



持续集成 常见问题 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

常见问题

Jenkinsfile 语法相关问题 构建执行相关问题 持续集成与代码仓库相关 持续集成与制品库相关 自定义构建节点相关 常见错误码



常见问题 Jenkinsfile 语法相关问题

最近更新时间: 2023-06-15 11:48:47

为什么使用 ci-init 提示无法拉取代码?

2019 年 10 月 10 日 之前创建的构建计划(Job)中的 ci-init 命令会为用户创建一对公私钥,并使其能够拉取项目中的代码仓库。之后创建的构建计划在调 用 ci-init 时,将不会创建拉取代码的公私钥对了。

在此之后新创建的构建计划,我们都会为用户内置一个可以用于拉取对应代码仓库的凭据 ID,直接使用 env.CREDENTIALS_ID 作为 userRemoteConfigs 的 credentialsId 即可。

旧的语法

// 旧版本的语法含有 ci-init sh 'ci-init' checkout([\$class: 'GitSCM', branches: [[name: env.GIT_BUILD_REF]], userRemoteConfigs: [[url: env.GIT_REPO_URL]]]) }	pipeline { agent any stages { stage('检出') { steps {	
checkout([\$class: 'GitSCM', branches: [[name: env.GIT_BUILD_REF]], userRemoteConfigs: [[url: env.GIT_REPO_URL]]]) }	// 旧版本的语法含有 ci-init sh 'ci-init'	
	<pre>checkout([\$class: 'GitSCM', branches: [[name: env.GIT_BUILD_REF]], userRemoteConfigs: [[url: env.GIT_REPO_URL]]]) } }</pre>	

新的语法

pipeline {	
agent any	
stages {	
stage('检出') {	
steps {	
checkout([
\$class: 'GitSCM',	
branches: [[name: env.GIT_BUILD_REF]],	
// 请注意新版的检出语法比旧版新增了 credentialsId: env.CREDENTIALS_ID	
userRemoteConfigs: [[url: env.GIT_REPO_URL, credentialsId: env.CREDENTIALS_ID]]	
])	
}	
}	
}	
}	

CODING 目前已经支持了凭据管理,我们建议用户使用更安全的凭据 ID 来代替之前的 ci-init 操作。



? 说明:

关于凭据如果您想了解更多请参见凭据管理。

单引号和双引号用法差异是什么?

使用 CODING 持续集成时经常需要在 Jenkinsfile 内拼接字符串或使用环境变量作为参数, Jenkinsfile 中的单引号和双引号在使用时,会有些许差异, 以下演示常用的 echo 与 sh 两个命令的差异。

pipeline { agent any environment { MY_ENV = 'this is my env' stages { stage('Test') { steps { script { def MY_ENV = 'define in script' echo "\${env.MY_ENV}" echo "\\${env.MY ENV}" echo "\${MY_ENV}" echo '\${MY_ENV}' }

- echo 在使用单引号时,并不会解析里面的 \$ 符号,而是直接输出原文;在使用双引号时,会打印出环境变量里的 MY_ENV。
- sh 在使用单引号时,将原文当作我们平时在终端里 sh 的命令一样执行,所以可以打印出环境变量里的 MY_ENV。

持续集成流程配置来源的区别是什么?

• 选择使用代码仓库中的 Jenkinsfile 后,该文件将存储至代码仓库中。修改 Jenkinsfile 意味着需在代码仓库中提交修改记录,若修改持续集成的触发条件,还可以自动触发集成任务。



• 使用静态配置的 Jenkinsfile 后,该文件将不会存储在代码仓库中,修改 Jenkinsfile 不会更新代码仓库内容,执行构建时将统一使用静态配置,保障构 建流程的一致性。

如何查看工作空间目录?

在持续集成的部署流程中添加"执行 Shell 脚本"步骤,并在其中添加 pwd 命令。持续集成运行后将输出工作空间目录。



如何自定义环境变量?

.

使用 enviroment 语法创建变量
 以生产日期变量为例・

environment { DATE2 = sh(returnStdout: true, script: 'date +%Y%m').trim() }
← 脚本变量 ☑ ▲础信息 流程配置 触发规则 变量与缓存 通知提醒
静态配置的 Jenkinsfile ⑦ 图形化编辑器 文本编辑器
<pre>80 sh '''cd java-spring-example111111111 82 ls -lart''' 83</pre>
在全局或局部中使用变量 在全局中使用变量:
script { env.cusversionall=sh(returnStdout: true, script: 'date +%Y%m').trim() echo "\${cusversionall}"}





仅在局部(某项步骤中)使用变量:

远程 SSH 执行命令时环境变量不生效怎么办?

由于在使用构建机连接远程 SSH 时使用了"非交互非登录式"连接,因此无法引用远程机器的 /etc/profile 、 ~/.bashrc 等文件配置中的环境变量。 您可以参见以下示例,使用 export 命令再设置变量且用 && 符号连续输入命令。

export PATH=/opt/jdk1.8.0_281/bin:**\$PATH** && java -version



构建执行相关问题

最近更新时间: 2023-06-15 11:44:52

目前主流的计算机操作系统内任何进程的退出都会留下 exit code,并以此判定进程是否按照预期运行。因此持续集成过程中执行进程的 exit code(退出 码)不为 0 就会判定为构建失败。以下是构建执行过程中失败的常见原因:

持续集成的配置文件语法有错误如何处理?

与大多数的编程语言一样,Jenkinsfile 也是由特定领域的语言 (DSL) 组成,语法错误就会导致编译或者运行失败。

测试不通过如何处理?

大多数主流的测试工具或测试框架,在测试逻辑不通过时,默认都会将退出码设置为非 0。

构建超时或构建配额不足如何处理?

每一个团队在使用 CODING 持续集成的时候,都会有一定的配额。为防止恶意使用持续集成,每一个构建任务都会有超时的限制,超时或者构建次数超过配 额系统将会主动中止构建任务。用户遇到配额不足时,可以在团队管理内进行配额调整,购买满足自己实际需求的配额。

如何查看构建日志与构建快照?

CODING 持续集成为用户提供了构建日志,用户可以根据日志内容,判断构建失败的原因。除此之外,CODING 持续集成还提供了每一次构建的配置快照, 用户可以根据快照获取构建使用的配置文件内容和参数,得知是否是配置差异导致的构建失败。

构建日志

← 构建记录#652	构建过程 构建快照 改动记录 测试报告 通用报告 构建产物	设置 启动新构建
❷ 构建成功	推送到标签 20210601.1 时触发	♀ 重新构建
构建过程		
		查看完整日志 🖸
▶ 开始	→ ✓ 检出 2 s → ✓ 安装依頼 14 s → O 检查代码规范	
	→ 从代码仓库检出 2 s → 执行 Shell 脚本 14 s	
	↔ ✓执行 Shell 脚本 <1 s	

构建快照

 构建 	记录#652 构建过程 构建快照	改动记录 〗	则试报告	通用报告	构建产物		✿ 设置	启动新构建
启动参数	环境变量 Jenkinsfile 构建节点							
序号	变量名					变量值		
1	TRIGGER_USER_NAME					88		
2	TRIGGER_USER_GK					UHwXAfctZP		
3	CCI_TRIGGER_METHOD					PUSH		
4	TRIGGER_USER_EMAIL					@coding.net		
5	TRIGGER_USER_ID					207901		

如何在本地运行自动化任务?



用户可以再将自动化的逻辑重新执行一遍(例如:在本地重新运行测试代码)或者实时修改代码获得更多的信息反馈,以此来排查问题。

使用了交互式命令行程序有什么影响?

在持续集成的过程中,用户无法直接使用交互式命令,若使用了呼出交互式命令行窗口的程序会导致构建失败。 常见的命令有 npm login docker login -u xxx (在持续集成登录 docker 时需使用 docker -u xx -p xx 命令)

如何 Debug 构建任务?

如果您需要 Debug 构建运行过程,可以通过在构建过程中添加以下步骤的方式提供 ssh:

steps {
sh 'apt-get update'
sh 'apt-get install -y tmate openssh-client'
sh '''echo -e \'y
\' ssh-keygen -q -t rsa -N "" -f ~/.ssh/id_rsa'''
sh 'tmate -S /tmp/tmate.sock new-session -d'
sh 'tmate -S /tmp/tmate.sock wait tmate-ready'
sh '''
tmate -S /tmp/tmate.sock display -p \'#{tmate_ssh}\'
tmate -S /tmp/tmate.sock display -p \'#{tmate_web}\'
echo "WebURL: \${tmateWeb}"
echo "SSH: \${tmateSSH}"
sh 'sleep 3600'
}

为什么无法连接阿里云服务器?

执行 SSH 命令访问阿里云服务器时提示 Connection reset 错误。

-) SSH Steps: sshCommand Execute command on remote node. ☑ ① <1秒 // 全屏
 - 1
 [2021-08-04 15:58:29] Executing command on ______: mkdir -p /opt/app-service/medical-datahub-api-web-user sudo: false

 2
 Session.connect: java.net.SocketException: Connection reset

此问题是阿里云侧白名单未放行 CODING IP 所致。前往阿里云**安全管控平台 > 安全管控 > 新增访问白名单**,将构建机的 IP 加入至白名单中可以防止其在访 问云服务器时被拦截。

CODING 构建机所使用的出口 IP 如下:

# 中国上海节点			
111.231.92.100			
81.68.101.44			
# 中国香港节点			
124.156.164.25			
119.28.15.65			
# 美国硅谷节点			



170.106.136.17

170.106.83.77

? 说明:

若用户直接复制构建计划页提供的构建机 IP 地址,请去掉 /32 结尾以防格式错误。

持续集成构建并行数最大支持多少?

标准版团队的构建并发数是 1 ,高级版和购买高性能包的团队默认并发数是 20 ,高级版团队如果需要更高的并发数可以让团队负责人在工单中提交申请。标 准版团队可以通过购买高级版或者性能包增加构建配额。

构建计划页面将展示当前通道的情况和机器的配额:



使用镜像推送插件时执行失败如何处理?

镜像推送插件执行失败并出现以下错误提示:

runtime error: invalid memory address or nil pointer dereference

请前往构建计划的流程配置页,使用文本编辑器将 codingcorp 改成 coding-public ,保存后重新启动构建任务。



构建计划获取变量失败

在读取环境变量时失败,此时需检查流程配置中的文本编辑器中,引用该变量时是否采用双引号""包裹。如果确认已使用双引号进行包裹,切换至"文本编 辑器"中查看该变量是否有多余的引号。



(e)	■本变量 ☑		基础信息 流程	配置 触发	规则 变量	与缓存 通知提	醒					尼 前往最新构建	操作 > 🕨 立	即构建
静态配	置的 Jenkinsfile ⑦	图	彩化编辑器 文本编辑器									↓ ↑ 环境变量	丟弃修改	保存
											🗴 上传到 Gen	eric 制品库		0
;码		8-1	使用凭据拉取代码	(+)_(+)	→ 9-1 阶段	9-1		+ 増加阶段]	• 0 结束	插件配置	高级配置		
本		88	withCredentials 凭据		品 上传到	Generic 制品库					将指定的文件」	と传到对应的 Generic 制品仓	;库内,查看完整帮助文	で档じ
本		6 64	执行 Shell 脚本 +		+ 増加	加并行阶段					文件 *⑦ "\${GIT_BRAI	NCH}"		
阶段			+ 増加并行阶段								选择 Generic f	3库*		
											a_b_c			•
											制品版本			
											设置制品版	本号,若不填写,则默认为		
											♦ 显示高级选择	Д		

例如编写了 ""\${GIT_BRANCH}"',此时需要将额外的单引号删除,保存后重新进行构建。

◆ 脚本变量 区 基础信息 流程配置 触发规则 变量与缓存 通知提醒	E 前往最新构建	操作 ~
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
<pre>80 sh '''cd java-spring-example1111111111 82 ls -lart''' 83</pre>		

推送到 TCR 提示"获取临时秘钥失败"错误

当使用持续集成推送至 TCR 时若出现下图中的"获取临时秘钥失败"错误:

集成 / 新建构建计划

🗵 获取临时秘钥时失败

← 构建镜像并推送至 TCR 个人版 (容器服务-镜像仓库)

构建计划名称*

需检查是否在 第三方应用 中已绑定腾讯云账号。确认已绑定后,联系团队负责人或管理员前往腾讯云控制台的"访问管理"页添加 CODING_QCSRole 角 色。

🔗 腾讯云	テᢪ品▼					搜索产品、文档 Q	🕜 小程序 🖸	99+) 集团账号 ▼	备案	工具 マ 支持	;▼ 费用、	9 -
访问管理	角色										CAM角	。色使用说明 🖸
器 概览 으 用户 冬 用户组	0	为什么我的账户出现了新角色? 在云服务中完成特定操作(如授 或者,如果您在某项服务开始)	权创建服务角色) 时,云服务会向用户 支持服务相关角色之前已在使用该服务	发送创建服务角色的授权请求,您问意并授权后,会自动创建推 ,通过邮件等方式各知您后,则会自动在您的账户中创建新角包	服务角色并关联相关策略。 6。							
🖾 策略	8/73	新國角色										© Q ‡
3 角色												
回 身份提供商	角	自色名称	角色ID	角色载体	角色描述			会话最大持续	时间	创建时间	操作	
🕞 联合账号	c	CODING_QCSRole		产品服务 - coding	当前角色为CODING DevOps服务角色,	该角色将在已关联策略的权限范	围内访问您的其他云	2 小时	:	2020-01-07 10:00):10 删除	
(12) 访问密钥	#	40项						10	▼ 条/页	H H 1	/1页	► H

构建失败提示"解析环境变量异常"



大部分构建记录为正常,但偶尔出现"解析环境变量异常"错误。

19)注4/22	RK(又)百/云·	비미미코
✔ 构建成功	wanjian 手动触发 #77 & develop	🛗 36 分钟前 🕓 1 分钟 29 秒
<	外部代码仓库 #76 % develop ->	🛗 43 分钟前 🕓 -
✔ 构建成功	外部代码仓库 #75 % develop -	🗎 17 小时前 🕓 1 分钟 23 秒
✔ 构建成功	主账号 手动触发 #74 😵 develop ≺ 👘 🗍	🗎 3 天前 ① 1 分钟 9 秒

这种情况通常由仓库授权问题导致。若构建计划使用的代码源为关联仓库,需要前往**代码仓库 > 关联仓库**中找到此仓库,取消关联后重新关联,再次重新触发 构建任务。

- > 演示项目 ->						搜索	् 🗳 🔅 🎽
🗵 项目协同	代码仓库 关联仓库						+ 关联代码仓库
全部产品 へ	仓库来源 全部 ▼ 认证方式 全部 ▼ 已开启项	目模块 全部 - 关联人 全部	▼ 搜索仓库 Q				
① 项目概览	代码仓库	仓库来源	认证方式	已开启项目模块	关联人	关联时间	操作
🗵 项目协同	pi-doc	GitHub	OAuth	全部模块	4	2021-09-14 10:34	
♪ 代码仓库</th <th>idu</th> <th>GitHub</th> <th>OAuth</th> <th>全部模块</th> <th>4</th> <th>2021-10-11 16:42</th> <th></th>	idu	GitHub	OAuth	全部模块	4	2021-10-11 16:42	
⑦ 代码扫描 beta >	1-2 个代码库, 共 2 个						
00 持续集成 >							
♣ 持续部署 >							
□ 制品管理 >							
遇 測试管理 >							
■ 文档管理 >							

? 说明:

若本文未能收录您实际遇见的问题,欢迎前往 工单中心 提交使用疑惑,我们将按照实际情况及时补充相关问题的处理方法。



持续集成与代码仓库相关

最近更新时间: 2023-08-09 11:37:32

如何在持续集成中推送代码?

在某些场景下,您可能需要在持续集成阶段推送代码。CODING 的持续集成内置了 Git、SVN 等命令工具,您可以参见如下示例。

pipeline {
agent any
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('修改') {
steps {
sh "echo '# Hello CODING' > README.md"
sh "git add ."
sh "git commit -m 'add README.md' "
}
}
stage('推送') {
steps {
// 使用了 CODING 持续集成系统预置的项目令牌环境变量 PROJECT_TOKEN_GK 和 PROJECT_TOKEN 来推送
// 若希望推送到非本项目或第三方平台的代码仓库,需要自行换成有效的凭据信息
sh "git push https://\${PROJECT_TOKEN_GK}:\${PROJECT_TOKEN}
@e.coding.net/myteam/myrepo.git HEAD:master"
}
}
}
}

如何调用 SVN 仓库?

在默认的持续集成计划的配置过程中,所运行的代码源默认是 Git 类型仓库。若希望使用 SVN 仓库 运行持续集成,下文给出了指引。

前提条件

在开始之前,请先创建项目令牌与申请用户名+密码凭据。

步骤1: 创建项目令牌



1. 前往**项目设置 > 开发者选项 > 项目令牌**页面,单击**新建项目令牌**。设置过期时间后并勾选持续集成所有的权限。

← 项目设置	项目设置 / 项目令牌 / 新建项目令牌					
只 项目与成员 □ 项目公告	新建项目令牌					
<♪ 开发者选项	令牌名称	过期	期时间			
	svn_ci	2	2020-06-19	~		
	 ・	查询、编辑、删除	 WIKI 新建、宣询、编辑、删除 	项目公告 新建、查询、编辑、删除		
	构建(持续集成)权限					
	 ✓ 构建节点 ✓ API 魚 允许构建节点接入 使用 	虫发 API 触发持续集成构建				
	新建取消					

2. 创建完成后会给出用户名及密码。

← 项目设置	开发者选项	项目令牌(1)								
A 项目与成员	接口与事件	项目令牌只用于操作项目。	3令牌只用于操作项目内的功能模块,只对当前项目有效。 不能跟个人令牌通用,如需要设置个人令牌, <mark>请点击这里</mark> 。							
□ 项目公告	外部仓库管理									國目令牌
、小 开发者选项	项目令牌	令牌名称	用户名	密码	创建时间	过期时间	操作			
	WebHook	Token			2020-06-16	2020-06-26	音看密码	编辑权限	禁用	册店会
	凭据管理									

步骤2:申请用户名和密码凭据

前往**项目设置 > 开发者选项 > 凭据管理**页面,单击**录入凭据**录入用户名和密码凭据。用户名和密码需要填写在创建项目令牌时给出的用户名及密码。

← 项目设置	开发者选项	凭据管理(1)						
A 项目与成员	接口与事件	将密码、私钥、证书 整帮助文档 ^[2]	等信息存储到凭据	管理中,可最大程度的提高凭据的安全性和管控使用权	限。在持续集成与部署等模块中使用时	寸,无需重复填写	,直接选择使	用即可。查看完
☑ 项目公告	外部仓库管理							
// 开发者选项	项目今牌				_			录入凭据
	WebHook	凭据名称	已授权数	凭据 ID	凭据描述	凭据类型	更新时间	操作
	係据管理	SVN token	1			用户名 + 密	2 小时	(1946 単)(194
	700 B · 1	SVIV_token	1	0	_	码	前	2003 Pds

创建完成后会给出凭据 ID,稍后需要将此 ID 录入至构建计划的流程配置中。

步骤3: 配置构建计划



1. 在**持续集成 > 构建计划**中单击**新建构建计划配置**,进入**选择构建计划模板 > 基础**页面,选择基础栏中的**空白模板**,这样可以自定义流程配置。 自定义构建过程 ← 选择构建计划模版 <♪ 代码仓库 构建计划是持续集成的基本单元,在这里你可以快速创建一个构建计划,更多内容可以到构建计划详情中进行配置。宣看帮助文档 [2] ∞ 持续集成 构建计划 全部 编程语言 镜像仓库 制品库 基础 API 文档 构建节点 beta ♪ 持续部署 简易模板 并行过程模板 ••• 文档管理 通过检出代码、构建、测试和部署来设置常规步骤。在模板中填充上真实构建和.. 通过检出代码、构建、测试和存档来设置并行构建步骤。其中测试和存档两个步.. 空白模版 多分支操作模版 Ξ. . 允许您根据 Jenkinsfile 的规范来随意定制持续集成流水线过程。 演示如何根据不同的分支集成到测试环境和部署到生产环境 人工确认模版 CODING 合并请求添加评审者 . ×... 该模版演示如何在 CODING 持续集成加入人工评审的步骤 使用 CODING 代码托管发起合并请求时,会自动为合并请求添加选定的评审者。 通过账号密码发起的 SSH 连接 通过私钥发起的 SSH 连接 >_ SSH >_ SSH 通过账号密码发起 SSH 连接 通过私钥发起 SSH 连接 ◎ 项目设置 2. 命名构建计划后,代码源选择**不使用**。 构建计划名称 * SVN_Test 小 代码仓库 ∞ 持续集成 构建过程 构建计划 Jenkinsfile 预览 1 代码仓库 构建节点 beta ↓ 持续部署 代码源 > pipeline { agent any □ 文档管理 > **~**~~~ stages { $\left(\right)$ stage('自定义构建过程') { steps { CODING GitHub.com GitLab.com 私有 GitLab echo "自定义构建过程开始" // 请在此处补充您的构建过程] **F** ' } Ø } } 码云 不使用 3 2 配置来源 ● 使用静态配置的 Jenkinsfile⑦ ✓ 是否前往配置详情 ◎ 项目设置 \ll 取消 确定

3. 完成后在流程配置中填写相应的配置。

pipeline {		
hihemie (
agent any		
aba waa f		
stages {		
stage('检出 S\/N 代码') {		
steps {		
checkout([\$class: 'SubversionSCM',		
// 此外可以添加额外认证		
additionalCredentials: [],		

版权所有:腾讯云计算(北京)有限责任公司



excludedRegions: ", excludedRevprop: ", excludedUsers: ", filterChangelog: false, ignoreDirPropChanges: false, includedRegions: ", locations: [[// 输入上文中创建的凭据 ID credentialsId: '5e25f6a9-675c-4b38-97b0-e907b5fe27cd', // 检出代码时所取出代码的范围 depthOption: 'infinity', // 是否将 SVN 上的外部引用一并检出 ignoreExternalsOption: true, // SVN 的检出目录,此目录是该 Job 工作目录的相对路径 local: '.', // SVN 代码仓库地址 remote: "svn://subversion.e.coding.net/StrayBirds/svn"]], workspaceUpdater: [\$class: 'UpdateUpdater']]) } }

步骤4:添加环境变量

在变量与缓存中添加环境变量,类别选择 CODING 凭据里的用户名 + 密码。

슯	项目概览		← SVN_Test ⊠		通知提醒	操作 ~
	代码仓库			添加		
∞	持续集成	~	流程环境变量	变量名称	+ 添加环境变量	
	构建计划		添加构建计划的环境变量,在手动	请输入变量名称	1文档 亿	
	构建节点 beta		变量名 类别	类别	操作	
4	持续部署	>	SVN_Code Coding 凭	Coding 凭据 🛛 👻	C Ô	
Ľ	文档管理	>		凭据类型		
			缓存目录	● 使用所有类型的凭据		
			1. 开启缓存能够避免每次构建重复	()使用指定的凭据类型		
			 2. 当您的构建缓存出现错误时,可 3. 建议您为 Mayen, Gradle, npr 	默认值		
					重置缓存	
			建议缓存目录: 🗌 项目目录	请选择凭据 ▼		
			(清你)(た) (同新紀方)(日日)	请搜索		
			· 肩总袖八斋安绂仔时日来	用户名 + 密码		
				SVN_token(! 查看凭据		
			保存修改 取消			
				朔定 取消		
ŝ	项目设置	«				

步骤5: 触发构建



您可以选择手动构建或配置 触发方式 进行自动构建,构建成功后如图所示。

<u>ن</u>	项目概览		← 构建记录#1	×	Check out from version control 2 ⊙ 1 秒	,* 全屏
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~	持续集成	~	✓ 构建成功 主账号 手动触发	Ì	1 Checking out a fresh workspace because /root/workspace doesn't exist $2$ Cleaning local Directory .	
	构建计划				3 Checking out svn://subversion.e.coding.net/StrayBirds/svn at revision '2020-06- 16T17:37:48.472 +0800'	
	构建节点 beta		<b>尤修订版本信</b> 息		4 Using sole credentials /******* (SW_LOKEN) in reaum ' <svn: subversion.e.coding.net:3690=""> StrayBirds/svn' 5 A branches</svn:>	
\$	持续部署	>			6 A trunk 7 A tags	
۵	文档管理	>	<b>构建过程</b> 构建快照 改动记录 测试报告 构建产	初	8 At revision 1 9	
			► 开始 检出 SVN 1 秒			
			✓ Check out from v 1 秒			
鏱	项目设置	~				

## 如何拉取多仓库?

## 1. 创建代码仓库项目令牌

在**项目设置 > 开发者选项 > 项目令牌**中,单击**新建项目令牌**,并勾选**读取**代码仓库权限。因涉及到两个仓库,需在代码仓库权限选择**统一配置所有代码仓库** 权限,创建完成后获取用户名与密码。

← 项目设置	项目设置 / 项目令牌 / 新建项目令牌					
<b>北</b> 项目与成员	新建项目今牌					
🗹 项目协同						
☑ 项目公告	令牌名称		过期时间			
▶ 开发者选项</td <td>读取本项目内所有代码仓库</td> <td></td> <td>请选择日期 ~</td> <td></td> <td></td>	读取本项目内所有代码仓库		请选择日期 ~			
℃ 研发规范 beta						
	项目管理权限					
	<b>项目协同</b> 读取与操作项目协同模块	文件 新建、查询、编辑、删除	WIKI 新建、查询、编辑、删除	项目公告 新建、查询、编辑、删除		
	API 文档 发布 API 文档	关联资源 新建、查询、编辑、删除	<b>项目成员</b> 读取与操作项目成员	<b>项目权限</b> 读取与操作项目权限		
	代码仓库权限 ⑦					
	<ul> <li>统一配置所有代码仓库权限</li> </ul>	指定代码仓库配置权限				
	仓库名称	访问权限	操作权限			
	* 项目内所有代码仓库	✓ 读取 读取代码仓库	□ <b>读写</b> 推送至代码仓库	合并请求 新建、查询、编辑、删除	<b>版本发布</b> 新建、查询、编辑、删除	



#### 2. 在持续集成配置中选择不使用代码源。

🔶 prod 🗹	基础信息 流程配置 触	发规则 变量与缓存 通	知提醒	已前往最新构建	操作 ~ ◆ 立即构建
代码源	CODING GitHub.com GitL	ab.com	<b>6</b> 码云 工蜂 通用 Git 仓库	[]] 不使用	
配置来源	● 使用静态配置的 Jenkinsfile ⑦				
节点池配置⑦	● 使用 CODING 提供的云主机进行构建	🕜 F 团队 CI 构建配额信息			
	★: 上海 中国	<b>香港</b> 中国	<b>硅谷</b> 美国		
	公网出口: /32, 公网	別出口: /32,	公网出口: /32,		
	🔵 使用自定义的构建节点进行构建 😨				
保存修改	取消				

3. 编写 Jenkinsfile 配置文件,填写需拉取的代码仓库地址。

pipeline {
 agent any
 stages {
 stage('检出1') {
 steps {
 st get (检出1') {
 steps {
 sh 'git clone "https://\${GIT_USER}:\${GIT_PASSWORD}@e.coding.net/codes-farm/laravel-demo.git"'
 sh 'ls -la'
 }
 }
 stage('检出2') {
 stage('检出2') {
 steps {
 sh 'git clone "https://\${GIT_USER}:\${GIT_PASSWORD}@e.coding.net/codes-farm/laravel-demo/config.git"'
 sh 'ls -la'
 }
 }
 }
}



#### 4. 将第一步申请的项目令牌的用户名与密码添加至持续集成的环境变量中。

¢	debug ② 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 ∨ ▶
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
1 2 3 4 5		<b>流程环境变量</b> 添加构建计划的环境变量, _{变量名}	: 在手动启动构建任务时,环境变量 类别	H 批量添加字符串类型环境变量 + 也将作为启动参数的默认值, <b>查看完整帮助文</b> 默认值
7 8 9	pipeline { agent any	GIT_USER	字符串	*****
10 11 12 13 14	stages { stage('检出1') { steps { sh 'git clone " <u>https://\${GIT_USER}:</u> sh 'ls -la'	\${GIT_PASSWORD}@e.co	ding.net/codes-farm/lar	avel-demo.git"'
15 16 17 18 19 20 21	<pre>} stage('检出2') { steps {     sh 'git clone "<u>https://\${GIT_USER}:</u>     sh 'ls -la' }</pre>	\${GIT_PASSWORD}@e.co	ding.net/codes-farm/lar	avel-demo/config.git"'
22 23 24	} }			

## 如何检出 Git Submodule 代码?

在持续集成构建计划中,若要将子仓库代码作为代码源,需通过流程配置检出 Git Submodule 子仓库代码。

在配置持续集成流程前,请先将子仓库添加至父仓库中。使用 git submodule add 命令添加拟跟踪项目的仓库地址作为子仓库,

git submodule add https://e.coding.net/test/git-sub-module.git

#### 代码提交成功后,在父仓库页将看到此图标:

← demo → 浏览	提交 分支 合并请求	版本 对比 设置		+ 创建代码仓库 🖌 🔫
ndemo	😢 master 👻 🏠	查找文件 🗸 输入以查找文件		<b></b>
🗋 sub	<b>文件</b> 历史 14			们建合并清求
🗋 .gitmodules				00 DXE I / 时小
MI README.md	🐠 主账号 Merge branch 'r	最后提交 fa5fb6474e 🗊 于 几秒前		
	Ca sub @ b43bac2			
	🗋 .gitmodules	主账号	refactor: 新增子账号	几秒前
	MI README.md	主账号	新的代码仓库	2 天前
	README.md			
	demo			
	第一行代码			

#### 步骤1:录入仓库访问凭据

通常情况下,子仓库的访问凭据与父仓库的凭据有差异,也为了避免在持续集成配置中暴露敏感信息,可以先行将父子仓库的访问凭据都录入至项目设置中。



划。		
4	项目设置	项目设置 / 凭据管理 / 录入凭据
22	项目与成员	录入凭据
2	项目协同	凭据类型
$\square$	项目公告	用户名 + 密码 🛛 🗸
	开发者选项	用户名 + 密码
		SSH 私钥 字符
		云 API 密钥
		腾讯云临时授权
		Kubernetes 凭据
		山山 Appdroid 签名证书
		请设置密码,不超过 1000 个字符
		凭据描述
		请输入凭证描述,不超过 100 个字符
		凭据授权
		被选中的构建计划将有权限使用此凭据如何在持续集成中使用凭据 [2]
		持续集成
	**	授权所有持续集成构建计划 ⑦

1. 进入**项目设置 > 开发者选项 > 凭据管理**页面,单击**录入凭据**,在**凭据类型**选择用户名 + 密码或 SSH 私钥并在凭据授权下勾选授权所有持续集成构建计



## 2. 录入完成后获取两者的凭据 ID。

← 项目设置	开发者选项	凭据管理(2	2)					
<ul> <li>项目与成员</li> <li>项目协同</li> </ul>	接口与事件	将密码、私钥、 看完整帮助文档	证书等信息存储 12	¥到凭据管理中,可最大程度的提高凭据的安全性和管控使用权限。若	持续集成与部署等模	块中使用时,无需重	复填写,直接	选择使用即可。查
□ 项目公告	Service Hook							录入凭据
▶ 开发者选项</th <th>凭据管理</th> <th>凭据名称</th> <th>已授权数</th> <th>凭据 ID</th> <th>凭据描述</th> <th>凭据类型</th> <th>更新时间</th> <th>操作</th>	凭据管理	凭据名称	已授权数	凭据 ID	凭据描述	凭据类型	更新时间	操作
		父仓库 git 账号密码	2	Ū	-	用户名 + 密 码	几秒前	编辑 删除
		子仓库 git 账号密码	2	0	-	用户名 + 密 码	几秒前	编辑 <mark>删除</mark>

## 步骤2:配置持续集成流程

参见使用以下 Jenkinsfile 配置:

ipeline {
gent any
ages {
age('检出') {
eps {
neckout([
class: 'GitSCM',
ranches: [[name: GIT_BUILD_REF]],
oGenerateSubmoduleConfigurations: false,
此处配置 Submodule 的检出规则
xtensions: [[
class: 'SubmoduleOption',
是否禁用检出 Submodule
isableSubmodules: false,
是否允许检出时使用 Parent Project 的用户凭据
arentCredentials: false,
是否递归检出所有 Submodule 的更新
cursiveSubmodules: true,
指定参考仓库的路径
eference: ",
是否追踪 .gitmodules 文件中配置的分支的最新提交
ackingSubmodules: false
,
ubmoduleCfg: [
此处配置远程 Parent Project 和 Submodules的检出信息
serRemoteConfigs: [
此处配置远程 Parent Project 仓库 SSH 凭据和仓库地址
redentialsId: '93207d20-****_****-410850900d86',
rl: 'https://e.coding.net/StrayBirds/Parent/parent.git'
此处配置远程 Submodule 仓库凭 SSH 凭据和仓库地址
redentialsId: '560bdc1e-****-***-c8e3ccb3ccc6',
rl: 'https://e.coding.net/StrayBirds/Submodule/sub.git'
如果有更多的 Submodules ,可以在这里增加配置



- ])
- }
- ı
- S
- }
- }

## 运行成功后的日志如下:

● 构建记录#2 例建过程 构建快照 以动记录 测试报告 通用报告 ※ 从代码仓库检	出 12 ③ 1秒
<ul> <li>◇ 构建成功</li> <li>主账号 推送到分支 master 时触发</li> <li>10 12021-08-25</li> <li>https://e.co</li> <li>11 [2021-08-25</li> <li>12 [2021-08-25</li> <li>+refs/heads,</li> <li>12 [2021-08-25</li> </ul>	<pre>14:32:48] &gt; git conrig remote.origin.urt ding.net/StrayBirds/demo/demo.git # timeout=10 14:32:48] &gt; git configadd remote.origin.fetch *rrefs/remotes/origin/* # timeout=10 14:32:48] &gt; git config remote.origin.url</pre>
https://e.cd 构建过程 构建过程 13 [2021-08-25 https://e.cd 14 [2021-08-25 15 [2021-08-25 https://e.cd +refs/merge, 16 [2021-08-25	<pre>ding.net/StrayBirdS/demo/demo.git # timeout=10 14:32:48] Fetching upstream changes from ding.net/StrayBirdS/demo/demo.git 14:32:48] using GIT_ASKPASS to set credentials git 账号密码 14:32:48] &gt; git fetchtagsforceprogress ding.net/StrayBirdS/demo/demo.git +refs/heads/*:refs/remotes/origin/* *:refs/remotes/origin/merge/* 14:32:48] &gt; git config remote.origin1.url</pre>
★ 推出 1s ★ 推出 1s ★ 从代码仓库检出 1s 18 12021-08-25 ★ 从代码仓库检出 1s 19 12021-08-25 ★ 以代码仓库检出 1s 10 12021-08-25 10 12021-08-25 10 12021-08-25 10 12021-08-25 10 12021-08-25 10 12021-08-25 10 12021-08-25 10 12021-08-25 10 12021-08-25 10 12021-08-25 10 12021-08-25 10 12021-08-25 10 12021-08-25 10 12021-08-25 10 12021-08-25 10 12021-08-25 10 12021-08-25 10 12021-08-25 10 12021-08-25 10 12021-08-25 10 12021-08-25 10 12021-08-25 10 12021-08-25 10 12021-08-25 10 12021-08-25 10 12021-08-25 10 12021-08-25 10 12021-08-25 10 12021-08-25 10 12021-08-25 10 12021-08-25 10 12021-08-25 10 12021-08-25 10 12021-08-25 10 12021-08-25 10 12021-08-25 10 12021-08-25 10 12021-08-25 10 12021-08-25 10 12021-08-25 10 12021-08-25 10 12021-08-25 10 12021-08-25 10 12021-08-25 10 12021-08-25 10 12021-08-25 10 12021-08-25 10 12021-08-25 10 12021-08-25 10 12021-08-25 10 12021-08-25 10 12021-08-25 10 12021-08-25 10 12021-08-25 10 12021-08-25 10 12021-08-25 10 12021-08-25 10 12021-08-25 10 12021-08-25 10 1201-08-25 10 1201-08-25 10 1201-08-25 10 1201-08-25 10 1201-08-25 10 1201-08-25 10 1201-08-25 10 1201-08-25 10 1201-08-25 10 1201-08-25 10 1201-08-25 10 1201-08-25 10 1201-08-25 10 1201-08-25 10 1201-08-25 10 1201-08-25 10 1201-08-25 10 1201-08-25 10 1201-08-25 10 1201-08-25 10 1201-08-25 10 1201-08-25 10 1201-08-25 10 1201-08-25 10 1201-08-25 10 1201-08-25 10 1201-08-25 10 1201-08-25 10 1201-08-25 <p< td=""><td>14:32:48]       &gt; git coning memote/on/gin.unt         14:32:48]       Fetching upstream changes from         ding.net/StrayBirdS/demo/sub.git         14:32:48]       soing GT_ASKPASS to set credentials git 账号密码         14:32:48]       &gt; git fetchtagsforceprogress         ding.net/StrayBirdS/demo/sub.git +refs/heads/*:refs/remotes/origin/*         *:refs/remotes/origin/merge/*         14:32:49]       &gt; git rev-parse         24a423a6476fc0440e4af7a77973^{commit} # timeout=10         14:32:49]       &gt; git config core.sparsecheckout # timeout=10         14:32:49]       &gt; git checkout -f a301dbcc629c24a423a6476fc0440e4af7a77973         14:32:49]       &gt; git checkout -f a301dbcc629c24a423a6476fc0440e4af7a77973         14:32:49]       &gt; git checkout -f a301dbcc629c24a423a6476fc0440e4af7a77973         14:32:49]       &gt; git remote # timeout=10         14:32:49]       &gt; git config core.sparsecheckout # timeout=10         14:32:49]       &gt; git submodule init # timeout=10         14:32:49]       &gt; git submodule init # timeout=10         14:32:49]       &gt; git configget remote.origin.url # timeout=10         14:32:49]       &gt; git config -f .gitmodulesget regexp ^submodule\.(.+)\.url         14:32:49]       &gt; git configget submodule.sub.url # timeout=10         14:32:49]       &gt; git configget submodule.sub.url</td></p<>	14:32:48]       > git coning memote/on/gin.unt         14:32:48]       Fetching upstream changes from         ding.net/StrayBirdS/demo/sub.git         14:32:48]       soing GT_ASKPASS to set credentials git 账号密码         14:32:48]       > git fetchtagsforceprogress         ding.net/StrayBirdS/demo/sub.git +refs/heads/*:refs/remotes/origin/*         *:refs/remotes/origin/merge/*         14:32:49]       > git rev-parse         24a423a6476fc0440e4af7a77973^{commit} # timeout=10         14:32:49]       > git config core.sparsecheckout # timeout=10         14:32:49]       > git checkout -f a301dbcc629c24a423a6476fc0440e4af7a77973         14:32:49]       > git checkout -f a301dbcc629c24a423a6476fc0440e4af7a77973         14:32:49]       > git checkout -f a301dbcc629c24a423a6476fc0440e4af7a77973         14:32:49]       > git remote # timeout=10         14:32:49]       > git config core.sparsecheckout # timeout=10         14:32:49]       > git submodule init # timeout=10         14:32:49]       > git submodule init # timeout=10         14:32:49]       > git configget remote.origin.url # timeout=10         14:32:49]       > git config -f .gitmodulesget regexp ^submodule\.(.+)\.url         14:32:49]       > git configget submodule.sub.url # timeout=10         14:32:49]       > git configget submodule.sub.url

## 如何检出其它项目的代码仓库?

在持续集成中,您可以通过 项目令牌 的方式检出其它项目内的 CODING 仓库代码。 为了方便您区分即将要操作的两个不同项目,我们统一将:

- 需要被检出的代码仓库所在项目称为 "项目 A" 。
- 执行检出持续集成任务所在的项目称为 "项目 B" 。

## 步骤1:在项目 A 内创建项目令牌

## 1. 进入项目 A 项目设置 > 开发者选项 > 项目令牌页面,单击新建项目令牌。

← 项目设置	开发者选项	项目令牌(0)							
A 项目与成员	接口与事件	项目令牌只用于操作项目	内的功能模块,只对当前项目有	效。不能跟个人令牌通用,如	需要设置个人令牌, <mark>请点击这里</mark> 。				
团 项目协同	外部仓库管理							新	建项目令牌
□ 项目公告	项目令牌	令牌名称	用户名	密码	创建时间	过期时间	操作		
<♪ 升友者选坝	WebHook				暂无数据				
	任据管理								



## 2. 选择需要检出的代码仓库,按需求配置操作权限。

← 项目设置	项目设置 / 项目令牌 / 新	建项目令牌				
A 项目与成员 团 项目协同	新建项目令牌					
☑ 项目公告	令牌名称		过期时间			
开发者选项	test-aa-checkout		2020-08-10 ~			
	项目管理权限       这代       新雄、童海、编辑、删除       缺陷       新雄、童海、编辑、删除       API文档	<ul> <li>史诗 新建、查询、编辑、删除</li> <li>文件 新建、查询、编辑、删除</li> </ul>	<ul> <li>需求 新建、查询、编辑、删除</li> <li>Wiki 新建、查询、编辑、删除</li> </ul>	<ul> <li>任务 新建、宣询、编辑、删除</li> <li>项目公告 新建、宣询、编辑、删除</li> </ul>		
	代码仓库权限					
	仓库名称	访问权限		操作权限		
	test-dd	✓ 读取 读取代码仓	库	✓ 读写 推送至代码仓库	✓ 合并请求 新建、查询、编辑、删除	✓ 版本发布 新建、查询、编辑、删除

## 3. 单击确定后创建成功。

## 步骤2:在项目 B 创建凭据

#### 1. 进入项目 B 进入**项目设置 > 开发者选项 > 凭据管理**页面,单击**录入凭据**。

← 项目设置	开发者选项	凭据管理(0)						
A 项目与成员	接口与事件	将密码、私钥、证书	等信息存储到凭据管理中	,可最大程度的提高凭据的安全性	生和管控使用权限。在持续集成与部署等模块中使用时, 🔅	无需重复填写,直接选择	使用即可。查看完整帮助文档 🖸	
₩ 项目协同	外部仓库管理							录入凭据
□ 项目公告	项目令牌	凭据名称	已授权数	凭据 ID	凭据描述	凭据类型	更新时间 操	ſF
↔ 开发者选项	WebHook				暂无数据			
	凭据管理							

## 2. 回到之前创建好的项目 A 项目令牌页面,单击查看密码。

← 项目设置	开发者选项	项目令牌(1)					
A 项目与成员	接口与事件	项目令牌只用于操作项目内的功	)能模块,只对当前项目有效。 不能	能跟个人令牌通用,如需要设置个人令的	牌,请点击这里。		
• 项目协同	外部仓库管理						新建项目令牌
□ 项目公告	项目令牌	令牌名称	用户名	密码	创建时间		操作
<h> <h> <h> <h> <h> <h> <h> <h> <h> &lt;</h></h></h></h></h></h></h></h></h>	WebHook	test-aa-checkout			2020-07-10	2020-08-10	查看密码编辑权限禁用删除
	凭据管理						
			请妥善保管您的 令牌用户名: 令牌密码 (token): 我知道了	ý令牌 	×		



3. 在项目 B 的录入凭据窗口凭据类型选择用户名 + 密码,粘贴项目令牌对应信息。

← 项目设置	项目设置 / 凭据管理 / 录入凭据					
A 项目与成员	录入凭据					
<b>衄</b> 项目协同	凭据类型					
□ 项目公告	用户名 + 密码 🖌					
✓ 开发者选项						
	凭据名称*					
	checkout-test-dd					
	用户名* <b>令牌用户名</b>					
	密码* 令牌密码 (token)					
	·····					
	凭据描述					
	请输入凭证描述,不超过 100 个字符					

4. 勾选授权的持续集成项目,单击**保存**。



步骤3:在项目 B 持续集成任务中配置对应的环境变量



. 进入持续	续集成设置 >	流程配置,	添加 <b>从代码仓库检出</b> 步骤,单击 <b>环境变</b>	0		
	既览	empty-	example II 基础信息 流程配置 触发规则 变	量与缓存 通知提醒		操作 ~
☑ 项目协	办同	*4 ** <b>37 88 45 1</b>				inte I.
<♪ 代码仓	全库	静念配直的 Jenkin	Shie ⑦ 图形化类描述 又本集相器			除仔   :
E 代码分	分析 beta >				🗵 从代码仓库检出	0
∞ 持续算 約建計	長戌 ∨ +80	1-1 开始	检出代码	+ 增加阶段	Scm *	
构建于	うら beta		→ 从代码仓库检出		1 - [] \$class: 'GitSCM',	
↓ 持续音	S編 bata		+		3 branches: [lname: env.GIT_BUILD_REF] 4 - userRemoteConfigs: [[ 5 url: env.GIT_REP0_URL,	1,
● 制品庫			+ 增加并行阶段		6 credentialsId: env.CREDENTIALS_ID 7 ]]]	
□ 测试管	音理 >					
D 文档管	音理 >					
					2 改动记录	
					1 轮询	
也可以很	在添加检出流	程之后,进	入持续集成设置 > <b>变量与缓存</b> 中单击 <b>添加</b>	环境变量。		
A 1	프 디 HRI IK					
山上	贝曰慨苋		<ul> <li>empty-example</li> </ul>	基础信息 流程配置	触发规则 <b>变量与缓存</b> 通知提醒	
V	页目协同					
	NTL A C					
17	て的它库		流程环境变量		+ 添加环境变量	
▶_ f	弋码分析 beta	>	添加构建计划的环境变量,在手动启动构建作	王务时,环境变量也将作为启动参数的	默认值,查看完整帮助文档 🖸	
~~ t	土法住 亡					
• f	寸绥未风	Ý	变量名	默认值	操作	
柞	勾建计划			暂无数据		
+	句建국는 바라					
Ť	今年 I L Deta					
<b>⊕</b> ‡	寺续部署	>	缓存目录			
m #			1 开户探方能够避免复次构建重复下载优益。	7份 大幅坦升构建速度		
ш- ц	刘印/牛		2. 当您的构建缓存出现错误时,可以进行重	《叶,八幅旋/闪海连述反。 置缓存操作。		
四 河	则试管理	>	3. 建议您为 Maven, Gradle, npm 等缓存	目录开启缓存。		
					重置缓存	
63	又档官埋	>	建议缓存目录: 项目目录	Maven Gradle npr	n	
			请您输入需要缓存的目录		启用 🔵 ×	
				, 植物 中 马		
				+ 垣加日來		
			保存修改 取消			

#### 2. 分别添加以下两个环境变量:

变量名	默认值
GIT_REPO_URL	需要检出的仓库克隆地址(HTTPS)
CREDENTIALS_ID	在 步骤2 录入的凭据 ID

## • GIT_REPO_URL

	♦ vue-cos > 浏览 提交 分支	合并请求 版本 对比 设置	新建代码仓库
☑ 项目协同	♠ vue-cos	12 master > 🏫 / 输入以查找文件	累 克隆
・小 代码仓库	> 🖿 public	<b>*#</b> EP =	
▶ 代码分析 beta >	> src		克隆仓库 查路华码全面时 终端提示的用户条具你在 CODING 个人设置 田垣写的「毛机」或「桌筒」
∞ 持续集成 >	.gitignore	no message 需要检出的仓库地址	
♣ 持续部署 >	🗋 Jenkinsfile	BIT_REPO_URL	HTTPS V https:// lgit 0
J 制品库	ML README.md	in src Dian Yu customized template	2 天前



#### • CREDENTIALS_ID

← 项目设置	开发者选项	凭据管理(1)							
A 项目与成员	接口与事件	将密码、私钥、证书等信息存	7储到凭据管理中	9, 可最大程度的提高凭据的安全性和管控使用权限。在	E持续集成与部署等模块中使用时,无需重复填	写,直接选择使用即可。	查看完整帮助文档	2	
团 项目协同	外部仓库管理	CREDEN	NTIALS ID						录入凭据
☑ 项目公告	项目今牌	使据名称 已经	協权物	◆ 使握 ID	侵堀描述	任据类型	更新时间	操作	
<♪ 开发者选项	WebHook				2 MAN FRANK	7 UND CON	July 1 Party	DR.IP.	
	WEDITOOK	check-out-test-	10.00	0	-	用户名 + 密码	2 小时前	编辑	删除
	凭据管理	dd							

## 填好后的环境变量:

项目概览     项目物	empty-example © 基础信息 <b>流程配置</b> 触发规则 变量与缓存 通知提醒 操作 ~	
✓→ 代码仓库	静态配置的 Jenkinsfile ③ 図形化磁磁器 文本編編器 文本編編器 文本編編器 (保存 ):	
▶ 代码分析 beta >	流程环境变量	量
∞ 持续集成 ∨	添加构建计划的环境变量,在手动启动构建任务时,环境变量也将作为启动参数的默认值, <b>查看完整帮助文档</b> [2]	
构建计划	<u> </u>	
构建节点 beta		
♣ 持续部署 >	CREDENTIALS JD 员 A 字符串 ****** 区 自	
毌 制品库	+ 增加并行阶段	
△ 測试管理 >		
□ 文档管理 >		

#### 步骤4:开始构建任务,成功检出代码

		← 构建记录#4	×	Check out from version control I2 ① 1秒
☑ 项目协同				
<♪ 代码仓库		✓ 构建成功 ■■●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●		1 using credential fe1bc98c-9e90-4796-9d3e-714f0fe323d1 2 Cloning the remote Git repository
▶ 代码分析 beta	>			4 > git init /root/workspace # timeout=30
∞ 持续集成	~	无修订版本信息		5 Fetching upstream changes from https: 6 > gitversion # timeout=30
构建计划				7 using GIT_ASKPASS to set credentials check-out-test-dd 8 > git fetchtagsprogress https 
构建节点 beta		<b>构建过程</b> 构建快照 改动记录 测试报告 构建产物		9 > git config remote.origin.url https: git #
♣ 持续部署	>			<pre>timeout=30 3 git configadd remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=30</pre>
毌 制品库				<pre>11 &gt; git config remote.origin.url httpsgit #</pre>
△ 測试管理	>	► 开始 ← 检出代码 1s		timeout=30 12 Fetching upstream changes from https 13 using GIT_ASKPASS to set credentials check-out-test-dd
文档管理	>	✓ Check out from versio1 s		<pre>14 &gt; git fetchtagsprogress https</pre>
		✓ Shell Script <1 s		16 Seen 1 remote branch
				b) git jmor Revision b/37/39/2/446411/2b7cc9b582defd470a9418c (origin/master) > git config core.sparsecheckout # timeout=30 > git config core.sparsecheckout # timeout=30 Commit message: "no message" 22 First time build.skipping changelog.

## 如何检出使用 Git LFS 的仓库

在持续集成中用户可以通过流程配置检出使用 Git LFS(Large File Storage)插件管理的代码仓库,实现带有大文件的 Git 仓库持续集成。

## Git LFS 简介

Git LFS 插件加速了带有频繁变动的大文件 (例如图片、视频等)的 git clone 和 git fetch 操作。

每当您在仓库中添加了大文件时,Git LFS 插件会将它储存在本地的 Git LFS cache 中,同时将代码仓库中的大文件内容代替为指向缓存地址的引用。当您 提交代码时,本次提交所涵盖的所有大文件会被提交到远程 Git LFS cache 中,该缓存和您的远程仓库相关联。当您检出带有大文件引用的提交时,插件会 将其替换为缓存中的文件实际内容。

因此,通过 Git LFS 插件的管理,大文件只会在 git checkout 的时候被加载。

#### 如何在构建计划中检出代码?



## 在 构建计划设置 > 流程配置页面,单击从代码仓库检出添加步骤,添加 Git-LFS-Pull 插件。

← test ⊠ 基础信息 流程配置 触发规则 变量与缓存 通知提醒	操作 ~
静态配置的 Jenkinsfile ⑦ 图形化编辑器 文本编辑器	团环境变量 丢弃修改 保存   :
	😵 从代码仓库检出 🕜
1-1 开始	Scm *
→ 从代码仓库检出 + 増加并行阶段	1* []       \$class: 'GitSCM', branches: [[name: env.GIT_BUILD_REF]], extensions: []         4 -       extensions: [ // 添加 GitLFSPull 插件 6 -         5 -       (sclass: 'GitLFSPull'], ], extension(figs: [] url: env.GIT_REPO_URL, credentialsId: env.CREDENTIALS_ID         11       )]]

#### Jenkinsfile

peline {	
gent any	
ages {	
:age('检出') {	
eps {	
neckout([	
class: 'GitSCM',	
ranches: [[name: env.GIT_BUILD_REF]],	
xtensions: [	
添加 GitLFSPull 插件	
iclass: 'GitLFSPull'],	
serRemoteConfigs: [[	
rl: env.GIT REPO URL,	
redentialsId: env.CREDENTIALS ID	
-	

#### 已关联私有 GitLab 仓库但检出代码失败

造成此问题的原因为您在绑定私有 GitLab 时所使用的 URL 与仓库实际的 URL 不一致,有可能导致构建计划执行时代码拉取失败。构建计划执行时,拉取 的仓库地址将从下图的配置信息中拉取。



```
"id":1,
 "description":"",
 "name":"ccjtest",
 "name_with_namespace":"Administrator / ccjtest",
 "path":"ccjtest",
 "path with namespace": "root/ccjtest",
 "created at":"2022-01-06T08:23:45.439Z",
 "default_branch":"master",
 "tag list":[
cont.<u>/ccjtest</u>",
 "web_url":"<u>http:/</u>
 "readme_url":"http://102
                                 >ot/ccjtest/-/blob/master/README.md",
 "avatar url":null,
 "star_count":0,
 "forks_count":0,
 "last_activity_at":"2022-01-06T08:23:45.439Z",
 "namespace":{
     "id":1,
    "name":"Administrator",
     "path":"root",
    "kind":"user",
    "full_path":"root",
    "parent_id":null,
    "avatar_url":"https://www.gravatar.com/avatar/e64c7d89f26bd1972efa854d13d7dd61?s=80\u0026d=identicon",
    "web_url":"http://1
                          /root"
```

? 说明:

访问 http://your-gitlab-address/api/v4/projects 获取此配置信息。

若配置中的地址无法被公网访问,例如为内网地址,那么将拉取失败。

#### 解决办法:

在构建计划中新增 GIT_REPO_URL 环境变量,将可被公网访问的仓库地址填入其中。

~ 构建1	亡录#13	构建过程	科建快照	成动记录	测试报告	週用报告	构建产物	
启动参数	环境变量	Jenkinsfile	构建节点					
1 big	celine {							
2	agent any							
3 9	stages {							
4	stage('检	出') {						
5	steps {	[						
6	check	cout([\$class:	'GitSCM',					
7	brand	hes: [[name:	GIT_BUILD_	REF]],				
8	userF	RemoteConfigs	: [[					
9	url	L: GIT REPO U	RL,					
10	cre	edentialsId:	CREDENTIALS	_ID				
11	1110							
12	1 3							
15	1							
·	. D#	100	<b>主白</b> 法	102300	在市场2十回回山	赤马卜梅方		
•	Ľ	基础	言思 流	柱配直	黓友规则	受重与拨行	子 週料促睡	
流程环境	亦量			≡ 批量:	あ加字符串巻き	刊环培变量	+ 添加环境亦作	
DIVIE	8.×.181			- 100.000		E 1 . 70. Ac. 485	10/04-1-20.00	-
添加构建记	+划的环境变	量,在手动启动构	匈建任务时,环	境变量也将作	为启动参数的影	(认值,查看完整	帮助文档 🖸	
<u>家变量名</u> 將	(会覆盖系统)	内置的环境变量		默认住	1		操作	
GIT_REP		字	守串	http:/	/x.x.x.x/root/cc	jtest.git		

检出代码时提示网络连接异常怎么办?



#### 问题描述:使用持续集成检出代码时报错并提示连接异常。

**解决办法:**造成此问题的原因有可能是构建计划使用了关联仓库,而关联仓库的 OAuth 鉴权已过期。您可以参见 导入或关联外部仓库 重新授权并关联外部仓 库。

## 关联的工蜂仓库无法同步至外部仓库列表

目前需在工蜂授权时选择**当前账号**的授权范围才能成功同步到外部仓库列表,并在持续集成构建任务重被检出,如果您选择的授权范围是**项目组**或<mark>项目</mark>,则无 法成功同步。



授权 CODING DevOps 使用 Coding.net 访问您的

帐号

授权范围:		•	
当前账号	项目组	项目	
选择一个项	īΞ		•
<b>Western</b>	-		
	an Inn		



## 持续集成与制品库相关

最近更新时间:2023-08-0116:27:21

#### 为什么会提示 reached your pull rate limit 错误?

使用 CI 拉取镜像时提示 reached your pull rate limit 报错,如下图所示:

```
[2021-04-29 13:27:41] Step 1/8 : FROM openjdk:8
[2021-04-29 13:27:41] 8: Pulling from library/openjdk
[2021-04-29 13:27:44] toomanyrequests: You have reached your pull rate limit. You may increase
the limit by authenticating and upgrading: https://www.docker.com/increase-rate-limit
script returned exit code 1
```

这是因为 dockerhub 的免费账户存在镜像拉取次数限制,CODING 的出口 IP 达到了 dockerhub 的拉取次数限制而出现的错误,您可以参见下文中的两 个办法解决此问题:

- 将镜像托管至 CODING Docker 制品仓库,详情请参见 Docker 制品库。
- 使用个人 Dockerhub 账号。

若您没有 dockerhub 账号,请在 Docker Hub 官网注册官方账号。

注册完成后修改构建计划配置,在 docker 执行命令前添加此行,填入已注册的账号:

docker login -u <dockerhub username> -p <dockerhub password>
username=\$(docker info | sed '/Username:/!d;s/.* //');
echo \$username

执行时可以在日志查看到正在使用的 dockerhub 账号,若账号符合拉取次数限制条件即可解决此问题。





### 如何自动获取代码 TAG 作为制品的版本号?

\${GIT_TAG} 是内置的环境变量,设计初衷是为了在配有监听 TAG 触发的构建计划中,监听到 TAG 后获取最新的 TAG 值以保存至变量中,因此只有在通 过 TAG 触发构建计划的时候此变量才有参数值,其他的触发方式(手动、定时、远程触发)触发构建时此值为空。

#### Illegal character in path at index

错误详情截图如下:

上传到 Generic 制品库 ② ① <1秒</p>
「2021-08-06 16:36:29] [INF0] codingArtifactsGeneric version 20210324.1 chuck
[2021-08-06 16:36:29] [INF0] upload files: test\2.txt
[2021-08-06 16:36:29] codingArtifactsGeneric [EXCEPTION]: java.lang.IllegalArgumentException:
Illegal character in path at index 66: https://jiyunkeji-generic.pkg.coding.net/ccj-demo/test/chunks/test\2.txt?
Version=latest&fileTag=12f86d52af791131e45abfed58cc386b&fileSize=12&action=part-init
[2021-08-06 16:36:29] at java.net.URI.create(Unknown Source)
[2021-08-06 16:36:29] at org.apache.http.client.methods.HttpPost.<init>(HttpPost.java:73)
[2021-08-06 16:36:29] at net.coding.jenkins.plugins.artifacts.generic.excutor.ChunkExecutor.getUploadId(ChunkExecutor.java:53)
[2021-08-06 16:36:29] at net.coding.jenkins.plugins.artifacts.generic.excutor.ChunkExecutor.exec(ChunkExecutor.java:45)
[2021-08-06 16:36:29] at net.coding.jenkins.plugins.artifacts.generic.excutor.ChunkExecutor.exec(ChunkExecutor.java:45)
[2021-08-06 16:36:29] at net.coding.jenkins.plugins.artifacts.generic.excutor.ChunkExecutor.exec(ChunkExecutor.java:45)
[2021-08-06 16:36:29] at net.coding.jenkins.plugins.artifacts.generic.excutor.ChunkExecutor.exec(ChunkExecutor.java:45)

**解决方案:**此错误常见于使用 Windows 环境作为自定义构建节点。将 Windows 作为自定义节点并使用"上传到 Generic 制品库插件"时不支持携带目 录上传,您需要在步骤中设置进入目录后再上传指定文件。

		×	变更目录子步骤 ————————————————————————————————————	
3-1 阶段 3-1	+ 增加阶段	→ O 结束 + 折	<b>插件配置</b> 高级配置	
↓↑ 变更目录子步骤		5	路径 *	
+			./test	
+ 增加并行阶段				
			上传到 Generic 制品库	$\otimes$
			将指定的文件上传到对应的 Generic 制品仓库内, 重看完整帮助文档 🖸	
			文件 * ⑦	
			2.txt	
			选择 Generic 仓库 *	
			请选择制品库	•
			制品版本	
			设置制品版本号,若不填写,则默认为 latest	

## 在持续集成中打包 maven 时指定了环境变量,为什么上传到制品库没有生效?

您好,请确认在使用 mvn deploy 命令推送到 CODING maven 制品库时是否有加 -- P 参数指定环境(例如 -- Ptest)





持续集成

#### # test配置文件:

#### # 编译打包:

mvn clean package -DskipTests -Ptest -s settinas.xml deploy到CODING maven制品库: mvn deploy -DskipTests -Ptest -s settings.xml

#### # prod 配置文件:

# 编译打包: mvn clean package -DskipTests -Pprod -s settings.xml deployE|CODING maven#1: mvn deplov -DskipTests -Pdev -s settinas.xml



# 自定义构建节点相关

最近更新时间: 2023-02-15 16:19:10

## CODING 中有几种节点类型?

当您在使用 CODING 持续集成进行构建时,本质上是调用计算资源作为**构建节点**完成构建任务。您可以选择使用官方默认提供的云计算资源或自行接入自定 义构建节点两种方式运行构建任务。

## 默认构建节点

CODING 官方提供中国上海、中国香港、美国硅谷三地的计算资源用于执行构建任务,计算资源的限额策略为:

服务名称	标准版	高性能包
构建并发数量	1	弹性伸缩
单次构建时间上线	30 分钟	120 分钟
每月总构建时长	300 分钟	10,000 分钟
构建计划缓存	2 GB	10 GB

默认节点内置了构建环境,其中预装了开发语言 SDK、命令行工具等服务,请参见 默认节点环境。

#### 自定义构建节点

相关内容可参见 构建节点类型 进行查看。

## 不同类型的构建节点配置是什么?

标准版国队所提供的构建节点配置为 **2核4G**:高级版国队、购买高性能包的国队,构建节点的配置为 **8核16G**;自定义构建节点的配置取决于接入机配置大小。

若您需要提升云服务器配置,可以参见文档:调整实例配置。

#### 节点状态一直处于准备中如何解决?

1. 在终端中输入命令 java -version 命令查看版本号是否为 8 或 11。

2. 检查是否正确安装 Jenkins 服务。

若版本号有差异或漏装上述服务,请参见 环境依赖 进行服务重装。

运行 qci_worker remove 命令删除旧有 Worker 服务,参见自定义节点 重新接入。

节点一直处于占用状态如何解决?



#### 按照下图提示单击清理环境按钮。

> qci_worker_te	est v					搜索		¢ 🔅 🍰
构建节点		构建节点池 ⑦	$\otimes$	linux 项目节点池			按)	新节点
全部产后 ^		并行构建数:         0/3         单个构建任务超时时间:30分钟           每月构建分钟数:         133/1000 ⑦         云主机配置:2 核 / 4GB / 100GB 磁盘空间 / Ubuntu		暂无节点池描述内容 🖸			150	
☐ 项目概览		⑦ 您可前往 服务订购 进行配额升级,或联系客服		节点列表 构建任务	授权构建计	划		
☑ 项目协同								
♪ 代码仓库</td <td></td> <td>自定义构建节点池 ③ beta 节点池类型:所有</td> <td></td> <td>▲ 当前构建节点池中存在离线节 删除当前节点后重新接入新节点,</td> <td>点,您可以进入[。] 恢复节点在线。</td> <td>节点所在目录,执行命令</td> <td>&amp; " qci_worker up</td> <td>-d " , 或尝试</td>		自定义构建节点池 ③ beta 节点池类型:所有		▲ 当前构建节点池中存在离线节 删除当前节点后重新接入新节点,	点,您可以进入 [。] 恢复节点在线。	节点所在目录,执行命令	& " qci_worker up	-d " , 或尝试
<ul> <li>① 代码扫描 beta</li> <li>○ 持续集成</li> </ul>	>	default 团队节点地 团队联队		节点	占用状态	最后运行时间	是否开启	操作
♪ 持续部署	>	可用节点: 0/0 已授权构建计划: 13		• VM-8-2-ubuntu 🚍	已离线	1 天前		⇔ ⊛
□ 制品管理	>			• P_PLZZHAO-PC0 =	闲置	20 分钟前	清理环境	- ≙ ⊗
🍝 测试管理	>	windows 项目节点池		• VM-8-17-ubuntu 🚍	占用	1 分钟内		⇔ ⊛
文档管理	>	· 可用节点: 1/1 已授权构建计划: 13		1-3个, 共3个				

若仍未恢复,运行 qci_worker remove 命令删除旧有 Worker 服务,参见 自定义节点 重新接入。

#### 节点处于离线状态如何解决?

执行命令 qci_worker up -d 。若仍无法解决问题,请尝试删除当前节点后,参见 自定义节点 重新接入。

#### 如何解决 Jenkins 启动异常?

当构建任务执行失败后出现**Jenkins 启动异常**错误提示符时,可以尝试重新构建。若无法恢复请运行 qci_worker remove 命令删除旧有 Worker 服务,使 用 自定义节点 一键生成接入命令方式重新接入。

## Windows 系统使用一键脚本接入时失败

检查 powershell 版本是否 >= 5.1.17。

#### 提示凭据不存在

若遇到凭据不存在问题,例如错误日志含有类似信息: Credentialld could not be found ,请清理 cci-agent 服务进程,同时删除主目录下的 .coding 文件夹。删除后参见 自定义节点 接入 worker 服务。

#### 如何访问其他服务器上的 Jenkins?

在装有 Jenkins 服务的自定义节点中运行以下命令:

```
qci_worker stop
qci_worker config JENKINS_HOST=0.0.0.0
qci_worker up -d
```

#### 在浏览器中访问以下网址:

http://目标服务器 IP:15740

## 如何解决 Jenkins 服务启动异常?

您好,当构建任务执行失败后出现**Jenkins 启动异常**错误提示符时,可以尝试重新构建。若无法恢复请运行 qci_worker remove 命令删除旧有 Worker 服务,使用 自定义节点 一键生成接入命令方式重新接入。

安装时提示 qci_worker: command not found 错误



#### 问题详情:

```
如下图所示,安装自定义节点时提示 qci_worker: command not found 错误。
Installing collected packages: certifi
  Found existing installation: certifi 2022.5.18.1
    Uninstalling certifi-2022.5.18.1:
     Successfully uninstalled certifi-2022.5.18.1
Successfully installed certifi-2022.6.15
bash:行440: qci worker: 未找到命令
bash:行479: qci_worker: 未找到命令
INFO : installing jenkins ...
  % Total
          % Received % Xferd
                              Average Speed Time
                                                    Time
                                                             Time Current
                              Dload Upload
                                             Total
                                                     Spent
                                                             Left Speed
100
     145 100
                145
                      0
                            0
                                365
                                         0 --:--:-- --:--
                                                                     365
100 68.5M 100 68.5M
                      0
                            0
                              3502k
                                         0 0:00:20 0:00:20 --:-- 3421k
  % Total
           % Received % Xferd Average Speed Time
                                                    Time
                                                            Time Current
                              Dload Upload
                                             Total
                                                    Spent
                                                             Left
                                                                  Speed
100
     145 100
                                        0 --:--:-- --:--:--
               145
                      0
                            0
                                416
                                                                     416
100 292M 100 292M
                      0
                            0
                             4915k
                                        0 0:01:00 0:01:00 --:-- 5134k
bash:行497: qci worker: 未找到命令
bash:行499: qci_worker: 未找到命令
bash:行500: qci_worker: 未找到命令
INFO : ----> Register
bash:行506: qci_worker: 未找到命令
INFO : ----> Start agent
bash:行519: qci_worker: 未找到命令
INFO : finished.
```

#### 解决办法:

1. 在终端中执行 whereis qci_worker 或 which qci_worker 命令查看 qci_worker 的所在路径。

#### 2. 检查是否将 qci_worker 的路径添加至 PATH 中。

root@localhost ~j# which qci_worker	
/usr/bin/which: no qci_worker in (.:/opt/node-v14.18.1-linux-x64/bin:/opt/apache-maven-3.8.1/bin:/opt/jdk1.8.0_281/	bin:/usr/local/sbin:/usr/sbin:/usr/bin:/ro
ot/bin)	
[root@localhost ~]#	DATH沿右/usr/local/hin
[root@localhost ~]# whereis qci_worker	
qci_worker: /usr/local/bin/qci_worker	➡ 所以找不到qci worker
[root@localhost ~]#	· · · · · · · · · · · · · · · · · · ·
[root@localhost ~]# export PATH=/usr/local/bin:\$PATH 添加 /usr/local/bin 到 PATH	
[root@localhost ~]#	

3. 添加至 PATH 后重新执行安装命令。

## 执行构建时提示 npm not found

## 问题详情:

已在自定义构建节点中安装了 npm,但是在运行构建过程中依然报错 npm not found 。

#### 解决办法:

此问题常见于接入自定义构建节点池后,再安装 npm 的构建节点,因为 qci_worker 未能读取 npm 的所在路径。

1. 执行 qci_worker stop 命令停止服务。

- 2. 再分别执行 ps -ef |grep jenkins、ps -ef|grep qci 命令查看是否仍存在残留的 qci 、Jenkins 进程。若存在则手动 kill 对应的 PID 进程。
- 3. 最后执行 qci_worker up -d 命令重新启动服务。

#### 若问题依旧存在,请参考以下步骤:

1. 执行 which npm 命令查看 npm 所在路径。



- 2. 执行 qci_worker stop 命令暂停服务。
- 3. 执行 In -s + 在第一步运行 which npm 所得到的路径命令,例如执行 In -s /usr/bin/npm。
- **4. 执行** qci_worker up -d 命令重启 qci 服务。

## 自定义节点如何缓存目录?

- 1. 在终端中运行 qci_worker config NODE_LOCAL_WORKSPACE_CACHE=True 命令指定本机中的缓存目录。
- 2. 运行 qci_worker stop 命令停止服务。
- 3. 运行 qci_worker up -d 命令重启服务。

#### ? 说明:

构建运行完成后依旧会清除目录,下一次构建会把缓存放入本次构建的工作目录,缓存本地存储路径默认为: ~/codingci/jenkins_cache。



# 常见错误码

最近更新时间: 2023-06-15 11:35:21

您可以使用网页搜索功能检索错误码以快速定位文档。

## ./gradlew: not found

此问题常见于使用了错误的持续集成模板。您可以检查所使用的代码仓库中是否有 gradlew 文件或者文件的目录位置是否与持续集成中定义的文件路径相符。

← java-spring-example -	<b>浏览</b> 提交 分支 合并请	球 版本 对比	; 设置		
🔒 java-spring-example	P master ▼     ↑ 直找文件 ∨	输入以查找文件			
> gradle/wrapper	<b>文件</b> 历史 1				
] .gitignore	eff 专用演示账号 Initial commit				
Dockerfile	gradle/wrapper	专用演示	Initial commit		
MI README.md	src src	专用演示	Initial commit		
build.gradle	🗋 .gitignore	专用演示	Initial commit		
🗋 gradlew.bat	Dockerfile	专用演示	Initial commit		
🗅 settings.gradle	MI README.md	专用演示	Initial commit		
	🗅 build.gradle	专用演示	Initial commit		
	🗅 gradlew	专用演示	Initial commit		
	🗋 gradlew.bat	专用演示	Initial commit		
	🗋 settings.gradle	专用演示	Initial commit		

## ./gradlew: Permission denied

此错误是因为文件缺少执行权限,将文件赋予执行权限即可。例如在执行文件之前添加命令: chmod +x gradlew 。

## codingArtifactsGeneric

完整错误码: codingArtifactsGeneric ERROR: no file found 。

出现此错误需检查构建过程的步骤中所定义的文件名或路径是否正确,所在的文件的相对路径为: /root/workspace ,例如按下图所示填写了 README.md 文件,那么实际所使用的路径为: /root/workspace/README.md 。



	♦ 环境变量 丢弃修改	保存:
	🗵 上传到 Generic 制品库	0
][	+ 增加 插件配置 高级配置	
	将指定的文件上传到对应的 Generic 制品仓库内,查看完	整帮助文档 🖸
1	文件*⑦	
	README.md	
	选择 Generic 仓库 *	
	请选择制品库	•
	制品版本	
	设置制品版本号,若不填写,则默认为 latest	
	₩ 显示高级选项	

## MissingPropertyException

完整错误码: groovy.lang.MissingPropertyException: No such property: REPO_URL for class: WorkflowScript\r 此错误一般由没有注入环境变量,执行失败引起。错误码中将提示由哪个环境变量所引起,例如此提示为 REPO_URL 变量缺失。

#### file does not exist

完整错误码: The specified user settings file does not exist: /root/workspace/./settings.xml 此错误表示执行在 Maven 命令时找不到 settings.xml 文件。可以通过 ls -ltr 命令看看当前目录是否有这个文件,如果没有,可以把本地的 settings.xml 和 pom.xml 文件提交到代码仓库,然后通过检出代码将其拉取至编译机器中,请参见 参考代码仓库。

## reached pull rate limit

完整错误码: toomanyrequests: You have reached your pull rate limit 此错误提示由于 dockerhub 对免费用户拉取镜像次数限制导致的。请参见 文档详情 以解决此问题。

## no checksums available

使用 Maven 编译时报错: Failed to execute goal on project xxxxxx Checksum validation failed, no checksums available 。 您可以在持续集成设置 → 流程配置中的命令行添加 -c 参数,例如: mvn clean package -c -DskipTests -gs settings.xml 后解决此问题。

## result 9001



错误信息详情:使用镜像更新插件时报错:result 9001 {project_auth_null=无法解析项目令牌。

				♦Ŷ 环境变量 丢弃修改	保存
	_	<b>流程环境变量</b> 添加构建计划的环境变量,在手动启动构刻	<b>能任务时,环境</b> 变	Ⅲ 批量添加字符串类型环境变量 + 量也将作为启动参数的默认值, 查看完整帮助文	添加环境变量 档 12
→ 4-1 部署到远端 Kubern	+ 增加阶段 4	变量名	类别	默认值	操作
镜像更新		DOCKER_IMAGE_NAME	字符串	my-docker-image	๔⊗
H STRO EQ		DOCKER_BUILD_CONTEXT	字符串		๔⊗
1 MI +X		CD_ACCOUNT_NAME	字符串	lhk	๔⊗
		DOCKER_IMAGE_VERSION	字符串	\${GIT_LOCAL_BRANCH:-branch}-\${	๔⊗
		DOCKER_REPO_NAME	字符串	docker	28
		DOCKERFILE_PATH	字符串	Dockerfile	28
		CD_NAMESPACE_NAME	字符串	liaohongkun-1	๔⊗
		CD_MANIFEST_TYPE	字符串	Deployment	๔⊗
		CD_CONTAINER_NAME	字符串	туарр	๔⊗
		CD_MANIFEST_NAME	字符串	myapp-deployment	๔⊗
		CD_CREDENTIAL_INDEX	字符串	979785	๔⊗
		CD_CREDENTIAL_ID	字符串	8e6f9fff00fb40ab828cccaac74d8bae	28
		CD_PERSONAL_ACCESS_TOK 🔒	字符串	***	28

删除 CD_PERSONAL_ACCESS_TOKEN 变量,然后再次编辑插件生成新的个人令牌,保存即可。

### Credentialld could not be found

出现此报错的原因为凭据不存在,需清理 cci-agent 服务进程,同时删除主目录下的 .coding 文件夹。删除后重新接入 自定义节点 Worker 服务。

## Cannot run program "nohup"

完整错误码: Cannot run program "nohup" (in directory "C:\codingci\tools\jenkins_home\workspace\xxxx") 完成的报错记录如下图所示:

## 执行 Shell 脚本 2 0 < 1秒

yarn

1 Cannot run program "nohup" (in directory "C:\codingci\tools\jenkins_home\workspace\939544-cci-13280258-705231"): CreateProcess error=2,系统找不到指定的文件。

✓ 全屏



#### 这是因为采用 Windows 自定义构建节点时执行了 shell 脚本,将 shell 脚本换成 bat 脚本即可。



## runtime error

完整错误码: runtime error: invalid memory address or nil pointer dereference 问题描述:使用镜像推送插件时失败,返回此错误码。



解决办法:请前往构建计划设置 > 流程配置 > 文本编辑器修改图中所示的命令行,将参数 codingcorp 改为 coding-public ,保存后重新启动构建。



### Switch Maven repository

完整错误码: Switch Maven repository 'maven(<http://mirrors.tencentyun.com/nexus/repository/maven-public/>)' to redirect to a secure protocol (like HTTPS) or allow insecure protocols

问题描述:使用 gradle 时返回此错误码。

解决办法:需要前往持续集成设置 > 流程配置,在制品编译步骤前添加一条 shell 命令。

rm /root/.gradle/init.gradle

## check pkg manager fail

在接入自定义节点时,若节点中的 yum 包管理工具版本过低则有可能出现此错误码。在自定义节点中执行 yum --version 2 命令查看是否能够正常返回版本 号。若版本过低则需执行 yum update 命令升级版本。

exec: "qci-plugin": executable file not found in \$PATH: unknown custom plugin execute failed.

运行持续集成任务失败后返回此错误码:

12 [2022-03-17 10:47:11] OCI runtime exec failed: exec failed: container_linux.go:367: starting container process caused: exec: "qci-plugin": executable file not found in \$PATH: unknown 13 custom plugin execute failed.

此错误码常见于使用自定义构建环境执行持续集成任务。因为自定义环境中的镜像没有预置 qci-plugin 环境。



製造配置           特選环境           特選环境           「市法联运行全局特理任务的环境,直看帮助文档 2           使用成为经理环境           日定义方式           日定义方式           石均趣环境安装指定镜像           日定义方式           石均趣环境安装指定镜像           日定义方式           日定义方式           日定义方式           日定义方式           日定义方式           日定义方式           日定文方式           日定义方式           日定、日本           日本           「CODING 口方のにない           「使用目表の           「使用なり、「「「「「「「「「「「「」」」」」」」」」」           「「「「「「」」」」」」」」           「「」」」」」           「「」」」」」           「」」」」」」           「」」」」」」」」」」」」」」」」           「」」」」」」」」」」」」」」」」」」」」」」」」」」」」」」」」」」」」
神趣环境         「「法建写合局物建任务的环境」童看物意文旨 2         使用就认得建环境         自定义均建环境         自定义均建环境         自定义均建环境         自定义方式         在均速环境でのDicker 镜像         ② openjdk8         ○ openjdk8
講选課运行全局物建任务的环境, 童者帮助文档 2         使用就认均速环境         ● 自定义构建环境         日定义方式         在均速环境安装指定镜像         日定以方式         在均速环境安装指定镜像         日常の空気指定镜像         日常の空気指定镜像         日常の空気指定镜像         日常の空気指定镜像         日常の空気指定镜像         日常の空気指定镜像         日常の空気指定镜像         日常の空気指定鏡像         日常の空気指定鏡像         日常の空気指定鏡像         日常の空気になった。         日常の空気になった。         日常の空気になった。         日常の空気         日常の空気         日常の空気の         「日常の日本         「日常の空気         日常の空気の         「日本         「日
<ul> <li>● 自定义构建环境 自定义方式</li> <li>在构建环境安装指定镜像</li> <li></li></ul>
自定义方式         在均建环境安装指定镜像         複像未源         CODING 官方 Docker 镜像         Docker 镜像。         ② openjdk8         Docker 镜像。         ② denjdk8         Docker 镜像。         ③ openjdk8         Docker 镜像。         ③ openjdk8         Docker 镜像。         ③ openjdk8         Docker 镜像运行参数         第1- v/ etc / hosts         ④ 使用银节点的工作空间 ③         不規变量 ③         CODING_DOCKER_IMAGE         CODING_DOCKER_IMAGE         * SIPPOJECT_NAME.toLow         + 添加参数                 Dicker 鏡像         ●         ①         ①         ○         ○         ○         ○         ○         ○         ○         ○         ○         ○         ○         ○         ○         ○         ○         ○         ○         ○         ○         ○         ○
福島市島市中央へをはなんのあい         現像来源         CODING 官方 Docker 银像         Docker 银像・         ① openjdk8         ① openjdk8         ② openjdk8         ② dp用間市点的工作空间 ③         X現空量 ③         CODING_DOCKER_REG_H       = "SjCCI_CURRENT_TEAMI+( CODING_DOCKER_REG_H         CODING_DOCKER_REG_H       = "SjCCI_CURRENT_TEAMI+( CODING_DOCKER_IMAGE         CODING_DOCKER_IMAGE       = "SjCROJECT_NAME toLow         + 添加参数       * ① CODING_DOCKER_IMAGE         ● 前往最新构建       操作 ✓       ● 立即体 ● 公 如即体
CODING 官方 Docker 張像 CODING 官方 Docker 張像 Docker 張 Docker ReG H = "SiCCI_CURRENT_TEAM]-( Docker ReG Docker ReG Dock
Docker 镜像*         ④ openjidk8         Docker 镜像运行参数         回: v/ etc / hosts : / etc / hosts         ● 使用根节点的工作空间 ③         K現安量 ③         CODING_DOCKER_REG_H = "\$(CC1_CURRENT_TEAM)+(CODING_DOCKER_IMAGE = "\$)(PROJECT_INAME.toLow)         + 添加参数 <b>CODING DOCKER_IMAGE</b> = "\$)(PROJECT_INAME.toLow)
● openjak3           Docker 镜像运行参数           回: ~ / etc / bots : / etc / bots           ● 使用限节点的工作空间 ①           环境变量 ①           CODING_DOCKER_REG_H = "SICCI_CURRENT_TEAM!~           CODING_DOCKER_IMAGE = "SIPROJECT_NAME.to.low           + 添加参数           面是改用在特定的阶段中再使用自定义构建环境           配 前往最新构建         操作 ✓           ● 立即料
Docker 機像运行参数         回: v/ ec / hosts: / etc / hosts         逆 使用限节点的工作空间 ③         环境变量 ③         CODING_DOCKER_REG_H = "\$(CCI_CURRENT_TEAM)+( CODING_DOCKER_IMAGE = "\$(PROJECT_NAME.toLow)         + 添加参数         面是改用在特定的阶段中再使用自定义构建环境         堅 前往最新构建       操作 ✓       ① 立即样         於 环境变量       医弃修改       係
SI: - v / vtc / hosts 使用限节点的工作空间 ⑦ 不填交量 ⑦ CODING_DOCKER_REG_H = "SICCI_CURRENT_TEAMI+- CODING_DOCKER_IMAGE = "SIPROJECT_NAME.toLow + 添加参数
<ul> <li>◆使用根节点的工作空间 ③</li> <li>环境变量 ③</li> <li>CODING_DOCKER_REG_H = 'SICCI_CURRENT_TEAMI-( CODING_DOCKER_IMAGE = 'SIPROJECT_NAME.to.low</li> <li>+ 添加參数</li> <li>而是改用在特定的阶段中再使用自定义构建环境</li> <li>配前往最新构建 操作 ✓ ● 立即格</li> <li>於环境变量   医弃修改   係</li> </ul>
K項变量 ⑦          CODING_DOCKER_REG_H       =       *\$(CCL_CURRENT_TEAM)+         CODING_DOCKER_IMAGE       =       *\$(PROJECT_NAME to Low)         + 添加参数       -       *         配       前往最新构建       操作 ∨       ● 立即林         於       环境变量        丢弃修改
coDING_DOCKER_REG_H       = "\$(CCI_CURRENT_TEAM)-(         coDING_DOCKER_IMAGE       = "\$(PROJECT_NAME:toLow)         + 添加参数       - 添加参数         配前往最新构建       操作 ✓       ① 立即校         於 环境变量       丢弃修改       係
coDING_DOCKER_JMAGE       = "SIPROJECT_NAME.toLow"         + 添加参数         而是改用在特定的阶段中再使用自定义构建环境         尼前往最新构建       操作 ✓
+ 源加參数 而是改用在特定的阶段中再使用自定义构建环境 配前往最新构建 操作 ✓ ● 立即料 ∳ 环境变量   丢弃修改   保
而是改用在特定的阶段中再使用自定义构建环境
示太阶段构建任务的环境 查考帮助文档 12
11-10-2020月20日 日
/ 北小功王×1-9%
となった
建环境安装指定镜像 🗸 🗸
冬源
DING 官方 Docker 镜像 V
ər 镜像 *
openjdk:11 ×
ər 镜像运行参数
- v / etc / hosts : / etc / hosts
DI ko

docker: not found

可以在构建任务执行过程中进行人工确认并填写用户自定义的表单,通知确

认者进行确认操作。查看完整帮助文档 🖸



使用持续集成推送镜像时出现下图报错:

	[2022-03-17 10:43:23] + docker build -t ccj-demo/php-laravel/java-spring-app:master-
	bdb58a99b0885e459bb108284b2a2931b0fae072 -f Dockerfile .
2	[2022-03-17 10:43:23] /root/workspace@tmp/durable-343dca2c/script.sh: 1:
	/root/workspace@tmp/durable-343dca2c/ <u>script.sh</u> : docker: not found

3 script returned exit code 127

出现此错误的原因有可能是在全局中使用了自定义镜像作为构建环境。建议不要在"开始"阶段中使用自定义构建环境,否则持续集成任务全局都会置于此自 定义环境下执行,而是改用在特定的阶段中再使用自定义构建环境。

变量与缓存 通知提醒	☑ 前往最新构建 操作 ∨ ● 立即构建
	♦ 环境变量 丢弃修改 保存
4-1 阶段 4-1       ▲         + 增加阶段	
	如 : - v / etc / hosts : / etc / hosts ✓ 使用根节点的工作空间 ⑦
	<b>人工确认</b> 可以在构建任务执行过程中进行人工确认并填写用户自定义的表单,通知确 认者进行确认操作。查看完整帮助文档 🖸

如果要在全局中使用自定义镜像作为构建环境,那么可以在 Docker 镜像中运行参数中填写下列命令:

-v /var/run/docker.sock:/var/run/docker.sock -v /usr/bin/docker:/usr/bin/docker



● 小 小 小 小 小 小 小 小 小 小 小 小 小 小 小 小 小 小 小	← tcr-example 区 基础信息 流程配置 触发规则	变量与缓存 通知提醒	図 前往最新构建 操作 ∨ ● 立即构建
开始         1-1         检出         2-1         触发 CD llu′cheng         3-1         推送镜像         构建环境	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
在内建环境安装指定镜像 〜 	▶ 开始 → Hh → 从代码仓库检出 → 从代码仓库检出 + 増加并行阶段	• • • • • • • • • • • • • • •	★ 基础配置 构建环境 请选择运行全局构建任务的环境。查看帮助文档 Ø 使用默认构建环境 自定义方式 在构建环境安装指定镜像 ~ 镜像来源 CODINS 官方 Docker 镜像 ~ Docker 镜像 * Docker 镜像正行参数 - v/var/run/docker.sock -v /usr/bin/d

## selenium.common.exceptions.WebDriver

## 问题描述:

使用持续集成中的默认构建节点运行自动化测试任务时失败,返回错误码为: selenium.common.exceptions.WebDriver。

#### 解决办法:

出现此报错的原因是执行自动化测试任务时需使用到 chrome driver 服务,而持续集成的 默认节点环境 暂时未能提供此服务。您可以参见 自定义节点 自行 接入已安装 chrome driver 服务的计算节点,并重新执行持续集成任务。

## exec: "docker": executable file not found in \$PATH: unknown

## 问题描述:

使用持续集成中的代码扫描插件时出现此错误,截图如下:

8	代码扫描 🕐 🕜 2秒	" 全屏
aster ∼ bdb58a9 ସ	<ol> <li>[2022-03-17 14:25:44] version: 5.8.5.STANDARD</li> <li>[2022-03-17 14:25:45] [SCAN-CI] 2022-03-17 14:25:45.708   Initialing a un manual type sc threshold obtained by jenkins file.</li> <li>[2022-03-17 14:25:45] [SCAN-CI] 2022-03-17 14:25:45 708   scanning headshift is master.</li> </ol>	an,
	4 [2022-03-17 14:25:45] [SCAN-CI] 2022-03-17 14:25:45.708   using scan plan publicasdasd 行 方案	全库默认
	5 [2022-03-17 14:25:45] [SCAN-CI] 2022-03-17 14:25:45.708   scan plan id from jk file is 9 6 [2022-03-17 14:25:45] [SCAN-CI] 2022-03-17 14:25:45.708   actually use scan plan id 9715	971565 665
	<pre>/ [2022-03-1/ 14:25:46] 24 8 [2022-03-17 14:25:46.029   STEP1 -&gt; patching the client a report running status to coding</pre>	ind
→ × 阶段 3-1 2 s	9 [2022-03-17 14:25:46] [SCAN-CI] 2022-03-17 14:25:46.030   Ci Build has not initial 10 [2022-03-17 14:25:46] [SCAN-CI] 2022-03-17 14:25:46.417   ci Build status init 11 [2022-03-17 14:25:46] [SCAN-CI] 2022-03-17 14:25:46.417   STEP2 -> exec client	
× 代码扫描 2 s	12 [2022-03-17 14:25:46] OCI runtime exec failed: exec failed: container_linux.go:367: star container process caused: exec: "docker": executable file not found in \$PATH: unknown	ting
	13 [2022-03-1/ 14:25:46] [SCAN-CI] 2022-03-1/ 14:25:46.739   ===================================	it Exec
	<pre>15 [2022-03-17 14:25:47] [SCAN-CI] 2022-03-17 14:25:47.032   error occur and report to codi {"ciBuildNumber":5,"jobId":1172501,"scanMetaId":285107,"scanId":0,"scanMetaIdToGetNotify :285107}</pre>	ng: Config"

## 解决办法:

出现此问题的原因有可能是使用了自定义构建环境。



← 测试镜像使用插件2 🗵	基础信	息 流程配置	触发规则 变量与缓存	通知提醒				<b>区</b> 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							64 X	下境变量	丟弃修改	保存
							基础配置				
▶ 开始	-(+)	-1 检出	+ 2-1 编译		<del></del>	阶段 3-1	 构建环境				
	4	→ 从代码仓库检出	。	ihell 脚本 +	Ä	代码扫描	请选择运行: 使用默试	全局构建任务的环 人构建环境	「境。查看	帮助文档 🖸	
		+ 増加并行阶段	+ 増	加并行阶段		+ 増加并行阶段	<ul> <li>自定义</li> <li>自定义:</li> </ul>	勾建环境 方式			
							在构刻镜像来》	韭环境安装指定镜 源	像		~
							CODI	NG 官方 Docker	鏡像		~
							Docker	镜像 *			
							٢	openjdk:8			~
							Docker	镜像运行参数			
							如:>				
							⊻ 使月	<b>书根节点的工作</b> 空	间 ⑦		
							环境变量 ⑦				
							CODING_DO	CKER_REG_H	= "\${CC	I_CURRENT_TEAM}	-( 🛞
							CODING_DO	CKER_IMAGE	= "\${PR	OJECT_NAME.toLov	~ ⊗
							+ 添加参数				

建议在全局或代码扫描阶段中选择默认构建环境。若仍希望使用自定义节点作为构建环境,那么请在自定义节点中预先安装 Docker 环境。

## Illegal character in path at index

错误详情截图如下:

## **上传到 Generic 制品库** [2] ③ < 1秒

_		
		[2021-08-06 16:36:29] [INFO] codingArtifactsGeneric version 20210324.1 chuck
		[2021-08-06 16:36:29] [INFO] upload files: test\2.txt
		[2021-08-06 16:36:29] codingArtifactsGeneric [EXCEPTION]: java.lang.IllegalArgumentException
		Illegal character in path at index 66: https://jiyunkeji-generic.pkg.coding.net/ccj-
		demo/test/chunks/test\2.txt?
		version=latest&fileTag=12f86d52af791131e45abfed58cc386b&fileSize=12&action=part-init
		[2021-08-06 16:36:29] at java.net.URI.create(Unknown Source)
		[2021-08-06 16:36:29] at org.apache.http.client.methods.HttpPost. <init>(HttpPost.java:73)</init>
		[2021-08-06 16:36:29] at
		net.coding.jenkins.plugins.artifacts.generic.excutor.ChunkExecutor.getUploadId(ChunkExecutor
		ava:53)
		[2021-08-06 16:36:29] at
		net.coding.jenkins.plugins.artifacts.generic.excutor.ChunkExecutor.exec(ChunkExecutor.java:4
		[2021-08-06 16:36:29] at
		net.coding.jenkins.plugins.artifacts.generic.ArtifactsGenericStep\$ArtifactsGenericStepExecut
		<pre>n.doChunkUpload(ArtifactsGenericStep.java:531)</pre>
此	错误常	常见于使用 Windows 环境作为自定义构建节点。将 Windows 作为自定义节点并使用"上传到 Generic 制品库插件"时不支持携带目录上传

)

您

┛ 全性



需要在步骤中设置进入目录后再上传指定文件。			
		※ 变更目录子步骤	0
3-1 阶段 3-1	+ 增加阶段	<b>结束</b> + <b>插件配置</b> 高级配置	
∲ 变更目录子步骤		路径 *	
		./test	
+ 增加并行阶段			
		上传到 Generic 制品库	8
		将指定的文件上传到对应的 Generic 制品 建看完整帮助文档 🖸	仓库内,
		<b>工</b> 件 * ⑦	
		2.txt	
		选择 Generic 仓库 *	
		请选择制品库	~
		制品版本	
		设置制品版本号,若不填写,则默认	为 latest