

CODING DevOps

Continuous Deployment



Copyright Notice

©2013–2024 Tencent Cloud. All rights reserved.

The complete copyright of this document, including all text, data, images, and other content, is solely and exclusively owned by Tencent Cloud Computing (Beijing) Co., Ltd. ("Tencent Cloud"); Without prior explicit written permission from Tencent Cloud, no entity shall reproduce, modify, use, plagiarize, or disseminate the entire or partial content of this document in any form. Such actions constitute an infringement of Tencent Cloud's copyright, and Tencent Cloud will take legal measures to pursue liability under the applicable laws.

Trademark Notice



This trademark and its related service trademarks are owned by Tencent Cloud Computing (Beijing) Co., Ltd. and its affiliated companies ("Tencent Cloud"). The trademarks of third parties mentioned in this document are the property of their respective owners under the applicable laws. Without the written permission of Tencent Cloud and the relevant trademark rights owners, no entity shall use, reproduce, modify, disseminate, or copy the trademarks as mentioned above in any way. Any such actions will constitute an infringement of Tencent Cloud's and the relevant owners' trademark rights, and Tencent Cloud will take legal measures to pursue liability under the applicable laws.

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

- Cloud Account

- Deployment Pipeline Management

- Deployment mode

 - Trigger When Docker Artifacts Are Auto Released

 - Add Deployment Stage to Build Plan

 - Manually Submit Release Order

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.

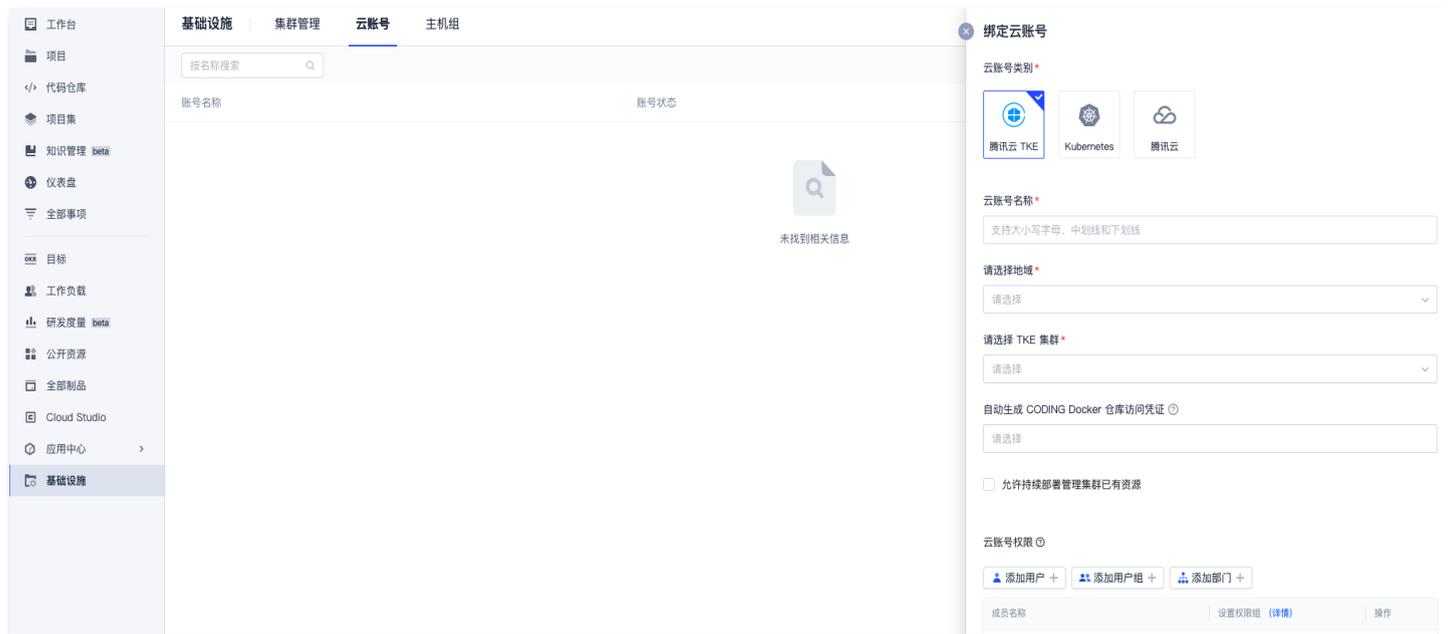
Operation step

Step 1: Obtain and associate a cloud account

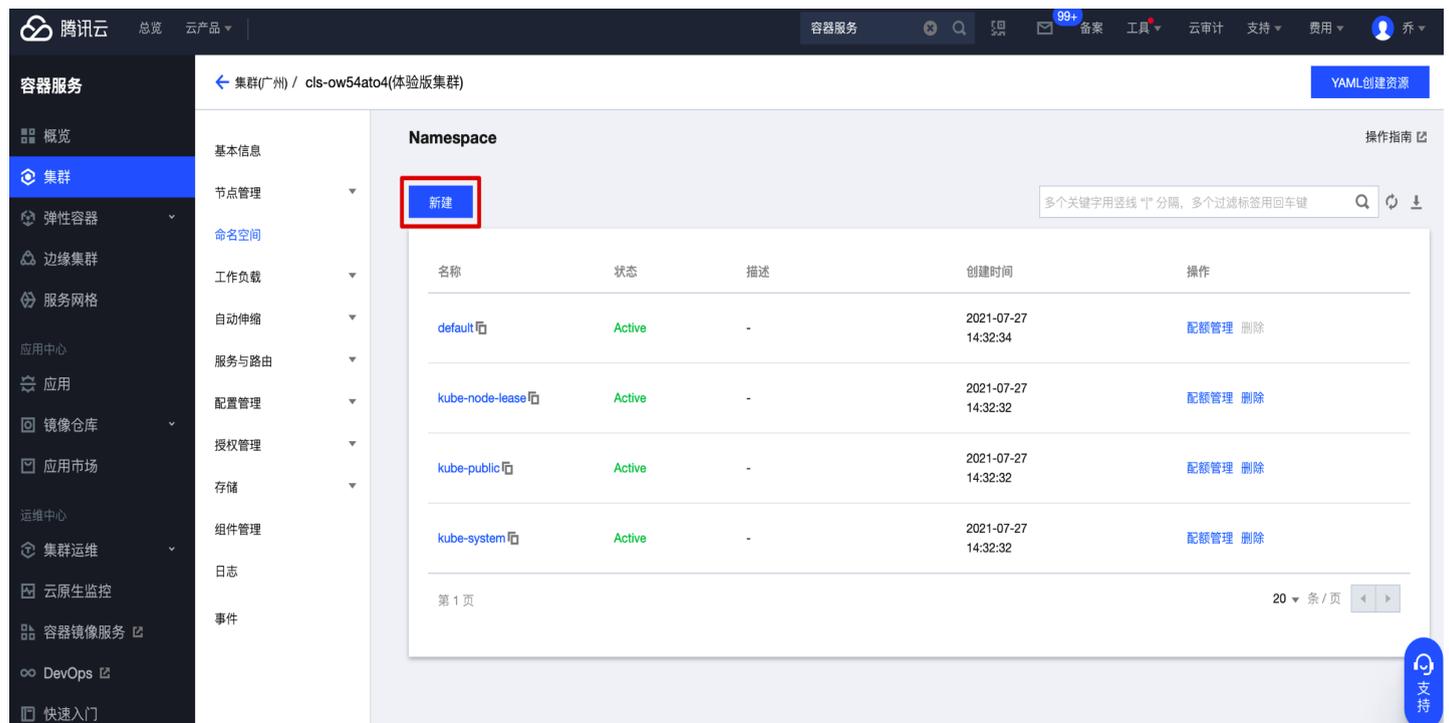
ⓘ Note:

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.

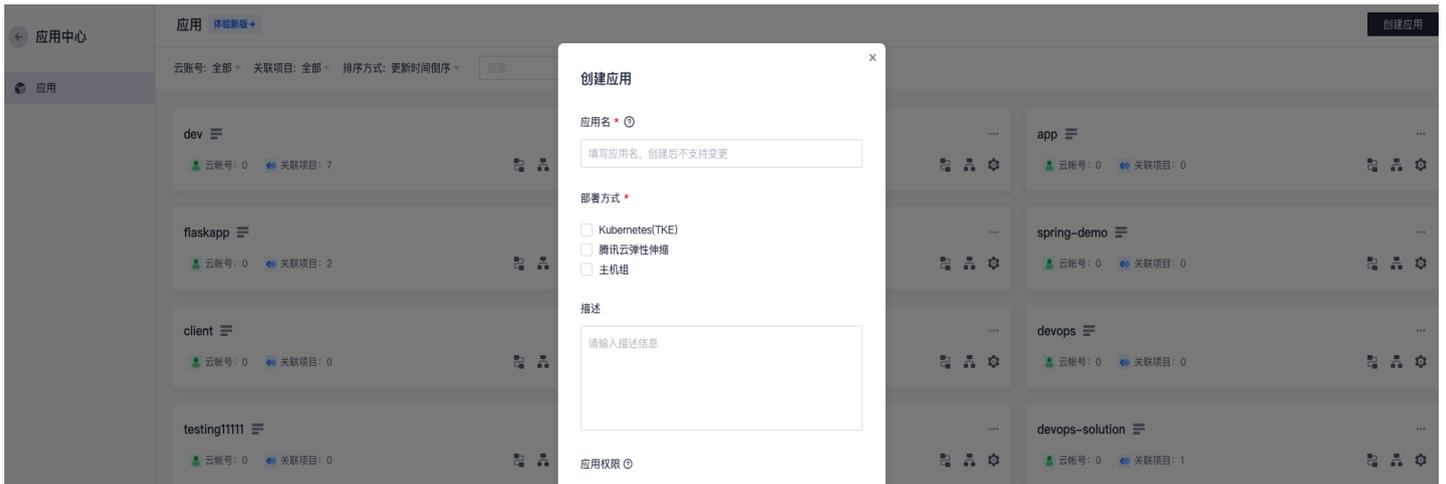


In the cluster, create a new `namespace` to store the automatically generated artifact repository access credentials. The cluster used in this document is named: `cd-demo`.

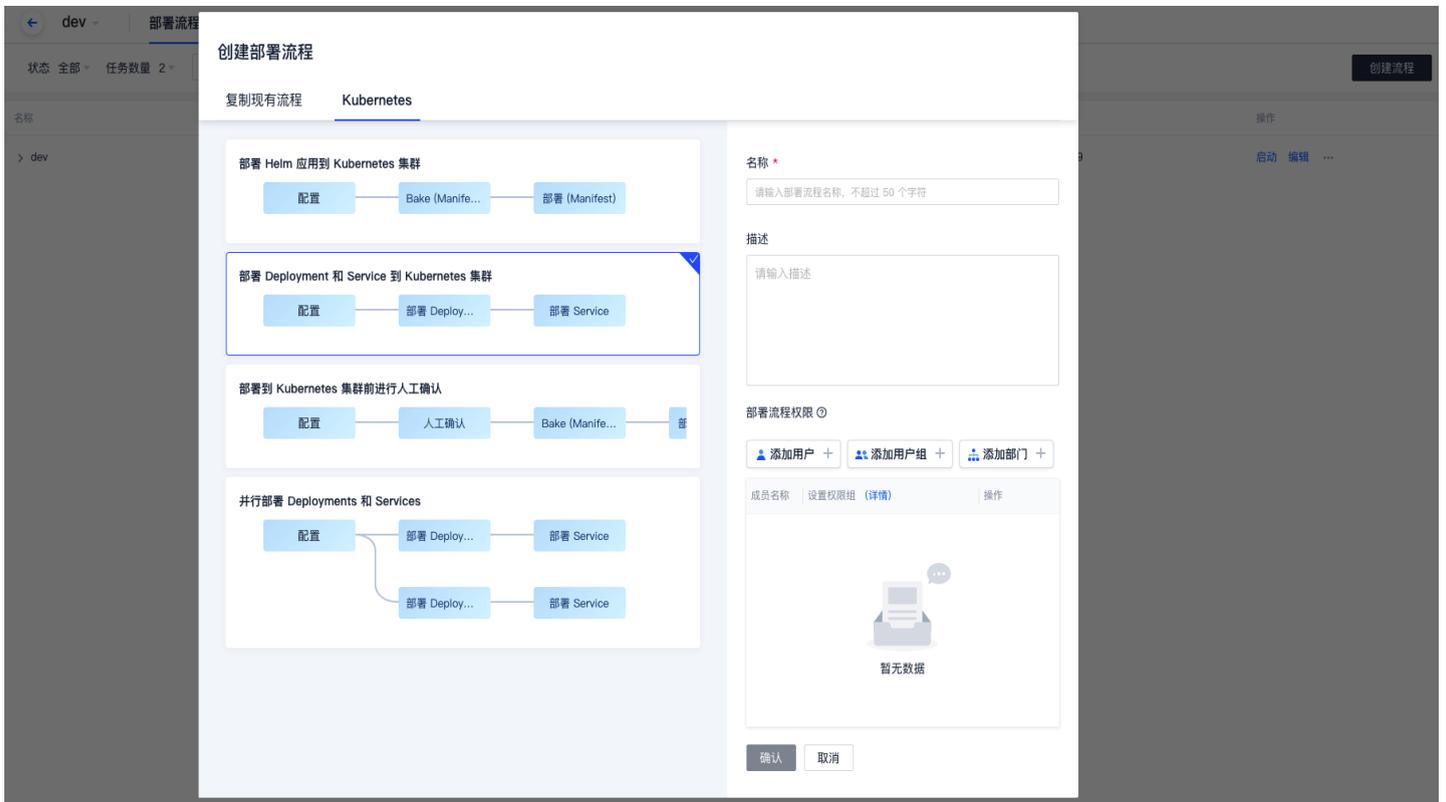


Step 2: Configure the application

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](#).

The screenshot shows a web interface for pushing Docker artifacts. On the left is a sidebar with navigation options: '操作指引' (Operation Guide), '配置凭据' (Configure Credentials), '推送' (Push), '拉取' (Pull), and '镜像源加速' (Image Source Acceleration). The main content area is titled '推送' (Push) and contains the following instructions:

输入以下推送相关信息，生成推送命令：

本地镜像 tag:

制品名称:

制品版本:

1. 请在命令行执行以下命令给本地镜像打标签：

```
docker tag <LOCAL_IMAGE_TAG> chasylee-docker.pkg.coding.net/typical/docker/<PACKAGE>:<VERSION>
```

2. 请在命令行执行以下命令进行推送：

```
docker push [REDACTED].pkg.coding.net/typical/docker/<PACKAGE>:<VERSION>
```

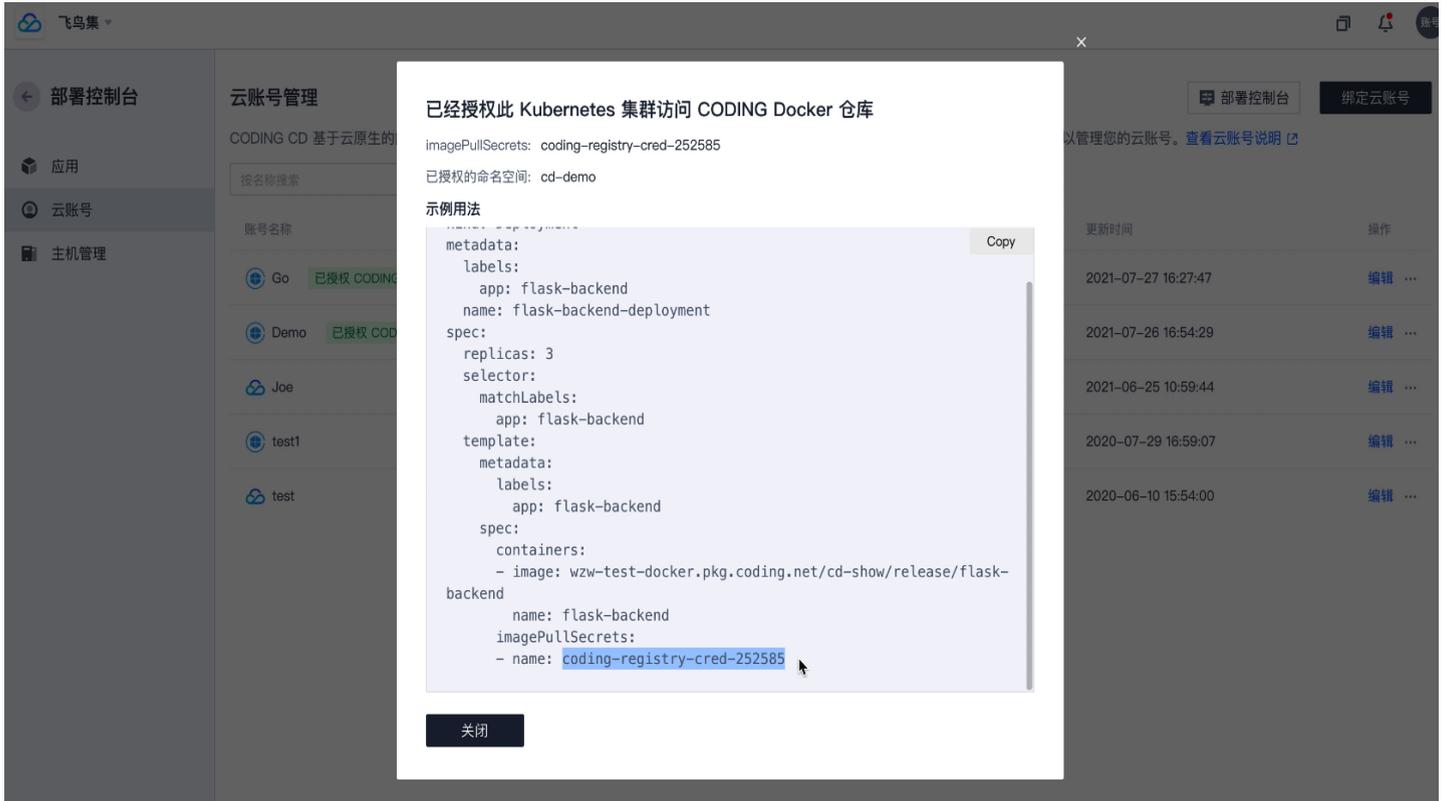
A 'Copy' button is located next to the second command.

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.

The screenshot displays the Tencent Cloud CODING DevOps interface for a repository named 'k8sDemo'. The left sidebar shows a file tree with folders 'gradle/wrapper' and 'k8s', and files 'deployment...', 'service.yaml', '.gitignore', 'Dockerfile', 'build.gradle', 'gradlew', 'gradlew.bat', and 'settings.gradle'. The main area shows the 'deployment.yaml' file in the 'k8s' folder, with a commit history of 6 items. The file content is as follows:

```
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    labels:
5      app: k8sdemo
6    name: k8sdemo-deployment
7    namespace: cd-demo
8  spec:
9    replicas: 1
10   selector:
11     matchLabels:
12       app: k8sdemo
13   template:
14     metadata:
15       labels:
16         app: k8sdemo
17     spec:
18       imagePullSecrets:
19         - name: coding-registry-cred-252585
20       containers:
21         - image: 'StrayBirds-docker.pkg.coding.net/flask-demo/cd-demo/hello-world'
22           name: k8sdemo
23           ports:
24             - containerPort: 8080
```

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.

```
1 apiVersion: apps/v1
2 kind: Deployment
3 metadata:
4   labels:
5     app: k8sdemo
6   name: k8sdemo-deployment
7   namespace: cd-demo
8 spec:
9   replicas: 1
10  selector:
11    matchLabels:
12      app: k8sdemo
13  template:
14    metadata:
15      labels:
16        app: k8sdemo
17    spec:
18      imagePullSecrets:
19        - name: coding-registry-cred-252585
20      containers:
21        - image: 'StrayBirds-docker.pkg.coding.net/flask-demo/cd-demo/hello-world'
22          name: k8sdemo
23          ports:
24            - containerPort: 8080
```

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

```
1 apiVersion: v1
2 kind: Service
3 metadata:
4   name: k8sdemo
5   namespace: cd-demo
6 spec:
7   selector:
8     app: k8sdemo
9   ports:
10    - port: 8080
11      targetPort: 8080
12   type: LoadBalancer
```

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.

The screenshot displays the configuration for the '部署 Deployment' stage in a pipeline. The configuration panel is titled '部署 (Manifest) 配置' and includes the following settings:

- Manifest 来源:** 使用制品 输入内容
- 制品来源:** CODING 代码库
- 项目:** Flask Demo
- 仓库:** k8sDemo
- 默认分支或标签:** master
- 文件路径:** k8s/deployment.yaml

Additional options include '高级配置' (Advanced Configuration) and '跳过 SpEL 表达式计算' (Skip SpEL Expression Calculation).

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

部署 Service [✎](#) ...

依赖阶段: 部署 Deployment ▾

部署 (Manifest) 配置 执行选项 通知 描述

Manifest 来源 ⓘ

使用制品 输入内容

制品来源

CODING 代码库 ▾

项目

演示项目 ▾

仓库

CD-demo ▾

默认分支或标签 ⓘ

master ▾

文件路径

k8s/service.yaml ▾

高级配置

跳过 SpEL 表达式计算 ⓘ

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.

▼ 镜像版本配置

Manifest 中的镜像版本默认支持动态替换，即启动部署流程时可以指定版本覆盖 Manifest 中的默认版本。查看 [帮助文档](#)

镜像来源

自动获取 上游阶段生成

镜像名称

镜像地址

高级配置

自定义版本规则 锁定默认版本 忽略版本

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.

cd-demo

基础配置

部署 Deployment
阶段类型: 部署 (Manifest)

制品

- k8s/deployment.yaml
master
- k8s/service.yaml
master
- hello-world
--

基础配置

执行选项 自动触发器 启动参数 通知 描述

自动触发器

▼ CODING docker 仓库触发器

触发器启用开关

触发器类型

CODING docker 仓库触发器

- CODING docker 仓库触发器
- TCR 个人版仓库触发器
- TCR 企业版仓库触发器
- TCR Helm 仓库触发器
- Git 仓库触发器

webhook 触发器

- 定时触发器
- CODING Generic 仓库触发器

通知

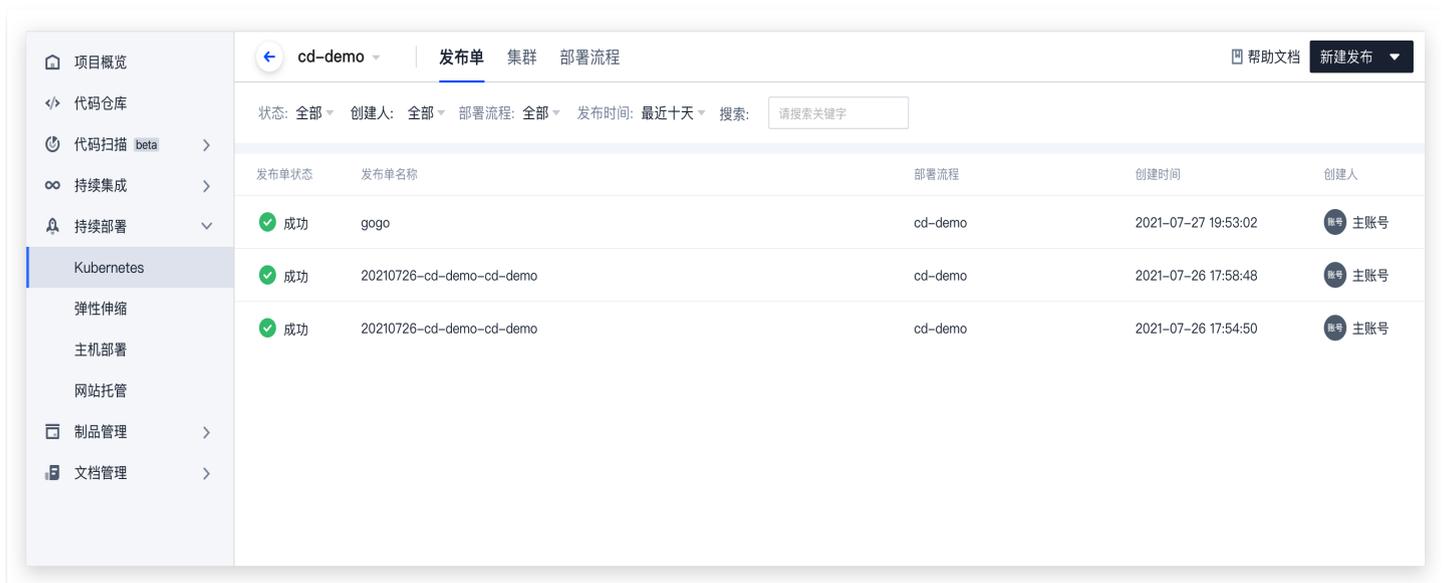
暂无通知

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.

gogo 成功

部署 Deployment

状态 成功 开始时间 2021-07-27 19:53:03
耗时 19 秒

阶段

成功
部署 Deployment
耗时: 19 秒

基础信息

手动触发

主账号

2021-07-27 19:53:02

25 秒

制品

- StrayBirds-docker.pkg.coding.net/flask-demo/cd-demo/hello-world latest
- k8s/deployment.yaml master
- k8s/service.yaml master

阶段详情

状态	脚本名称	启动时间	耗时
成功	部署 Deployment	2021-07-27 19:53:03	19 秒

DeployStatus Task Status Artifact Status

Deployment k8sdemo-deployment [查看 Yaml 内容](#) [跳转查看资源详情](#)

ScalingReplicaSet

3 分钟 以前

Scaled up replica set k8sdemo-deployment-994479977 to 1

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).

cd-demo ▾ | 发布单 集群 部署流程

工作负载 服务 云账号: 全部 ▾ 命名空间: 全部 ▾ 类型: 全部 ▾ 状态: 全部 ▾

名称	命名空间	云账号
deployment k8sdemo-deployment	cd-demo	Go

V001 StrayBirds-docker.pkg.coding.net/flask-demo/cd-demo/hello-world:latest  Load B

k8sdemo-deployment-994479977

操作 ▾

基础信息

创建时间 2021-07-27 19:53:05
云账号 [Go](#)
命名空间 cd-demo
资源类型 replicaSet
控制器 [deploymentk8sdemo-deployment](#)

镜像

StrayBirds-docker.pkg.coding.net/flask-demo/cd-demo/hello-world:latest ⓘ

事件

1 x SuccessfulCreate

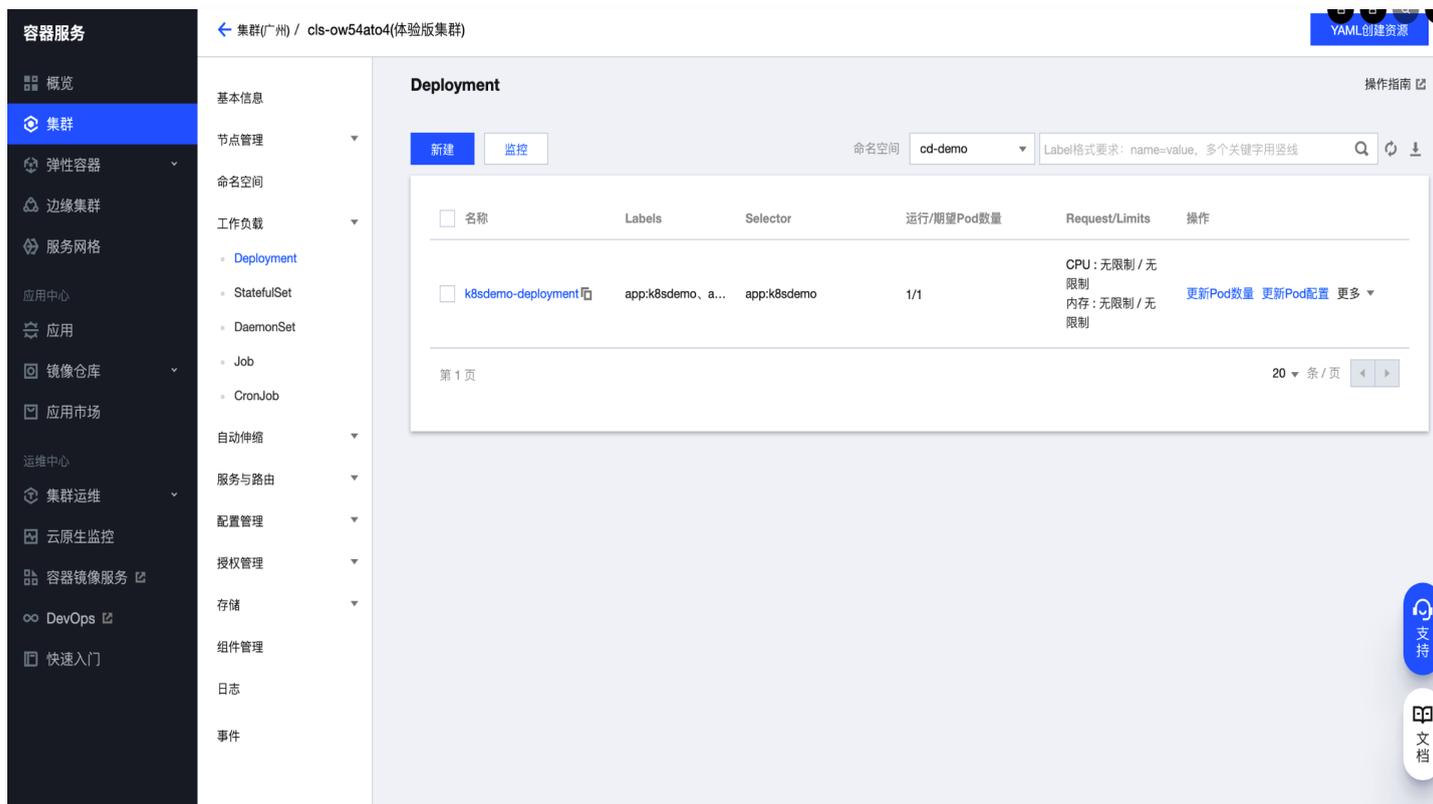
- 以前

Created pod: k8sdemo-deployment-994479977-nmkdx

LABELS

- app:k8sdemo
- app.kubernetes.io/managed-by:spinnaker
- app.kubernetes.io/name:cd-demoteam174750
- pod-template-hash:994479977

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.

制品仓库 | 全部制品 | 仓库管理 创建制品仓库

ruby 设置仓库 代理设置 版本覆盖策略

类型 Docker | 权限 项目内

镜像列表

发布状态 全部 + 制品属性 搜索制品名称...

镜像名	最新推送版本	最近更新时间	版本号	操作
ruby	dc0c37fa405025c573807dc673cfad...	2021-12-08 11:24:35	1	...
v1.0	dc0c37fa405025c573807dc673cfad...	2021-12-08 10:38:17	1	...

1-2 个, 共 2 个 每页显示行数 15

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

cd-demo

基础配置

制品

- k8s/deployment.yaml master
- k8s/service.yaml master
- hello-world

部署 Deployment
阶段类型: 部署 (Manifest)

基础配置

执行选项 自动触发器 启动参数 通知 描述

自动触发器

CODING docker 仓库触发器

触发器启用开关

触发器类型

CODING docker 仓库触发器

- CODING docker 仓库触发器
- TCR 个人版仓库触发器
- TCR 企业版仓库触发器
- TCR Helm 仓库触发器
- Git 仓库触发器

webhook 触发器

- 定时触发器
- CODING Generic 仓库触发器

通知

暂无通知

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.

The screenshot displays the '绑定云账号' (Bind Cloud Account) interface. On the left, a sidebar lists navigation options like '工作台', '项目', and '基础设施'. The main content area is titled '云账号' and contains a table with the following data:

账号名称	账号状态
adolf	已验证

On the right, the configuration form for a new cloud account includes:

- 云账号类别***: Radio buttons for 腾讯云 TKE (selected), Kubernetes, and 腾讯云.
- 云账号名称***: Input field with a search hint: 支持大小写字母、中划线和下划线.
- 请选择地域***: Dropdown menu.
- 请选择 TKE 集群***: Dropdown menu.
- 自动生成 CODING Docker 仓库访问凭证**: Input field.
- 允许持续部署管理集群已有资源
- 云账号权限**: Buttons for 添加用户 +, 添加用户组 +, and 添加部门 +.
- Table with columns: 成员名称, 设置权限组 (详情), and 操作.

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

腾讯云 总览 云产品 网站备案 +

容器服务

← 集群(上海) / cls-bjvylwsb(k8s-dev)

集群API Server信息

访问地址 https://cls-bjvylwsb.ccs.tencent-cloud.com

外网访问 已开启
已放通IP地址: 212.64.105.0/24; 212.129.144.0/24

内网访问 未开启

Kubeconfig 以下kubeconfig文件为当前子账号的kubeconfig内容:

```
apiVersion: v1
clusters:
- cluster:
  certificate-authority-data:
LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUN5RENDQW
REFWTVJNd0VRWURWUWFERXdwcmRXSmwKY201bGRHVnpNQjRYRF
```

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

基础设施 集群管理 云账号 主机组

按名称搜索

账号名称 账号状态

adolf 已验证

绑定云账号

云账号类别 *

腾讯云 TKE Kubernetes 腾讯云

云账号名称 *

支持大小写字母、中划线和下划线

提示

请确保您的 Kubernetes 集群已开放公网访问, 并将 CODING 持续部署的公网 IP 段添加到集群访问控制列表白名单。

CODING 持续部署的公网 IP 段:

212.64.105.0/24
212.129.144.0/24

选择认证方式 *

Kubeconfig Service Account

Kubeconfig *

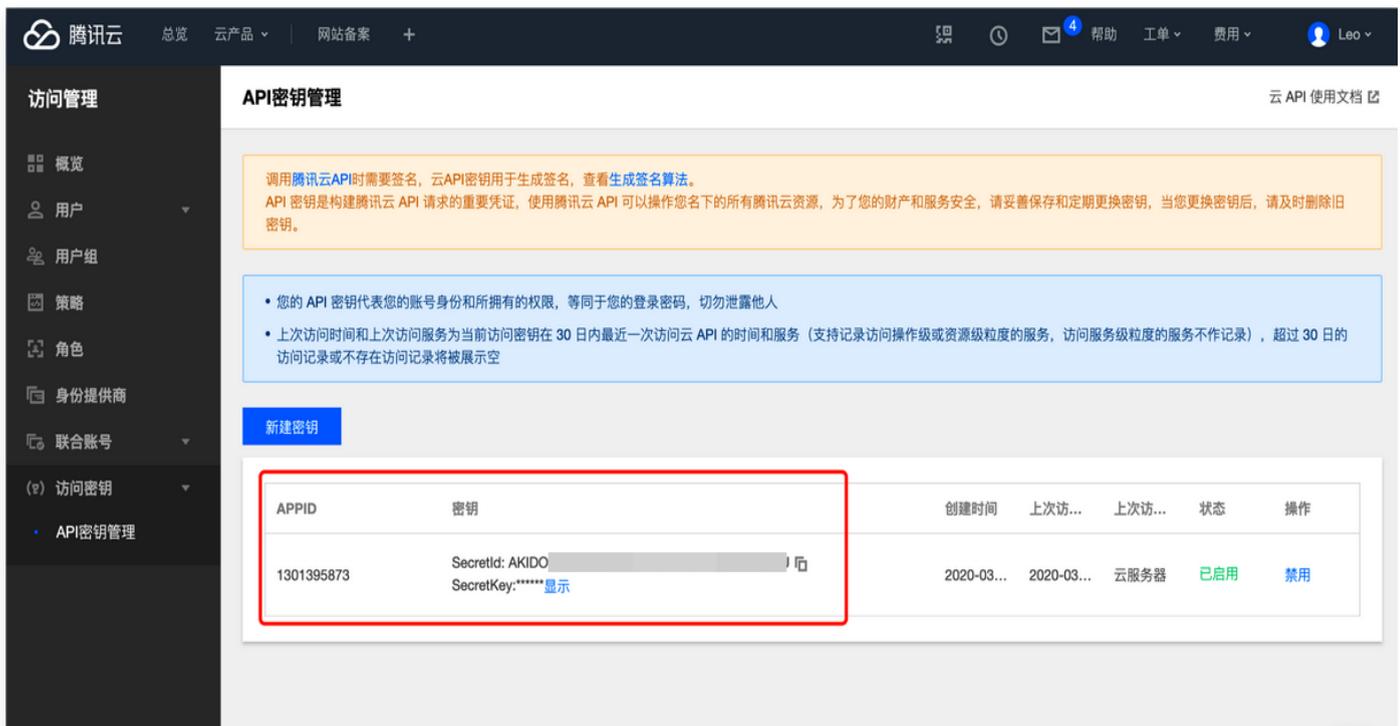
```
apiVersion: v1
clusters:
- cluster:
  certificate-authority-data:
LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUN5RENDQW
REFWTVJNd0VRWURWUWFERXdwcmRXSmwKY201bGRHVnpNQjRYRF
```

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

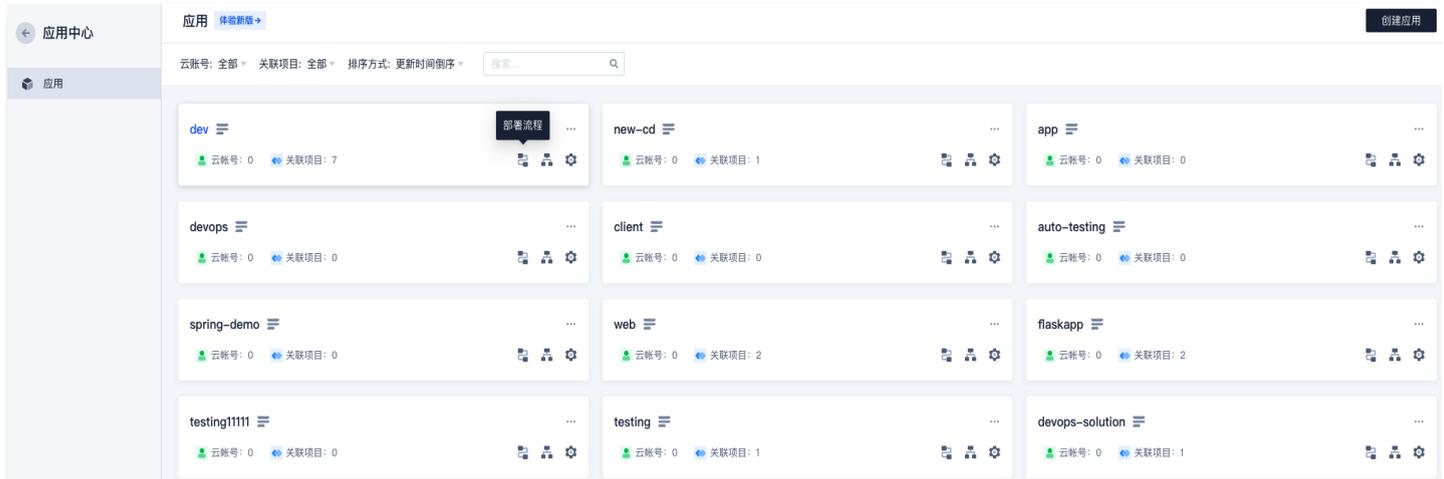
The deployment pipeline is the core module of Continuous Deployment. Its advantage lies in supporting the combination of stages in any sequence, which provides excellent flexibility, consistency, and repeatability.

- **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.

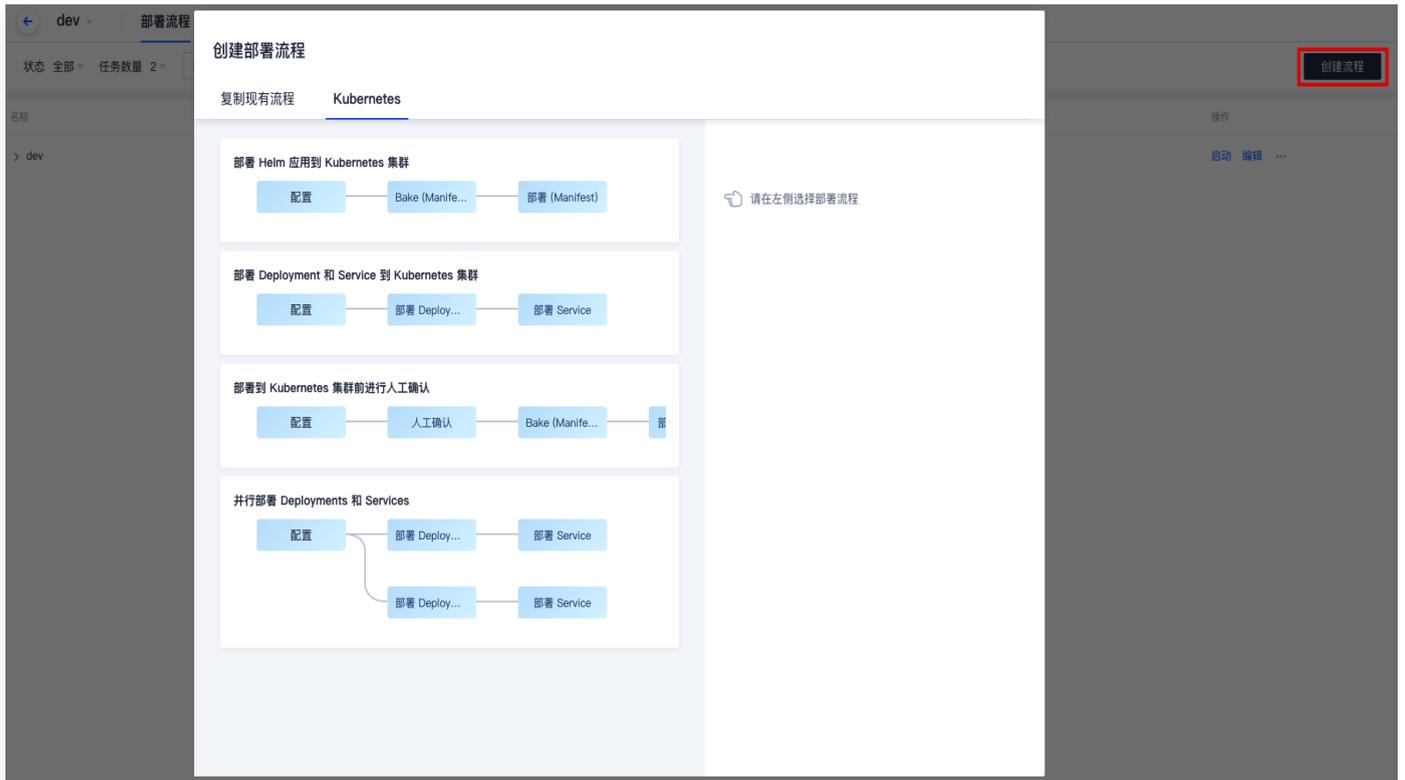
Users can configure a fully automated deployment pipeline or add manual judgment conditions at certain stages. Additionally, the deployment pipeline supports automation triggers by various events, such as webhooks and other deployment pipelines.

Create Deployment Pipeline

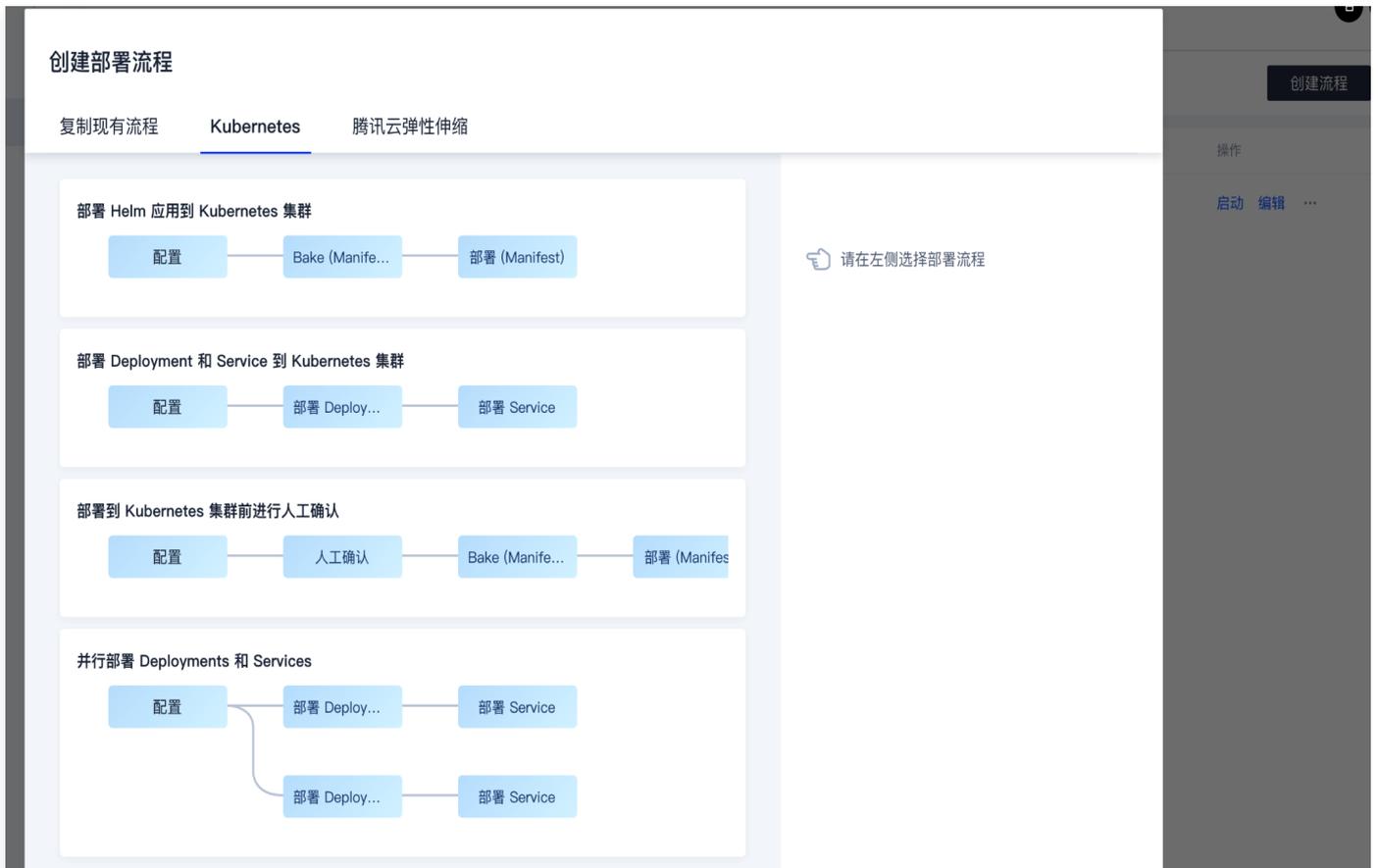
Go to the Application Center, and click the deployment pipeline icon in the lower-right corner of the application card.



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.

基础配置

执行选项 自动触发器 启动参数 通知 描述

执行选项

- 禁止本流程并行执行 (同一时间只能执行一个部署)
- 不要自动取消在排队状态的部署执行任务

自动触发器

暂无触发器

+ 添加触发器

启动参数

暂无启动参数

+ 添加启动参数

通知

暂无通知

+ 添加通知设置

描述

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.

The screenshot displays the 'Deployment Pipeline Configuration' interface. On the left, a pipeline diagram shows a '基础配置' (Basic Configuration) stage connected to a '请选择阶段' (Please select a stage) stage. The '基础配置' stage is expanded to show a '制品' (Artifact) section with the text '暂无制品 可在阶段中配置' (No artifacts, can be configured in the stage). The right sidebar is titled '基础配置' and has tabs for '执行选项' (Execution Options), '自动触发器' (Automatic Triggers), '启动参数' (Start Parameters), '通知' (Notifications), and '描述' (Description). The '启动参数' tab is active, showing a '启动参数' (Start Parameter) section with a trash icon. Below this, there are fields for '参数名' (Parameter Name) with a '请输入' (Please enter) placeholder, a checkbox for '必填参数' (Required Parameter), a '参数类型' (Parameter Type) dropdown menu set to '字符串' (String), a '默认值' (Default Value) field with a '请输入' (Please enter) placeholder, and a '描述信息' (Description Information) field with a '请输入' (Please enter) placeholder. At the bottom of the sidebar, there is a '+ 添加启动参数' (Add Start Parameter) button.

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.

GO

请选择阶段
阶段类

添加阶段

请选择阶段
阶段类型: --

选择阶段

搜索阶段名称

依赖阶段: 请选择阶段

Kubernetes 通用类型

- 部署 (Manifest)**
部署 yaml/json 格式的 Kubernetes manifest 文件 选择
- 过滤 (Manifest)**
过滤 (Manifest) 选择
- 扩缩容 (Manifest)**
对 Kubernetes 对象执行扩缩容 选择
- 回滚 (Manifest)**
回滚至目标版本 选择
- 删除 (Manifest)**
删除 Kubernetes (Manifest) 选择
- Bake (Manifest)**
使用 Helm Bake manifest 文件 选择
- Patch (Manifest)**
Patch a Kubernetes object in place. 选择

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.

- 🏠 项目概览
- 👤 项目协同
- 📁 代码仓库
- 🔄 代码扫描 beta
- ∞ 持续集成
- 🔔 持续部署
- Kubernetes
- 弹性伸缩
- 主机部署
- 网站托管
- 📦 制品管理
- 🧪 测试管理
- 📄 文档管理

Kubernetes 应用部署

您可以提交发布单部署 Kubernetes 应用，并在部署成功后查看应用信息。更多内容查看 [帮助文档](#)

🏠 📄

状态: 全部 ▾

排序方式: 全部 ▾

请输入应用名称 🔍

flaskapp

🔔

该应用最近一次发布单状态

○ 发布单

🏠 集群

test

🔔

最近一次发布于 2020-04-13 16:43:37

- 发布单

🏠 集群

Deployment Pipeline Configuration

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

← app 测试 [🔗](#)
撤销变更
已同步
⋮

⚙️ 基础配置

制品

- 🔗 flaskapp
-

+

🏠 部署 (Manifest)

阶段类型: 部署 (Manifest)

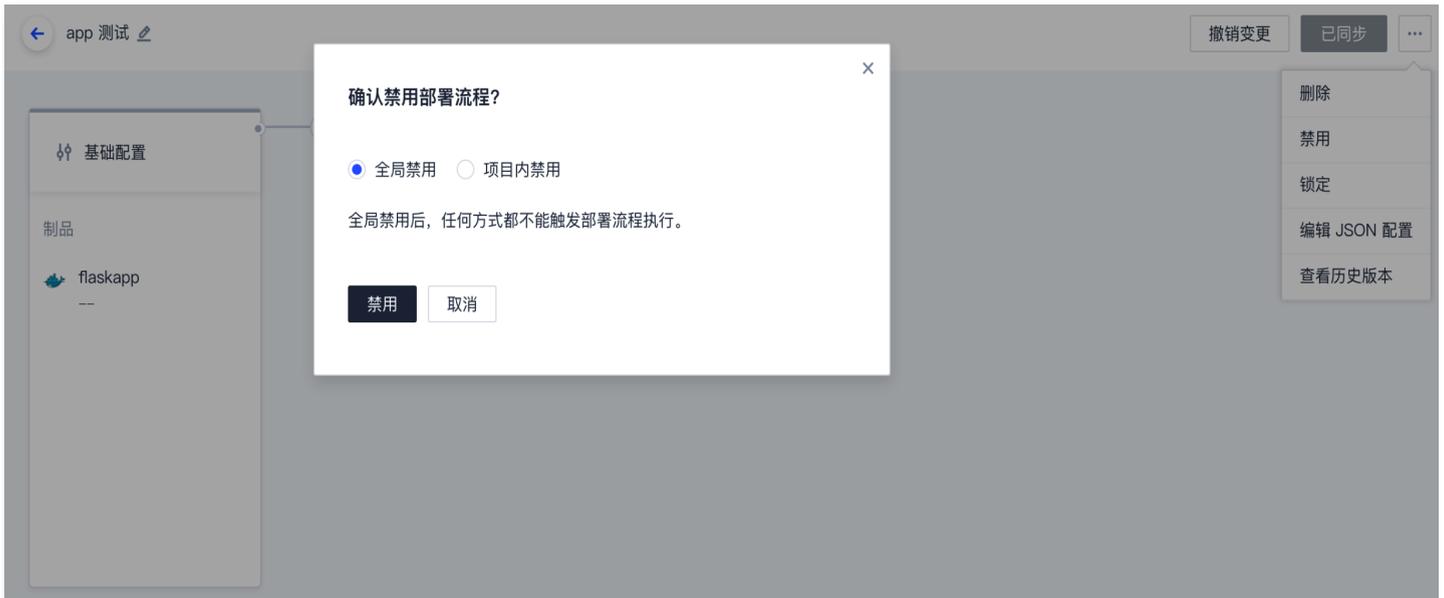
- 删除
- 禁用
- 锁定
- 编辑 JSON 配置
- 查看历史版本

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.

修订历史

修订版本 2021-06-16 10:25:41 (当前版本)

对比 上一版本

```
"application": "flaskapp",
"codingNickname": "主账号",
"desc": "",
"expectedArtifacts": [
  {
    "defaultArtifact": {
      "customKind": false,
      "id": "8b215t...",
      "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": "ff025686-...",
      "name": "lhkprod-docker.pkg.coding.net/cd-demo/cd-demo/flaskapp",
      "parentType": "docker",
      "pkgName": "flaskapp",
      "type": "coding_docker/image"
    },
    "useDefaultArtifact": true,
    "usePriorArtifact": false
  }
]
```

关闭

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

```
1 {
2   "codingNickname": "主账号",
3   "desc": "",
4   "expectedArtifacts": [
5     {
6       "defaultArtifact": {
7         "customKind": false,
8         "id": "8b215b5c-...",
9         "name": "lhkprod-docker.pkg.coding.net/cd-demo/cd-demo/flaskapp",
10        "parentType": "docker",
11        "pkgName": "flaskapp",
12        "reference": "lhkprod-docker.pkg.coding.net/cd-demo/cd-demo/flaskapp",
13        "type": "coding_docker/image",
14        "version": ""
15      },
16      "displayName": "flaskapp",
17      "id": "cf1aedbd-...",
18      "matchArtifact": {
19        "customKind": false,
20        "id": "ff025-...",
21        "name": "lhkprod-docker.pkg.coding.net/cd-demo/cd-demo/flaskapp",
22        "parentType": "docker",
23        "pkgName": "flaskapp",
24        "type": "coding_docker/image"
25      },
26      "useDefaultArtifact": true,
27      "usePriorArtifact": false
28    }
29  ]
30 }
```



取消

保存变更

Deployment mode Trigger When Docker Artifacts Are Auto Released

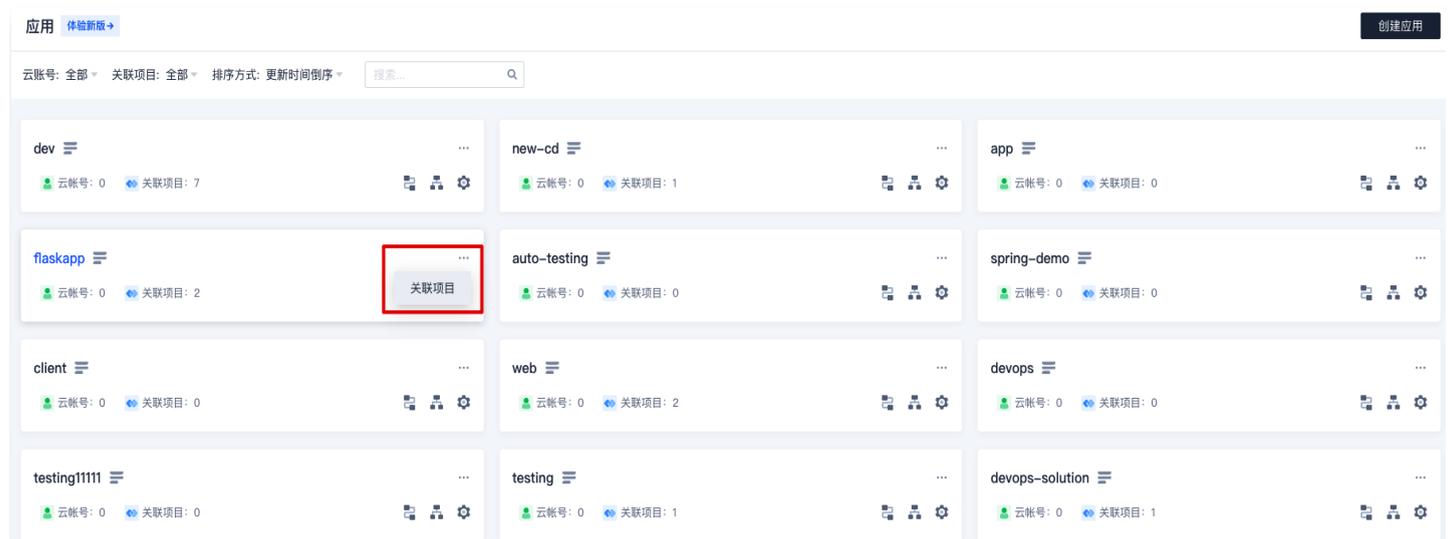
Last updated: 2024-09-05 16:44:07

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.

选择构建计划模版

自定义构建过程

构建计划是持续集成的基本单元，在这里你可以快速创建一个构建计划，更多内容可以到构建计划详情中进行配置。[查看帮助文档](#)

全部 团队模版 编程语言 Serverless **镜像仓库** 制品库 部署 基础 API 文档

请输入模版关键字进行搜索



构建镜像并推送到 TCR 企业版

基于源代码变更自动触发镜像构建，并推送镜像至容器镜像服务TCR企业版...



构建镜像并推送至 TCR 个人版 (容器服务-镜像仓库)

基于源代码变更自动触发镜像构建，并推送镜像至容器镜像服务TCR个人版...



CODING Docker 镜像推送

将一个构建完毕的 Docker 镜像推送到当前项目下的 Docker 制品库中

若没有找到合适的模版，可选择自定义构建过程



自定义构建过程

允许您根据 Jenkinsfile 的规范来随意定制持续集成流水线过程。

Add this stage to the CI process:

flask-docker | 基本信息 **流程配置** 触发规则 变量与缓存 通知提醒 前往最新构建 操作 立即构建

静态配置的 Jenkinsfile 图形化编辑器 文本编辑器 环境变量 丢弃修改 保存

5-1 构建 Docker 镜像 → 6-1 推送到 CODING D... + 增加阶段

执行 Shell 脚本 执行 Pipeline 脚本 + 增加并行阶段

执行 Pipeline 脚本

插件配置 高级配置

```

Pipeline 脚本 *
1 docker.withRegistry(
2   "${env.CCI_CURRENT_WEB_PROTOCOL}://${env.CODING_D
3   "${env.CODING_ARTIFACTS_CREDENTIALS_ID}"
4 ) {
5   docker.image("${env.CODING_DOCKER_IMAGE_NAME}:${env.DOC
6 }

```

Jenkinsfile reference:

```

stage('Push to CODING Docker artifact repository') {
  steps {
    script {
      docker.withRegistry(
        "${env.CCI_CURRENT_WEB_PROTOCOL}://${env.CODING_DOCKER_REG_HOST}",
        "${env.CODING_ARTIFACTS_CREDENTIALS_ID}"
      ) {
        docker.image("${env.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.

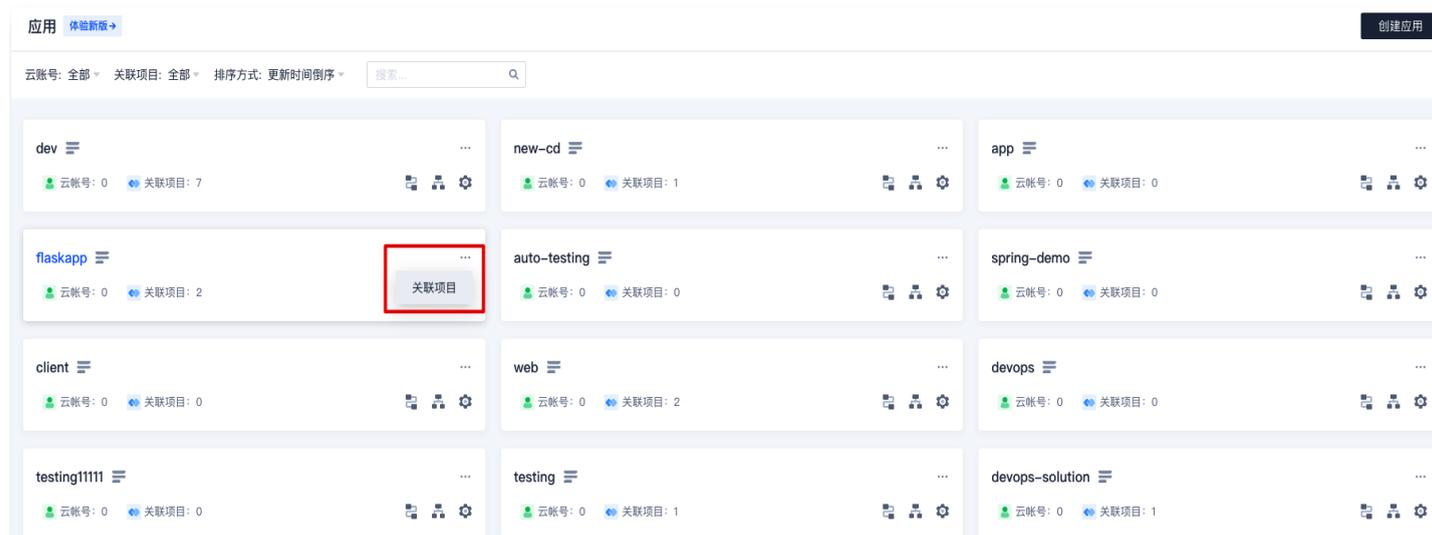
The screenshot displays the '部署控制台' (Deployment Console) for an application named 'app 测试'. The main workflow consists of two stages: '基础配置' (Basic Configuration) and '部署 (Manifest)' (Deployment (Manifest)). The '基础配置' stage is highlighted with a red box. The '部署 (Manifest)' stage is also highlighted with a red box. The '基础配置' stage is currently selected, and its configuration is shown in the right-hand panel. The '基础配置' panel shows the '制品' (Artifact) section with 'flaskapp' listed. The '自动触发器' (Automatic Trigger) section is expanded, showing the following settings:

- 执行选项: 自动触发器, 启动参数, 通知, 描述
- 自动触发器
 - ▼ CODING docker 仓库触发器
 - 触发器启用开关:
 - 触发器类型: CODING docker 仓库触发器
 - 绑定现有制品 自定义
 - 制品名称: flaskapp
- 添加触发器
- 启动参数
 - 暂无启动参数
 - 添加启动参数
- 通知
 - 暂无通知

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**.

选择构建计划模板

自定义构建过程

构建计划是持续集成的基本单元，在这里你可以快速创建一个构建计划，更多内容可以到构建计划详情中进行配置。[查看帮助文档](#)

全部

团队模板

编程语言

Serverless

镜像仓库

制品库

部署

基础

API 文档

请输入模板关键字进行搜索



CODING Docker 镜像推送并部署到 Kubernetes

将一个构建完毕的 Docker 镜像推送到当前项目下的 Docker 制品库中并...

若没有找到合适的模板，可选择自定义构建过程



自定义构建过程

允许您根据 Jenkinsfile 的规范来随意定制持续集成流水线过程。

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.

2 构建 Docker 镜像

Docker 镜像名称 *

my-docker-image

Dockerfile 文件位置 *

Dockerfile

Docker 构建目录 *

.

Docker 镜像版本 *

分支名-修订版本号 (\${GIT_LOCAL_BRANCH:-branch})-\$

3 推送到 CODING Docker 制品库

Docker 制品库 *

cd-demo

4 部署到远端 Kubernetes 集群

集群 *

Go

命名空间 *

demo

```

}

stage('构建镜像并推送到 CODING Docker 制品库') {
  steps {
    script {
      docker.withRegistry(
        "${CCI_CURRENT_WEB_PROTOCOL}://${CODING_DOCKER_REG_HOST}",
        "${CODING_ARTIFACTS_CREDENTIALS_ID}"
      ) {
        def dockerImage = docker.build("${CODING_DOCKER_IMAGE_NAME}:${DOCKER_IMAGE_VERSION}",
          dockerImage.push()
        }
      }
    }
  }
}

stage('部署到远端 Kubernetes 集群') {
  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}:${DOCKER_IMAGE_VERSION}",
      cloudAccountName: "${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}",
    ])
  }
}
}
}

```

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.

The screenshot displays the Jenkins graphical editor interface. The top navigation bar includes tabs for '基础信息', '流程配置', '触发规则', '变量与缓存', and '通知提醒'. The main workspace shows a pipeline flow with a stage '5-1 推送到 CODING D...' containing a step '镜像更新'. A configuration panel for this step is open on the right, with the following fields:

- 镜像 ***: cd-demo
- 集群 ***: Go
- 命名空间 ***: 请选择命名空间
- 资源类型 ***: 请选择资源类型
- 资源名称 ***: 请选择资源名称
- Pod 容器 ***: 请选择 Pod

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.

快速发布配置

快速发布适用于简单发布场景，如需配置更灵活复杂的部署流程，请前往[部署控制台](#)

① 集群配置 ———— ② 制品配置 ———— ③ 应用部署

制品类型 镜像 Manifest 文件 Helm 包

仓库类型

镜像仓库

镜像名称

镜像版本

制品配置说明

- 当制品类型选择镜像时，仓库类型支持 CODING Docker 仓库、Tencent TCR 企业版和 Tencent TCR 个人版。
 - [1. Tencent TCR 企业版说明](#)
 - [2. Tencent TCR 个人版说明](#)
- 当制品类型为 Manifest 文件时，需要输入 Manifest 文件在代码仓库中的相对路径，例如：k8s/deployment.yaml

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

cd-demo-快速发布 ✔ 成功

Patch (Manifest)

状态 成功 开始时间 2021-07-29 14:47:05
耗时 6 秒

阶段详情

状态	脚本名称	启动时间	耗时
✔ 成功	Patch (Manifest)	2021-07-29 14:47:05	6 秒

DeployStatus Task Status Artifact Status

Deployment k8sdemo-deployment

[查看 Yaml 内容](#) [跳转查看资源详情](#)

ScalingReplicaSet

1 小时 17 分钟 以前

Scaled down replica set k8sdemo-deployment-994479977 to 0

ScalingReplicaSet

1 小时 28 分钟 以前

Scaled down replica set k8sdemo-deployment-7485579c9c to 0

ScalingReplicaSet

1 小时 44 分钟 以前

Scaled up replica set k8sdemo-deployment-7485579c9c to 1

ScalingReplicaSet

基础信息

手动触发

账号 主账号

2021-07-29 14:45:06

6 秒

制品



暂无制品

阶段

✔ 成功

Patch (Manifest)

耗时: 6 秒