持续部署

操作指南

产品文档

腾讯云

文档目录

操作指南
权限控制

最近更新时间：2024-01-03 11:39:45

本文为您详细介绍在 CODING 持续部署中的权限控制。

前提条件

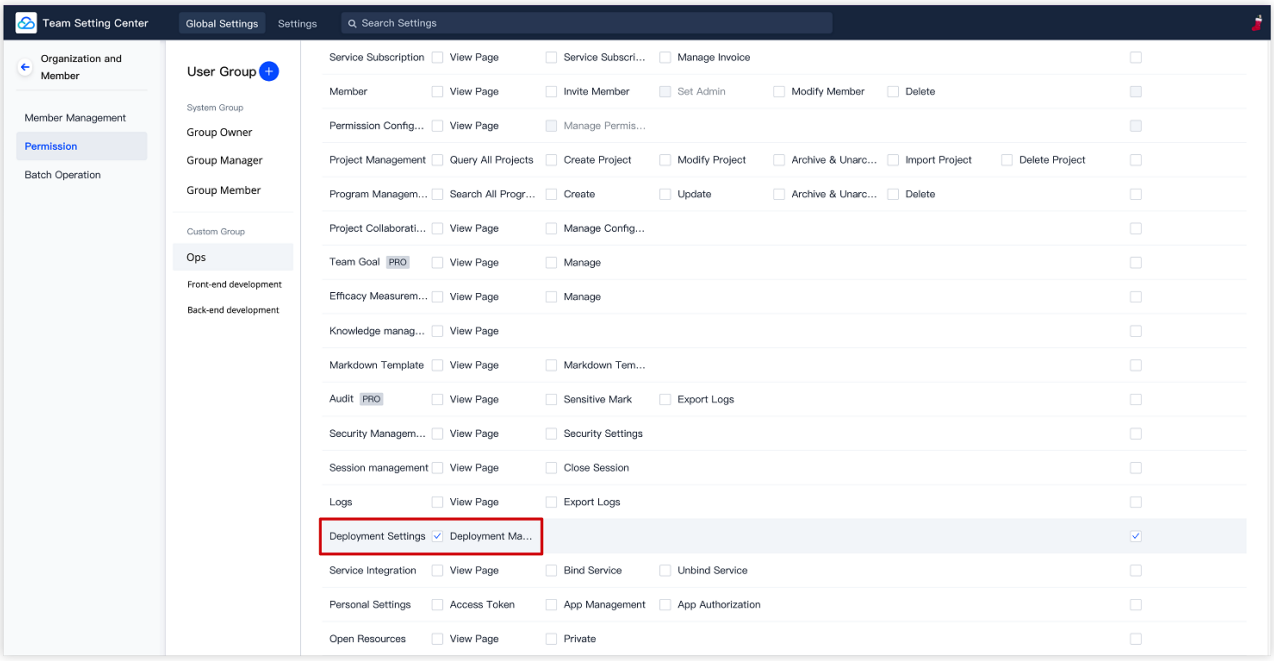
使用 CODING 项目管理的前提是，您的腾讯云账号需要开通 CODING DevOps 服务。

进入管理设置

1. 登录 CODING 控制台，单击团队域名进入 CODING 使用页面。
2. 鼠标移至右上角头像后会出现下拉菜单，进入团队管理界面，单击权限配置进入管理界面。

权限控制

在默认情况下，CODING 持续部署的权限控制：
团队所有者：具备部署管理权限
团队管理员：具备部署管理权限
团队普通成员：不具备部署管理权限
CODING 推荐团队中建立独立的**运维**角色，并赋予**部署管理**的权限，并且指定团队成员担任运维角色，以实现最小权限原则。如下图所示，设置一个**运维**角色，可单击**添加用户组**设置更多角色。



推荐在 DevOps 实践中，将持续部署过程的操作简单地划分为两类角色：

运维：配置持续部署过程（配置应用、配置发布流程、配置审批流程）

开发：提交发布单进行发布（提交发布单、等待审批完成、查看发布过程）

部署控制台

最近更新时间：2024-01-03 11:40:20

本文为您详细介绍 CODING 持续部署中的控制台。

前提条件

使用 CODING 项目管理的前提是，您的腾讯云账号需要开通 CODING DevOps 服务。

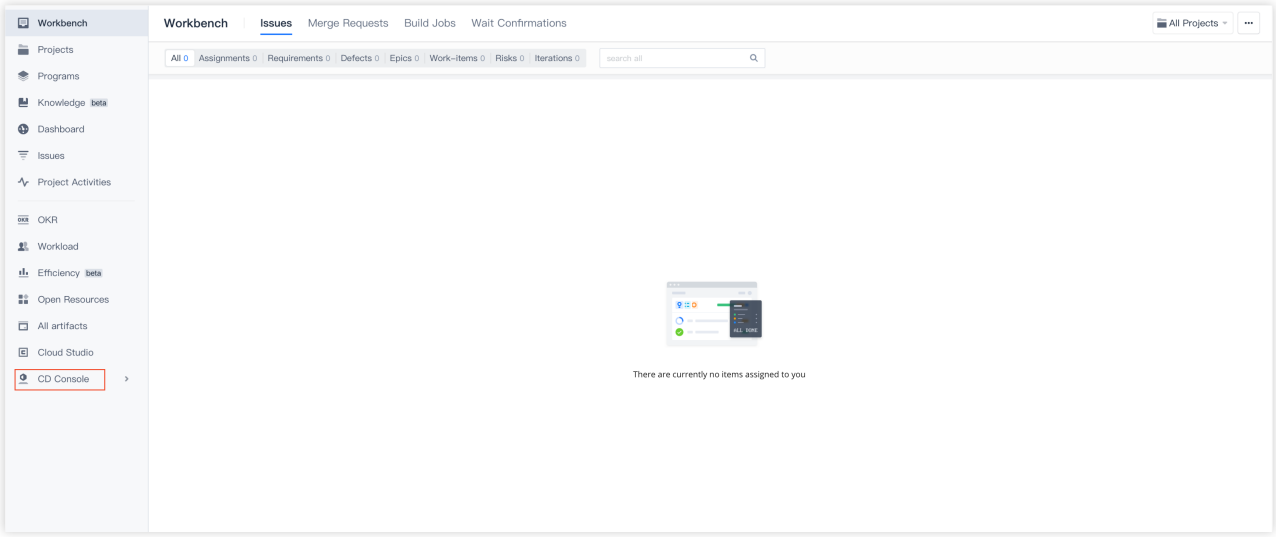
进入项目

1. 登录 CODING 控制台，单击**立即使用**进入 CODING 使用页面。
2. 单击工作台首页左侧的

，进入持续部署控制台。

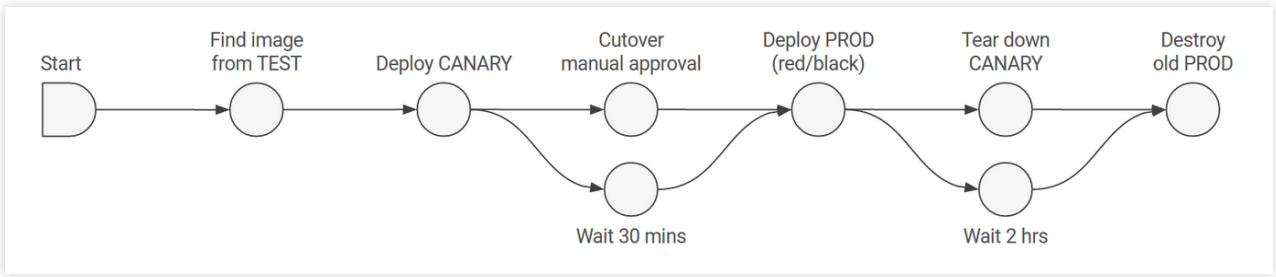
功能介绍

CODING 持续部署控制台基于 Spinnaker 实现，是一个能够管理应用与云账号的综合控制中心。控制台中运维类角色可以在控制台中管理待部署的应用列表，配置部署流程，查看和管理应用集群，对集群进行一些点对点的操作（扩缩容，停止，回退等）。
您可以在 CODING 团队首页中快速进入部署控制台。



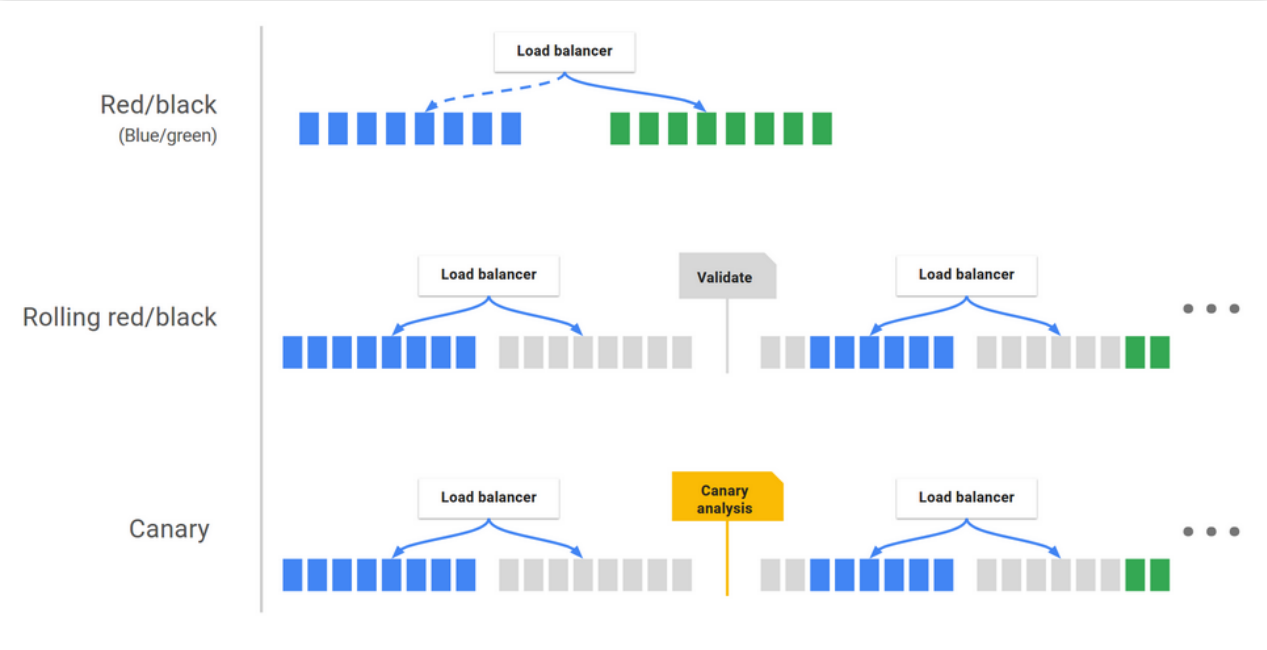
部署流程

部署流程由一系列的**阶段**组成，能够将持续部署流水线化。部署流程可以手动触发执行，也支持配置自动触发，例如由 CODING Docker 仓库触发、Webhook 触发、定时触发等。此外，可以配置引用制品、参数、通知和串行并行逻辑。**阶段**是持续部署流程里的一个自动构建模块，您可以在部署流程中定义各个阶段的执行顺序，以实现灵活的自动化部署。CODING 持续部署提供了很多阶段模板供您选择使用，例如人工确认、前置条件检查、Deploy（部署）等。



部署策略

CODING 持续部署支持精细化的部署策略，例如红/黑（蓝/绿）部署、滚动红/黑策略和灰度部署等。用户可以为每个环境使用不同的部署策略，可以在测试环境中使用红/黑策略，生产环境里使用滚动红/黑策略。在部署策略中已经对必要的步骤进行封装，不需要复杂操作就可以实现企业级发布。



基础设施管理

CODING 持续部署基于 Spinnaker CloudDriver 组件开发，能够兼容适配不同的云平台，实现高效云资源管理。基础设施包含如下几块：

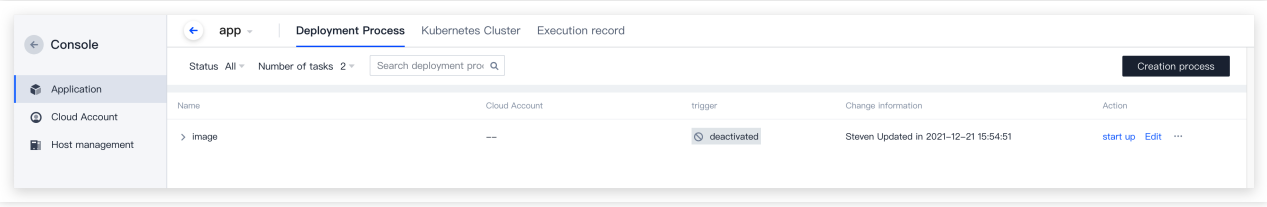
服务组：服务组是最基本的资源管理单元，用于标识可部署的制品（如：VM 镜像、Docker 镜像）以及实例数量、自动伸缩策略、元数据等可配置项。服务组还可能关联负载均衡器和安全组。当部署完成后，服务组就相当于一组运行中的软件实例集合（如：腾讯云弹性伸缩组、Kubernetes pods）。

负载均衡器：用于将外部网络流量重定向到服务组中的运行实例，支持指定一系列规则对运行实例做健康检测。

安全组：定义了网络访问权限，由 IP、端口和通信协议组成安全组规则。

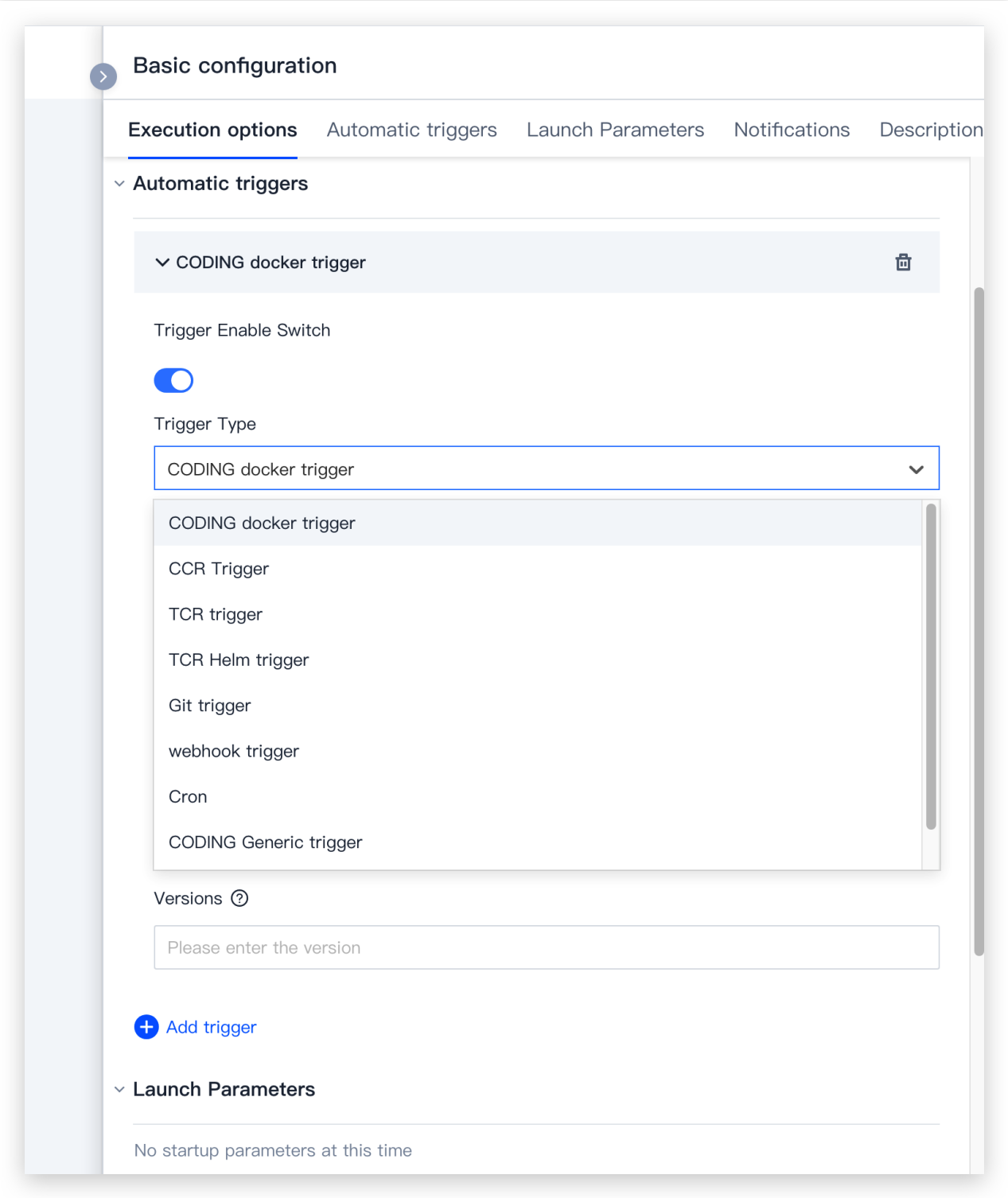
集群：由用户定义的，对服务组的逻辑分组。

应用：CODING 持续部署以应用作为基本部署单位。应用包含若干个应用集群、负载均衡器和安全组等。应用通常代表您想要部署的服务、配置、以及运行所需的基础设置。推荐将一个应用对应至微服务架构中的一个服务。



触发器

在保留 Spinnaker 部分原生触发器类型的基础上， CODING 部署控制台扩充了触发器类型，使之能够与 CODING 上游制品库匹配。



制品类型改造

在保留 Spinnaker 部分原生制品类型的基础上，CODING 部署控制台在 **Git 仓库文件**制品类型中扩充了对 CODING 代码库的支持，在 **Docker 镜像**制品类型中扩充了对 CODING Docker 镜像制品的支持，已支持 War 包、Helm 包等更多的制品类型。

名词解释

实例：运行中的容器或 VM 实例。
Stack：由用户自定义的，对集群的逻辑分组，如 `prod` ，`staging` ，`test` 。
Detail：由用户自定义的，用于标识集群的三级字段。例如具有相同 `\\${Application}-\\${Stack}-${Detail}` 属性的服务组属于同一个集群。

云账号

最近更新时间：2024-01-03 11:41:06

本文为您详细介绍 CODING 持续部署中的云账号。

前提条件

使用 CODING 项目管理的前提是，您的腾讯云账号需要开通 CODING DevOps 服务。

进入项目

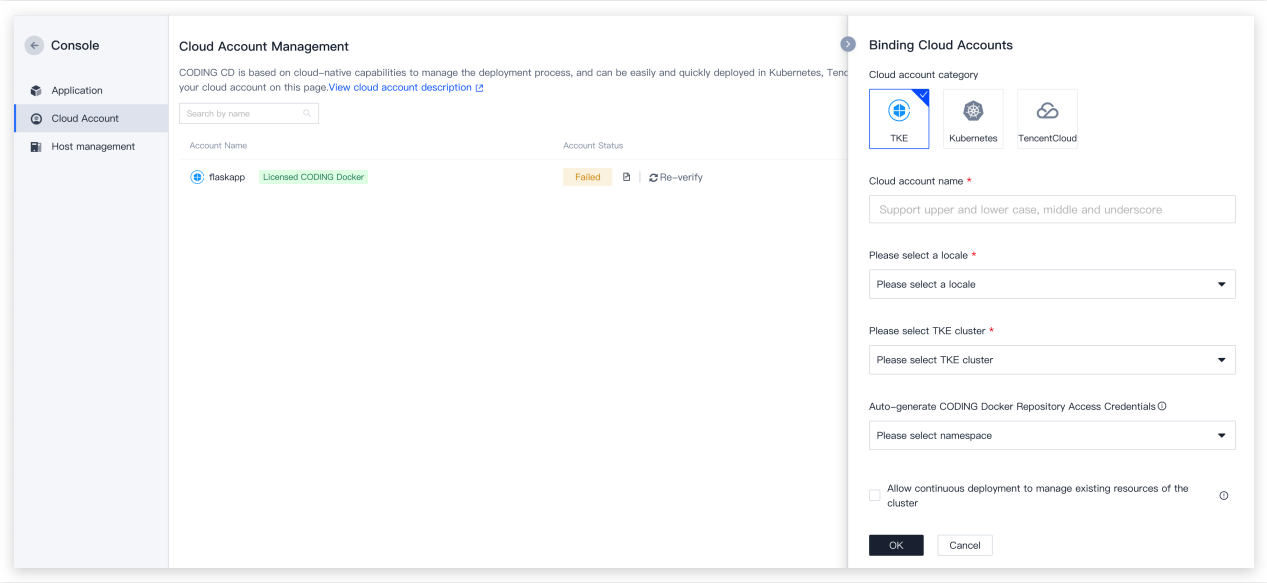
1. 登录 CODING 控制台，单击团队域名进入 CODING 使用页面。
2. 进入 CODING 工作台首页后，选择左侧**功能设置** > **持续部署**，进入云账号管理页面。

功能介绍

云账号是访问云资源的凭证，只有配置了云账号，CODING 持续部署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：如果您是从腾讯云开发者平台入口注册登录，才会显示此类账号。
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。
您可以在 CODING 首页工作台中的**功能设置** > **持续部署** > **云账号**页面管理云账号。

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。

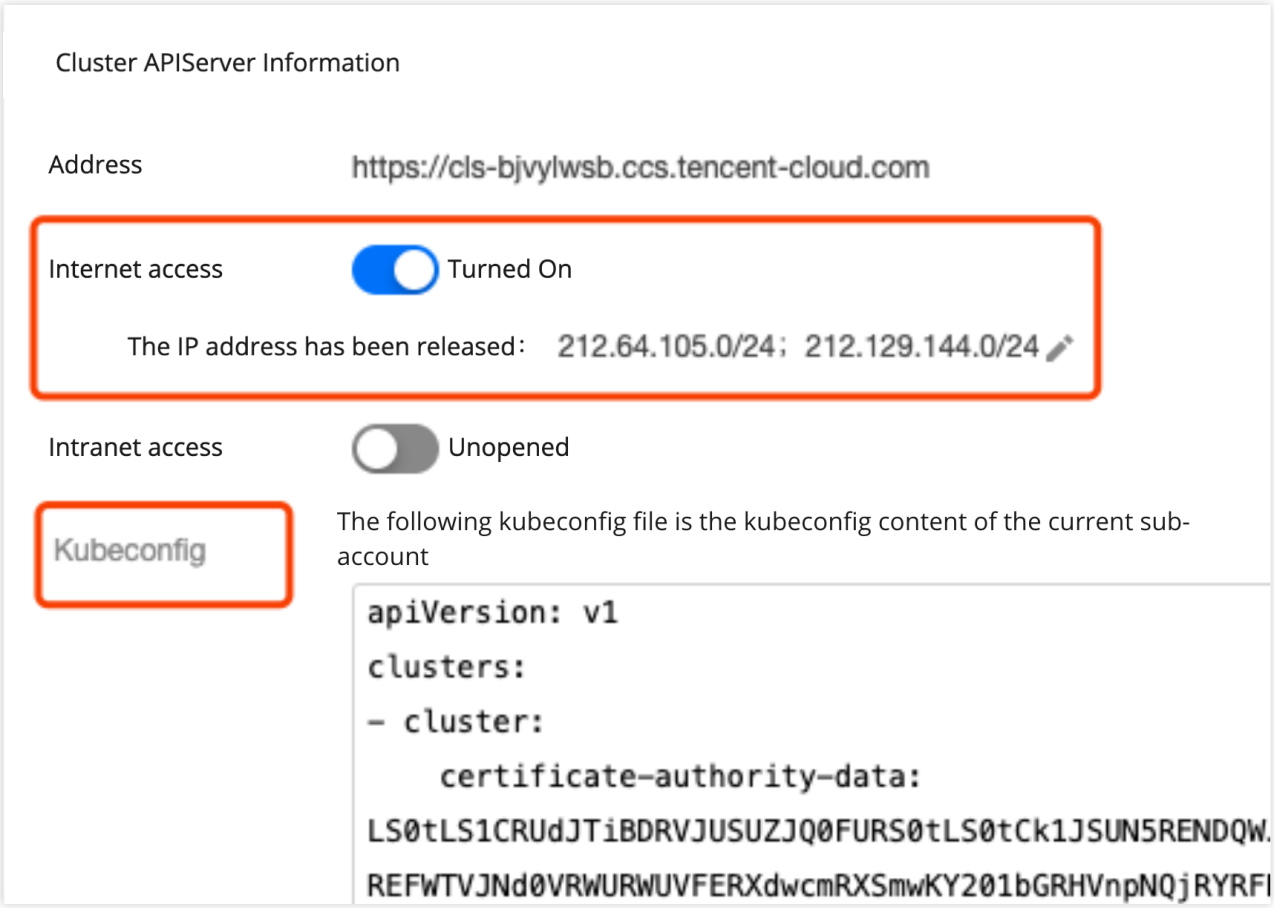
Kubernetes

Kubernetes 云账号支持 Kubeconfig 和 Service Account 两种常用凭据。以 Kubeconfig 为例：

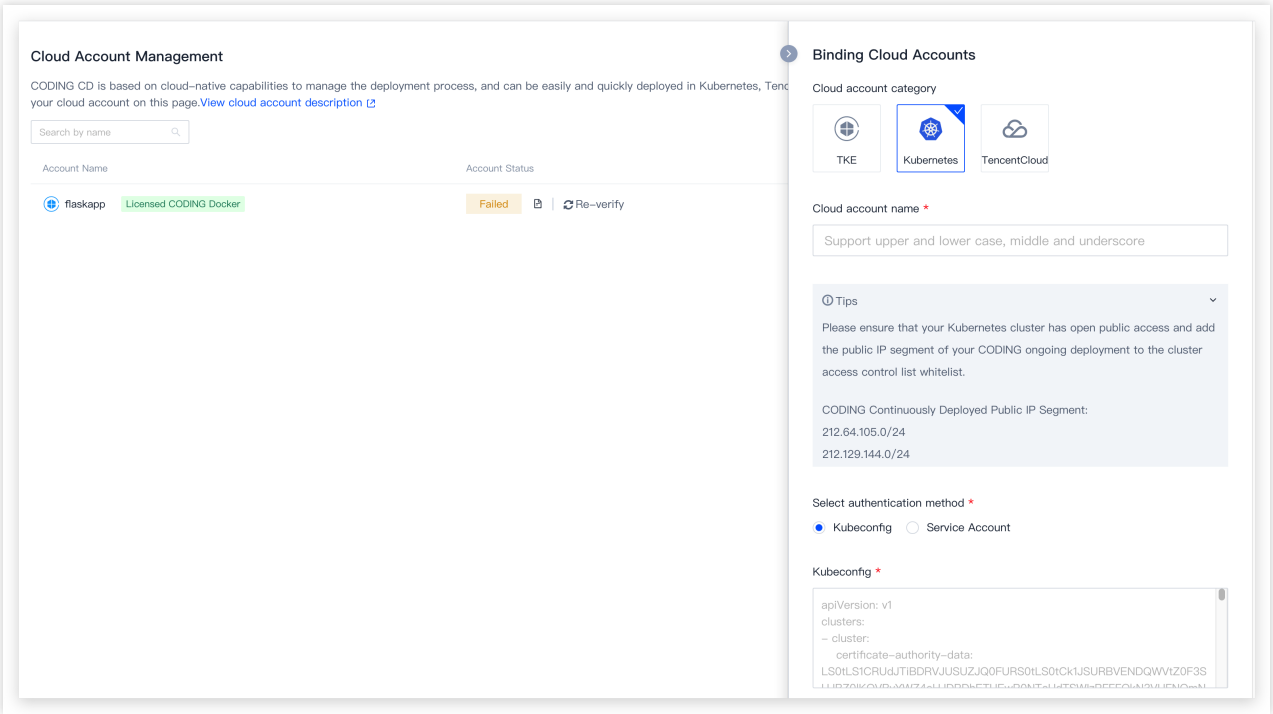
登录云计算网页控制台，复制 Kubeconfig，并将 CODING IP 段添加至集群外网访问控制列表（白名单）。

CODING 持续部署的公网 IP 段：

212.64.105.0/24、212.129.144.0/24

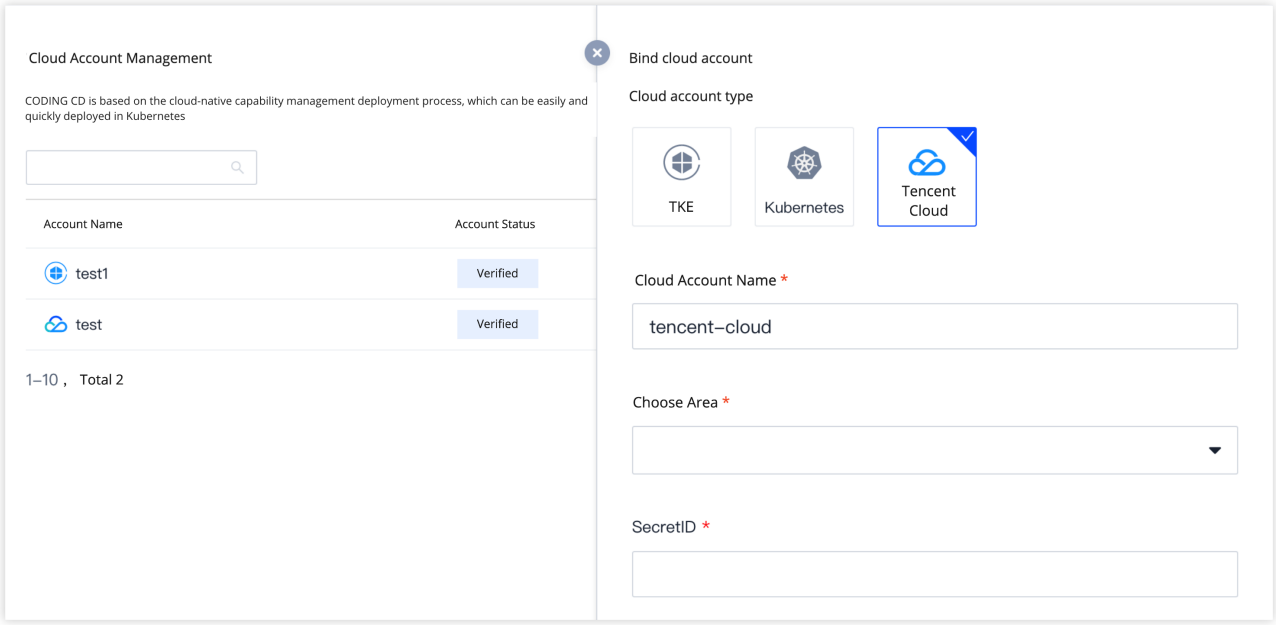


将 Kubeconfig 粘贴到 CODING 中，选择 Cluster Context，完成云账号添加。

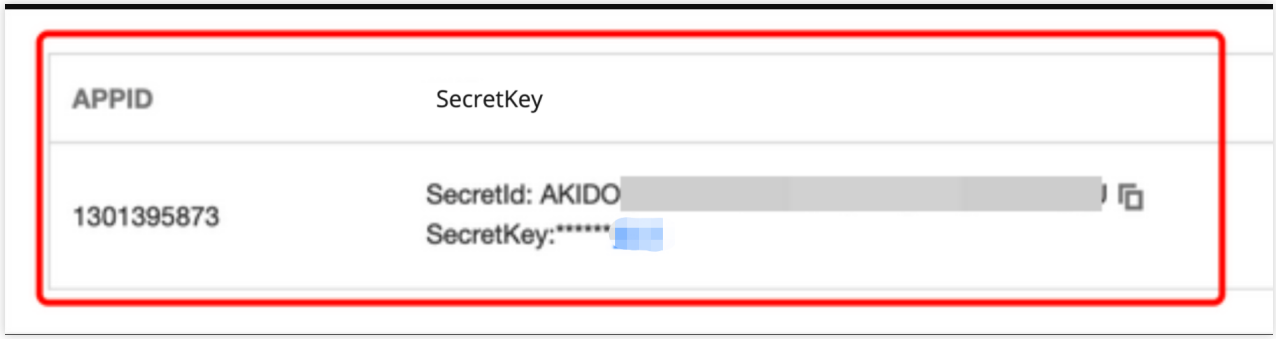


腾讯云账号

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. 从腾讯云 访问管理控制台 拷贝密钥信息。



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。

应用与项目

最近更新时间：2024-01-03 11:41:55

本文为您详细介绍 CODING 持续部署中的应用与项目。

前提条件

使用 CODING 项目管理的前提是，您的腾讯云账号需要开通 CODING DevOps 服务。

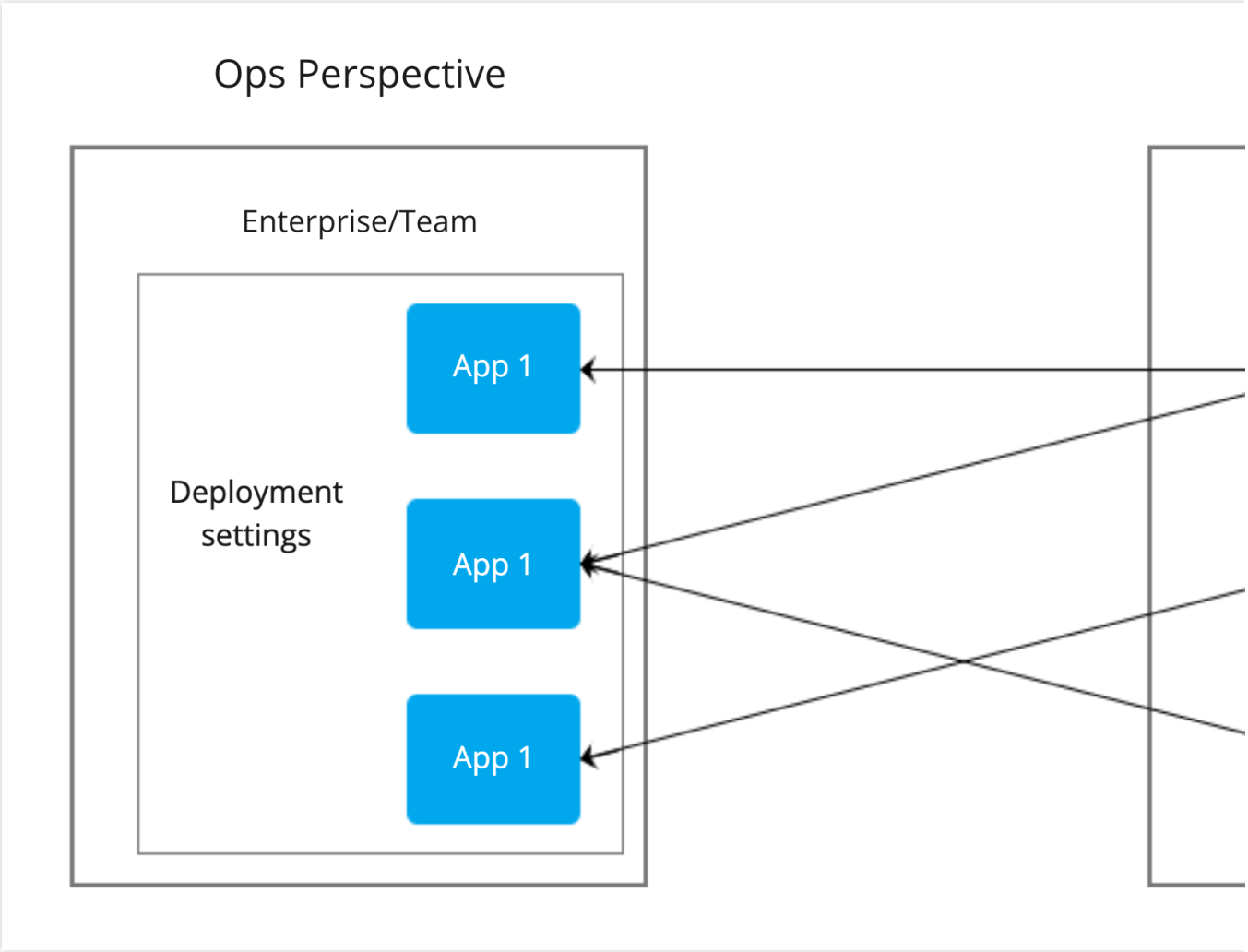
进入项目

1. 登录 CODING 控制台，单击**立即使用**进入 CODING 使用页面。
2. 单击工作台首页左侧的

，进入持续部署控制台。

功能介绍

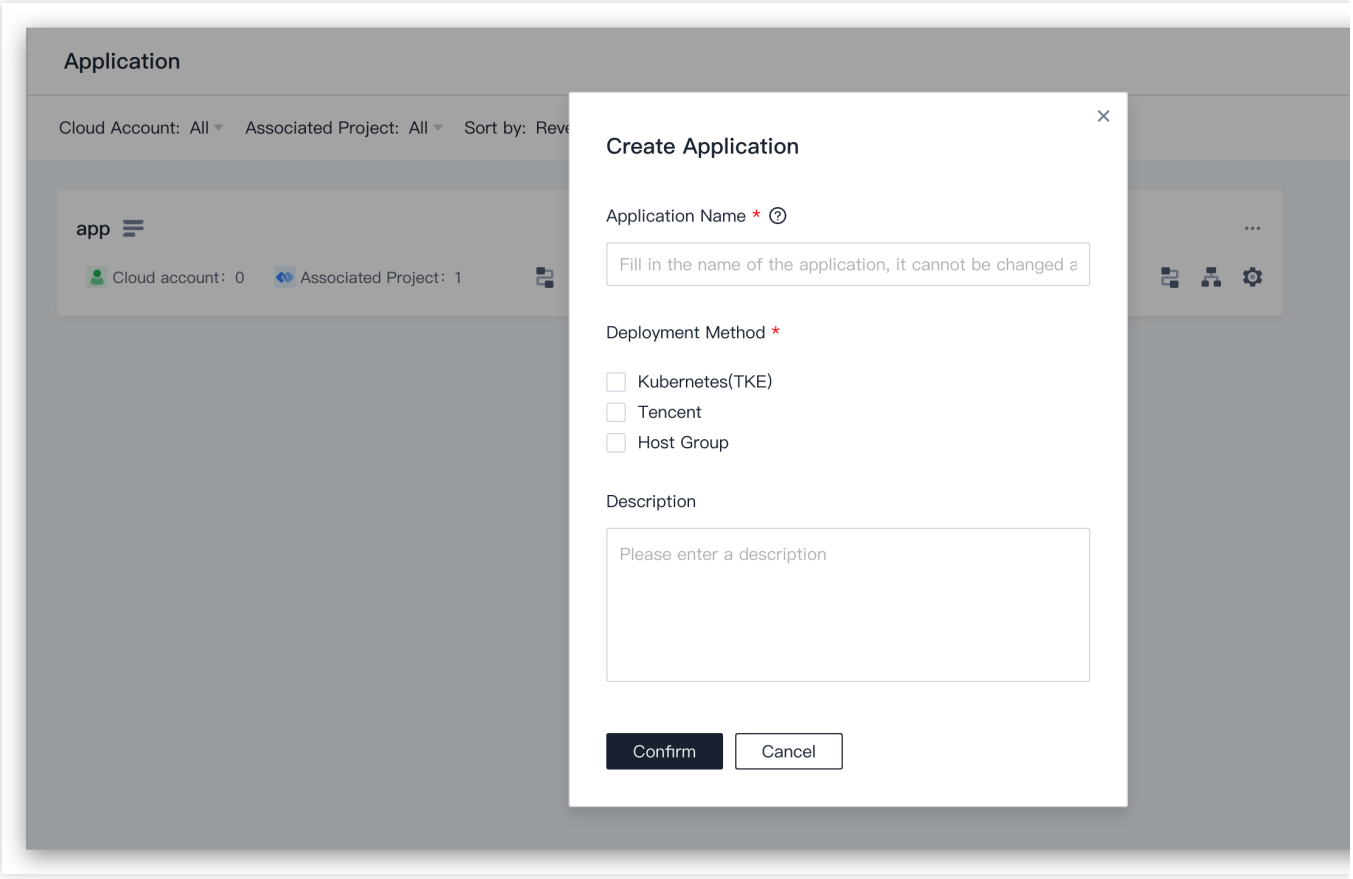
CODING 持续部署中的应用和项目都是属于企业/团队的一级资源，它们之间为一对多关系，即一个项目包含多个应用，一个应用从属于多个项目。



在该设计下，运维人员可以专注于应用的持续部署管理（部署流程、基础设施等），而非运维人员（一般指开发）只需要在项目维度操作（提交发布单，查看发布详情），使运维能够专注于在云的基础上做基础设施运维，开发在项目内就能够进行大部分业务运维并完成从需求到发布的完整闭环。

应用

应用是 CODING CD 中的基本部署单位。应用包含若干个应用集群，以及安全组和负载均衡器等。应用对部署的软件集合进行抽象，通常代表您想要部署的服务、配置、以及运行所需的基础设置。推荐的做法是一个应用对应微服务架构中的一个服务。



对应关系示例

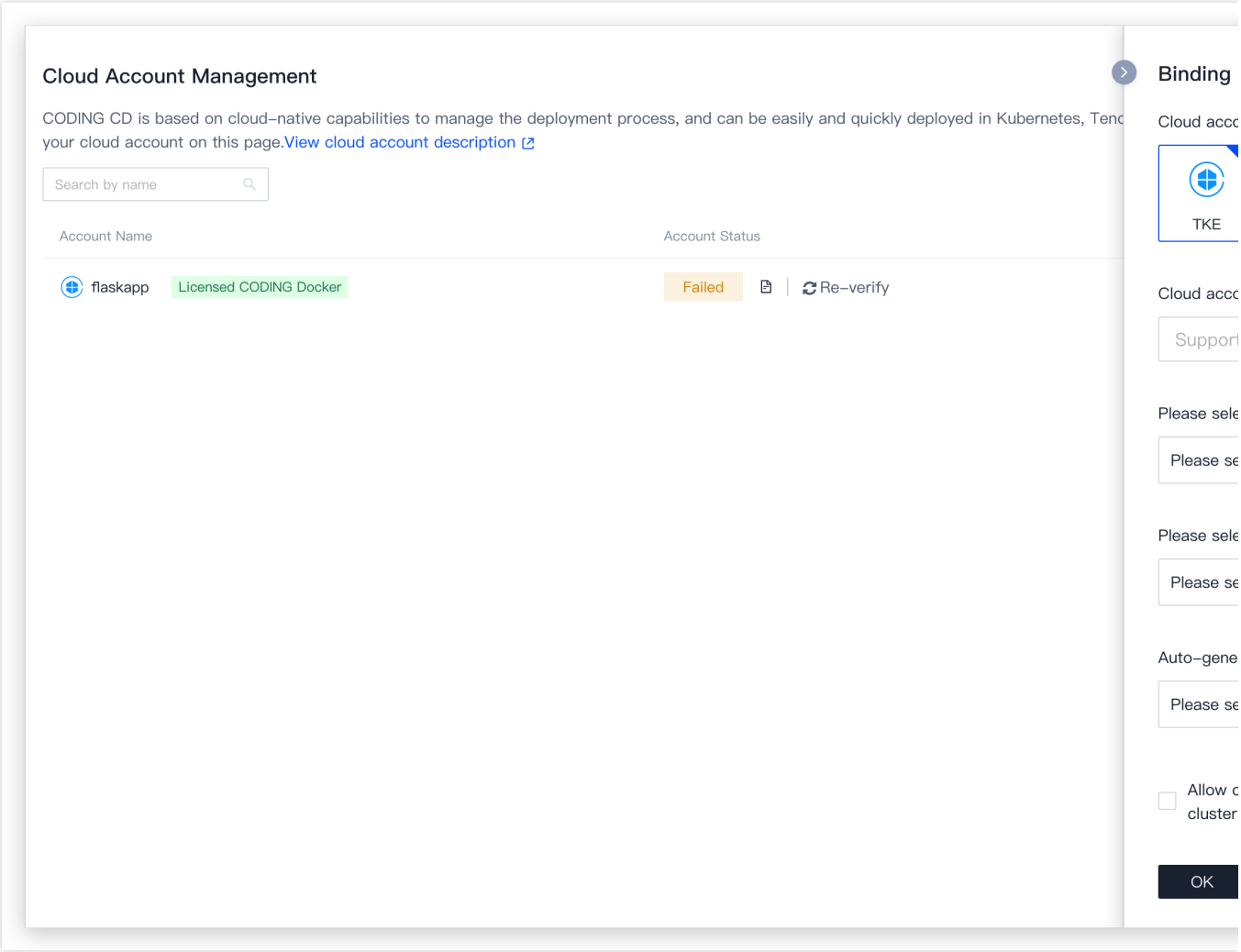
在微服务架构下的微服务对应于一个 CODING 持续部署的应用，当然您也可以在了解对应关系的情况下依照自己的偏好来设定对应关系。下面是一个典型的团队、项目、应用、集群、云账号之间的关系示例：

团队：XXX 科技有限公司	
云账号	自建 Kubernetes Service Account 腾讯云北京 TKE 集群 Service Account 腾讯云中国香港 API Key
项目1：车载用品电商站点项目	应用1：车载电商后端 应用2：车载电商前端 应用3：物流管理服务
项目2：服装电商站点项目	应用1：服装电商后端 应用2：服装电商前端 应用3：物流管理服务
部署控制台	应用1：车载电商后端 测试集群 生产集群 应用2：车载电商前端

	测试集群 生产集群 应用3：物流管理服务 车载电商测试集群 车载电商生产集群 服装电商测试集群 服装电商生产集群 应用4：服装电商后端 测试集群 生产集群 应用5：服装电商前端 测试集群 生产集群

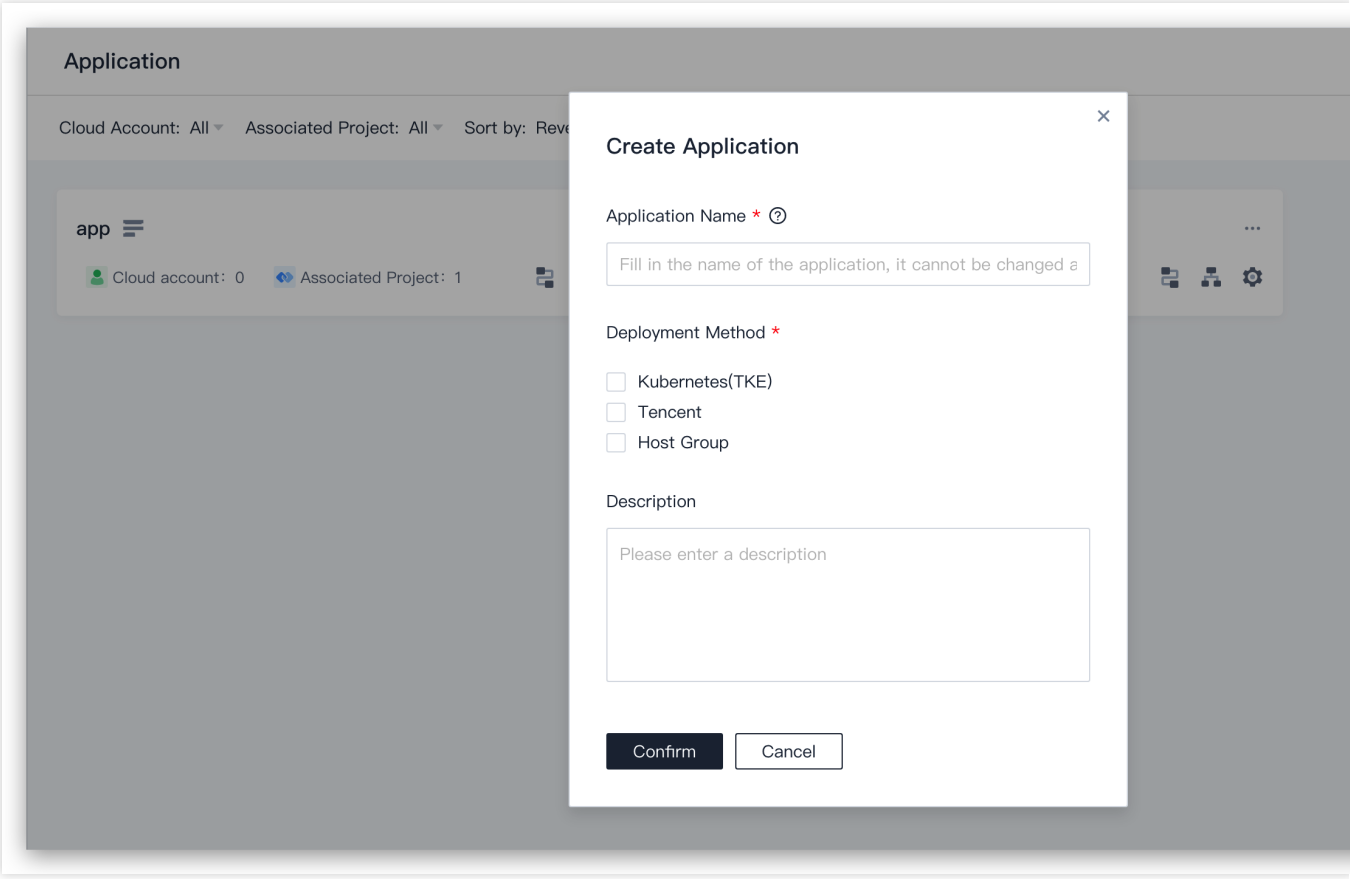
云账号绑定

云账号是访问云资源的凭证，进入 CODING 部署控制台创建应用，单击导航栏**应用 > 创建应用**。在进行应用创建之前，请确保您已经完成了 云账号绑定。



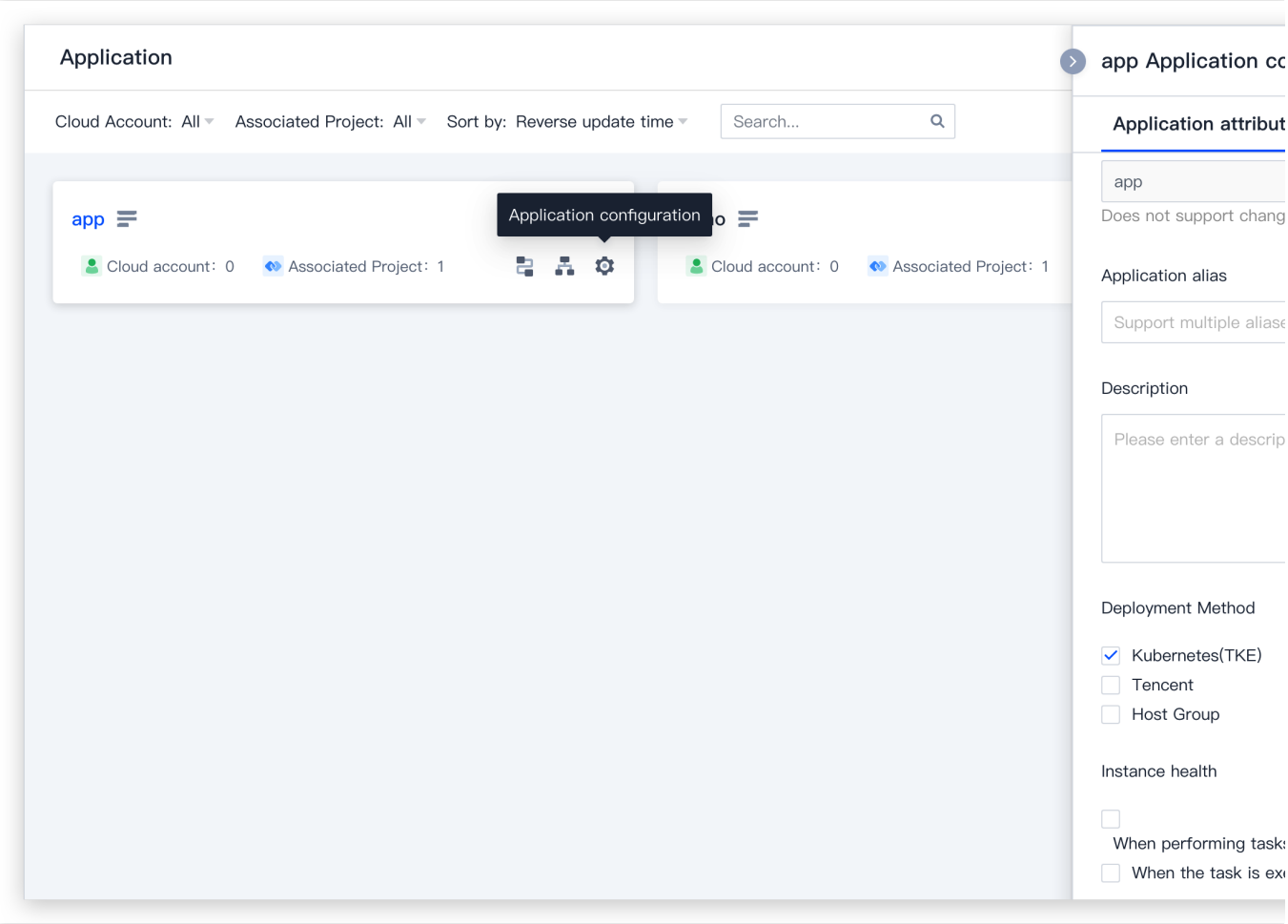
应用创建

单击首页左侧的控制台，单击右上角**创建应用**。



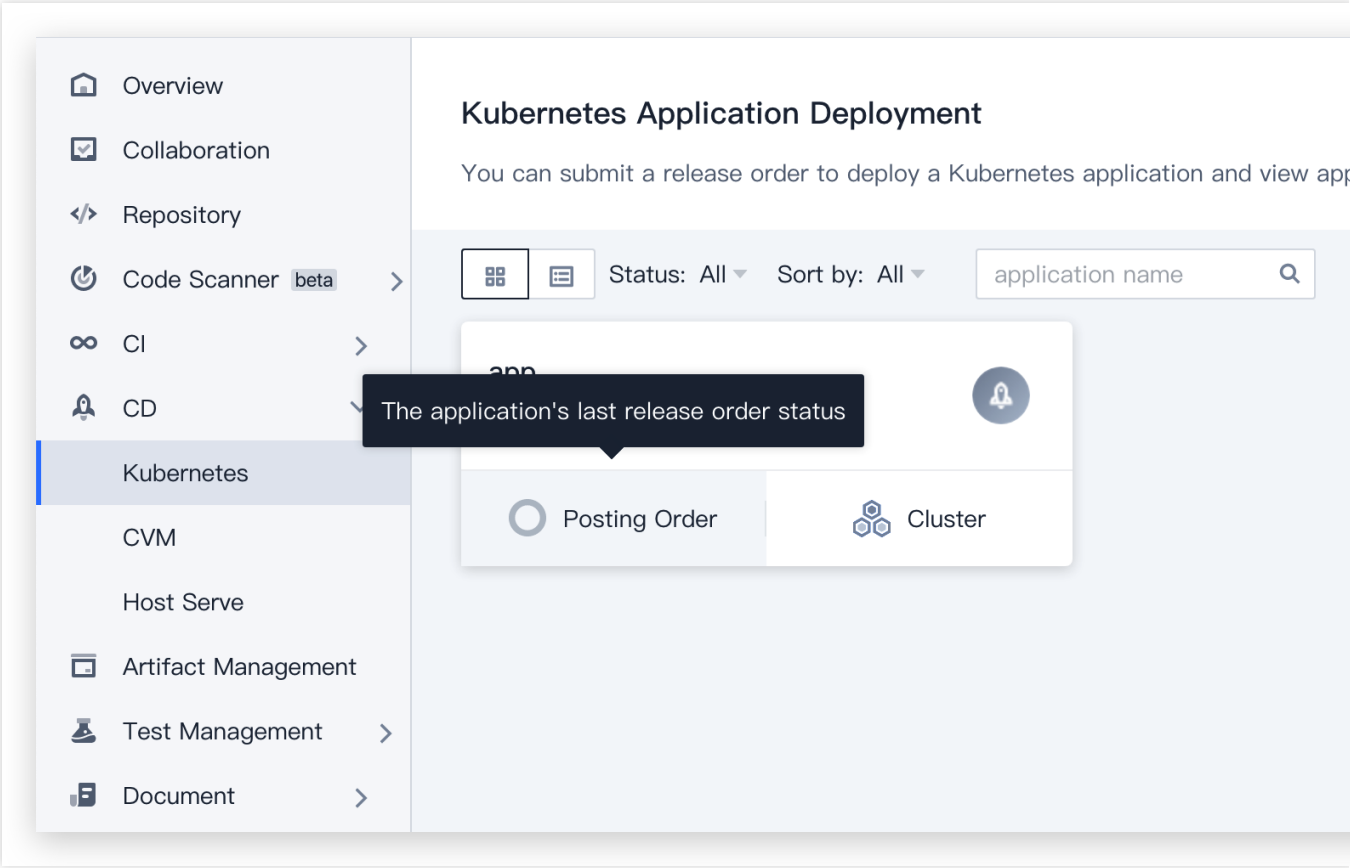
与项目关联

在部署控制台内完成应用创建后，可以直接在控制台首页将应用与项目相关联。



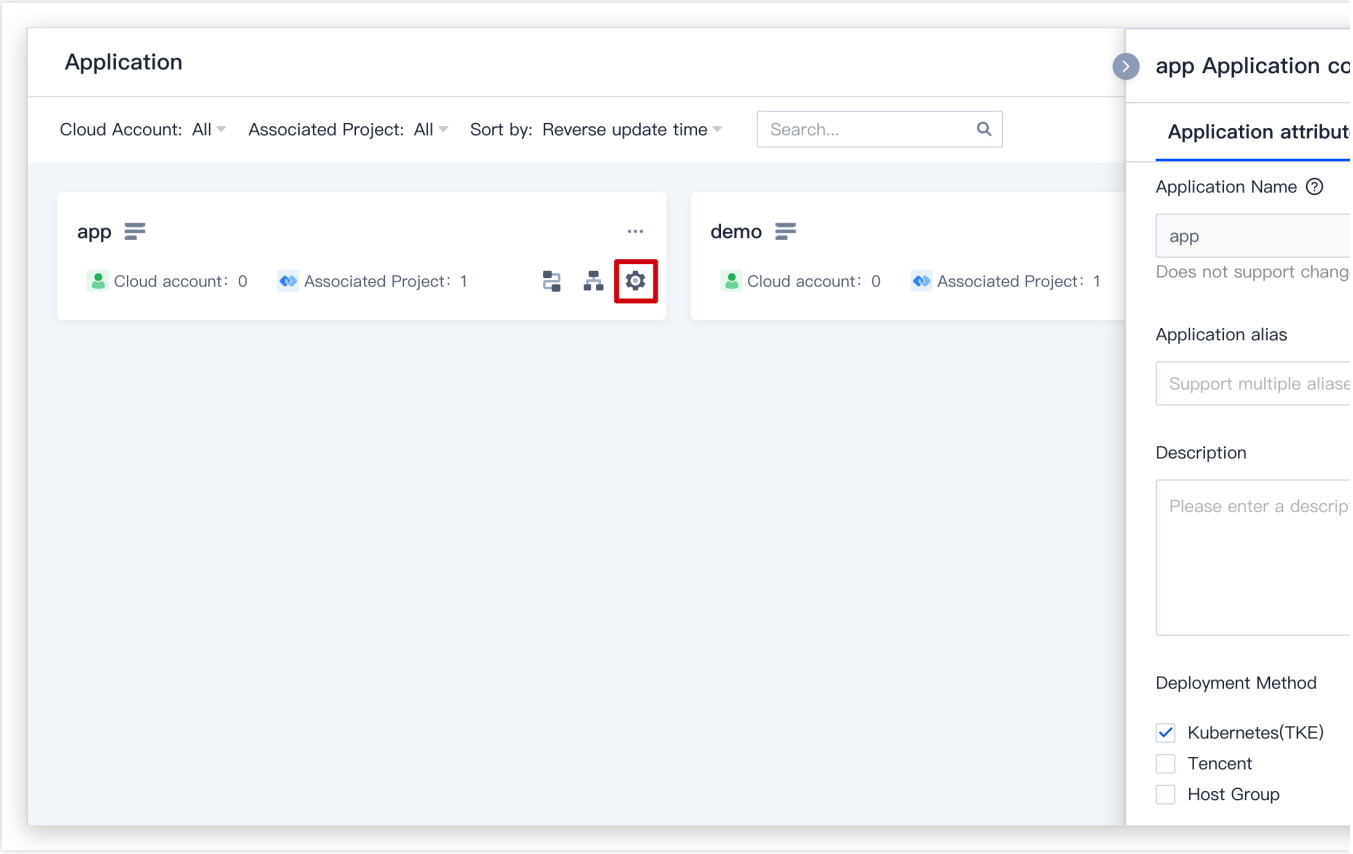
新建发布单

当运维人员完成对应用的 部署流程配置 后，开发人员在项目内就可以实现从项目协同到应用发布的 DevOps 闭环。典型的场景是当有新版本需要发布时，开发人员在**持续部署 > Kubernetes** 页面新建发布单，发布单自动触发部署流程执行，开发可随时查看发布状态和历史详情。



管理应用

在部署控制台新建应用后可以在应用的**配置**中调整应用属性与通知，或删除应用。

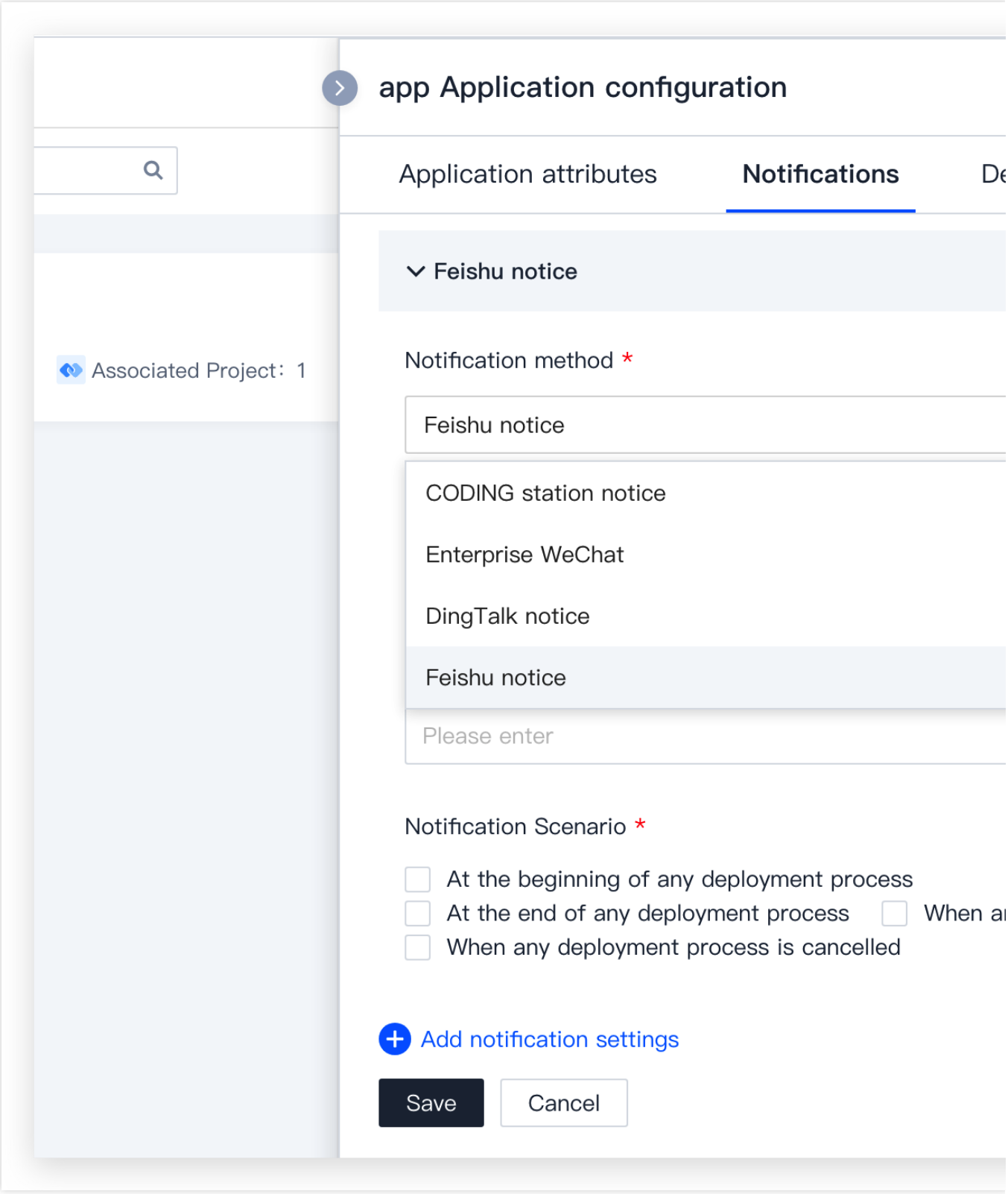


应用通知

目前支持 CODING 站内通知、企业微信、钉钉和飞书四种通知方式。

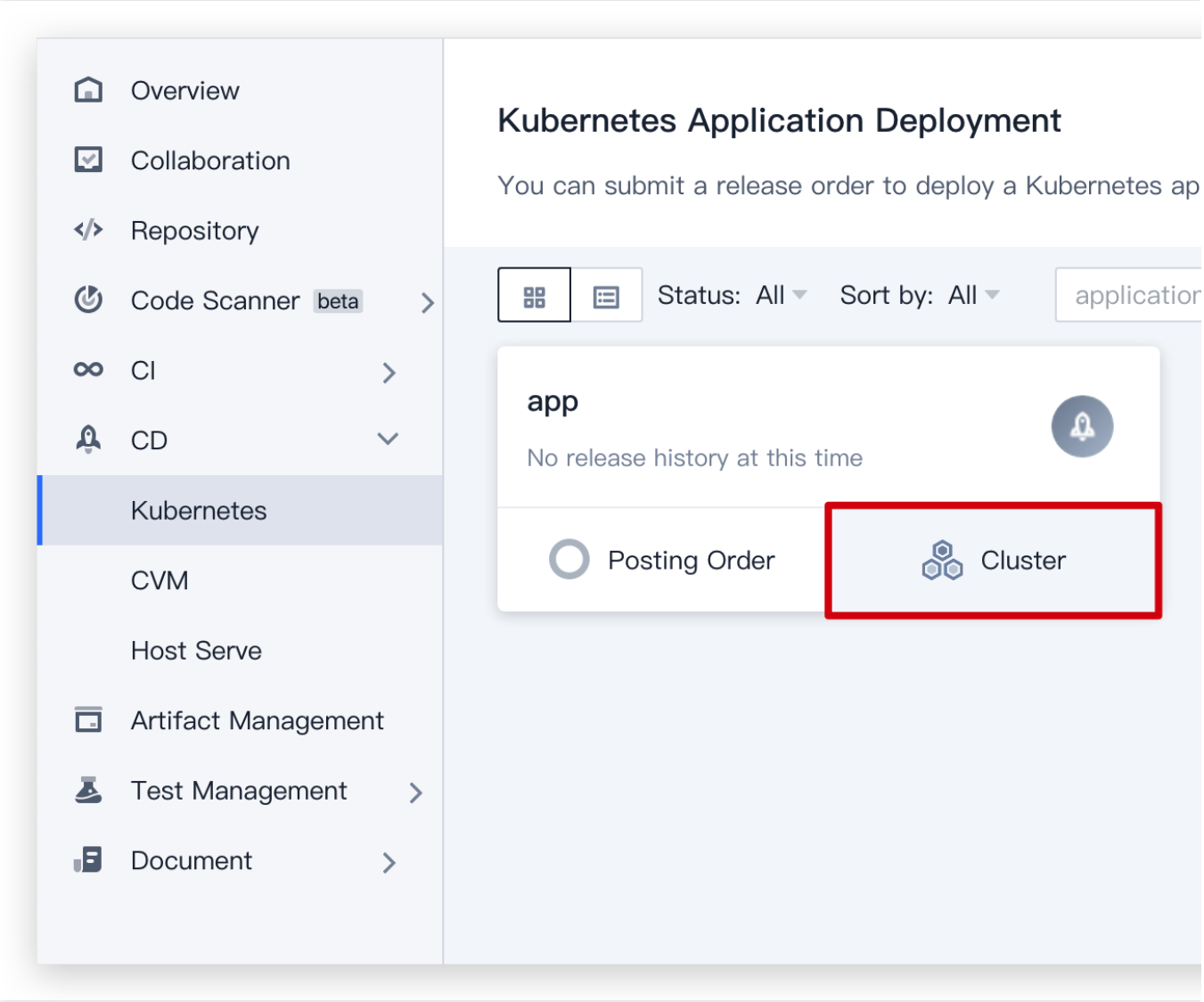
显示 / 隐藏功能入口

对于不需要显示的功能入口，可以在**特性**栏将其禁用，这里的禁用并不会删除相应的数据，仅表示在控制台界面隐藏。可以隐藏部署流程、集群、负载均衡器和安全组功能入口：



添加实例的自定义属性链接

在**集群** > **服务组** > **实例详情**面板可以查看运行实例的自定义链接，自定义链接提供关于实例的简略信息，如：日志、健康状态等。



自定义链接对应的 IP 可以是公有或私有 IP，默认端口为 80；如果需要设置其他端口号，在 Path 文本框以 `:` 开头，如：`:7002/health` 。

1. 在**链接**一栏，单击 **Add Section**。
2. **Section Heading** 输入自定义链接标题。
3. **Links** 输入自定义链接名称，以及 URL。

说明：

URL 字段支持使用表达式引用更多的实例属性。例如对于腾讯云实例，可以使用 `{region}` 引用实例的所在地域。

4. 单击 **Add Link** 在同一属性下添加更多链接。
5. 单击 **Add Section** 添加新的自定义属性链接。
6. 单击**撤销**取消添加操作。**撤销**不会删除已保存的自定义属性链接。
7. 单击**保存**完成操作。

流量保护

说明：

流量保护旨在确保任何时间至少有一个实例处于正常运行状态。

启用流量保护功能后，如果用户或者脚本尝试删除、禁用或对服务组进行伸缩容操作，CODING 控制台会对操作进行验证以确保集群中至少有一个实例在正常运行，否则将会拒绝用户或脚本请求。

1. 在**流量保护**栏，单击**添加流量保护**。
2. 以下是需要填写的字段：

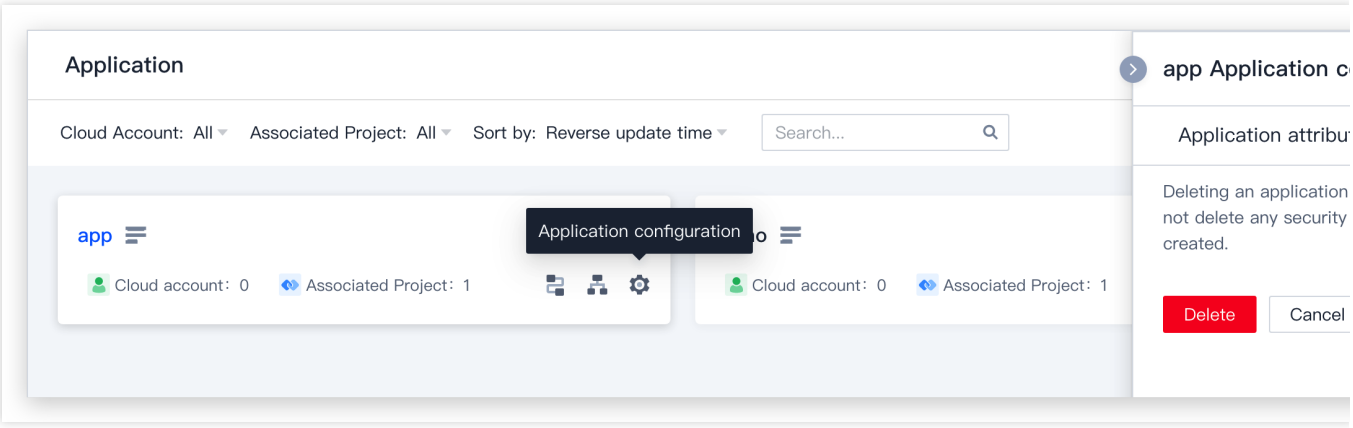
字段	必填项	说明
云账号	是	设置流量保护的云账号
地域	是	可选的地域，`*` 表示选择所有地域
分组	否	设置流量保护的集群分组，如果留空表示选择不属于任何分组的集群
详情	否	详情是区分集群的三级字段，具有相同 `${Application}-${Stack}-${Detail}` 的服务组属于同一个集群

3. 单击**保存**使配置生效。

应用删除

如果应用中有服务组，需要先删除服务组。

进入 部署控制台 后，单击应用右下角**齿轮图标**，进入应用配置页后单击**删除**。



部署流程
部署流程介绍

最近更新时间：2024-01-03 11:42:34

本文为您详细介绍 CODING 持续部署中的部署流程。

前提条件

使用 CODING 项目管理的前提是，您的腾讯云账号需要开通 CODING DevOps 服务。

进入项目

1. 登录 CODING 控制台，单击**立即使用**进入 CODING 使用页面。
2. 单击工作台首页左侧的

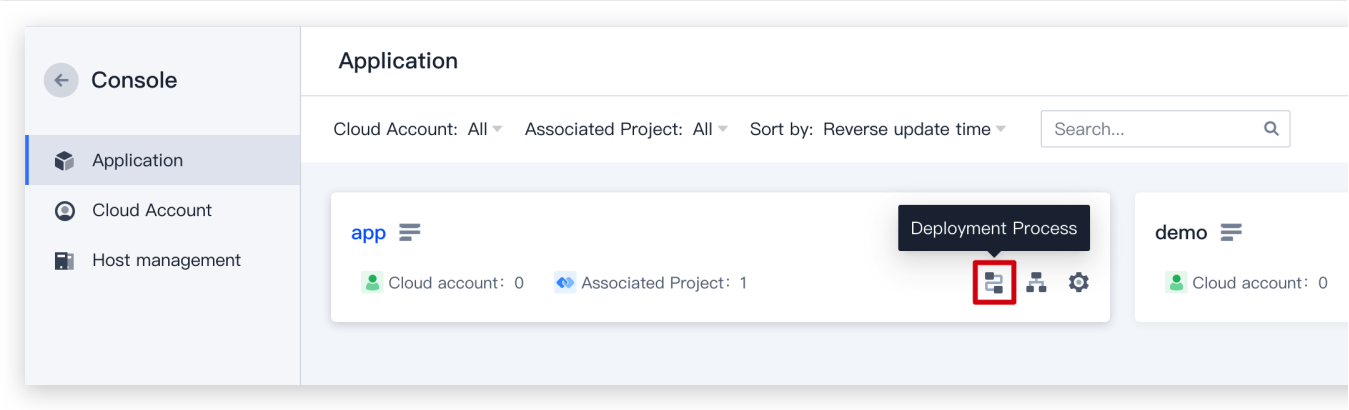
，进入持续部署控制台。

功能介绍

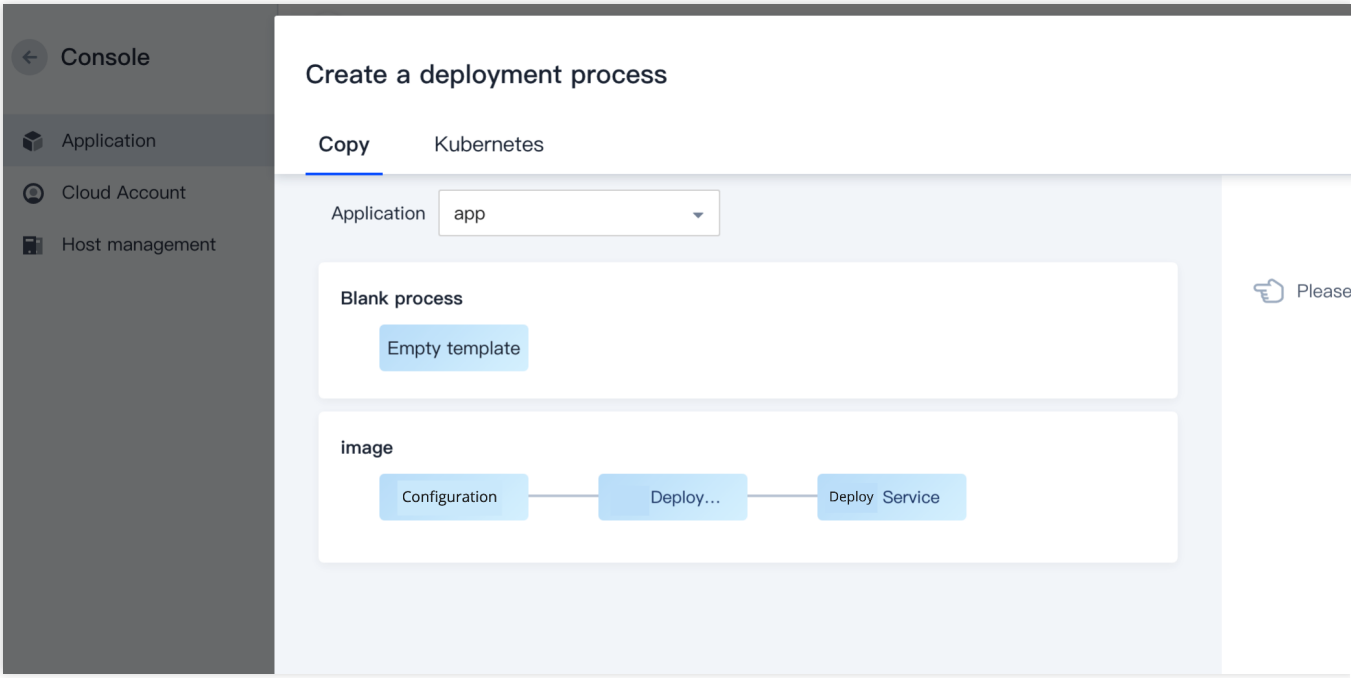
部署流程是实现持续部署最核心的模块。其强大之处在于支持阶段以任意的顺序组合，这样的能力让部署流程具备出色的灵活性、一致性和可重复性。
灵活性：支持串行、并行控制
一致性：支持多种部署策略，回滚能力，确保发布结果符合预期
可重复性：部署流程可重复执行，阶段可被其他部署流程复制使用
您可以配置完全自动化的部署流程，也可以在某些阶段加入手工判断条件。此外部署流程支持多种事件的自动化触发，如 Webhook 触发、由其他部署流程触发等。

新建部署流程

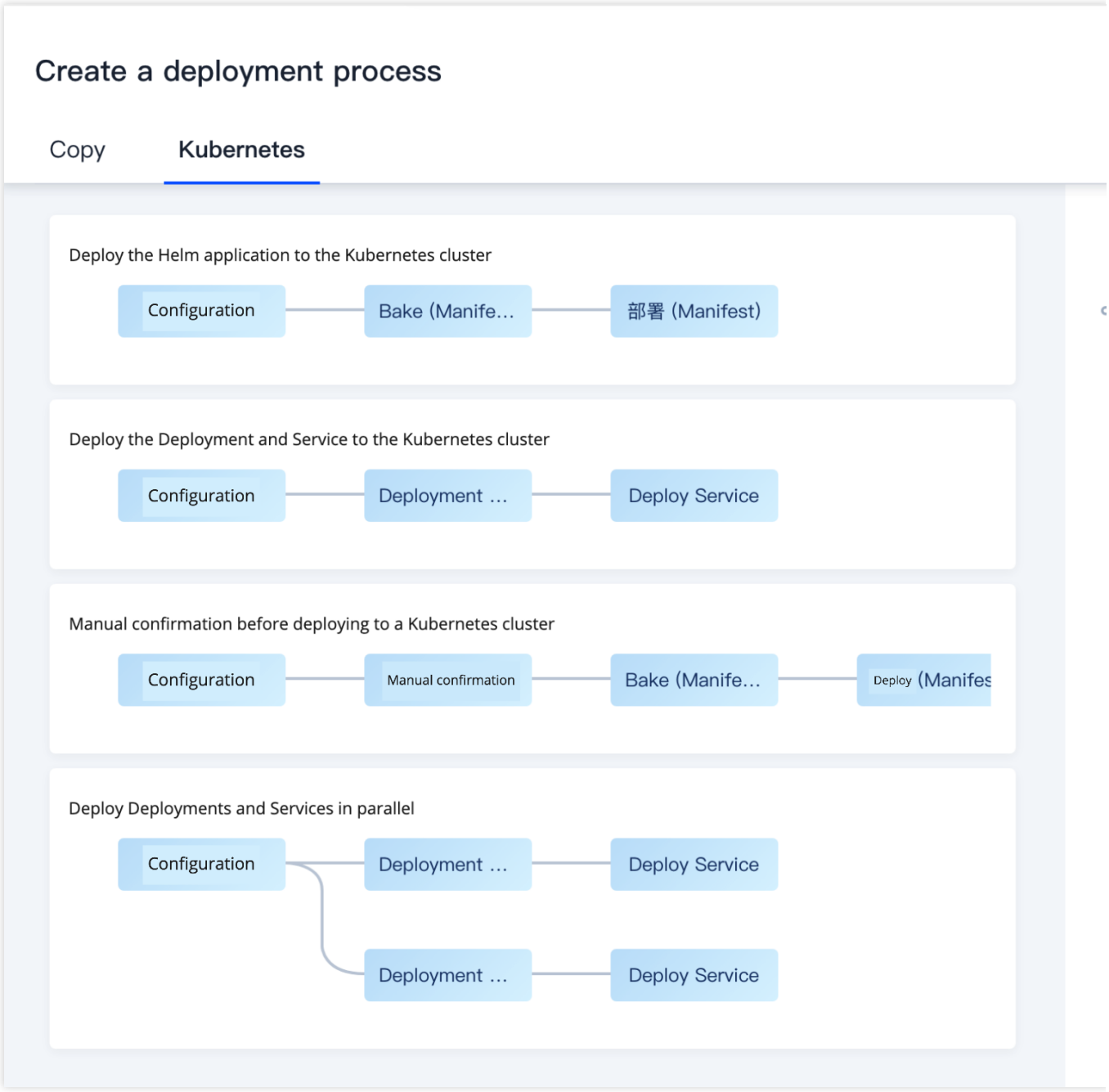
前往部署控制台，单击应用卡片右下方的部署流程按钮。



1. 单击右上角的**创建流程**按钮。

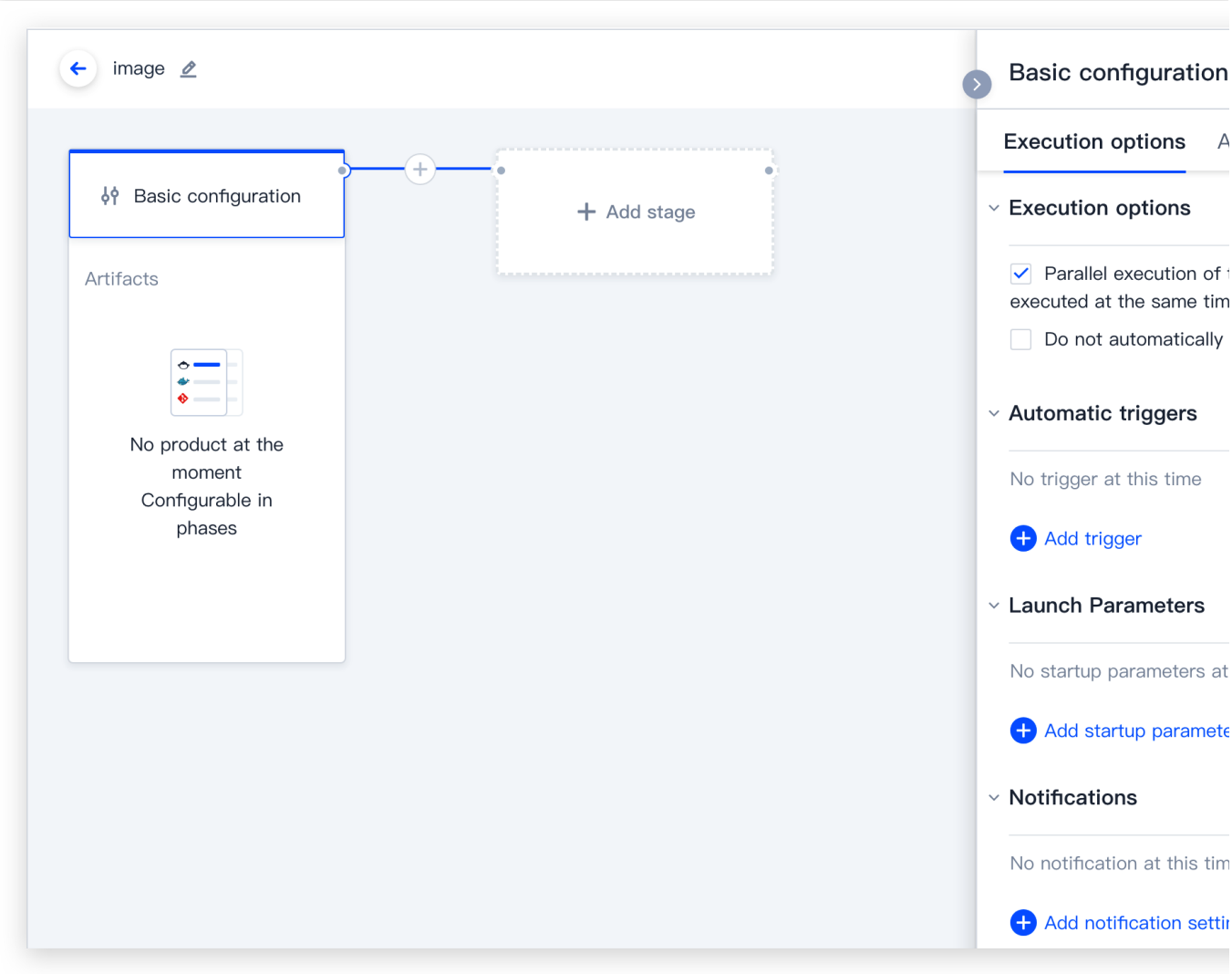


2. 您可以复制在其他应用中创建的流程，或通过空白流程自行创建。CODING 亦提供了 Kubernetes 与腾讯云弹性伸缩参考流程模板。



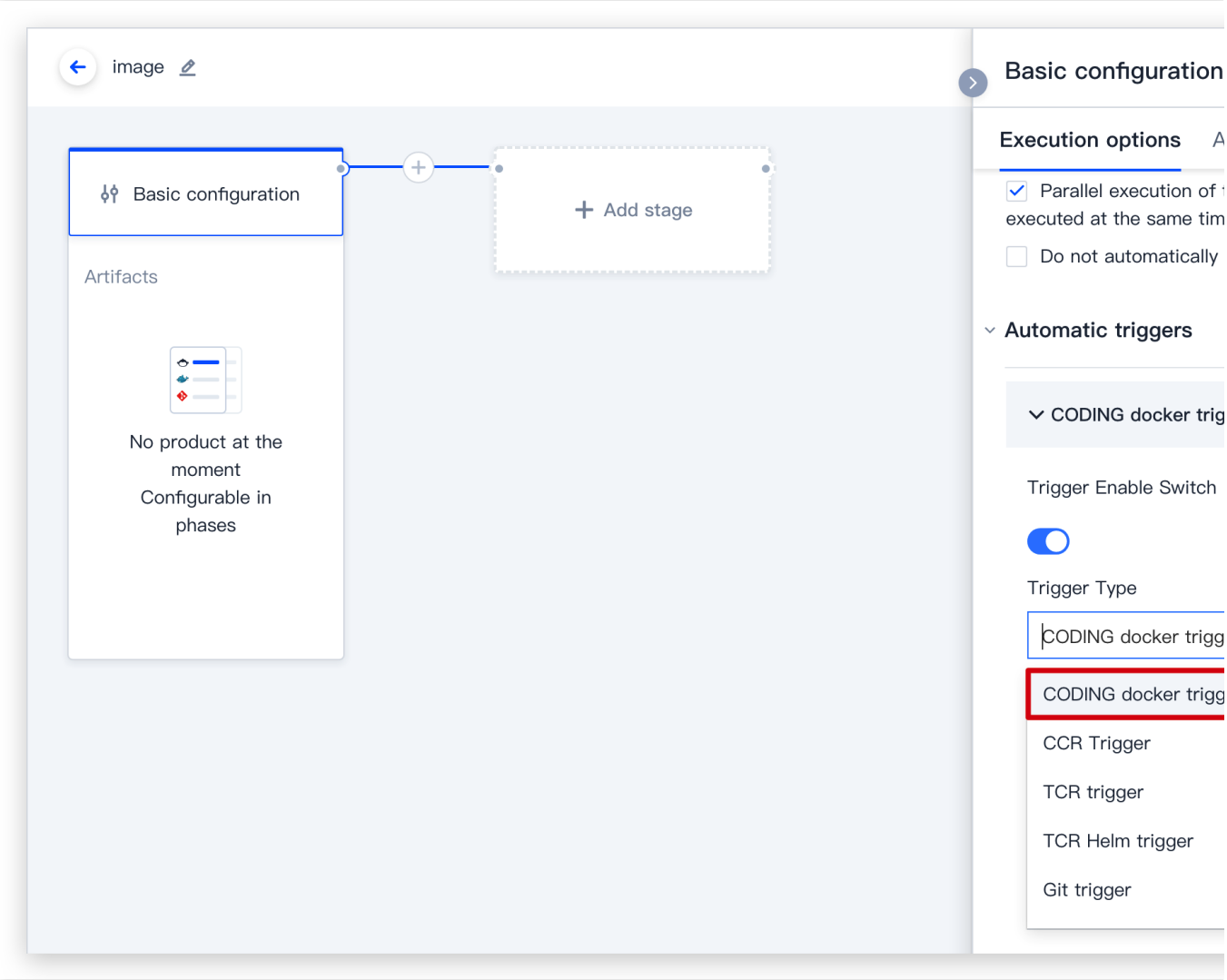
基础配置

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。



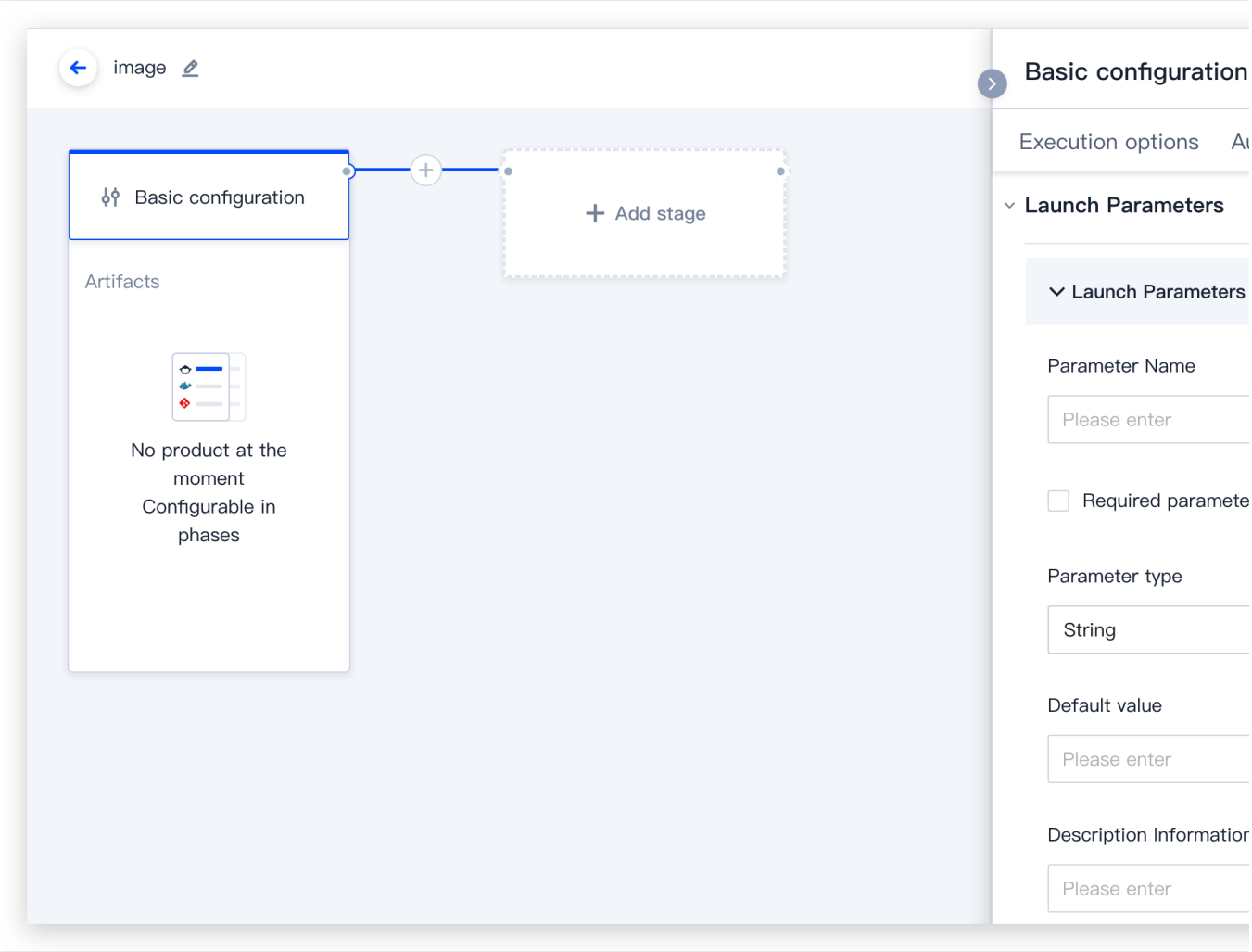
自动触发器

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。



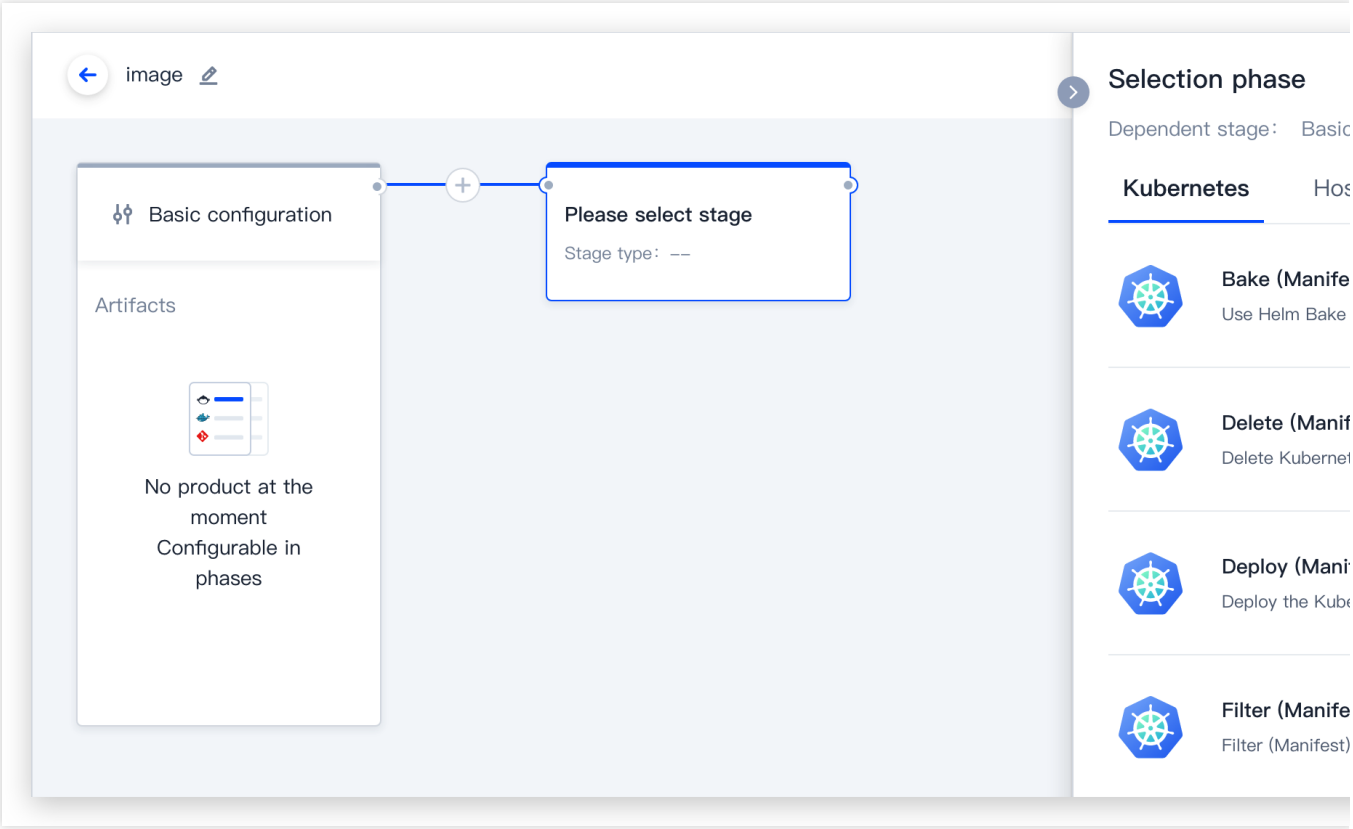
添加部署流程参数

在部署流程配置页面，单击**添加启动参数**，即可开始填写参数。



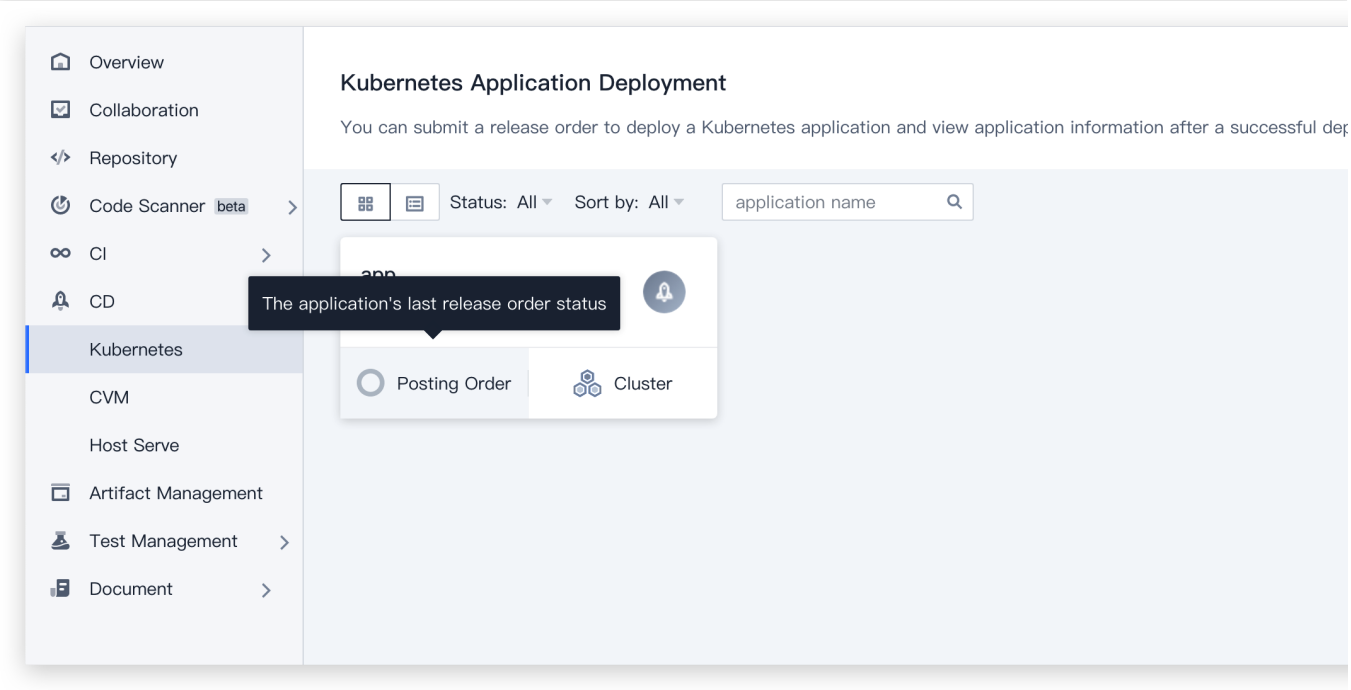
添加阶段

在部署流程配置页面单击 + 即可添加新的阶段，右侧列表中支持选择阶段类型。



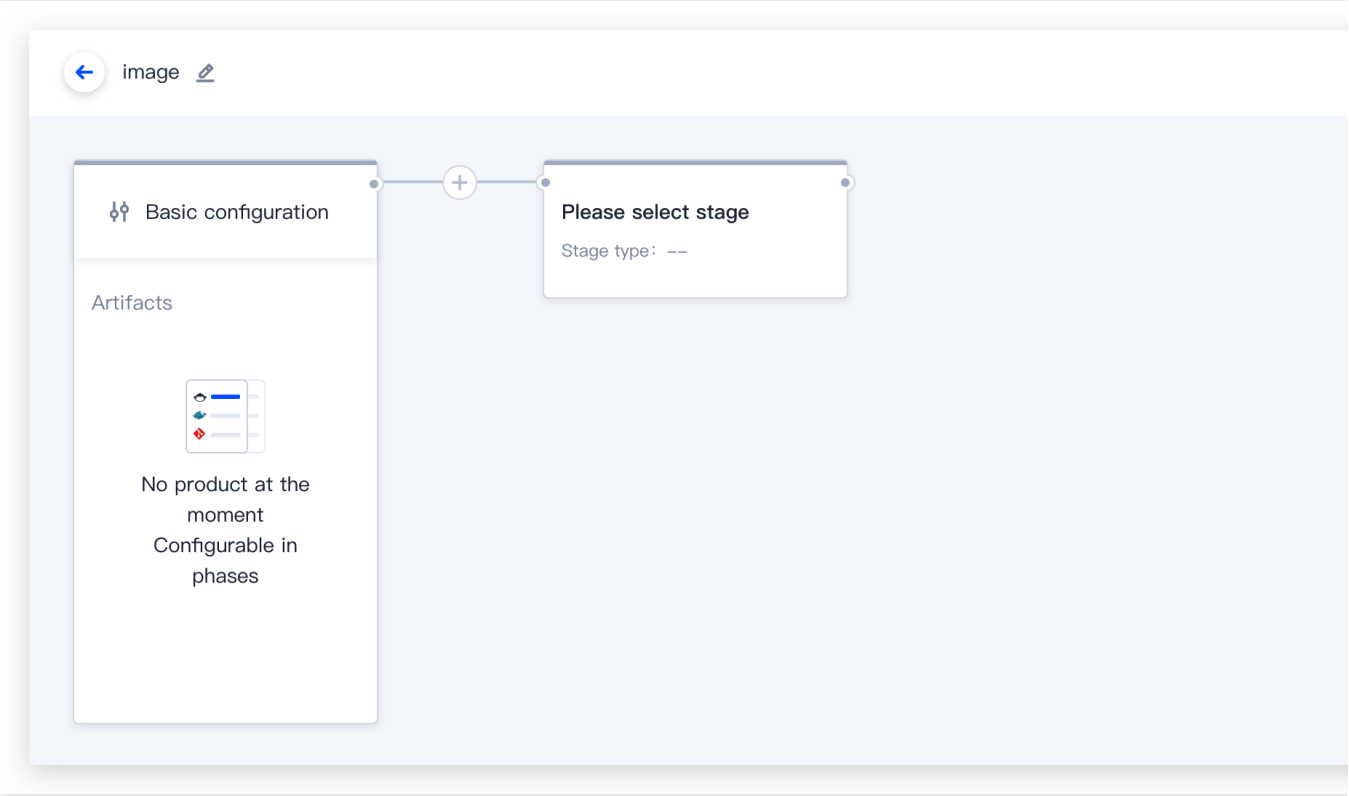
执行部署流程

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



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。

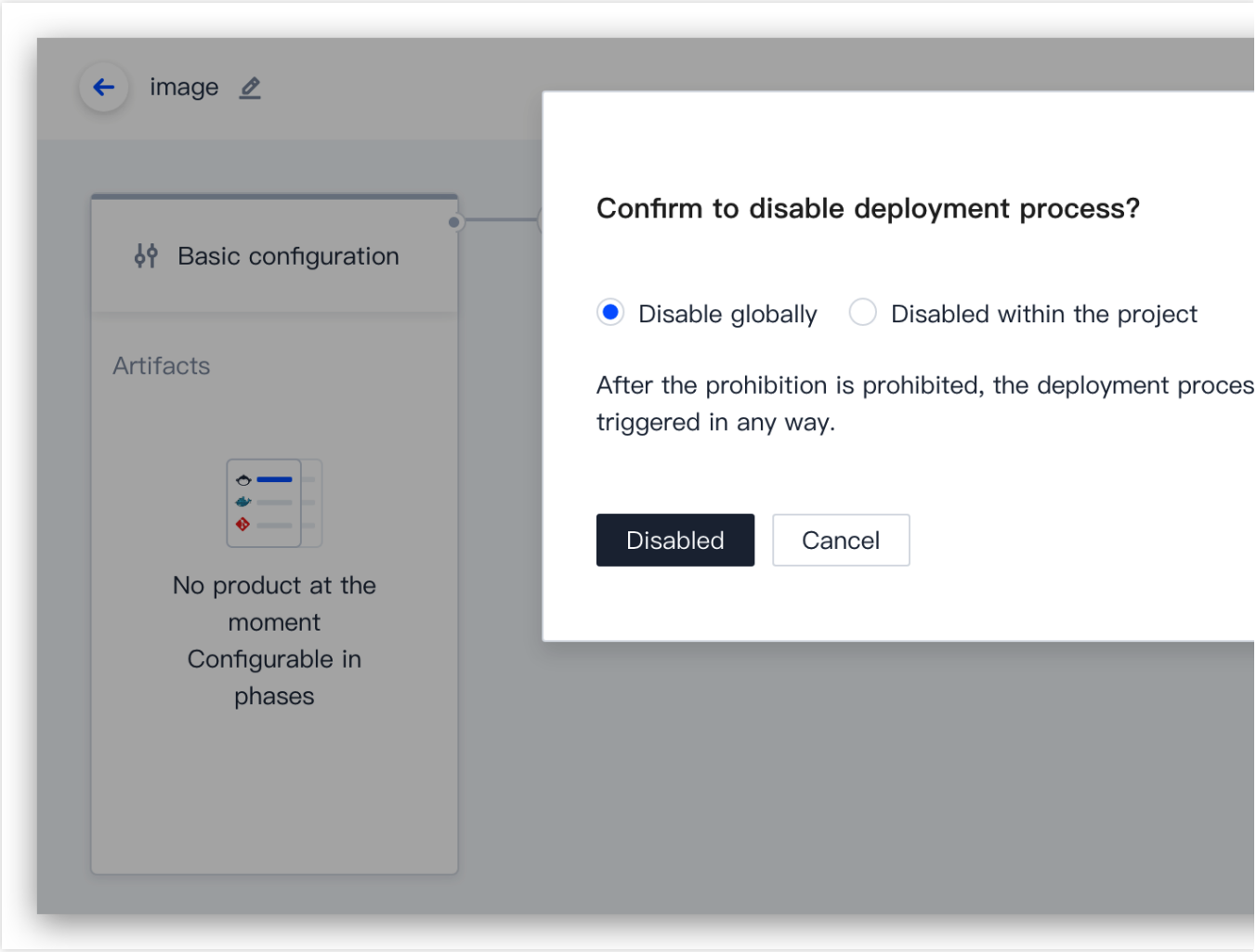


删除部署流程

设置后，将删除此部署流程。

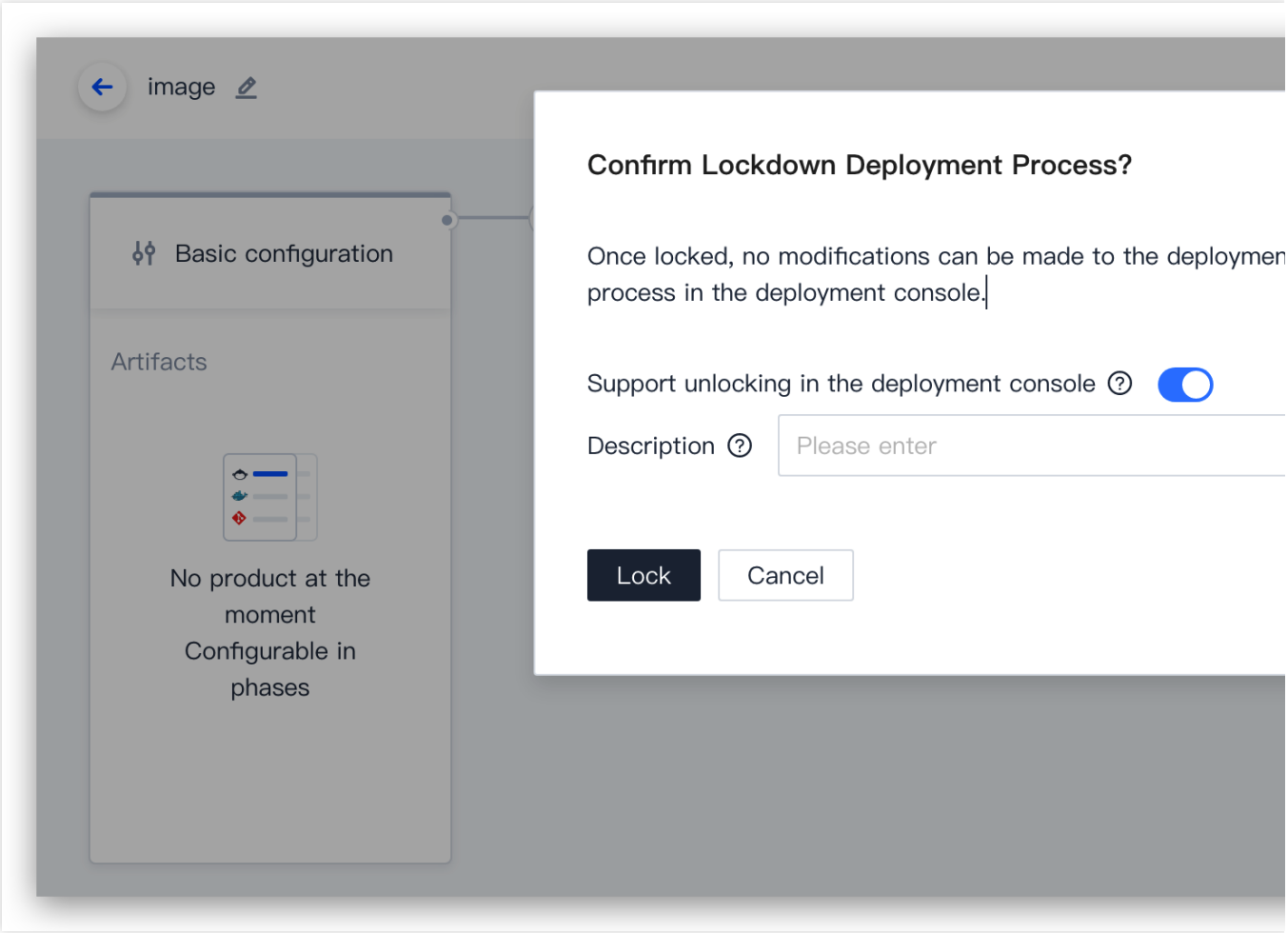
禁用部署流程

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。



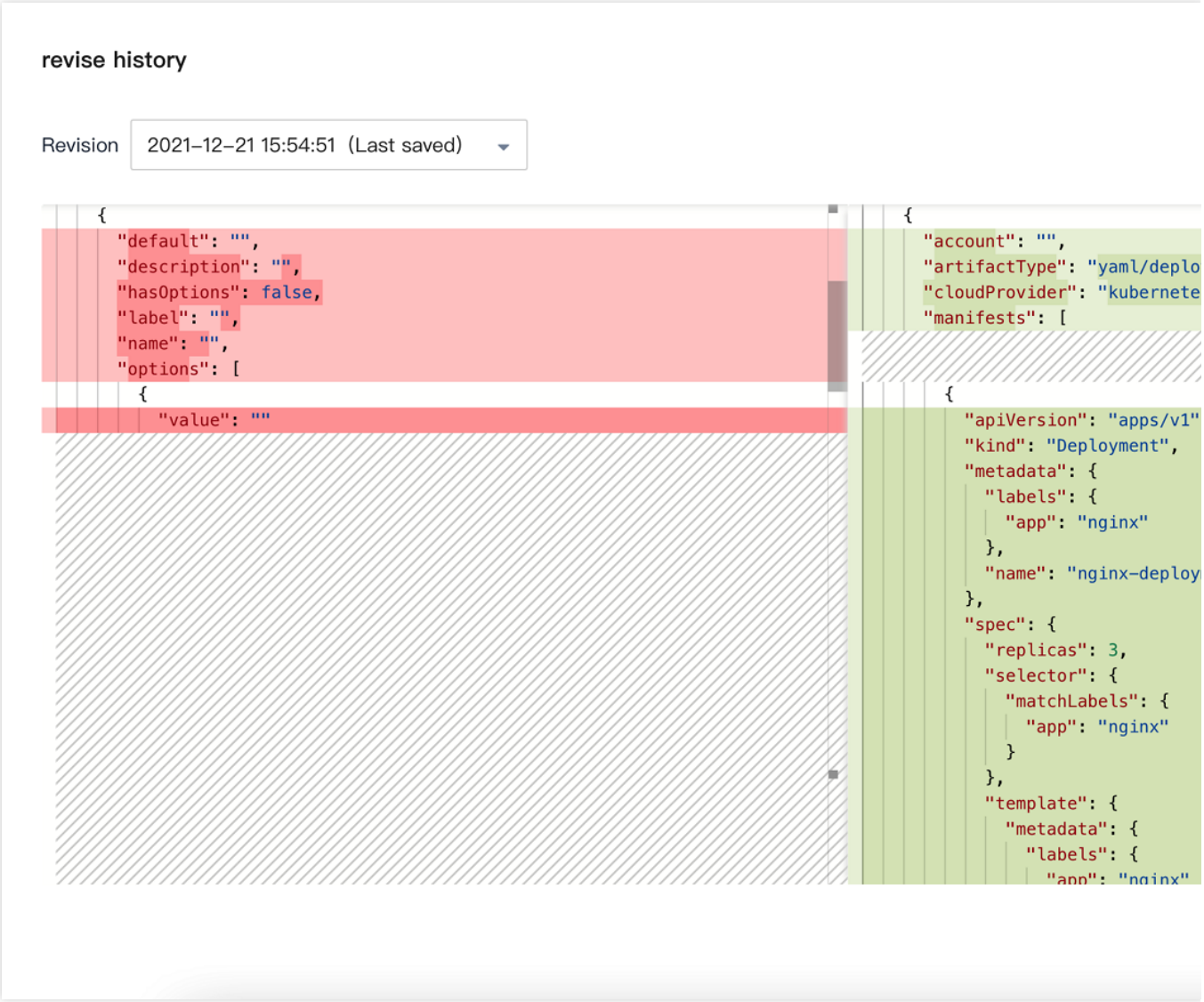
锁定部署流程

锁定部署流程后，将不能通过部署控制台编辑部署流程。可以选择是否允许通过 API 接口对部署流程进行更新。



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。



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。


image
Edit JSON configuration
"desc": "",
"expectedArtifacts": [],
"keepWaitingPipelines": false,
"lastModifiedBy": "xeOeNZIEVz",
"limitConcurrent": true,
"parameterConfig": [
{
"default": "",
"description": "",
"hasOptions": false,
"label": "",
"name": "",
"options": [
{
"value": ""
}
],
"pinned": false,
"required": false
}
],
"relationProjects": [
{
"bind": true,
"bindTime": "2022-02-14 16:25:51",
Canc

阶段类型

最近更新时间：2024-01-03 11:45:47

本文为您详细介绍在 CODING 持续部署中部署流程的阶段类型。

前提条件

使用 CODING 项目管理的前提是，您的腾讯云账号需要开通 CODING DevOps 服务。

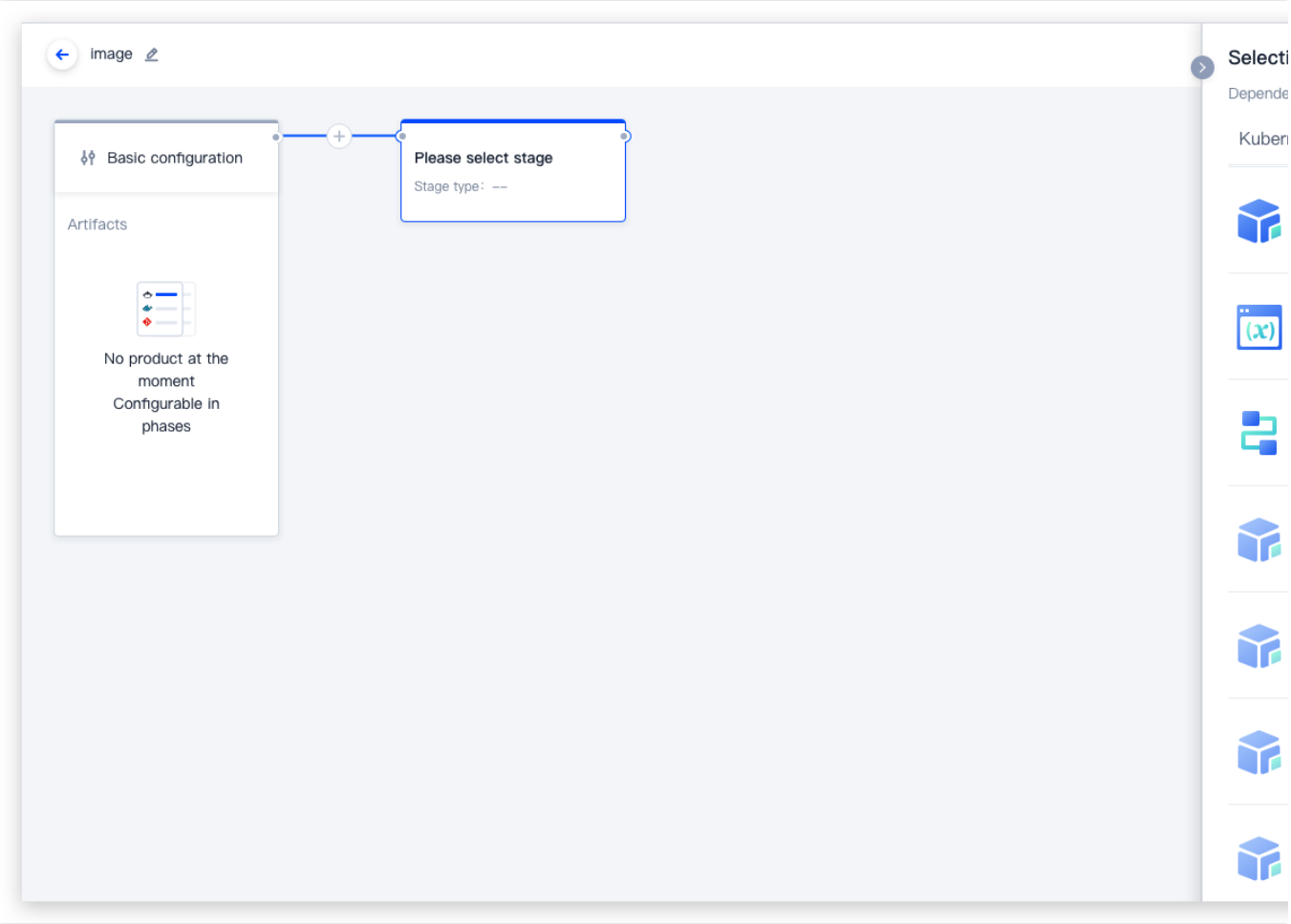
进入项目

1. 登录 CODING 控制台，单击**立即使用**进入 CODING 使用页面。
2. 单击工作台首页左侧的

，进入持续部署控制台。

通用类型

编辑部署流程时可以为阶段选择阶段类型。

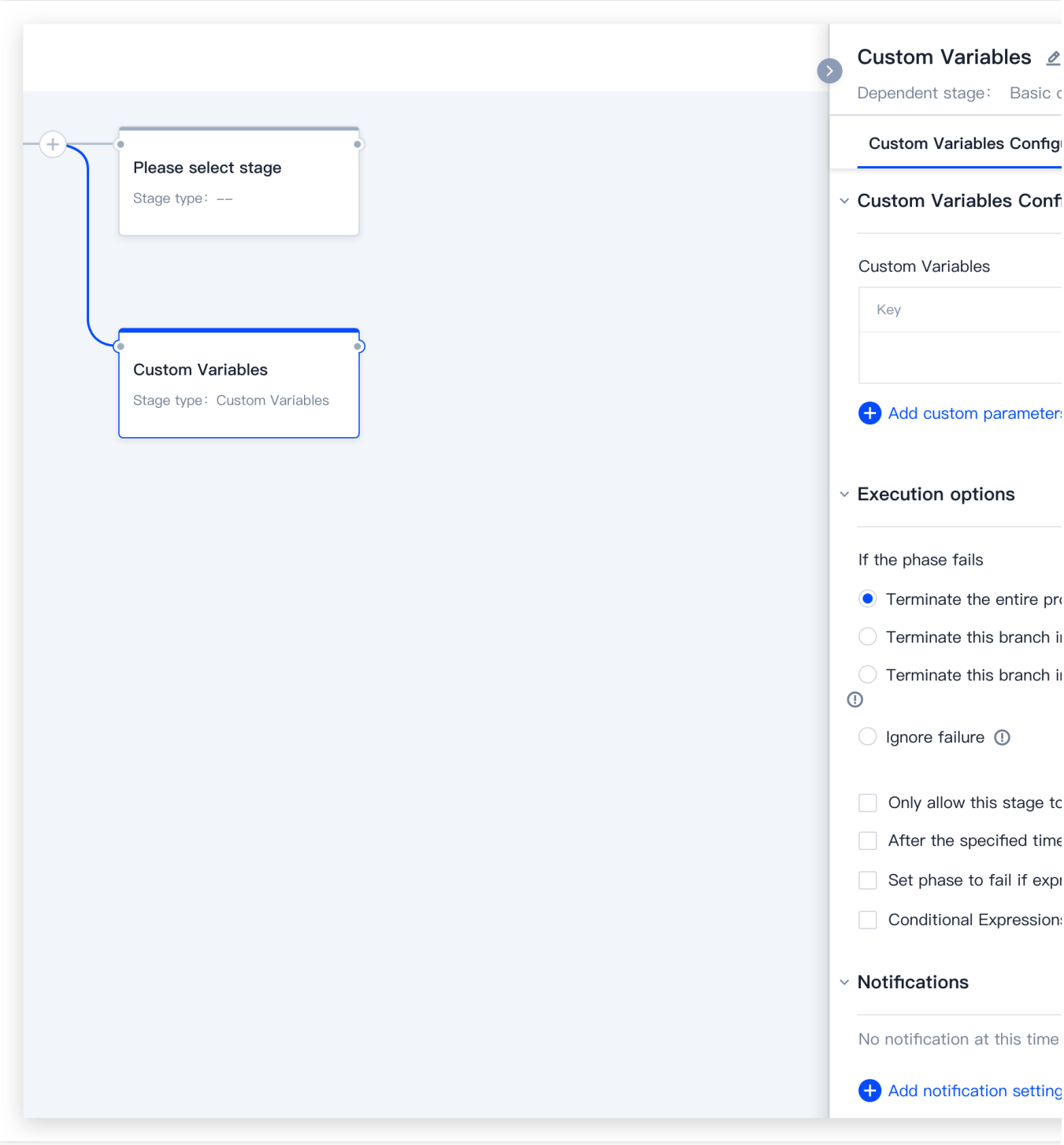


预置条件检查

在执行下一步之前检查前置条件，例如检查集群规模或某个阶段的状态。

自定义变量

添加自定义变量（`key/value` 键值对），在此阶段的定义的变量可以被下游阶段引用。



人工确认

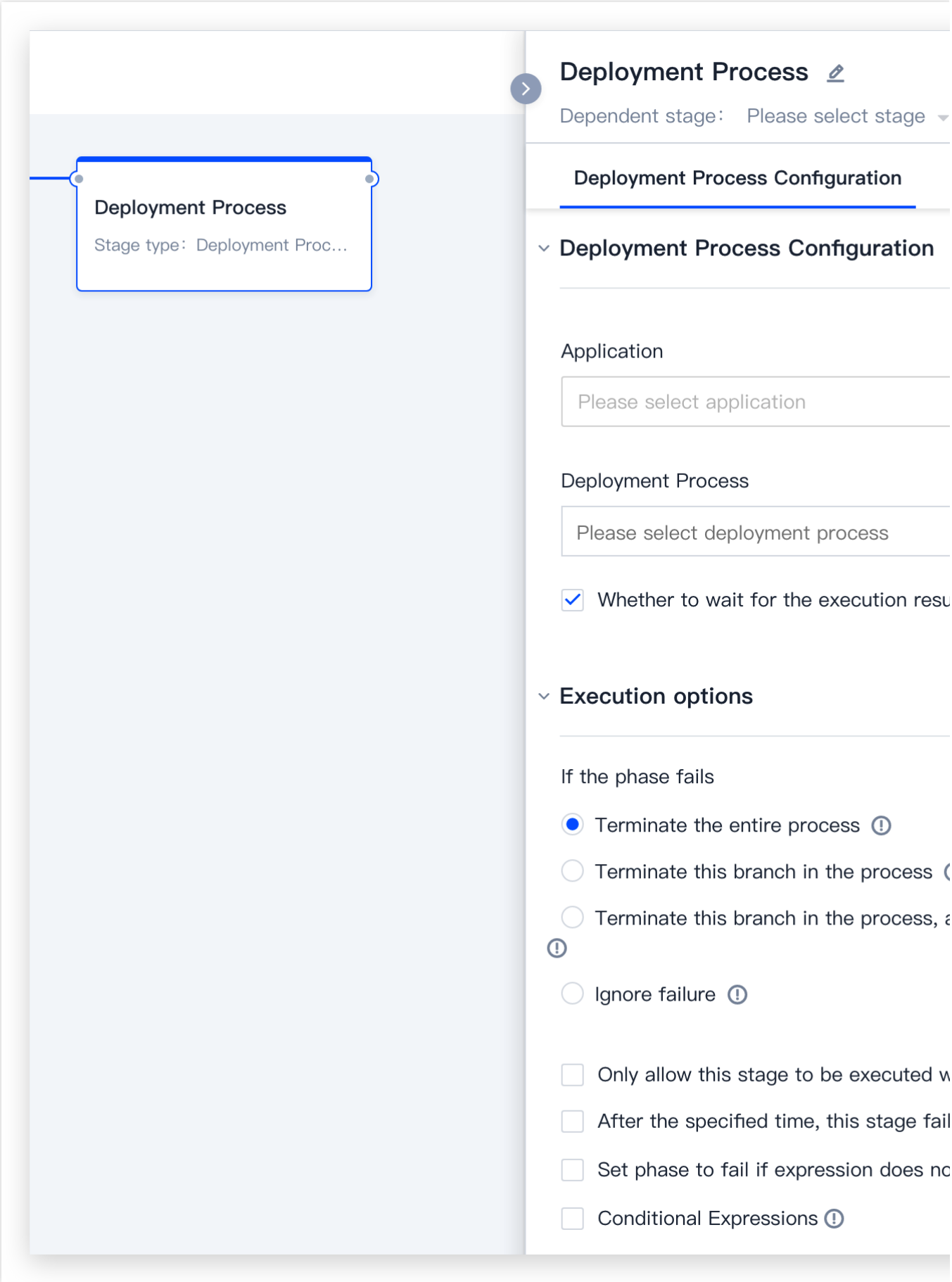
在执行下一步之前等待人工确认。可以在人工确认阶段增加指引说明帮助确认人进行人工确认，或者添加输入选项让用户选择。这些输入选项可以决定下游阶段的执行行为。例如，可以使用 `预置条件检查` 来确保只有满足特定的条件时才执行相应的阶段。

部署流程

将其他部署流程作为子部署流程执行。您可以执行当前应用的部署流程，也可以执行具有访问权限的其他应用的部署流程。在阶段执行结束前可以选择是否等待子部署流程的执行结果。如果选择等待执行结果，此阶段的状态即为子部署流程的最终执行状态；否则，只要子部署流程开始执行此阶段的状态就会被标记为“成功”。

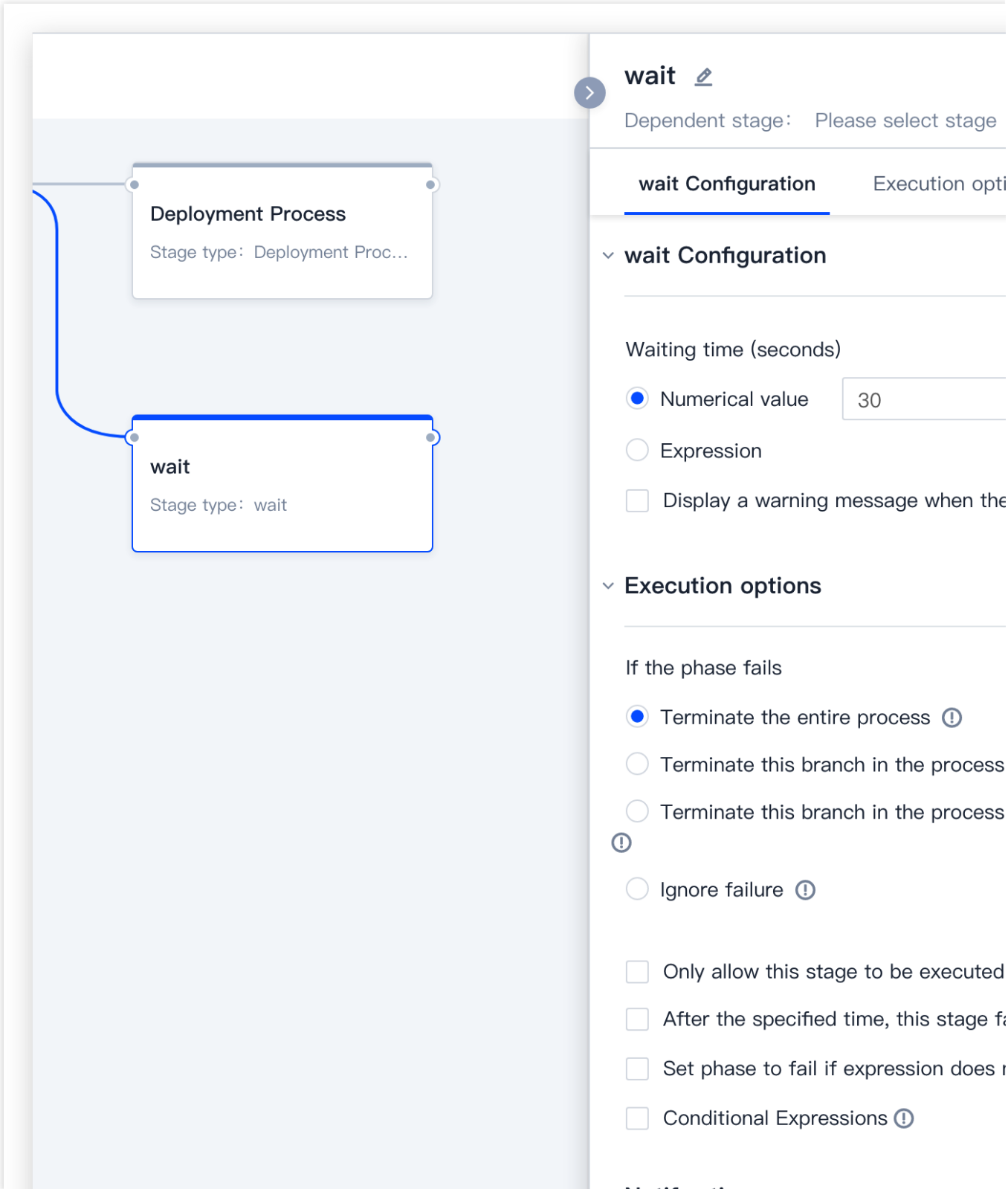
配置选项说明：

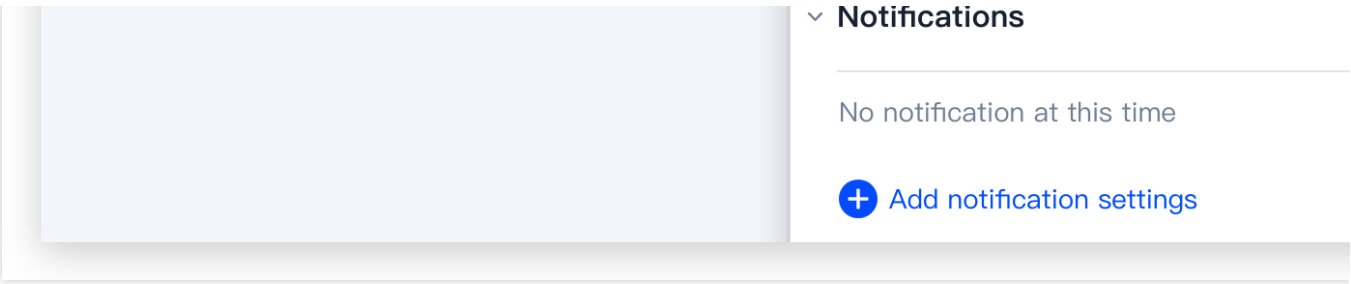
字段	是否必填	说明
应用	是	列出所有具体访问权限的应用
部署流程	是	列出应用下所有的部署流程
是否等待执行结果	否	如果选择等待执行结果，此阶段的状态即为子部署流程的最终执行状态；否则，只要子部署流程开始执行此阶段的状态就会被标记为“成功”。


Deployment Process
Dependent stage: Please select stage
Deployment Process
Stage type: Deployment Proc...
Deployment Process Configuration
Deployment Process Configuration
Application
Please select application
Deployment Process
Please select deployment process
Whether to wait for the execution resu
Execution options
If the phase fails
Terminate the entire process
Terminate this branch in the process
Terminate this branch in the process, a
Ignore failure
Only allow this stage to be executed w
After the specified time, this stage fail
Set phase to fail if expression does nc
Conditional Expressions

等待

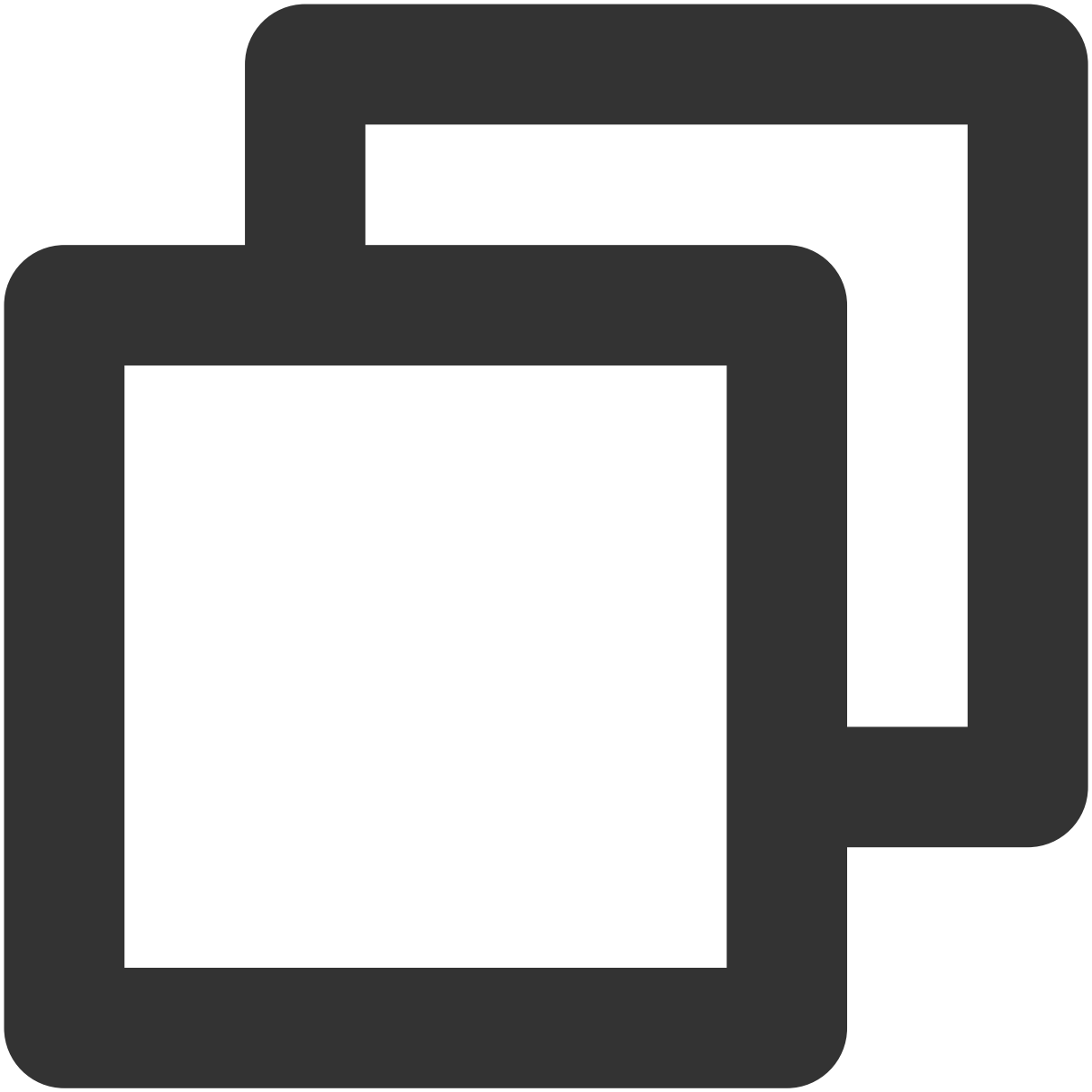
等待一定的时间段后继续执行。在部署流程执行过程中可以手动减少等待时间或直接跳过等待。等待时间支持表达式。



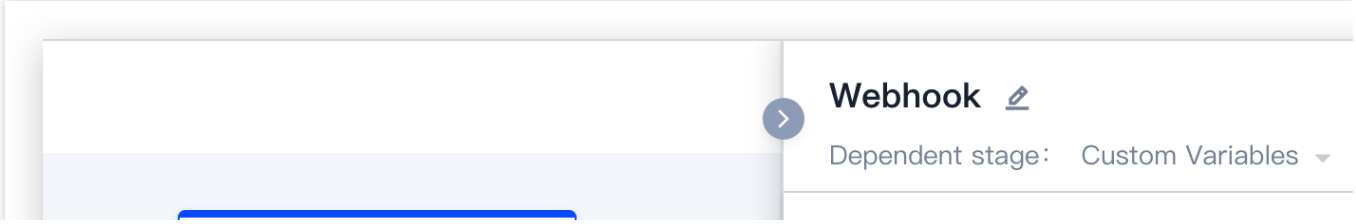


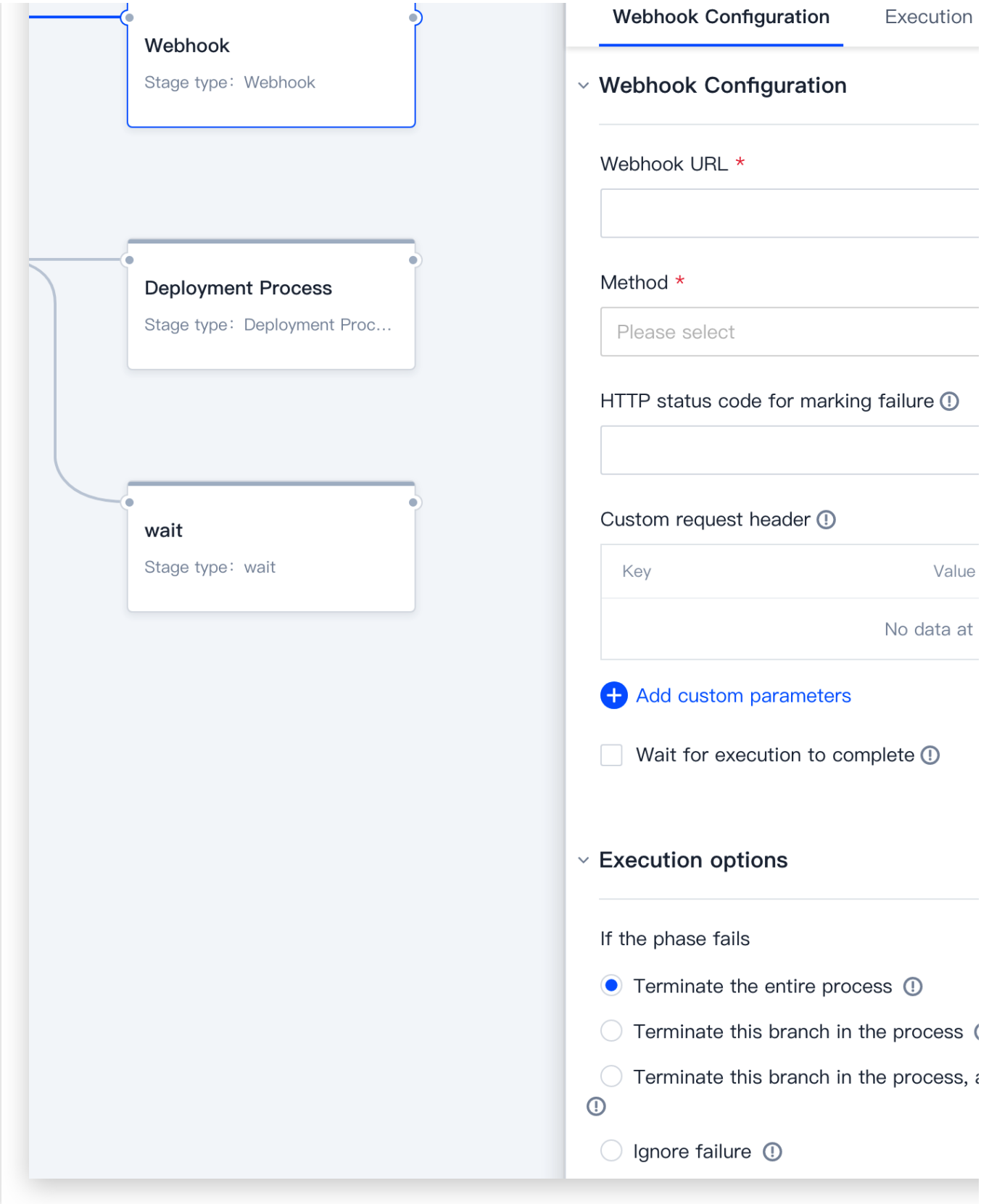
Webhook

支持调用外部系统 API 作为部署流程的阶段。
利用指定 Webhook 的目标 URL 和 HTTP 方法，支持自定义 `header` 和 JSON 格式的 payload。默认情况下，如果 Webhook 调用返回 `2XX` 或 `3XX` 表示阶段执行成功，返回 `4XX` 或 `5XX` 表示执行失败。Webhook 的 URL、payload 调用的最终状态都会展示在部署流程执行详情中。
用户可以在 URL 字段和 payload 中使用部署流程表达式。当阶段执行完成后，阶段上下文的 `Webhook` 对象包含了 payload 内容，可以在后续的部署流程表达式中引用 payload。例如以下表达式可以获取 Webhook 执行的最终状态：



```
${#stage("My Webhook Stage")["context"]["webhook"]["statusCode"]}
```




Webhook
Stage type：Webhook
Deployment Process
Stage type：Deployment Proc...
wait
Stage type：wait
Webhook Configuration
Execution
Webhook Configuration
Webhook URL *
Method *
Please select
HTTP status code for marking failure
Custom request header
Key
Value
No data at
Add custom parameters
Wait for execution to complete
Execution options
If the phase fails
Terminate the entire process
Terminate this branch in the process
Terminate this branch in the process,
Ignore failure

禁用集群

禁用集群意味着让集群保持运行状态但不处理流量。如果需要的话，用户可以指定运行一定数量的服务组而禁用剩下的。

配置选项说明：

云服务（Provider）选择 Kubernetes

字段	是否必填	说明
云服务	是	云服务类型，目前支持 Kubernetes 和 腾讯云
云账号	是	管理资源对象的云账号
Namespaces	是	服务组所属的命名空间
集群	是	服务组所属的集群
禁用选项	是	指定具体的禁用规则

云服务（Provider）选择 Kubernetes

字段	是否必填	说明
云服务	是	云服务类型，目前支持 Kubernetes 和 腾讯云
云账号	是	管理资源对象的云账号
Namespaces	是	服务组所属的命名空间
集群	是	服务组所属的集群
禁用选项	是	指定具体的禁用规则
健康检查	是	执行此任务时，仅参考腾讯云提供的健康检查

集群缩容

可以选择是否对活跃的（正常运行）的服务组缩容；另外支持使指定数量的服务组保持当前规模，将其他服务组缩容。

云服务（Provider）选择 Kubernetes

字段	是否必填	说明
云服务	是	云服务类型，目前支持 Kubernetes 和 腾讯云
云账号	是	管理资源对象的云账号
Namespaces	是	服务组所属的命名空间

集群	是	服务组所属的集群
缩容选项	是	指定具体的缩容选项

云服务（Provider）选择 腾讯云

字段	是否必填	说明
云服务	是	云服务类型，目前支持 Kubernetes 和 腾讯云
云账号	是	管理资源对象的云账号
地域	是	服务组所属的地域
集群	是	服务组所属的集群
健康检查	是	执行此任务时，仅参考腾讯云提供的健康检查

启用服务组

启动服务组即让被禁用的服务组重新处理请求流量。负载均衡器的配置决定了新旧服务组之间的路由规则。启用服务组的同时也启用伸缩容策略。

配置选项说明：

云服务（Provider）选择 Kubernetes

字段	是否必填	说明
云服务	是	云服务类型，目前支持 Kubernetes 和 腾讯云
云账号	是	管理资源对象的云账号
Namespaces	是	服务组所属的命名空间
集群	是	服务组所属的集群
目标服务组	是	指定服务组需满足的匹配规则

云服务（Provider）选择 腾讯云

字段	是否必填	说明
云服务	是	云服务类型，目前支持 Kubernetes 和 腾讯云

云账号	是	管理资源对象的云账号
地域	是	服务组所属的地域
集群	是	服务组所属的集群
目标服务组	是	指定服务组需满足的匹配规则
健康检查	是	执行此任务时，仅参考腾讯云提供的健康检查

禁用服务组

禁用服务组意味着让服务组保持运行状态但不处理流量。另外针对禁用服务组的伸缩容操作也会被禁用。禁用服务组使得在新旧服务组之间做流量切换变得很简单。在阶段启动之前用户必须指定禁用最新、最旧或次新的服务组。

说明：

配置选项参见 启用服务组。

销毁服务组

销毁指定集群的服务组和相应资源。在阶段启动之前用户必须指定销毁最新、最旧或次新的服务组。

配置选项说明：

云服务（Provider）选择 Kubernetes

字段	是否必填	说明
云服务	是	云服务类型，目前支持 Kubernetes 和 腾讯云
云账号	是	管理资源对象的云账号
Namespaces	是	服务组所属的命名空间
集群	是	服务组所属的集群
目标服务组	是	指定服务组需满足的匹配规则

云服务（Provider）选择 腾讯云

字段	是否必填	说明
云服务	是	云服务类型，目前支持 Kubernetes 和 腾讯云
云账号	是	管理资源对象的云账号

地域	是	服务组所属的地域
集群	是	服务组所属的集群
目标服务组	是	指定服务组需满足的匹配规则

调整服务组规模

以当前服务组规模的百分比或指定数量来调整服务组大小。支持如下调整策略：
扩容：增大目标服务组的规模
缩容：减小目标服务组的规模
扩容至相对最大规模：将目标服务组规模扩容至与当前集群中最大服务组一致
调整为特定规模：将目标服务组调整至特定规模

腾讯云类型

Bake

从指定的软件包 Bake 云服务器镜像，Bake 是指创建云服务器镜像的过程。CODING 部署控制台基于 Hashicorp 的 Packer 抽象出 Baker 阶段，提供了默认的 Packer 模板和基本的云服务器镜像，以帮助用户快速入门。
需要注意的是，如果 Spinnaker 检测到不需要进行新的 Bake，则跳过 Bake 过程。 CODING 部署控制台会根据 Bake 阶段参数（基础操作系统，版本化的软件包等）为每次 Bake 生成唯一的键（key）。如果软件包或 Bake 阶段参数发生了更改，则会触发新的 Bake 操作。 要更改默认行为并在部署流程每次运行时重新 Bake 镜像，请在阶段阶段配置中勾选 `Rebake` 。详情请参见 Hashicorp Packer。

部署

通过指定部署策略部署提前准备好的镜像。CODING 持续部署提供了部分内置的部署策略，如：红黑（蓝绿）部署、Highlander 部署。您还可以选择对已有服务组无侵入式的部署方式，或者创建自定义的部署策略。

回滚集群

回滚集群中一个或多个区域的实例。
配置选项说明：

字段	是否必填	说明
云账号	是	腾讯云账号
地域	是	集群所属的地域
集群	是	指定需要回滚的集群
健康检查	是	执行此任务时，仅参考腾讯云提供的健康检查

克隆服务组

将已有服务组的所有属性克隆到新服务组（依赖镜像、容器等）。在创建新服务组时可以选择覆盖克隆服务组的任意属性。

Shrink Cluster

指定保留一定数量的最新或规模最大的服务组，将其他的服务组全部删除。用户可以选择是否删除不满足指定条件的活跃（正常运行）服务组

Modify Scaling Process

暂停/恢复 扩缩容操作

Kubernetes 类型

Bake (Manifest)

使用诸如 Helm 这样的模板渲染器 Bake 资源清单。

部署（Manifest）

包含两个主要的步骤：
指定要部署的 mainfest
在 manifest 中指定需要覆盖的制品（如 `Docker 镜像` ）

配置 manifest

根据需求，有两个方法可以指定 manifest：

静态：直接在部署流程中指定

动态：运行时使用绑定的制品

不管使用哪种方式，都需提前选择 `Deploy (Manifest)` 阶段：

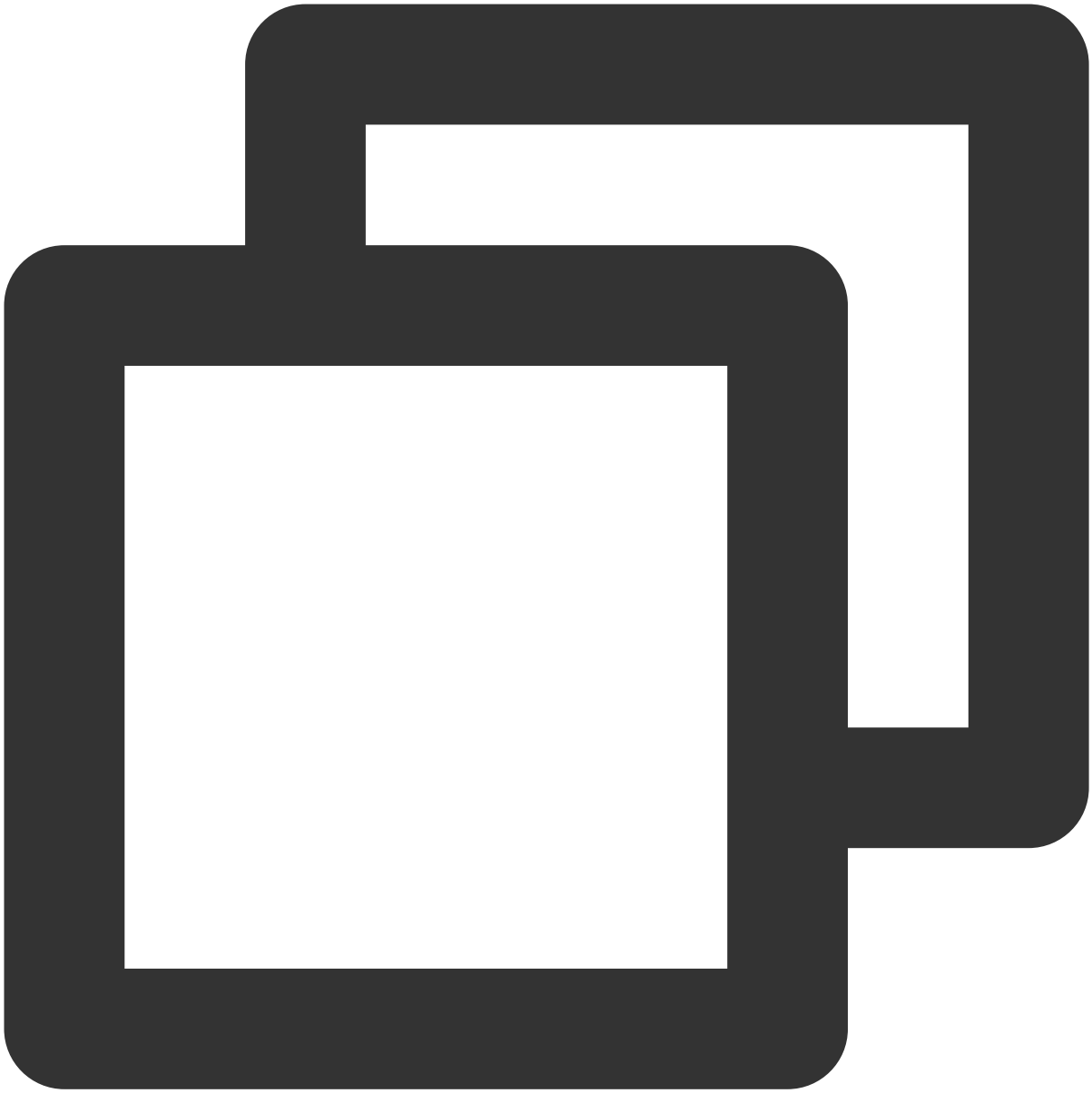
配置静态的 manifest

如果您提前知道即将部署的资源对应的 manifest（即使不知道制品的版本），那么可以在 `Deployment(Manifest)` 阶段配置中直接提供 manifest 纯文本内容。

说明：

选择 `Text` 类型后，用户在文本框直接编辑 YAML 文件内容。

如果是使用 JSON 定义的部署流程，对应的内容如下：



```
{
  "name": "Deploy my manifest",    // human-readable name
  "type": "deployManifest",        // tells orchestration engine what to run
  "account": "nudge",              // account (k8s cluster) to deploy to
  "cloudProvider": "kubernetes",
  "source": "text",
  "manifest": {
                                   // manifest contents go here
  }
}
```

配置动态的 manifests

如果制品没有存储在部署流程仓库中，或者当一个阶段需要部署多种制品时，可以通过绑定制品来配置 manifest。CODING 持续部署的制品允许用户引用任何远程的可部署资源。`Deploy(Manifest)` 阶段引用的制品必须是包含 Manifest 定义的文本文件，这样的文本文件可能存在于 GitHub 仓库或 GCS。详情请参见 部署流程详细设置。

假设您已经在上游阶段声明了 `Expected Artifacts`，那么可以在 `Deploy(Manifest)` 阶段引用：

说明：

Manifest Source 勾选 `Artifact` 后，用户可以选择部署上游提供的制品。请确保云账号（Account）拥有下载制品的权限。

上游阶段可能会匹配到多个制品，例如我们可以配置正则表达式 `.*\\yml`，将所有 `yml` 文件作为制品。`Deploy(Manifest)` 阶段执行时会部署所有匹配到的 `yml` 文件。

覆盖制品

通常当我们对基础设施做部署更新操作时，大多数的变更都是涉及 Docker 镜像或 ConfigMap 中的 `flag`。因此，CODING CD 针对这些资源类型的变更提供了优秀的支持。

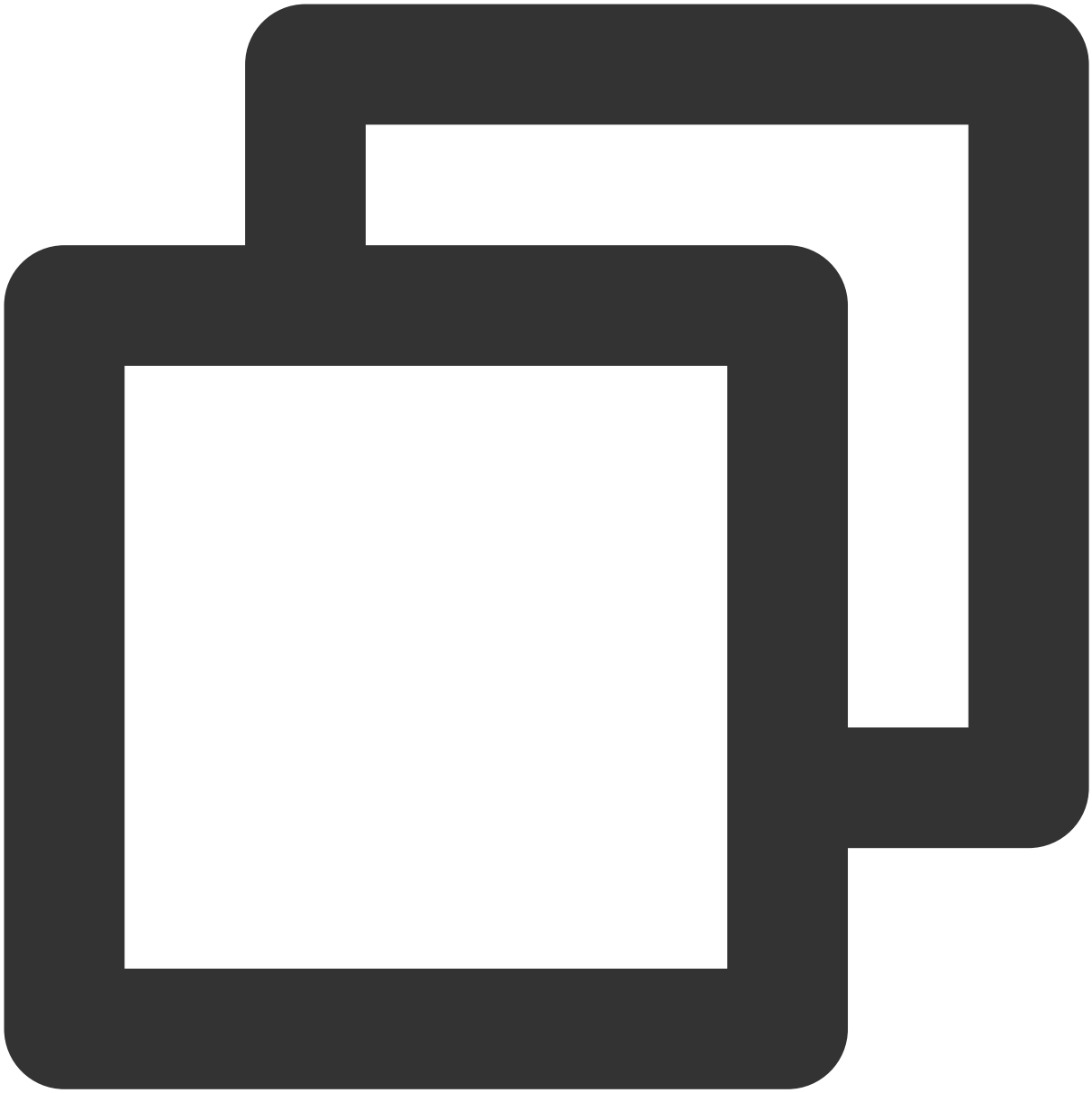
Docker 镜像

Kubernetes ConfigMap

Kubernetes Secret

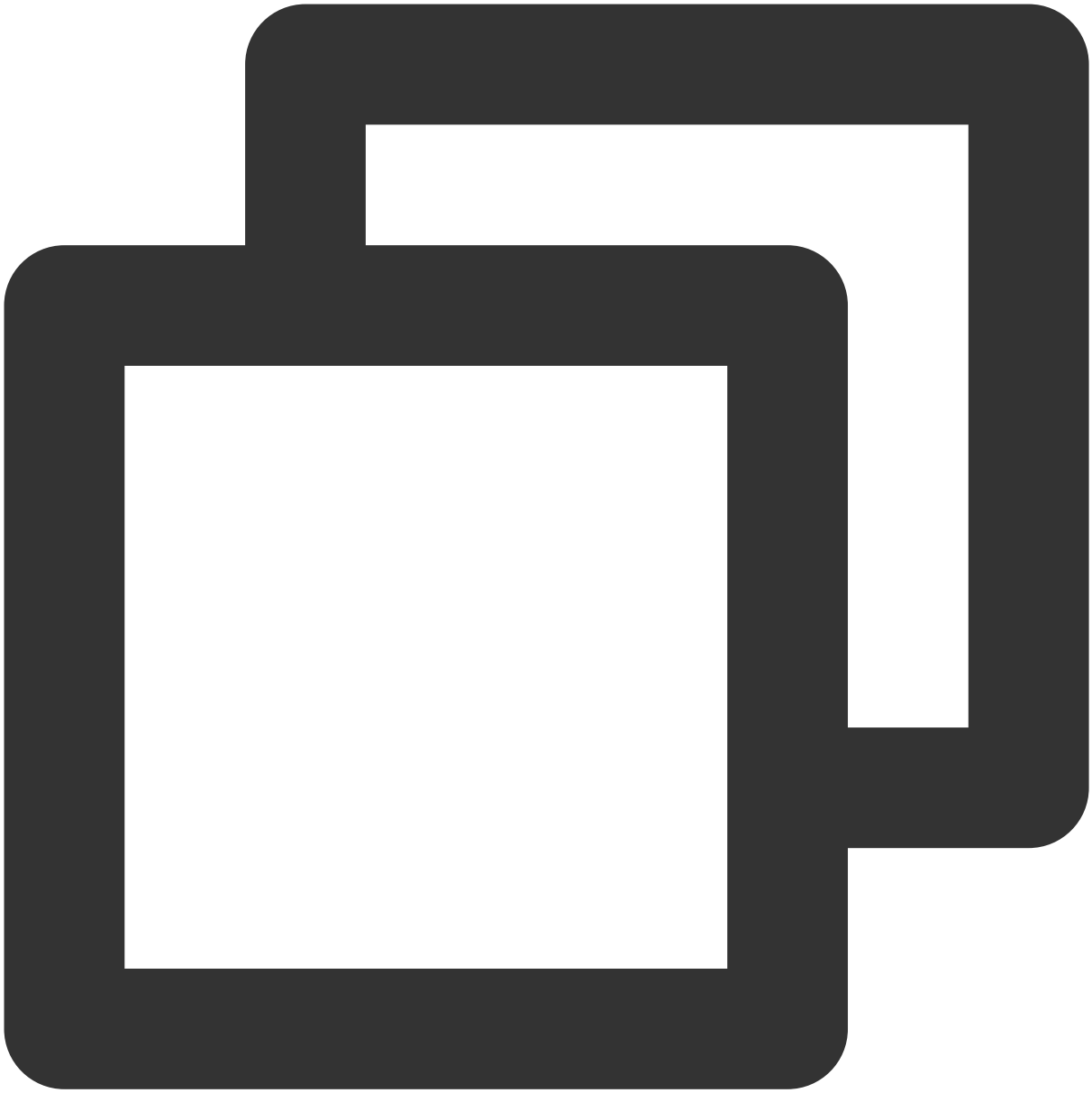
如果上游阶段的部署流程上下文中存在这些资源对象，CODING CD 将会自动地尝试将它们注入到正在部署的 manifest 文件中。

例如，假设 Docker 镜像仓库类型的触发器触发部署流程执行，并且触发器携带了镜像 `gcr.io/my-project/my-image`，其 digest 值为 `sha256:c81e41ef5e...`。在部署流程中，您配置了 manifest 内容如下的部署阶段。



```
# ... rest of manifest
  containers:
  - name: my-container
    image: gcr.io/my-project/my-image
# rest of manifest ...
```

因为部署流程是由 `Docker 镜像内容变更` 所触发，所以部署流程编排引擎会将 Docker 镜像制品连同部署阶段中的 manifest 一起派发给 `clouddriver` 组件处理。最终得到部署的 manifest 内容如下：



```
# ... rest of manifest
  containers:
  - name: my-container
    image: gcr.io/my-project/my-image@:sha256:c81e41ef5e...
# rest of manifest ...
```

为了确保部署阶段获取到正确的制品，您可以强制阶段绑定所有需要的制品，如果绑定失败，阶段将启动失败。以下配置的含义是 Docker 镜像 `gcr.io/my-project/my-image` 必须被绑定到 manifest，否则阶段将会执行失败：

启用（Manifset）

启用 Kubernetes 对象。

配置选项说明：

`Selector` 选择 `静态指定目标`

字段	是否必填	说明
云账号	是	管理资源对象的云账号
Namespace	是	资源对象所在的命名空间
Selector	是	通过名称静态指定删除的目标资源
Kind	是	资源对象类型
Name	是	资源对象名称（例如 replicaSet 类型的资源 nginx-deployment-5dfd77bbf9）

`Selector` 选择 `动态选择目标`

字段	是否必填	说明
云账号	是	管理资源对象的云账号
Namespace	是	资源对象所在的命名空间
Selector	是	通过名称静态指定删除的目标资源
Kind	是	资源对象类型
集群	是	资源对象名称（例如 replicaSet 类型的资源 nginx-deployment-5dfd77bbf9）
Target	是	选择规则匹配资源对象

删除（Manifest）

删除通过资源清单创建的 Kubernetes 对象。

配置选项说明：

`Selector` 选择 `静态指定目标`

字段	是否必填	说明
云账号	是	管理资源对象的云账号
Namespace	是	资源对象所在的命名空间

Selector	是	通过名称静态指定删除的目标资源
Kind	是	资源对象类型
Name	是	资源对象名称（例如 replicaSet 类型的资源 nginx-deployment-5dfd77bbf9）

`Slector` 选择 动态选择目标

字段	是否必填	说明
云账号	是	管理资源对象的云账号
Namespace	是	资源对象所在的命名空间
Selector	是	通过集群和 Target 字段动态选择资源对象
Kind	是	资源对象类型
Cluster	是	资源对象所在的集群
Target	是	选择规则匹配资源对象

`Selector` 选择 使用标签匹配目标

字段	是否必填	说明
云账号	是	管理资源对象的云账号
Namespace	是	资源对象所在的命名空间
Selector	是	通过指定标签规则匹配资源对象
Kind	是	资源对象类型
Labels	是	不填写规则表示匹配所有满足指定类型的资源对象

设置选项说明：

字段	是否必填	说明
级联删除	否	勾选后，表示删除此资源对象管理的所有资源对象（例如：Replica Set 管理的所有 Pods），不勾选可能会产生孤儿资源
Grace Period	否	（可选）指定资源对象正式终止的时间，将会覆盖 manifest 设置的时间（如果已设置）

触发器配置

最近更新时间：2024-01-03 11:55:53

本文为您详细介绍在 CODING 持续部署中部署流程的触发器配置。

前提条件

使用 CODING 项目管理的前提是，您的腾讯云账号需要开通 CODING DevOps 服务。

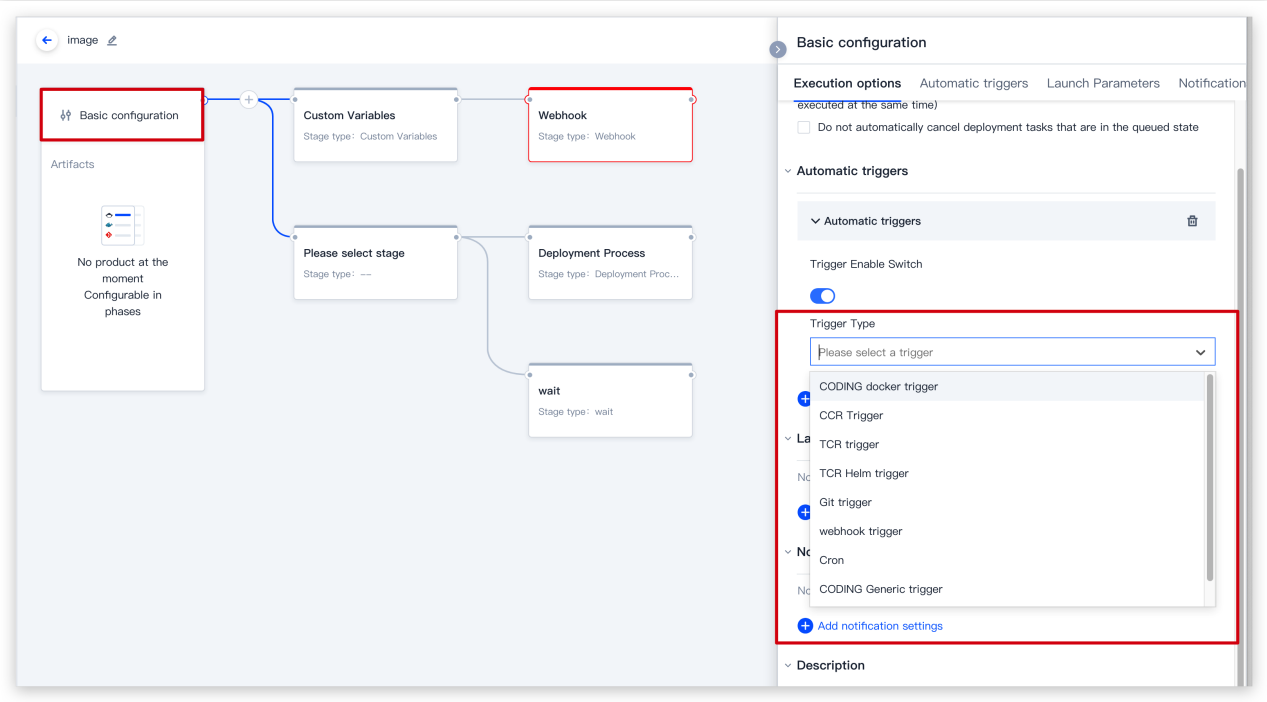
进入项目

1. 登录 CODING 控制台，单击**立即使用**进入 CODING 使用页面。
2. 单击工作台首页左侧的

，进入持续部署控制台。

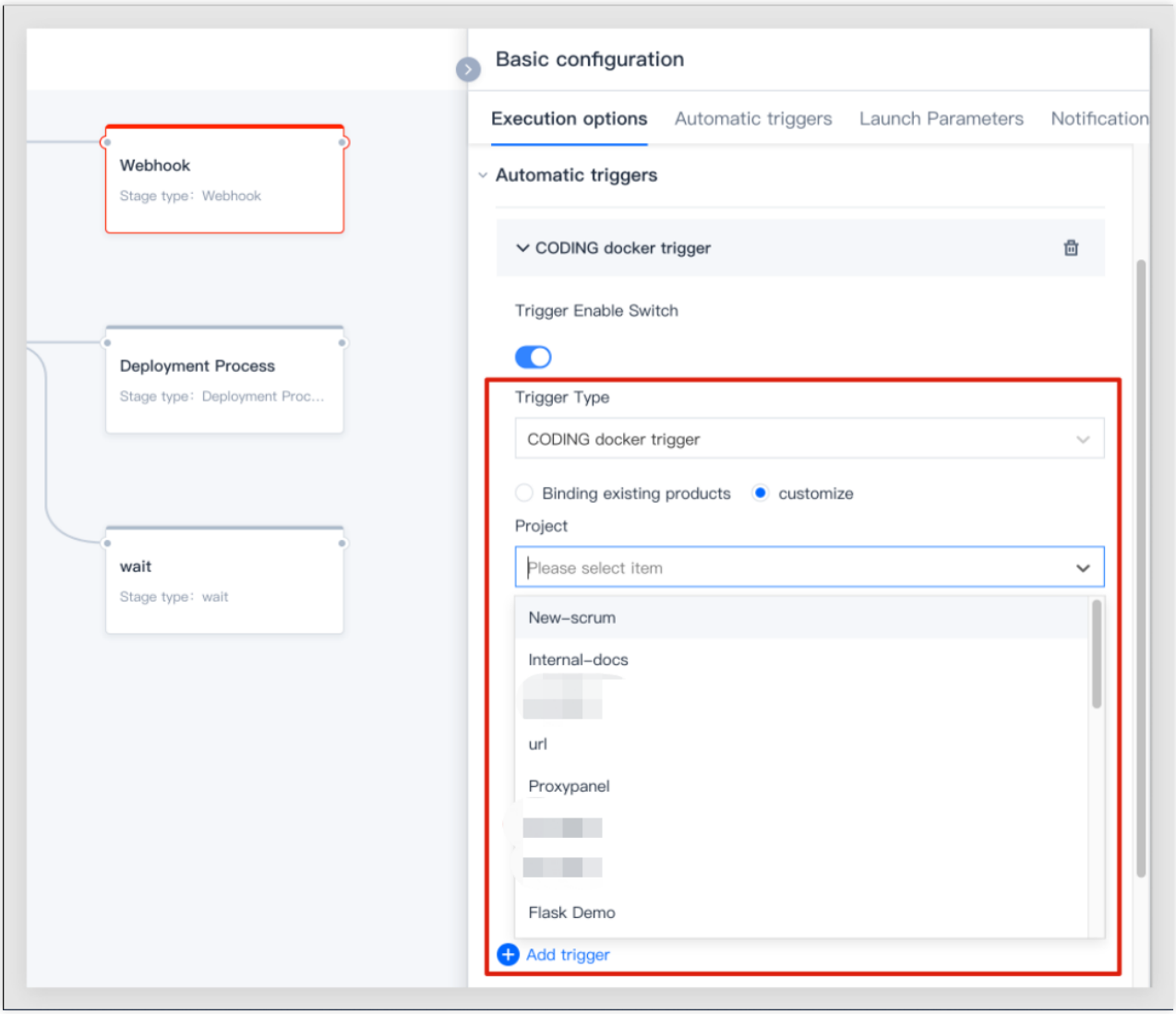
功能介绍

CODING 部署控制台支持多种自动触发条件，使之能够与 CODING 中的流水线相互匹配。目前支持 Docker 仓库触发器、TCR 个人版仓库触发器、TCR 企业版仓库触发器、Git 仓库触发器等触发条件。



Docker 仓库触发

通过配置 Docker 仓库触发器，能够监听制品仓库下的更新情况。若有镜像更新，将自动触发 CD 的部署流程。



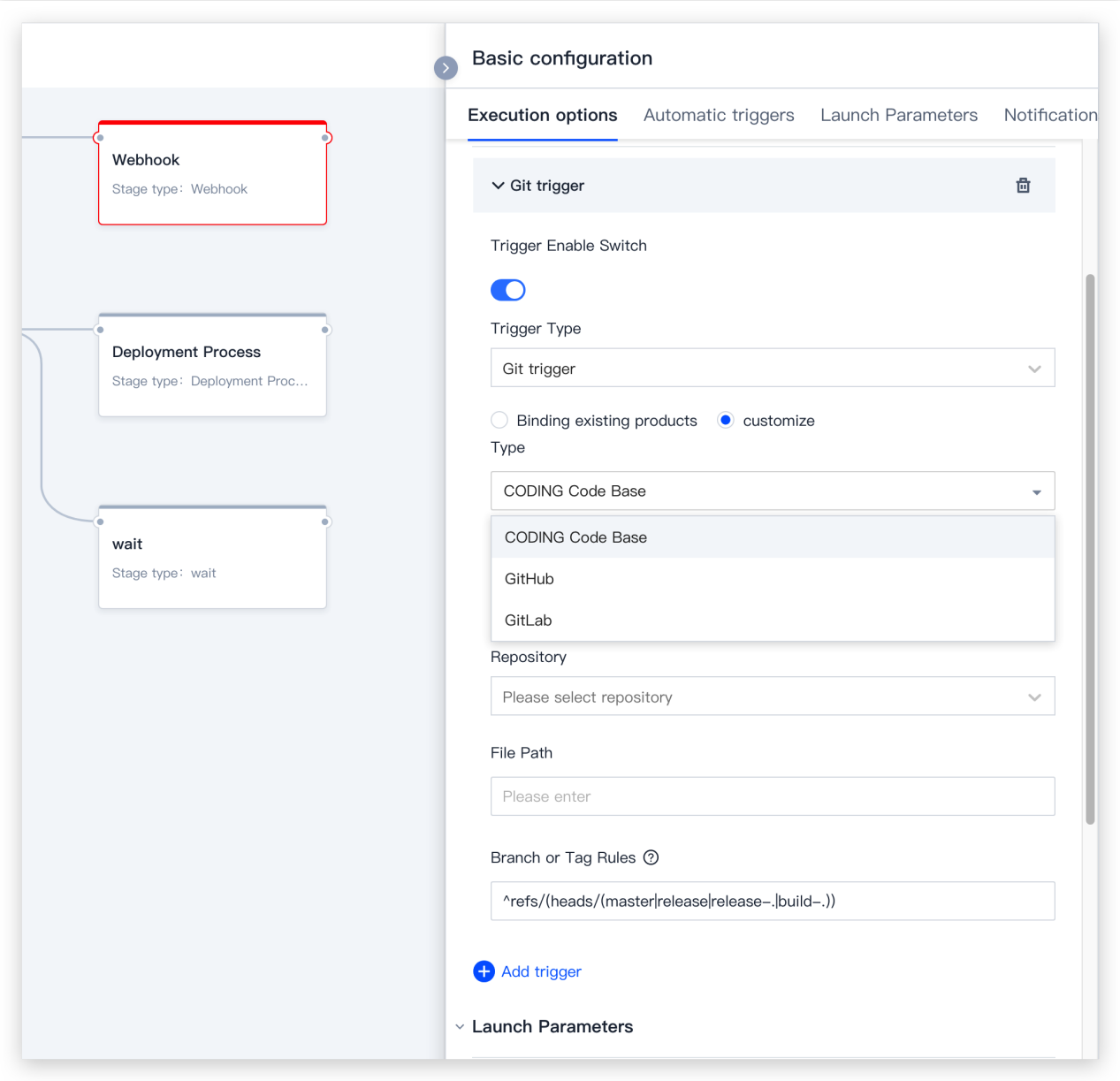
Git 仓库触发

支持 CODING 代码库、GitHub、GitLab 三种类型的 Git 仓库。

字段	说明
仓库类型	支持 CODING 代码库、GitHub、GitLab 三种类型的 git 仓库
项目	列出登录账号加入的所有项目
仓库	列出项目下的所有代码仓库
分支或标签规则	支持正则表达式，留空或 `.*` 表示不对分支或标签做限制

CODING 代码库

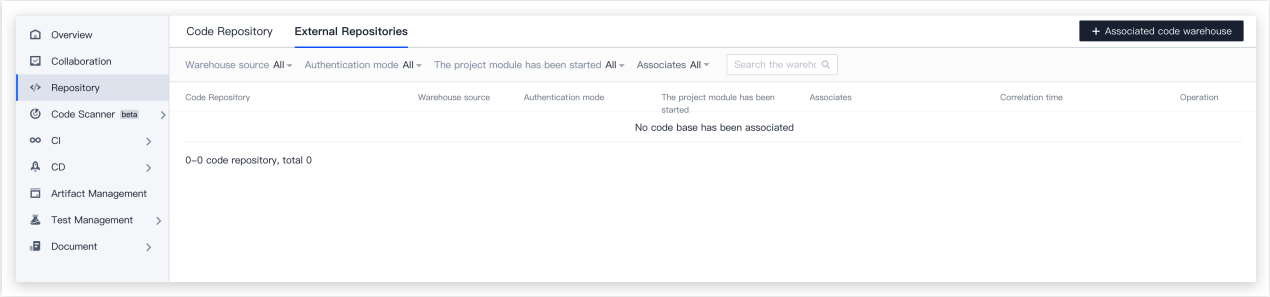
配置 cd-demo 项目下的 cd-demo 代码仓库作为触发器，分支或标签规则 release.* 表示所有以 release 开头命名的分支或 tag 才会触发部署流程执行。



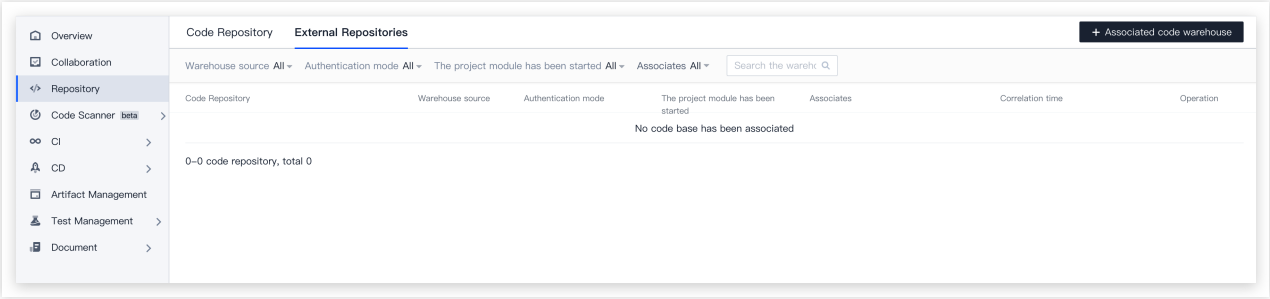
Github

Github 代码库的支持需要提前在项目设置中关联代码库，具体步骤如下：

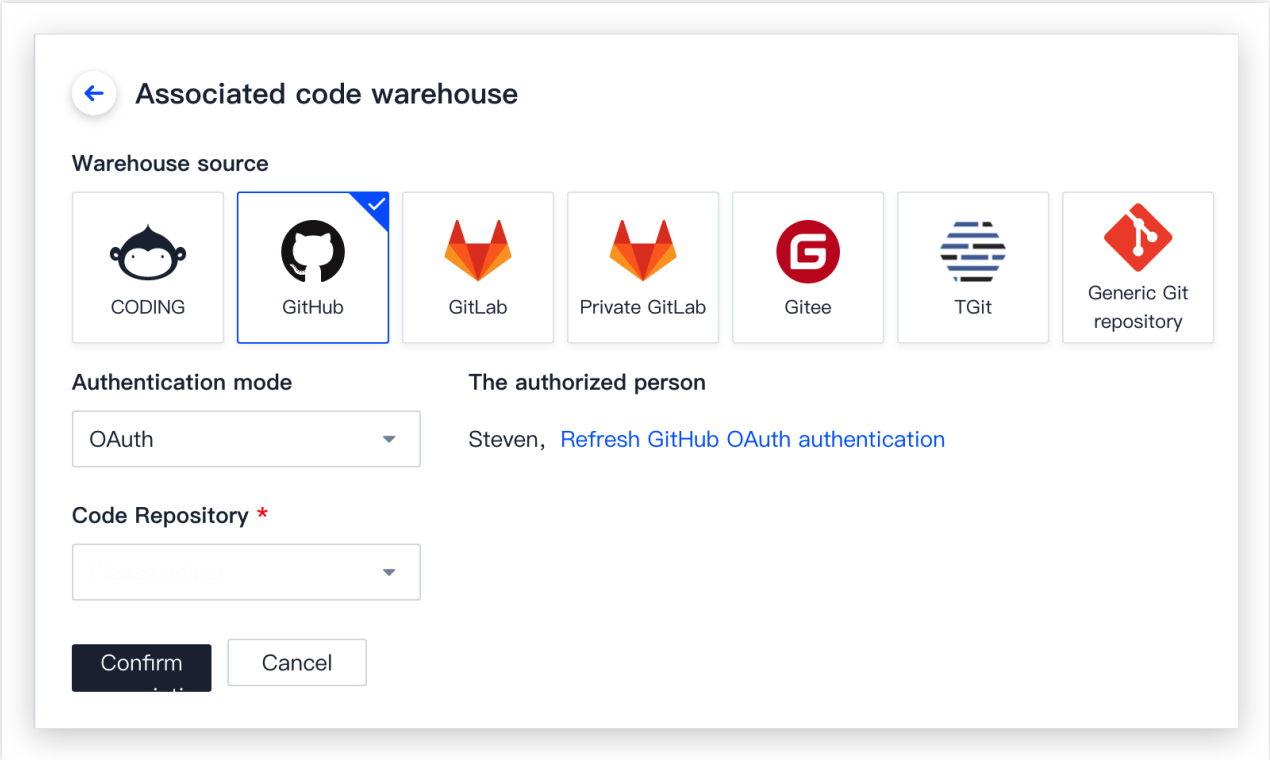
1. 进入项目概览页，单击**代码库**。



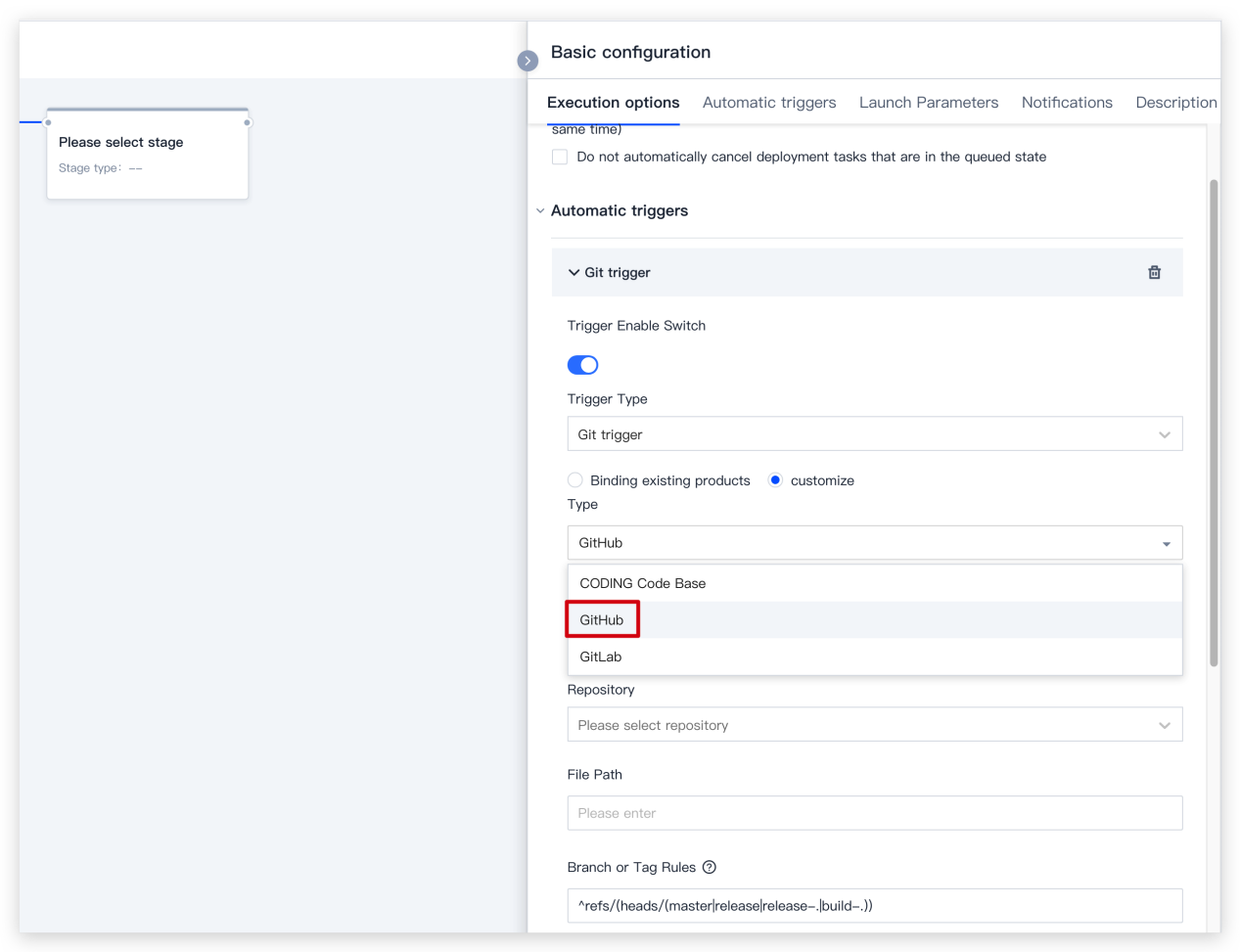
2. 单击右上角的关联代码库按钮。



3. 使用OAuth跳转到GitHub关联的帐户，并选择名称下的代码库。

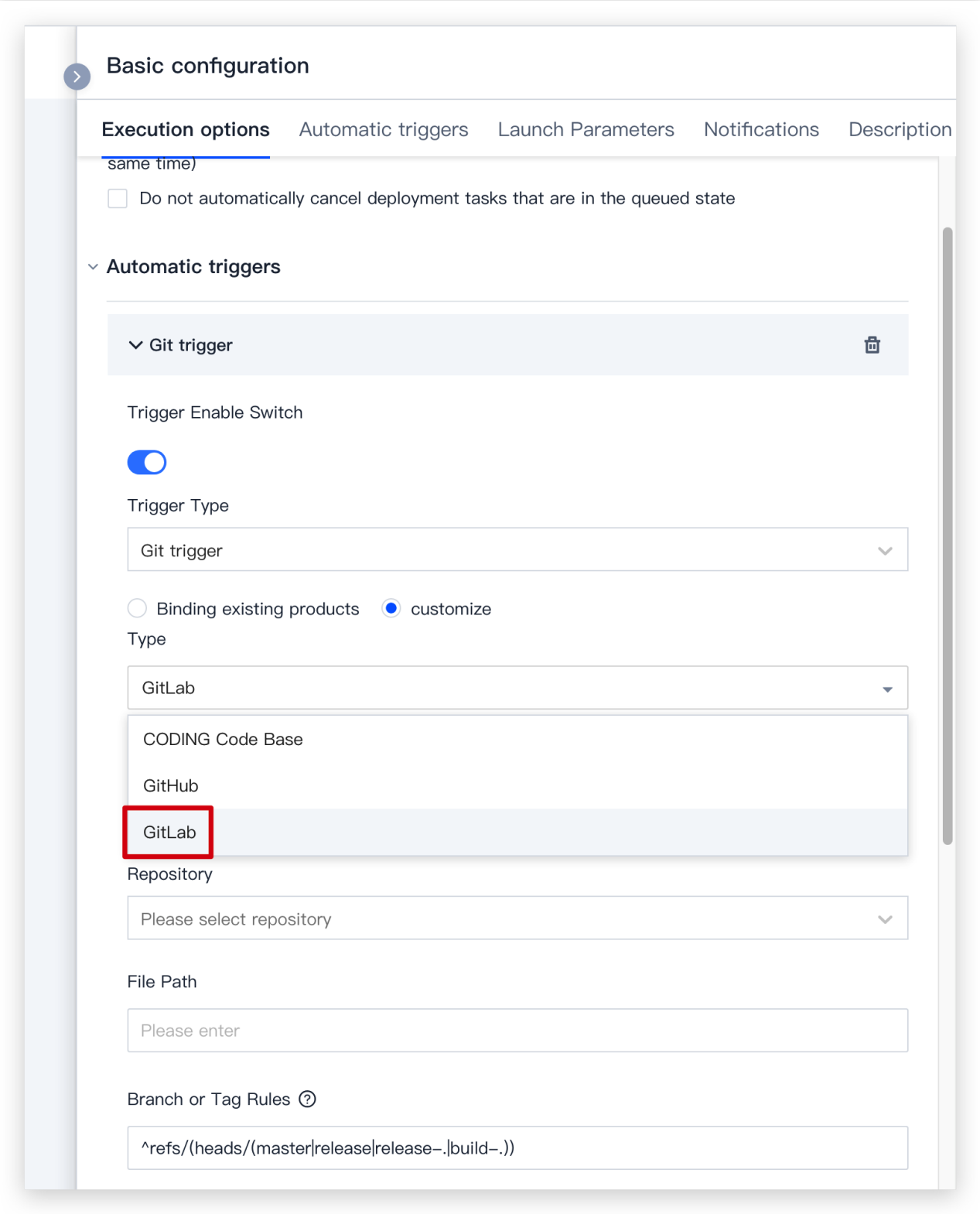


4. 关联完成后返回**基础配置 > 执行选项**，选择**GitHub**仓库类型。



GitLab

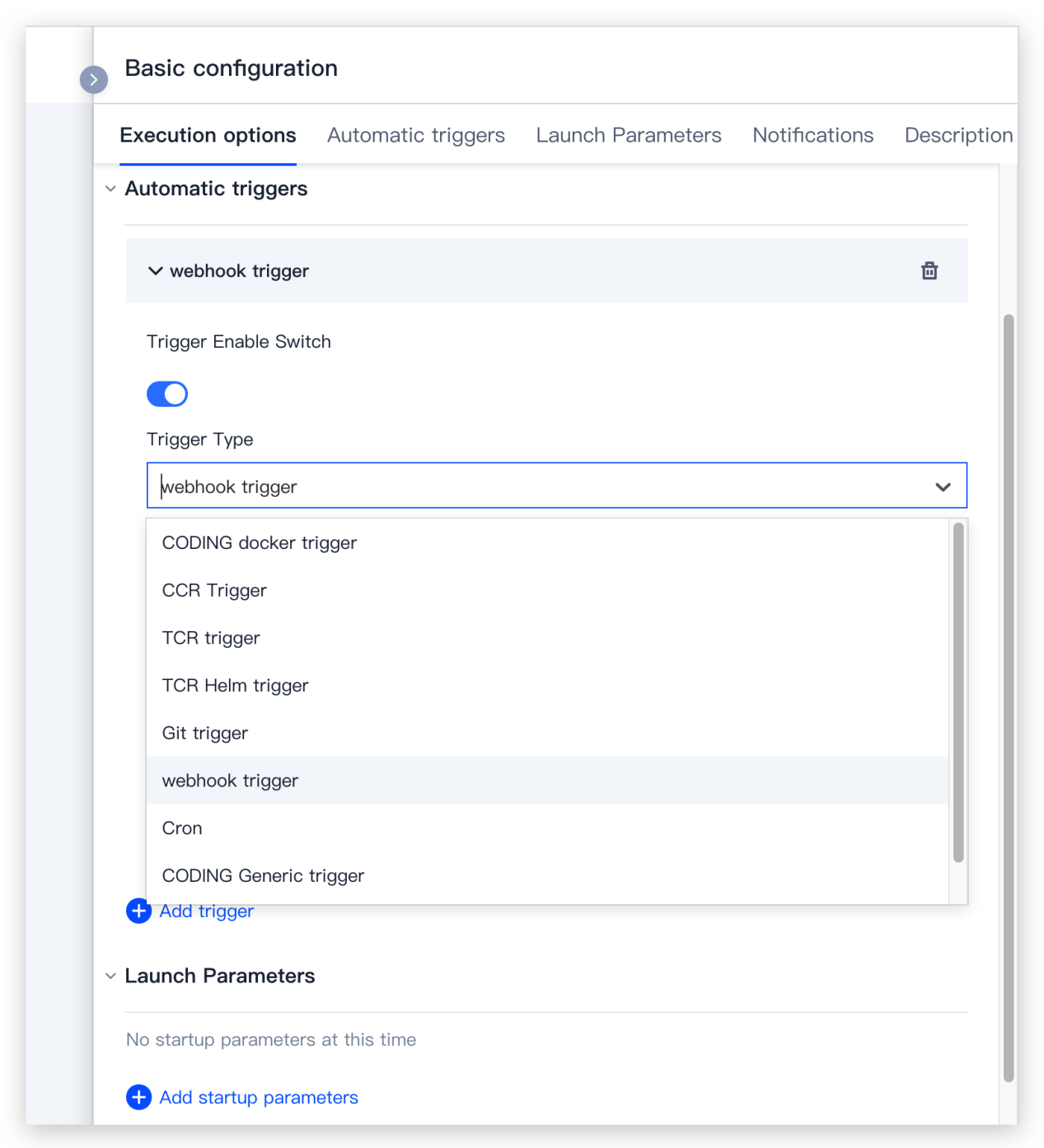
需要提前关联 GitLab 账号后（关联步骤请参见 Github），在**基础配置 > 执行选项**中选择**GitLab**仓库类型。



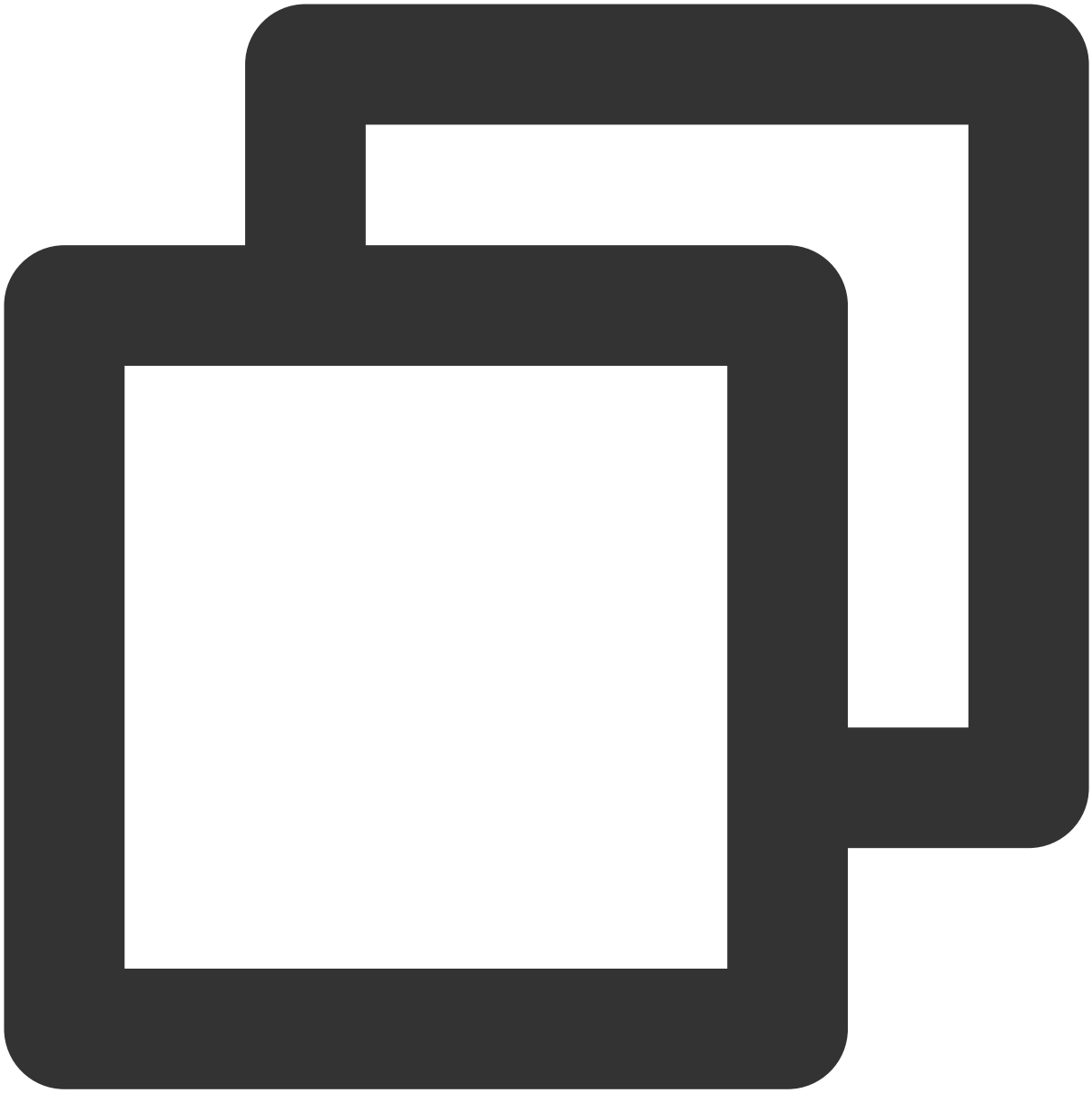
Webhook 触发

选择 Webhook 触发，将生成全局唯一的 URL 触发地址，`Payload Constraints` 定义 `Payload` 请求内容必须提供的参数，支持正则表达式，留空或 `.*` 表示不对 Key 的 Value 值做限制。

Payload Constraints：如果需要使用特定内容的 payload 触发 Webhook，可以在 `Payload Constraints` 一栏添加 `key/value` 键值对，当部署流程收到 Webhook 请求时，将会 `payload` 内容进行校验，`value` 支持正则表达式。



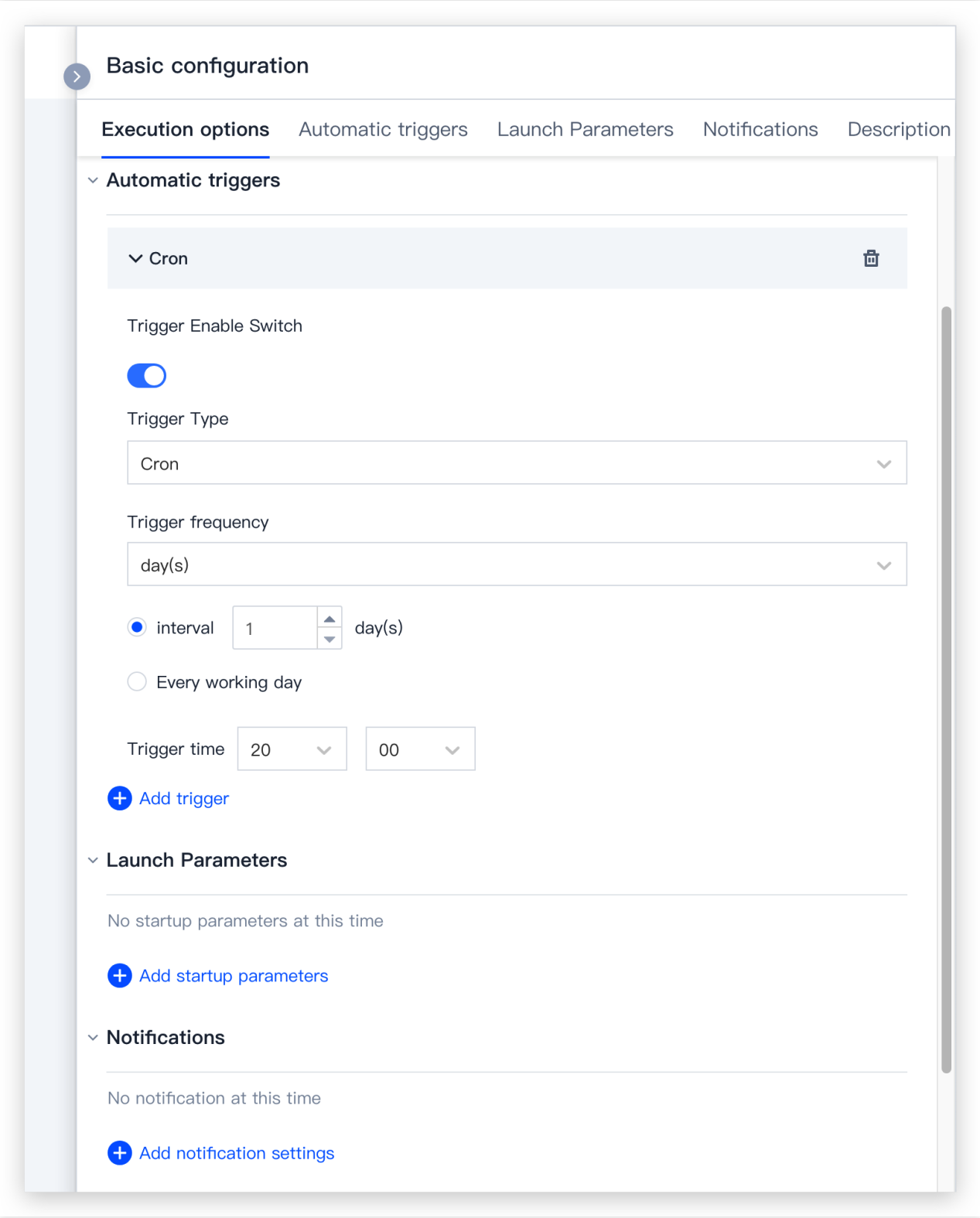
使用场景举例：部署流程 Webhook 地址对公网开放，但只有提供认证凭据才能触发部署流程执行。如下 `Payload` 请求会成功触发部署流程执行：



```
curl --location --request POST 'http://codingcorp.coding.com/api/cd/webhooks/webhoo
--header 'Content-Type: application/json' \\
--data-raw '{"secret": "faiM4&KqJTTuEy8J"}'
```

定时触发

例如每天晚上 8 点触发部署流程：


Basic configuration
Execution options
Automatic triggers
Launch Parameters
Notifications
Description
Automatic triggers
Cron
Trigger Enable Switch
Trigger Type
Cron
Trigger frequency
day(s)
interval
1
day(s)
Every working day
Trigger time
20
00
Add trigger
Launch Parameters
No startup parameters at this time
Add startup parameters
Notifications
No notification at this time
Add notification settings

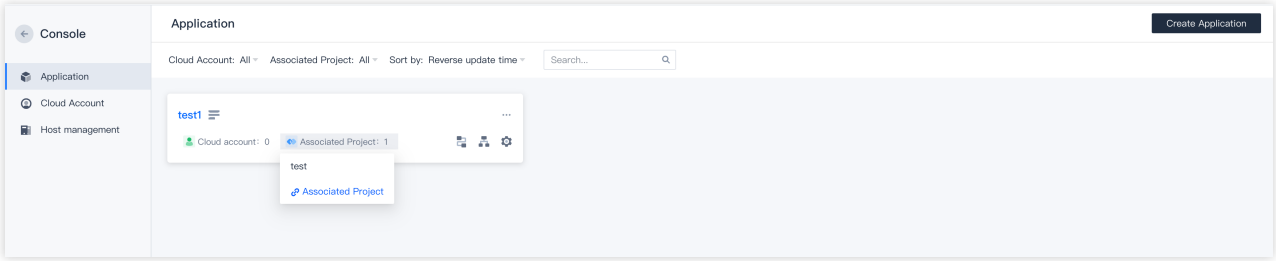
部署方式
自动发布 Docker 制品时触发

最近更新时间：2024-01-03 11:57:28

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工作流，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。

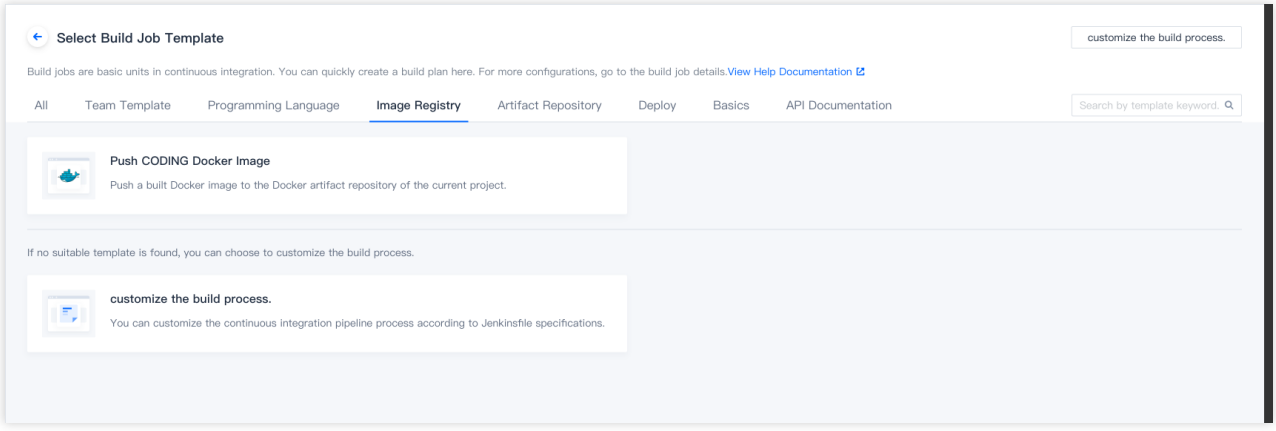
步骤1：应用与项目关联

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

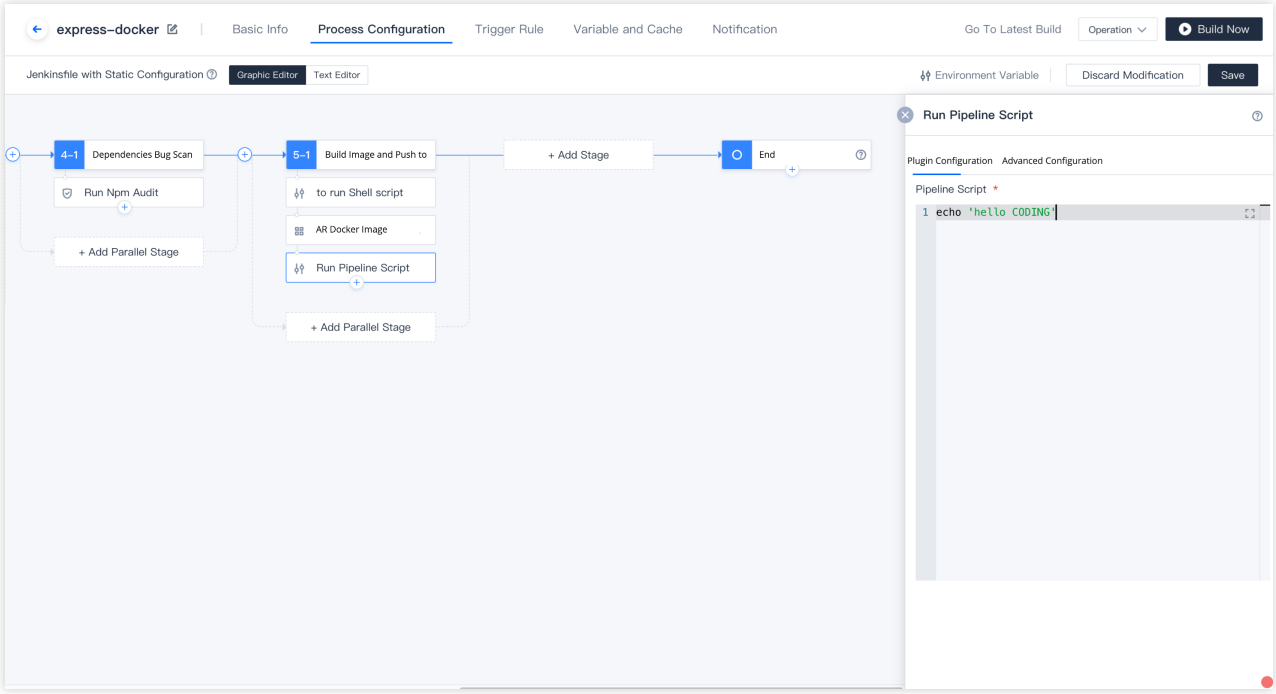


步骤2：配置持续集成

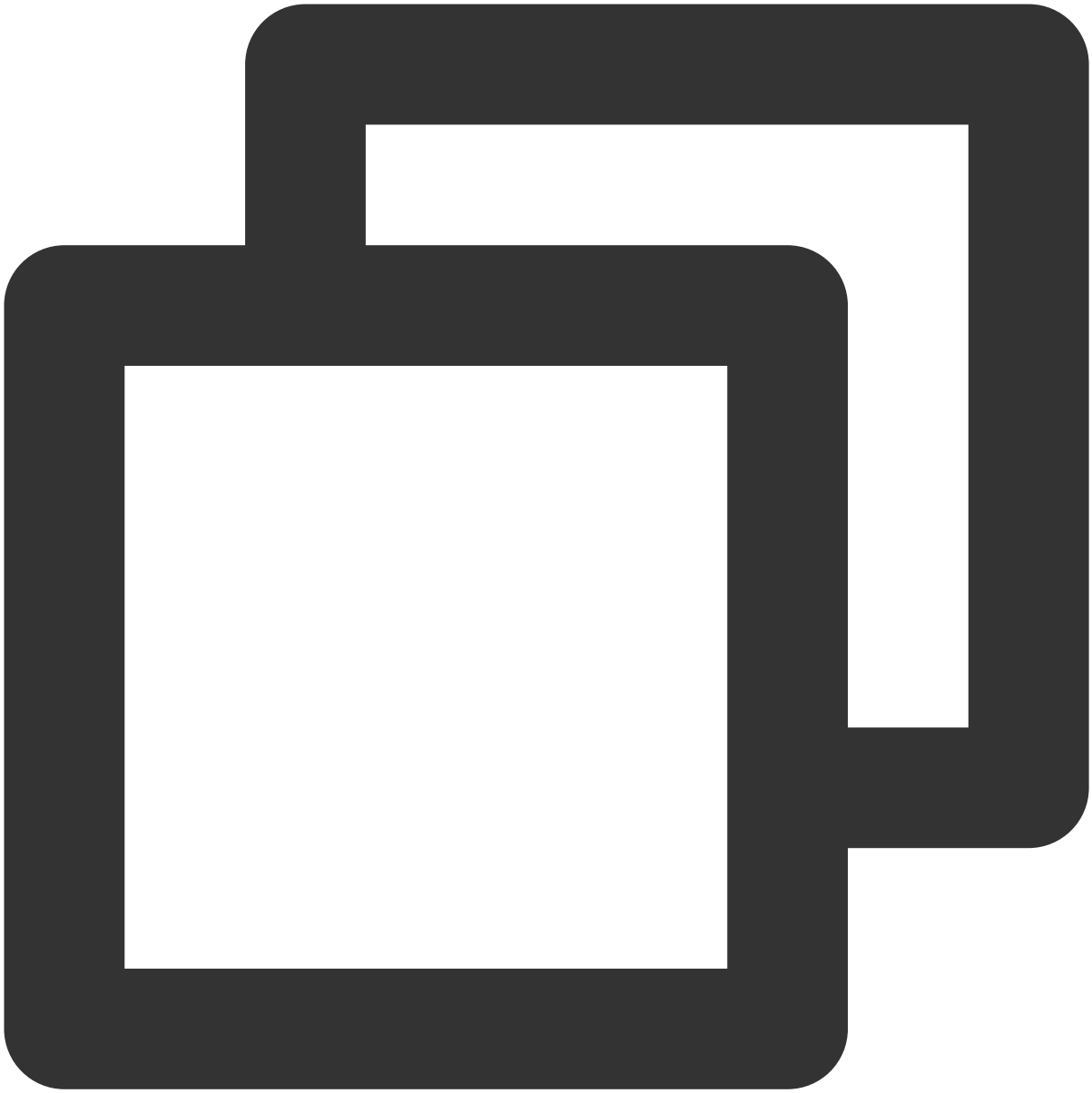
此步骤使用持续集成将制品推送至制品仓库。您可以通过持续集成计划模板创建，或直接编写 Jenkinsfile 手动增加此阶段。



在持续集成流程中手动增加此阶段：



Jenkinsfile 参考



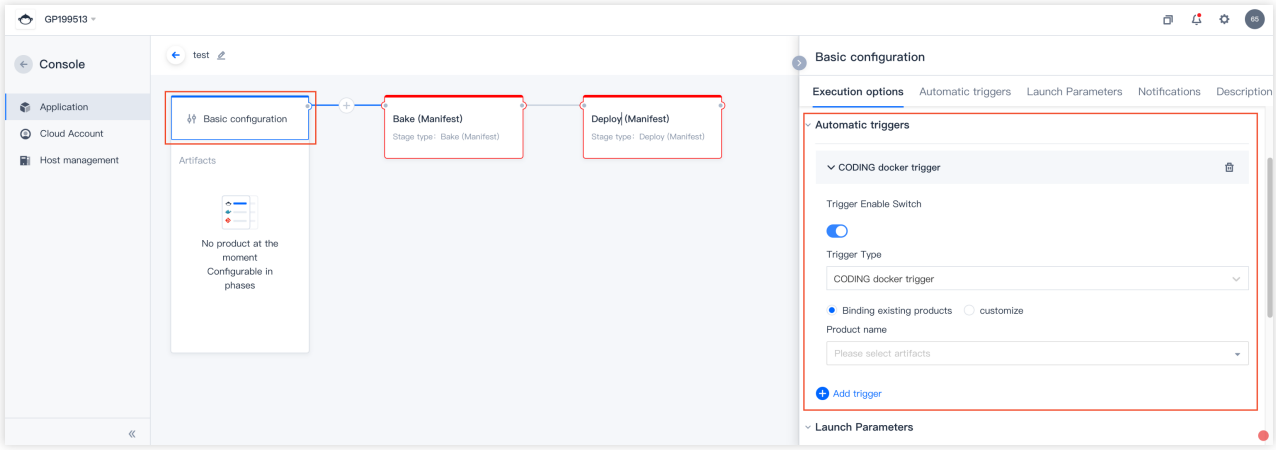
```
stage('部署到远端 Kubernetes 集群') {
      steps {
        cdDeploy([
          deployType: 'PATCH_IMAGE',
          application: "${CCI_CURRENT_TEAM}",
          pipelineName:  "${PROJECT_NAME}-${CCI_JOB_NAME}-${CD_CREDENTIAL_INDEX}",
          image: "${CODING_DOCKER_REG_HOST}/${CODING_DOCKER_IMAGE_NAME}:${DOCKER_IM
          cloudAccountName: "${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}",
          ])
        }
      }
```

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，将自定触发部署流程；选择**自定义**能够监听其他项目的制品仓库更新情况。

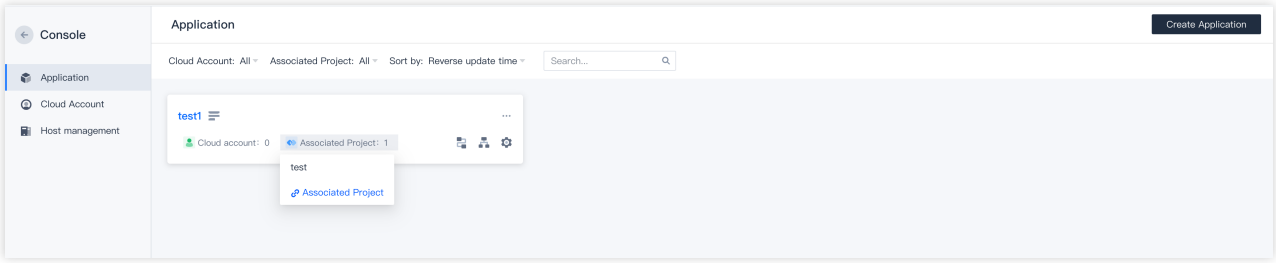
除了通过 CODING Docker 制品更新触发，您还可以通过 Git 仓库或定时器触发此部署流程。



在构建计划中添加部署阶段

最近更新时间：2024-01-03 11:58:05

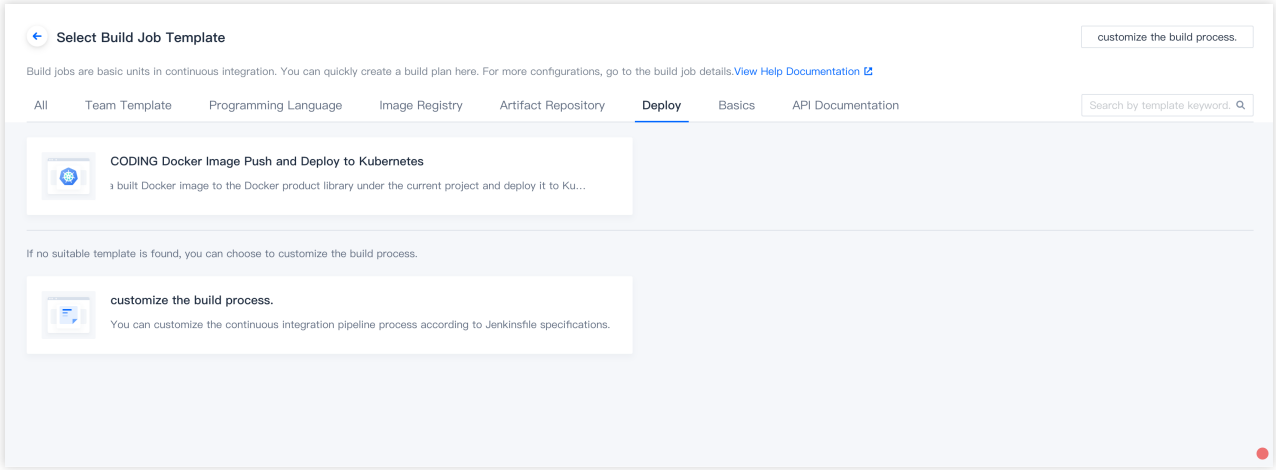
在持续集成中触发部署时请提前前往**部署控制台**，将应用与项目关联。



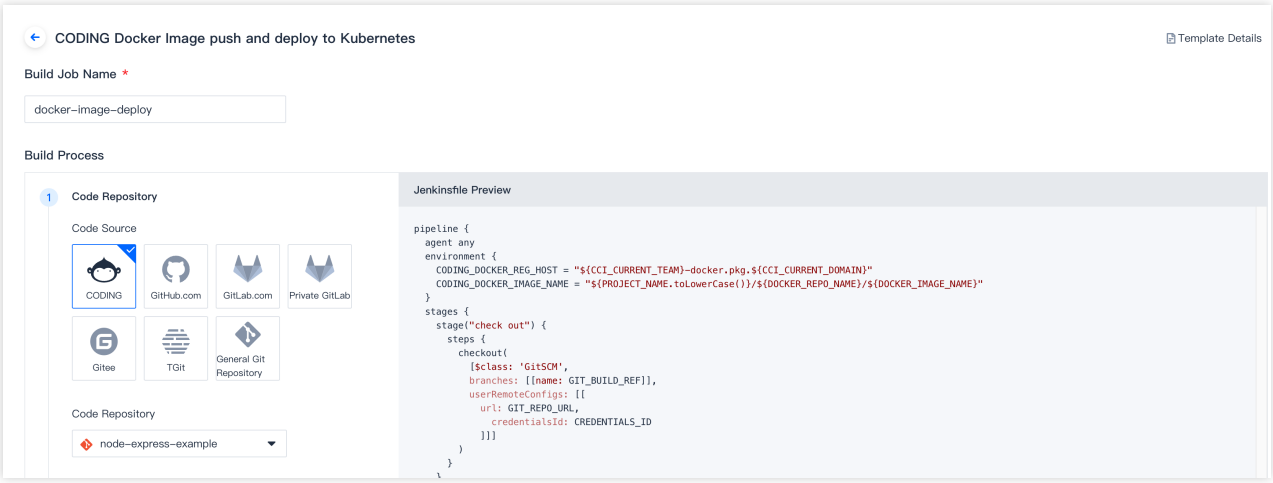
本文给出了两种设置方法，您可以按照需求有选择性阅读。
直接使用构建计划模板
在已有构建计划中添加部署阶段

使用构建计划模板

注意：
请在部署控制台中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。



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。



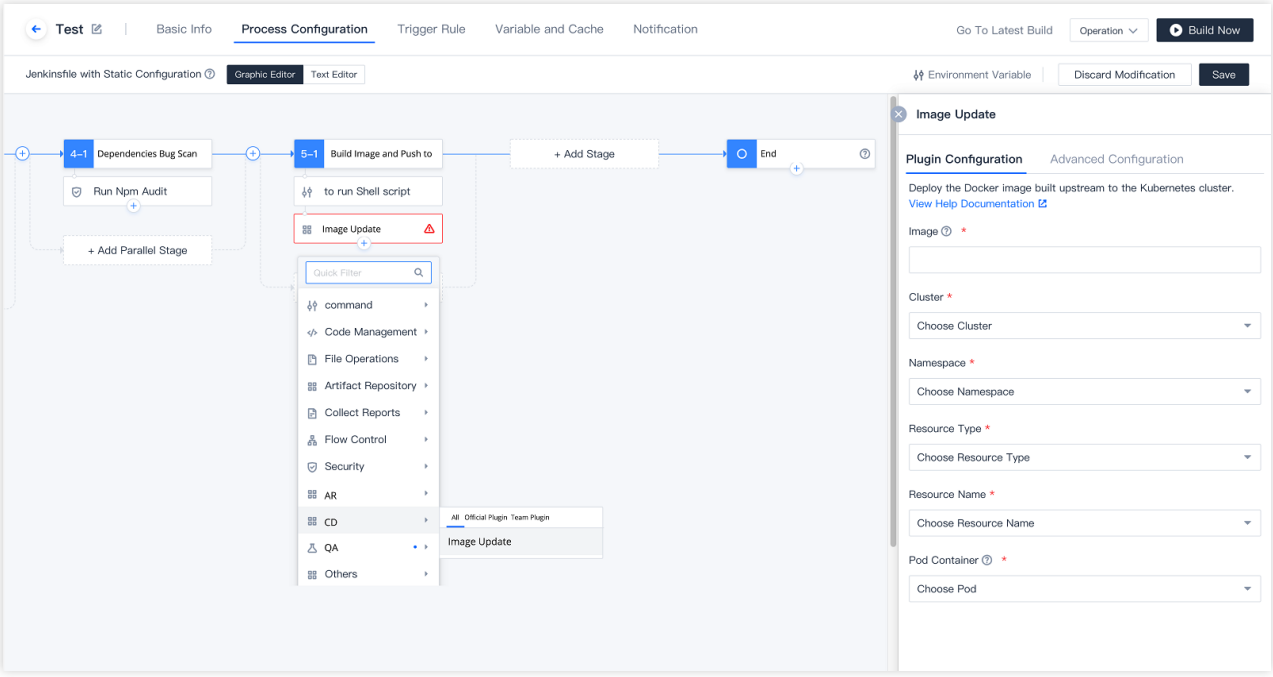
设置完成后，运行持续构建计划即可完成自动发布。

添加部署阶段

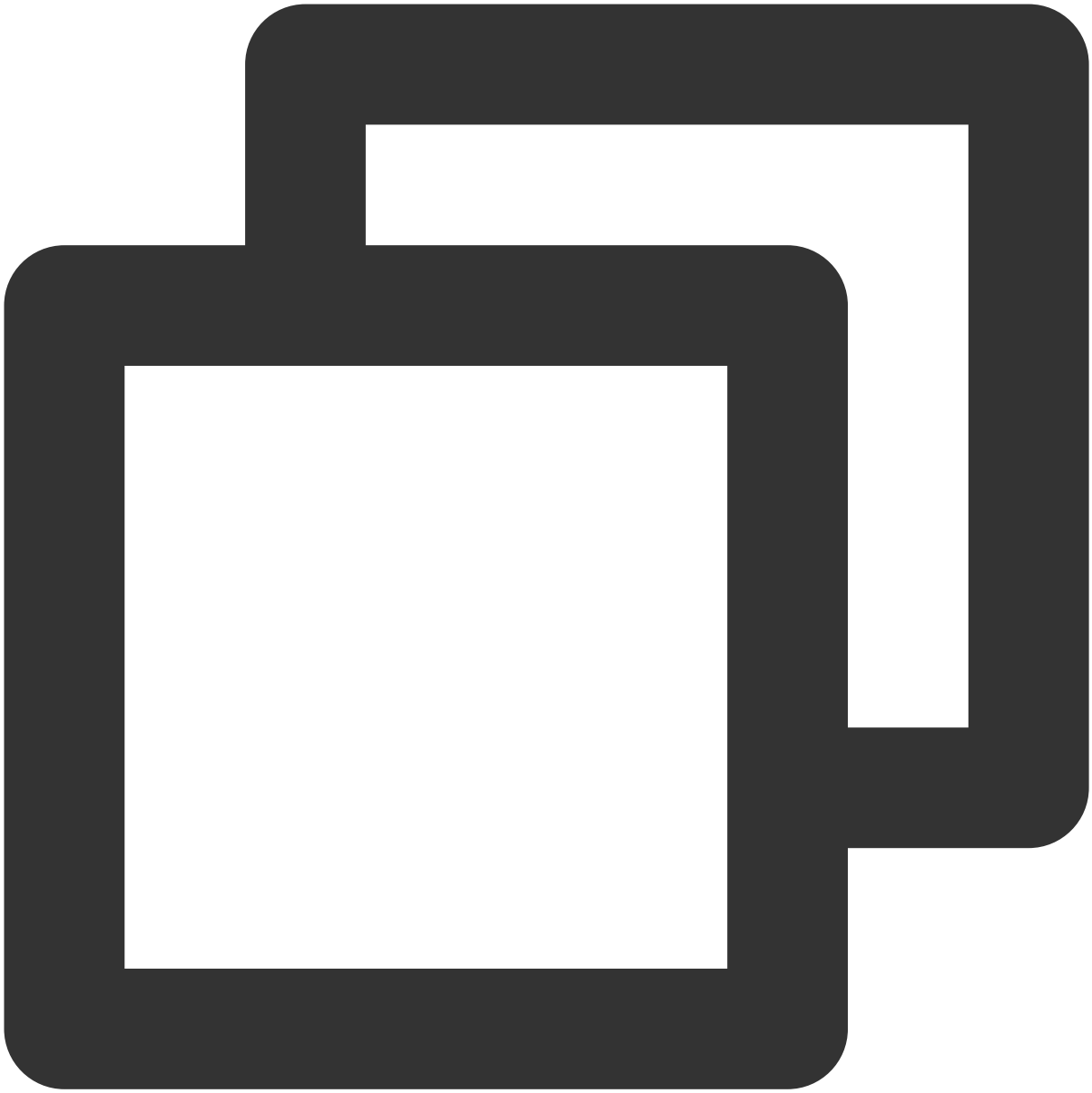
在此方法中您可以在**构建计划 > 流程配置**中使用编辑器或填写命令添加部署阶段。

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。



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}")

        }
```

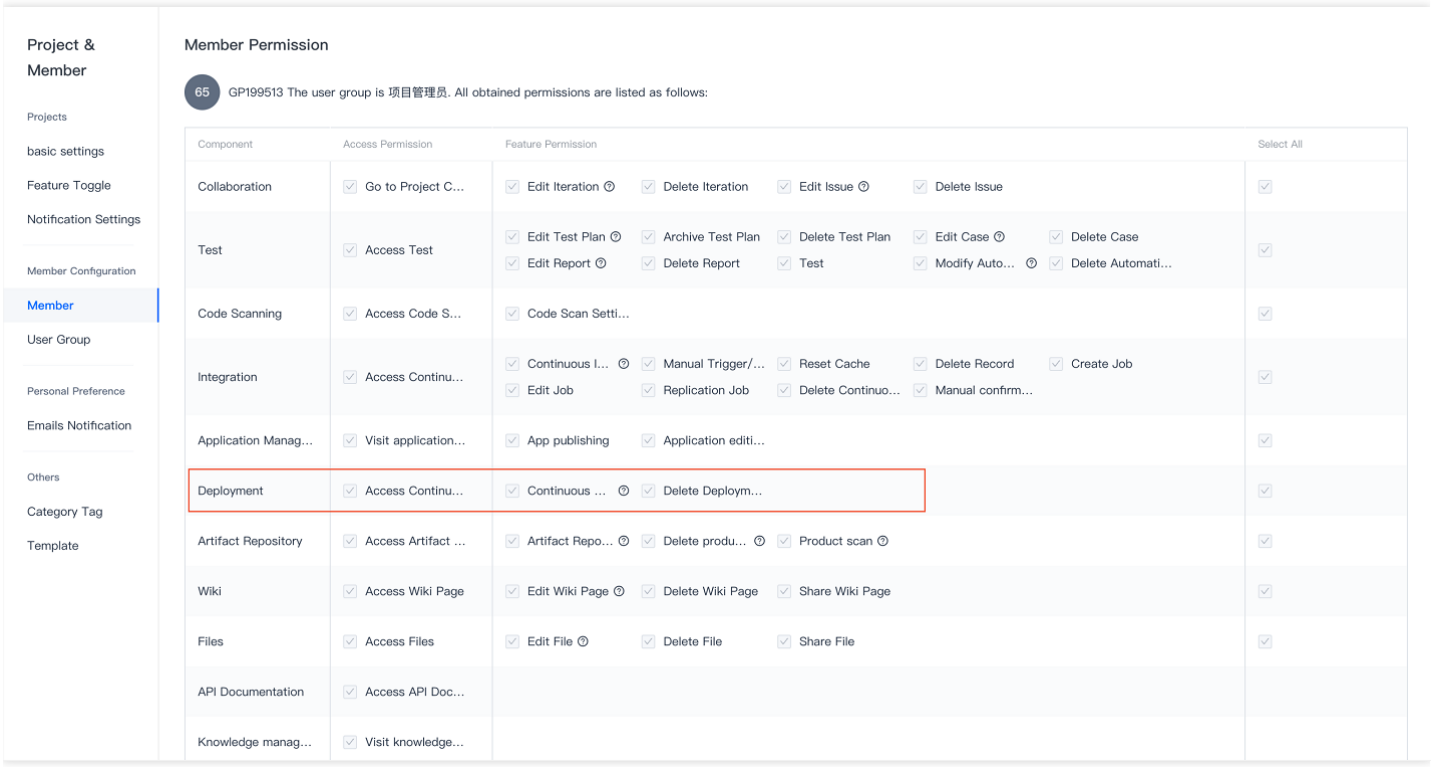
```
        }
```

手动提交发布单

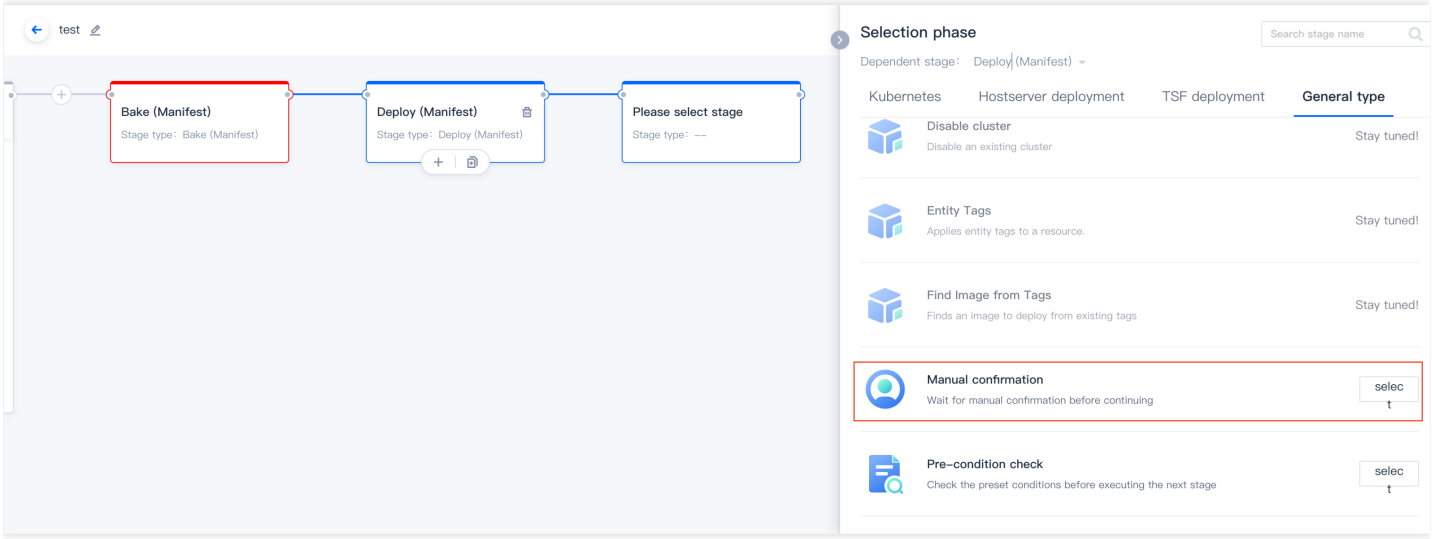
最近更新时间：2022-03-09 15:29:00

新建发布单

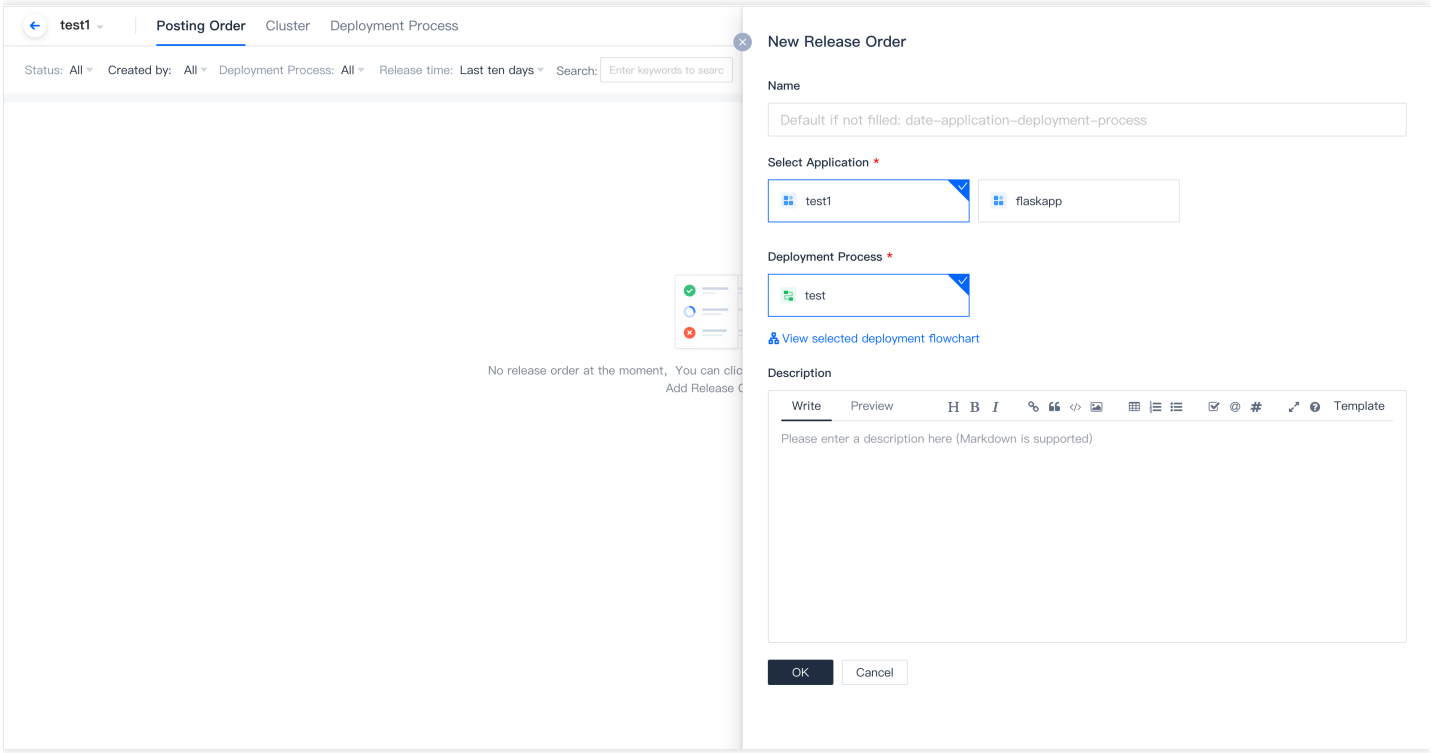
我们推荐赋予**开发**用户组访问与持续部署管理权限。详情请参见 权限控制 。



设置后，开发具备提交发布单的权限，不具备前往部署控制台修改部署配置的权限。运维组用户还可以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。

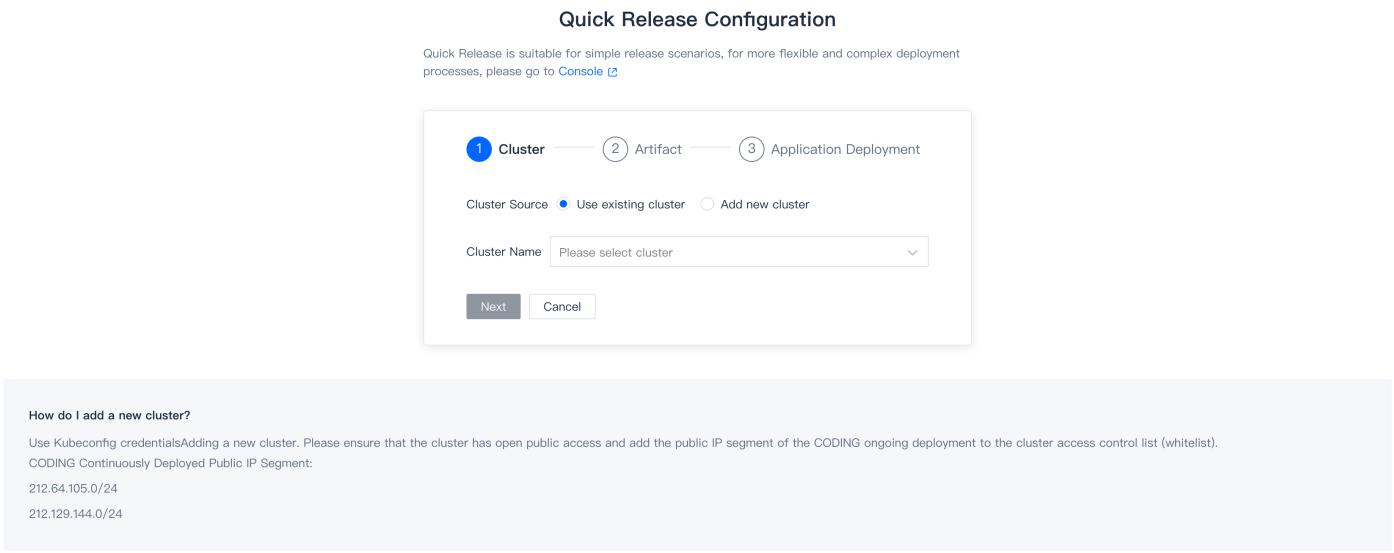


单击**新建发布单**，可以运行已有应用及部署流程。

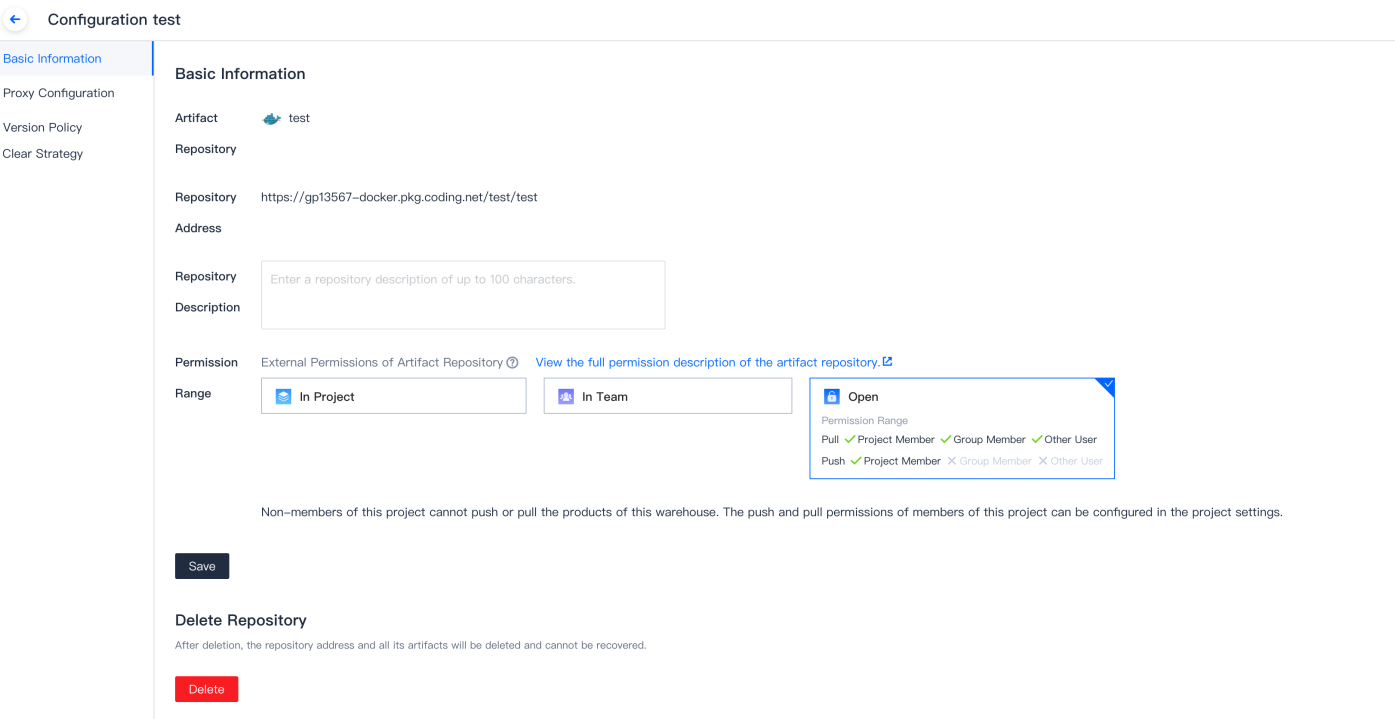


快速发布

若不希望在团队设置复杂的权限限制，而直接希望体验持续部署功能，可以使用**快速发布**功能。无需在控制台中配置部署流程即可将镜像发布至集群中，适用于更加灵活复杂的部署流程的场景，例如临时镜像变更等突发场景，无需快速将制品发布至集群中。



成员所在用户组需具备部署管理权限，所发布的制品的权限范围需设置为公开状态，从而能够被集群访问。



发布完成后可以在持续部署中查看发布详情。