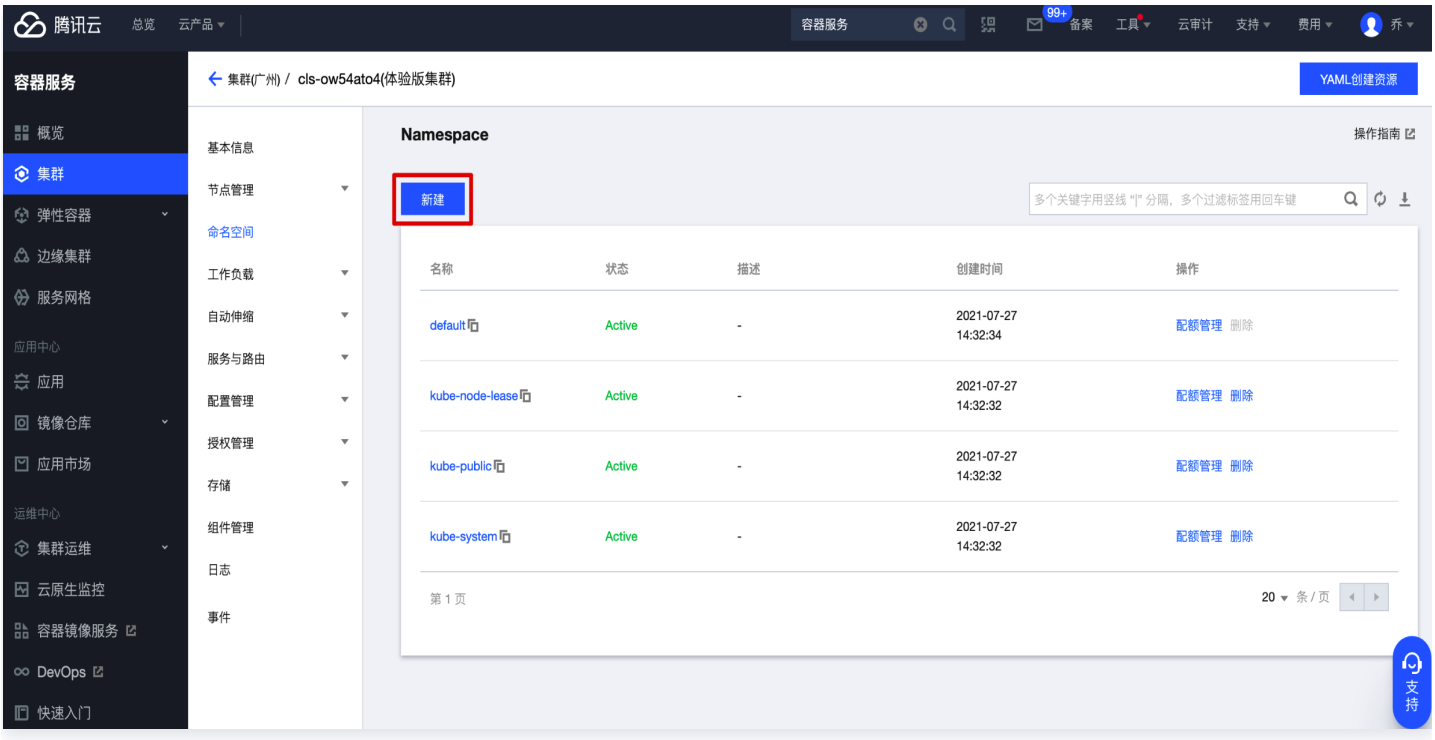
Operation step

Step 1: Obtain and associate a cloud account

> **Note:**
> Because Tencent Cloud TKE is used, the deployed application will be released to the cluster. The team account used in this example has been associated with **Team Settings Center > Third Party Application** in Tencent Cloud Account.

Click on **Infrastructure** on the left of the homepage, and bind the Tencent Cloud account in **Cloud Account**. You can customize your cloud account name. After selecting a region, you will automatically get the corresponding cluster.

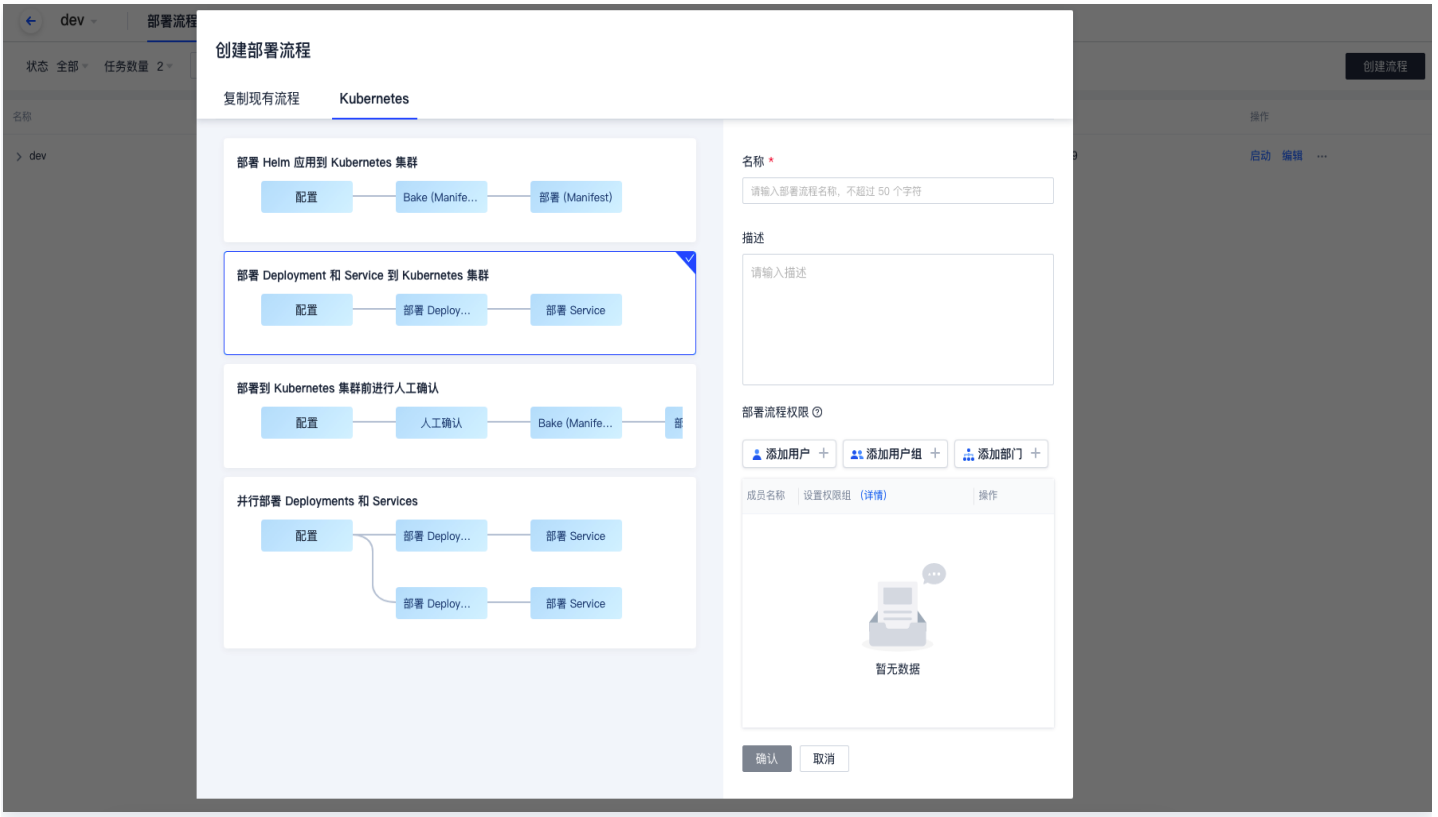
In the cluster, create a new `namespace` to store the automatically generated artifact repository access credentials. The cluster used in this document is named: cd-demo.



Step 2: Configure the application

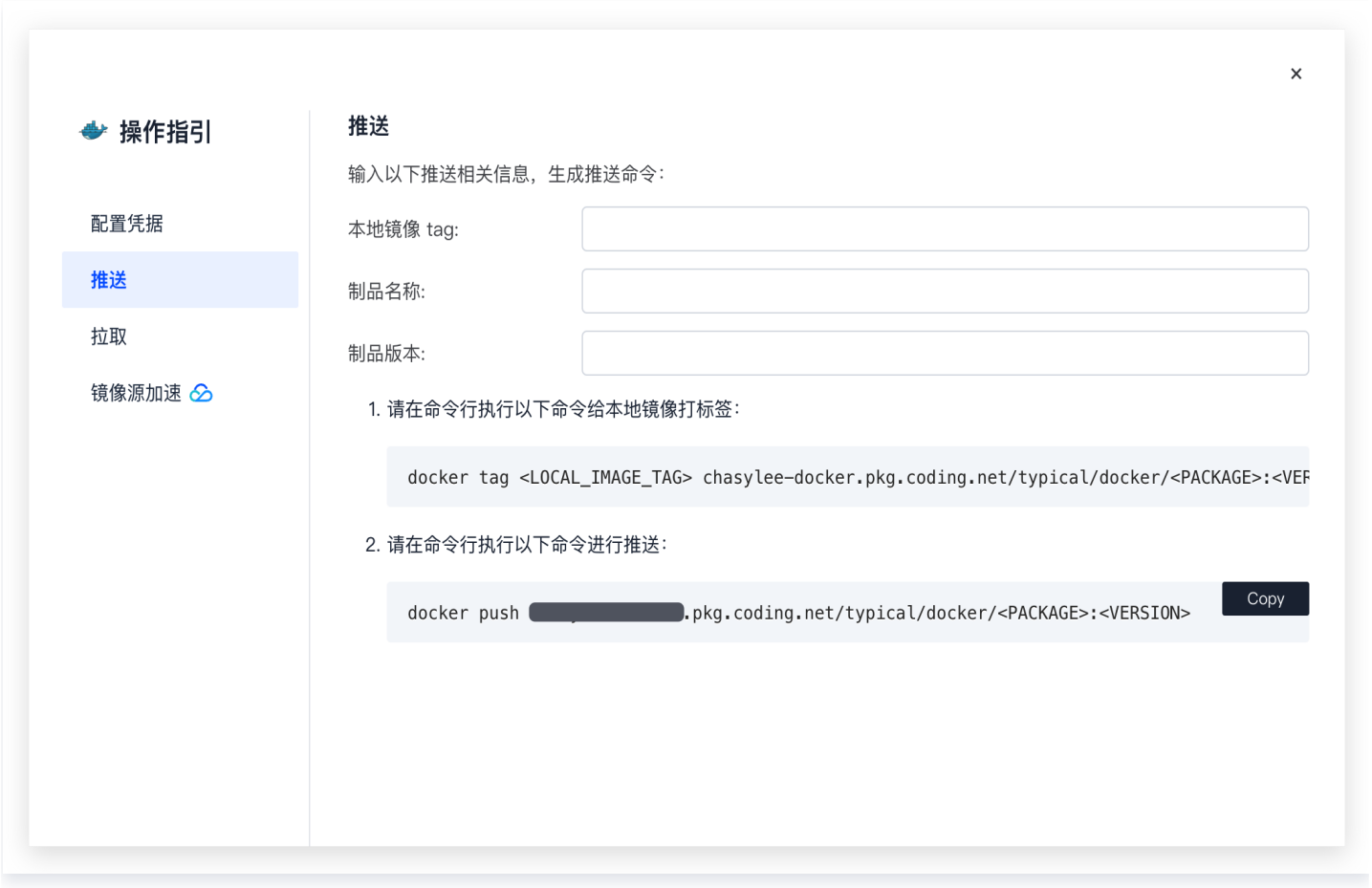
After successfully adding the cloud account, click **Create Application** in the Deployment Console, fill in the application name, and select the deployment method. For more details about **Application**, refer to Application Management.

Select **Deploy to Kubernetes Cluster** template, fill in the name and description, and complete the creation.

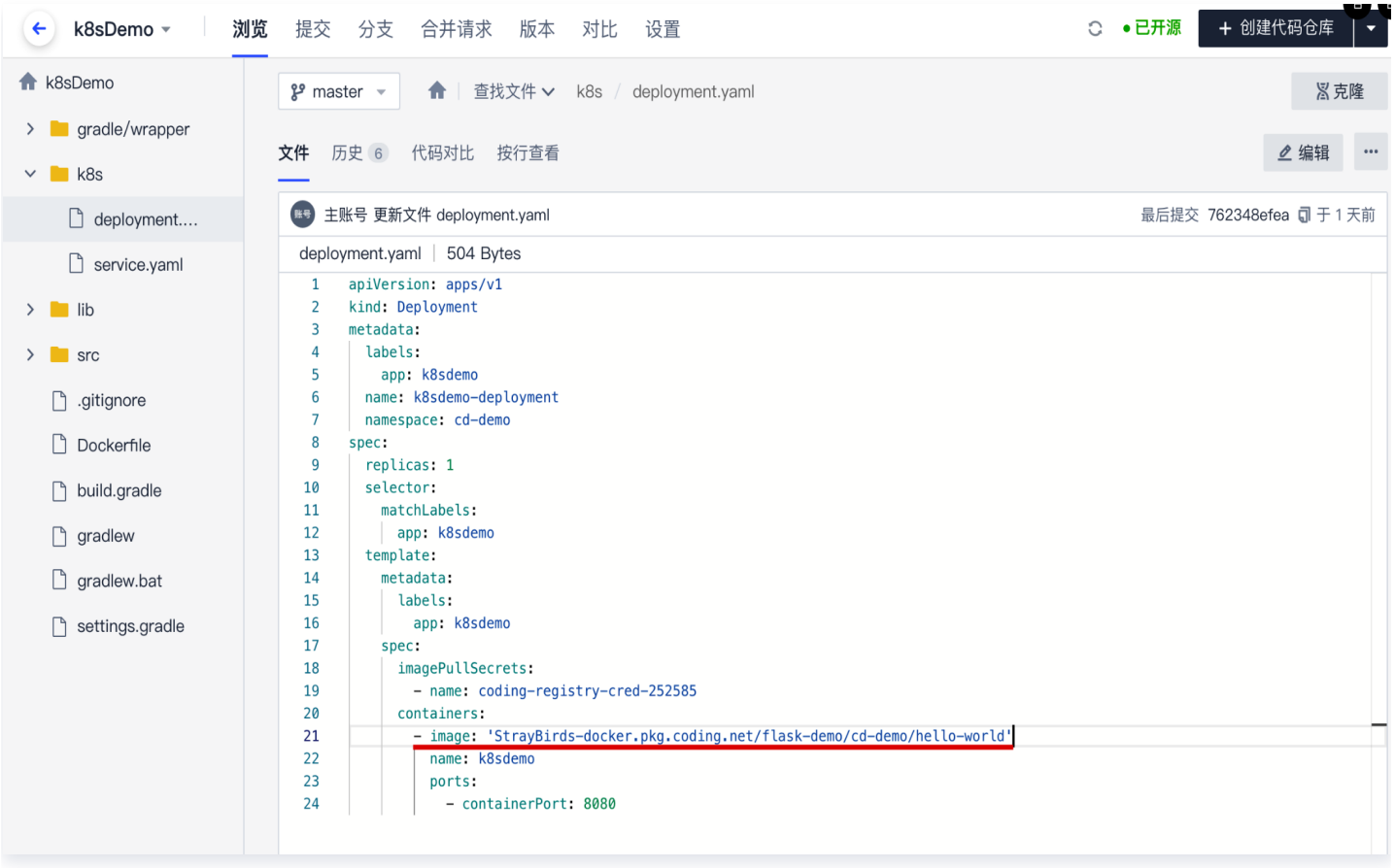


Step 3: Initialize the project

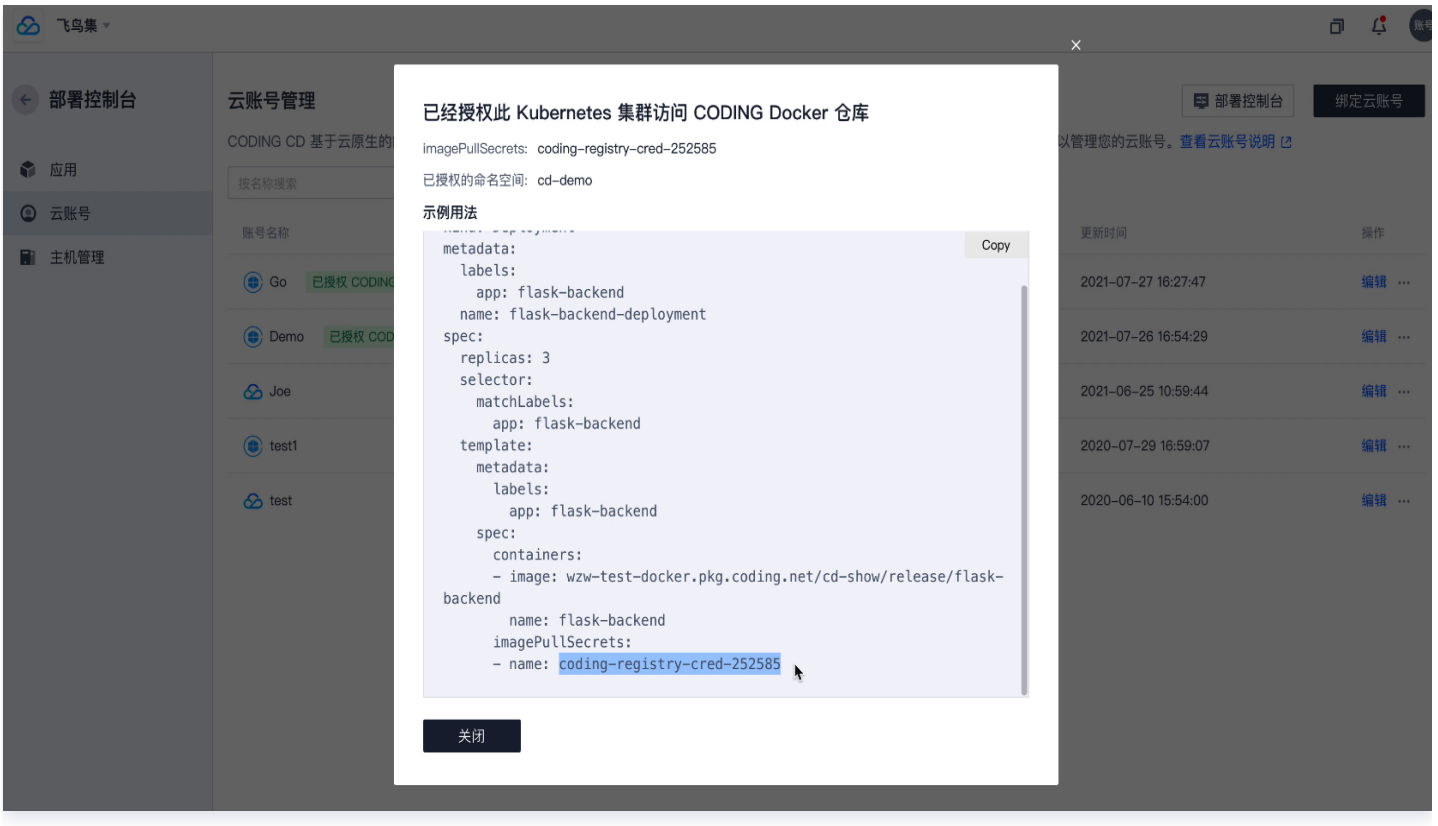
This step is mainly for configuring the project files involved in Continuous Deployment. It is assumed that you have already imported the sample code repository and created the Docker artifact repository in the prerequisite preparation.
First, push the local Docker artifact to the CODING artifact repository. For specific operations, refer to Docker.



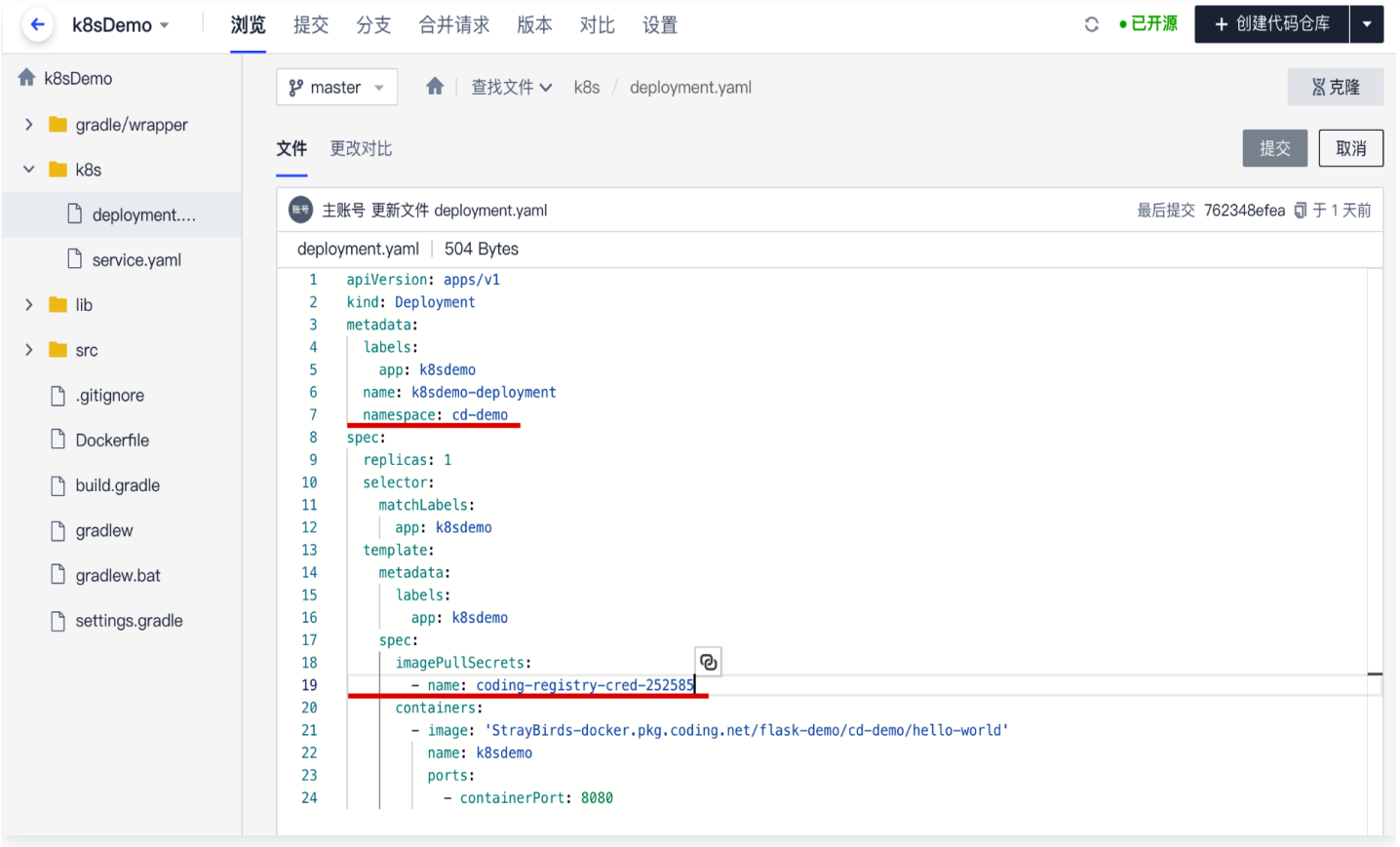
After pushing the artifacts to the artifact repository, get the artifact's pull address (i.e., the image path in the "Operation Guide" > "Pull" command) and enter it as the image address in the code repository's `/k8s/deployment.yaml` file.



Next, you need to import the cloud account's imagePullSecrets into the code repository. In **Infrastructure** > **Cloud Account**, click to view details and then copy the name.



Paste it into the `deployment.yaml` file in the code repository. Also, in the namespace parameter field, fill in the namespace 'cd-demo' created above.



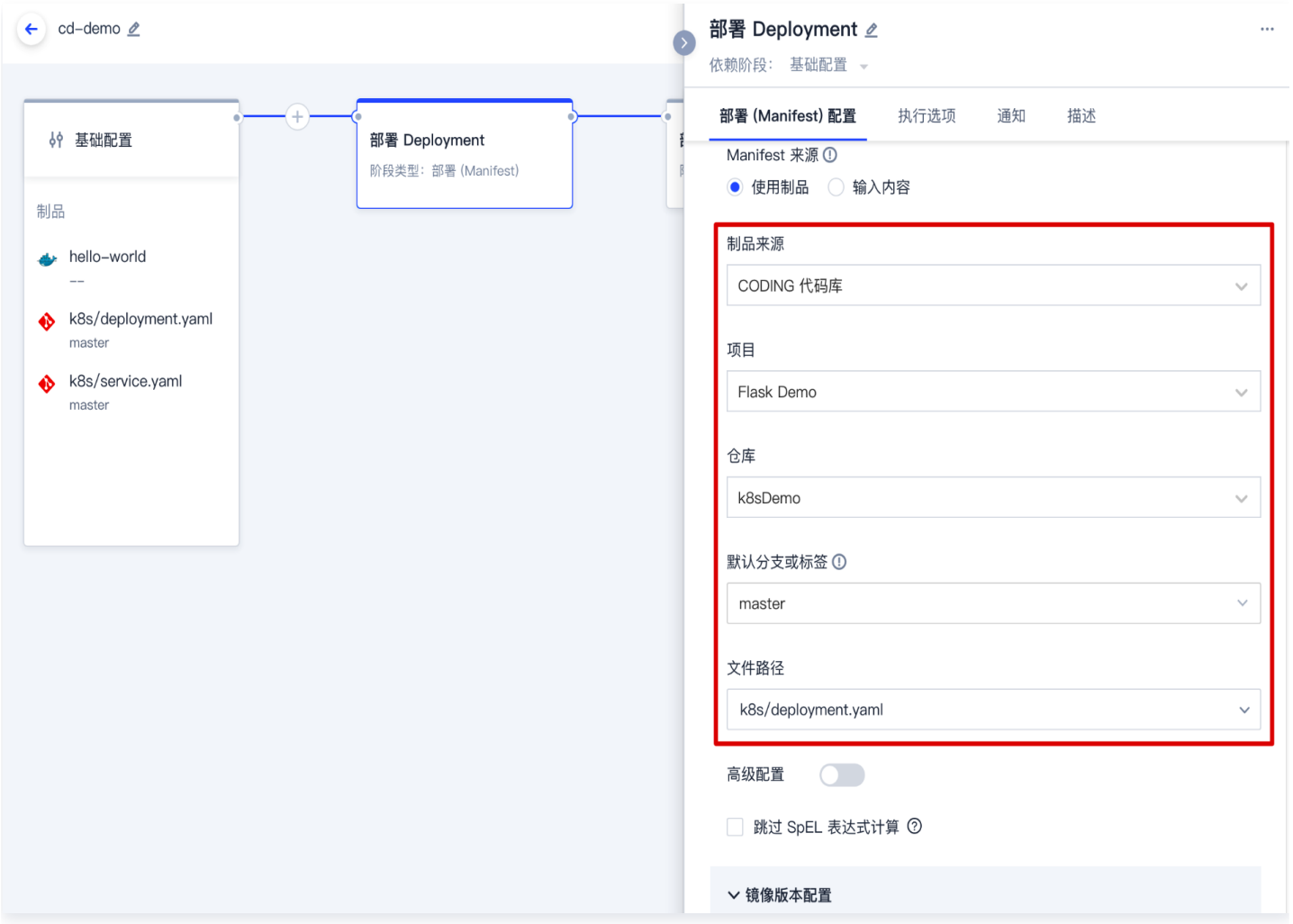
The namespace in the service.yaml file at the same level must also match.

Step 4: Deployment Pipeline

Find the application created in Step 2 in the Application Center, click its name to go to the Deployment Pipeline Configuration page, and set up the pipeline:

- Pipeline execution options (in this demo, all the default values are retained).
- Artifacts needed in the deployment and service deployment stages.
- Manual or automatic trigger.

First, configure the deployment (Manifest) stage. In Basic Settings, select the bound cloud account, choose Manifest Source as **CODING Code Repository**, and fill in the corresponding path.



Configure the service deployment stage by following the same steps as above. However, for the file path, select the `k8s/service.yaml` file.

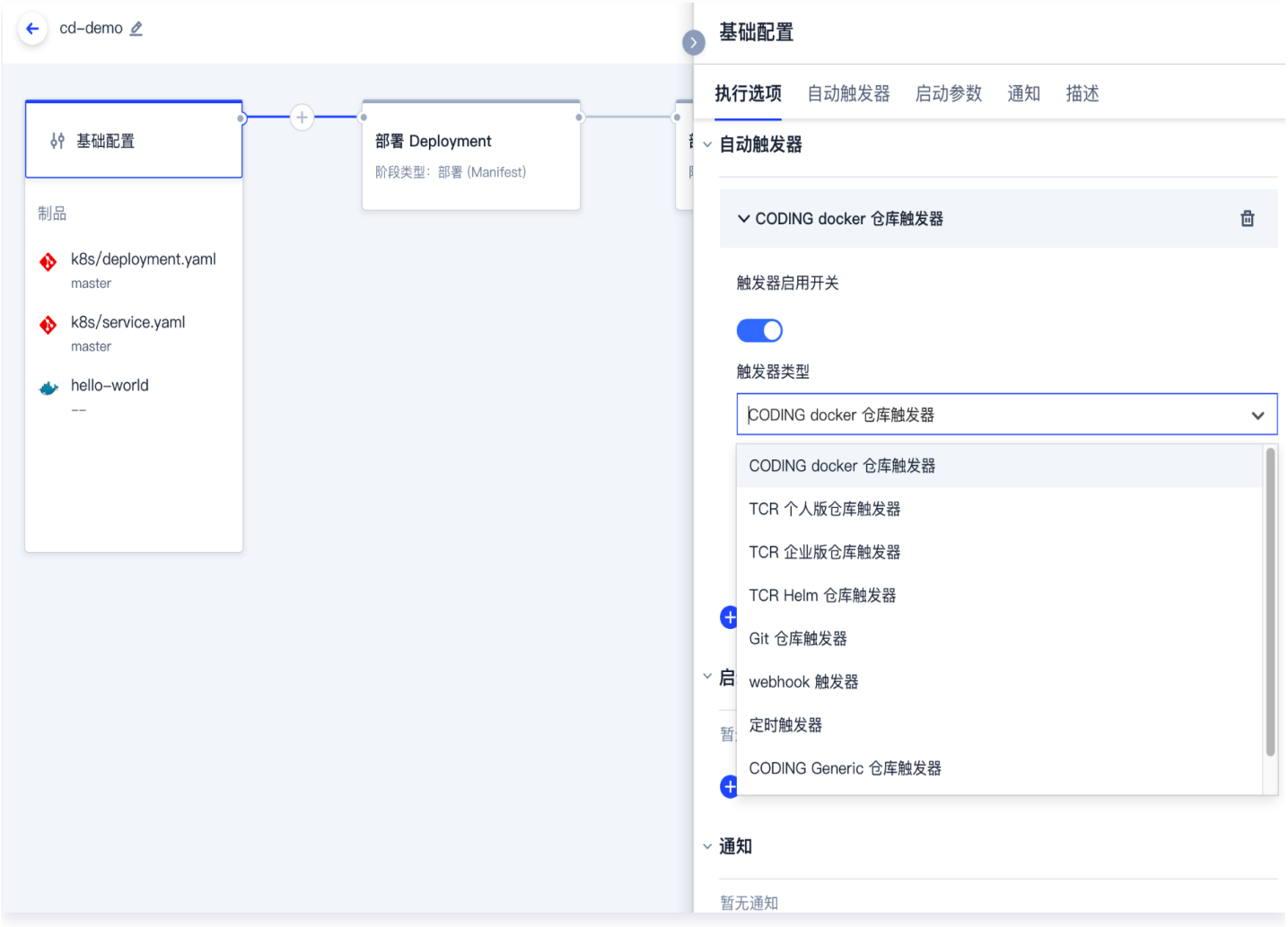
In Image Version Configuration, default to automatically fetching the image source. If you set Custom Version Rules, specific image version information will be sent to the cluster.

Step 5: Trigger Configuration

After configuring the deployment stages, you can use automated triggers or manually submit a release order to execute the deployment.

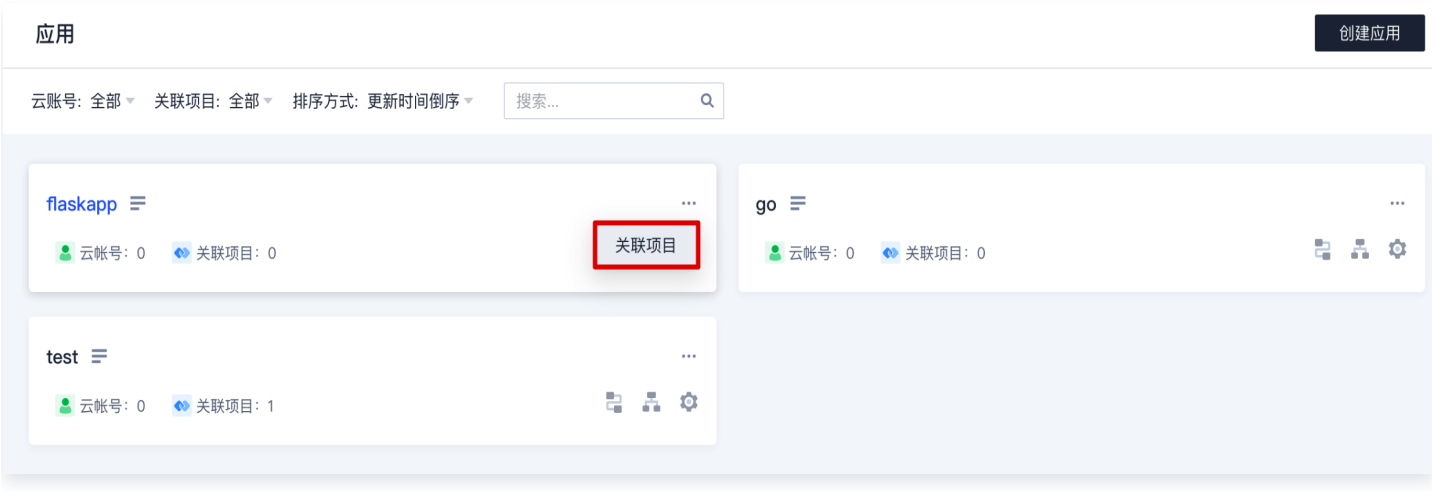
Automatic Trigger

In **Basic Configuration**, select the trigger type and choose Docker Repository Triggers. When developers update the code repository and use CI to package and push the image to the artifact repository, updating the Docker image version will automatically trigger the deployment pipeline, releasing the application to the Kubernetes (TKE) cluster. Upon completion, check and confirm the successful deployment in the Application Center.

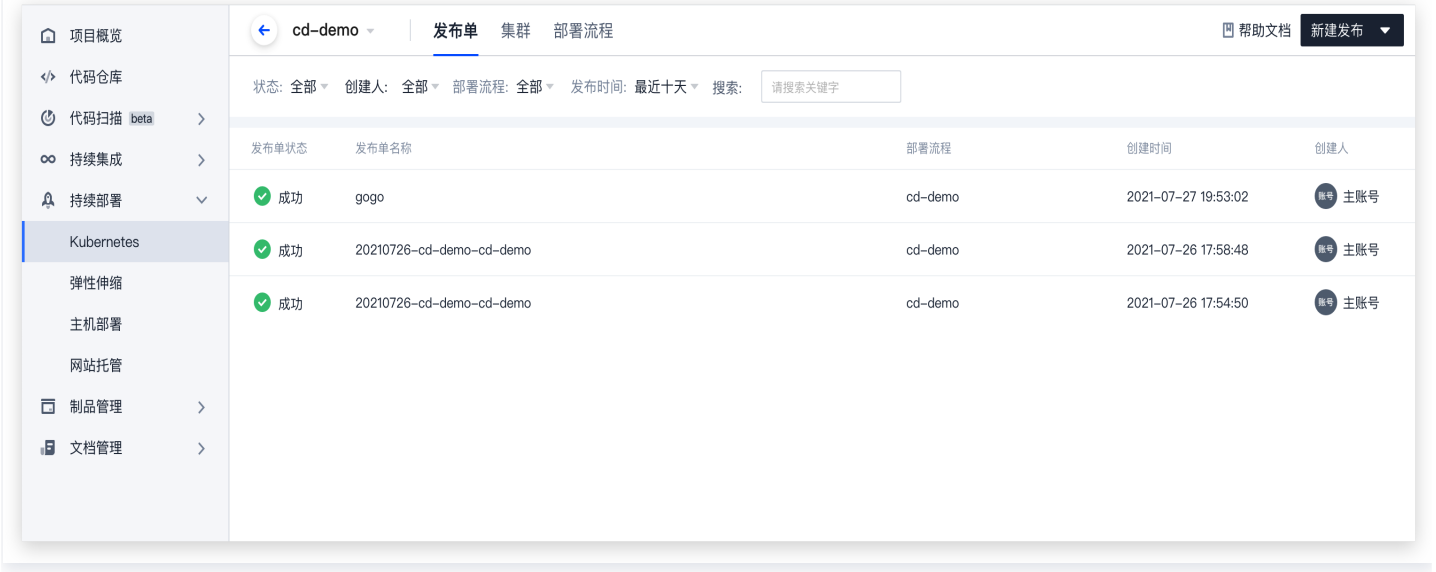


Manually Submit Release Order

To trigger the deployment process by manually submitting a release order, associate the application (for example, flaskapp in this case) with a project. Search for the project to be associated in the Application Center:

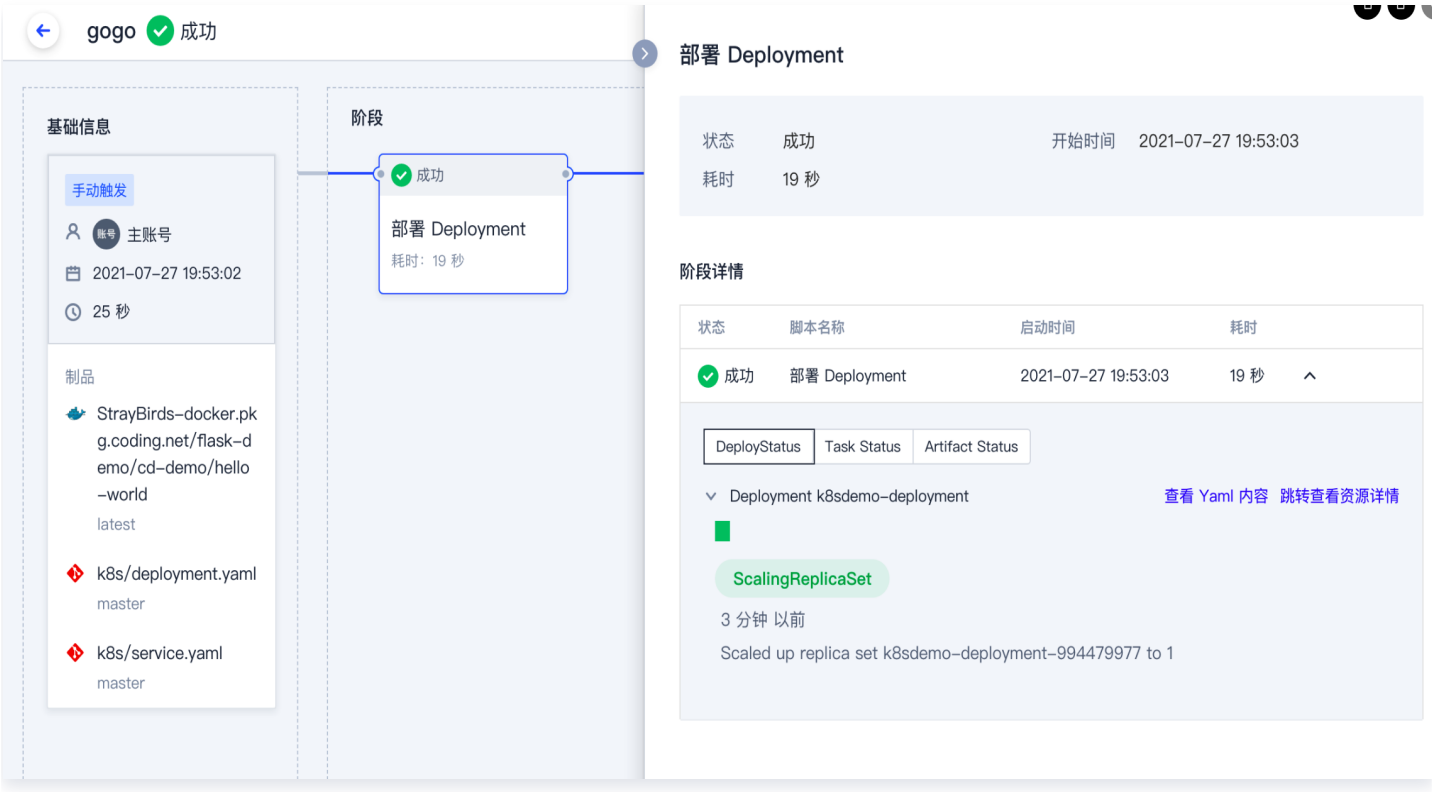


After the association is complete, click on **Continuous Deployment>Kubernetes** in the project to manually submit a release form.

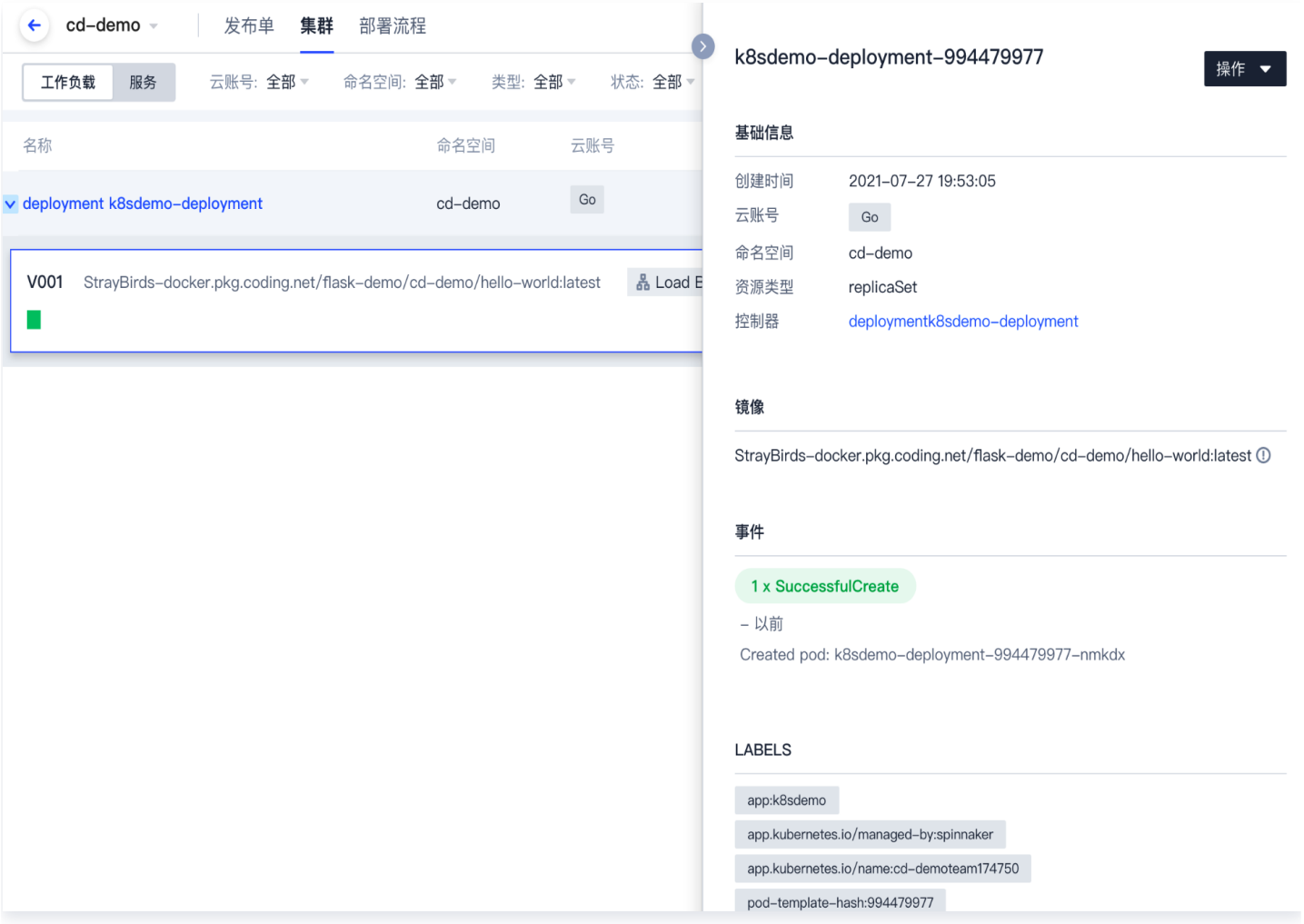


Step 6: Release Complete

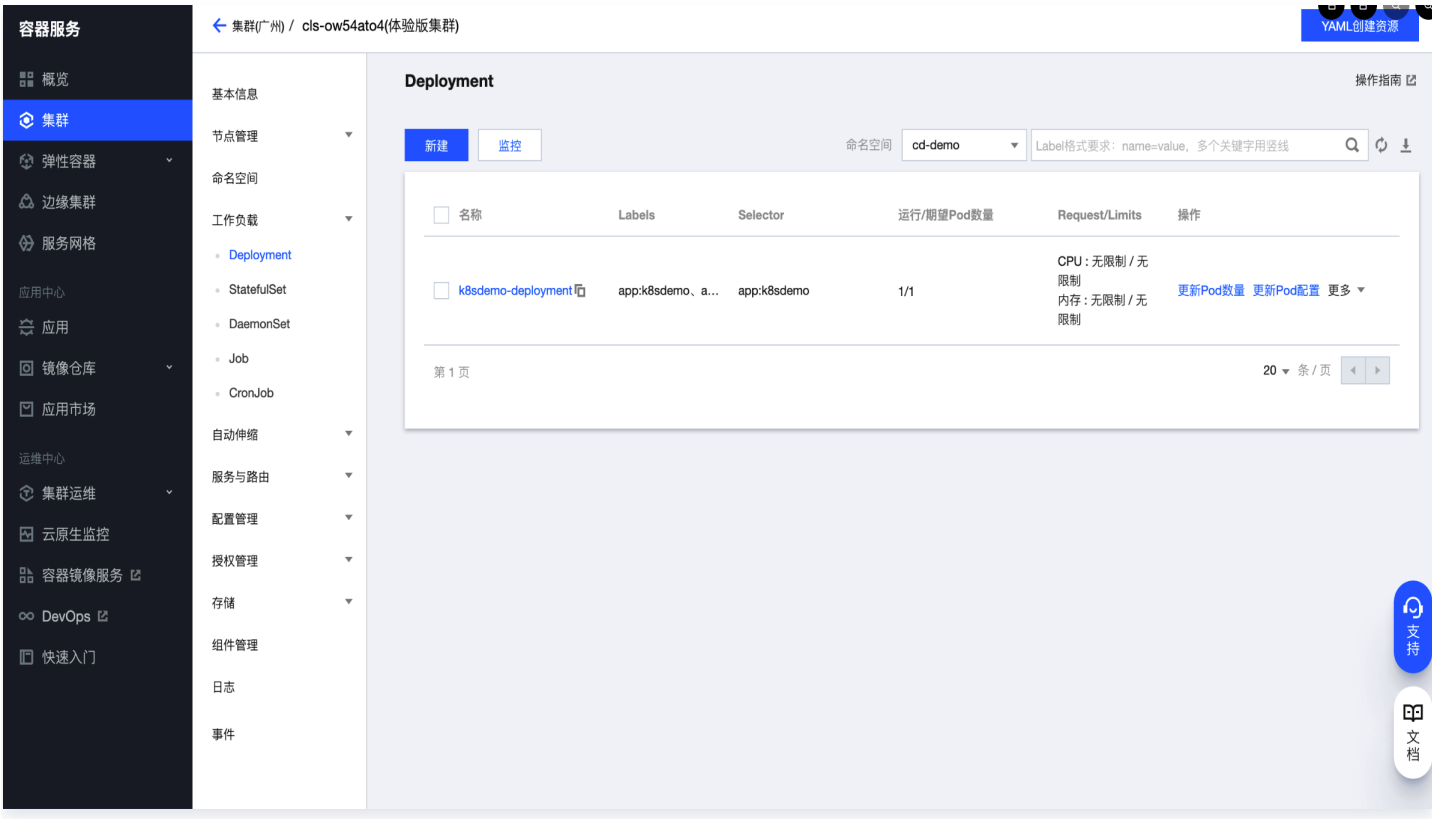
After a successful release, you can view the released artifacts, launch parameters, and stage execution details.



When you need to view the operational status of a resource in a cluster, click on the workload under **Cluster** to see the details (such as Pod instances of the workload, logs, and other information).



View workload in Tencent Kubernetes Engine.

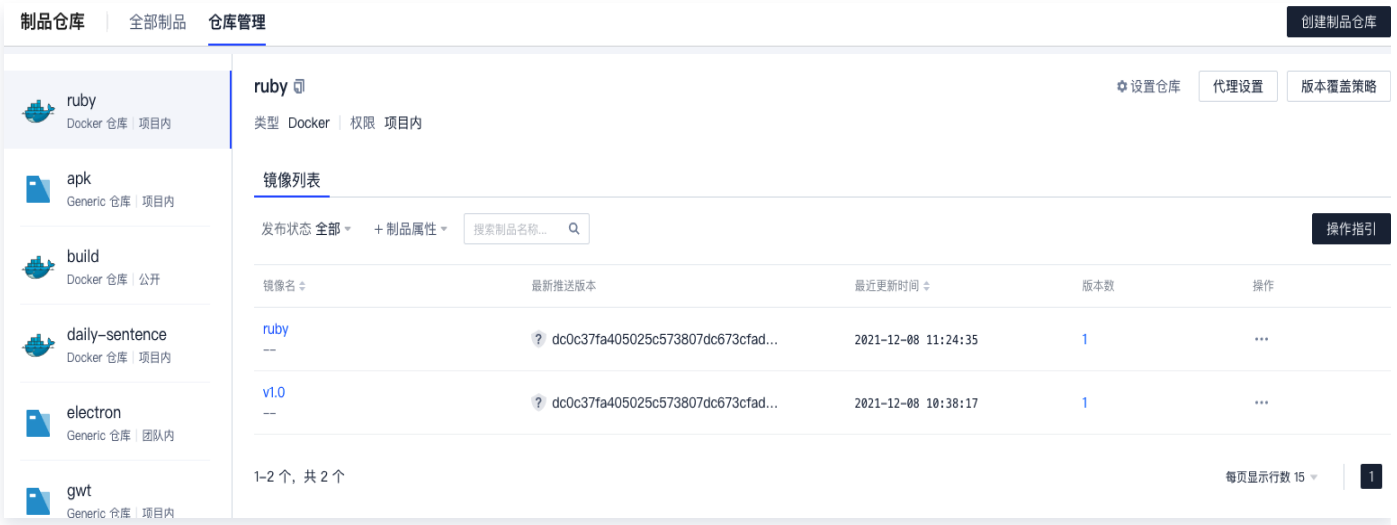


Configure the automated build process

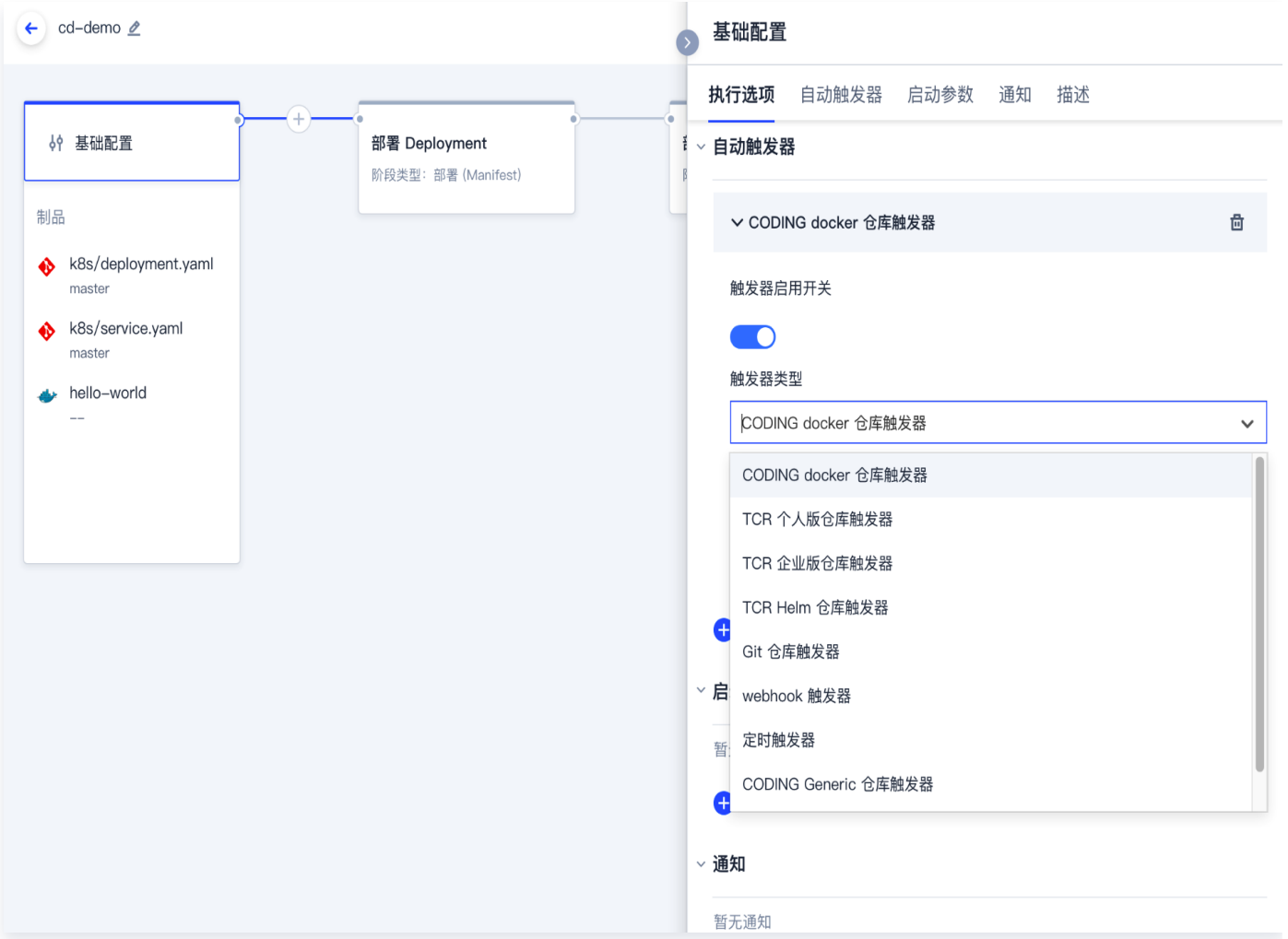
If frequent project development is needed later, can you rely solely on code pushes to automatically complete the release? The answer is yes. The classic design approach for continuous delivery workflow is as follows:

1. Configure continuous integration tasks to trigger upon detecting code updates.

2. Package the project code into an artifact and release it to the artifact repository.



3. Once continuous deployment detects a new artifact version, it automatically releases to the cluster.



Cloud Account

Last updated：2024-09-05 16:43:33

A cloud account is a credential for accessing cloud resources. Only after a cloud account is configured can the Continuous Deployment Console implement the management of cloud resources and infrastructure. Currently, the following three types of cloud accounts are supported:

- Tencent Cloud TKE: Use clusters in Tencent Cloud TKE.
- Kubernetes: Two commonly used credentials are supported, i.e. Kubeconfig and the Service Account.
- Tencent Cloud Account: Tencent Cloud API key.

Click on 'Infrastructure' on the left side of the team's homepage to enter the **Cloud Account** management page, then click 'Bind Cloud Account' at the top right corner.

Tencent Cloud TKE

1. Select Tencent Cloud TKE as the cloud account type and follow the instructions to complete the binding with the cluster under the cloud account. If you do not have a cluster, please go to Tencent Cloud TKE to create one.

2. Select the cluster you plan to deploy, and click **Confirm** to automatically verify the cluster under the account and complete the interconnection.

Kubernetes

A Kubernetes cloud account supports two commonly used credentials, i.e. Kubeconfig and the Service Account. Taking Kubeconfig as an example:
Log in to the Cloud Computing Web Console, copy Kubeconfig, and then add the CODING IP range to the extranet access control list (allowlist) of the cluster.

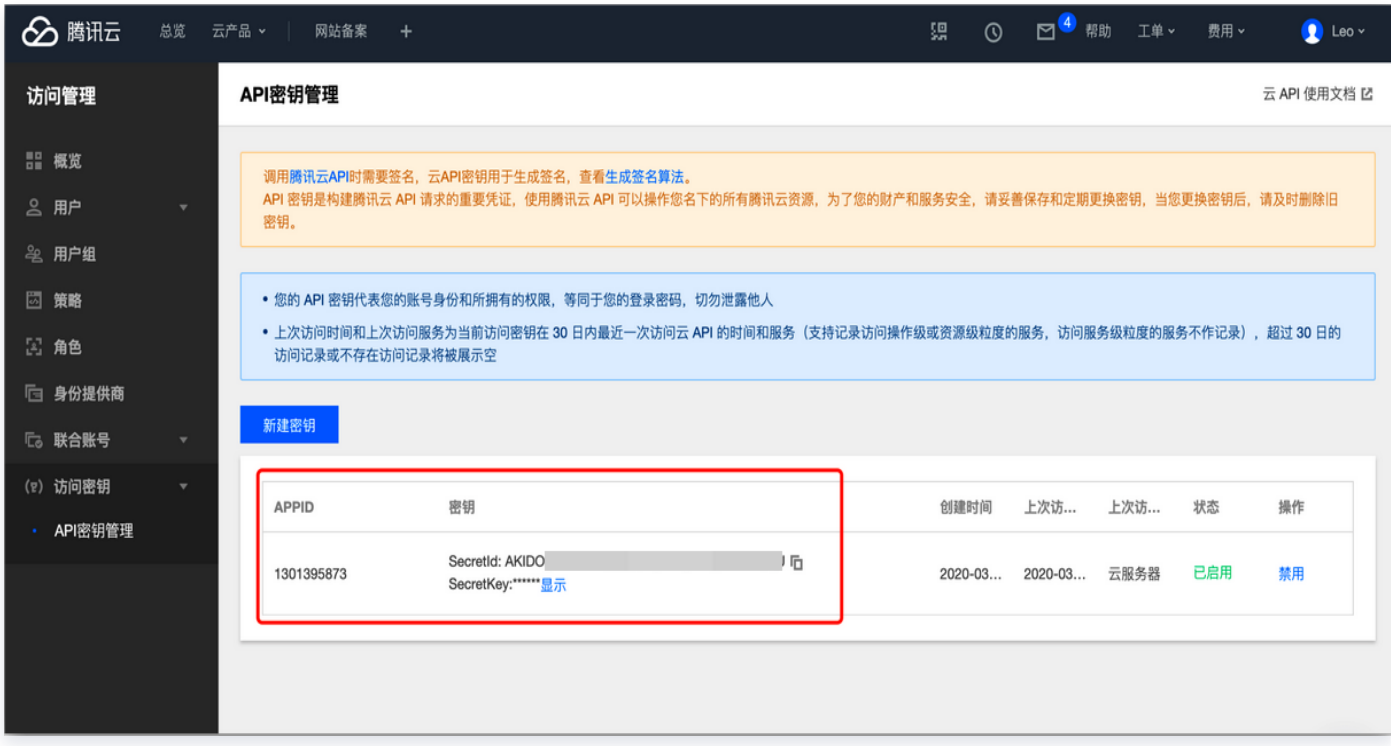
> **Note:**
> Public IP Range of CODING-CD:
> 212.64.105.0/24
> 212.129.144.0/24
> 49.234.127.0/24
> 49.235.224.0/24
> 49.234.65.0/24
> 81.69.101.0/24

Paste the Kubeconfig into the cloud account and select Cluster Context to add the cloud account.

Tencent Cloud Account

1. Select Tencent Cloud Account as the cloud account type, enter the cloud account name, and select regions. Multiple regions can be selected, allowing CODING-CD to obtain the resource management permissions of the selected regions.

2. Copy the API key information on the Tencent Cloud CAM Console page.



3. Paste the copied SecretID and SecretKey into the respective text boxes, and click **Confirm** to complete adding the cloud account.

Deployment Pipeline Management

Last updated: 2024-09-05 16:43:51

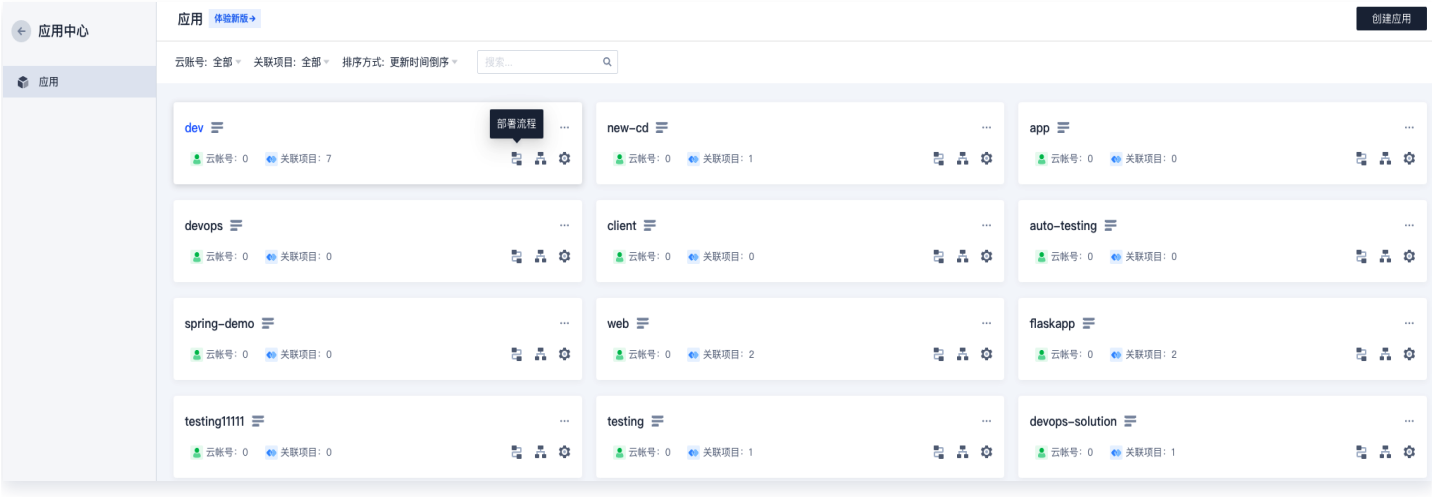
The deployment pipeline is the core module of Continuous Deployment. Its advantage lies in supporting the combination of stages in any sequence, which provides excellent flexibility, consistency, and repeatability.

- Flexibility: Supports serial and parallel control.
- Consistency: Supports multiple deployment strategies and rollback, ensuring the release results as expected.
- Repeatability: The deployment pipeline can be executed repeatedly, and stages can be copied and used by other deployment pipelines.

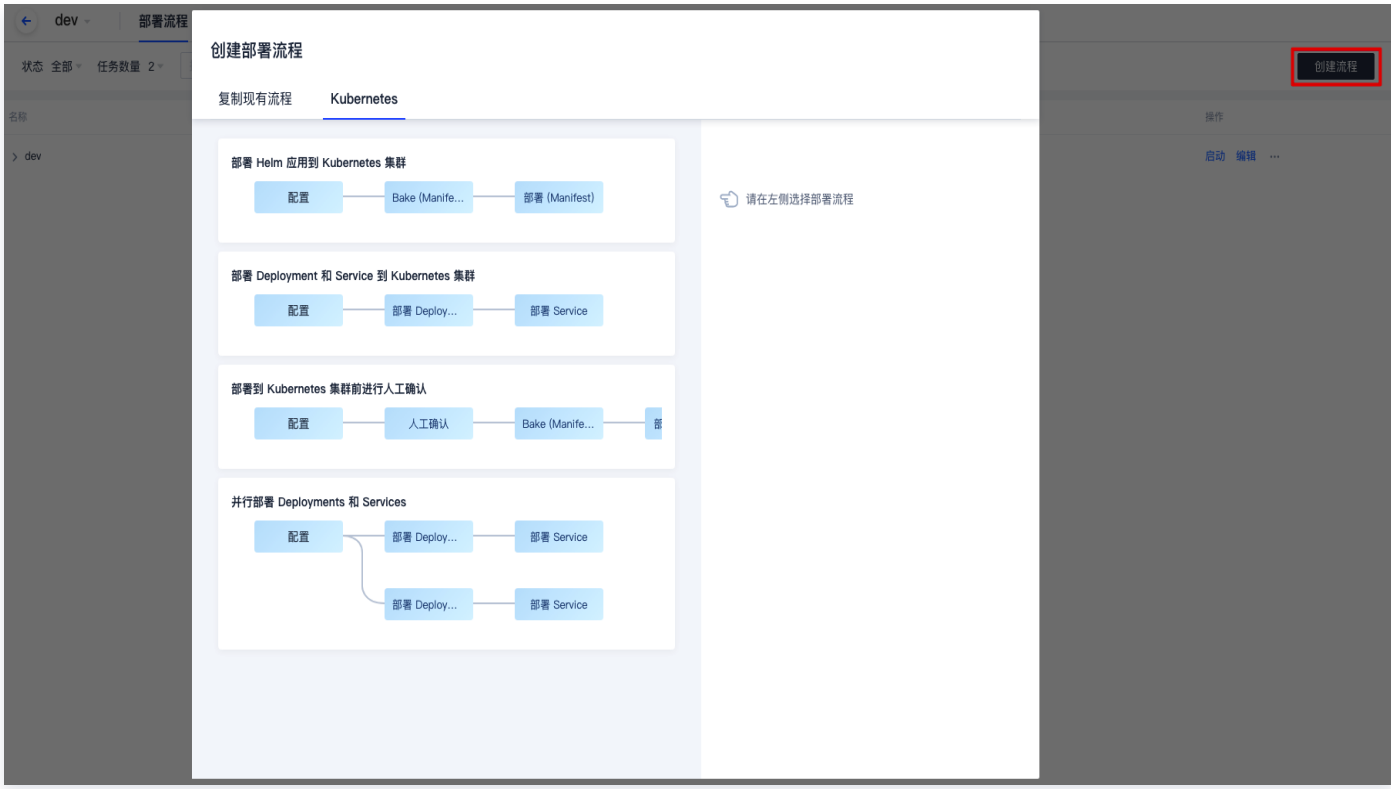
Users can configure a fully automated deployment pipeline or add manual judgment conditions at certain stages. Additionally, the deployment pipeline supports automation triggers by various events, such as webhooks and other deployment pipelines.

Create Deployment Pipeline

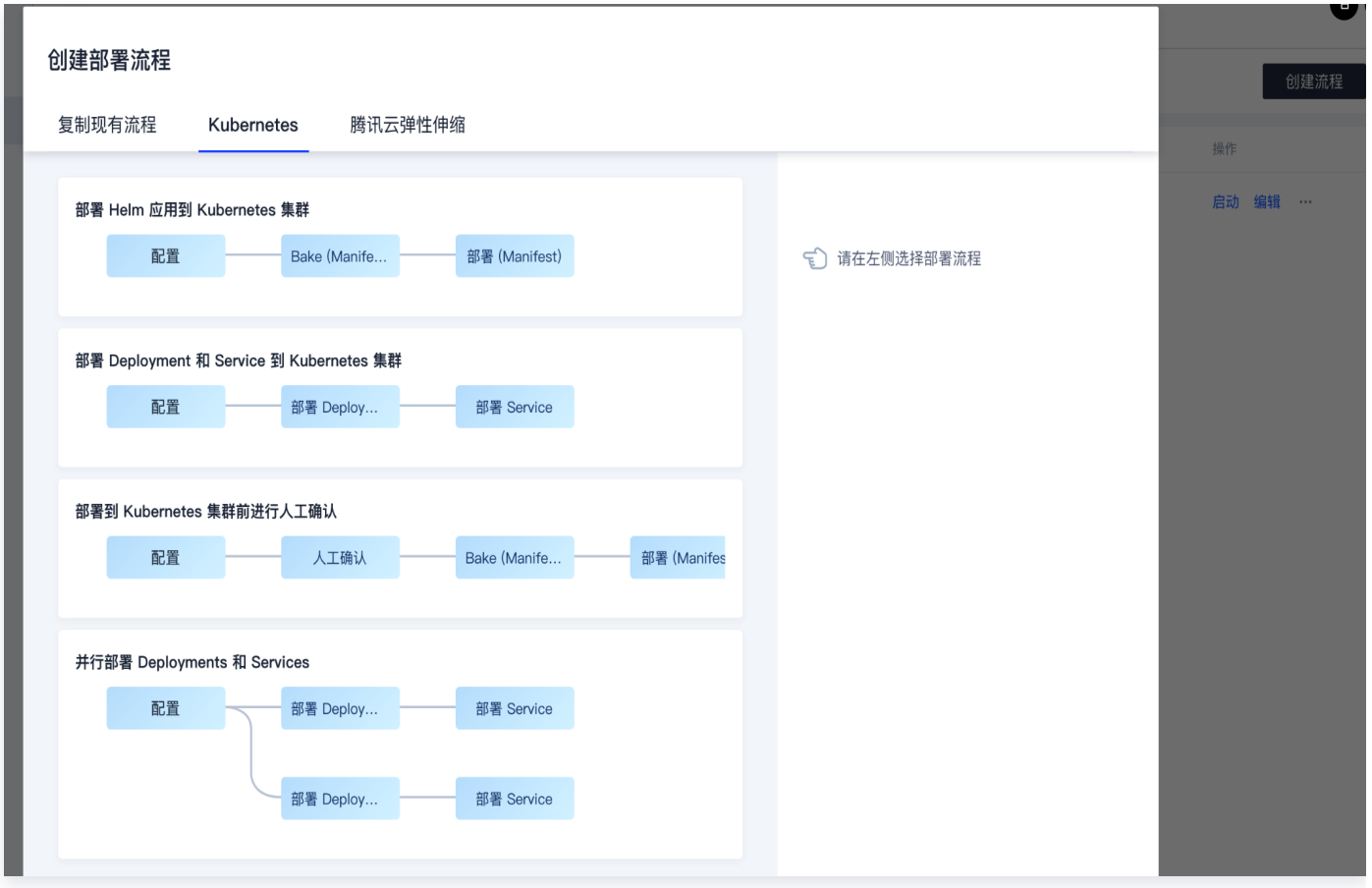
Go to the Application Center, and click the deployment pipeline icon in the lower-right corner of the application card.



1. Click **Create Process** in the top right corner.

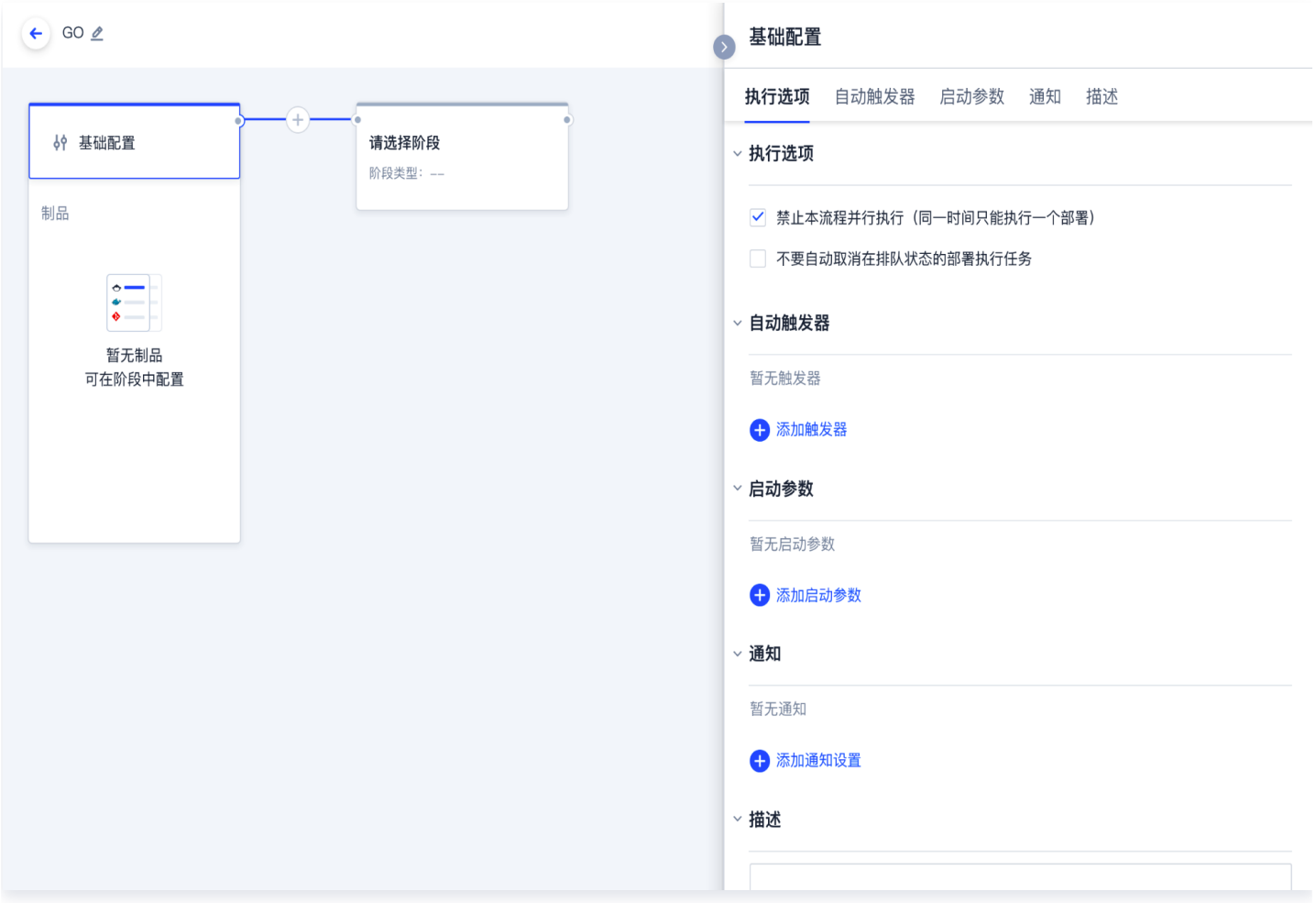


2. You can copy a pipeline created in another application or create a new one from scratch.



Basic configuration

Basic configurations of an application can be regarded as the starting point for a full build. This allows you to set trigger conditions, or configure the notifications for a pipeline.

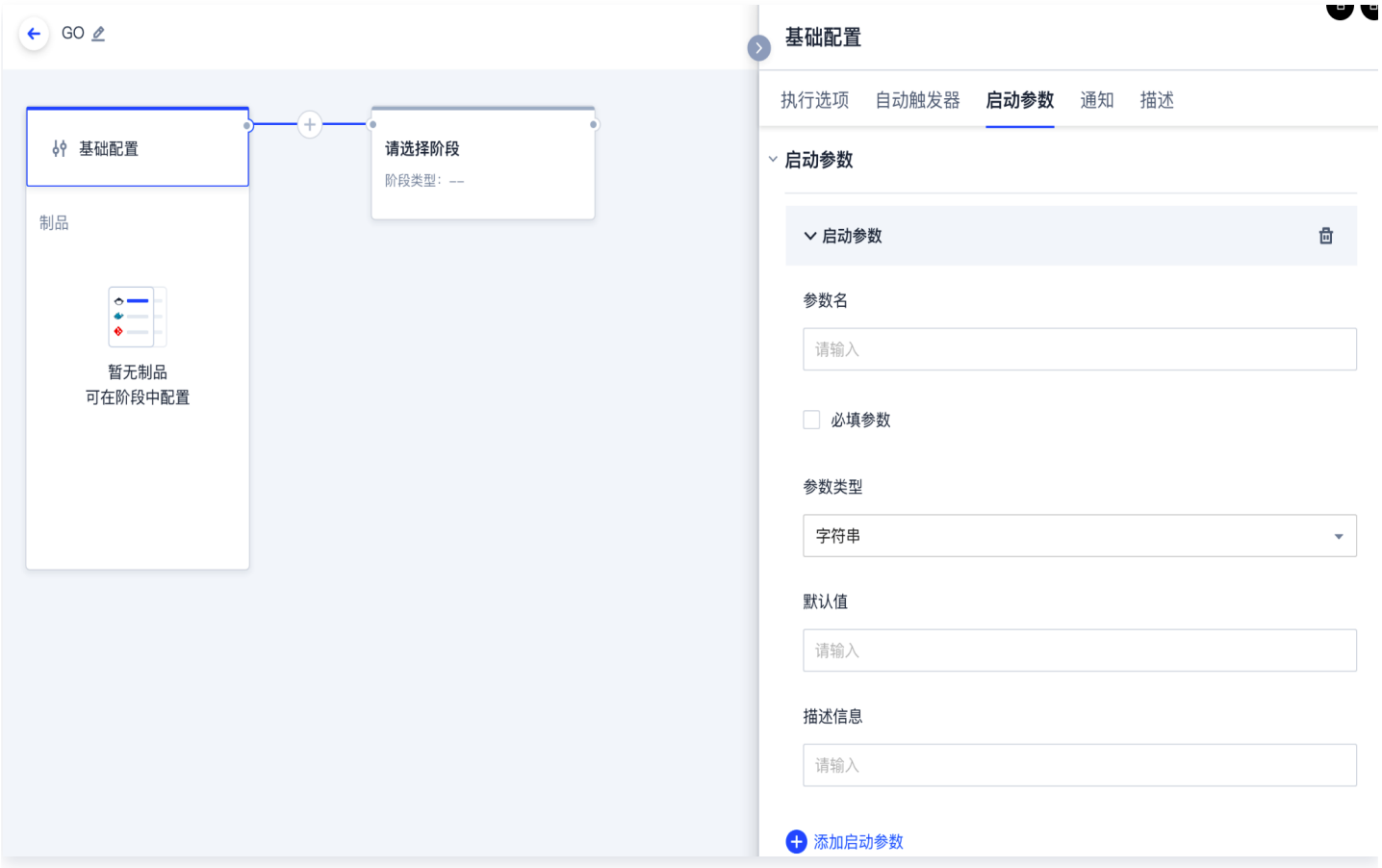


Auto Trigger

The auto trigger supports CODING Docker Repository Trigger, TCR Personal Repository Trigger, and Git Repository Trigger.

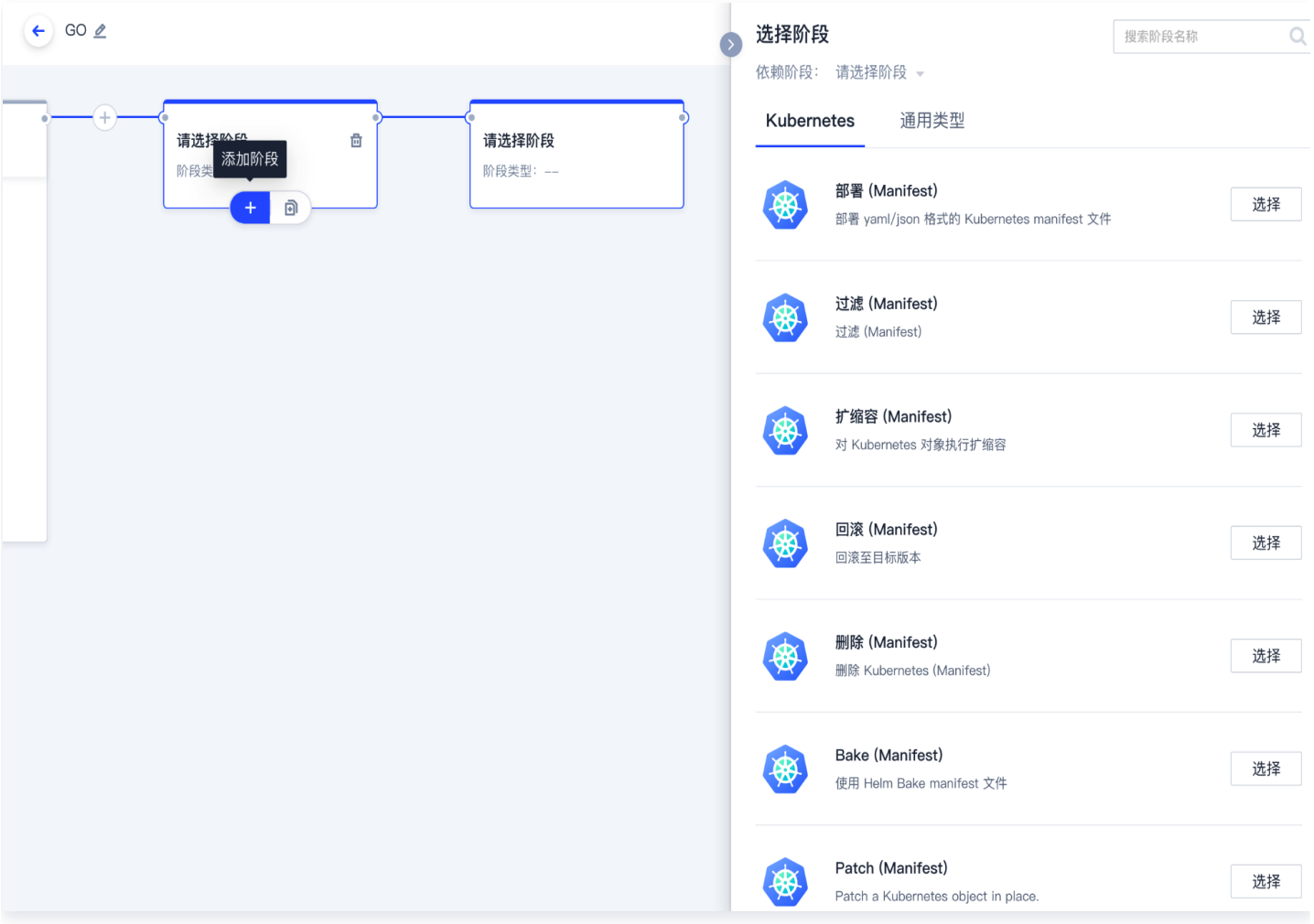
Add deployment pipeline parameters

On the process deployment configuration page, click **Add Parameter** to start entering parameters.



Add stage

On the Deployment Pipeline Configuration page, click the plus icon to add a new stage. You can select the stage type from the list on the right.

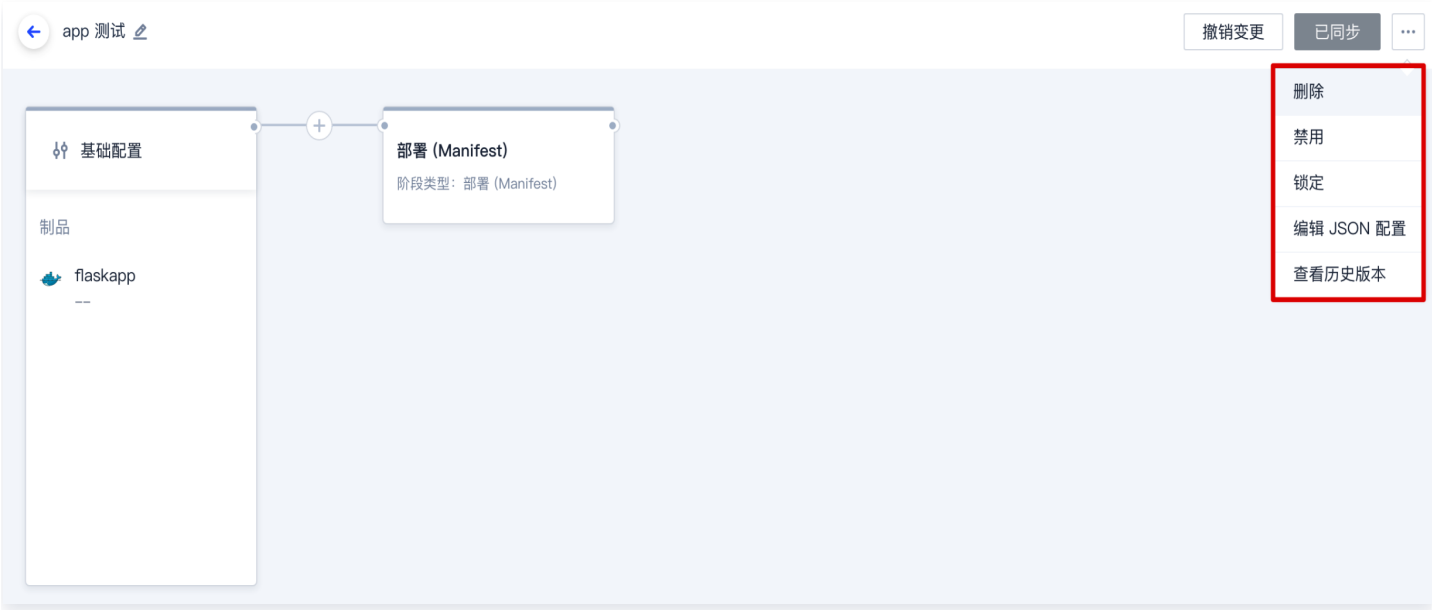


Execute Deployment Pipeline

After configuring the deployment pipeline, you can set it to execute automatically based on the configured triggers or trigger it manually by submitting a release sheet in Continuous Deployment.

Deployment Pipeline Configuration

You can delete, disable, or lock a deployment pipeline, view its earlier versions, and edit JSON configuration.



Delete Deployment Pipeline

After setting, this deployment pipeline will be deleted.

Disable deployment pipeline

Once set, all triggers, including manual triggers, will be disabled from starting the deployment process. You can choose to disable it for the entire team or only within a project.

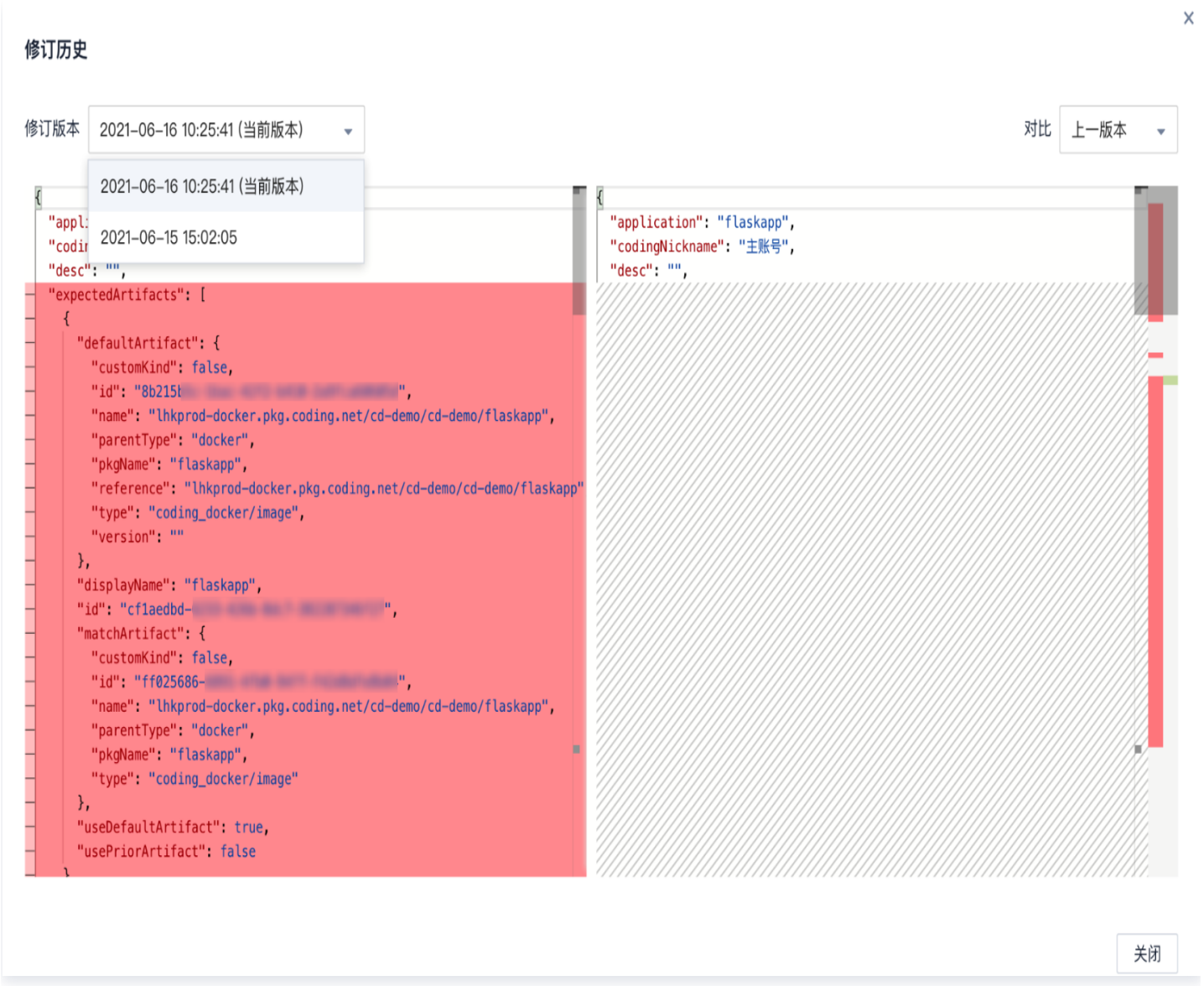
Lock deployment pipeline

Once the deployment process is locked, no modifications can be made to it in the Application Center.

View revision history

When a new pipeline configuration is saved, the previous version will be added to revision history. On the Revision History page, you can make a comparison between different versions,

and restore to any earlier version.

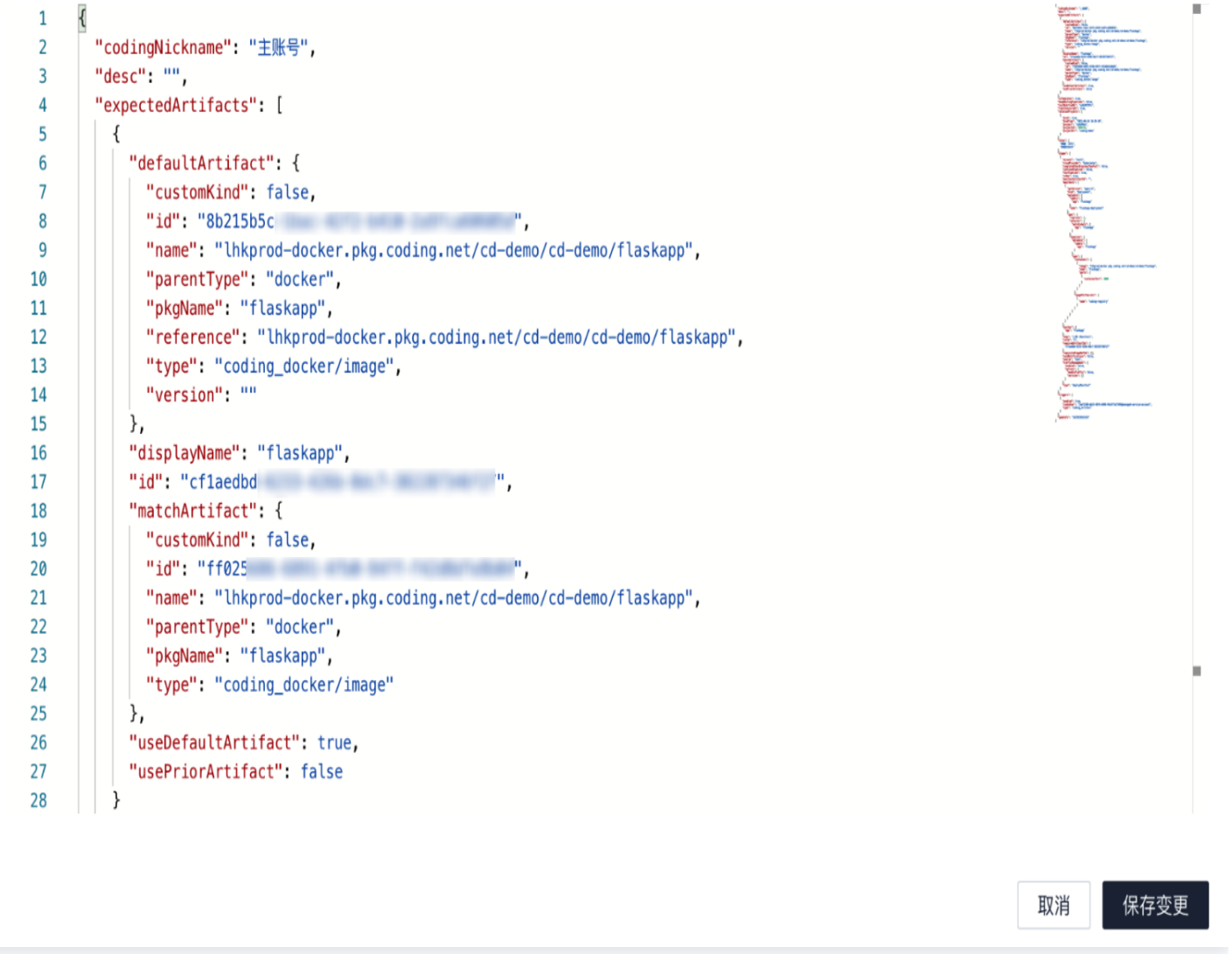


Edit JSON configuration

Any changes made in the CODING-CD Console are saved in JSON files. You can add new fields to a pipeline, or edit the JSON content to customize configuration items not displayed in UI.

> **Note:**
> This allows you to edit a deployment pipeline in the text box, but it may affect the availability of the pipeline. We support restoring to any specific version in the revision history.

编辑 JSON 配置



```
{
  "codingNickname": "主账号",
  "desc": "",
  "expectedArtifacts": [
    {
      "defaultArtifact": {
        "customKind": false,
        "id": "8b215b5c",
        "name": "lhkprod-docker.pkg.coding.net/cd-demo/cd-demo/flaskapp",
        "parentType": "docker",
        "pkgName": "flaskapp",
        "reference": "lhkprod-docker.pkg.coding.net/cd-demo/cd-demo/flaskapp",
        "type": "coding_docker/image",
        "version": ""
      },
      "displayName": "flaskapp",
      "id": "cf1aedbd",
      "matchArtifact": {
        "customKind": false,
        "id": "ff025",
        "name": "lhkprod-docker.pkg.coding.net/cd-demo/cd-demo/flaskapp",
        "parentType": "docker",
        "pkgName": "flaskapp",
        "type": "coding_docker/image"
      },
      "useDefaultArtifact": true,
      "usePriorArtifact": false
    }
```

取消 保存变更

Deployment mode Trigger When Docker Artifacts Are Auto Released

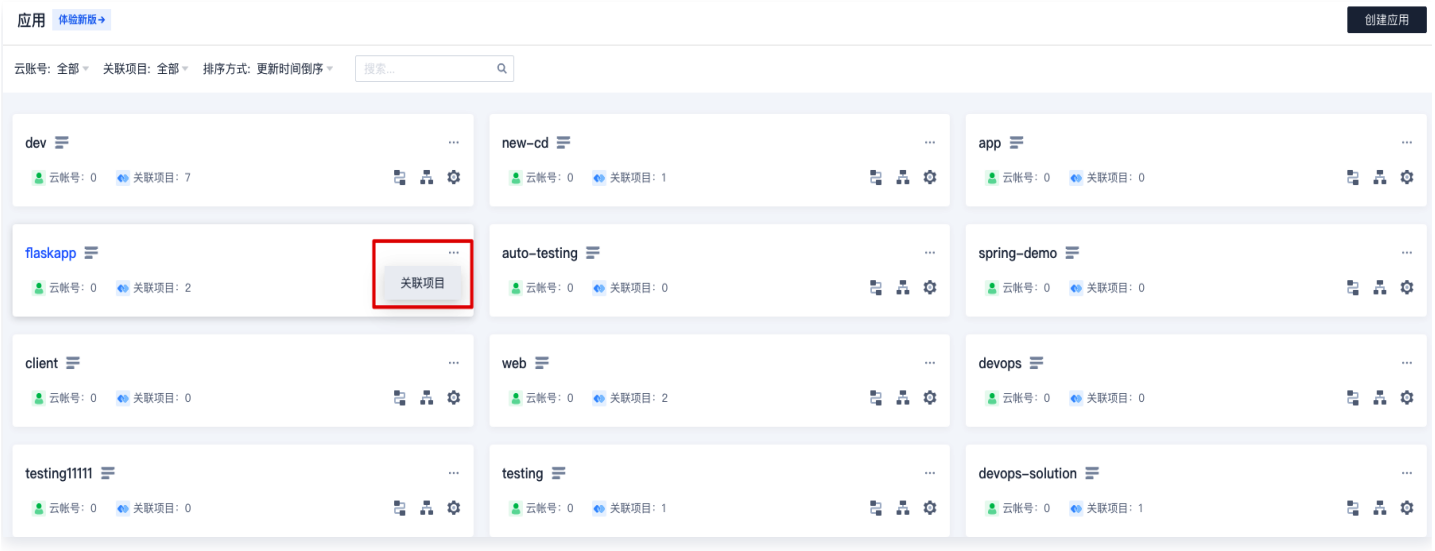
Last updated: 2024-09-05 16:44:07

One major advantage of CODING-CD is its ability to easily integrate upstream and downstream products into the workflow. The following will demonstrate how to implement **continuous integration task push artifacts** > **artifact repository image update** > **trigger deployment process**, a basic automated pipeline configuration.

Operation step

Step 1: Associate the application with the project

In the deployment process console, the **application** needs to be associated with the project in advance. Go to the App Center, click the **project association icon** in the application, select the project where the continuous integration configuration is located, and associate it.



Step 2: Configure CI process

This step pushes artifacts to the artifact repository through CI. You can create a CI process from the CI plan template, or add this stage by writing a Jenkinsfile.

Add this stage to the CI process:

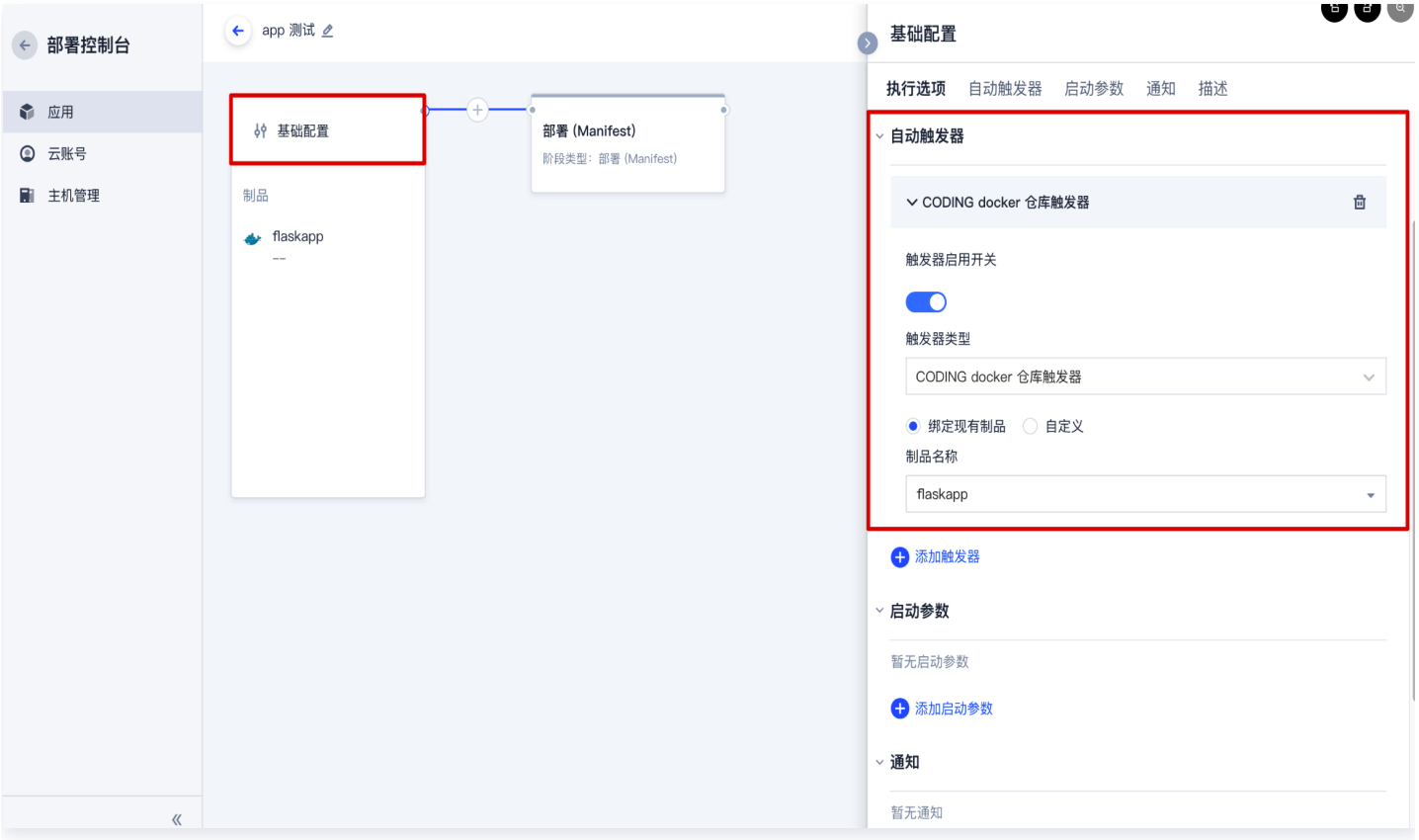
Jenkinsfile reference:

```
stage('Push to CODING Docker artifact repository') {
      steps {
        script {
          docker.withRegistry(

"${env.CCI_CURRENT_WEB_PROTOCOL}://${env.CODING_DOCKER_REG_HOST}",
            "${env.CODING_ARTIFACTS_CREDENTIALS_ID}"
          ) {

docker.image("${CODING_DOCKER_IMAGE_NAME}:${env.DOCKER_IMAGE_VERSION}").
```

```
push()
            }
          }


        }
```

Step 3: Trigger According to Artifact Image Version

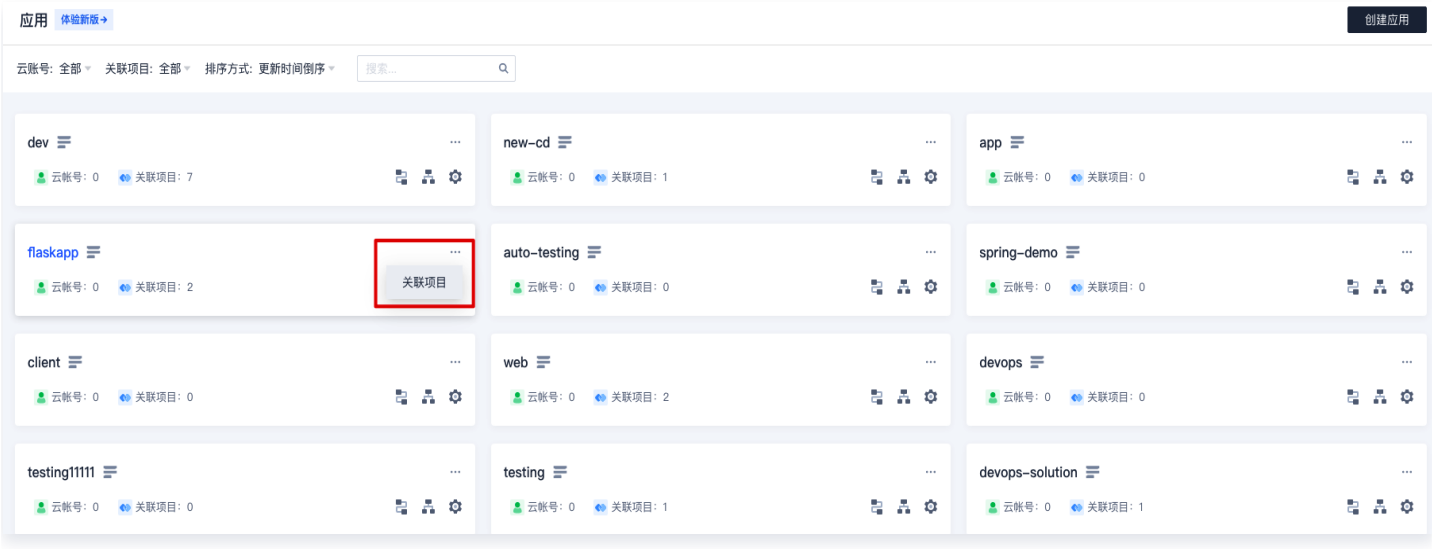
Go to the application deployment process in continuous deployment, click the trigger switch in **basic configuration**. Here, choose to trigger by CODING Docker artifact update, monitoring the artifact version number in the associated project. If continuous integration pushes the artifact to the artifact repository, it will automatically trigger the deployment process; choosing **by Definition** can monitor artifact repository updates from other projects.
In addition to CODING Docker Repository Trigger, you can also select Git Repository Trigger or Scheduled Trigger.



Add Deployment Stage to Build Plan

Last updated: 2024-09-05 16:44:21

Before you trigger deployment in Continuous Integration, go to the **Application Center** to associate your application with the project.



This document provides two methods of configuration. You can selectively read based on your needs.

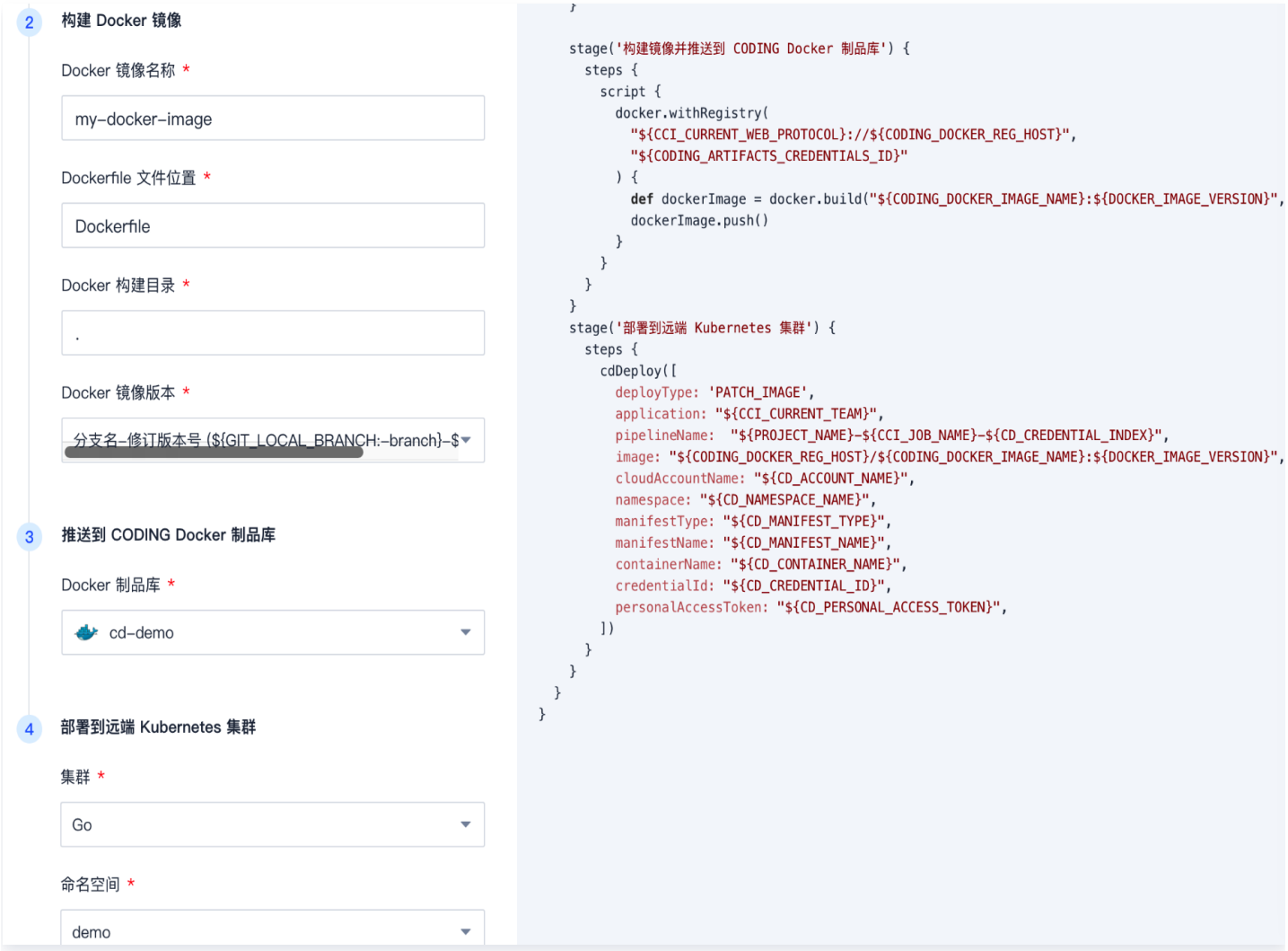
- Directly use a build plan template
- Add a deployment stage to an existing build plan

Use Build Plan Template

Associate the cloud account for the relevant cluster in the Application Center. For more information, see Cloud Account.

Click on the **Continuous Integration** in the left product column, then click on the upper right corner to create a build plan. Select the template under **Deployment** category, which is **Push to Kubernetes**.

Follow the template prompts to select the appropriate artifact repository, remote cluster address, and other information. After completion, check the option to trigger build after creation.



After the setup is complete, you can run the continuous build plan to automate the release process.

Add Deployment Stage

In this method, you can use the editor or fill in commands to add deployment stages in **Build Plan > Flow Configuration**.

Graphical editor

Add a **Deployment** stage in the existing build plan settings, and then fill in the image URL, cluster, namespace, etc.

Jenkinsfile

```
stage('Deploys to the remote Kubernetes cluster') {
      steps {
        cdDeploy([
          deployType: 'PATCH_IMAGE',
          application: "${CCI_CURRENT_TEAM}",
```

```
          pipelineName:
"${PROJECT_NAME}-${CCI_JOB_NAME}-${CD_CREDENTIAL_INDEX}",
          image:
"${CODING_DOCKER_REG_HOST}/${CODING_DOCKER_IMAGE_NAME}:${DOCKER_IMAGE_VE
RSION}",
          cloudAccountName: "${CD_ACCOUNT_NAME}",
          namespace: "${CD_NAMESPACE_NAME}",
          manifestType: "${CD_MANIFEST_TYPE}",
          manifestName: "${CD_MANIFEST_NAME}",
          containerName: "${CD_CONTAINER_NAME}",
          credentialId: "${CD_CREDENTIAL_ID}",
          personalAccessToken: "${CD_PERSONAL_ACCESS_TOKEN}",
        ])
      }
    }
```

Manually Submit Release Order

Last updated：2024-09-05 16:44:38

Create Release Order

Add a Manual Confirmation step in the application's deployment procedure to ensure that releases are double-checked via the Release Sheet and to maintain release quality through permission control.

1. To use the Release feature, you must link the app with the project.
2. After linking, go to the project's **Continuous Deployment**, find the app, and click **Release** to enter the release management page.

3. After clicking **Create Release**, you can run existing apps and deployment processes.

Quick Release

If you do not wish to set complex permission restrictions in team settings and want to directly experience the Continuous Deployment feature, you can use the **Quick Release** feature. You

do not need to configure the deployment process in the console to deploy images to the cluster, suitable for scenarios where deployment processes are more flexible and complex, such as temporary image changes and other emergency situations, allowing for quick artifact deployment to the cluster.

After release, you can view the release details in Continuous Deployment.