CodeArts Build

Getting Started

Issue   01
Date   2024-11-30

Security Declaration

Vulnerability

Huawei's regulations on product vulnerability management are subject to the *Vul. Response Process.* For details about this process, visit the following web page:
https://www.huawei.com/en/psirt/vul-response-process
For vulnerability information, enterprise customers can visit the following web page:
https://securitybulletin.huawei.com/enterprise/en/security-advisory

Contents

1 Building with Ant and Uploading the Package to a Release Repo (x86, Preset Image, GUI)

In this section, you use CodeArts Build to build a project with Ant on a x86 server and upload the resulting software package to a release repo. These steps will be carried out through a graphical user interface (GUI).

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:

    a. Visit **Huawei Cloud official website**.

    b. Click **Sign Up** and create your account as instructed.

    Once your account is created, the system automatically redirects you to your personal information page.

    c. Complete individual or enterprise real-name authentication. For details, see **Real-Name Authentication**.

- You have enabled CodeArts Free Edition. If not, enable it by referring to **Purchasing a CodeArts Package**.

Creating a CodeArts Repo Repository

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 Click **Access Service**. The homepage of CodeArts Build is displayed. On the navigation bar, click **Homepage**.

Step 4 Click **Create Project**, and select the **Scrum** template. Set the project name to **Scrum01** and keep the default values for other parameters. Click **OK** to access the project.

Step 5 In the navigation pane, choose **Code** > **Repo**.

Step 6 On the displayed page, click **New Repository**. Select **Template**, and click **Next**.

Step 7 You will then be redirected to the **Create Template Repository** page. In the **Select Repository Template** stage, search for **Java Ant Demo** and select the template. Click **Next**.

Step 8 Set **Repository Name** to **Repo01** and keep the default values for other parameters. Click **OK**.

----End

Creating a Build Task

Step 1 In the navigation pane, choose **CICD** > **Build**.

Step 2 Click **Create Task**. The **Basic Information** page is displayed.

Step 3 Configure the following parameters and click **Next**.

Table 1-1 Basic information

Parameter	Description
Name	Assign a custom name to the build task, for example, **BuildTask01**.
Code Source	Select **Repo** to pull the code hosted in a CodeArts Repo repository for your build.
Repository	Select **Repo01**, the code repository created in section **"Creating a CodeArts Repo Repository"**.
Default Branch	Keep the default value **master**.
Description	Optional. Enter additional information to describe the build task.

Step 4 Select the **Ant** template and click **OK** to create the build task. Then you will be redirected to the page for configuring build actions.

Step 5 On the **Build Actions** page, keep the default settings and click **Save**.

----End

Running the Build Task

Step 1 Click the name of the build task.

Step 2 On the displayed **Build History** page, click **Run** to start the build task. In the displayed dialog box, click **Confirm**.

- The page shown in **Figure 1-1** displays a successful run of the task.

Figure 1-1 Build success

- If the task fails, troubleshoot the issue by reviewing the prompt or analyzing the logs.

Figure 1-2 Build failure
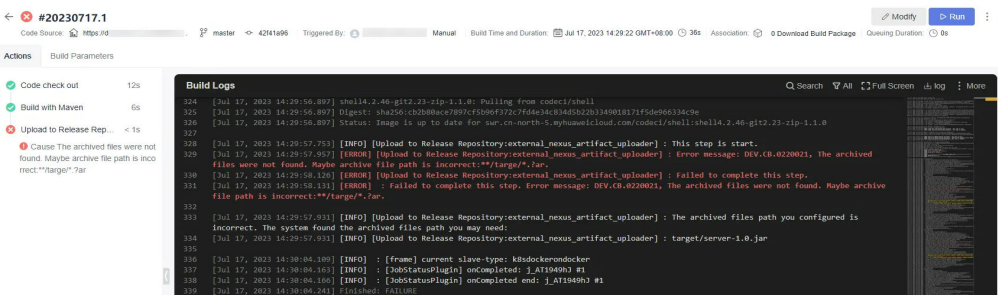
----End

Viewing Build Results

Step 1 In the navigation pane, choose **Artifact** > **Release Repos**.

Step 2 On the displayed page, find the folder that shares the same name as the build task (the name you specify when **creating a build task**), as shown in **Figure 1-3**. The software package can be found within this folder.

Figure 1-3 Checking the software package
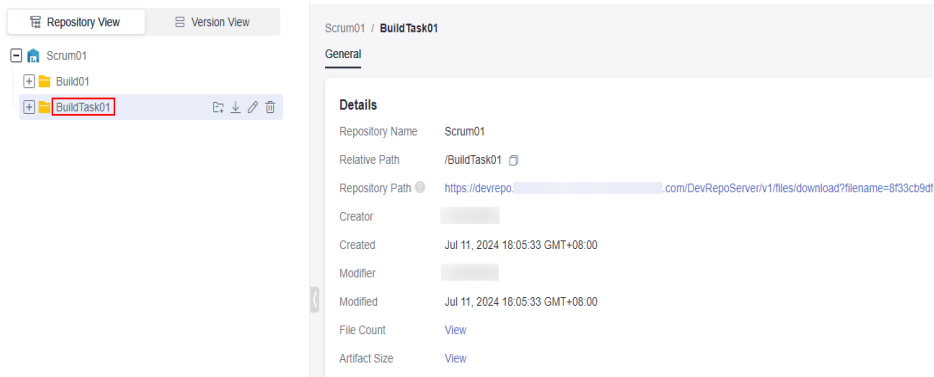
----End

Clearing Resources

To avoid unnecessary charges, you are advised to release the following resources once your builds are finished.

- In CodeArts Repo: Delete code repositories.
- In the release repo: Delete software packages and **clear the recycle bin** by referring to *CodeArts Artifact User Guide* > "Release Repo (New Version)" > "Managing the Recycle Bin".

NOTICE

Released resources cannot be recovered. Exercise caution when performing these operations.

---

2 Building with CMake and Uploading the Package to a Release Repo (Arm, Preset Image, GUI)

In this section, you use CodeArts Build to build a project with CMake in an Arm environment and upload the resulting software package to a release repo.

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:

    a. Visit **Huawei Cloud official website**.

    b. Click **Sign Up** and create your account as instructed.

       Once your account is created, the system automatically redirects you to your personal information page.

    c. Complete individual or enterprise real-name authentication. For details, see **Real-Name Authentication**.

- You have enabled CodeArts Free Edition. If not, enable it by referring to **Purchasing a CodeArts Package**.

Creating a CodeArts Repo Repository

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 Click **Access Service**. The homepage of CodeArts Build is displayed. On the navigation bar, click **Homepage**.

Step 4 Click **Create Project**, and select the **Scrum** template. Set the project name to **Scrum01** and keep the default values for other parameters. Click **OK** to access the project.

Step 5 In the navigation pane, choose **Code** > **Repo**.

Step 6 On the displayed page, click **New Repository**. Select **Template**, and click **Next**.

Step 7 You will then be redirected to the **Create Template Repository** page. In the **Select Repository Template** stage, search for **Cpp Demo** and select the template. Click **Next**.

Step 8 Set **Repository Name** to **Repo01** and keep the default values for other parameters. Click **OK**.

----End

Creating a Build Task

Step 1 In the navigation pane, choose **CICD** > **Build**.

Step 2 Click **Create Task**. The **Basic Information** page is displayed.

Step 3 Configure the following parameters and click **Next**.

Table 2-1 Basic information

Parameter	Description
Name	Assign a custom name to the build task, for example, **BuildTask01**.
Code Source	Select **Repo**.
Repository	Select **Repo01**, the code repository created in section **"Creating a CodeArts Repo Repository"**.
Default Branch	Keep the default value **master**.
Description	Optional. Enter additional information to describe the build task.

Step 4 Select the **CMake** template.

Step 5 Click **OK**. On the displayed **Build Actions** page, set **Environment** to **Kunpeng (Arm)**, retain default settings for other build actions, and click **Save**.

----End

Running the Build Task

Step 1 Click the name of the build task.

Step 2 On the displayed **Build History** page, click **Run** to start the build task.

- The following page displays a successful run of the task.
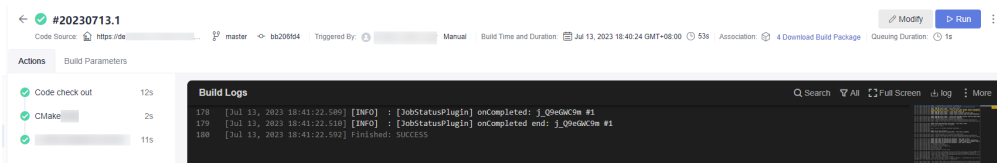
- If the task fails, troubleshoot the issue by reviewing the prompt or analyzing the logs.
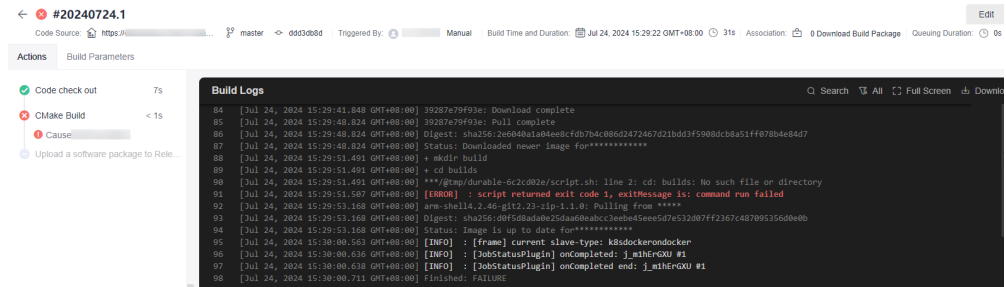
----End

Viewing Build Results

Step 1 In the navigation pane, choose **Artifact** > **Release Repos**.

Step 2 On the displayed page, find the folder that shares the same name as the build task (the name you specify when **creating a build task**). The software package can be found within this folder.

----End

CodeArts Build
Getting Started

3 Building with Maven, Uploading the Software
Package, and Pushing the Image (x86, Preset Image,
Code)

3 Building with Maven, Uploading the Software Package, and Pushing the Image (x86, Preset Image, 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** in this practice). 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, making the process traceable, recoverable, secure, and reliable.

In this section, you define your Maven build as code, build your project using the x86 server and the preset image, upload the resulting software package to the release repo, and push the image to SoftWare Repository for Container (SWR).

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:

  a. Visit **Huawei Cloud official website**.

  b. Click **Sign Up** and create your account as instructed.

      Once your account is created, the system automatically redirects you to your personal information page.

  c. Complete individual or enterprise real-name authentication. For details, see **Real-Name Authentication**.

- You have enabled CodeArts Free Edition. If not, enable it by referring to **Purchasing a CodeArts Package**.

- You have created an organization in SWR. If no organization is available, create one by referring to **Creating an Organization**.

Preparing a Project

Step 1 **Log in to the Huawei Cloud console** with your Huawei Cloud account.

CodeArts Build
Getting Started

3 Building with Maven, Uploading the Software
Package, and Pushing the Image (x86, Preset Image,
Code)

Step 2 Click ☰ in the upper left corner and choose **Developer Services** > **CodeArts Build** from the service list.

Step 3 Click **Access Service**. The homepage of CodeArts Build is displayed. On the navigation bar, click **Homepage**.

Step 4 Click **Create Project**, and select the **Scrum** template. Set the project name to **Scrum01** and keep the default values for other parameters. Click **OK** to access the project.

----End

Creating a CodeArts Repo Repository

Step 1 In the navigation pane, choose **Code** > **Repo**.

Step 2 On the displayed page, click **New Repository**. Select **Common**, and click **Next**.

Step 3 Set parameters according to **Table 3-1** and click **OK**.

Table 3-1 Creating a code repository

Parameter	Description
Repository Name	Assign a custom name to the code repository, for example, **maven_yml_build**. ● The name starts with a digit, letter, or underscore (_). ● The name can contain periods (.) and hyphens (-). ● The name cannot end with **.git**, **.atom**, or a period (.).
Description	(Optional) Enter additional information to describe the code repository. Max. 2,000 characters.
.gitignore Programming Language	Select the appropriate programming language, such as **Java**, for the **.gitignore** file.
Initial Settings	Select all. ● **Generate README**: Select this option to create a README file where you can add details about the project's architecture and purpose, similar to a repository-wide comment. ● **Automatically create check task (free of charge)**: Select this option to auto-generate a code check task for the repository upon creation. The check task will appear in the check task list.
Visibility	Select **Private** to make the repository visible only to its members. These members can access the repository and commit code.

----End

CodeArts Build
Getting Started

3 Building with Maven, Uploading the Software
Package, and Pushing the Image (x86, Preset Image,
Code)

Creating a build.yml File

Step 1 In the navigation pane, choose **Code** > **Repo**.

Step 2 Click the name of the code repository you created (see **Creating a CodeArts Repo Repository**).

Step 3 Click **Create** and select **Create Directory** from the drop-down list, as shown in **Figure 3-1**.

Figure 3-1 Creating a directory
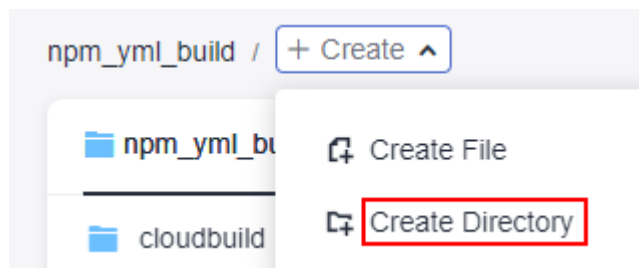
Step 4 Set parameters according to **Table 3-2** and click **OK**.

Table 3-2 Creating a directory

Parameter	Description
Directory Name	Assign a custom name to the code directory, for example, **.cloudbuild**. Use 1 to 100 characters, including letters, digits, slashes (/), underscores (_), hyphens (-), and periods (.).
Commit Message	Describe the files within the directory. Use 1 to 2,000 characters.

Step 5 Click the name of the directory created in **Step 4**.

Step 6 Click **Create** and select **Create File** from the drop-down list, as shown in **Figure 3-2**.

Figure 3-2 Creating a file
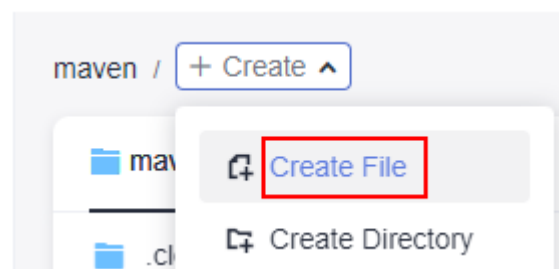
Step 7 Name the file **build.yml** and copy the following code to the file:

```
# The YAML provided is a default template that can be edited as needed.
---
version: 2.0
steps:
```

CodeArts Build
Getting Started

3 Building with Maven, Uploading the Software
Package, and Pushing the Image (x86, Preset Image,
Code)

```
BUILD:
- maven:
    image: cloudbuild@maven3.5.3-jdk8-open
    inputs:
      settings:
        public_repos:
          - https://mirrors.huawei.com/maven
      cache: true # Determine whether to enable caching.
      command: mvn package -Dmaven.test.failure.ignore=true -U -e -X -B
- upload_artifact:
    inputs:
      path: "*/target/*.?ar"
- build_image:
    inputs:
      organization: codeci_gray # Organization name
      image_name: maven_demo # Image name
      image_tag: 1.0 # Image tag
      dockerfile_path: ./Dockerfile
```

Step 8 Click **OK**.

----End

Creating a Java File

Step 1 Create a directory named **src/main/java** by referring to **Step 4**.

Step 2 Create a file named **HelloWorld.java** in the **src/main/java** directory by referring to **Step 6** and **Step 7**. The code in the file is as follows:

```
/**
 * Hello world
 *
 */

public class HelloWorld {

  public static void main(String[] args) {
    System.out.println("Hello World!");
  }

}
```

Step 3 Click **OK**.

----End

Creating a Dockerfile

Step 1 In the root directory, create a file named **Dockerfile** by following **Step 6** and **Step 7**. The code in the file is as follows:

```
FROM swr.regionid.myhuaweicloud.com/codeci/special_base_image:centos7-base-1.0.2-in
MAINTAINER <devcloud@demo.com>
USER root
RUN mkdir /demo
COPY ./target/server-1.0.jar /demo/app.jar
```

Step 2 Click **OK**.

----End

Creating a pom.xml File

Step 1 In the root directory, create a file named **pom.xml** by following **Step 6** and **Step 7**. The code in the file is as follows:

CodeArts Build
Getting Started

3 Building with Maven, Uploading the Software
Package, and Pushing the Image (x86, Preset Image,
Code)

```xml
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-
v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.huawei.demo</groupId>
  <artifactId>server</artifactId>
  <packaging>jar</packaging>
  <version>1.0</version>
  <name>server</name>
  <url>http://maven.apache.org</url>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
  </dependencies>


  <build>
    <pluginManagement>
      <plugins>
        <plugin>
          <groupId>org.apache.maven.plugins</groupId>
          <artifactId>maven-jar-plugin</artifactId>
          <version>2.6</version>
          <configuration>
            <archive>
              <manifest>
                <addClasspath>true</addClasspath>
              </manifest>
              <manifestEntries>
                <Main-Class>
                  HelloWorld
                </Main-Class>
              </manifestEntries>
            </archive>
          </configuration>
        </plugin>
      </plugins>
    </pluginManagement>
  </build>
</project>
```

Step 2 Click **OK**.

----End

Creating a Build Task

Step 1 In the navigation pane, choose **CICD** > **Build**.

Step 2 Click **Create Task**. On the displayed page, set parameters according to **Table 3-3**. Then, click **Next**.

Table 3-3 Basic information

Parameter	Description
Name	Assign a custom name to the build task, for example, **maven_yml_build**.

CodeArts Build
Getting Started

3 Building with Maven, Uploading the Software
Package, and Pushing the Image (x86, Preset Image,
Code)

Parameter	Description
Code Source	Select **Repo**.
Repository	Select **Repo01**, the code repository created in section **"Creating a CodeArts Repo Repository"**.
Default Branch	Select the branch created in section **"Creating a CodeArts Repo Repository"**. Keep the default value **master**.

Step 3 Select **Blank Template** and click **OK**. The **Build Actions** page is displayed.

Step 4 Click the **Code** tab. Then you can view the imported build script, as shown in **Figure 3-3**.

Figure 3-3 Code tab
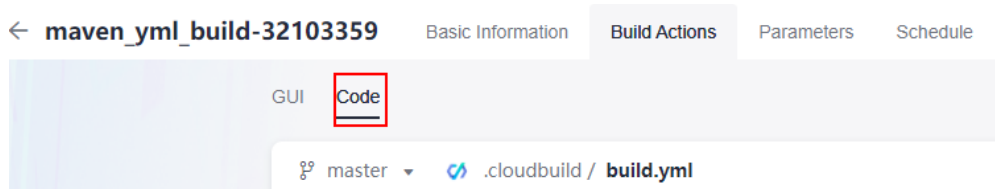
Step 5 Click **Save and Run**.

----End

Viewing and Verifying the Build Results

- Checking the uploaded the software package

  a. In the navigation pane, choose **Artifact** > **Release Repos**.

  b. On the displayed page, find the folder that shares the same name as the build task (the name you specify when **creating a build task**), as shown in **Figure 3-4**. The software package can be found within this folder.

  Figure 3-4 Checking the software package
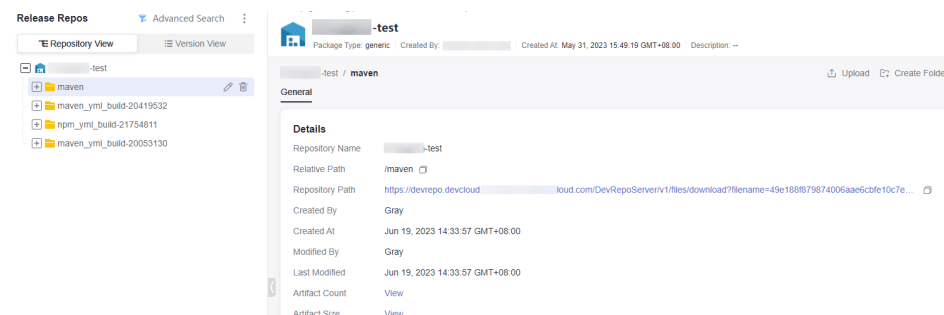
- Checking the pushed image

  a. Go to the **SWR** console.

  b. In the navigation pane, choose **My Images**. In the search box, select **Organization** as the filter field, and type the organization name

CodeArts Build
Getting Started

3 Building with Maven, Uploading the Software
Package, and Pushing the Image (x86, Preset Image,
Code)

(**codeci_gray** is used in this example) you configure when **creating a build.yml file**.

c. In the filtered results, click the image name (**maven_demo** is used in this example) you configure when **creating a build.yml file**, as shown in **Figure 3-5**.

Figure 3-5 Filtering images
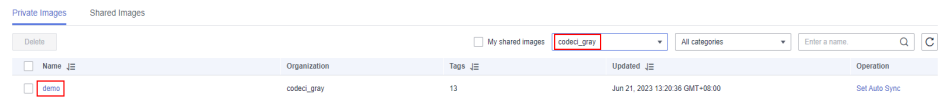
4 Common Practices for Beginners

After enabling CodeArts Build, you can use it in different service scenarios.

This section describes common build practices.

Build on GUI

Practice	Description
Creating a Custom Image via Maven Build (Built-in Executors, GUI)	Use CodeArts Build to build a Maven project, to create a Docker image from the resulting build package and push the image to SWR.