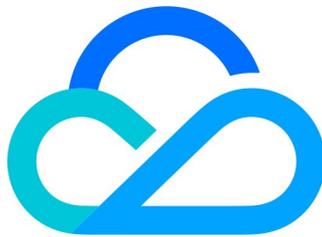


CODING DevOps

持续部署



腾讯云

【 版权声明 】

©2013–2025 腾讯云版权所有

本文档（含所有文字、数据、图片等内容）完整的著作权归腾讯云计算（北京）有限责任公司单独所有，未经腾讯云事先明确书面许可，任何主体不得以任何形式复制、修改、使用、抄袭、传播本文档全部或部分內容。前述行为构成对腾讯云著作权的侵犯，腾讯云将依法采取措施追究法律责任。

【 商标声明 】



及其它腾讯云服务相关的商标均为腾讯云计算（北京）有限责任公司及其关联公司所有。本文档涉及的第三方主体的商标，依法由权利人所有。未经腾讯云及有关权利人书面许可，任何主体不得以任何方式对前述商标进行使用、复制、修改、传播、抄录等行为，否则将构成对腾讯云及有关权利人商标权的侵犯，腾讯云将依法采取措施追究法律责任。

【 服务声明 】

本文档意在向您介绍腾讯云全部或部分产品、服务的当时的相关概况，部分产品、服务的内容可能不时有所调整。您所购买的腾讯云产品、服务的种类、服务标准等应由您与腾讯云之间的商业合同约定，除非双方另有约定，否则，腾讯云对本文档内容不做任何明示或默示的承诺或保证。

【 联系我们 】

我们致力于为您提供个性化的售前购买咨询服务，及相应的技术售后服务，任何问题请联系 4009100100或 95716。

文档目录

持续部署

快速开始

云账号

部署流程管理

部署方式

自动发布 Docker 制品时触发

在构建计划中添加部署阶段

手动提交发布单

持续部署 快速开始

最近更新时间：2025-03-17 15:33:52

说明：

CODING DevOps 于2025年9月1日起更新 CODING 订购方案，取消原标准版套餐，下线部分功能（制品安全扫描、测试管理、测试协同、仪表盘、研发度量），新注册团队用户界面无持续部署、应用管理功能，为确保您的使用权益和资产数据安全，请及时关注并处理，[了解更多详情](#)。

功能介绍

CODING 持续部署用于把控构建之后的项目发布与部署交付流程。能够无缝对接上游 Git 仓库、制品仓库以实现全自动化部署。在稳定的技术架构、运维工具等基础上，具备蓝绿发布，灰度发布（金丝雀发布），滚动发布，快速回滚等能力。

下文将以一个简单的 Demo 项目为例，演示如何使用 CODING 持续部署控制台将应用发布至腾讯集群。

前置准备

- 开启持续部署设置权限，请参见 [权限详情](#)。
- 导入 [示例代码库](#)。点击了解如何 [导入或关联外部仓库](#)。
- Docker 制品仓库，单击了解如何使用项目中的 [Docker 制品仓库](#)。
- 一个可被 CODING 持续部署访问的 Kubernetes 集群，单击了解如何申请 [腾讯云标准集群](#)。

说明：

集群的推荐配置为 8 核 32 GB，需开启公网访问权限。

操作步骤

步骤1：获取并关联云账号

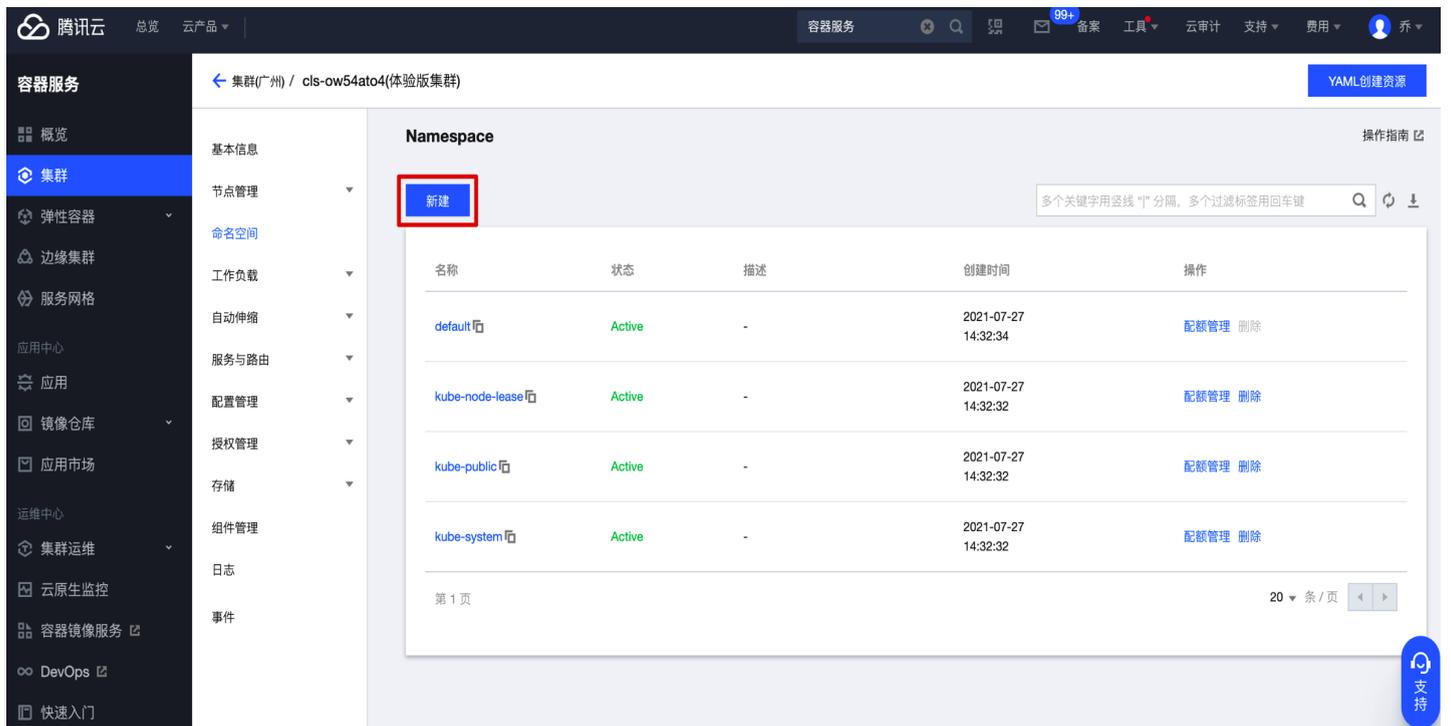
说明：

因使用了腾讯云容器服务，部署后应用将发布至集群，本示例使用的团队账号已在 [团队设置中心](#) > [第三方应用中关联 腾讯云账号](#)。

单击首页左侧的[基础设施](#)，在[云账号](#)中绑定腾讯云账号。云账号名称可以自拟，选择地域后将自动获取对应集群。

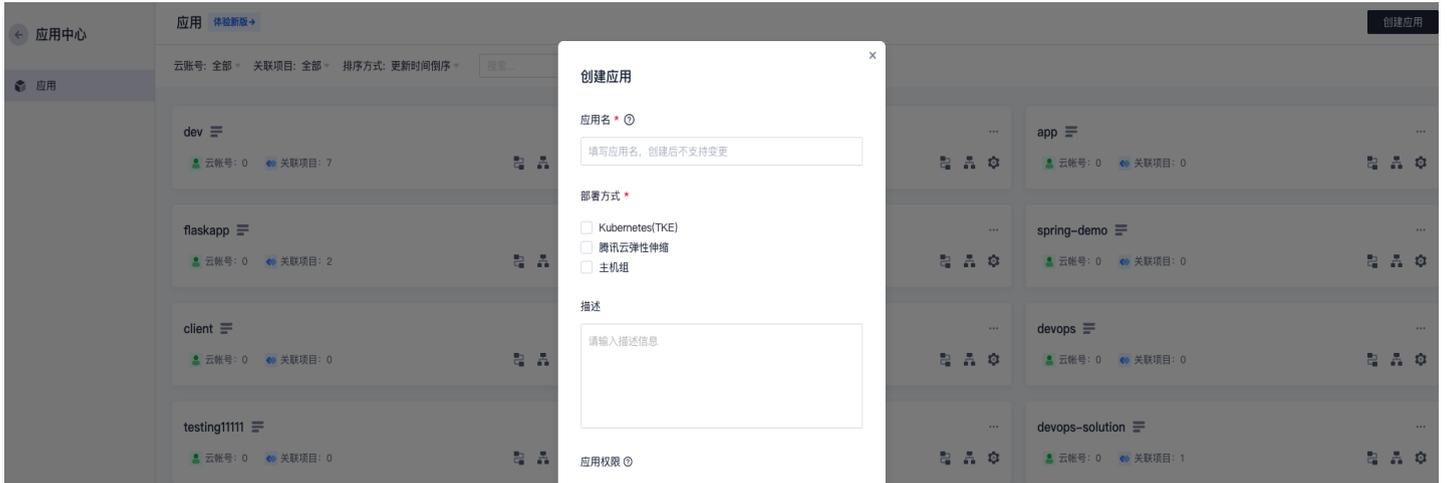


在集群中新建命名空间 (Namespace) 用于存储自动生成的制品仓库访问凭证，本文中所使用的集群命名为：cd-demo。

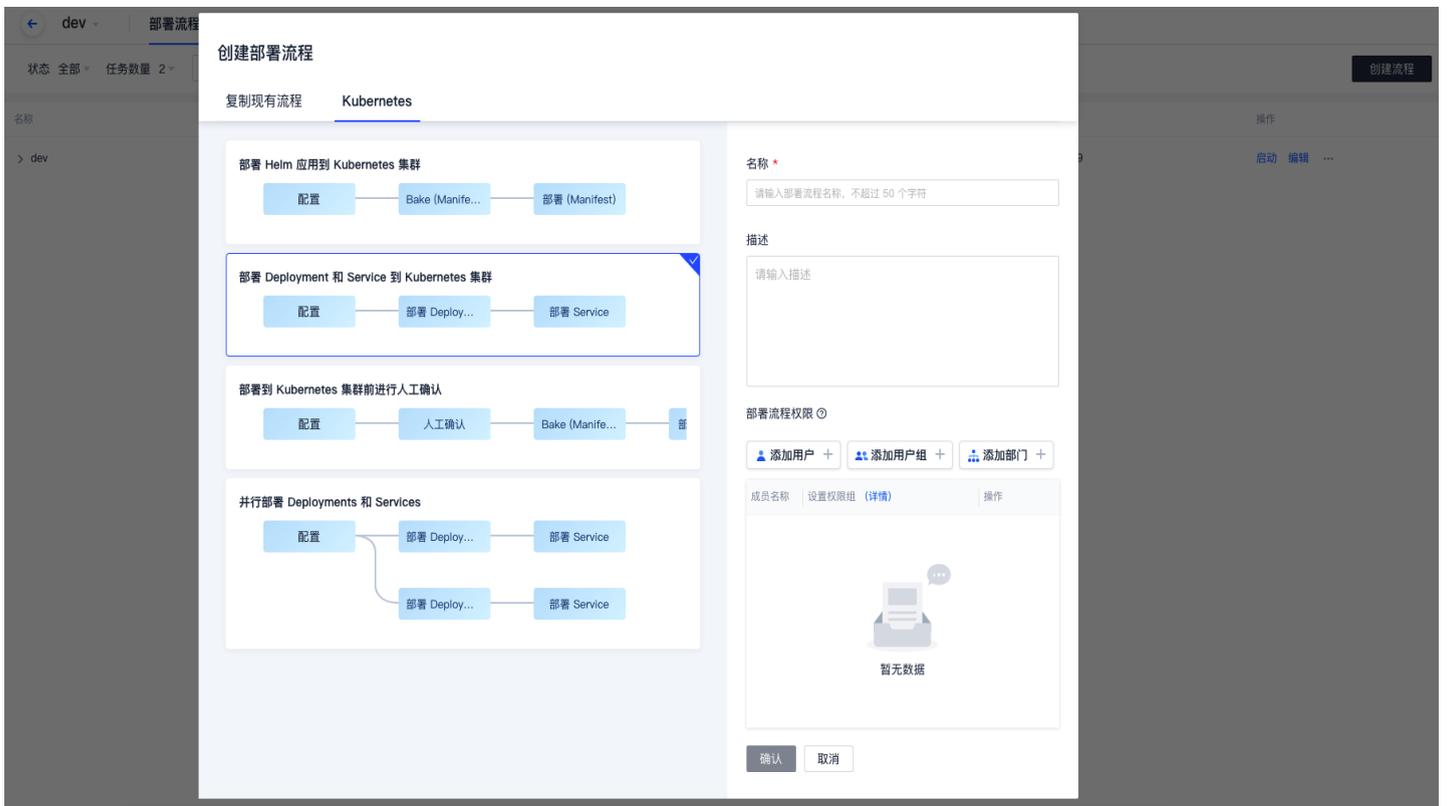


步骤2：配置应用

成功添加云账号后，在部署控制台中单击创建应用，填写应用名与选择部署方式。有关于应用的更多介绍，详细说明请参见 [应用管理](#)。



选择部署到 Kubernetes 集群模板，填写名称与描述后完成创建。



步骤3：初始化项目

此步骤主要用于配置持续部署所涉及的项目文件，默认您已完成前置准备中的导入示例代码库及创建 Docker 制品库。

首先，将本地拟发布的 Docker 制品推送至 CODING 制品仓库，具体操作可参考 [Docker](#)。

×

操作指引

- 配置凭据
- 推送**
- 拉取
- 镜像源加速 

推送

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:

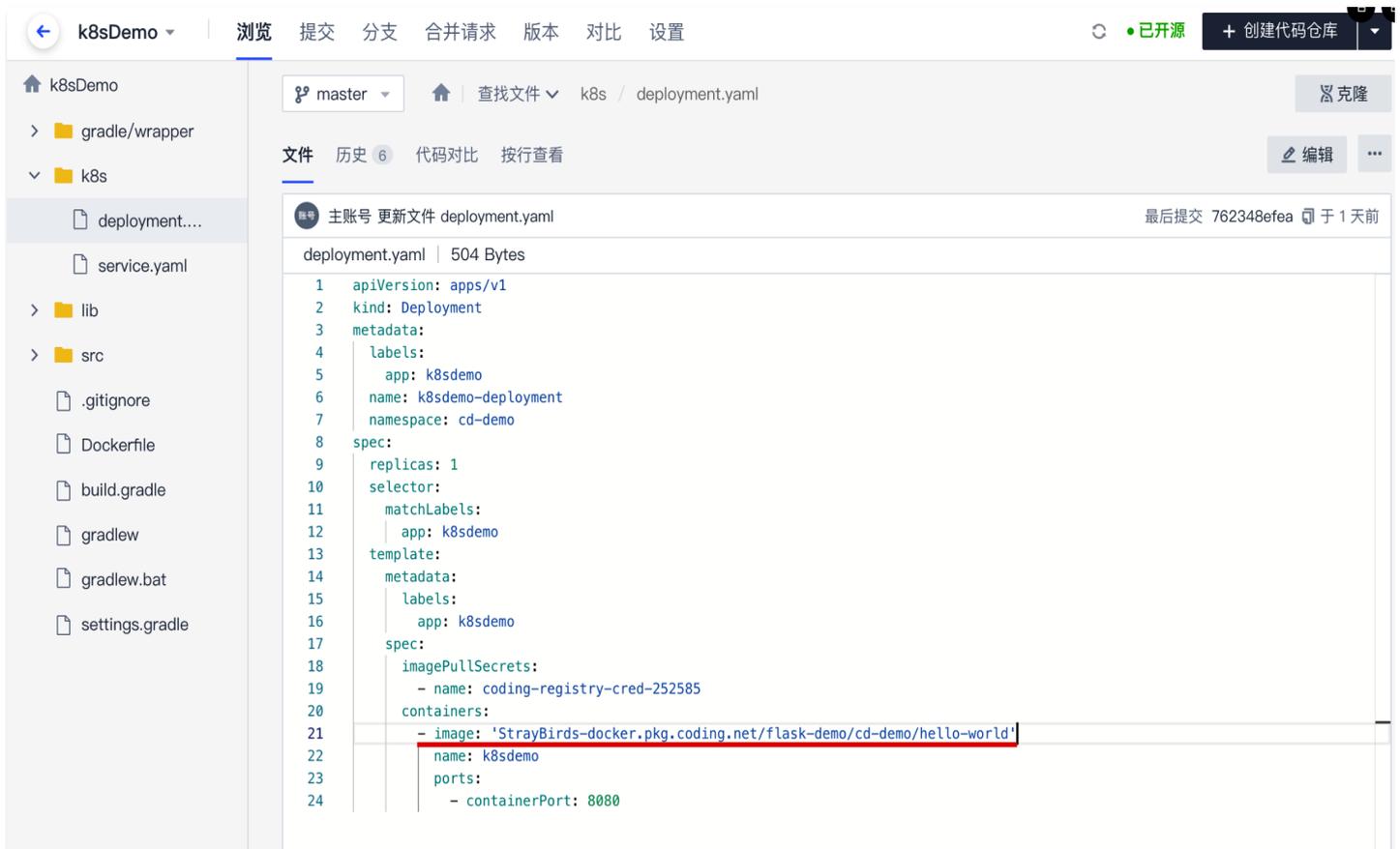
- 请在命令行执行以下命令给本地镜像打标签：

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

Copy

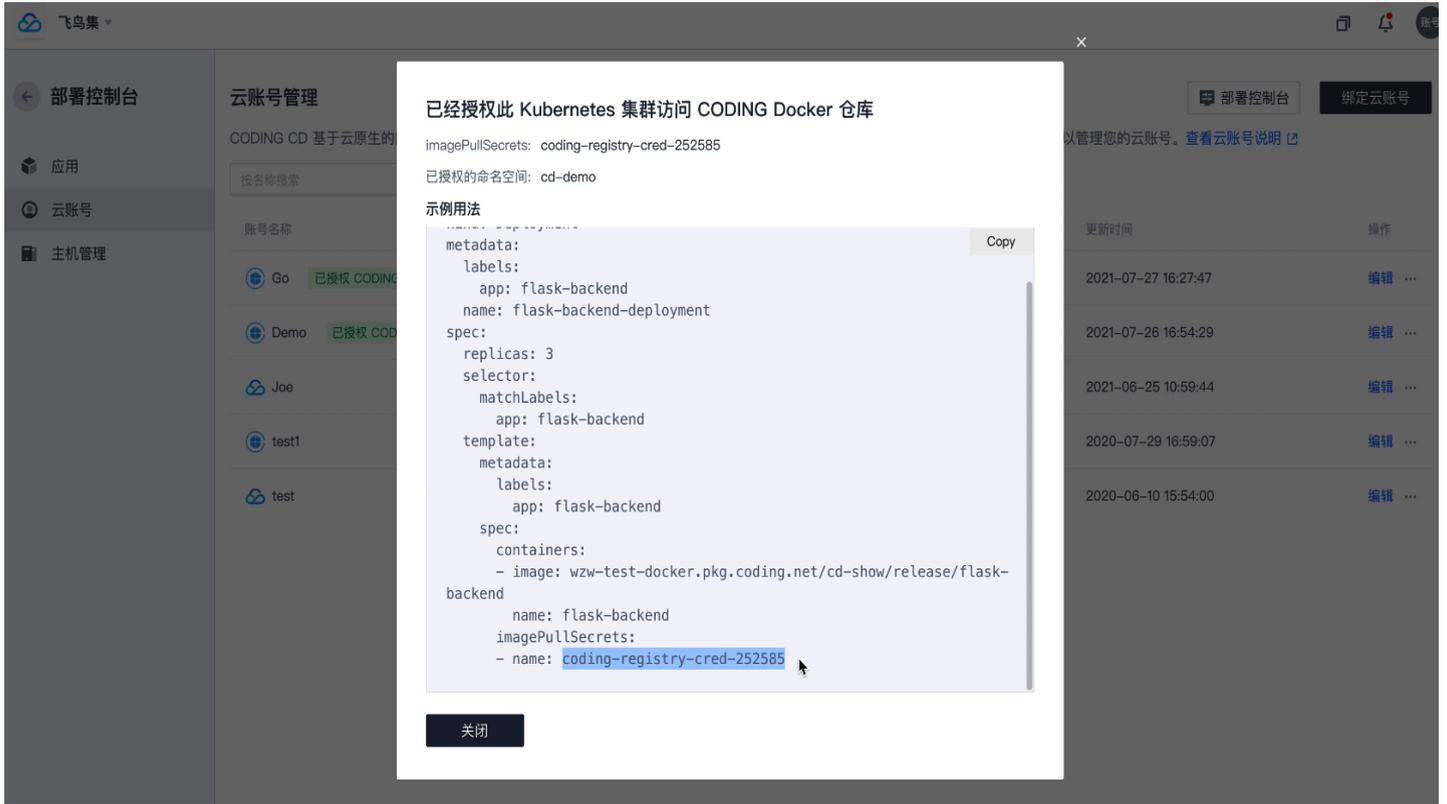
推送至制品仓库后，获取制品的拉取地址（即“操作指引” > “拉取”命令中的镜像路径）并填写至代码仓库中 `/k8s/deployment.yaml` 中的 image 地址。



The screenshot displays the 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 (504 Bytes) with the following content:

```
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
```

接下来需导入云账号的 `imagePullSecrets` 至代码仓库中。在**基础设施 > 云账号**中单击查看详情后，复制名称。



粘贴至代码仓库中的 `deployment.yaml` 文件中，同时在 `namespace` 参数一栏中填写在上文中所创建的命名空间 `cd-demo`。

← k8sDemo 浏览 提交 分支 合并请求 版本 对比 设置 ● 已开源 + 创建代码仓库

🏠 k8sDemo

- gradle/wrapper
- ▼ k8s
 - deployment...
 - service.yaml
- lib
- src
- .gitignore
- Dockerfile
- build.gradle
- gradlew
- gradlew.bat
- settings.gradle

🔍 master | 🏠 查找文件 | k8s / deployment.yaml 🔄 克隆

文件 更改对比 提交 取消

👤 主账号 更新文件 deployment.yaml 最后提交 762348efea 于 1 天前

deployment.yaml | 504 Bytes

```

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
    
```

同一层级的 service.yaml 文件中的 namespace 内容也需保持一致。

← k8sDemo 浏览 提交 分支 合并请求 版本 对比 设置 ● 已开源 + 创建代码仓库

🏠 k8sDemo

- gradle/wrapper
- ▼ k8s
 - deployment...
 - service.yaml
- lib
- src
- .gitignore
- Dockerfile
- build.gradle
- gradlew
- gradlew.bat
- settings.gradle

🔍 master | 🏠 查找文件 | k8s / service.yaml 🔄 克隆

文件 历史 4 代码对比 按行查看 编辑 ...

👤 管理员 更新文件 service.yaml 最后提交 01c504d016 于 3 个月前

service.yaml | 181 Bytes

```

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
    
```

步骤4：部署流程

在应用中心找到上述步骤 2 创建的应用，点击其名称进入部署流程配置页面，可以为此流程设定：

- 流程的执行选项（在此示例中保持默认即可）。
- 部署 Deployment 阶段以及部署 Service 阶段所需制品。
- 手动或自动触发。

首先配置部署（Manifest）阶段。基础设置选择已绑定的云账号，在 Manifest 来源选择 **CODING 代码库**，填写相应的路径。

The screenshot displays the configuration page for a '部署 Deployment' stage in a pipeline. The left sidebar shows the pipeline structure with '基础配置' and '部署 Deployment' stages. The main area shows the configuration for the '部署 Deployment' stage, with a red box highlighting the 'Manifest 来源' section. The configuration includes:

- Manifest 来源: 使用制品 (selected)
- 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).

配置部署 Service 阶段时步骤同上，但在文件路径处需选择 `k8s/service.yaml` 文件。

部署 Service

阶段类型: 部署 (Manifest)

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 ⓘ

镜像版本配置默认选择自动获取镜像来源。若设置自定义版本规则，将仅传送特定的 image 版本信息号至集群中。

▼ 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 `hello-world`

镜像地址 [?](#) `StrayBirds-docker.pkg.coding.net/demo/docker/hello-world`

高级配置

自定义版本规则 [?](#) 锁定默认版本 [?](#) 忽略版本 [?](#)

步骤5：触发器配置

完成部署阶段配置后，您可以使用自动化触发器、手动提交发布单执行部署。

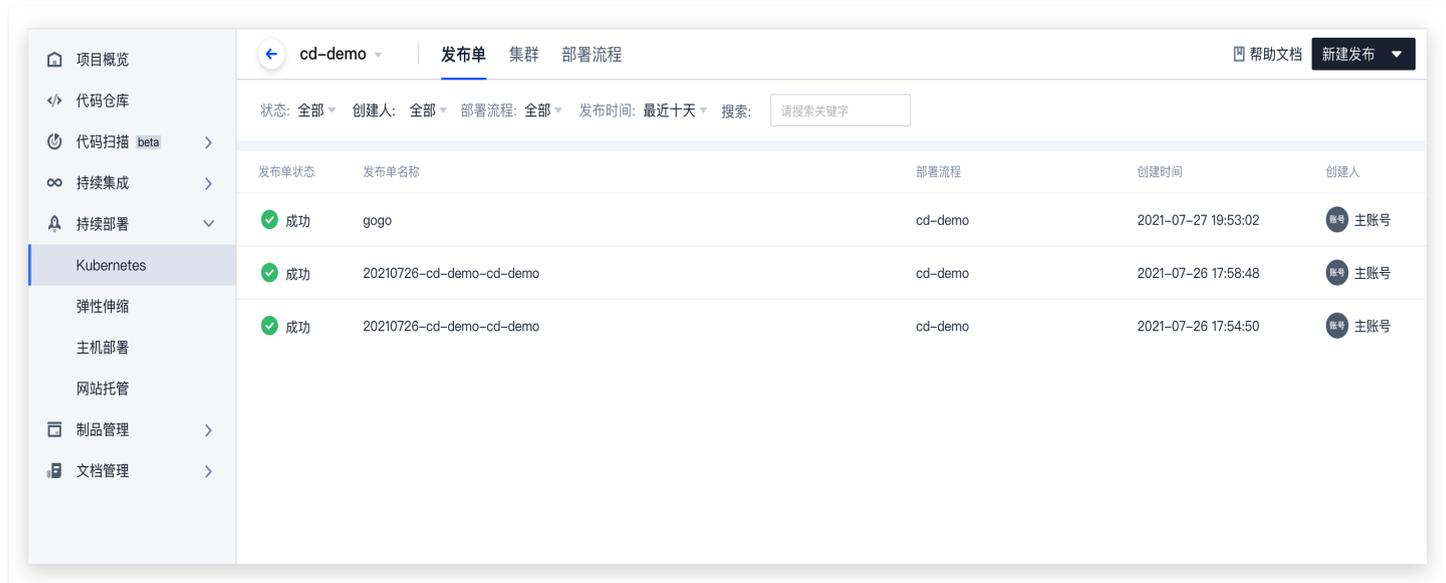
自动触发

在基础配置中点选触发器类型，选择 Docker 仓库触发器。当开发人员更新代码仓库并使用 CI 将镜像打包推送至制品库后，Docker 镜像版本的更新将自动触发部署流程并将应用发布至 Kubernetes（TKE）集群，完成后可以在应用中心查看并确认应用是否发布成功。

手动提交发布单

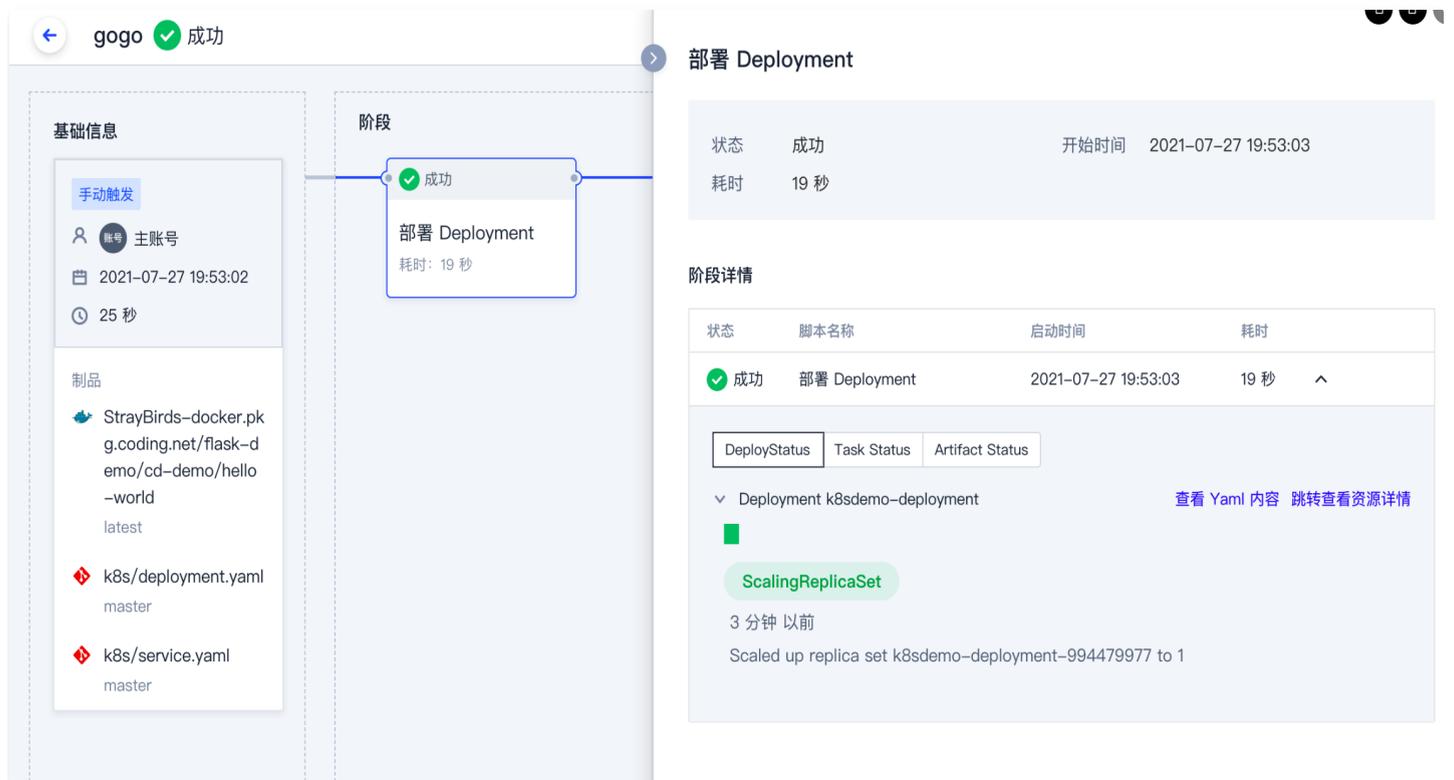
若希望通过手动提交发布单的形式触发部署流程，那么可以将应用（例如本范例的 flaskapp）与项目关联。在应用中心搜索项目名称进行关联：

关联完成后，单击项目中的**持续部署 > Kubernetes** 手动提交发布单。



步骤6：发布完成

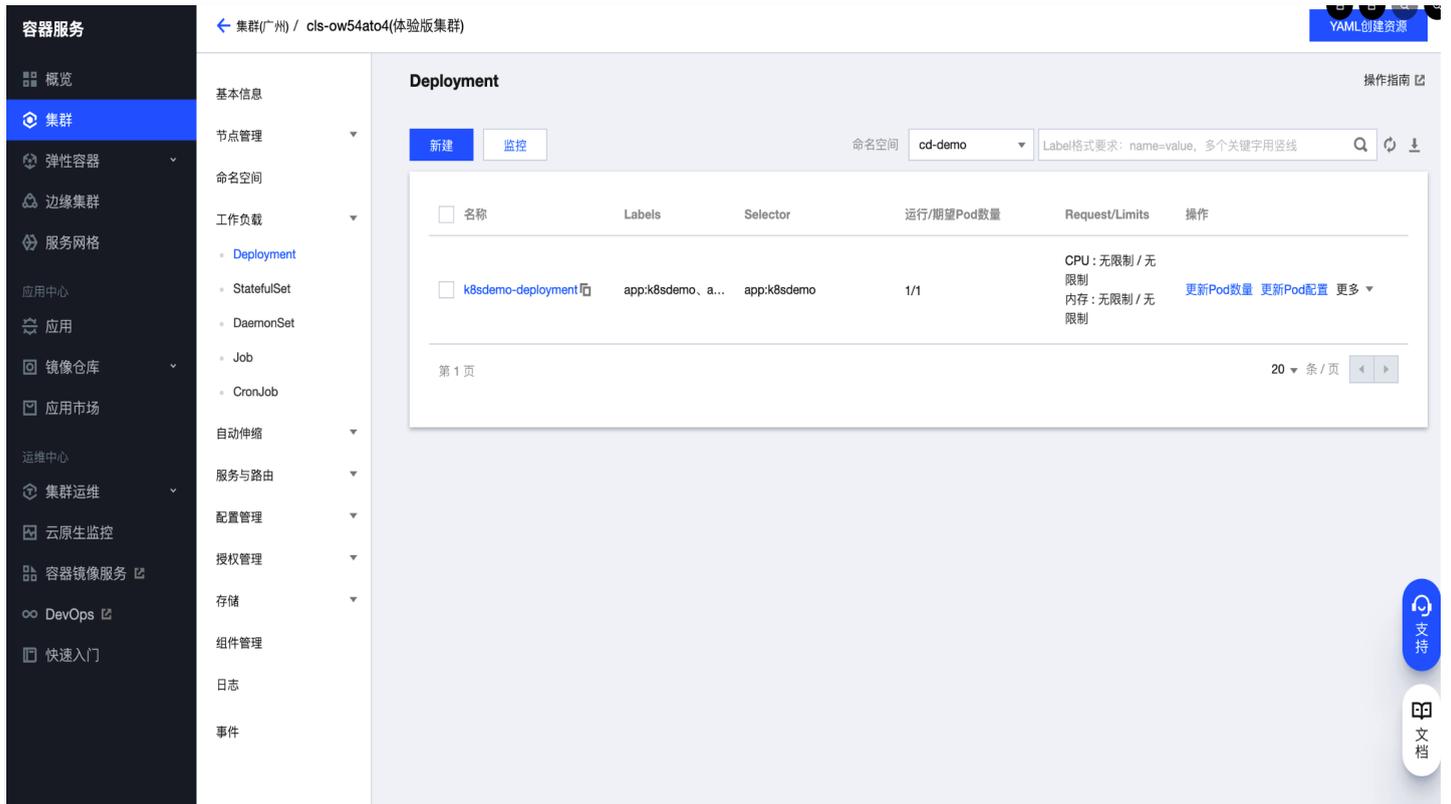
发布成功后，可以查看发布的制品及启动参数及阶段执行详情等信息。



当需要查看某个资源在集群中的运行状态时，单击集群下的工作负载即可查看详情（例如工作负载的 Pod 实例，日志等信息）。

The screenshot displays the Tencent Cloud DevOps console interface. The top navigation bar includes 'cd-demo', '发布单', '集群', and '部署流程'. Below this, there are tabs for '工作负载' (Workloads) and '服务' (Services), along with filters for '云账号: 全部', '命名空间: 全部', '类型: 全部', and '状态: 全部'. The main content area is divided into two panels. The left panel shows a table of workloads with columns for '名称', '命名空间', and '云账号'. A row is selected for 'deployment k8sdemo-deployment' in the 'cd-demo' namespace, with a 'Go' button. Below this, a specific pod instance 'V001' is shown with its image 'StrayBirds-docker.pkg.coding.net/flask-demo/cd-demo/hello-world:latest' and a 'Load E' button. The right panel provides detailed information for the selected deployment 'k8sdemo-deployment-994479977'. It includes a '操作' (Actions) dropdown menu and sections for '基本信息' (Basic Information), '镜像' (Image), '事件' (Events), and 'LABELS'. The '基本信息' section lists: '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', and '控制器: deploymentk8sdemo-deployment'. The '镜像' section shows the image name 'StrayBirds-docker.pkg.coding.net/flask-demo/cd-demo/hello-world:latest'. The '事件' section shows '1 x SuccessfulCreate' and 'Created pod: k8sdemo-deployment-994479977-nmkdx'. The 'LABELS' section lists: 'app:k8sdemo', 'app.kubernetes.io/managed-by:spinnaker', 'app.kubernetes.io/name:cd-demoteam174750', and 'pod-template-hash:994479977'.

在腾讯云的容器服务中查看工作负载。



配置自动化构建流程

若在后期需要频繁开发项目，能否仅依靠代码推送就能够自动完成发布呢？答案是肯定的。经典的持续交付 workflow 设计思路如下：

1. 配置持续集成任务，设置监听代码更新后触发。



2. 将项目代码打包成制品后，发布至制品仓库。

制品仓库 | 全部制品 | 仓库管理 创建制品仓库

- ruby**
Docker 仓库 | 项目内
- apk
Generic 仓库 | 项目内
- build
Docker 仓库 | 公开
- daily-sentence
Docker 仓库 | 项目内
- electron
Generic 仓库 | 团队内
- gwt
Generic 仓库 | 项目内

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	版本数	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. 持续部署监听到制品版本号更新后，自动发布至集群。

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

- 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

云账号

最近更新时间：2023-09-11 16:05:27

云账号是访问云资源的凭证，只有配置了云账号，持续部署控制台才能实现对云资源的部署管理和基础设施管理。目前支持三种云账号类型：

- 腾讯云 TKE：使用腾讯云容器服务中的集群。
- Kubernetes：支持 Kubeconfig 和 Service Account 两个常用凭据。
- 腾讯云账号：即腾讯云 API 密钥。

单击团队首页左侧的基础设施，进入云账号管理页，单击右上方的绑定云账号。



腾讯云 TKE

1. 云账号类型选择腾讯云 TKE，按照指引完成与云账号名下的集群绑定。若没有集群请前往 [腾讯云 TKE](#) 创建集群。



2. 选择拟部署的集群，单击**确定**后会自动验证该账号名下的集群并完成互联。

Kubernetes

Kubernetes 云账号支持 Kubeconfig 和 Service Account 两种常用凭据。以 Kubeconfig 为例：登录云计算网页控制台，复制 Kubeconfig，并将 CODING IP 段添加至集群外网访问控制列表（白名单）。

说明：

CODING 持续部署的公网 IP 段：

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



将 Kubeconfig 粘贴到云账号中，选择 Cluster Context 完成云账号添加。

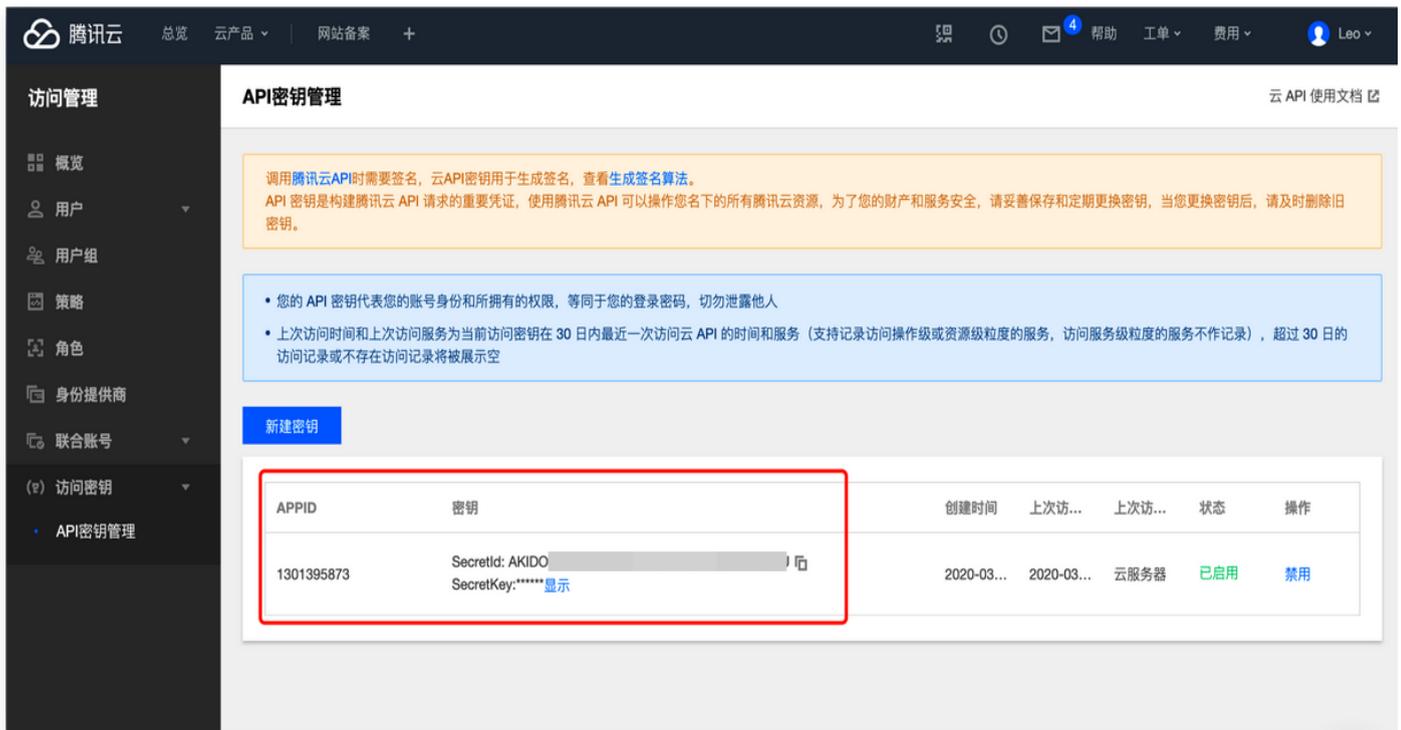


腾讯云账号

1. 云账号类型选择腾讯云账号，输入云账号名称，并选择区域。支持多选区域，CODING 持续部署将获得勾选区域的腾讯云资源管理权限。



2. 在腾讯云 [访问管理控制台](#) 页面中拷贝 API 密钥信息。



3. 将拷贝的 SecretID 和 SecretKey 粘贴到对应的文本框，单击**确定**完成云账号添加。

部署流程管理

最近更新时间：2024-08-02 15:23:01

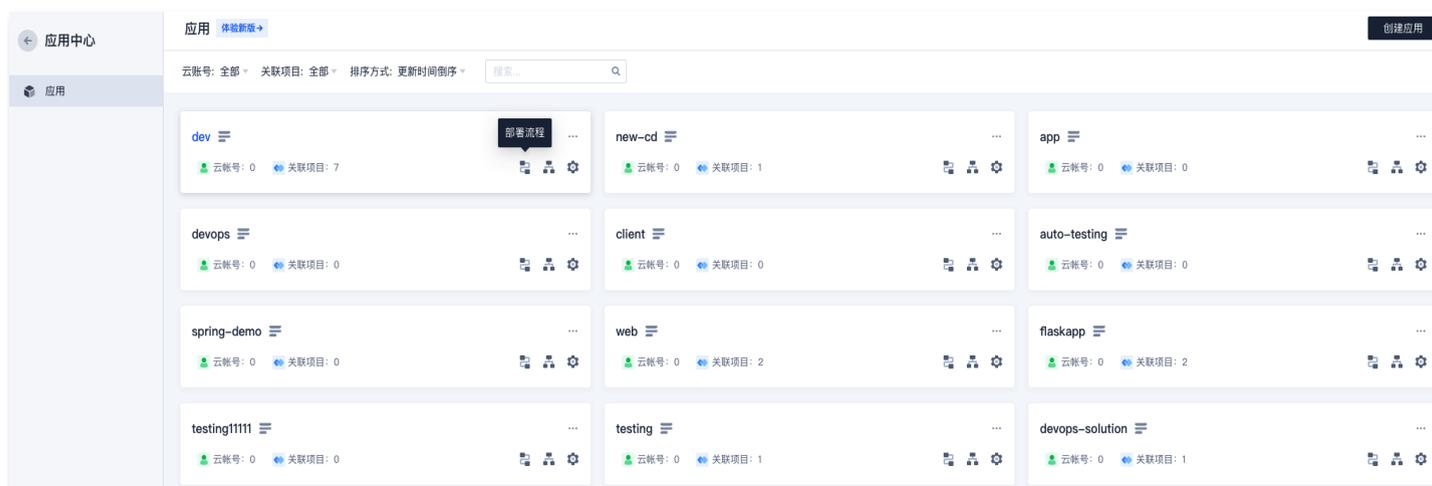
部署流程是实现持续部署最核心的模块。其优势在于支持阶段以任意的顺序组合，这样的能力让部署流程具备出色的灵活性、一致性和可重复性。

- 灵活性：支持串行、并行控制。
- 一致性：支持多种部署策略，回滚能力，确保发布结果符合预期。
- 可重复性：部署流程可重复执行，阶段可被其他部署流程复制使用。

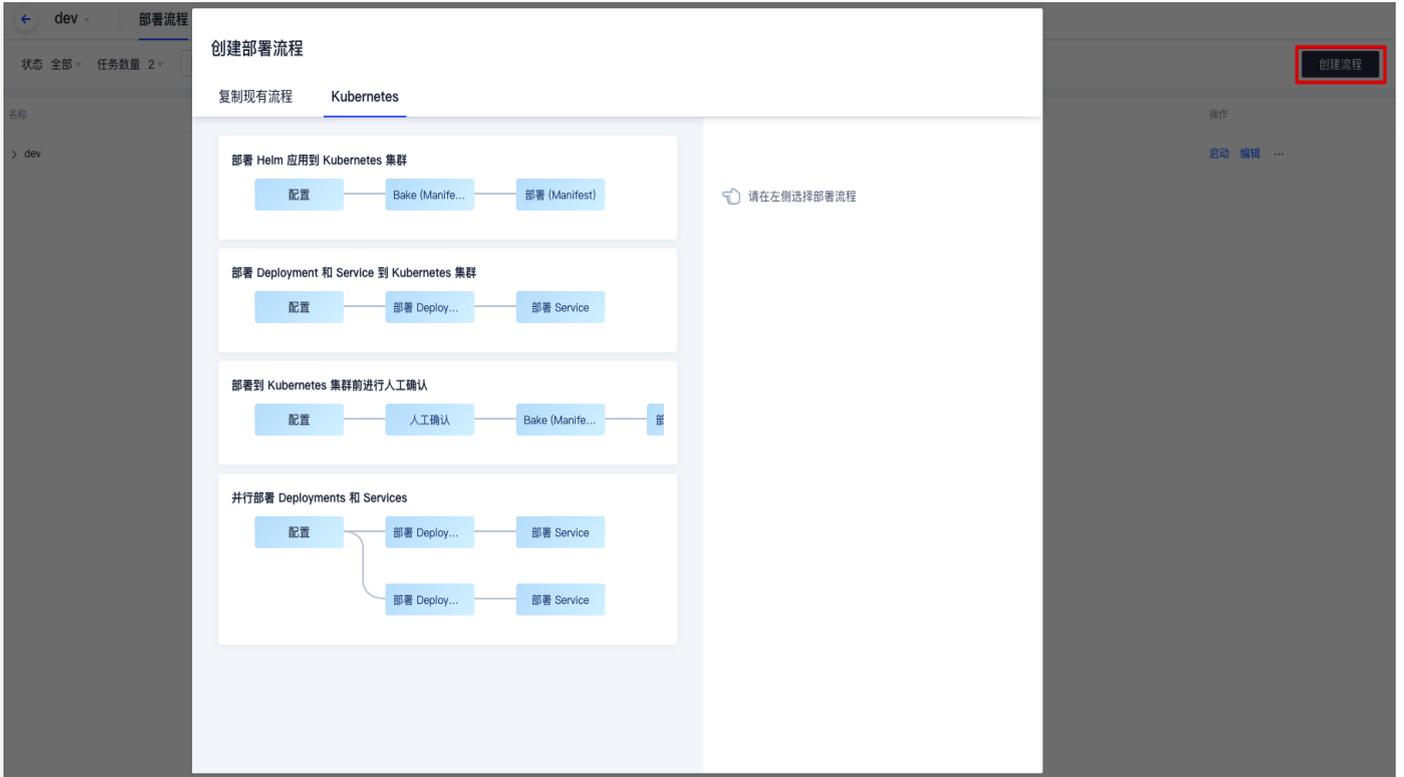
用户可以配置完全自动化的部署流程，也可以在某些阶段加入手工判断条件。此外部署流程支持多种事件的自动化触发，例如 Webhook 触发、由其他部署流程触发等。

新建部署流程

前往应用中心，单击应用卡片右下方的部署流程按钮。



1. 单击右上角的**创建流程**。



2. 您可以复制在其他应用中创建的流程，或通过空白流程自行创建。

创建部署流程

复制现有流程
Kubernetes
腾讯云弹性伸缩

部署 Helm 应用到 Kubernetes 集群

```

                    graph LR
                    A[配置] --> B[Bake (Manife...)]
                    B --> C[部署 (Manifest)]
                
```

部署 Deployment 和 Service 到 Kubernetes 集群

```

                    graph LR
                    A[配置] --> B[部署 Deploy...]
                    B --> C[部署 Service]
                
```

部署到 Kubernetes 集群前进行人工确认

```

                    graph LR
                    A[配置] --> B[人工确认]
                    B --> C[Bake (Manife...)]
                    C --> D[部署 (Manifes)]
                
```

并行部署 Deployments 和 Services

```

                    graph LR
                    A[配置] --> B[部署 Deploy...]
                    A --> C[部署 Deploy...]
                    B --> D[部署 Service]
                    C --> E[部署 Service]
                
```

👉 请在左侧选择部署流程

创建流程
操作
启动 编辑 ...

基础配置

应用的基础配置可以理解为构建整体的初始环节，既可以设置触发条件，也可以配置部署流程的通知方式等。



基础配置

制品

暂无制品
可在阶段中配置

请选择阶段
阶段类型: --

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

自动触发器

自动触发器支持 CODING Docker 制品仓库、TCR 个人版仓库触发器、Git 仓库触发器等触发条件。

添加部署流程参数

在部署流程配置页面，单击**添加参数**，即可开始填写参数。

The screenshot displays the configuration interface for a workflow stage. On the left, a workflow diagram shows a stage named '基础配置' (Basic Configuration) with a plus sign next to it, indicating the option to add a new stage. Below this, a panel titled '制品' (Artifacts) shows '暂无制品 可在阶段中配置' (No artifacts can be configured in this stage). On the right, the '基础配置' (Basic Configuration) panel is active, showing 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 selected, showing a list of start parameters. The first parameter is '启动参数' (Start Parameter) with a trash icon. Below it are input fields for '参数名' (Parameter Name), '必填参数' (Required Parameter), '参数类型' (Parameter Type), '默认值' (Default Value), and '描述信息' (Description Information). A '+ 添加启动参数' (Add Start Parameter) button is at the bottom.

添加阶段

在部署流程配置页面单击加号即可添加新的阶段，右侧列表中支持选择阶段类型。

← 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	<input type="button" value="选择"/>
	过滤 (Manifest) 过滤 (Manifest)	<input type="button" value="选择"/>
	扩缩容 (Manifest) 对 Kubernetes 对象执行扩缩容	<input type="button" value="选择"/>
	回滚 (Manifest) 回滚至目标版本	<input type="button" value="选择"/>
	删除 (Manifest) 删除 Kubernetes (Manifest)	<input type="button" value="选择"/>
	Bake (Manifest) 使用 Helm Bake manifest 文件	<input type="button" value="选择"/>
	Patch (Manifest) Patch a Kubernetes object in place.	<input type="button" value="选择"/>

执行部署流程

部署流程配置完成后，您可以通过设置好的触发器提交自动执行，或在持续部署中提交发布单手动触发部署流程。

- 🏠 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 ▾

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

部署流程配置

部署流程支持删除、禁用、锁定、查看历史版本与编辑 JSON 配置。

← 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

删除部署流程

设置后，将删除此部署流程。

禁用部署流程

设置后，将禁止任意触发器启动部署流程，包括手动触发。可以选择在团队内整体禁用或仅在项目内禁用。



锁定部署流程

锁定部署流程后，不能在应用中心对部署流程进行任何修改。



查看修订历史

保存新的部署流程配置后，旧版本将会添加到修订历史。您可以在修订历史页对比各版本信息，选择并还原到任意历史版本。

修订历史

修订版本 2021-06-16 10:25:41 (当前版本) 对比 上一版本

2021-06-16 10:25:41 (当前版本)

2021-06-15 15:02:05

```
{
  "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

编辑 JSON 配置

在部署控制台中所做的任何更改最终都会以 JSON 格式文件保存，直接编辑部署流程的 JSON 内容可以为部署流程添加新属性或自定义 UI 界面尚未显示的配置项。

注意：

此种方式允许用户将在文本框内自由编辑部署流程，但有可能会破坏部署流程的可用性，我们提供了从修订历史中恢复到任意指定版本的能力。

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  }
```

取消

保存变更

部署方式

自动发布 Docker 制品时触发

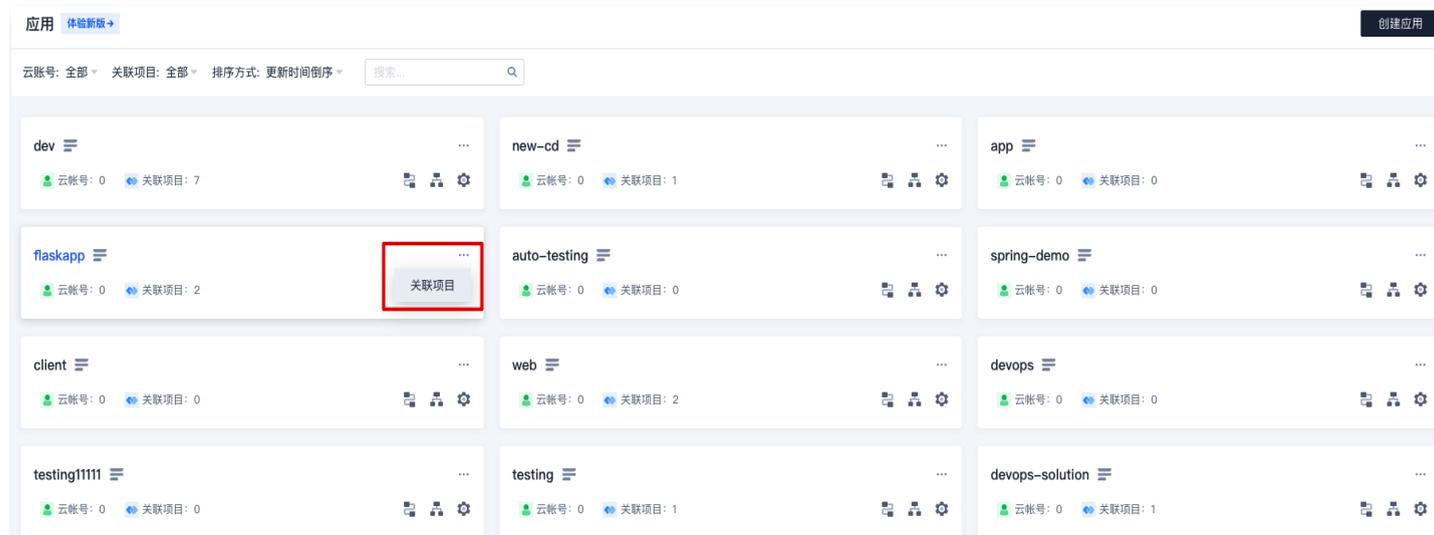
最近更新时间：2023-09-11 16:05:28

CODING 持续部署的一大优势在于能够便捷的集成上下游产品为 workflow，下文将演示如何通过三个步骤实现**持续集成任务推送制品 > 制品仓库镜像更新 > 触发部署流程**这一基础的自动化流水线配置。

操作步骤

步骤1：应用与项目关联

部署流程控制台中的**应用**需提前与项目相关联。前往应用中心，单击应用中的**关联项目**图标，选择持续集成配置所在的项目并进行关联。



步骤2：配置持续集成

此步骤使用持续集成将制品推送至制品仓库。您可以通过持续集成计划模板创建，或直接编写 Jenkinsfile 手动增加此阶段。

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

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。

在持续集成流程中手动增加此阶段：

flask-docker | 基础信息 **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

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 参考：

```

stage('推送到 CODING Docker 制品库') {
  steps {
    script {
      docker.withRegistry(
    
```

```

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

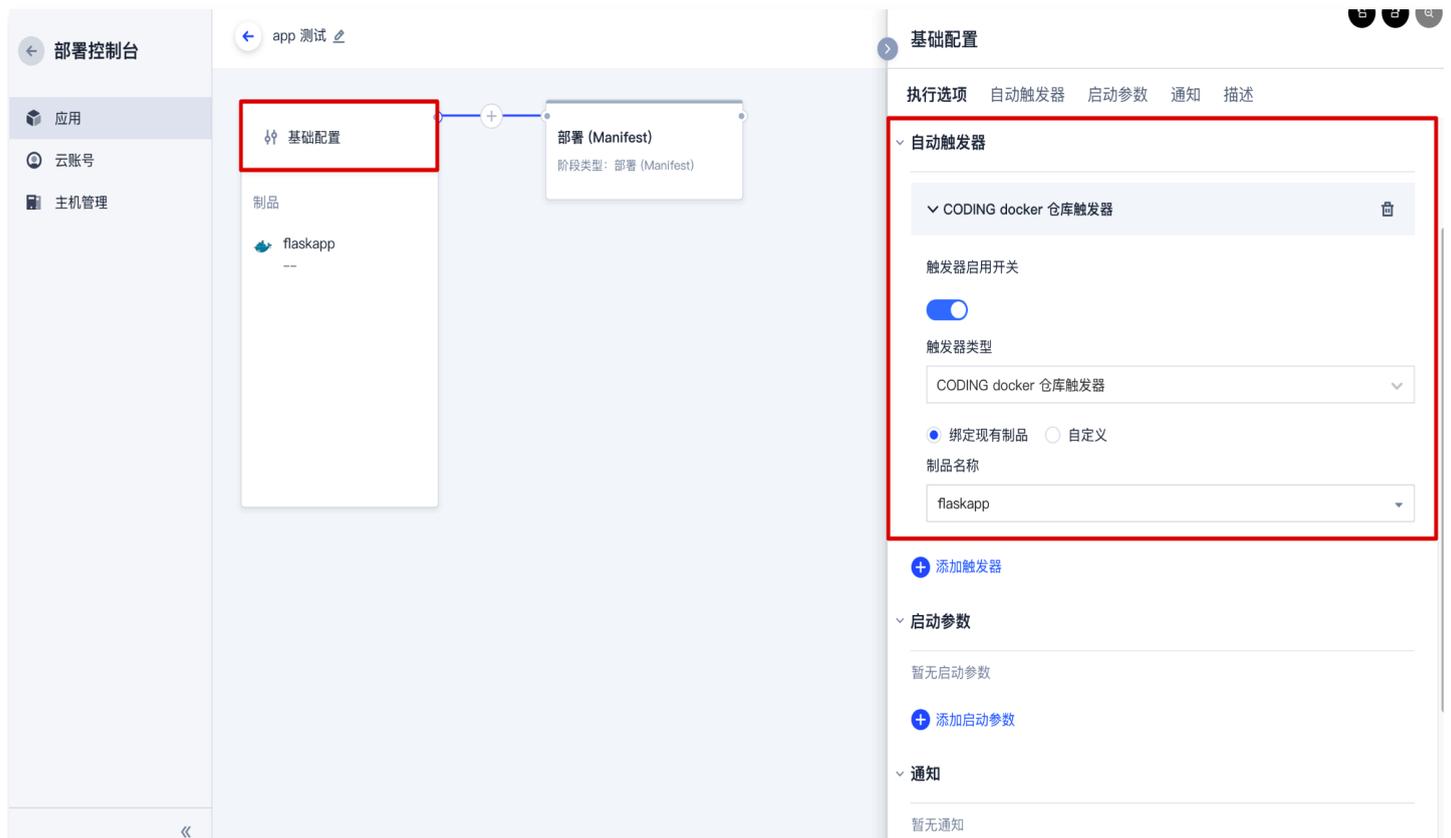
  docker.image("${CODING_DOCKER_IMAGE_NAME}:${env.DOCKER_IMAGE_VERSION}").
  push()
  }
}
}

```

步骤3: 根据制品镜像版本触发

前往持续部署中的应用部署流程，单击**基础配置**中的触发器启用开关。此处选择通过 CODING Docker 制品更新触发，将监听关联项目中制品版本号。若持续集成将制品推送至制品仓库时，将自动触发部署流程；选择**自定义**能够监听其他项目的制品仓库更新情况。

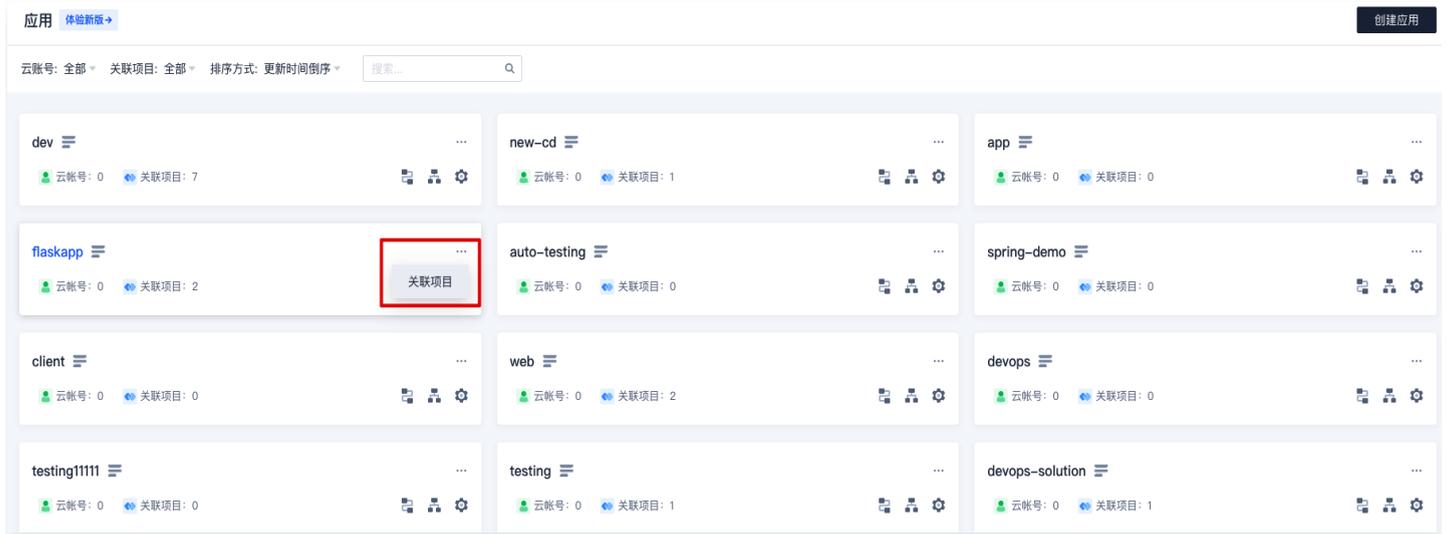
除了通过 CODING Docker 制品更新触发，您还可以通过 Git 仓库或定时器触发此部署流程。



在构建计划中添加部署阶段

最近更新时间：2023-09-11 16:05:28

在持续集成中触发部署时请提前前往应用中心，将应用与项目关联。



本文给出了两种设置方法，您可以按照需求有选择性阅读。

- 直接使用构建计划模板
- 在已有构建计划中添加部署阶段

使用构建计划模板

请在应用中心关联涉及到相关集群的云账号，详情请参见 [云账号](#)。

单击项目内左侧产品栏**持续集成**，右上角创建构建计划，选择部署分类下的**推送到 Kubernetes** 模板。

← 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

 **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。

按照模板提示选择相应的制品仓库、远端集群地址等信息，完成后勾选创建后触发构建。

2 构建 Docker 镜像

Docker 镜像名称 *

Dockerfile 文件位置 *

Docker 构建目录 *

Docker 镜像版本 *

3 推送到 CODING Docker 制品库

Docker 制品库 *

4 部署到远端 Kubernetes 集群

集群 *

命名空间 *

```
},
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}",
    ])
  }
}
```

设置完成后，运行持续构建计划即可完成自动发布。

添加部署阶段

在此方法中您可以在构建计划 > 流程配置中使用编辑器或填写命令添加部署阶段。

图形化编辑器

在已有构建计划设置中添加部署阶段，填写镜像地址、集群与命名空间等关键信息。

flask-docker | 基础信息 | **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

1 构建 Docker 镜像

5-1 推送到 CODING D...

镜像更新

快速筛选

- 命令
- 代码管理
- 文件操作
- 发布部署
- 收集报告
- 流程控制
- 安全
- 持续部署
 - 全部
 - 官方插件
 - 团队插件
 - 镜像更新
- 质量管理
- 其他
- 腾讯云插件
- 编译
- 消息通知

镜像更新

插件配置 | 高级配置

将上游构建的 Docker 镜像部署到 Kubernetes 集群。 [查看完整帮助文档](#)

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 参考:

```

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}",  
  ])  
}  
}
```

手动提交发布单

最近更新时间：2025-04-14 17:03:52

新建发布单

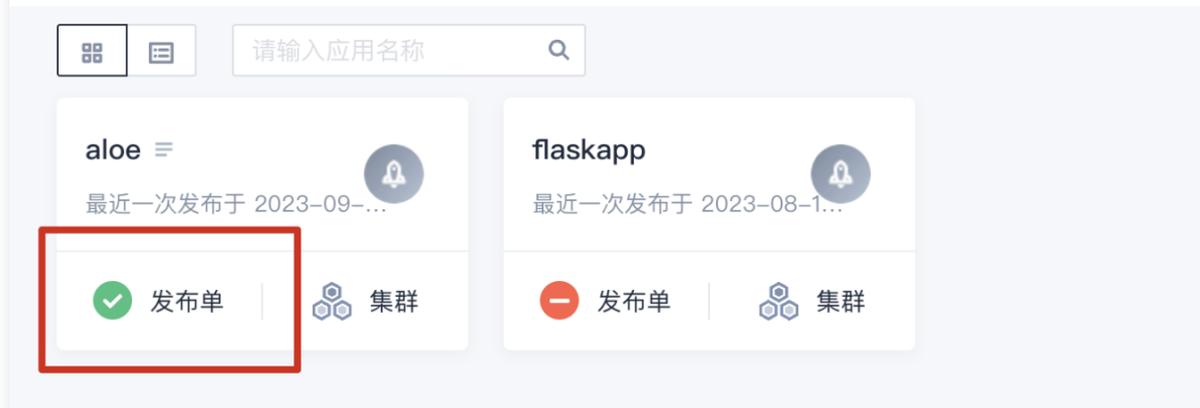
在应用的部署流程中添加人工确认步骤，确保通过发布单发布时是经二次确认的，通过权限控制把控发布质量。

The screenshot displays the 'Manual Release' configuration interface. On the left, a workflow diagram shows a '部署 Service' stage (Manifest type) followed by a '请选择阶段' stage. The right pane, titled '选择阶段', lists several stage options under the '通用类型' (General Type) tab. The '人工确认' (Manual Confirmation) option is highlighted with a red border, indicating it is the selected step for manual confirmation. Other options include '预置条件检查' (Pre-check), '部署流程' (Deployment Process), '自定义变量' (Custom Variables), '绑定部署流程制品' (Bind Deployment Process Artifacts), '等待' (Wait), and 'Entity Tags'.

1. 要使用发布单功能，必须将 [应用与项目关联](#)。
2. 关联完成后，进入项目中[持续部署](#)找到该应用，单击[发布单](#)进入发布单管理页面。

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](#)



3. 单击新建发布单后，可以运行已有应用及部署流程。



快速发布

若不希望团队设置复杂的权限限制，需要直接体验持续部署功能，可以使用快速发布功能。您无需在控制台中配置部署流程即可将镜像发布至集群中，适用于部署流程更加灵活复杂的场景，例如说临时镜像变更等突发场景，无需快速将制品发布至集群中。

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。
 - [Tencent TCR 企业版说明](#)
 - [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

发布完成后可以在持续部署中查看发布详情。

cd-demo-快速发布 ✔ 成功

Patch (Manifest)

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

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