制品库

操作指南

产品文档

腾讯云

版权所有：腾讯云计算（北京）有限责任公司

【版权声明】

©2013-2023 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内容。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】

腾讯云

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。

文档目录

版权所有：腾讯云计算（北京）有限责任公司

操作指南
开通用户
主账号

最近更新时间：2023-01-05 18:13:03

操作场景

本文为您介绍腾讯云主账号如何开通并使用 CODING DevOps 服务，您可以按照以下几种情况有选择性操作。

主账号未开通服务

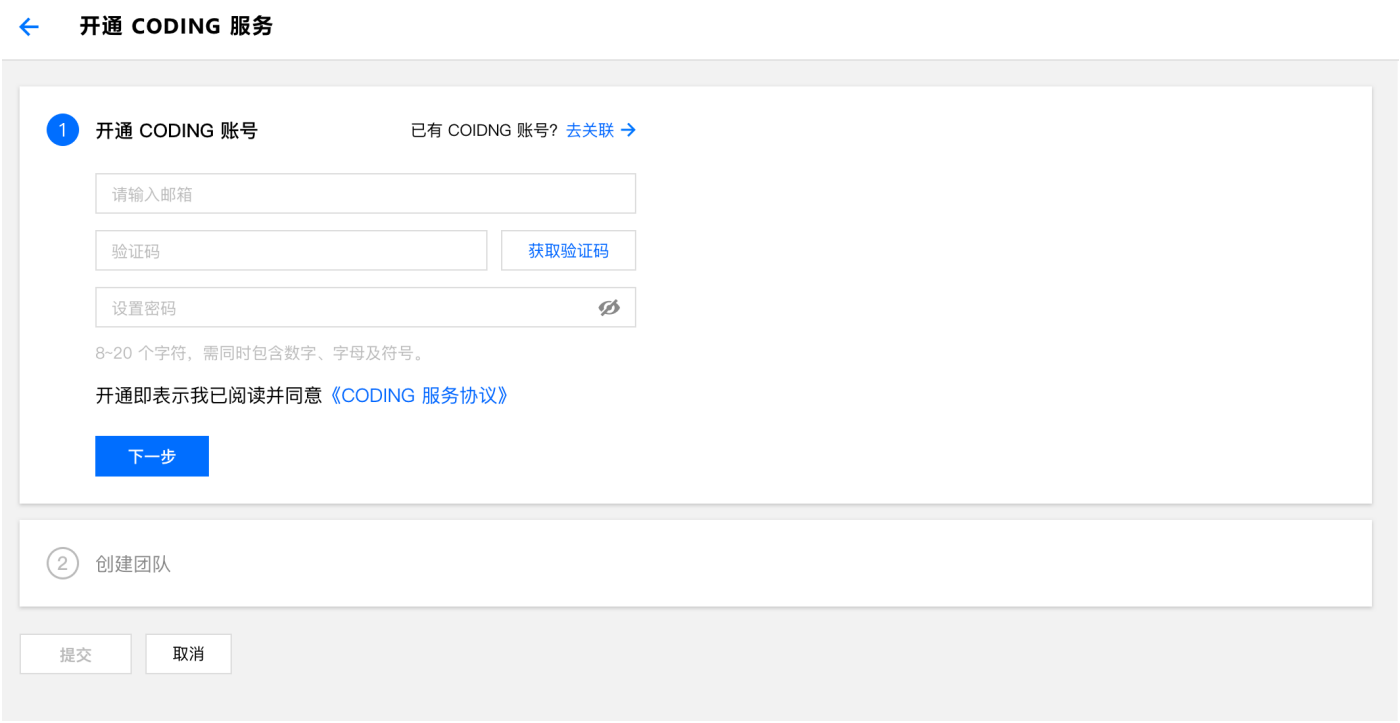
此场景适用于未曾在腾讯云中使用 CODING DevOps 服务的用户，请访问 CODING DevOps 控制台 并按照提示完成策略权限授权后，开通 CODING 服务。

> **说明：**
> 腾讯云主账号服务开通后，您还需要 CODING 账号才能正常使用服务。

未注册过 CODING 账号

1. 若您未注册过 CODING 账号，请单击**没有 CODING 账号，去开通**。

2. 按照提示填写邮箱、验证码、密码进入下一步，输入团队名称和域名后完成团队创建。



关联已有 CODING 账号

若您有印象曾使用过 CODING 服务，但并非通过腾讯云控制台入口注册，那么可以选择将腾讯云账号与已有 CODING 账号相关联。请单击控制台中的**已有 CODING 账号，去关联**。

输入已注册的 CODING 账号完成关联操作，关联完成后将继承原有 CODING 团队数据。

主账号已开通服务

单击 控制台 中的**立即使用**或**团队域名**，跳转至 CODING 团队页后，开始使用 CODING 服务。

子用户

最近更新时间：2021-11-24 10:58:19

操作场景

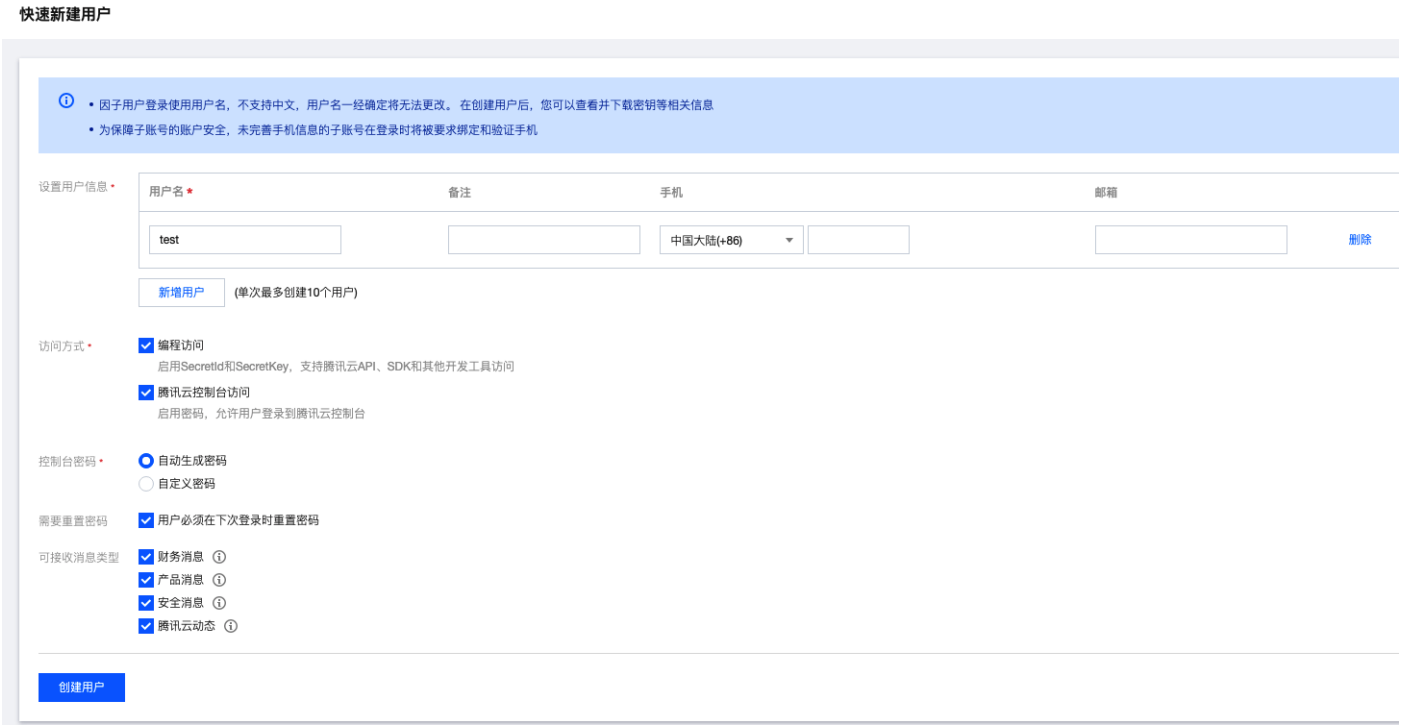
子用户是由主账号创建的实体，有确定的身份 ID 和身份凭证，能够登录并独立设置控制台，且具有 API 访问策略。若您希望通过腾讯云子用户使用 CODING DevOps 服务（以下简称服务），下文将按照多种场景进行指引。

1. 主账号已开通服务，需帮助子用户开通服务
2. 主账号未开通服务，需使用子用户协助主账号开通服务
3. 子用户已开通服务，但部分功能受限

主账号已开通服务

1. 主账号登录 CODING DevOps 控制台。
2. 单击**快速创建子用户**。

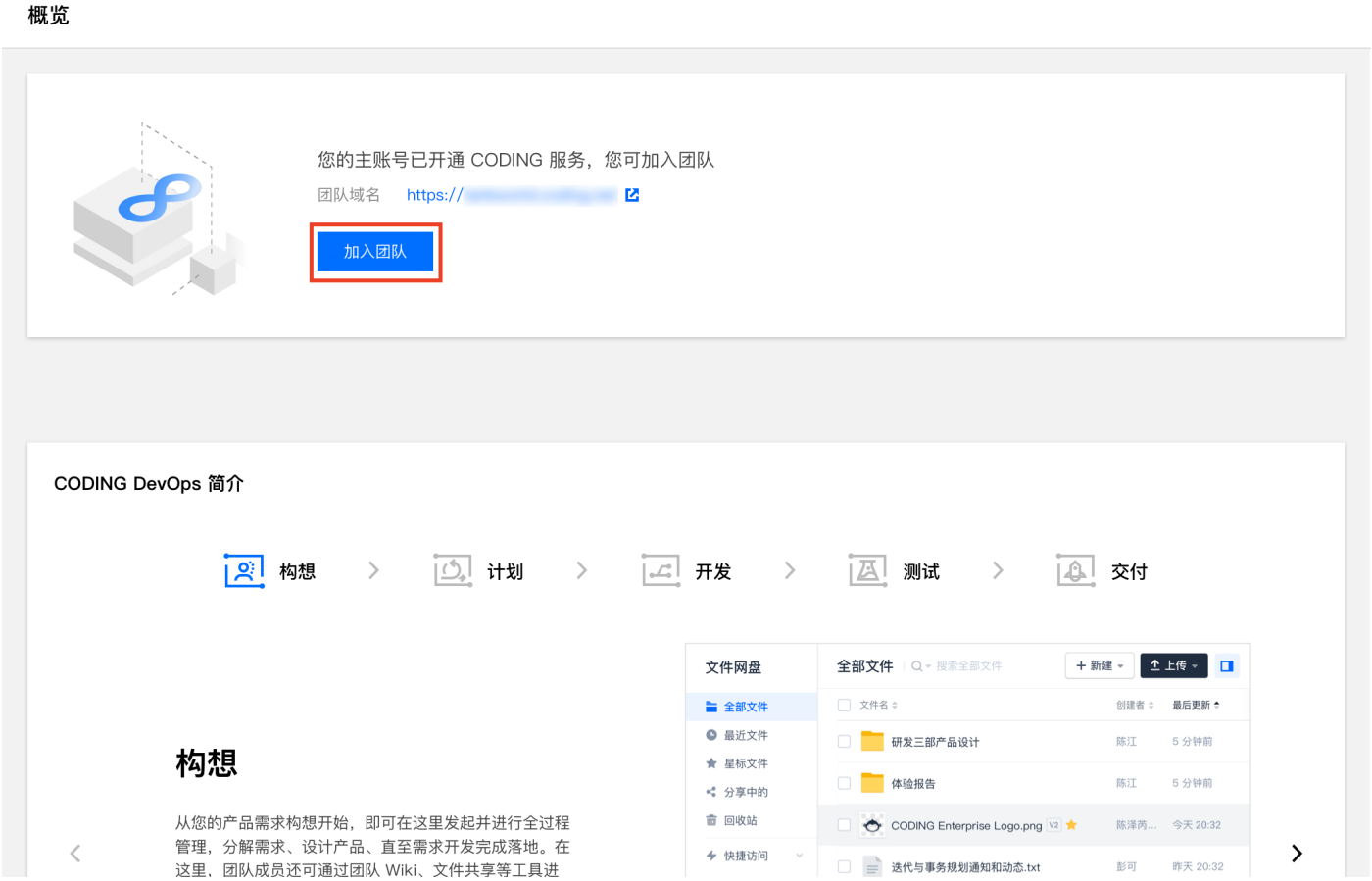
3. 在**快速新建用户**页面填写子用户相关信息，访问方式请同时勾选**编程访问**及**腾讯云控制台访问**，密码设置可选自动生成或自定义密码，单击**创建用户**后即可创建出拥有 CODING DevOps 策略的子用户。



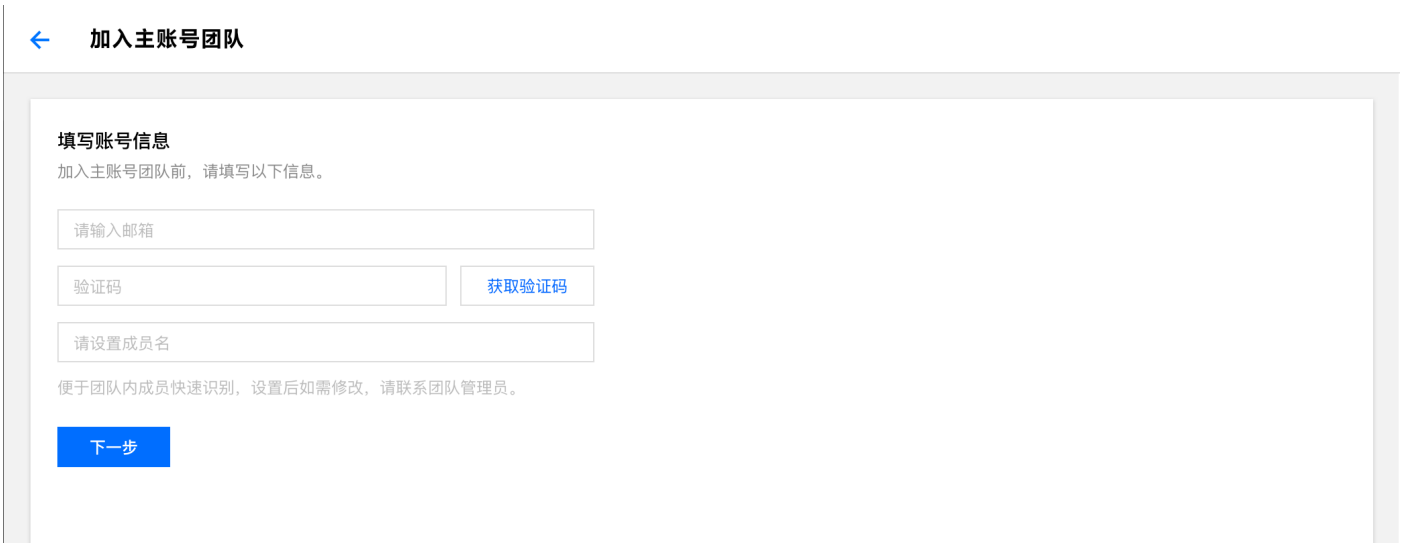
注意：

新建子用户时，请务必勾选访问方式的编程访问和腾讯云控制台访问，避免子用户无法登录 CODING DevOps 控制台的情况。

4. 创建完成后，使用子用户登录 控制台，单击**加入团队**。



按照提示填写账号信息后加入团队。



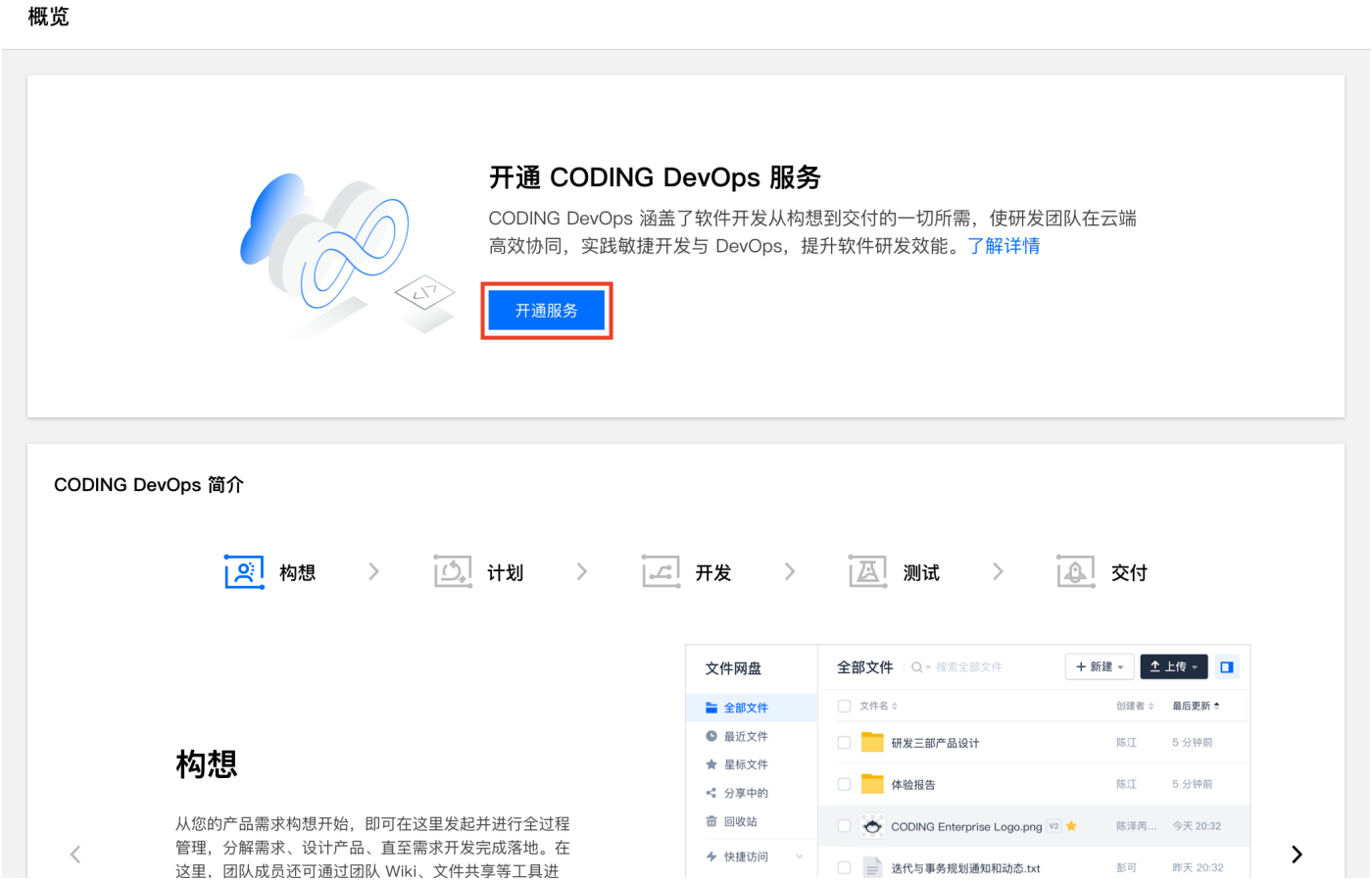
主账号未开通服务

使用主账号直接开通服务

请联系主账号拥有者，参见 主账号，了解如何使用主账号直接开通服务。

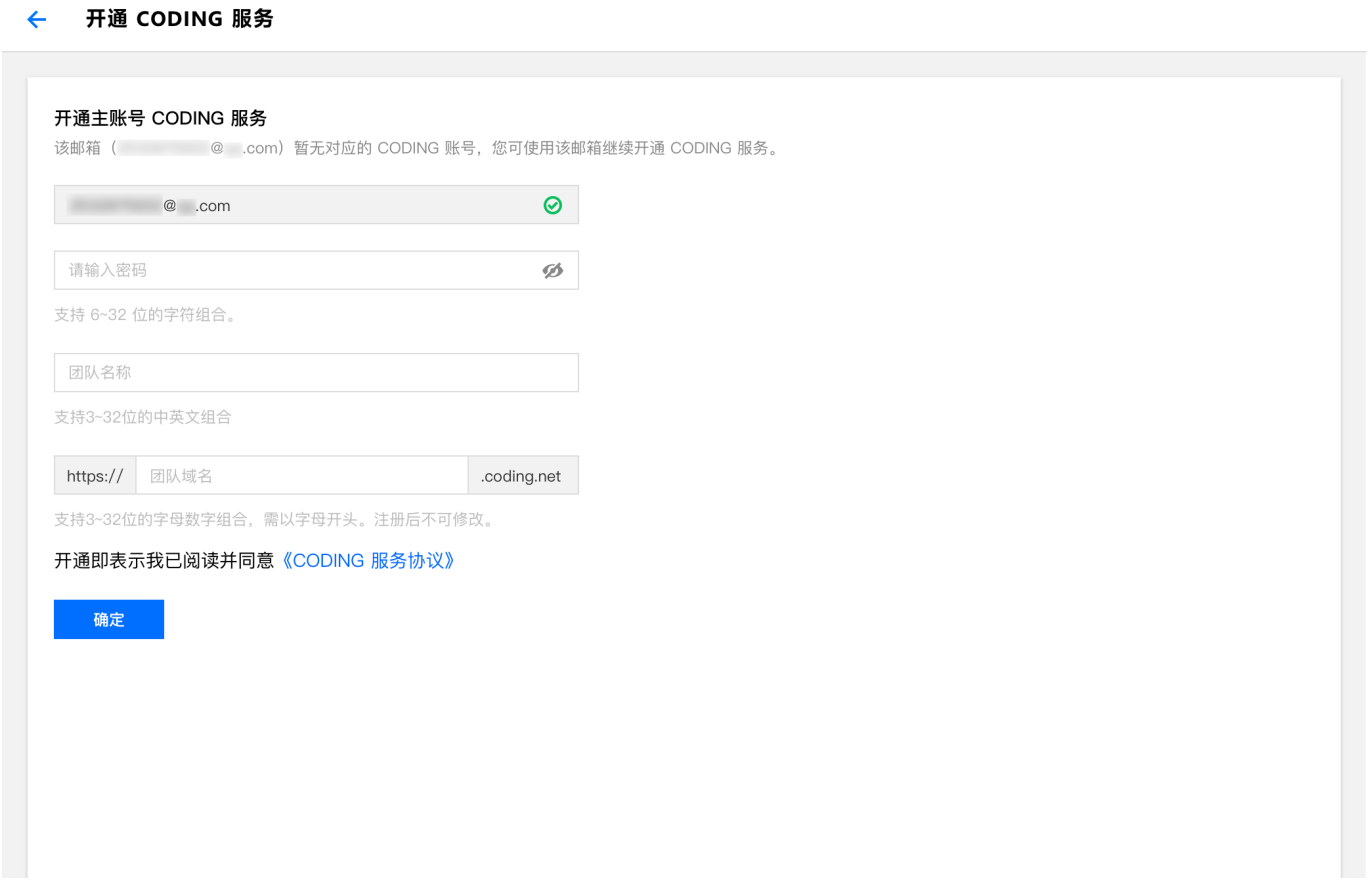
使用子用户帮助主账号开通服务

1. 子用户需前往 访问管理控制台，确认拥有 AdministratorAccess 权限策略，然后前往 CODING DevOps 控制台 > 概览页面。
2. 单击**开通服务**，进入主账号开通服务流程。

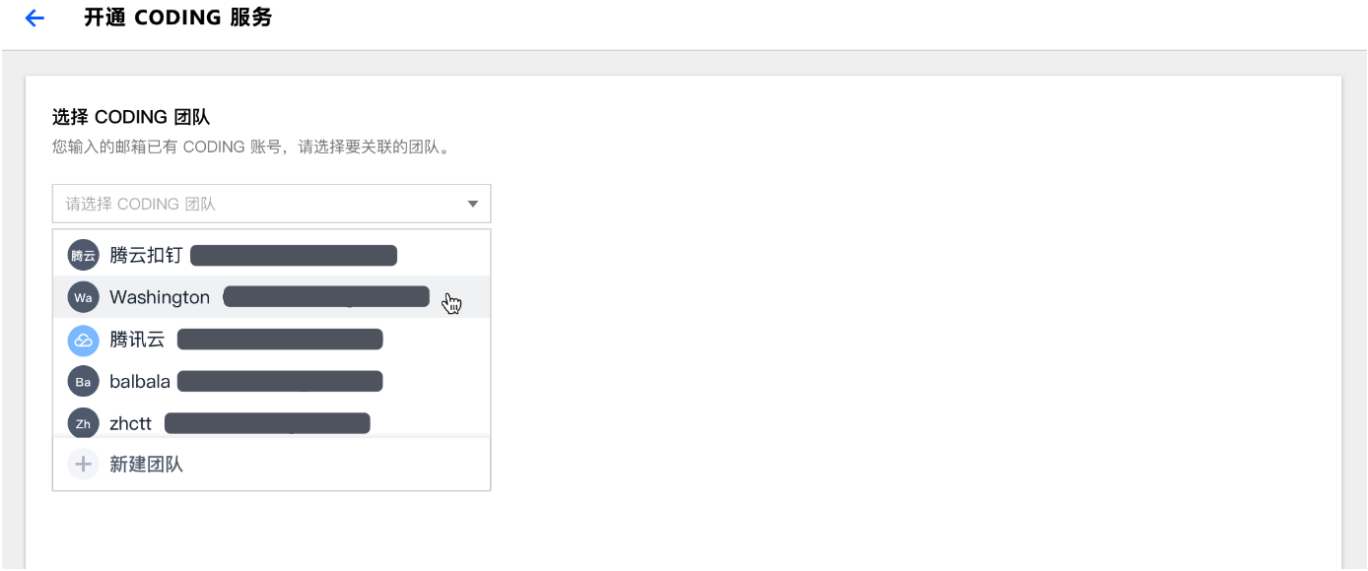


输入主账号邮箱与验证码完成开通。

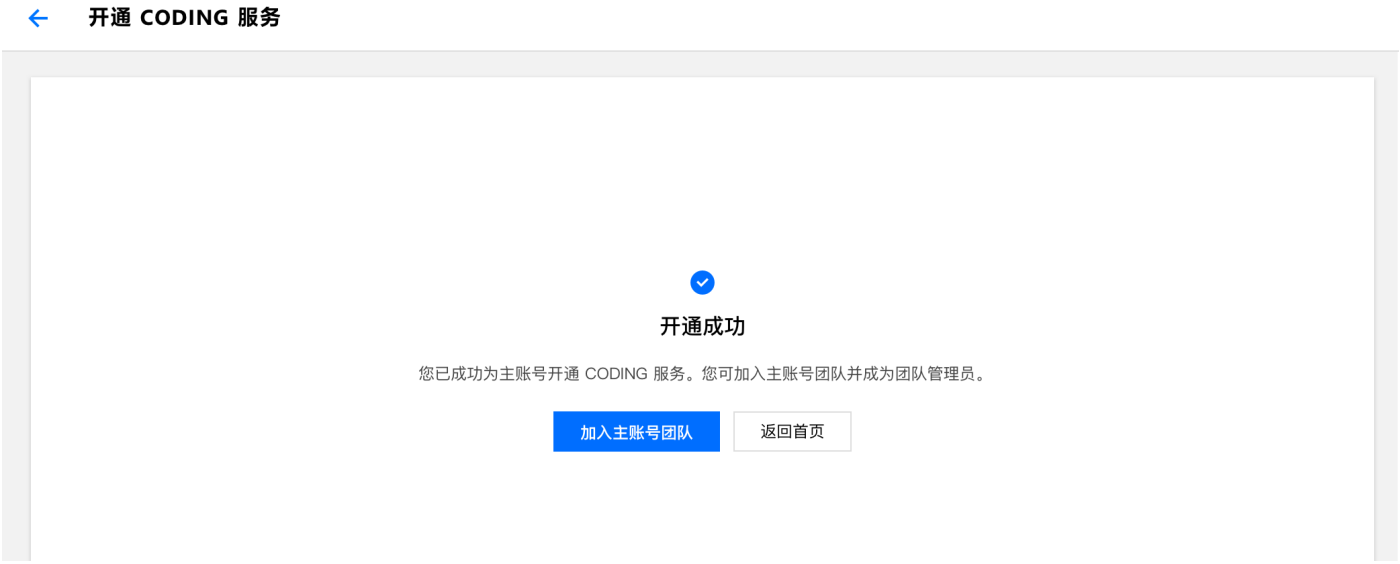
- 若主账号邮箱不存在，需进入主账号创建流程，输入密码与团队名称后开通账号。



- 若主账号邮箱已关联多个 CODING 团队，选择需要的团队即可。



3. 开通成功后，子用户即可加入主账号团队。

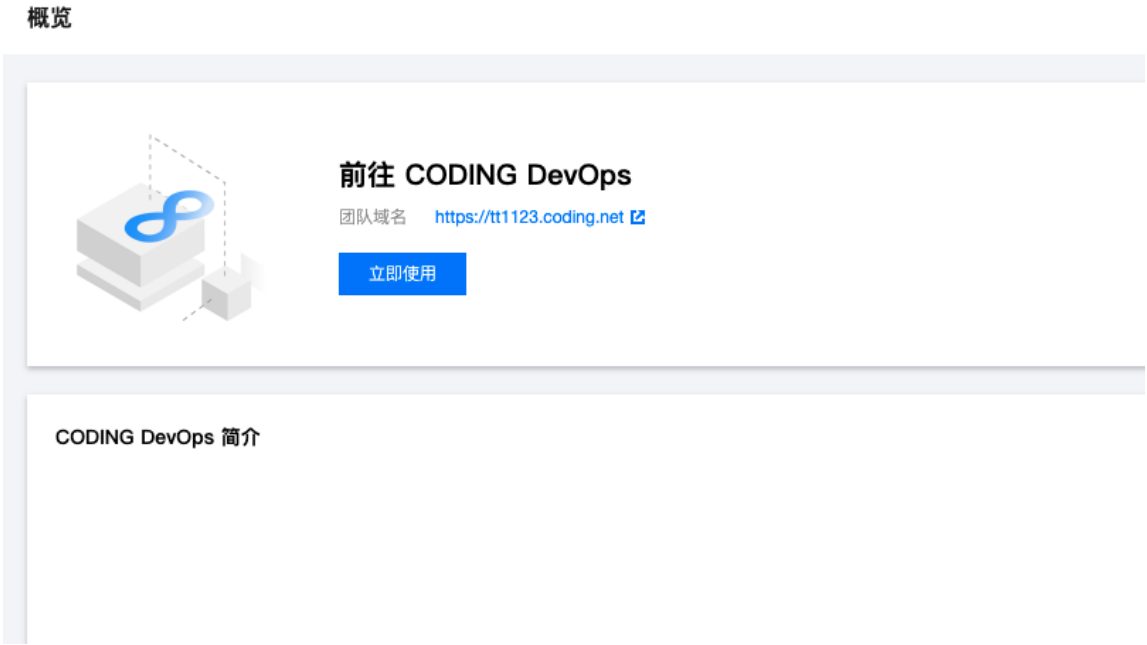


子用户已开通服务

子用户能否继续创建新的子用户，取决于 访问管理控制台 中所选择的策略。

当子用户具备 AdministratorAccess 策略时，控制台将出现**快速创建子用户**，意味着该子用户能够创建新的子用户。

若子用户仅具备 QcloudCODINGFULLAccess，控制台中将不再出现**快速创建子用户**，意味着该子用户仅具备服务的访问策略，无法继续新建子用户。



权限说明

最近更新时间：2022-04-22 15:08:44

在使用腾讯云 CODING Devops 产品的过程中，为了能够调取相关云资源，会遇到需要进行服务授权的场景。在使用该产品的过程中主要涉及 CODING_QCSRole 角色，本文档会展示此角色所包含的预设策略详情。

CODING_QCSRole 角色

开通 CODING Devops 服务后，腾讯云会授予您的账户 CODING_QCSRole 角色的权限。该角色默认关联多个预设策略，对应策略会出现在该角色的已授权策略列表中。CODING_QCSRole 角色默认关联的预设策略包含如下：

- QcloudAccessForCODINGRole：CODING 服务对云资源的访问权限。
- QcloudAccessForCODINGRoleInAccessTKE：CODING 服务所需要的容器服务（TKE）权限。
- QcloudAccessForCODINGRoleInThroughTCR：CODING 服务所需要的容器镜像（TCR）权限。

预设策略 QcloudAccessForCODINGRole

授权场景

当您已注册并登录腾讯云账号后，首次登录 CODING 控制台 时，需前往“访问管理”页面对当前账号赋予 CODING DevOps、访问管理（CAM）等权限。

授权步骤

1. 首次登录 CODING 控制台 时会弹出**服务授权**窗口。
2. 单击**前往访问管理**，进入角色管理页面。
3. 单击**同意授权**，完成身份验证后即可成功授权。

权限内容

- 访问管理相关

权限名称	权限说明
cam:ListPolicies	查询策略列表

- 容器镜像服务相关

权限名称	权限说明
tcr:DescribeInstances	查询实例信息
tcr:CreateInstanceToken	创建实例访问凭证
tcr:DescribeNamespaces	查询命名空间信息

权限名称	权限说明
tcr:DescribeRepositories	查询镜像仓库信息
tcr:DescribeRepositoryOwnerPersonal	查询个人版所有仓库

预设策略 QcloudAccessForCODINGRoleInAccessTKE

授权场景

开通 CODING Devops 服务并完成 CODING_QCSRole 角色授权后，该策略将会与 CODING_QCSRole 角色相关联，完成操作后即可获得腾讯云容器服务（TKE）云资源的相关权限。

授权步骤

该策略与预设策略 QcloudAccessForCODINGRole 同时授权，无需额外操作。

权限内容

权限名称	权限说明
ccs:DescribeCluster	查询集群列表
ccs:CreateClusterEndpoint	创建集群访问端口
ccs:CreateClusterEndpointVip	创建托管集群外网访问端口
ccs:ModifyClusterEndpointSP	修改托管集群外网端口的安全策略
ccs:DescribeClusterEndpointVipStatus	查询托管集群开启外网端口流程状态

预设策略 QcloudAccessForCODINGRoleInThroughTCR

授权场景

开通 CODING Devops 服务并完成 CODING_QCSRole 角色授权后，该策略将会与 CODING_QCSRole 角色相关联，完成操作后即可获得腾讯云容器镜像服务（TCR）云资源的相关权限。

授权步骤

该策略与预设策略 QcloudAccessForCODINGRole 同时授权，无需额外操作。

权限内容

- 容器镜像服务相关

权限名称	权限说明
tcr:CreateWebhookTrigger	创建触发器

权限名称	权限说明
tcr:ModifyWebhookTrigger	更新触发器
tcr:DeleteWebhookTrigger	删除触发器
tcr:DescribeWebhookTriggerLog	获取触发器日志
tcr:PushRepository	推送镜像
tcr:PushRepositoryPersonal	个人版推送镜像

版权所有：腾讯云计算（北京）有限责任公司

制品库权限

最近更新时间：2022-12-28 16:43:51

本文为您介绍如何使用制品库中的权限设置。

进入制品库功能页

1. 登录 CODING 控制台，单击**立即使用**进入 CODING 使用页面。
2. 单击团队首页左侧的**项目**，进入项目列表页，选择目标项目。
3. 单击左侧菜单栏的**制品管理**，进入制品库功能页面。

功能介绍

制品作为团队的重要资产，它的权限管理是至关重要的：例如某些制品需要开放给第三方外部使用（例如开源组件包）、某些制品需要开放给团队内的其他项目组成员使用（例如基础公共组件包）、某些制品只需开放给本项目成员使用（例如应用安装包）。CODING 制品库提供了完善的权限管理，满足不同场景下的权限需求。

权限规则

1. CODING 制品库提供了三种权限范围
   - 项目内
   - 团队内
   - 公开
2. 不同成员对制品的操作
   - 拉取：从制品库拉取任意指定的制品。
   - 推送：推送任意制品到制品库。
   - 代理：从代理中同步不在缓存中的制品。详情请参见 制品库代理。
3. 鉴权规则

成员类别	项目内			团队内			公开		
	拉取	推送	代理	拉取	推送	代理	拉取	推送	代理
本项目成员	✓	✓	✓	✓	✓	✓	✓	✓	✓
团队内成员	×	×	×	✓	×	×	✓	×	×
匿名成员	×	×	×	×	×	×	✓	×	×

设置权限

了解上述权限对应的鉴权规则后，您可以在创建制品仓库时按需设置权限范围，对于已经创建好的制品仓库您也可以进行权限范围修改。

创建仓库时设置权限

在**制品库**页面，新建制品仓库时即可设置权限范围。

对已存在的仓库修改权限

1. 单击仓库右上方的**设置仓库**。

2. 在**基本信息**中即可对权限范围进行更新。

制品库认证

最近更新时间：2023-08-09 11:08:22

本文为您介绍如何使用制品库中的认证机制。

进入制品库功能页

1. 登录 CODING 控制台，单击**立即使用**进入 CODING 使用页面。
2. 单击团队首页左侧的**项目**，进入项目列表页，选择目标项目。
3. 单击左侧菜单栏的**制品管理**，进入制品库功能页面。

功能介绍

当用户访问制品库时，制品库会对用户提供的凭证进行鉴权，以确保用户对制品库拥有操作权限。详情请参见 制品库权限。制品库支持多种鉴权方式：

- 个人访问令牌
- 用户账号密码
- 项目令牌

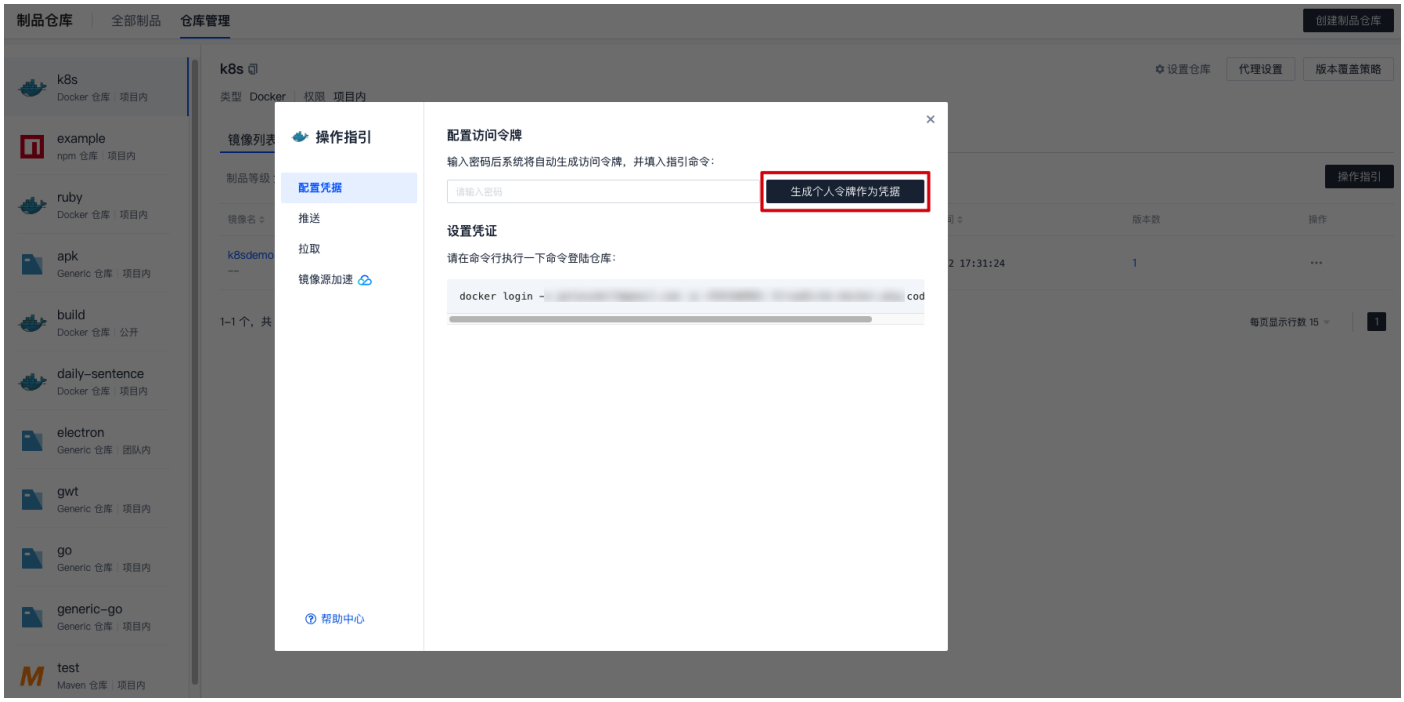
每种制品库在本地配置凭证的命令会有区别，但逻辑相似，并且在制品库页面都有设置凭证的指引。本文以 Docker 制品库为例，演示用户配置鉴权凭证的三种方式。

个人访问令牌

个人访问令牌（Access token）包含了用户的安全认证信息，利用访问令牌可以拥有查看或操作相应资源的权限。推荐使用个人访问令牌进行制品库认证，相比直接配置用户账号密码更加安全。

版权所有：腾讯云计算（北京）有限责任公司

1. 单击制品仓库指引页面的**使用访问令牌生成配置**。

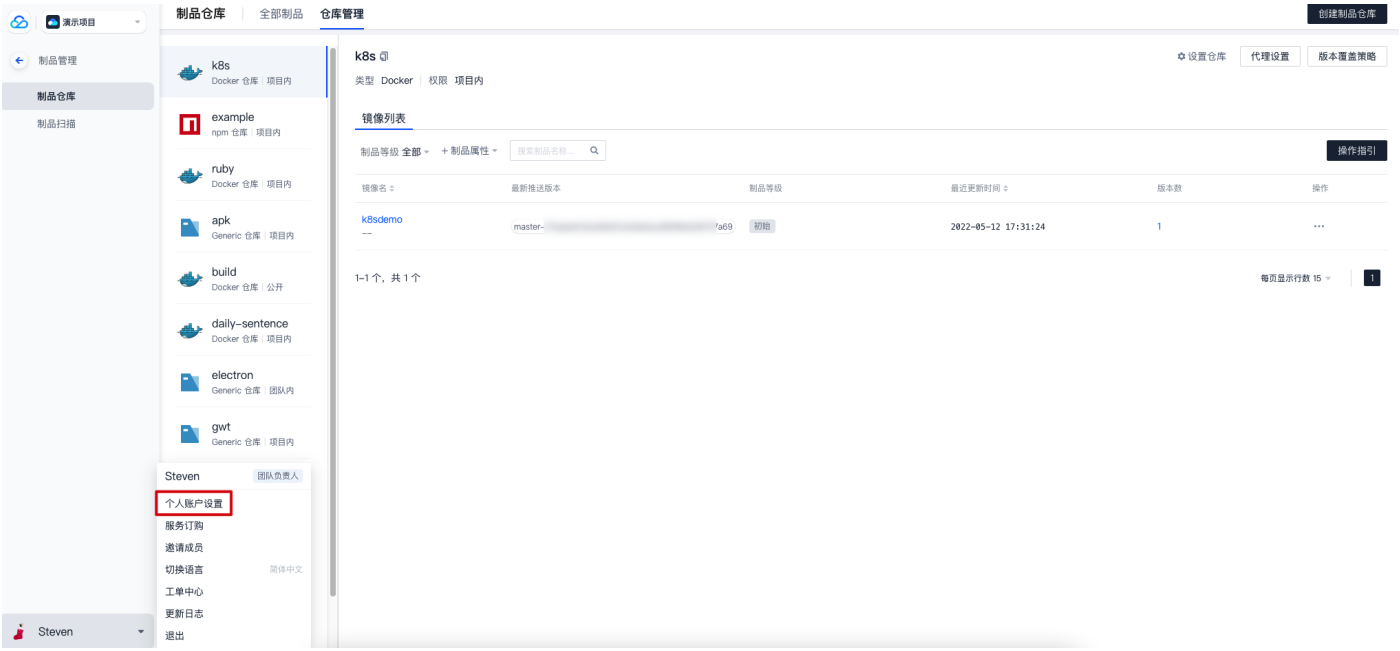


2. 输入认证信息后，即可看到已携带新访问令牌的执行命令，单击“copy”复制命令。

3. 在本地 docker 环境中，执行刚刚复制的 docker login 命令，提示登录成功即可进行下一步的推送或拉取操作。

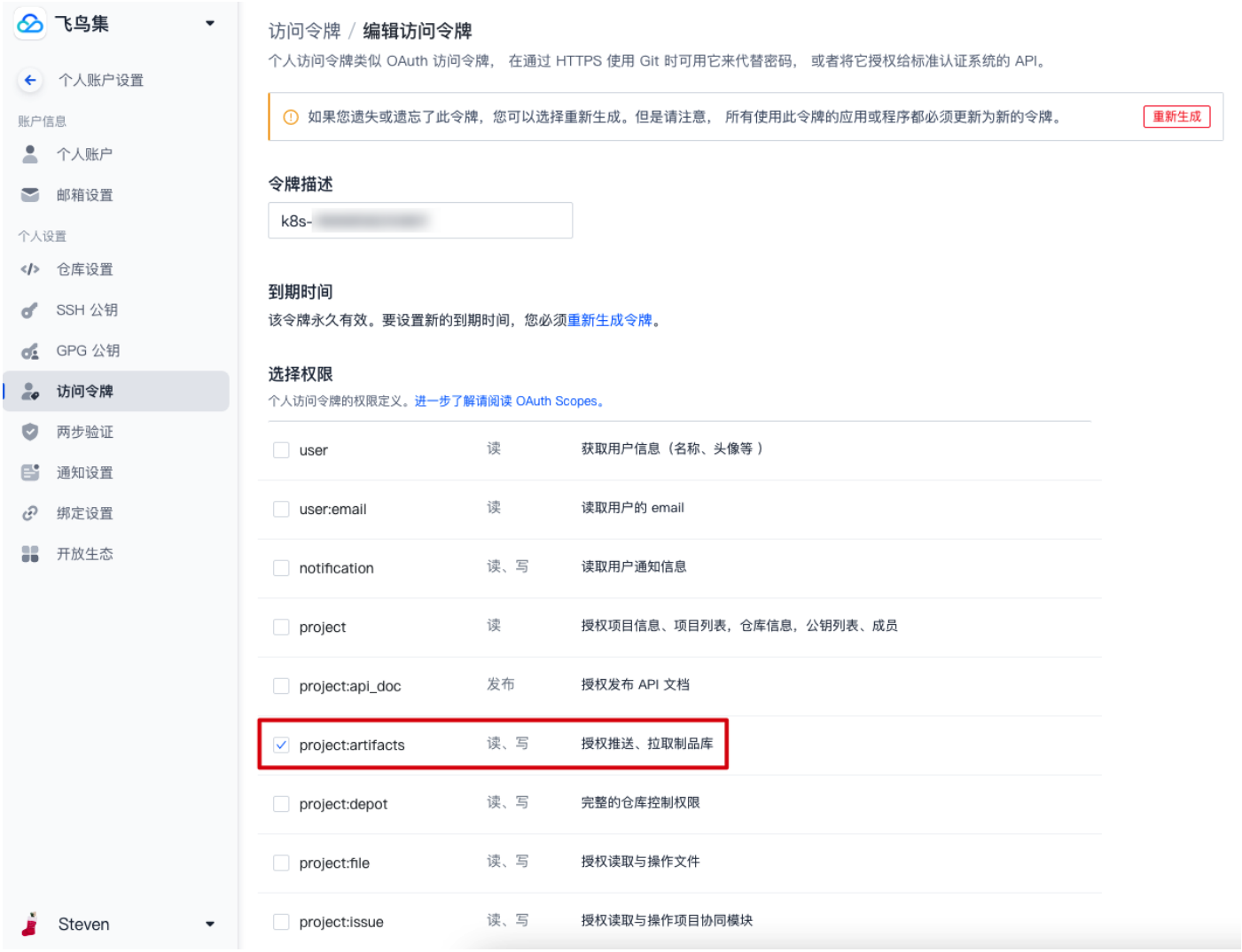
查看个人令牌

1. 单击左下角头像框中的**个人账户设置**。



2. 在个人账户页面，单击**访问令牌**，可以看到上述步骤中通过制品库新建的个人访问令牌信息，在此页面您也可以看到个人访问令牌被使用的记录。

3. 单击令牌后面的**编辑**，可以查看或修改该访问令牌的权限信息。



用户账号密码

通过用户账号（手机号或邮箱）密码也可设置凭证信息。

1. 在制品仓库指引页面，直接复制默认提供的 Docker 命令，将其中的 password 替换为个人账户登录密码。

2. 在本地 docker 页面执行命令，输入密码，即可认证成功进行下一步推送或拉取操作。

```
[root@docker-test-12315 ~]#
[root@docker-test-12315 ~]# docker login -u [redacted]@qq.com anywhere-docker.pkg.coding.net
Password:
Login Succeeded
```

制品属性

最近更新时间：2022-12-28 16:44:00

本文为您介绍如何使用制品库中的制品属性及 REST API 操作说明。

进入制品库功能页

1. 登录 CODING 控制台，单击**立即使用**进入 CODING 使用页面。
2. 单击团队首页左侧的**项目**，进入项目列表页，选择目标项目。
3. 单击左侧菜单栏的**制品管理**，进入制品库功能页面。

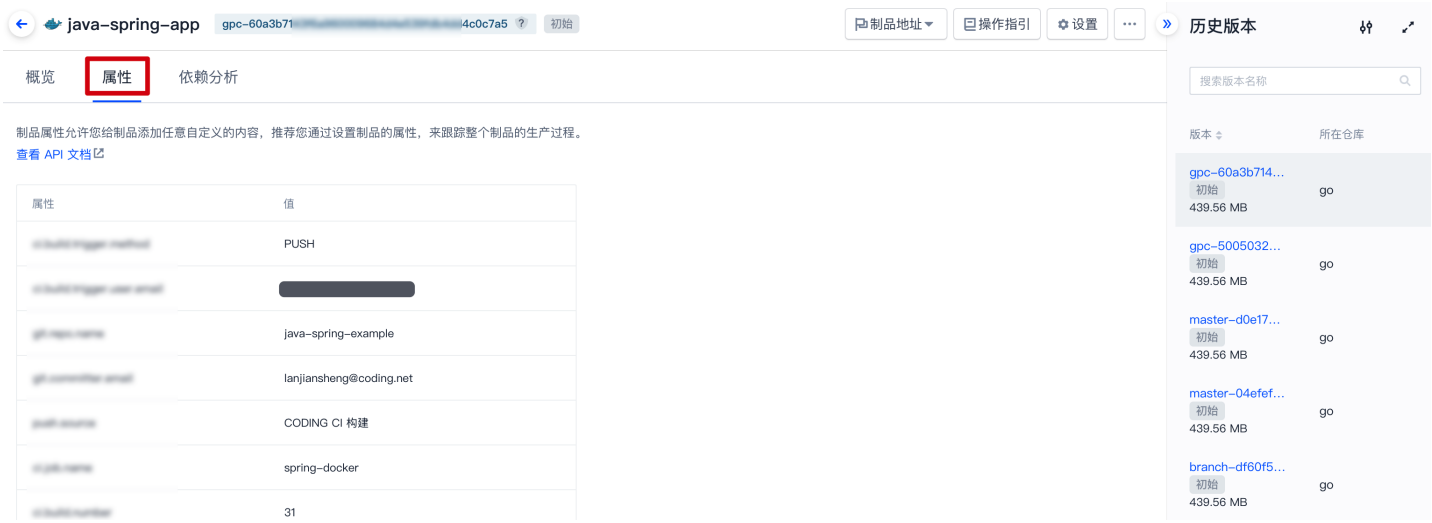
功能介绍

CODING 制品属性支持用户对制品版本的自定义属性，可以进行**查询、新增、删除、修改**的操作。制品属性同时支持通过**页面操作**以及 **REST API** 进行管理。

> 说明：
> 制品属性制品元数据的区别：制品属性不同于制品元数据，**制品元数据通常为制品类型的原生属性**，例如 npm 的 packageName 和 version 等信息。**制品属性更多的是用来描述元数据无法定义的内容**，您可以利用制品属性写入在 CODING 持续集成中的制品产出信息，或其它自定义内容。

通过页面操作管理属性

在**制品库**页面，单击指定仓库下的指定包名，进入包页面后，单击**属性**，可在页面上对制品属性进行**查看、新增、修改、删除**操作。



如何在 CODING 持续集成中收集制品属性

下面给出一个 Docker 制品的属性收集的 Jenkinsfile 示例，可以使用该示例文件创建一个持续集成 Job。

```
pipeline {
agent any
environment {
ENTERPRISE = "myteam"
PROJECT = "myproject"
ARTIFACT_REPO = "myrepo"
PACKAGE = "mypkg"
VERSION = "myversion"

ARTIFACT_BASE = "${ENTERPRISE}-docker.pkg.coding.net"

ARTIFACT_IMAGE = "${ARTIFACT_BASE}/${PROJECT}/${ARTIFACT_REPO}/${PACKAGE}:${VERSIO
N}"
// 该 docker 镜像用于收集制品属性
PROP_COLLECTOR = 'docker.pkg.coding.net/ci-props-collector:0.1.0'

}
stages {
stage('检出') {
steps {
checkout([$class: 'GitSCM', branches: [[name: env.GIT_BUILD_REF]],
userRemoteConfigs: [[url: env.GIT_REPO_URL, credentialsId: env.CREDENTIALS_ID]]])
}
}
stage('推送到制品库') {
steps {
// 此处使用 hello-world 的 docker 镜像作为演示
// 您可以根据自己的实际情况将此处修改成其他制品类型的推送逻辑
sh 'docker pull hello-world'
sh 'docker tag hello-world ${ARTIFACT_IMAGE}'
script {
docker.withRegistry("https://${ARTIFACT_BASE}", "${env.DOCKER_REGISTRY_CREDENTIALS_ID}") {
docker.image("${ARTIFACT_IMAGE}").push()
}
}
}
}
stage('收集制品属性') {
steps {
script {
// 使用 CODING 持续集成内置的服务连接作制品属性接口的认证方式
// 您也可以使用自己创建的项目令牌，写入到 USERNAME 和 PASSWORD 中
```

```
withCredentials([
usernamePassword(
credentialsId: env.DOCKER_REGISTRY_CREDENTIALS_ID,
usernameVariable: 'USERNAME',
passwordVariable: 'PASSWORD'
)]) {
// 将本次构建的信息写入到对应的制品属性中
sh '''
docker run --rm \
-e USERNAME=${USERNAME} \
-e PASSWORD=${PASSWORD} \
-e PROJECT=${PROJECT} \
-e REPO=${ARTIFACT_REPO} \
-e PACKAGE=${PACKAGE} \
-e VERSION=${VERSION} \
-e ENTERPRISE=${ENTERPRISE} \
-e CI_BUILD_NUMBER=${CI_BUILD_NUMBER} \
-e JOB_ID=${JOB_ID} \
-e JOB_NAME=${JOB_NAME} \
-e PROJECT_NAME=${env.PROJECT_NAME} \
-e GIT_REPO_URL=${GIT_REPO_URL} \
-e GIT_COMMITTER_NAME=${GIT_COMMITTER_NAME} \
-e GIT_COMMITTER_EMAIL=${GIT_COMMITTER_EMAIL} \
-e GIT_LOCAL_BRANCH=${GIT_LOCAL_BRANCH} \
-e GIT_COMMIT=${GIT_COMMIT} \
${PROP_COLLECTOR}
'''
}
}
}
}
}
}
```

通过 REST API 管理属性

制品属性支持用户使用 REST API 来直接对制品属性进行**查询**、**新增**、**删除**、**修改**的操作。

> **说明：**
> 目前仅支持使用**项目令牌**调用制品属性的 REST API。

申请拥有制品属性权限的项目令牌

1. 单击左侧菜单**项目设置 > 开发者选项 > 项目令牌 > 新建项目令牌**。

2. 在新建令牌页面，填写令牌名称，选择令牌过期时间，勾选**制品属性**，单击**新建**。

3. 新建成功后，在项目令牌列表中，单击**查看密码**，校验身份后即可获得该令牌的用户名及密码。

$URL 的组成

```
https://{teamGK}.coding.net/api/projects/{projectName}/repositories/{repoName}/packages/{pkgName}/versions/{versionName}/properties
```

$URL 参数说明

参数	说明
teamGK	团队域名，例如：codingcorp.coding.net 的 teamGK 为 codingcorp
projectName	项目名称
repoName	制品仓库名称
pkgName	制品包名称
versionName	制品版本名称

示例：

```
https://myteam.coding.net/api/projects/myproject/repositories/myrepo/packages/mypkg/versions/myversion/properties
```

快速上手

新增制品属性

```
curl -u 项目令牌:密码 -H "Content-Type: application/json" -d '{"properties":[{"name":"demo.name","value":"demo value"}]}' -X POST $URL
```

查询制品属性

```
curl -u 项目令牌:密码 $URL
```

修改制品属性

```
curl -u 项目令牌:密码 -H "Content-Type: application/json" -d '{"properties":[{"name":"demo.name","value":"new demo value"}]}' -X PUT $URL
```

删除制品属性

```
curl -u 项目令牌:密码 -H "Content-Type: application/json" -d '{"names":["demo.name"]}' -X DELETE $URL
```

返回 code 为 0 则表示操作成功

```
{"code":0}
```

详细说明

body 参数说明

参数	说明
name	属性名
value	属性值

其他说明

- 制品属性名称以 **coding.** 开头的属性名将作为 CODING 平台内部的保留字段，这一类字段将会有特殊的解析含义和联动。保留字段的属性目前**不支持删除和修改**操作，仅支持**新增**及**查询**。
- 制品属性值支持多个值的形式，以分号分隔即可，具体为： **value1;value2;value3** 。

命名规则

- 制品属性名称仅支持 1-128 位的小写英文字母、数字、下划线 (_)、小数点 (.)；

- 制品属性名称不允许以小数点 (.) 开头或结尾，且不允许包含连续的小数点 (.)；
- 制品属性名称和值均不能为空；
- 制品属性值不能以分号 (;) 开头或结尾；
- 制品属性值为以分号分隔的多个值时，不允许连续相同的值。
- 具体命令

1. **新增制品属性**

```
curl -u 项目令牌:密码 \
-H "Content-Type: application/json" \
-d '{"properties":[{"name":"demo.name","value":"demo value"}]}' \
-X POST $URL
curl -u 项目令牌:密码 \
-H "Content-Type: application/json" \
-d '{"properties":[{"name":"demo.name","value":"demo value"},{"name":"demo.name2","valu
e":"demo value2"}]}' \
-X POST $URL
```

示例：

2. **查询制品属性**

```
curl -u 项目令牌:密码 $URL
```

示例：

3. **修改制品属性**

```
curl -u 项目令牌:密码 \
-H "Content-Type: application/json" \
-d '{"properties":[{"name":"demo.name","value":"new demo value"}]}' \
-X PUT $URL
curl -u 项目令牌:密码 \
-H "Content-Type: application/json" \
-d '{"properties":[{"name":"demo.name","value":"new demo value"},{"name":"demo.name2","va
lue":"new demo value2"}]}' \
-X PUT $URL
```

版权所有：腾讯云计算（北京）有限责任公司

示例：

```
❯ curl -u ptjfqnebv7xj:                                                    \
... ❯ -H "Content-Type: application/json" \
... ❯ -d '{"properties":[{"name":"push_time","value":"15:20:35"},{"name":"push_date","value":"2019-09-30"}]}' \
... ❯ -X PUT http://codingcorp.coding.com/api/projects/coding-artifacts/artifacts/repositories/registry/packages/hello-world/versions/        /properties
{"code":0}
```

4. **删除制品属性**

```
curl -u 项目令牌:密码 \
-H "Content-Type: application/json" \
-d '{"names":["demo.name"]}' \
-X DELETE $URL
```

支持同时删除多个制品属性，指定想要删除的制品属性名称即可：

```
curl -u 项目令牌:密码 \
-H "Content-Type: application/json" \
-d '{"names":["demo.name","demo.name2","demo.name3"]}' \
-X DELETE $URL
```

示例：

```
❯ curl -u ptjfqnebv7xj:                                                    \
... ❯ -H "Content-Type: application/json" \
... ❯ -d '{"names":["push_date","push_time"]}' \
... ❯ -X DELETE http://codingcorp.coding.com/api/projects/coding-artifacts/artifacts/repositories/registry/packages/hello-world/versions/        /properties
{"code":0}
```

返回 code 为 0 即表示操作成功

```
{"code":0}
```

若不为 0 则表示操作有误，可根据提示来进行调整，例：

```
❯ curl -u ptjfqnebv7xj:                                                    \
... ❯ -H "Content-Type: application/json" \
... ❯ -d '{"names":["coding.ci.job.id"]}' \
... ❯ -X DELETE http://codingcorp.coding.com/api/projects/coding-artifacts/artifacts/repositories/registry/packages/hello-world/versions/        /properties
{"code":11608,"msg":{"names":"不能更改或删除只读属性 [coding.ci.job.id]"}}
```

版权所有：腾讯云计算（北京）有限责任公司

制品晋级

最近更新时间：2022-12-28 16:44:10

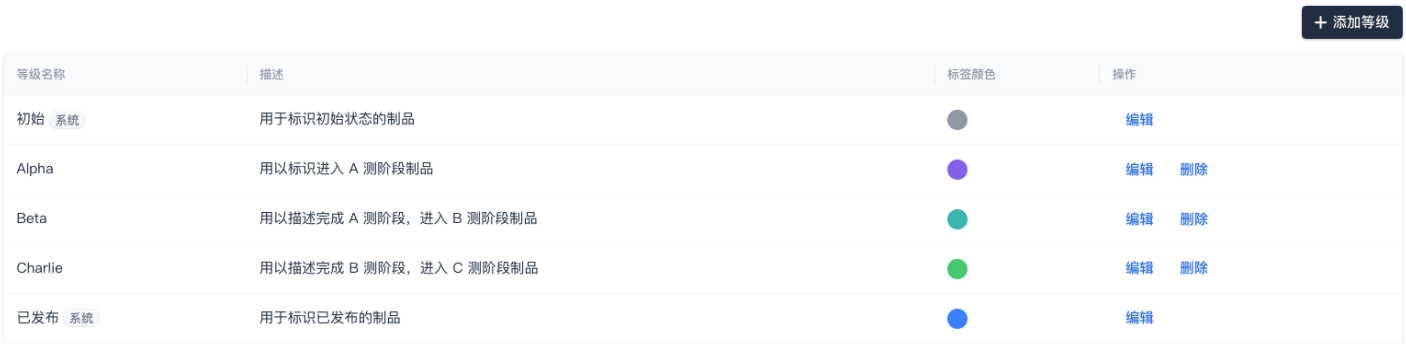
本文为您介绍在制品库中使用制品晋级功能。

进入制品库功能页

1. 登录 CODING 控制台，单击**立即使用**进入 CODING 使用页面。
2. 单击团队首页左侧的**项目**进入项目列表页面。
3. 单击项目中左侧菜单栏的**制品管理**，进入制品库功能页面。

功能介绍

仅通过简单的制品版本号无法有效判断制品是否达到了可交付水平。制品晋级功能提供自定义制品等级和晋级规则，帮助研发团队有效且直观地区分制品版本的成熟度情况，使得最终交付的制品版本是合格且可信赖的。团队成员能够根据业务需求，为制品仓库配置不同的晋级规则，完善制品质量及版本区分。

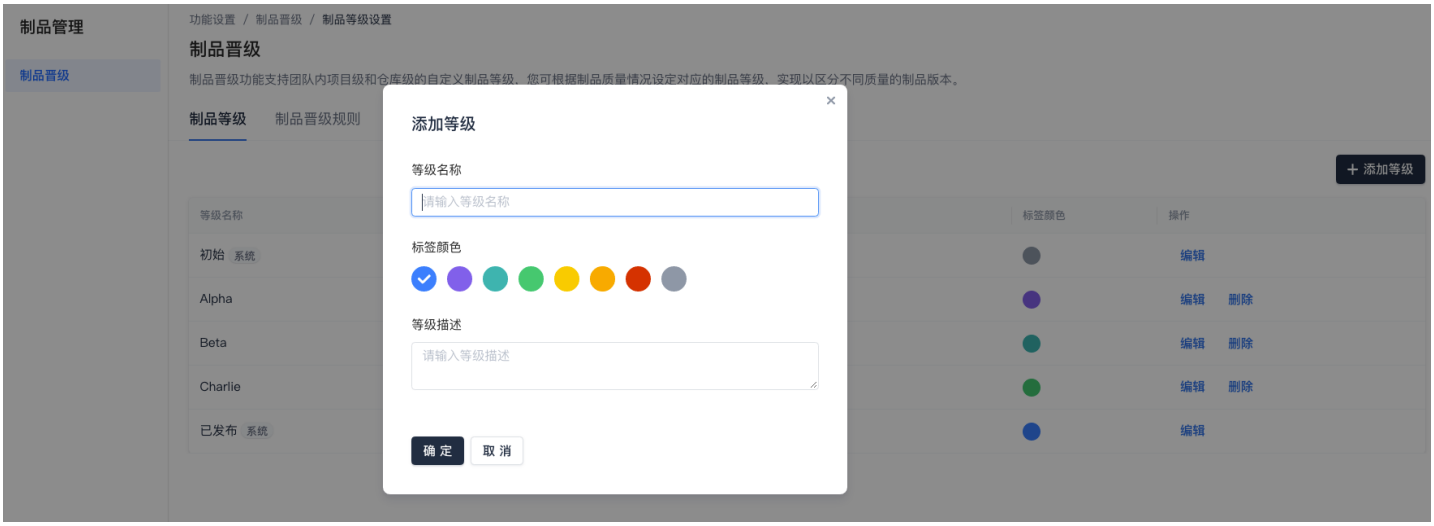


设定制品等级

类似于项目协同中的事项流转，制品晋级本质上是在直观描述制品的交付状态。前往**团队设置中心** > **功能设置** > **制品晋级**创建制品等级。

您可以按照测试交付流程定义制品等级，例如初始 > A 测 > B 测 > 待发布 > 已发布；或根据团队内部对于制品质量的共识

制定等级，输入名称并勾选颜色后完成创建。

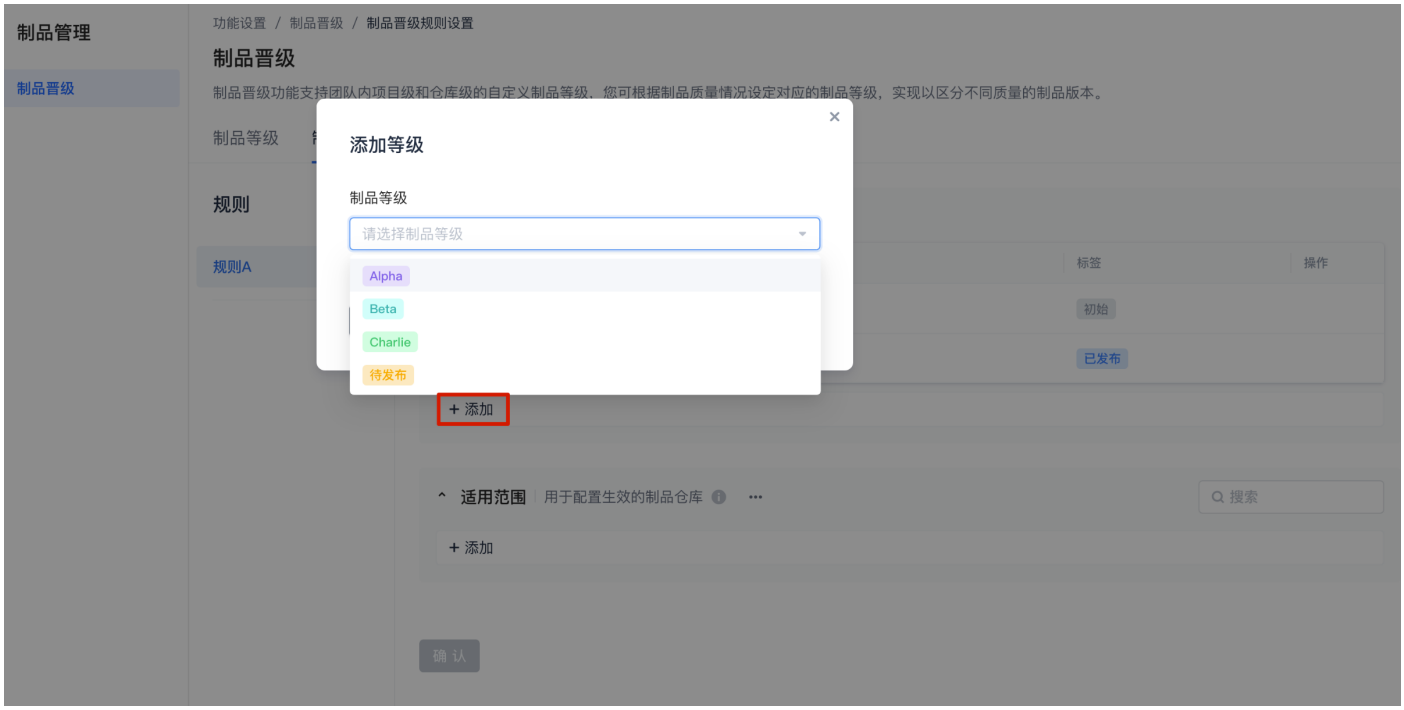


等级创建完成后，还需要继续设定晋级规则才能够应用至制品仓库。

设定制品晋级规则

晋级规则用于定义制品等级的流转顺序，例如规定制品状态需从“初始”等级历经各项测试后才能达到“待发布”等级。进入制品晋级选项后，前往“制品晋级规则”页创建晋级规则。

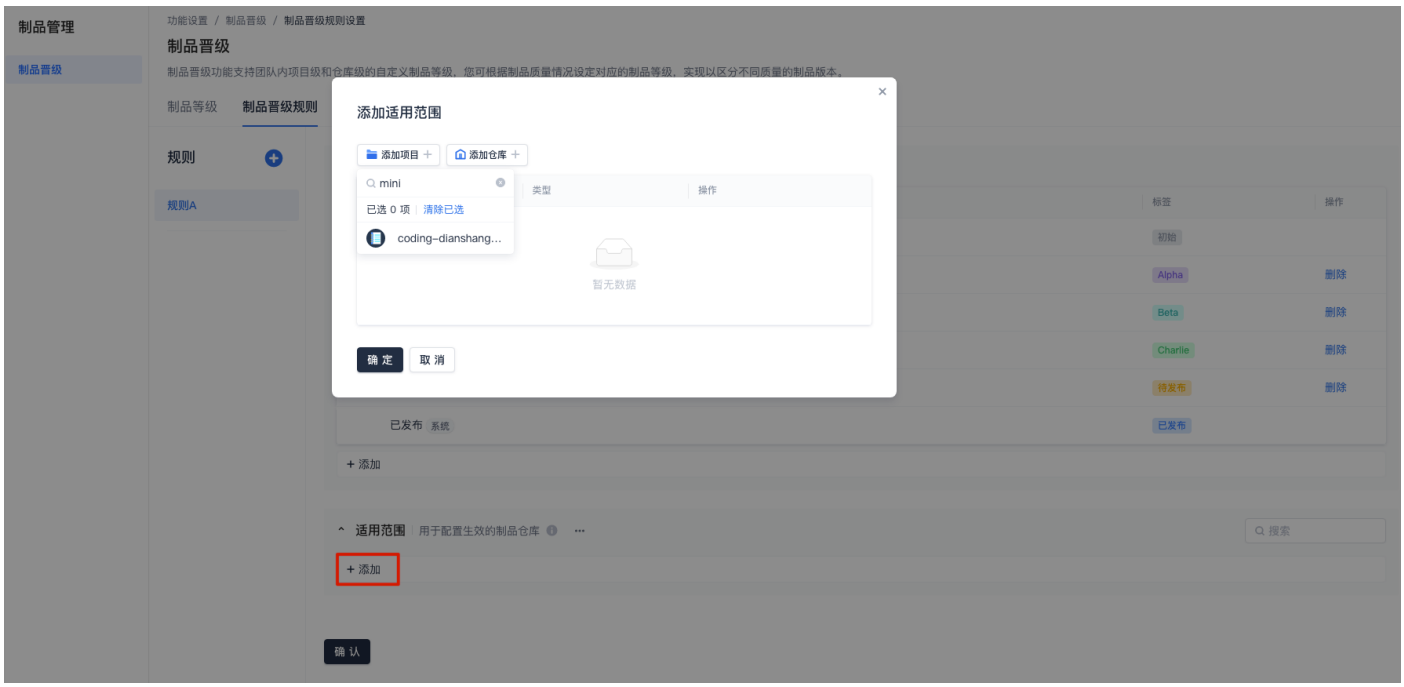
1. 将制品等级圈选至“晋级规则”中。



2. 拖拽左侧按钮调整等级的流转顺序。

3. 勾选适用该晋级规则的制品仓库。完成后单击下方的**确认**进行保存即可。

> **说明：**
> 单个项目或制品仓库支持应用多项规则，在实际的制品流转过程中任意选择一项等级。

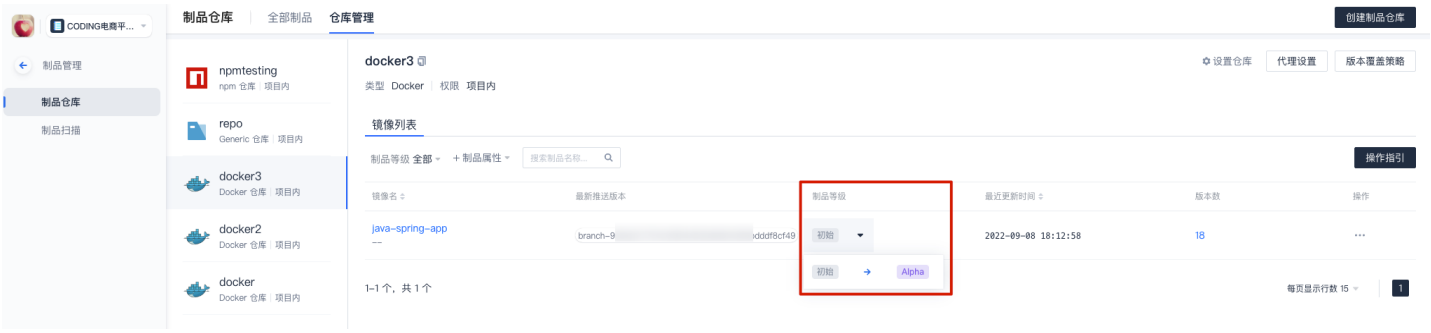


应用制品晋级规则

在制品管理中的制品列表与制品历史版本页调整制品等级，方便团队成员掌握制品目前的所属状态，快速了解当前版本的制品质量是否达到了待发布等级。

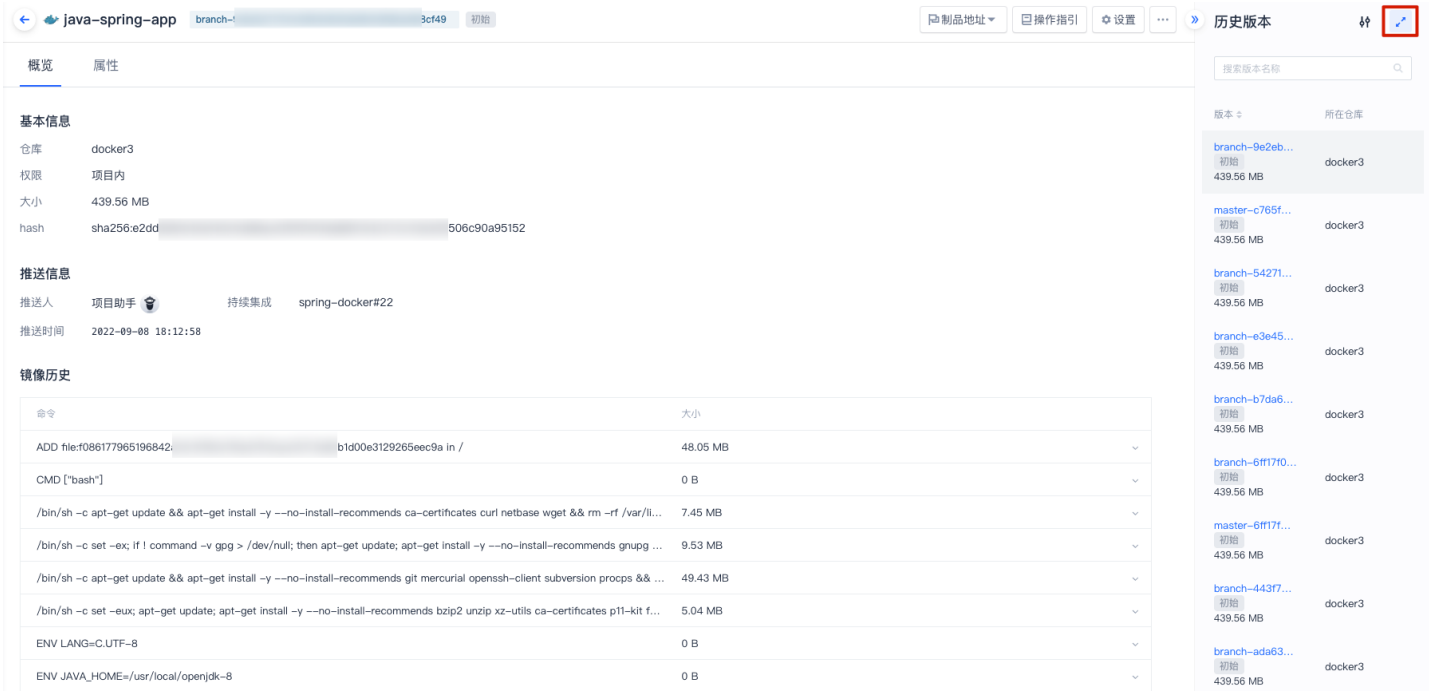
制品列表

进入仓库管理页，在列表中手动调整制品等级。

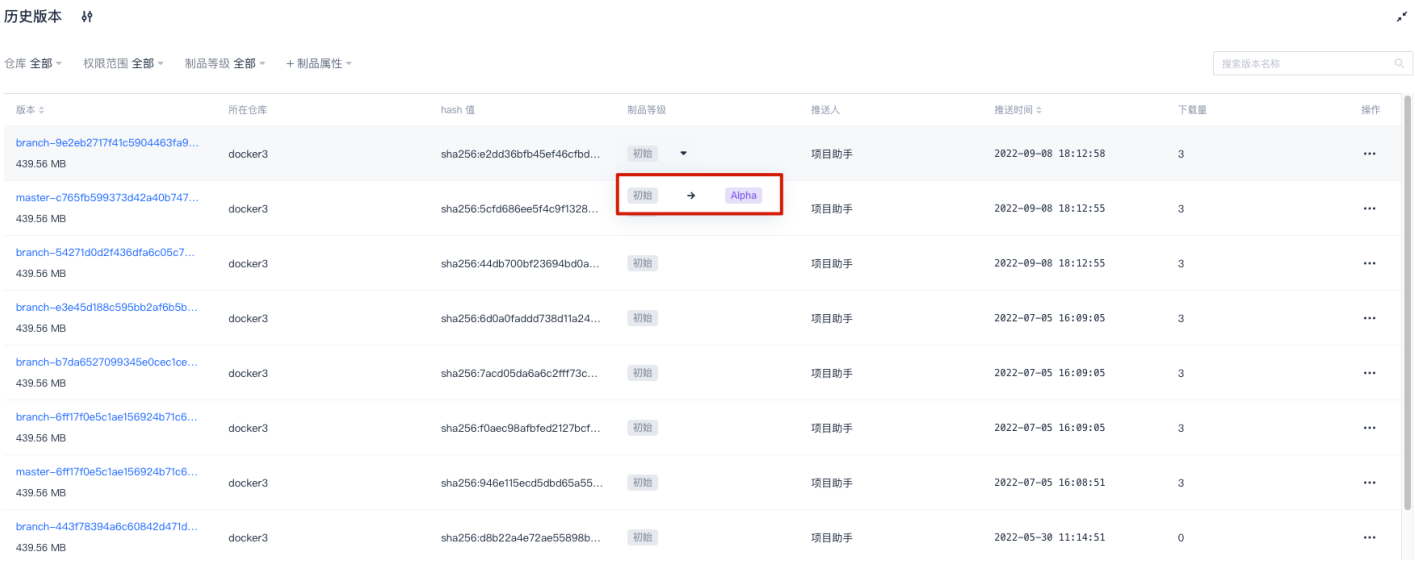


历史版本

单击制品概览右上角的全屏按钮，展开查看制品的历史版本。



在制品等级中手动调整制品状态。



制品版本覆盖策略

最近更新时间：2022-12-28 16:44:15

本文为您介绍制品库中的版本覆盖策略。

进入制品库功能页

1. 登录 CODING 控制台，单击**立即使用**进入 CODING 使用页面。
2. 单击团队首页左侧的**项目**，进入项目列表页，选择目标项目。
3. 单击左侧菜单栏的**制品管理**，进入制品库功能页面。

功能介绍

在开发阶段，每一次的代码修改都有可能会产生新的制品。开发人员可能需要依赖方频繁修改版本号来使用最新的版本。这将会非常不利于开发调试，因为如果在生产阶段随意覆盖同一个制品的版本，可能会带来管理上的混乱。**让制品拥有唯一的版本号，可以保障同一个版本的制品永远保持相同的行为，这对于部署及应用生命周期管理而言都非常有意义。**

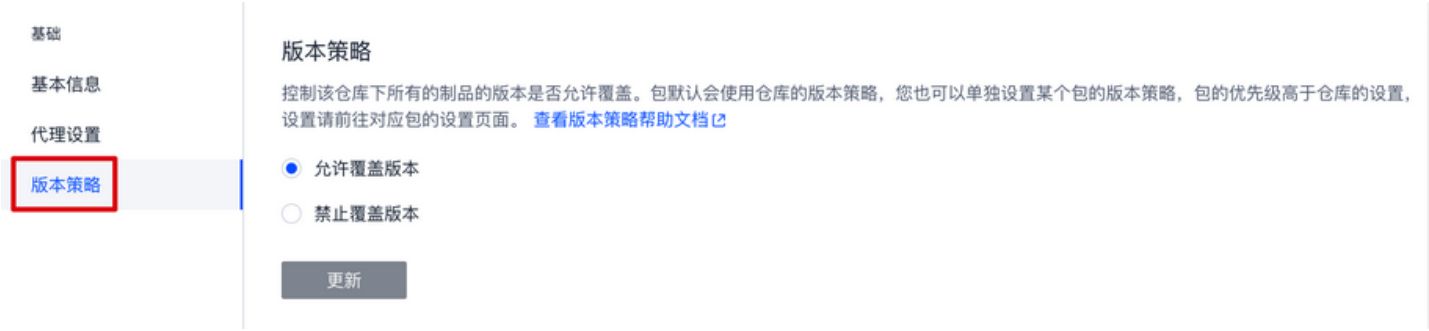
CODING 制品库提供了灵活的版本覆盖策略，可以保障 Docker 镜像版本的唯一，也可以重复发布同一个 npm 包的版本。您可以根据需要，针对制品的生命周期，设置**仓库**、**包**、**版本**的策略。

接下来本文按照这三个层级来介绍如何设置制品版本覆盖策略、以及制品库提供的默认版本覆盖策略。

仓库的版本覆盖策略

单击制品库 > 设置仓库。

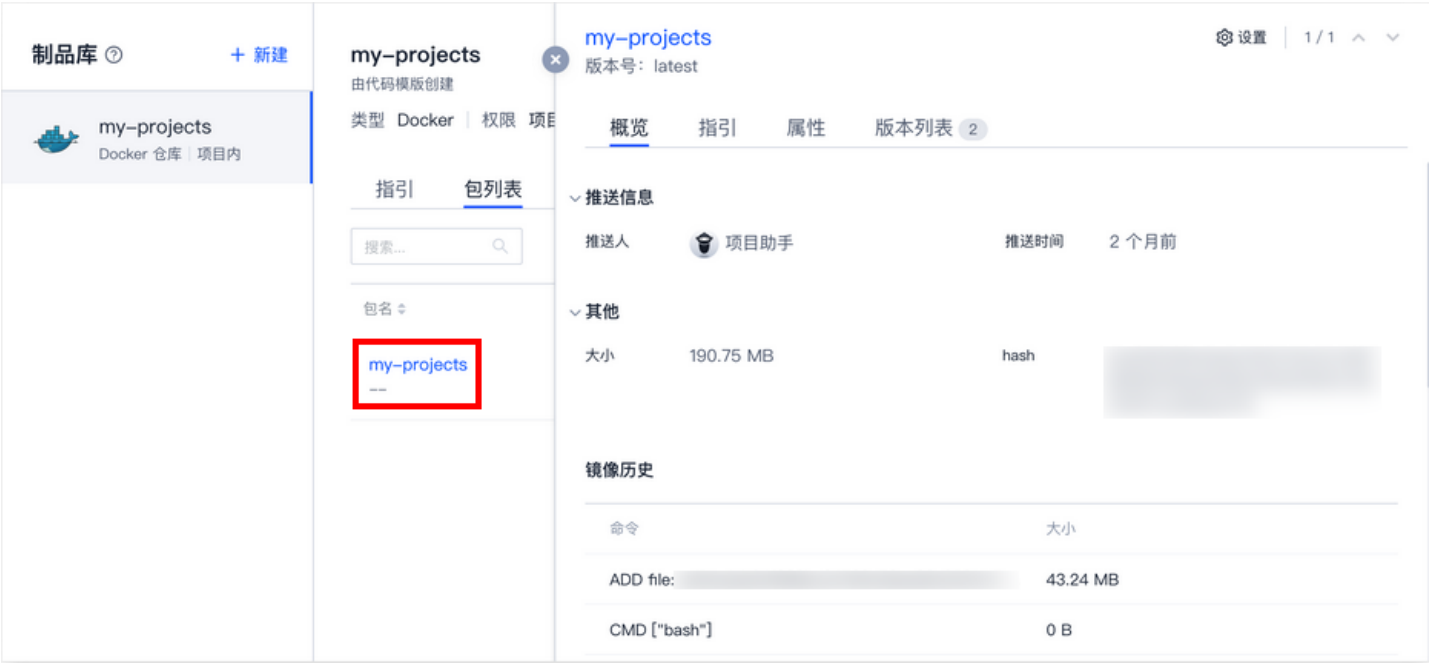
单击**版本策略**，此处可设置该仓库下所有制品的版本是否允许覆盖。



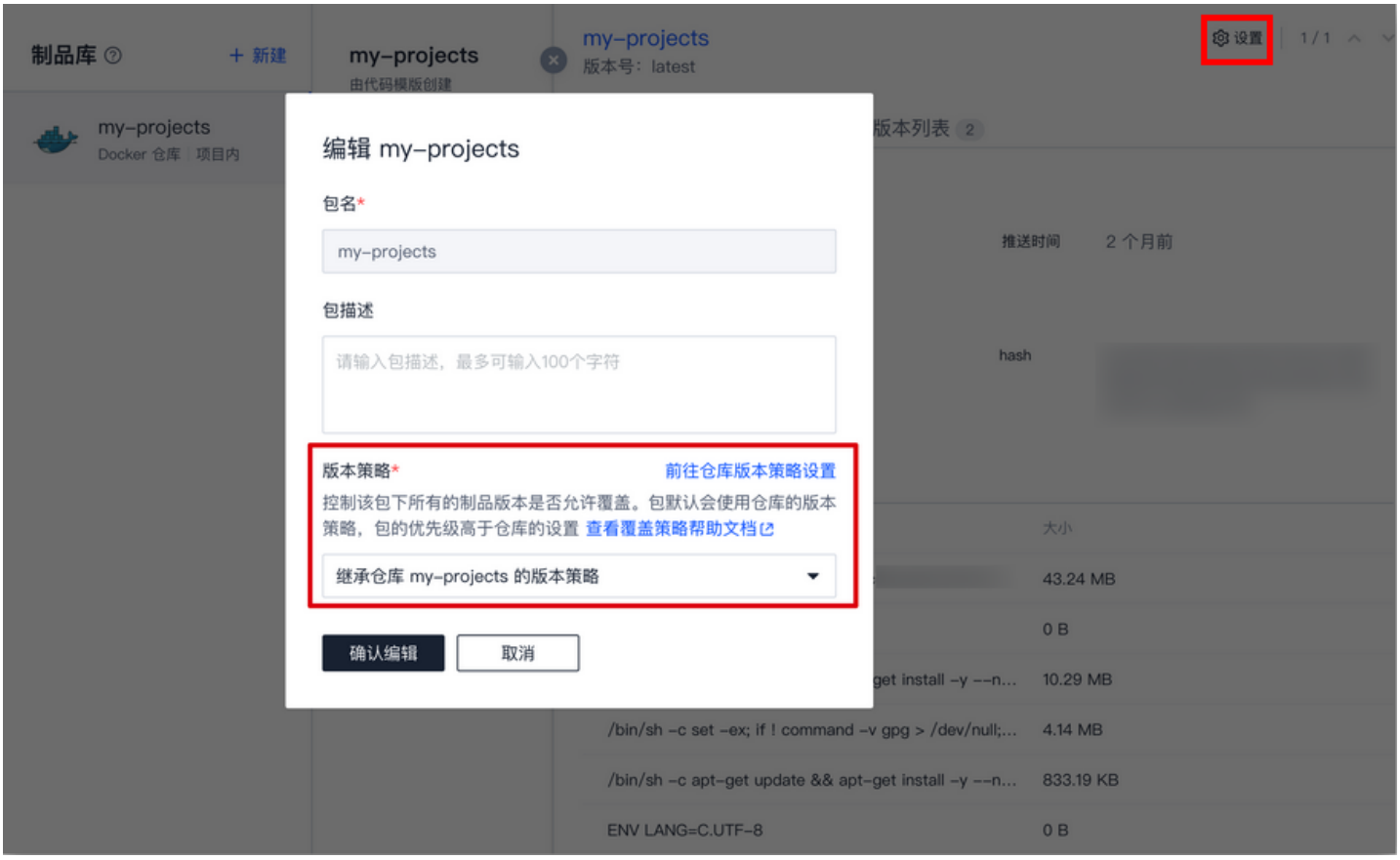
> ⚠ **注意：**
>
> 目前 Maven 制品库较为特殊，多了一项：使用 Maven SNAPSHOT（快照）。

包的版本覆盖策略

单击具体包名可看到右侧的包详情页面。



单击**设置**即可对包的版本策略进行选择，默认情况下包使用本仓库的版本覆盖策略。

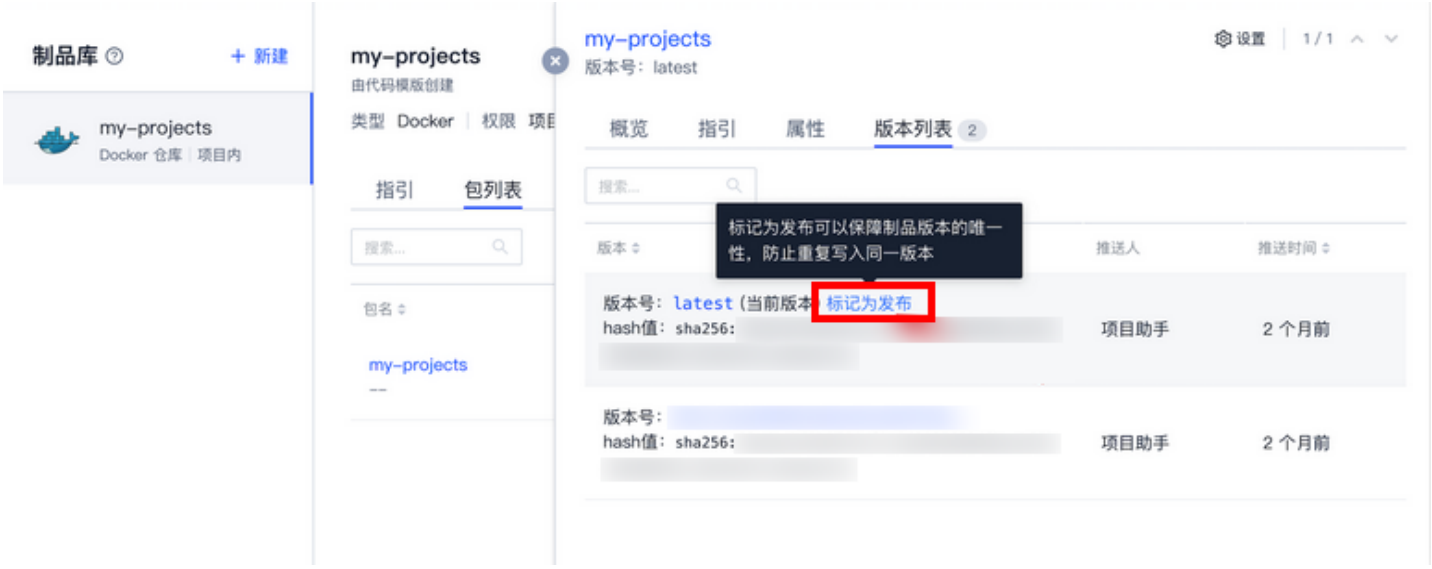


版本的发布策略

用户可以将某个版本设置为**已发布**，标记为发布可以保障制品版本的唯一性，防止重复写入同一版本。

> **注意：**
> 发布操作是不可逆的，当一个版本被设置为发布后，将无法取消发布状态。

将鼠标停放在某个版本上，会显示**标记为发布**按钮。



单击**标记为发布**按钮后，出现发布版本弹窗，单击**确认**即可。

发布成功后，该版本会显示一个**已发布**标签。

默认的版本覆盖策略

制品库按照该制品类型的原生逻辑，提供了默认的版本覆盖策略，详情如下：

制品类型	仓库	包	版本
Docker	允许发布相同版本	继承仓库规则	未发布
Maven	Maven SNAPSHOT	继承仓库规则	未发布
npm	不允许发布相同版本	继承仓库规则	未发布
PyPI	不允许发布相同版本	继承仓库规则	未发布
Generic	允许发布相同版本	继承仓库规则	未发布
Helm	允许发布相同版本	继承仓库规则	未发布
Composer	不允许发布相同版本	继承仓库规则	未发布
NuGet	不允许发布相同版本	继承仓库规则	未发布
Conan	允许发布相同版本	继承仓库规则	未发布

版权所有：腾讯云计算（北京）有限责任公司

清理策略

最近更新时间：2022-12-28 16:44:20

本文为您介绍制品库中的清理策略，能够帮助您及时清理老旧版本的制品。您可以通过设置清理策略快速清理多余制品，释放储存空间。

进入制品库功能页

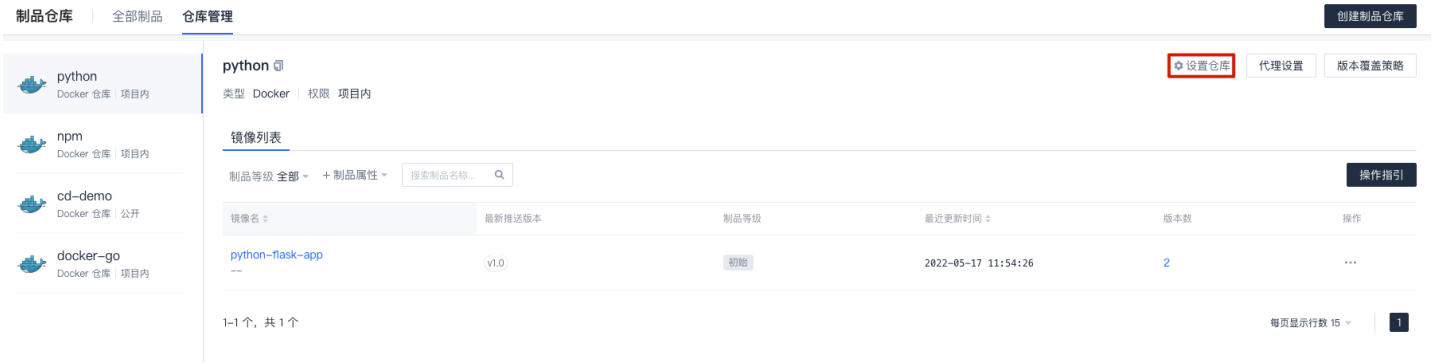
1. 登录 CODING 控制台，单击**立即使用**进入 CODING 使用页面。
2. 单击团队首页左侧的**项目**，进入项目列表页，选择目标项目。
3. 单击左侧菜单栏的**制品管理**，进入制品库功能页面。

配置清理策略

目前支持清理策略设置的制品类型：

- Docker
- Generic
- Gradle
- Helm
- npm

进入上述三种类型的制品仓库页后，单击**设置仓库** > **清理策略**。



您需要先填写清理设置中的两个触发条件，再选择执行自动或手动清理，只有同时符合两个触发清理条件的制品才会被纳入清

理列表。

查看清理记录

单击团队首页左侧的团队设置中心，在日志中查看制品仓库的操作日志。

在制品仓库日志页将显示近期的制品仓库日志。

制品库代理

最近更新时间：2022-12-28 16:45:18

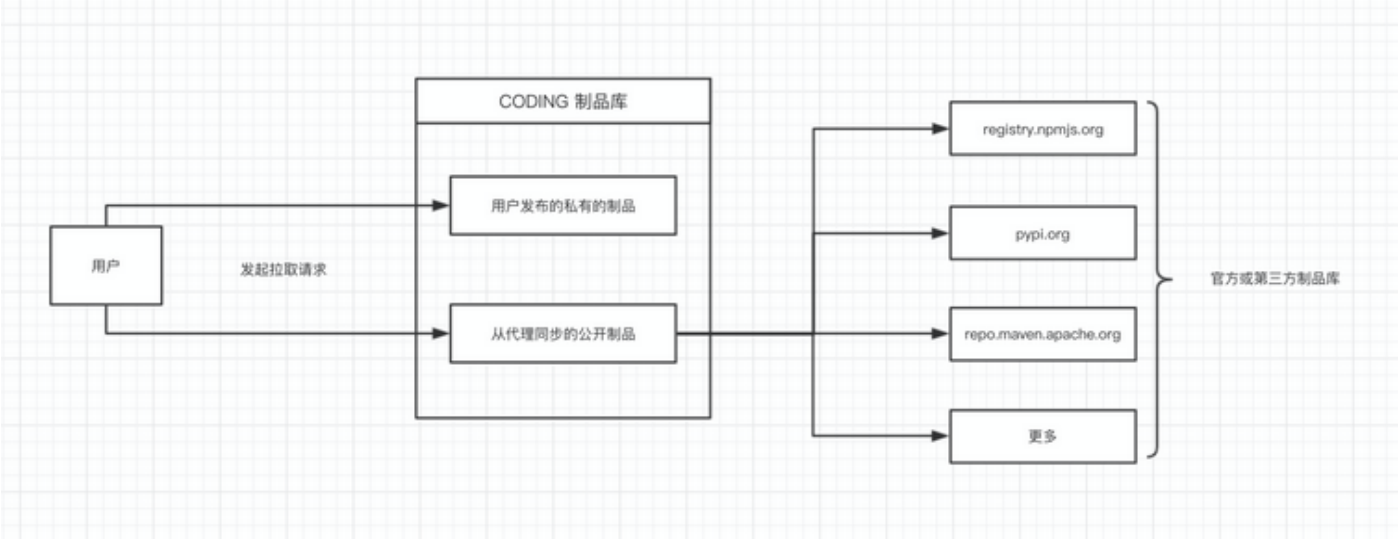
本文为您介绍如何使用制品库中的代理功能。

进入制品库功能页

1. 登录 CODING 控制台，单击**立即使用**进入 CODING 使用页面。
2. 单击团队首页左侧的**项目**，进入项目列表页，选择目标项目。
3. 单击左侧菜单栏的**制品管理**，进入制品库功能页面。

功能介绍

制品库的代理功能支持用户配置仓库代理多个源，当私有仓库内找不到对应的包时，会尝试去配置的源拉取对应的包返回给用户。同时也支持用户配置代理源认证的账号信息。制品库代理功能可作为统一入口帮助用户管理依赖的第三方制品。该功能仅支持 Maven、npm、PyPI 和 Composer 制品库。



> 说明：
>
> 制品拉取的顺序
>
> 1. 优先获取私有仓库内的包。
> 2. 私有仓库内无法找到时，再从配置代理的源按照从上到下顺序查找。

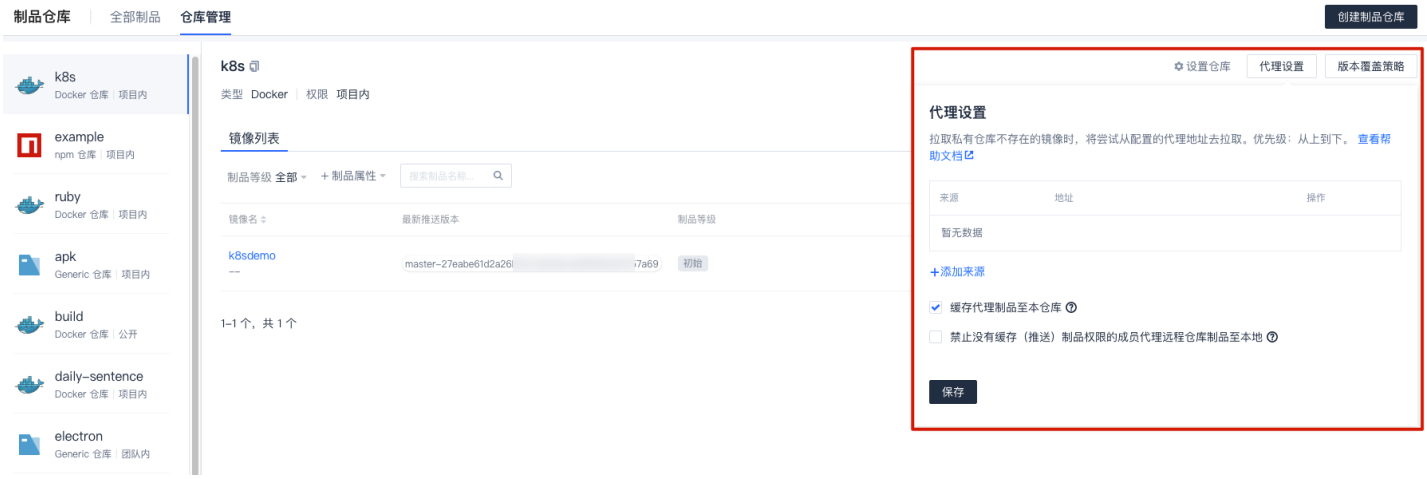
开启代理

在**制品库**下新建制品库时，可选择**启用代理**，默认此项打开。

配置代理

在制品库列表页面，单击右上角按钮**设置仓库**进入某个制品库的设置页面。

在制品库设置页面，单击**代理设置**，可以添加/删除代理来源、调整代理来源优先级、配置鉴权信息。



添加来源

在单击**添加来源**按钮后，进入创建来源页面，填写地址、名称，如有必要再填写配置鉴权信息。单击**添加**按钮即可。

修改鉴权

如需修改代理源的鉴权信息，在代理源列表页面，单击**配置**按钮，即可进行修改。

制品库内置的代理地址如下：

类型	名称	地址
npm	npmjs	访问地址
	cnpm	访问地址
	TencentCloud npm	访问地址
PyPI	PyPI	访问地址
	TencentCloud PyPI	访问地址
Maven	Maven Central	访问地址
	TencentCloud Maven	访问地址
	JCenter	访问地址
Composer	Aliyun Composer	访问地址

常见问题

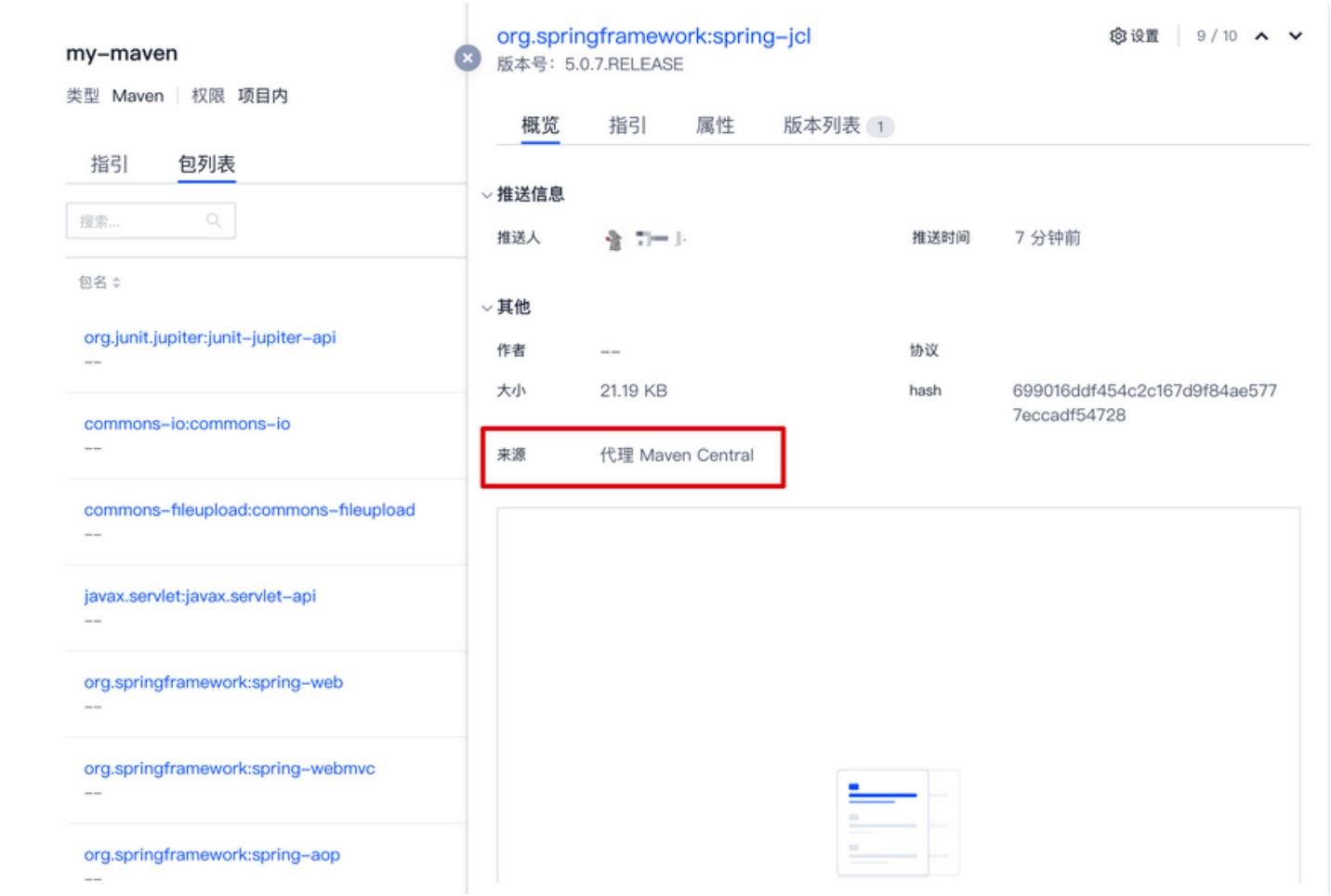
在代理配置成功后就可以使用代理源拉取依赖了。但是如何识别制品库中的制品来源是不是从代理同步而来的?

1. 以 Maven 类型制品为例，您可以在本地执行 maven install 时看到类似如下的制品拉取日志：

```
[INFO] Downloading from：https://xxxxxxxx-maven.pkg.coding.net/repository/coding-demo/my-maven/org/springframework/spring-jcl/5.0.7.RELEASE/spring-jcl-5.0.7.RELEASE.pom
[INFO] Downloaded from：https://xxxxxxxx-maven.pkg.coding.net/repository/coding-demo/my-ma
```

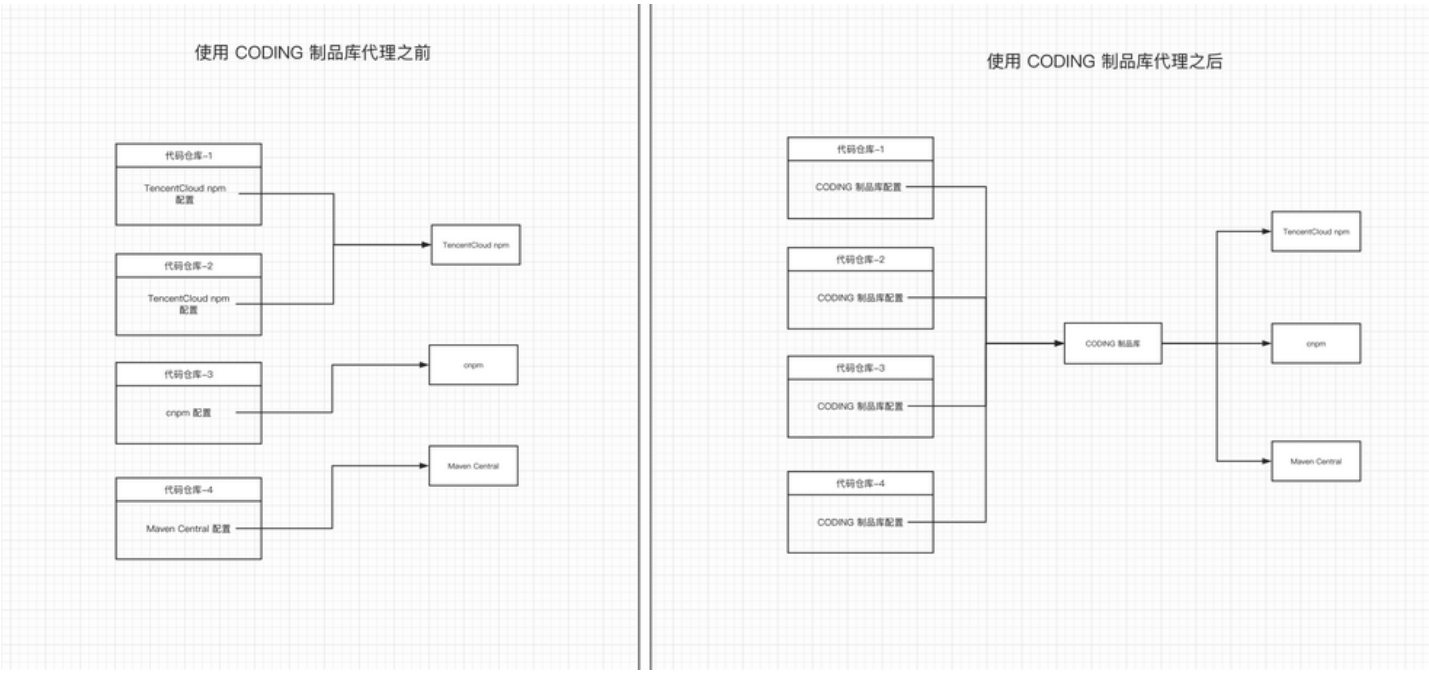
```
ven/org/springframework/spring-jcl/5.0.7.RELEASE/spring-jcl-5.0.7.RELEASE.pom (1.9 kB at 735 B/s)
```

2. 同时，在 CODING 制品库上，您也可以看到该制品的来源。



直接从第三方制品源拉取制品和通过 CODING 制品库代理拉取有什么区别?

制品库可以帮助您统一管理团队内的制品源配置，您可以在 CODING 制品库内追踪团队内成员的使用情况，也可以通过 CODING 制品扫描统一检测出有安全缺陷，直接对团队内的制品安全进行审计。


使用 CODING 制品库代理之前
代码仓库-1
TencentCloud npm 配置
TencentCloud npm
代码仓库-2
TencentCloud npm 配置
代码仓库-3
cnpm 配置
cnpm
代码仓库-4
Maven Central 配置
Maven Central
使用 CODING 制品库代理之后
代码仓库-1
CODING 制品库配置
代码仓库-2
CODING 制品库配置
代码仓库-3
CODING 制品库配置
代码仓库-4
CODING 制品库配置
CODING 制品库
TencentCloud npm
cnpm
Maven Central

制品扫描
功能介绍

最近更新时间：2022-12-30 19:32:58

本文为您介绍制品仓库中的扫描功能。

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. 单击团队首页左侧的**项目**，进入项目列表页，选择目标项目。
3. 选择左侧项目菜单栏的**制品管理** > **制品扫描**。

> 说明：
> 制品扫描为旗舰版功能。

功能介绍

CODING 制品仓库的扫描功能可以在不访问源代码的情况下，通过扫描二进制组件及其元数据，找寻组件中存在的漏洞。制品扫描功能支持与持续集成和持续部署模块相集成。您可以在方案中预设质量红线标准，杜绝问题组件发布至生产环境。同时，扫描方案还支持提供详细扫描记录和缺陷统计。

产品功能

扫描方案

用于规定扫描规则、质量红线，以及被执行扫描的制品包。扫描方案只能应用于当前项目内的制品仓库，每个扫描方案都有唯一的扫描 ID。扫描方案可以应用于当前项目下的任意仓库、任意制品包与任意制品版本。触发方式可以选择制品包更新时自动触发或选择制品版本手动触发。

扫描规则

用于规定扫描时关注的制品缺陷内容，可以指定关注的漏洞等级并设置漏洞白名单。

- 漏洞危险等级规则
  定义该扫描方案关注的漏洞危险等级。若不勾选 “低危” 等级，则所有低危等级的漏洞都不会被扫描出来，不计入漏洞统计。

- 漏洞白名单（仅安全扫描）
  将具体的 CVE 漏洞编号纳入至白名单后，所有公共组件中如果存在该编号的 CVE 漏洞都不会被扫描；将具体所属依赖组件列入白名单后，该组件中的所有漏洞都不会被扫描。

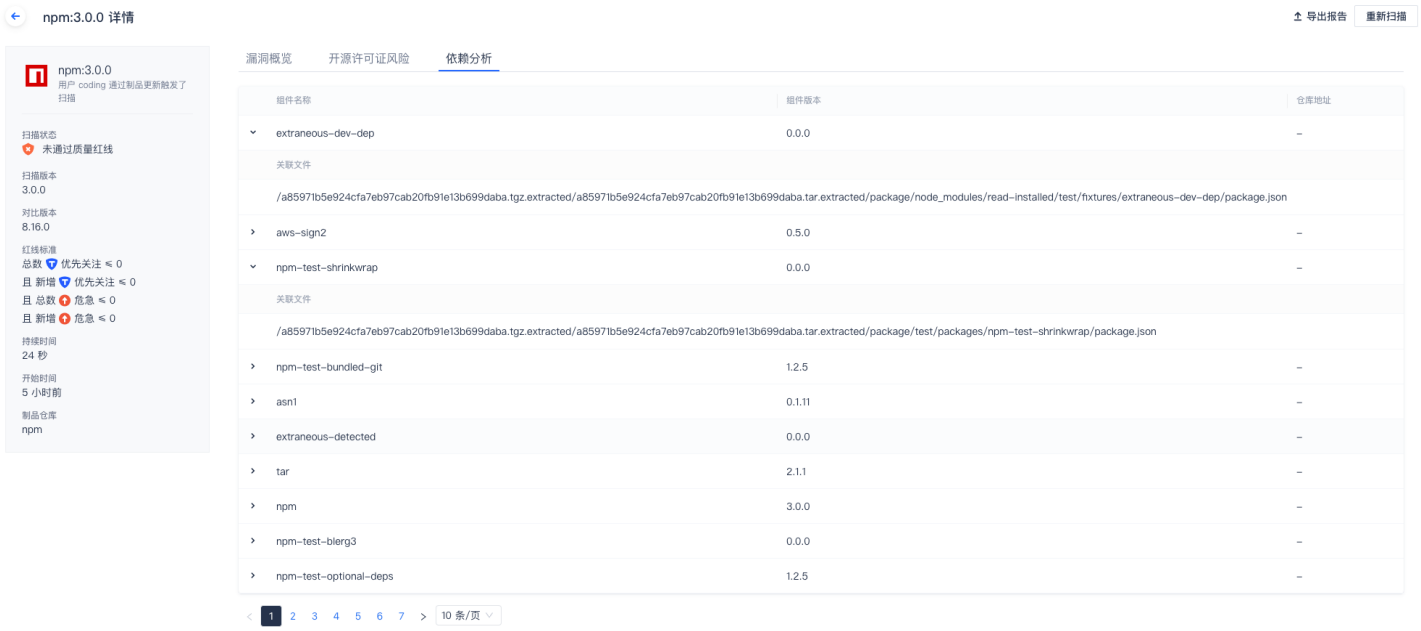
质量红线

用于规定扫描结果是否通过的标准。您可以根据团队的安全需求定义红线标准，包括对漏洞、开源许可证等级和数量限制。

依赖分析

> **说明：**
> 目前仅针对 Generic、Maven、Docker 和 npm 类型制品提供分析功能。

依赖分析功能可以揭示制品中所使用的依赖关系，默认内置在制品扫描中。触发扫描方案后将同时启用依赖分析功能，分析结束后将展示依赖列表、组件名称、组件版本及组件所在仓库。



开源许可证风险

开发者在使用开源软件的过程中需注意许可证是否存在违规风险。制品扫描在执行漏洞扫描的同时也会对开源组件许可证进行扫描，展示许可证风险等级、来源、约束与关联组件等信息。

方案应用

单个扫描方案可以复用于各类型的制品仓库中的任意制品版本，触发方式支持制品包更新时自动触发与手动触发。

方案类型

扫描方案支持安全漏洞扫描与移动端安装包质量检查两种类型。移动端安装包质量检查方案由**腾讯云安装包质量检查 IPT** 提供主要能力。除了两种类型外，制品扫描还将继续拓展更多制品扫描能力。

- **安全漏洞扫描**
  扫描制品及其依赖的公共组件中存在的安全漏洞。漏洞的定义采用 CVE 国际通用标准及评级；制品扫描采用的组件漏洞特征库由云鼎实验室维护，为腾讯安全提供超过 20 年的行业经验积累，由专门的安全运营团队实时更新。
  支持以下类型制品仓库：Docker、Maven、npm、PyPI、Generic。
- **移动端安装包质量扫描**
  对 Android / iOS 操作系统的安装包进行安装包大小、重复文件、图片等进行规范检查。
  支持 Generic 类型制品仓库中 .ipa 和 .apk 格式的制品文件。

安全漏洞

漏洞的定义采用 CVE 国际通用标准定义，其危险等级遵循 CVSS 标准。CODING 制品扫描通过对制品及其依赖解析，暴露该制品存在的安全漏洞。漏洞信息包括 CVE 编号、引入依赖组件、危险等级及漏洞描述等。

漏洞的定义由 CVE 编号及其存在的公共组件共同决定。例如，我们在公共组件 com.alibaba.fastjson:1.1.15 中，发现了编号为 CVE-2017-18349 的安全漏洞，则这条漏洞在扫描结果中会显示为：

- CVE 编号：CVE-2017-18349
- 所属依赖：com.alibaba.fastjson
- 引入版本：1.1.15

制品漏洞库

CODING 制品扫描采用腾讯安全漏洞特征库。

腾讯安全漏洞特征库是腾讯科恩实验室自研的、长期维护的商业化制品漏洞库。在漏洞信息方面，包含了国内外的开源漏洞库信息（包括：CVE、NVD、CNVD、 CNNVD 等）与腾讯自身发现的独有商业漏洞信息。腾讯安全团队对开源漏洞信息进行了误报校验与中文翻译，同时提供修复优先级与修复建议信息，扫描准确性行业领先。对于自身发现的独有商业漏洞信息，提供独有漏洞 ID、漏洞详情描述、漏洞修复建议等内容，涵盖漏洞基本信息。

对于私有化客户，我们提供了实时更新和离线更新两种更新方式。

实时更新

在实时更新方面，支持在本地漏洞库配置国内互联网更新地址 (<https: bsca.tencentcloudapi.com="">) 和腾讯云账号密钥认证，在每次扫描执行的时候，会检查漏洞库是否为最新版本。如果不是最新版本，则会先更新漏洞库再执行扫描。

离线更新

在离线更新方面，也支持将漏洞库离线打包部署至企业内网，提供更新服务。

快速上手

最近更新时间：2022-12-28 16:44:35

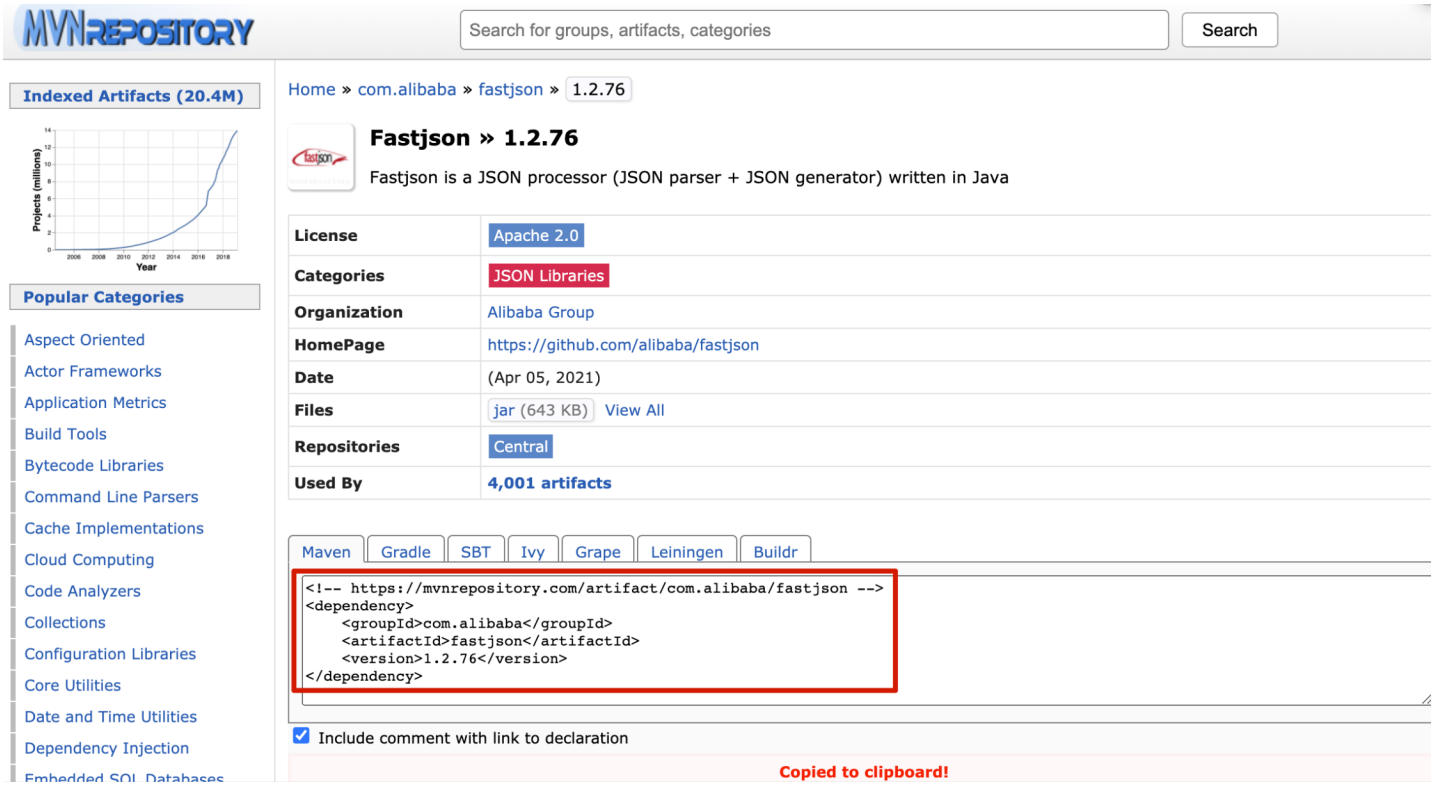
本文将以示例制品 fastjson 作为扫描对象，介绍如何使用扫描功能。

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. 单击团队首页左侧的**项目**，进入项目列表页，选择目标项目。
3. 选择左侧菜单栏的**制品管理** > **制品扫描**。

本文以示例制品 fastjson 作为扫描对象，若项目内已有制品，可以直接进行扫描。

得益于 CODING 制品代理功能，在拉取制品至本地时将自动上传至制品仓库。单击 示例制品 获取相关信息，若不清楚如何使用 Maven 制品仓库，请参见 快速开始。



创建扫描方案

前往**制品管理** >**制品扫描**，单击左上角的蓝色 + 号创建一个新的扫描方案，输入扫描方案名称、描述、扫描规则和质量红线标准，即可完成创建。


创建扫描方案
规则配置
基础信息
方案名称*
我的扫描方案-1
方案类型
安全漏洞扫描
方案描述
请输入仓库描述，最多可输入100个字符
扫描规则
漏洞等级 *
危急
高危
中危
低危
CVE 漏洞白名单
请填入 CVE 漏洞编号，如 CVE-2020-9794
+ 添加 CVE 漏洞白名单
质量红线
红线标准
危急漏洞
总数 <=
0
且
高危漏洞
总数 <=
0
且
危急漏洞
新增 <=
0
且
高危漏洞
新增 <=
0

方案类型

扫描方案支持安全漏洞扫描与移动端安装包质量检查两种类型；使用安全漏洞扫描类型能够筛选漏洞等级与 CVE 漏洞白名单。移动端安装包质量检查方案由**腾讯云安装包质量检查 IPT** 提供主要能力，请参见 功能介绍。

扫描规则

扫描规则决定了该扫描方案中能够被检查出来的漏洞。若仅勾选了**高危**的漏洞等级，则其他等级的漏洞将不会被统计，即使存在高于**高危**等级的漏洞也不会被纳入检查。

CVE 漏洞白名单

用于在扫描过程中忽略特定类型的漏洞。您可以给某种类型的漏洞定义一个 CVE 漏洞编号并在白名单处填写，若制品存在 CVE-2020-9794 类型的漏洞则不会被纳入统计范围，单击查看 CVE 漏洞收录。

⚠ 注意：

> 在扫描过程中，漏洞扫描仅会在扫描规则中勾选的等级范围中进行，其他等级均会被过滤，然后才会判断 CVE 漏洞白名单中规定的具体漏洞编号。例如用户在漏洞扫描中只选择了 “危急” 的漏洞等级，而又在 CVE 漏洞白名单中填入了一个 “低危” 等级的漏洞，此等级的漏洞会被忽略。

高级选项

在高级配置中可以自动禁止未扫描完成或存在质量问题的制品被下载，防止存在漏洞的制品被意外使用。

编辑扫描方案

对于已创建的扫描方案，单击右上角的**设置**按钮进入设置页面。

单击**规则配置**也可以进入设置页面。

您可以在此处编辑方案名称或描述、查看重新勾选扫描规则中的漏洞等级和质量红线标准。

触发扫描方案

扫描方案支持自动与手动两种触发方式。

自动扫描

单击右上角的 ··· 按钮，进入 方案应用页，您可以在此处配置自动触发扫描方案。

打开自动扫描开关后，当前方案所应用的制品仓库有更新时将自动触发扫描。扫描筛选默认为**全部**，即任一制品仓库有更新时会自动触发此扫描方案。

勾选**按条件筛选**后可以为扫描方案设置应用范围与触发条件。例如下图，当 pypi-go 和 write-go 制品仓库有任何制品更新时就会自动触发该扫描方案。

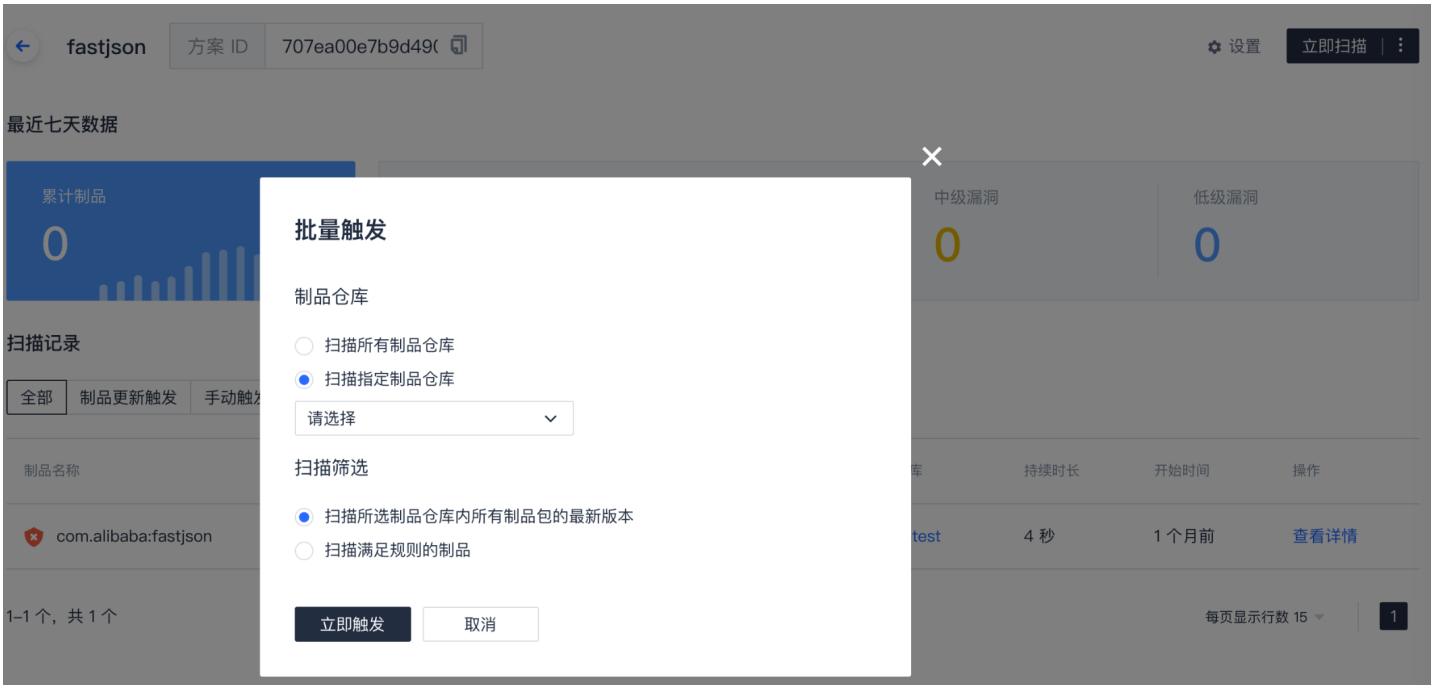
您还可以对扫描方案添加更加细致的制品筛选条件，如下图当 test 制品的 release 版本有更新时，将对此制品单独进行扫

描。

手动扫描

有三种手动触发扫描的方式：触发单个扫描方案、批量触发制品扫描与在持续集成流水线中添加制品扫描。

单击扫描方案右上角的批量扫描可以在全部制品仓库中执行扫描，或设置扫描范围与扫描筛选条件。



在持续集成流水线中也支持手动添加制品扫描单元。

分析扫描结果

图标标识

各标示结果的图标意义：

将鼠标移至标示处将会弹窗扫描结果。

查看扫描结果

进入扫描方案详情，您可以看到该扫描方案被应用的所有历史累计数据和扫描记录。

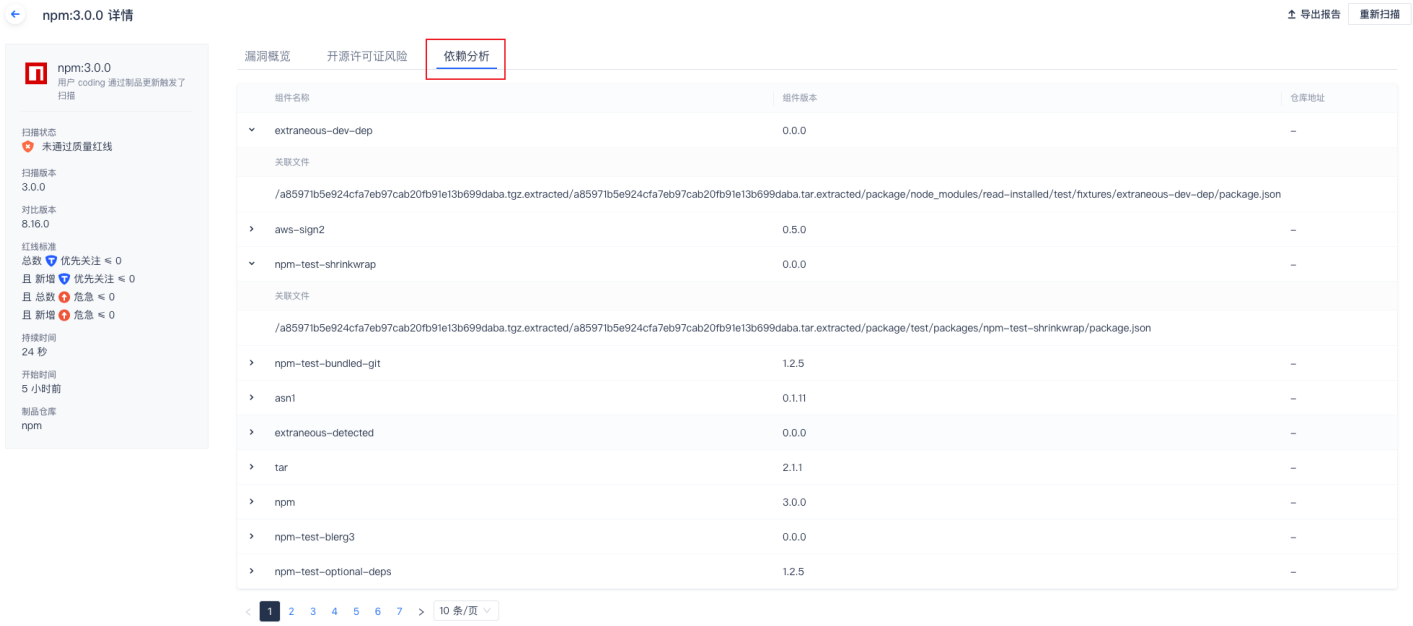
开源许可证风险

单击**开源许可证风险**可以查看该制品所有组件的开源 license 名称、风险等级、来源信息与关联组件等。

依赖分析

> **说明：**
>
> 目前仅针对 Generic、Maven、Docker、npm 类型制品提供分析功能。

触发扫描方案会自动对制品进行依赖分析，分析完成后将展示该制品所有依赖组件的名称、版本、组件所在仓库地址等信息。

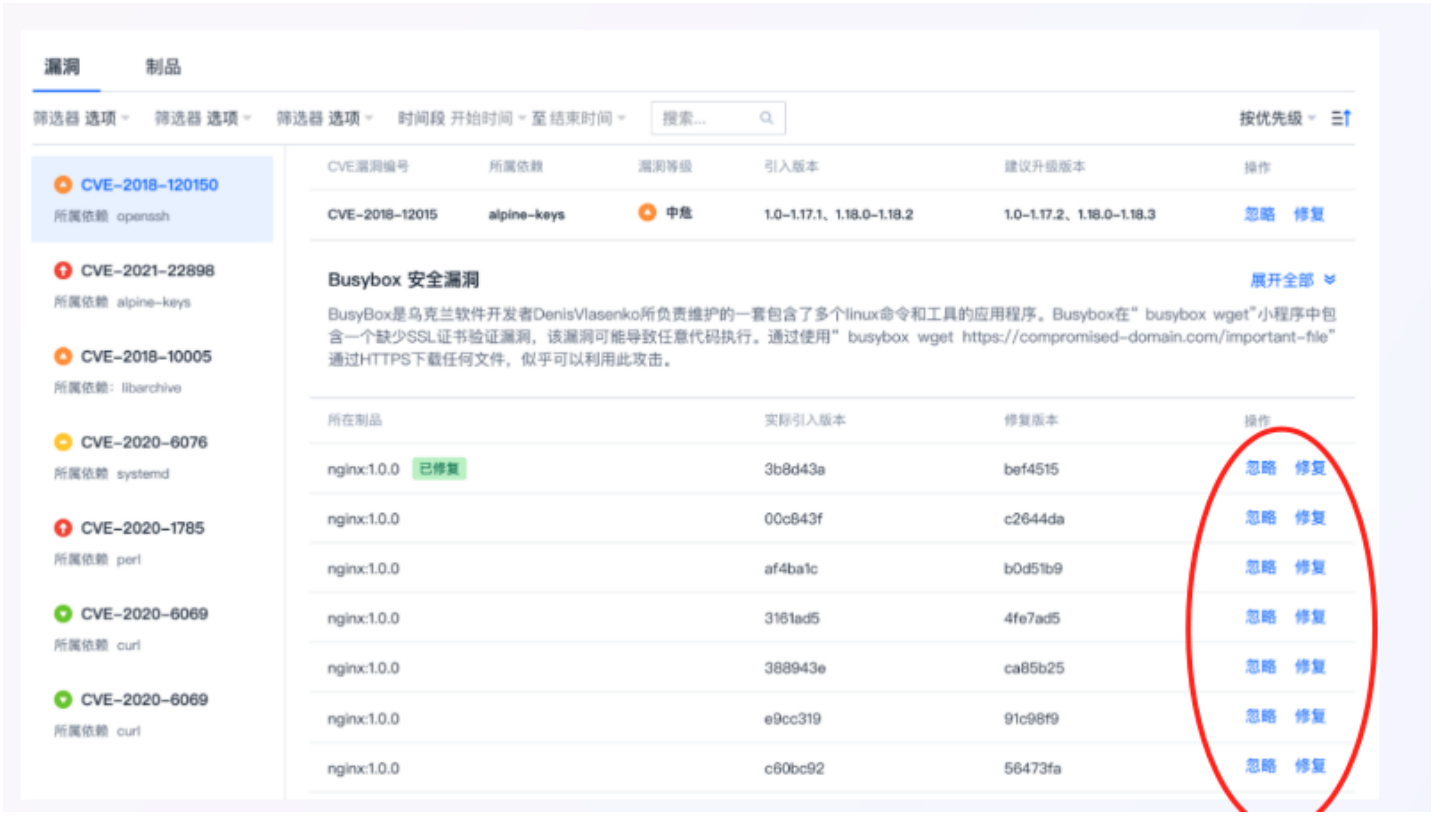


解决漏洞并记录

单击“查看详情”或制品名称即可查看该制品版本的漏洞详情。详情页展示了所有漏洞的详细信息，包括漏洞编码、等级、所属依赖、版本、修复建议等。

针对已暴露的漏洞，您可以按照漏洞概览中的修复建议进行漏洞修复。

若漏洞并不会造成较为严重的后果，可以将其记录为忽略；若已对漏洞采取修补动作，可以将其记录为修复。



若希望上手扫描更多制品，请参见 扫描对象。

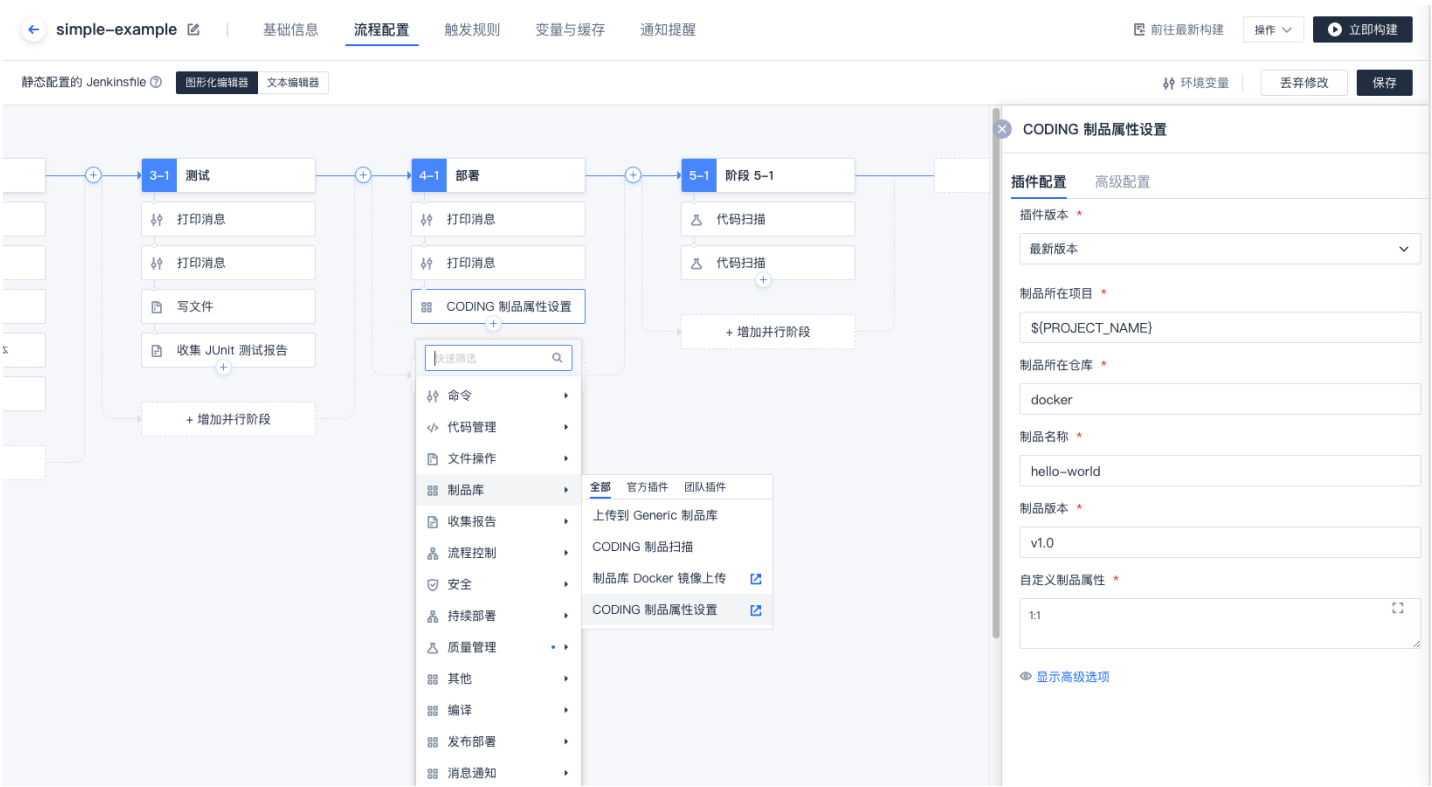
自动化插件
在持续集成中推送制品

最近更新时间：2022-12-28 16:44:59

本插件主要用于将构建过程中的代码仓库、分支、commit、构建计划、测试结果等关键信息写入制品属性。

快速开始

进入持续集成设置中的流程配置，选择 CODING 制品属性设置，在插件配置中按照提示填写相应的值。



Jenkinsfile

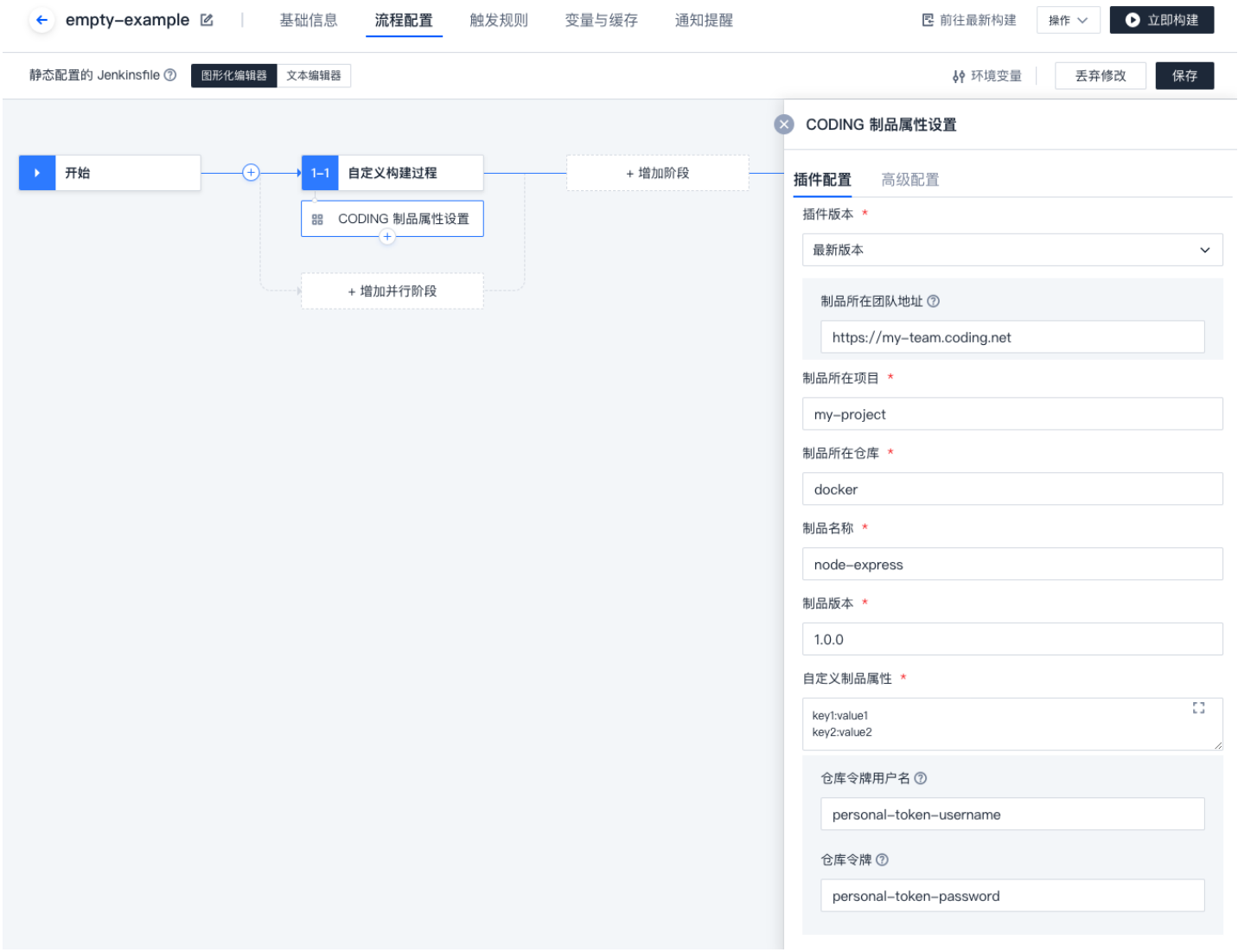
您也可以参考下列命令在文本编辑器中填写命令。

```
stage {
steps {
useCustomStepPlugin(
key: 'coding-public:artifact_set_properties',
version: 'latest',
params: [
repo: 'docker'
artifactName: 'node-express',
```

```
version: "1.0.0",
properties: """key1:value1\nkey2:value2\nkey3:value3"""
]
)
}
}
```

详细参数

本插件还支持填写其他 CODING 团队的制品仓库地址（权限需开放访问）或 WePack 制品仓库的制品属性。




```
stage {
steps {
useCustomStepPlugin(
key: 'coding-public:artifact_docker_push',
version: 'latest',
```

```
params: [
repo: 'docker'
artifactName: 'node-express',
version: "1.0.0",
properties: """key1:value1\nkey2:value2\nkey3:value3"""
// 远程系统仓库时用的 Docker 仓库地址
systemHost: 'https://my-team.coding.net',
// 项目名 / 命名空间名
project: 'my-project',
// 拥有权限写入制品属性的用户名 / 密码 / token
username: 'personal-token-username',
password: 'personal-token-password'
]
)
}
}
```

参数详情

参数	描述	默认值	备注
artifactName	您需要写入属性的制品名称，例如 node-express	-	必填
project	制品所在项目	${PROJECT_NAME}	非必填；请注意将项目名称改为小写
repo	制品所在仓库	-	必填
properites	写入制品的属性	-	非必填

参数	描述	默认值	备注
username	拥有权限写入制品属性的用户名	${PROJECT_TOKEN_GK}	请输入项目令牌或个人令牌的用户名
password	拥有权限写入制品属性的密码	${PROJECT_TOKEN}	请输入项目令牌或个人令牌
systemHost	其他团队或远程 WePack 系统地址	${CCI_CURRENT_TEAM}.${CCI_CURRENT_DOMAIN}	非必填

自动填充制品属性

最近更新时间：2023-05-21 13:26:47

本插件用于将构建环境中的 Docker 镜像自动推送至制品仓库，并支持将构建过程中的关键信息写入制品属性。

快速开始

仅需要填写构建环境中的镜像 tag，以及需要推送的仓库名称（默认推送到本项目仓库），便可推送制品到 CODING 制品仓库。

文本编辑器插件配置如下：

```
stage {
steps {
useCustomStepPlugin(
key: '64:artifact_docker_push',
version: '1.0',
params: [
image: 'node-express',
repo: 'docker'
]
)
}
}
```

详细参数

本插件还支持您将镜像推送到任意 CODING 企业或 WePack 制品仓库中，同时支持自定义制品属性。


```
stage {
steps {
useCustomStepPlugin(
key: 'coding-public:artifact_docker_push',
version: 'latest',
params: [
image: 'my-image:1.0.0',
repo: 'my-repo',
// 登陆 Docker 仓库时用的 Docker 仓库地址
host: 'myteam-docker.pkg.coding.net',
// 项目名 / 命名空间名
project: 'my-project',
// 推送至仓库所用的用户名 / 密码 / token
username: 'repo-login-username',
password: 'repo-login-password',
// 自定义属性
properties: '[{\"name\":\"key1\",\"value\":\"value1\"},{\"name\":\"key2\",\"value\":\"value2\"}]',
// 当制品仓库 host 的域名和系统域名不一致时填写，一般情况下不用填写
systemHost: 'wepack.hello.net'
]
)
```

```
}
}
```

参数详情

参数	描述	默认值	备注
image	您需要推送的当前环境中制品 tag，例如 my-image:1.0.0	-	必填；如果当前构建环境中，没有该 tag，系统会尝试匹配该制品名称的其他版本
host	推送目标制品仓库 host	${CCI_CURRENT_TEAM}-docker.pkg.${CCI_CURRENT_DOMAIN}	非必填；如果您需要推送的制品仓库 host 协议为 http，需要明确定义
project	推送目标制品仓库所在项目	${PROJECT_NAME.toLowerCase()}	非必填
repo	推送目标制品仓库	-	必填

参数	描述	默认值	备注
properites	推送到目标仓库时自动写入制品的属性	默认属性列表详见下表	非必填；如果您自定义的属性名如果与默认值重合，则自定义属性值会覆盖默认属性值
username	访问推送目标仓库认证用户名	${PROJECT_TOKEN_GK}	请输入项目令牌或个人令牌的用户名
password	访问推送目标仓库认证密码	${PROJECT_TOKEN}	请输入项目令牌或个人令牌
systemHost	团队或系统 host	${${CCI_CURRENT_TEAM}}.${CCI_CURRENT_DOMAIN}	非必填；只有当制品仓库域名和系统域名不一致时，才需要填写本字段

参数默认值相关 CODING CI 环境变量说明

环境变量名	描述
CCI_CURRENT_WEB_PROTOCOL	当前构建环境的 PROTOCOL，例如 https
CCI_CURRENT_TEAM	当前构建环境的团队名，例如 codingcorp
CCI_CURRENT_DOMAIN	当前构建环境的域名，例如 coding.net

版权所有：腾讯云计算（北京）有限责任公司

环境变量名	描述
PROJECT_NAME	当前 CI job 所在项目名，例如 coding-artifacts
PROJECT_TOKEN_GK	项目令牌用户名
PROJECT_TOKEN	项目令牌密码

properties 默认值

制品属性名	制品属性值(环境变量名)	描述
ci.project.url	PROJECT_WEB_URL	CI 所在项目 url
ci.project.name	PROJECT_NAME	CI 所在项目名
ci.job.id	JOB_ID	CI 构建计划 id
ci.job.name	CCI_JOB_NAME	CI 构建计划名称
ci.build.number	CI_BUILD_NUMBER	CI 构建记录编号
ci.build.trigger.method	CCI_TRIGGER_METHOD	CI 构建记录触发方式
ci.build.trigger.user.name	TRIGGER_USER_NAME	CI 构建记录触发者
ci.build.trigger.user.email	TRIGGER_USER_EMAIL	CI 构建记录触发 email
git.repo.http.url	GIT_HTTP_URL	构建目标代码仓库地址（https 协议）
git.repo.ssh.url	GIT_SSH_URL	构建目标代码仓库地址（ssh 协议）
git.repo.name	DEPOT_NAME	构建目标代码仓库名
git.branch	GIT_LOCAL_BRANCH	构建目标分支
git.commit	GIT_COMMIT	构建目标 commit
git.tag	GIT_TAG	构建目标 tag
git.committer.name	GIT_COMMITTER_NAME	代码提交者
git.committer.email	GIT_COMMITTER_EMAIL	代码提交者 email

版权所有：腾讯云计算（北京）有限责任公司