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

最近更新时间：2023-12-29 11:44:51

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.
                ])
            }
        }
    }
}
```

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。

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。

在创建构建计划时选择使用代码仓库中的 Jenkinsfile 与静态配置的 Jenkinsfile 有什么区别？

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。

构建执行相关问题

最近更新时间：2024-05-28 10:15:10

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。

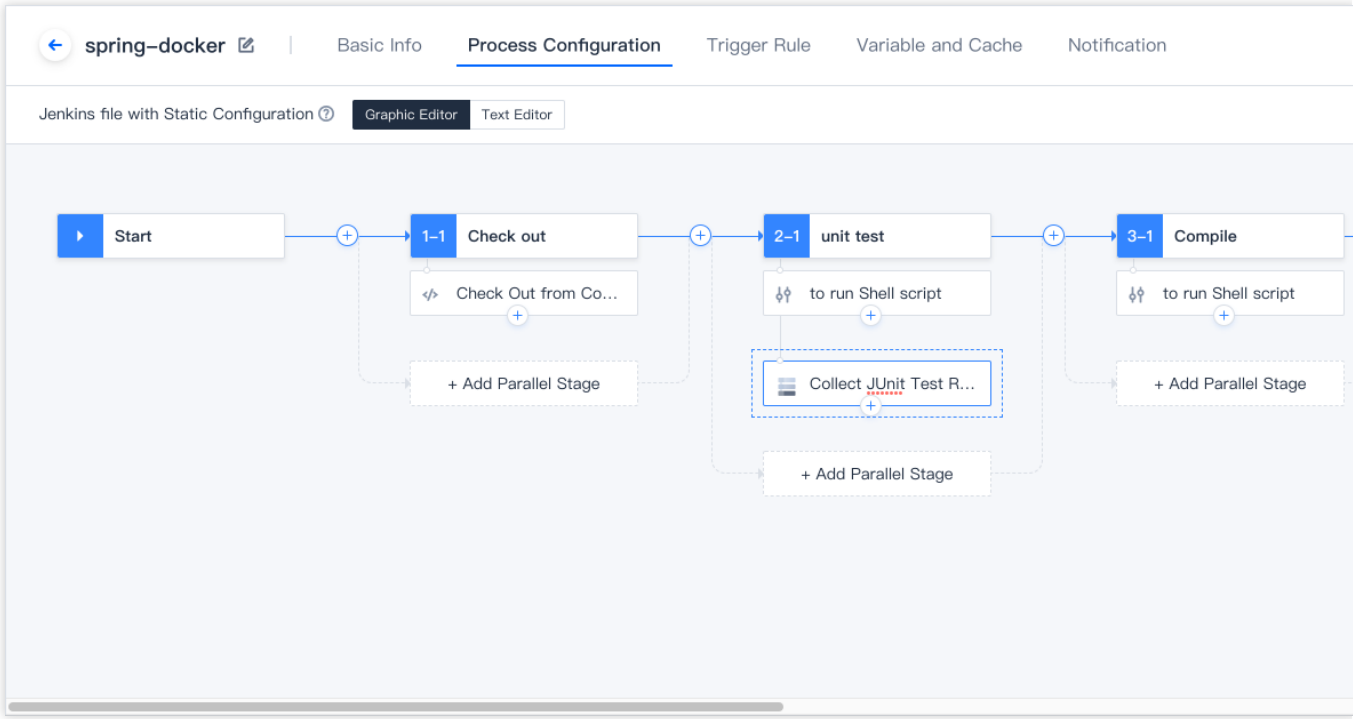
构建超时或构建配额不足如何处理？

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。

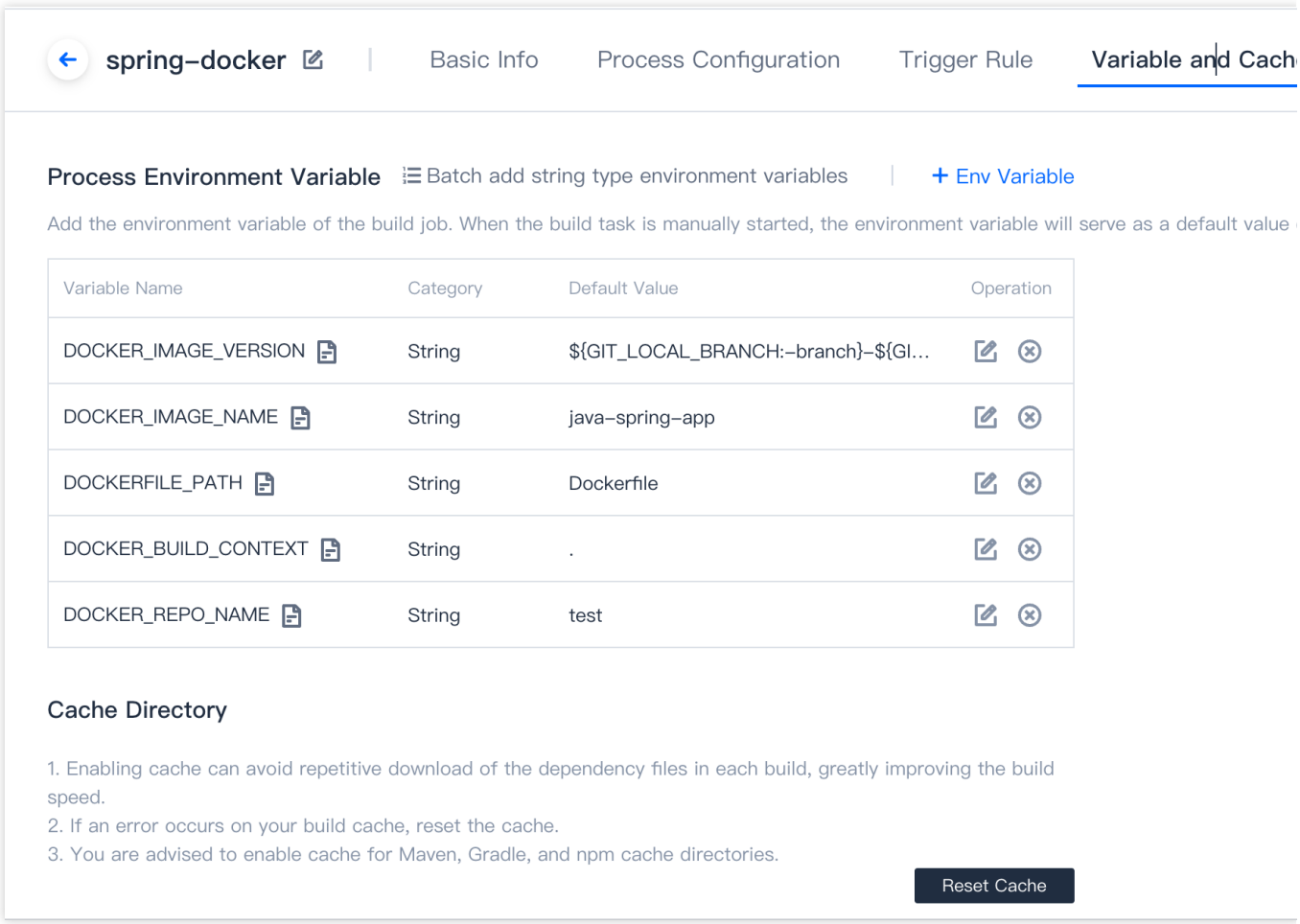
如何查看构建日志与构建快照？

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


spring-docker
Basic Info
Process Configuration
Trigger Rule
Variable and Cache
Notification
Jenkins file with Static Configuration
Graphic Editor
Text Editor
Start
1-1
Check out
Check Out from Co...
+ Add Parallel Stage
2-1
unit test
to run Shell script
Collect JUnit Test R...
+ Add Parallel Stage
3-1
Compile
to run Shell script
+ Add Parallel Stage

构建快照



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（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）

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。

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'
}
```

持续集成与代码仓库相关

最近更新时间：2023-12-29 11:44:51

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，您可以参考如下示例。

```
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.CREDENTI
      }
    }
    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 PROJEC
            // 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？

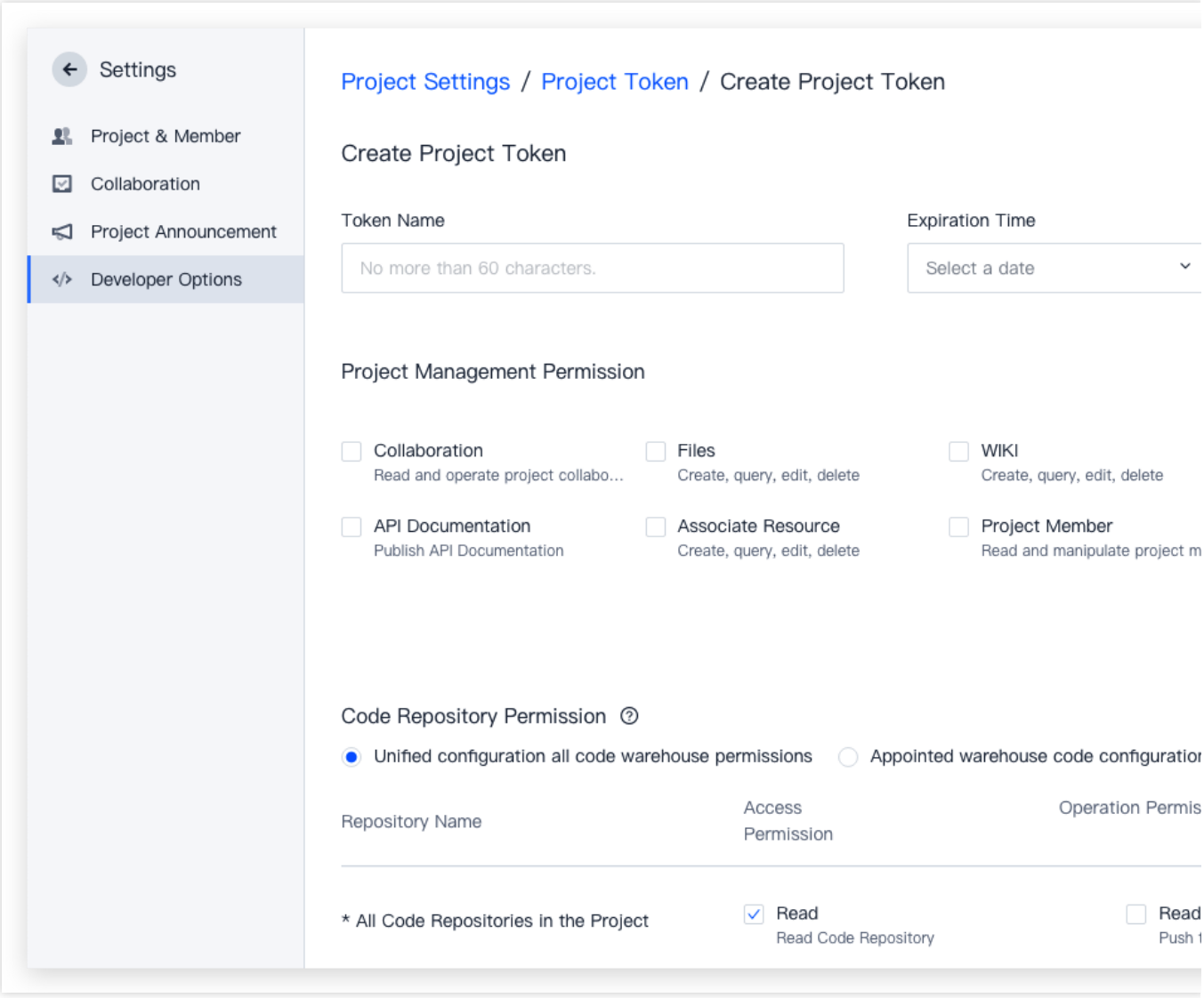
在默认的持续集成计划的配置过程中，所运行的代码源默认是 Git 类型仓库。若希望使用 SVN 仓库 运行持续集成，下文给出了指引。

前提条件

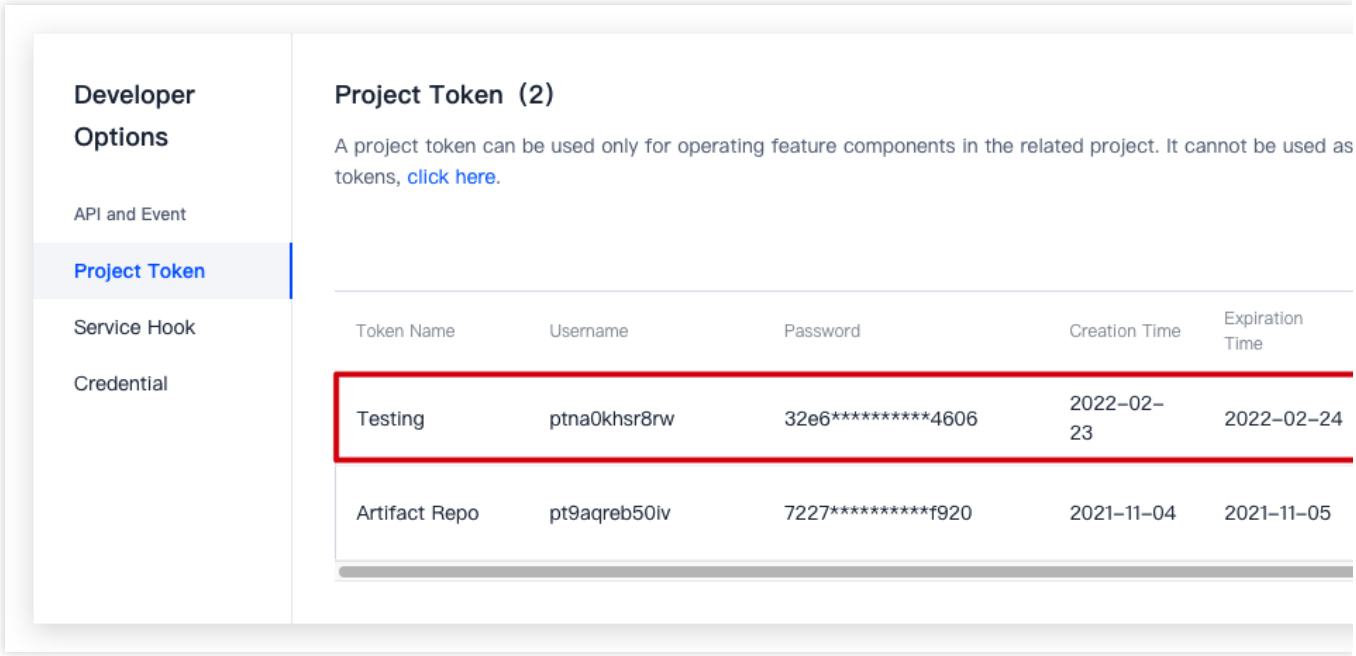
在开始之前，请先创建项目令牌与申请用户名+密码凭据。

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。

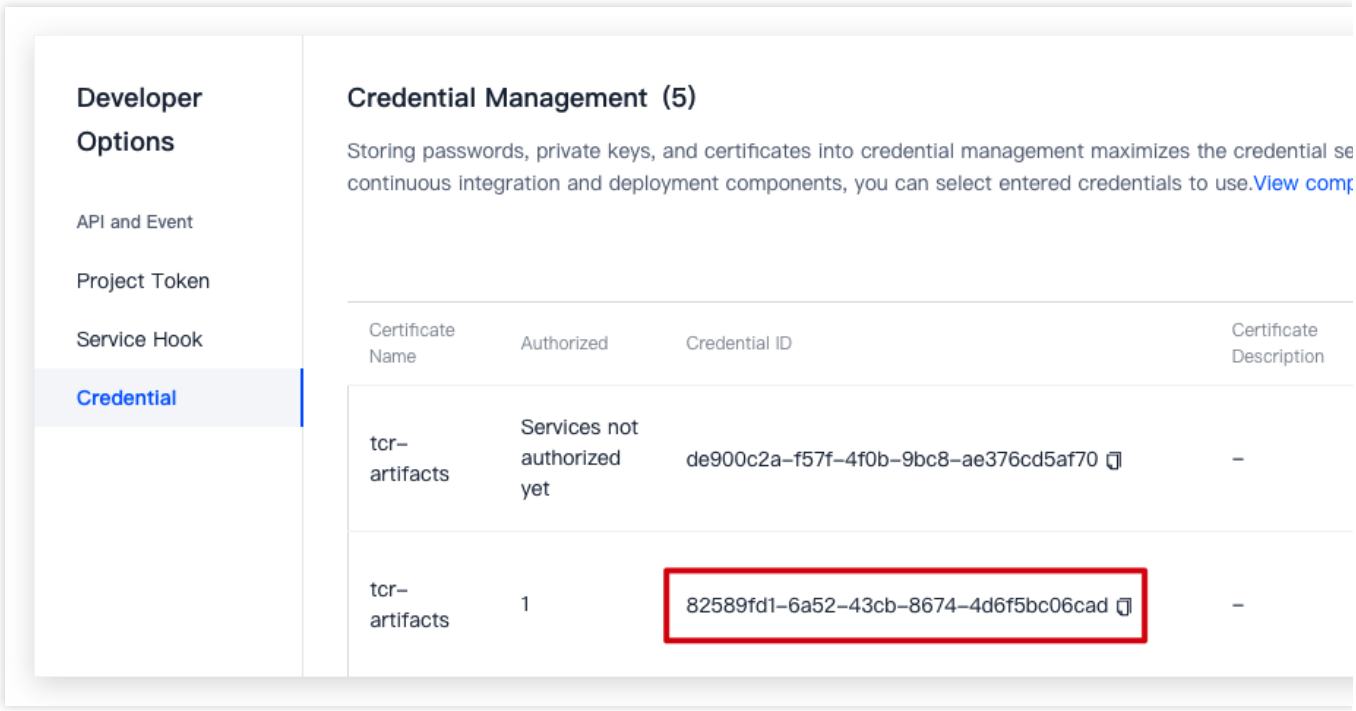


2. 创建完成后会给出用户名及密码。



步骤2：申请用户名和密码凭据

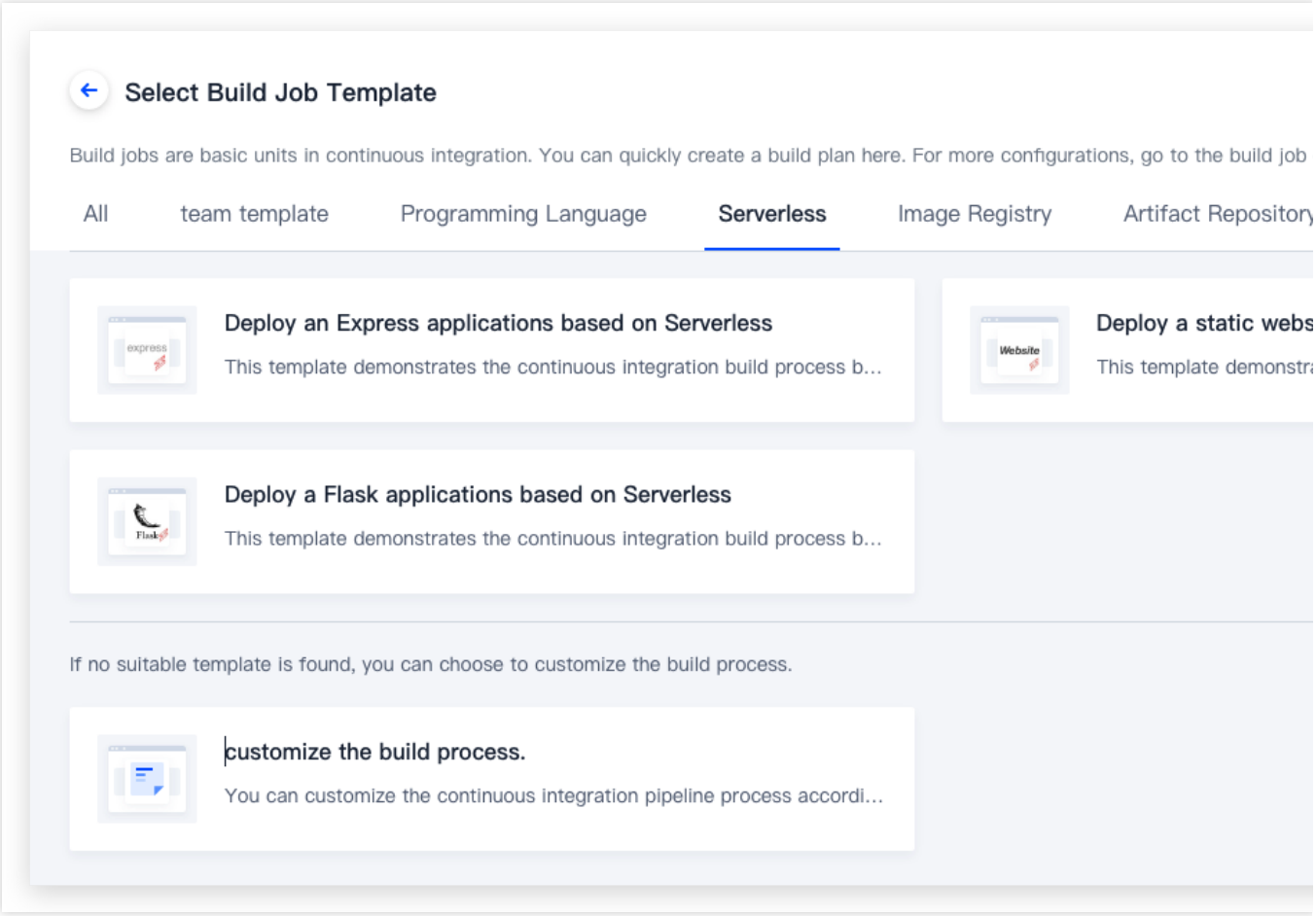
前往**项目设置 > 开发者选项 > 凭据管理**页面，单击**录入凭据**录入用户名和密码凭据。用户名和密码需要填写在创建项目令牌时给出的用户名及密码。



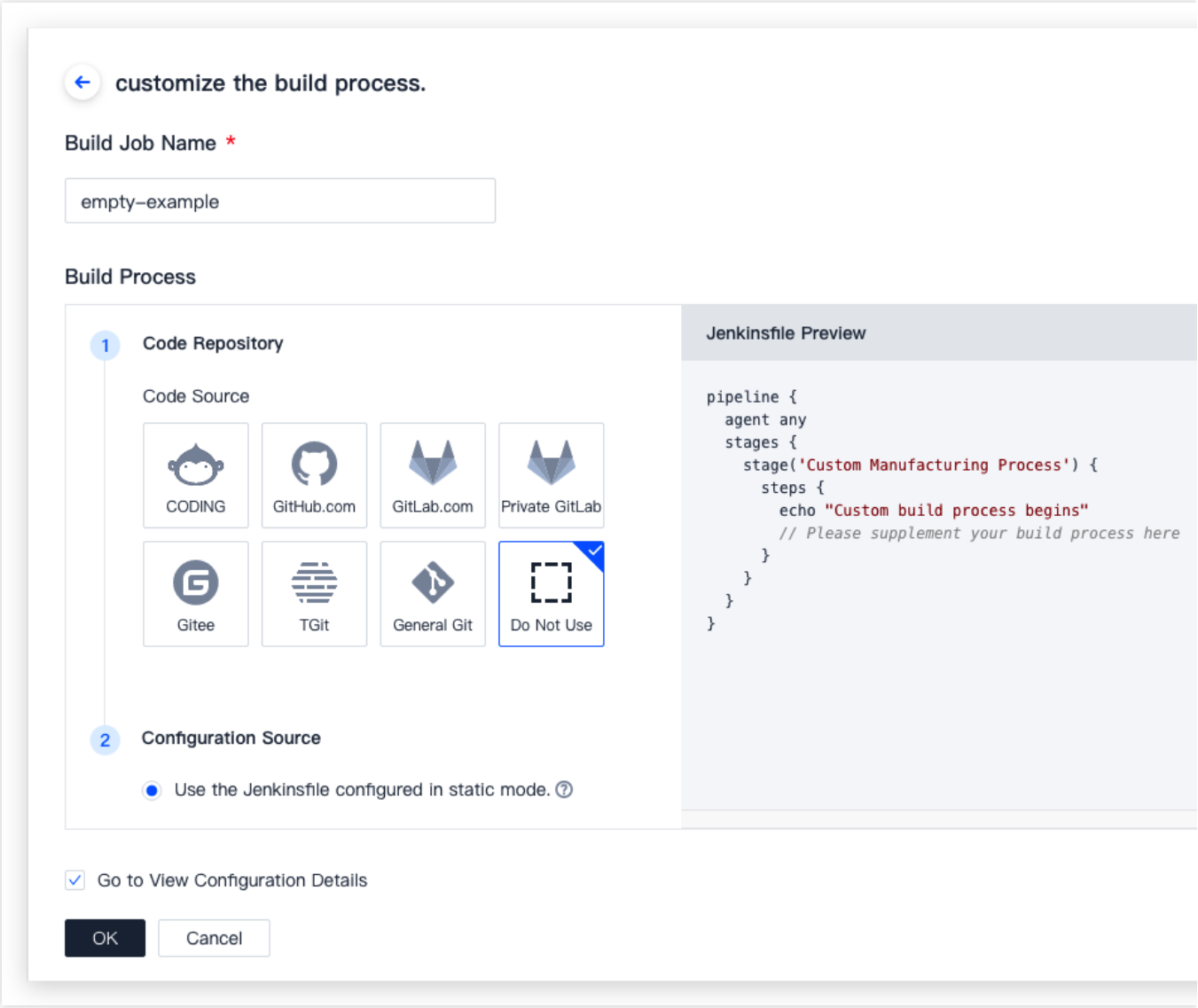
创建完成后会给出凭据 ID，稍后需要将此 ID 录入至构建计划的流程配置中。

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**。



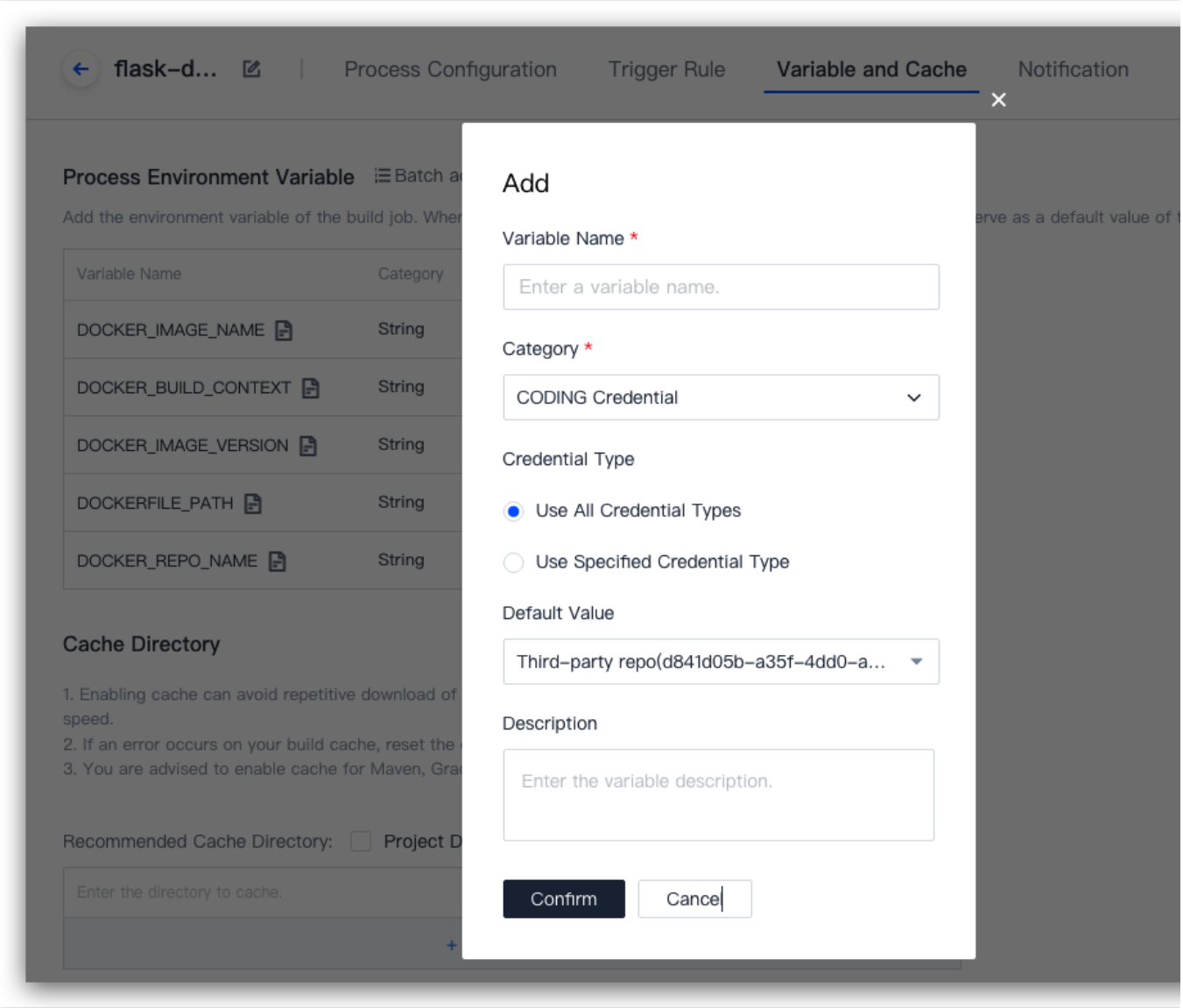
3. 完成后在流程配置中填写相应的配置。

```
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/sv
                    workspaceUpdater: [$class: 'UpdateUpdater']])
        }
    }
    }
}
```

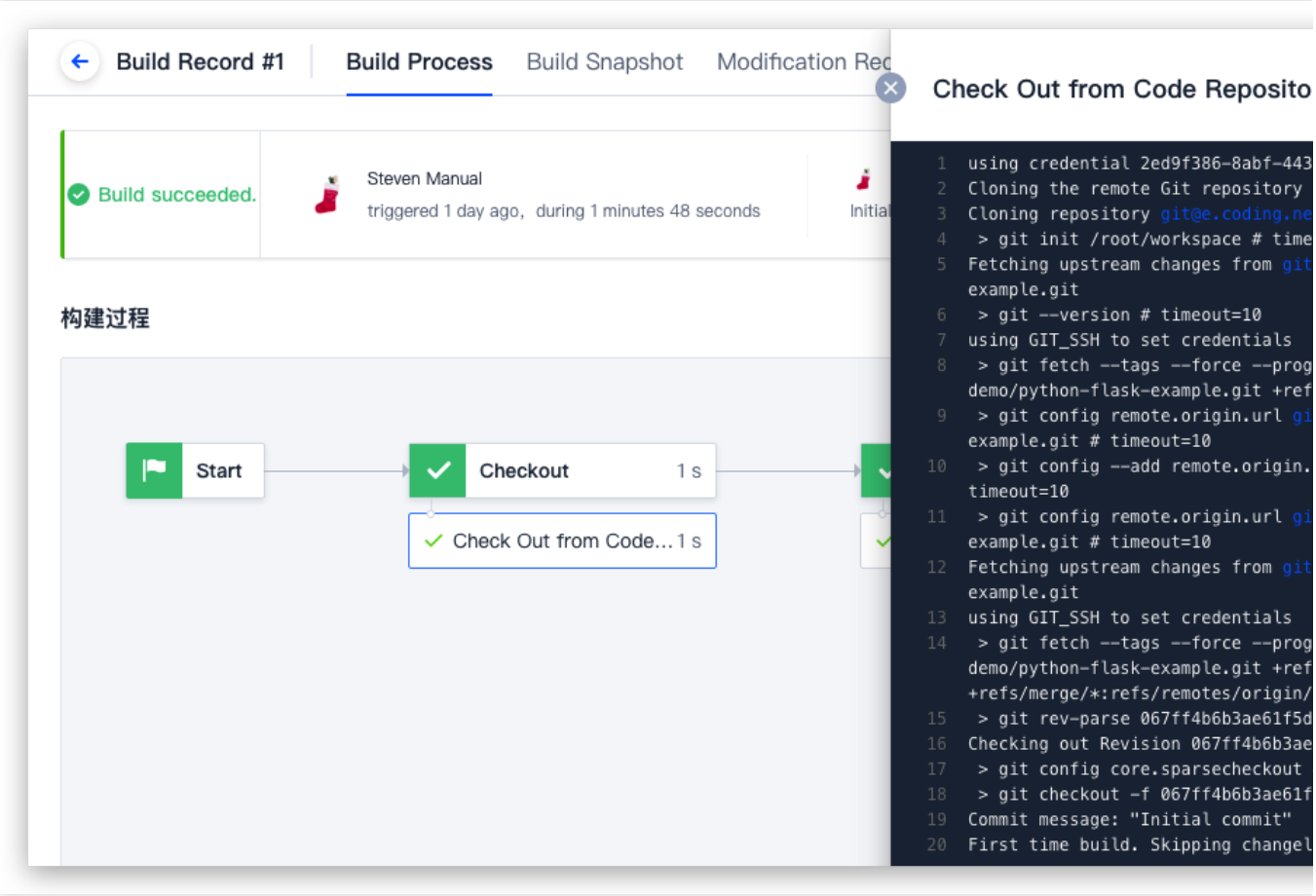
步骤4：添加环境变量

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

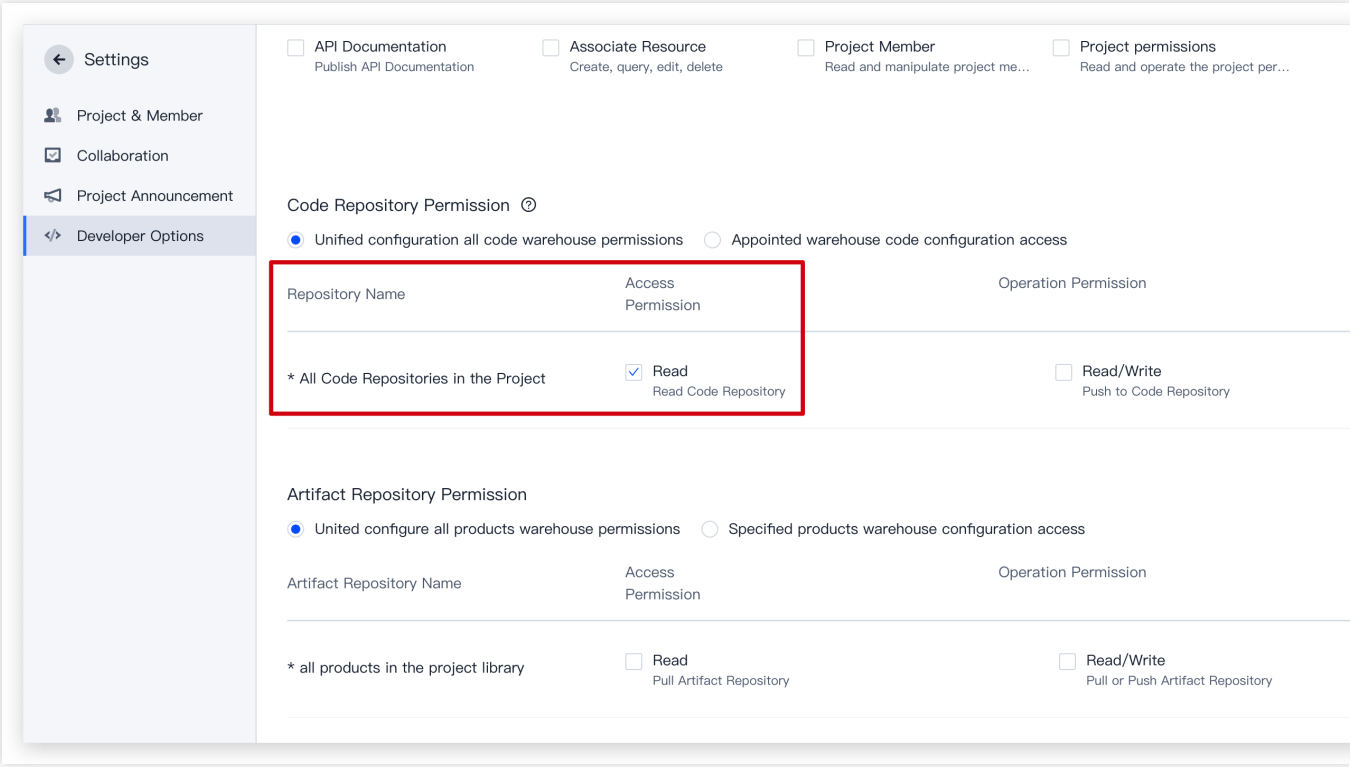
您可以选择手动构建或配置触发方式进行自动构建，构建成功后如图所示。



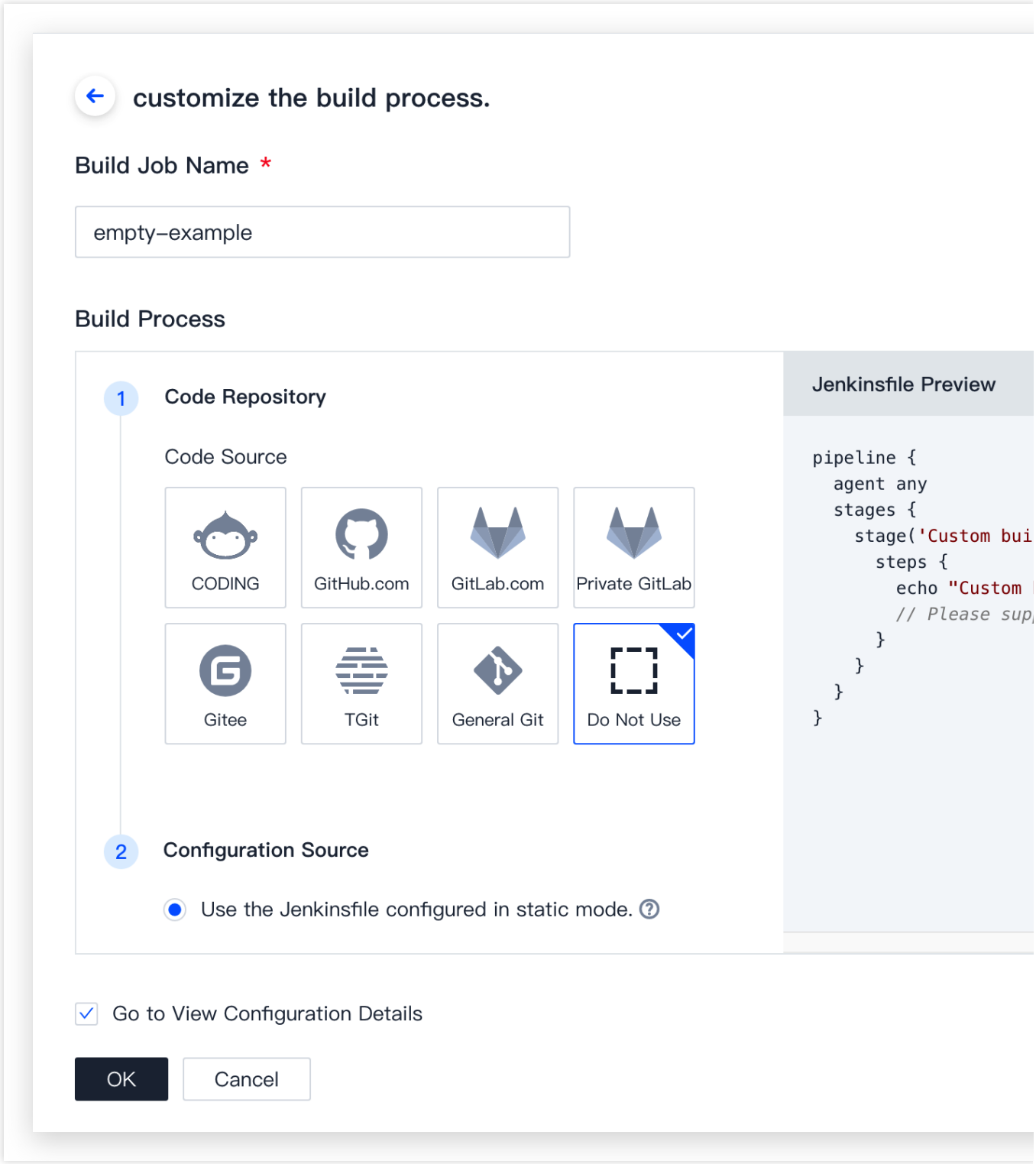
如何拉取多仓库？

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。

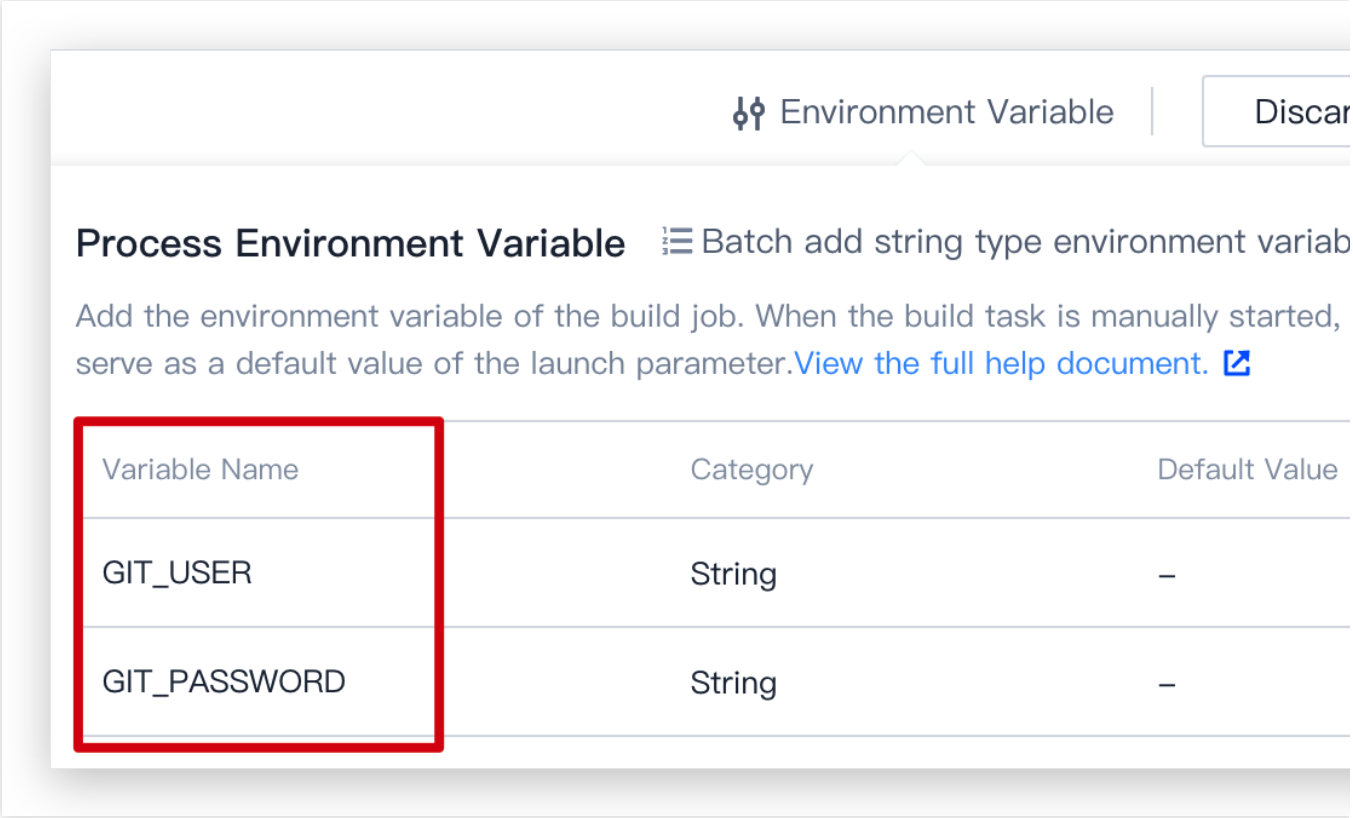


3. 编写 Jenkinsfile 配置文件，填写需拉取的代码仓库地址。

```
pipeline {
    agent any
    stages {
    stage('检出1') {
      steps {
        sh 'git clone "https://${GIT_USER}:${GIT_PASSWORD}@e.coding.net/codes-farm/
```

```
        sh 'ls -la'
      }
    }
    stage('检出2') {
      steps {
        sh 'git clone "https://${GIT_USER}:${GIT_PASSWORD}@e.coding.net/codes-farm/
        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。

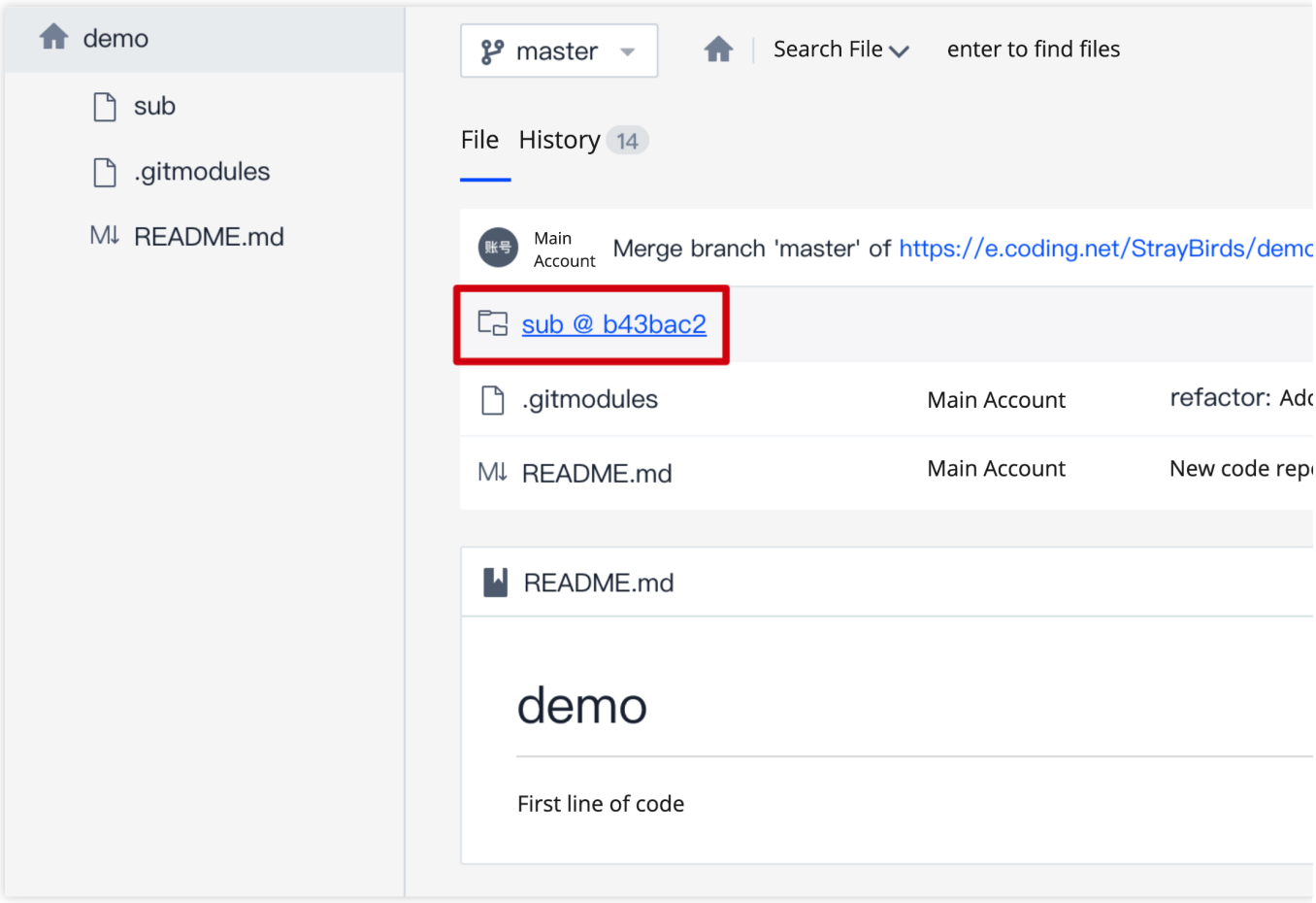


如何检出 Git Submodule 代码？

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
```

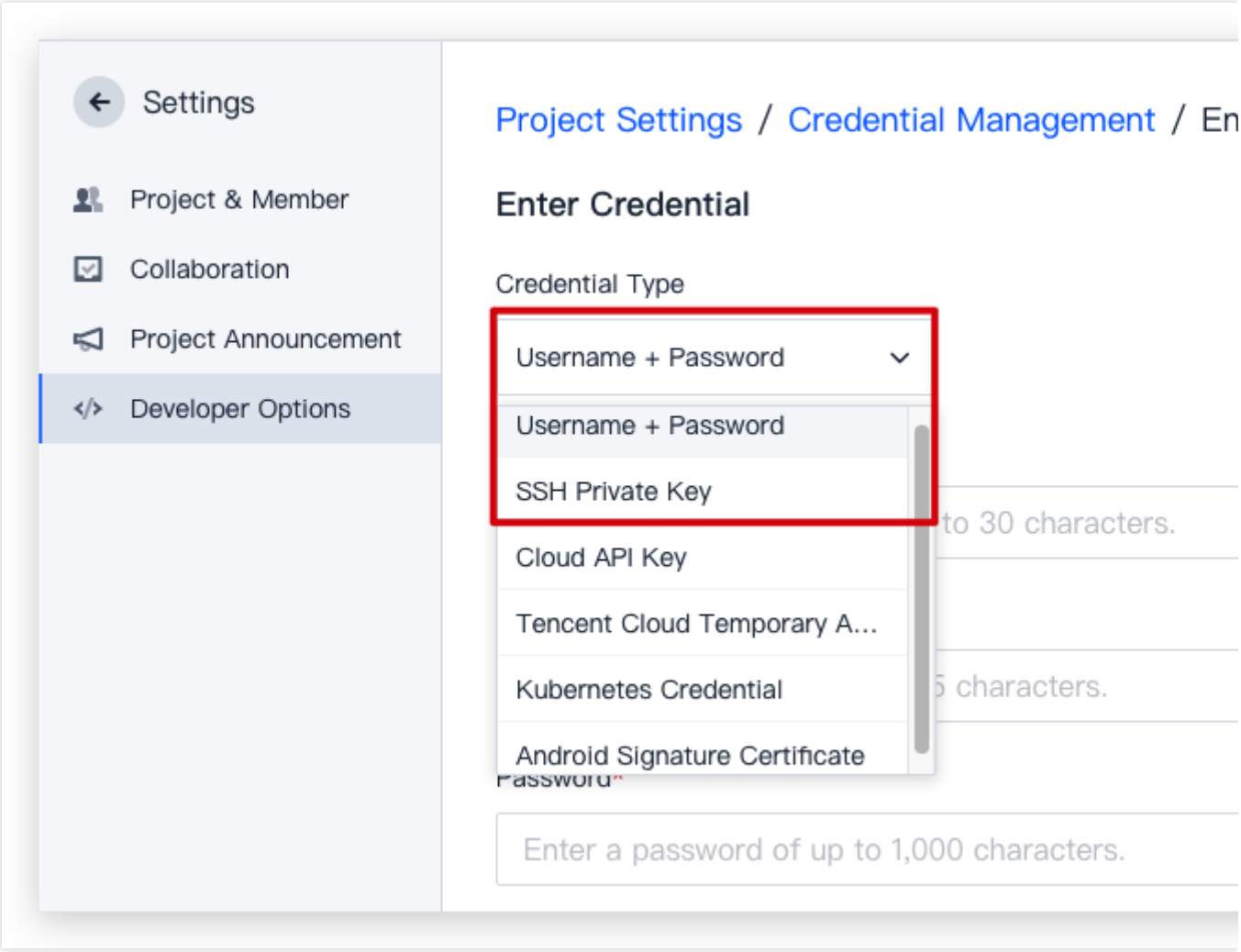
代码提交成功后，在父仓库页将看到此图标：



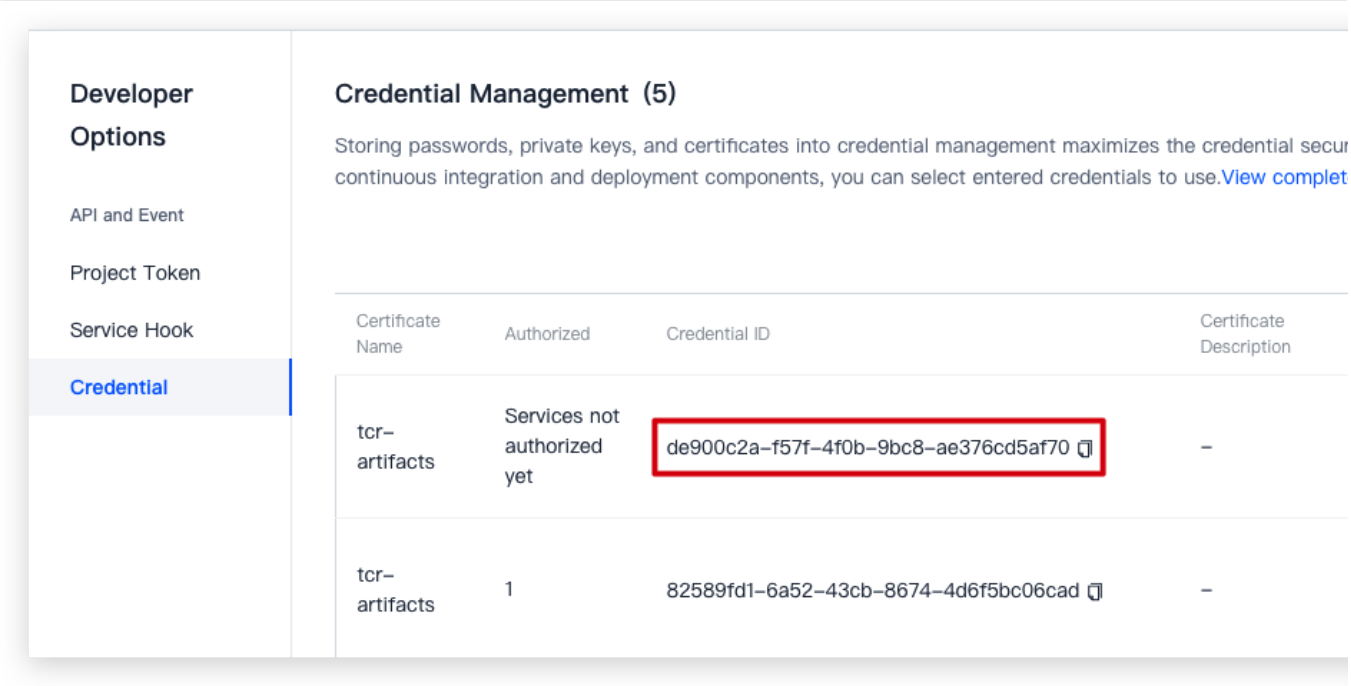
步骤1：录入仓库访问凭据

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。



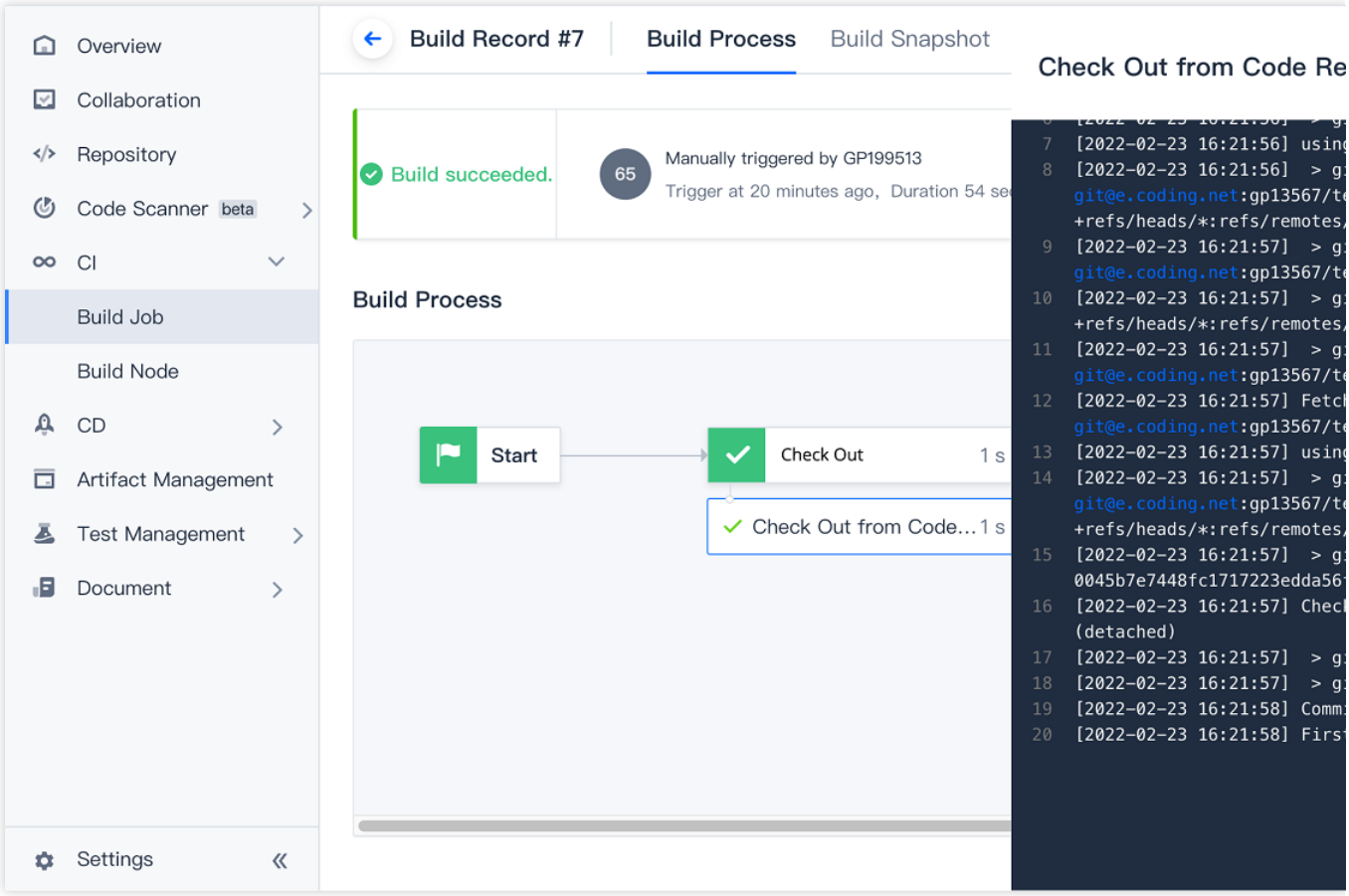
步骤2：配置持续集成流程

参考使用以下 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。

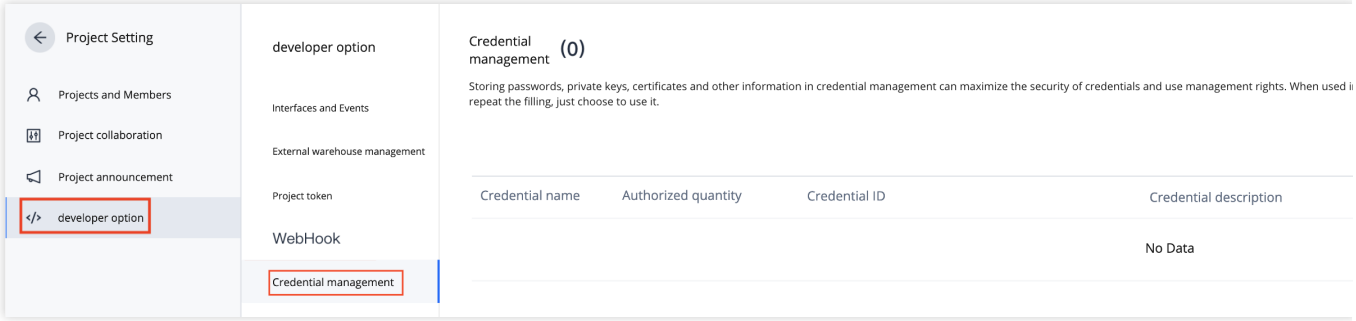
为了方便您区分即将要操作的两个不同项目，我们统一将：

需要被检出的代码仓库所在项目称为 “项目 A”

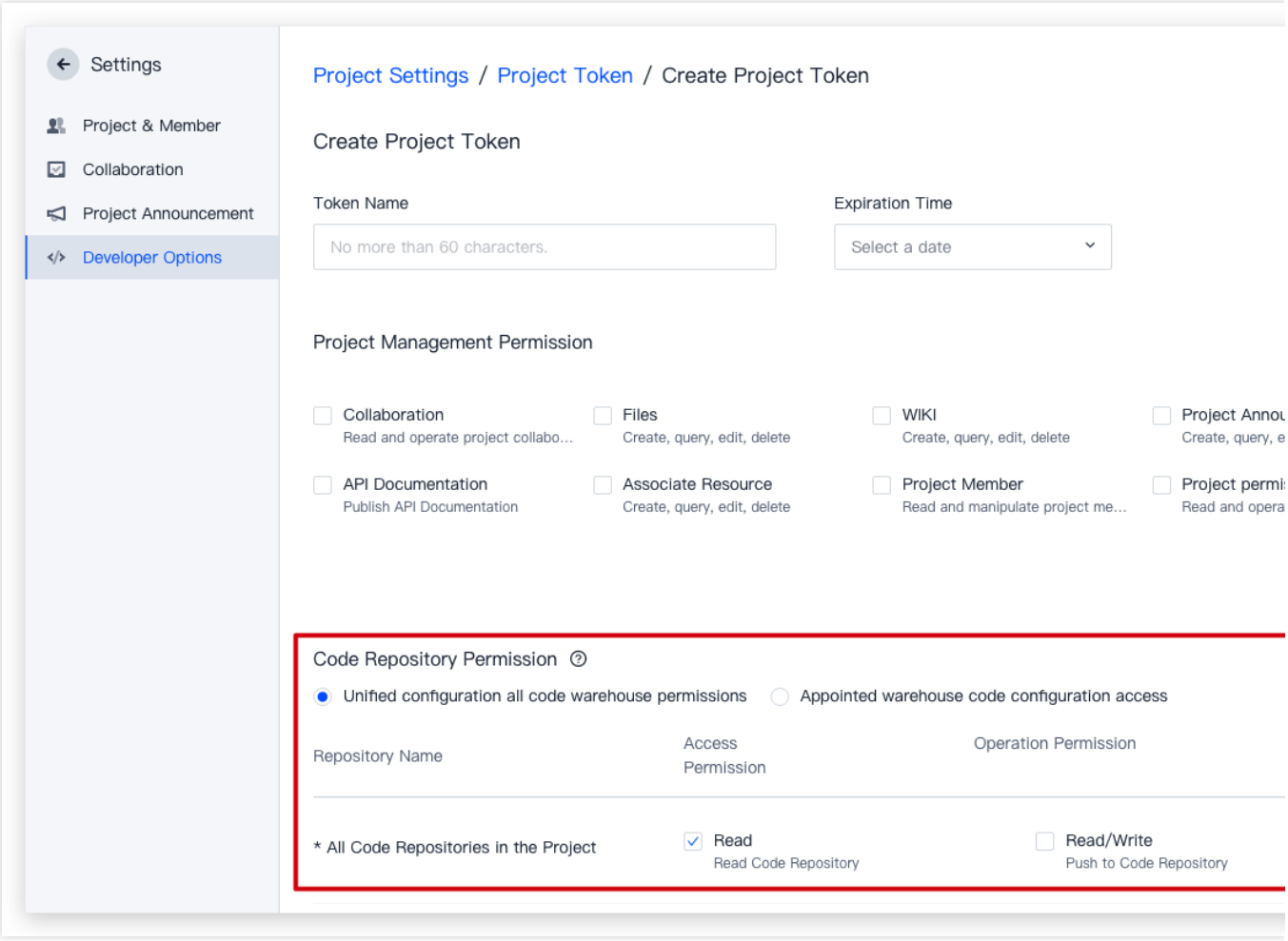
执行检出持续集成任务所在的项目称为 “项目 B”

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** 。



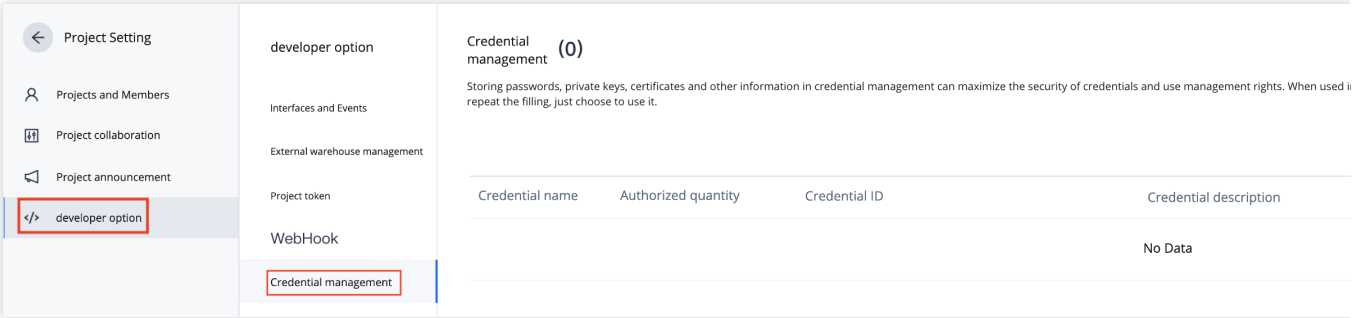
2. 选择需要检出的代码仓库，按需求配置操作权限。



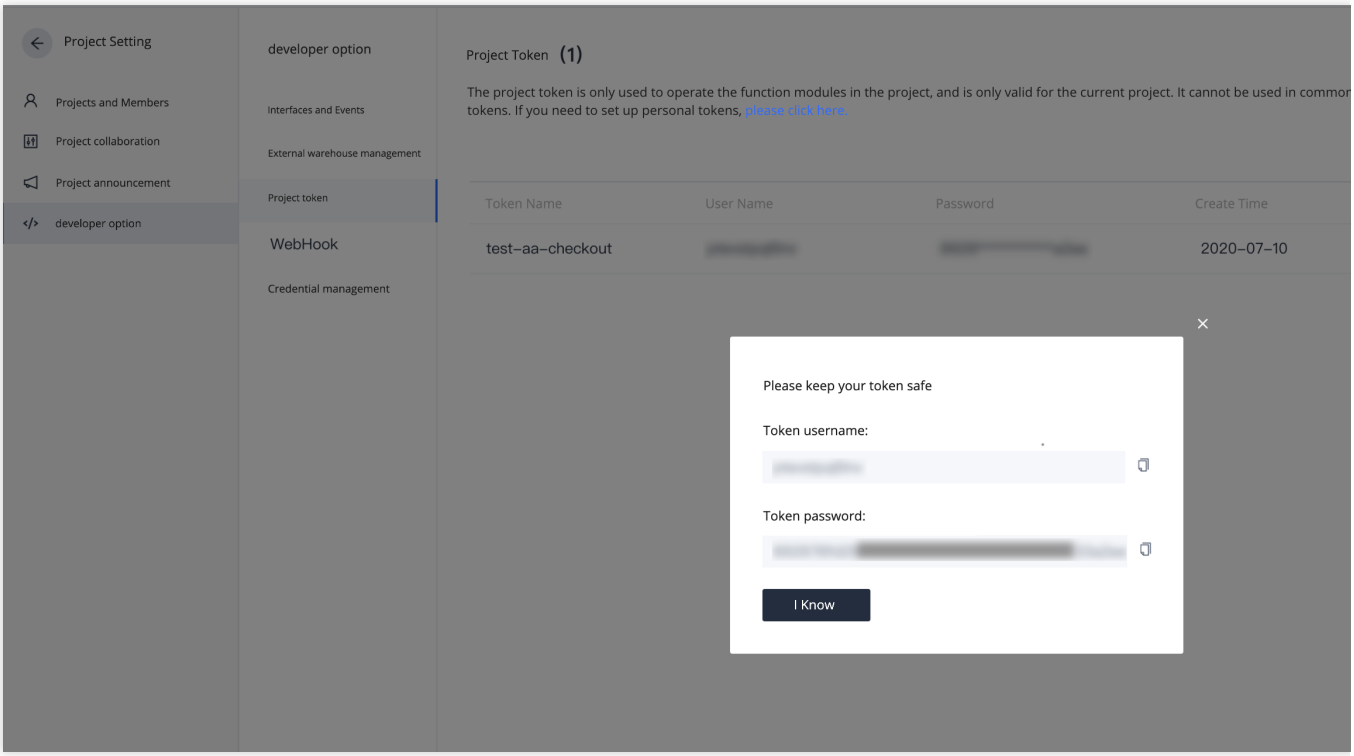
3. 单击**确定**后创建成功。

步骤2：在项目 B 创建凭据

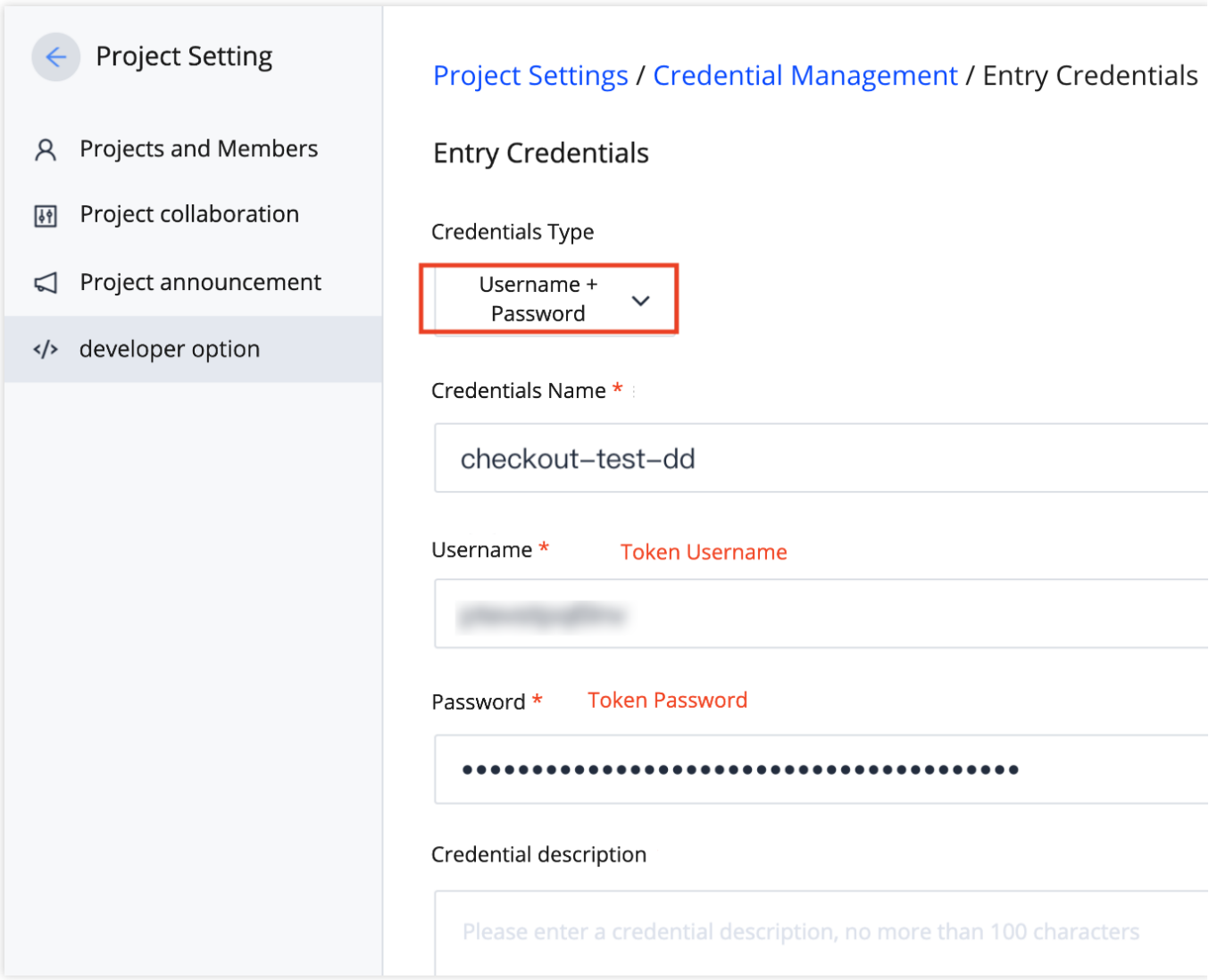
1. 进入项目 B 进入**项目设置 > 开发者选项 > 凭据管理**页面，单击**录入凭据**。



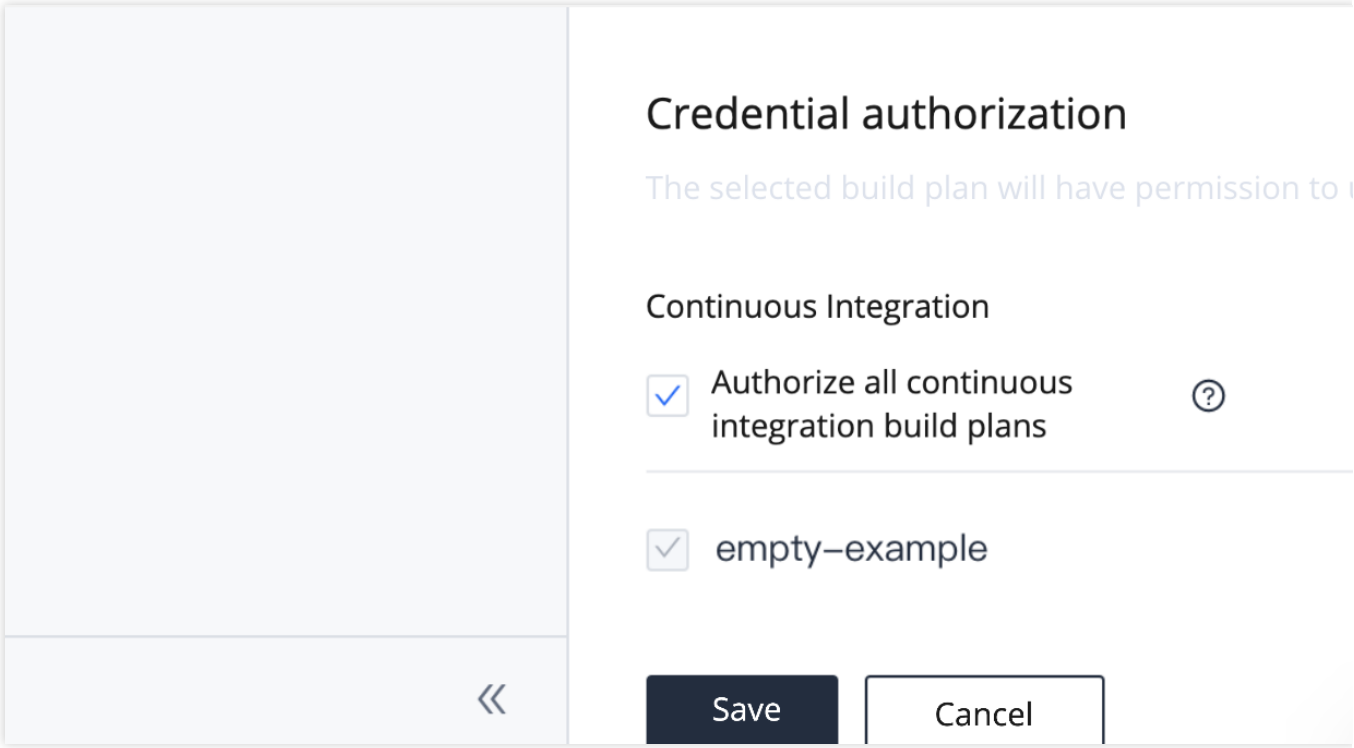
2. 回到之前创建好的项目 A 项目令牌页面，单击**查看密码**。



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。

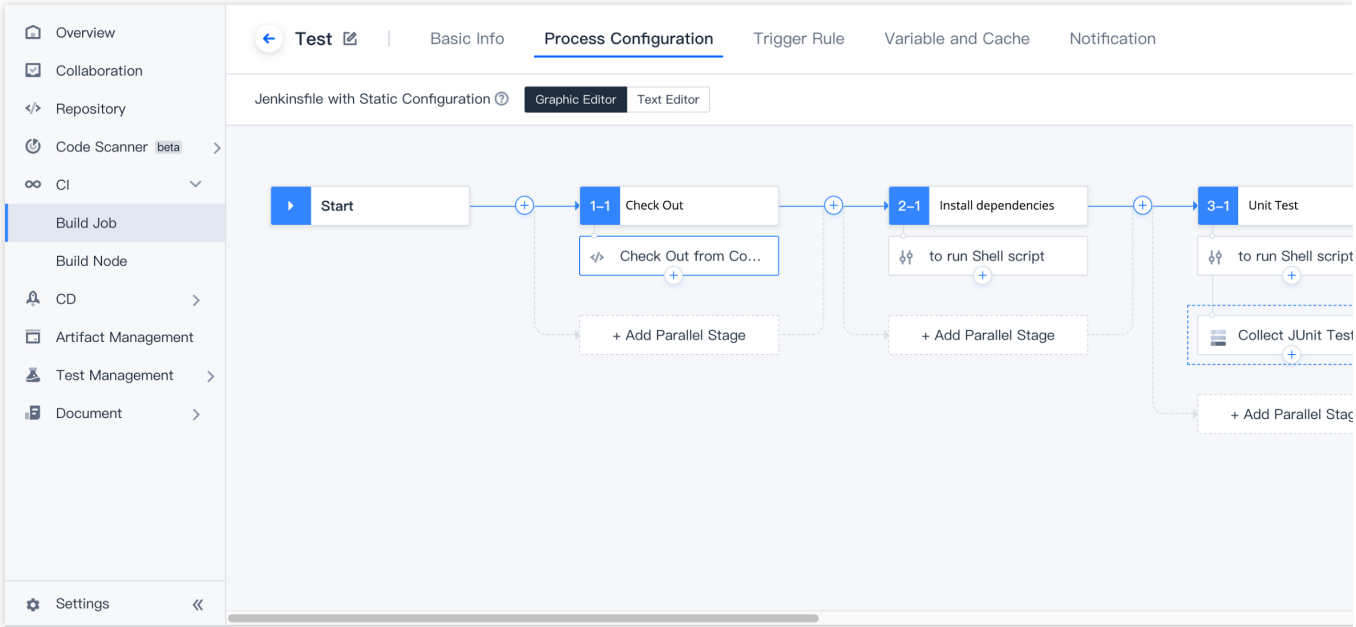


4. 勾选授权的持续集成项目，单击**保存**。

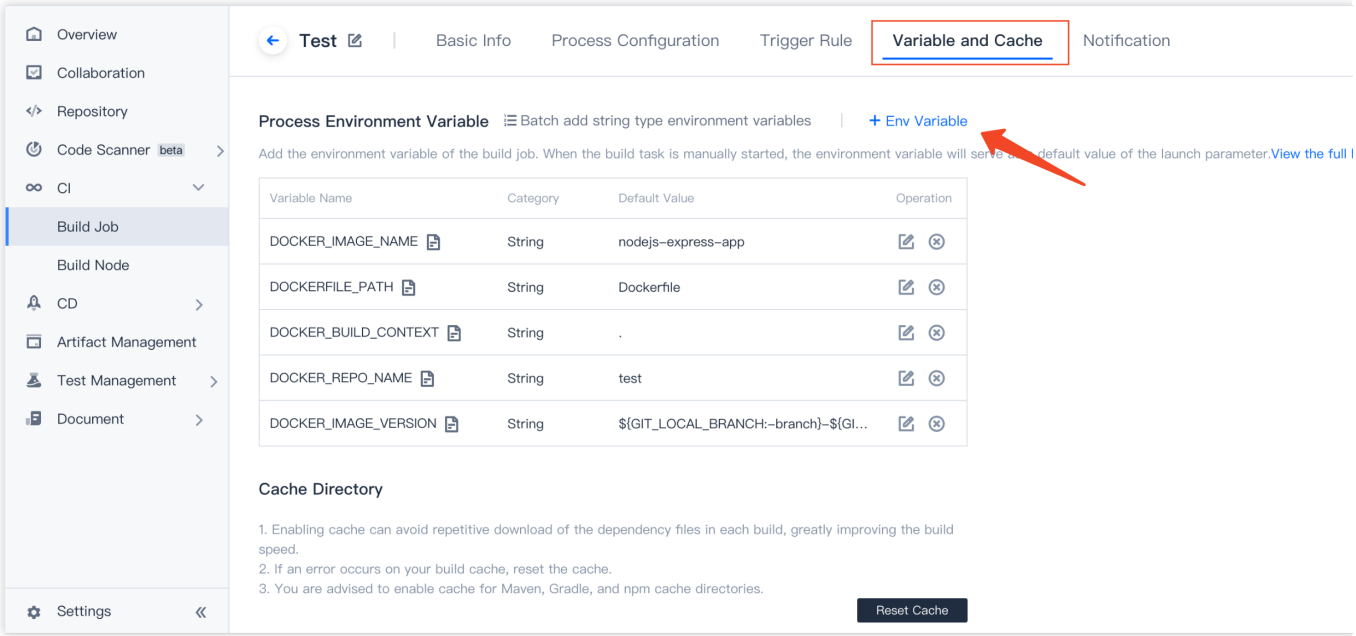


步骤3：在项目 B 持续集成任务中配置对应的环境变量

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**。



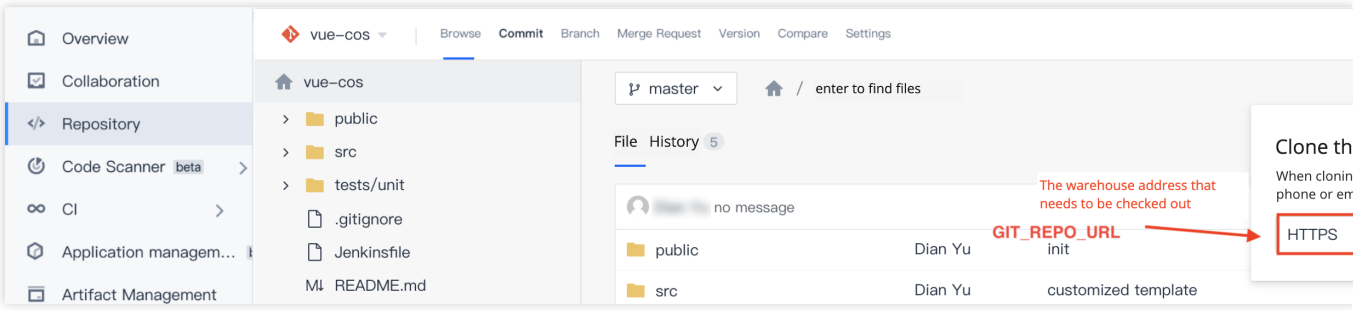
也可以在添加检出流程之后，进入持续集成设置 > **变量与缓存**中单击**添加环境变量**。



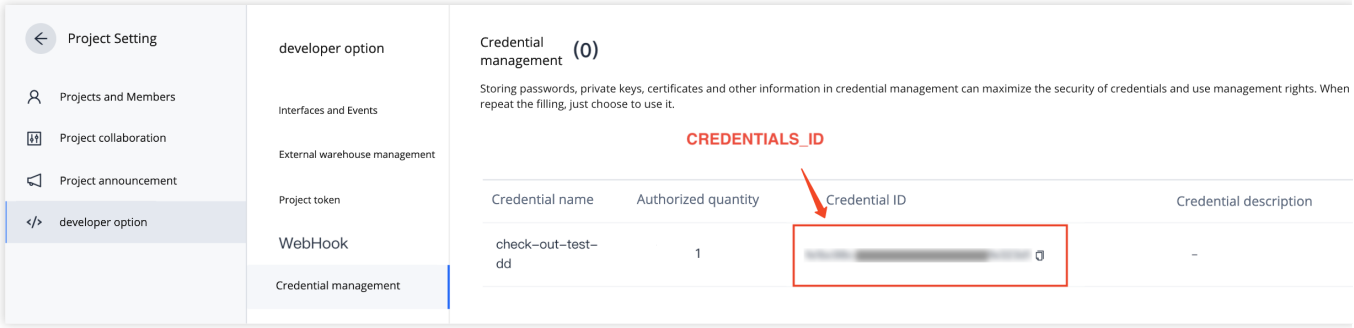
2. 分别添加以下两个环境变量：

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

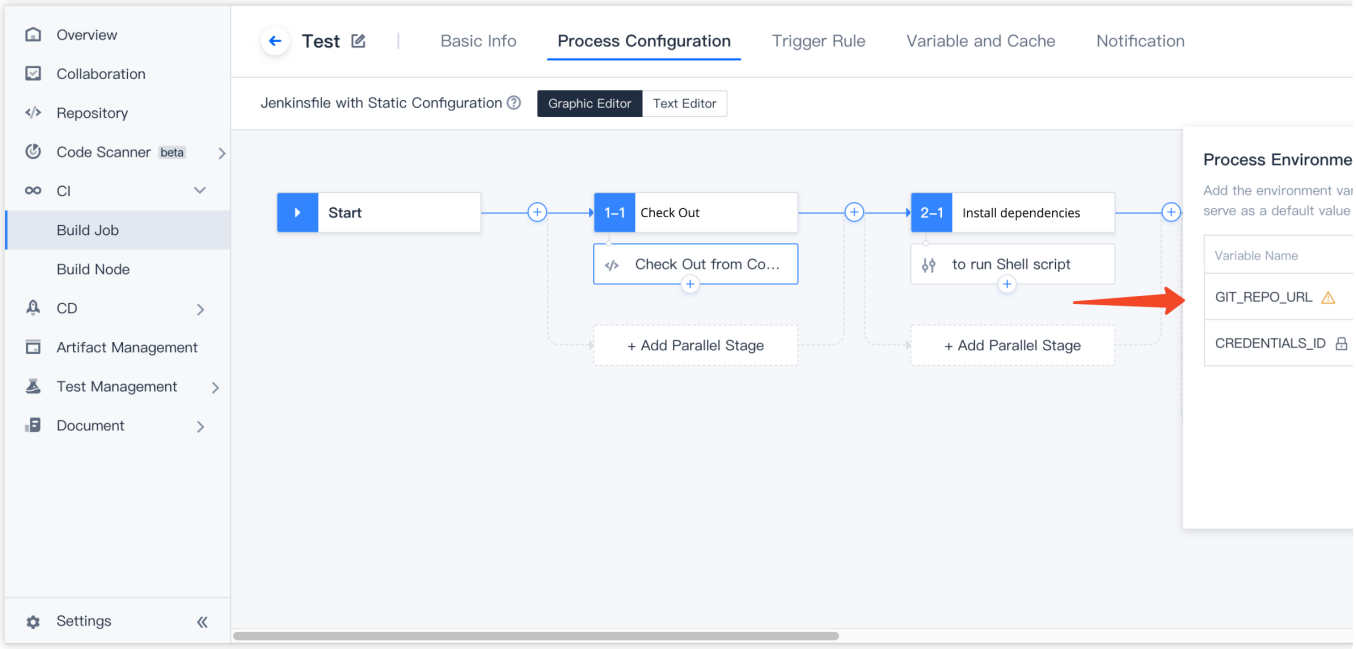
GIT_REPO_URL



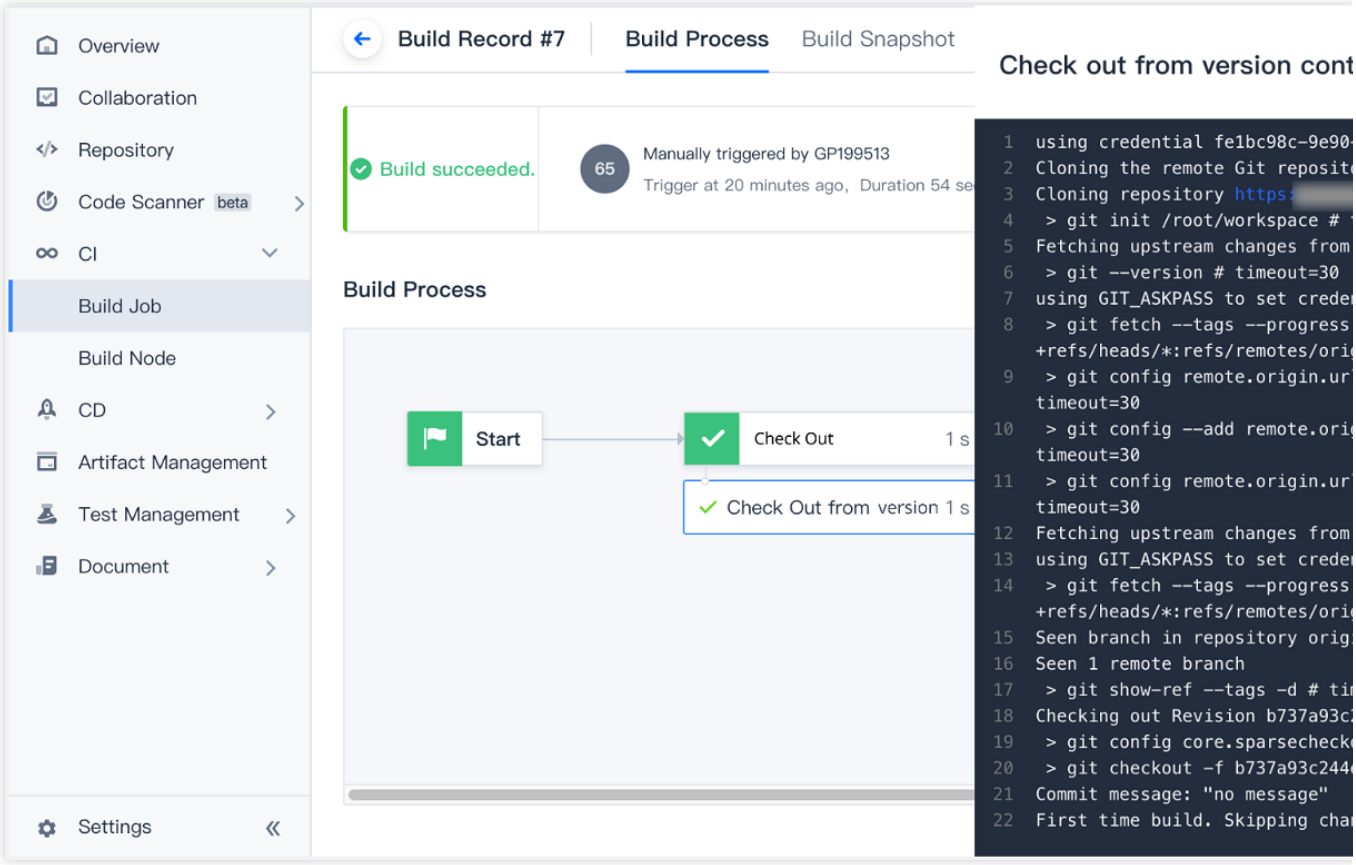
CREDENTIALS_ID



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

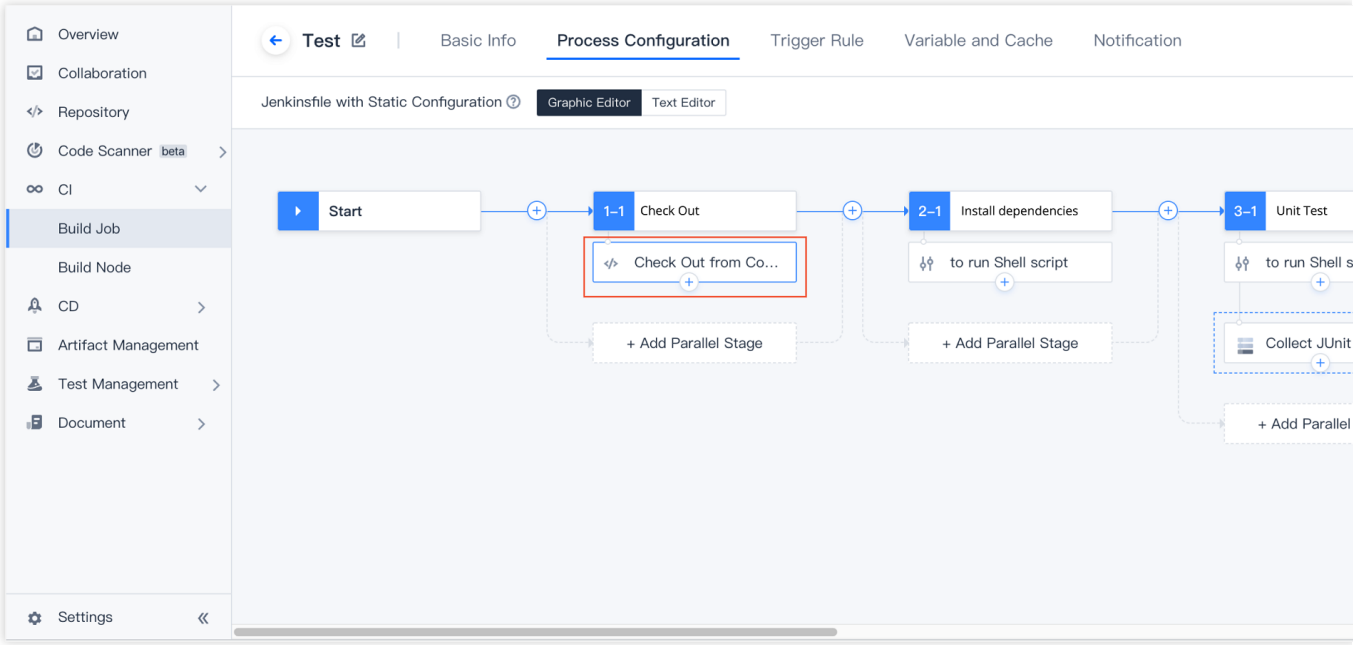
Git LFS 插件加速了带有频繁变动的大文件（例如图片、视频等）的 `git clone` 和 `git fetch` 操作。

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。

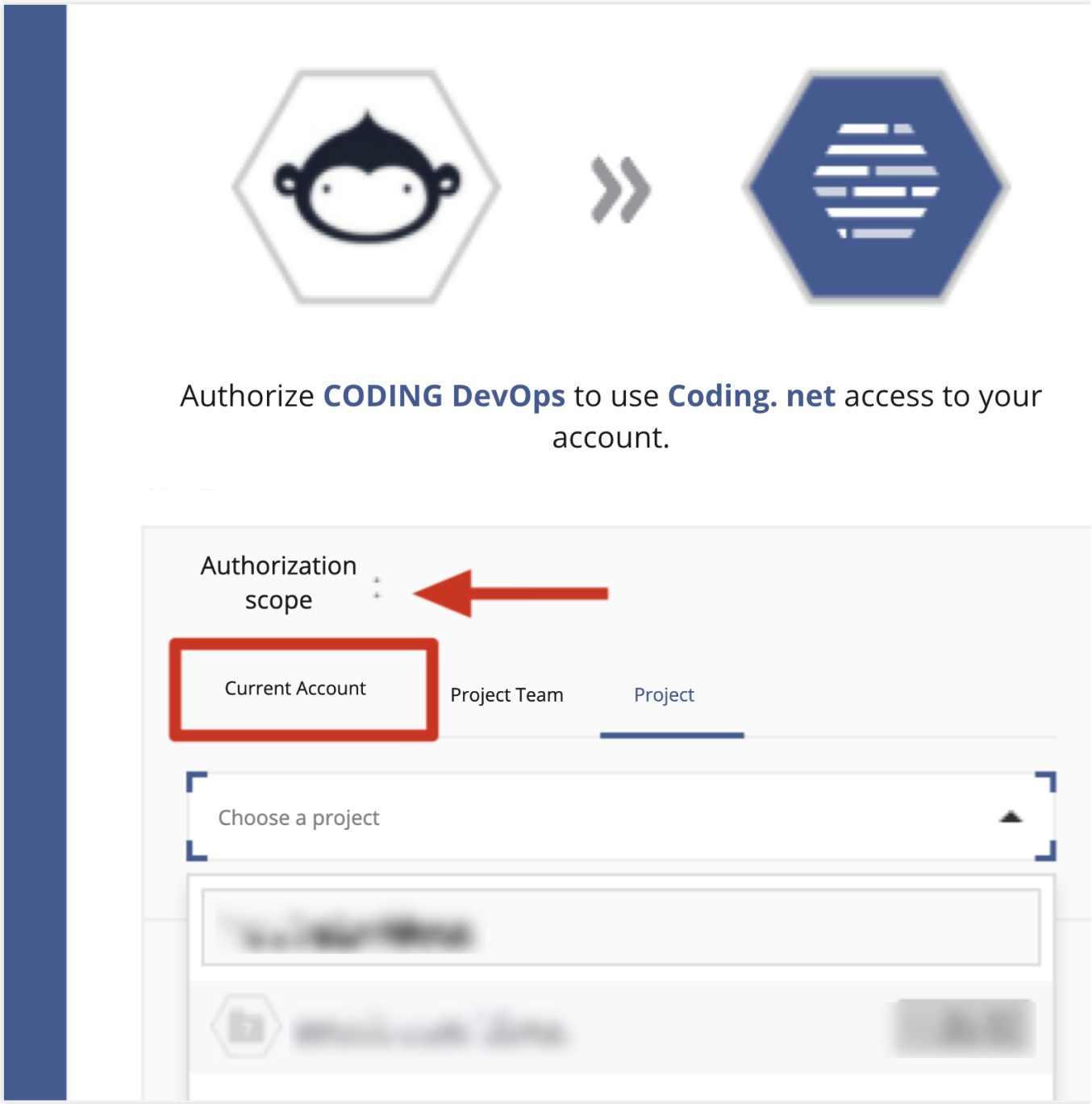


Jenkinsfile

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

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。

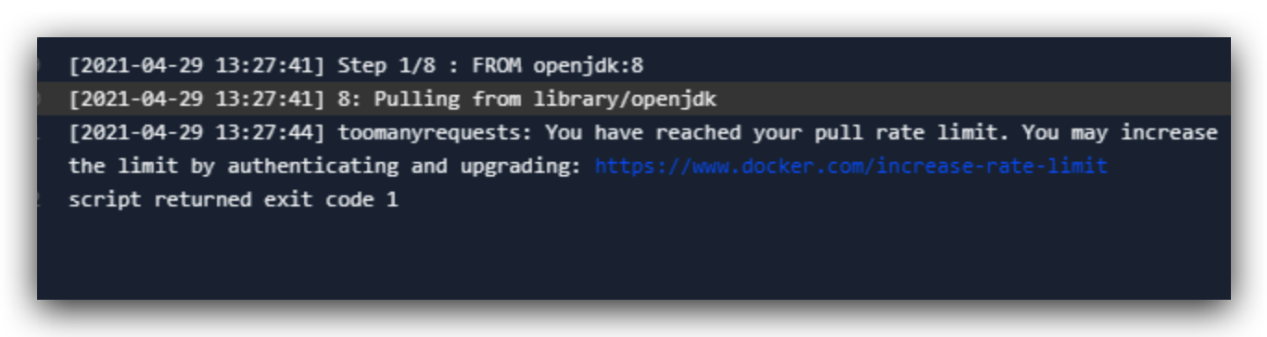


持续集成与制品库相关

最近更新时间：2023-12-29 11:44:51

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：



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，您可以参考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。

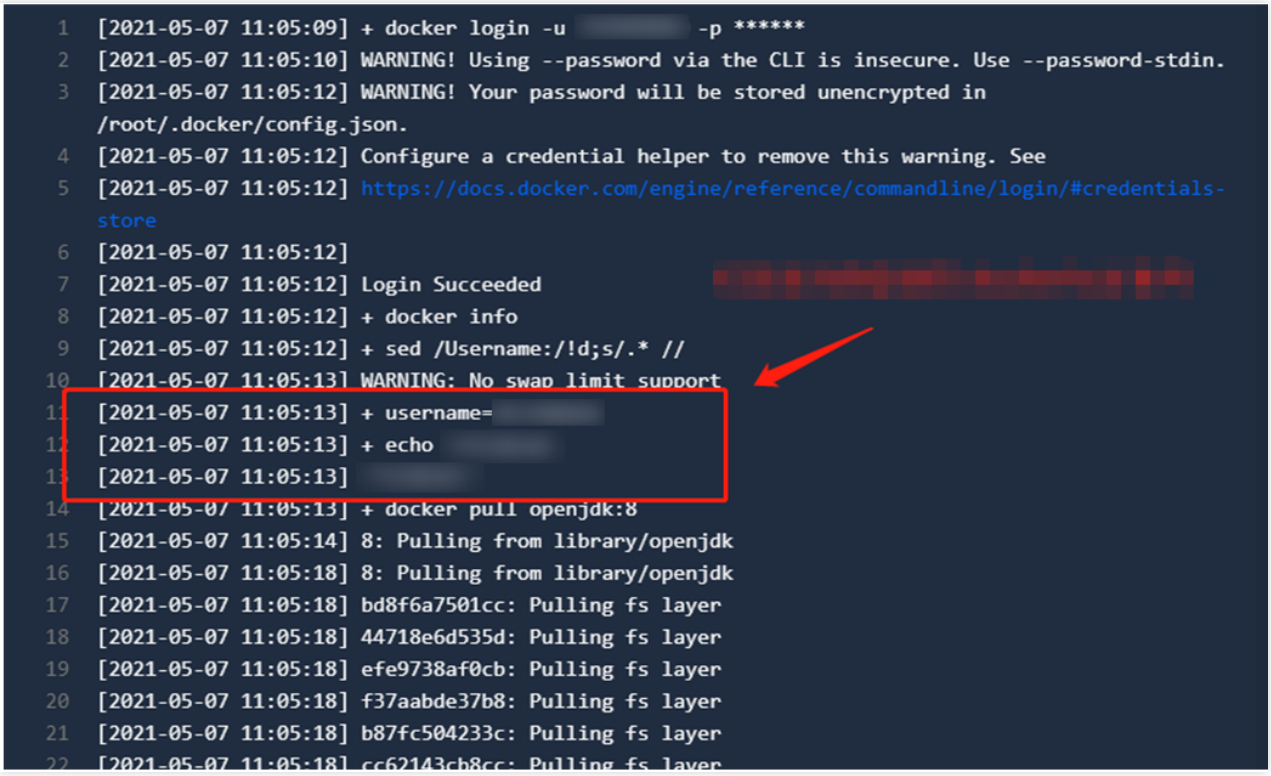
使用个人 Dockerhub 账号。

若您没有 dockerhub 账号，请 注册账号。

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。


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

自定义构建节点相关

最近更新时间：2023-12-29 11:44:51

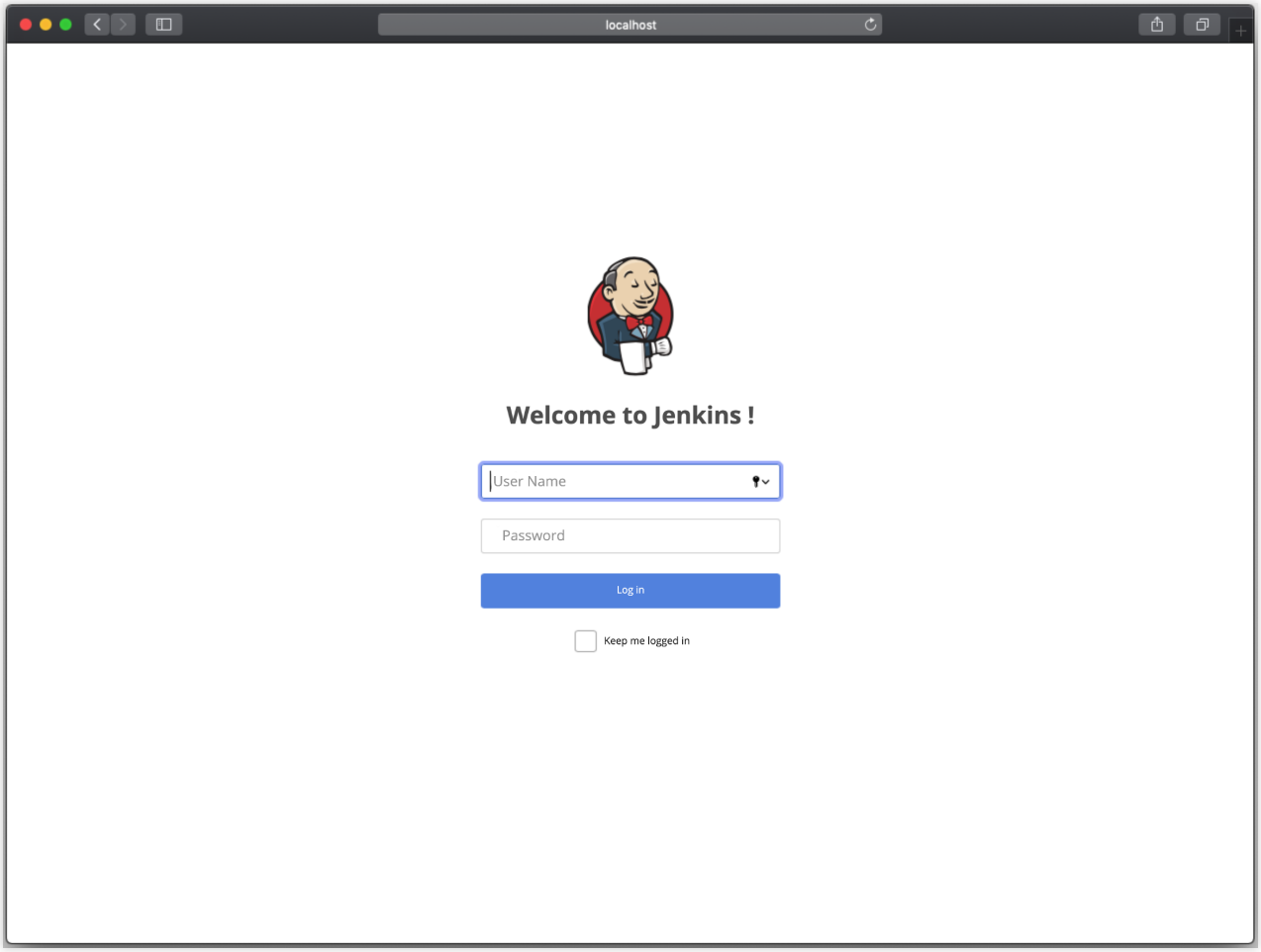
在自定义构建节点中访问本地 Jenkins？

步骤1：访问 Jenkins

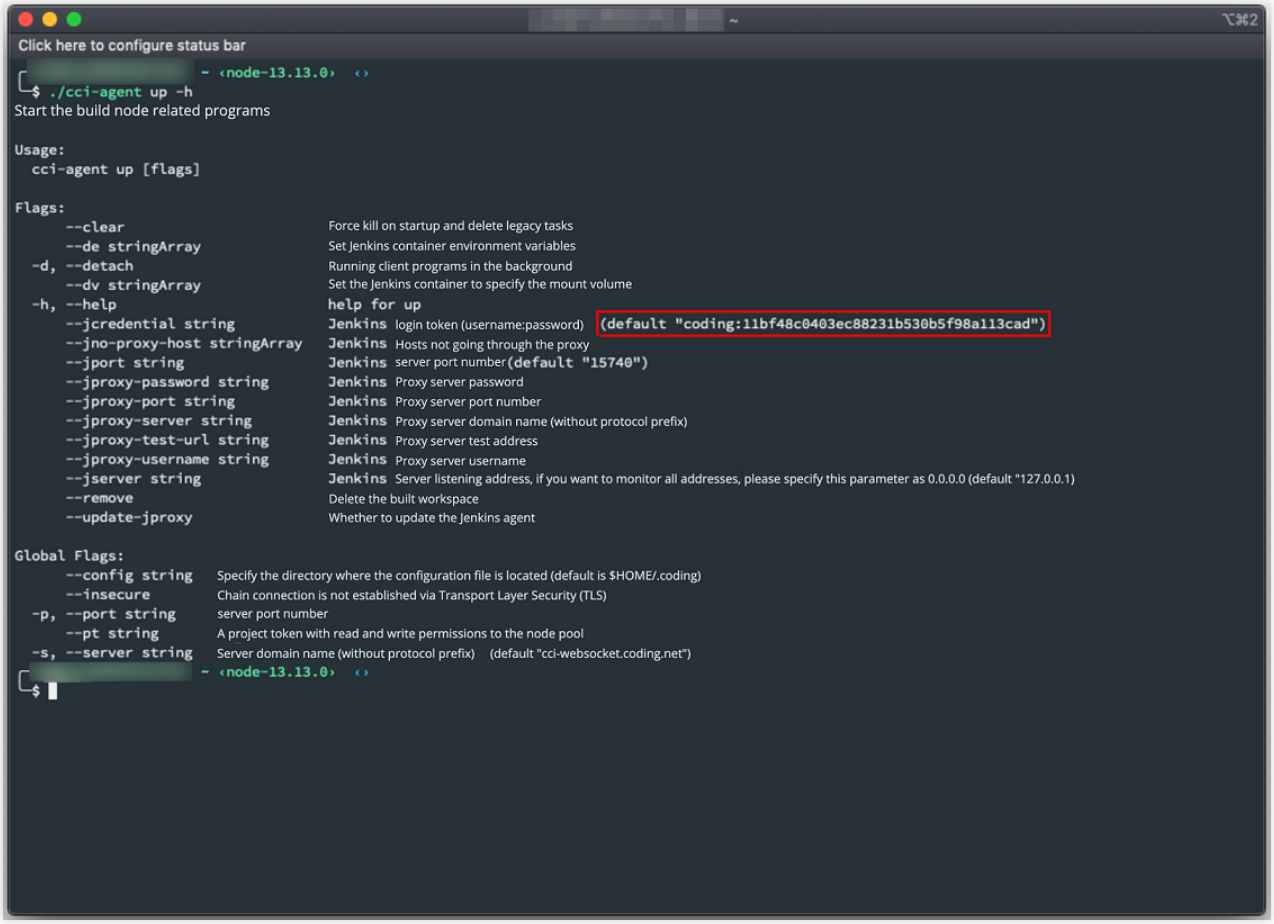
1. 首先需要将您的设备作为自定义构建节点接入
2. 为了避免对外暴露端口，CODING CI 自定义节点默认启动的 Jenkins 只会监听本地回环地址（127.0.0.1），默认的监听端口为 `15740`，此时，您只能在构建节点机器通过 localhost 或者 127.0.0.1 进行访问，具体的访问地址为 `<http://localhost:15740>`。
3. 如果无法访问，可以通过命令 `cat ~/.coding/cci-agent.yml` 查看端口 (publicPort)
4. 如果您希望在构建节点外部访问 Jenkins，可以在执行 `up` 命令启动程序时添加 `--jserver 0.0.0.0` 参数，同时也可以使用 `--jport` 指定监听端口，假设构建节点 IP 为 `NODE_IP`，指定的监听端口为 `PORT` 此时的访问地址为 `<http://NODE_IP:PORT>`。

步骤2：Jenkins 登录令牌

在浏览器中输入 Jenkins 访问地址，会看到登录页面。



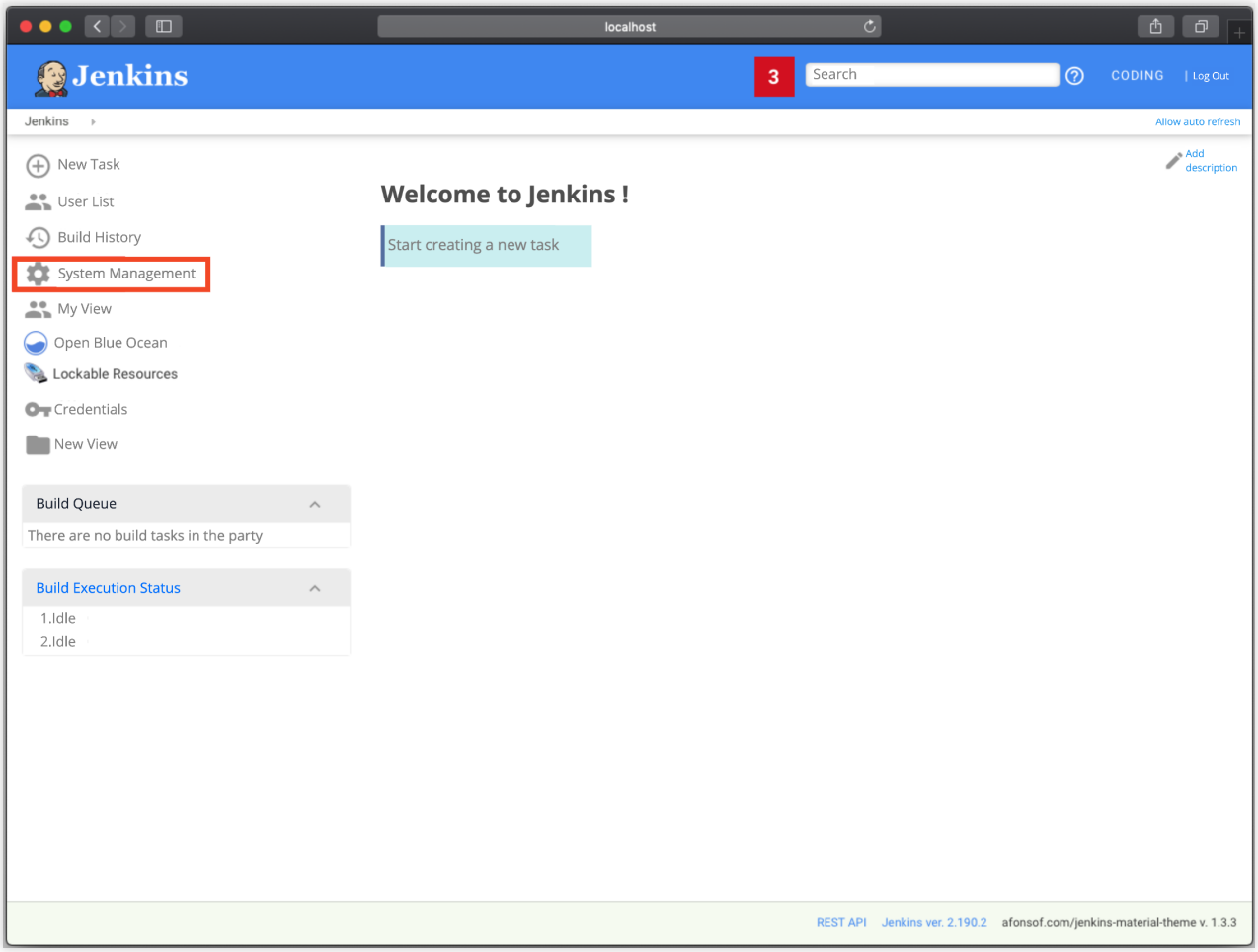
Jenkins 登录用户名和密码分别为 `coding` 和 `11bf48c0403ec88231b530b5f98a113cad`，您可以执行 ./cci-agent up -h 命令。



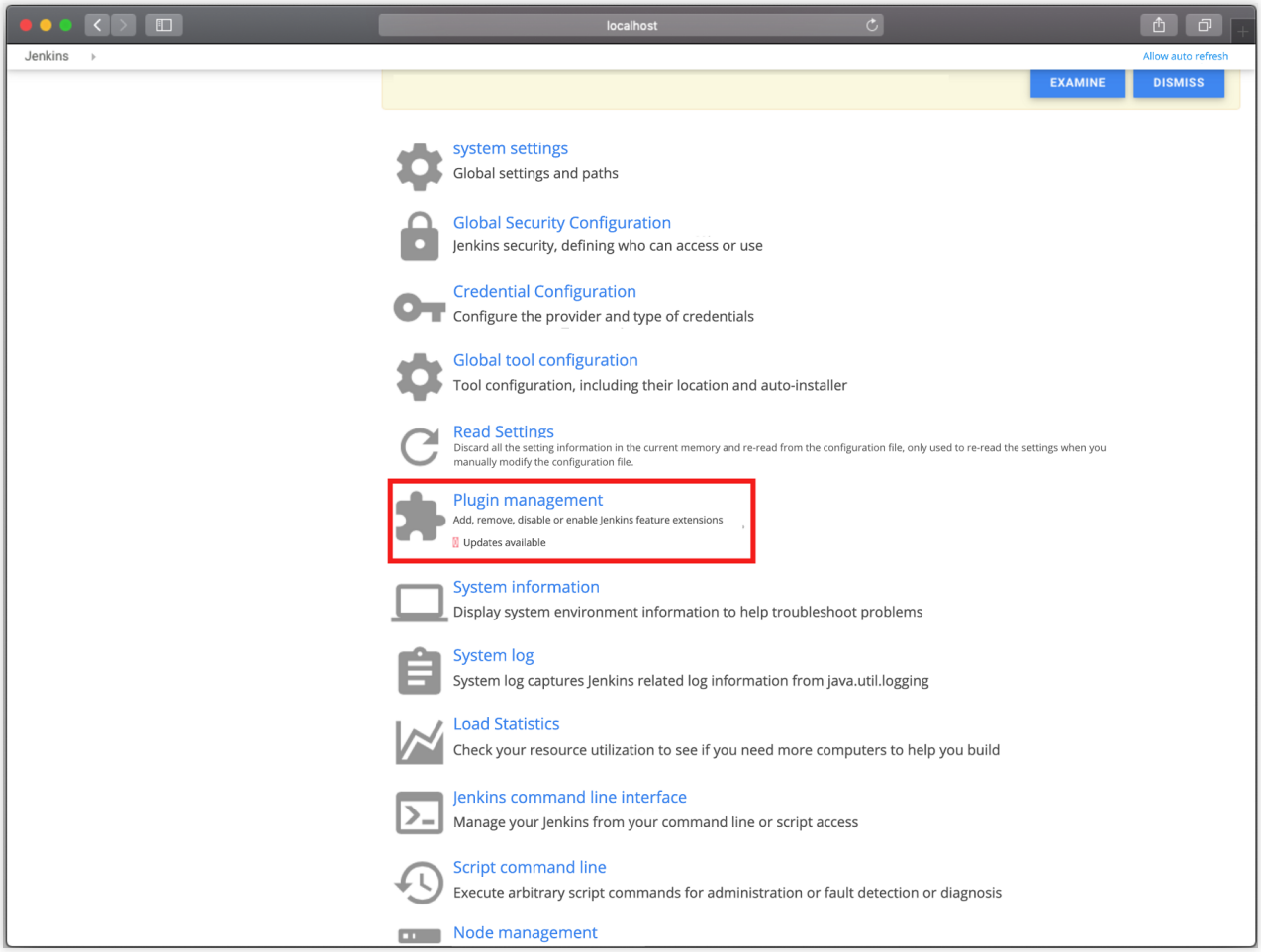
如何在自定义构建节点安装插件？

自定义构建节点是以开源软件 Jenkins 为引擎进行构建的，Jenkins 提供了超过 1000 个插件支持构建、部署、自动化。CODING CI 自定义节点默认提供的 Jenkins 仅内置了最常用的部分插件，如果不能满足您项目的需求，您可以自行安装需要的插件。

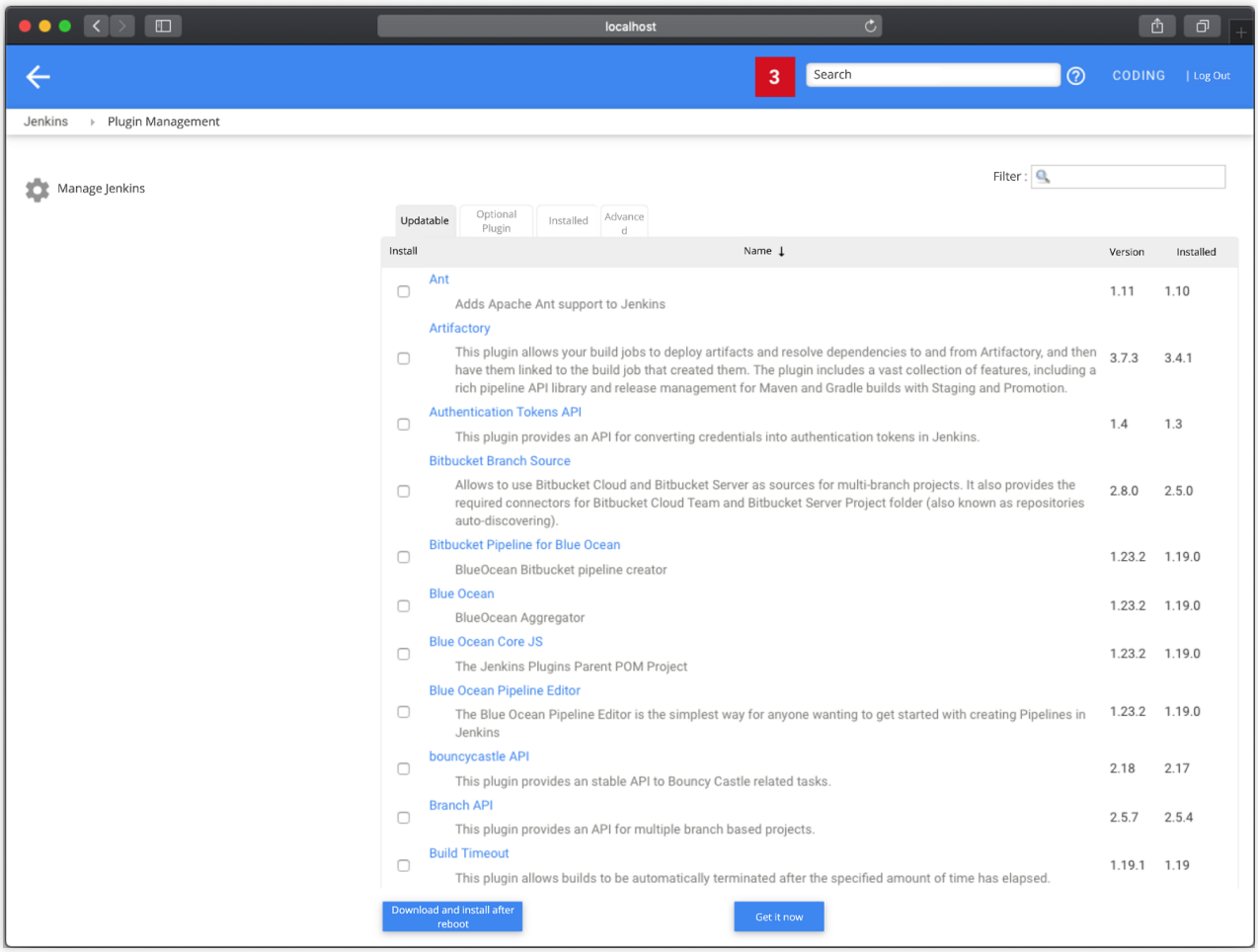
1. 首先需要您登录 Jenkins。
2. 登录到 Jenkins 后，可以看到 Jenkins 管理界面，进入**系统管理** > **插件管理**页面。
3. 左侧菜单栏中单击**系统管理**。



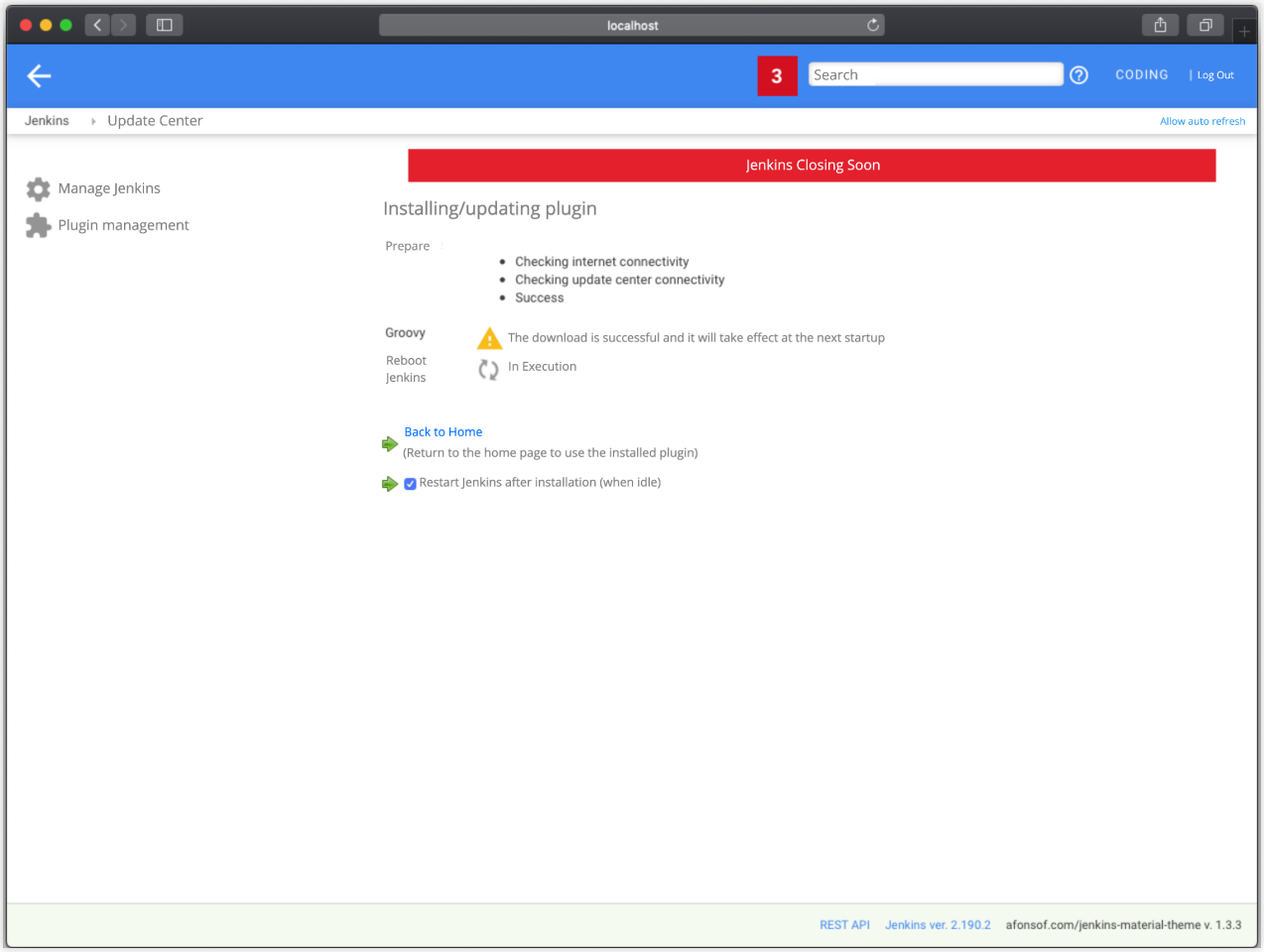
4. 下拉页面，单击**插件管理**。


localhost
Jenkins
Allow auto refresh
EXAMINE
DISMISS
system settings
Global settings and paths
Global Security Configuration
Jenkins security, defining who can access or use
Credential Configuration
Configure the provider and type of credentials
Global tool configuration
Tool configuration, including their location and auto-installer
Read Settings
Discard all the setting information in the current memory and re-read from the configuration file, only used to re-read the settings when you manually modify the configuration file.
Plugin management
Add, remove, disable or enable Jenkins feature extensions
Updates available
System information
Display system environment information to help troubleshoot problems
System log
System log captures Jenkins related log information from java.util.logging
Load Statistics
Check your resource utilization to see if you need more computers to help you build
Jenkins command line interface
Manage your Jenkins from your command line or script access
Script command line
Execute arbitrary script commands for administration or fault detection or diagnosis
Node management

5. 打开插件管理页面。



6. 在插件管理页面，找到**可选插件**选项页，搜索并勾选需要安装的插件，点击页面下方的**下载待重启后安装**，在弹出的**更新中心**页面勾选**安装完成后重启 Jenkins**，等待 Jenkins 安装完成后自动重启，即可使用。

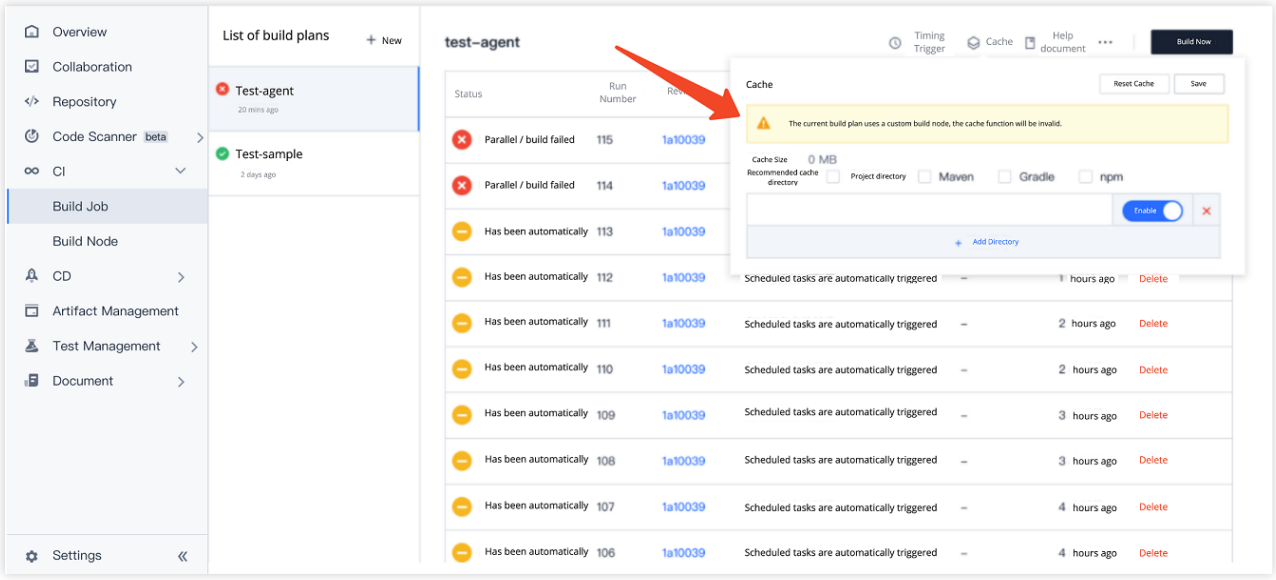


如何自定义构建节点配额？

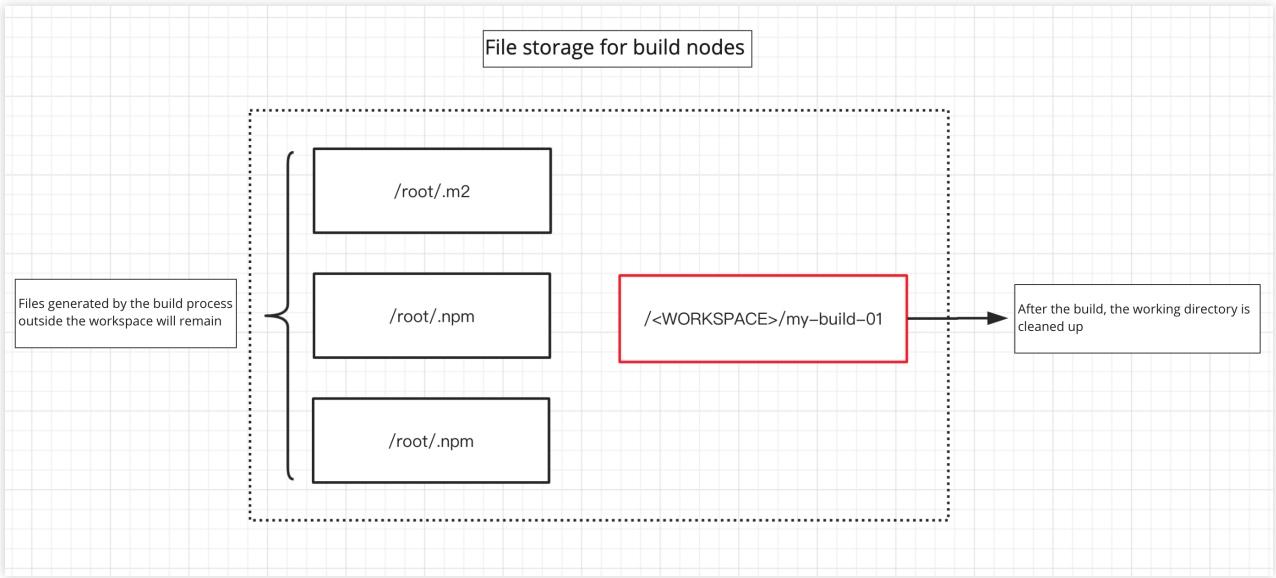
接入的自定义节点不会受到 CODING 团队配额限制；
自定义构建节点不计入团队每周构建次数配额限制；
自定义构建节点不受团队并行上限配额限制；
自定义构建节点不受超时配额限制；

如何自定义构建节点的缓存？

配置了自定义构建节点池的构建计划将使用构建节点自身的缓存。

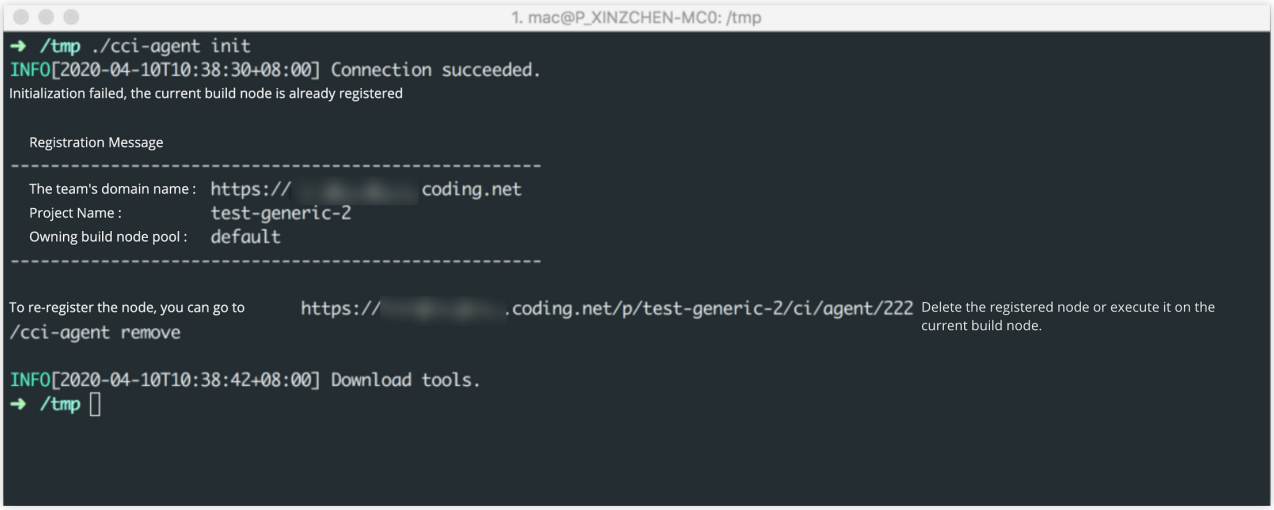


使用自定义构建节点执行的构建计划，在每次执行的时候会创建一个独立的 `WorkSpace`，构建结束后会将其清理。构建过程中产生在工作空间外的文件将会保留（如 maven / npm 等制品库的全局缓存）。



重复注册 cci-agent 如何处理？

默认的情况下，如果在一台机器上重复注册 cci-agent 会提示节点已被注册，并且需要删除注册节点后才能重新注册。



这是因为注册节点需要提供 config 目录（默认为 `~/.config` ）和端口号（默认为 15740），您只需要手动指定不冲突的 config 目录和端口号即可重复注册节点。
在进行操作之前，请确保您已经在**项目设置 > 开发者选项 > 项目令牌**申请了具备构建节点权限的项目令牌密码。
下面为使用 `~/.coding2` 和 `15741` 端口 进行重复注册的示例操作。

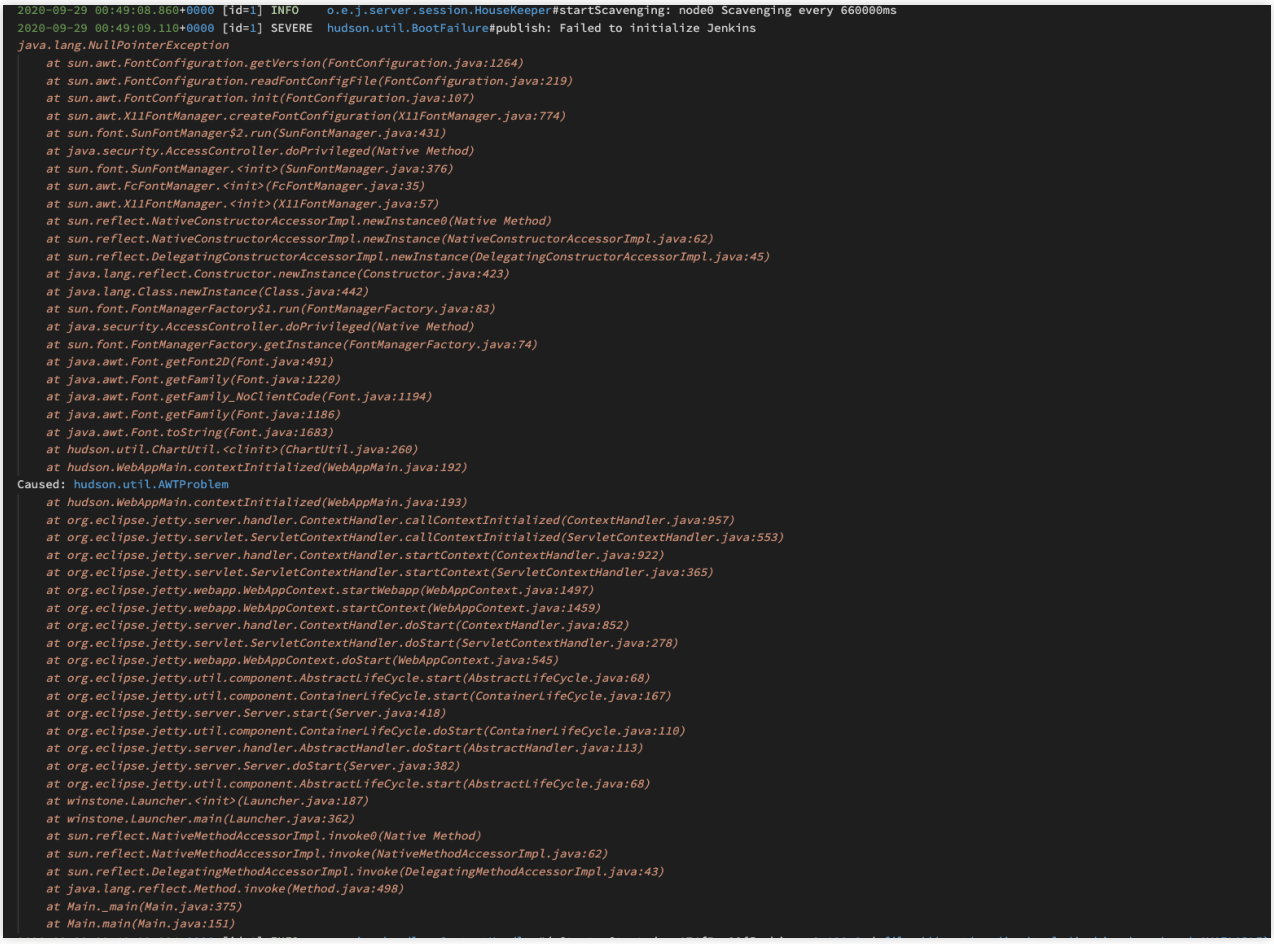
```
./cci-agent init --config ~/.coding2 --pt <项目令牌密码>
```

```
./cci-agent up --config ~/.coding2 --jport 15741
```

自定义节点异常问题收集

自定义节点操作系统为最小化安装的 CentOS

如果您的自定义节点 Jenkins 无法启动，如下情况报 Warning 时，可能是由于节点的操作系统为最小化安装的 CentOS (Minimal Install)，而 Jenkins 的网页页面依赖一些图形化组件。



解决方案为执行一下命令：

```
yum install fontconfig
```

异常状态处理方式

在实际生产中，接入自定义构建节点可能会存在如客户端的网络环境，配置环境缺失等诸多不稳定因素，下面将罗列出已知异常状态的处理方式。

删除构建节点池

相关位置	构建记录错误提示	处理方式
配置构建计划	-	构建计划节点池配置中不会出现已经被删除的构建节点池
已配置构建计划	配置页面上会提示构建节点池已被删除	在构建计划中已配置的构建节点池被删除

触发构建任务	该构建计划配置的构建节点池 my\\-pool 已被删除，请重新配置	允许触发，但构建任务会立即失败
队列中、初始化、准备构建、构建中	该构建计划配置的构建节点池 my\\-pool 已被删除，请重新配置	这几种状态的构建任务会立即失败

删除处于"占用"状态的构建节点

相关位置	构建记录错误提示	处理方式
队列中的构建任务	-	不受到影响，队列中的构建任务尚未分配具体的构建节点，在寻找到有效构建节点之前会一直处于队列中的状态
初始化、准备构建、构建中	构建节点 xxx 已离线	这几种状态的构建任务会立即失败

处于“占用"状态的构建节点掉线

说明：

由于客户端网络环境不稳定，构建节点可能会存在掉线的情况

掉线后客户端（构建节点）会尝试进行重试，服务端会尝试等待客户端重新连接

等待超时（3 分钟）则会判断构建节点离线，并将构建任务标记为失败

若重连成功，客户端将继续上报构建的内容

相关位置	构建记录错误提示	处理方式
队列中的构建任务	-	不受到影响，队列中的构建任务尚未分配具体的构建节点，在寻找到有效构建节点之前会一直处于队列中的状态
初始化、准备构建、构建中	构建节点 xxx 已离线	这几种状态的构建任务会立即失败

配置的构建节点池没有接入节点

相关位置	构建记录错误提示	处理方式
配置构建计划	-	因为构建计划并不会直接关联构建节点，所以不影响构建计划对配置，若节点池内不存在构建节点，配置页面中会给出相应的警告

已配置构建计划	-	因为构建计划并不会直接关联构建节点，所以不影响构建计划对配置，若节点池内不存在构建节点，配置页面中会给出相应的警告
初始化、准备构建、构建中	该构建计划配置的构建节点池 my\\-pool 已被删除，请重新配置	这几种状态的构建任务会立即失败

取消构建计划授权

相关位置	构建记录错误提示	处理方式
配置构建计划	会给出相对应的提示信息	不允许用户选中未授权的构建计划
已配置构建计划	-	构建记录列表页面和构建配置页面会给出未授权的提示信息，用户需要手动调整节点池配置
触发构建任务	该构建计划没有获得构建节点池 default 的授权，请授权	允许触发，但构建任务会立即失败
初始化、准备构建、构建中	该构建计划配置的构建节点池 my\\-pool 已被删除，请重新配置	这几种状态的构建任务会立即失败