

# **CODING Continuous Integration**

## **FAQs**

### **Product Documentation**



## Copyright Notice

©2013-2024 Tencent Cloud. All rights reserved.

Copyright in this document is exclusively owned by Tencent Cloud. You must not reproduce, modify, copy or distribute in any way, in whole or in part, the contents of this document without Tencent Cloud's the prior written consent.

## Trademark Notice



All trademarks associated with Tencent Cloud and its services are owned by Tencent Cloud Computing (Beijing) Company Limited and its affiliated companies. Trademarks of third parties referred to in this document are owned by their respective proprietors.

## Service Statement

This document is intended to provide users with general information about Tencent Cloud's products and services only and does not form part of Tencent Cloud's terms and conditions. Tencent Cloud's products or services are subject to change. Specific products and services and the standards applicable to them are exclusively provided for in Tencent Cloud's applicable terms and conditions.

# Contents

## FAQs

Jenkinsfile Syntax

Build Execution Issues

Continuous Integration and Code Repositories

Continuous Integration and Artifact Repositories

Custom Build Nodes

# FAQs

## Jenkinsfile Syntax

Last updated : 2023-12-29 11:44:51

### Why am I prompted with "cannot pull code" when using ci-init?

When the ci-init command is used for a build plan (job) created before October 10, 2019, a public/private key pair is created for the user, so they can pull code from repositories in the project. A key pair will not be created when calling ci-init for a build plan created after this date.

We have built in a credential ID for newly created build plans so users can pull code from repositories. Simply use `env.CREDENTIALS_ID` as the `credentialsId` of `userRemoteConfigs`.

### Old syntax





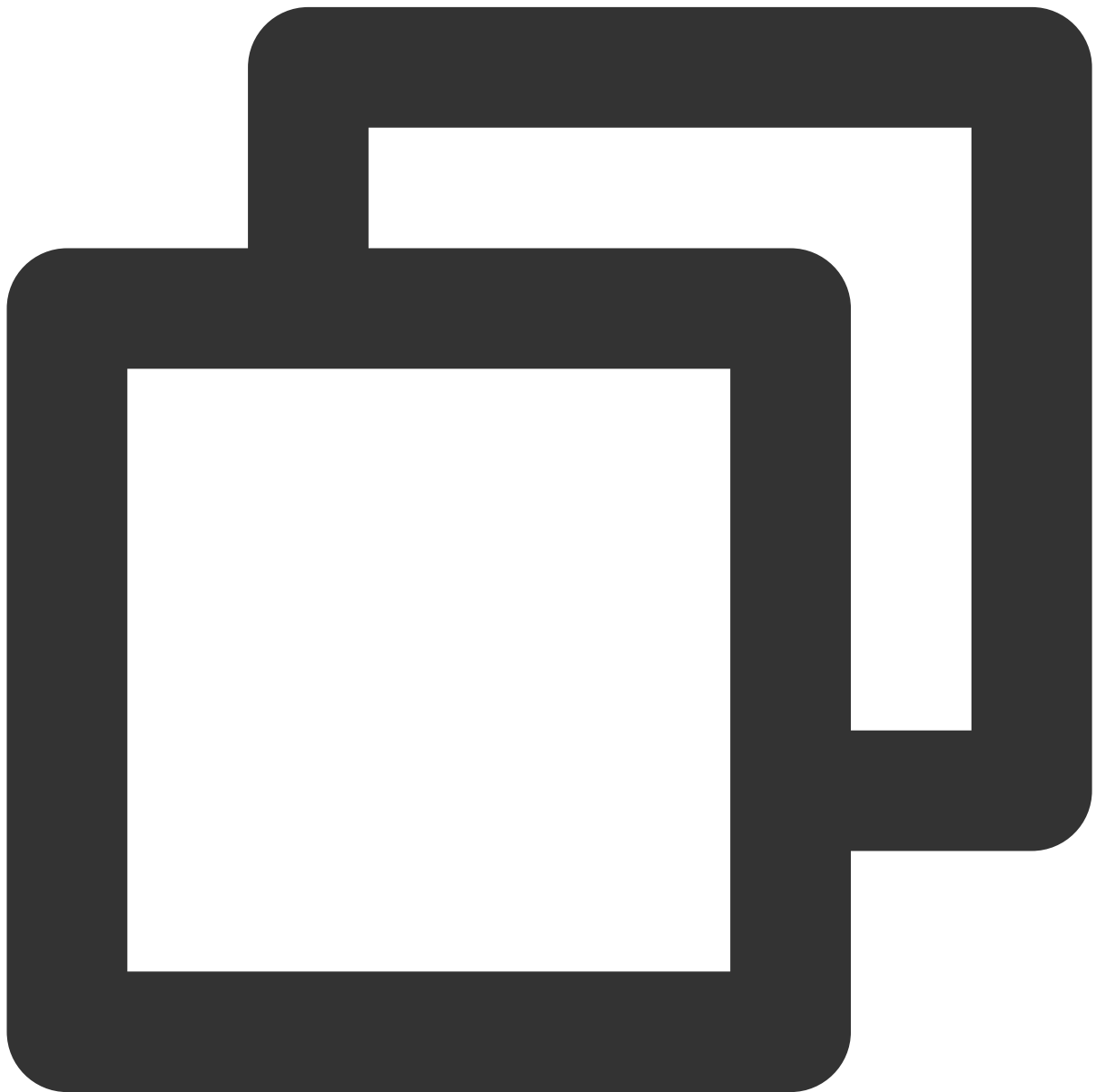
```
pipeline {
  agent any
  stages {
    stage('check out') {
      steps {
        // The old syntax includes ci-init
        sh 'ci-init'

        checkout([
          $class: 'GitSCM',
          branches: [[name: env.GIT_BUILD_REF]],

```

```
        userRemoteConfigs: [[url: env.GIT_REPO_URL]]
    })
}
}
```

### New syntax



```
pipeline {
    agent any
```

```
stages {
  stage('check out') {
    steps {
      checkout([
        $class: 'GitSCM',
        branches: [[name: env.GIT_BUILD_REF]],
        // Note that the new checkout syntax has the additional credential
        userRemoteConfigs: [[url: env.GIT_REPO_URL, credentialsId: env.
      ]])
    }
  }
}
```

**As CODING already supports credential management, you should use the more secure credential ID in place of ci-init.**

### When should I use single or double quotes?

When using CODING Continuous Integration (CODING-CI), you often need to splice strings or use environment variables as parameters in a Jenkinsfile. Single and double quotes are used differently in a Jenkinsfile. The following demonstrates the differences between the commonly used echo and sh commands.



```
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}"
    // Output: this is my env

    echo "\\${env.MY_ENV}"
    // Output: ${env.MY_ENV}

    echo "${MY_ENV}"
    // Output: define in script

    echo '${MY_ENV}'
    // Output: ${MY_ENV}

    sh 'echo ${MY_ENV}'
    // Output: this is my env

    sh "echo ${MY_ENV}"
    // Output: define in script

    sh "echo ${env.MY_ENV}"
    // Output: this is my env
}
}
}
}
```

echo: When using single quotes, the \$ symbols inside are not parsed, but directly included in the output. When using double quotes, MY\_ENV in the environment variables is printed.

sh: When using single quotes, the original text is executed as the sh command normally used in the terminal, so MY\_ENV in the environment variables can be printed.

## When creating a build plan, what is the difference between choosing to use a Jenkinsfile from a code repository or a static, configured Jenkinsfile?

When you select a Jenkinsfile from a code repository, the file is stored in the repository. Modifications to the Jenkinsfile require commits to the code repository. If you modify the trigger conditions for continuous integration, integration tasks can still be automatically triggered.

When you use a static, configured Jenkinsfile, the file is not stored in a code repository, so modifications to the Jenkinsfile do not involve repository updates. During build execution, you should use only static, configured files to ensure the consistency of the build process.

# Build Execution Issues

Last updated : 2024-05-28 10:15:28

In most computer operating systems, any process that exits leaves an exit code that indicates whether the process ran as expected. Therefore, if the exit code of the execution process in continuous integration (CI) is not 0, the system judges the build to have failed. The following are common causes of build execution failure:

## How do I fix CI configuration file syntax errors?

Like most programming languages, a Jenkinsfile is composed of a domain-specific language (DSL), so syntax errors can cause compilation or runtime failures.

## How do I resolve failed tests?

Most mainstream testing tools and frameworks set the exit code to a non-zero value by default when the test logic fails.

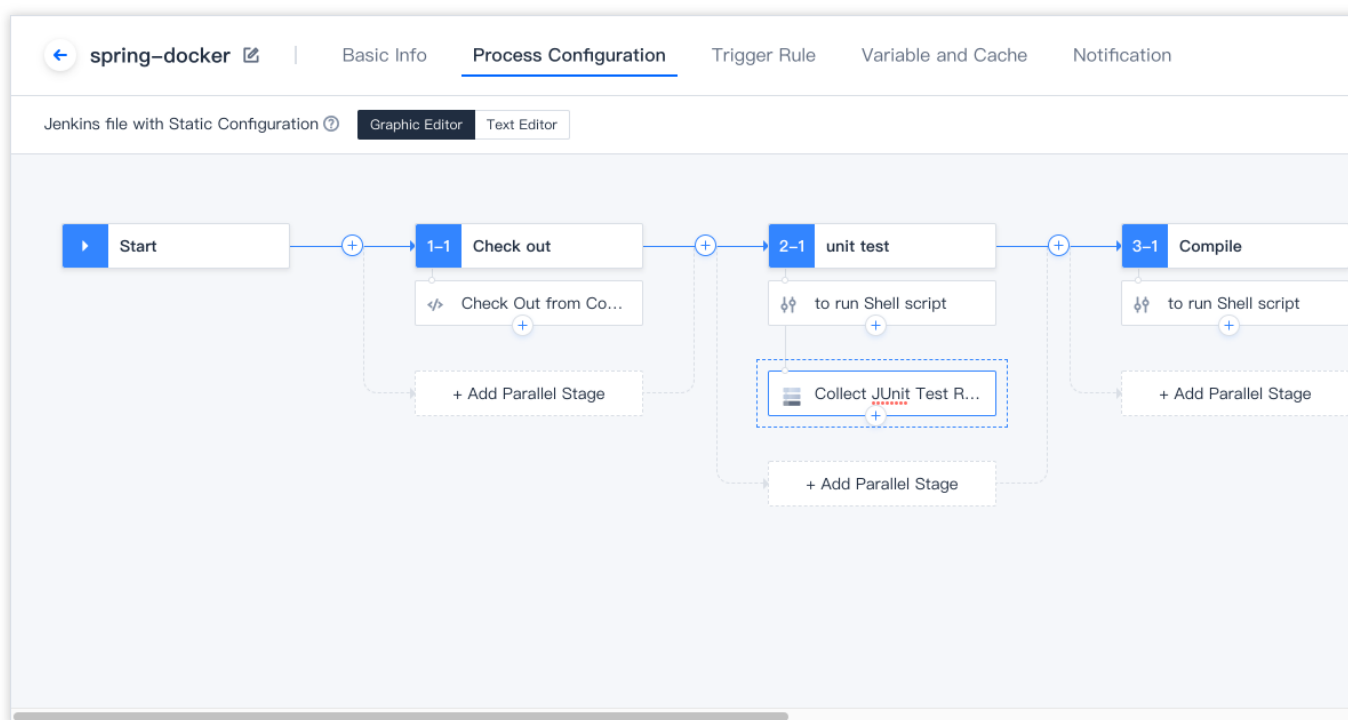
## How do I resolve a build timeout or an insufficient build quota?

When using CODING Continuous Integration (CODING-CI), each team has a certain build quota. To prevent the malicious use of CI in cyberattacks, each build task has a timeout limit. Build tasks that time out or exceed the build quota are terminated by the system. If you need a higher quota, you can adjust the quota in Team Management by purchasing the quota you require.


## How do I view build logs and build snapshots?


CODING-CI provides build logs, which allow users to determine the causes of faults. In addition, CODING-CI provides a configuration snapshot for each build. You can use the snapshot to get the configuration file content and parameters used in the build. This way, you can see if configuration issues caused the build to fail.

### Build log


















## Build snapshot

← **spring-docker**  | Basic Info Process Configuration Trigger Rule Variable and Cache

**Process Environment Variable**  Batch add string type environment variables | [+ Env Variable](#)

Add the environment variable of the build job. When the build task is manually started, the environment variable will serve as a default value.

Variable Name	Category	Default Value	Operation
DOCKER_IMAGE_VERSION 	String	\${GIT_LOCAL_BRANCH:-branch}-\${GI...	 
DOCKER_IMAGE_NAME 	String	java-spring-app	 
DOCKERFILE_PATH 	String	Dockerfile	 
DOCKER_BUILD_CONTEXT 	String	.	 
DOCKER_REPO_NAME 	String	test	 

**Cache Directory**

- Enabling cache can avoid repetitive download of the dependency files in each build, greatly improving the build speed.
- If an error occurs on your build cache, reset the cache.
- You are advised to enable cache for Maven, Gradle, and npm cache directories.

Reset Cache

## How do I run automated tasks locally?

You can re-execute the automated logic (for example, re-run the test code locally) or modify the code in real time to get more feedback for troubleshooting.

## What happens if I use an interactive command-line program?

In the CI process, you cannot directly use interactive commands. If you use a program that calls up an interactive command-line window, the build will fail.

A common command is `npm login` `docker login -u xxx` (the `docker -u xx -p xx` command is required when logging in to Docker during CI).

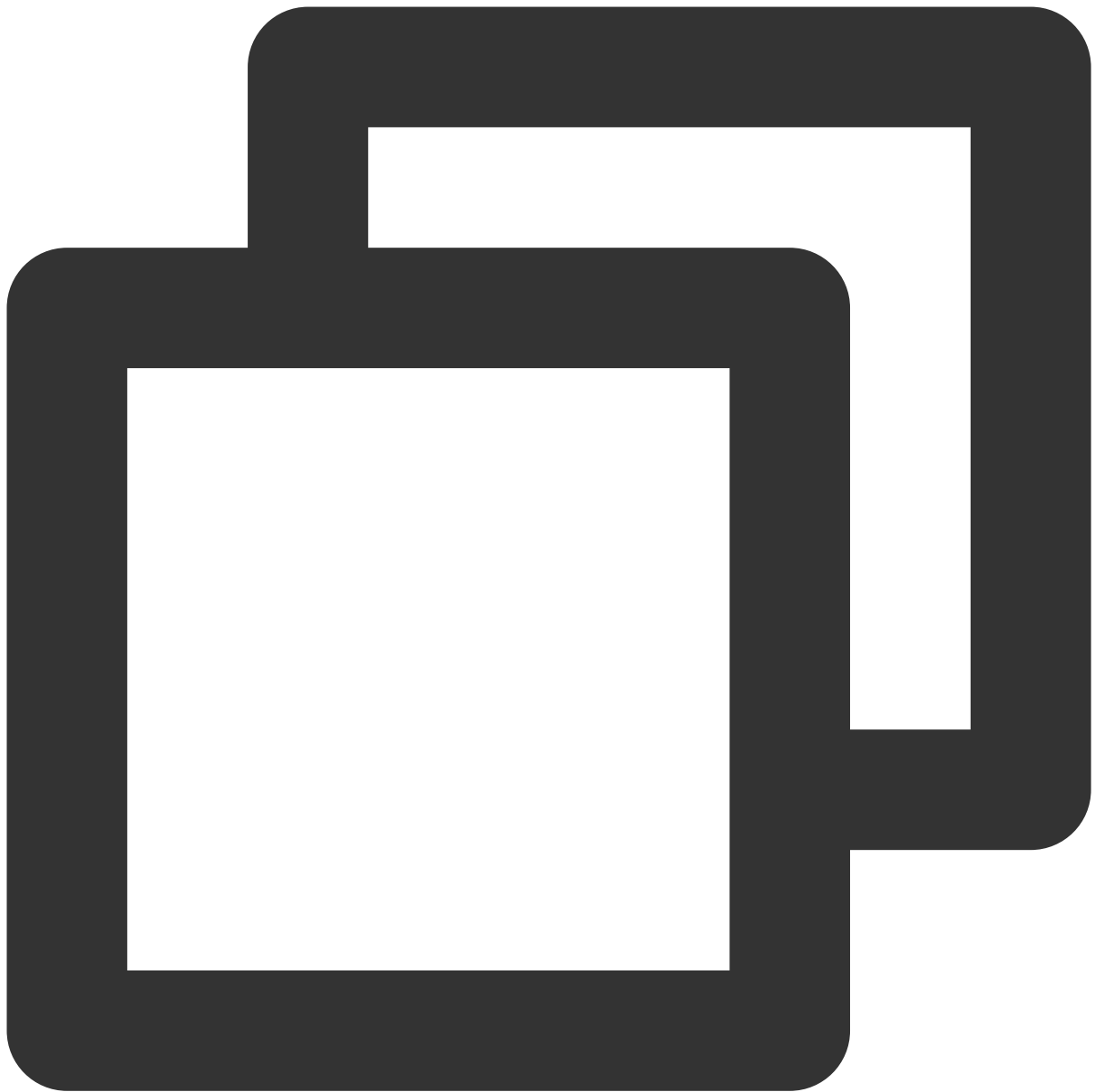
### Note:

If you cannot find an answer to your question in this document, please go to the [Ticket Center](#) to submit an issue. We will promptly provide a solution to your problem.

## How do I debug build tasks?

If you need to debug a build run process, you can provide the SSH by adding the following steps to the build process:





```
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}"
```

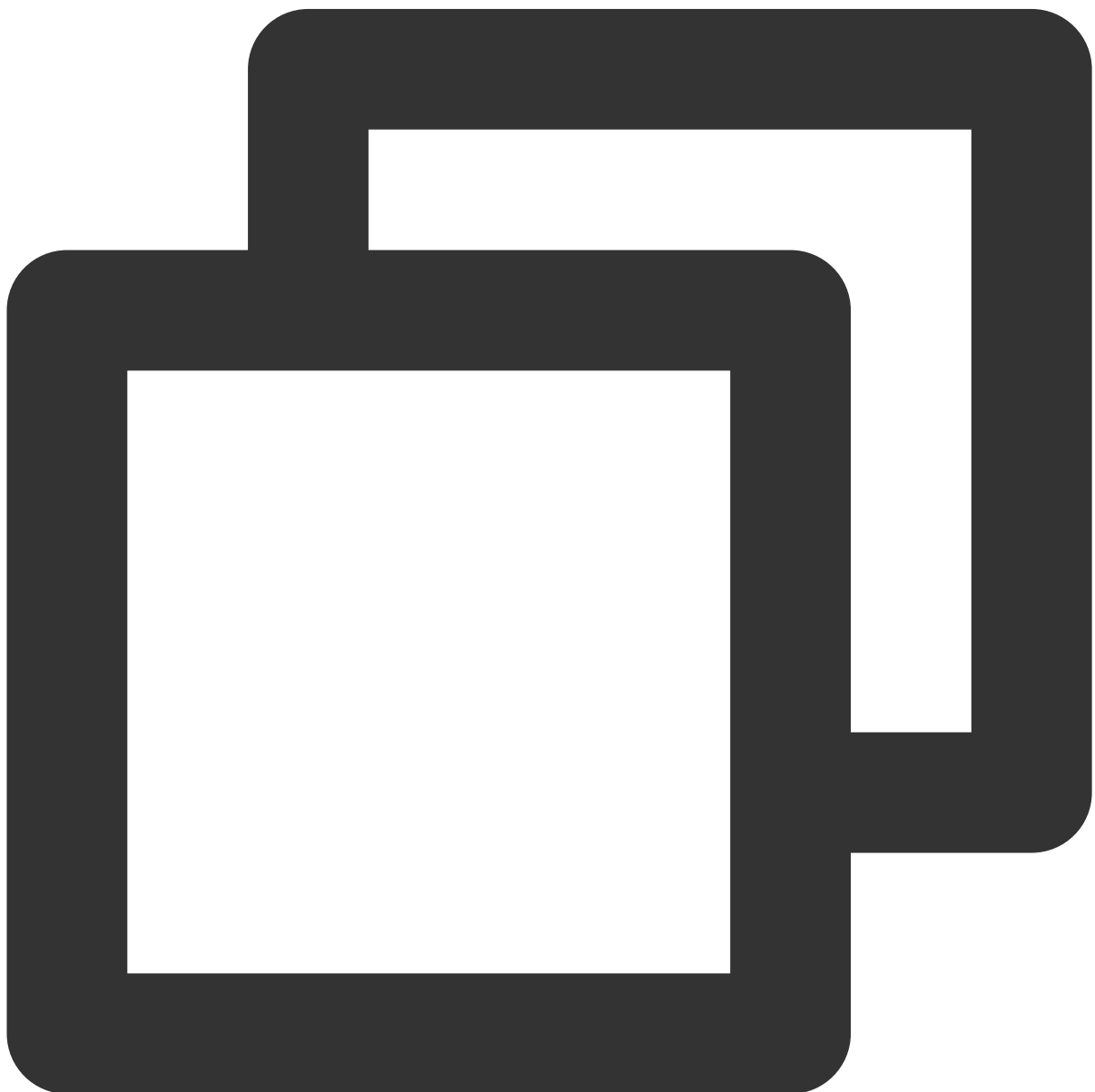
```
echo "SSH: ${tmateSSH}"  
'''  
    sh 'sleep 3600'  
}
```

# Continuous Integration and Code Repositories

Last updated : 2023-12-29 11:44:51

## How do I push code during continuous integration?

In some scenarios, you may have to push code during continuous integration (CI). CODING Continuous Integration (CODING-CI) provides built-in command tools, including Git and SVN. You can refer to the example below.



```
pipeline {
  agent any
  stages {
    stage('check out') {
      steps {
        checkout([
          $class: 'GitSCM',
          branches: [[name: env.GIT_BUILD_REF]],
          userRemoteConfigs: [[url: env.GIT_REPO_URL, credentialsId: env.CREDENTI
        ]
      ]
    }
    stage('modification') {
      steps {
        sh "echo '# Hello CODING' > README.md"
        sh "git add ."
        sh "git commit -m 'add README.md' "

      }
    }
    stage('push') {
      steps {
        // The CODING-CI system's preset project token environment variables PR
        // To push to the code repository of another project or a third-party p
        sh "git push https://${PROJECT_TOKEN_GK}:${PROJECT_TOKEN}
          @e.coding.net/myteam/myrepo.git HEAD:master"

      }
    }
  }
}
```

## How do I call SVN repositories?

In the default CI plan configuration process, the code source is a Git repository by default. To use an SVN repository for continuous integration, follow the instructions below.

### Prerequisites

Before starting, create a project token and apply for username + password credentials.

### Step 1: Create project token

1. Go to **Project Settings > Developer Options > Project Token** and click *Create Project Token*. Set the expiration date and select all CI permissions.

← Settings

Project & Member

Collaboration

Project Announcement

Developer Options

Project Settings / Project Token / Create Project Token

Create Project Token

Token Name

No more than 60 characters.

Expiration Time

Select a date

Project Management Permission

☐ Collaboration  
Read and operate project collabo...

☐ Files  
Create, query, edit, delete

☐ WIKI  
Create, query, edit, delete

☐ API Documentation  
Publish API Documentation

☐ Associate Resource  
Create, query, edit, delete

☐ Project Member  
Read and manipulate projei

Code Repository Permission ⓘ

☒ Unified configuration all code warehouse permissions ☐ Appointed warehouse code configura

Repository Name	Access Permission	Operation Per
* All Code Repositories in the Project	<input checked="" type="checkbox"/> Read Read Code Repository	<input type="checkbox"/> R Pt

2. Once the token is created, you will receive a username and password.

Developer Options

API and Event

Project Token

Service Hook

Credential

### Project Token (2)

A project token can be used only for operating feature components in the related project. It cannot be used for other components, [click here](#).

Token Name	Username	Password	Creation Time	Expiration Time
Testing	ptna0khsr8rw	32e6*****4606	2022-02-23	2022-02-23
Artifact Repo	pt9aqreb50iv	7227*****f920	2021-11-04	2021-11-04

## Step 2: Apply for username and password credentials

Go to **Project Settings > Developer Options > Credential Management**, click **Enter Credential**, and enter your username and password. You must enter the username and password generated upon project token creation.

Developer Options

API and Event

Project Token

Service Hook

Credential

### Credential Management (5)

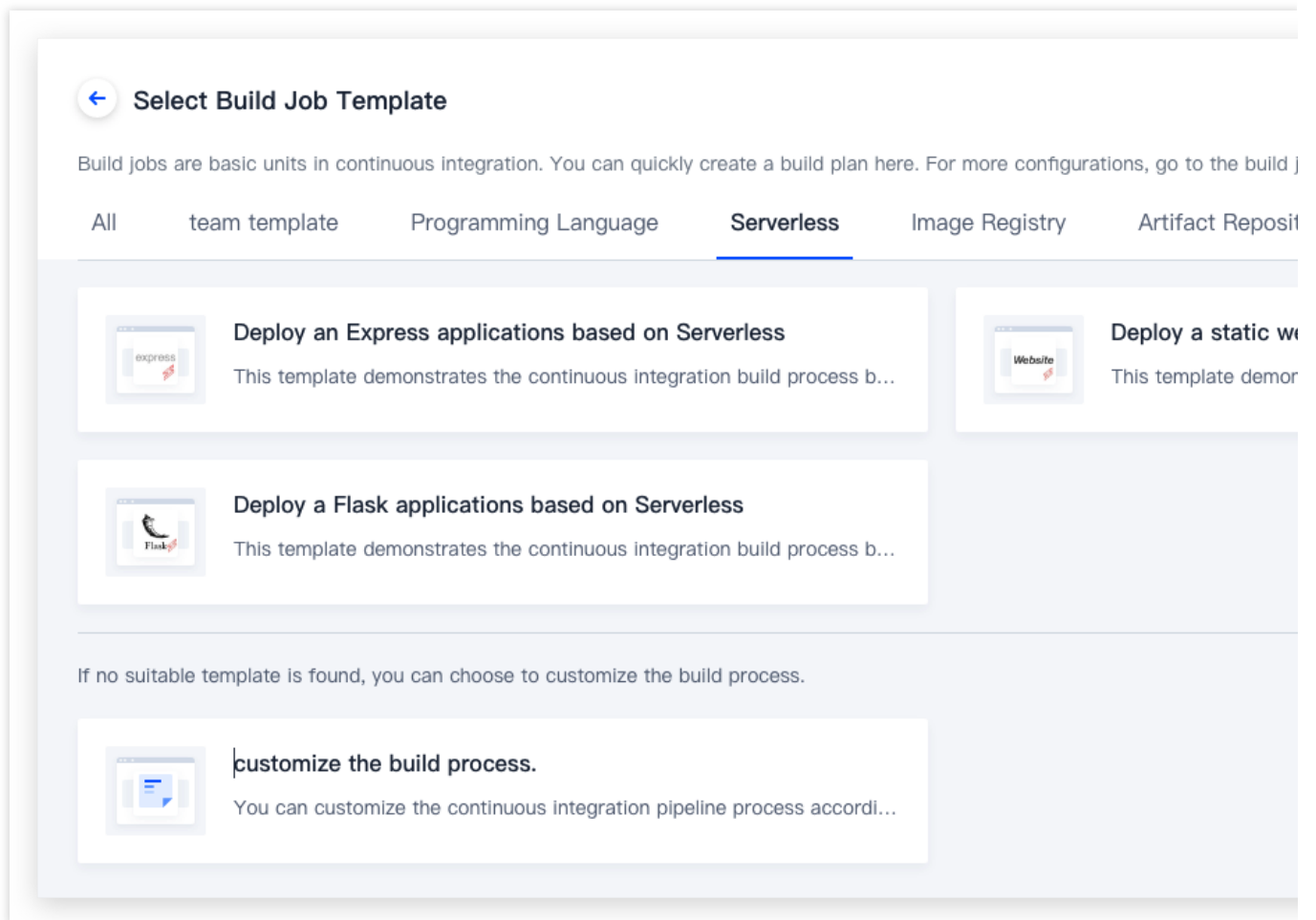
Storing passwords, private keys, and certificates into credential management maximizes the credential security. For continuous integration and deployment components, you can select entered credentials to use. [View comments](#)

Certificate Name	Authorized	Credential ID	Certificate Description
tcr-artifacts	Services not authorized yet	de900c2a-f57f-4f0b-9bc8-ae376cd5af70	-
tcr-artifacts	1	82589fd1-6a52-43cb-8674-4d6f5bc06cad	-

After your credentials are created, you will receive a credential ID. Later, you must input this ID in the build plan process configuration.

## Step 3: Configure build plan

1. Go to **Continuous Integration > Build Plans**, click **New Build Plan Configuration**, and go to **Select Build Plan Template > Basic**. On this page, select **Blank Template** in the Basic field. This allows you to customize the process configuration.



2. After naming the build plan, select **Not use** for the code source.

← customize the build process.


Build Job Name \*


empty-example


Build Process


1 Code Repository


Code Source


CODING


GitHub.com


GitLab.com

Private GitLab

Gitee

TGit

General Git

Do Not Use

2 Configuration Source

☒ Use the Jenkinsfile configured in static mode. ?

☒ Go to View Configuration Details

OK

Cancel

Jenkinsfile Preview

```
pipeline {
  agent any
  stages {
    stage('Custom Manufacturing Process') {
      steps {
        echo "Custom build process begins"
        // Please supplement your build process her
      }
    }
  }
}
```

3. Then, enter the relevant settings in the process configuration.



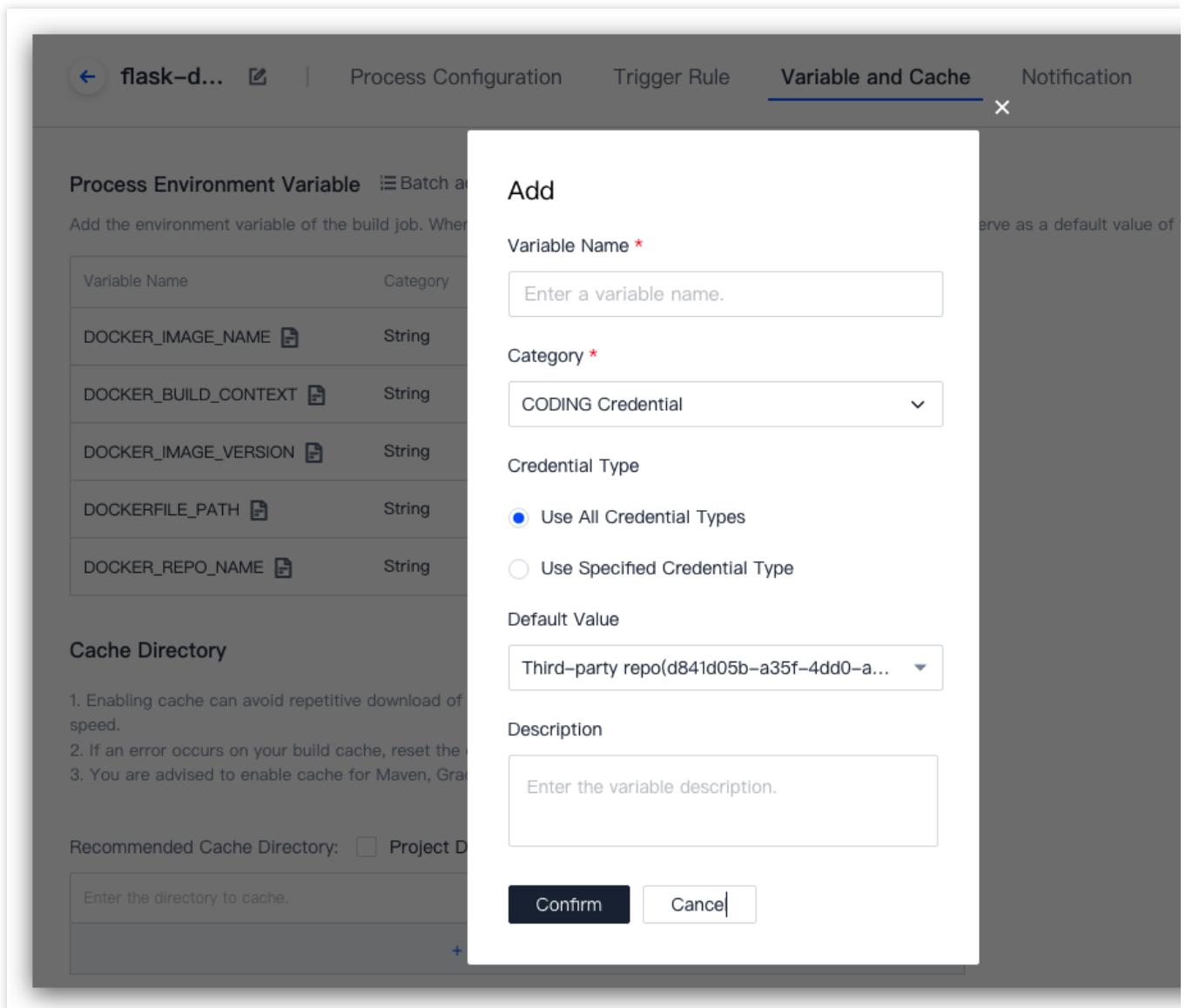


```
pipeline {
  agent any
  stages {
    stage('check out SVN code') {
      steps {
        checkout([$class: 'SubversionSCM',
                  // You can add additional credentials here
                  additionalCredentials: [],
                  excludedCommitMessages: '',
                  excludedRegions: '',
                  excludedRevprop: '',
```

```
        excludedUsers: '',
        filterChangelog: false,
        ignoreDirPropChanges: false,
        includedRegions: '',
        locations: [[
            // Enter the credential ID created above
            credentialsId: '5e25f6a9-675c-4b38-97b0-e907b5fe27cd',
            // The range of code to check out
            depthOption: 'infinity',
            // Whether to check out SVN external references as w
            ignoreExternalsOption: true,
            // SVN checkout directory, which is a relative path
            local: '.',
            // SVN code repository URL
            remote: "svn://subversion.e.coding.net/StrayBirds/sv
workspaceUpdater: [$class: 'UpdateUpdater']]
    }
}
}
```

#### Step 4: Add environment variable

Add an environment variable in Variables and Caches. As the type, select Username + Password in CODING Credential.



## Step 5: Trigger build

You can choose manual build or configure a trigger method for auto building. After a successful build, you will see the following:

**Build Record #1** | **Build Process** | Build Snapshot | Modification Record

Build succeeded. Steven Manual triggered 1 day ago, during 1 minutes 48 seconds

构建过程

```

graph LR
    Start[Start] --> Checkout[Checkout 1 s]
    Checkout --> CheckOut[Check Out from Code Repository 1 s]
  
```

Check Out from Code Repository

```

1 using credential 2ed9f386-8abf-443
2 Cloning the remote Git repository
3 Cloning repository git@e.coding.net:demo/python-flask-example.git
4 > git init /root/workspace # time
5 Fetching upstream changes from git
6 example.git
7 > git --version # timeout=10
8 using GIT_SSH to set credentials
9 > git fetch --tags --force --prog
10 demo/python-flask-example.git +ref
11 > git config remote.origin.url gi
12 example.git # timeout=10
13 > git config --add remote.origin.
14 timeout=10
15 > git config remote.origin.url gi
16 example.git # timeout=10
17 Fetching upstream changes from git
18 example.git
19 using GIT_SSH to set credentials
20 > git fetch --tags --force --prog
21 demo/python-flask-example.git +ref
22 +refs/merge/*:refs/remotes/origin/
23 > git rev-parse 067ff4b6b3ae61f5d
24 Checking out Revision 067ff4b6b3ae
25 > git config core.sparsecheckout
26 > git checkout -f 067ff4b6b3ae61f
27 Commit message: "Initial commit"
28 First time build. Skipping changel
  
```

## How do I pull multiple repositories?

1. Create a code repository project token

Go to **Project Settings > Developer Options > Project Token**, click *Create Project Token*, and select **Read** in Code Repository Permissions. As we need to read two code repositories, select **Configure all the code repository permissions** in Code Repository Permissions. When you create the token, you will receive a username and password.

Settings

Project & Member

Collaboration

Project Announcement

Developer Options

☐ API Documentation  
Publish API Documentation

☐ Associate Resource  
Create, query, edit, delete

☐ Project Member  
Read and manipulate project me...

☐ Project permissions  
Read and operate the project per...

Code Repository Permission ⓘ

☒ Unified configuration all code warehouse permissions

☐ Appointed warehouse code configuration access

Repository Name	Access Permission	Operation Permission
* All Code Repositories in the Project	<input checked="" type="checkbox"/> Read Read Code Repository	<input type="checkbox"/> Read/Write Push to Code Repository

Artifact Repository Permission

☒ United configure all products warehouse permissions

☐ Specified products warehouse configuration access

Artifact Repository Name	Access Permission	Operation Permission
* all products in the project library	<input type="checkbox"/> Read Pull Artifact Repository	<input type="checkbox"/> Read/Write Pull or Push Artifact Repository

2. In the CI configuration, select **Not use** for the code source.

← customize the build process.

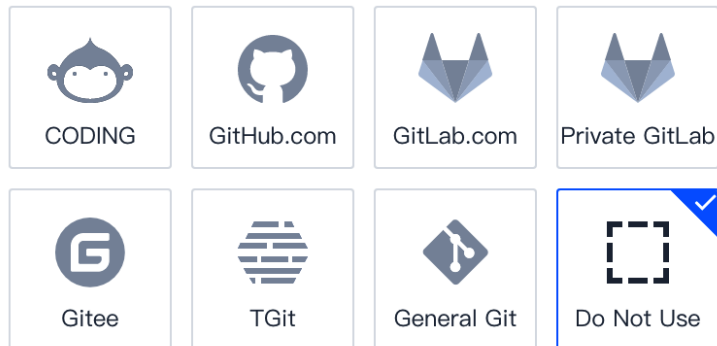
Build Job Name \*

empty-example

### Build Process

#### 1 Code Repository

Code Source



#### 2 Configuration Source

☒ Use the Jenkinsfile configured in static mode. ?

#### Jenkinsfile Preview

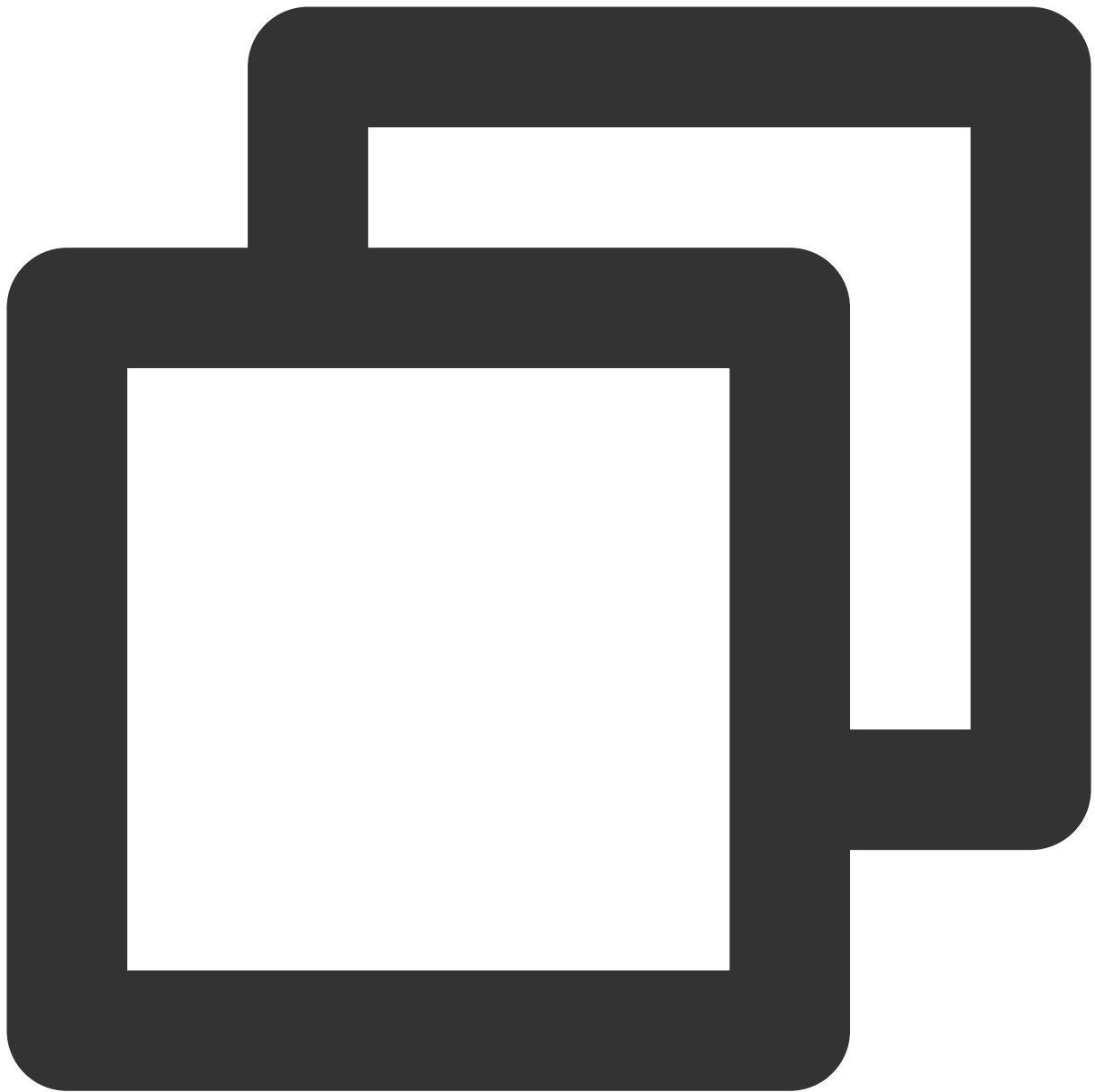
```
pipeline {
  agent any
  stages {
    stage('Custom b
      steps {
        echo "Custo
        // Please s
      }
    }
  }
}
```

☒ Go to View Configuration Details

OK

Cancel

3. Write a Jenkinsfile configuration file and enter the URLs of the code repositories to pull from.



```
pipeline {
  agent any
  stages {
    stage('checkout 1') {
      steps {
        sh 'git clone "https://${GIT_USER}:${GIT_PASSWORD}@e.coding.net/codes-farm/'
        sh 'ls -la'
      }
    }
    stage('checkout 2') {
      steps {
```

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

4. Add the username and password generated when you applied for a project token in the CI environment variables.

🔑 Environment Variable | Disc

### Process Environment Variable

Batch add string type environment variable

Add the environment variable of the build job. When the build task is manually started, it will serve as a default value of the launch parameter. [View the full help document.](#)

Variable Name	Category	Default Value
GIT_USER	String	—
GIT_PASSWORD	String	—

## How do I check out Git submodule code?

To set a submodule of a repository as the code source in a CI build plan, you must use the process configuration to check out the Git submodule repository code.

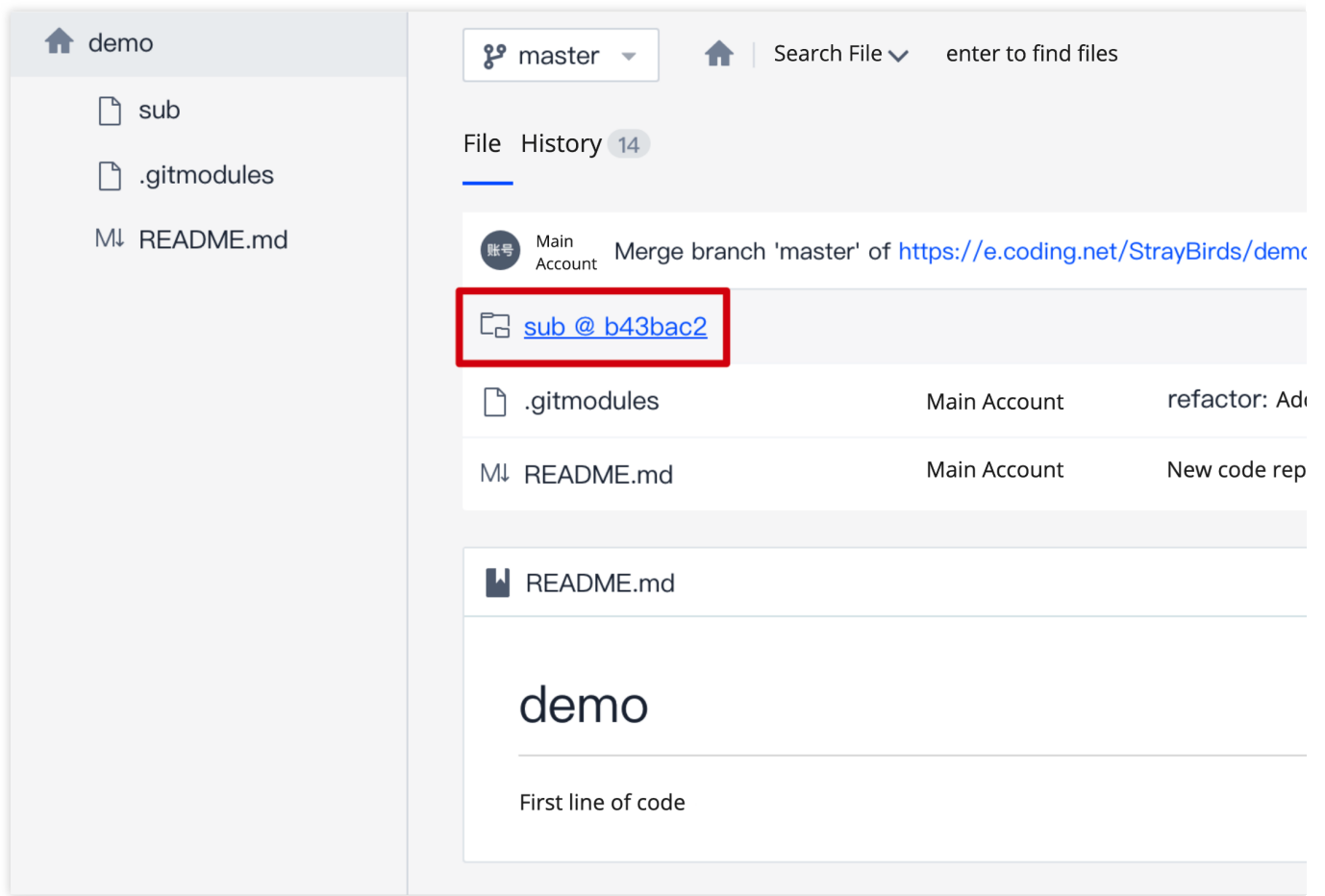
Before configuring the CI process, add the sub-repository to the parent repository. Use the `git submodule add` command to add the repository URL of the project to be tracked as a sub-repository.





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

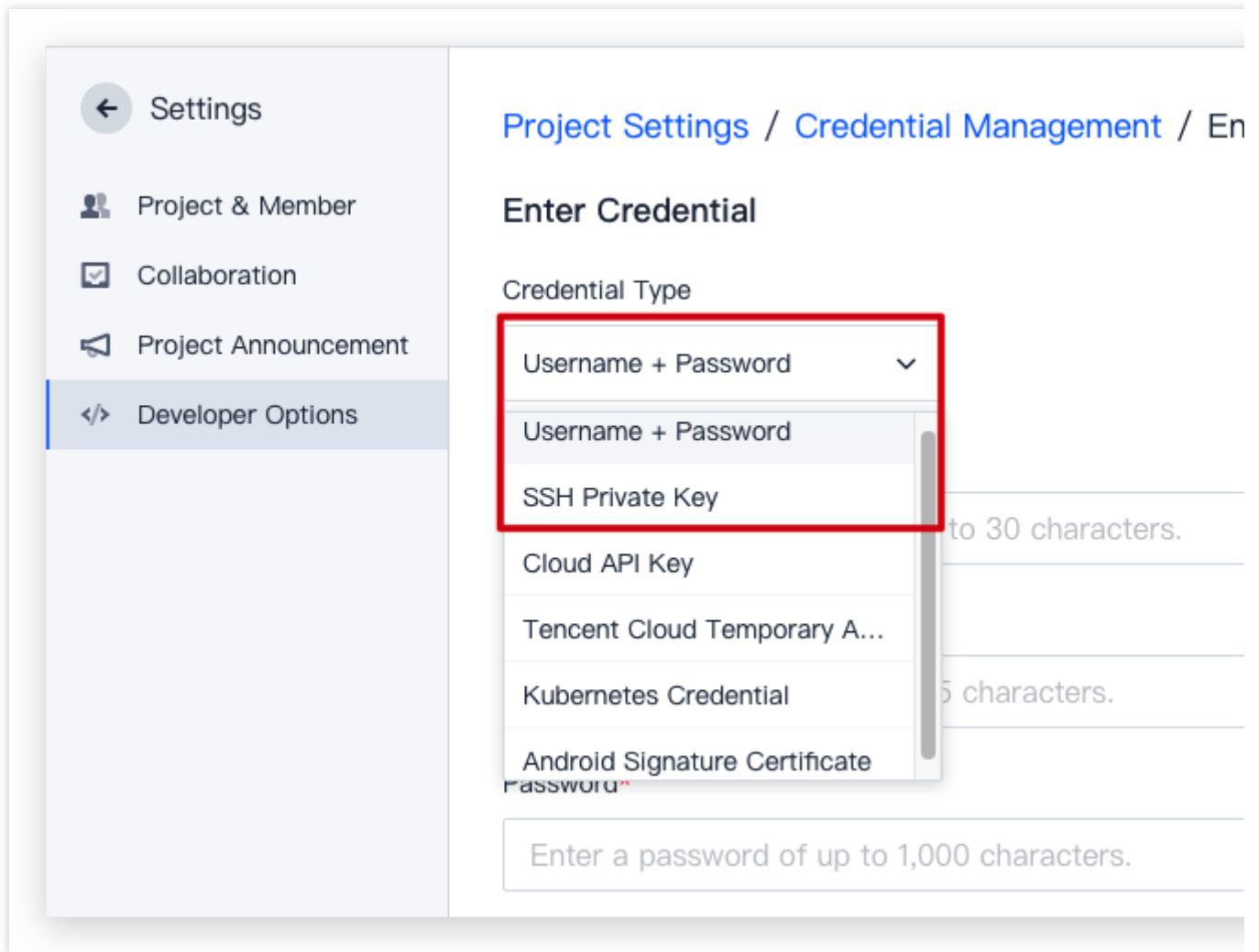
After a successful code commit, you will see this icon on the parent repository page:



### Step 1: Enter repository access credentials

Generally, the credentials for accessing a sub-repository are different from those of the parent repository. To avoid exposing sensitive information in CI configurations, you can enter the access credentials of the parent and sub-repositories in the project settings first.

1. Go to **Project Settings > Developer Options > Credential Management** and click **Enter Credential**. For the **Credential Type** select **Username + Password** or **SSH Private Key**. Under **Credential Authorization**, select **Authorize all the CI build plans**.



2. After entering the necessary information, you will receive two credential IDs.

Developer Options

API and Event

Project Token

Service Hook

Credential

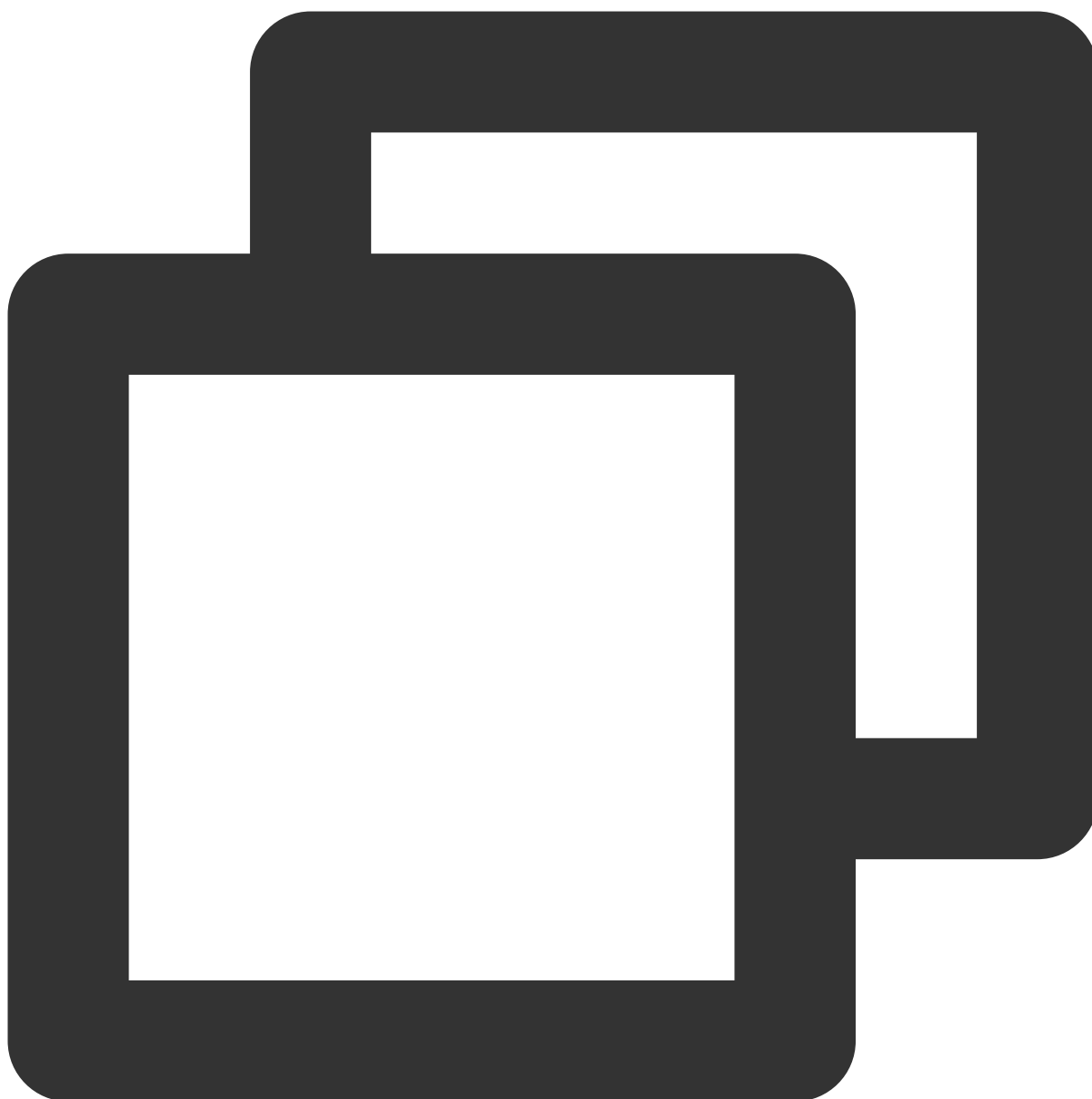
Credential Management (5)

Storing passwords, private keys, and certificates into credential management maximizes the credential security of continuous integration and deployment components, you can select entered credentials to use.[View console](#)

Certificate Name	Authorized	Credential ID	Certificate Description
tcr-artifacts	Services not authorized yet	de900c2a-f57f-4f0b-9bc8-ae376cd5af70	-
tcr-artifacts	1	82589fd1-6a52-43cb-8674-4d6f5bc06cad	-

## Step 2: Configure CI process

Refer to the following Jenkinsfile configuration:



```
pipeline {
  agent any
  stages {
    stage('check out') {
      steps {
        checkout([
          $class: 'GitSCM',
          branches: [[name: GIT_BUILD_REF]],
          doGenerateSubmoduleConfigurations: false,
          // Configure submodule checkout rules here
          extensions: [[
```

```

    $class: 'SubmoduleOption',
    // Whether to prohibit submodule checkout
    disableSubmodules: false,
    // Whether to allow the use of parent project user credentials for
    parentCredentials: false,
    // Whether to recursively check out all submodule updates
    recursiveSubmodules: true,
    // Specify the reference repository path
    reference: '',
    // Whether to track the latest commits to the branch configured in
    trackingSubmodules: false
  ]],
  submoduleCfg: [
  ],
  // Configure the remote parent project and submodule checkout info
  userRemoteConfigs: [
  [
    // Configure the remote parent project repository SSH credentials
    credentialsId: '93207d20-****-****-****-410850900d86',
    url: 'https://e.coding.net/StrayBirds/Parent/parent.git'
  ],
  // Configure the remote submodule repository SSH credentials and
  [
    credentialsId: '560bdc1e-****-****-****-c8e3ccb3ccc6',
    url: 'https://e.coding.net/StrayBirds/Submodule/sub.git'
  ],
  // If there are more submodules, add their configurations here
  ]
  ])
}
}
}
}
}

```

After successful operation, the log will read as follows:

The screenshot displays the Tencent Cloud CODING CI interface. On the left is a navigation menu with options: Overview, Collaboration, Repository, Code Scanner (beta), CI, Build Job (selected), Build Node, CD, Artifact Management, Test Management, Document, and Settings. The main area shows 'Build Record #7' with tabs for 'Build Record #7', 'Build Process', and 'Build Snapshot'. A green checkmark indicates 'Build succeeded.' with a status '65' and a note 'Manually triggered by GP199513' and 'Trigger at 20 minutes ago, Duration 54 s'. Below this, the 'Build Process' section shows a flow from 'Start' to 'Check Out' (1 s), which is further detailed as 'Check Out from Code...' (1 s). On the right, a terminal window shows a series of git commands and their outputs, including cloning a repository and checking out a specific commit.

## How do I check out code repositories from other projects?

During CI, you can use project tokens to check out code from CODING repositories in other projects.

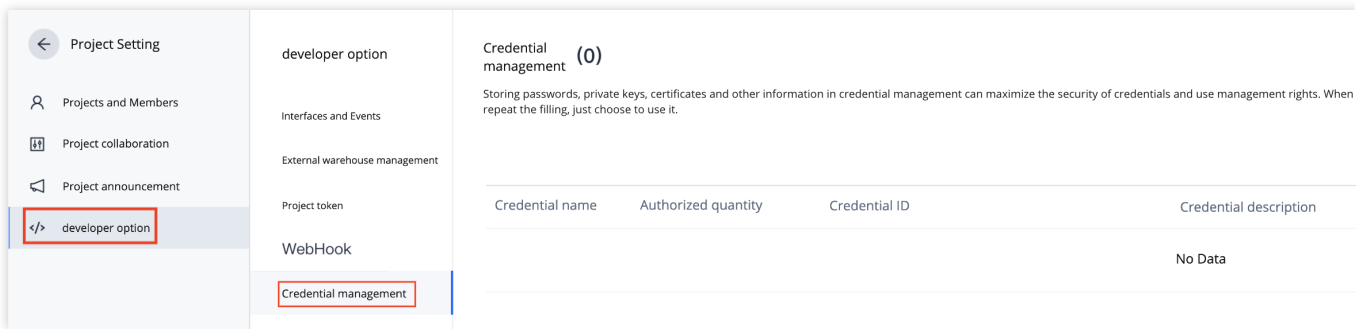
In this example, we will use two different projects:

"Project A" is the project that contains the code repository that we will need to check out.

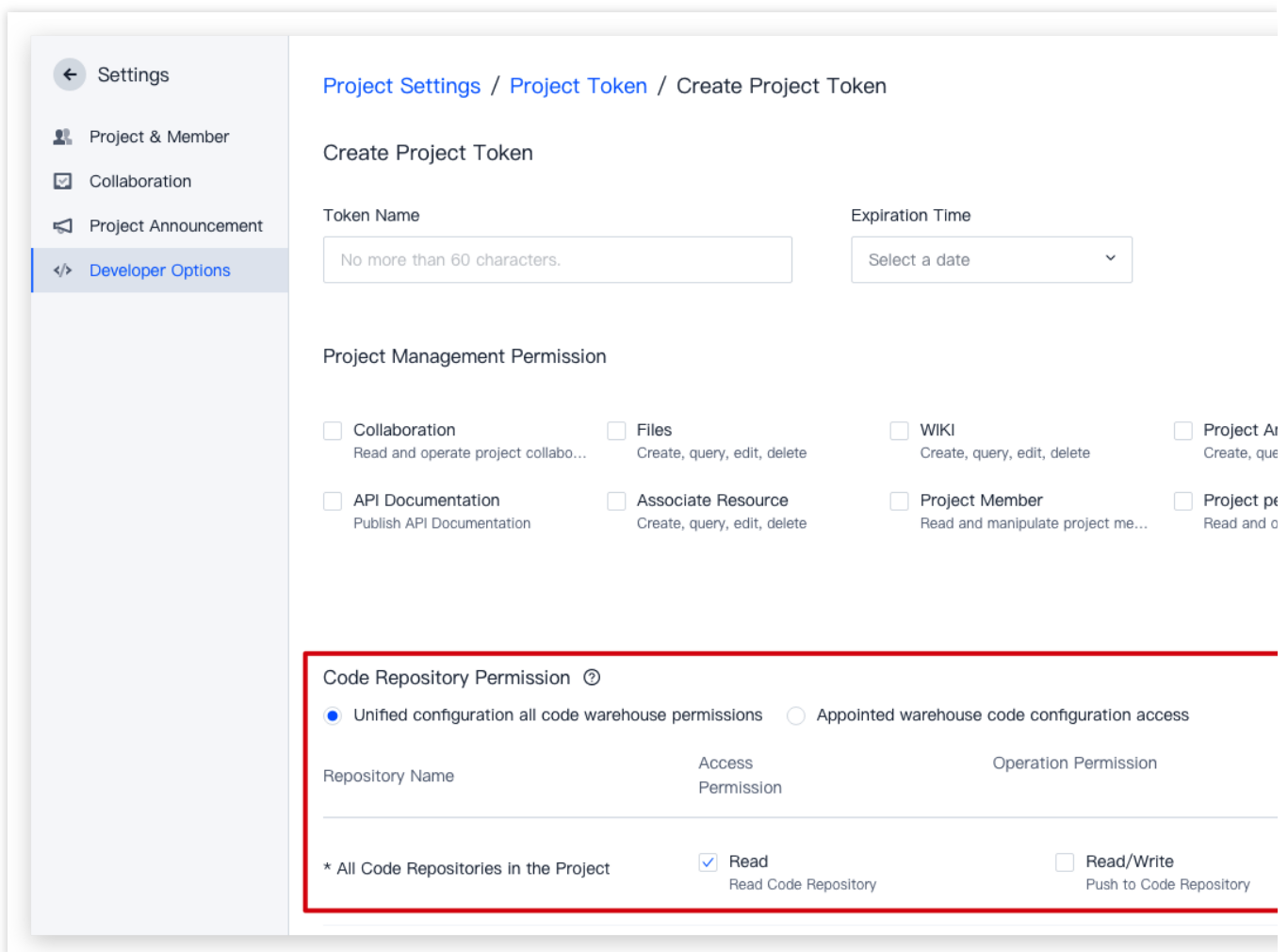
"Project B" is the project that contains the CI checkout task.

### Step 1: Create project token in Project A

1. Open Project A, go to **Project Settings > Developer Options > Project Token**, and click **Create Project Token**.



2. Select the code repository for checkout and configure the necessary operation permissions.

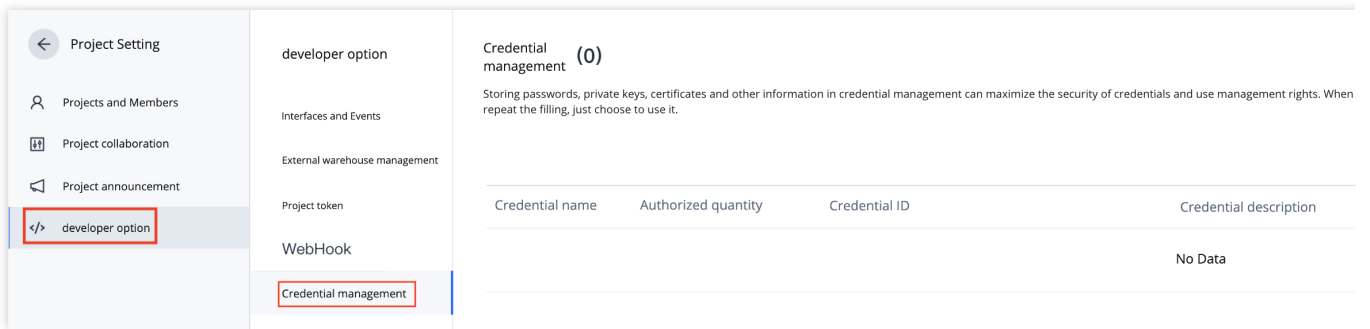


3. Click **OK** to create the token.

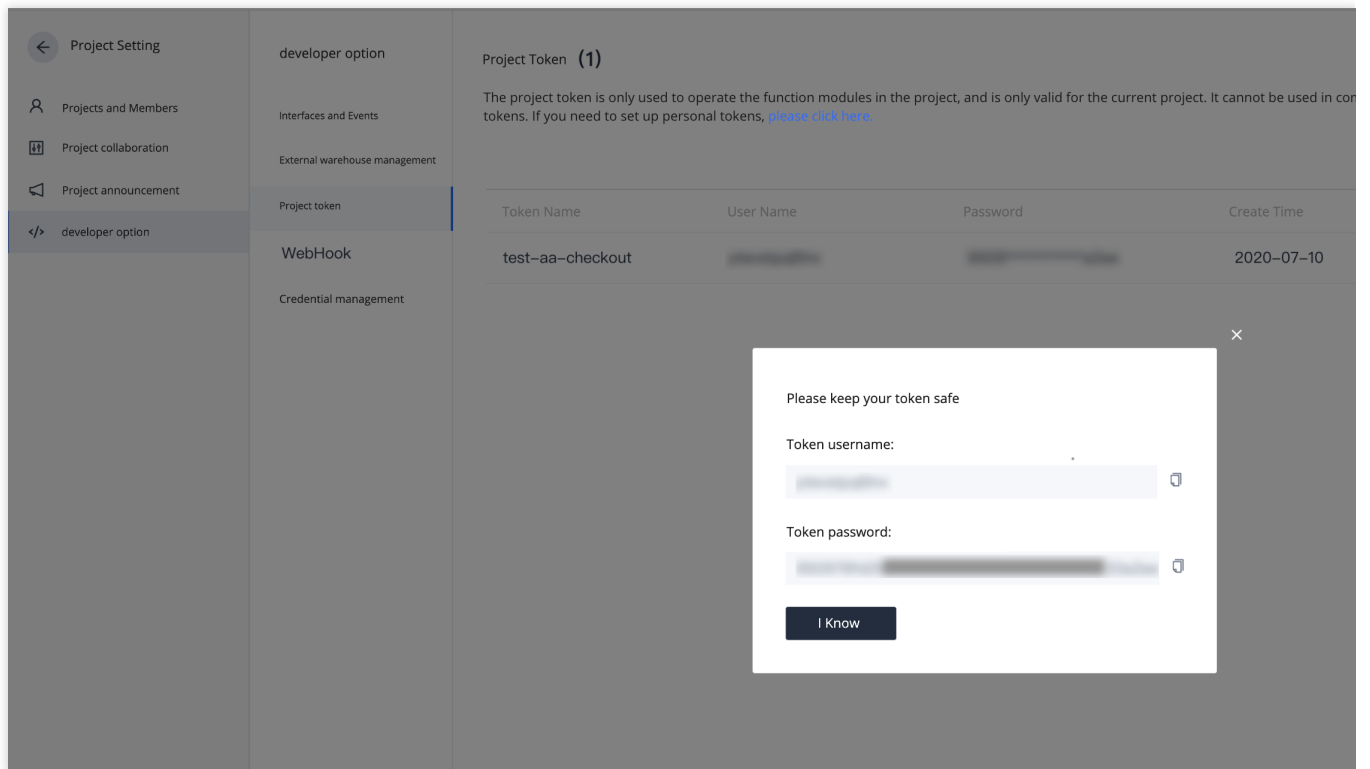
## Step 2: Create credentials in Project B

1. Open Project B, go to **Project Settings > Developer Options > Credential Management**, and click **Enter Credential**.





2. Go back to the page of the token created for Project A and click **View Password**.



3. In the **Enter Credential** window for Project B, select **Username + Password** as the **Credential Type** and paste the corresponding project token information.

← Project Setting

👤 Projects and Members

👥 Project collaboration

📢 Project announcement

</> developer option

Project Settings / Credential Management / Entry Credential

Entry Credentials

Credentials Type

Username + Password

Credentials Name \*

checkout-test-dd

Username \* Token Username

Password \* Token Password

Credential description

Please enter a credential description, no more than 100 characters

4. Select the CI project to authorize and click **Save**.

## Credential authorization

The selected build plan will have permission to

### Continuous Integration

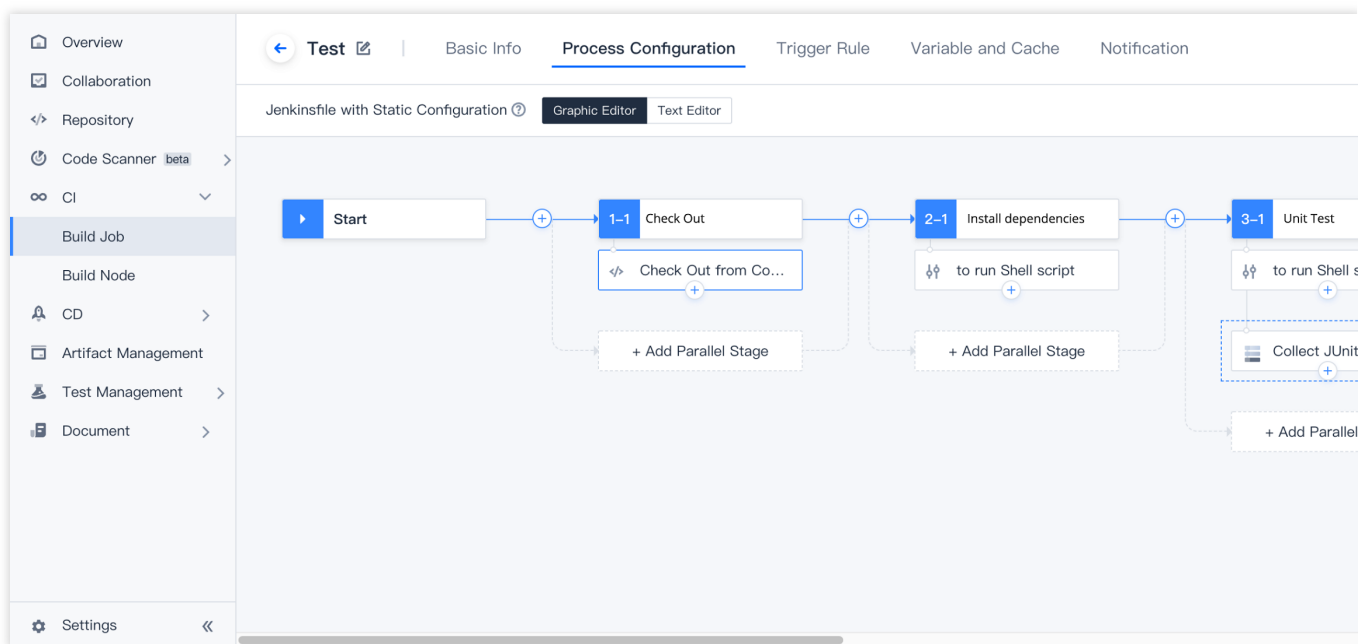
☒ Authorize all continuous integration build plans ?

☒ empty-example

<<SaveCancel

### Step 3: Configure corresponding environment variables in CI task in Project B

1. Go to CI Settings > **Process Configuration**, add a **Check out from code repository** step, and click **Environment Variable**.



After adding a checkout process, you can also go to CI Settings > **Variables and Caches** and click **Add Environment Variable**.

**Variable and Cache**

**Process Environment Variable** [+ Env Variable](#)

Add the environment variable of the build job. When the build task is manually started, the environment variable will serve as the default value of the launch parameter. [View the](#)

Variable Name	Category	Default Value	Operation
DOCKER_IMAGE_NAME	String	nodejs-express-app	<a href="#">Edit</a> <a href="#">Delete</a>
DOCKERFILE_PATH	String	Dockerfile	<a href="#">Edit</a> <a href="#">Delete</a>
DOCKER_BUILD_CONTEXT	String	.	<a href="#">Edit</a> <a href="#">Delete</a>
DOCKER_REPO_NAME	String	test	<a href="#">Edit</a> <a href="#">Delete</a>
DOCKER_IMAGE_VERSION	String	\${GIT_LOCAL_BRANCH:-branch}-\${GI...	<a href="#">Edit</a> <a href="#">Delete</a>

**Cache Directory**

- Enabling cache can avoid repetitive download of the dependency files in each build, greatly improving the build speed.
- If an error occurs on your build cache, reset the cache.
- You are advised to enable cache for Maven, Gradle, and npm cache directories.

[Reset Cache](#)

2. Add the following two environment variables:

Variable	Default Value
GIT_REPO_URL	Clone URL of the repository to be checked out (HTTPS)
CREDENTIALS_ID	The credential ID entered in <a href="#">Step 2</a>

GIT\_REPO\_URL

**vue-cos** [Browse](#) [Commit](#) [Branch](#) [Merge Request](#) [Version](#) [Compare](#) [Settings](#)

[vue-cos](#)

[master](#) [enter to find files](#)

**File History** 5

no message

public Dian Yu

src Dian Yu customized template

**GIT\_REPO\_URL** [init](#)

The warehouse address that needs to be checked out

**Clone the repository** When cloning the repository or on the command line, use the following command:

**HTTPS**

CREDENTIALS\_ID

Project Setting

developer option

Interfaces and Events

External warehouse management

Project token

WebHook

Credential management

Credential management (0)

Storing passwords, private keys, certificates and other information in credential management can maximize the security of credentials and use management rights. When repeat the filling, just choose to use it.

CREDENTIALS\_ID

Credential name	Authorized quantity	Credential ID	Credential description
check-out-test-dd	1	[Redacted]	-

The environment variables have been entered:

Overview

Collaboration

Repository

Code Scanner beta

CI

Build Job

Build Node

CD

Artifact Management

Test Management

Document

Settings

Test

Basic Info

Process Configuration

Trigger Rule

Variable and Cache

Notification

Jenkinsfile with Static Configuration

Graphic Editor

Text Editor

Start

1-1 Check Out

2-1 Install dependencies

Check Out from Co...

to run Shell script

+ Add Parallel Stage

+ Add Parallel Stage

Process Environment

Add the environment variable as a default value

Variable Name

GIT\_REPO\_URL

CREDENTIALS\_ID

#### Step 4: Start build task and check out code

The screenshot displays the Tencent Cloud CODING CI interface. On the left is a navigation menu with options: Overview, Collaboration, Repository, Code Scanner (beta), CI, Build Job (selected), Build Node, CD, Artifact Management, Test Management, Document, and Settings. The main area shows 'Build Record #7' with tabs for 'Build Record #7', 'Build Process', and 'Build Snapshot'. The 'Build Record #7' tab indicates 'Build succeeded.' with a green checkmark and a duration of 65 seconds. The 'Build Process' tab shows a workflow diagram with steps: 'Start' (green flag icon) and 'Check Out' (green checkmark icon, 1 s). A tooltip for the 'Check Out' step shows 'Check Out from version 1 s'. On the right, a terminal window displays the following commands and output:

```

1 using credential felbc98c-9e90
2 Cloning the remote Git repository
3 Cloning repository https://
4 > git init /root/workspace #
5 > git --version # timeout=30
6 > git fetch --tags --progress
7 using GIT_ASKPASS to set credentials
8 > git fetch --tags --progress
9 > git config remote.origin.url
10 > git config --add remote.origin.url
11 > git config remote.origin.url
12 Fetching upstream changes from
13 using GIT_ASKPASS to set credentials
14 > git fetch --tags --progress
15 Seen branch in repository origin
16 Seen 1 remote branch
17 > git show-ref --tags -d # ti
18 Checking out Revision b737a93c
19 > git config core.sparsecheckout
20 > git checkout -f b737a93c244
21 Commit message: "no message"
22 First time build. Skipping cha
  
```

## How do I check out from a repository using Git LFS?

During CI, you can use process configuration to check out code from a repository managed by the Git Large File Storage (LFS) plugin. This allows CI with Git repositories containing large files.

### Introduction to Git LFS

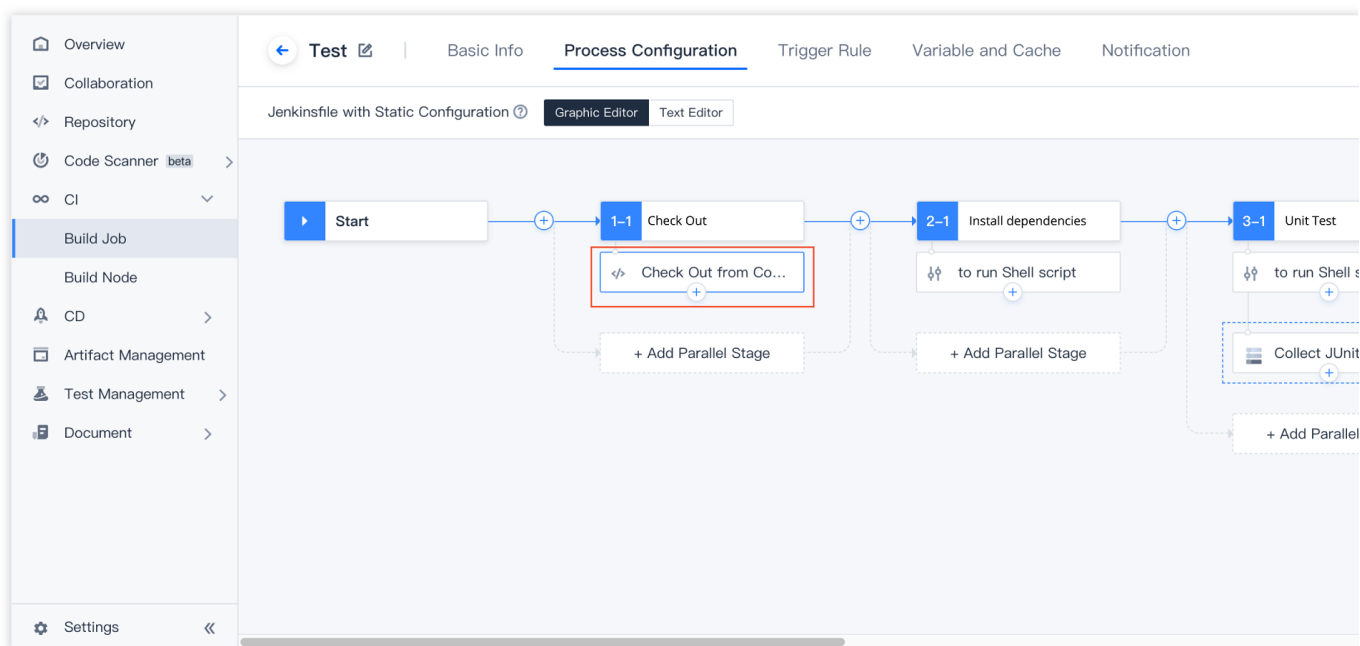
The Git LFS plugin accelerates `git clone` and `git fetch` operations that involve frequently changed large files (such as images and videos).

Each time you add large files to the repository, the Git LFS plugin will store the files in the local Git LFS cache and replace large file content in the code repository with references to the cache address. When you commit code, all large files involved in the commit are committed to the remote Git LFS cache, which is associated with your remote repository. When you check out commits that reference large files, the plugin will replace the references with the actual file content from the cache.

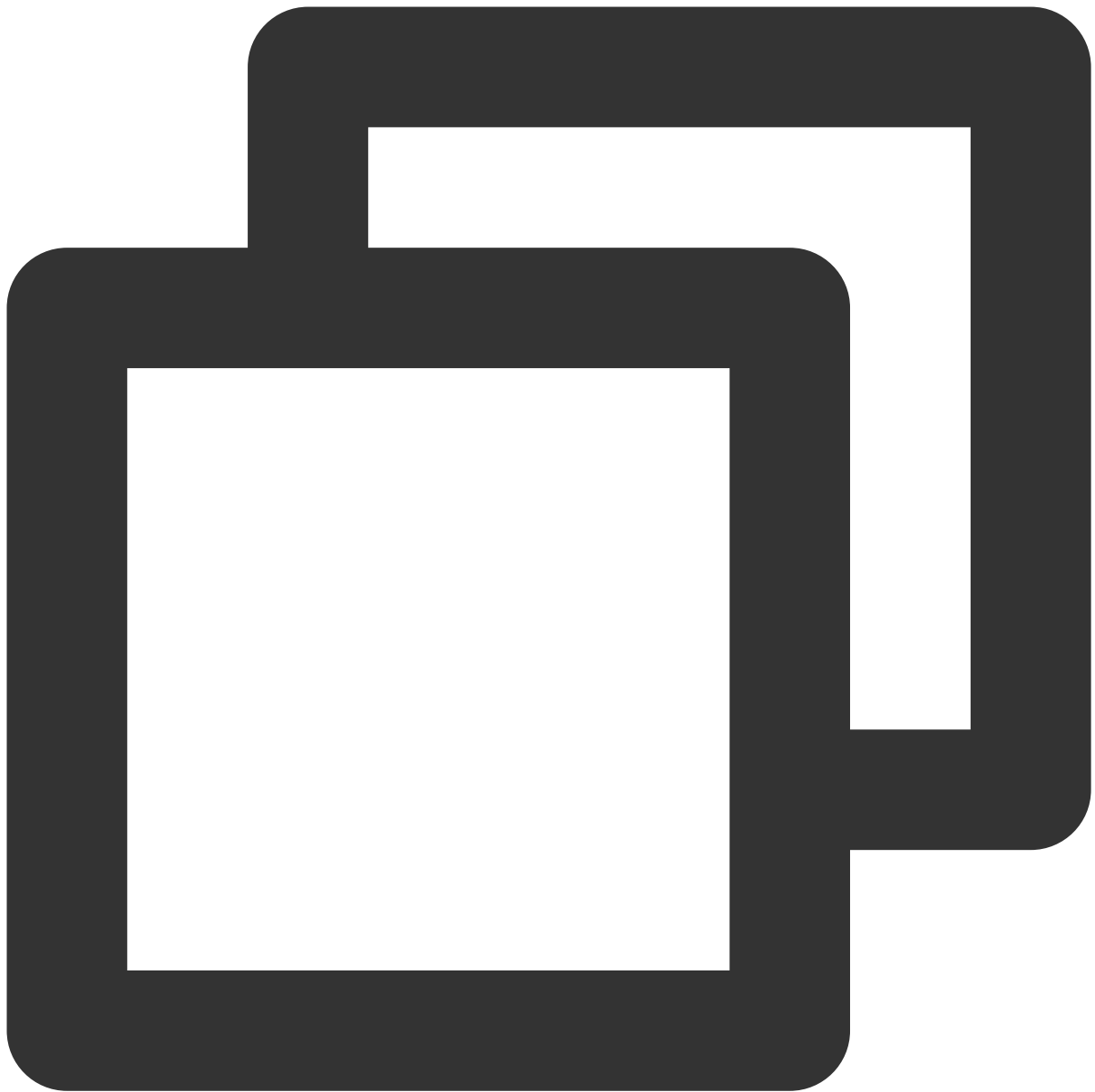
Therefore, when using the Git LFS plugin, large files are only loaded for `git checkout`.

### How do I check out code from a build plan?

Go to **Build Plan Settings > Process Configuration**, click **Check out from code repository** to add this step, and then add the Git-LFS-Pull plugin.



## Jenkinsfile



```
pipeline {  
  agent any  
  stages {  
    stage('check out') {  
      steps {  
        checkout([  
          $class: 'GitSCM',  
          branches: [[name: env.GIT_BUILD_REF]],  
          extensions: [  
            // Add GitLFSPull plugin  
            [$class: 'GitLFSPull'],  
          ],  
        )  
      }  
    }  
  }  
}
```



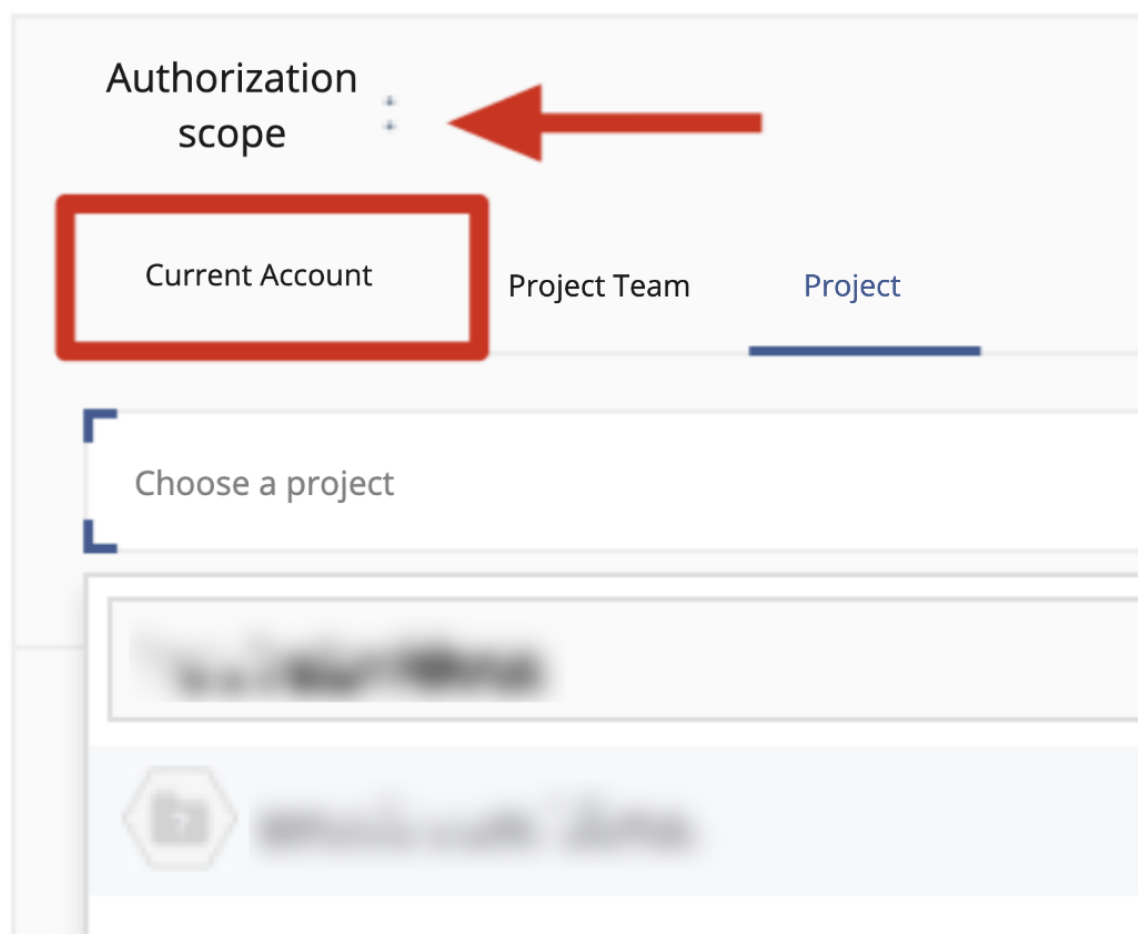
```
    ],
    userRemoteConfigs: [[
      url: env.GIT_REPO_URL,
      credentialsId: env.CREDENTIALS_ID
    ]]
  ])
}
}
```

### Why can't I sync associated TGit repositories to the external repository list?

Currently, you must select **Current Account** as the authorization scope during TGit authorization in order to sync these repositories to the external repository list and check them out in a CI build task. Repositories with **Project Group** or **Project** authorization scopes cannot be synced.



Authorize **CODING DevOps** to use **Coding.net** account.

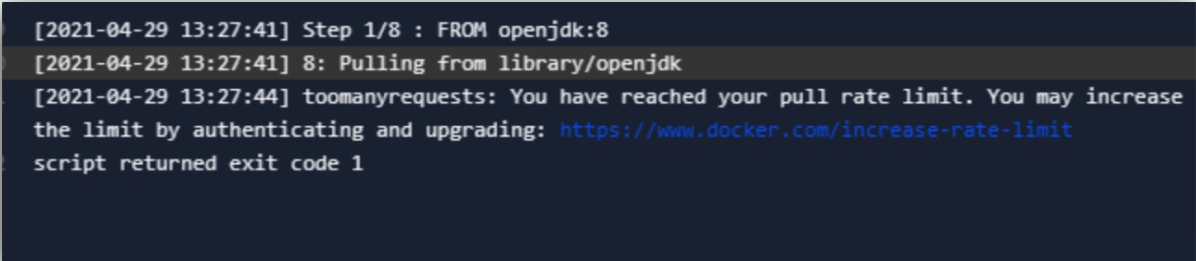


# Continuous Integration and Artifact Repositories

Last updated : 2023-12-29 11:44:51

## Why does the system return the error `reached your pull rate limit` ?

When pulling images using CI, you may be prompted with the error `reached your pull rate limit` , as shown below:



```
[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
```

This occurs when users with the trial version of Docker Hub reach their image pull limit, due to the CODING egress IP address reaching the Docker Hub pull limit. Use one of the two methods below to solve this problem:

Host images in the CODING Docker artifact repository. For details, see [Docker Artifact Repository](#).

Use your personal Docker Hub account.

If you don't have a Docker Hub account, you can [sign up](#).

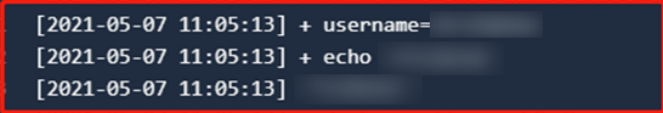
After signing up for an account, modify the build plan configuration by adding this line and entering the account before executing commands in Docker.



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

During execution, you can view the current Docker Hub account in the log. An account that has not reached its pull limit will not have this problem.

```
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.docker.com/engine/reference/commandline/login/#credentials-store
6 [2021-05-07 11:05:12]
7 [2021-05-07 11:05:12] Login Succeeded [REDACTED]
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]
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
22 [2021-05-07 11:05:18] cc62143cb8cc: Pulling fs layer
```



# Custom Build Nodes

Last updated : 2023-12-29 11:44:51

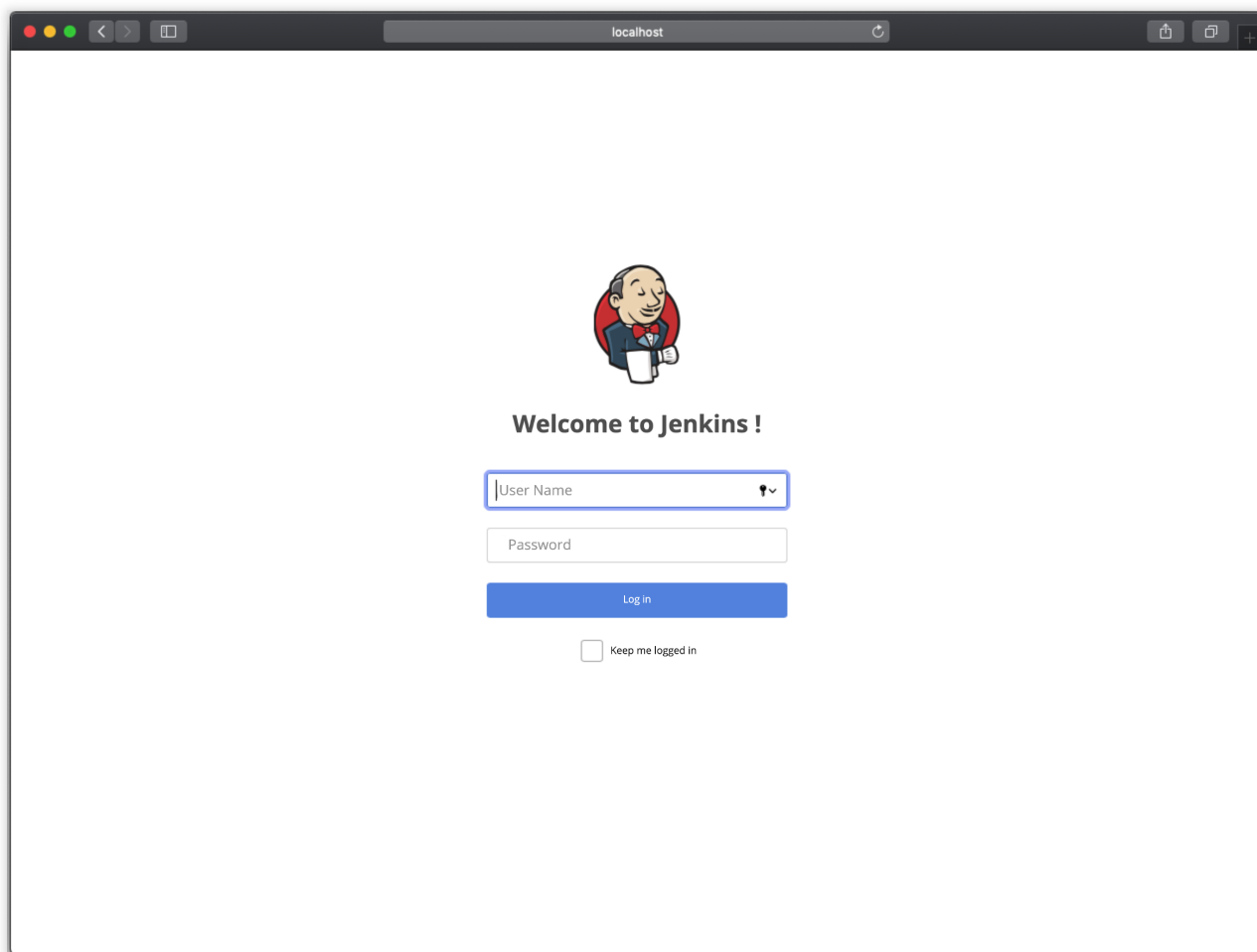
## How do I access a local Jenkins instance in a custom build node?

### Step 1: Access Jenkins

1. First, you must set your device as a custom build node access point.
2. To avoid exposing ports, Jenkins instances started by CODING-CI custom nodes only listen to the local loopback address (127.0.0.1) by default. The default listener port is `15740` . In this case, you can only access Jenkins from the build node machine through localhost or 127.0.0.1. The specific access address is `<http://localhost:15740>` .
3. If you cannot access Jenkins, run the command `cat ~/.coding/cci-agent.yml` to see the port (publicPort).
4. If you want to access Jenkins from outside the build node, run the `up` command to launch the program and add the `--jserver 0.0.0.0` parameter. At the same time, you can use `--jport` to specify the listening port. Assuming the build node IP address is `NODE_IP` and the listening port is `PORT` , the access address will be `<http://NODE_IP:PORT>` .

### Step 2: Jenkins login token

Enter the Jenkins access address in your browser to view the login page.



Here, we assume the Jenkins username and password are `coding` and `11bf48c0403ec88231b530b5f98a113cad`. You can run the `./cci-agent up -h` command.

```

Click here to configure status bar
[ ] - node-13.13.0
$ ./cci-agent up -h
Start the build node related programs

Usage:
  cci-agent up [flags]

Flags:
  --clear                Force kill on startup and delete legacy tasks
  --de stringArray       Set Jenkins container environment variables
  -d, --detach           Running client programs in the background
  --dv stringArray       Set the Jenkins container to specify the mount volume
  -h, --help            help for up
  --jcredential string   Jenkins login token (username:password) (default "coding:11bf48c0403ec88231b530b5f98a113cad")
  --jno-proxy-host stringArray Jenkins Hosts not going through the proxy
  --jport string         Jenkins server port number (default "15740")
  --jproxy-password string Jenkins Proxy server password
  --jproxy-port string   Jenkins Proxy server port number
  --jproxy-server string Jenkins Proxy server domain name (without protocol prefix)
  --jproxy-test-url string Jenkins Proxy server test address
  --jproxy-username string Jenkins Proxy server username
  --jserver string       Jenkins Server listening address, If you want to monitor all addresses, please specify this parameter as 0.0.0.0 (default "127.0.0.1")
  --remove              Delete the built workspace
  --update-jproxy        Whether to update the Jenkins agent

Global Flags:
  --config string  Specify the directory where the configuration file is located (default is $HOME/.coding)
  --insecure       Chain connection is not established via Transport Layer Security (TLS)
  -p, --port string server port number
  --pt string      A project token with read and write permissions to the node pool
  -s, --server string Server domain name (without protocol prefix) (default "cci-websocket.coding.net")
$

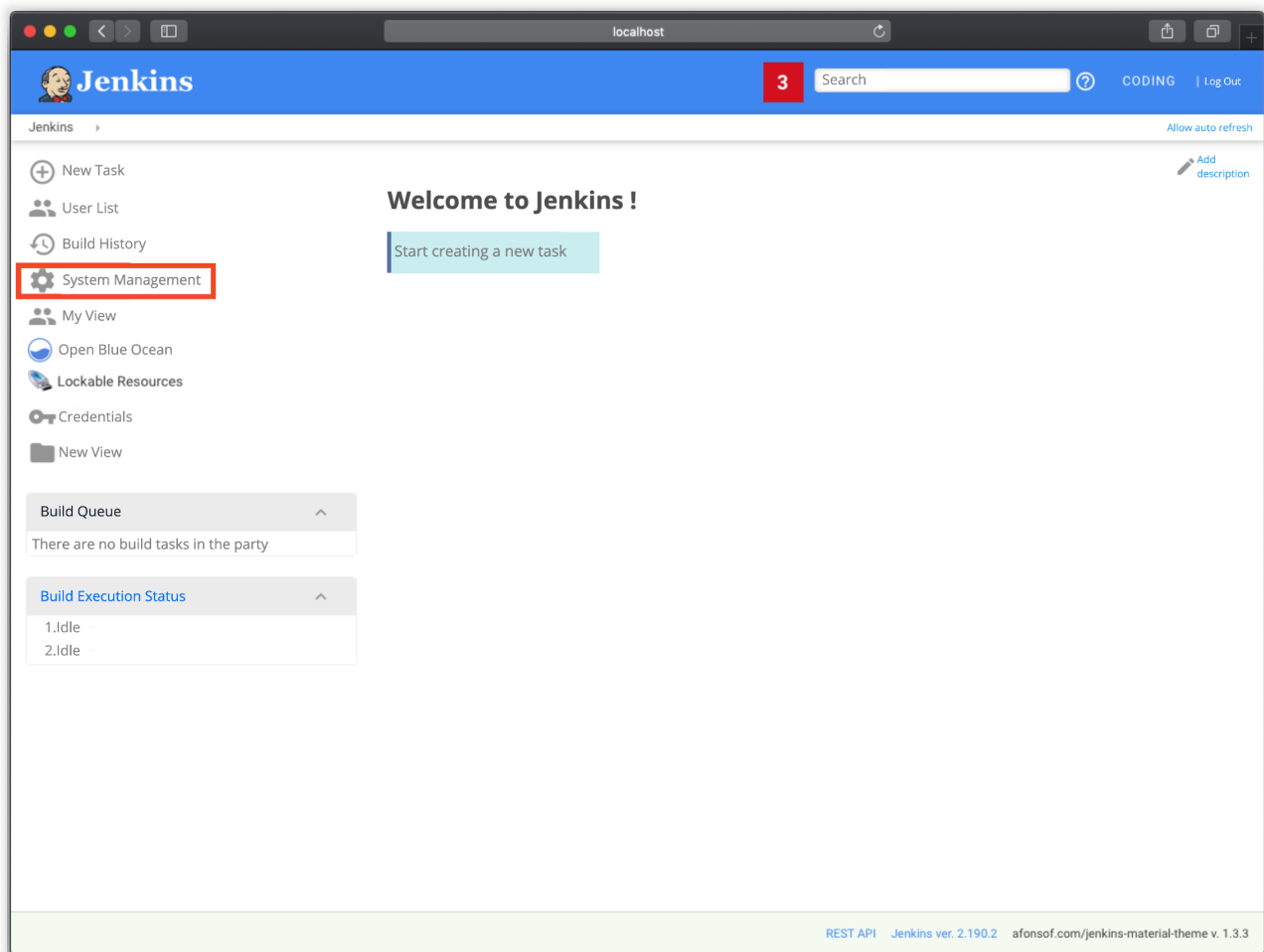
```

## How do I install plugins in custom build nodes?

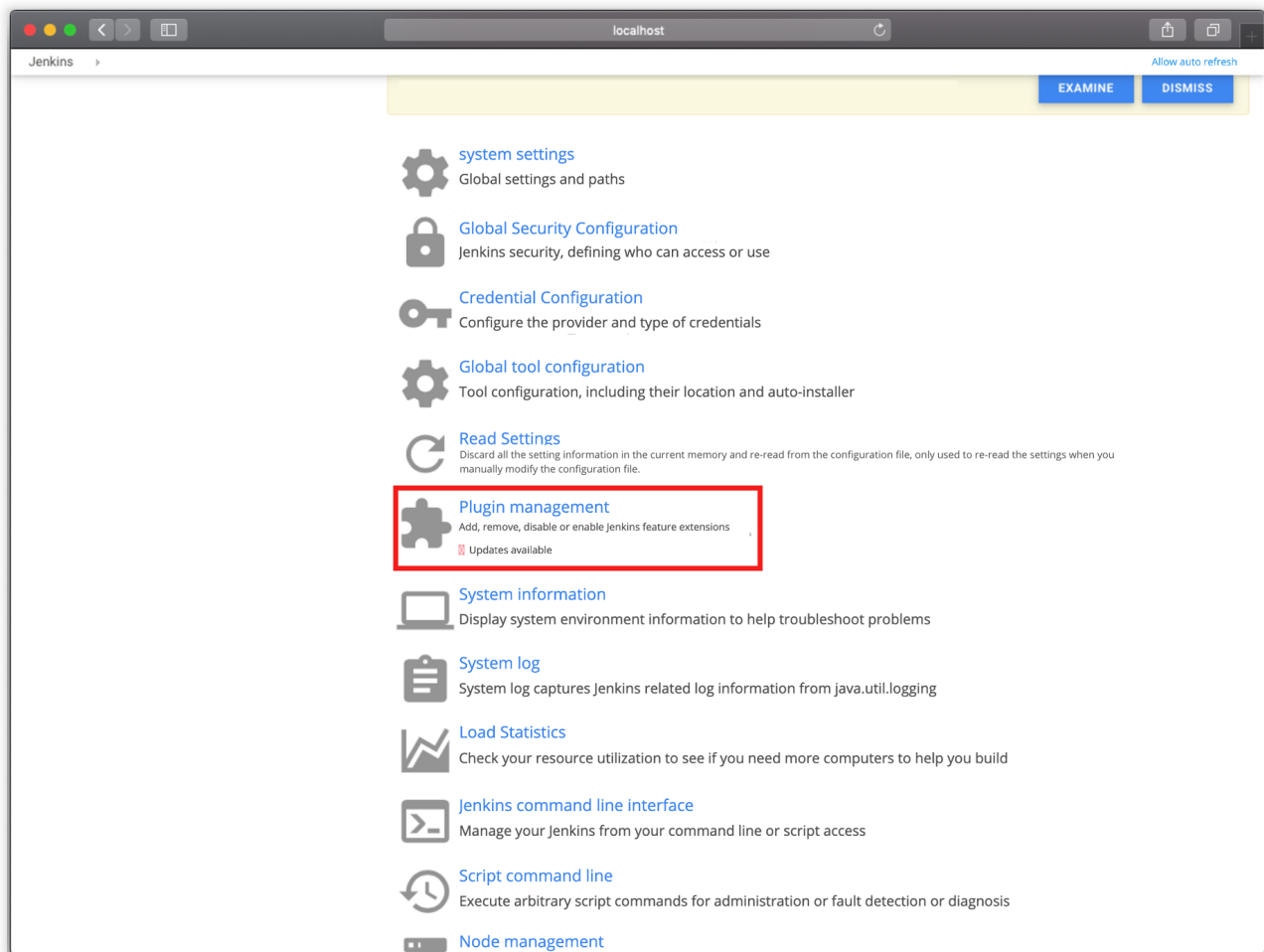
Custom build nodes that use the open-source software [Jenkins](#) as their build engines can use the over 1,000 plugins Jenkins provides for build, deployment, and automated operations. By default, CODING CI custom build nodes only have the most common Jenkins plugins. You can manually install other plugins as needed.

1. First, sign in to Jenkins.
2. After you sign in to Jenkins, you will see the Jenkins management interface. Go to **Manage Jenkins > Manage Plugins**.
3. In the menu on the left, click **Manage Jenkins**.

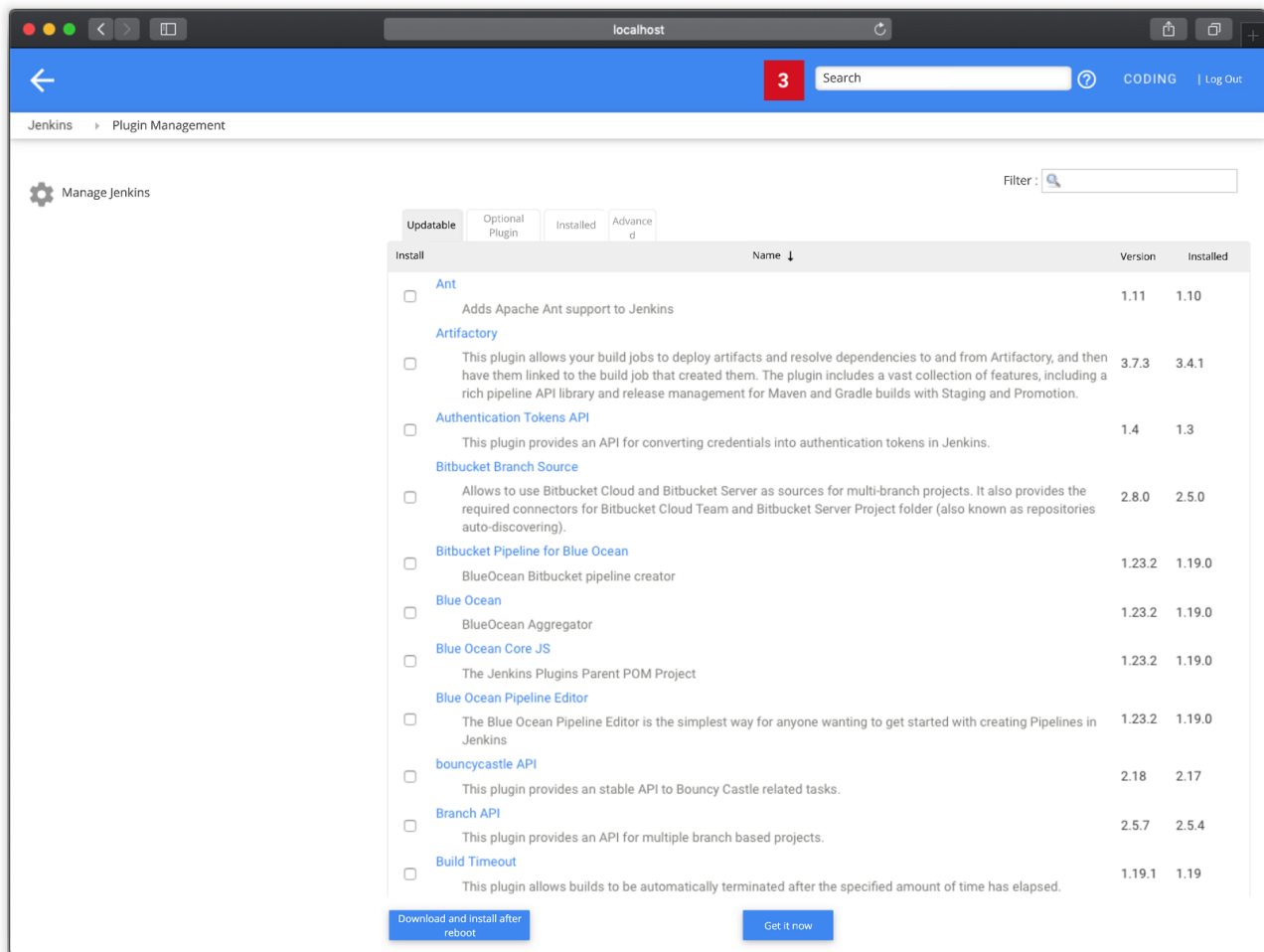




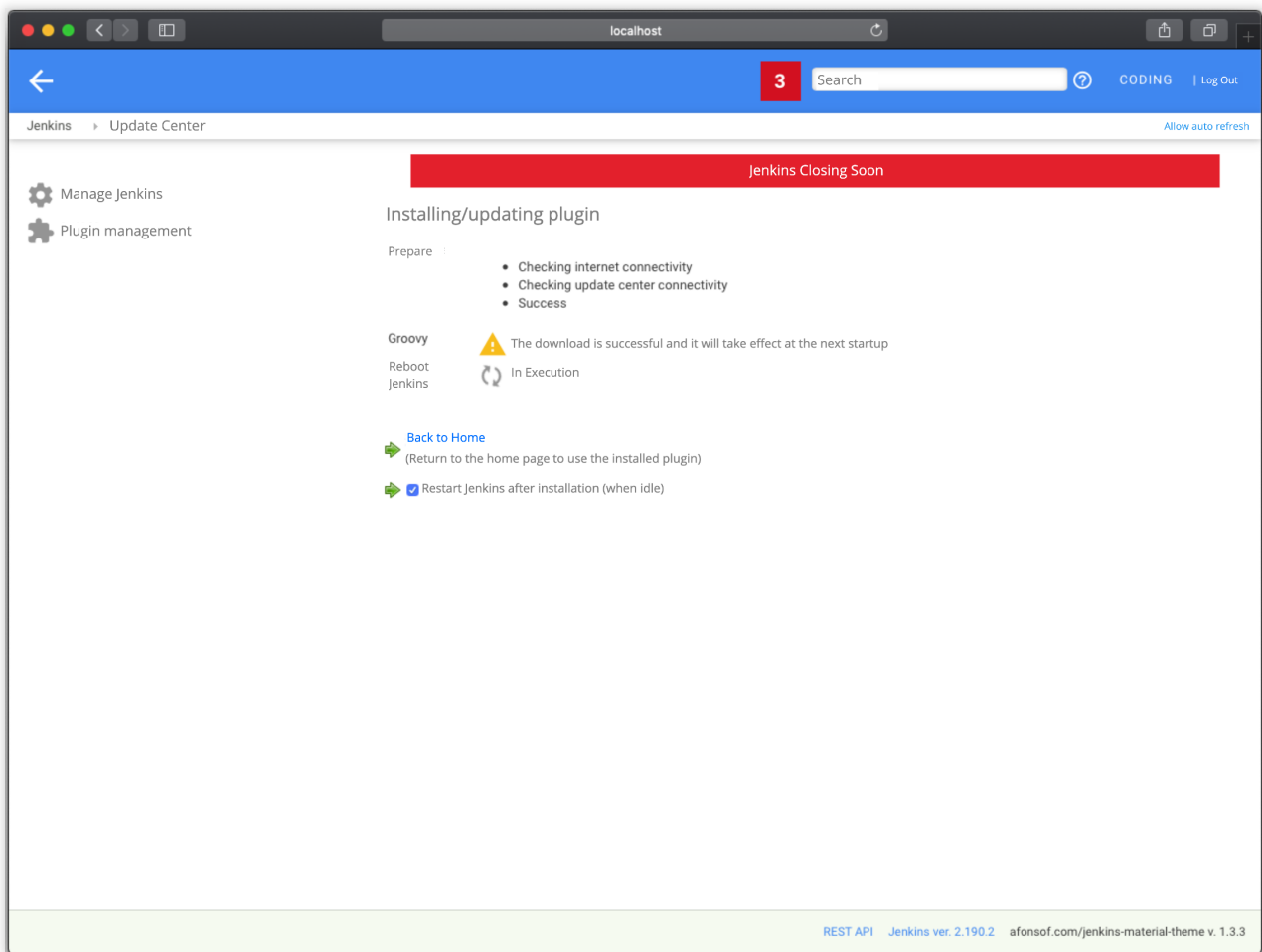
4. On the page, select **Manage Plugins**.



5. Open the Plugin Manager page.



6. On the Plugin Manager page, find the **Available** tab, select the plugins to install, and click **Download now and install after restart** at the bottom. On the **Update Center** page that pops up, select **Restart Jenkins when installation is complete**. Wait for Jenkins to install the plugins and automatically restart. You can now use the plugins.



## How do quotas work for custom build nodes?

Access custom nodes are not limited by CODING team quotas.

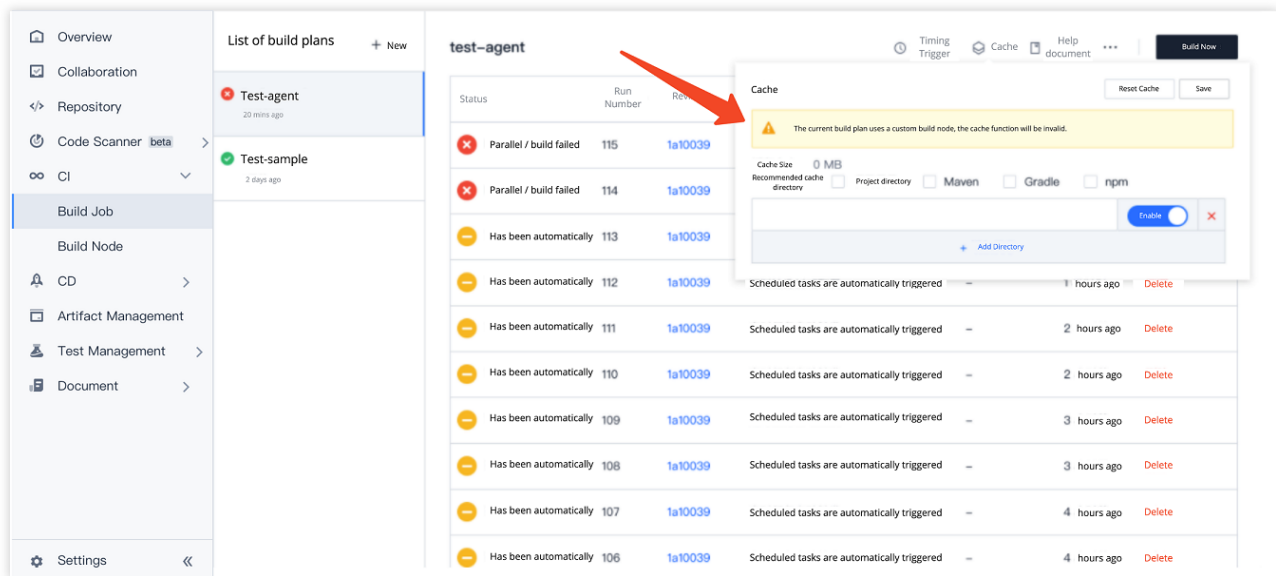
Custom build nodes are not counted towards the team's weekly build quota.

Custom build nodes are not limited by the team's parallel operation quota.

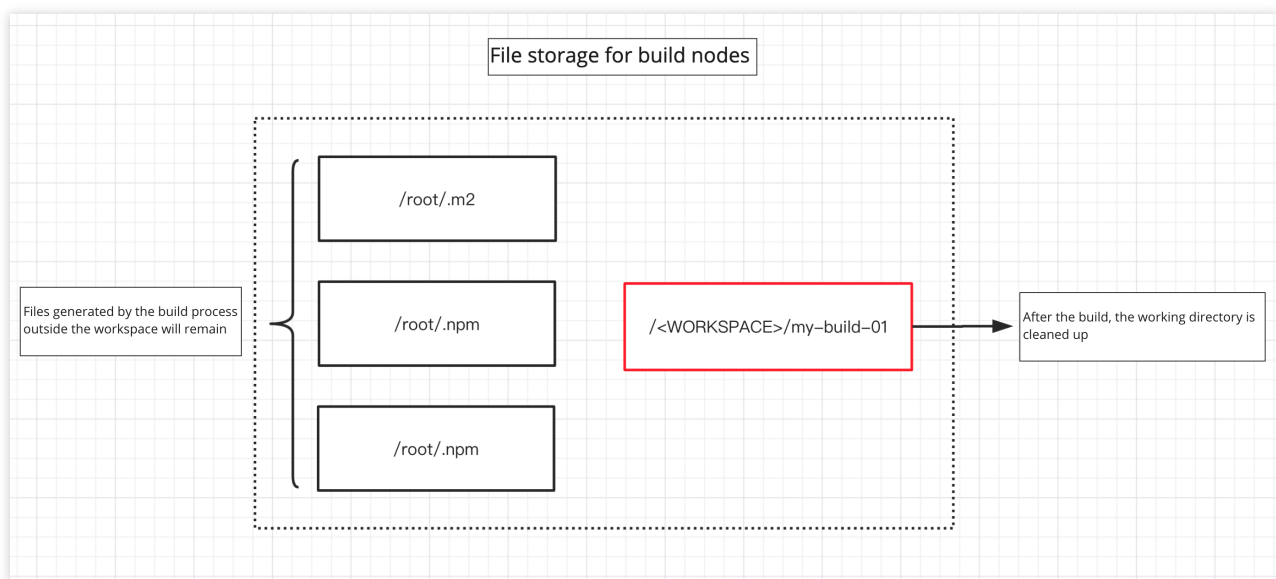
Custom build nodes are not limited by timeout quotas.

## How do caches work for custom build nodes?

A build plan configured with a custom build node pool will use the build node's own cache.



For a build plan executed by custom build nodes, each execution creates an independent `Workspace`, which is cleared after the build is completed. Files produced outside of the workspace during the build process are retained (such as files in the global cache for maven, npm, and other artifact repositories).



## How do I re-register cci-agent?

Under normal conditions, if you repeatedly register cci-agent on a machine, you will be prompted that the node is already registered, and you must delete the registered node before registering it again.

```
1. mac@P_XINZCHEN-MC0: /tmp
→ /tmp ./cci-agent init
INFO[2020-04-10T10:38:30+08:00] Connection succeeded.
Initialization failed, the current build node is already registered

Registration Message
-----
The team's domain name : https://[redacted] coding.net
Project Name : test-generic-2
Owning build node pool : default
-----

To re-register the node, you can go to https://[redacted].coding.net/p/test-generic-2/ci/agent/222 Delete the registered node or execute it on the
/ci-agent remove current build node.

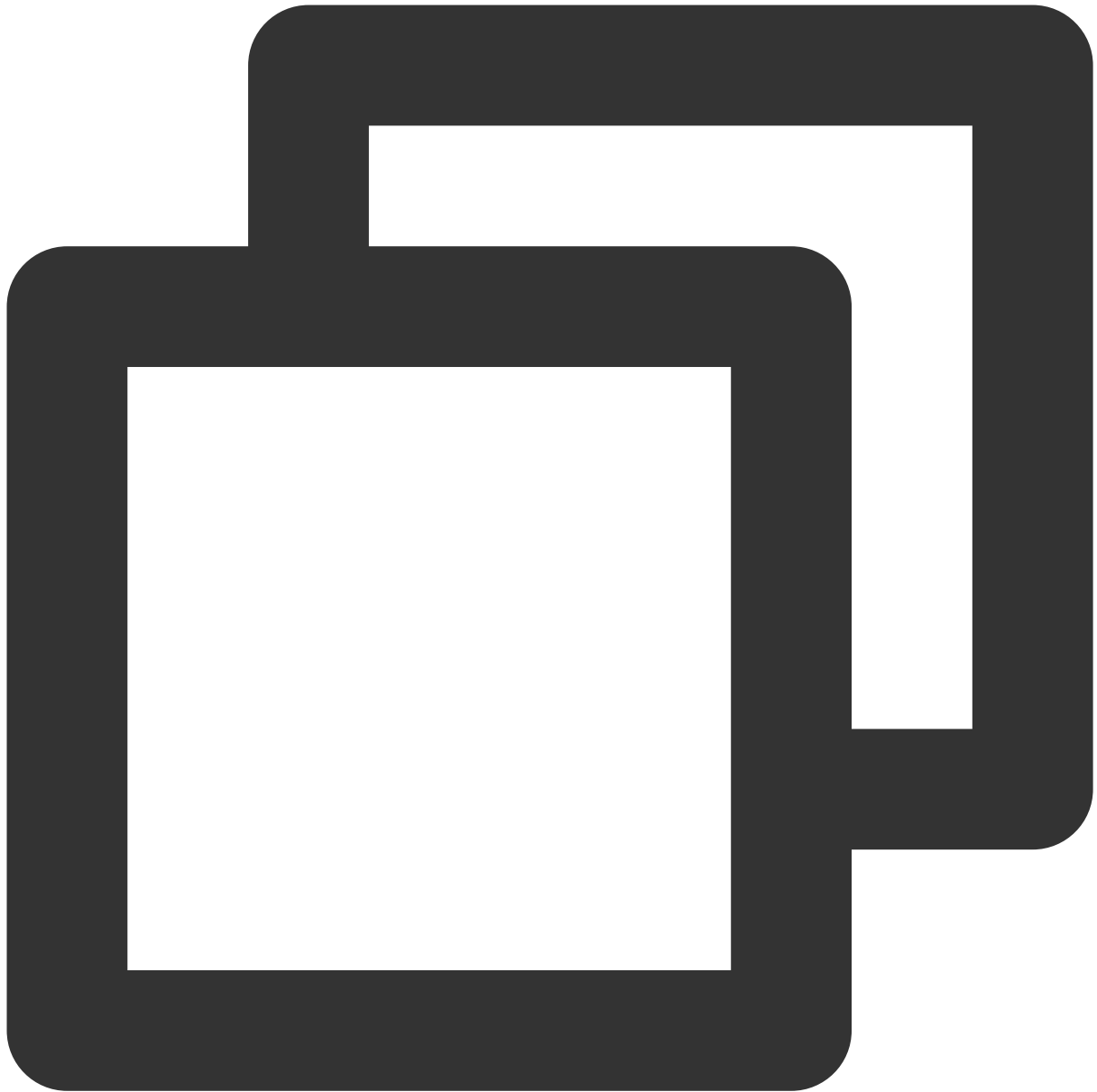
INFO[2020-04-10T10:38:42+08:00] Download tools.
→ /tmp []
```

This is because you must provide a config directory (default: `~/.config` ) and a port number (default: 15740) when registering a node. To re-register a node, manually specify a non-conflicting config directory and port number. Before performing this operation, you must have already applied for a project token password with permissions for the build node by going to **Project Settings > Developer Options > Project Token**.

In the following example, we use `~/.coding2` and `port 15741` to re-register a node.



```
./cci-agent init --config ~/.coding2 --pt <project token password>
```



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

## Custom node exceptions

### Custom node operating system: CentOS Minimal Install

If your custom node Jenkins fails to start and the warning below is shown, it may be because the node's operating system is CentOS (Minimal Install), and the Jenkins webpage relies on some graphical components.

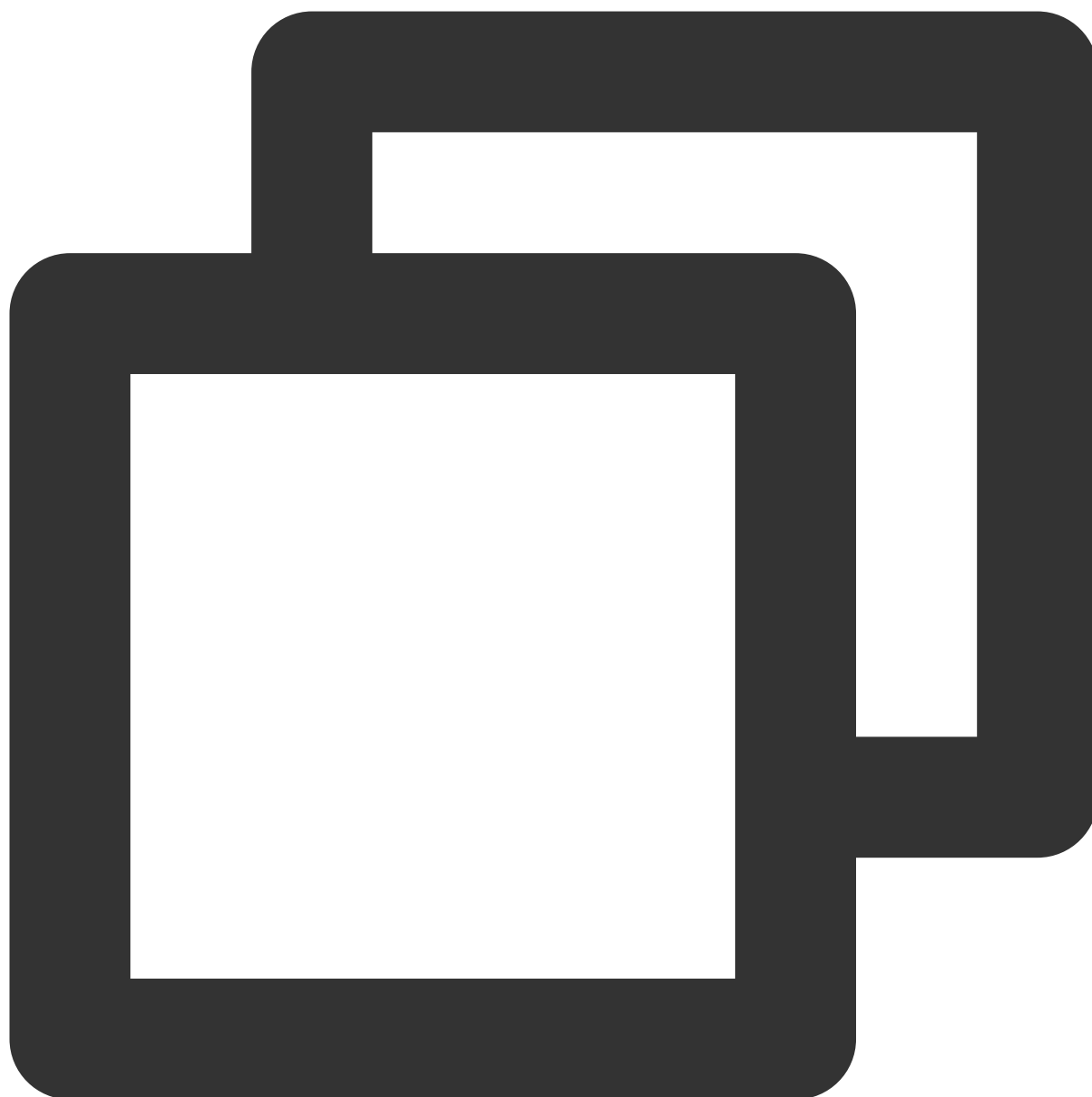


```

2020-09-29 00:49:09.860+0800 [id=1] INFO    o.e.j.server.session.HouseKeeper#startScavenging: node0 Scavenging every 66000ms
2020-09-29 00:49:09.110+0800 [id=1] SEVERE  hudson.util.BootFailure#publish: Failed to initialize Jenkins
java.lang.NullPointerException
    at sun.awt.FontConfiguration.getVersion(FontConfiguration.java:1264)
    at sun.awt.FontConfiguration.readFontConfigFile(FontConfiguration.java:219)
    at sun.awt.FontConfiguration.init(FontConfiguration.java:107)
    at sun.awt.X11FontManager.createFontConfiguration(X11FontManager.java:774)
    at sun.font.SunFontManager$2.run(SunFontManager.java:431)
    at java.security.AccessController.doPrivileged(Native Method)
    at sun.font.SunFontManager.<init>(SunFontManager.java:376)
    at sun.awt.FcFontManager.<init>(FcFontManager.java:35)
    at sun.awt.X11FontManager.<init>(X11FontManager.java:57)
    at sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method)
    at sun.reflect.NativeConstructorAccessorImpl.newInstance(NativeConstructorAccessorImpl.java:62)
    at sun.reflect.DelegatingConstructorAccessorImpl.newInstance(DelegatingConstructorAccessorImpl.java:45)
    at java.lang.reflect.Constructor.newInstance(Constructor.java:423)
    at java.lang.Class.newInstance(Class.java:442)
    at sun.font.FontManagerFactory$1.run(FontManagerFactory.java:83)
    at java.security.AccessController.doPrivileged(Native Method)
    at sun.font.FontManagerFactory.getInstance(FontManagerFactory.java:74)
    at java.awt.Font.getFont2D(Font.java:491)
    at java.awt.Font.getFamily(Font.java:1220)
    at java.awt.Font.getFamily_NoClientCode(Font.java:1194)
    at java.awt.Font.getFamily(Font.java:1186)
    at java.awt.Font.toString(Font.java:1683)
    at hudson.util.ChartUtil.<clinit>(ChartUtil.java:260)
    at hudson.WebAppMain.contextInitialized(WebAppMain.java:192)
Caused: hudson.util.AWTProblem
    at hudson.WebAppMain.contextInitialized(WebAppMain.java:193)
    at org.eclipse.jetty.server.handler.ContextHandler.callContextInitialized(ContextHandler.java:957)
    at org.eclipse.jetty.servlet.ServletContextHandler.callContextInitialized(ServletContextHandler.java:553)
    at org.eclipse.jetty.server.handler.ContextHandler.startContext(ContextHandler.java:922)
    at org.eclipse.jetty.servlet.ServletContextHandler.startContext(ServletContextHandler.java:365)
    at org.eclipse.jetty.webapp.WebAppContext.startWebapp(WebAppContext.java:1497)
    at org.eclipse.jetty.webapp.WebAppContext.startContext(WebAppContext.java:1459)
    at org.eclipse.jetty.server.handler.ContextHandler.doStart(ContextHandler.java:852)
    at org.eclipse.jetty.servlet.ServletContextHandler.doStart(ServletContextHandler.java:278)
    at org.eclipse.jetty.webapp.WebAppContext.doStart(WebAppContext.java:545)
    at org.eclipse.jetty.util.component.AbstractLifecycle.start(AbstractLifecycle.java:68)
    at org.eclipse.jetty.util.component.ContainerLifecycle.start(ContainerLifecycle.java:167)
    at org.eclipse.jetty.server.Server.start(Server.java:418)
    at org.eclipse.jetty.util.component.ContainerLifecycle.doStart(ContainerLifecycle.java:110)
    at org.eclipse.jetty.server.handler.AbstractHandler.doStart(AbstractHandler.java:113)
    at org.eclipse.jetty.server.Server.doStart(Server.java:382)
    at org.eclipse.jetty.util.component.AbstractLifecycle.start(AbstractLifecycle.java:68)
    at winstone.Launcher.<init>(Launcher.java:187)
    at winstone.Launcher.main(Launcher.java:362)
    at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
    at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
    at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
    at java.lang.reflect.Method.invoke(Method.java:498)
    at Main._main(Main.java:375)
    at Main.main(Main.java:151)

```

To solve this problem, run the following command:



```
yum install fontconfig
```

## Abnormal status handling

In actual production, unstable factors such as the client's network environment or a missing configuration environment can affect access to custom build nodes. The list below provides the methods used to handle abnormal statuses.

### Deleted build node pool

--	--	--

Location	Build Record Error Prompt	Handling Method
Build plan configuration	-	Deleted build node pools will not appear in the build plan node pool configuration.
Configured build plan	The configuration page indicates that the build node pool has been deleted.	A build node pool configured in the build plan has been deleted.
Trigger build task	The build node pool my\\-pool configured for this build plan has been deleted. Please reconfigure.	The build task can be triggered, but it will immediately fail.
In queue, initializing, preparing build, building	The build node pool my\\-pool configured for this build plan has been deleted. Please reconfigure.	Build tasks in these statuses will fail immediately.

### "In Use" build node deleted

Location	Build Record Error Prompt	Handling Method
Build tasks in queue	-	Unaffected, the build tasks in the queue have not been assigned specific build nodes and will remain in the queue until a valid build node is found.
Initializing, preparing build, building	Build node xxx offline	Build tasks in these statuses will fail immediately.

### "In Use" build node is offline

#### Note:

The build node may be offline due to the unstable network of the client.

After going offline, the client (build node) will attempt to reconnect. The server will retry the operation after the client reconnects.

If the operation times out (after three minutes), the build node is judged to be offline and the build task is marked as failed.

Upon successful reconnection, the client will continue to report build content.

Location	Build Record Error Prompt	Handling Method
Build tasks in queue	-	Unaffected, the build tasks in the queue have not been assigned specific build nodes and will remain in the queue until a valid build node is found.
Initializing, preparing build, building	Build node xxx offline	Build tasks in these statuses will fail immediately.

### Configured build pool has no accessible nodes

Location	Build Record Error Prompt	Handling Method
Build plan configuration	-	Because the build plan is not directly associated with build nodes, this does not affect the build plan configuration. If there are no build nodes in the node pool, the configuration page will display the corresponding warning.
Configured build plan	-	Because the build plan is not directly associated with build nodes, this does not affect the build plan configuration. If there are no build nodes in the node pool, the configuration page will display the corresponding warning.
Initializing, preparing build, building	The build node pool my\\-pool configured for this build plan has been deleted. Please reconfigure.	Build tasks in these statuses will fail immediately.

### Build plan authorization canceled

Location	Build Record Error Prompt	Handling Method
Build plan configuration	The corresponding prompt is displayed.	Users cannot select unauthorized build plans.
Configured build plan	-	The Build Record List page and Build Configuration page will display an unauthorized prompt. You must manually adjust the node pool configuration.
Trigger build	This build plan does not have	The build task can be triggered, but it will immediately

task	authorization for the build node pool default. Please authorize it.	fail.
Initializing, preparing build, building	The build node pool my\\-pool configured for this build plan has been deleted. Please reconfigure.	Build tasks in these statuses will fail immediately.