CodeArts Build

User Guide

Issue 01

Date 2025-08-29





Copyright © Huawei Cloud Computing Technologies Co., Ltd. 2025. All rights reserved.

No part of this document may be reproduced or transmitted in any form or by any means without prior written consent of Huawei Cloud Computing Technologies Co., Ltd.

Trademarks and Permissions

HUAWEI and other Huawei trademarks are the property of Huawei Technologies Co., Ltd. All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between Huawei Cloud and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, quarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

i

Contents

1 Working with CodeArts Build	1
2 Purchasing CodeArts Build	4
2.1 Purchasing CodeArts Build	4
3 Configuring Project-Level Role Permissions	5
4 Creating a Build Task	g
4.1 About Build Tasks	
4.2 Creating and Managing Groups	10
4.3 Defining a Build Task on GUI	12
4.4 Defining a Build Task with Code	22
4.4.1 Creating a Build Task with Code	22
4.4.2 Crafting Your build.xml for a Single Task	24
4.4.3 Crafting Your build.xml for Multiple Tasks	25
5 Configuring a Build Task	28
5.1 Performing Basic Configurations	28
5.1.1 Configuring the Build Environment	28
5.1.2 Configuring the Code Download	32
5.2 Selecting Build Actions	36
5.3 Configuring Build Actions	37
5.3.1 Building with Maven	37
5.3.2 Building with Android	50
5.3.3 Building with npm	58
5.3.4 Building with Gradle	60
5.3.5 Building with Yarn	61
5.3.6 Building with Gulp	63
5.3.7 Building with Grunt	64
5.3.8 Building with Mono	65
5.3.9 Building in PHP	67
5.3.10 Building with Setuptools	68
5.3.11 Building with PyInstaller	70
5.3.12 Running Shell Commands	72
5.3.13 Building with GNU Arm	73
5.3.14 Building with CMake	75

5.3.15 Building with Ant	77
5.3.16 Building with Kotlin	78
5.3.17 Building with Go	79
5.3.18 Building Android App with Ionic	80
5.3.19 Building an Android Quick App	82
5.3.20 Building with sbt	83
5.3.21 Building with Grails	85
5.3.22 Building with Bazel	
5.3.23 Building with Flutter	
5.3.24 Building with HarmonyOS	
5.3.25 Running Docker Commands to Operate Images	
5.3.26 Generating a Unit Test Report	
5.3.27 Building an Image and Pushing It to SWR	
5.3.28 Using a Custom Build Environment	
5.3.29 Downloading a Software Package from Release Repos	
5.3.30 Downloading File from File Manager	
5.3.31 Uploading a Software Package to Release Repos	
5.3.32 Uploading Files to OBS	
5.4 Configuring Parameters	
5.5 Configuring Schedules	
5.6 Configuring Roles and Permissions	
5.7 Configuring Notifications	
6 Running a Build Task	118
7 Viewing a Build Task	119
8 Managing Build Tasks	122
9 Querying Audit Logs	125
10 References	127
10.1 Syntax Guide to YAML File Configuration	
10.2 Guide to Cache Directory	
11 Old User Guide	143
11.1 Signing Android APK	
11.2 Managing Files	
11.3 Custom Build Environments	
11.4 Custom Templates	
11.5 Editing, Deleting, Cloning, Favoriting, and Stopping a Build Task	150

Working with CodeArts Build

Build refers to the process of compiling source code into one or more target files, and packaging these target files along with configuration and resource files.

CodeArts Build provides an easy-to-use, cloud-based build platform that supports multiple programming languages, helping you achieve continuous delivery with higher efficiency. With just a few clicks, you can easily create, configure, and run build tasks to automate code retrieval, build, and packaging. CodeArts Build also monitors build status in real time.

CodeArts Build is a service provided within the **CodeArts** solution. For details about its role in the solution, see **CodeArts Architecture**.

For more information about CodeArts Build, see Service Overview.

Steps

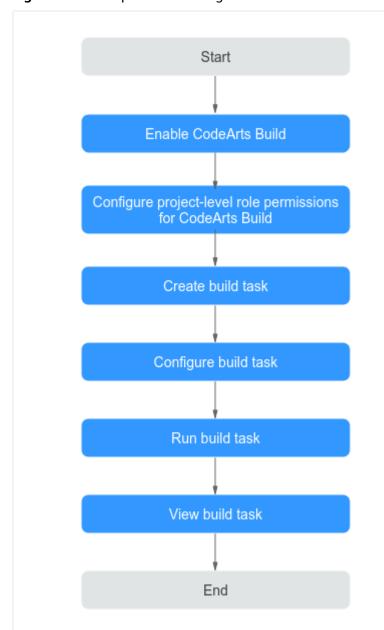


Figure 1-1 Basic process of using CodeArts Build

Table 1-1 Steps in using CodeArts Build

Step	Description
Enable CodeArts Build	In this step, you enable CodeArts Build .
Configure project- level role permissions for CodeArts Build	Before using CodeArts Build, you need to configure basic project-level permissions. Additionally, you can access the CodeArts Build homepage to gain an overall understanding of its features.

Step	Description
Create build task	In this step, you create a build task through either the GUI or YAML. You can configure parameters, schedules, roles and permissions, and event notifications for the task.
Configure build actions	You can choose from over 30 build tools to configure your build by following the GUI guide or referring to the sample code for the YAML file.
Run build task	In this step, you run a build task , which can be triggered by pipelines or schedulers.
View build task	In this step, you check the information and execution results of the build task.

2 Purchasing CodeArts Build

2.1 Purchasing CodeArts Build

Prerequisites

You have registered with Huawei Cloud and completed real-name authentication. If you do not have a HUAWEI ID yet, follow these steps to create one:

- Visit Huawei Cloud official website.
- Click **Sign Up** and create your account as instructed.
 Once your account is created, the system automatically redirects you to your personal information page.
- 3. Complete individual or enterprise real-name authentication. For details, see **Real-Name Authentication**.

Procedure

For details, see **Purchasing CodeArts**.

3 Configuring Project-Level Role Permissions

Assign a specific role to the new member. Each role comes with its own default permissions. For details, see **Table 3-1**.

Table 3-1 Default roles and permissions matrix for CodeArts Build

Role	Creat e	Edit	Delet e	View	Execu te	Clone	Disab le	Assig n Permi ssions	Gro up
Projec t creato r	③	>	③	>	>	>	③	>	S
Projec t mana ger	②	②	②	⊘	⊘	②	②	⊘	8
Produ ct mana ger	3	8	0	②	0	0	0	0	8
Test mana ger	3	8	3	S	8	8	8	8	3
Opera tion mana ger	8	8	8	8	8	8	8	8	8

Role	Creat e	Edit	Delet e	View	Execu te	Clone	Disab le	Assig n Permi ssions	Gro up
Syste m engin eer	③	>	②	②	⊘	⊘	②	€3	S
Com mitter	>	>	②	②	>	>	>	8	3
Devel oper	>		②	②		>	>	8	3
Tester	8	8	8	8	8	8	8	8	3
Partici pant	8	8	8	8	8	8	8	8	3
Viewe r	8	3	8	3	8	8	8	8	3
Projec t admi n	③	>	②	②	>	>	③	>	S

□ NOTE

indicates that the roles have the permission, and indicates that they do not.

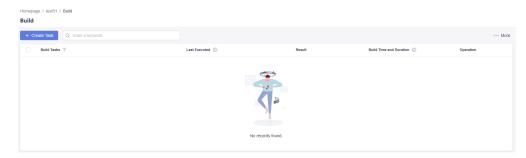
Prerequisites

- You have enabled CodeArts Build.
- You have added members by referring to CodeArts User Guide >
 "Preparations" > "Adding Project Members", and assigned roles to the new members by referring to "Managing Permissions".

Accessing the CodeArts Build Homepage

- **Step 1** Log in to the Huawei Cloud console with your Huawei Cloud account.
- Step 2 Click in the upper left corner and choose Developer Services > CodeArts Build from the service list.
- **Step 3** You can access CodeArts Build from either the homepage or the project list.
 - From the homepage

 Click Access Service to go to the CodeArts Build homepage, where you can find your build task list.



• From the project list

- a. Click Access Service to see the CodeArts Build homepage.
- b. Click **Homepage** on the menu bar.
- c. Click the name of the target project.
- d. Choose **CICD** > **Build**. The build task list page of the specified project is displayed.

----End

Configuring Role Permissions

- 1. Access the CodeArts Build homepage from the project list.
- 2. In the navigation pane, choose **Settings** > **Permissions**.
- 3. On the displayed page, configure permissions for different roles on CodeArts Build resources.

Figure 3-1 Configuring project-level role permissions

For details about different roles' operation permissions on the specified build task, see **Configuring Roles and Permissions**.

Menu Icons and Their Usage on the CodeArts Build Homepage

CodeArts Build provides multiple choices of UI themes and layouts. This section walks you through the navigation bar that uses the **Infinite** theme and the **Classic** layout.

Table 3-2 Menu icons and their usage

Menu Icon	Description
o ~	Click this icon to expand the drop-down list and select the region you want to switch to.
	Data and resources are isolated between regions. Use your resources in the region where you purchased it.
Services ∨	Click this icon to expand the drop-down list and select Build to go to the CodeArts Build homepage, where you can find your build task list.
Scrum01 ▼	Click this project icon to expand the drop-down list and select the project you want to switch to when accessing CodeArts Build from the project page.
· · · More	Click this icon to expand the drop-down list and select the desired item to manage custom templates , custom build environments, files , recycle bin , or pools.
\triangleright	Click this icon to run the build task.
☆	Click this icon to favorite a build task .
***	Click this icon to expand the drop-down list and select the desired action to edit , clone , disable , or delete the build task.

4 Creating a Build Task

4.1 About Build Tasks

CodeArts Build gives you two build task setups so you can pick which fits your needs: a graphical user interface (GUI) or a code-centric YAML.

Build on GUI

In this mode, you simply assemble tools, set parameters, and then start a build on the GUI.

Advantages:

- **Ease of use**: For those who value a low barrier to kick off, this mode allows them to configure builds with clicks and drags rather than hand-crafting scripts.
- **Clear visuals**: The GUI shows your execution graph and dependencies, so you can watch each step unfold, pinpoint faults, and squash bugs on the spot.
- **Rapid onboarding**: For starters, the GUI mode is the fast lane to a working build without wrestling with scripting languages and build tools.

Build with Code

CodeArts Build allows you to define your build as code using YAML. Your configurations, such as build environments, parameters, commands, and actions, reside in a YAML file named **build.yml**. After creating this file, add it along with the source code to a code repository. The file will be used as a script by the system to run a build.

Advantages:

Flexibility

- By designing complex logic and conditions in a YAML file, you have granular controls on your build process.
- You can implement dynamic config, conditional branching, and loops as your needs evolve.

Maintainability

- YAML versioning benefits collaboration and management.
- With everything versioned, you get control over tweaks to your build config and roll back changes to sidestep risk from rogue edits.

Reusability

- Once a script is created, you can spin up multiple builds across projects in one go without duplicating configurations.
- Codify your common build steps as modules or functions, so you can plug identical components every time you need.

Automation

- To achieve automated build-and-deploy flows, plug code-based builds into the CI/CD system with less ceremony.
- Scripting your way to automation makes complex builds faster and rocksolid.

4.2 Creating and Managing Groups

For build tasks targeted at different modules or scenarios in the same project, you can group the tasks for easy management in CodeArts Build. A group named **Ungrouped** is automatically created along with the first created group to store ungrouped build tasks.

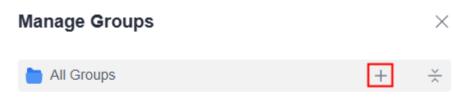
Constraints

- You can create a maximum of 50 groups.
- A group consists of a maximum of three levels of directories.

Creating a Group

- 1. Access the CodeArts Build homepage from the project list.
- 2. On the displayed page, click **Create one**. In the displayed dialog box, click 🛨.

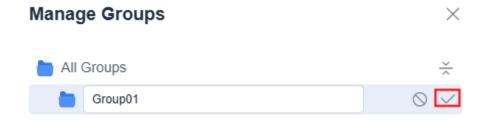
Figure 4-1 Creating a group





3. Set a group name and click .

Figure 4-2 Assigning a name to the group

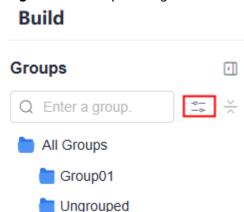


4. Click Close.

Managing Groups

1. Click - . The dialog box for group management is displayed.

Figure 4-3 Group management



2. Move the cursor to the row where the group is located.

Click to edit the group name.

Click to delete the group or adjust its order.

After a group is deleted, its tasks are automatically moved to **Ungrouped**.

4.3 Defining a Build Task on GUI

CodeArts Build provides a graphical user interface (GUI) where you can configure build tools and parameters.

Constraints

To enhance the security of your local code repository while using CodeArts Build, you can add the following IP addresses to your repository server's whitelist. This ensures that only CodeArts Build can access your repository.

AP-Singapore: 114.119.185.21

AF-Cairo: 101.46.68.29

LA-Santiago: 119.8.154.190

AF-Johannesburg: 182.160.17.185

ME-Riyadh: 101.46.48.183

LA-Mexico City2: 122.8.183.54 and 110.238.80.148

• Brazil: 119.8.85.121

• TR-Istanbul: 101.44.33.34, 101.44.34.216, and 101.44.36.238

Preparations

- To use CodeArts Repo repositories, you must have the operation permissions on CodeArts Repo.
- Create a CodeArts project by referring to CodeArts User Guide >
 "Preparations" > "Creating a Project".

If you already have a project available, skip this step.

 Create a repository by referring to CodeArts Repo User Guide > "Creating a CodeArts Repo Repository".

If you already have a CodeArts Repo repository or are using a third-party repository, skip this step.

Creating a Build Task with GUI

- 1. Access the CodeArts Build homepage from the project list.
- 2. Click **Create Task**. On the displayed page, configure the basic information of the build task by referring to **Table 4-1**. Click **Next**. The page for selecting a build template is displayed.

Table 4-1 Basic information

Parame ter	Description
Name	Assign a custom name to the build task. • Letters, digits, underscores (_), and hyphens (-) allowed. • 1 to 115 characters.
Project	 Select the project to which the build task belongs. This parameter is autofilled when you access CodeArts Build through the project list, so you can leave it as default. When accessing the service through the service homepage, select the project created in preparations.
Code Source	 Select the source from where the code to be compiled is pulled. Repo: Code is pulled from CodeArts Repo for your build. Other Repo: Code is pulled from CodeArts Repo repositories of other projects for your build. Select a project, and then select a code repository and a default branch. Pipeline: If Pipeline is selected as a code source, the build tasks can be executed only by running the corresponding pipeline and cannot be executed independently. The following code repositories are provided by third-party sources and not by CodeArts. GitHub: Code is pulled from GitHub for your build. Git: Code is pulled from other services for your build.
Service Endpoin t	Optional. You need to set this parameter when the Code Source is set to a third-party code repository. If you are using a third-party code repository for the first time, you will need to create a service endpoint. For details, see Creating a Service Endpoint .
Reposito ry	Select the repository from where the code to be compiled is pulled.
Default Branch	Select a default branch.

Parame ter	Description
Descripti	Optional. Enter additional information to describe the build task.
on	Max. 512 characters.

- 3. CodeArts Build has more than 30 built-in build templates. You can select a template that suits your requirements and click **OK** to create the build task. For details about templates, see **Table 4-2**.
 - You can also select Blank Template and add desired build actions when configuring a build task.
 - If preset templates do not meet your needs, you can also customize a template.

Table 4-2 System template description

Temp late	Description	Lang uage	Stage
Mave n	Build a Java project with Apache Maven.	Java	Configure the build environment, download code, build with Maven, and upload the software package to a release repo.
Quick App	Build a quick app with npm.	Java Scrip t	Configure the build environment, download code, build an Android quick app, and upload the software package to a release repo.
Djang o	Build a Django project.	Pyth on	Configure the build environment, download code, build with Setuptools, and upload the software package to a release repo.
lonic Andro id App	Develop an Android app with Ionic, an HTML5 mobile app development framework.	Java	Configure the build environment, download code, build an Android app with Ionic, and upload the software package to a release repo.

Temp late	Description	Lang uage	Stage
Ant	Apache Ant is a tool used to compile and deploy Java projects.	Java	Configure the build environment, download code, build with Ant, and upload the software package to a release repo.
CMak e	Build a cross-platform project with CMake.	C/C+ +	Configure the build environment, download code, build with CMake, and upload the software package to a release repo.
Grails	Build a Web application with Grails, a Web application development framework.	Groo vy	Configure the build environment, download code, build with Grails, and upload the software package to a release repo.
Grunt	Build a JavaScript project and run the project with Grunt.	Java Scrip t	Configure the build environment, download code, build with Grunt, and upload the software package to a release repo.
gulp	Use gulp to build an automation workflow and a frontend IDE.	Java Scrip t	Configure the build environment, download code, build with gulp, and upload the software package to a release repo.
Go	Build a Go project.	Go	Configure the build environment, download code, build with Go, and upload the software package to a release repo.
mono -linux	Compile a project with MSBuild and .NET on Mono Linux (x86 and Arm).	Java Scrip t	Configure the build environment, download code, build on Mono Linux, and upload the software package to a release repo.

Temp late	Description	Lang uage	Stage
Mave n- Conta iner	Use Apache Maven to build a Java project, create an image from Dockerfile, and push the image to SWR.	Java	Configure the build environment, download code, Build with Maven, create an image, and push the image to SWR.
Servic eStag e- Mave n- Imag e Build	Use Apache Maven to build a Java project, create an image from Dockerfile, and push the image to SWR.	Java	Configure the build environment, download code, Build with Maven, create an image, push the image to SWR, run shell commands, and upload the software package to a release repo.
servic estag e- npm- war	Build a WAR package with npm and upload the package to a release repo.	Java Scrip t	Configure the build environment, download code, Build with npm, run shell commands, Build with Ant, and upload the software package to a release repo.
Relea se Mave n Privat e Packa ge	To build a Java project with Maven using private dependency packages, you must first upload these packages to a self-hosted Maven repo in CodeArts Release.	Java	Configure the build environment, download code, and Build with Maven.
PHP	Declare, install, and package the PHP libraries your project depends on by using the PHP runtime environment and Composer.	PHP	Configure the build environment, download code, build in PHP, and upload the software package to a release repo.
Bazel	Build a project with Bazel.	Java	Configure the build environment, download code, build with Bazel, and upload the software package to a release repo.

Temp late	Description	Lang uage	Stage	
Shell	Run shell commands.	N/A	Configure the build environment, download code, and run shell commands.	
sbt	Build a Scala or Java project with sbt.		Configure the build environment, download code, build with sbt, and upload the software package to a release repo.	
npm	Build Vue and Webpack projects with npm.	Java Scrip t	Configure the build environment, download code, build with npm, and upload the software package to a release repo.	
Andro id APK	The Android build system compiles application resources and source code, and then packages them into APKs that can be deployed, signed, and distributed.	Java	Configure the build environment, download code, build with Android, and upload the software package to a release repo.	
Gradl e	Build a Java, Groovy, or Scala project with Gradle.	Java	Configure the build environment, download code, build with Gradle, and upload the software package to a release repo.	
server less- npm	Build Vue and Webpack projects with npm.	Java Configure the build environment, download code, build with npm, and upload the softwar package to a release repo.		
DevSt ar- Mave n- Conta iner Build	Use Apache Maven to build a Java project, create an image from Dockerfile, and push the image to SWR.		Configure the build environment, download code, Build with Maven, create an image, and push the image to SWR.	

Temp late	Description	Lang uage	Stage
DevSt ar- npm- Conta iner Build	Use npm to build a JavaScript project, create an image from Dockerfile, and push the image to SWR.	Java Scrip t	Configure the build environment, download code, Build with npm, create an image, and push the image to SWR.
DevSt ar- Pytho n- Conta iner Build	Build a Python project, create an image from Dockerfile, and push the image to SWR.	Pyth on	Configure the build environment, download code, Build with Setuptools, create an image, and push the image to SWR.
Yarn	Build a JavaScript project with Yarn.		Configure the build environment, download code, build with Yarn, and upload the software package to a release repo.
buildI mage	Build an image with a Dockerfile and push the image to SWR.	le and push the environment, downlo	
PyInst aller	Use the PyInstaller tool to compress the Python file into an executable program.	Pyth on	Configure the build environment, download code, build with PyInstaller, and upload the software package to a release repo.
Setup tools	ools Setuptools. on environment, code, build wi Setuptools, ar the software p		Configure the build environment, download code, build with Setuptools, and upload the software package to a release repo.
Build with Ark Comp iler	rith Ark Compiler on Ubuntu. rk omp		Configure the build environment, download code, build with Ark Compiler, and upload the software package to a release repo.

Temp late	Description	Lang uage	Stage
GNU- ARM	Design, develop, and use an Arm simulator with the GNU Arm toolchain.	С	Configure the build environment, download code, build with GNU Arm, and upload the software package to a release repo.

4. On the displayed **Build Actions** page, click **Save**. The build task is created. The new task appears in the build task list.

You can continue to configure this build task by referring to **Configuring a Build Task**.

Turning a Task Into a Template

- You can save the current build task as a template for later build task creation. The procedure is as follows:
 - a. On the **Build History** page, click ··· in the upper right corner and select **Make Template** from the drop-down list.
 - b. Enter the template name and description, and click **Save**.
 - c. Click the username, and select **All Account Settings** from the drop-down list.
 - d. In the navigation pane, choose **Build** > **Templates**. The saved template is displayed in the list.

The following table lists what you can do straight from the build templates.

Note that template edits and deletions are locked down to the template creator and the tenant account.

Table 4-3 Managing custom templates

Operation	Description
Search for a template	Enter a keyword in the search box to search for a template.
Favorite a template	Locate the desired template and click $\stackrel{\frown}{\Omega}$ to add the template to favorites.
Delete a template	Locate the desired template, click ••• , choose Delete , and click OK .

Creating a Service Endpoint

When you select any third-party repository on the **Basic Information** page, the **Endpoint** is a mandatory setting.

Service endpoints are extensions or plug-ins of CodeArts and provide the capability of connecting to third-party services.

By default, CodeArts Build pulls code from CodeArts Repo for your build. CodeArts Build also uses service endpoints to connect to third-party repositories to obtain project source code.

□ NOTE

- The network may be unstable or other problems may occur when a third-party repository is used.
- Use the code import function of CodeArts Repo for secure, stable, and efficient download and build.

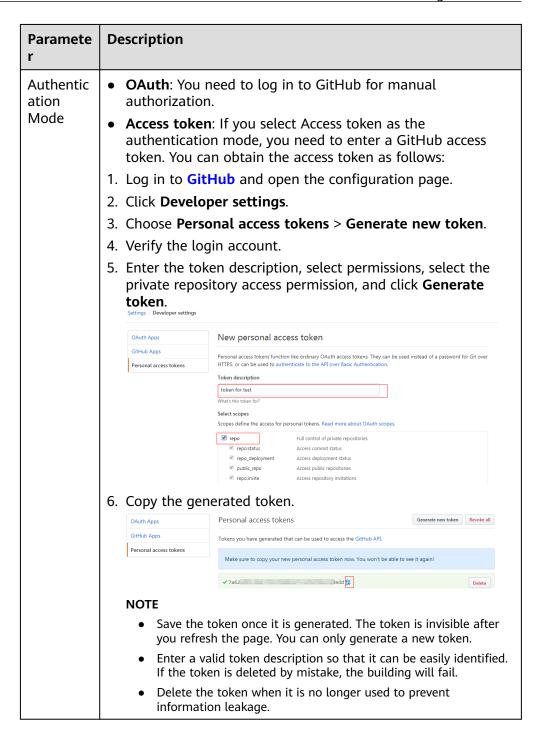
GitHub

To restrict CodeArts Build from accessing the GitHub repository, you can use OAuth or access token for authentication, as long as the code can still be pulled to complete the build.

You can also delete connections or withdraw authorization at any time to prevent password leakage.

- 1. Click **Create** next to **Service Endpoint**.
- 2. In the displayed dialog box, configure the following parameters.

Paramete r	Description
Service Endpoint Name	Assign a custom name to the service endpoint. Enter a maximum of 256 characters, including letters, digits, hyphens (-), underscores (_), periods (.), and spaces.



After the authorization is successful, return to the page for creating the build task.

Git

- 1. Click Create next to Service Endpoint.
- 2. In the displayed dialog box, configure the following parameters.

Parameter	Description
Service Endpoint Name	Assign a custom name to the service endpoint. Enter a maximum of 256 characters, including letters, digits, hyphens (-), underscores (_), periods (.), and spaces.
Git Repository URL	Enter the HTTPS address of the target Git repository.
Username	Optional. Enter the username of the target Git repository. Max. 300 characters.
Password or Access Token	Optional. Enter the password of the target Git repository. Max. 300 characters.

Click Confirm.

4.4 Defining a Build Task with Code

4.4.1 Creating a Build Task with Code

CodeArts Build allows you to define your build as code using YAML. Your configurations, such as build environments, parameters, commands, and actions, reside in a YAML file named **build.yml**. After creating this file, add it along with the source code to a code repository. The file will be used as a script by the system to run a build.

Constraints

You can only use the code hosted in CodeArts Repo for code-based builds.

Preparations

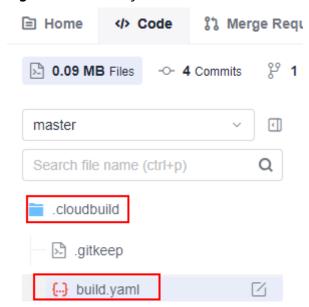
- Obtain permissions for CodeArts Repo. For details, see Granting Permissions for Using CodeArts Repo.
- Create a CodeArts project by referring to CodeArts User Guide >
 "Preparations" > "Creating a Project".

If you already have a project available, skip this step.

Creating a YAML File for Your Code-based Build

- 1. Access the CodeArts Build homepage from the project list.
- In the navigation pane, choose Code > Repo. Create a code repository by referring to Creating a Custom Repository.
- 3. Click the repository name. On the displayed page, choose **Create** > **Create Directory** and name the directory **.cloudbuild**.
- 4. In the .cloudbuild directory, choose Create > Create File to create a file named build.yml. Figure 4-4 shows the directory that stores files of the code repository.

Figure 4-4 Directory



If the YAML file is not stored in the .cloudbuild directory, you can use parameter CB_BUILD_YAML_PATH to specify the path of the YAML file in the code repository. For details about parameter settings, see Adding Custom Parameters.

5. Click and write the **build.yml** file by referring to the sample code in **Crafting Your build.xml for a Single Task** or **Crafting Your build.xml for Multiple Tasks**.

Configuring Basic Information

- 1. In the navigation pane, choose **CICD** > **Build**.
- 2. Click **Create Task**. On the displayed page, configure the basic information of the build task by referring to **Table 4-4**.

Table 4-4 Basic information

Parame ter	Description	
Name	Assign a custom name to the build task.	
	• Letters, digits, underscores (_), and hyphens (-) allowed.	
	• 1 to 115 characters.	
Project	Select the project to which the build task belongs.	
	This parameter is autofilled when you access CodeArts Build through the project list, so you can leave it as default.	
	 When accessing the service through the service homepage, select the project created in preparations. 	

Parame ter	Description
Code Source	If you select Repo , code is pulled from CodeArts Repo for your build.
Reposito ry	Select the repository from where the code to be compiled is pulled.
Default Branch	Select a default branch.
Descript ion	Optional. Enter additional information to describe the build task. Max. 512 characters.

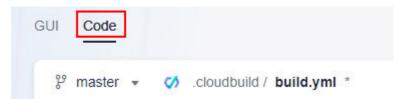
3. Click **Next**. The page for selecting a build template is displayed.

Selecting a Build Template

- Keep the default template selected.
 No matter which build template you select, code-based builds remain unaffected.
- 2. Click **OK**. The **Build Actions** page is displayed.

Configuring Build Actions

In the upper left corner of the **Build Actions** page, click the **Code** tab. The system automatically reads the YAML file from the code repository and the branch **configured in basic information**.



You can modify the YAML file by referring to the sample code in sections "Performing Basic Configurations" and "Configuring Build Actions" of **Configuring Build Task**. Any changes made to the YAML file will overwrite the original **YAML file you create for your code-based build** once the build task is completed.

Complete all the configurations and click **Save** to create a build task. The new task will be displayed on the build task list.

4.4.2 Crafting Your build.xml for a Single Task

The following example uses a built-in x86 executor with 8 vCPUs and 16 GB memory. The tasks involve compiling and building code from CodeArts Repo using Maven, and then uploading the resulting software package to a release repo.

For details about how to define build actions with code, navigate **Configuring a Build Task** > "Performing Basic Configurations" > "Configure Build Actions", locate your desired actions, and examine the sample code under "Build with Code". For

details about the YAML file structure of multiple tasks, see **Crafting Your build.xml for Multiple Tasks**.

```
version: 2.0 # The version number is a mandatory and unique parameter that must be set to 2.0.
params: # Build parameters that can be referenced during your build. If no build parameters are set here,
parameters created during task configuration are preferentially used.

    name: paramA

  value: valueA
 - name: paramB
  value: valueB
env: # Optional. It defines the build environment. x86 is used by default if neither env or envs is set.
 resource:
  type: docker # Agent pool type. The value can be docker or custom. docker indicates that the default
executor is used, and custom indicates that a custom executor is used.
  arch: X86 # The host type of the build environment can be either x86 or Arm.
  class: 8 vCPUs | 16 GB # The specification can be: 2 vCPUs | 8 GB, 4 vCPUs | 8 GB, 8 vCPUs | 16 GB, 16
vCPUs | 32 GB, or 16 vCPUs | 64 GB. This parameter is not required when the agent pool type is set to a
  pool: Mydocker #Agent pool name. This parameter is required when the agent pool type is set to a
custom one.
steps:
 PRE_BUILD: # It prepares for the build process by performing tasks such as downloading code and
running shell commands.
  - checkout:
     name: Code download # Optional
     inputs: # Action parameters
      scm: codehub # Code source: CodeArts Repo only
      url: xxxxxxxxx # This refers to the SSH address of the repository that the code is pulled from.
      branch: ${codeBranch} # Pulled code branch, which can be parameterized.
     inputs:
      command: echo ${paramA}
 BUILD: # It defines the build action. Only one BUILD can be configured.
  - maven: # Action keyword. Only specified keywords are supported.
  name: maven build # Optional
  image: xxx # Image address.
  inputs:
   command: mvn clean package
   upload_artifact:
  inputs:
```

4.4.3 Crafting Your build.xml for Multiple Tasks

path: "**/target/*.?ar"

A build task is the smallest unit that a build project can be broken down into for simple service scenarios. However, for more complex requirements, you may need to:

- Use a multi-repo approach to distribute build tasks that depend on each other across multiple machines.
- Set up multiple build tasks in a modular and fine-grained way, and run them in a specific order. Each task depends on the successful completion of its dependency task.

To handle such complex builds, CodeArts Build offers a task model called BuildFlow, which organizes multiple build tasks in a directed acyclic graph (DAG) and runs them concurrently based on their dependencies.

The following is a YAML file example for multiple tasks:

```
version: 2.0 # The version number is a mandatory and unique parameter that must be set to 2.0. params: # Define global parameters that apply across all subtasks.

- name: p
value: 1
```

env and envs are optional. Use envs if you need to specify conditions to determine the host

specification and type.

env: # This parameter takes precedence once being set. The host specification and type defined here will be used instead of the ones set in the build environment configuration.

resource:

type:docker # Agent pool type. The value can be **docker** or **custom**. **docker** indicates that the default executor is used, and **custom** indicates that a custom executor is used.

arch:X86 # The host type of the build environment can be either x86 or Arm.

class:8 vCPUs | 16 GB # The specification can be: 2 vCPUs | 8 GB, 4 vCPUs | 8 GB, 8 vCPUs | 16 GB, 16 vCPUs | 32 GB, or 16 vCPUs | 64 GB. This parameter is not required when the agent pool type is set to a custom one.

pool:Mydocker #Agent pool name. This parameter is required when the agent pool type is set to a custom one.

envs:

- condition: p == 1 # The following host specification and type are used if this condition is met. resource:

type: docker arch: ARM

- condition: p == 0 # The following host specification and type are not used if this condition is not met.

type: docker arch: X86

Configure either buildflow or buildflows. Use buildflows if you need to specify conditions to control job executions.

huildflow

strategy: lazy # Define the running policy of buildflow, which can be lazy or eager. The eager mode is used by default if this parameter is not defined.

lazy: prioritizes high-importance subtasks first. Once those succeed, downstream subtasks kick off. This policy helps conserve build resources when concurrency is tight. However, the overall build time can run longer.

eager: triggers every subtask at the same time. Dependents will prepare the environment and code and then wait for their prerequisite builds. Although resources may spend time on idle waiting, this policy slashes build time when you have plenty of concurrency capacity.

jobs: # defines subtasks to be orchestrated. Each subtask must have a unique name as its identifier. If subtask A depends on B, B has a higher priority. Subtasks with the same priority are triggered at the same time. This example defines three subtasks: Job1, Job2, and Job3. Job3 depends on Job1 and Job2 and will be run only after they are completed. Job1 and Job2 have equal priority and will be triggered at the same time.

- job: Job3 # Assign a custom name to the child task.

depends_on: # Define the task dependency. In this practice, the configuration indicates that Job3 depends on Job1 and Job2.

- Job1 - Job2

build_ref: .cloudbuild/build3.yml # Define the YAML build script to run during a job build.

- iob: Job1

build_ref: .cloudbuild/build1.yml

- job: Job2

build_ref: .cloudbuild/build2.yml

buildflows:

- condition: p == 1 # All jobs under this collection are executed if this condition is met. jobs: # It defines the tasks to be orchestrated.

- job: Job1 # Assign a custom name to the child task.

build_ref: 1.yml # YAML build script that needs to be run during a build.

params:

- name: abc value: 123

- condition: p == 1 # Job2 is executed if this condition is met.

job: Job2 build_ref: 2.yml params: - name: abc value: 123

During a build with YAML, parameters are used in the following priority (as shown in the preceding sample code):

Runtime parameters configured on the **Parameters** page > Default values of parameters configured on the **Parameters** page > Parameters defined in **build ref**

> Parameters defined in **params** under a job > Global parameters defined in **params** under BuildFlow

For tailored performance per scenario, you may define params in a different way:

You can configure **params** as global parameters that apply to all subtasks your orchestrated builds. If you demand job-specific parameters rather than **params** at the top, you can scope parameters directly inside your desired subtasks. The following code scopes parameter **isSubmodule** to **Job1** and **Job2**.

```
buildflow:
  jobs:
    - job: Job3
     depends_on:
        - Build Job1
        - Build job2
     build_ref: .cloudbuild/build3.yml
    - job: Job1
     params:
       - name: isSubmodule
        value: true
     build_ref: .cloudbuild/build1.yml
    - job: Job2
     params:
       - name: isSubmodule
        value: true
     build_ref: .cloudbuild/build2.yml
```

5 Configuring a Build Task

5.1 Performing Basic Configurations

5.1.1 Configuring the Build Environment

Configure a global runtime environment for a build task.

CodeArts Build also allows you to use custom executors, such as **LINUX**, **LINUX_DOCKER**, **WINDOWS**, and **MAC** (Linux, Linux Docker, Windows, and macOS executors). For build scenarios supported by these executors, see **Table 5-1**.

Table 5-1 Executor types and their description

Execu tor Type	Description
LINUX	 You can run shell commands to run the build task on a Linux VM. Before using CodeArts Build, you need to install build tools, such as Maven and Gradle, on custom executors.
	 Only the following build actions are available: Running Shell Commands, Uploading a Software Package to Release Repos, and Downloading a Package from Release Repos.

Execu tor Type	Description
LINUX _DOC	 When you run the build task, CodeArts Build starts a Linux Docker container in which the task is run.
KER	 The entire build process runs in the container. Once the task is finished, the container automatically removes the build image, which includes the code pulled during the build, the process data, and the build products.
	 You can configure the mapping between the host directory and container directory to share the host directory in the image.
	 The following build actions are functions are unavailable: Download File from File Manager, Building with Android, and Unit Test in Building with Maven.
WIND OWS	You will run build tasks on the Windows executor. This allows you to execute Windows-related build tasks.
	You can use Git Bash to run the shell script for your build.
	 Only the following build actions are available: Running Shell Commands, Uploading a Software Package to Release Repos, and Downloading a Package from Release Repos.
	 You can use Windows 7, Windows 10, Windows Server 2012, or Windows Server 2016.
	 Before customizing a Windows executor, ensure that you have installed the JDK and Git.
	Install the compilation tool. For example, install Maven if you will use it for your build.
MAC	You will run shell commands on the macOS executor for your build. This allows you to execute macOS-related build tasks.
	 Only the following build actions are available: Running Shell Commands, Uploading a Software Package to Release Repos, and Downloading a Package from Release Repos.
	You can select any macOS version in use.

Constraints

When building with YAML, you have an available environment with custom executors.

Build on GUI

- 1. Access the CodeArts Build homepage from the project list.
- 2. On the CodeArts Build homepage, search for the target task and click its name. The build task configuration page is displayed.
 - CodeArts Build presets the **Configure Build Environment** action. Set the parameters according to **Table 5-2**.

Table 5-2 Build environment parameters

Parameter	Description	
Environmen t	x86/Arm (Kunpeng) server NOTE Select the appropriate type of host you intend to use for software running on different chipset architecture. For example, if your software is designed for Arm (Kunpeng) servers, select Arm (Kunpeng).	
Mapping Between Host and Container Directories	Set up the directory mapping between the custom executor and the container, and then you can mount files like dependencies from the custom executor to the container for your builds. (This mapping needs to be set when Execution Host is set to Custom executor .)	
	If the Host Directory is set to /home and the Container Directory is set to /opt , then the content in the executor's local /home directory will be mounted to the /opt directory in the container.	

Build with Code

Modify **env** settings in **the YAML file** by referring to the following sample code of build environment configurations.

version: 2.0 # The version number is a mandatory and unique parameter that must be set to 2.0. env: # Optional. It defines the build environment. x86 is used by default if neither env or envs is set. resource:

type:docker # Agent pool type. The value can be **docker** or **custom**. **docker** indicates that the default executor is used, and **custom** indicates that a custom executor is used.

arch:X86 # The host type of the build environment can be x86 or Arm.

class:8 vCPUs | 16 GB # The specification can be: 2 vCPUs | 8 GB, 4 vCPUs | 8 GB, 8 vCPUs | 16 GB, 16 vCPUs | 32 GB, or 16 vCPUs | 64 GB. This parameter is not required when the agent pool type is set to a custom one.

pool:Mydocker #Agent pool name. This parameter is required when the agent pool type is set to a custom one.

Add the following code information to **the YAML file for your code-based build** by referring to the following sample code of BuildSpace.

Ⅲ NOTE

You have an available environment with custom executors.

version: 2.0

buildspace: # BuildSpace is used.

fixed: true path: kk clean: true clean_exclude:

- cache # Excluded path
- aa # Excluded path
- bb # Excluded path

Table 5-3 Parameters in the sample code of BuildSpace

Paramet er	Туре	Description
fixed	String	Optional. Set this parameter when you need a fixed directory for build executions.
		In CodeArts Build, an empty path (for example, / devcloud/ws/sMMM/workspace/j_X/) is randomly assigned to a build task as the root directory by default. This directory is called a "BuildSpace". Even for build tasks in the same project, BuildSpaces are randomly assigned to them.
		However, a fixed BuildSpace path is necessary in some scenarios. CodeArts Build allows users to configure BuildSpace to specify a fixed directory for a build.
		• true: A fixed path is used.
		false: A random path is used. The default value is false.
path	String	Optional. Set this parameter when you use a fixed directory for build executions.
		The fixed path is in the following format: /devcloud/slavespace/usr1/+"\${domainId}"+/. You can set the path parameter to add a path after the fixed path.
		For example, if the path is set to kk , the fixed path is / devcloud/slavespace/usr1/+"\${ domainId}"+/kk .
clean	String	Optional. Set this parameter when you require the fixed path to be cleared.
		true: The fixed path will be cleared. Files in the fixed path will be deleted each time the build task is complete.
		false: The fixed path will not be cleared. When the total size of files reaches the maximum capacity of the workspace, you need to manually free up space by setting clean to true. NOTE
		 If there is no clearance setting for the fixed path, all files within the current tenant's fixed path will be automatically deleted once the total file size reaches the upper limit of the workspace. The workspace refers to the custom executor specification.
		The default value is true .
clean_exc lude	String	Optional. Set this parameter when you want to exclude a few paths from the cleanup of the fixed path.
		Specify paths to exclude from the cleanup. Only level-1 folders in a fixed path can be specified.

5.1.2 Configuring the Code Download

In this action, you set the download mode for pulling code from the repository during the build process.

Build on GUI

You can specify a source version with a code repository tag or commit ID. Besides, you can enable auto update of submodules and Git Large File Storage (LFS) for your build.

The **Configure Code Download** action is preset. Set the parameters according to **Table 5-4**.

Table 5-4 Parameters for configuring the code download

Parameter	Description
Specify Repository Tag or Commit ID	Specify whether to specify a tag or a commit ID when running a build task.
	Do Not specify: All code is pulled for your build.
	Tag: Only the code with the specified tag is pulled for your build. When you run the build task, a dialog box will appear prompting you to enter a tag. A tag marks a point in a code repository. If you select Repo as the code source, you can create a tag by referring to Managing Tags. When using a third-party code repository as the source, you need to create a tag within that repository.
	Commit ID: Only the code with the specified commit ID is pulled for your build. When you run the build task, a dialog box will appear prompting you to enter a commit ID. A commit ID is the number generated when the code is committed. For example, the commit ID in a CodeArts Repo repository is shown in Figure 5-1.
	Figure 5-1 Commit ID
	© Home © COMP 12 Morph Responsion 8 © Reviews Associated Work Remark of Responsion Statistics Activities Activities Members O Settings SAMEWORK O Control 1 Section 0 time 1 companies
	master © build_docum_mape_repor / + Create + 2 Second by terrored (Chief) © build_docum_mape_repor / + Create + 2 Build_docum_mape_repor Chiefer
	west stant date - select one date:
	© Dozente III Hellowerd.pox ©

Parameter	Description
Clone Depth	Optional.
	The clone depth is the number of recent commits that will be cloned. A larger value indicates more commits will be fetched. The clone depth must be a positive integer. The recommended maximum value is 25.
	For example, setting the clone depth to 5 instructs the system to fetch only the five most recent commits, but no earlier records.
Auto Update	A submodule is a Git tool used to manage shared repositories for higher team efficiency. Submodules allow you to keep a shared repository as a subdirectory of your repository. You can isolate and reuse repositories, and pull latest changes from or push commits to shared repositories. For details, see Configuring a Submodule.
	Enabled: If the code repository contains submodules, the system automatically pulls the code from the submodule repository during a build.
	Disabled: The system does not automatically pull the code of the submodule repository.
Enable Git LFS	Determine whether to enable Git LFS to pull all files, including large files, such as audios, videos, and images, during a build. By default, these files are not pulled.

Build with Code (Downloading Code from a Single Repo)

Modify the code in the **PRE_BUILD** block in **Creating a YAML File for Your Code-based Build** by referring to the following sample code:

```
version: 2.0 # The value must be 2.0.

steps:

PRE_BUILD:

- checkout:

name: checkout

inputs:

scm: codehub # Code source: CodeArts Repo only

url: xxxxxxxxx # This refers to the SSH address of the repository that the code is pulled from.

branch: ${codeBranch} # Mandatory at any time and can be parameterized.

commit: ${commitId}

lfs: true

submodule: true

depth: 100

tag: ${tag}

path: test
```

Table 5-5 Parameters in the sample code for downloading a single repository

Paramet er	Туре	Description
scm	String	Enter a code source. Currently, only CodeArts Repo is supported. If this parameter is not configured in the YAML file, the code repository information configured in the build task will be used. The default value is codehub .
url	String	Enter the SSH address of the code repository from which the code will be pulled.
branch	String	Specify the branch from which to pull the code. You can use <i>\${codeBranch}</i> to reference this parameter.
commit	String	Optional. If needed, you can enter a commit ID to indicate the specific version of the source code to be pulled for your build.
		You can also use <i>\${commitId}</i> to reference this parameter.
tag	String	Optional. If needed, you can enter a tag to indicate the specific version of the source code to be pulled for your build.
		You can also use <i>\${tag}</i> to reference this parameter. If you provide both the commit ID and tag, the build using the commit ID will be run first.
depth	Int	Optional. Shallow clone depth. When a commit ID is specified for builds, depth must be greater than or equal to the depth of the commit ID.
		The default value is 1 .
submod ule	Bool	Optional. Specify whether to pull the submodules. • true: Pull. • false: Do not pull.
		The default value is false .
lfs	Bool	 Optional. Specify whether to enable Git LFS. true: Enable. false: Disable. By default, CodeArts Build does not pull large files such as
		audios, videos, and images. You can enable Git LFS to pull all files. The default value is false .
path	String	Optional. Sub-path for cloning: The code is downloaded to the sub-path.

Build with Code (Downloading Code from Multiple Repos via Manifest)

In scenarios such as Android and HarmonyOS, hundreds or even thousands of code repositories need to be integrated at the same time during one build. The efficiency of integrating and downloading multiple code repositories is critical.

CodeArts Build integrates the Repo download tool. You only need to perform simple configurations to download multiple code repositories. Currently, this feature only applies to CodeArts Repo repositories.

Modify the code in the PRE_BUILD block in Creating a YAML File for Your Code-based Build by referring to the following sample code:

```
version: 2.0 # The value must be 2.0.
steps:

PRE_BUILD:
- manifest_checkout:
    name: "manifest"
    inputs:
    manifest_branch: "master"
    manifest_file: "default.xml"
    path: "dir/dir02"
    repo_url: "https://example.example.example.example.example.com/xx/manifest.git"
    manifest_file: "default.xml"
    path: "dir/dir02"
    repo_url: "https://example.example.example.example.example.com/xx/git-repo.git"
    repo_branch: "master"
    username: "someone"
    password: "${PASSWD}"
```

Table 5-6 Parameters in the sample code for downloading multiple repositories via Manifest

Parameter	Туре	Description
name	Strin g	Optional. Enter the action name. The default value is manifest_checkout.
manifest_u rl	Strin g	Enter the address of a Manifest repository that includes an XML file.
manifest_b ranch	Strin g	Optional. Enter the Manifest branch or revision. The default value is HEAD .
manifest_fi le	Strin g	Optional. Manifest file path. The default value is default.xml.
		The repositories defined in manifest_file must be of the same code source.
path	Strin g	Optional. Download path of all sub-repositories of the custom Manifest file, which is the relative path of the working path.
		The path cannot start with a slash (/) and cannot contain any period (.). The default value is the working path.

Parameter	Туре	Description
repo_url	Strin g	Optional. Enter the address of the CodeArts Repo repository. The default value is https://gerrit.googlesource.com/git-repo.
		You must have been granted the download permission for the CodeArts Repo repository specified by repo_url (the repository can either be open-source or private, but configured with account and password).
repo_branc h	Strin g	Optional. Enter the branch of the CodeArts Repo repository. The default value is stable .
username	Strin g	Optional. Enter the username used for downloading the repository. This parameter is mandatory for private repositories.
password	Strin g	Optional. Enter the HTTPS password used for downloading the repository. This parameter is mandatory for private repositories.

- manifest_url and manifest_file must use the same code source. For private repositories, the account indicated by the username and password you use must have been granted the download permission.
- If the optional parameters mentioned above are left empty, default values will be used.
- When using private repositories, you are advised to configure the username and password using private build parameters. For details, see Configuring Parameters.

5.2 Selecting Build Actions

You can select desired build actions that suits your scenarios.

Build on GUI

Go to the **Build Actions** page, where you can see the default action combination of the selected template.

- You can click on a build action to add it to your build task. For details about how to configure each build action, see their "Build on GUI" instructions in Configuring Build Actions.
 - If the tool version preset in the build action cannot meet your requirements, you can **customize the build environment**.
- To delete a build action, select it and choose *** > **Delete**.
- To copy a build action, select it and choose *** > Clone.
- To disable a build action that is reserved, select it and choose > Disable.
 To enable the action again, select it and choose > Enable.

Upload to Release Repo
Upload a software package to Release Repo.

Clone

Add Action

Delete

Disable

Figure 5-2 Adding, cloning, deleting, and disabling a build action

- To set up the environment for running your code-based build task, set env of the YAML file by referring to the sample code in the "Build with Code" of Configuring the Build Environment.
- To set up the code download mode, set PRE_BUILD of the YAML file by referring to the sample code in the "Build with Code" of Configuring the Code Download.
- To set up the build actions, set **BUILD** of **the YAML file** by referring to the sample code in the "Build with Code" of each build action in **Configuring Build Actions**.

5.3 Configuring Build Actions

5.3.1 Building with Maven

Maven's primary goal is to build Java-based projects and manage project information and dependencies.

Constraints

- If you need to use a custom **settings** file for a Maven build, ensure it meets the following requirements:
 - The maximum file size is 100 KB.
 - The file type must be .xml, .key, .keystore, .jks, .crt, or .pem.
 - A maximum of 20 files can be uploaded.

Build on GUI

Add **Build with Maven**, when **configuring build actions**. Set the parameters according to **Table 5-7**.

Table 5-7 Parameters for building with Maven

Paramet er	Description
Action Name	 Assign a custom name to the build action. The name can contain: Letters, digits, hyphens (-), underscores (_), commas (,), semicolons (;), colons (:), periods (.), slashes (/), and parentheses. 1 to 128 characters.
Tool Version	Select a tool version that matches your current development environment. For tool versions supported by CodeArts Build, see build tools and versions. If the current tools and versions do not meet your requirements, you can customize a build environment.
Comman ds	Configure the Maven commands, or use the default ones. For more commands, see the Maven official website.
Continue After Failure	Specify whether to proceed after the current action fails by setting the parameter to either Yes or No .

Paramet er	Description
setting File Configura tion	• Generate setting File with Repositories: The optimal repository access mode is automatically configured based on your IP address when you access the setting.xml file provided by CodeArts. Your IP address may be in regions in or outside the current country. You are advised to retain the default settings.
	The setting.xml file defines the default dependency pull sequence and mirror source proxy. If you need to use a custom setting.xml file, add a custom setting.xml file and add settings settings.xml to the end of the default packaging command. Then, you can use the added settings.xml file to build with Maven.
	# Package a project without performing unit tests. mvn package -Dmaven.test.skip=true -U -e -X -Bsettings settings.xml
	 Public Repositories: By default, Huawei Mirrors is added, and Huawei SDK repositories has been configured. This configuration is used only when you need to add a public repository that is not provided by CodeArts. The procedure is as follows: Click Add.
	 Enter the repository address, and select Release and Snapshot as required. Select either Release or Snapshot, or both. Release: If this option is selected, the build process attempts to download the release version dependency from the
	repository. Snapshot : If this option is selected, the build process attempts to download the snapshot version dependency from the repository.
	 Private Repositories: Self-hosted repos provided by CodeArts have been configured by default. This configuration is used only when you need to add other private repository. The procedure is as follows:
	1. Create a Nexus repository service endpoint.
	 Click Add, select the service endpoint created in the previous step, and select Release and Snapshot as required. Release and Snapshot are two types of repositories. Pay attention to their differences. If you upload a dependency to a release repository, it cannot be downloaded during a build.
	 Snapshot: For private dependency packages released for debugging, add the -SNAPSHOT suffix to the dependency version (for example, 1.0.0-SNAPSHOT). During each release, the dependency is automatically released to the snapshot repository. The version does not need to be updated each time the dependency is released. You can

Paramet er	Description
	add the -U parameter to the build command to obtain the latest version.
	 For officially released private dependency packages, do not add the -SNAPSHOT suffix to the dependency version (for example, 1.0.0). During each release, the dependency is automatically released to the release repository. The version must be updated each time the dependency is released. Otherwise, the latest dependency package cannot be obtained during the build.
Release to Self- hosted Repos	By default, CodeArts Build uses the self-hosted repos as the download source of private dependency. The configuration is required for uploading build products to the self-hosted repos and store the build products as dependencies for other projects. Before the configuration, create a self-hosted repo . The configuration procedure is as follows:
	• Do not configure POM : Private dependencies do not need to be released to the self-hosted repo of CodeArts Artifact.
	 Configure all POMs: Deployment configurations are added to all pom.xml files of the project. The mvn deploy command is used to upload the built dependency package to a self-hosted repo. In the command window, use the number sign (#) to comment out the mvn package -Dmaven.test.skip=true -U -e -X -B command, as shown in the following figure.
	<pre># Package a project without performing unit tests. #mvn package -Dmaven.test.skip=true -U -e -X -B</pre>
	Delete the number sign (#) before the #mvn deploy - Dmaven.test.skip=true - U - e - X - B command, as shown in the following figure.
	<pre># Package a project and release dependencies to Self-hosted Repos. # Release build results to Self-hosted Repos for other Maven projects. # Release the build results to Self-hosted Repos, not Release Repos. mvn deploy -Dmaven.test.skip=true -U -e -X -B</pre>
	The uploaded private dependency can be referenced by adding the groupId , artifactId , and version coordinates in the pom.xml file to other projects.
Unit Test	To process unit test results, set the parameters. For details, see Configuring a Unit Test.
Cache	Opt to use the cache to improve the build speed. If you set Use Dependency Cache to Yes , the downloaded dependency package is cached during each build. In this way, the dependency package does not need to be pulled repeatedly during subsequent builds, which effectively improves the build speed.
	The dependency cache files of a Maven build task cannot be updated. The cache directory will be updated only when new dependencies are imported to the task.

Modify the code in the **BUILD** block in **Creating a YAML File for Your Code-based Build** by referring to the following sample code:

```
version: 2.0 # The value must be 2.0.
steps:
 BUILD:
  - maven:
      image: cloudbuild@maven3.5.3-jdk8-open
      inputs:
       settings:
         public_repos:
          - https://mirrors.example.com/maven
       cache: true # Determine whether to enable caching.
       unit_test:
         coverage: true
         ignore_errors: false
         report_path: "**/TEST*.xml"
         enable: true
         coverage_report_path: "**/site/jacoco"
       command: mvn package -Dmaven.test.failure.ignore=true -U -e -X -B
       ignore_fail: true
```

Table 5-8 Parameters in the sample code

Param eter	Туре	Description
image	String	The image address can be in either of the following formats:
		• cloudbuild@maven3.5.3-jdk8-open: This address starts with cloudbuild and uses the at sign (@) as a separator, with the default image provided by CodeArts Build following it. For image build tool versions supported by CodeArts Build, see Build Tool Versions.
		Use a complete SWR image path, for example, swr.example.example.com/codeci_test/demo:141d26c455abd6d7.xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
settings	Мар	Optional. Set this parameter if you need a custom settings.xml file.
		If this parameter is not set, the setting.xml file provided by CodeArts is used by default. If you need to use a custom settings.xml file, add a custom setting.xml file and add settings settings.xml to the end of the default packaging command mvn package - Dmaven.test.failure.ignore=true -U -e -X -B .

Param eter	Туре	Description
cache	Bool	Optional. Specify whether to enable cache. • true: Enable. • false: Disable. The default value is false.
comma nd	String	Configure the Maven command. For more commands, see the Maven official website.
unit_tes t	Мар	Optional. Configure the unit test. For details, see Configuring a Unit Test.
ignore_ fail	String	Whether to proceed after the current action fails. • true: Yes • Empty: No

Adding a Custom setting.xml File

The file restrictions are as follows:

- The maximum file size is 100 KB.
- The file type must be .xml, .key, .keystore, .jks, .crt, or .pem.
- A maximum of 20 files can be uploaded.

You can add files in either of the following ways:

Build on GUI

- a. In the Commands window of the Build with Maven action, run the cat / home/build/.m2/settings.xml command. After the task is complete, the content of the settings.xml file will be displayed in the build logs.
- b. Customize a new **settings.xml** file according to the **settings.xml** file's information displayed in the build logs.
- Add the Download File from File Manager action before the Build with Maven action.
 - Assign a custom name to the action and select a tool version. Currently, only **shell4.2.46-git1.8.3-zip6.00** is supported.
- d. Click **Upload**. In the displayed dialog box, select the file created in **b**, add a description, select the agreements, and click **Save**.
- e. Expand the **File Name** drop-down list and select the uploaded **setting.xml** file.

• Build with Code

Modify the code in the **BUILD** block in **Creating a YAML File for Your Code-based Build** by referring to the following sample code:

version: 2.0 # The value must be 2.0. steps:

```
BUILD:
- download_file:
inputs:
name: settings.xml
ignore_fail: true
```

Table 5-9 Parameters in the sample code for downloading a file

Para mete r	Туре	Description
nam e	String	Name of the setting file.
ignor e_fail	String	Whether to proceed after the current action fails. • true: Yes • Empty: No

You can access the uploaded files in either of the two ways.

- On the CodeArts Build homepage, click **More** and select **Files**.
- Alternatively, click **Manage Files** in the **Download File From File Manager** action.

On the **Files** page, you can edit, download, and delete files, as well as configure operation permissions for other users.

- Enter a keyword in the search box to search for a file.
- Click in the **Operation** column to modify the file name and specify whether to allow all members of your account to use the file in CodeArts Build
- Click in the **Operation** column to download the file.
- Click ••• in the **Operation** column and select **Delete** from the drop-down list. Confirm the deletion as prompted.
- Click ••• in the Operation column and select Modify Permissions from the drop-down list. In the displayed dialog box, configure file operation permissions for the user.

By default, the creator has all permissions, which cannot be deleted or modified.

Figure 5-3 Configuring file operation permissions for a user



Permission	Role with the Permission
Add users	All users in the project
View a file	File creator and users under the same account
Use a file	File creator and users with the use permissions configured by the file creator
Update a file	File creator and users with the update permissions configured by the file creator
Delete a file	File creator and users with the delete permissions configured by the file creator
Modify permissions	File creator

Table 5-10 Roles and their permissions on files

Other Operations

Configuring a Unit Test and Generating a Unit Test Coverage Report Using JaCoCo are optional. Configure them if needed.

- Configuring a unit test: The Maven build action leverages unit tests to ensure that your code's core functions operate as expected. Within a Maven project, testing frameworks like JUnit are used to write and execute these tests.
- Generating a unit test coverage report using JaCoCo: The unit test coverage report distinguishes between the code that has been examined through unit tests and those that have not been covered. It helps you understand whether your tests have checked all code paths and logic.

Configuring a Unit Test

- 1. Before configuring a unit test, you need to write unit test code in the project and ensure that the following conditions are met:
 - The storage location of unit test code must comply with the default unit test case directory specifications and naming specifications of Maven. You can specify the case location in the configuration.
 - For example, if the unit test cases are stored in **src/test/java/ {{package}}/**, the unit test is automatically executed during a Maven build.
 - The project cannot contain the configuration code for ignoring unit test cases. Specifically, ensure that the **pom.xml** file of the project does not contain the following code:

 The JUnit dependency needs to be added to the pom.xml file. The following is the sample code that needs to be added:

```
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
         <version>4.13.1</version>
</dependency>
```

2. Create a unit test class in the code repository, as shown in Figure 5-4.

Figure 5-4 Unit test file directory



The following sample code is from the **Demo.java** file:

package test;

```
public class Demo {
  public String test(Integer i) {
    switch (i) {
      case 1:
         return "1";
      case 2:
         return "2";
      default:
         return "0";
    }
}
```

The following sample code is from the **DemoTest.java** file:

```
package test;
import org.junit.Test;

public class DemoTest {
    private Demo demo=new Demo();
    @Test
    public void test(){
        assert demo.test(1).equals("1");
        assert demo.test(2).equals("2");
        assert demo.test(3).equals("0");
    }
}
```

- 3. Configure a unit test in the build action.
 - Build on GUI
 - In the command window displayed in action Build with Maven, use the number sign (#) to comment out the mvn package -Dmaven.test.skip=true -U -e -X -B command.

```
# Package a project without performing unit tests.
#mvn package -Dmaven.test.skip=true -U -e -X -B
```

ii. Delete the number sign (#) before the #mvn package -Dmaven.test.failure.ignore=true -U -e -X -B command.

```
# Package a project, perform unit tests while ignoring failures, and check dependency updates.

# Perform unit tests and use test reports for analysis.

# Enable test report printing and specify the storage location.

mvn package -Dmaven.test.failure.ignore=true -U -e -X -B
```

iii. Expand **Unit Test** and set the following parameters **Table 5-11**.

Table 5-11 Parameters for unit test configuration

Parameter	Description
Print Test Results	 Specify whether to process unit test results. Yes: Process unit test results. No: Do not process unit test results.
Ignore Test Failure	If you choose to process the unit test result, you need to configure whether to ignore the case failure.
	Yes: If the case fails, the build task will continue.
	No: If the case fails, the build task will also fail.

Parameter	Description
Test Report	The unit test results are aggregated into a visual test report. To retrieve these results, you need to specify the path of the unit test result files.
	In most cases, retain the default path **/TEST*.xml. To improve the accuracy of the result, specify a precise report path, for example, target/surefire-reports/TEST*.xml.
Print Unit Test Results	Specify whether to process unit test coverage results. If you select Yes , a coverage test report is generated. For details about the configuration, see Generating a Unit Test Coverage Report Using JaCoCo .
Report Location	If you choose to process the unit test coverage results, enter the relative path to the project root directory, for example, target/site/jacoco. Once you have done this, all files in that directory will be packaged and uploaded.

Modify the code in the **BUILD** block in **Creating a YAML File for Your Code-based Build** by referring to the following sample code:

```
version: 2.0 # The value must be 2.0.

steps:

BUILD:
- maven:

    unit_test:
    coverage: true
    ignore_errors: false
    report_path: "**/TEST*.xml"
    enable: true
    coverage_report_path: "**/site/jacoco"
    command: mvn package -Dmaven.test.failure.ignore=true -U -e -X -B
```

Table 5-12 Unit test parameters

Paramet er	Туре	Description
enable	Bool	Optional.
		Specify whether to process unit test results.
		 true: Process unit test results. To set this parameter to true, add the - Dmaven.test.failure.ignore=true parameter to the end of the mvn command.
		false: Do not process unit test results.
		The default value is true .

Paramet er	Туре	Description
ignore_er	Bool	Optional.
rors		Specify whether to ignore test failure.
		• true : Ignore test failure. If the case fails, the build task will continue.
		• false: Do not ignore test failure. If the case fails, the build task will also fail.
		The default value is true .
report_pa th	Strin g	Enter the path for storing unit test data. Specify an accurate report path, for example, target/surefire-reports/TEST*.xml.
coverage	Bool	Optional.
		Specify whether to process coverage data. If you want to set this parameter to true , see Generating a Unit Test Coverage Report Using JaCoCo .
		• true: Process coverage data.
		false: Do not process coverage data.
		The default value is false .
coverage_	Strin	Optional.
report_pa th	g	If you choose to process the unit test coverage results, enter the relative path to the project root directory, for example, target/site/jacoco . Once you have done this, all files in that directory will be packaged and uploaded.

4. Once the task is successfully completed, you can access the test report on the testing tab of the task execution details page. When you opt to print the unit test coverage report, a report is generated. You can download it by clicking the button for downloading the test coverage report.

Generating a Unit Test Coverage Report Using JaCoCo

If you set **Print Unit Test Results** to **Yes**, complete configurations as follows:

• Configuration method for a single-module project

To generate the unit coverage report, add **jacoco-maven-plugin** to the project by including the following configuration to the **pom.xml** file.

By default, the JaCoCo report goal is bound to the **verify** phase. You need to change the report goal to the **test** phase.

```
<plugin>
  <groupId>org.jacoco</groupId>
  <artifactId>jacoco-maven-plugin</artifactId>
  <version>0.8.5</version>
  <executions>
   <execution>
  <goals>
```

Configuration method for a multi-module project

Assume that the code structure of a multi-module project looks like the following example. You will walk through how to configure and generate a unit test report.

a. Add an aggregation module named **report** to the project. Your code directory structure then looks like this:

```
module1
pom.xml
module2
pom.xml
module3
pom.xml
pom.xml
pom.xml
pom.xml
pom.xml
pom.xml
```

b. Add **jacoco-maven-plugin** to the pom.xml file in the root directory of the project. The following is a code sample:

c. Configure the **pom.xml** file of the aggregation module.

Use **dependency** elements to introduce all dependency modules and use **report-aggregate** to define the JaCoCo aggregation goal.

```
<dependency>
    <groupId>${project.groupId}</groupId>
     <artifactId>module3</artifactId>
     <version>${project.version}</version>
  </dependency>
</dependencies>
<build>
  <plugins>
     <plugin>
       <groupId>org.jacoco</groupId>
       <artifactId>jacoco-maven-plugin</artifactId>
       <version>0.8.3</version>
       <executions>
          <execution>
            <id>report-aggregate</id>
             <phase>test</phase>
             <goals>
               <goal>report-aggregate</goal>
             </goals>
          </execution>
       </executions>
     </plugin>
  </plugins>
</build>
```

d. Once the configuration is completed, run mvn test in the root directory of the project. After the command is successfully executed, the coverage report of each module is generated in the report/target/site/jacocoaggregate directory. You can also customize an output path for the reports by configuring outputDirectory.

5.3.2 Building with Android

The Android build system compiles application resources and source code, and then packages them into APKs that can be deployed, signed, and distributed.

Build on GUI

 Add Build with Android, when configuring build actions. Set the parameters according to Table 5-13.

Table 5-13 Parameters for building with Android

Paramete r	Description		
Action Name	Assign a custom name to the build action. The name can contain:		
	• Letters, digits, hyphens (-), underscores (_), commas (,), semicolons (;), colons (:), periods (.), slashes (/), and parentheses.		
	• 1 to 128 characters.		
Gradle	Select a tool version that matches your current development environment.		
	For tool versions supported by CodeArts Build, see build tools and versions. If the current tools and versions do not meet your requirements, you can customize a build environment.		
JDK	Select a tool version that matches your current development environment.		
	For tool versions supported by CodeArts Build, see build tools and versions. If the current tools and versions do not meet your requirements, you can customize a build environment.		
NDK	Select a tool version that matches your current development environment.		
	For tool versions supported by CodeArts Build, see build tools and versions . If the current tools and versions do not meet your requirements, you can customize a build environment .		
Command s	Configure the Gradle commands, or use the default ones. For more commands, see the Gradle official website .		
Continue After Failure	Specify whether to proceed after the current action fails by setting the parameter to either Yes or No .		

2. If you need to use apksigner to sign Android APKs, add the **Sign Android APK** action and configure the parameters according to the following table.

Parameter	Description	
Action Name	Assign a custom name to the build action. The name can contain:	
	 Letters, digits, hyphens (-), underscores (_), commas (,), semicolons (;), colons (:), periods (.), slashes (/), and parentheses. 	
	• 1 to 128 characters.	
APK Location	Location of the .apk file to be signed after the Android build. You can use regular expressions, such as build/bin/*.apk, to match the built APK package.	

Parameter	Description
Keystore File	Select the keystore file used for signature from the drop- down list. For details about how to create and upload the file, see Generating the Keystore Signature File and Uploading It for Management.
Keystore Password	Optional. Enter the custom password of the keystore file.
Alias	 Assign a custom alias to the key. The value must start with a letter and can contain letters, digits, underscores (_), hyphens (-), and periods (.). The value contains 1 to 128 characters.
Key Password	Optional. Enter a custom key password.
Apksigner Command	Enter a custom signature parameter. By default,verbose is added to display the signature details.
Continue After Failure	Specify whether to proceed after the current action fails by setting the parameter to either Yes or No .

Once you have finished configuration, run the build task. If the task is executed successfully, check the build logs. If the logs of the **Sign Android APK** action display **Signed**, it means that the signing process was successful.

Build with Code

Modify the code in the **BUILD** block in **Creating a YAML File for Your Code-based Build** by referring to the following sample code:

1. The following sample code is for the Android build:

```
version: 2.0 # The value must be 2.0.
steps:
 BUILD:
  - android:
     inputs:
      gradle: 4.8
      jdk: 1.8
      ndk: 17
      command: |
       cat ~/.gradle/init.gradle
        cat ~/.gradle/gradle.properties
        cat ~/.gradle/init_template.gradle
        rm -rf ~/.gradle/init.gradle
        rm -rf /home/build/.gradle/init.gradle
        # Gradle Wrapper provided by CodeArts Build is used for cache acceleration.
        cp /cache/android/wrapper/gradle-wrapper.jar ./gradle/wrapper/gradle-wrapper.jar
        # Build an unsigned APK.
        /bin/bash ./gradlew assembleDebug -Dorg.gradle.daemon=false -d --stacktrace
     ignore_fail: true
```

Table 5-14 Parameters in the sample code for the Android build

Para meter	Туре	Description
comm and	String	Enter the Gradle command. For more commands, see the Gradle official website.
gradle	String	Select a tool version that matches your current development environment. For tool versions supported by CodeArts Build, see build tools and versions. If the current tools and versions do not meet your requirements, you can customize a build environment.
jdk	String	Select a tool version that matches your current development environment. For tool versions supported by CodeArts Build, see build tools and versions. If the current tools and versions do not meet your requirements, you can customize a build environment.
ndk	String	Select a tool version that matches your current development environment. For tool versions supported by CodeArts Build, see build tools and versions. If the current tools and versions do not meet your requirements, you can customize a build environment.
ignore _fail	String	Whether to proceed after the current action fails.true: YesEmpty: No

2. The following sample code is to sign the Android APK.

```
version: 2.0 # The value must be 2.0.
steps:
BUILD:
- android_sign:
    inputs:
    file_path: build/bin/*.apk
        keystore_file: androidapk.jks
        keystore_password: xxxxxx
        alias: keyalias
        key_password: xxxxxx
        apksigner_commond: --verbose
        ignore_fail: true
```

Table 5-15 Parameters in the sample code for signing the Android APK

Paramete r	Туре	Description
file_path	Strin g	Location of the .apk file to be signed after the Android build.
		You can use regular expressions, such as build/bin/ *.apk, to match the built APK package.
keystore_f ile	Strin g	Name of the keystore file. For details about how to create and upload the file, see Generating the Keystore Signature File and Uploading It for Management.
keystore_	Strin g	Optional.
password		Enter the custom password of the keystore file.
alias	Strin	Alias of the keystore file.
	g	The value must start with a letter and can contain letters, digits, underscores (_), hyphens (-), and periods (.).
		The value contains 1 to 128 characters.
key_pass	Strin	Optional.
word	g	Enter a custom key password.
apksigner _common d	Strin g	Enter a custom signature parameter. By default, verbose is added to display the signature details.
ignore_fai	Strin	Whether to proceed after the current action fails.
l	g	• true: Yes
		Empty: No

Android Version Description

- SDK: used to specify **compileSdkVersion**.
- Build Tools: used to specify **buildToolsVersion**.

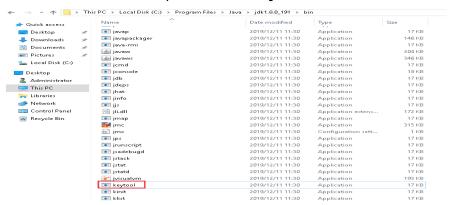
You can find the two versions in the **build.gradle** file or the global configuration file (user-defined) of the project.

```
Size: 959 bytes
app/build.gradle
      apply plugin: 'com.android.application'
  3
      android {
  4
          compileSdkVersion 23
  5
           buildToolsVersion '25.0.0'
  6
  8
  q
           defaultConfig {
              applicationId "cn.bluemobi.dylan.step"
 10
              minSdkVersion 17
 11
              targetSdkVersion 23
 12
 13
               versionCode 1
               versionName "1.0"
 14
 15
              testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
 16
```

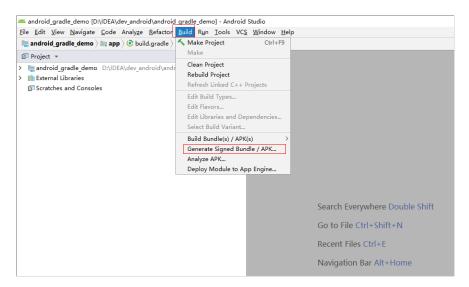

- Select compileSdkVersion or buildToolsVersion based on project requirements.
- The Gradle wrapper build mode is also supported. If the provided Gradle version does
 not meet your requirements, you can run the gradlew command for build using the
 wrapper. The required Gradle version will be automatically downloaded. Example of the
 build command: ./gradlew clean build.

Generating the Keystore Signature File and Uploading It for Management

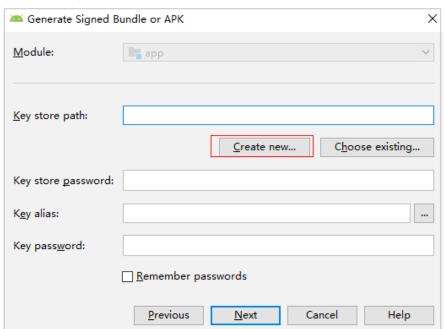
- 1. The keystore signature file can be generated in either of the following ways:
 - Using Keytool in JDK to Generate Signature Files
 - i. Find the JDK installation path and run keytool.exe.



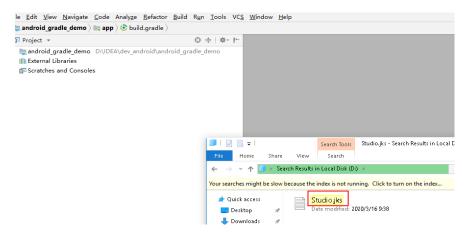
- ii. Run the following command to generate a **.jks** file: keytool -genkeypair -storepass 123456 -alias apksign -keypass 123456 -keyalg RSA validity 20000 -keystore D:/android.jks
- Using Android Studio to Generate Signature Files
 - Open the Studio client and choose Build > Generate Signed Bundle/APK.



- ii. Select APK and click Next.
- iii. Click **Create new...**. In the displayed dialog box, enter related information, and click **OK**. Then click **Next**.



iv. View the generated signature file.



- 2. You can upload the keystore signature file to the **Files** page in either of the following ways:
 - In the Build with Android action, click Upload next to Keystore File. In the displayed dialog box, select a file, add a description, select the related agreements, and click Save.
 - On the CodeArts Build homepage, choose More > Files. On the displayed page, click Upload File. In the displayed dialog box, select a file, add a description, select the related agreements, and click Save.

On the **Files** page, you can edit, download, and delete files, as well as configure operation permissions for other users.

- Enter a keyword in the search box to search for a file.
- Click in the Operation column to modify the file name and specify whether to allow all members of your account to use the file in CodeArts Build.
- Click in the Operation column to download the file.
- Click ••• in the Operation column and select Delete from the dropdown list. Confirm the deletion as prompted.
- Click ••• in the Operation column and select Modify Permissions from the drop-down list. In the displayed dialog box, configure file operation permissions for the user.

By default, the creator has all permissions, which cannot be deleted or modified.

Figure 5-5 Configuring file operation permissions for a user



Permission Role with the Permission Add users All users in the project View a file File creator and users under the same account Use a file File creator and users with the use permissions configured by the file creator File creator and users with the update permissions Update a file configured by the file creator Delete a file File creator and users with the delete permissions configured by the file creator Modify File creator permissions

Table 5-16 Roles and their permissions on files

Helpful Links

Hit snags in Android builds? See Android Build FAQs.

5.3.3 Building with npm

Build Vue and Webpack projects with npm.

Build on GUI

Add **Build with npm**, when **configuring build actions**. Set the parameters according to **Table 5-17**.

Table 5-17 Parameters for building with npm

Param eter	Description	
Action Name	 Assign a custom name to the build action. The name can contain: Letters, digits, hyphens (-), underscores (_), commas (,), semicolons (;), colons (:), periods (.), slashes (/), and parentheses. 1 to 128 characters. 	
Tool Version	Select a tool version that matches your current development environment.	
	For tool versions supported by CodeArts Build, see build tools and versions. If the current tools and versions do not meet your requirements, you can customize a build environment.	
Comm ands	Configure the npm commands, or use the default ones. If you have special build requirements, enter your custom build script in the text box. For more commands, see the Node.js official website .	

Param eter	Description
Contin ue After Failure	Specify whether to proceed after the current action fails by setting the parameter to either Yes or No .

Modify the code in the **BUILD** block in **Creating a YAML File for Your Code-based Build** by referring to the following sample code:

```
version: 2.0 # The value must be 2.0.
steps:
 BUILD:
  - npm:
     image: cloudbuild@nodejs8.11.2
     inputs:
      command: |
         export PATH=$PATH:~/.npm-global/bin
         #Set the cache directory.
         npm config set cache /npmcache
         npm config set prefix '~/.npm-global'
         npm config set registry https://repo.example.com/repository/npm/
         #For Node.js 18 or higher: Configure the .npmrc file. For earlier versions: use these command
lines.
         npm config set disturl https://repo.example.com/nodejs
         npm config set sass_binary_site https://repo.example.com/node-sass/
         npm config set phantomjs_cdnurl https://repo.example.com/phantomjs
         npm config set chromedriver_cdnurl https://repo.example.com/chromedriver
         npm config set operadriver_cdnurl https://repo.example.com/operadriver
         npm config set electron_mirror https://repo.example.com/electron/
         npm config set python_mirror https://repo.example.com/python
         npm install --verbose
         npm run build
     ignore_fail: true
```

Table 5-18 Parameters in the sample code

Para meter	Typ e	Description
image	Stri ng	 The image address can be in either of the following formats: Use cloudbuild@nodejs8.11.2. This address starts with cloudbuild and uses the at sign (@) as a separator, with the default image version provided by CodeArts Build following it. Use a complete SWR image path, for example, swr.example.example.com/codeci_test/demo:141d26c455abd6d7xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
comm and	Stri ng	Configure the npm commands. For more commands, see the Node.js official website .

Para meter	Typ e	Description
ignore _fail	Stri ng	Whether to proceed after the current action fails. • true: Yes • Empty: No

Helpful Links

Hit snags in npm builds? See npm Build FAQs.

5.3.4 Building with Gradle

Build a Java, Groovy, or Scala project with Gradle.

Build on GUI

Add **Build with Gradle**, when **configuring build actions**. Set the parameters according to **Table 5-19**.

Table 5-19 Parameters for building with Gradle

Param	Description
eter	Description
Action Name	 Assign a custom name to the build action. The name can contain: Letters, digits, hyphens (-), underscores (_), commas (,), semicolons (;), colons (:), periods (.), slashes (/), and parentheses. 1 to 128 characters.
Gradle	Select a tool version that matches your current development environment.
	For tool versions supported by CodeArts Build, see build tools and versions. If the current tools and versions do not meet your requirements, you can customize a build environment.
JDK	Select a tool version that matches your current development environment. For tool versions supported by CodeArts Build, see build tools and
	versions. If the current tools and versions do not meet your requirements, you can customize a build environment.
Comm ands	Configure the Gradle commands, or use the default ones. If you have special build requirements, enter your custom build script in the text box. For more commands, see the Gradle official website .
Contin ue After Failure	Specify whether to proceed after the current action fails by setting the parameter to either Yes or No .

Modify the code in the **BUILD** block in **Creating a YAML File for Your Code-based Build** by referring to the following sample code:

```
version: 2.0 # The value must be 2.0.

steps:

BUILD:
- gradle:
    inputs:
        gradle: 4.8
        jdk: 1.8
        command: |
            # Gradle Wrapper provided by CodeArts is used for cache acceleration.
            cp /cache/android/wrapper/gradle-wrapper.jar ./gradle/wrapper/gradle-wrapper.jar
        # Build an unsigned APK.
        /bin/bash ./gradlew build --init-script ./.codeci/.gradle/init_template.gradle -

Dorg.gradle.daemon=false -Dorg.gradle.internal.http.connectionTimeout=800000
        ignore_fail: true
```

Table 5-20 Parameters in the sample code

Para mete r	Typ e	Description
com mand	Stri ng	Configure the Gradle commands. For more commands, see the Gradle official website.
gradl e	Stri ng	Select a tool version that matches your current development environment. For tool versions supported by CodeArts Build, see build tools and versions. If the current tools and versions do not meet your requirements, you can customize a build environment.
jdk	Stri ng	Select a tool version that matches your current development environment. For tool versions supported by CodeArts Build, see build tools and versions. If the current tools and versions do not meet your requirements, you can customize a build environment.
ignor e_fail	Stri ng	Whether to proceed after the current action fails. • true: Yes • Empty: No

Helpful Links

Hit snags in Gradle builds? See Gradle Build FAQs.

5.3.5 Building with Yarn

Build a JavaScript project with Yarn.

Build on GUI

Add **Build with Yarn**, when **configuring build actions**. Set the parameters according to **Table 5-21**.

Table 5-21 Parameters for building with Yarn

Param eter	Description
Action Name	 Assign a custom name to the build action. The name can contain: Letters, digits, hyphens (-), underscores (_), commas (,), semicolons (;), colons (:), periods (.), slashes (/), and parentheses. 1 to 128 characters.
Tool Version	Select a tool version that matches your current development environment. For tool versions supported by CodeArts Build, see build tools and versions. If the current tools and versions do not meet your requirements, you can customize a build environment.
Comm ands	Configure the Yarn commands, or use the default ones. If you have special build requirements, enter your custom build script in the text box. For more commands, see the Yarn official website.
Contin ue After Failure	Specify whether to proceed after the current action fails by setting the parameter to either Yes or No .

Build with Code

Modify the code in the **BUILD** block in **Creating a YAML File for Your Code-based Build** by referring to the following sample code:

```
version: 2.0 # The value must be 2.0.
steps:
 BUILD:
  - yarn:
    inputs:
      command: |-
  #If the Node.js version is earlier than 18, the settings can be as follows:
                  npm config set cache-folder /yarncache
                  npm config set registry http://mirrors.tools.huawei.com/npm/
                  npm config set disturl http://mirrors.tools.huawei.com/nodejs
                  npm config set sass_binary_site http://mirrors.tools.huawei.com/node-sass/
                  npm config set phantomjs_cdnurl http://mirrors.tools.huawei.com/phantomjs
                  npm config set chromedriver_cdnurl http://mirrors.tools.huawei.com/chromedriver
                  npm config set operadriver_cdnurl http://mirrors.tools.huawei.com/operadriver
                  npm config set electron_mirror http://mirrors.tools.huawei.com/electron/
                  npm config set python_mirror http://mirrors.tools.huawei.com/python
                  #If the Node.js version is 18 or later, the settings can be as follows:
                  #npm config set registry http://mirrors.tools.huawei.com/npm/
                  npm config set prefix '~/.npm-global'
                  export PATH=$PATH:~/.npm-global/bin
                  #yarn add node-sass-import --verbose
                  yarn install --verbose
```

yarn run build tar -zcvf demo.tar.gz ./** ignore_fail: true

Table 5-22 Parameters in the sample code

Param eter	Туре	Description
comm and	Strin g	Configure the Yarn commands. For more commands, see the Yarn official website.
ignore_ fail	Strin g	Whether to proceed after the current action fails. • true: Yes • Empty: No

5.3.6 Building with Gulp

Build a frontend IDE with Gulp.

Build on GUI

Add **Build with Gulp**, when **configuring build actions**. Set the parameters according to **Table 5-23**.

Table 5-23 Parameters for building with Gulp

Parame ter	Description
Action Name	 Assign a custom name to the build action. The name can contain: Letters, digits, hyphens (-), underscores (_), commas (,), semicolons (;), colons (:), periods (.), slashes (/), and parentheses. 1 to 128 characters.
Tool Version	Select a tool version that matches your current development environment. For tool versions supported by CodeArts Build, see build tools and versions. If the current tools and versions do not meet your requirements, you can customize a build environment.
Comma nds	Configure the Gulp commands, or use the default ones. If you have special build requirements, enter your custom build script in the text box. For more commands, see the Gulp official website.
Continu e After Failure	Specify whether to proceed after the current action fails by setting the parameter to either Yes or No .

Modify the code in the **BUILD** block in **Creating a YAML File for Your Code-based Build** by referring to the following sample code:

```
version: 2.0 # The value must be 2.0.

steps:
BUILD:
- gulp:
inputs:

command: |-

export PATH=$PATH:~/.npm-global/bin

npm config set registry http://mirrors.tools.huawei.com/npm/

npm config set prefix '~/.npm-global'

#If node-sass needs to be installed

#npm config set sass_binary_site https://repo.huaweicloud.com/node-sass/

#npm install node-sass

#Load dependencies

npm install -verbose

gulp

ignore_fail: true
```

Table 5-24 Parameters in the sample code

Para meter	Typ e	Description
comm and	Strin g	Configure the Gulp commands. For more commands, see the Gulp official website.
ignore _fail	Strin g	Whether to proceed after the current action fails. • true: Yes • Empty: No

5.3.7 Building with Grunt

Build a JavaScript project with Grunt.

Build on GUI

Add **Build with Grunt**, when **configuring build actions**. Set the parameters according to **Table 5-25**.

Table 5-25 Parameters for building with Grunt

Paramet er	Description
Action Name	 Assign a custom name to the build action. The name can contain: Letters, digits, hyphens (-), underscores (_), commas (,), semicolons (;), colons (:), periods (.), slashes (/), and parentheses. 1 to 128 characters.

Paramet er	Description
Tool Version	Select a tool version that matches your current development environment.
	For tool versions supported by CodeArts Build, see build tools and versions. If the current tools and versions do not meet your requirements, you can customize a build environment.
Comman ds	Configure the Grunt commands, or use the default ones. If you have special build requirements, enter your custom build script in the text box. For more commands, see the Grunt official website .
Continue After Failure	Specify whether to proceed after the current action fails by setting the parameter to either Yes or No .

Modify the code in the **BUILD** block in **Creating a YAML File for Your Code-based Build** by referring to the following sample code:

Table 5-26 Parameters in the sample code

Para meter	Туре	Description
comm and	String	Configure the Grunt commands. For more commands, see the Grunt official website .
ignore _fail	String	Whether to proceed after the current action fails. • true: Yes • Empty: No

5.3.8 Building with Mono

Use Mono for MSBuild and .NET builds.

Build on GUI

Add **mono**, when **configuring build actions**. Set the parameters according to **Table 5-27**.

Table 5-27 Parameters for building with Mono

Paramete r	Description	
Action Name	 Assign a custom name to the build action. The name can contain: Letters, digits, hyphens (-), underscores (_), commas (,), semicolons (;), colons (:), periods (.), slashes (/), and parentheses. 1 to 128 characters. 	
Tool Version	Select a tool version that matches your current development environment. For tool versions supported by CodeArts Build, see build tools and versions. If the current tools and versions do not meet your requirements, you can customize a build environment.	
Comman ds	Configure the Mono commands, or use the default ones. If you have special build requirements, enter your custom build script in the text box.	
Continue After Failure	Specify whether to proceed after the current action fails by setting the parameter to either Yes or No .	

Build with Code

Modify the code in the **BUILD** block in **Creating a YAML File for Your Code-based Build** by referring to the following sample code:

```
version: 2.0 # The value must be 2.0.

steps:
BUILD:
- mono:
inputs:
command: |
nuget sources Disable -Name 'nuget.org'
nuget sources add -Name 'xxcloud' -Source 'https://repo.xxcloud.com/repository/nuget/v3/
index.json'
nuget restore
msbuild /p:OutputPath=../buildResult/Release/bin
zip -rq ./archive.zip ./buildResult/Release/bin/*
ignore_fail: true
```

Table 5-28 Parameters in the sample code

Para mete r	Туре	Description
com mand	Strin g	Configure the Mono commands.
ignor e_fail	Strin g	Whether to proceed after the current action fails. • true: Yes • Empty: No

5.3.9 Building in PHP

Use PHP to build within an installation and packaging environment that includes the necessary PHP code libraries for the project.

Build on GUI

Add **Build in PHP**, when **configuring build actions**. Set the parameters according to **Table 5-29**.

Table 5-29 Parameters for building in PHP

Param eter	Description
Action Name	 Assign a custom name to the build action. The name can contain: Letters, digits, hyphens (-), underscores (_), commas (,), semicolons (;), colons (:), periods (.), slashes (/), and parentheses. 1 to 128 characters.
Tool Version	Select a tool version that matches your current development environment. For tool versions supported by CodeArts Build, see build tools and versions. If the current tools and versions do not meet your requirements, you can customize a build environment.
Comma nds	Configure the PHP commands, or use the default ones. If you have special build requirements, enter your custom build script in the text box. For more commands, see the PHP official website.
Continu e After Failure	Specify whether to proceed after the current action fails by setting the parameter to either Yes or No .

Modify the code in the **BUILD** block in **Creating a YAML File for Your Code-based Build** by referring to the following sample code:

Table 5-30 Parameters in the sample code

Para mete r	Туре	Description
com mand	String	Configure the PHP commands. For more commands, see the PHP official website.
ignor e_fail	String	Whether to proceed after the current action fails. • true: Yes • Empty: No

5.3.10 Building with Setuptools

Use Setuptools to package Python applications.

Prerequisites

When using Setuptools to pack the code, ensure that the **setup.py** file exists in the root directory of the code. For details on how to write the setup file, see the **official instructions of Python**.

Build on GUI

Add **Build with Setuptools**, when **configuring build actions**. Set the parameters according to **Table 5-31**.

Table 5-31 Parameters for building with Setuptools

Param eter	Description	
Action Name	 Assign a custom name to the build action. The name can contain: Letters, digits, hyphens (-), underscores (_), commas (,), semicolons (;), colons (:), periods (.), slashes (/), and parentheses. 1 to 128 characters. 	
Tool Version	Select a tool version that matches your current development environment. For tool versions supported by CodeArts Build, see build tools and versions. If the current tools and versions do not meet your requirements, you can customize a build environment.	
Comm ands	 Configure the pack commands. You can use the default commands to pack the file into an .egg file. For Python 2.7 or later, it is advised to use python setup.py sdist bdist_wheel to pack the source code package and .whl installation package for pip installation. For more commands, see the Setuptools official website. 	
Contin ue After Failure	Specify whether to proceed after the current action fails by setting the parameter to either Yes or No .	

```
version: 2.0 # The value must be 2.0.

steps:

BUILD:

- python:

name: Build with Setuptools

image: cloudbuild@python3.6

inputs:

command: |

pip config set global.index-url https://pypi.org/simple

pip config set global.trusted-host repo.xxcloud.com

python setup.py bdist_egg

ignore_fail: true
```

Table 5-32 Parameters in the sample code

Para mete r	Туре	Description
name	String	Optional. Assign a custom name to the build action. The name can contain: • Letters, digits, hyphens (-), underscores (_), commas (,), semicolons (;), colons (:), periods (.), slashes (/), and parentheses. • 1 to 128 characters.
imag e	String	Optional. Enter the image version, which should include the fixed part cloudbuild@ and the supported Python version following it. For tool versions supported by CodeArts Build, see build tools and versions. If the current tools and versions do not meet your requirements, you can customize a build environment. The default value is cloudbuild@python3.6.
com mand	String	 Configure the pack commands. You can use the default commands to pack the file into an .egg file. For Python 2.7 or later, it is advised to use python setup.py sdist bdist_wheel to pack the source code package and .whl installation package for pip installation. For more commands, see the Setuptools official website.
ignor e_fail	String	Whether to proceed after the current action fails. • true: Yes • Empty: No

5.3.11 Building with PyInstaller

Use PyInstaller to package Python scripts into standalone executables.

Build on GUI

Add **Build with PyInstaller**, when **configuring build actions**. Set the parameters according to **Table 5-33**.

Table 5-33 Parameters for building with PyInstaller

Param eter	Description
Action Name	 Assign a custom name to the build action. The name can contain: Letters, digits, hyphens (-), underscores (_), commas (,), semicolons (;), colons (:), periods (.), slashes (/), and parentheses. 1 to 128 characters.
Tool Versio n	Select a tool version that matches your current development environment. For tool versions supported by CodeArts Build, see build tools and versions. If the current tools and versions do not meet your requirements, you can customize a build environment.
Comm ands	Configure the build and packaging commands or use the default commands, which will package the project into an executable. For more commands, see the PyInstaller official website .
Contin ue After Failure	Specify whether to proceed after the current action fails by setting the parameter to either Yes or No .

```
version: 2.0 # The value must be 2.0.

steps:
BUILD:
- python:
    name: Build with PyInstaller
    image: cloudbuild@python3.6
    inputs:
    command: |
        pip config set global.index-url https://pypi.org/simple
        pip config set global.trusted-host repo.xxcloud.com
        # Create a single executable file in the dist directory with -F.
        # For command details, see https://pyinstaller.readthedocs.io/en/stable/usage.html.
        pyinstaller -F *.py
        ignore_fail: true
```

Table 5-34 Parameters in the sample code

Para mete r	Туре	Description	
name	Strin	Optional.	
	g	Assign a custom name to the build action. The name can contain:	
		• Letters, digits, hyphens (-), underscores (_), commas (,), semicolons (;), colons (:), periods (.), slashes (/), and parentheses.	
		• 1 to 128 characters.	
imag	Strin	Optional.	
е	g	Enter the image version, which should include the fixed part cloudbuild@ and the supported Python version following it.	
		For tool versions supported by CodeArts Build, see build tools and versions . If the current tools and versions do not meet your requirements, you can customize a build environment .	
		The default value is cloudbuild@python3.6 .	
com mand	Strin g	Configure the build and packaging commands or use the default commands, which will package the project into an executable. For more commands, see the PyInstaller official website .	
ignor	Strin g	Whether to proceed after the current action fails.	
e_fail		• true: Yes	
		Empty: No	

5.3.12 Running Shell Commands

You can use the action **Run Shell Commands** to create and run a build task. You can also use this action in conjunction with other build tools. For instance, in a Maven build, you can add the **Run Shell Commands** action to generate the necessary files for the subsequent build process.

Build on GUI

Add **Run Shell Commands**, when **configuring build actions**. Set the parameters according to **Table 5-35**.

Description
 Assign a custom name to the build action. The name can contain: Letters, digits, hyphens (-), underscores (_), commas (,), semicolons (;), colons (:), periods (.), slashes (/), and parentheses. 1 to 128 characters.
Select a tool version that matches your current development environment. For tool versions supported by CodeArts Build, see build tools and versions. If the current tools and versions do not meet your requirements, you can customize a build environment.
Enter the shell commands for your build.

Specify whether to proceed after the current action fails by setting

Table 5-35 Parameters for running shell commands

the parameter to either **Yes** or **No**.

Build with Code

Modify the code in the **PRE_BUILD** block in **Creating a YAML File for Your Code-based Build** by referring to the following sample code:

```
version: 2.0 # The value must be 2.0.
steps:
PRE_BUILD:
- sh:
inputs:
command: echo ${a}
ignore_fail: true
```

Table 5-36 Parameters in the sample code

Continu

e After Failure

Param eter	Туре	Description
comma nd	String	Enter the shell commands for your build.
ignore_ fail	String	Whether to proceed after the current action fails. • true: Yes • Empty: No

5.3.13 Building with GNU Arm

Compile and build software for Arm processors.

Build on GUI

Add **Build with GNU Arm**, when **configuring build actions**. Set the parameters according to **Table 5-37**.

Table 5-37 Parameters for building with GNU Arm

Parame ter	Description	
Action Name	 Assign a custom name to the build action. The name can contain: Letters, digits, hyphens (-), underscores (_), commas (,), semicolons (;), colons (:), periods (.), slashes (/), and parentheses. 1 to 128 characters. 	
Tool Version	Select a tool version that matches your current development environment. For tool versions supported by CodeArts Build, see build tools and versions. If the current tools and versions do not meet your requirements, you can customize a build environment.	
Comma	Configure the GNU Arm build commands, or use the default make command. If Makefile is not in the root directory of the code, run the cd command to access the correct directory and then run the make command. If you do not want to run the make command, you can refer to the build commands provided by the following images: Optional: Image gnuarm201405: Run the arm-none-linux-gnueabi-gcc command as follows: arm-none-linux-gnueabi-gcc -o main main.c Image gnuarm-linux-gcc-4.4.3: Run the arm-linux-gcc command as follows: arm-linux-gcc -o main main.c Image gnuarm-7-2018-q2-update: Run the arm-none-eabi-gcc command as follows: arm-none-eabi-gccspecs=nosys.specs -o main main.c NOTE For details about how to write the GNU makefile in Linux, see the official website. Makefile contains only line comment tags (#). If you want to use or output the number sign (#), escape the number sign, for example, using \#.	
Continu e After Failure	Specify whether to proceed after the current action fails by setting the parameter to either Yes or No .	

Build with Code

```
version: 2.0 # The value must be 2.0.
steps:
BUILD:
- gnu_arm:
inputs:
command: make
ignore_fail: true
```

Table 5-38 Parameters in the sample code

Para met er	Туре	Description			
com man d	Strin g	 Configure the command for building with GNU Arm. If Makefile is not in the root directory of the code, run the cd command to access the correct directory and then run the make command. If you do not want to run the make command, you can refer to the build commands provided by the following images: Optional: Image gnuarm201405: Run the arm-none-linux-gnueabi-gcc command as follows: arm-none-linux-gnueabi-gcc -o main main.c Image gnuarm-linux-gcc-4.4.3: Run the arm-linux-gcc command as follows: arm-linux-gcc -o main main.c Image gnuarm-7-2018-q2-update: Run the arm-none-eabi-gcc command as follows: arm-none-eabi-gccspecs=nosys.specs -o main main.c NOTE For details about how to write the GNU makefile in Linux, see the official website. Makefile contains only line comment tags (#). If you want to use or output the number sign (#), escape the number sign, for example, using \#. 			
ignor e_fail	Strin g	Whether to proceed after the current action fails. • true: Yes • Empty: No			

5.3.14 Building with CMake

Build a cross-platform project with CMake.

Build on GUI

Add **Build with CMake**, when **configuring build actions**. Set the parameters according to **Table 5-39**.

Table 5-39 Parameters for building with CMake

Parame ter	Description
Action Name	 Assign a custom name to the build action. The name can contain: Letters, digits, hyphens (-), underscores (_), commas (,), semicolons (;), colons (:), periods (.), slashes (/), and parentheses. 1 to 128 characters.
Tool Version	Select a tool version that matches your current development environment. For tool versions supported by CodeArts Build, see build tools and versions. If the current tools and versions do not meet your requirements, you can customize a build environment.
Comma nds	Configure the CMake commands, or use the default ones. If you have special build requirements, enter your custom build script in the text box. For more commands, see the CMake official website.
Continu e After Failure	Specify whether to proceed after the current action fails by setting the parameter to either Yes or No .

```
version: 2.0 # The value must be 2.0.

steps:

BUILD:

- cmake:

inputs:

command: |

# Create the build directory and switch to the build directory.

mkdir build && cd build

# Generate makefiles for the Unix platform and perform the build.

cmake -G 'Unix Makefiles' ../ && make -j

ignore_fail: true
```

Table 5-40 Parameters in the sample code

Para mete r	Туре	Description
com mand	Strin g	Configure the CMake commands. For more commands, see the CMake official website.
ignor e_fail	Strin g	Whether to proceed after the current action fails. • true: Yes • Empty: No

5.3.15 Building with Ant

Build, test, and deploy a Java project using Ant.

Build on GUI

Add **Build with Ant**, when **configuring build actions**. Set the parameters according to **Table 5-41**.

Table 5-41 Parameters for building with Ant

Paramet er	Description
Action Name	 Assign a custom name to the build action. The name can contain: Letters, digits, hyphens (-), underscores (_), commas (,), semicolons (;), colons (:), periods (.), slashes (/), and parentheses. 1 to 128 characters.
Tool Version	Select a tool version that matches your current development environment. For tool versions supported by CodeArts Build, see build tools and versions. If the current tools and versions do not meet your requirements, you can customize a build environment.
Comma nds	Configure the Ant build commands, or use the default ones. If you have special build requirements, enter your custom build script in the text box. For more commands, see the Ant official website .
Continue After Failure	Specify whether to proceed after the current action fails by setting the parameter to either Yes or No .

Build with Code

```
version: 2.0 # The value must be 2.0.
steps:
BUILD:
- ant:
inputs:
command: ant -f build.xml
ignore_fail: true
```

Table 5-42 Parameters in the sample code

Paramet er	Туре	Description
comman d	String	Configure the Ant build commands. For more commands, see the Ant official website .

Paramet er	Туре	Description
ignore_f ail	String	Whether to proceed after the current action fails.true: YesEmpty: No

5.3.16 Building with Kotlin

Build, test, and deploy a project with Kotlin.

□ NOTE

Currently, this action is available only in LA-Sao Paulo1.

Build on GUI

Add **Build with Kotlin**, when **configuring build actions**. Set the parameters according to **Table 5-43**.

Table 5-43 Parameters for building with Kotlin

Parame ter	Description
Action Name	 Assign a custom name to the build action. The name can contain: Letters, digits, hyphens (-), underscores (_), commas (,), semicolons (;), colons (:), periods (.), slashes (/), and parentheses. 1 to 128 characters.
Tool Version	Select a tool version that matches your current development environment. For tool versions supported by CodeArts Build, see build tools and versions. If the current tools and versions do not meet your requirements, you can customize a build environment.
Comma nds	Configure the Kotlin build commands, or use the default ones. If you have special build requirements, enter your custom build script in the text box. For more commands, see the Kotlin official website.
Continu e After Failure	Specify whether to proceed after the current action fails by setting the parameter to either Yes or No .

Build with Code

Modify the code in the **BUILD** block in **Creating a YAML File for Your Code-based Build** by referring to the following sample code:

version: 2.0 # The value must be 2.0. steps:

BUILD:
- kotlin:
inputs:
command: gradle build
ignore_fail: true

Table 5-44 Parameters in the sample code

Para meter	Туре	Description
comm and	String	Configure the Kotlin build commands. For more commands, see the Kotlin official website.
ignore _fail	String	Whether to proceed after the current action fails. • true: Yes • Empty: No

5.3.17 Building with Go

In this action, you build a project with the Go language. This involves compiling the source code to produce executables, managing project dependencies, and customizing the build process.

Build on GUI

Add **Build with Go**, when **configuring build actions**. Set the parameters according to **Table 5-45**.

Table 5-45 Parameters for building with Go

Param eter	Description		
Action Name	 Assign a custom name to the build action. The name can contain: Letters, digits, hyphens (-), underscores (_), commas (,), semicolons (;), colons (:), periods (.), slashes (/), and parentheses. 1 to 128 characters. 		
Tool Version	Select a tool version that matches your current development environment. For tool versions supported by CodeArts Build, see build tools and versions. If the current tools and versions do not meet your requirements, you can customize a build environment.		
Comma nds	Configure the Go project build command, or use the default ones. If you have special build requirements, enter your custom build script in the text box. For more commands, see the Go official website.		
Continu e After Failure	Specify whether to proceed after the current action fails by setting the parameter to either Yes or No .		

Modify the code in the **BUILD** block in **Creating a YAML File for Your Code-based Build** by referring to the following sample code:

```
version: 2.0 # The value must be 2.0.

steps:

BUILD:

go:
inputs:
command: |
export GO15VENDOREXPERIMENT=1
export GOPROXY=https://goproxy.cn
mkdir -p $GOPATH/src/example.com/demo/
cp -rf . $GOPATH/src/example.com/demo/
go install example.com/demo
cp -rf $GOPATH/bin/ ./bin
ignore_fail: true
```

Table 5-46 Parameters in the sample code

Param eter	Туре	Description
comm and	Strin g	Configure the command for building a Go project. For more commands, see the Go official website .
ignore_ fail	Strin g	Whether to proceed after the current action fails. • true: Yes • Empty: No

5.3.18 Building Android App with Ionic

In this action, you build an Ionic Android app, which is a mobile app that works across multiple platforms. This action allows you to quickly develop mobile apps, mobile web pages, hybrid apps, and web pages.

The project contains the project compilation description files such as **ionic.config.json**, **package.json**, and **angular.json**.

Build on GUI

Add **Build Android App with Ionic**, when **configuring build actions**. Set the parameters according to **Table 5-47**.

Table 5-47 Parameters for building an Android app with Ionic

Param eter	Description
Action Name	 Assign a custom name to the build action. The name can contain: Letters, digits, hyphens (-), underscores (_), commas (,), semicolons (;), colons (:), periods (.), slashes (/), and parentheses. 1 to 128 characters.
Gradle	Select a Gradle version that matches your current development environment. For tool versions supported by CodeArts Build, see build tools and versions. If the current tools and versions do not meet your requirements, you can customize a build environment.
JDK	Select a JDK version that matches your current development environment. For tool versions supported by CodeArts Build, see build tools and versions. If the current tools and versions do not meet your requirements, you can customize a build environment.
NDK	Select an NDK version that matches your current development environment. For tool versions supported by CodeArts Build, see build tools and versions. If the current tools and versions do not meet your requirements, you can customize a build environment.
Comm ands	Configure the packaging script in the command box. For more commands, see the Ionic official website .
Contin ue After Failure	Specify whether to proceed after the current action fails by setting the parameter to either Yes or No .

```
version: 2.0 # The value must be 2.0.
steps:
BUILD:
   - ionic_android_app:
   inputs:
        gradle: '4.8'
        jdk: '33'
        ndk: '17'
        command: ./instrumented.apk
        ignore_fail: true
```

Table 5-48 Parameters in the sample code

Para mete r	Туре	Description
gradl e	String	Select a Gradle version that matches your current development environment.
		For tool versions supported by CodeArts Build, see build tools and versions . If the current tools and versions do not meet your requirements, you can customize a build environment .
jdk	String	Select a JDK version that matches your current development environment.
		For tool versions supported by CodeArts Build, see build tools and versions. If the current tools and versions do not meet your requirements, you can customize a build environment.
ndk	String	Select an NDK version that matches your current development environment.
		For tool versions supported by CodeArts Build, see build tools and versions. If the current tools and versions do not meet your requirements, you can customize a build environment.
com mand	String	Configure the packaging script in the command box. For more commands, see the Ionic official website .
ignor e_fail	String	Whether to proceed after the current action fails. • true: Yes
		• Empty: No

5.3.19 Building an Android Quick App

Run the npm configuration command to build an Android quick app.

Build on GUI

Add **Build Android Quick App**, when **configuring build actions**. Set the parameters according to **Table 5-49**.

Table 5-49 Parameters for building an Android quick app

Param eter	Description
Action Name	 Assign a custom name to the build action. The name can contain: Letters, digits, hyphens (-), underscores (_), commas (,), semicolons (;), colons (:), periods (.), slashes (/), and parentheses. 1 to 128 characters.

Param eter	Description
Tool Version	Select a tool version that matches your current development environment.
	For tool versions supported by CodeArts Build, see build tools and versions. If the current tools and versions do not meet your requirements, you can customize a build environment.
Comm ands	Configure npm commands. For more commands, see the Node.js official website.
Contin ue After Failure	Specify whether to proceed after the current action fails by setting the parameter to either Yes or No .

Modify the code in the **BUILD** block in **Creating a YAML File for Your Code-based Build** by referring to the following sample code:

```
version: 2.0 # The value must be 2.0.

steps:

BUILD:

- quick_app:
    inputs:
        command: |-
        npm config set registry http://7.223.219.40/npm/
        # Load dependencies
        npm install --verbose
        # Default build
        npm run build
        ignore_fail: true
```

Table 5-50 Parameters in the sample code

Para meter	Туре	Description
comm and	Strin g	Configure npm commands. For more commands, see the Node.js official website .
ignore _fail	Strin g	Whether to proceed after the current action fails. • true: Yes • Empty: No

5.3.20 Building with sbt

Build a Scala or Java project with sbt.

Build on GUI

Add **Build with sbt**, when **configuring build actions**. Set the parameters according to **Table 5-51**.

Table 5-51 Parameters for building with sbt

Parame ter	Description
Action Name	 Assign a custom name to the build action. The name can contain: Letters, digits, hyphens (-), underscores (_), commas (,), semicolons (;), colons (:), periods (.), slashes (/), and parentheses. 1 to 128 characters.
Tool Version	Only the default version sbt1.3.2-jdk1.8 is supported currently.
Comma nds	Configure the sbt commands, or use the default ones. If you have special build requirements, enter your custom build script in the text box. For more commands, see the sbt official website.
Continu e After Failure	Specify whether to proceed after the current action fails by setting the parameter to either Yes or No .

Build with Code

```
version: 2.0 # The value must be 2.0.
steps:
BUILD:
- sbt:
inputs:
command: |
sbt package
ignore_fail: true
```

Table 5-52 Parameters in the sample code

Para meter	Туре	Description
comm and	Strin g	Configure the sbt commands. For more commands, see the sbt official website .
ignore _fail	Strin g	 Whether to proceed after the current action fails. true: Yes Empty: No

5.3.21 Building with Grails

Build a web application with Grails.

Build on GUI

Add **Build with Grails**, when **configuring build actions**. Set the parameters according to **Table 5-53**.

Table 5-53 Parameters for building with Grails

Parame ter	Description		
Action Name	Assign a custom name to the build action. The name can contain: • Letters, digits, hyphens (-), underscores (_), commas (,), semicolons (;), colons (:), periods (.), slashes (/), and parentheses. • 1 to 128 characters.		
Tool Version	Select a tool version that matches your current development environment. For tool versions supported by CodeArts Build, see build tools and versions. If the current tools and versions do not meet your requirements, you can customize a build environment.		
Comma nds	Configure the Grails commands, or use the default ones. If you have special build requirements, enter your custom build script in the text box. For more commands, see the Grails official website .		
Continu e After Failure	Specify whether to proceed after the current action fails by setting the parameter to either Yes or No .		

Build with Code

```
version: 2.0 # The value must be 2.0.
steps:
BUILD:
- grails:
inputs:
command: grails war
ignore_fail: true
```

Table 5-54 Parameters in the sample code

Para meter	Туре	Description
comm and	String	Configure the Grails commands. For more commands, see the Grails official website .

Para meter	Туре	Description
ignore	String	Whether to proceed after the current action fails.
_fail		• true: Yes
		Empty: No

5.3.22 Building with Bazel

Use Bazel to compile and build.

Build on GUI

Add **Build with Bazel**, when **configuring build actions**. Set the parameters according to **Table 5-55**.

Table 5-55 Parameters for building with Bazel

Paramet er	Description		
Action Name	 Assign a custom name to the build action. The name can contain: Letters, digits, hyphens (-), underscores (_), commas (,), semicolons (;), colons (:), periods (.), slashes (/), and parentheses. 1 to 128 characters. 		
Tool Version	Select a tool version that matches your current development environment.		
	For tool versions supported by CodeArts Build, see build tools and versions . If the current tools and versions do not meet your requirements, you can customize a build environment .		
Comman ds	Configure the Bazel commands, or use the default ones. If you have special build requirements, enter your custom build script in the text box.		
Continue After Failure	Specify whether to proceed after the current action fails by setting the parameter to either Yes or No .		

Build with Code

```
version: 2.0 # The value must be 2.0.
steps:
BUILD:
- bazel:
inputs:
```

```
command: |
cd java-maven
bazel build //:java-maven_deploy.jar
mkdir build_out
cp -r bazel-bin/* build_out/
ignore_fail: true
```

Table 5-56 Parameters in the sample code

Para meter	Туре	Description
comm and	Strin g	Configure the Bazel commands.
ignore Strin Whether to proce _fail g • true: Yes • Empty: No		

5.3.23 Building with Flutter

Build Android applications with Flutter.

Build on GUI

Add **Build with Flutter**, when **configuring build actions**. Set the parameters according to **Table 5-57**.

Table 5-57 Parameters for building with Flutter

Param eter	Description	
Action Name	 Assign a custom name to the build action. The name can contain: Letters, digits, hyphens (-), underscores (_), commas (,), semicolons (;), colons (:), periods (.), slashes (/), and parentheses. 1 to 128 characters. 	
Flutter	Select a Flutter version that matches your current development environment. For tool versions supported by CodeArts Build, see build tools and versions. If the current tools and versions do not meet your requirements, you can customize a build environment.	
JDK	Select a JDK version that matches your current development environment. For tool versions supported by CodeArts Build, see build tools and versions. If the current tools and versions do not meet your requirements, you can customize a build environment.	

Param eter	Description
NDK	Select an NDK version that matches your current development environment.
	For tool versions supported by CodeArts Build, see build tools and versions. If the current tools and versions do not meet your requirements, you can customize a build environment.
Comma nds	Configure the Flutter commands, or use the default ones. If you have special build requirements, enter your custom build script in the text box. For more commands, see the Flutter official website .
Continu e After Failure	Specify whether to proceed after the current action fails by setting the parameter to either Yes or No .

```
version: 2.0 # The value must be 2.0.
steps:
BUILD:
- flutter:
inputs:
    flutter: '1.17.5'
    jdk: '3333'
    ndk: '23.1.7779620'
    command: ./instrumented.apk
    ignore_fail: true
```

Table 5-58 Parameters in the sample code

Para met er	Туре	Description	
flutt er	String	Select a Flutter version that matches your current development environment.	
		For tool versions supported by CodeArts Build, see build tools and versions . If the current tools and versions do not meet your requirements, you can customize a build environment .	
jdk	String	Select a JDK version that matches your current development environment.	
		For tool versions supported by CodeArts Build, see build tools and versions . If the current tools and versions do not meet your requirements, you can customize a build environment .	

Para met er	Туре	Description	
ndk	String	Select an NDK version that matches your current development environment.	
		For tool versions supported by CodeArts Build, see build tools and versions . If the current tools and versions do not meet your requirements, you can customize a build environment .	
com man d	String	Configure the Flutter commands. For more commands, see the Flutter official website.	
igno re_fa il	String	Whether to proceed after the current action fails. • true: Yes • Empty: No	

5.3.24 Building with HarmonyOS

Build, test, and deploy a project with Hvigor.

Constraints

You are advised to use executors with 4 vCPUs and 16 GB memory (at least 8 GB) or higher for Hvigor builds.

Build on GUI

Add **Build with HarmonyOS**, when **configuring build actions**. Set the parameters according to **Table 5-59**.

Table 5-59 Parameters for building with HarmonyOS

Param eter	Description	
Action Name	 Assign a custom name to the build action. The name can contain: Letters, digits, hyphens (-), underscores (_), commas (,), semicolons (;), colons (:), periods (.), slashes (/), and parentheses. 1 to 128 characters. 	
Tool Version	The default version is HarmonyOS-5.0.0-API12 . Currently, HarmonyOS-5.0.0-API12 and HarmonyOS-API9 are supported.	
Comma nds	Configure commands, or use the default ones. If you have special build requirements, enter your custom build script in the text box.	

Param eter	Description
Contin ue After Failure	Specify whether to proceed after the current action fails by setting the parameter to either Yes or No .

Modify the code in the **BUILD** block in **Creating a YAML File for Your Code-based Build** by referring to the following sample code:

```
version: 2.0 # The value must be 2.0.
steps:
BUILD:
- harmonyos:
    name: "HarmonyOS Build"
    inputs:
    command: |
        npm config set strict-ssl false
        npm config set registry=https://repo.huaweicloud.com/repository/npm/
        npm config set @ohos:registry=https://repo.harmonyos.com/npm/
        chmod +x hvigorw
        ./hvigorw clean assembleApp --no-daemon
    ignore_fail: true
```

Table 5-60 Parameters in the sample code

Paramet er	Туре	Description
comman d	String	Configure the HarmonyOS commands.
ignore_fa il	String	Whether to proceed after the current action fails. • true: Yes • Empty: No

5.3.25 Running Docker Commands to Operate Images

Run Docker commands to perform image operations, such as login, push, and download.

Build on GUI

Add **Run Docker Commands**, when **configuring build actions**. Set the parameters according to **Table 5-61**.

Table 5-61 Parameters for running docker commands

Parame ter	Description		
Action Name	 Assign a custom name to the build action. The name can contain: Letters, digits, hyphens (-), underscores (_), commas (,), semicolons (;), colons (:), periods (.), slashes (/), and parentheses. 1 to 128 characters. 		
Tool Version	Select a tool version that matches your current development environment. For tool versions supported by CodeArts Build, see build tools and versions. If the current tools and versions do not meet your requirements, you can customize a build environment.		
Comma nds	Click Add to add a command, and configure it as required. For details about the Docker commands supported by CodeArts Build, see Docker Commands Supported by CodeArts Build . You can drag and drop the commands into the desired execution sequence.		
Continu e After Failure	Specify whether to proceed after the current action fails by setting the parameter to either Yes or No .		

```
version: 2.0 # The value must be 2.0.
steps:
BUILD:
- docker:
    inputs:
        command: |
        docker pull swr.xx-xxxxx-x.myxxcloud.com/codeci/dockerindocker:dockerindocker18.09-1.3.2
        ignore_fail: true
```

Table 5-62 Parameters in the sample code

Param eter	Туре	Description
comm and	String	Each command takes up one line. For details about the supported Docker commands, see Docker Commands Supported by CodeArts Build.
ignore _fail	String	Whether to proceed after the current action fails. • true: Yes
		Empty: No

Docker Commands Supported by CodeArts Build

• docker login: Log in to the Docker repository.

Usage: docker login [options] [server]

The following table describes how to set **options**. **server** indicates the Docker repository address.

Option	Short Form	Description
 passwor d	-р	Password for logging in to the repository.
 usernam e	-u	Username for logging in to the repository.
 passwor d	-stdin	Password obtained from stdin

Example: docker login -u jack -p 12345 mydocker-registry.com

In this example, user **jack** remotely logs in to the **mydocker-registry.com** repository using password **12345**.

Advanced Usage

To read a password from the file, run cat ~/my_password.txt | docker login --username jack --password-stdin.



docker build: Build an image from a Dockerfile or context. The context can
be a local path (Path) where the build is executed, a remote URL (such as a
Git repository, tarball, or text file), or a hyphen (-).

Usage: docker build [options] Path | URL | -

The following table describes how to set **options**. **Path/URL/-** indicates the context source.

Option	Short Form	Description
file	-f	Dockerfile path. The default value is ./Dockerfile.
tag	-t	In the format of " <i>Image name</i> : <i>Tag</i> "

Example: docker build -t mydocker-registry.com/org/alpine:1.0 -f ./ Dockerfile .

In this example, this command uses the Dockerfile with the tag **mydocker-registry.com/org/alpine:1.0** in the current directory to create an image.

• docker push: Push an image to a specified registry.

Usage: docker push [options] name[:tag]

Example: docker push mydocker-registry.com/org/alpine:1.0

In this example, this command pushes tag 1.0 of the **mydocker-registry.com/org/alpine** image to the remote repository.

4 push > swr >1.my com/codeci_gray/test1.0 + [

• docker pull: Pull an image from a registry.

Usage: docker pull [options] name[:tag|@digest]

The following table describes how to set **options**.

Option	Short Form	Description
all-tags	-a	Download all tagged images.

Example: docker pull mydocker-registry.com/org/alpine:1.0

In this example, this command pulls the **mydocker-registry.com/org/alpine** image whose tag is **1.0** from the remote repository.

• **docker tag**: Modify the tag of the image.

Usage: docker tag source_image[:tag] target_image[:tag]

source_image[:tag] indicates the image whose tag needs to be modified, and **target_image[:tag]** indicates the target image with a new tag.

Example: docker tag mydocker-registry.com/org/alpine:1.0 mydocker-registry/neworg/alpine:2.0

In this example, this command changes the tag of the **mydocker-registry.com/org/alpine** image from **1.0** to **2.0**.

• **docker save**: Save one or more images to a .tar file (streamed to the standard output by default).

Usage: docker save [options] image [image...]

The following table describes how to set **options**.

Option	Short Form	Description
output	-0	Write to a file instead of using standard output.

Example: docker save -o alpine.tar mydocker-registry.com/org/alpine:1.0 mydocker-registry.com/org/alpine:2.0

In this example, this command packages the **mydocker-registry.com/org/ alpine:1.0** and **mydocker-registry.com/org/alpine:2.0** images into **alpine.tar**.

• docker logout: Log out of a Docker repository.

Usage: docker logout [server]

Example: docker logout mydocker-registry.com

This example indicates that the image repository whose address is **mydocker-registry.com** is logged out.

5.3.26 Generating a Unit Test Report

The **Unit Test Report** action generates a visualized report by parsing the unit test result file that you have generated.

Currently, this action is only available in the **ME-Riyadh** region.

Prerequisites

Before running the **Unit Test Report** action, ensure that the test result file has been generated and the file framework is supported by CodeArts Build.

Build on GUI

Add **Unit Test Report**, when **configuring build actions**. Set the parameters according to **Table 5-63**.

Table 5-63 Parameters for generating the unit test report

Param eter	Description		
Action Name	 Assign a custom name to the build action. The name can contain: Letters, digits, hyphens (-), underscores (_), commas (,), semicolons (;), colons (:), periods (.), slashes (/), and parentheses. 1 to 128 characters. 		
Tool Version	Select a tool version that matches your current development environment. For tool versions supported by CodeArts Build, see build tools and versions. If the current tools and versions do not meet your requirements, you can customize a build environment.		
Contin ue After Failure	Specify whether to proceed after the current action fails by setting the parameter to either Yes or No .		
Unit Test	 Report Type: Select a unit test framework. Currently, only JUnit is supported. Test Report: Enter a relative path to specify the location relative to the project root directory, for example, target/surefire-reports/TEST*.xml. Currently, only the test report in the .xml format is supported. Print Unit Test Results: Specify whether to process unit test coverage report. If you select Yes, ensure that your project can use jacoco-maven-plugin to generate unit coverage reports. 		

5.3.27 Building an Image and Pushing It to SWR

Upload the build artifact to SWR.

If necessary dependency packages and tools are missing, you can create a custom image from a Dockerfile and push the image to SWR. For details about how to use the pushed image, see **Using a Custom Build Environment**.

Preparations

- You have created an organization in SWR. For details about organization restrictions, see notes and constraints of SWR.
- If you want to push the created image to SWR of other Huawei Cloud users, perform the following operations.
 - a. Access the CodeArts Build homepage from the project list.
 - b. In the navigation pane, choose **Settings** > **General** > **Service Endpoints**.
 - c. Select **IAM user** from the **Create Endpoint** drop-down list box. In the displayed dialog box, enter the following information and click **Confirm**.
 - Service Endpoint Name: Assign a custom name to the endpoint. Enter a maximum of 256 characters, including letters, digits, hyphens (-), underscores (_), periods (.), and spaces.
 - Access key ID (AK) and secret access key (SK) are used like passwords to authenticate users who make API requests.
 On the CodeArts Build homepage, click Console, hover the cursor on the username in the upper right corner, and choose My Credentials from the drop-down list. In the navigation pane on the left, choose Access Keys to create a user key.
- If you want to push the created image to other image repositories, perform the following operations.
 - a. Access the CodeArts Build homepage from the project list.
 - b. In the navigation pane, choose **Settings** > **General** > **Service Endpoints**.
 - Select **Docker repository** from the **Create Endpoint** drop-down list box.
 In the displayed dialog box, enter the following information and click **Confirm**.
 - Service Endpoint Name: Assign a custom name to the endpoint. Enter a maximum of 256 characters, including letters, digits, hyphens (-), underscores (_), periods (.), and spaces.
 - Repository Address: Enter the address of the target image repository.
 - Username: Enter the username for logging in to the repository.
 - Password: Enter the password used for logging in to the repository.

Build on GUI

Add **Build Image and Push to SWR** after **Build with Maven**, when **configuring build actions**.

Retain the default values for the **Build with Maven** action. If the current parameter settings do not meet your requirements, modify the parameter settings by referring to **Building with Maven**. For details about the parameters for the **Build Image and Push to SWR** action, see **Table 5-64**.

Table 5-64 Parameters for creating an image and pushing it to SWR

Paramet er	Description		
Action Name	 Assign a custom name to the build action. The name can contain: Letters, digits, hyphens (-), underscores (_), commas (,), semicolons (;), colons (:), periods (.), slashes (/), and parentheses. 1 to 128 characters. 		
Tool Version	Select the Docker version, or use the default one. Currently, CodeArts Build supports Docker 18.03 and Docker 20.10.		
Image Reposito ry	Select the target image repository. You can push the image to Huawei Cloud SWR and other image repositories.		
Authoriz ed User	Specify the user to which the target image repository belongs. You can push the image to the current user or other user's image repository.		
	Ensure that you have permissions to edit or manage all images in the organization. For details, see User Permissions .		
IAM Account	Expand the drop-down list and select the service endpoint created in Preparations for the specific IAM account. Then, use the service endpoint to push the files to the user's SWR.		
	This parameter is mandatory when Authorized User is set to Other .		
Push Region	Select the target region of your push. The built image will be pushed to the SWR repository in this region.		
Docker Reposito ry Endpoin t	Select the Docker repository service endpoint created in Preparations and push the image to the corresponding repository through the service endpoint.		
Organiz ation	Select the organization created in Preparations from the drop-down list box. The image will be placed in this organization after being pushed to SWR.		
Image Name	Enter the name of the created image. The value must start with a digit or letter and can contain 1 to 255 characters, including only lowercase letters, digits, underscores (_), and hyphens (-).		

Paramet er	Description
Image Tag	Specify the image tag, which can be customized. You can use <i>Image name</i> : <i>Tag</i> to uniquely specify an image.
	The value can contain 1 to 128 characters, including only letters, digits, periods (.), underscores (_), and hyphens (-). It cannot start with periods or hyphens.
Working Director y	Work directory of docker build . The context path parameter in the docker build command is the relative path of the root directory of the repository.
	When Docker builds an image, the docker build command packs all content under the context path and sends it to the container engine to help build the image.
Dockerfi le Path	Path of the Dockerfile. Set this parameter to a path relative to the working directory. For example, if the working directory is the root directory and the Dockerfile is in the root directory, set this parameter to ./Dockerfile.
Add Build Metadat a to Image	Specify whether to add the build information to the image. After the image is created, run the docker inspect command to view the image metadata.
Continu e After Failure	Specify whether to proceed after the current action fails by setting the parameter to either Yes or No .

```
version: 2.0 # The value must be 2.0.
steps:
 BUILD:
  - build_image:
     name: buildImage
     inputs:
      regions: ["x-x-x", "x-x-xxx"]
      organization: codeci_test
      image_name: demo
      image_tag: ${GIT_COMMIT}
      dockerfile_path: dockerfile/Dockerfile
      build_args:
       - name: aaa
        value: iuhascihsd
      # set_meta_data: true
      ignore_fail: true
```

Table 5-65 Parameters in the sample code for creating an image and pushing it to SWR

Parame ter	Туре	Description
regions	List	Optional. Select the region of SWR where the image is to be uploaded to. By default, the image is uploaded to SWR in the region where the current task is located. If multiple regions are configured, the built image will be pushed to SWR in each region in sequence after the image is created.
organiz ation	String	Enter the name of the organization to which the image belongs after being pushed to SWR. The organization name is the name of the organization created in Preparations .
image_ name	String	Optional. Enter the name of the created image. The value must start with a digit or letter and can contain 1 to 255 characters, including only lowercase letters, digits, underscores (_), and hyphens (-). The default value is demo .
image_t ag	String	Optional. Specify the image tag, which can be customized. You can use <i>Image name</i> : <i>Tag</i> to uniquely specify an image. The value can contain 1 to 128 characters, including only letters, digits, periods (.), underscores (_), and hyphens (-). It cannot start with periods or hyphens. The default value is v1.1 .
context _path	String	Optional. Work directory of docker build. The context path parameter in the docker build command is the relative path of the root directory of the repository. When Docker builds an image, the docker build command packs all content under the context path and sends it to the container engine to help build the image. The default value is
dockerfi le_path	String	Optional. Path of the Dockerfile. Set this parameter to a path relative to the working directory. For example, if the working directory is the root directory and the Dockerfile is in the root directory, set this parameter to ./Dockerfile. The default value is ./Dockerfile.
build_ar gs	String	

Parame ter	Туре	Description
set_met a_data	Bool	Optional. Specify whether to add the build information to the image. After the image is created, run the docker inspect command to view the image metadata. • true: Add the build information to the image. • false: Do not add the build information to the image. The default value is false.
ignore_f ail	String	Whether to proceed after the current action fails.true: YesEmpty: No

5.3.28 Using a Custom Build Environment

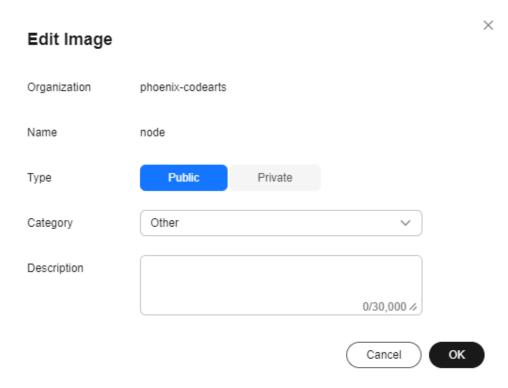
If the tool version supported by CodeArts Build does not meet your requirements, you can use a custom image that has been uploaded to SWR.

Setting the Image Type to Public

Private images in SWR cannot be pulled by CodeArts Build during the build process. Therefore, you need to set the image type to **Public** before starting the build.

- 1. Log in to SWR.
- 2. In the navigation pane, choose **My Images**, click the image name to go to the image details page, and click **Edit** in the upper right corner.
- 3. In the displayed dialog box, set **Type** to **Public** and click **OK**.

Figure 5-6 Editing an image



4. To obtain the complete image path, click to copy the image download command. The part following **docker pull** is the image path.



Build on GUI

Add **Use SWR Public Image** when **configuring build actions**. Set the parameters according to **Table 5-66**.

Table 5-66 Parameters for using an SWR public image

Param eter	Description
Action Name	 Assign a custom name to the build action. The name can contain: Letters, digits, hyphens (-), underscores (_), commas (,), semicolons (;), colons (:), periods (.), slashes (/), and parentheses. 1 to 128 characters.
Image Addres s	Enter the image path obtained in 4 .

Param eter	Description
Comm ands	Configure commands, or use the default ones. If you have special build requirements, enter your custom build script in the text box.
	For example, if the image is used for a Maven build, configure commands for building with Maven. For an npm build, configure commands for building with npm. This rule also applies to other builds.
Contin ue After Failure	Specify whether to proceed after the current action fails by setting the parameter to either Yes or No .

```
version: 2.0 # The value must be 2.0.
steps:
BUILD:
- swr:
image: cloudbuild@ddd
inputs:
command: echo 'hello'
ignore_fail: true
```

Table 5-67 Parameters in the sample code for using an SWR image

Param eter	Туре	Description
image	String	 Set the image path in either of the following ways: Use an address that starts with cloudbuild and uses the tag sign (@) as a separator, with the tool version supported by CodeArts Build following it. For example, cloudbuild@maven3.5.3-jdk8-open, where maven3.5.3-jdk8-open is the version of Maven being used. Use the image path obtained in 4.
comma nd	String	Configure the command. For example, if the image is used for a Maven build, configure commands for building with Maven. For an npm build, configure commands for building with npm. This rule also applies to other builds.
ignore_ fail	String	Whether to proceed after the current action fails. • true: Yes • Empty: No

Image FAQs

For questions about images, see Image FAQs.

5.3.29 Downloading a Software Package from Release Repos

CodeArts Build allows you to download packages or files from the release repo to the build task root directory for use in subsequent build actions.

Obtaining the Package Download Address

- **Step 1** In the navigation pane, choose **Artifact** > **Release Repos**.
- **Step 2** Click the name of the package to be downloaded. On the package details page, the **Repository Path** is the download URL. Click next to the address to copy it.

Figure 5-7 Software package address



----End

Build on GUI

Add **Download Package from Release Repos** when **configuring build actions**. Set the parameters according to **Table 5-68**.

Table 5-68 Parameters for downloading a package from the release repo

Parame ter	Description	
Action Name	 Assign a custom name to the build action. The name can contain: Letters, digits, hyphens (-), underscores (_), commas (,), semicolons (;), colons (:), periods (.), slashes (/), and parentheses. 1 to 128 characters. 	
Tool Version	Select a tool version.	
Package Address	Paste the address copied in Step 2 to the text box.	
Continu e After Failure	Specify whether to proceed after the current action fails by setting the parameter to either Yes or No .	

Modify the code in the **BUILD** block in **Creating a YAML File for Your Code-based Build** by referring to the following sample code:

```
version: 2.0 # The value must be 2.0.
steps:
BUILD:
- download_artifact:
    inputs:
    url: xxxxxxxxxxx
    ignore_fail: true
```

Table 5-69 Parameters in the sample code

Para mete r	Туре	Description
url	String	Paste the address copied in Step 2 .
ignor e_fail	String	Whether to proceed after the current action fails. • true: Yes • Empty: No

5.3.30 Downloading File from File Manager

CodeArts Build stores your Android APK signature files and **settings.xml** files of Maven builds, and helps you manage these files. For example, you can create, edit, and delete these files, and modify users' permissions on them. For details about how to upload files, see **Managing Files**. Add the **Download File from File Manager** action to download files from **Files** to the working directory for use.

Build on GUI

Add **Download File from File Manager**, when **configuring build actions**.

The parameters are described in the following table.

Parameter	Description
Action Name	Assign a custom name to the build action.
Tool Version	Select a tool version.
File Name	 Select an uploaded file from the drop-down list. Click Upload to upload a local file to File Manager. Click Manage Files to manage files on the Files page.

Build with Code

version: 2.0 # The value must be 2.0. steps:

BUILD:		
DOILD.		
download_file:		
inputs:		
name: android22.jks		

Para met er	Typ e	Description	Ma nd ato ry	Default Value
nam e	Stri ng	File name.	Yes	None

5.3.31 Uploading a Software Package to Release Repos

For details about the restrictions on uploading software packages, see **constraints** of CodeArts Artifact.

- Only files can be uploaded, folders cannot be uploaded, and directories cannot be automatically created.
 - For example, the **a** directory contains the **aa** file and **b** directory that contains the **bb** file, and the build package directory is set to **a**/**.
 - When the **a** directory is scanned, both **aa** and **bb** will be uploaded to the same directory, and the system will not create a **b** directory in release repos.
- To upload a folder, package it before adding the Upload to Release Repo action. You can package the folder by running the packaging command or adding the Run Shell Commands action.

Build on GUI

Add **Upload to Release Repo**, when **configuring build actions**. Set the parameters according to **Table 5-70**.

□ NOTE

When you select Windows **executors**, add action **Upload Software Package to Release Repos (Windows)**.

Table 5-70 Parameters for uploading a software package to the release repo

Parame ter	Description
Action Name	 Assign a custom name to the build action. The name can contain: Letters, digits, hyphens (-), underscores (_), commas (,), semicolons (;), colons (:), periods (.), slashes (/), and parentheses. 1 to 128 characters.

Parame ter	Description					
Package Locatio n	 Directory for storing the build result. The build package directory supports regular expression matching. ** means that the system recursively traverses the current directory. * indicates zero or multiple characters. ? indicates one character. The system file uses slashes (/) as separators, and the path is not case-sensitive. Examples: 					
	 *.class: Matches files whose names end with .class in the current directory. **/*.class: Recursively matches all files whose names end with .class in the current directory. test/a??.java: Matches Java files whose names start with a followed by two characters in the test directory. **/test/**/XYZ*: Recursively matches all files whose parent directory is test and whose names start with XYZ, for example, abc/test/def/ghi/XYZ123. 					
Version	 Optional. Set the name of the directory where the software package generate by the build task will be uploaded to the release repo. Not specified (recommended): Use the build number to name the directory for storing files uploaded to release repos. Specified: Files in the directory with the same name may be overwritten. 					
Package Name	 Optional. Set the name for the software package generated by the build task. The name will be used when the package is uploaded to the release repo. Not specified (recommended): Use the original file name to name the file uploaded to release repos. Leave Package Name unspecified so that all files matching the build package directory can be uploaded. Specified: A file may be overwritten when another file with the same name is uploaded. For multiple file uploads with different package names, repeat the upload action for each file. 					
Custom Director y	Optional. After you specify the custom upload directory, the uploaded software package is uploaded to the <i>custom upload directory</i> version number Software package name directory.					
Continu e After Failure	Specify whether to proceed after the current action fails by setting the parameter to either Yes or No .					

Build with Code

Modify the code in the **BUILD** block in **Creating a YAML File for Your Code-based Build** by referring to the following sample code:

```
version: 2.0 # The value must be 2.0.
steps:
BUILD:
- upload_artifact:
    inputs:
    path: "**/target/*.?ar"
    version: 2.1
    name: packageName
    custom_upload_path: /phoenix-sample-ci/
    ignore_fail: true
```

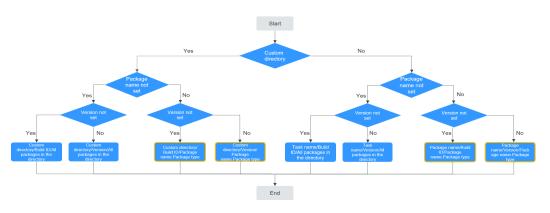
Table 5-71 Parameters in the sample code

Par am ete r	Туре	Description
pat h	Strin g	Directory for storing the build result.The build package directory supports regular expression
		matching. ** means that the system recursively traverses the current directory. * indicates zero or multiple characters. ? indicates one character.
		The system file uses slashes (/) as separators, and the path is not case-sensitive.
		Examples:
		*.class: Matches files whose names end with .class in the current directory.
		**/*.class: Recursively matches all files whose names end with .class in the current directory.
		• test/a??.java : Matches Java files whose names start with a followed by two characters in the test directory.
		 /test//XYZ*: Recursively matches all files whose parent directory is test and whose names start with XYZ, for example, abc/test/def/ghi/XYZ123.
vers	Strin	Optional.
ion	g	Enter the release version number.
		Not specified (recommended): Use the build number to name the directory for storing files uploaded to release repos.
		Specified: Files in the directory with the same name may be overwritten.

Par am ete r	Туре	Description
na	Strin	Optional.
me	g	Enter the name of the package generated during the build.
		Not specified (recommended): Use the original file name to name the file uploaded to release repos.
		Specified: A file may be overwritten when another file with the same name is uploaded.
cust	Strin	Optional.
om _upl oad _pa th	g	After you specify the custom upload directory, the uploaded software package is uploaded to the custom upload directory/version number/Software package name directory.
ign	Strin	Whether to proceed after the current action fails.
ore_ fail	g	• true: Yes
Idil		Empty: No

How Release Versions and Package Names Impact Uploads

Figure 5-8 Impact of an unspecified release version and package name on uploads



5.3.32 Uploading Files to OBS

CodeArts Build allows you to upload build products to OBS. You can use this build action as required.

For details about the restrictions on using OBS, see **Restrictions and Limitations**.

Preparations

To upload files to OBS of other users, perform the following operations.

- 1. Access the CodeArts Build homepage from the project list.
- 2. In the navigation pane, choose **Settings** > **General** > **Service Endpoints**.
- 3. Select **IAM user** from the **Create Endpoint** drop-down list box. In the displayed dialog box, enter the following information and click **Confirm**.
 - Service Endpoint Name: Assign a custom name to the endpoint. Enter a maximum of 256 characters, including letters, digits, hyphens (-), underscores (_), periods (.), and spaces.
 - Access key ID (AK) and secret access key (SK) are used like passwords to authenticate users who make API requests.
 - On the CodeArts Build homepage, click **Console**, hover the cursor on the username in the upper right corner, and choose **My Credentials** from the drop-down list. In the navigation pane on the left, choose **Access Keys** to create a user key.

Build on GUI

Add **Upload Files to OBS**, when **configuring build actions**. Set the parameters according to **Table 5-72**.

Table 5-72 Parameters for uploading files to OBS

Paramet er	Description
Action Name	 Assign a custom name to the build action. The name can contain: Letters, digits, hyphens (-), underscores (_), commas (,), semicolons (;), colons (:), periods (.), slashes (/), and parentheses. 1 to 128 characters.
Authoriz ed User	 Select the user. Your files will be pushed to the user's OBS. Current: Upload files to an OBS bucket of the current user. Other: Upload files to OBS of a specific user by specifying an IAM account.
IAM Account	Expand the drop-down list and select the service endpoint created in Preparations for the specific IAM account. Then, use the service endpoint to push the files to the user's OBS. This parameter is mandatory when Authorized User is set to Other .

Paramet er	Description
Build Directory	Directory for storing build results. If no file name is specified for OBS storage, use wildcard characters to upload multiple files. Example: **/target/*.?ar uploads all JAR and WAR packages built with Maven. Examples: **.class: Matches files whose names end with .class in the current directory. **/*.class: Recursively matches all files whose names end with .class in the current directory. *test/a??.java: Matches Java files whose names start with a followed by two characters in the test directory. **/test/**/XYZ*: Recursively matches all files whose parent
	directory is test and whose names start with XYZ , for example, abc/test/def/ghi/XYZ123 .
Bucket Name	Name of the target OBS bucket. (Cross-region upload is not supported.)
OBS Directory	Optional. Directory for storing build results on OBS (for example, application/version/). You can leave this parameter blank or enter ./ to store build results to the OBS root directory.
File Name	 Optional. Enter the name (without the directory) that the resulting build file will be stored as in OBS. If leave it as blank, you can upload multiple files and use the build product file name as the name it will be stored as in OBS. If you do not leave it as blank, you can upload only one file, such as application.jar.
Upload Folder	You can choose whether to enable the function of uploading folders. • Yes: The folder is also uploaded. • No: The file is uploaded, but not the folder.
Headers	Optional. Add one or more custom response headers during the file upload. The headers will be included in the response to download objects or query the object metadata. For example, you can set the key to x-frame-options and value to false to prevent web pages stored in OBS from being embedded into by third-party web pages.
Continue After Failure	Specify whether to proceed after the current action fails by setting the parameter to either Yes or No .

Build with Code

Modify the code in the **BUILD** block in **Creating a YAML File for Your Code-based Build** by referring to the following sample code:

```
version: 2.0 # The value must be 2.0.
steps:
BUILD:
    - upload_obs:
    inputs:
        artifact_path: "**/target/*.?ar"
        bucket_name: codecitest-obs
        obs_directory: "./"
        # artifact_dest_name: ""
        # upload_directory: true
        # headers:
        # x-frame-options: true
        # test: test
        # commit: ${commitId}
        ignore_fail: true
```

Table 5-73 Parameters in the sample code

Paramet er	Туре	Description
artifact_ path	String	Optional. Directory for storing build results. If no file name is specified for OBS storage, use wildcard characters to upload multiple files. Example: **/target/*.?ar uploads all JAR and WAR packages built with Maven. Examples: *.class: Matches files whose names end with .class in the current directory. **/*.class: Recursively matches all files whose names end with .class in the current directory. test/a??.java: Matches Java files whose names start with a followed by two characters in the test directory. **/test/**/XYZ*: Recursively matches all files whose parent directory is test and whose names start with XYZ, for example, abc/test/def/ghi/XYZ123. The default value is bin/*.
bucket_n ame	String	Name of the target OBS bucket. (Cross-region upload is not supported.)
obs_dire ctory	String	Optional. Directory for storing build results on OBS (for example, application/version/). You can leave this parameter blank or enter ./ to store build results to the OBS root directory. The default value is ./.

Paramet er	Туре	Description
artifact_	String	Optional.
dest_na me		Enter the name (without the directory) that the resulting build file will be stored as in OBS.
		 If leave it as blank, you can upload multiple files and use the build product file name as the name it will be stored as in OBS.
		 If you do not leave it as blank, you can upload only one file, such as application.jar.
upload_	Bool	Optional.
directory		Specify whether to enable the function of uploading folders.
		• true : The folder of the build product is also uploaded.
		 false: All matched build products are uploaded to obs_directory in tile mode.
		The default value is false .
headers	Мар	Optional.
		Add one or more custom response headers during the file upload. The headers will be included in the response to download objects or query the object metadata.
		For example, you can set the value of x-frame-options to false to prevent web pages stored in OBS from being embedded into by third-party web pages.
ignore_f	String	Whether to proceed after the current action fails.
ail		• true: Yes
		Empty: No

5.4 Configuring Parameters

Predefined Parameters

The values of predefined parameters are automatically generated by the system and do not need to be defined, as shown in **Table 5-74**. You can use **\${Parameter name}** to reference the parameters in the code.

Table 5-74 Predefined parameters

Parameter	Description
BUILDNUMB ER	Build ID in the format of <i>date.times that this build task is run on that day</i> . For example: 20200312.3 .
TIMESTAMP	Build task running timestamp. For example: 20190219191621.

Parameter	Description
INCREASENU M	Total number of times that the build task is run. The value starts from 1 and is incremented by 1 each time the task is run.
PROJECT_ID	ID of the project where the build task is located.
WORKSPACE	Root directory of the source code pulled by the build task, also known as the workspace.
GIT_TAG	Code tag name. This parameter only has a value if you have specified the downloaded code by tag.
COMMIT_ID_ SHORTER	First eight digits of the code commit ID. This parameter only has a value if you have specified the downloaded code by commit ID.
COMMIT_ID	Code commit ID. For example: b6192120acc67074990127864d3fecaf259b20f5.

Adding Custom Parameters

On the page for configuring the build task, click the **Parameters** tab. On the displayed page, click **Create Parameter**, and set parameters according to **Table 5-75**. The system encrypts the parameters for storage and decrypts them before usage.

Table 5-75 Adding custom parameters

Name	Typ e	Default Value	Private Paramete r	Runtime Settings	Para ms Descr iption
Name of a custom parameter. The value can contain a maximum of 128 characters, including	Stri ng	Default value of the custom parameter. Max. 8,192 characters.	Specify whether the parameter is private	Specify whether to set this parameter when running the build task. If Runtime Settings is enabled, the parameter value can be changed when you click to run the build task, and the system reports the parameter to CodeArts Pipeline.	Enter additi onal infor matio n to descri be the para meter. Max. 1,024 chara cters.
letters, digits, and underscores (_). NOTE • Do not use the following fields: LD_PRELOAD, LD_LIBRARY_PAT H, PATH, BASH_ENV, GIT_SSH_COMMA ND, and path. • Symbols are not supported.	Enu mer atio n	In the displayed dialog box, enter enumerated values in the Value text box. Each parameter value must end with a semicolon (;). Max. 8,192 characters. Once you have set the enumerated values, select a default value for the parameter from the Default Value dropdown list box.	or not. If the parameter is private, the system encrypts the input parameter for storage and only decrypts the parameter for usage. Private parameter s are not displayed in run logs.		
	Aut o Incr eme nt	Default value of the custom parameter. Max. 8,192 characters.			

Using Parameters

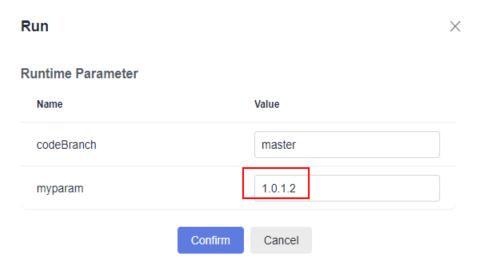
The following section describes how to use custom parameters, as shown in Figure 5-9.

Figure 5-9 Custom parameters



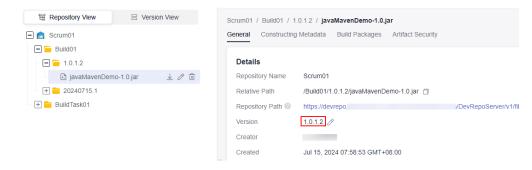
- On the page for configuring the build task, click the Build Actions tab, enter \${myparam} in the Version field of the Upload to Release Repo action, and click Save and Execute.
- 2. In the displayed dialog box, change the value of **myparam** to **1.0.1.2** and click **Confirm**. Wait until the build task is completed.

Figure 5-10 Setting the runtime parameters



3. Go to the release repo and find the resulting build package. The version number is the modified value of **myparam**.

Figure 5-11 Viewing the build package



5.5 Configuring Schedules

With CodeArts Build, you can configure triggers and schedule tasks, so developers can achieve continuous project integration.

Constraints

- The schedule setup is unavailable if the code source is set to **Pipeline**.
- Continuous integration is only supported when the code source is set to Repo.
- If a scheduled build task fails to run 10 times in a row, it will automatically stop being triggered further.

Scheduling a Build Task

On the page for editing the build task, click the **Schedule** tab and configure an execution plan.

• Continuous Integration: Once Run upon Code Commit is enabled, committing the referenced code source will trigger a build task.

This option can be enabled only when **Repo** is selected as the code source.

Figure 5-12 Configuring continuous integration

Continuous Integration

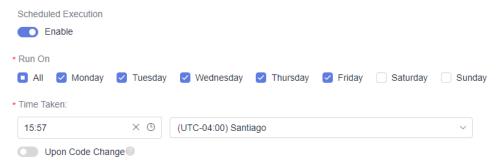
Run upon Code Commit

• **Scheduled Execution**: Enable this option and schedule the run time for the build task. Enable **Run upon code change** if needed.

After this function is enabled, the build task is run at the specified date and time.

If you enable **Run upon code change**, the build task will only run at the specified date and time if there have been changes to the code since the last build.

If the scheduled build task fails to be run 10 consecutive times, it will not be triggered again.



5.6 Configuring Roles and Permissions

CodeArts Build allows you to configure permissions for each role of the build task.

1. On the page for editing the build task, click the **Permissions** tab page and configure operation permissions for different roles. For details about the default permissions of each role, see **Table 5-76**.

Changes to task creator's permissions are locked down to the project creator themselves and the project manager.

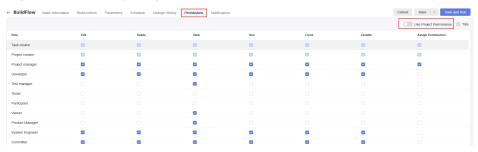
Table 5-76 Default roles and permissions matrix for CodeArts Build

Role	Creat e	Edit	Delet e	View	Exec ute	Clon	Disa ble	Assig n Perm issio ns	Gr ou p
Task creat or	②	②	②	②	Ø	②	⊘	②	②
Proje ct creat or	②	②	②	Ø	②	②	②	Ø	>
Proje ct mana ger	Ø	⊘	Ø	⊘	S	©	Ø	>	S
Prod uct mana ger	8	8	0	>	0	0	0	8	8
Test mana ger	8	8	8	②	8	8	8	8	8
Oper ation mana ger	8	8	8	3	•	0	0	3	8
Syste m engin eer	②	②	Ø	Ø	Ø	②	②	8	⊘
Com mitte r	②	②	②	②	②	②	②	8	8
Devel oper	②	②	②	②	②	②	②	8	3
Teste r	3	8	8	8	8	3	3	8	8
Partic ipant	8	8	8	8	8	3	8	8	0
View er	8	8	8	8	8	8	8	8	8

Role	Creat e	Edit	Delet e	View	Exec ute	Clon e	Disa ble	Assig n Perm issio ns	Gr ou p
Proje ct admi n	>	()	>	③	③	③	③	③	()

2. Toggle on **Use Project Permissions** to synchronize the current build task's permissions with the project's permissions. For details about how to configure project permissions, see **Configuring Role Permissions**.

Figure 5-13 Configuring roles and permissions for a build task



5.7 Configuring Notifications

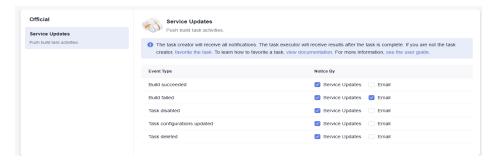
CodeArts Build can send you event notifications on your build task updates, including success, failure, disabling, changes, and deletion.

On the page for configuring a build task, click the **Notifications** tab and set the parameters.

Internal Messages and Emails

Click Service Updates under Official.

By default, you will receive internal messages for all events and emails for build failures. Adjust the notification settings as needed.



6 Running a Build Task

A build task can be triggered in the following ways:

- Run a single build task on the CodeArts Build page.
- Trigger task execution upon code commits to a CodeArts Repo repository. For details, see **Continuous Integration**.
- Execute the task on a scheduled basis. Alternatively, trigger task execution at specified time, but only if the code has changed since the last build. For details, see **Configuring Scheduled Execution**.
- Trigger a build task by running a pipeline.

This section describes how to run a single build task on the CodeArts Build page.

Prerequisites

You have **created a build task** and you have permissions to run or disable the build task.

Procedure

- 1. Access the CodeArts Build homepage from the project list.
- 2. Search for the target build task on the CodeArts Build homepage and click to run the task.

If **runtime parameters have been configured** for the build task and are referenced, the parameter setting dialog box is displayed. Set the parameters as required and click **OK**.

□ NOTE

Build FAQs

For questions about build task execution, see General FAQs.

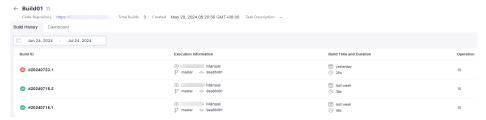
Viewing a Build Task

- 1. Access the CodeArts Build homepage from the project list.
- 2. The build task list related to the current user is displayed, showing the following information.

Item	Description
Build Tasks	Name of the project to which the build task belongs and the build task name. You can click the project name to go to the build list of the project and click the task name to go to the build history page.
Last Executed	Information such as the task executor, triggering mode, branch of the used repository, and commit ID.
Result	The most recent executions are shown from right to left in real time. You can tell the task status from the bar color (green: Succeeded, blue: Running, red: Failed, and gray: Not built).
Build Time and Duration	Build task start time and build duration.
Operation	Click to start builds, to favorite tasks, and to expand the drop-down list (edit, clone, disable, and delete tasks.) For details, see Managing Build Tasks.

3. Click the build task name to go to the **Build History** page. You can view the latest build history. (The build records in latest 31 days are displayed by default. You can customize the period using the date selection component in the upper left corner of the page.)

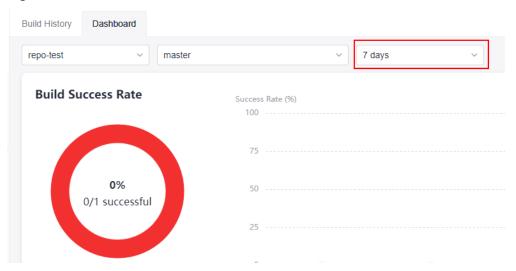
Figure 7-1 Build history



4. Click the **Dashboard** tab to view the build success rate and build performance distribution in the last seven days in pie, line, and bar charts.

You can select a period from the drop-down list box.

Figure 7-2 Dashboard



5. Click a build ID on the **Build History** tab to view details, including the code source, trigger source, build time and duration, associations, queuing duration, action logs, and build parameters.



- Click the code source link in the upper left corner to access the code repository page.
- Click **Download Build Package** and expand the drop-down list. To download all build packages, click **Download All**. To view all build packages, click **Go to Artifact** and go to the **Release Repos** page. To download a specified package, click the name of the package.
- Click an action node (such as Code checkout) on the left to view the build logs.
- When viewing logs, click Full Screen in the upper right corner of the log window to maximize the log window, click Exit Full Screen to exit the maximized log window, choose Download > Download Logs to

download all log files, and click an action node on the left to view logs of the corresponding action.

-	Click Edit or Execute in the upper right corner to edit or run the build
	task. Click and clone the task, save the task as a template, view the badge status, or disable the task.

8 Managing Build Tasks

Ensure you have the required permissions before performing any operations on build tasks.

Editing a Build Task

- 1. Access the CodeArts Build homepage from the project list.
- 2. Search for the target build task.
- In the row of the target build task, click --- and select Edit from the dropdown list.
 - On the **Basic Information** tab page, configure the task name, code source, code repository, default branch, and task description.
 - On the **Build Actions** tab page, configure build actions and parameters.
 - On the Parameters tab page, customize parameters for running the build task.
 - On the Schedule tab page, configure continuous integration (the triggering event) and scheduled execution.
 - On the Change History tab page, view the change history of the build task. Click Compare Difference to see what has been adjusted compared to the previous execution.
 - On the **Permissions** tab page, configure permissions for different roles.
 - On the Notifications tab page, configure notifications for different types of events (including Build succeeded, Build failed, Task deleted, Task configurations updated, and Task disabled).
- 4. Edit the information on a tab page, and click **Save**.

Deleting the Build Task

Click ••• in the row that contains the target build task and select **Delete** from the drop-down list. Exercise caution when performing this operation.

You can view the deleted build task in the recycle bin. In the upper right corner of the CodeArts Build homepage, click **More** and select **Recycle Bin** from the dropdown list.

The page displays deleted build tasks and allows the operations listed in the following table.

Operation	Description
Modify the task retention period	Click the select box next to Task Retention Period and select from 1 to 30 days.
Search for a task	Enter a keyword in the search box and click Q .
Delete a task	Select the task to be deleted from the list and click Delete to delete the task from the recycle bin.
Restore a task	Select the task to be restored from the list and click Restore . Then you can find this task again in the task list of CodeArts Build.
Clear the recycle bin	Click Empty Recycle Bin to delete all tasks from the recycle bin.

Cloning the Build Task

- Click --- in the row of the build task and select Clone from the drop-down list.
- 2. On the displayed page, modify the task information as required and click **Save** to create a clone of the build task.

If you want to both clone the build task and run the clone, click **Save and Execute**.

□ NOTE

Cloning a task will duplicate all of its permissions. The new task has identical access control settings as the original.

Disabling a Task

- The build task that is currently running cannot be disabled or deleted.
- After the build task is disabled, **Disabled** is displayed next to the build task
 name. To run the build task, click in the row that contains the build task
 and select **Enable** from the drop-down list.
- 1. Click in the row that contains the target build task and select **Disable** from the drop-down list.
- 2. In the displayed **Disable Task** dialog box, enter the reason and click **OK**.

Favoriting the Build Task

 After you favorite a build task, the task is displayed on the top of the task list when you refresh the page or access the task list next time. If you favorite many build tasks, the tasks are sorted by task creation time in descending order.

- If you favorite a task that is not created by yourself, you can obtain the corresponding notification based on the notification event type set for the task.
- 1. Move the cursor to the row of the build task and click \Box . If the color of the icon changes, the task is successfully favorited.
- 2. (Optional) Click to unfavorite the task.

Stopping a Build Task

- 1. Click the name of a running build task. The **Build History** page is displayed.
- 2. Click the **Build ID**.
- 3. On the displayed page, click **Stop** in the upper right corner.

Managing Build Tasks in Batches

Check the boxes next to the desired build tasks. In the displayed dialog box, you can click different buttons to perform batch operations on the selected build tasks: **Move To** changes the group of the build tasks to a specified one. **Permissions** sets permissions for build task roles. **Delete** removes the build tasks.

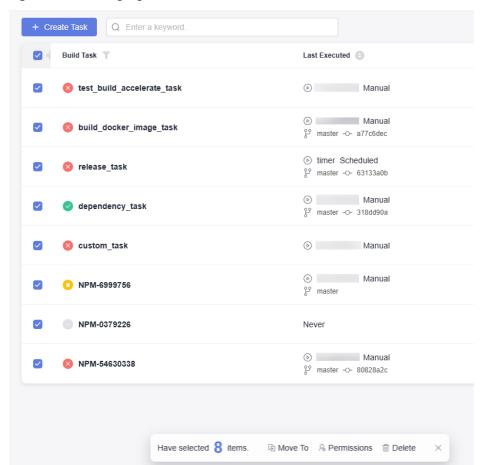


Figure 8-1 Managing build tasks in batches

9 Querying Audit Logs

Cloud Trace Service (CTS) records operations on CodeArts Build for query, audit, and backtrack.

Operations Recorded by CTS

Table 9-1 CodeArts Build operations recorded by CTS

Operation	Resource Type	Event
Creating a build task	BuildProjectService	createJob
Running a build task	BuildProjectService	buildJob
Deleting a build task	BuildProjectService	deleteJob
Updating a build task	BuildProjectService	updateJob
Disabling a build task	BuildProjectService	disableJob
Enabling a build task	BuildProjectService	enableJob
Uploading a keystore file	BuildProjectService	uploadKeystore
Updating a keystore file	BuildProjectService	updateKeystore
Deleting a keystore file	BuildProjectService	deleteKeystore
Initializing the EFS directory and storage quota	CloudBuildCache	initEFSDirAndQuota
Uploading a report (including the unit test and dependency analysis)	CloudBuildReport	uploadReport
Creating a custom template	BuildProjectService	createCustomTemplate

Operation	Resource Type	Event
Deleting a custom template	BuildProjectService	deleteCustomTemplate
Updating nextfs information	nextfsInfo	updateNextfsInfo
Creating nextfs	nextfsInfo	createNextfsInfo
Associating nextfs with a tenant	tenantNextfs	createTenantNextfs
Disassociating a tenant from nextfs	tenantNextfs	deleteTenantNextfs
Modifying License information	licenseInfo	updateLicenseInfo
Creating a tenant license	licenseInfo	createLicenseInfo
Creating code cache information	codeCacheInfo	createCodeCacheInfo
Deleting code cache information	codeCacheInfo	deleteCodeCacheInfo
Creating records of using code cache	cacheHistoryInfo	createCacheHistoryInfo
Updating usage info of code cache	cacheHistoryInfo	updateCacheHistoryInfo

Viewing Audit Logs

Query CodeArts Build traces on the CTS console. For details, see **Viewing Audit Logs**.

10 References

10.1 Syntax Guide to YAML File Configuration

Sample Code for Single-task Build

```
version: 2.0
#Parameters are specified in pairs, with a name and a corresponding value. If no value is assigned to a
parameter, it will default to an empty string. When referencing a parameter, use the syntax ${parameter}
params:
 - name: machineArch
  value: X86
#(Optional) Configure either env or envs to set up the build environment. Use envs if you need to specify
conditions to determine the host specification and type.
 resource:
  type: docker
  arch: X86
  class: 8U16G
  pool: Mydocker
envs:
 - condition: machineArch == 'ARM'
  resource:
    type: docker
    arch: ARM
 - condition: machineArch == 'X86'
  resource:
    type: docker
    arch: X86
#Build actions
 PRE BUILD:
  - checkout:
     name: checkout
     inputs:
      scm: codehub
      url: git@codehub.devcloud.cn-north-7.ulanqab.huawei.com:huang-test00001/maven.git
      branch: master
      commit: commitId
      lfs: true
      submodule: true
```

```
depth: 100
     tag: tag
     path: test
 - manifest_checkout:
   name: "manifest"
   inputs:
     manifest_url: https://codehub.devcloud.xxxxxxx.ulanqab.huawei.com/IPD-xxxxxx/manifest.git
     manifest_branch: master
     manifest_file: default.xml
     path: dir/dir02
     lfs: true
     repo_url: https://codehub.devcloud.xxxxxxxx.ulanqab.huawei.com/IPD-xxxxxx/git-repo.git
     repo_branch: master
     username: someone
     password: PASSWD
 - sh:
     command: echo ${machineArch}
BUILD: # Build actions
 - maven:
   name: Build with Maven
   image: cloudbuild@maven3.5.3-jdk8-open
   inputs:
     settings:
      public_repos:
        - https://mirrors.huawei.com/maven
     cache: true
     unit_test:
      coverage: true
      ignore_errors: false
      report_path: "**/TEST*.xml"
      enable: true
      coverage_report_path: "**/site/jacoco"
     command: mvn package -Dmaven.test.failure.ignore=true -U -e -X -B
     check:
      project_dir: ./
      settings: ~/.m2/settings.xml
      param: "
 - upload_artifact:
   inputs:
     path: "**/target/*.?ar"
     version: 2.1
     name: packageName
```

Table 10-1 Single-task syntax configuration

Para mete r	Ty pe	Description	Ma nd ato ry
versio n	Str ing	YAML file version specified with a fixed value. Currently, the only supported version is 2.0.	Yes

Para mete r	Ty pe	Description	Ma nd ato ry
para ms	Ma p	Global parameter configuration. These parameters must be specified in pairs, with a name and a corresponding value. If no value is assigned to a parameter, it will default to an empty string. When referencing a parameter, use the syntax \${parameter name}.	No
		In the preceding sample code, the defined parameter is referenced as an input parameter by following the format \$ {machineArch}. When used as a condition, the declared parameter name machineArch is applied.	
		name: Parameter name.value: The value paired with the parameter name.	
env	Ma p	This build environment configuration serves the same purpose as envs. You can configure either of them. However, env does not support condition statements. For details, see the example in Configuring the Build Environment. • resource: Build environment resource information. • type: (Mandatory) Agent pool type. The value can be docker or custom. docker indicates that the default executor is used, and custom indicates that a custom executor is used. • arch: (Mandatory) Build executor architecture. The value can be either x86 or Arm. • class: (Optional) Build executor specifications. The value can be: 2U8G (2 vCPUs 8 GB), 4U8G (4 vCPUs 8 GB), 8U16G (8 vCPUs 16 GB), 16U32G (16 vCPUs 32 GB), or 16U64G (16 vCPUs 64 GB). This parameter is applicable when the agent pool type (type) is set to docker. The default value is 2U8G. If you need different specifications, purchase the concurrent execution packages that match those specifications. • pool: (Optional) Agent pool name. This parameter is applicable when the agent pool type (type) is set to custom.	No

Para mete r	Ty pe	Description	Ma nd ato ry
envs	Ma p	 This build environment configuration serves the same purpose as env. You can configure either of them. Unlike env, envs supports condition statements, allowing for more flexible usage of the same YAML file in different scenarios. condition: The statement allows you to specify environment information based on certain conditions. The corresponding environment settings from the resource configuration will be applied if the current condition is met. resource: Build environment resource information. type: (Mandatory) Agent pool type. The value can be docker or custom. docker indicates that the default executor is used, and custom indicates that a custom executor is used. arch: (Mandatory) Build executor architecture. The value can be either x86 or Arm. class: (Optional) Build executor specifications. The value can be: 2U8G (2 vCPUs 8 GB), 4U8G (4 vCPUs 8 GB), 8U16G (8 vCPUs 16 GB), 16U32G (16 vCPUs 32 GB), or 16U64G (16 vCPUs 64 GB). This parameter is required when the agent pool type (type) is set to docker. The default value is 2U8G. If you need different specifications, purchase the concurrent execution packages that match those specifications. pool: (Optional) Agent pool name. This parameter is required when the agent pool type (type) is set to custom. 	No
steps	Ma p	A collection of build actions. This parameter configures the build process by specifying: PRE_BUILD: defines build preparations, typically code downloads before the actual build process begins. BUILD: defines and runs service-specific build tasks.	Yes

DRE_BUILD Description Des	Para mete r	Ty pe	Description	Ma nd ato ry
 manifest_checkout: Multi-repo download. For details, see the example in Build with Code (Downloading Code from Multiple Repos via Manifest). name: (Optional) Build action name. You can assign a 	mete r steps: PRE_BUIL	pe Ma	This parameter defines build preparations, typically code downloads before the actual build process begins. Currently, only checkout, manifest_checkout, and sh are supported. Generally, you only need to configure one of them. • checkout: Single-repo download. For details, see the example in Build with Code (Downloading Code from a Single Repo). - name: (Optional) Build action name. You can assign a custom value. If not specified, the value defaults to checkout. - inputs: (Mandatory) Action input parameters. These parameters vary for each action. For details, see Build Actions. - scm: (Optional) Code source. The supported code source is currently limited to codehub, which is also the default value. - url: (Mandatory) SSH or HTTPS address used to pull code. SSH (ssh) is used when code is pulled from CodeArts Repo (codehub), and HTTPS (https) is used for other code sources. - branch: (Mandatory) Name of the code branch to be pulled. - commit: (Optional) Commit ID specified for the build process. - lfs: (Optional) Whether to enable git lfs. The default value is false. - submodule: (Optional) Whether to pull submodules. The default value is false. - depth: (Optional) Shallow clone depth. When a commit ID is specified for builds, depth must be greater than or equal to the depth of the commit ID. The default value is 1. - tag: (Optional) Tag specified for the build process. If both a commit ID and a tag are provided, the build will prioritize the commit ID and run based on it. - path: (Optional) Subdirectory for cloning. The code is	nd ato
manifest_checkout.			 manifest_checkout: Multi-repo download. For details, see the example in Build with Code (Downloading Code from Multiple Repos via Manifest). name: (Optional) Build action name. You can assign a custom value. If not specified, the value defaults to 	

Para mete r	Ty pe	Description	Ma nd ato ry
		 inputs: (Mandatory) Action input parameters. These parameters vary for each action. For details, see Build Actions. 	
		 manifest_url: (Mandatory) Address of a manifest repository that includes one or more XML files. 	
		 manifest_branch: (Optional) The specified manifest branch or revision. The default value is HEAD. 	
		 manifest_file: (Optional) Path of the manifest file. The defined repositories must be of the same code source. The default value is default.xml. 	
		 path: (Optional) The download path for all sub-repositories of the manifest. This path is relative to the working directory, so it cannot start with a forward slash (/) or contain any periods (.). The default value is the current working directory. 	
		 - Ifs: (Optional) Whether to enable git Ifs. The default value is false. 	
		 repo_url: (Optional) Repository address. 	
		 repo_branch: (Optional) Repository branch. The default value is stable. 	
		 username: (Optional) Username used for downloading the repository. This parameter is required for private repositories. 	
		 password: (Optional) Password used for downloading the repository. This parameter is required for private repositories. 	
		• sh : Run Shell commands	
		 inputs: (Mandatory) Action input parameters. These parameters vary for each action. For details, see Build Actions. 	
		 command: (Mandatory) Configuration for running shell commands. If checkout or manifest_checkout cannot meet service requirements, you can customize shell commands to prepare for the build. 	

Para mete r	Ty pe	Description	Ma nd ato ry
steps: BUIL D	Ma p	This parameter defines and runs service-specific build tasks. Only a particular set of build actions are supported and can be flexibly combined to meet service requirements. For details about the build actions, see Selecting Build Actions .	Yes
		maven: defines the Maven build action.	
		 name: (Optional) Build action name. You can assign a custom value. If not specified, the value defaults to Build with Maven. 	
		 image: (Mandatory) Container image used for the build. You can customize an image or use the default one. The default image name follows the format cloudbuild@ Tool version. For details about the tool version, see Build Tools and Versions. 	
		 inputs: (Mandatory) Action input parameters. These parameters vary for each action. For details, see Build Actions. 	
		 settings: (Optional) The settings file configuration used for the Maven build. 	
		 public_repos: Addresses of the repositories from which the dependency packages are downloaded. 	
		 cache: (Optional) Whether to enable the cache. The default value is false. 	
		 unit_test: (Optional) Unit test configuration. 	
		 coverage: (Optional) Whether to process coverage data. The default value is false. 	
		 ignore_errors: (Optional) Whether to ignore unit test errors. The default value is true. 	
		– report_path : (Mandatory) Unit test data path.	
		 enable: (Optional) Whether to process unit test data. The default value is true. 	
		 coverage_report_path: (Optional) Coverage data path. 	
		 command: (Mandatory) Build command. 	
		– check : (Optional) Check configuration.	
		project_dir: (Mandatory) Project path.	
		 settings: (Optional) Path of the settings file used for the Maven build. 	
		– param : (Optional) Maven parameter.	
		upload_artifact: defines the action of uploading the binary package to the artifact repository.	

Para mete r	Ty pe	Description	Ma nd ato ry
		 inputs: (Mandatory) Action input parameters. These parameters vary for each action. For details, see Build Actions. 	
		 path: (Mandatory) Path and name of the file to be uploaded. You can use wildcard characters. 	
		- version : (Optional) The default value is the build ID.	
		 name: (Optional) File name. The default value is the original file name. 	

Multi-task Build

```
version: 2.0
#Parameters are specified in pairs, with a name and a corresponding value. If no value is assigned to a
parameter, it will default to an empty string. When referencing a parameter, use the syntax ${parameter}
name}.
params:
 - name: machineArch
  value: X86
 - name: jobCondition
  value: 1
 - name: jobsCondition
  value: 1
# (Optional) Configure either env or envs to set up the build environment. Use envs if you need to specify
conditions to determine the host specification and type.
env:
 resource:
  type: docker
  arch: X86
  class: 8U16G
  pool: Mydocker
envs:
 - condition: machineArch == 'ARM'
  resource:
   type: docker
   arch: ARM
 - condition: machineArch == 'X86'
  resource:
   type: docker
    arch: X86
# Configure either buildflow or buildflows. Use buildflows if you need to specify conditions to control job
executions.
buildflow:
 strategy: Lazy
  jobs:
    - job: Job3
     depends_on:
      - Job1
       - Job2
     build_ref: .cloudbuild/build3.yml
    - job: Job1
     build_ref: .cloudbuild/build1.yml
    - job: Job2
     build_ref: .cloudbuild/build2.yml
```

```
buildflows:
 - condition: jobsCondition == 1
  jobs:
    - job: Job1
     build_ref: .cloudbuild/build1.yml
     params:
      - name: job1Params
       value: 1
    - condition: jobCondition == 1
     job: Job2
     build_ref: .cloudbuild/build2.yml
     params:
      - name: job2Params
        value: 2
    - job: Job3
     depends_on:
      - Job1
      - Job2
     build_ref: .cloudbuild/build3.yml
 - condition: jobsCondition == 2
  jobs:
     - job: Job3
     build_ref: .cloudbuild/build3.yml
```

Table 10-2 Multi-task syntax configuration

Par ame ter	Туре	Description	Ma nda tor y
versi on	Strin g	YAML file version specified with a fixed value. Currently, the only supported version is 2.0.	Yes
para ms	Мар	Global parameter configuration. These parameters must be specified in pairs, with a name and a corresponding value. If no value is assigned to a parameter, it will default to an empty string. When referencing a parameter, use the syntax \${parameter name}.	No
		In the preceding sample code, the defined parameter is referenced as an input parameter by following the format \$ {machineArch}. When used as a condition, the declared parameter name machineArch is applied.	
		name: Parameter name.value: The value paired with the parameter name.	

Par ame ter	Туре	Description	Ma nda tor y
env	Мар	 This build environment configuration serves the same purpose as envs. You can configure either of them. However, env does not support condition statements. resource: Build environment resource information. type: (Mandatory) Agent pool type. The value can be docker or custom. docker indicates that the default executor is used, and custom indicates that a custom executor is used. arch: (Mandatory) Build executor architecture. The value can be either x86 or Arm. class: (Optional) Build executor specifications. The value can be: 2U8G (2 vCPUs 8 GB), 4U8G (4 vCPUs 8 GB), 8U16G (8 vCPUs 16 GB), 16U32G (16 vCPUs 32 GB), or 16U64G (16 vCPUs 64 GB). This parameter is applicable when the agent pool type (type) is set to docker. The default value is 2U8G. If you need different specifications, purchase the concurrent execution packages that match those specifications. pool: (Optional) Agent pool name. This parameter is applicable when the agent pool type (type) is set to custom. 	No

Par ame ter	Туре	Description	Ma nda tor y
envs	Мар	 This build environment configuration serves the same purpose as env. You can configure either of them. Unlike env, envs supports condition statements, allowing for more flexible usage of the same YAML file in different scenarios. condition: The statement allows you to specify environment information based on certain conditions. The corresponding environment settings from the resource configuration will be applied if the current condition is met. resource: Build environment resource information. type: (Mandatory) Agent pool type. The value can be docker or custom. docker indicates that the default executor is used, and custom indicates that a custom executor is used. arch: (Mandatory) Build executor architecture. The value can be either x86 or Arm. class: (Optional) Build executor specifications. The value can be: 2U8G (2 vCPUs 8 GB), 4U8G (4 vCPUs 8 GB), 8U16G (8 vCPUs 16 GB), 16U32G (16 vCPUs 32 GB), or 16U64G (16 vCPUs 64 GB). This parameter is required when the agent pool type (type) is set to docker. The default value is 2U8G. If you need different specifications, purchase the concurrent execution packages that match those specifications. pool: (Optional) Agent pool name. This parameter is required when the agent pool type (type) is set to custom. 	No

Par ame ter	Туре	Description	Ma nda tor y
buil dflo w	Мар	A build job (referring to the build task on the GUI) is the smallest unit that a build project can be broken down into for simple service scenarios. However, for more complex requirements, you may need to use a multi-repo approach to distribute build jobs that depend on each other across multiple machines. In certain scenarios, you may need to set up multiple build tasks in a modular and fine-grained way, and run them in a specific order. Each task depends on the successful completion of its dependency task. To handle such complex builds, CodeArts Build offers a task model called BuildFlow, which organizes multiple build jobs in a directed acyclic graph (DAG) and runs them concurrently with the maximum capacity based on job dependencies. By doing so, CodeArts Build helps improve build efficiency. • strategy: (Optional) buildflow running policy. The value can be either Lazy or Eager. Lazy: The build takes a long time but saves resources. Therefore, you are advised to use this method when the agent pool is limited. This method triggers jobs with higher priority. Eager: Resources may be idle, but the build time can be shortened. You are advised to use this method when the agent pool quota is sufficient. This method triggers the all jobs at the same time. The default value is Eager.	Yes
		• jobs : (Mandatory) A collection of orchestrated jobs. This item defines the dependency between jobs (sub-tasks).	
		• job : (Mandatory) Name of a job.	
		• depends_on : (Optional) Whether a job depends on other jobs. Enter the name of the dependency jobs. The current job depends on Job1 and Job2.	
		build_ref: (Mandatory) Path (relative to the root directory of the repository) of the YAML file used for building the current job. The YAML file is a complete standalone file that can be executed for the build process. For details, see Sample Code for Single-task Build.	
		• job: (Mandatory) Name of a job. build_ref: (Mandatory) Path (relative to the repository root directory) of the YAML file used for building the current job.	
		• job : (Mandatory) Name of a job. build_ref : (Mandatory) Path (relative to the repository root directory) of the YAML file used for building the current job.	

Par ame ter	Туре	Description	Ma nda tor y	
buil dflo ws	Мар	Configure buildflows if you need to specify conditions. Adapt to different service scenarios for a better use of YAML files. • condition: The statement placed under buildflows allows you to specify job configurations based on certain conditions. The corresponding jobs configuration will be applied if the current condition is met. • jobs: (Mandatory) A collection of orchestrated jobs. This item defines the dependency between jobs (sub-tasks).	Yes	
		 job: (Mandatory) Name of a job. build_ref: (Mandatory) Path (relative to the root directory of the repository) of the YAML file used for building the current job. The YAML file is a complete standalone file that can be executed for the build process. For details, see Sample Code for Single-task Build. 		
		 params: (Optional) Parameters defined by a job. These parameters are scoped to the YAML file referenced by jobs, allowing them to be referenced within the YAML file used by jobs. 		
		– name : Parameter name.		
		 value: The value paired with the parameter name. 		
		 condition: The statement allows you to specify job configurations based on certain conditions. The corresponding job configuration will be applied if the current condition is met. 		
		 job: (Mandatory) Name of a job. build_ref: (Mandatory) Path (relative to the repository root directory) of the YAML file used for building the current job. 		
			 params: (Optional) Parameters defined by a job. These parameters are scoped to the YAML file referenced by jobs. name: Parameter name. 	
		value: The value paired with the parameter name.		
		• job: (Mandatory) Name of a job. depends_on: (Optional) Whether a job depends on other jobs. Enter the name of the dependency jobs. The current job depends on Job1 and Job2.		
		build_ref: (Mandatory) Path (relative to the repository root directory) of the YAML file used for building the current job.		

10.2 Guide to Cache Directory

CodeArts Build offers a dependency cache feature in certain build actions. It significantly enhances the efficiency of downloading dependency packages during the build process. When executing the build task, CodeArts Build mounts the remote cache directory specific to the tenant on the build task executor. This directory can be directly utilized during the build process, eliminating the need for repetitive downloads. For build actions that supported caching, see **Table 10-3**.

Constraints

Before clearing cache, be well aware of the following caveats:

- Since the cache directory is shared among multiple users in the same tenant, frequent clearing of the cache may cause exceptions (usually a message indicating that "xxx file does not exist") for other users during their builds. Therefore, clear the cache only when it is abnormal. Once the task is successful, remove the cache clearing command. While clearing the cache, avoid running other build tasks that use this cache.
- For security purposes, the cache clearing command can only be run in the build action. Running this command in other actions may encounter an error (for example, the clearing operation failed or the directory not found).

Table 10-3 Usage of the cache directory in each Build action

Build Action	Cache Director y (Enter an absolut e director y rather than a relative director y starting with ./.)	Cache Usage	Cache Clearing Command
Build with Maven	/ reposito ry/local/ maven	For details about the GUI configuration, see Building with Maven.	The cache clearing command follows this format: rm -rf / repository/local/maven/ {groupld}! {artifactId}! {version}. Enter the parameters for groupld, artifactId, and version based on the dependency coordinates. The periods in groupld are automatically converted into directory separators (/) and form a directory structure. For example, assume that the dependency coordinates are as follows: <dependency> <groupld>com.codearts.java</groupld> <artifactid>demo</artifactid> <version>1.0-SNAPSHOT</version> </dependency> The command for clearing the dependency is rm -rf / repository/local/maven/com/codearts/java/demo/1.0-SNAPSHOT.
Build with npm	/ npmcac he	Input the following build command: npm config set cache / npmcache.	npm cache cleanforce
Build with Grunt	/ npmcac he	Input the following build command: npm config set cache / npmcache.	npm cache cleanforce

Build Action	Cache Director y (Enter an absolut e director y rather than a relative director y starting with ./.)	Cache Usage	Cache Clearing Command
Build with Gulp	/ npmcac he	Input the following build command: npm config set cache / npmcache.	npm cache cleanforce
Build Android Quick App	/ npmcac he	Input the following build command: npm config set cache / npmcache.	npm cache cleanforce
Build with Yarn	/ npmcac he	Input the following build command: yarn config set cachefolder /npmcache.	yarn cache clean
Build with Gradle (Only for the Gradle wrapper version)	./gradle/ wrapper	Input the following build command: cp / cache/android/wrapper/gradle-wrapper.jar ./gradle/wrapper/gradle-wrapper.jar.	rm -rf ./gradle/wrapper/
Build with Android (Only for the Gradle wrapper version)	./gradle/ wrapper	Input the following build command: cp / cache/android/ wrapper/gradle-wrapper.jar ./gradle/wrapper.jar.	rm -rf ./gradle/wrapper/

1 1 Old User Guide

11.1 Signing Android APK

Sign an APK with apksigner.

Build on GUI

 Add Sign Android APK after Build with Android, when configuring build actions.

The parameters are described in the following table.

Parameter	Description
Action Name	Assign a custom name to the build action.
APK Location	Location of the APK file to be signed generated after Android building. Regular expressions are supported. For example, build/bin/*.apk can be used to match the built APK package.
Keystore File	Used for signing. You can create the file by referring to Generating Keystore Signature Files . Select one from those uploaded on the file management page.
Keystore Password	Keystore password.
Alias	Alias of the keystore file.
Key Password	Password of the key.
apksigner CLI	Custom signature parameter. By default,verbose is added to display the signature details.

2. Check whether the signing is successful.

After the configuration is complete, run the build task. After the task is executed successfully, view the build log. If "Result: Signed" is displayed in the Android APK signature log, the signing is successful.

Build with Code

```
version: 2.0 # The value must be 2.0.
steps:
BUILD:
- android_sign:
    inputs:
    file_path: build/bin/*.apk
        keystore_file: androidapk.jks
        keystore_password: xxxxxx
        alias: keyalias
        key_password: xxxxxx
        apksigner_commond: --verbose
```

Parameter	Туре	Description	Man dato ry	Default Value
file_path	String	Directory of the APK that needed to be signed	Yes	None
keystore_file	String	Keystore file name	Yes	None
keystore_pa ssword	String	Keystore file password	No	None
alias	String	Alias	Yes	None
key_passwor d	String	Password	No	None
apksigner_c ommond	String	apksigner command	Yes	None

11.2 Managing Files

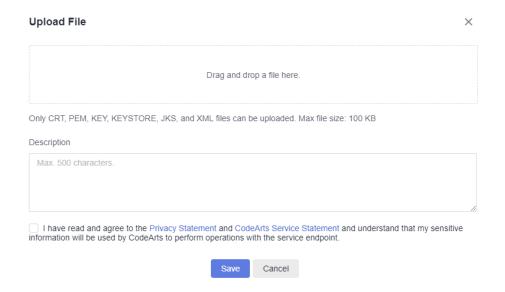
CodeArts Build stores your **Android APK signature files** and **settings.xml** files of **Maven builds**, and helps you manage these files. For example, you can create, edit, and delete these files, and modify users' permissions on them.

Constraints

- The maximum file size is 100 KB.
- The file type must be .xml, .key, .keystore, .jks, .crt, or .pem.
- A maximum of 20 files can be uploaded.

Uploading a File

- 1. Access the CodeArts Build homepage.
- 2. Click More and select Files.
- 3. Click Upload File.
- 4. In the displayed dialog box, select a file, add a description, select the check box to agree to the statements, and click **Save**.



Managing Files

After uploading a file, you can edit, download, and delete it, and configure file operation permissions for other users.

- Enter a keyword in the search box to search for a file.
- Click in the Operation column to modify the file name and specify whether to allow all members of your account to use the file in CodeArts Build.
- Click in the Operation column to download the file.
- Click *** in the Operation column and select Delete from the drop-down list.
 Confirm the deletion as prompted.
- Click *** in the Operation column and select Modify Permissions from the drop-down list. In the displayed dialog box, configure file operation permissions for the user.



Table 11-1 Roles and their permissions on files

Permission	Role with the Permission
Add users	All users in the project
View a file	File creator and users under the same account

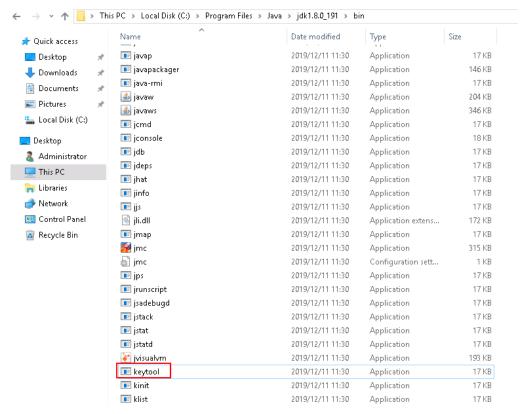
Permission	Role with the Permission
Use a file	File creator and users with the use permissions configured by the file creator
Update a file	File creator and users with the update permissions configured by the file creator
Delete a file	File creator and users with the delete permissions configured by the file creator
Modify permissions	File creator

M NOTE

By default, the creator has all permissions, which cannot be deleted or modified.

Generating Keystore Signature Files

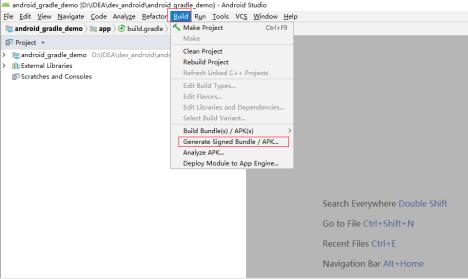
- Using Keytool in JDK to Generate Signature Files
 - a. Find the JDK installation path and run keytool.exe.



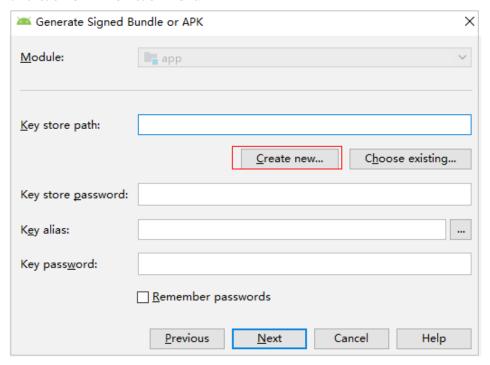
- b. Run the following command to generate a .jks file: keytool -genkeypair -storepass 123456 -alias apksign -keypass 123456 -keyalg RSA -validity 20000 -keystore D:/android.jks
- Using Android Studio to Generate Signature Files

本 android_gradle_demo [D:\IDEA\dev_android\android_gradle_demo] - Android Studio <u>File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help</u> android_gradle_demo > 📭 app > 🕙 build.gradle > 🔨 Make Project Make Project ▼ Clean Project

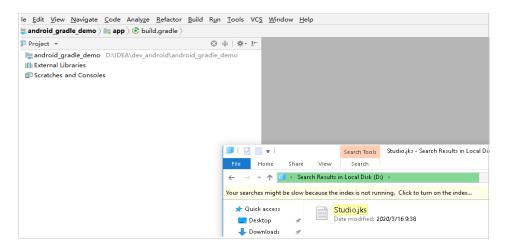
Open Android Studio and choose **Build** > **Generate Signed Bundle/APK**.



- b. Select APK and click Next.
- Click Create new. In the displayed dialog box, enter related information, and click OK. Then click Next.



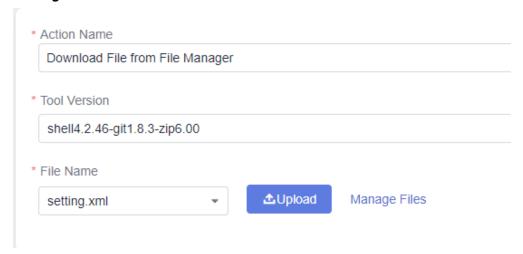
View the generated signature file.



You can upload the generated signature file to Files for unified management.

Using the settings.xml File

When creating or editing a Maven build task, add the **Download File from** File Manager action on the **Build Actions** tab page, and select the uploaded settings.xml file.



2. Add **--settings settings.xml** to the end of the default Maven build command so that you can use the **settings.xml** file for build.

11.3 Custom Build Environments

Background

If the common build environments do not meet your requirements, **customize one** by adding required dependencies and tools to a base image to make a Dockerfile. Then you can **use the custom environment** for your build.

Base Images

CodeArts Build uses CentOS 7 and Ubuntu 18 as the base images, which are provided with multiple common environment tools. You can configure custom environments as required.

The built-in environment tools include:

JDK 1.8, Maven, Git, Ant, zip, unzip, GCC, CMake, and Make.

Procedure

- Step 1 Access the CodeArts Build homepage.
- **Step 2** In the upper right corner of the CodeArts Build homepage, click **More** and select **Custom Build Environments** from the drop-down list.
- **Step 3** On the **Custom Build Environments** page, click a base image to download the Dockerfile template.

CentOS 7-based ARM Base Image This base image is based on CentOS 7 and is used for ARM builds. It has the following tools installed: OpenJDK 1.8.0_40, Maven 3.5.3, Ant 1.10.3, Git, Wget, zip, unzip, bzip2, GCC, Make, and CMake. (Aug 17, 2021 00:00:00 GMT+08:00 Ubuntu 18-based ARM Base Image This base image is based on Ubuntu 18 and is used for ARM builds. It has the following tools installed: OpenJDK 1.8.0_40, Maven 3.5.3, Ant 1.10.3, Git, Wget, zip, unzip, bzip2, GCC, Make, and CMake. (Aug 17, 2021 00:00:00 GMT+08:00

Step 4 Edit the downloaded Dockerfile.

You can add other dependencies and tools required by the project to customize the Dockerfile. The following figure shows an example of adding JDK and Maven tools.

RUN yum install -y java-1.8.0-openjdk.x86_64 RUN yum install -y maven RUN echo 'hello world!' RUN yum clean all

----End

11.4 Custom Templates

Build template selection: If the preset build templates cannot meet your build requirements, you can customize a build template.

- **Step 1** Log in to the CodeArts Build homepage.
- **Step 2** Select a build task from the list and click the task name. The **Build History** page is displayed.

□ NOTE

If no task exists in the list, create a build task on the GUI.

Step 3 Click in the upper right corner. Select Make Template from the drop-down list.

Ⅲ NOTE

A build task that uses any private parameters cannot be saved as a template. For details about how to set build parameters, see **Configuring Parameters**.

- **Step 4** Enter the template name and description, and click **Save**.
- **Step 5** Click the username in the upper right corner, and select **All Account Settings** from the drop-down list.
- **Step 6** In the navigation pane, choose **Build** > **Templates**. The saved template is displayed in the list.

You can perform the following operations on saved templates.

Table 11-2 Managing custom templates

Operation	Description
Favorite a template	Click to add the template to your favorites.
Delete a template	Click . In the displayed dialog box, click Yes to delete the template.

----End

11.5 Editing, Deleting, Cloning, Favoriting, and Stopping a Build Task

Ensure you have the required permissions before you perform operations on build tasks.

Editing a Build Task

- 1. Log in to the CodeArts Build homepage.
- 2. Search for the target build task.
- 3. In the row of the target build task, click --- and select **Edit** from the drop-down list.
 - On the Basic Information tab page, configure the task name, code source, code repository, branch, and task description.
 - On the **Build Actions** tab page, configure build actions and parameters.

- On the Parameters tab page, customize parameters for running the build task
- On the Schedule tab page, configure continuous integration (the triggering event) and scheduled execution.
- On the Change History tab page, view the change history of the build task.
- On the **Permissions** tab page, configure permissions for different roles.
- On the Notifications tab page, configure notifications for different types of events (including Build succeeded, Build failed, Task deleted, Task configurations updated, and Task disabled).
- 4. Edit the information on a tab page, and click **Save and Execute** > **Save**.

Deleting the Build Task

- 1. Search for the target build task.
- 2. Click ••• in the row of the build task and choose **Delete** from the drop-down list. Exercise caution when performing this operation.

You can view the deleted build task in the recycle bin.

Cloning the Build Task

- 1. Search for the target build task.
- 2. Click ••• in the row of the build task and select **Clone** from the drop-down list.
- 3. On the displayed page, modify the task information as required and click **Clone**.

Cloning a task will duplicate all of its permissions. The new task has identical access control settings as the original.

Favoriting the Build Task

- 1. Search for the target build task.
- 2. Move the cursor to the row of the build task and click $\stackrel{\bullet}{\Omega}$. If the color of the icon changes, the task is successfully favorited.
- 3. (Optional) Click 📩 to unfavorite the task.

- After you favorite a build task, the task is displayed on the top of the task list when you refresh the page or access the task list next time. If you favorite many build tasks, the tasks are sorted by task creation time in descending order.
- If you favorite a task that is not created by yourself, you can obtain the corresponding notification based on the notification event type set for the task.

Stopping a Build Task

1. Search for the target build task.

- 2. Click the name of a running build task. The **Build History** page is displayed.
- 3. Click the **Build ID**.
- 4. On the displayed page, click **Stop** in the upper right corner.