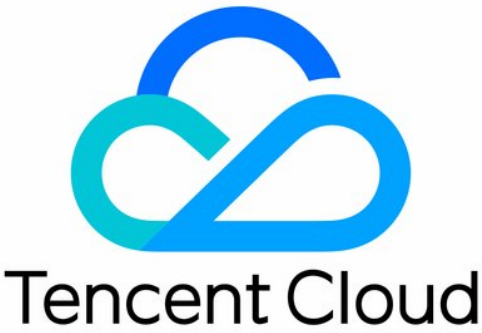
CODING Continuous Integration

Operation Guide

Product Documentation



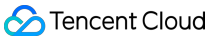
©2013-2022 Tencent Cloud. All rights reserved.

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

Operation Guide
- Compile Build Process
  - Text Editor
  - Process Configuration Details
  - Graphical Editor
- Configure Build Plan
  - Trigger Rules
  - Environment Variables
  - Build Snapshots
  - Cache Directory
- Build Artifacts
  - Docker
- Manage Build Plans
  - Group Management
  - Build Plan Templates
- System Plugins
  - Error
  - Upload Generic Artifacts

©2013-2022 Tencent Cloud. All rights reserved.

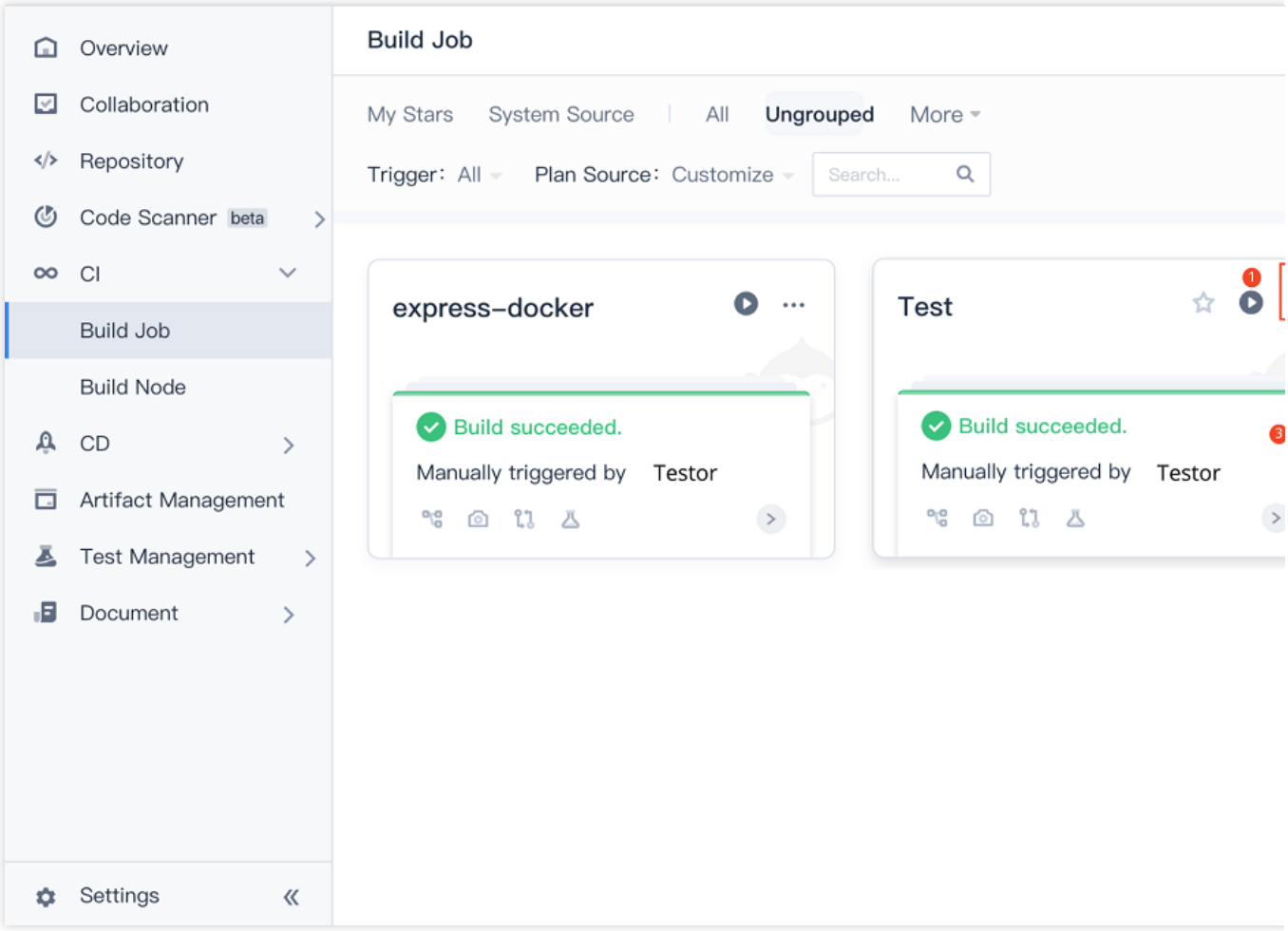
Operation Guide
Compile Build Process
Text Editor

Last updated：2023-12-29 11:44:51

title: Text Editor - CODING Help Center
pageTitle: Text Editor
pagePrevTitle: Getting Started
pagePrev: ci/start.html
pageNextTitle: Configuration Details
pageNext: ci/process/detail.html

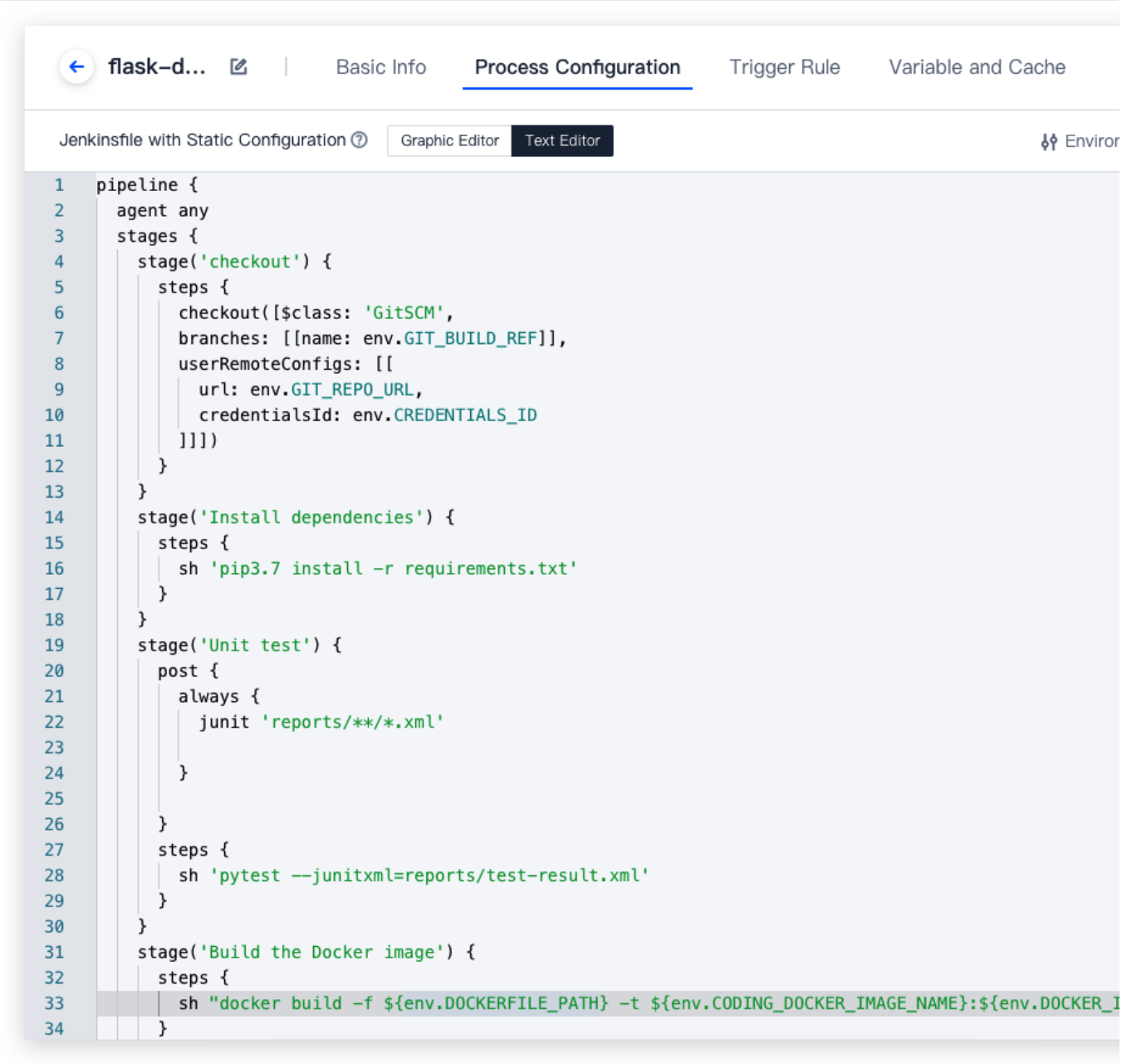
In essence, build tasks follow the processes and steps defined in configuration files. CODING Continuous Integration (CODING-CI) is fully compatible with Jenkinsfiles. Configuration files composed in the text editor can be run as long as they follow the syntax specifications of Jenkinsfiles.
Open a project, click "Continuous Integration" on the left, and then click "Settings" on a build plan.

©2013-2022 Tencent Cloud. All rights reserved.



Select "Process Configuration" > "Text Editor".

©2013-2022 Tencent Cloud. All rights reserved.



Follow the syntax for Jenkinsfile in the build process. For more information, refer to the following documentation.

More information

Official Jenkinsfile Documentation

Configuration Details

==== 2021/08/14 ====

©2013-2022 Tencent Cloud. All rights reserved.

Process Configuration Details

Last updated：2023-12-29 11:44:51

title: Process Configuration Details - CODING Help Center
pageTitle: Process Configuration Details
pagePrevTitle: Text Editor
pagePrev: ci/process/text.html
pageNextTitle: Graphical Editor
pageNext: ci/process/visual-editor.html
alias: process/detail.html

This document provides you a guidance on compiling a build process and describes parameters in each step.

Code repositories

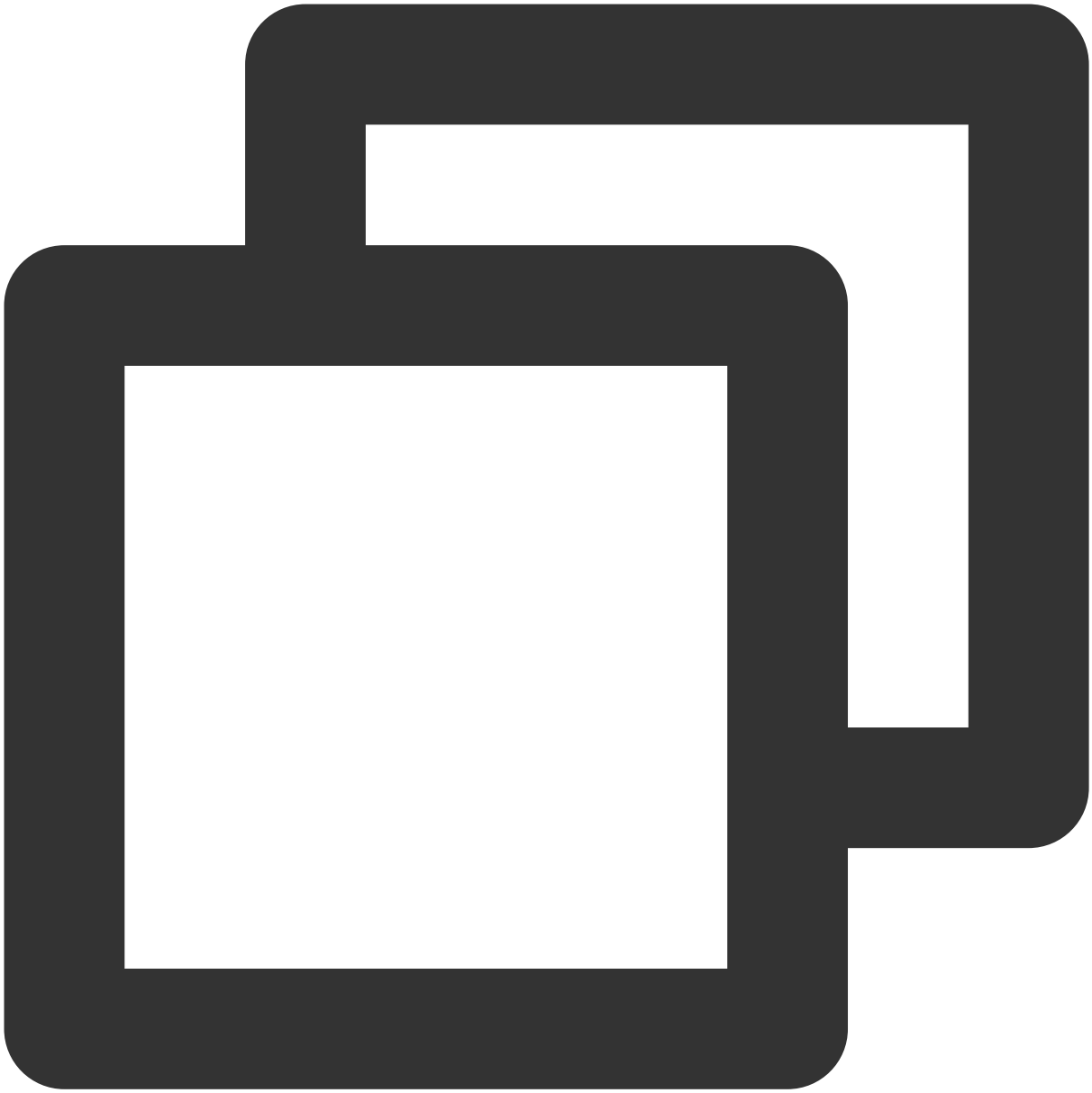
Git

Checks out source code from a Git repository in the current project. This command is a simpler version of the checkout command.
Parameter list:
Git URL `url` : string
Branch `branch` : string
Change log `changelog` : string
Identity authentication ID `credentialsId` : string
Poll `poll` : boolean

Check out from version control

Universally checks out SCM code (Git or SVN).
This step returns content in a map format. If you are using Git, you could use the following:

©2013-2022 Tencent Cloud. All rights reserved.

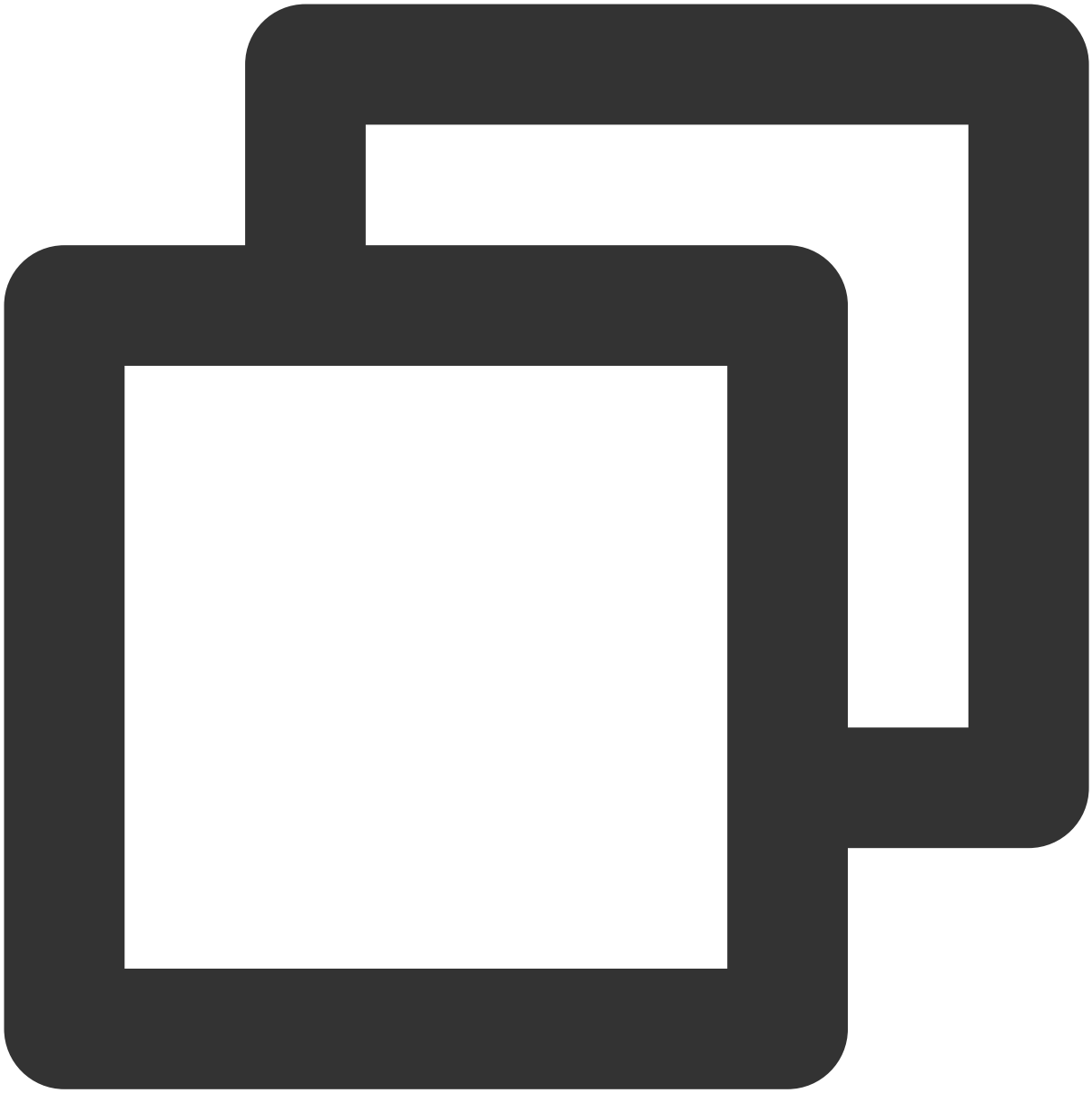


```
def scmVars = checkout scm
def commitHash = scmVars.GIT_COMMIT
// or
def commitHash = checkout(scm).GIT_COMMIT
```

The parameter scm is an object that allows the SCM type to be configured, such as the following:

GitSCM example:

©2013-2022 Tencent Cloud. All rights reserved.



```
checkout([$class: 'GitSCM', branches: [[name: env.GIT_BUILD_REF]],
              userRemoteConfigs: [[url: env.GIT_REPO_URL]]])
```

userRemoteConfigs parameter list:

`url` : string

`name` : string, name of a remote repository such as "origin"

`refspec` : string, for more information, see Git Internals - The Refspec.

`branches` : array of objects (optional)

`changelog` : boolean (optional)

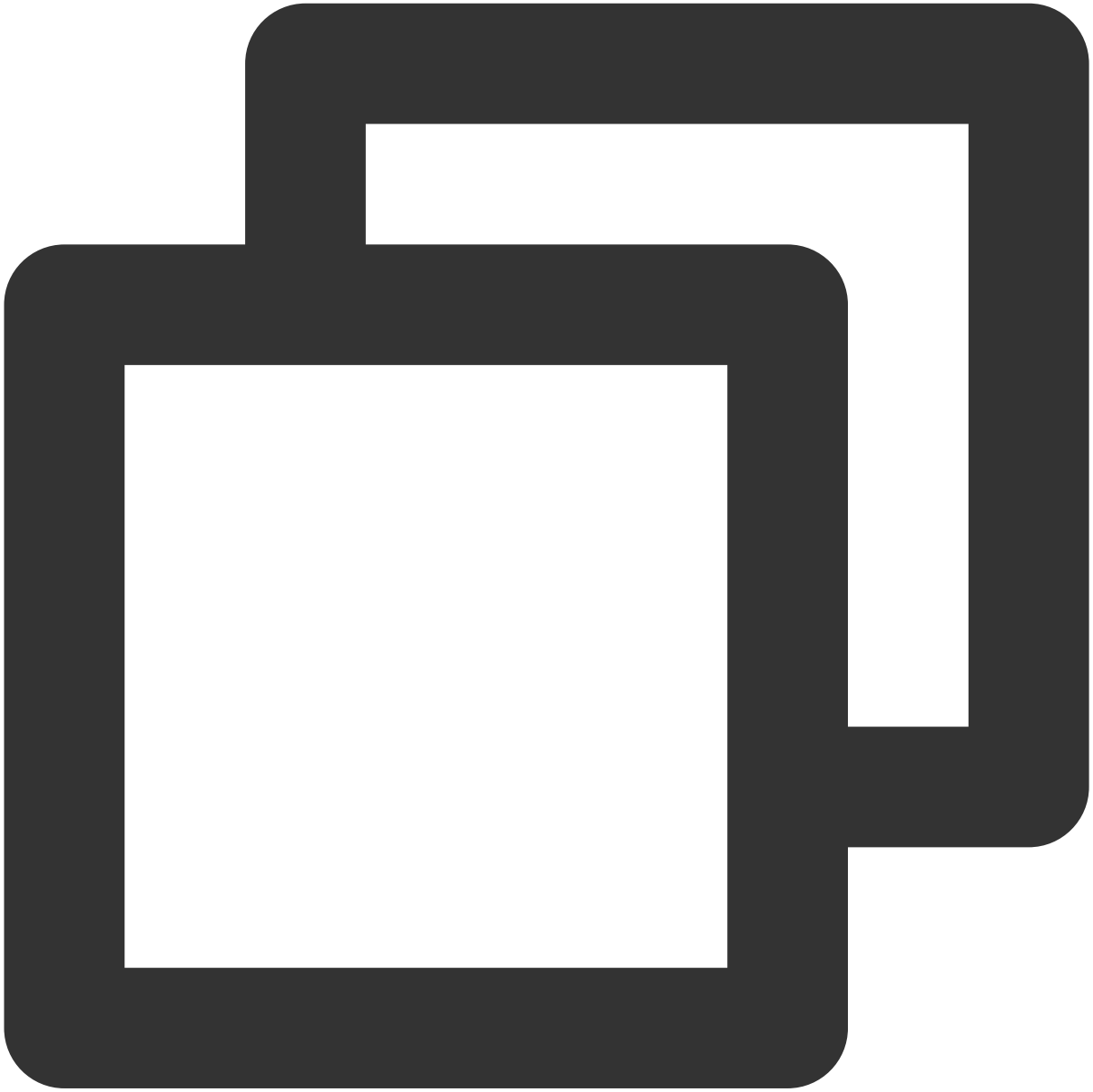
©2013-2022 Tencent Cloud. All rights reserved.

`credentialsId` : string (optional)

`doGenerateSubmoduleConfigurations` : boolean (optional)

`submoduleCfg` : array (optional)

SubversionSCM: Checks out code from an SVN server. Example:



```
checkout([$class: 'SubversionSCM', remote: 'http://sv-server/repository/trunk']])
```

Parameter list:

`locations` : array of objects

`remote` : string

©2013-2022 Tencent Cloud. All rights reserved.

`credentialsId` : string

`local` : string, specifies a local directory (relative to the workspace) as the location of code to be checked out

`depthOption` : string. Corresponds to --depth. The default value is unlimited. Learn More.

`ignoreExternalsOption` : boolean

Build process

Subnodes

Parameter list:

`label` : string, environment label name such as java-8

Collect artifacts

Collects build results (such as jar, war, or apk). Note that the artifacts collected are saved and deleted along with the build history. This is just a temporary storage space. We recommend using "Artifact Management" for version management of build results.

Parameter list:

`artifacts` : string, you can use the wildcard * to specify the path pattern of files in the workspace to be collected while keeping to the Apache Ant Path Rules

`allowEmptyArchive` : boolean (optional). Generally, this command results in "Building Failed" if no file appropriate to the collection pattern is found. If this parameter is set to "true", a build process returns a warning if there are no artifacts, instead of a failure result.

`caseSensitive` : boolean (optional). By default, file path rules are case-sensitive. If this parameter is set to "false", the rules are non-case-sensitive.

`defaultExcludes` : boolean (optional)

`excludes` : boolean (optional), you can exclude certain files when setting the path pattern while keeping to the Apache Ant Path Rules

`fingerprint` : boolean (optional), file hashes are also calculated during collection

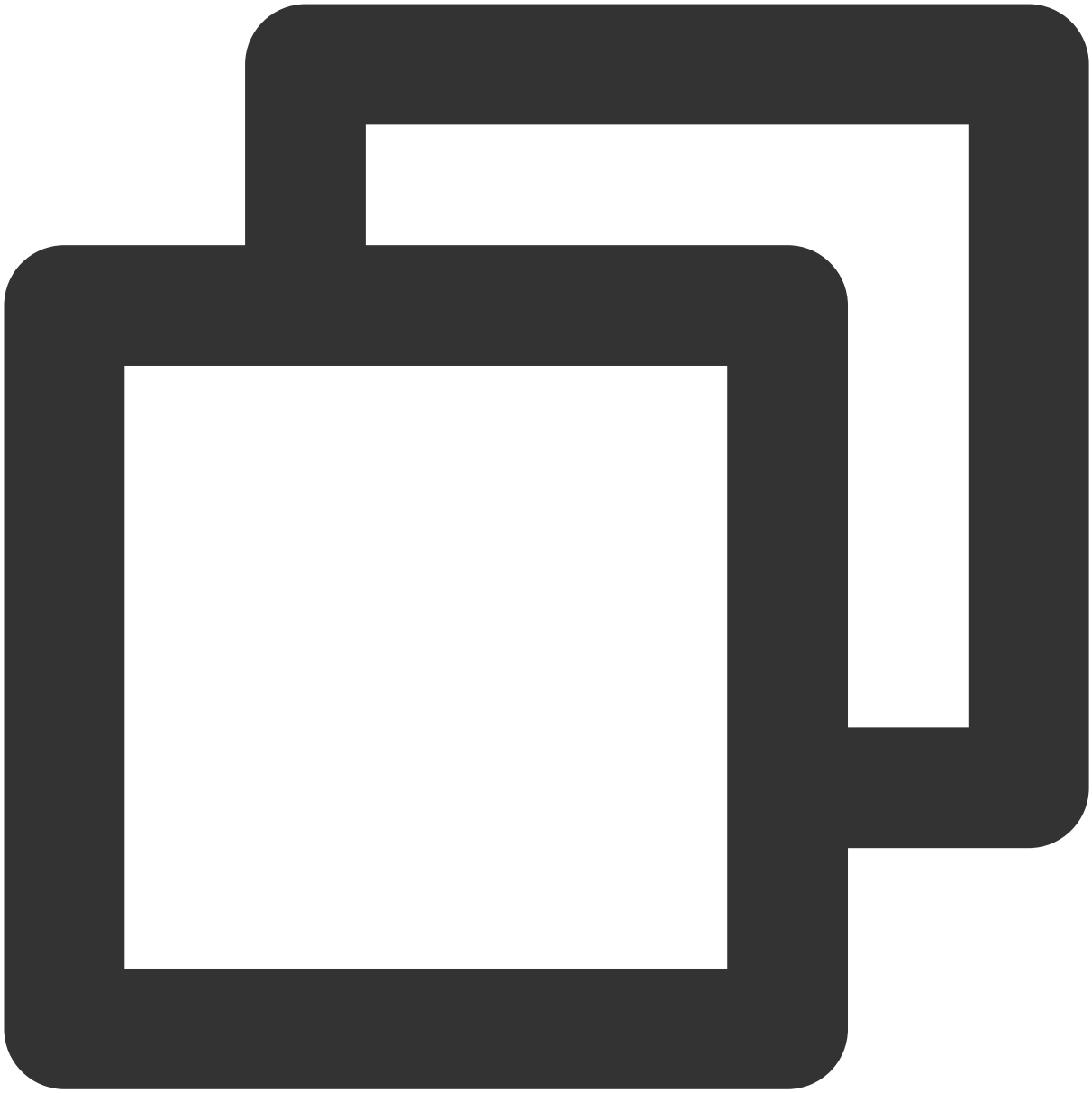
`onlyIfSuccessful` : boolean (optional), only collected when "Build successful" is returned

Execute shell script

Executes a shell script.

Example:

©2013-2022 Tencent Cloud. All rights reserved.



```
pipeline {
    agent any
    stages {
        stage('Example') {
            steps {
                echo 'Hello World'
                sh 'ls -al'
            }
        }
    }
}
```

©2013-2022 Tencent Cloud. All rights reserved.

Collect JUnit test reports

Collects JUnit and TestNG test reports (in XML). You can specify the XML files to be collected using `**/build/test-reports/*.xml`, for example. Do not include XML files that are not reports. You can use commas to separate multiple rules.

Parameter list:

`testResults`: string

`allowEmptyResults`: boolean (optional), "file does not exist" or "file is empty" is allowed

`keepLongStdio`: boolean (optional), all test logs, including those of passed test cases, are kept

Others

Change directory substep

Changes directory substeps. You can fill in some substeps in the "dir" block, which will be run in the specified directory path.

Parameter list:

`path`: string

Sleep

Pauses for a period of time until the set due time. Similar to `sleep xxx` in Unix.

Parameter list:

`time`: int

`unit`: Select one from `NANOSECONDS, MICROSECONDS, MILLISECONDS, SECONDS, MINUTES, HOURS, and DAYS`.

Error

Sends an error signal. Used when you need to partially terminate an execution process. You can also use throw new Exception(), but the exception stack printed is shorter if you use "error".

Parameter list:

`message`: string

Current directory

Returns the current directory path as a string.

Parameter list:

`tmp`: boolean (optional). This parameter returns a temporary directory associated with the workspace. Generally, it is used when you need to store some temporary files without confusing the workspace directory.

Write file

©2013-2022 Tencent Cloud. All rights reserved.

Writes the specified content into a file.

Parameter list:

`file` : string

`text` : string

`encoding` : string. File encoding. If left empty, the default encoding in the current run environment is used. In the case of a binary file, a base-64-encoded result is returned.

Read file

Reads a file from a relative path and returns the file content as a string.

Parameter list:

`file` : string, path address relative to the workspace directory

encoding: string. File encoding. If left empty, the default encoding in the current run environment is used. In the case of a binary file, a base-64-encoded result is returned.

Retry substep

Retries the specified block until the set maximum number of retries is reached. Stops retrying if the execution process ends normally. Keeps retrying until the set maximum number of retries is reached if an exception occurs in the execution process. The build process terminates if an exception occurs during the last try.

Parameter list:

`count` : int

Time-limited substep

Executes the process in a block within a limited time. When the time is up, the exception `org.jenkinsci.plugins.workflow.steps.FlowInterruptedException` is returned. The optional unit parameter is minutes by default.

Parameter list:

`time` : int

`activity` : boolean. Time is calculated when there is no new content in the log rather than based on an absolute execution time.

`unit` : Select one from `NANOSECONDS, MICROSECONDS, MILLISECONDS, SECONDS, MINUTES, HOURS, and DAYS` .

Catch error in substep

Catches errors in the specified substep.

Timed substep

Records the execution time of the specified substep in the form of a Unix timestamp.

Loop substep

©2013-2022 Tencent Cloud. All rights reserved.

Loops the execution of the specified substep for the designated number of times.

Conditional loop substep

Loops the execution of the specified substep until the substep returns "true".

Print information

Prints information in the log.

Parameter list:

`message` : string

Run arbitrary pipeline script

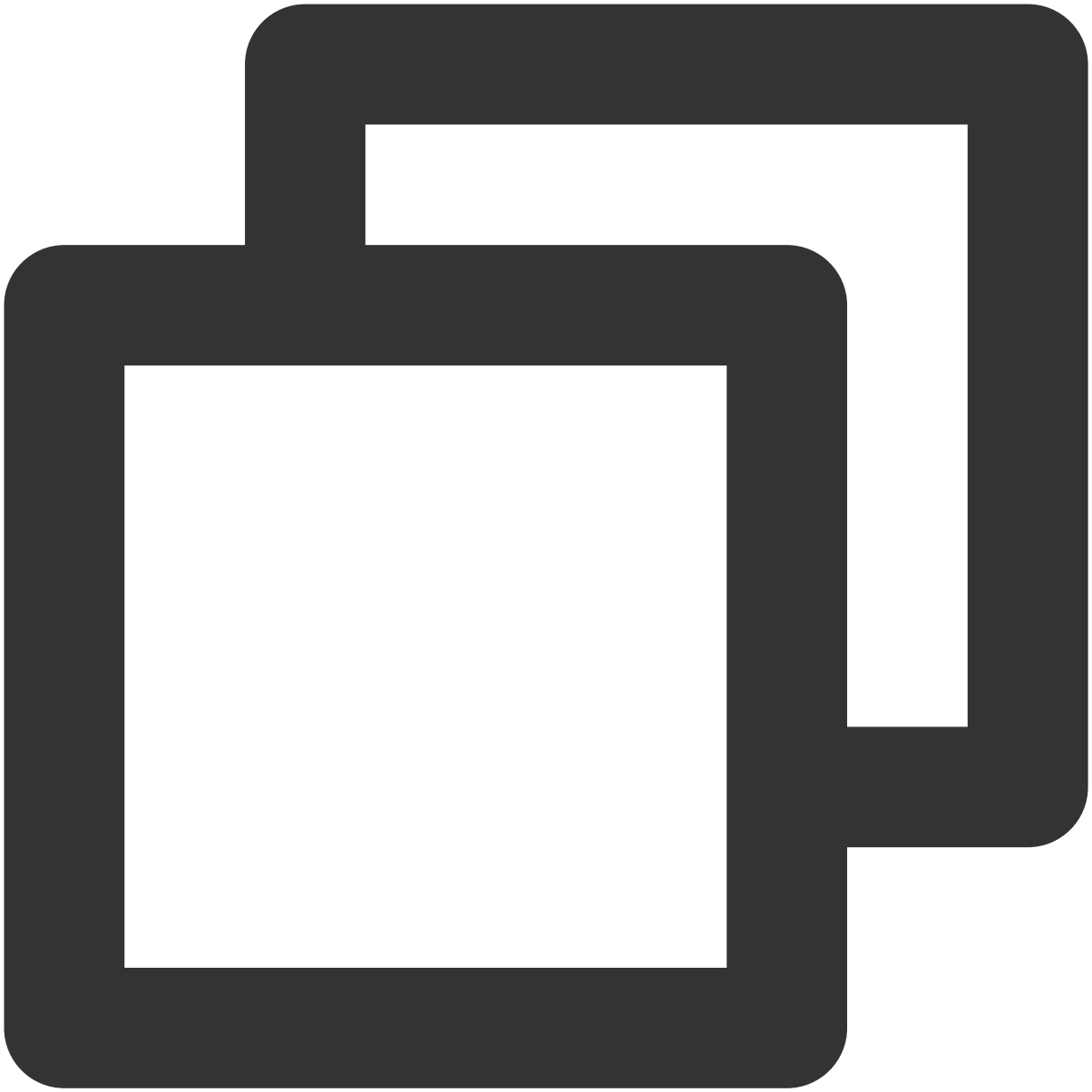
Runs an arbitrary pipeline script.

Run Groovy source file

Runs a Groovy source file in this location during the build process.

Example:

©2013-2022 Tencent Cloud. All rights reserved.



```
pipeline {
    agent any
    stages {
        stage('Example') {
            steps {
                echo 'Hello World'
                load 'test.groovy'
            }
        }
    }
}
```

©2013-2022 Tencent Cloud. All rights reserved.

Run yarn audit

Runs a yarn audit in the specified directory. In Continuous Integration, you can see the vulnerabilities found during the yarn dependency audit on the results page.
Parameter list:
`directory` : string (optional). Fill in the directory location of yarn.lock. Runs in the root directory of the project by default.
`collectResult` : boolean (optional). Collects yarn audit reports.

Run npm audit

Runs an npm audit in the specified directory. In Continuous Integration, you can see the vulnerabilities found during the npm dependency audit on the results page.
Parameter list:
`directory` : string (optional). Fill in the directory location of package.json. Runs in the root directory of the project by default.
`collectResult` : boolean (optional). Collects npm audit reports.

Merge merge request

Merges code. You can merge the specified merge request.
Parameter list:
`token` : string, project token
`depot` : string, repository name
`mrResourceId` : string, specified resource ID
`commitMessage` : string, merged commit message template
`deleteSourceBranch` : boolean (optional), deletes source branch
`fastForward` : boolean (optional), tries fast-forward merge

Merge request comment

Comments on a merge request. You can comment on a specified merge request.
Parameter list:
`token` : string, project token
`depot` : string, repository name
`mrResourceId` : string, specified resource ID
`commentContent` : string, comment template
==== 2020/08/13 ====

©2013-2022 Tencent Cloud. All rights reserved.

Graphical Editor

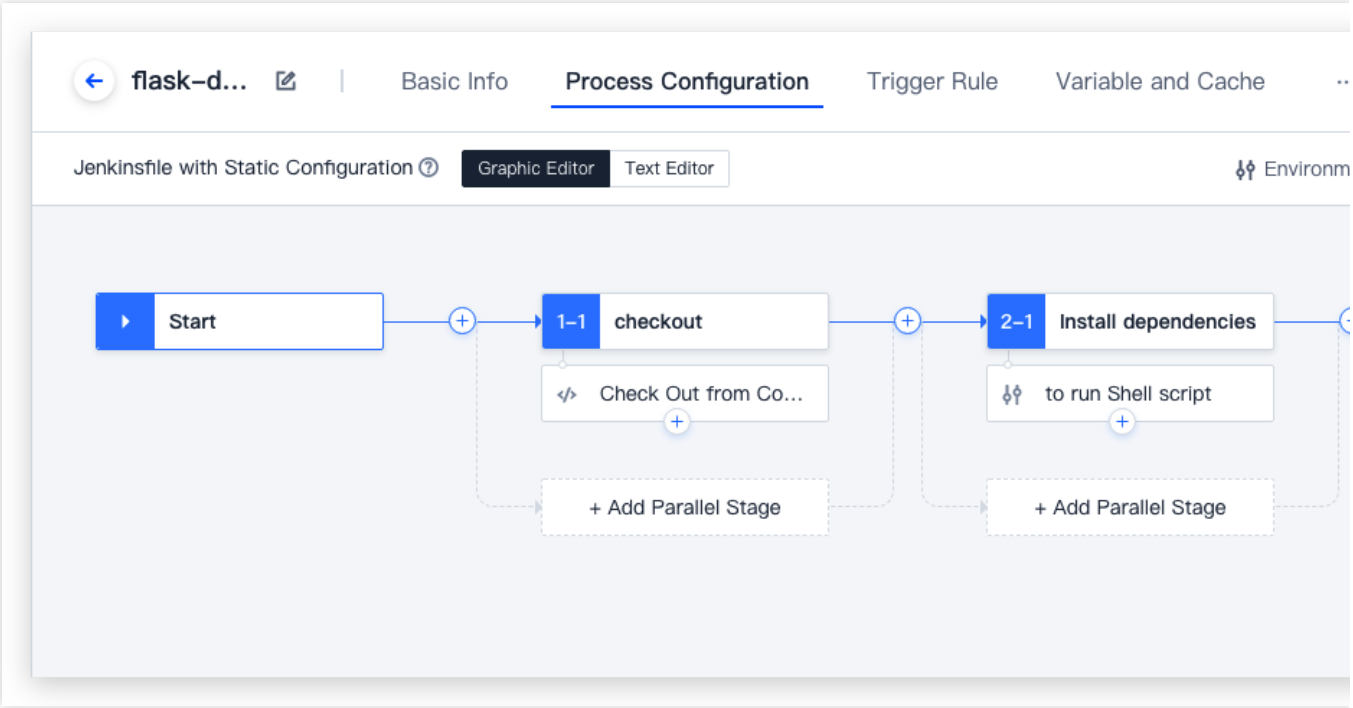
Last updated：2023-12-29 11:44:51

title: Graphical Editor - CODING Help Center
pageTitle: Graphical Editor
pagePrevTitle: Process Configuration Details
pagePrev: ci/process/detail.html
pageNextTitle: Trigger Rules
pageNext: ci/configuration/trigger.html
alias:
devops/ci/visualeditor.html
ci/visualeditor.html
ci/visual-editor.html

Function Overview

Editing a Jenkinsfile (a file that describes a build process) using a command-line editor is the most basic mode of human-computer interaction. Based on its core text editing function, CODING has been designed with an innovative graphical editor that is compatible with most custom command-line operations. Enjoy an intuitive WYSIWYG editing experience as you can view while building.
To access the function, go to "Build Plan Settings" > "Process Configuration".

©2013-2022 Tencent Cloud. All rights reserved.



Build Process Concepts

In essence, both graphical and text editors allow users to view and edit the core of the build process—the Jenkinsfile (a file that describes the process). Before we go into the details of editors, let's take a look at a few important concepts concerning the "file that describes the process".

Note:

This document focuses on the syntax rules for declarative files.

Pipeline

A `pipeline` is a customizable working model that defines an entire process for delivering software. In general, it includes build, test, and deployment phases.

Execution Environment

The execution environment describes the execution environment of the entire process or a certain `stage` of executing a `pipeline`. It must appear in the top grid of a `descriptive file` or at every `stage`.

Required?	Yes
Parameter list	See below
Permitted location	Must appear in the top grid of a `descriptive file` or at every stage

Stage

©2013-2022 Tencent Cloud. All rights reserved.

A `stage` defines a series of closely related `steps`. Each `stage`, such as the "build stage", the "test stage", or the "deployment stage", undertakes an independent, clear responsibility in an entire pipeline. Generally, all actual build processes are provided in stages.

Required?	At least one
Parameter list	A required string parameter that specifies the name of a stage
Permitted location	In the `stage` block

Stage List

The `stage list` includes a series of `stages`. A `stage list` will include at least one `stage`. A `pipeline` must have and only have one `stage list`.

Required?	Yes
Parameter list	None
Permitted location	Can only appear once in the `pipeline`

Step List

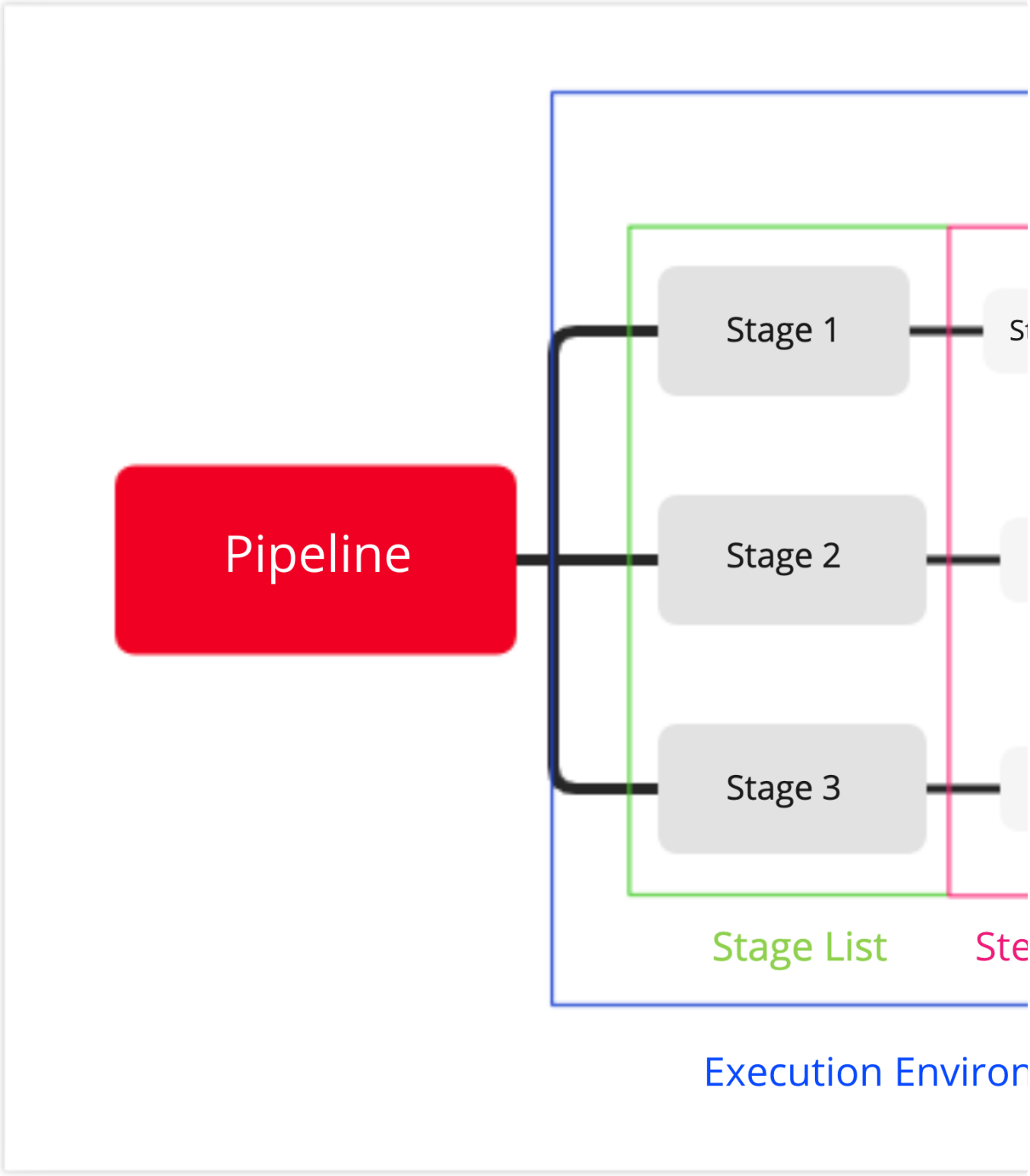
The `step list` describes what to do at a `stage` and what specific commands to run. For example, a `step` needs the system to print a "Building" message and run the command `echo 'building...'`.

Required?	Yes
Parameter list	None
Permitted location	In every `stage` block

Parallel

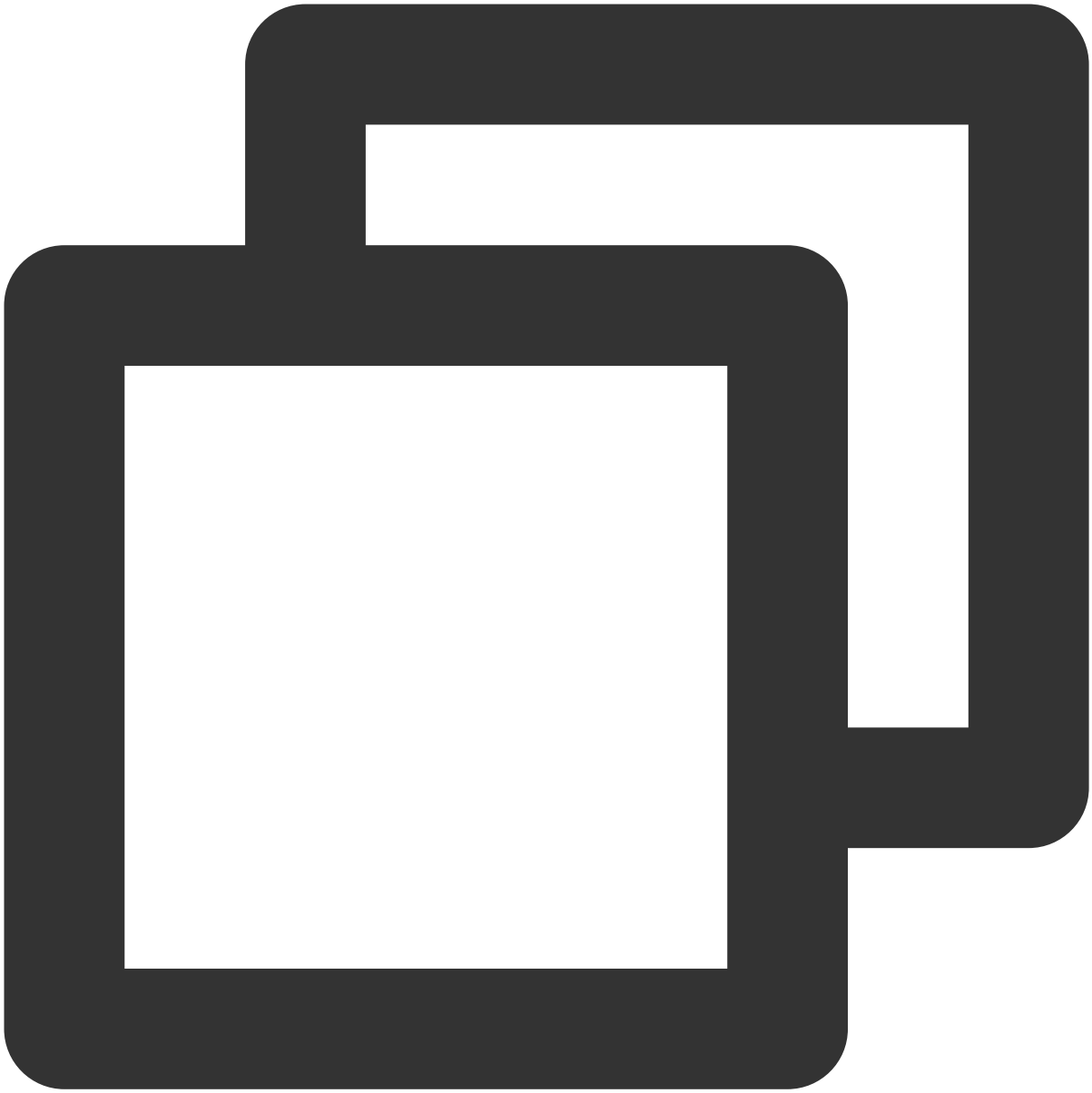
"Parallel" is used to declare some `stages` executed in parallel to accelerate the execution speed, especially when a `stage` and another `stage` are independent of each other. Take note that you cannot set the `execution environment` for any `stage` with a `parallel` block.

©2013-2022 Tencent Cloud. All rights reserved.



Sample File

©2013-2022 Tencent Cloud. All rights reserved.



```
pipeline {
  agent any
  stages {
    stage('check out') {
      steps {
        sh 'ci-init'
        checkout([$class: 'GitSCM', branches: [[name: env.GIT_BUILD_REF]],
                                    userRemoteConfigs: [[url: env.GIT_REPO_URL]]])
      }
    }
    stage('build') {
```

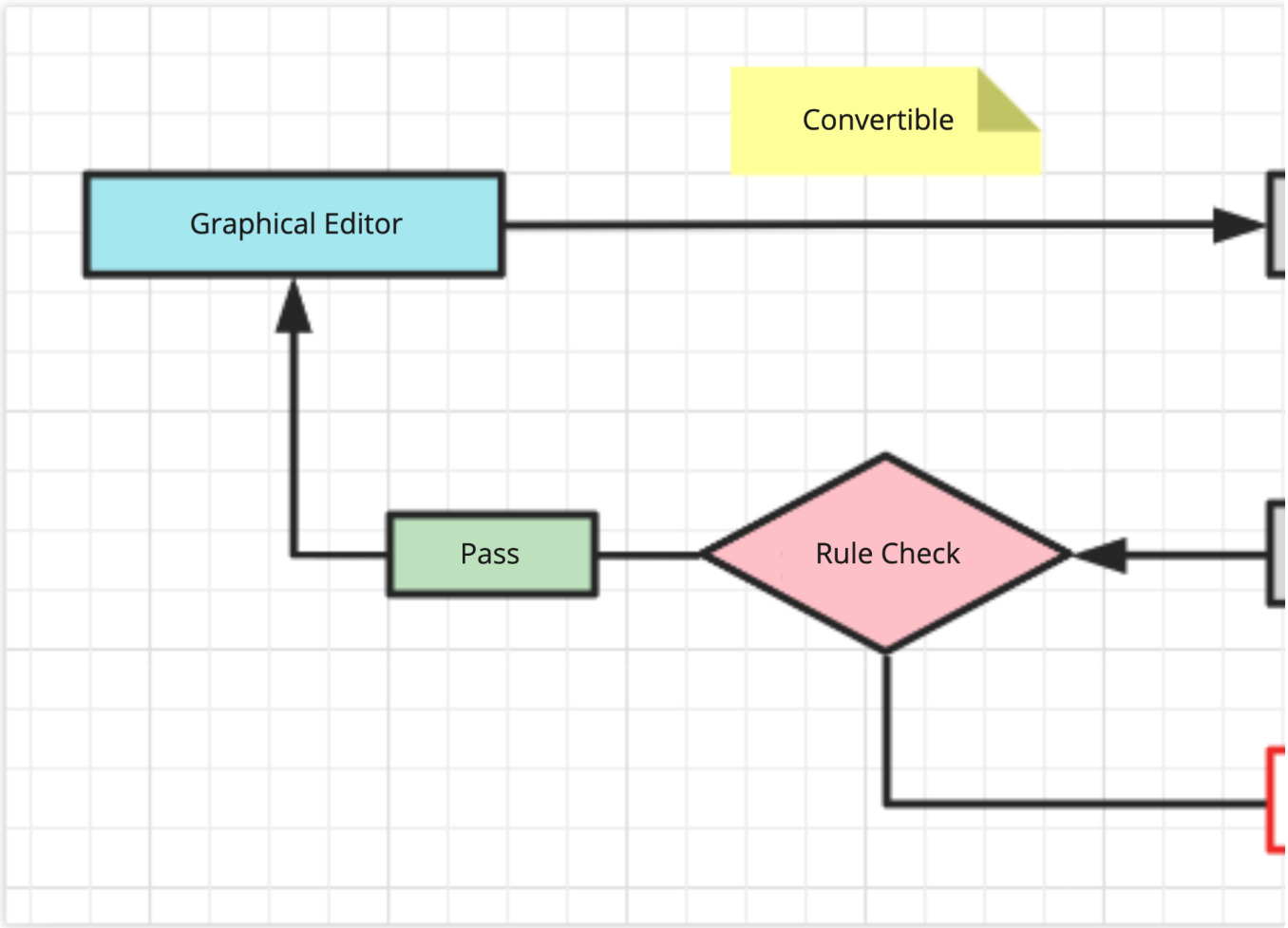
©2013-2022 Tencent Cloud. All rights reserved.

```
        steps {
          echo 'building...'
          sh 'make'
          echo 'build complete.'
        }
      }
      stage('Test') {
        steps {
          echo 'unit testing...'
          sh 'make check'
          junit 'reports/**/*.xml'
          echo 'unit testing complete.'
        }
      }
      stage('deploy') {
        steps {
          echo 'deploying...'
          sh 'make publish'
          echo 'deployment complete'
        }
      }
    }
}
```

Switching Between Editors

In essence, the graphical editor is preset code, allowing you to switch seamlessly to the text editor. However, you cannot switch from the text editor to the graphical editor. Code added or deleted in the text editor must pass a "rule check" before it can be converted into an editable view.

©2013-2022 Tencent Cloud. All rights reserved.



The text editor supports a wider range of custom operations than the graphical editor. As the graphical editor is preset with numerous commonly used steps, you can use it for pattern-based and standardized work. The text editor has no limitations and only requires conformance to Jenkins syntax, lending itself to specific and special tasks.

==== 2020/09/07 ====

©2013-2022 Tencent Cloud. All rights reserved.

Configure Build Plan Trigger Rules

Last updated：2023-12-29 11:44:51

title: Trigger Rules - CODING Help Center
pageTitle: Trigger Rules
pagePrevTitle: Graphical Editor
pagePrev: ci/process/visual-editor.html
pageNextTitle: Environment Variables
pageNext: ci/configuration/env.html
alias:
devops/ci/trigger.html
ci/trigger.html

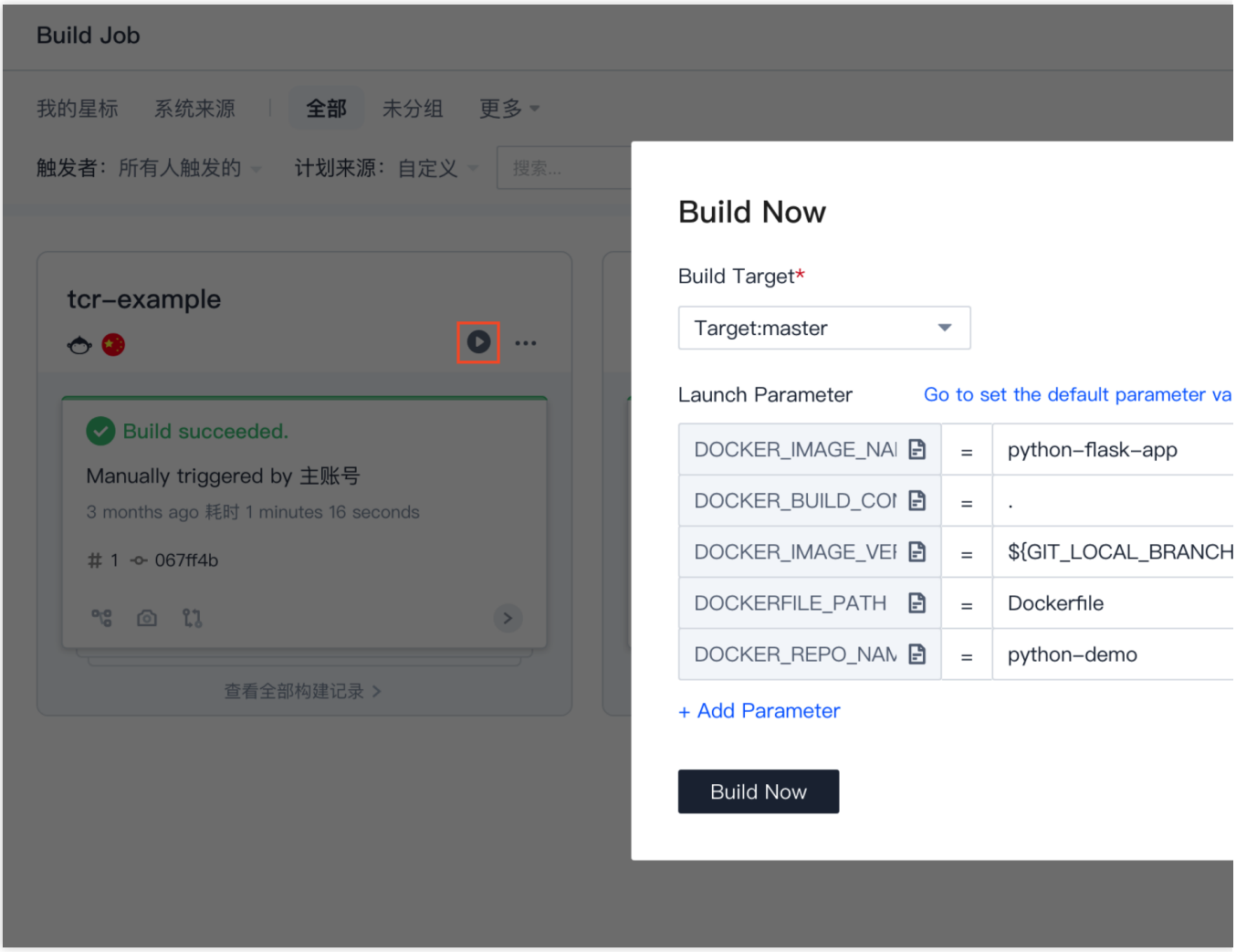
Function Overview

In the process of configuring a continuous integration (CI) plan, you can set trigger rules as necessary for running the build plan. These include the run frequency and trigger conditions of the build plan. The following trigger methods can be used in every CI build plan.
Manual trigger
Triggered by code changes
Scheduled trigger
API trigger
You can combine these methods.

Manual trigger

You can manually trigger a build plan by entering the build parameters, which will be added to the build environment in the form of environment variables.
On the build plans page, select "Build Now". In the pop-up window, select the build targets (tag, branch, or revision number), enter the required build parameters to complete the trigger build.

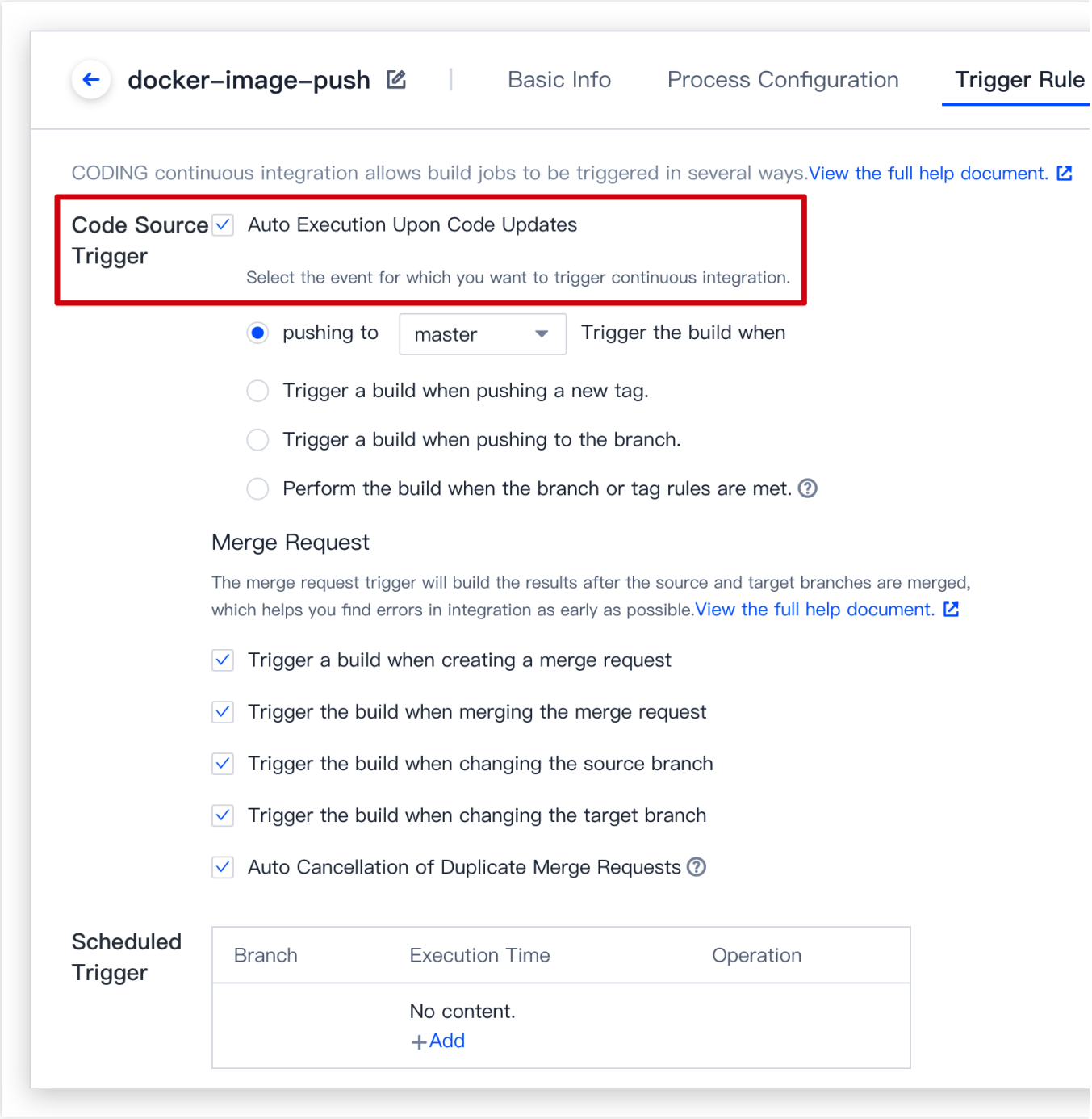
©2013-2022 Tencent Cloud. All rights reserved.



Triggered by code changes

For a build plan configured to be triggered by code changes, the code repository selected for the build plan will be monitored to trigger the build plan according to its changes.

©2013-2022 Tencent Cloud. All rights reserved.



Code updates

Trigger build when pushing to <branch>

A build is triggered only when the code of the specified branch is updated.

Trigger build when pushing a new tag

A build is triggered only when a new Git tag is created.

Trigger build when pushing to a branch

A build is triggered when the code of any branch is updated.

Build when branch or tag rules met

©2013-2022 Tencent Cloud. All rights reserved.

Supports matches of regular expressions:

1.1 If you want to trigger a build when the code of a master branch is updated, "refs/heads/master" and "master" both match the condition.

1.2 If you want to trigger a build only when a master or dev branch is updated, use the following: ^refs/heads/(master|dev).

Merge requests

A merge request triggers a build in the following circumstances:

When a merge request is initiated

A change has occurred in the source branch of a merge request

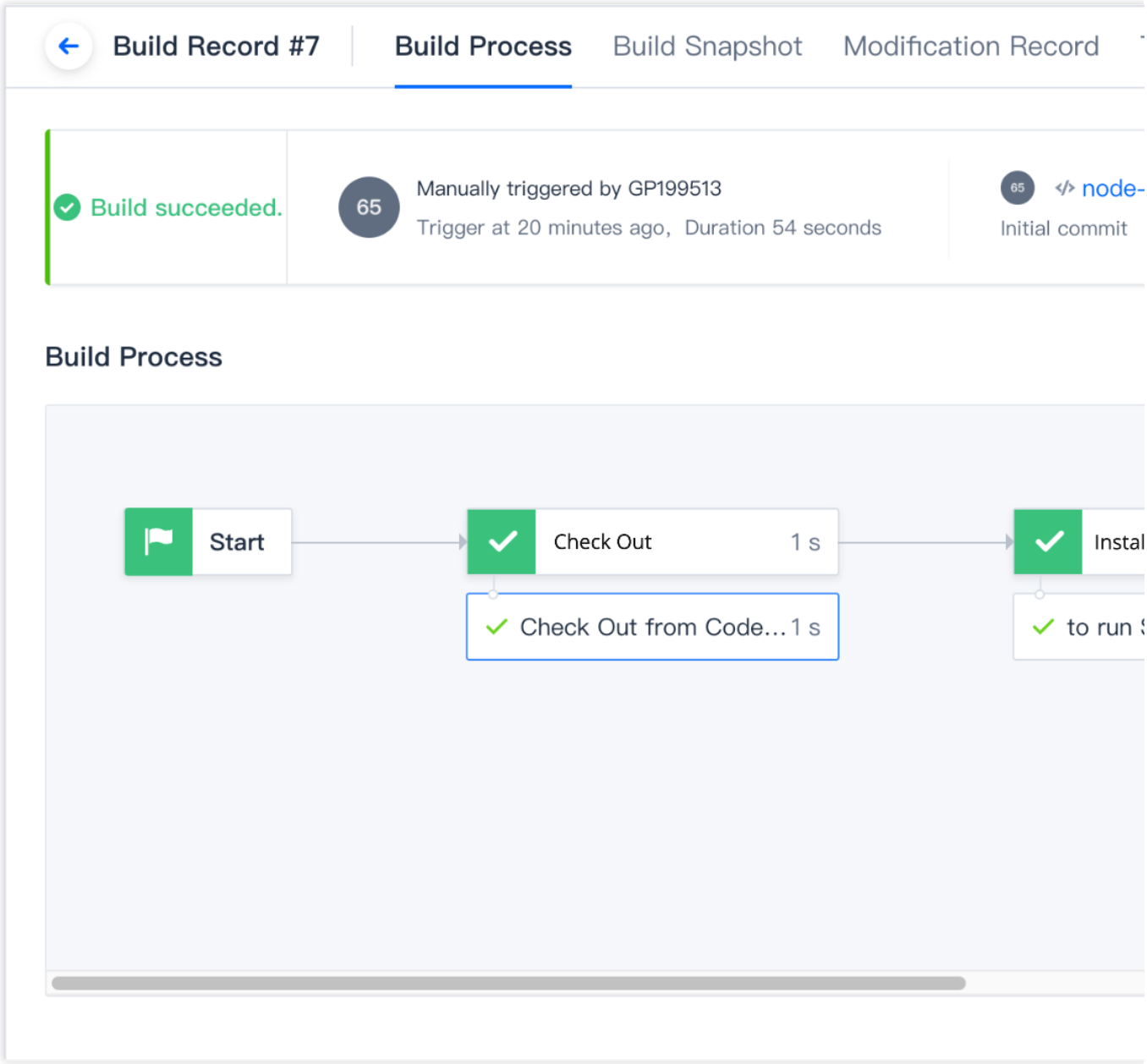
A change has occurred in the target branch of a merge request

When a merge request is merged

Note:

When a build is **triggered by a merge request**, the **result after the source branch and target branch is merged** is built, so integration errors can be identified at the earliest. This is not possible when a build is **triggered by code updates**.

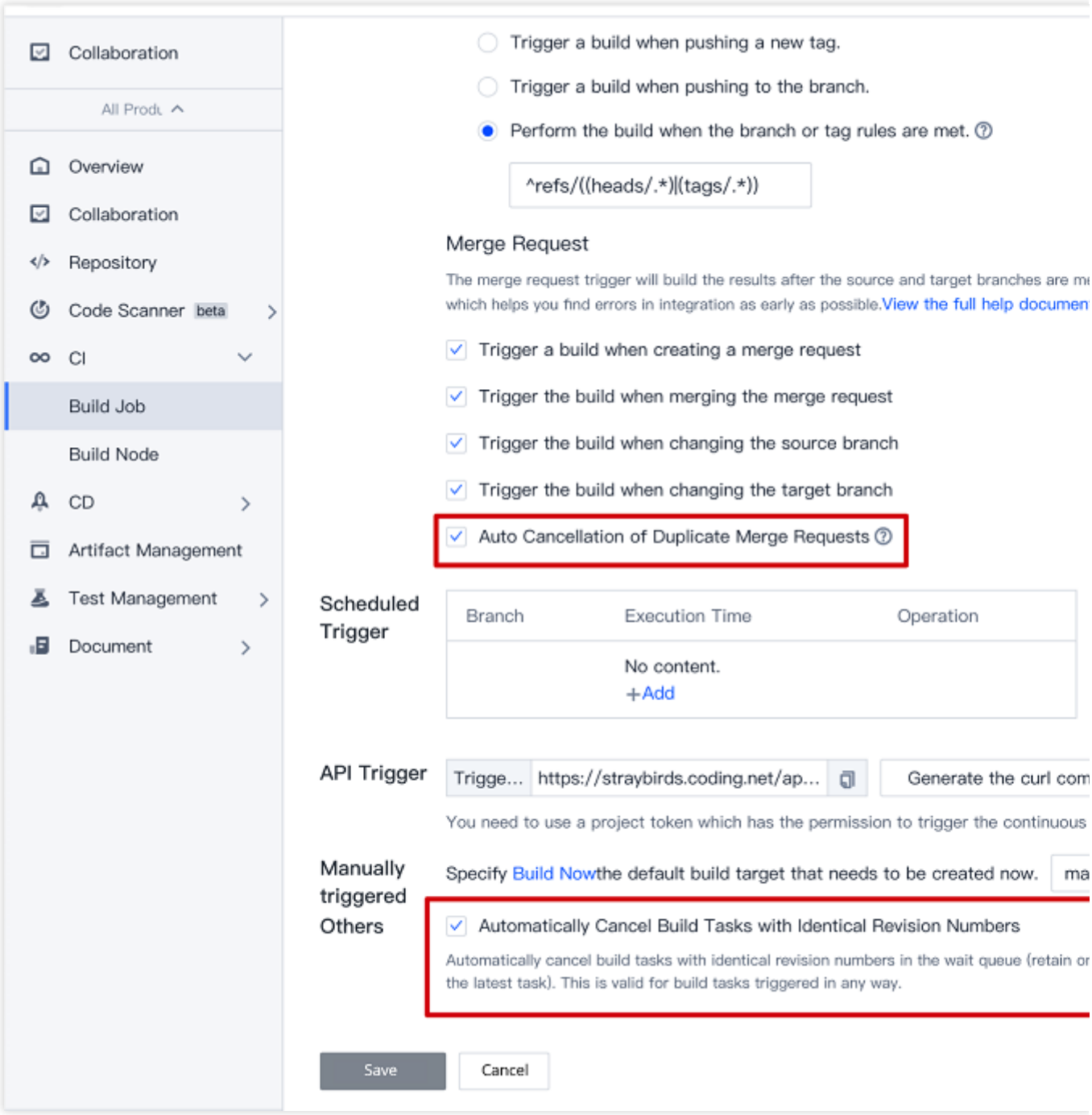
©2013-2022 Tencent Cloud. All rights reserved.



Automatically cancel identical builds

In the settings, you can select "Automatically cancel identical version numbers" and "Automatically cancel identical merge requests" to cancel identical builds that are triggered (and only keep the latest one).

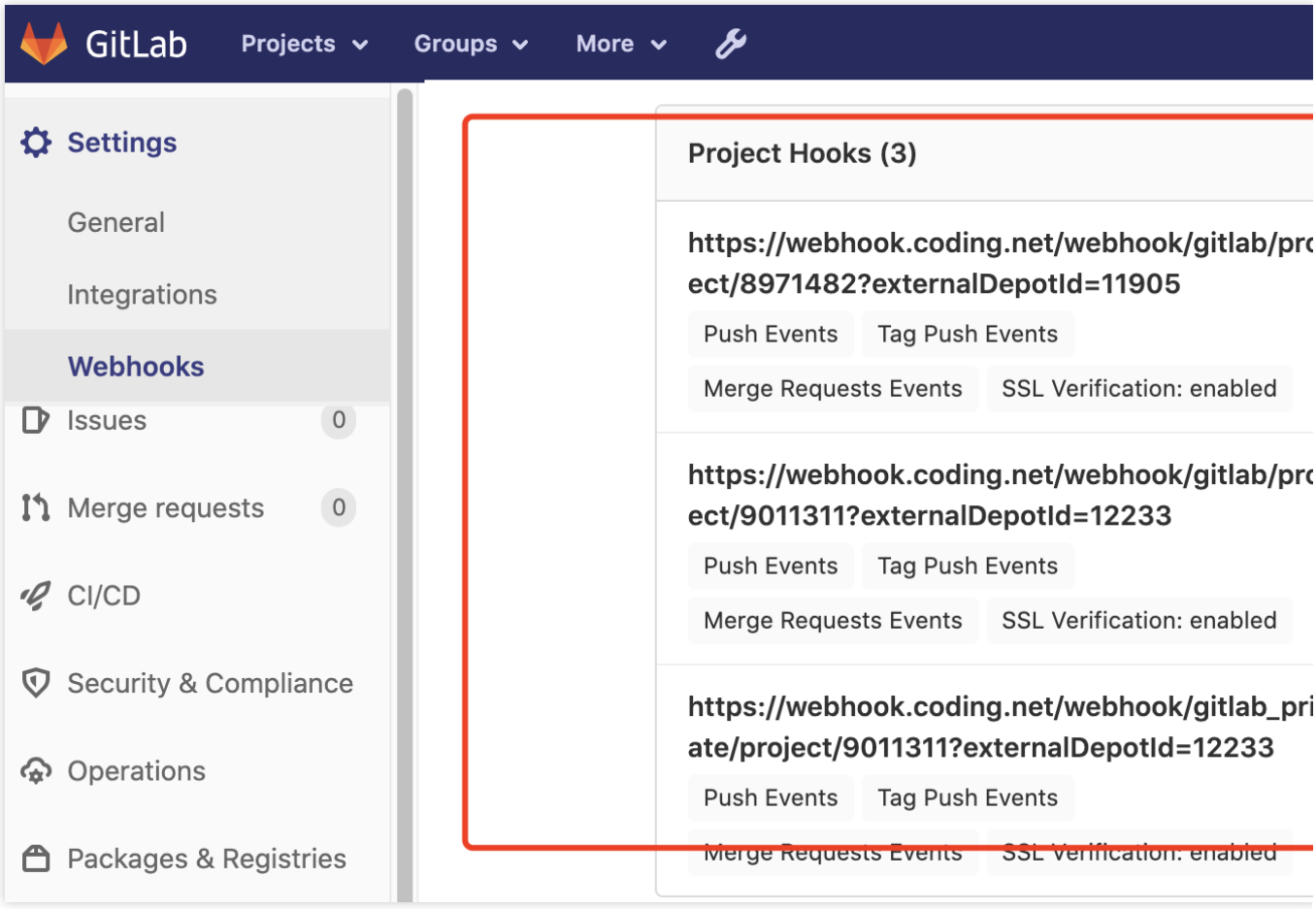
©2013-2022 Tencent Cloud. All rights reserved.



Private GitLab

In the case of Bind Private GitLab, GitLab Webhooks will be automatically created. Subsequently, CODING will be notified of events to match the above-set trigger rules.

©2013-2022 Tencent Cloud. All rights reserved.

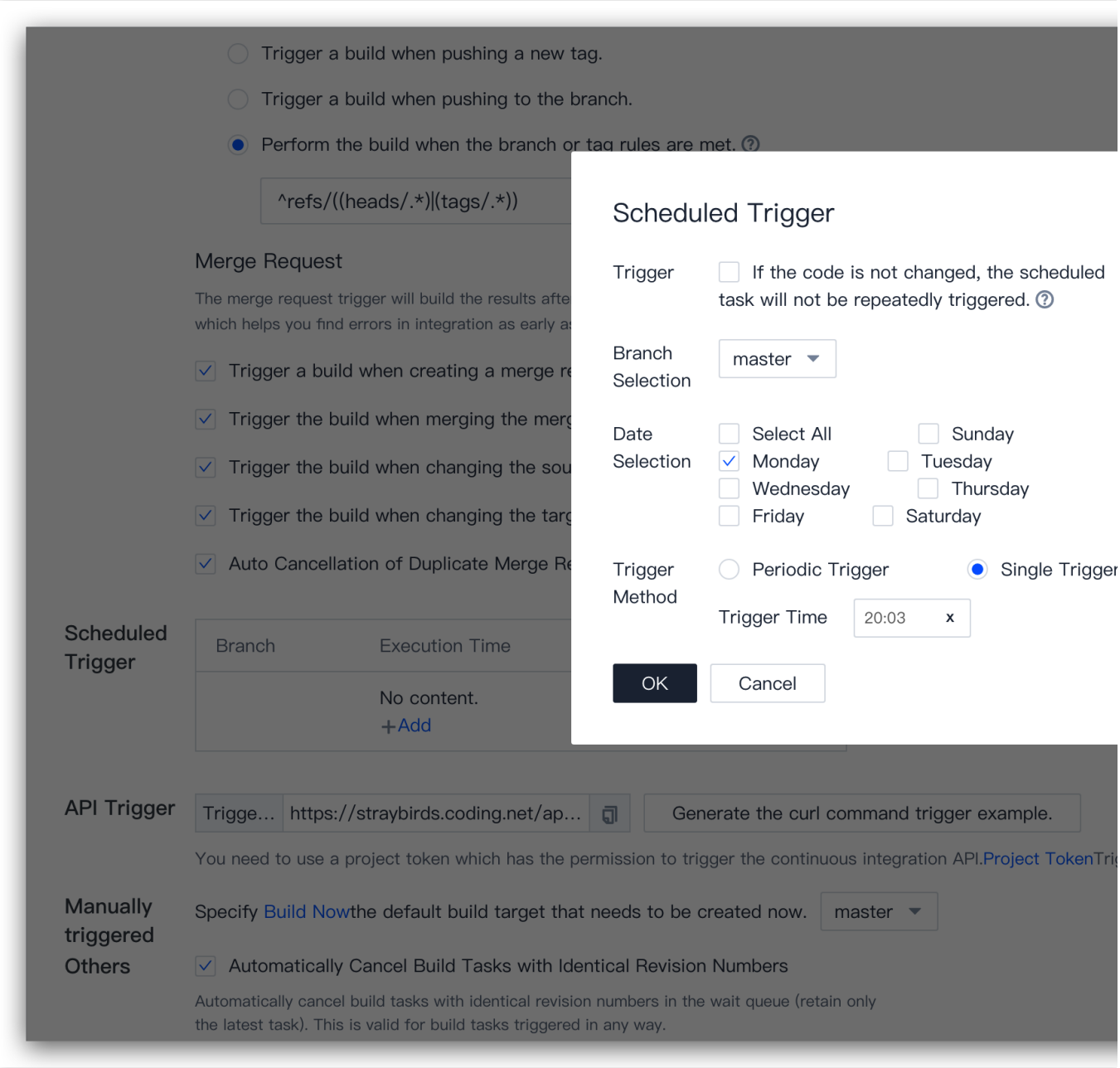


Scheduled trigger

By configuring a scheduled trigger for a build plan, you can set the build plan to be triggered periodically or at specific times to generate build tasks.

You can add multiple scheduled triggers in no order of priority. If scheduled triggers overlap, multiple builds are triggered.

©2013-2022 Tencent Cloud. All rights reserved.



Trigger condition: Do not repeatedly trigger scheduled task if code is unchanged

If the code of the selected branch has not changed since the last trigger, a build will not be triggered at the trigger time.

Select Date

You can select multiple days in a week.

Repeatedly

You can select a time range between 00:00 and 24:00 (accurate to the hour) to trigger tasks at the specified interval.

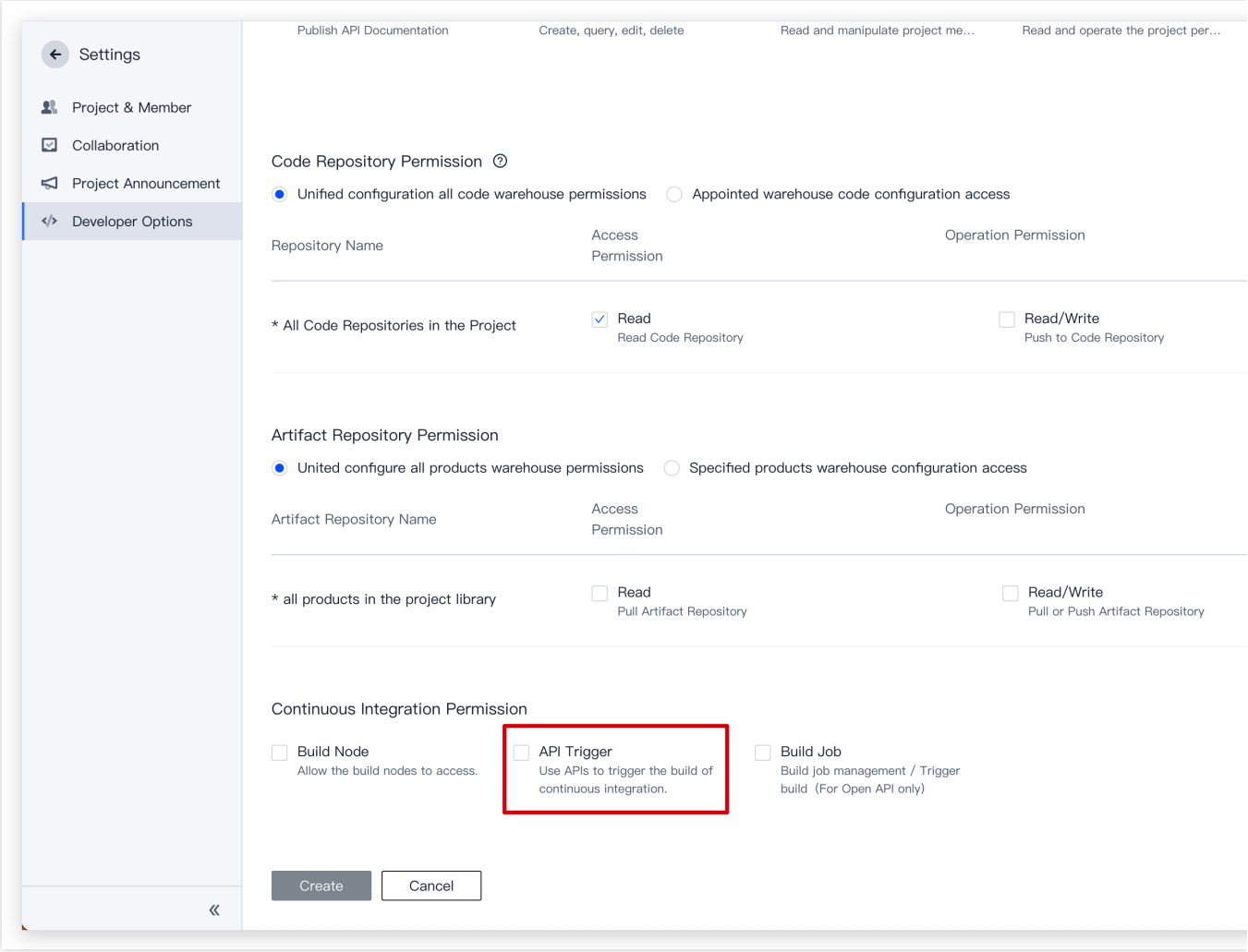
Once

You can select a trigger time between 00:00 and 24:00 (accurate to the minute).

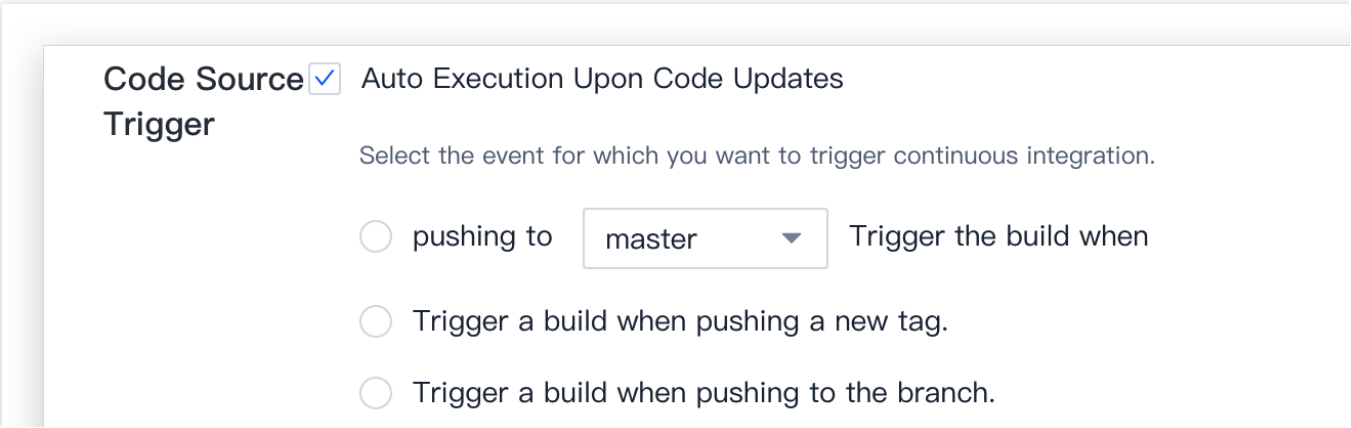
API trigger

©2013-2022 Tencent Cloud. All rights reserved.

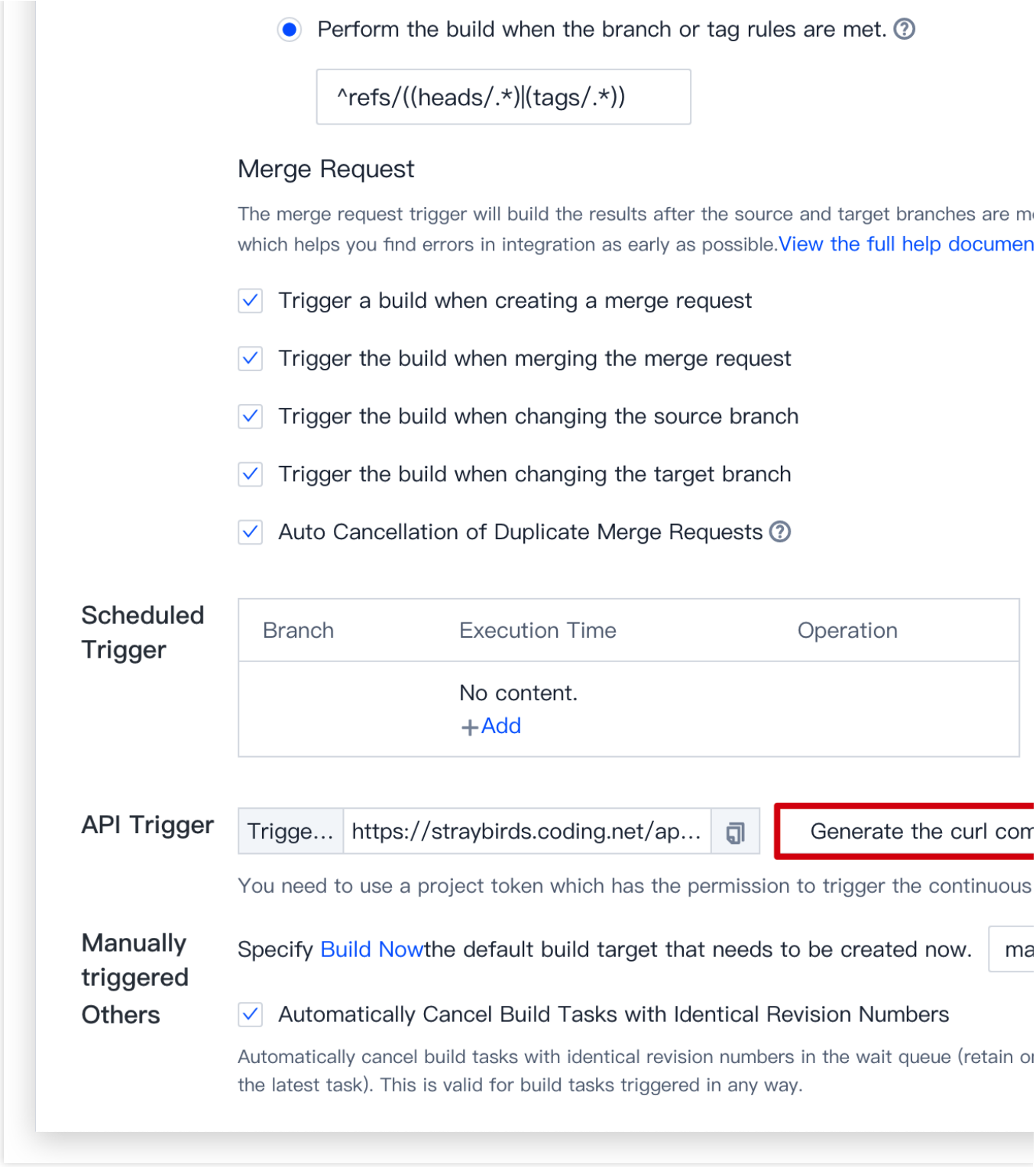
Before using this feature, select "Project Settings" > "Developer Options" > "Project Token" > "Create Token" to generate the token authorizing you to trigger the Continuous Integration API.



With the token, you can call the API in a build plan. Select "Generate cURL command to test trigger" to generate the call command.



©2013-2022 Tencent Cloud. All rights reserved.

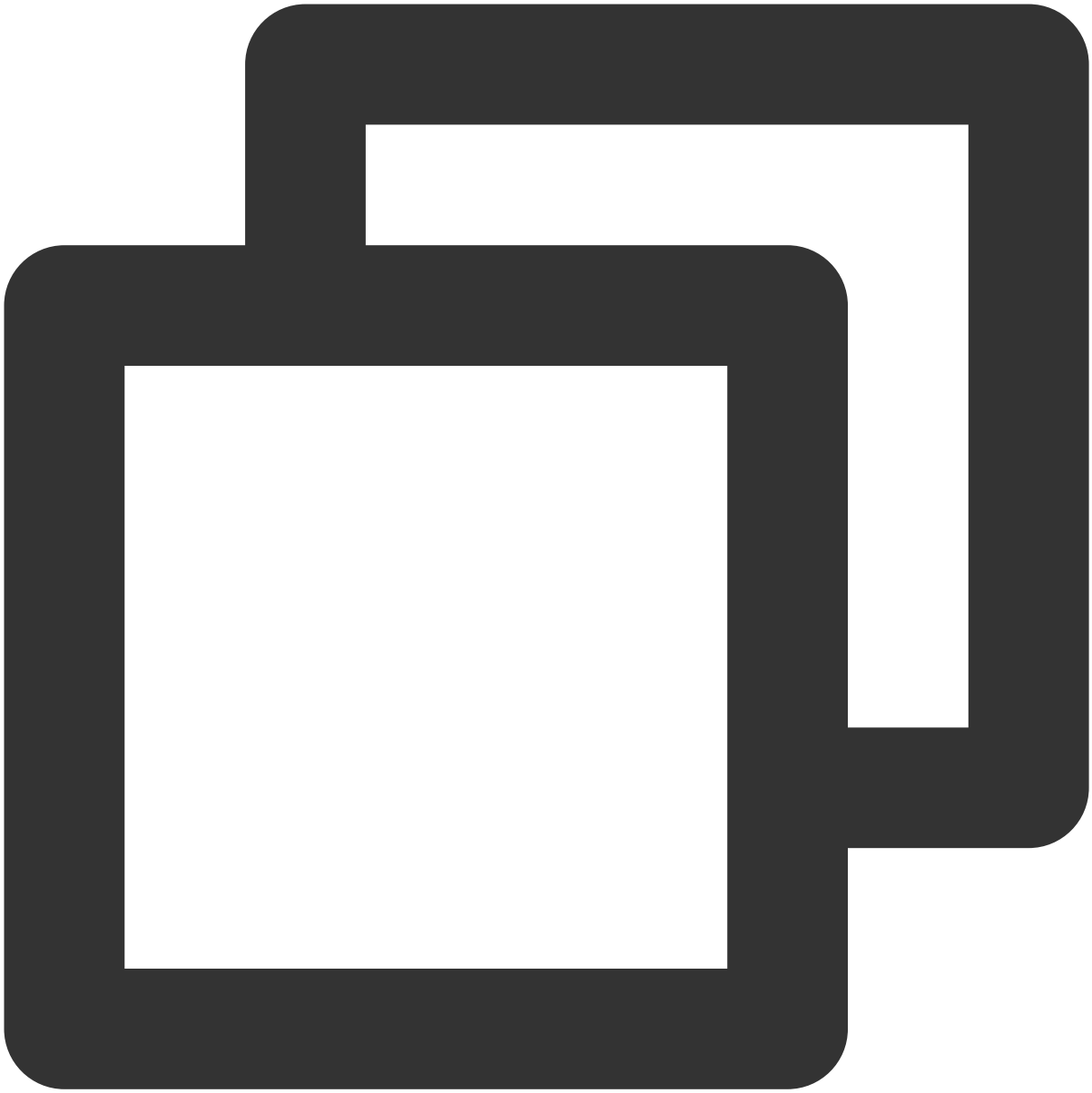


API

When the project token calls the CODING Continuous Integration (CODING-CI) API, the authentication method is `Basic Auth` . See the API parameters below.

Trigger build task

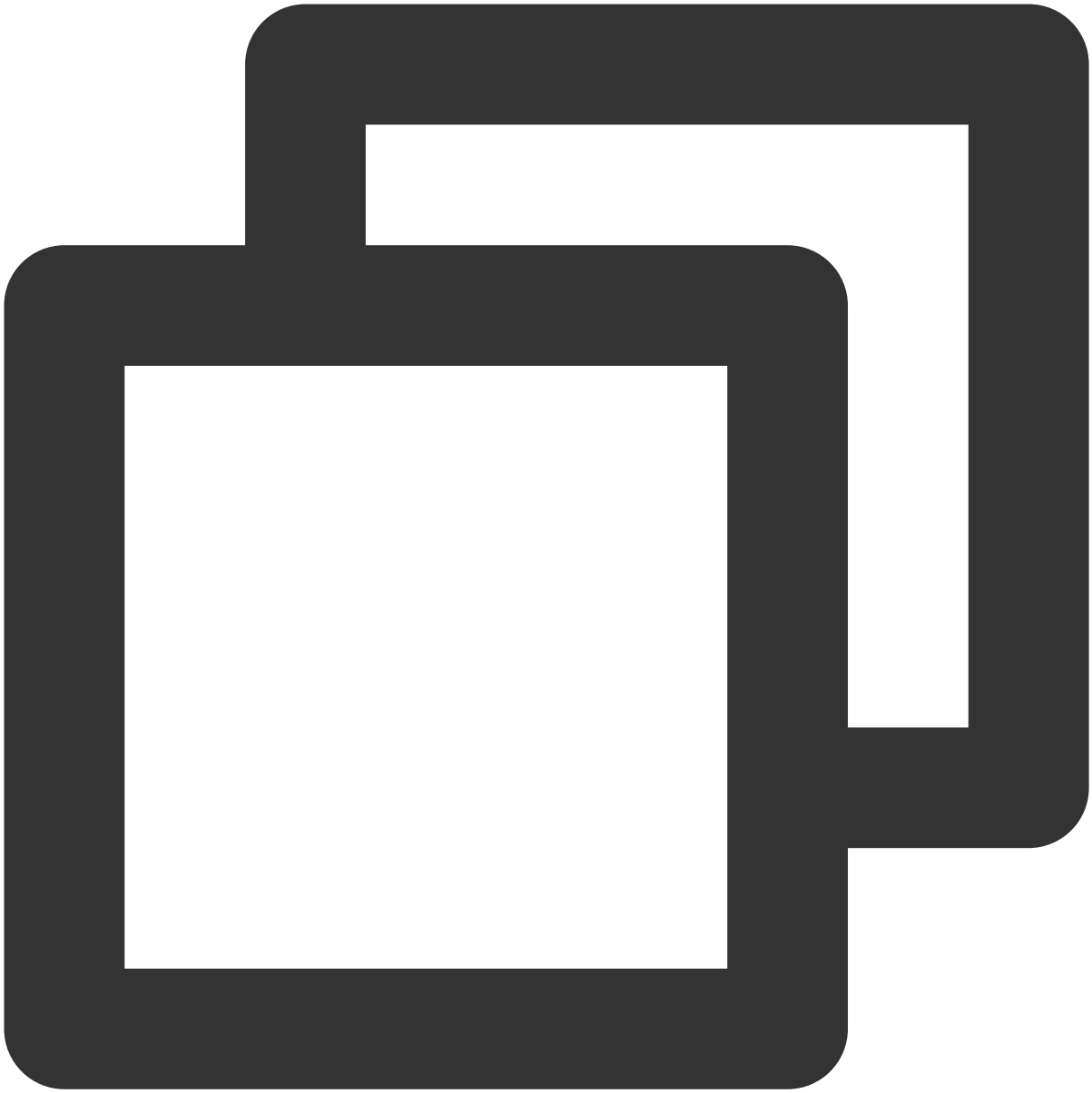
©2013-2022 Tencent Cloud. All rights reserved.



```
POST https://< TEAM_GK >.coding.net/api/cci/job/< JOB_ID >/trigger
```

Request body

©2013-2022 Tencent Cloud. All rights reserved.

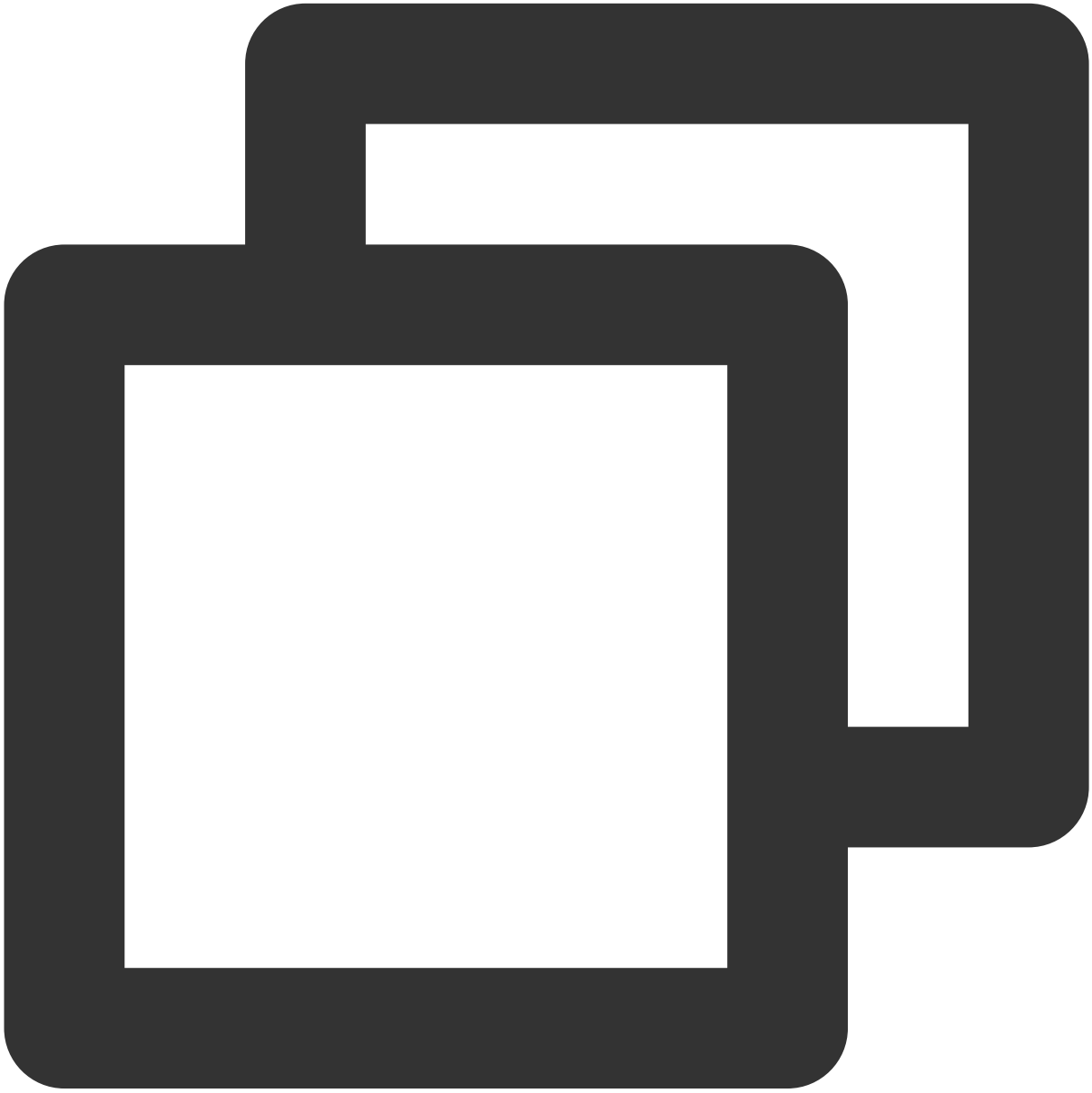


```
{
  "ref": "master",
  "envs": [
    {
      "name": "my-params-1",
      "value": "hello",
      "sensitive": 1
    },
    {
      "name": "my-params-2",
      "value": "world",
```

©2013-2022 Tencent Cloud. All rights reserved.

```
        "sensitive": 0
      }
    ]
}
```

Return body



```
{
  "code": 0
}
```

©2013-2022 Tencent Cloud. All rights reserved.

Parameter descriptions

Parameter	Location	Required?	Type	Default value	Description
ref	body	No	string	master	Built target ref (`commit sha` / `tag` / `branch`), ignore if code repository isn't used in build plan
envs	body	No	env[]	-	Startup parameter of build plan

envItem

Parameter	Location	Required?	Type	Default value	Description
name	envItem.name	Yes	string	master	Name of startup parameter of build plan
value	envItem.value	No	string	-	Startup value of build plan
sensitive	envItem.sensitive	No	number	0	Whether to keep startup parameters confidential and do not show startup parameters in log 1: confidential, 0: plaintext

==== 2021/08/24 ====

©2013-2022 Tencent Cloud. All rights reserved.

Environment Variables

Last updated：2023-12-29 11:44:51

title: Environment Variables - CODING Help Center
pageTitle: Environment Variables
pagePrevTitle: Trigger Rules
pagePrev: ci/configuration/trigger.html
pageNextTitle: Build Snapshots
pageNext: ci/configuration/snapshot.html
alias:
devops/ci/env.html
ci/env.html

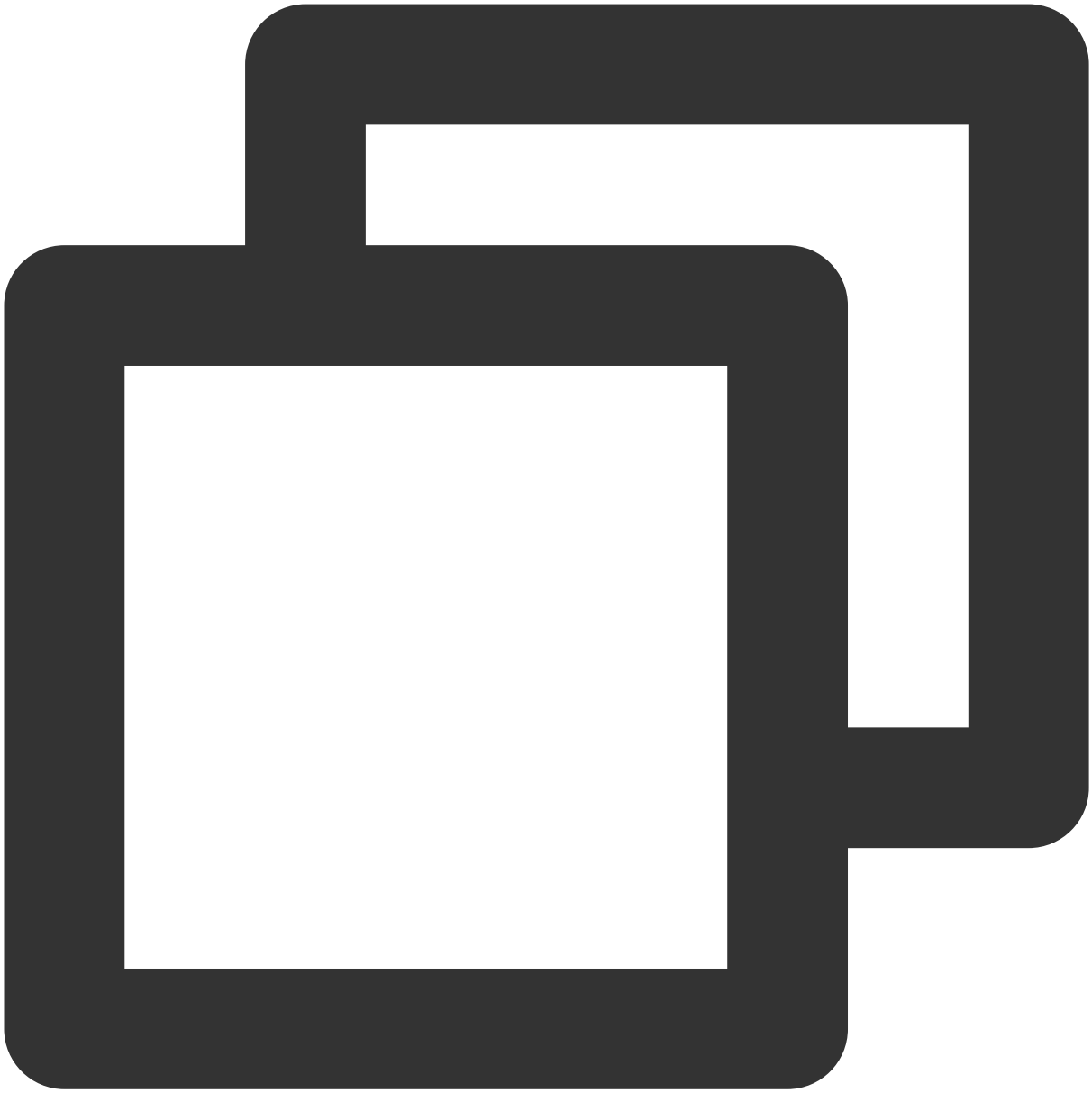
Function Overview

In a continuous integration process, we may incorporate configurations (such as an account password or version number) into the build process as environment variables. CODING Continuous Integration (CODING-CI) supports environment variables in multiple formats. You can incorporate variables into a build process with the following methods (in order of highest priority to lowest):
"withEnv" in a Jenkinsfile
"environment" in a Jenkinsfile
Startup parameters in a build plan (job)
Environment variables in a build plan (job)
Built-in system environment variables during a build process
This document describes these methods in detail.

withEnv and environment

You can use "environment" to define environment variables in a Jenkinsfile (as follows):

©2013-2022 Tencent Cloud. All rights reserved.



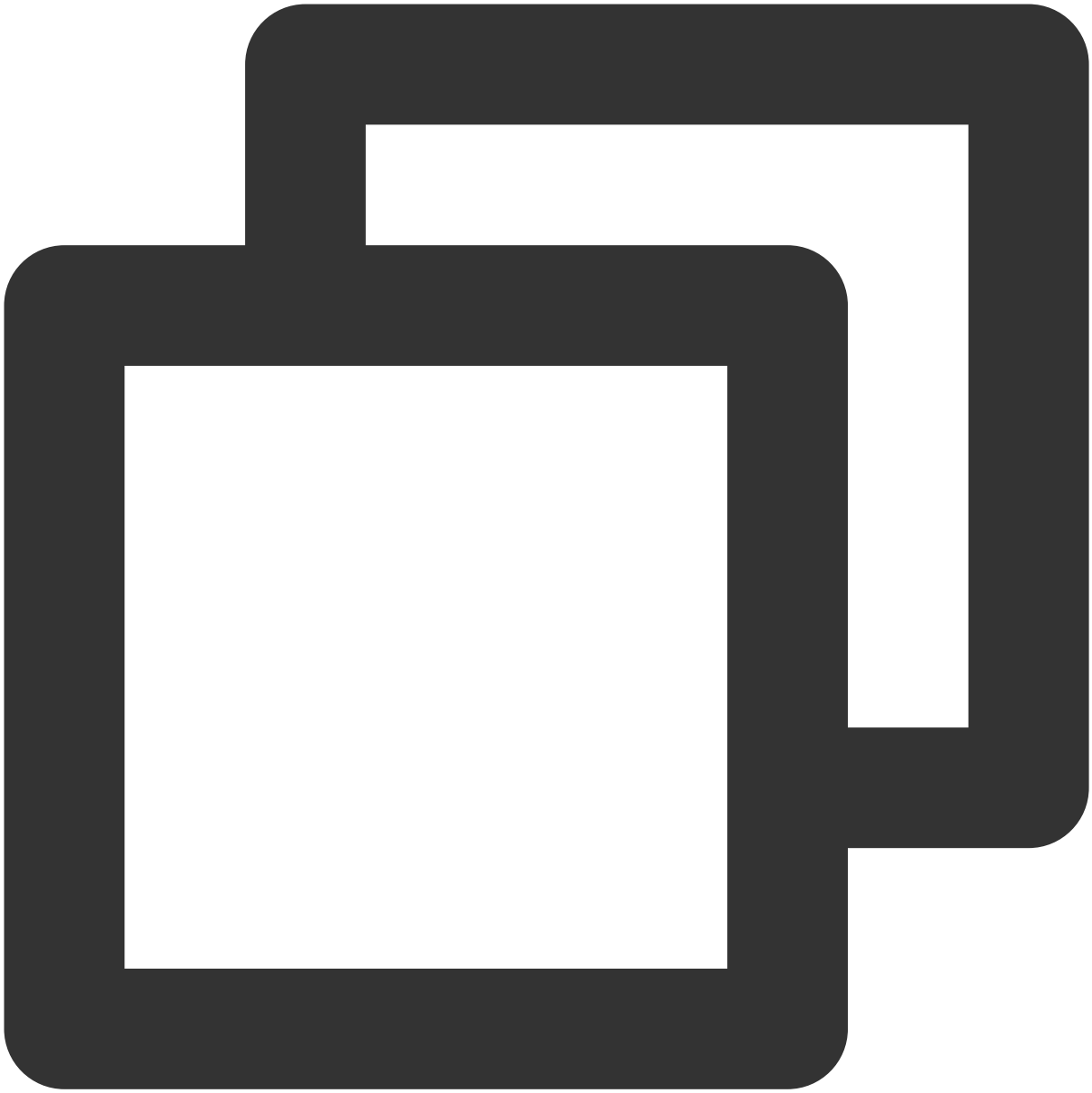
```
pipeline {
    agent any
    environment {
        MY_PROJECT = 'project-1'
        MY_TEAM    = 'team-1'
    }
    stages {
        stage('Build') {
            steps {

                echo "MY_PROJECT is ${MY_PROJECT}"
```

©2013-2022 Tencent Cloud. All rights reserved.

```
                echo "MY_TEAM is ${MY_TEAM}"
                // The output is as follows:
                // MY_PROJECT is project-1
                // MY_TEAM is team-1
            }
        }
    }
}
```

In a build process, you may need to use environment variables of the same name at different stages. You can use "withEnv" to set the environment variables for some operations to avoid confusing the global environment variables. Steps executed with "withEnv" will prioritize the set environment variables. Refer to the following example:

©2013-2022 Tencent Cloud. All rights reserved.



```
pipeline {
    agent any
    environment {
        MY_PROJECT = 'project-1'
        MY_TEAM    = 'team-1'
    }
    stages {
        stage('Build') {
            steps {

                echo "MY_PROJECT is ${MY_PROJECT}"
```

©2013-2022 Tencent Cloud. All rights reserved.

```
                echo "MY_TEAM is ${MY_TEAM}"
                // The output is as follows:
                // MY_PROJECT is project-1
                // MY_TEAM is team-1

                // Environment variables set with "withEnv" are only valid for step
                withEnv(['MY_PROJECT=project-2']) {

                    echo "MY_PROJECT is ${MY_PROJECT}"
                    echo "MY_TEAM is ${MY_TEAM}"
                    // The output is as follows:
                    // MY_PROJECT is project-2
                    // MY_TEAM is team-1

                }
            }
        }
    }
}
```

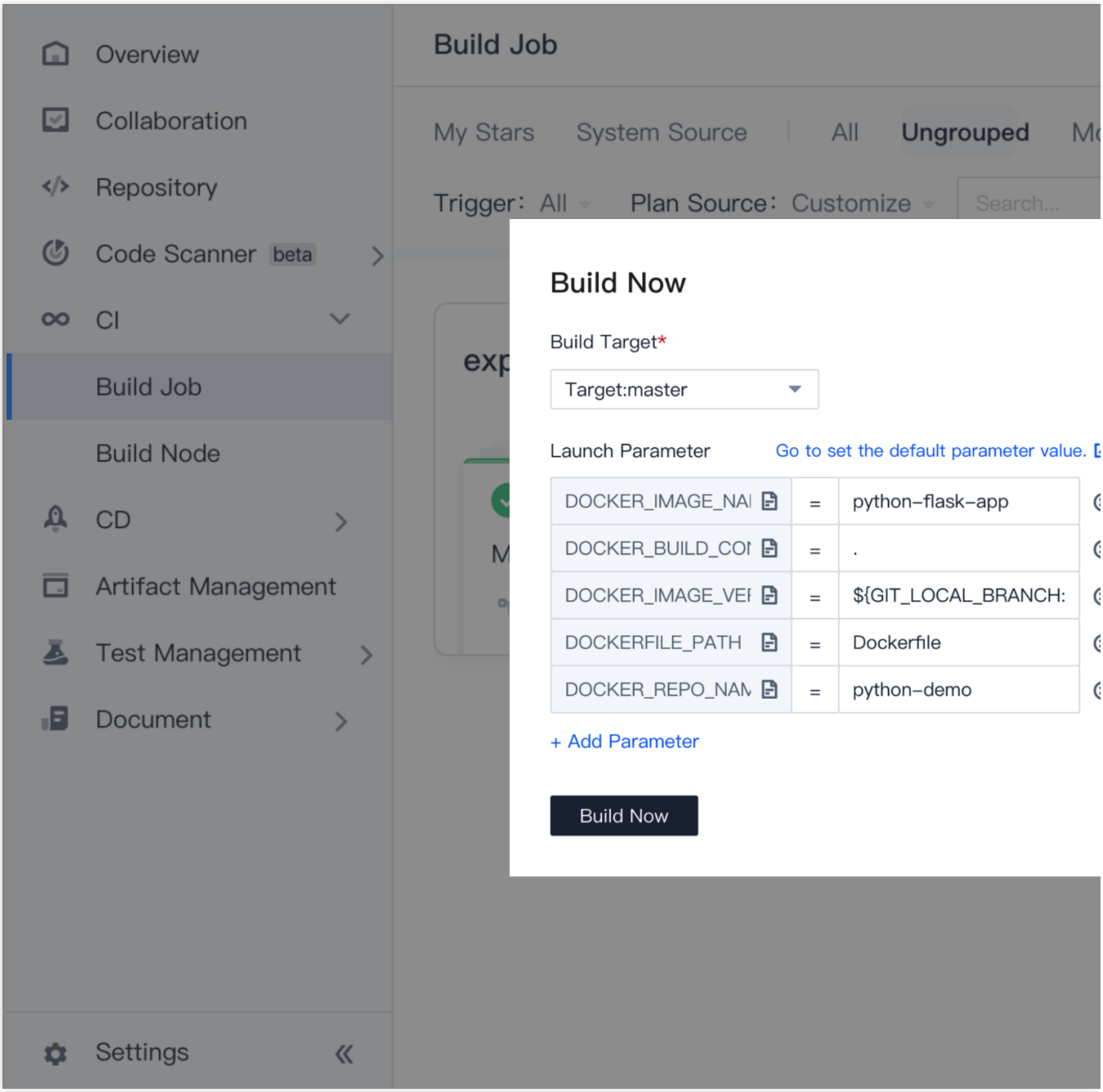
Note:

For more information, see the official Jenkins documentation—Using environment variables.

Startup parameters in build plans

Startup parameters are the next most important environment variables. You can select or fill in their values when starting a build plan.

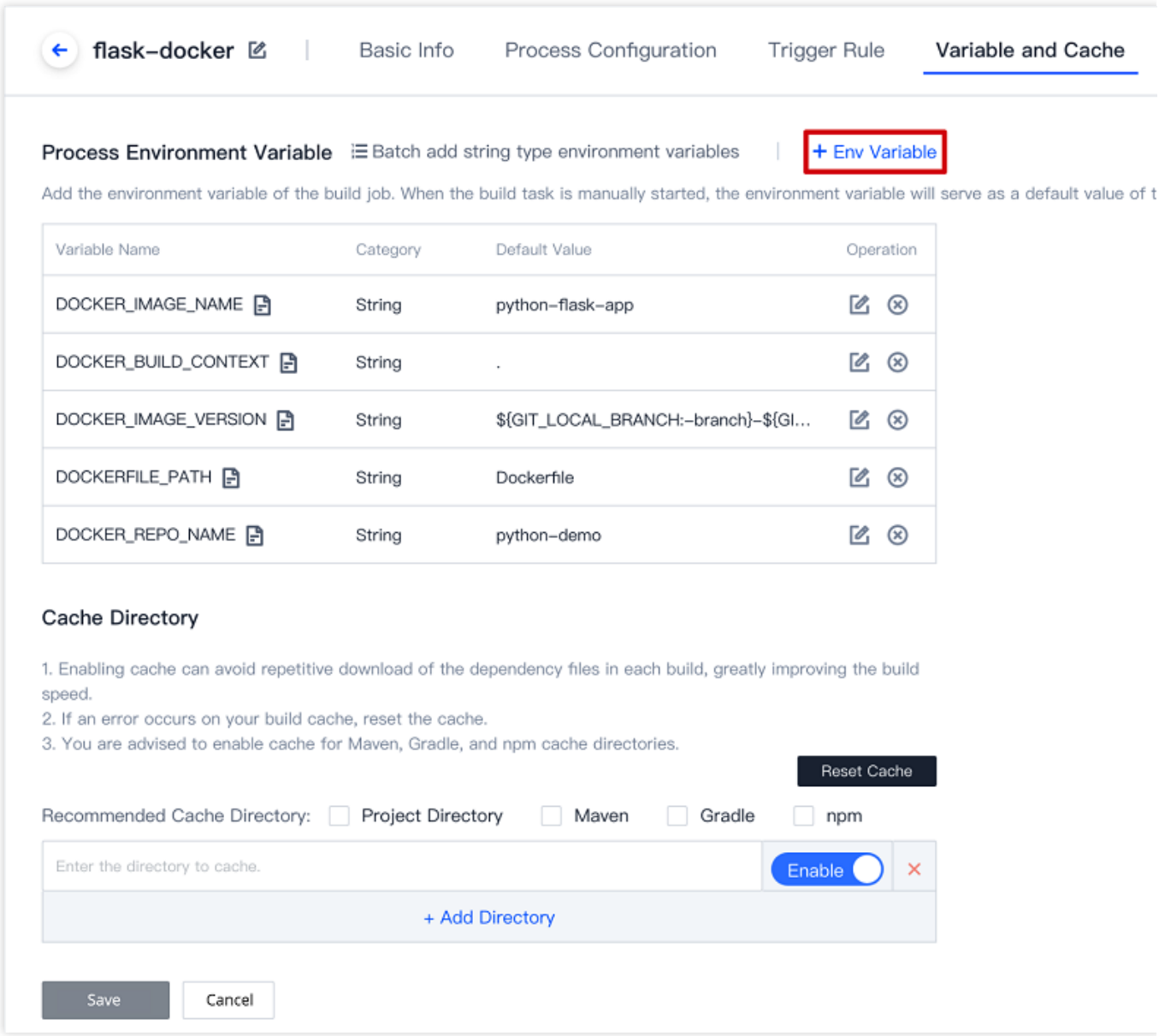
©2013-2022 Tencent Cloud. All rights reserved.



Add environment variable

Besides hardcoding environment variables into a Jenkinsfile, you can also set variables when configuring a build plan. CODING supports four types of environment variables: string, single-selection, multi-selection, and CODING credentials. You can also configure environment variables in a build plan as the default values of the startup parameters.

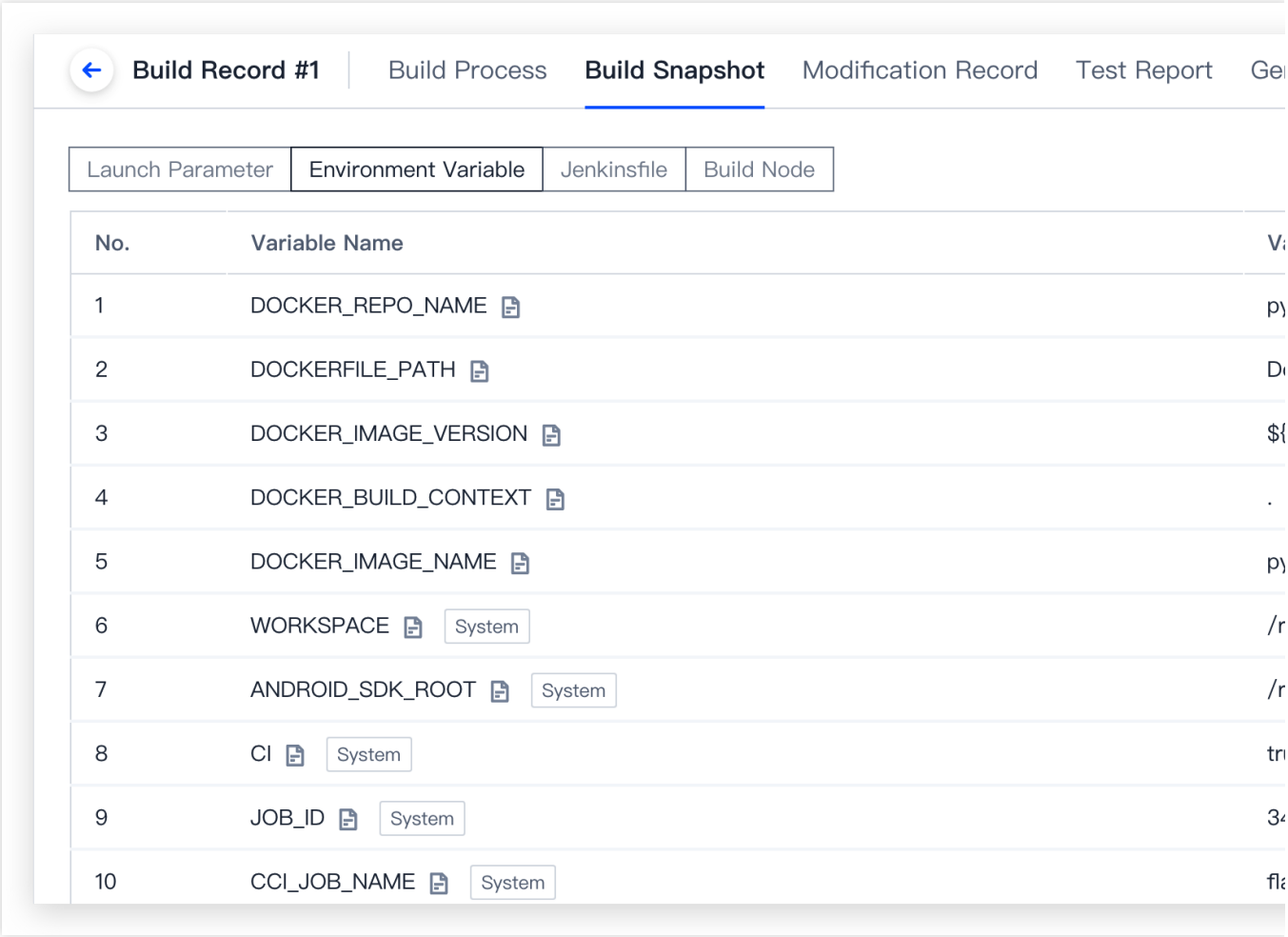
©2013-2022 Tencent Cloud. All rights reserved.



Built-in system environment variables

In CODING-CI build processes, he corresponding environment variables are incorporated to every build task. You can view the list of default environment variables in "Build Snapshot":

©2013-2022 Tencent Cloud. All rights reserved.



The environment variables are summarized below by trigger rules (triggered upon code updates, scheduled trigger, or triggered upon merge requests):

No.	Variable	Description
1	CREDENTIALS_ID	Private deploy key CredentialsId, for pullir repositories
2	DOCKER_REGISTRY_CREDENTIALS_ID	Docker private key CredentialsId (equival to CODING_ARTIFACTS_CREDENTIALS_
3	CREDENTIALS_ID	Repository private key CredentialsId, for pulling repositories in the project
4	GIT_HTTP_URL	Code repository HTTPS URL
5	GIT_BUILD_REF	The Git revision number for the build

©2013-2022 Tencent Cloud. All rights reserved.

6	GIT_DEPLOY_KEY	Public deploy key of code repositories
7	GIT_COMMIT	Revision number of the current version
7	GIT_COMMIT_SHORT	First seven digits of the revision number
8	GIT_PREVIOUS_COMMIT	Revision number of the last build run No.
9	GIT_AUTHOR_EMAIL	Email address of the latest author of this version
10	GIT_SSH_URL	SSH URL of the code repository
11	GIT_COMMITTER_NAME	Name of the latest committer of this versic
12	GIT_AUTHOR_NAME	Name of the latest author of this version
13	REF	Version to be built
14	GIT_PREVIOUS_COMMIT	Revision number of the last successful bu run
15	GIT_COMMITTER_EMAIL	Email address of the latest committer of th version
16	GIT_BRANCH	Branch triggering the build
17	GIT_URL	SSH URL of the repository
18	GIT_LOCAL_BRANCH/BRANCH_NAME	Local branch name
19	FETCH_REF_SPECS	refs to be checked out by git
20	GIT_REPO_URL	SSH URL of the repository
21	JOB_ID	Build plan ID
22	JOB_NAME	Build plan name
23	CI_BUILD_NUMBER	Build No.
24	PROJECT_ID	Project ID
25	PROJECT_NAME	Project name
26	PROJECT_WEB_URL	Project website URL
27	PROJECT_API_URL	URL of project's backend API
28	PROJECT_TOKEN	Project token password, for reading the

©2013-2022 Tencent Cloud. All rights reserved.

		project
29	PROJECT_TOKEN_GK	Project token user
30	GIT_TAG	Git tag triggering the build (only applicable when building with tags)
31	DEPOT_NAME	Current code repository name
32	CCI_CURRENT_PROJECT_COMMON_CREDENTIALS_ID (soon to be released)	Built-in project token's CredentialsId
33	CCI_CURRENT_TEAM (soon to be released)	Company name for the current build environment, such as "myteam" in myteam.coding.net
34	CCI_CURRENT_DOMAIN (soon to be released)	Domain name of the current build environment, such as "coding.net" in myteam.coding.net
35	MR_RESOURCE_ID	Merge request ID
36	MR_TARGET_BRANCH	Target branch of the merge request
37	MR_TARGET_SHA	Version number of the target branch of the merge request
38	MR_MERGED_SHA	Simulated merged version number
39	MR_SOURCE_BRANCH	Source branch of the merge request
40	MR_STATUS	Status of the merge request
41	MR_SOURCE_SHA	Version number of the source branch of th merge request

==== 2020/10/14 ====

©2013-2022 Tencent Cloud. All rights reserved.

Build Snapshots

Last updated：2023-12-29 11:44:51

title: Build Snapshots - CODING Help Center
pageTitle: Build Snapshots
pagePrevTitle: Environment Variables
pagePrev: ci/configuration/env.html
pageNextTitle: Cache Directory
pageNext: ci/configuration/cache.html
alias:
devops/ci/snapshot.html
ci/snapshot.html

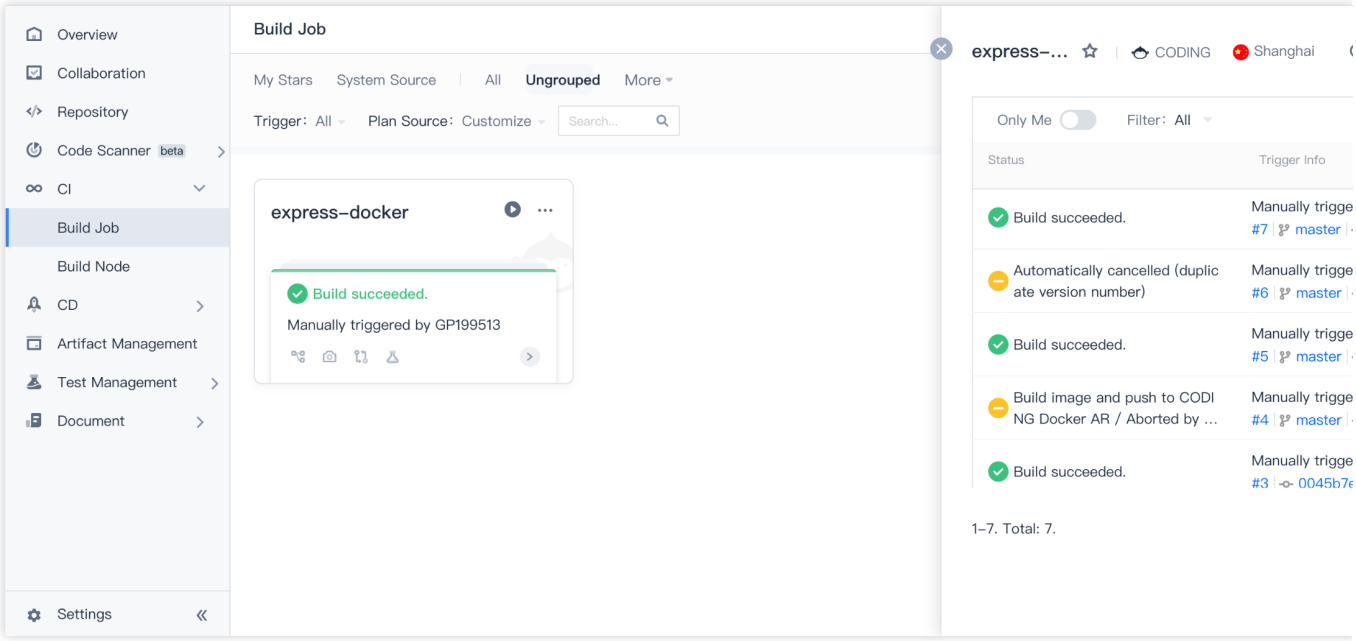
Function overview

You may use different configuration files or build parameters for each build task in Continuous Integration. CODING Continuous Integration (CODING-CI) features build snapshots to allow you to review the execution process of a build task. Build snapshots clearly show the configuration parameters of every build record.

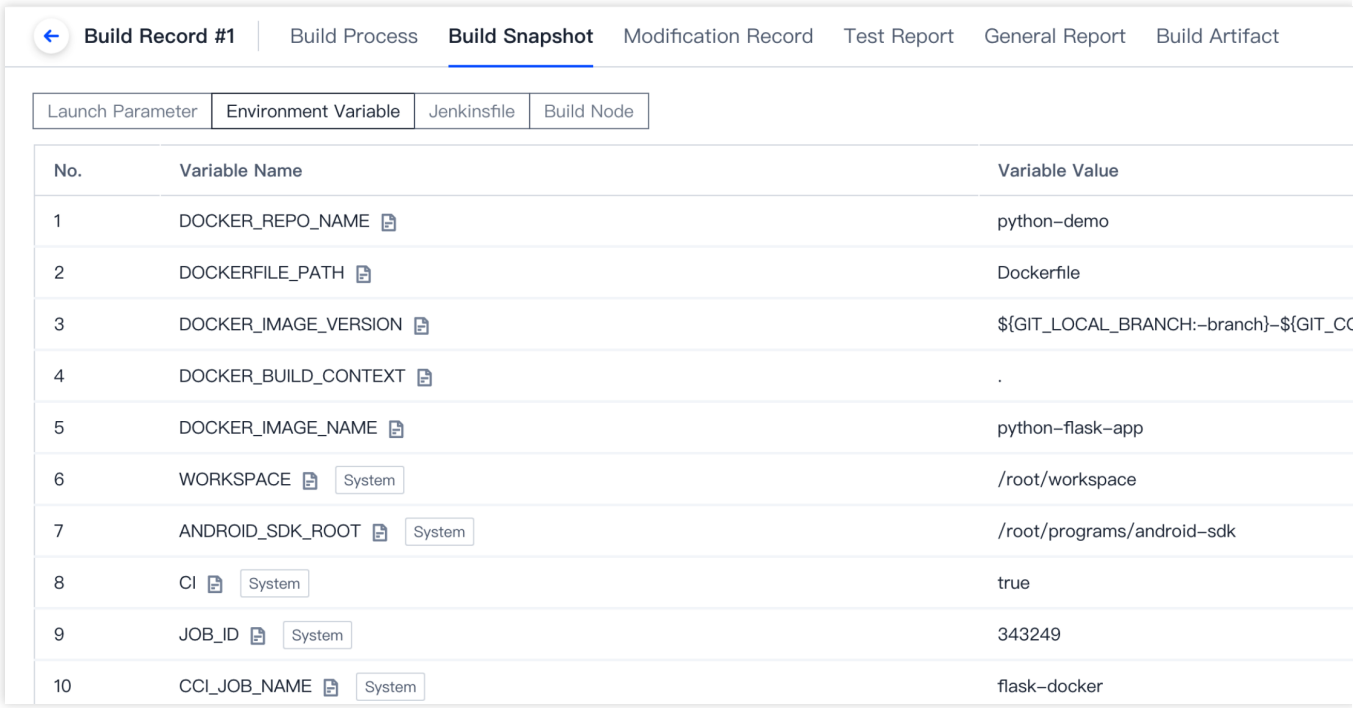
View build configuration

1. In a project, click "Continuous Integration" > "Build Plans". Select the title of a build plan to view all build records of the plan. Click a record to open it:

©2013-2022 Tencent Cloud. All rights reserved.



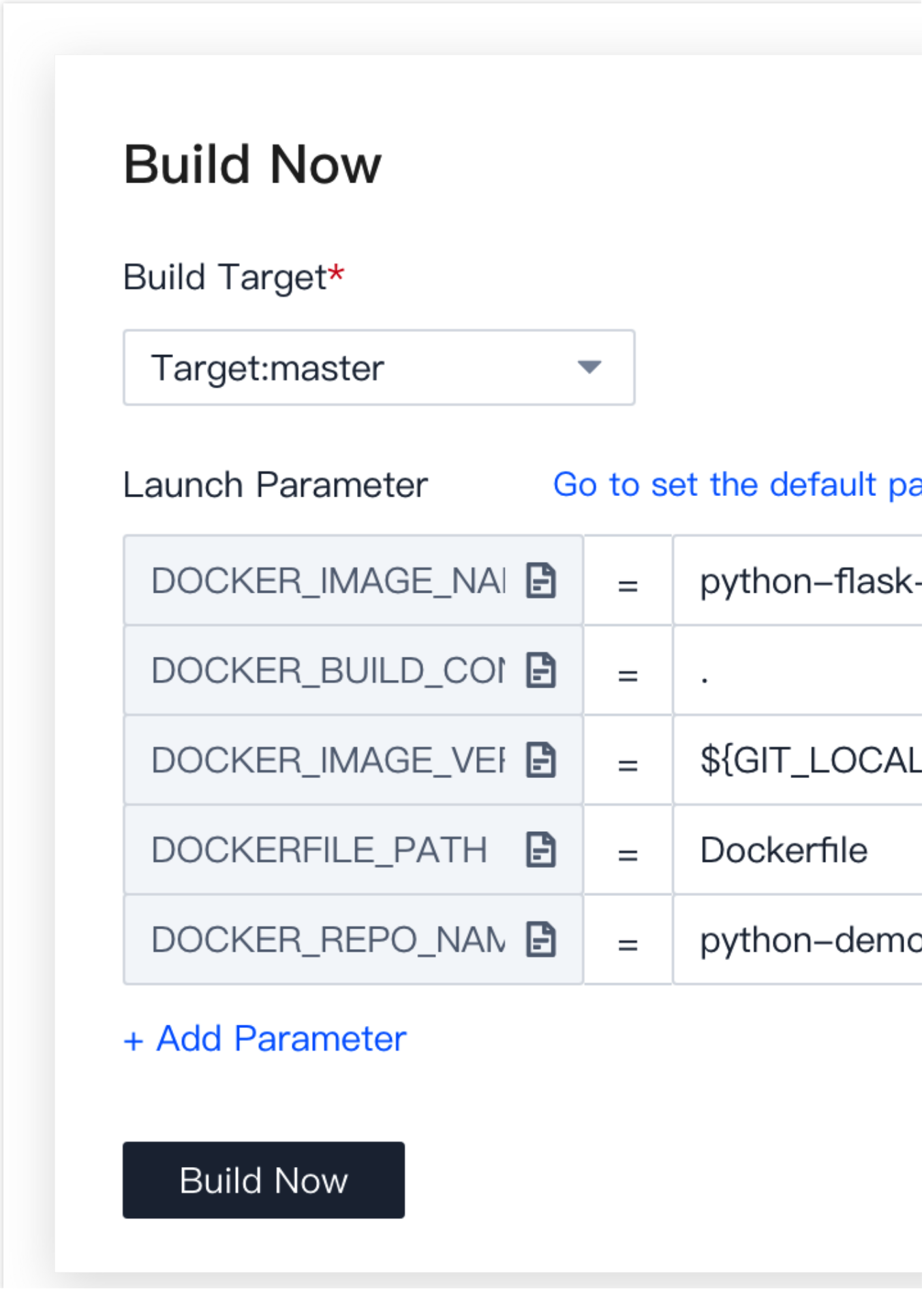
2. In a build record, you can click "Build Snapshot" to view the configuration snapshots of the build record—the startup parameters, environment variables, and process configuration file.



Startup parameters

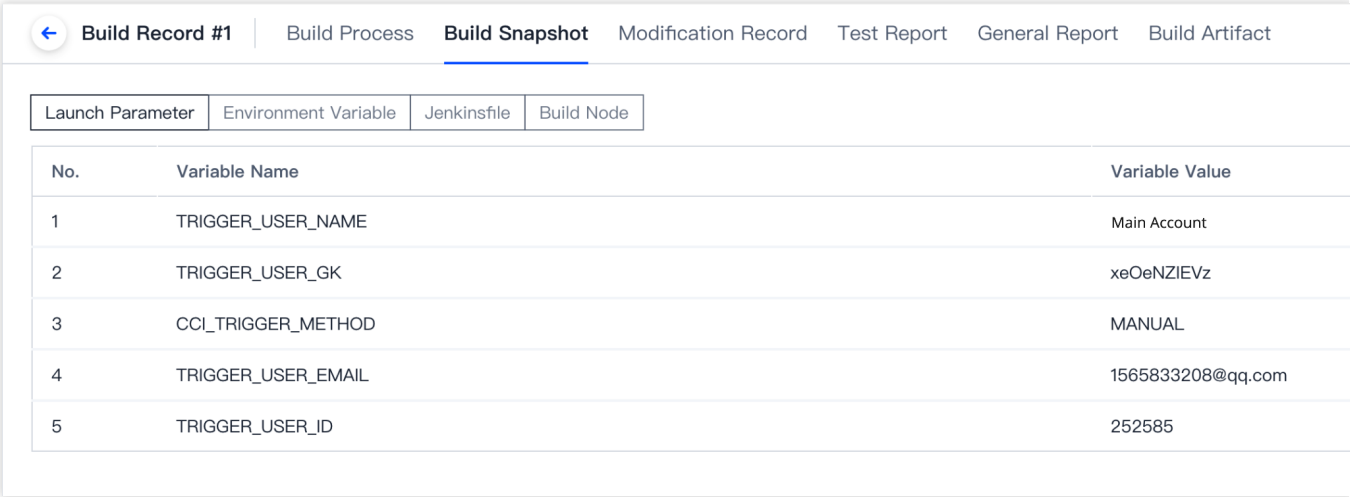
The startup parameters are the parameters that you entered when starting a build task. They are incorporated into the run environment of the build task as environment variables.

©2013-2022 Tencent Cloud. All rights reserved.



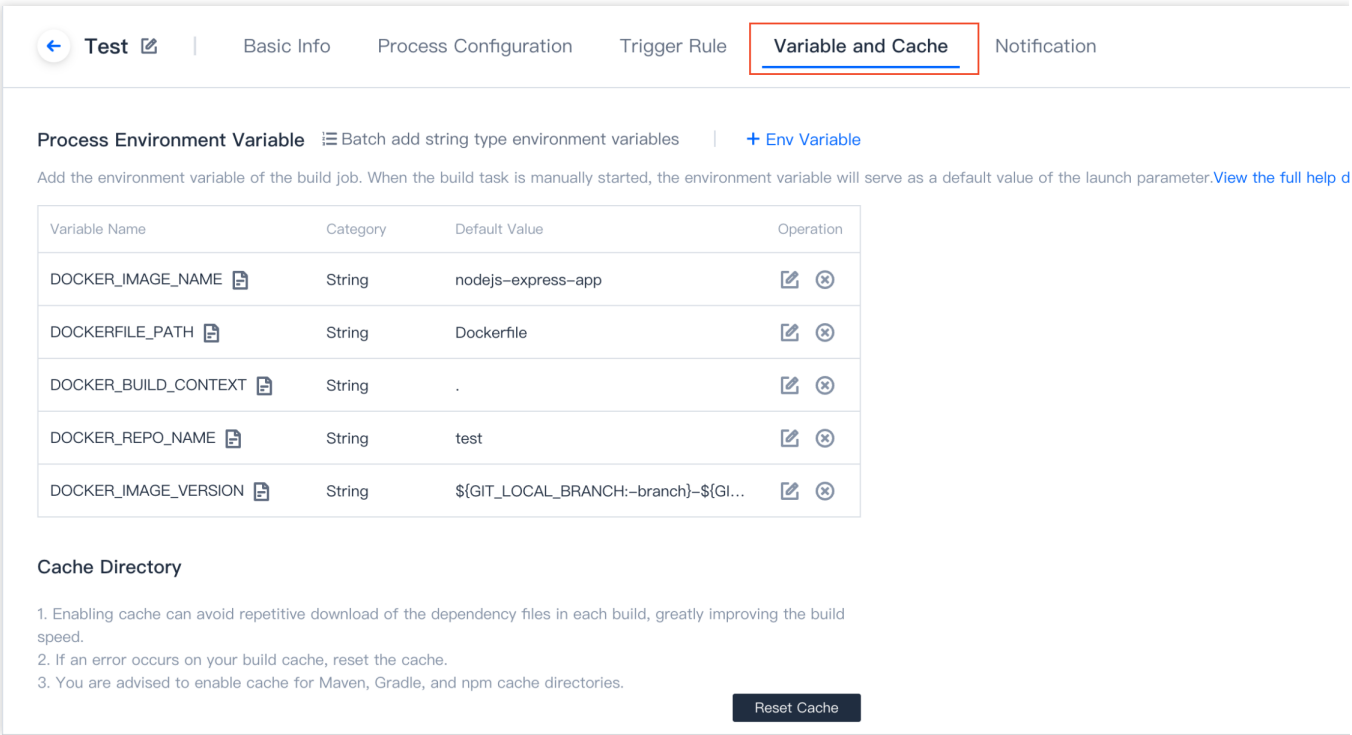
©2013-2022 Tencent Cloud. All rights reserved.

After the build is completed, you can view the configured startup parameters in "Build Snapshot".



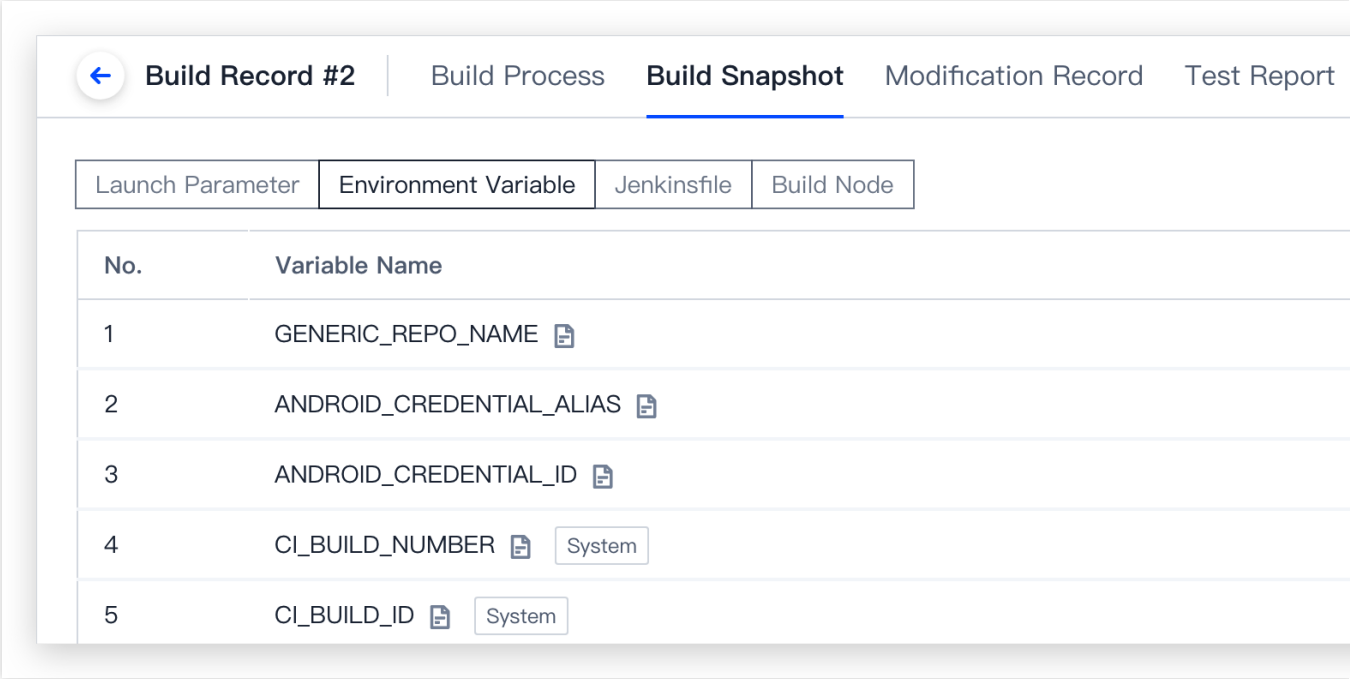
Environment variables

The environment variables only include those you configured when starting the task, and exclude all environment variables generated or dynamically set in the run process.



©2013-2022 Tencent Cloud. All rights reserved.

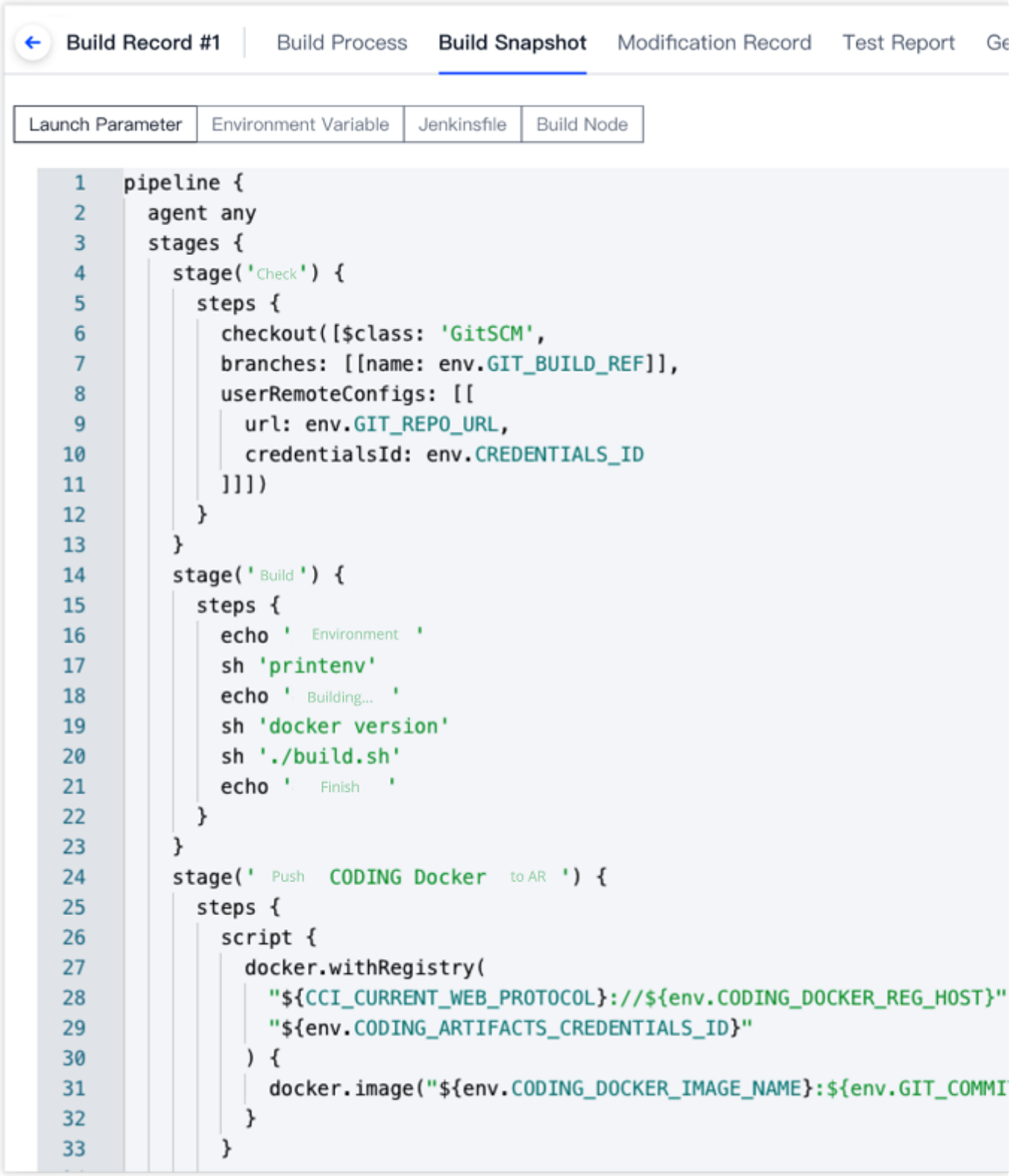
Select "Environment Variables" to view the environment variables set by the system and user for the build task when the task was started.



Process configuration file

Select the tab for the process configuration to view the configuration file (Jenkinsfile) used for the build record.

©2013-2022 Tencent Cloud. All rights reserved.



==== 2020/08/13 ====

©2013-2022 Tencent Cloud. All rights reserved.

Cache Directory

Last updated：2023-12-29 11:44:51

title: Cache Directory - CODING Help Center
pageTitle: Cache Directory
pagePrevTitle: Build Snapshots
pagePrev: ci/configuration/snapshot.html
pageNextTitle: Build Node Types
pageNext: ci/node/type.html
alias:
devops/ci/cache.html
ci/cache.html
practice/jenkins-dockerfile.html

Function Overview

When installing dependencies for a local project, the downloaded files are cached for the next installation. After you run the `npm install` command, `./node_modules` is generated in the project and is cached in the `~/.npm` directory, which is more compact and universal.
Default build nodes
CODING distributes computing resources for each build plan, which are destroyed once a build is finished. As a new build node is assigned for each build, the "cache directory" needs to be specified to accelerate the next build.
Custom build nodes
If you would like to access computing resources and execute the task in the build plan using a custom build node, the server will not be destroyed once the build is finished, so you do not need to specify the "cache directory".
When using Docker in Continuous Integration, you will need to enable the "cache directory" in Docker.

Default build nodes

CODING provides the basic task computing resources for build plans. A CVM is assigned for each task in a Linux build environment with root user permissions, the cache directory is as follows:

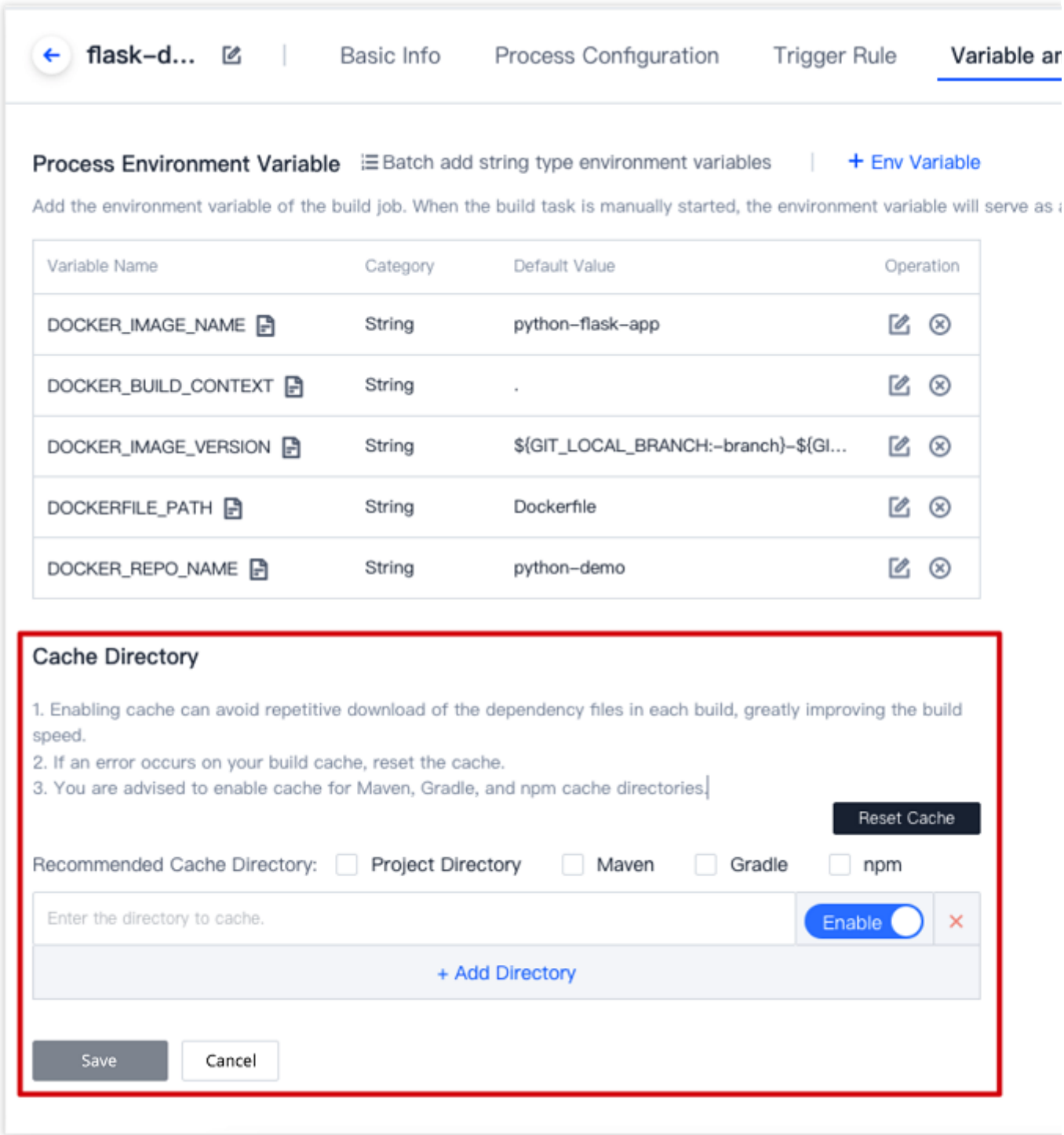
Package management tool	Cache directory
Maven	/root/.m2/
Gradle	/root/.gradle/
npm	/root/.npm/

©2013-2022 Tencent Cloud. All rights reserved.

composer	/root/.cache/composer/
pip3	/root/.cache/pip/
yarn	/usr/local/share/.cache/yarn/

In "Build Plan Settings" > "Variables and Caches", select or enter the cache directory.

©2013-2022 Tencent Cloud. All rights reserved.

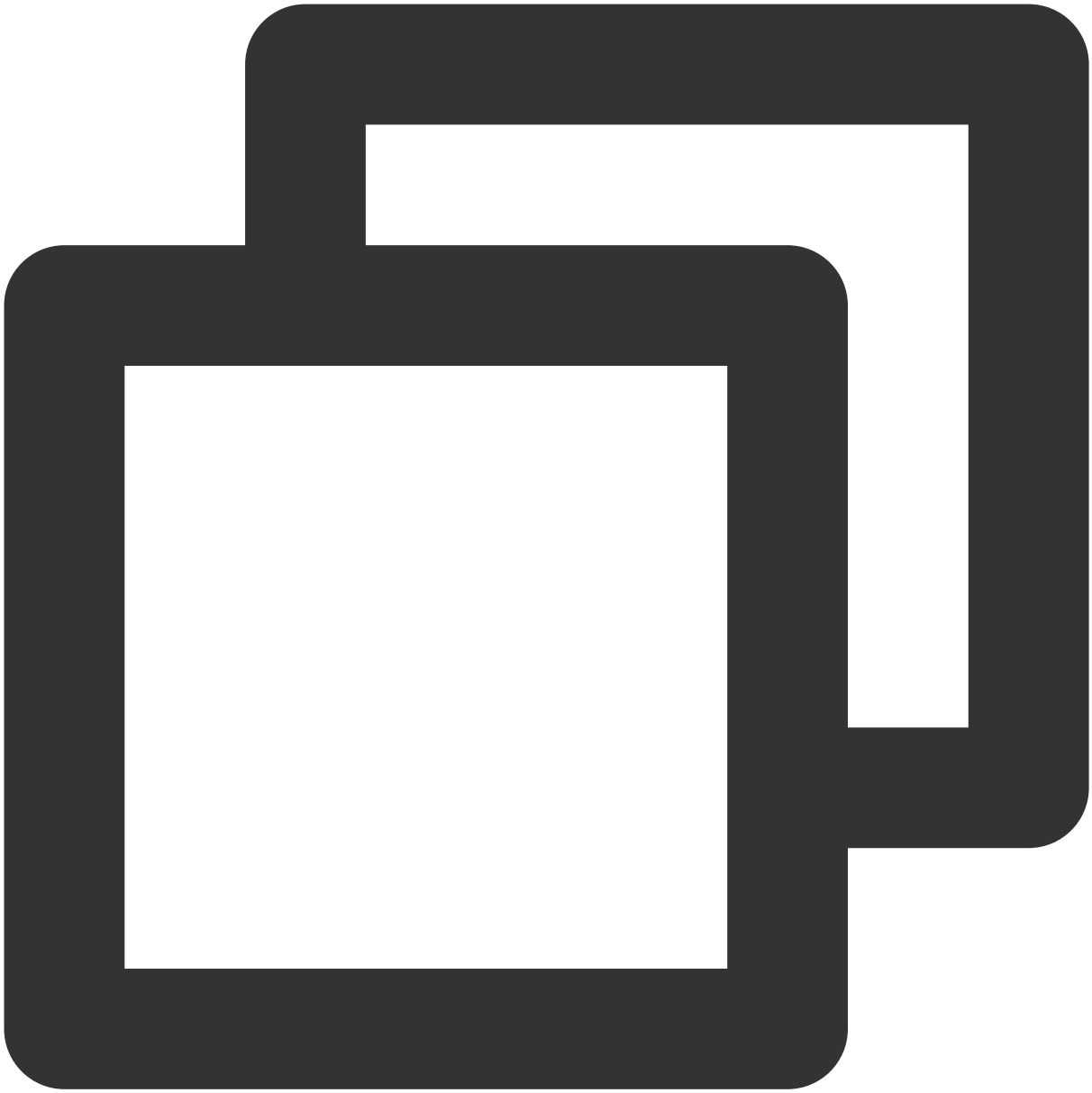


Docker build environment

If you are using Docker in a build plan, go to "Variables and Caches", select the cache directory, and then enable it in Docker.

Jenkinsfile

©2013-2022 Tencent Cloud. All rights reserved.



```
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.CREDENTIAL
        ])
      }
```

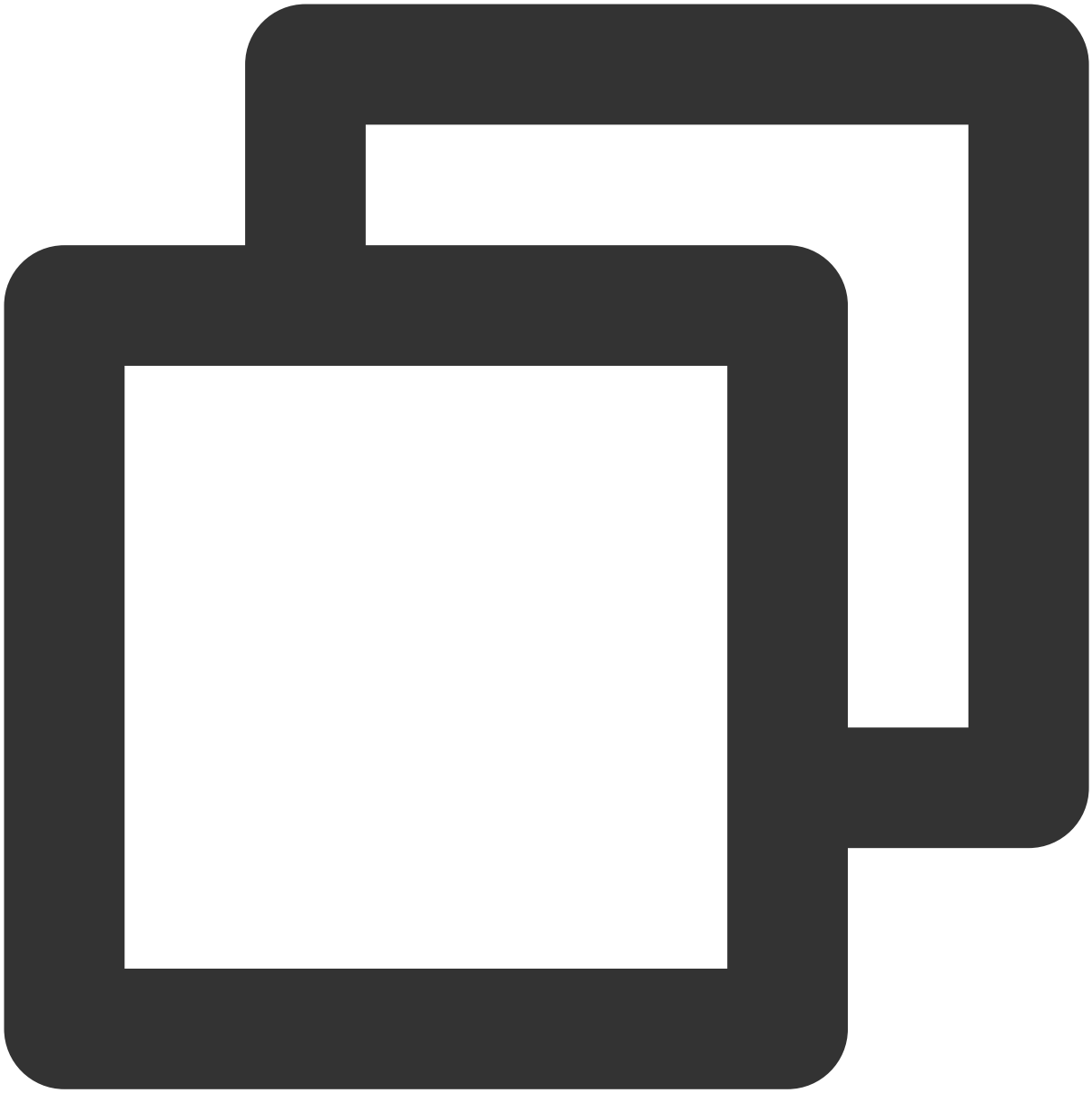
©2013-2022 Tencent Cloud. All rights reserved.

```
    }
    stage('Java cache') {
      agent {
        docker {
          image 'adoptopenjdk:11-jdk-hotspot'
          args '-v /root/.gradle/:/root/.gradle/ -v /root/.m2/:/root/.m2/'
          reuseNode true
        }
      }
      steps {
        sh './gradlew test'
      }
    }
    stage('npm cache') {
      steps {
        script {
          docker.image('node:14').inside('-v /root/.npm/:/root/.npm/') {
            sh 'npm install'
          }
        }
      }
    }
  }
}
```

Custom build nodes

When using a Docker environment in a custom build node, find the cache directory corresponding to the server username. For example, if the default username for Ubuntu servers is ubuntu, the cache directory is `/home/ubuntu/.npm/` , and the codes are as follows:

©2013-2022 Tencent Cloud. All rights reserved.



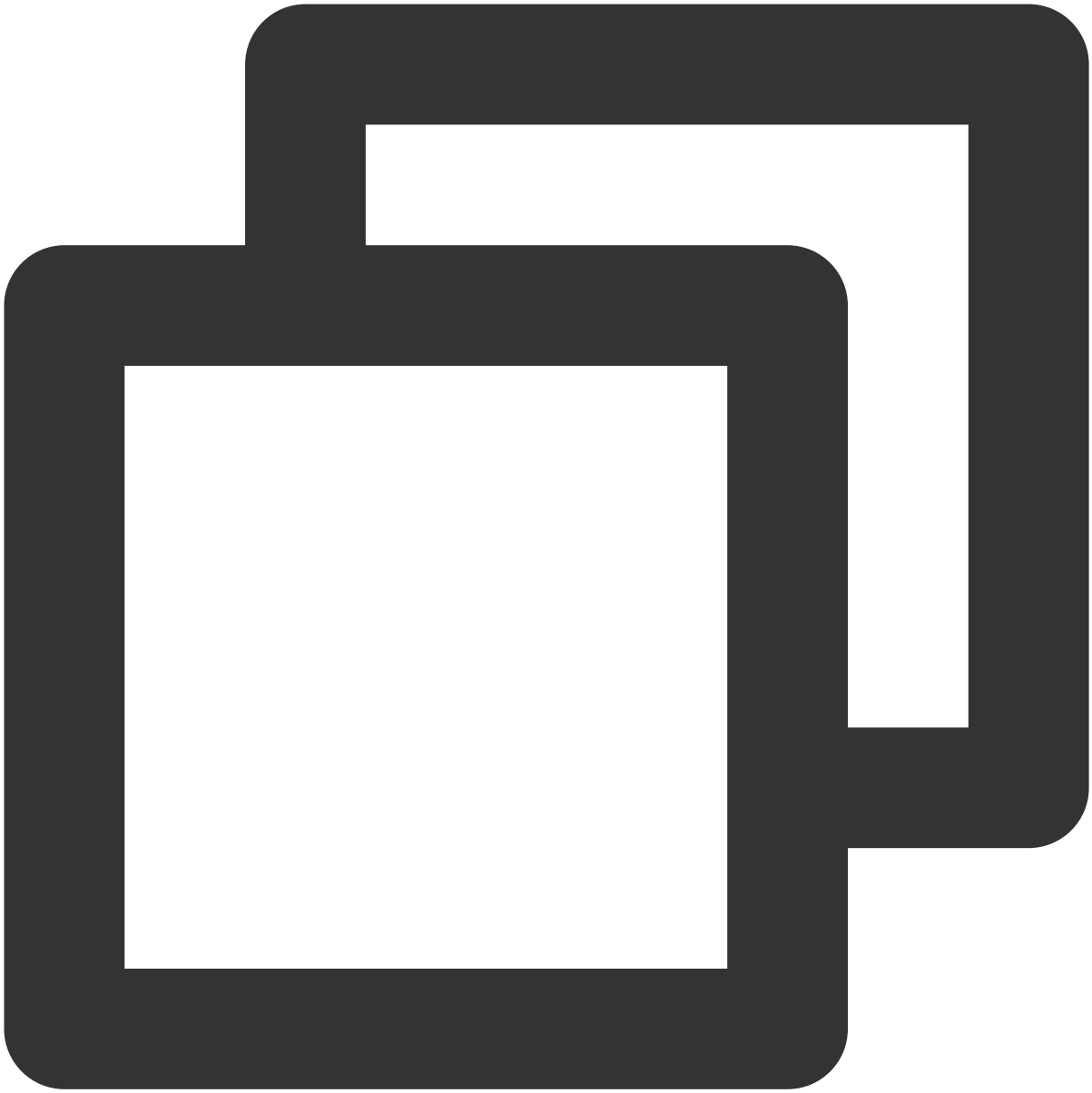
```
docker.image('node:14').inside('-v /home/ubuntu/.npm/:/root/.npm/') {
  sh 'npm install'
}
```

Cache basic docker images

As the basic Docker images, such as the basic `Dockerfile` image and CI agent image, need to be pulled for every build, the process can be accelerated by caching the images.

You can use the `Jenkinsfile` below by modifying the image name:

©2013-2022 Tencent Cloud. All rights reserved.



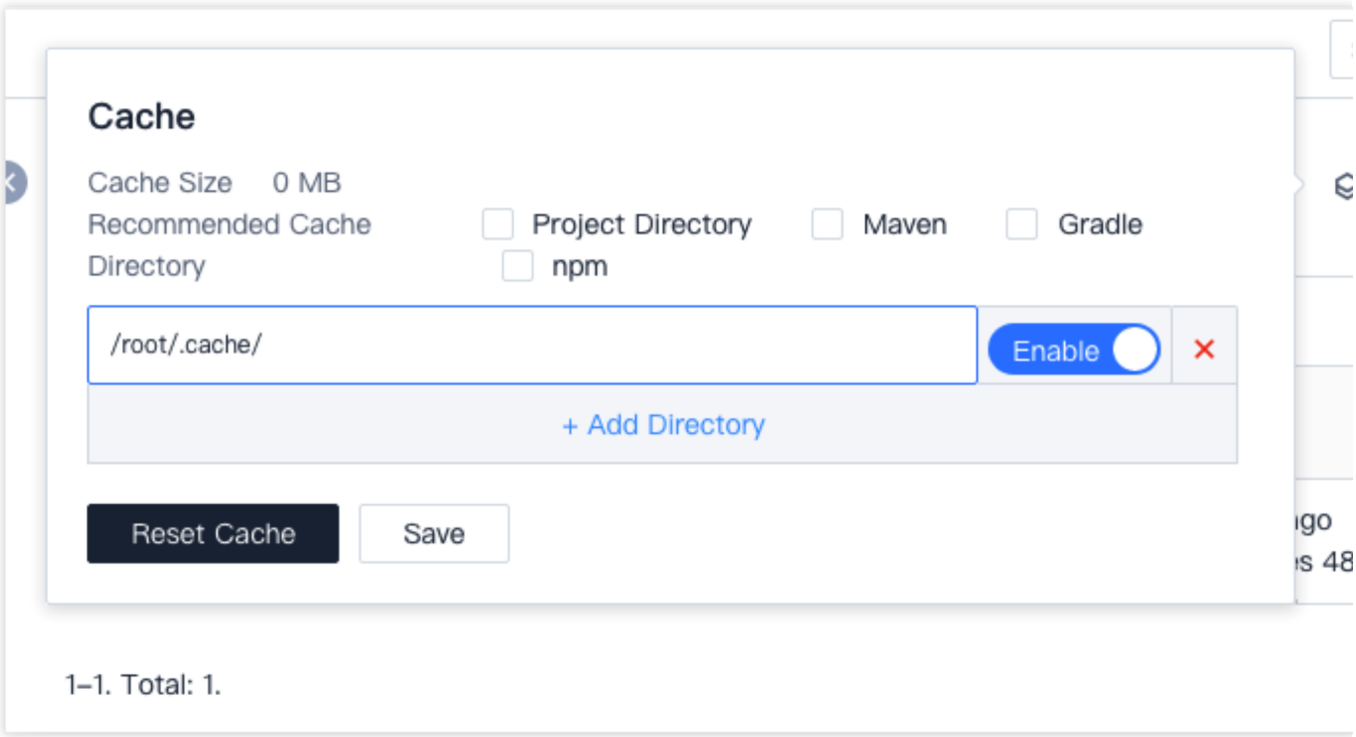
```
pipeline {
  agent any
  environment{
    DOCKER_CACHE_EXISTS = fileExists '/root/.cache/docker/php-8.0-cli.tar'
  }
  stages {
    stage('Load cache') {
      when { expression { DOCKER_CACHE_EXISTS == 'true' } }
      steps {
        sh 'docker load -i /root/.cache/docker/php-8.0-cli.tar'
      }
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    }
    stage('Use images (modify this section)') {
      agent {
        docker {
          image 'php:8.0-cli'
          args '-v /root/.cache/:/root/.cache/'
          reuseNode 'true'
        }
      }
      steps {
        sh "php -v"
      }
    }
    stage('Generate cache (run once only)') {
      when { expression { DOCKER_CACHE_EXISTS == 'false' } }
      steps {
        sh 'mkdir -p /root/.cache/docker/'
        sh 'docker save -o /root/.cache/docker/php-8.0-cli.tar php:8.0-cli'
      }
    }
  }
}
```

Add the path `/root/.cache/` in "Cache Directory". The duration of the second build is significantly shorten due to the cache:

©2013-2022 Tencent Cloud. All rights reserved.



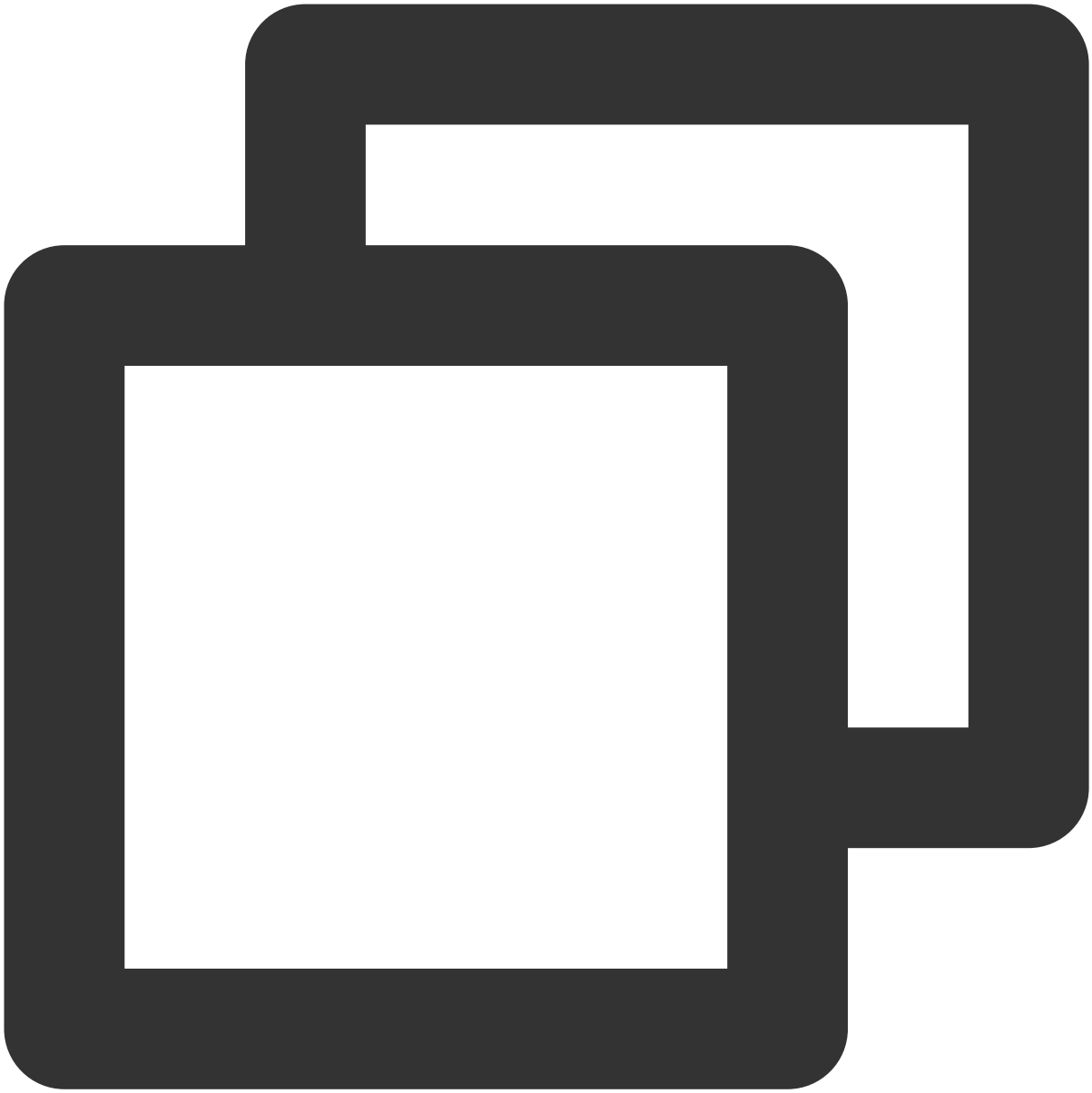
⚠Keep cached images up to date with official updates.

Save Dockerfiles

If you are using a `Dockerfile` as the build environment in Continuous Integration, instead of running the `docker build` command at initialization, save the built Docker images to a repository to pull and reuse them again.

Jenkinsfile

©2013-2022 Tencent Cloud. All rights reserved.



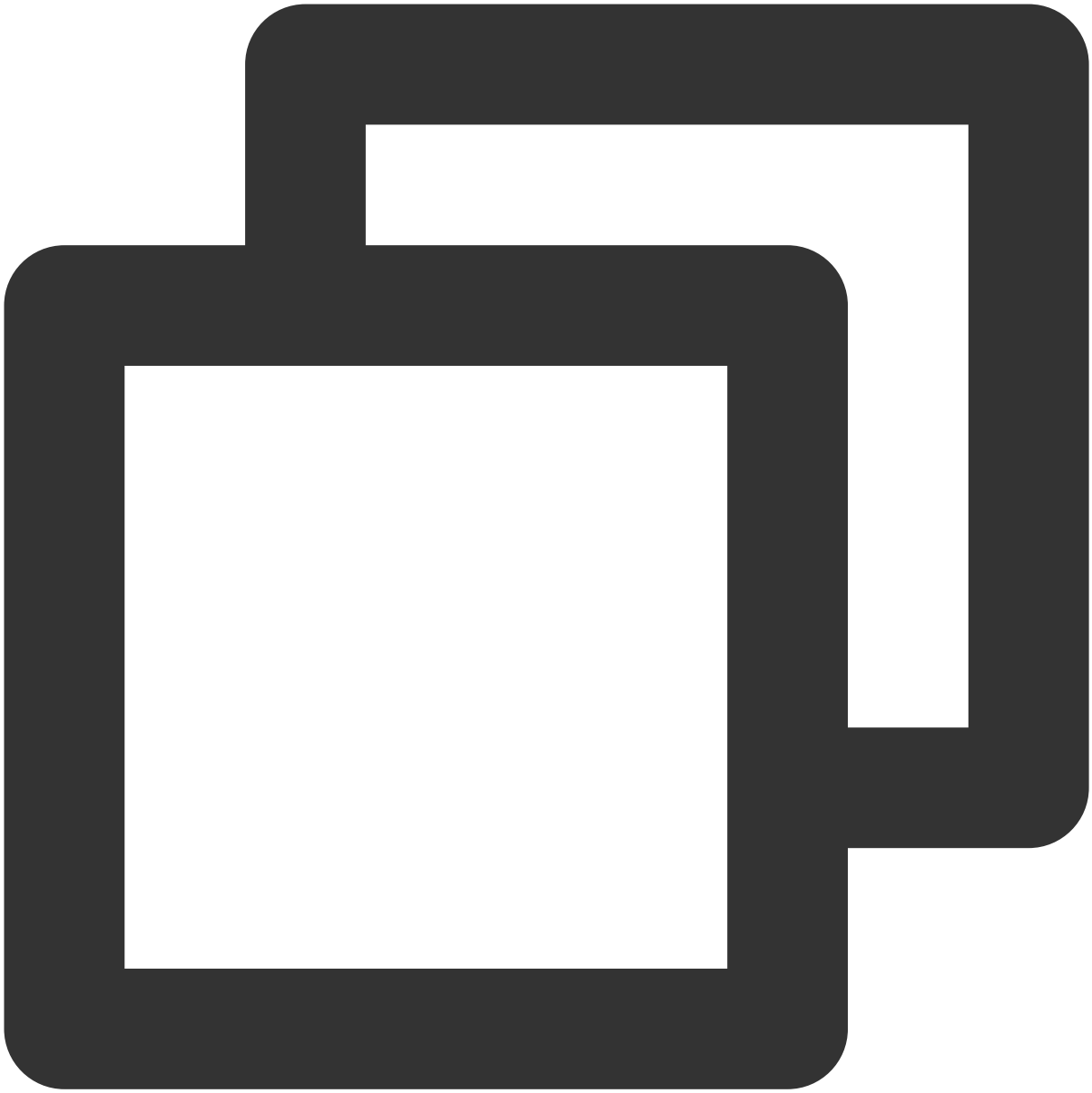
```
// Creates a CODING Docker repository and obtains the username, password, and repos
sh "docker login -u $DOCKER_USER -p $DOCKER_PASSWORD my-team-docker.pkg.coding.net"

// Use MD5 of Dockerfile as tag
md5 = sh(script: "md5sum Dockerfile | awk '{print \\$1}'", returnStdout: true).trim
imageFullName = "my-team-docker.pkg.coding.net/my-project/my-repo/my-app:dev-${md5}

// Check if images exist in remote repository
dockerNotExists = sh(script: "docker manifest inspect $imageFullName > /dev/null",
def testImage = null
if (dockerNotExists) {
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    testImage = docker.build("$imageFullName", "--build-arg APP_ENV=testing ./")
    sh "docker push $imageFullName"
} else {
    testImage = docker.image(imageFullName)
}

// Use images for automated testing
testImage.inside("-e 'APP_ENV=testing'") {
    stage('Test') {
        echo 'testing...'
        sh 'ls'
        echo 'test done.'
    }
}
```

Explanation: By running the following command in the shell, you can determine "if the images exist" based on the returned value.

©2013-2022 Tencent Cloud. All rights reserved.



```
docker manifest inspect ecoding/foo:bar
no such manifest
$ echo $?
1
```

==== 2021/09/17 ====

©2013-2022 Tencent Cloud. All rights reserved.

Build Artifacts
Docker

Last updated：2023-12-29 11:44:51

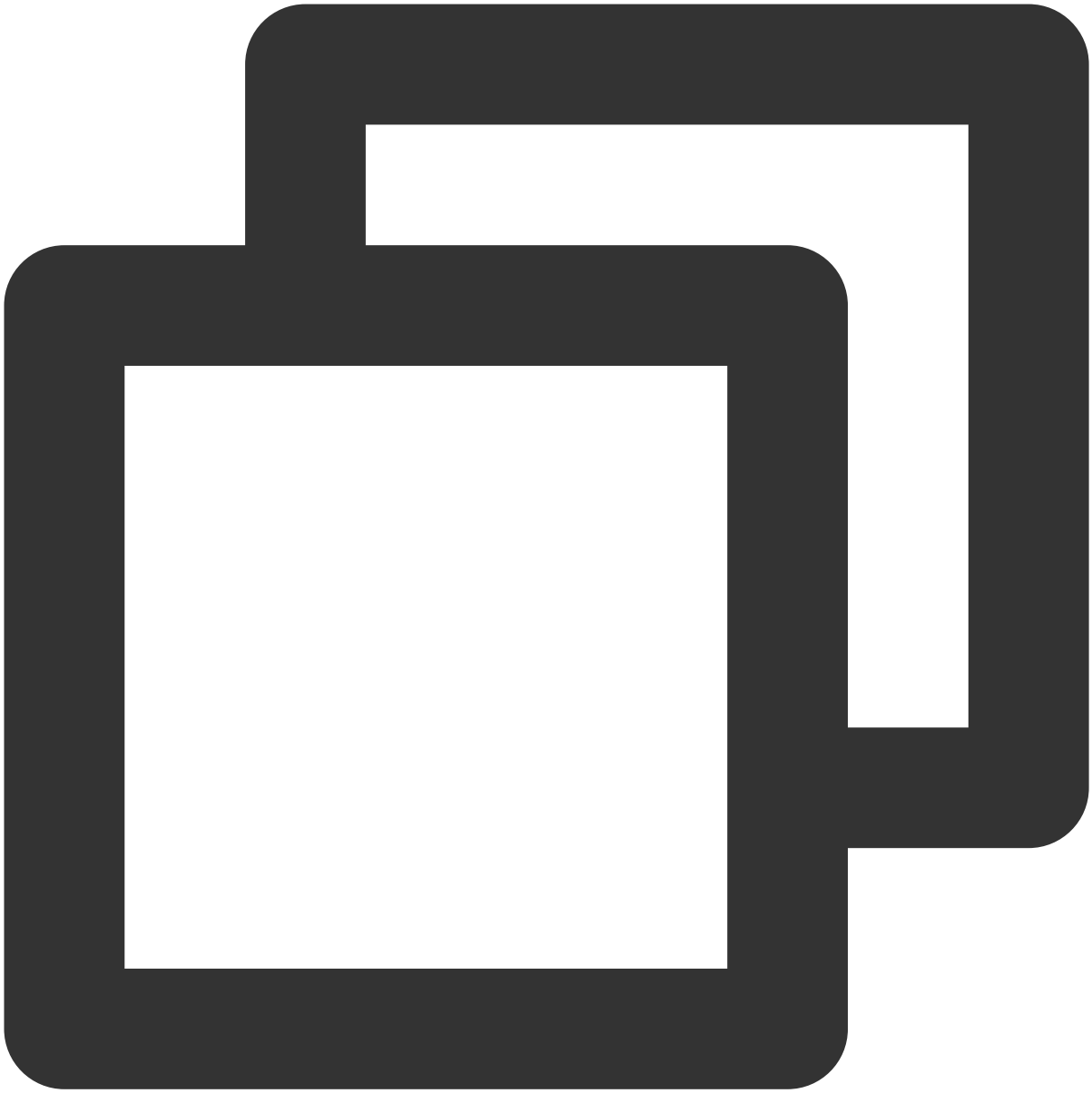
title: Docker - CODING Help Center
pageTitle: Docker
pagePrevTitle: Build Composer Artifacts
pagePrev: ci/artifacts/composer.html
pageNextTitle: Build File Type Artifact
pageNext: ci/artifacts/generic.html
alias:
devops/ci/artifacts/docker.html
ci/artifacts/docker.html

Feature Overview

This document provides an example of a Jenkinsfile for building a Docker image with a continuous integration task. After you build the Docker image, you can use a preset plugin to upload it to the CODING Artifact Repository (CODING-AR). Before using this function, ensure that you have a basic understanding of Docker artifact repositories.

Jenkinsfile

©2013-2022 Tencent Cloud. All rights reserved.



```
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.CREDENTIAL
        ])
      }
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    }
    stage('Build Docker image') {
      steps {
        script {
          ARTIFACT_VERSION = "1.2.0"
          // Note: When creating a project, use hyphens in the link ID instead of u
          // Modify build/my-api to your artifact repository name and image name
          CODING_DOCKER_IMAGE_NAME = "${env.PROJECT_NAME.toLowerCase()}/build/my-ap
          // The environment variable CODING_ARTIFACTS_CREDENTIALS_ID has been buil
          docker.withRegistry("https://${env.CCI_CURRENT_TEAM}-docker.pkg.coding.ne
            docker.build("${CODING_DOCKER_IMAGE_NAME}:${ARTIFACT_VERSION}").push()
          }
        }
      }
    }
  }
}
```

©2013-2022 Tencent Cloud. All rights reserved.

Manage Build Plans
Group Management

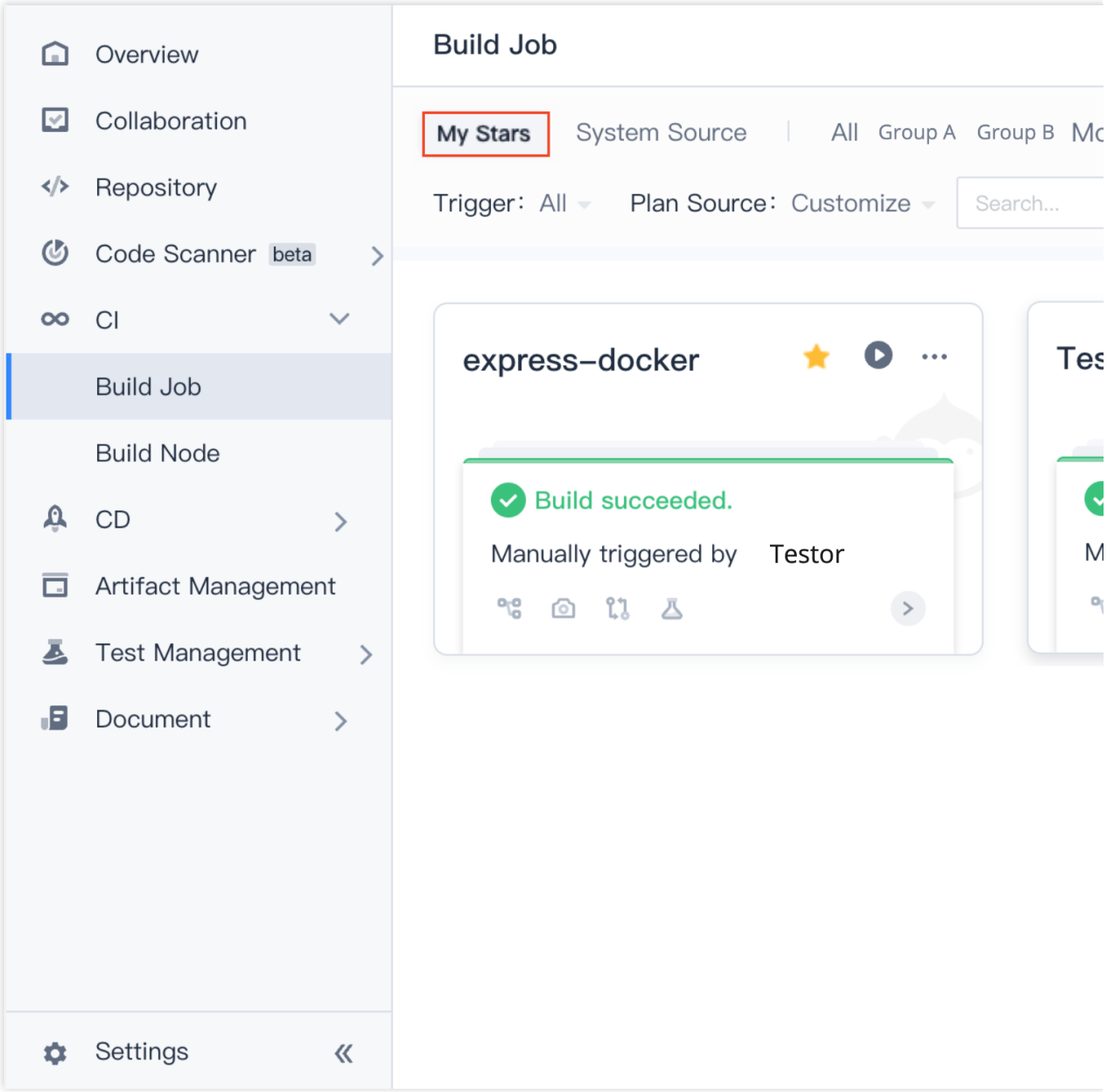
Last updated：2023-12-29 11:44:51

title: Group Management - CODING Help Center
pageTitle: Group Management
pagePrevTitle: Build node pool
pagePrev: ci/node/pool.html
pageNextTitle: Team Build Template
pageNext: ci/manage/team-template.html
alias:
devops/ci/group.html
ci/group.html

Starring and grouping

Build plans can be starred and grouped to help you quickly locate build plans that you are following.
Starring
This is a personal setting that only takes effect for the user who has starred a plan. Click the star on a build plan area, and you can view only starred plans in "My Stars" tab.

©2013-2022 Tencent Cloud. All rights reserved.



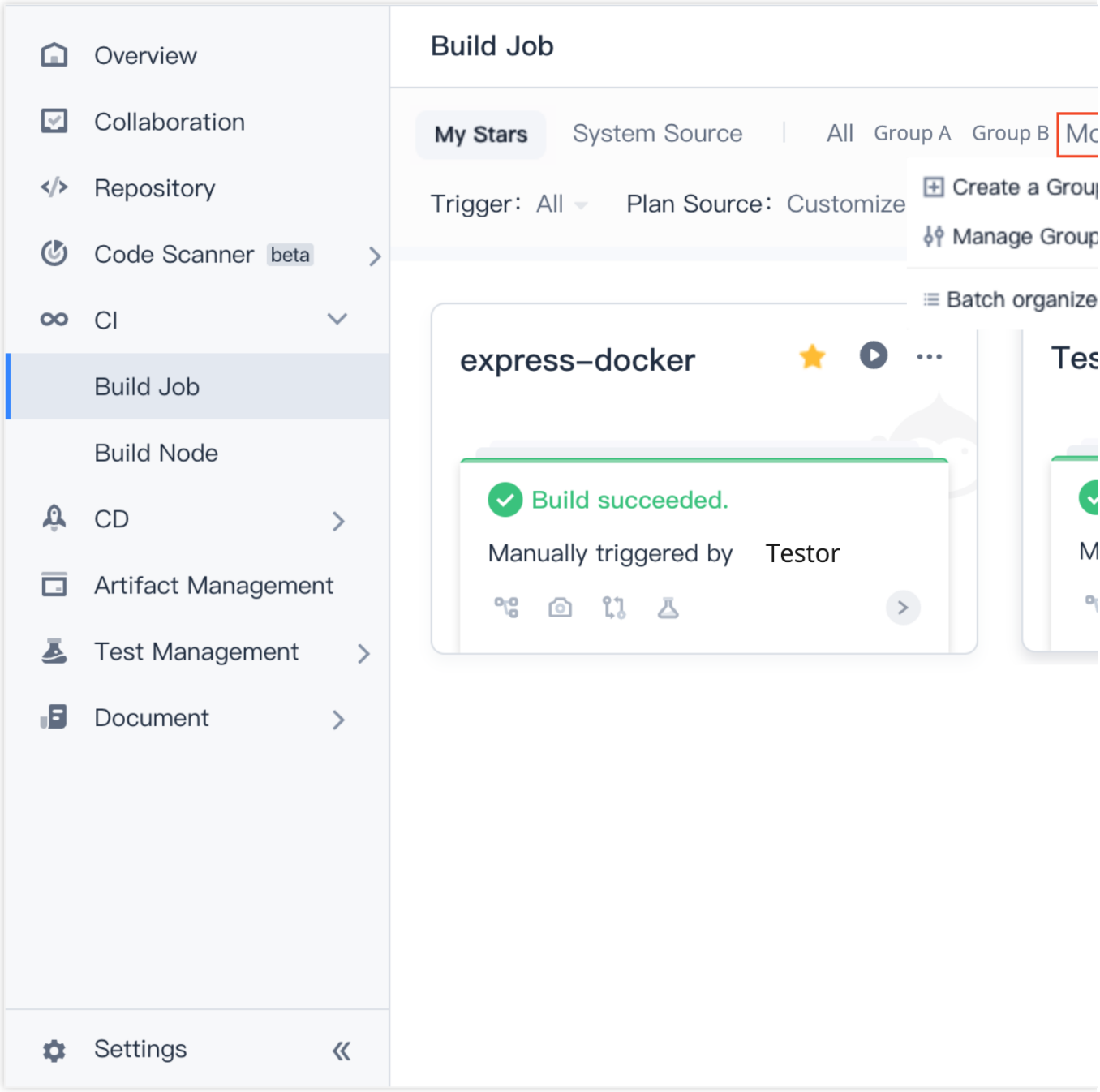
Grouping

This is a global setting that is only accessible to users with permission to "Continuous Integration Management".

Members of a project can view the groups and categories configured for build plans and conveniently sort plans.

Click "More" > "Create Group" and enter a group name to create a group.

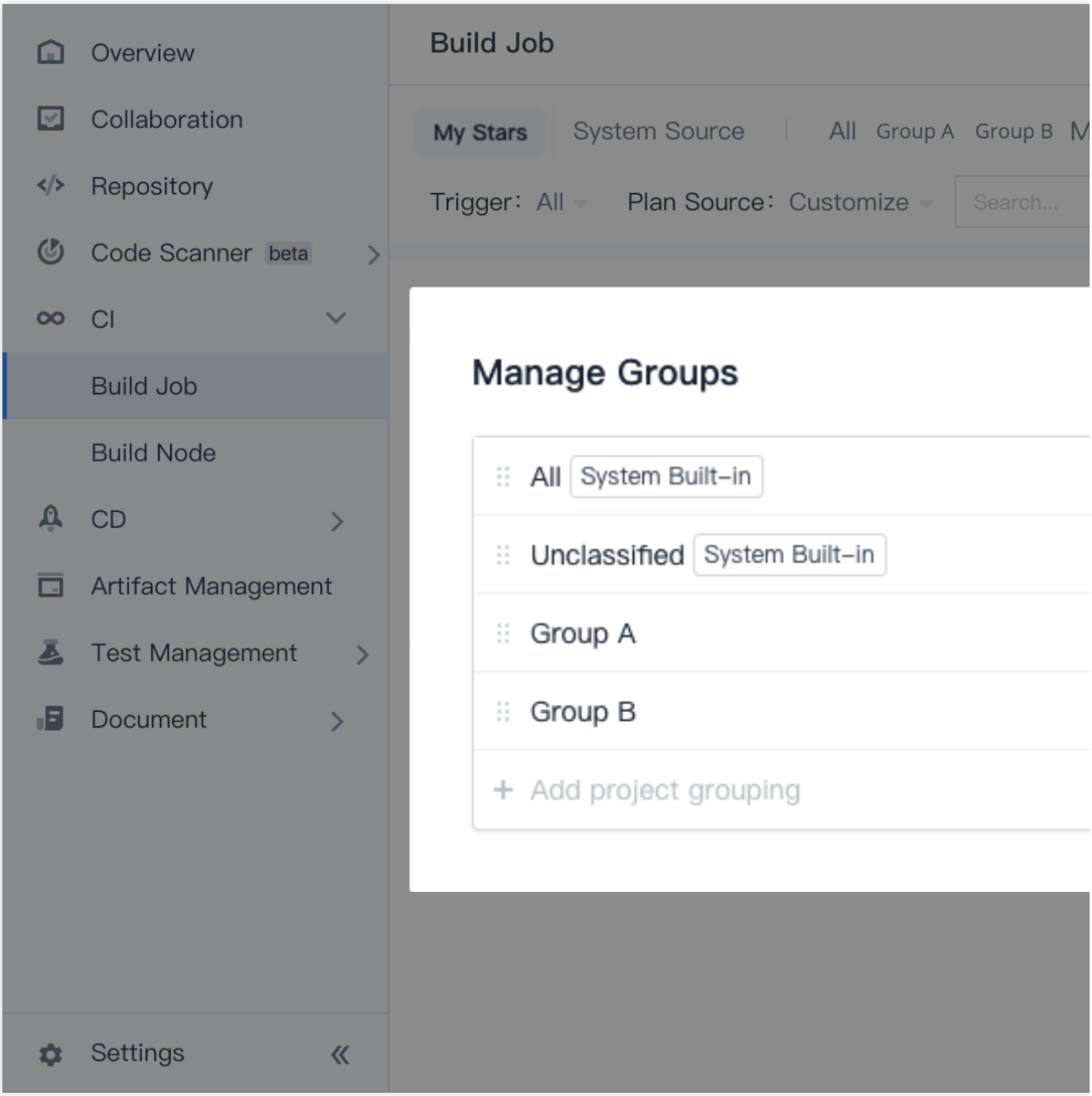
©2013-2022 Tencent Cloud. All rights reserved.



You can modify group names, change the order, and create and delete groups.

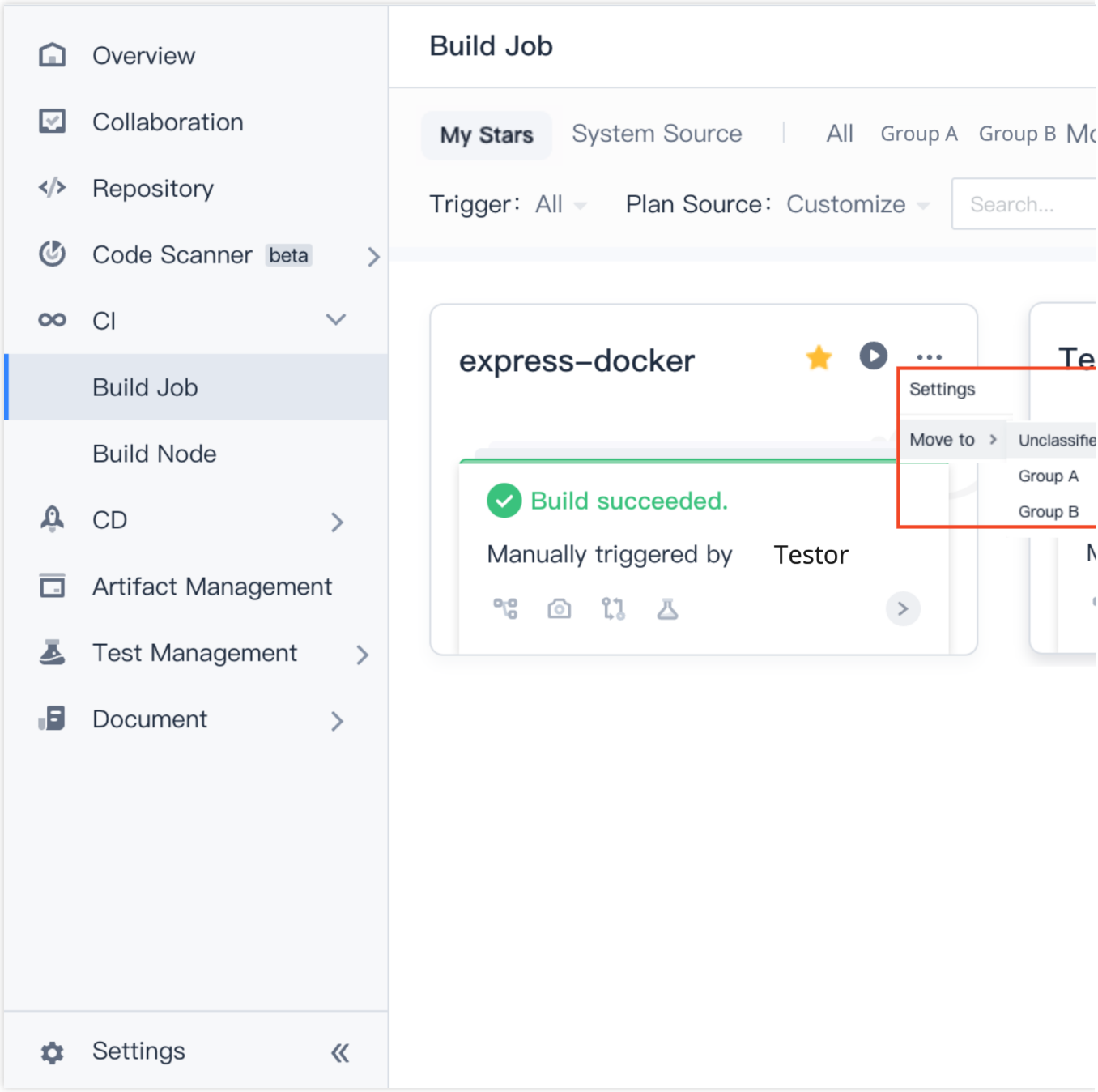
Note: Deleting a group does not delete the build plans in the group. After you delete a group, the build plans in the group will be categorized as "Not Grouped".

©2013-2022 Tencent Cloud. All rights reserved.



Click "Batch Sort Build Plans" to enter the build plan sorting page. You can select multiple and move them to the same group at once. Then the selected build plans can be seen a separate group tab after being added.

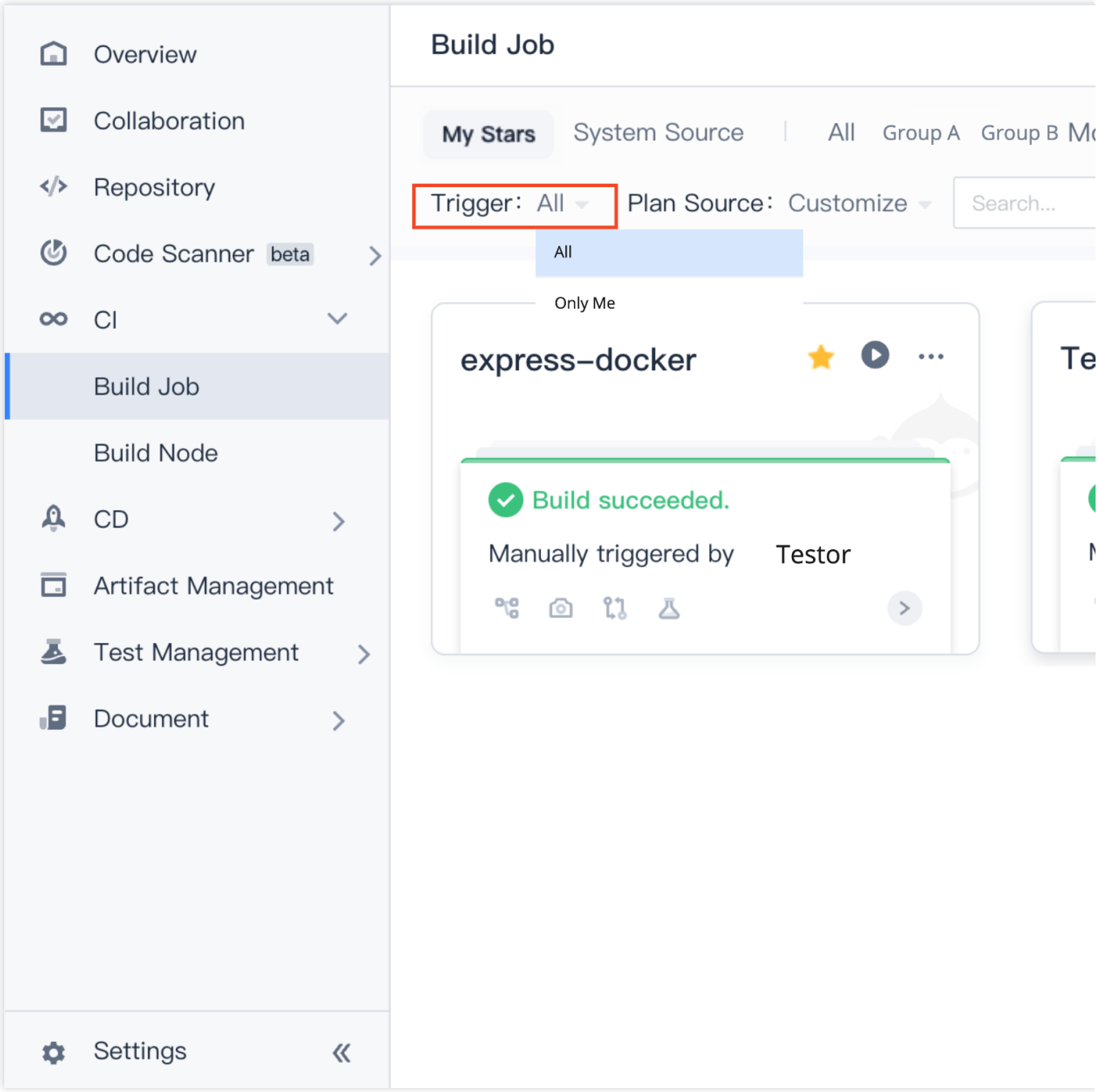
©2013-2022 Tencent Cloud. All rights reserved.



Filtering and sorting

In the search bar on the right of the build plan page, you can filter build plans by name. Select "Filters" > "Only Me". Only the latest build plans triggered by you will be shown. This filter can also be enabled in the build records.

©2013-2022 Tencent Cloud. All rights reserved.



You can also sort the build plans by the trigger time of the latest build records.

==== 2020/09/25 ====

©2013-2022 Tencent Cloud. All rights reserved.

Build Plan Templates

Last updated：2023-12-29 11:44:51

title: Build Plan Templates - CODING Help Center
pageTitle: Build Plan Templates
pagePrevTitle: Group Management
pagePrev: ci/manage/group.html
pageNextTitle: Upload Generic Artifacts
pageNext: ci/plugins/generic.html
alias:
ci/node/overview.html
ci/team-template.html

Function Overview

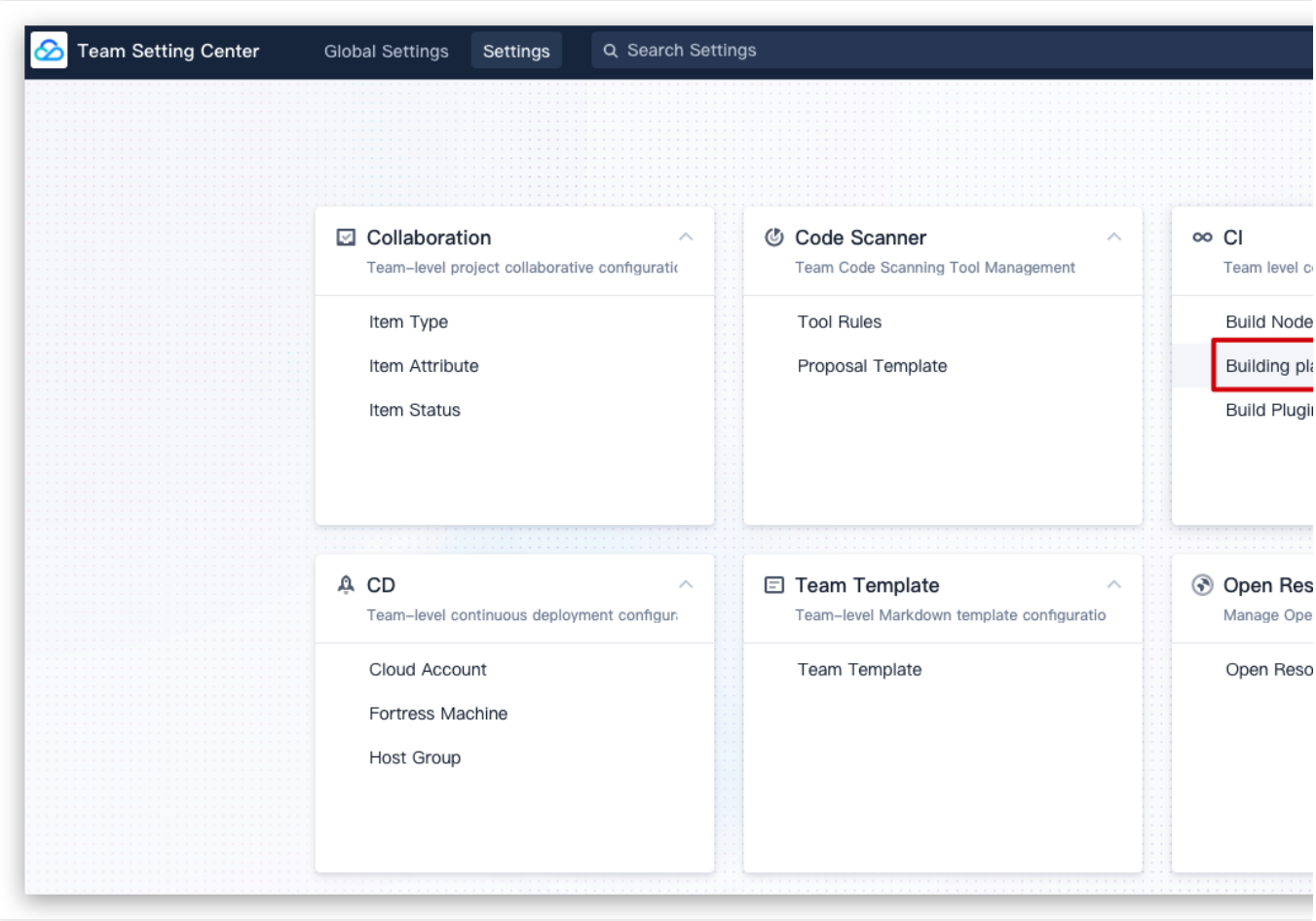
In CODING Continuous Integration (CODING-CI), you can create unified **build plan templates**. Members in a team can reuse configured standard templates across projects to configure build processes more efficiently and centrally manage universal build plans.

New build plan template

Click the gear icon

in the upper-right corner of your team homepage to go to the team settings. Select "Feature Settings" > "Build Plan Templates" to create a new build template.

©2013-2022 Tencent Cloud. All rights reserved.



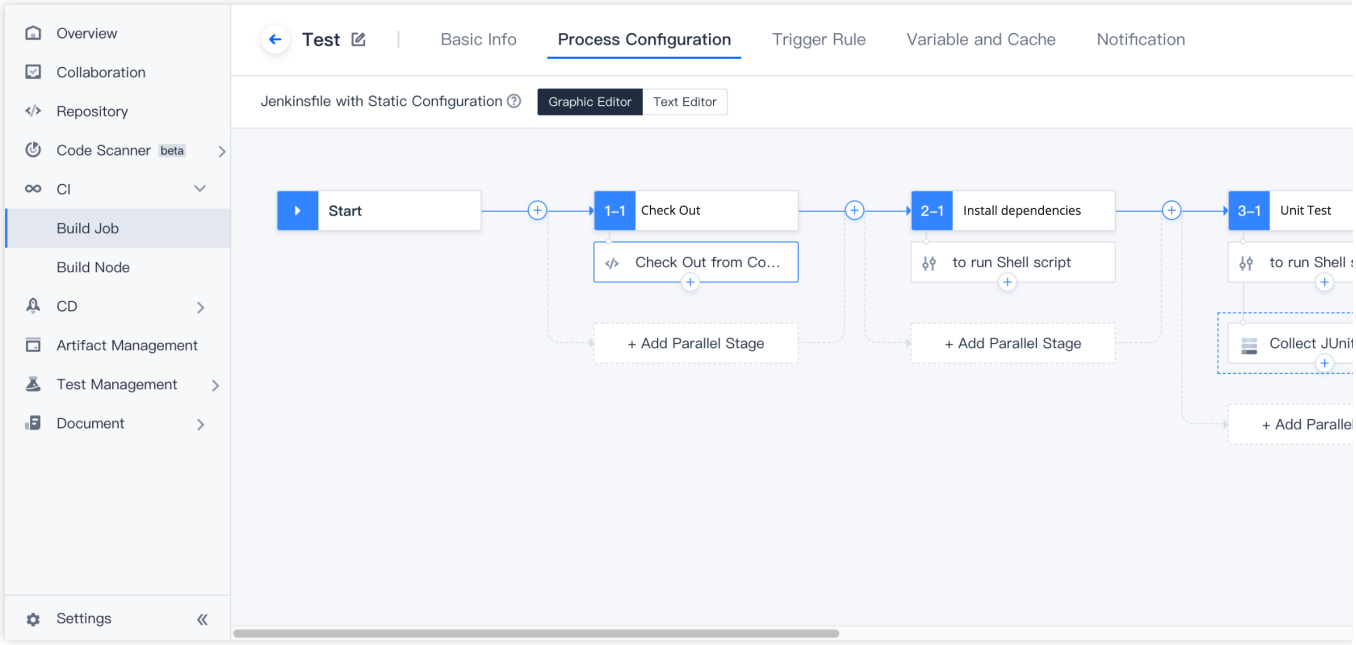
Edit build plan template

In a template, you can edit the process configuration, basic configurations, trigger rules, and variables and caches.

Process configuration

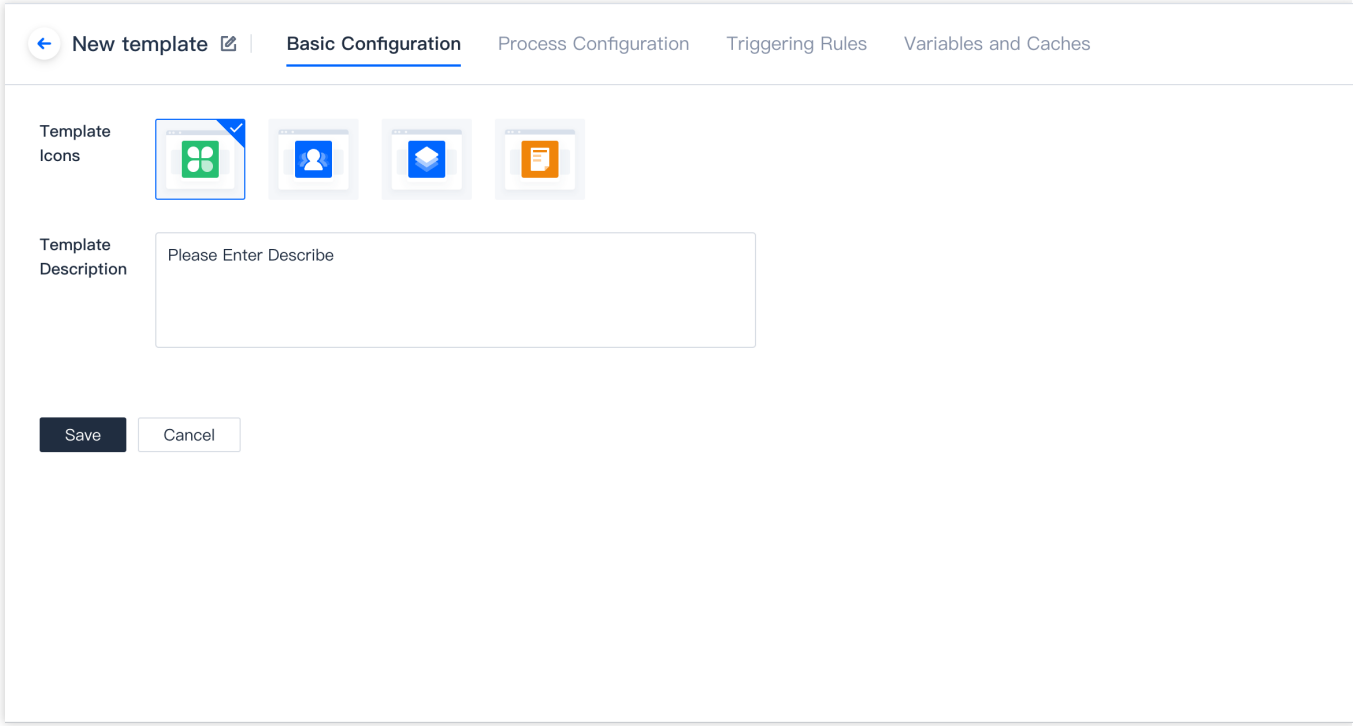
Use the "Graphical Editor" or "Text Editor" to compose the execution process of the build template. The graphical editor allows you to view while writing a build process. You can add and delete steps in the graphical editor and convert the result to text. However, steps composed in the text editor may not be fully converted to graphics.

©2013-2022 Tencent Cloud. All rights reserved.



Basic configurations

In "Basic Configurations", you can change the template name and icon. From the "Actions" dropdown menu on the right, you can select "Delete Template" or Sync Template.



If a template is updated, the creator of the template can click Sync Template to sync the updates to all build plans created from the template. Selecting "Sync Template" will overwrite the configurations in the related build plans. Refer

©2013-2022 Tencent Cloud. All rights reserved.

to the sample scenarios.

Trigger rules

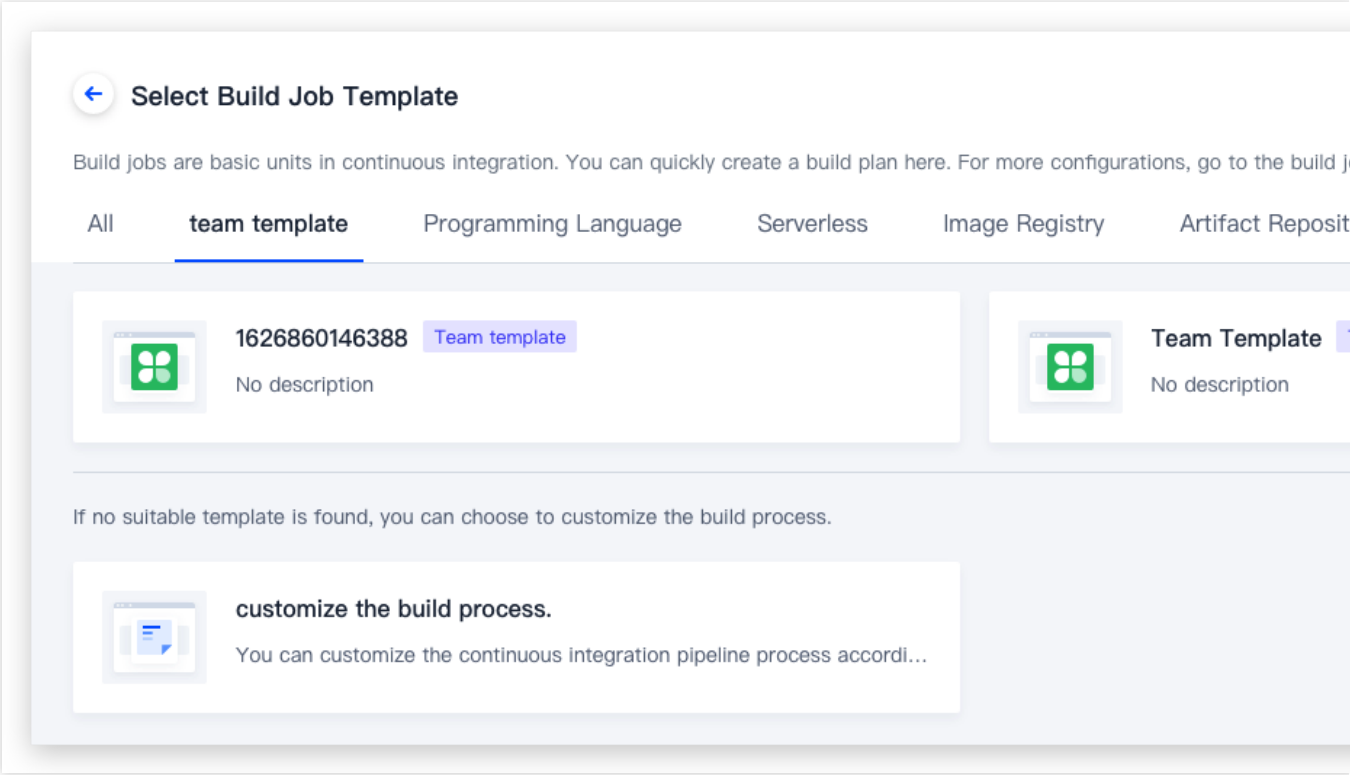
You can configure code source trigger, scheduled trigger, or manual trigger rules. The settings are the same as those of normal build plans. For more information, see Trigger Rules.

Variables and caches

You can add environment variables to a build plan. If you manually enable a build task, the environment variables will be used as the default values for startup parameters. For more information, see Environment Variables.

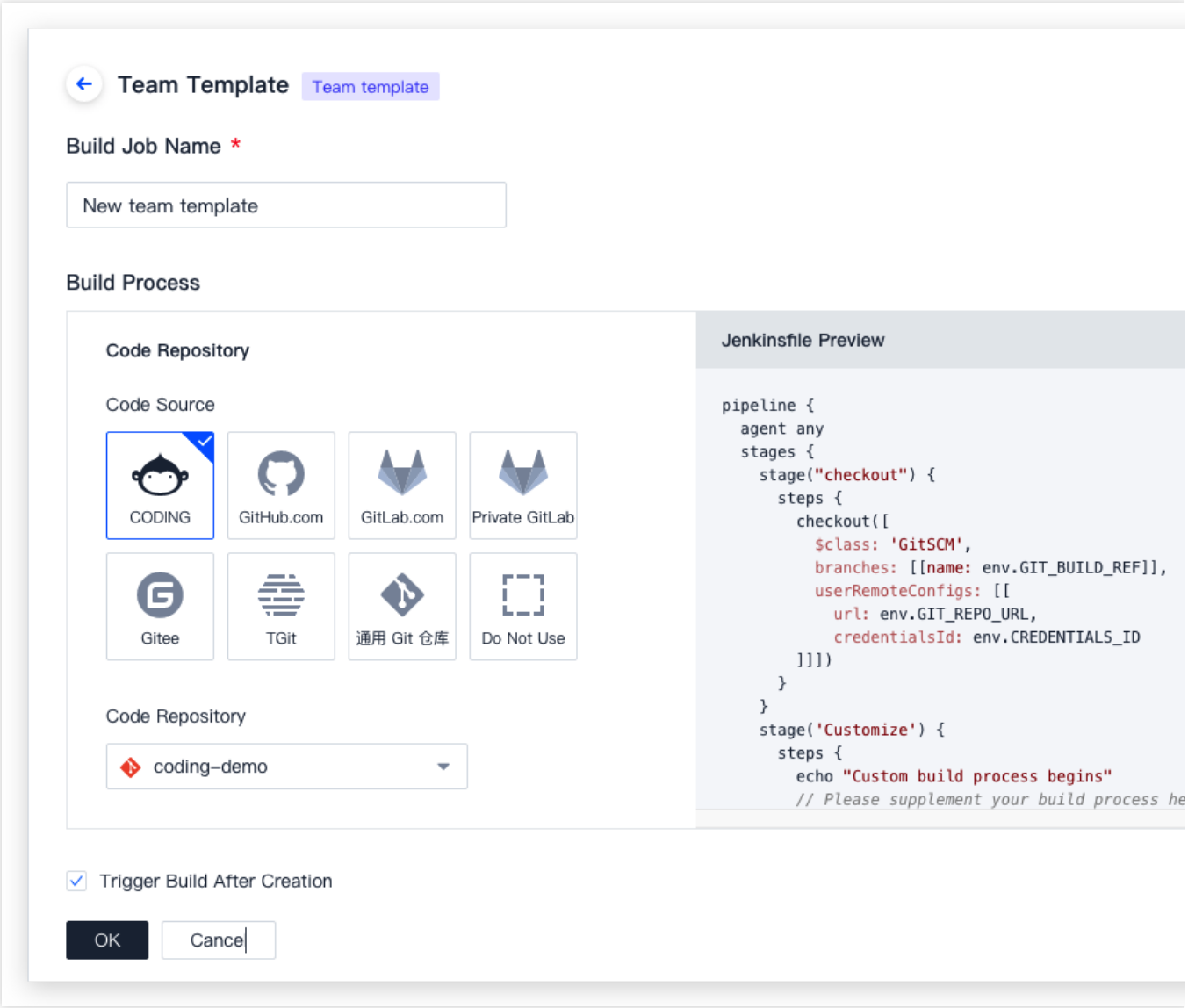
Use build plan template

After a build template is created, members of the team can use the build plan template in any project.



"Template" will be shown in the upper-left corner of such a build plan. You can select the code source as required for a project.

©2013-2022 Tencent Cloud. All rights reserved.



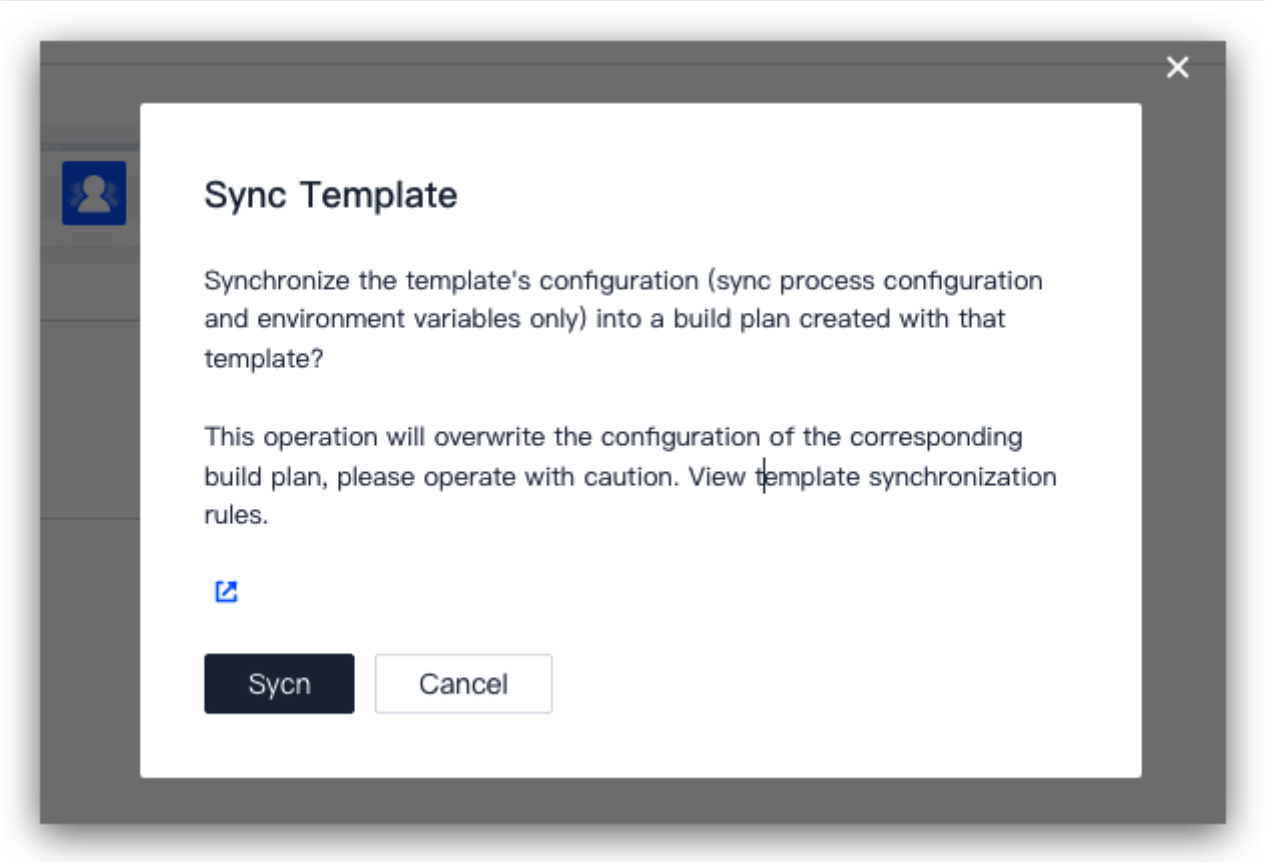
After you create the build plan, the build plan process, trigger configurations, environment variables, and default values are the same as the template, you can modify them according to the project. These modifications will not apply to the template. To modify the template, from your team settings, select "Feature Settings" > "Build Plan Templates".

Sync build plan template

After a build plan template is modified, the creator of a template can sync the updates to all build plans created from the template.

Sample scenario: Team A implements continuous integration build specifications. Most build plans are created from a standard build plan template. After some time, the previous specifications need to be updated and the template creator simply needs to modify the build plan template and sync the updates to all build plans created from the template.

©2013-2022 Tencent Cloud. All rights reserved.



Syncing updates will not overwrite all contents in build plans. See the figure below for an example of "changes" that would be applied.

©2013-2022 Tencent Cloud. All rights reserved.

⚠ Make sure you have known the effect before you perform the sync operation.

==== 2020/11/27 ====

©2013-2022 Tencent Cloud. All rights reserved.

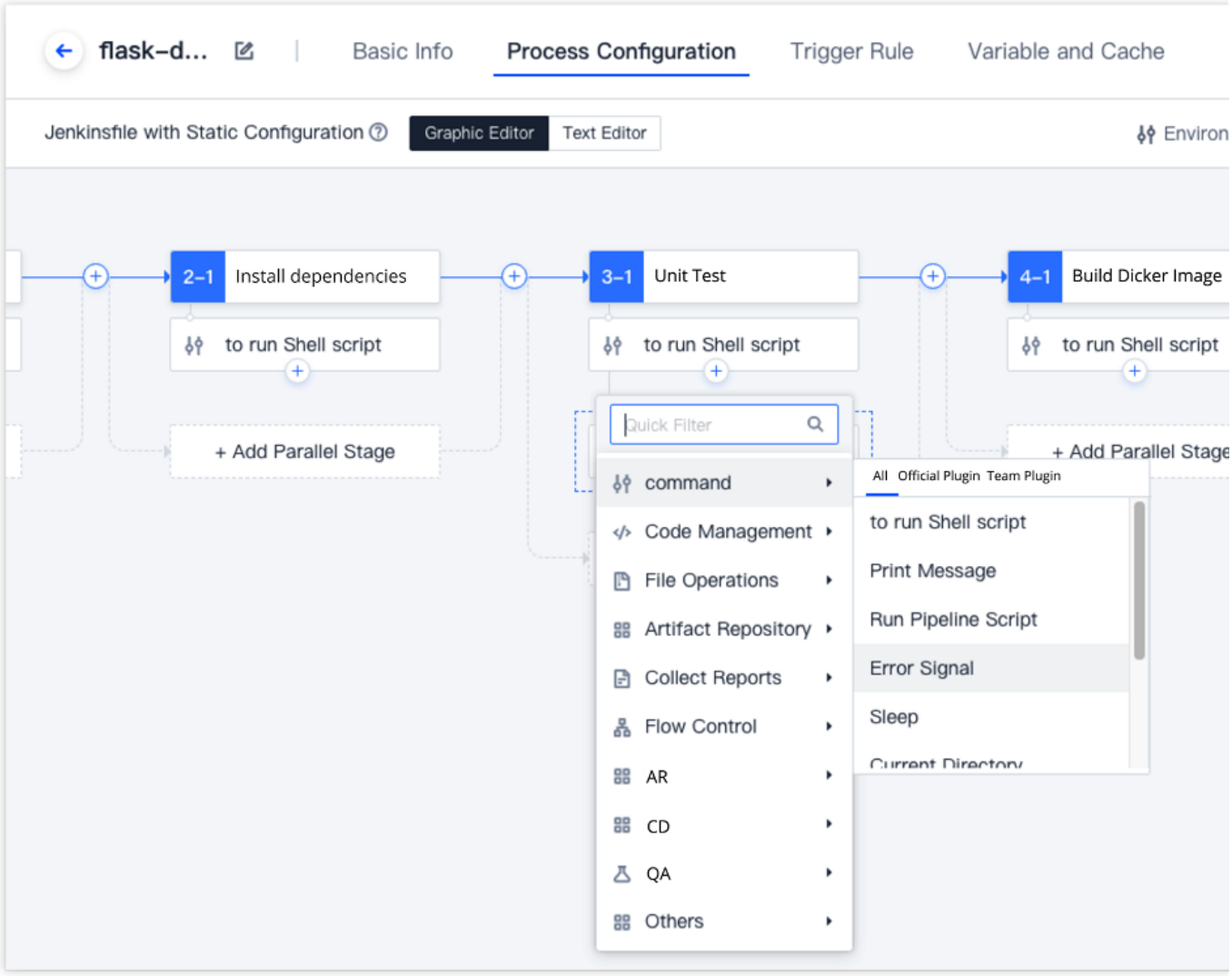
System Plugins
Error

Last updated：2023-12-29 11:44:51

title: Error - CODING Help Center
pageTitle: Error
pagePrevTitle: Update Images in K8s Clusters
pagePrev: ci/plugins/html-report.html
pageNextTitle: Push to CODING Docker Artifact Repository
pageNext: ci/plugins/cci-push-docker.html
alias: devops/ci/plugins/api-doc.html

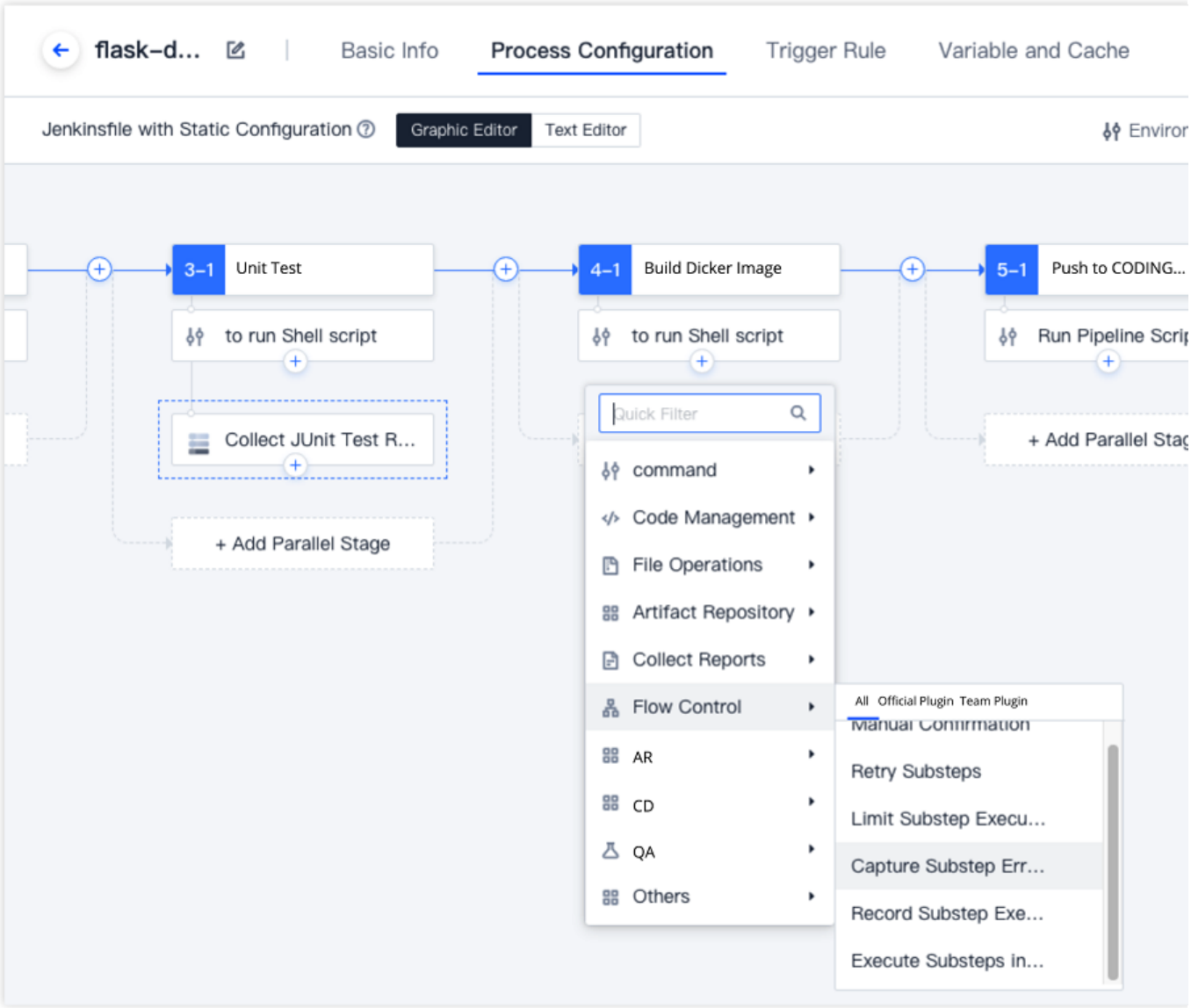
In a sense, the "errorl" in Continuous Integration is a terminator to stop the remaining steps and suspend the build process.

©2013-2022 Tencent Cloud. All rights reserved.



In Continuous Integration, add "Catch incorrect substep". The result will serve as a signal for whether to suspend the continuous integration task. If the result is successful, the remaining steps are run. Even if the result fails, the remaining steps are still run, but the build task is deemed to have failed.

©2013-2022 Tencent Cloud. All rights reserved.



==== 2021/06/30 ====

©2013-2022 Tencent Cloud. All rights reserved.

Upload Generic Artifacts

Last updated：2023-12-29 11:44:51

title: Upload Generic Artifacts - CODING Help Center
pageTitle: Upload Generic Artifacts
pagePrevTitle: Team Template
pagePrev: ci/manage/team-template.html
pageNextTitle: Retrieve Entered Credentials
pageNext: ci/plugins/credentials.html
alias: devops/ci/plugins/generic.html

Feature Overview

In an actual production environment, many tasks are repetitive. CODING Continuous Integration (CODING-CI) features plugins that can help you handle tedious and repetitive tasks efficiently. You can also use custom parameters to address unique needs. More built-in plugins are coming soon. At the moment, you can use the following convenient plugin types in CODING-CI:
Upload generic artifacts
Retrieve uploaded credentials
Automatically add reviewers in merge requests
Manual confirmation

Use plugin to upload generic artifacts

In the process of building a CI task, you can choose to upload an artifact to the CODING Artifact Repository (CODING-AR). Generic artifacts upload plugin allows you to conveniently upload generic artifacts of up to 5 GB in Continuous Integration. Before using this function, ensure that you have a basic understanding of generic repositories. For more information, see Using Generic Repositories.

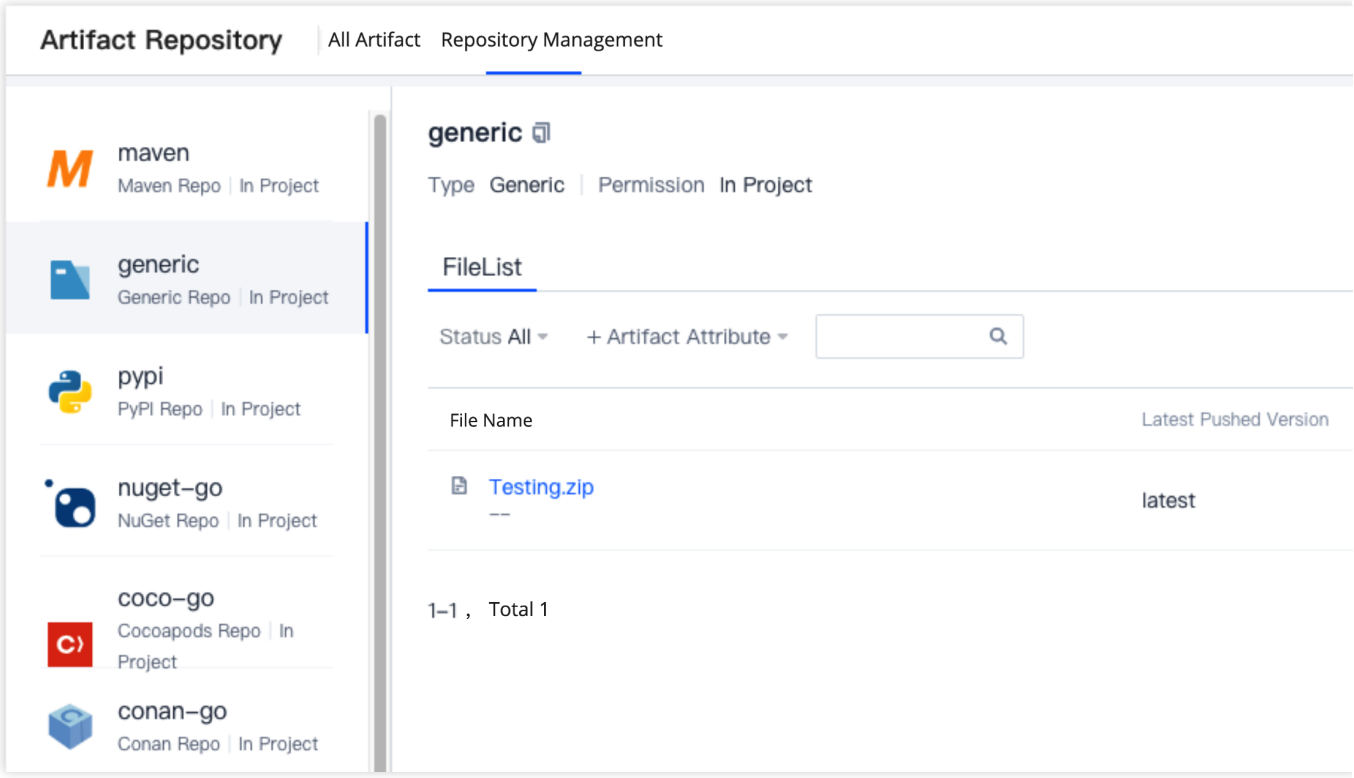
Getting Started

You can use a fixed template or a Jenkinsfile configuration to upload generic artifacts.

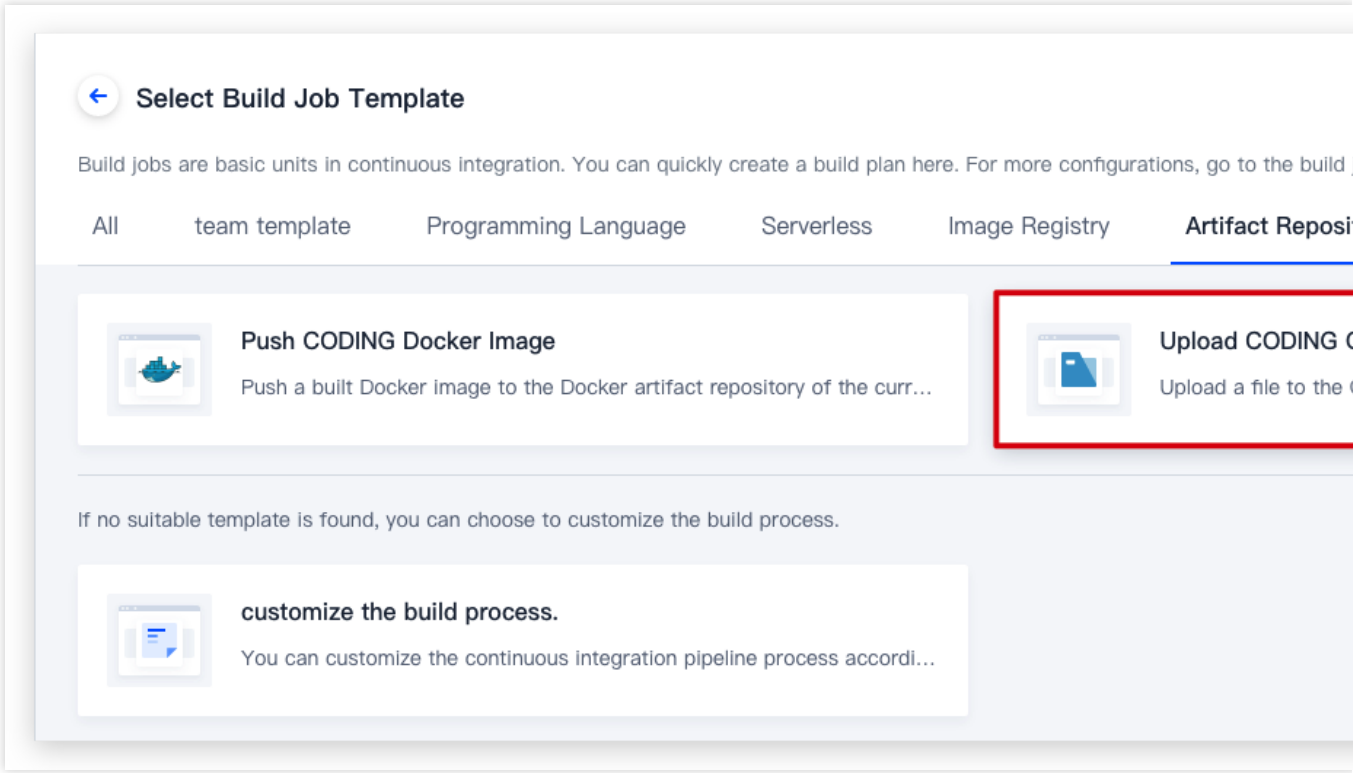
Fixed Template

1. Before using the plugin, make sure that you have created a generic repository. Take a test repository as an example below.

©2013-2022 Tencent Cloud. All rights reserved.

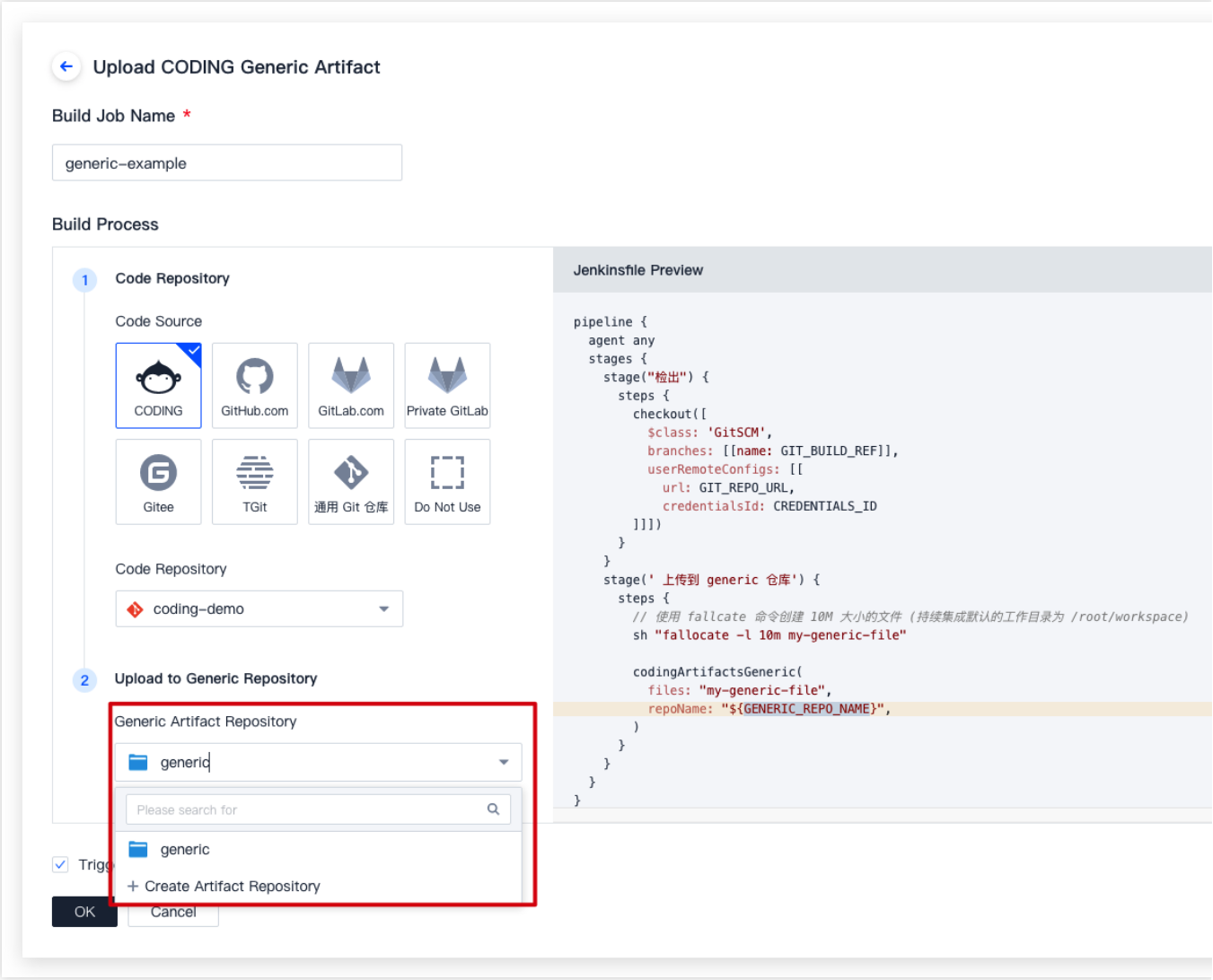


2. Click "New Build Plan" and select "Artifact Repository" > CODING Generic Artifact Upload.



3. Set the default value to "test" repository you have just created. You can also fill in your artifact repository URL in the code.

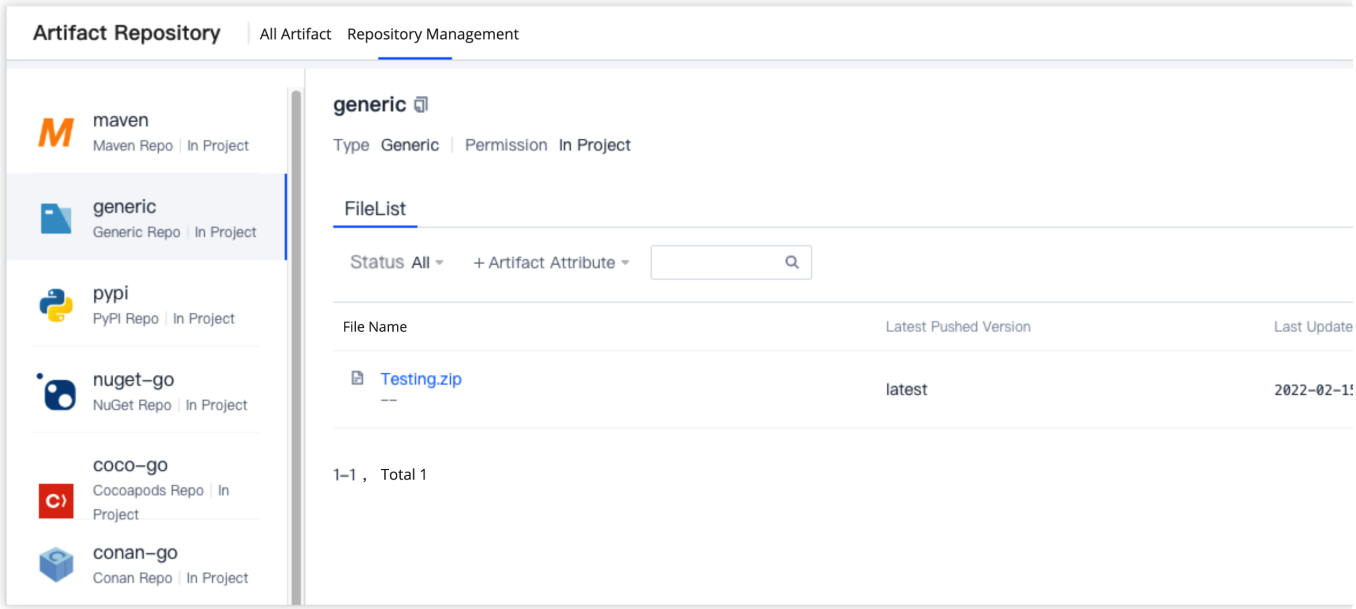
©2013-2022 Tencent Cloud. All rights reserved.



4. Select "Trigger build after creation" and click "OK" to run the build.

5. After the build is complete, the file uploaded beforer will appear in the artifact repository.

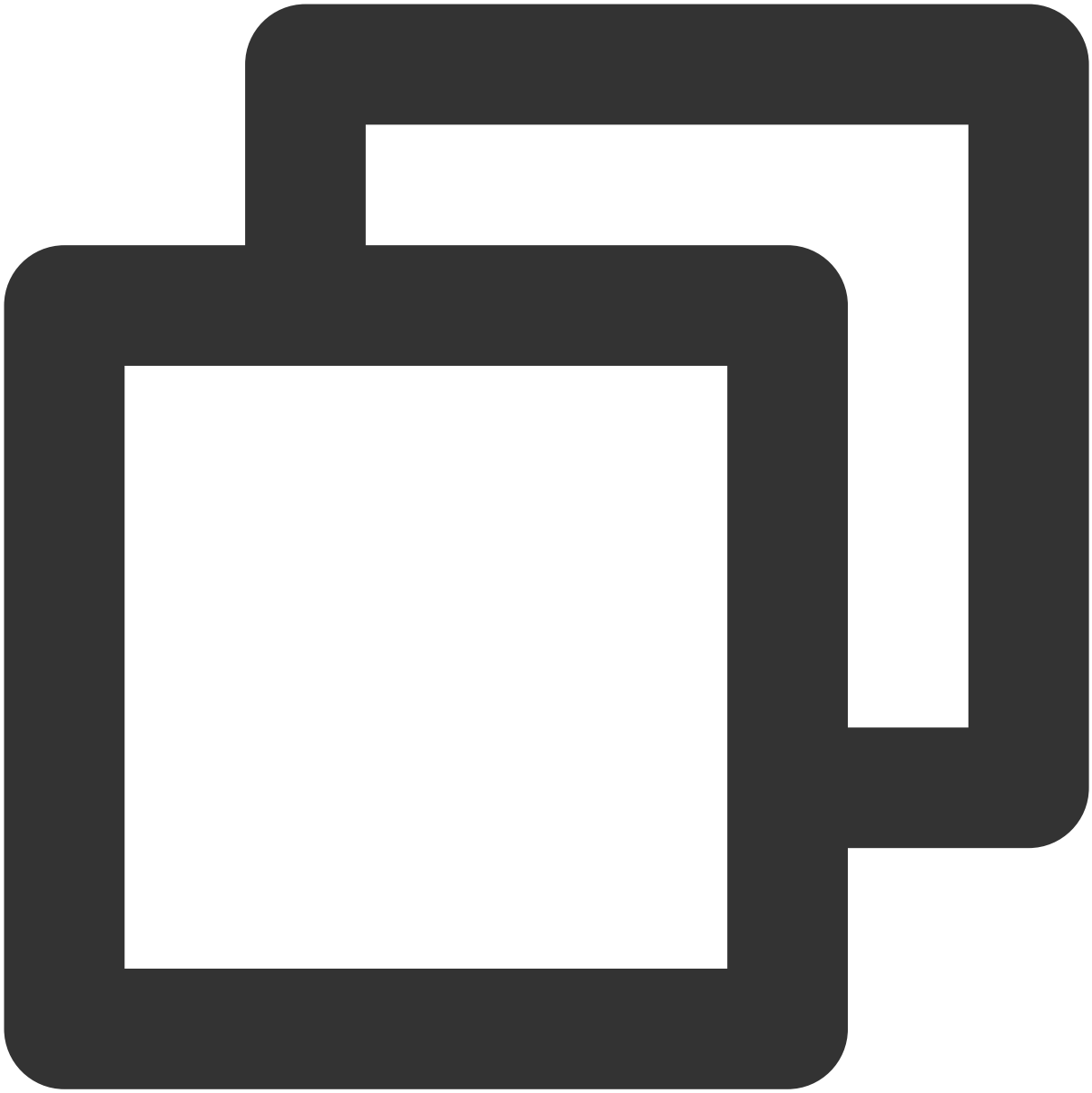
©2013-2022 Tencent Cloud. All rights reserved.



Jenkinsfile Configuration

1. When selecting a code repository, make sure the code repository contains a Jenkinsfile with the configuration below.

©2013-2022 Tencent Cloud. All rights reserved.

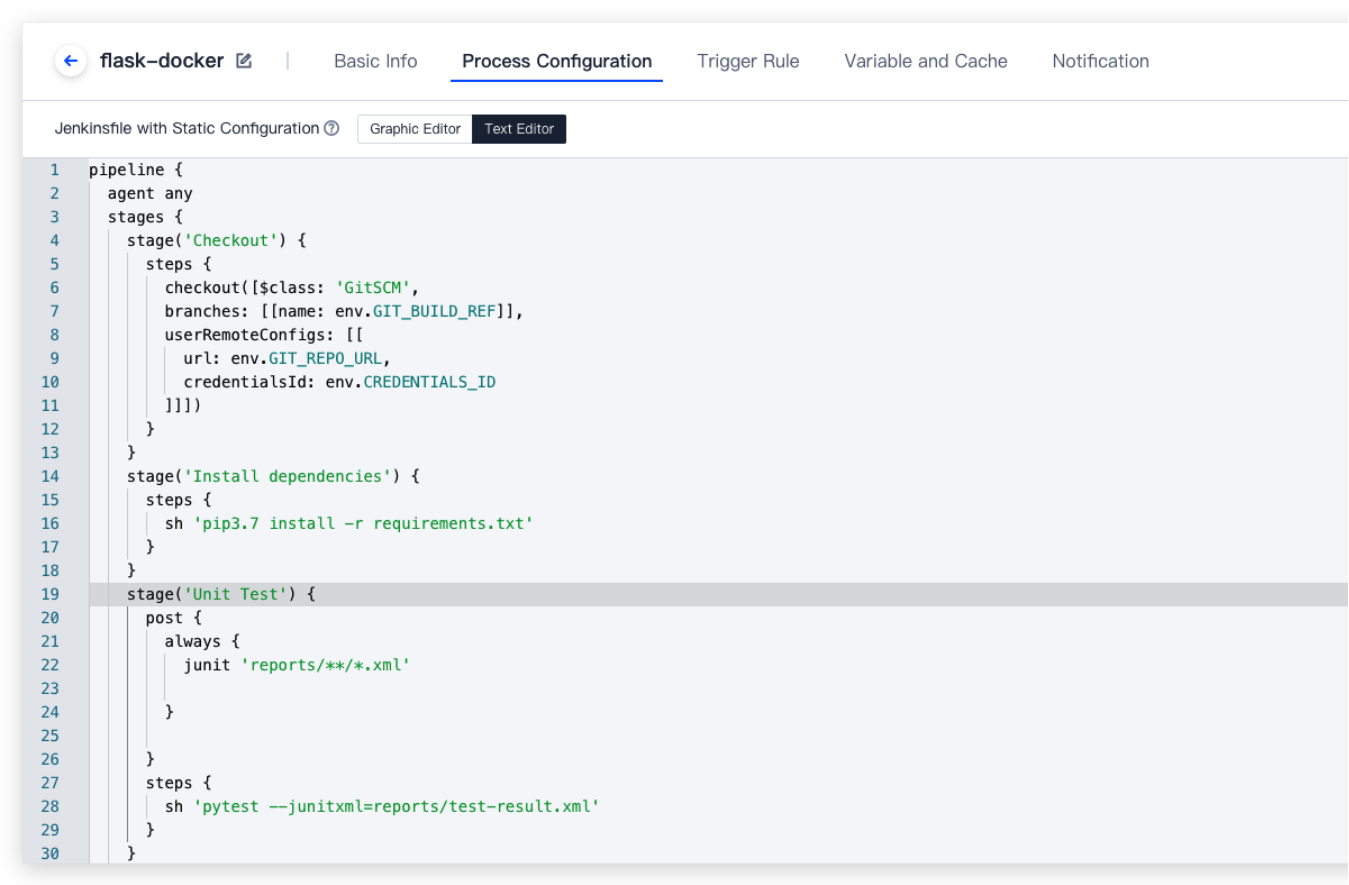


```
pipeline {
    agent any
    stages {
        stage('Upload to generic repository') {
            steps {
                // Use the fallocate command to create a file of 10 MB in size. (Th
                sh 'fallocate -l 10m my-generic-file'

                codingArtifactsGeneric(
                    files: 'my-generic-file',
                    repoName: 'myrepo', // Fill in your repository parameters here,
```

©2013-2022 Tencent Cloud. All rights reserved.

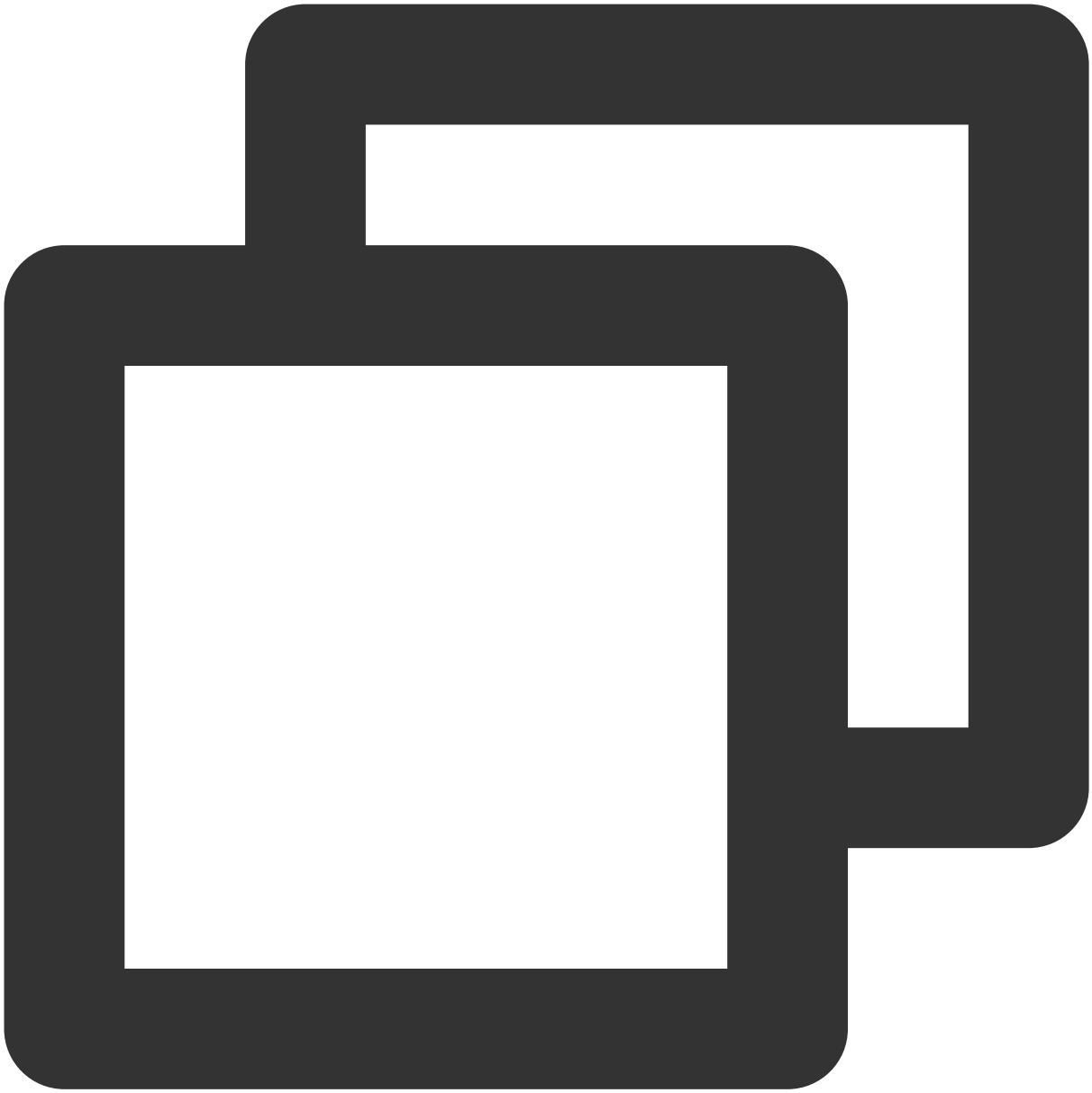
```
                        )
                    }
                }
            }
        }
```

2. In "Process Configuration", you can modify the Jenkinsfile configuration.



More examples of parameters

©2013-2022 Tencent Cloud. All rights reserved.



```
pipeline {
    agent any
    stages {
        stage('Upload to generic repository') {
            steps {
                // Use the fallocate command to create a file of 10 MB in size. (Th
                sh 'fallocate -l 10m my-generic-file'

                codingArtifactsGeneric(
                    files: 'my-generic-file',
                    repoName: 'myrepo',
```

©2013-2022 Tencent Cloud. All rights reserved.

```
                    )
                }
            }
        }
    }
```

Parameter Descriptions

Parameter	Required?	Parameter type	Graphical parameter type	Default value
files	Yes	-	List of files to be uploaded, wildcards are supported `build/libs/**/xx`	-
repoName	Yes (Not required if repoURL is set separately)	string	Repository name	-
version	No	string	string	latest
zip	No	-	string	string
credentialsId	No	string	Credentials (username+password)	env.CODING_ARTIFACTS_CREDI
repoURL	No	string	string	https:// `<` env.CCI_CURRENT_TE generic. `<` env.CCI_CURRENT_D

©2013-2022 Tencent Cloud. All rights reserved.

withBuildProps	No	boolean	boolean	true
workspace	No	string	No	-

==== 2020/08/24 ====

©2013-2022 Tencent Cloud. All rights reserved.