持续集成

常见问题

产品文档

腾讯云

文档目录

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

```
pipeline {
agent any
stages {
stage('检出') {
steps {
// 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]]
])
}
}
}
}
```

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}"
// 输出内容为 this is my env

echo "\${env.MY_ENV}"
// 输出内容为 ${env.MY_ENV}

echo "${MY_ENV}"
// 输出内容为 define in script

echo '${MY_ENV}'
// 输出内容为 ${MY_ENV}

sh 'echo ${MY_ENV}'
// 输出内容为 this is my env

sh "echo ${MY_ENV}"
// 输出内容为 define in script

sh "echo ${env.MY_ENV}"
// 输出内容为 this is 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。

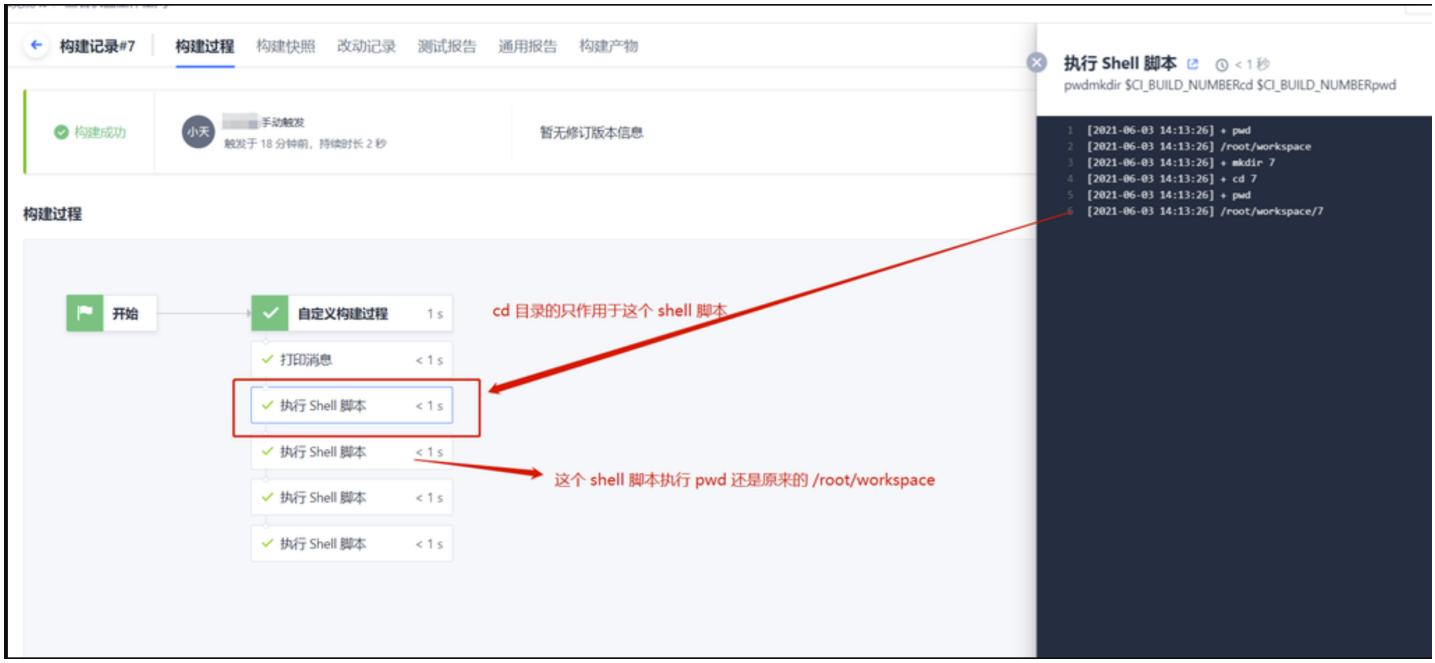
持续集成流程配置来源的区别是什么？

- 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()
}
```

- 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}"}
```

```
stage('定义日期变量-局部-全局') {
  steps {
    script {
      def cusversion=sh(returnStdout: true, script: 'date +%Y').trim()
      echo "${cusversion}"
      env.cusversionall=sh(returnStdout: true, script: 'date +%Y%m').trim()
      echo "${cusversionall}"
    }

    sh 'echo $DATE2'
    echo "${env.MAGIC_VERSION}"
  }
}
stage('打印全局变量cusversionall') {
  steps {
    script {
      echo "cusversionall: ${cusversionall}"
    }

  }
}
```

仅在局部（某项步骤中）使用变量：

```
script {
def cusversion=sh(returnStdout: true, script: 'date +%Y%m').trim() echo "${cusversion}"}
```

```
stage('定义日期变量-局部-全局') {
  steps {
    script {
      def cusversion=sh(returnStdout: true, script: 'date +%Y').trim()
      echo "${cusversion}"
      env.cusversionall=sh(returnStdout: true, script: 'date +%Y%m').trim()
      echo "${cusversionall}"
    }

    sh 'echo $DATE2'
    echo "${env.MAGIC_VERSION}"
  }
}
```

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

构建快照

序号	变量名	变量值
1	TRIGGER_USER_NAME	[redacted]
2	TRIGGER_USER_GK	UHwXAfctZP
3	CCI_TRIGGER_METHOD	PUSH
4	TRIGGER_USER_EMAIL	[redacted]@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）

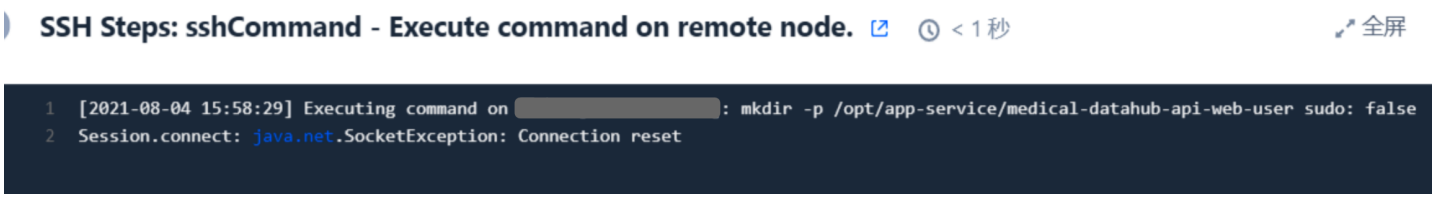
如何 Debug 构建任务?

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。



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
```

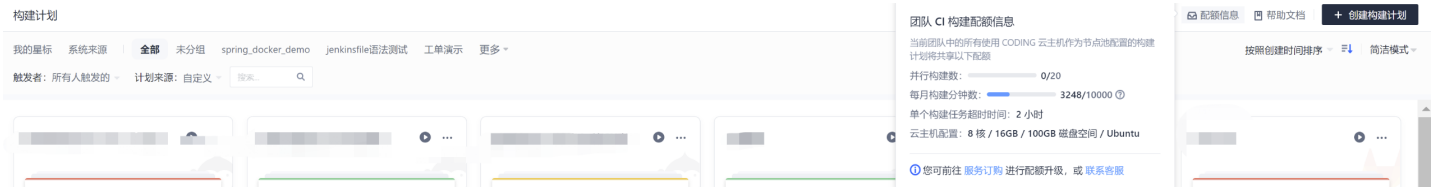
```
170.106.136.17

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。

```
pipeline {
  agent any
  stages {
    stage('检出') {
      steps {
        checkout([$class: 'GitSCM',
        branches: [[name: GIT_BUILD_REF]],
        userRemoteConfigs: [[
          url: GIT_REPO_URL,
          credentialsId: CREDENTIALS_ID
        ]]])
      }
    }
    stage('编译') {
      steps {
        sh 'rm /root/.gradle/init.gradle'
        sh './gradlew build --debug'
      }
    }
    stage('构建镜像并推送到 CODING Docker 制品库') {
      steps {
        sh "docker build -t ${CODING_DOCKER_IMAGE_NAME}:${DOCKER_IMAGE_VERSION} -f ${DOCKERFILE_PATH} ${DOCKER_BUILD_CONTEXT}"
        useCustomStepPlugin(key: 'codingcorp:artifact_docker_push', version: 'latest', params: [image:"${CODING_DOCKER_IMAGE_NAME}:${DOCKER_IMAGE_VERSION}",repo:"${DOCKER_REPO_NAME}",properties:'[]'])
      }
    }
  }
  environment {
    CODING_DOCKER_REG_HOST = "${CCI_CURRENT_TEAM}-docker.pkg.${CCI_CURRENT_DOMAIN}"
    CODING_DOCKER_IMAGE_NAME = "${PROJECT_NAME.toLowerCase()}/${DOCKER_REPO_NAME}/${DOCKER_IMAGE_NAME}"
  }
}
```

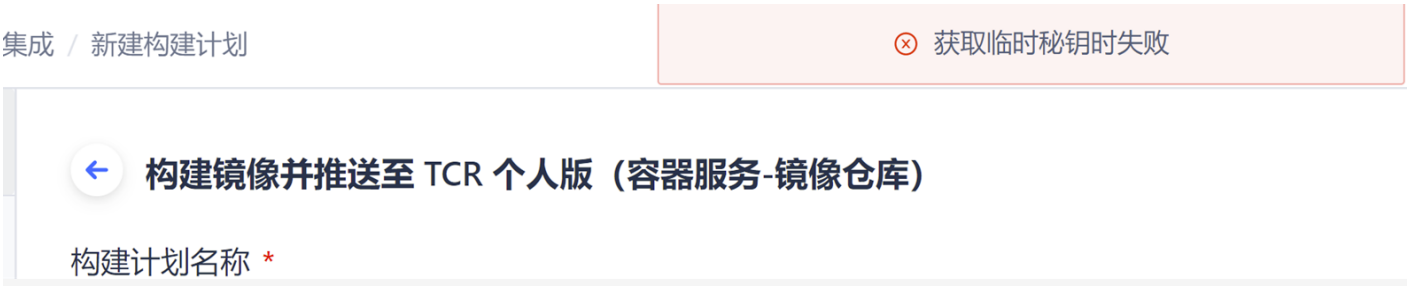
构建计划获取变量失败

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。

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。

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：



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。

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。

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。

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。

2. 创建完成后会给出用户名及密码。

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。

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。

2. 命名构建计划后，代码源选择**不使用**。

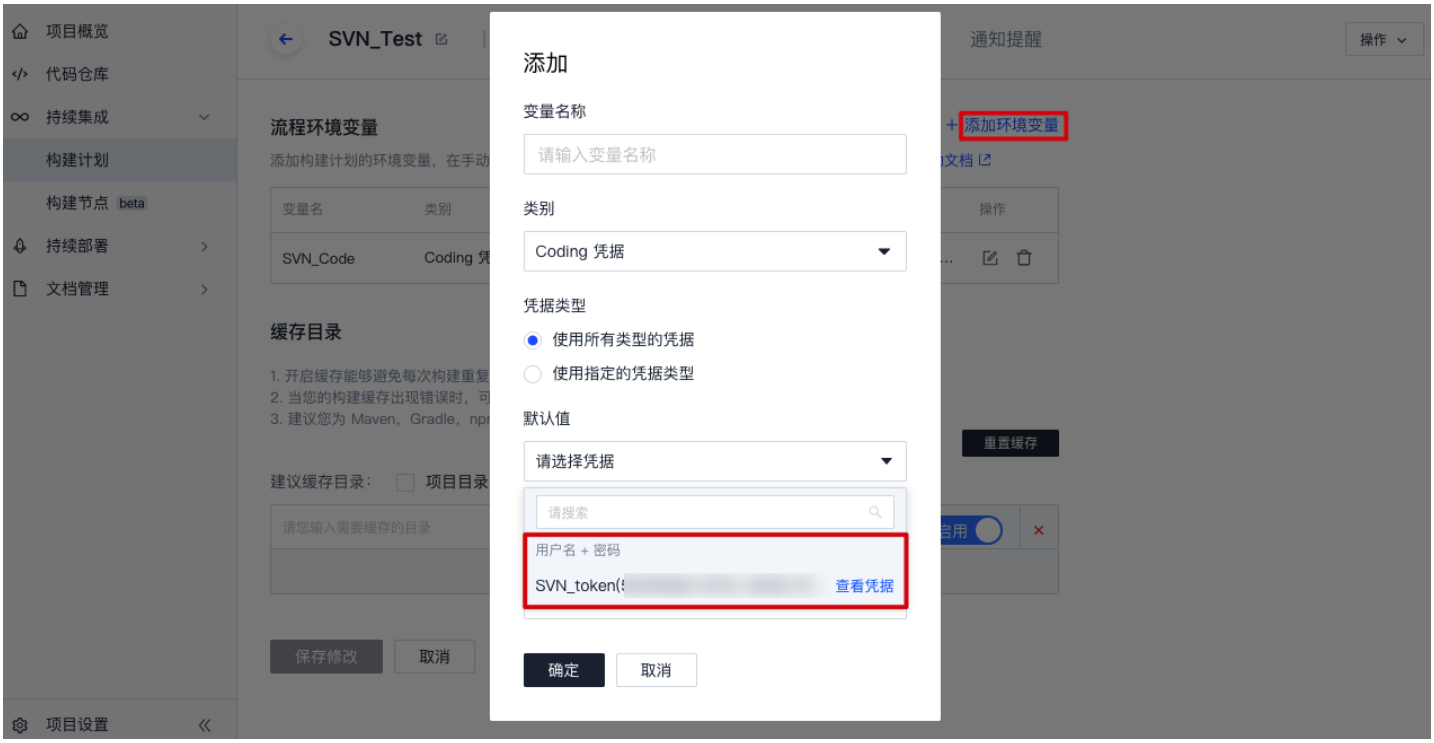
3. 完成后在流程配置中填写相应的配置。

```
pipeline {
agent any
stages {
stage('检出 SVN 代码') {
steps {
checkout([$class: 'SubversionSCM',
// 此处可以添加额外认证
additionalCredentials: [],
```

```
excludedCommitMessages: '',
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']])
}
}
}
}
```

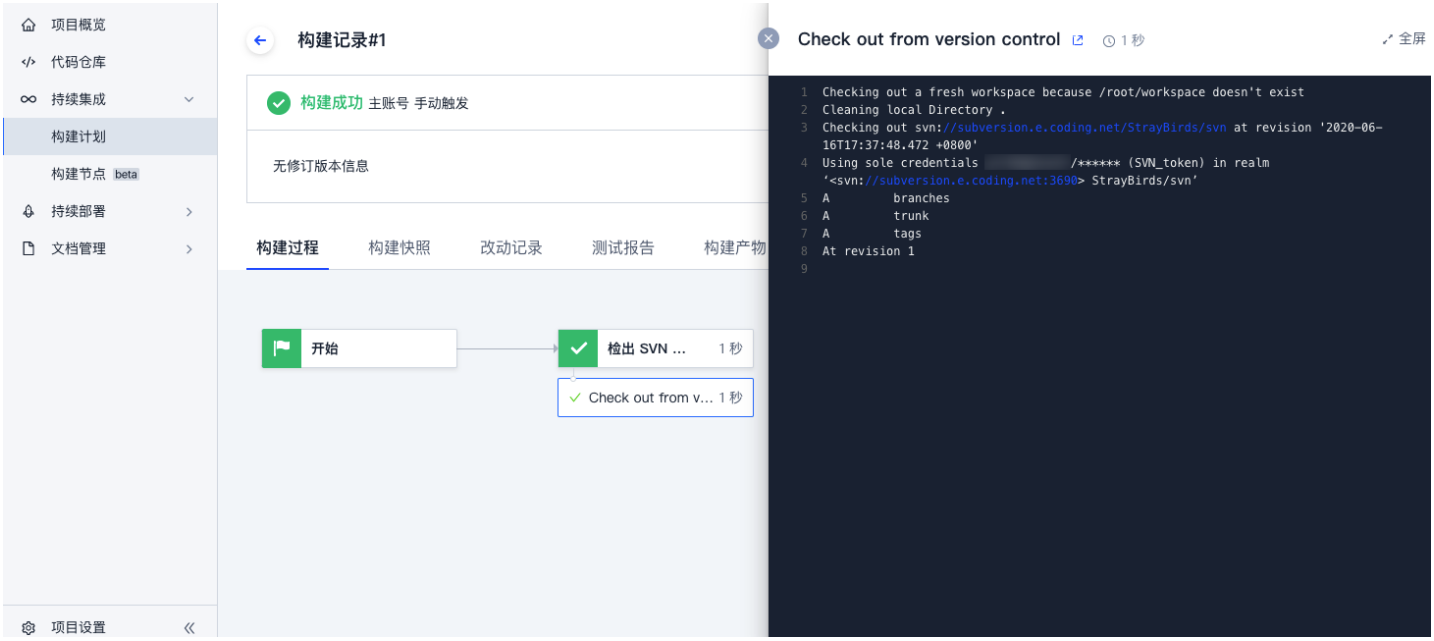
步骤4：添加环境变量

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。



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。



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。

2. 在持续集成配置中选择不使用代码源。

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 {
sh 'git clone "https://${GIT_USER}:${GIT_PASSWORD}@e.coding.net/codes-farm/laravel-demo.git"'
sh 'ls -la'
}
}
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。

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：

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。

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划**。

2. 录入完成后获取两者的凭据 ID。

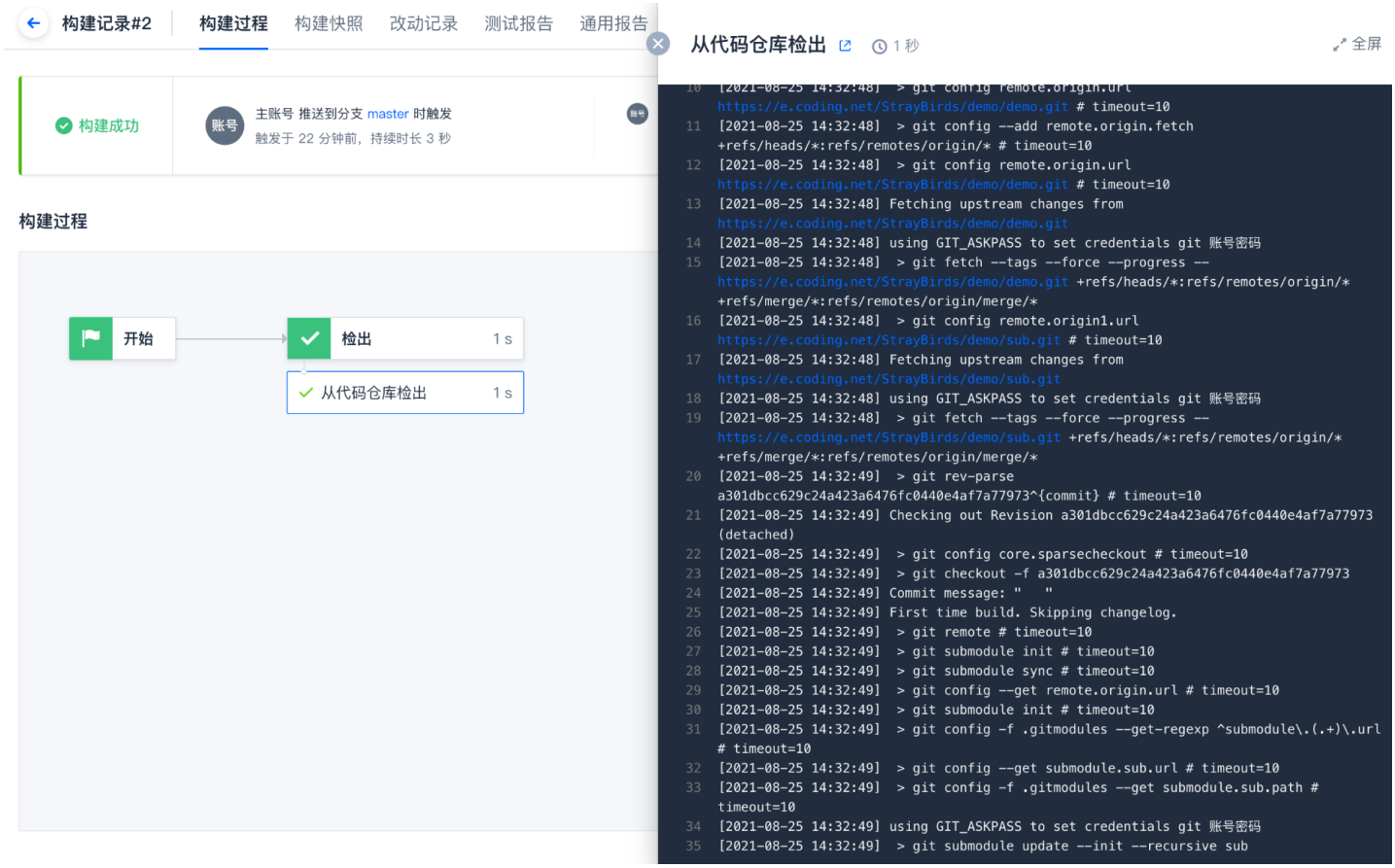
步骤2：配置持续集成流程

参见使用以下 Jenkinsfile 配置：

```
pipeline {
agent any
stages {
stage('检出') {
steps {
checkout([
$class: 'GitSCM',
branches: [[name: GIT_BUILD_REF]],
doGenerateSubmoduleConfigurations: false,
// 此处配置 Submodule 的检出规则
extensions: [[
$class: 'SubmoduleOption',
// 是否禁用检出 Submodule
disableSubmodules: false,
// 是否允许检出时使用 Parent Project 的用户凭据
parentCredentials: false,
// 是否递归检出所有 Submodule 的更新
recursiveSubmodules: true,
// 指定参考仓库的路径
reference: '',
// 是否追踪 .gitmodules 文件中配置的分支的最新提交
trackingSubmodules: false
]],
submoduleCfg: [
],
// 此处配置远程 Parent Project 和 Submodules的检出信息
userRemoteConfigs: [
[
// 此处配置远程 Parent Project 仓库 SSH 凭据和仓库地址
credentialsId: '93207d20-****-****-****-410850900d86',
url: '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
[
credentialsId: '560bdc1e-****-****-****-c8e3ccb3ccc6',
url: 'https://e.coding.net/StrayBirds/Submodule/sub.git'
],
// 如果有更多的 Submodules ，可以在这里增加配置
```

```
]
])
}
}
}
}
```

运行成功后的日志如下：



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** 。

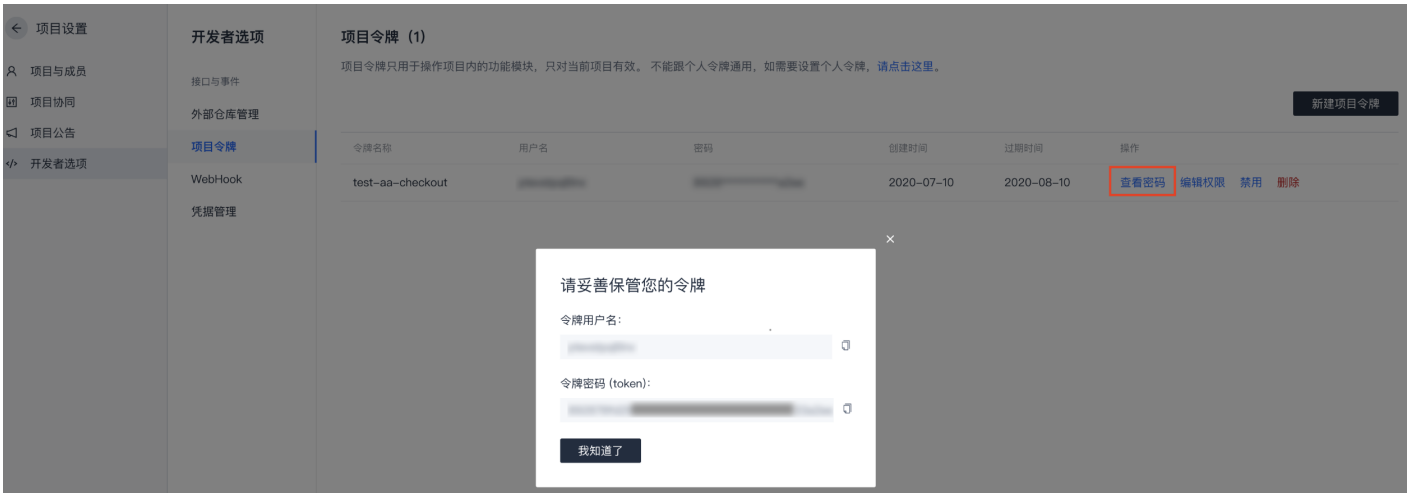
2. 选择需要检出的代码仓库，按需求配置操作权限。

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**。

2. 回到之前创建好的项目 A 项目令牌页面，单击**查看密码**。



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。

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

1. 进入持续集成设置 > **流程配置**，添加**从代码仓库检出**步骤，单击**环境变量**。

也可以在添加检出流程之后，进入持续集成设置 > **变量与缓存**中单击**添加环境变量**。

2. 分别添加以下两个环境变量：

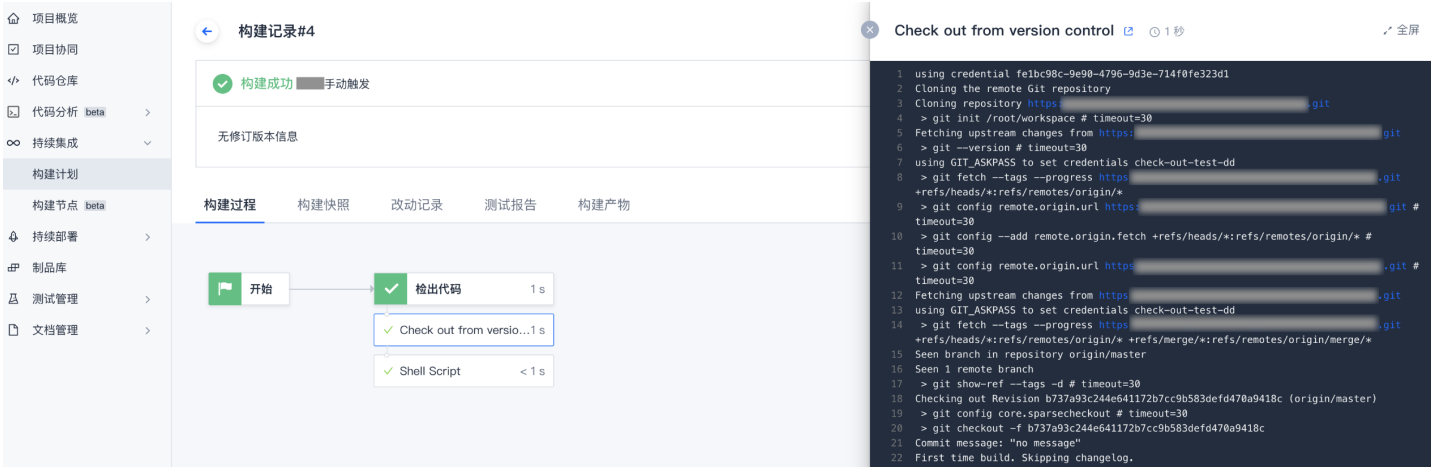
变量名	默认值
GIT_REPO_URL	需要检出的仓库克隆地址（HTTPS）
CREDENTIALS_ID	在 步骤2 录入的凭据 ID

- GIT_REPO_URL

- CREDENTIALS_ID

填好后的环境变量：

步骤4：开始构建任务，成功检出代码



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。

Jenkinsfile

```
pipeline {
agent any
stages {
stage('检出') {
steps {
checkout([
$class: 'GitSCM',
branches: [[name: env.GIT_BUILD_REF]],
extensions: [
// 添加 GitLFSPull 插件
[$class: 'GitLFSPull'],
],
userRemoteConfigs: [[
url: env.GIT_REPO_URL,
credentialsId: 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":[

    ],
    "ssh_url_to_repo":"git@1          :root/ccjtest.git",
    "http_url_to_repo":"http://15          48/root/ccjtest.git",
    "web_url":"http:/          oot/ccjtest",
    "readme_url":"http://159          oot/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://          /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。**

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**项目**，则无法成功同步。

持续集成与制品库相关

最近更新时间：2023-08-01 16:27:21

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。

```
1  [2021-05-07 11:05:09] + docker login -u [redacted] -p ******
2  [2021-05-07 11:05:10] WARNING! Using --password via the CLI is insecure. Use --password-stdin.
3  [2021-05-07 11:05:12] WARNING! Your password will be stored unencrypted in
   /root/.docker/config.json.
4  [2021-05-07 11:05:12] Configure a credential helper to remove this warning. See
5  [2021-05-07 11:05:12] https://docs.[redacted]login/#credentials-
   store
6  [2021-05-07 11:05:12]
7  [2021-05-07 11:05:12] Login Succeeded
8  [2021-05-07 11:05:12] + docker info
9  [2021-05-07 11:05:12] + sed /Username:/!d;s/.* //
10 [2021-05-07 11:05:13] WARNING: No swap limit support
11 [2021-05-07 11:05:13] + username=[redacted]
12 [2021-05-07 11:05:13] + echo [redacted]
13 [2021-05-07 11:05:13] [redacted]
14 [2021-05-07 11:05:13] + docker pull openjdk:8
15 [2021-05-07 11:05:14] 8: Pulling from library/openjdk
16 [2021-05-07 11:05:18] 8: Pulling from library/openjdk
17 [2021-05-07 11:05:18] bd8f6a7501cc: Pulling fs layer
18 [2021-05-07 11:05:18] 44718e6d535d: Pulling fs layer
19 [2021-05-07 11:05:18] efe9738af0cb: Pulling fs layer
20 [2021-05-07 11:05:18] f37aabde37b8: Pulling fs layer
21 [2021-05-07 11:05:18] b87fc504233c: Pulling fs layer
```

打印出当前登录的 dockerhub 账号

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 < 1 秒 全屏

```
[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)
[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.ArtifactsGenericStep$ArtifactsGenericStepExecution.doChunkUpload(ArtifactsGenericStep.java:531)
```

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。

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)

```
# dev 配置文件:
# 编译打包:
mvn clean package -DskipTests -Pdev -s settings.xml

# deploy 到 CODING maven 制品库:
mvn deploy -DskipTests -Pdev -s settings.xml
```

```
# 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#l:
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。

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： CredentialId 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   145    0     0    416      0 --:--:-- --:--:-- --:--:--   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。

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 ln −s + 在第一步运行 which npm 所得到的路径命令，例如执行 ln -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 。

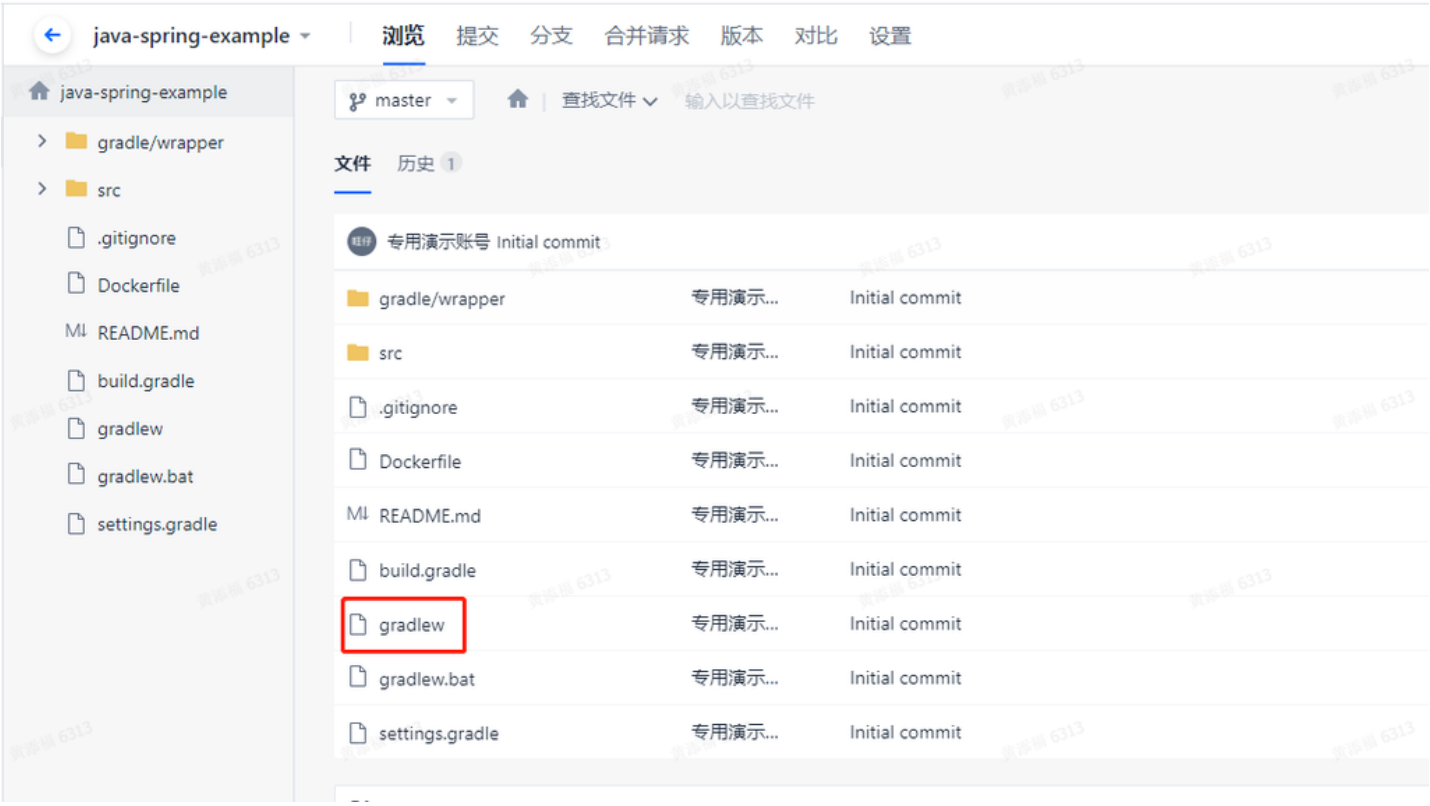
常见错误码

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。



./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 。

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。

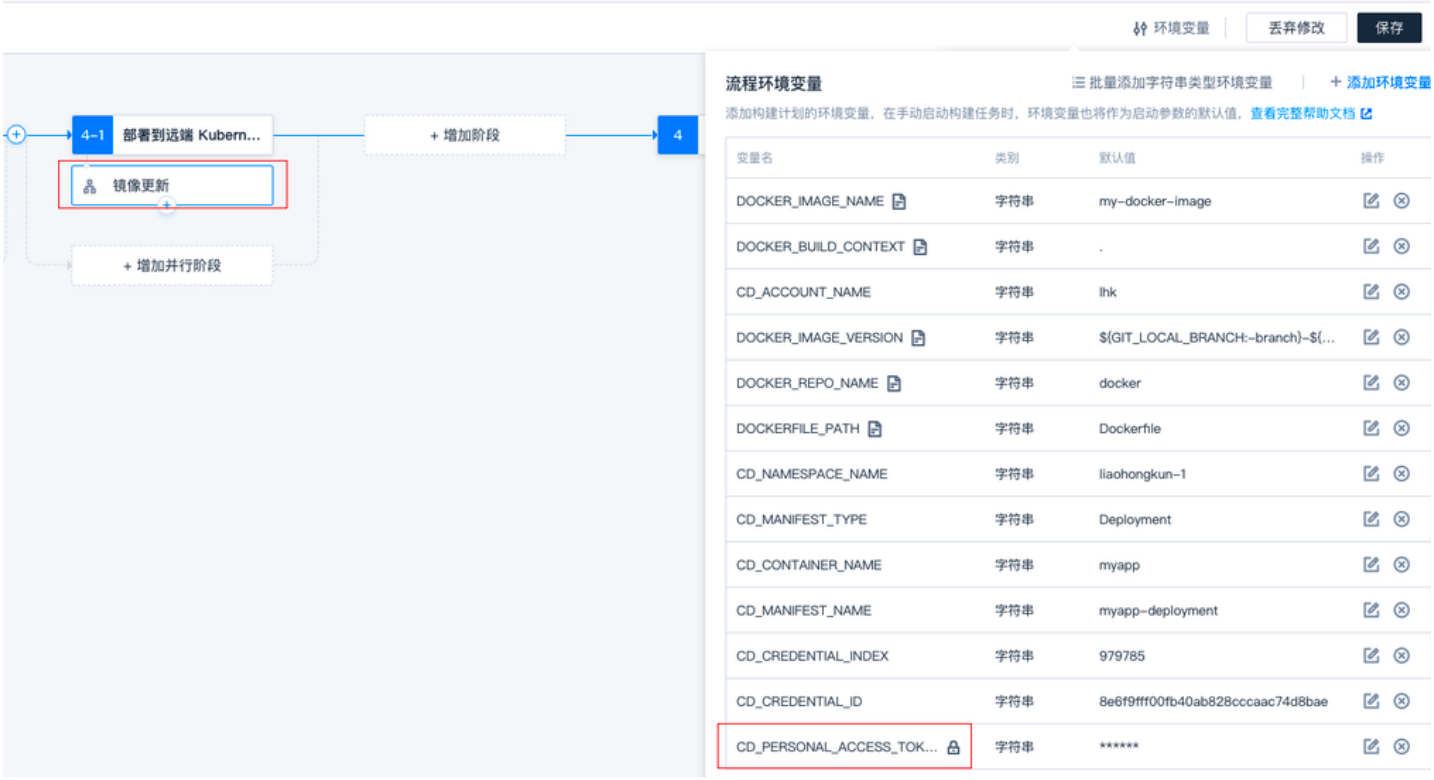
no checksums available

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。



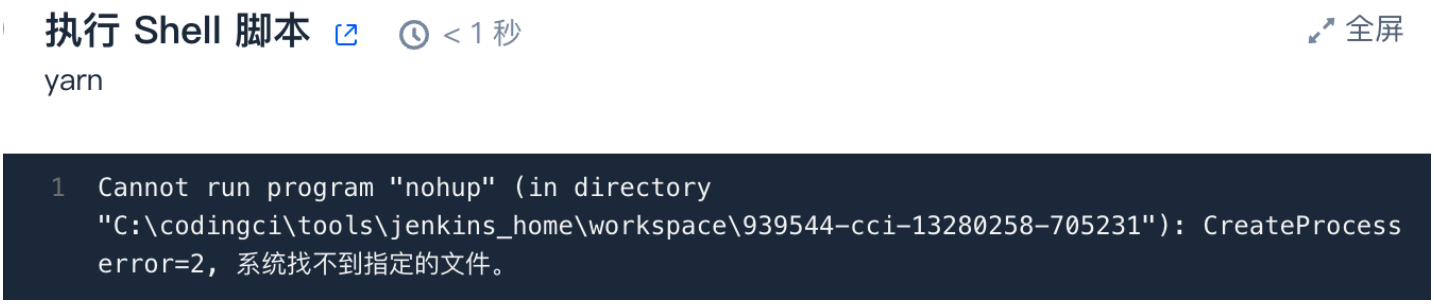
删除 CD_PERSONAL_ACCESS_TOKEN 变量，然后再次编辑插件生成新的个人令牌，保存即可。

CredentialId could not be found

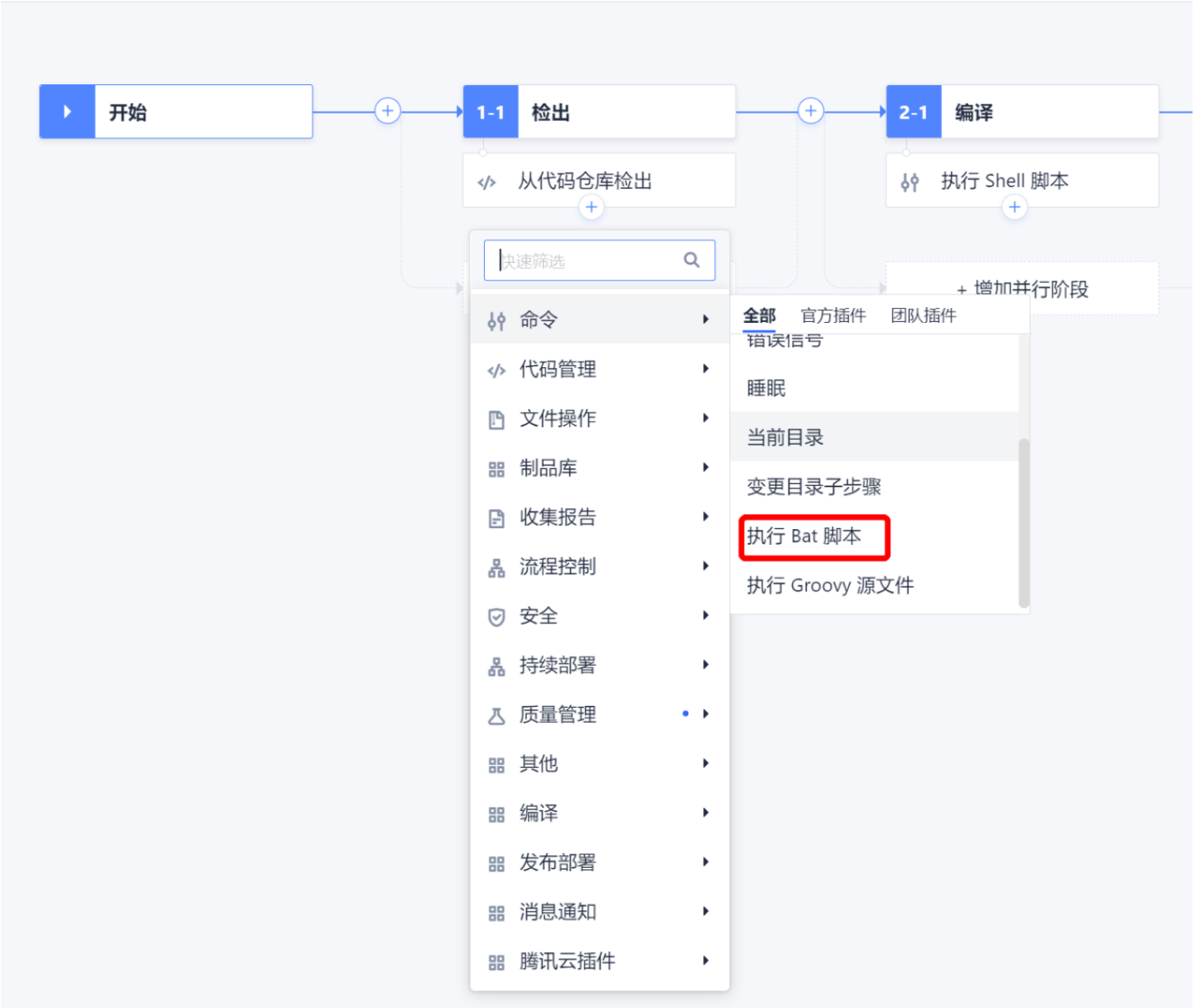
出现此报错的原因为凭据不存在，需清理 cci-agent 服务进程，同时删除主目录下的 .coding 文件夹。删除后重新接入 自定义节点 Worker 服务。

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：



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。

```
pipeline {
  agent any
  stages {
    stage('检出') {
      steps {
        checkout([$class: 'GitSCM',
        branches: [[name: GIT_BUILD_REF]],
        userRemoteConfigs: [[
          url: GIT_REPO_URL,
          credentialsId: CREDENTIALS_ID
        ]]])
      }
    }
    stage('编译') {
      steps {
        sh 'rm /root/.gradle/init.gradle'
        sh './gradlew build --debug'
      }
    }
    stage('构建镜像并推送到 CODING Docker 制品库') {
      steps {
        sh "docker build -t ${CODING_DOCKER_IMAGE_NAME}:${DOCKER_IMAGE_VERSION} -f ${DOCKERFILE_PATH} ${DOCKER_BUILD_CONTEXT}"
        useCustomStepPlugin(key: 'codingcorp:artifact_docker_push', version: 'latest', params: [image:"${CODING_DOCKER_IMAGE_NAME}:${DOCKER_IMAGE_VERSION}",repo:"${DOCKER_REPO_NAME}",properties:'[]'])
      }
    }
  }
  environment {
    CODING_DOCKER_REG_HOST = "${CCI_CURRENT_TEAM}-docker.pkg.${CCI_CURRENT_DOMAIN}"
    CODING_DOCKER_IMAGE_NAME = "${PROJECT_NAME.toLowerCase()}/${DOCKER_REPO_NAME}/${DOCKER_IMAGE_NAME}"
  }
}
```

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

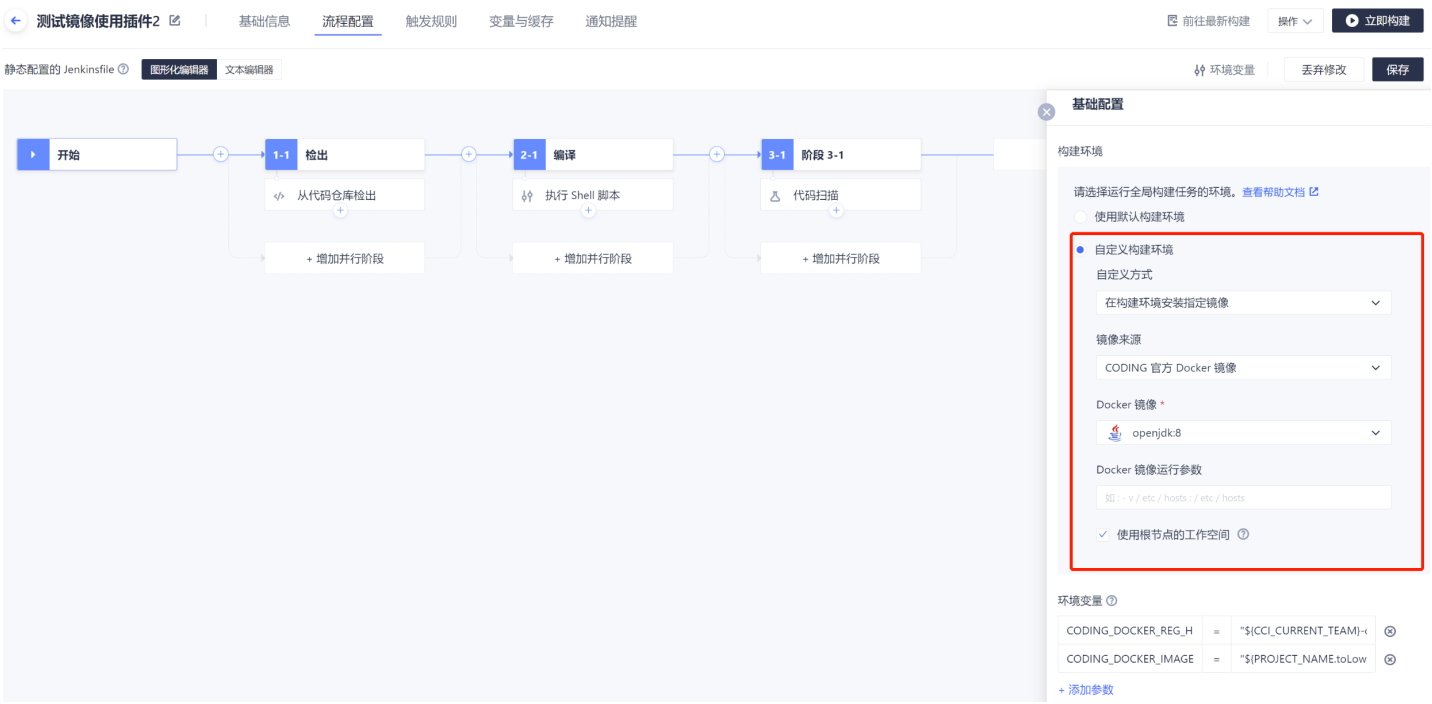
在接入自定义节点时，若节点中的 yum 包管理工具版本过低则有可能出现此错误码。在自定义节点中执行 yum --version 2 命令查看是否能够正常返回版本号。若版本过低则需执行 yum update 命令升级版本。

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

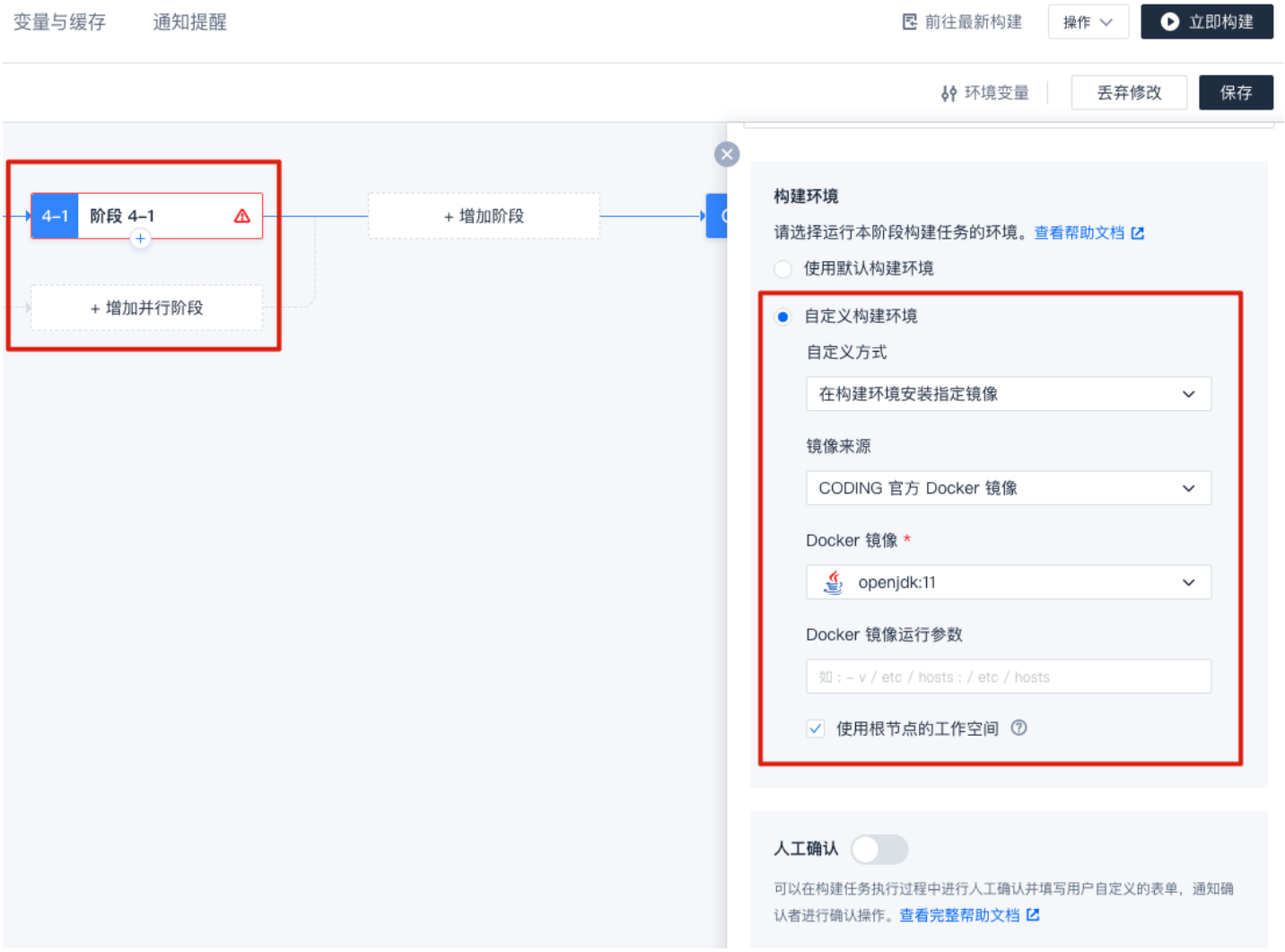
运行持续集成任务失败后返回此错误码：

```
[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
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。



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。



docker: not found

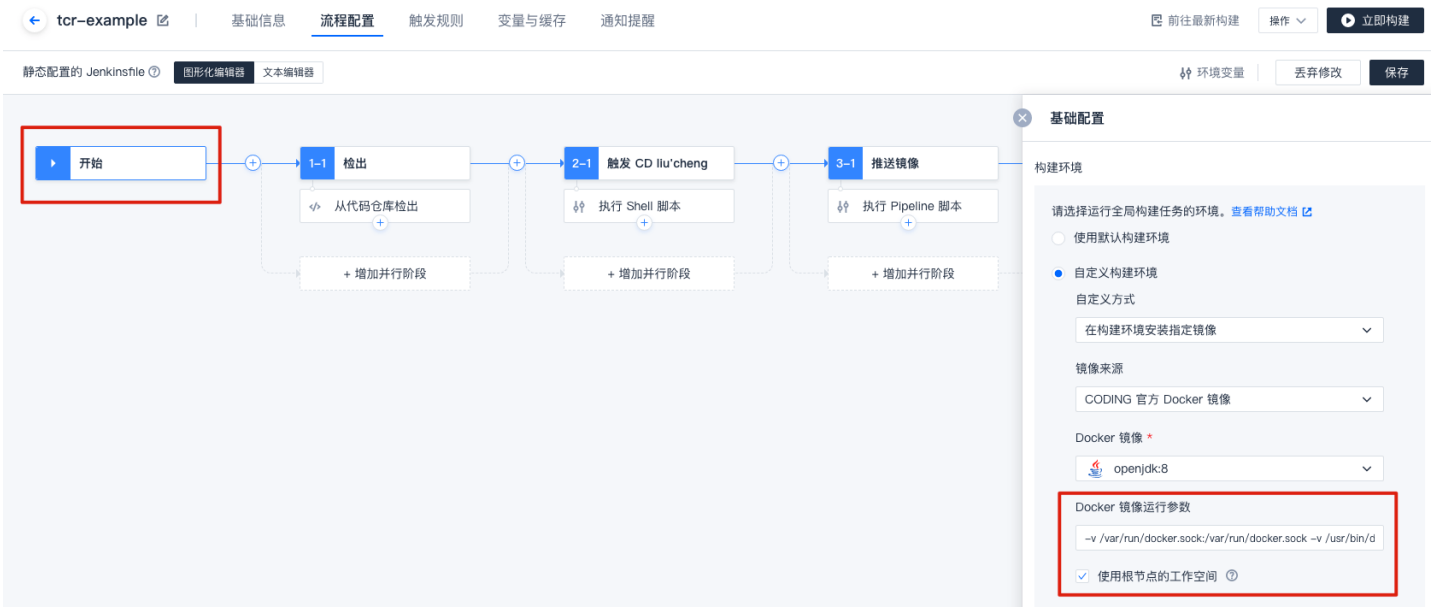
使用持续集成推送镜像时出现下图报错：

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

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
```



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 。

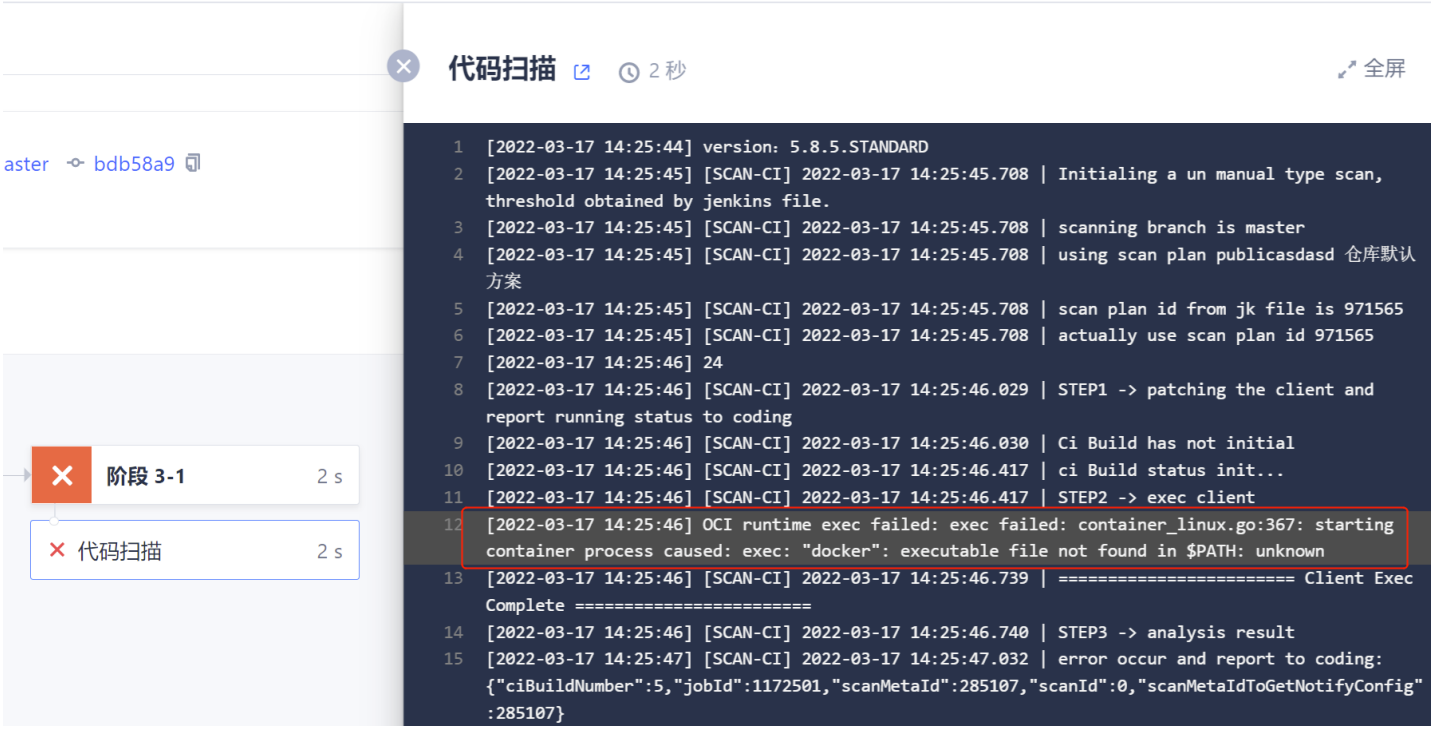
解决办法：

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

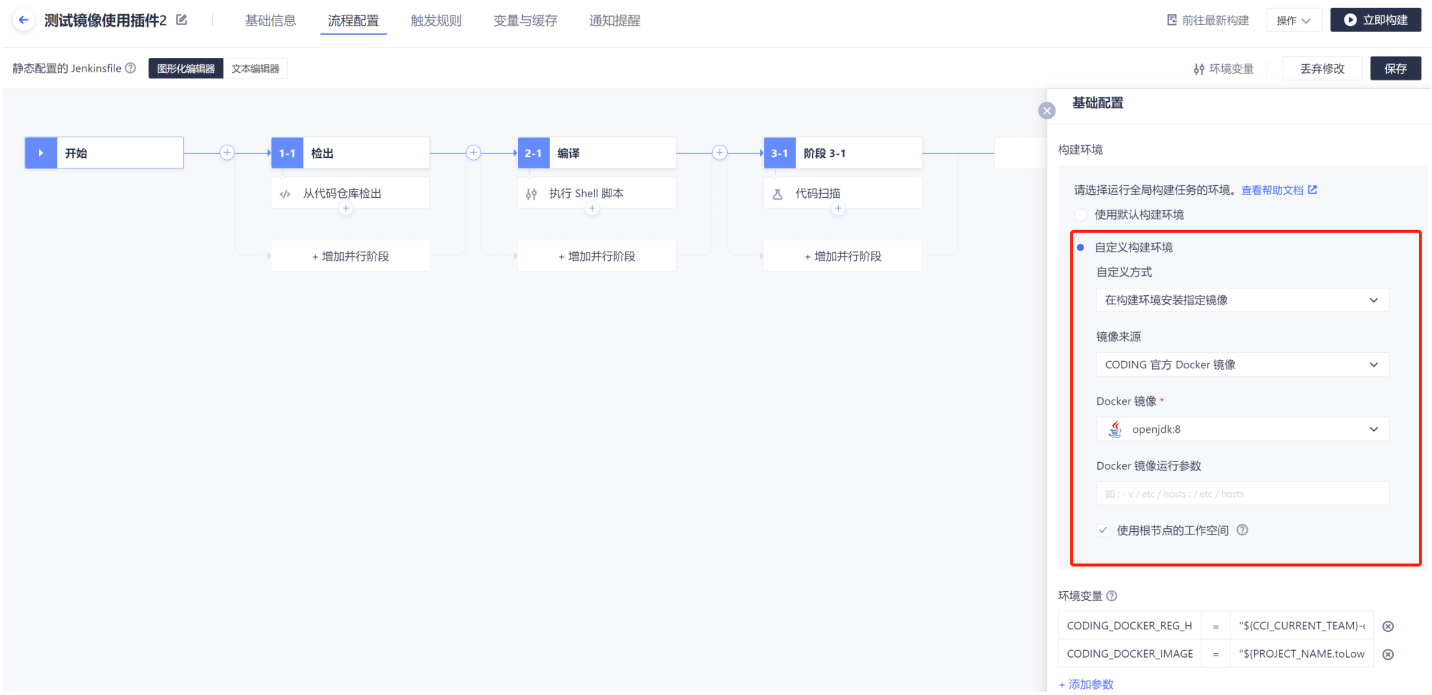
问题描述：

使用持续集成中的代码扫描插件时出现此错误，截图如下：



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。



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：

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。