

制品库 操作指南 产品文档





【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

操作指南 开通用户 主账号 子用户 权限说明 制品库权限 制品库认证 制品属性 制品晋级 制品版本覆盖策略 清理策略 制品库代理 制品扫描 功能介绍 快速上手 自动化插件 在持续集成中推送制品 自动填充制品属性



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。

🗲 角色管理



```
⑦ 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。

概览





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。

请输入 邮箱		
验证码	获取验证码	
设置密码	Ø	
8~20 个字符,需同时包含数字、	字母及符号。	
	《CODING 服务协议》	
开通即表示我已阅读并同意		
开通即表示我已阅读并同意		
开通即表示我已阅读并同意 下一步		
开通即表示我已阅读并同意 下一步		
 开通即表示我已阅读并同意 下ー步 创建团队 		

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**。

概览





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。

✔ 关联 CODING 账号 ☆ 送联 CODING 账号 ※ 没有 CODING 账号? 去开通 请输入邮箱 请输入邮箱地址 验证码 ※取验证码

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。

概览

	前往 CODING DevOps 团队域名 立即使用 立即选购	
快速创建子用户		
点击以下按钮快速创建拥有 COD	DING DevOps 权限的子用户。	

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。



快速新建用户

	用广名 *	备注	手机	郎箱	
	test		中国大陆(+86) 🔻		ł
	新增用户 (单次最多创建10个用户)				
j I. •	✓ 編程(MP) 居用密码,允许用户登录到降讯云控制台 信用密码,允许用户登录到降讯云控制台	K和其他开发工具访问			
台密码・	 自动生成密码 自定义密码 				
	✔ 用户必须在下次登录时重置密码				
重置密码					

▲ 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。

既觉			
您的主账号已开通 CODING 服务,您可加 团队域名 https:// 译 加入团队	入团队		
CODING DevOps 简介			
🕺 构想 👌 近 计划 👌 🛁	コ 开发 >	[五] 测试 > [4]	交付
	文件网盘	全部文件 ↓ Q ~ 搜索全额文件 + 第	ि → 上传 → 🔲
	► 全部文件	□ 文件名 \$	创建者 ⇔ 最后更新 ≑
构想	 ● 最近文件 ★ 星标文件 	□ 研发三部产品设计	陈江 5 分钟前
- 5.0.	< 分享中的	□ 体验报告	陈江 5 分钟前
从您的产品需求构想开始,即可在这里发起并进行全过程 管理,分解需求、设计产品,直至需求开发完成落地。在	亩 回收站	CODING Enterprise Logo.png 🛂 🖈	陈泽芮 今天 20:32
这里。团队成员还可通过团队 Wiki、文件共享等工具进	◆ 快捷访问 ~	送代与事务规划通知和动态.txt	彭可 昨天 20:32
照提示填写账号信息后加入团队 。			
← 加入主账号团队			
項与账亏恬忌 加入主账号团队前,请填写以下信息。			
请输入邮箱			

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 <mark>主账号</mark>,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。

概览

CODING DevOps 简介 文 构想 〉 ① 计划 〉 子	,开发 〉	🗵 测试 > 🛕	交付		
	文件网盘	全部文件 Q - 搜索全部文件 + 割	證 ▼		
	文件网盘	全部文件 □ Q + 搜索全部文件 + 割 □ 文件名 o	i建 →		
*17#8	文件网盘 金部文件 金部文件 金部文件 金部文件	全部文件 Q = 搜索全部文件 + 劉 文件名 ○ → 研发三部产品设计	 建 → ▲ 上传 → ■ 创建者 → 最后更新 ◆ 陈江 5 分钟前 		
构想	文件网盘 全部文件 ◎ 最近文件 ★ 星标文件 《 分原中的	全部文件 Q = 搜索全部文件 + 第 文件名。 研发三部产品设计 ● 体验报告	 · 上传 → □ · · ·		
构想 从您的产品需求构想开始,即可在这里发起并进行全过程	文件网盘 金部文件 ● 最近文件 ★ 星标文件 < 夕原中的 音 回收站	全部文件 Q < 搜索全部文件	 建 ● 上传 ● ■ 创建者 ○ 最后更新 ● 陈江 5 分钟前 陈江 5 分钟前 陈泽芮… 今天 20:32 		

输入主账号邮箱与验证码完成开通。

← 开通 CODING 服务

输入主账号邮箱			
证码	获取验证码		



- 。 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。
 - ← 开通 CODING 服务

🕥 腾讯云

请输入密	码		Ø		
支持 6~32	位的字符组合。				
团队名称					
支持3~32位	面的中英文组合				
https://	团队域名		.coding.net		
支持3~32位	前字母数字组合,需以	字母开头。注册后不可修	改。		
开通即表	示我已阅读并同意《	CODING 服务协议》			

- 。 若主账号邮箱已关联多个 CODING 团队,选择需要的团队即可。
 - ← 开通 CODING 服务

输入的	的邮箱已有 CODING 账号,请选择要关联的团	I队。
请选择	CODING 团队	
腾云	腾云扣钉	
Wa	Washington	6
B	腾讯云	
Ba	balbala	
Zh	zhctt	
+	新建团队	



3. 开通成功后,子用户即可加入主账号团队。

← 开通 CODING 服务



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 <mark>访问管理控制台</mark> 中所选择的策略。

选择策略

授权提示

> 腾讯云

- 如果您希望授予子账号当前账号下全部资源的全部访问权限,请单选 AdministratorAccess 即可
- 如果您希望授予子账号当前账号下除去访问管理(CAM)、费用中心以外的全部资源访问权限,请单选 QCloudResourceFullAccess 即可
- 如果您希望授予子账号当前账号下全部资源的只读访问权限,请单选 ReadOnlyAccess 即可

新建自定义策略			支持搜索策略名称/描述	Q,
策略列表 (共654条,已选择1条)				
策略名	描述	引用次数	策略类型 🍸	
AdministratorAccess	该策略允许您管理账户内所	2	预设策略	
ReadOnlyAccess	该策略允许您只读访问账户	0	预设策略	
QCloudResourceFullAccess	该策略允许您管理账户内所	0	预设策略	
QCloudFinanceFullAccess	该策略允许您管理账户内财	0	预设策略	
QcloudAAFullAccess	活动防刷(AA)全读写访	0	预设策略	
QcloudABFullAccess	代理记账(AB)全读写访	0	预设策略	

支持按住 shift 键进行多选

自助管理 API 密钥 () 自助管理 MFA 设备 ()

确定	取消





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。

概览

	前往 CODING DevOps ^{团队域名}	
	立即使用立即选购	
快速创建子用户		
点击以下按钮快速创建拥有 CO	DING DevOps 权限的子用户。	
快速创建子用户		

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。

概览





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



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

项目内			团队内		公开				
別	拉取	推送	代理	拉取	推 送	代 理	拉 取	推 送	代 理
本项目 成员	1	1	1	1	1	1	1	1	1
团队内 成员	×	×	×	1	×	×	1	×	×
匿名成 员	×	×	×	×	×	×	1	×	×

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。

۵	项目概览		新建仓库	
\checkmark	项目协同			
	代码仓库		制品仓库*	🖹 < M 🔲 🥐
٢	代码扫描 beta	>		Generic Docker Maven npm PyPI
~~~	持续集成	>		
۵	持续部署	>		Helm Composer NuGet Copan
₽	制品库			
Д	測试管理	>	仓库地址*	https://StrayBirds-npm.pkg.coding.net/coding-demo/ npm
۵	文档管理	>	仓库描述	请输入仓库描述,最多可输入100个字符
			权限范围	制品库仓库对外的权限 ③ 查看制品库完整权限说明 ②
				○ 团队内
				○ 公开
٢	项目设置	~		

# 对已存在的仓库修改权限

# 1. 单击仓库右上方的设置仓库。

制品库 ⑦ キ新建	© 设置仓库
Conan 仓库 项目内	类型 Conan 权限 项目内 指引 包列表
nuget NuGet 仓库   项目内	设置凭证
Composer Composer 仓库   项目内	推荐使用访问令牌生成命令。 查看所有认证方式 じ
helm Helm 仓库   项目内	使用访问令牌生成配置



# 2. 在基本信息中即可对权限范围进行更新。

基础		
基本信息	基本信息	
代理设置		•
版本策略	仓库奕型*	Conan 👻
	仓库名称*	conan
	仓库描述	请输入仓库描述,最多可输入100个字符
	权限范围	制品库仓库对外的权限 ⑦ 查看制品库完整权限说明 [2] ● 项目内
		○ 团队内
		○ 公开
	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 <mark>制品库权限</mark>。 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。



### 1. 单击制品仓库指引页面的使用访问令牌生成配置。

制品仓库 全部制品 行	全库管理				创建制品仓库
<b>k8s</b> Docker 仓库   項目内	<b>k8s 司</b> 类型 Docker 1 权限 项目内			\$2 设置仓库 代	理设置版本覆盖策略
example npm 仓库 □ 项目内	- 镜像列表   ◆ 操作指引	<b>配置访问令牌</b> 输入密码后系统将自动生成访问令牌,并填入指引命令:	×		操作指引
www.bocker 仓库 项目内	100 Hat 19 50 . <b>配置凭据</b> 银像名 ≑ 推送	建築入室69 生成个人令牌作为完施 设置凭证	ı) ¢	版本数	操作
apk Generic 仓库 I 项目内	k8sdemo  镜像源加速 ⊘	请在命令行执行一下命令登陆仓库: docker login - cod	2 17:31:24	1	
<b>build</b> Docker 仓库   公开	1-1 个,共				每页显示行数 15 👻 🔰 👖
electron					
Generic 仓库 团队内					
Generic 仓库 项目内 go Generic 仓库 项目内					
Generic-go Generic 仓库 项目内	⑦ 帮助中心				
M test Maven 仓库 项目内					

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。

◆ 操作指引	<b>配置访问令牌</b> 输入密码后系统将自动生成访问令牌,并填入指引命令:	>
配置凭据	•••••	清除
推送 拉取	<b>设置凭证</b> 请在命令行执行一下命令登陆仓库:	
镜像源加速 🔗	docker login -u -p	

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。

🗲 🔼 演示项目	■ 制品仓库 全部制品 仓	仓库管理					创建制品仓库
← 制品管理	<b>k8s</b> Dacker 企業 」源目内	k8s 回 学型 Docker 祝祝 项目内				₽ 设置仓库	代理设置版本覆盖策略
制品仓库		ATT THE PART AND					
制品扫描	example	镜像列表					
		制品等级 全部 → +制品属性	▼ 提案制品名称 Q				操作指引
	weby Docker 仓库   项目内	镜像名 ≑	最新推送版本	制品等级	最近更新时间 ≑	版本数	操作
	apk Generic 仓库   项目内	k8sdemo 	master-	7a69 ¥7)ta	2022-05-12 17:31:24	1	
	● build Docker 仓库 □ 公开	1-1个, 共1个					每页显示行数 15 - 1
	daily-sentence Docker 仓库   项目内						
	electron Generic 仓库 团队内						
	gwt Generic 仓库   项目内						
	Steven 团队负责人						
	个人账户设置						
	服务订购						
	邀请成员						
	切换语言 简体中文						
	工单中心						
	更新日志						

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。

<ul> <li>◇ 飞鸟集</li> <li>◆ 个人账户设置</li> </ul>	访问令牌 ② 个人访问令牌可用于访问 CODING API,您的令牌用户名为	新建令牌 …
账户信息 全 个人账户	k8s artifacts 该令牌永久有效。	从未使用 编辑 …
邮箱设置 个人设置	ex acts 这令牌永久有效。	从未使用 编辑 …
</th <th>e) ^{'acts} 该令牌永久有效。</th> <th>从未使用 编辑 …</th>	e) ^{'acts} 该令牌永久有效。	从未使用 编辑 …
♂£ GPG 公钥 ♪ 访问令牌	<b>de</b> rtifacts 该令牌永久有效。	最后使用于 6 个月前 编辑 ····
<ul><li>一两步验证</li><li>通知设置</li></ul>	ssru tifacts 该令牌永久有效。	最后使用于 6 个月前 编辑 ····
<ul><li> </li><li> <p< th=""><th>ssr facts 该令牌永久有效。</th><th>最后使用于 6 个月前 编辑 ····</th></p<></li></ul>	ssr facts 该令牌永久有效。	最后使用于 6 个月前 编辑 ····
	<b>coc</b> <i>t:artifacts</i> 该令牌永久有效。	从未使用 编辑 …



3.	单击	令牌后面的 <b>编辑</b> ,	可以查看或修改该访问	]令牌的权	限信息。	
	Ø	飞鸟集    ▼	访问令牌 / 编辑访问令	牌		
	¢	个人账户设置	个人访问令牌类似 OAuth 访问	1令牌, 在通过	HTTPS 使用 Git 时可用它米代替密码, 或者将它按权给标准认证系统的 API。	
	账户(	信息	<ol> <li>如果您遗失或遗忘了此名</li> </ol>	令牌,您可以选择	释重新生成。但是请注意, 所有使用此令牌的应用或程序都必须更新为新的令牌。	重新生成
	-	个人账户				
		邮箱设置	令牌描述			
	个人i	没置	k8s-			
		仓库设置	到期时间			
	ď	SSH 公钥	该令牌永久有效。要设置新的	到期时间,您必须	须重新生成令牌。	
	đ	GPG 公钥				
		访问令牌	近择权限 个人访问令牌的权限定义。进一步	了解请阅读 OAuth \$	Scopes.	
	٢	两步验证	user	读	获取用户信息(名称、头像等)	
	5	通知设置				
	ତ	绑定设置	user:email	读	读取用户的 email	
	*	开放生态	notification	读、写	读取用户通知信息	
			project	读	授权项目信息、项目列表,仓库信息,公钥列表、成员	
			project:api_doc	发布	授权发布 API 文档	
			✓ project:artifacts	读、写	授权推送、拉取制品库	
			project:depot	读、写	完整的仓库控制权限	
			project:file	读、写	授权读取与操作文件	
	1	Steven -	project:issue	读、写	授权读取与操作项目协同模块	

# 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。

➡ 操作指引	<b>配置访问令牌</b> 输入密码后系统将自动生成访问令牌,并填入指引命令:
配置凭据	请输入密码 生成个人令牌作为凭据
拉取 镜像源加速 🔗	<b>设置凭证</b> 请在命令行执行一下命令登陆仓库:
	docker login -u



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 @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。

🗲 🔶 java-spring-app 🛛	gpc-60a3b71	₽制品地址▼	回操作指引	♥设置	(	» 历史版本	<b>4</b> 4	1
概览 属性 依赖分	析					搜索版本名称		٩
制品属性允许您给制品添加任意自定义的 查看 API 文档 ^亿	的内容,推荐您通过设置制品的属性,来跟踪整个制品的生产过程					版本 🗢	所在仓库	
属性	值					gpc-60836714 初始 439.56 MB	go	
challinger reliat	PUSH					<b>gpc-5005032</b> … 初始	go	
chalcingge are shall						439.56 MB		
phone care	java-spring-example					master-d0e17 初始 439.56 MB	go	
gl-contribution and	lanjiansheng@coding.net					master-04efef		
patranea	CODING CI 构建					初始 439.56 MB	go	
10,00,000	spring-docker					branch-df60f5… 初始	go	
clubbrantar.	31					439.56 MB		

# 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
// 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 '''
-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} \
${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。

← 项目设置	开发者选项	项目令牌(0)					
<ul><li>只 项目与成员</li><li>団 项目协同</li><li>・ 开发者选项</li></ul>	接口与事件 外部仓库管理 酒日会建	项目令牌用以部署项目, 只 <b>这里。</b>	针对当前代码仓库, 可证	殳置拥有只读或者读╕	6权限(默认为只读)。 不能	跟个人令牌通用,如需要设计	置个人令牌,请点击 新建项目令牌
	<del>攻回文林</del> WebHook 凭据管理	令牌名称	用户名	密码	创建时间 无数据	过期时间	操作

# 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。

坝目设直 / 坝目令牌 / 新知	<b>王坝日</b> 令碑				
新建项目令牌					
令牌名称	ž	过期时间			
artifact-properties-credent	ial	2021-02-17 ~			
项目管理权限					
<ul> <li>送代</li> <li>新建、音询、编辑、删除</li> </ul>	<ul> <li>史诗</li> <li>新建, 音询, 编辑, 剃除</li> </ul>	<ul> <li>需求</li> <li>新建、音询、编辑、删除</li> </ul>	任务 新建、查询、编辑、删除	Ŷ	
<ul> <li>缺陷 新建、查询、编辑、删除</li> </ul>	<ul> <li>文件</li> <li>新建、查询、编辑、删除</li> </ul>	<ul> <li>WIKI</li> <li>新建、查询、编辑、删除</li> </ul>	<ul> <li>项目公告</li> <li>新建、查询、编辑、删除</li> </ul>		
API 文档 发布 API 文档					
代码仓库权限					
仓库名称	访问权限		操作权限		
my-projects	✓ 读取 读取代码 仓库		<ul> <li>读写 推送至代码 仓库</li> </ul>	<ul> <li>合并请求</li> <li>新建、查询、编</li> <li>辑、删除</li> </ul>	<ul> <li>版本发布</li> <li>新建、查询、编</li> <li>辑、删除</li> </ul>
制品度切開					
1000000000000000000000000000000000000	□ 读写	2 制品属性	7		
拉取制品库	□ 读句 拉取、推送制品库	✓ ๗๓๓๒ ⊑ 新建、查询、编辑、删除			
新建取消					



如需要设置个人令牌,请点击
新建项目令牌
路码 编辑权限 禁用

# \$URL 的组成

https://{teamGK}.coding.net/api/projects/{projectName}/repositories/{repoName}/packages/{pkgN ame}/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/my version/properties

# 快速上手



curl -u 项目令牌:密码 -H "Content-Type: application/json" -d '{"properties":[{"name":"demo.name","va lue":"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","va lue":"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 \$U RL

# 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:

curl -u ptffmebv%; http://coding.com/go/govjects/coding-artifacts/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/repositoria/

# 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



### 示例:

## 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:

### 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:



# 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。

			+ :	添加等级
等级名称	描述	标签颜色	操作	
初始系统	用于标识初始状态的制品	•	编辑	
Alpha	用以标识进入 A 测阶段制品	•	编辑  删除	
Beta	用以描述完成 A 测阶段,进入 B 测阶段制品	•	编辑  删除	
Charlie	用以描述完成 B 测阶段,进入 C 测阶段制品	•	编辑  删除	
已发布 系统	用于标识已发布的制品	•	编辑	

# 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。

Q 搜索设置项	☑ 项目协同 团队级项目协同配置	<ul> <li>         ・</li></ul>	<ul> <li>か 持续集成</li> <li>回队级持续构建配置</li> </ul>	へ 持续部署 図队毀持续部署配置	^
✿ 全局设置	配置方案	工具规则	构建节点池	云账号	
🖬 功能设置	事项类型	方案模版	构建计划模版	堡垒机	
第 生态能力	事项属性 事项状态		构建插件	主机组	
	⑦ 公开资源 管理公开代码仓库	へ 代码仓库 団队級代码仓库配置	△ 制品管理 团队级制品管理功能	◇ 菜单管理 回N级菜单配置	^
	公开资源	仓库设置 团队都袭公明 仓库规范	制品盲纹	菜单管理	

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。

制品管理	功能设置 / 制品晋级 / 制品等级设置 制品晋级	t					
制品晋级	制品晋级功能支持团队内项目级和仓库级的自定义制品等级、您可根据制品质量情况设定对应的制品等级、实现以区分不同质量的制品版本。						
	<b>制品等级</b> 制品晋级规则	添加等级					
		等级名称			+ 添加等级		
	等级名称	请输入等级名称	标签颜色	操作			
	初始 系统	标签颜色	•	编辑			
	Alpha		•	编辑  删除			
	Beta	等级描述 请输入等级描述	•	编辑  删除			
	Charlie	æ	•	编辑 删除			
	已发布 系统	确定 取消	•	编辑			

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。

制品管理	功能设置 / 制品晋级 / 制品晋级规则设置					
	制品晋级					
制品晋级	制品晋级功能支持团队内项目级和仓库级的自定义制品等级,您可根据制品质量情况设定对应的制品等级,实现以区分不同质量的制品版本。					
	制品等级 制品晋级规则					
	规则					
	规则 A					
	暂无制品晋级规则					



# 1. 将制品等级圈选至"晋级规则"中。

制品管理	功能设置 / 制品	晋级 / 制品晋级规则设置					
	制品晋级						
制品晋级	制品晋级功能支持	团队内项目级和仓库级的自定义制品等级、您可根据制品质量情况设定对应的制品等级,实现以区分不同质量	量的制品版本。				
	制品等级 行	× 添加等级					
	规则	制品等级 *					
	规则A	Alpha	标签	操作			
		Beta	初始				
		Charlie	已发布				
		待发布					
		+ 添加					
		▲ 适用范围 日子配置生效的制品仓库 ⑧ …					
		+ 添加					
		700 XA					

### 2. 拖拽左侧按钮调整等级的流转顺序。

制品管理制品普级	功能设置 / 制品晋级 / 制品晋级规则设置 制品晋级功能支持团队内项目级和仓库级的自定义制品等级,您可根据制品质量情况设定对应的制品等级,实现以区分不同质量的制品版本。 制品晋级规则						
	规则 🔶	▲ <b>晋级规则</b> 用于配置制品晋级路径 <b>③ …</b>					
	规则A	等级名称	标签	操作			
		初始 系統	初始				
		🗄 Alpha	Alpha	删除			
		Chartie	Charlie	删除			
		Beta	Beta	删除			
		·····································	待发布	删除			
		已发布 系统	已发布				
		+ 添加					
		▲ 适用范围 用于配置生效的制品仓库 ③ …	Q 搜索				
		+ 添加					

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。


制品管理	制品晋级	Q / 制品首数规则	<b>汉直</b>					
制品晋级	制品音级功能支持团队内项目级和仓库级的自定义制品等级,您可根据制品质量情况设定对应的制品等级,实现以区分不同质量的制品版本。							
	制品等级制品	品晋级规则	× 添加适用范围					
	规则	O	■ 添加项目 + ) <b>①</b> 添加仓库 +					
			Q.mini O _{类型} 操作	1000	40.05			
	规则A		已选 0 项   清除已选	标盘	採作			
			Coding-dianshang	初始				
			· · · · · · · · · · · · · · · · · · ·	Alpha	删除			
		确定取消		Beta	删除			
			确定 取消	Charlle	删除			
				待发布	删除			
			已发布 系统	已发布				
			+ 添加					
		^	<b>适用范围</b> 用于配置生效的制品仓库 ◎ ···					
		[	+ 添加					
		70						
		5#						

## 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。

õ	CODING电频平	制品仓库 全部制品 仓库	管理					创建制品仓库
¢	制品管理	npmtesting npm 仓库 □ 项目内	docker3 3 类型 Docker   权限 項目内				✿ 设置仓库 代理设	置 版本覆盖策略
	制品包库制品扫描	Fepo Generic 仓库   项目内	镜像列表					
		docker3	制品等级 全部 👻 🕂 制品属性 🎽 授索書	品名称 Q				操作指引
		Docker 智序 项目网	镜像名 ≑ Java–spring–app	最新推送版本 hranch_9 vidrif8cf49	制品等级	最近更新时间 ≑ 2022-00-08 18-12-58	版本数	操作
		Docker 仓库   项目内     docker			初始 → Alpha			
		Docker 仓库   项目内	1-1个, 共1个				每页显示行数 15 👻 🧧	

历史版本

## 单去制品概览右上备的全屈按钮。展开查看制品的历史版太

腾讯云

半山巾	加恢见有上用的主用按钮,废开旦有前面的历史做名	<b>~</b> 0						
🔶 🖝 ja	← ◆ java-spring-app branch- 3cf49 初始 № 認識							69 🖉
概览	属性						搜索版本名称	
基本信息							版本 ≑	所在仓库
仓库 权限	docker3 项目内						branch-9e2eb 初始 439.56 MB	docker3
大小 hash	大小 439.56 MB hash sha256.xe2d 506c90a95152						master-c765f 初始 439.56 MB	docker3
<b>推送信息</b> 推送人	<b>推送信息</b> 推送人 项目助手 ❤ 持续集成 spring-docker#22						branch54271 初始 439.56 MB	docker3
推送时间 <b>镜像历史</b>	2022-09-08 18:12:58						branch-e3e45 初始 439.56 MB	docker3
命令		大小					branch-b7da6 初始 439.56 MB	docker3
ADD file	1086177965196842. b1d00e3129265eec9a in / ash"]	48.05 MB 0 B			~		branch-6ff17f0 初始 439.56 MB	docker3
/bin/sh /bin/sh	/bin/sh - c apt-get update & apt-get install - yno-install-recommends ca-certificates curl netbase wget & m -rf /var/li 7.45 MB v /bin/sh - c set -ex; lf 1 command -v ggg > /dev/null; then apt-get update; apt-get install -yno-install-recommends gnupg 9.53 MB v						master-6ff17f 初始 439.56 MB	docker3
/bin/sh	/bin/sh -c apt-get update && apt-get install -yno-install-recommends git mercurial openssh-client subversion procps && 49.43 MB v					branch-443f7		
/bin/sh	/bin/sh -c set -eux; apt-get update; apt-get install -yno-install-recommends bzip2 unzip xz-utils ca-certificates p11-kit f 5.04 MB v					初始 439.56 MB	docker3	
ENV LA	ENV LANG=C.UTF-8         0 B         ~           ENV JAVA_HOME=/usr/local/open/dk-8         0 B         ~							docker3

## 在制品等级中手动调整制品状态。

### 历史版本 🕴

仓库全部▼ 权限范围全部▼ 制品	等級 全部 > → 制品属性 >					搜索版本名称	
版本 ≑	所在仓库	hash 值	制品等级	推送人	推送时间 ⇔	下载量	操作
branch-9e2eb2717f41c5904463fa9 439.56 MB	docker3	sha256:e2dd36bfb45ef46cfbd	初始	项目助手	2022-09-08 18:12:58	3	
master-c765fb599373d42a40b747 439.56 MB	docker3	sha256:5cfd686ee5f4c9f1328	初始 → Alpha	项目助手	2022-09-08 18:12:55	3	
branch-54271d0d2f436dfa6c05c7 439.56 MB	docker3	sha256:44db700bf23694bd0a	初始	项目助手	2022-09-08 18:12:55	3	
branch-e3e45d188c595bb2af6b5b 439.56 MB	docker3	sha256:6d0a0faddd738d11a24	初始	项目助手	2022-07-05 16:09:05	3	
branch-b7da6527099345e0cec1ce 439.56 MB	docker3	sha256:7acd05da6a6c2fff73c	初始	项目助手	2022-07-05 16:09:05	3	
branch-6ff17f0e5c1ae156924b71c6 439.56 MB	docker3	sha256:f0aec98afbfed2127bcf	初始	项目助手	2022-07-05 16:09:05	3	
master-6ff17f0e5c1ae156924b71c6 439.56 MB	docker3	sha256:946e115ecd5dbd65a55	初始	项目助手	2022-07-05 16:08:51	3	
branch-443f78394a6c60842d471d 439.56 MB	docker3	sha256:d8b22a4e72ae55898b	初始	项目助手	2022-05-30 11:14:51	0	



# 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。

۲	④ 项目概览		制品库 ⑦ 🛛 🕂	coding-demo	◎ 设置仓库	版本覆盖策略	
	代码仓库		Maven 金属 项目内	类型 Docker 权限 项目内			
٢	代码扫描 beta	>	Maven CA ADD	指引 镜像列表			
~	持续集成	>	test Generic 仓库 □项目内	親衆 Q			
٩	持续部署	>		镜像名 \$ 最新推送版本	最近更新时间 ≎	版本数	操作
₽	制品库		coding-demo				
₫	测试管理	>	Docker 仓库 项目内	nodejs-express-app master-	14 天前	3	
۵	文档管理	>					



### 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。

//x/T-25544	
<b>基本信息</b> 控制该仓库下所有的制品的版本是否允许覆盖。包默认会使用仓库的版本策略,您也可以单独设置某个包的版本策略,包的优先级高于仓库的设置	E,
<b>代理设置</b>	
● 允许覆盖版本	
○ 禁止覆盖版本	
更新	

## ▲ 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。

制品库 ⑦ 十新建	<b>my-projects</b> 由代码模版创建	my-projects 版本号: latest	◎设置 1/1 ∧ ∨
● my-projects Docker 仓库 □项目内	类型 Docker   权限 项目 指引 包列表 ^{提索} … Q 包名 \$ my-projects 	概览     指引     属性     版本列表     2       推送信息              推送人                推送人                                                                                                                         <	推送时间 2 个月前 hash
		镜像历史 命令 ADD file: CMD ["bash"]	大小 43.24 MB 0 B



## 制品库

### 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。

制品库⑦ + 新建	my-projects 由代码模版创建 my-projects 版本号: latest		<b>懲设置</b>   1/1 へ ~
my-projects Docker 仓库 项目内	编辑 my–projects	版本列表 2	
	包名*		
	my-projects	推送时间 2 个月	前
	包描述		
	请输入包描述,最多可输入100个字符	hash	
	版本策略* 前往仓库版本策略设置		
	控制该包下所有的制品版本是否元件覆盖。包斯认会使用它库的版本 策略,包的优先级高于仓库的设置 查看覆盖策略帮助文档 [2]	大小	
	继承仓库 my-projects 的版本策略 ▼	43.24 MB	
	确认编辑 取消	0 B	
		get install -yn 10.29 MB	
	/bin/sh -c set -ex; if ! comman	nd -v gpg > /dev/null; 4.14 MB	
	/bin/sh -c apt-get update && a	apt-get install -yn 833.19 KB	
	ENV LANG=C.UTF-8	0 B	

## 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。

制品库 ⑦ + 新建	my-projects 由代码模版创建	my-projects 版本号: latest	\$3设置 │ 1/1 ∧ ∨
my-projects Docker 仓库 项目内	类型 Docker   权限 项目	概览 指引 属性 版本列表 2	
	指引 包列表	_ 提索 Q	
	援索 Q	标记为发布可以保障制品版本的唯一 版本 章 性,防止重复写入同一版本	推送人 推送时间 ≑
	包名 ≎	版本号: <mark>latest</mark> (当前版本 <mark>标记为发布</mark> hash值: sha256:	项目助手 2 个月前
	my-projects		
		版本号: hash值: sha256:	项目助手 2个月前

## 单击**标记为发布**按钮后,出现发布版本弹窗,单击**确认**即可。

制品库 ② + 新建	my-projects 由代码模版创建 版本号: latest	
my-projects	类型 Docker 权限 项E 概览 指引 属性 版本列表 2	
Docker仓库,项目内	指引包列表 漫齋… 《	
	标记为发布	推送人
	您正在尝试将 my–projects 仓库里的 my–projects:lat est 标记为已发布。标记为发布后,您将永远无法覆盖	项目助手
	该版本。 宣看版本策略帮助又档记 您也可以通过设置 my-projects仓库版本覆盖策略 来控制整个 仓库的默认策略。	项目助手
	确认取消	



## 发布成功后,该版本会显示一个**已发布**标签。

my-p 版本号	orojects : latest 已发布				② 设置 │ 1/1 ∧ ∨
概》	览 指引	属性	版本列表 2		
搜索					
版本;	•			推送人	推送时间 ≑
版本 hash	:号: <mark>latest</mark> (当 h值: sha256:	前版本]已	发布	项目助手	2 个月前
版本 hash	·号: n值: sha256:			项目助手	2 个月前

## 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



# 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**。

wython Docker 仓库 □ 项目内	<b>python 词</b> 类型 Docker 权限 项目内	✿ 设置仓库 代理设置	版本覆盖策略			
● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ●	镜像列表					
_ cd-demo	制品等级 全部 ▼ +制品属性 ▼ 搜索制品名称		操作指引			
Docker 仓库 公开	镜像名 ≑	最新推送版本	制品等级	最近更新时间 ≑	版本数	操作
docker-go Docker 仓库   项目内	python-flask-app 	v1.0	初始	2022-05-17 11:54:26	2	
	1-1 个, 共 1 个				每页显示行	数 15 👻 🚺

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

创建制品仓库



### 理列表。

← 设置 python	
基本信息 代理设置	<b>清理策略</b> 制品仓库清理策略支持自动/手动批量删除满足清理条件的制品,避免长时间未被使用的老旧制品占据过大储存空间。 查看帮助文档 ^[2] 潇珊心罢
版本策略 清理策略	<b>消理设直</b> 单个制品包的制品版本保留 20 个
	<ul> <li>超出保留数量的版本若在最近 30 天没有被下载,将会被自动 / 手动清理。优先清理上一次被下载时间较早的版本。</li> <li>✓ 保留被标记为"已发布"的制品</li> <li> <b>开启自动清理</b> ⑦     </li> </ul>
	保存  手动清理

## 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。

索设置项	☑ 团队信息 团队基本信息设置	^	<b>组织和成员</b> 团队组织架构、成员管理和批量操作	∧ ● 服务订购 服务订购与订购信息管理	◇ 安全性 图队安全相关功能设置
后设置	基本信息		成员管理	概览	访问审计
能设置	实名认证		权限配置	订购	会话管理
	注销团队		批量操作	资源用量	登录设置
				账单管理	水印设置 Pro
				优惠券管理	日志
	■ 第三方应用 第三方服务绑定	^			
	第三方应用				

## 在**制品仓库日志**页将显示近期的制品仓库日志。

		*	日志管理									
Q			操作日志	登录日志	代码仓	库日志 制	品仓库日志					
¢			逸揖成员     >     逸揖朝品库     ▼     开始日雨     ◆     至     結束日雨     ●       服素条件:操作者:所有: 制品合库:所有: 計加日期: 无: 結束日期: 无:       ●     ●     ●							导出 Excel		
	会话管理		成员	制品仓库	仓库类型	操作类型	制品名称	制品版本	代理信息	IP	平台	操作时间
	登录设置 水印设置	Pro	项目助手	demo/k8s	Docker	拉取制品版本	k8sdemo	master- 27eabe61d2a 757a	69	172.17.16.156	docker/2 172 ()	2022-08-11 14:48:36
			项目助手	flask demo/python	Docker	拉取制品版本	python– flask– app	v1.0		10.0.32.229:44852	codinç ant ()	2022-05-17 11:54:27
			项目助手	flask- demo/python	Docker	拉取制品版本	python– flask– app	v1.0		10.0.32.229:51622	Gc 1 Q	2022-05-17 11:54:27
			项目助手	flask- demo/python	Docker	拉取制品版本	python– flask– app	v1.0		10.0.32.229:39032	Go-1 .1 ()	2022-05-17 11:54:27
			项目助手	flask- demo/python	Docker	删除制品版本	python flask app	v1.0		172.17.49.196	docke /872 ()	2022-05-17 11:54:26



# 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、PyPl 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



			-, -, -, -, -, -, -, -, -, -, -, -, -, -	
۵	项目概览		新建仓库	
V	项目协同			
<i>«</i> />	代码仓库		制品仓库*	
٢	代码扫描 beta	>		Generic Docker Maven npm PyPI
~	持续集成	>		
4	持续部署	>		Helm Composer NuGet Conan
₽	制品库			
A	测试管理	>	仓库地址*	https://StrayBirds-maven.pkg.coding.net/repository/coding-demo/ maven
۵	文档管理	>	仓库描述	请输入仓库描述,最多可输入100个字符
			权限范围	制品库仓库对外的权限 ② 查看制品库完整权限说明已
				• 项目内
				○ 团队内
				○ 公开
			启用代理	✓ 允许获取代理源的包 ③ 什么是制品代理 ○
٢	项目设置	«	确认	

### 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。

制品库 🔊 🛛 🕂	maven
Maven 仓库 项目内	类型 Maven 权限 项目内 指引 包列表
test Generic 仓库 项目内	M Apache Maven
eoding-demo Docker 仓库   项目内	设直凭证 自动生成配置
	推荐使用访问令牌生成认证配置。 查看所有认证方式 I2 使用访问令牌生成配置
	编辑您的 settings.xml。 如何找到 settings.xml 文件位置记
	请将下方 [PASSWORD] 替换成您的登录密码。请妥善保管好您的配置,不要随意分享给他人。
	一般情况 maven 的通用 settings.xml 在 .m2 文件夹下, 项目内 settings.xml 也可以进行设置, 优先级更高 -<br <settings></settings>
	omitted xml <servers></servers>



### 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。

<b>制品仓库</b> 全部制品	仓库管理	创建制品仓库
k8s     Donker 合度 道日内	k8s 司 ^{進型} Docker 邦限 道日内	✿设置仓库 代理设置 版本覆盖策略
example npm 仓库   项目内	镜像列表	代理设置 担取私有仓库不存在的镜像时,将尝试从配置的代理地址去担取。优先级:从上到下。 宣看帮 助文档2
ruby	制品等级 全部 + 制品属性 -	来源 地址 操作
Docker 仓库 山 坝 目内     apk	(現象名: 最新推送版本 制品等级 k8sdemo	暂无数据
Generic 仓库 项目内		+ 添加来源 ✓ 毎存代理制品至本合度 ⑦
<b>build</b> Docker 仓库   公开	1-1个, 共1个	□ 禁止没有缓存(推送)制品权限的成员代理远程仓库制品至本地 ⑦
→ daily-sentence Docker 仓库   项目内		保存
electron Generic 仓库   团队内		

### 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。

### 创建来源

基础信息

地址*:	Maven Central (https://repo.maven.apache	•
名称*:	Maven Central	

### 鉴权信息

您可以通过设置来源的鉴权信息来拉取私有包

账号:	
密码:	••••••
添	取消

## 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
	npmjs	访问地址
npm	cnpm	访问地址
	TencentCloud npm	访问地址
DyDI	PyPI	访问地址
ГУГІ	TencentCloud PyPI	访问地址
	Maven Central	访问地址
Maven	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://xxxxxxx-maven.pkg.coding.net/repository/coding-demo/my-m aven/org/springframework/spring-jcl/5.0.7.RELEASE/spring-jcl-5.0.7.RELEASE.pom [INFO] Downloaded from : https://xxxxxxx-maven.pkg.coding.net/repository/coding-demo/my-ma



/org/springframework/spring-jcl/5.0.7.RELEASE/spring-jcl-5.0.7.RELEASE.pom (1.9 kB at 735 B,

## 2. 同时,在 CODING 制品库上,您也可以看到该制品的来源。

my-maven	8	org.springframework:spring-jcl 版本号: 5.0.7.RELEASE					11 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ↔ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ▲ ● 10 ■ ■ ■ ● 10 ■ ■ ■ ● 10 ■ ■ ● 10 ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■
失望 Maven   仪限 项目内		概览	指引	属性	版本列表 1		
指引    包列表							
搜索 Q	~	· 推送信息 推送人	★ 11-1	li.		推送时间	7 分钟前
包名≑		其他					
org.junit.jupiter:junit-jupiter-api 		作者				协议	
commons-io:commons-io		大小	21.19 KB		_	hash	699016ddf454c2c167d9f84ae577 7eccadf54728
		来源	代理 Mave	en Central			
commons-fileupload:commons-fileupload	ľ				-		
javax.servlet:javax.servlet-api 							
org.springframework:spring-web							
org.springframework:spring-webmvc 					-	-	
org.springframework:spring-aop 						_	

### 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。







# 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。

扫描规则 ⑦

漏洞等级 * 🛛 🖌 🖌 高危 🖌 中危 🖌 低危



### • 漏洞白名单(仅安全扫描)

CVE 漏洞白名单 ⑦

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。

漏洞 CVE ID	所属组件 ⑦	
例如 CVE-2020-9794	例如 com.alibaba:fastjson	(
+ 添加 CVE 漏洞白名单		

### 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。

#### ← npm:3.0.0 详情

npm:3.0.0 用户 coding 通过制品更新触发了 扫描	漏洞概览 开源许可证风险 依赖分析
	個件名称 超件版本 合厚地址
<ul> <li>扫描状态</li> <li>* 未通过质量红线</li> <li>扫描版本</li> </ul>	<ul> <li>extraneous-dev-dep</li> <li>0.0.0</li> </ul>
	关联文件
3.0.0 对比版本	/a8597lb5e924cfa7eb97cab20fb91e13b699daba.tgz.extracted/a8597lb5e924cfa7eb97cab20fb91e13b699daba.tar.extracted/package/node_modules/read-installed/test/ftxtures/extraneous-dev-dep/package.json
21604+ 8.16.0 紅枝标淵 总数 ♥ 优先关注 ≤ 0 且 新増 ♥ 优先关注 ≤ 0 且 总数 ● 危急 ≤ 0	> aws-sign2 0.5.0 -
	<ul> <li>npm-test-shrinkwrap</li> <li>0.0.0</li> </ul>
	关现文件
且 新増 🔂 危急 ≤ 0 持续时间	/a8597lb5e924cfa7eb97cab20fb91e13b699daba.tgz.extracted/a8597lb5e924cfa7eb97cab20fb91e13b699daba.tar.extracted/package/test/packages/npm-test-shrinkwrap/package.json
24 秒 开始时间	npm-test-bundled-git 1.2.5 -
5 小时前	> asn1 0.1.11 -
npm	extraneous-detected 0.0.0 -
	> tar 2.1.1 -
	> npm 3.0.0 -
	> npm-test-blerg3 0.0.0 -
	npm-test-optional-deps 1.2.5
	< 1 2 3 4 5 6 7 > 10条/页 >

## 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

▲ 导出报告 重新扫描



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<mark>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</mark> 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。

MYNREPOSITORY		Search for groups, artifacts, categories	Search			
Indexed Artifacts (20.4M)	Home » com.alibaba	<ul> <li>» fastjson » 1.2.76</li> <li>n » 1.2.76</li> <li>a JSON processor (JSON parser + JSON generator) written in Java</li> </ul>				
GCA 2	License	Apache 2.0				
2006 2008 2010 2012 2014 2016 2018 Year	Categories	JSON Libraries				
Popular Categories	Organization	Alibaba Group				
Aspect Oriented	HomePage	https://github.com/alibaba/fastjson				
Actor Frameworks	Date	(Apr 05, 2021)				
Application Metrics	Files	jar (643 KB) View All				
Build Tools	Repositories	Central				
Bytecode Libraries	Used By	4,001 artifacts				
Cache Implementations						
Cloud Computing	Maven Gradle	SBT Ivy Grape Leiningen Buildr				
Code Analyzers	<pre><!-- https://mvni</pre--></pre>	repository.com/artifact/com.alibaba/fastjson>				
Collections	<pre><dependency>    <groupid>com</groupid></dependency></pre>	alibaba				
Configuration Libraries	<artifactid>fastjson</artifactid> <version>1.2.76</version>					
Core Utilities			,			
Date and Time Utilities		uite liel en de de vetien				
Dependency Injection						
Embedded SOL Databases	Copied to clipboard!					

## 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 $\sim$ 方案描述 请输入仓库描述,最多可输入100个字符 扫描规则 🕐 ✓ 危急 ✓ 危急 ✓ 高危 ✓ 中危 ✓ 低危 漏洞等级 * CVE 漏洞白名单 请填入 CVE 漏洞编号, 如 CVE-2020-9794 $\otimes$ + 添加 CVE 漏洞白名单 质量红线 ⑦ 红线标准 🔒 危急漏洞 $\sim$ 总数 <= 🗸 0 $\otimes$ 总数 <= 🗸 且 🛆 高危漏洞 0 $\otimes$ $\sim$ 且 🔒 危急漏洞 新增 <= 🗸 0 $\otimes$ $\sim$ 且 新增 <= 🗸 🛆 高危漏洞 0 $\otimes$ $\sim$

### 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 <mark>功能介绍</mark>。

## 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。

← 创建扫描	方案	
基础信息		
方案名称*	Docker 制品库扫描	
方案描述		
扫描规则 ⑦		
漏洞等级 *	✔ 危急	
CVE 漏洞白名单	CVE-2020-9794	8
	+ 添加 CVE 漏洞白名单	
质量红线 ⑦		
红线标准	+ 添加质量红线	
新建取消		
▲ 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。

扫描方案 🕂									帮助文档 🗹	
test-go					<b>制品仓库测试验证</b>					
历史累计数据					历史累计数据					
累计制品 1 ···································	危急漏洞 <b>0</b>	高级漏洞 <b>0</b>	中级漏洞 <b>0</b>	低级漏洞 0 酒	累计制品 8 constant f f f f f f f f f f f f f f f f f f f	危急漏洞 9	高级漏洞 <b>114</b>	中级漏洞 64 查	低级漏洞 <b>735</b> 酒详情 →	

### 单击规则配置也可以进入设置页面。

<ul> <li>◆ 制品仓库测试验证</li> <li>方案 III</li> </ul>	0146d19ac760456			◆ 规则配置 方案应用
历史累计数据				
累计制品	危急漏洞		中级漏洞	低级漏洞
8	9	114	64	735
扫描记录				
全部 制品更新触发 手动触发 自定	义构建计划			



### 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。

← 设置扫描方案 规则配置 方案应用				
基础信息				
方案名称*	fastjson			
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			
方案 ID	每个扫描计划都有全平台唯一的 ID,可用于自定义的持续集成编排时通过方案 ID 指定扫描配置。前往方案应用 方案 ID 707ea00e7b9d49( ]			
扫描规则 ⑦				
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 ⑦				
红线标准	<ul> <li>● 危急漏洞</li> <li>✓</li> <li>&gt; 总数 &lt;= ✓</li> <li>○</li> </ul>			

## 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。

扫描方案 🕂									帮助文档记
test-go	Sep420504-70-	67419		۰۰۰ م	制品仓库测试验证	560965do52cccc5	2506b	[	<b>≎ •••</b>
历史累计数据	0004095040790	07410			历史累计数据	000000000000000000000000000000000000000	0000		<u>万</u> 乘应用 删除
累计制品	危急漏洞	高级漏洞	中级漏洞	低级漏洞	累计制品	危急漏洞	高级漏洞	中级漏洞	低级漏洞
	0	0	0	0	8	9	114	64	735
			査	這看详情 →					查看详情 >

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。

← 设置	置扫描方案    规则配置   方案应用	添加筛选规则
自动扫描	开启后满足条件的制品有新的推送时,会自动使用当前方案进	筛选规则名称
	自动扫描将通过持续集成构建计划 制品扫描方案—fastjson—自	筛选规则–1
	已开启	制品仓库
扫描筛选	○ 全部	只有满足筛选条件的制品更新时,才会自动触发扫描。
	● 按条件筛选	○ 全部制品仓库
	+ 添加筛选条件	● 指定制品仓库
		M scan-test 🍦 pypi-go 💙 👉 write-go M test-go
		Incde-demo I py-go I video
		制品筛选
		<ul> <li>全部制品</li> </ul>
		○ 满足规则条件的制品
		添加取消

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



°⊞o		
← 设置	置扫描方案 规则配置 方案应用 ————————————————————————————————————	◎ 添加筛选规则
自动扫描	开启后满足条件的制品有新的推送时,会自动使用当前方案 自动扫描将通过持续集成构建计划制品扫描方案_fastison_	策进规则名称     简选规则−1     简选规则−1
扫描筛选	<ul><li>已开启</li><li>全部</li></ul>	<b>制品仓库</b> 只有满足筛选条件的制品更新时,才会自动触发扫描。 ○ 全部制品仓库
	<ul> <li>按条件筛选</li> <li>+ 添加筛选条件</li> </ul>	● 指定制品仓库 M scan-test
		★ node-demo ★ py-go ★ video
		○ 全部制品
		<ul> <li>满足规则条件的制品</li> <li>制品名称 ~ 等于 ~ test</li> </ul>
		或 制品版本 V 正则表达式 V ^release- 🛞
		+ 添加制品筛选
		添加取消

## 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。

← fastjson 方案 ID 707ea	a00e7b9d49(	Ì			✿ 设置		
最近七天数据						机重扫描	
累计制品	危急源			中级漏洞	低级漏洞		
	0	0		0	0		
扫描记录							
全部 制品更新触发 手动触发 自定	全部 制品更新触发 手动触发 自定义构建计划						
制品名称	制品版本	来源	制品仓库	持续时长	开始时间	操作	
om.alibaba:fastjson	1.1.15	用户 蓝健声 通过手动触发了扫描	scan-te:	st 4秒	1 个月前	查看详情	
1-1 个, 共 1 个 毎页显示行数 15 - 1							

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。



← fastjson 方案 ID	707ea00e7b9d49( 🗻				✿ 设置	立即扫描   :
最近七天数据		×				
^{累计制品}	批量触发	4	中级漏洞 <b>0</b>		低级漏洞 <b>0</b>	
	制品仓库					
扫描记录 全部 制品更新触发 手动触发	<ul> <li>              扫描所有制品仓库      </li> <li>             扫描指定制品仓库         </li> </ul>					
制品名称	扫描筛选	库		持续时长	开始时间	操作
🌝 com.alibaba:fastjson	<ul> <li>扫描所选制品仓库内所有制品包的最新版本</li> <li>扫描满足规则的制品</li> </ul>	test		4 秒	1个月前	查看详情
1-1个, 共1个	立即触发 取消				每页显示	行数 15 👻 🚺

## 在持续集成流水线中也支持手动添加制品扫描单元。

🔶 openapi-prod 🗵	基础信息 流程配置 触	的发规则 变量与缓存	通知提醒	尼前往最新构建 操作 ∨ ▶ 立即构建		
分支 master ▼ 中的 api–Jenki	nsfile ⑦ 图形化编辑器 文本编辑	毒器		↓ 夺环境变量		
				× CODING 制品扫描	0	
检出 (+)	→ 3–1 发布 API 文档	+	增加阶段	<b>插件配置</b> 高级配置		
从代码仓库检出	器 读取代码生成 API 文档			制品扫描在不需要访问源代码的情况下,通过扫描二进制组件及其元数据 找到组件中存在的漏洞。 查看完整帮助文档 🖸	居,	
+ 博加并行阶段	▲ CODING 制品扫描 +			制品仓库 *		
¥ 19107TJ91FX	快速筛选 <b>Q</b>			scan-test		
	↓ 命令			制品 *		
	♦ 代码管理			请选择制品	-	
	▶ 文件操作			制品版本 *		
	▶ 收集报告			请选择制品版本	-	
	品 流程控制 ▶			扫描方案 *		
	▽ 安全			请洗择扫描方室	-	
	△ 质量管理 • ▶	CODING 制品扫描		ארר ורו + ובא או		
	品 持续部署 ▶	代码扫描NEW		质量红线不通过时构建失败		
	器 其他 ▶					
	器编译 ▶					

## 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。

制品库 ⑦ 🛛 🕂 🕂	<b>npm-test</b> npm制品			✿ 设置仓库 代理	设置 版本覆盖策略
<b>pypi-test</b> PyPI 仓库   公开	类型 npm   权限 团队内				
npm-test npm 仓库 团队内	指引 包列表 _{搜索} Q				
<b>质量红线</b> 未通过质量红线		最新推送版本	最近更新时间 ≑	版本数	操作
1 个未通过,0 个通过,0 个未设置	,0 个扫描中	1.0.0 标记为发布	21 小时前	1	
😢 制品仓库测试	查看详情				
generic-002 Generic 仓库 公开					
Maven 仓库   项目内					
<b>java-demo</b> Docker 仓库   项目内					

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。

<ul> <li>项目组测试环境制品扫描</li> </ul>	方案 ID	3e00b3c2b2a14f8	a			<u>↑</u>	导出报告   💠 访	立即扫描   :
漏洞统计								
漏洞总数 16	11	危急漏洞		高危漏洞 15	中危漏洞 1		低危漏洞	
扫描记录								
仓库 全部 ▼ 扫描状态 全部 ▼ 触发方式	全部▽							搜索制品名称 Q
制品名称		制品版本	来源		制品仓库	持续时长	开始时间	操作
FacebookPlus_145.0_v1.6r-35.ipa		latest	用户 oon	1 通过手动触发了扫描	generic	4 分钟 23 秒	4 天前	查看详情
v apkextractorprov14.5.0.apk		latest	用户 Diar	ne 通过手动触发了扫描	generic	2 分钟 48 秒	4 天前	查看详情
com.thoughtworks.xstream:xstrean	n	1.4.17	用户 Diar	ne 通过手动触发了扫描	maven-proxy	3 分钟 12 秒	4 天前	查看详情
1-3个. 共3个							每页5	显示行数 15 ▼ 1

### 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。

```
← npm:3.0.0 详情
```

npm:3.0.0	漏洞概览 开源许可证风险 依赖分析	
■■■ 用户 coding 通过制品更新触发了 扫描	License 名称 License 风险	关联组件数量
扫描状态	> bsd-simplified ● 低风险	3
扫描版本	> ISC< ● 低风险	56
3.0.0 对比版本	✓ gpl-3.0 ③ 高风险	2
8.16.0 红线标准	License 详情	
总数 <b>寸</b> 优先关注 ≤ 0 目 新増 <b>寸</b> 优先关注 ≤ 0	License 名称 gpl-3.0	
且 总数 6 危急 ≤ 0	License 风险 🚯 高风险	
且 新唱 ♥ 泡志 ♥ 0	License 来源 https://spdx.org/licenses/GPL-3.0-only.html	
24 秒	License 约束 无	
5 小时前	License Copyright	
制品仓库 npm	GNU GENERAL PUBLIC LICENSE Version 3, 29 June 2007	
	Copyright (C) 2007 Free Software Foundation, Inc. <https: fsf.org=""></https:> Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed. Preamble	
	宣看更多	
	关联组件	
	组件名称	
	tar 2.1.1	

## 依赖分析

⑦ 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。

← npm:3.0.0 详情			全 导出报告	重新扫描
npm:3.0.0	運調概范 开源许可证风险 <b>依赖分析</b>			
用戶 coding 通过制品更新触发了 扫描	组件名称	组件版本	仓库地址	
扫描状态      オーズ オーズ (本語)     オーズ     オーズ	✓ extraneous-dev-dep	0.0.0	-	
	关联文件			
3.0.0 311/16 ±	/a85971b5e924cfa7eb97cab20fb91e13b699daba.tgz.extracted/a85971b5e924cfa7eb97cab20fb91e13	b699daba.tar.extracted/package/node_modules/read-installed/test/fixtures/extraneous-dev-dep/package.jso	'n	
对比版本 8.16.0	> aws-sign2	0.5.0	-	
1336/m //////////////////////////////////	<ul> <li>✓ npm-test-shrinkwrap</li> </ul>	0.0.0	-	
且 总数 ① 危急 ≤ 0	关联文件			
且 新增 [●] 危急 ≤ 0 持续时间	/a85971b5e924cfa7eb97cab20fb91e13b699daba.tgz.extracted/a85971b5e924cfa7eb97cab20fb91e13	b699daba.tar.extracted/package/test/packages/npm-test-shrinkwrap/package.json		
24 秒 开始时间	> npm-test-bundled-git	1.2.5	-	
5 小时前 制品仓库	> asn1	0.1.11	-	
npm	> extraneous-detected	0.0.0	-	
	> tar	2.1.1	-	
	> npm	3.0.0	-	
	> npm-test-blerg3	0.0.0	-	
	> npm-test-optional-deps	1.2.5	-	

#### 1 2 3 4 5 6 7 > 10条/页 >

## 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。

🍪 🗖 devops 👻	🕤 npm:3.0.0 详情							土 导出报告	重新扫描
<ul> <li>Q 理案</li> <li>◆ 新島管理</li> <li>初島仓库</li> <li>◆ 新島行道</li> </ul>	Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0 Pgr:30.0	面积极型         开源许可证风险         依赖分析 <b>(加久型金坑1日)</b> <	<ul> <li>● 低危4 • 4</li> <li>● 低危4 • 4</li> <li>● ① (双原示未被1)</li> <li>● ② ③ 危急</li> <li>○ ○ 危急</li> <li>○ ○ た急</li> <li>○ ○ た急</li> <li>○ ○ た急</li> </ul>	28美選列 ● 所順の面 extend 至3.0.2版本中存住注入職用。该職用用于用户 解析或解释方式错误。 nd/pull/48	<ol> <li>引入回本</li> <li>2.0.1</li> <li>論入传道命令、数量结构或记录</li> </ol>	将至原本 2.0.2,3.0.2 的操作过程中, 网络系统或/**	王王 CVI 道利毎年。 Q   有式利用 POC   元   (VSS 3.0   4) (VSS 3.0   4) ( 文広の単田   ( 文広の単田   ( 文広の単田)  ( 文広の単田)  ( 文広の単石)  ( 文広の単石)  ( 文)  ( 一)( 二)( 二)( 二)( 二)( 二)( 二)( 二)( 二)( 二)( 二	歴史が現める。 建立が現める。 3 5 5 5 5 5 5 5 5 5 5 5 5 5	0. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2.
		> CVE-2021-44906 新增	🔽 😯 危急	minimist	0.0.8	1.2.6	无		
		> CVE-2018-1000620 新増	70 危急	cryptiles	2.0.4	>4.1.1	无		
		➤ CVE-2018-3728 新增	😯 🔷 高危	hoek	2.14.0	4.2.0,5.0.3	无		
		> CVE-2017-6458 新增	🔽 🔿 高危	sntp	1.0.9	4.2.8,4.3.94	无		
		1-5 个, 共 5 个						页显示行数 15 👻	1

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。



#### com.alibaba:fastis

com.	.alibaba:fastjson:1.1.15 详情	<b>書</b> 月			重新扫描
M co 5 用户	m.alibaba:fastjson:1.1.1 ⁵ 蓝健声 通过手动触发了扫描	_{漏洞概览} 漏洞数量统计图			
量红线 务状态	未通过质量红线 扫描结束	<b>2</b> 漏洞数量 ■ 危急 0 ■ 高危 2 ■	中危 0 💿 低危 0		
品仓库	scan-test				
记录	592643#1	漏洞列表统计			
2	com.alibaba:fastjson:1.1. 15	全部 危急 高危 中危 低危	搜索漏洞编号		
卖时间	4 秒	CVF 漏洞编号	漏洞等级	所屋依赖	引入版本
台时间	2 分钟前	<ul> <li>✓ CVE-2017-18349</li> </ul>	△ 高危	com.alibaba:fastjson	1.1.15
		<b>Pippo FastjsonEngine Fastjs</b> Pippo是一款基于Java的Web框架。 器/生成器。Pippo1.11.0版本中的Fa 过发送特制的JSON请求利用该漏源	<b>on 输入验证漏洞</b> FastjsonEngine是其 astjsonEngine所使用的 执行任意代码。	中的一个JSON处理引擎。Fastjson是其中 的Fastjson1.2.25之前版本的parseObject存	的一个基于Java的JSON解析 荞在安全漏洞。远程攻击者可通

#### 修复建议

目前厂商已发布升级补丁以修复漏洞,补丁获取链接: https://github.com/alibaba/fastjson/wiki/security_update_20170315

#### 相关信息

https://fortiguard.com/encyclopedia/ips/44059 https://github.com/alibaba/fastjson/wiki/security_update_20170315 https://github.com/pippo-java/pippo/issues/466

### 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。

漏洞 制品						
筛选器 选项 → 筛选器 选项 → 筛选器	₩ <b>选项 &gt;  时间段</b> 开始	时间 - 童结束时间,	"提索	۹		按优先级 ▽ Ξ
CVE-2018-120150	CVE漏洞编号	所属依赖	漏洞等级	引入版本	建议升级版本	操作
所属依赖 openssh	CVE-2018-12015	alpine-keys	🗿 中危	1.0-1.17.1, 1.18.0-1.18.2	1.0-1.17.2、1.18.0-1.18.3	忽略 修复
<ul> <li>CVE-2021-22898</li> <li>所属依赖 alpine-keys</li> <li>CVE-2018-10005</li> </ul>	Busybox 安全漏漏 BusyBox是乌克兰软( 含一个缺少SSL证书 通过HTTPS下载任何	牛开发者DenisVlasen 全证漏洞,该漏洞可能 文件,似乎可以利用」	ko所负责维护的一 B导致任意代码执行 比攻击。	套包含了多个linux命令和工具的应 。通过使用" busybox wget http	用程序。Busybox在" busybox w ss://compromised-domain.com/	展开全部 ¥ get"小程序中包 important-file"
所属依赖: liberchive	所在制品			实际引入版本	修复版本	操作
所属依赖 systemd	nginx:1.0.0 已修复			3b8d43a	bef4515	忽略 修复
O CVE-2020-1785	nginx:1.0.0			00c843f	c2644da	忽略 修复
所属依赖 perl	nginx:1.0.0			af4ba1c	b0d51b9	忽略 修复
○ CVE-2020-6069 所属依赖 curl	nginx:1.0.0			3161ad5	4fe7ad5	忽略 修复
	nginx:1.0.0			388943e	ca85b25	忽略 修复
CVE-2020-6069 所属依赖 curl	nginx:1.0.0			e9cc319	91c98f9	忽略 修复
	nginx:1.0.0			c60bc92	56473fa	忽略 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。

← simple-example 区 基础信息 流程配置	<b>1</b> 触发规则 变量与	缓存 通知提醒	В 前往最新构建 操作 ∨ ● 立即构建
静态配置的 Jenkinsfile ⑦ 图形化编辑器 文本编辑器			♦♦ 环境变量 医异修改 保存
			× CODING 制品属性设置
→ <mark>3-1</mark> 测试	→ 4-1 部署	<b>5-1</b> 阶段 5-1	插件配置 高级配置
↓ 打印消息	。 ↓ 打印消息	乙 代码扫描	插件版本 *
↓ 打印消息	↓<>          ↓         打印消息	△ 代码扫描	最新版本 ~
同 写文件	·····································	•	制品所在项目 *
	+	+ 增加并行阶段	\${PROJECT_NAME}
x 图 收集 JUnit 测试报告	快速筛选 Q		制品所在仓库 *
	of 命令 •		docker
+ 增加并行阶段			制品名称 *
	文件操作		hollo world
	部制品库 ·	全部 官方插件 团队插件	neno-world
	• 收集报告	上传到 Generic 制品库	制品版本 *
	品 流程控制 →	CODING 制品扫描	v1.0
	☑ 安全 →	制品库 Docker 镜像上传 🛛 🖸	自定义制品属性 *
	品 持续部署 ▶	CODING 制品属性设置 🔼	11
	△ 质量管理 ••		
	器 其他 <b>,</b>		④显示高级选项
	品 编译 ·		
	器 发布部署 ·		
	器 消息通知 ▶		

## 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"""
1
}
}

## 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。

← empty-example ② 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 ⑦ 图形化编辑器 文本编辑器	♦ 环境变量 丢弃修改 保存
	S CODING 制品属性设置
开始	<ul> <li>CODING 制品属性设置</li> <li>插件配置 高级配置</li> <li>插件版本 *</li> <li>最新版本 *</li> <li>最新版本</li> <li>制品所在团队地址 ⑦</li> <li>https://my-team.coding.net</li> <li>制品所在项目 *</li> <li>my-project</li> <li>制品所在仓库 *</li> <li>docker</li> <li>制品名称 *</li> <li>node-express</li> <li>制品版本 *</li> <li>10.0</li> </ul>
	key1:value1 key2:value2 仓库令牌用户名 ⑦ personal-token-username
	仓库令牌 ⑦ personal-token-password

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

![](_page_69_Picture_0.jpeg)

### 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}	非填请意项名改小必;注将目称为写
repo	制品所在仓库	_	必填
properites	写入制品的属性	_	非必 填

![](_page_70_Picture_1.jpeg)

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}	请入目牌个令的户输项令或人牌用名
password	拥有权限写入制品 属性的密码	\${PROJECT_TOKEN}	请入 目 牌 个 令 令 或 人 牌
systemHost	其他团队或远程 WePack 系统地 址	\${CCI_CURRENT_TEAM}.\${CCI_CURRENT_DOMAIN}	非必 填

![](_page_71_Picture_0.jpeg)

# 自动填充制品属性

最近更新时间: 2023-05-2113:26:47

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。

🗲 flask-docker 🗹	基础信息 流程配置 触发规则	变量与缓存 通知提醒	2 前往最新构建 操作 ∨ ● 立即构建
静态配置的 Jenkinsfile ⑦	图形化编辑器 文本编辑器		<b>↓</b> 环境变量 医弃修改 保存
			😣 制品库 Docker 镜像上传
→ 2-1 安装依赖	→ 3-1 构建镜像并推送到 …	+ 增加阶段	<b>插件配置</b> 高级配置
↓ 执行 Shell 脚本	。 ∮Ŷ 执行 Shell 脚本		插件版本 *
•	盟 制品库 Docker 镜像		最新版本
+ 增加并行阶段			推送镜像 * ⑦
	+ 增加并行阶段		node-express
			推送目标制品仓库 *
			docker1 ~
			₩ 显示高级选项

## 文本编辑器插件配置如下:

![](_page_71_Picture_9.jpeg)

## 详细参数


2-1 安装依赖	<ul> <li>制品库 Docker 镜像上传</li> <li>插件配置 高级配置</li> <li>node-express</li> <li>推送目标制品仓库</li> <li>推送到团队内仓库</li> <li>推送到团队外 CODING / WePack 仓库</li> <li>项目 *</li> <li>empty</li> </ul>
2-1 安装依赖 + 增加阶段 + 增加阶段 + 增加阶段 + 增加并行阶段 + 10 + 10 + 10 + 10 + 10 + 10 + 10 + 1	插件配置       高级配置         node-express         推送目标制品仓库         ● 推送到团队内仓库         推送到团队外 CODING / WePack 仓库         项目 *         empty
↓	node-express 推送目标制品仓库 ● 推送到团队内仓库 推送到团队外 CODING / WePack 仓库 项目 * empty ~
+ 增加并行阶段 + 增加并行阶段	<ul> <li>推送到团队内仓库</li> <li>推送到团队外 CODING / WePack 仓库</li> <li>项目 *</li> <li>empty</li> </ul>
	制品仓库 * docker1 ~ 目标仓库令牌用户名 ⑦ "\${PROJECT_TOKEN_GK}" 目标仓库令牌 ⑦ "\${PROJECT_TOKEN}"

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		必如前环中有 ta系尝配品的版填果构境,该 g统试该名其本;当建善没,会匹制称他
host	推送目标制品 仓库 host	\${CCI_CURRENT_TEAM}- docker.pkg.\${CCI_CURRENT_DOMAIN}	非填果要的仓h协h需确必;您推制库 took的的。你能能是一个的人,我们就是不是一个的人,我们就是一个人,我们就是一个人,我们就是一个人,我们就是一个人,我们就是一个人,我们就是我们的人,我们就是我们
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	非填果定属如默重则义值盖属必;您义性果认合自属会默性如自的名与值,定性覆认值
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}	非填有品域系名致才填字必;当仓名统不时需写段只制库和域一,要本

## 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



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