

CodeArts Build

FAQs

Issue 01
Date 2024-10-08



Copyright © Huawei Technologies Co., Ltd. 2024. 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 Technologies Co., Ltd.

Trademarks and Permissions



HUAWEI and other Huawei trademarks are trademarks 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 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, guarantees 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.

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 General	1
1.1 Can I Run a Build Task with a Specific Server or Server Configuration?	1
1.2 Why Are the Required Project Files Not Found During Builds?	1
1.3 Files Not Found During Software Package Upload	2
1.4 Why Am I Prompted that My Permissions Are Insufficient When Running a Build Task?	3
1.5 Why Am I Prompted that the Build Task Does Not Exist When Calling It Through a Pipeline?	4
1.6 Why Is My Task Stopped?	4
1.7 How Do I Migrate Common Java Projects Created with Eclipse to the Cloud?	4
1.8 Service Endpoint Did Not Exist	10
1.9 Incomplete JAR Due to Parallel Build Tasks	10
1.10 How Do I Resolve the Error Occurred When Pulling the Sub-module During My Build?	11
1.11 How Do I Fix the Submodules Checkout Error Due to a Failure to Obtain Its Revision?	12
1.12 Why Are the Submodules Not Being Pulled During My Build?	13
2 Maven Builds	15
2.1 What Can I Do If License Check Fails?	15
2.2 Failed to Upload a Package Using the maven deploy Command	15
2.3 How Do I Deal With the Missing POM File?	16
2.4 How Do I Deal With the Missing Package or Symbol?	18
2.5 Using the exec-maven-plugin Extension for Maven and NPM Hybrid Builds	20
2.6 How Do I Fix the Errors Related to References Among Multiple Child and Parent Projects?	22
2.7 How Do I Configure and Clear the Cache for a Maven Build?	24
2.8 How Do I Find the Correct Build Package Path?	26
2.9 How Do I Use jib-maven-plugin to Build a Maven Project and Create an Image?	27
2.10 Why Is the New Package Still Built from the Old Code Even After the Code Has Been Updated?	29
2.11 Why Is My Download of Maven Components Slow?	29
3 Android Builds	31
3.1 How Do I Deal With the Unstable Jcenter () Configured for a Project?	31
3.2 What Should I Do If an Error Occurs During the Lint Check and the Task Stops?	32
3.3 What Should I Do When Dependency com.android.tools.build:gradle:3.0.1 Cannot Be Downloaded?	32
3.4 Why Am I Seeing the Error Message "Javadoc generation failed"?	33
3.5 Why Am I Seeing the Error Message "Could not find method google()"?	33
3.6 What Can I Do If the Gradle Version Is Too Early?	34

3.7 How Do I Fix the Android APK Signing Failure?.....	34
4 Using Gradle for Build.....	37
4.1 Failing to Find Gradle of the Specified Version.....	37
5 npm Builds.....	39
5.1 Why Am I Seeing the Error Message "JavaScript heap out of memory"?.....	39
5.2 Why Am I Seeing the Error Message "enoent ENOENT: no such file or directory"?.....	40
5.3 Why Am I Seeing the Error Message "Module not found: Error: Can't resolve ..."?.....	41
5.4 Why Did My Build Fail Without Displaying Any Errors?.....	42
5.5 Why Am I Seeing the Error Message "npm cb() never called"?.....	42
5.6 Why Am I Seeing the Error Message "gyp ERR!"? stack Error: EACCES: permission denied.....	43
5.7 Why Am I Seeing the Error Message "eslint: error 'CLODOP' is not defined"?.....	44
5.8 How Do I Fix the node-sass Download Failure?.....	44
5.9 Why Am I Seeing the Error Message "error: Could Not Write Config File"?.....	45
5.10 Why Does It Take Long to Install Dependencies and Build?.....	46
5.11 What Should I Do When the Dependency Version Cannot Be Found?.....	47
6 Images.....	48
6.1 Failed to Create an Image Using Dockerfile.....	48
6.2 Failed to Push Images to SWR.....	50
6.3 How Do I Resolve the Image Pull Failure During a Build Task?.....	52
6.4 No Permission to Pull Images When SWR Public Images Are Used.....	53
6.5 Failed to Log In to the Image Repository.....	54
6.6 How Do I Push an Image to Other Tenants?.....	54
6.7 Pulling Docker Hub Images Times Out or Exceeds the Max. Attempts.....	57

1 General

1.1 Can I Run a Build Task with a Specific Server or Server Configuration?

No, if you use the built-in executors provided by CodeArts Build.

CodeArts Build currently assigns idle servers at random to run build tasks, so you cannot specify a particular server.

However, custom executors allow for this option. You can create an agent pool with just one executor, and use it for your build.

1.2 Why Are the Required Project Files Not Found During Builds?

Symptoms

Build tools require specific build files to complete the build process. For example, a Maven build needs a **pom.xml** file. If these files are missing, the build will fail and display an error message stating that the *xxx* project cannot find the *xxx* file. The following table shows the common error messages.

Tool	Build File	Error Message
Maven	pom.xml	The goal you specified requires a project to execute but there is no POM in this directory (). Please verify you invoked Maven from the correct directory.
Ant	build.xml	Buildfile: build.xml does not exist!
npm	package.json	npm ERR! enoent ENOENT: no such file or directory, open '/package.json'

Tool	Build File	Error Message
Yarn	package.json	error Couldn't find a package.json file in ""

Cause Analysis

The preceding errors may occur due to the following causes:

- No build file exists in the code project directory.
- The project directory is nested, and the required file is not present in the directory where the build commands are run.

Solution

- Check whether the required build file is lost in the project.
- Verify if the build file is located in the project's root directory or the specified path in the build commands. If necessary, use the **cd** command to navigate to the relevant subdirectory first.

Example:

```
cd demo-root/demo
mvn package -Dmaven.test.failure.ignore=true
```

1.3 Files Not Found During Software Package Upload

Symptoms

If an incorrect build package path is entered in the action **Upload to Release Repos** of the build task, an error is reported during task execution, and the following error information is recorded in the log:

```
[ERROR] [Upload to Release Repository:Software Package] : Error message: DEV.CB.0220021, The archived files were not found. Maybe archive file path is incorrect:**/target/bb.war.
```

Cause Analysis

During action **Upload to Release Repos**, the required files fail to be found due to incorrect package path configurations. In the preceding error message, the package path is configured as ****/target/bb.war**. Package **bb.war** cannot be found because there is no such a package in the **target** directory.

Solution

- If you are sure that there is a WAR package in the **target** directory, but not sure if the name is **bb.war**:
Change the build package path to ****/target/*.war** to match WAR packages.
- The files in the target directory cannot be determined.

Add **ls -al target** to the end of the Shell script in the build action. All files in the **target** directory will be printed when you run the build task again. After locating the required file, reconfigure the build package path.

NOTE

- Build commands run in the **workspace** directory or its subdirectories. If the build package is not in the **workspace** directory, copy the package to the **workspace** directory, such as its subdirectory **mv /usr/bin/nginx ./**. Otherwise, the package will be lost in the next action.
- See [Uploading Software Packages to Release Repos.](#)

1.4 Why Am I Prompted that My Permissions Are Insufficient When Running a Build Task?

Symptoms

A build task fails and an error message appears indicating that you lack the necessary permissions.

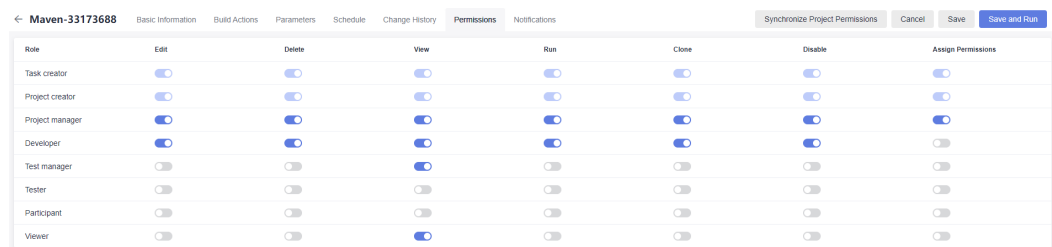
Cause Analysis

If you do not know your role or if it has been changed, you may lack the required permissions to perform builds. This means you will not be able to perform any operations on the task.

Solution

Step 1 Contact the task administrator (the task creator or project creator) to configure the task operation permissions.

Step 2 The administrator goes to the **Permissions** tab page of the task and enables the required operation permissions.



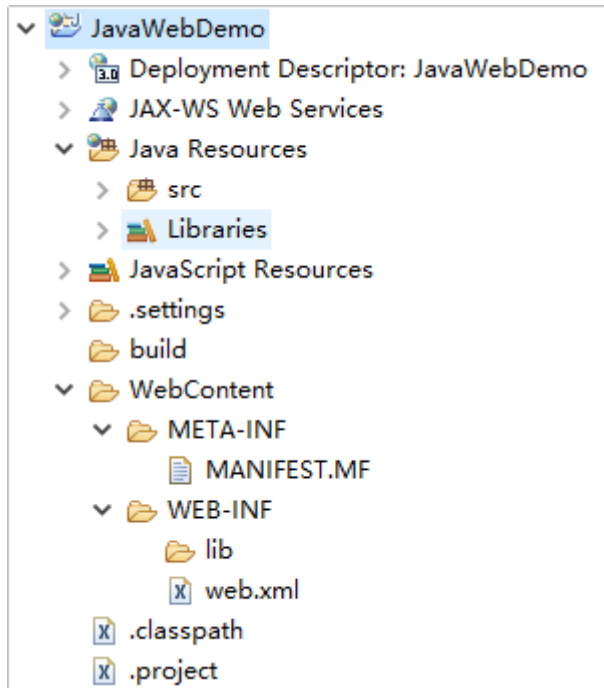
Role	Edit	Delete	View	Run	Close	Disable	Assign Permissions
Task creator	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Project creator	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Project manager	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Developer	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Test manager	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Tester	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Participant	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Viewer	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

----End

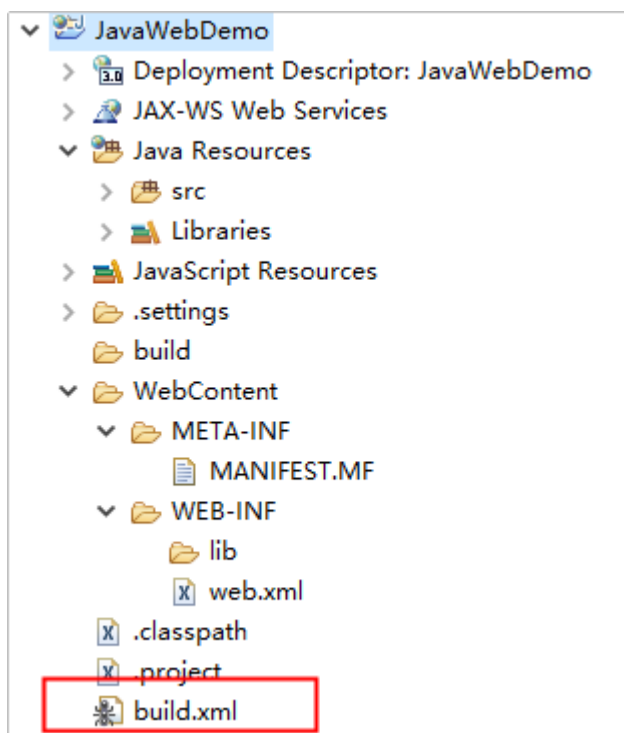
Solution

Step 1 Create a **build.xml** file in the project.

The following figure shows the directory structure of a web project demo created using Eclipse.



In the root directory, create a **build.xml** file. The updated directory structure is shown in the following figure.



The following provides a detailed description of the **build.xml** file, followed by a complete example of a **build.xml** file. You only need to change the attributes in the example to the actual value of the project.

1. Defining attributes

- Project name

```
<property name="project.name" value="JavaWebTest" />
```

- Package name: Name of the WAR file generated during packaging. The **project.name** is used for the name of the project. So the name of the generated WAR package is **project.name.war**.

```
<property name="package.name" value="${project.name}.war" />
```

- WAR package output path: When uploading a software package, use this path and package name as the build package path. The value is the path, which can be customized.

```
<property name="dist.war.dir" value="./targets" />
```

Note: If the ./targets directory does not exist, create one in the [init](#) step.

- Source code (*.java) path. The value indicates the path where the Java code of a project is stored.

```
<property name="src.dir" value="src" />
```

- **WebContent** directory in the source code. The value indicates the path for storing the **WebContent** code of a project.

```
<property name="webcontent.dir" value="./WebContent" />
```

- **WEB-INF** directory in the source code. The value indicates the path for storing the **WEB-INF** code of a project. Generally, the **WEB-INF** code is stored in the **WebContent** directory. Therefore, the **WebContent** directory is referenced.

```
<property name="webcontent.webinf.dir" value="${webcontent.dir}/WEB-INF" />
```

- CLASS file output directory: Generally, the compiled CLASS files are stored in the **WEB-INF** directory.

```
<property name="webcontent.webinf.classes.dir" value="${webcontent.webinf.dir}/classes" />
```

Note: If the classes directory does not exist, create one in the [init](#) step.

- **lib** path, which is the path for storing the referenced dependency package. Generally, the path is in the **WEB-INF** directory.

```
<property name="webcontent.webinf.lib.dir" value="${webcontent.webinf.dir}/lib" />
```

Note: If the lib directory does not exist, create one in the [init](#) step.

- Java version

```
<property name="source.version" value="1.8" />
```

```
<property name="target.version" value="1.8" />
```

- Path of the **web.xml** file, which does not need to be defined in non-web projects.

```
<property name="webxml.path" value="${webcontent.webinf.dir}/web.xml" />
```

2. Defining paths

- Define the **classpath** path. For example, if A references B, the compilation of **A.java** searches for the **B.class** file in this path.

```
<path id="classpath">
  <!-- JAR package of the project -->
  <fileset dir="${webcontent.webinf.lib.dir}">
    <include name="**/*.jar" />
  </fileset>
  <!-- classes file of the project -->
```

```
<path element location="${webcontent.webinf.classes.dir}" />
<!-- The web server package to be used, which can be downloaded and added. -->
<!-- JAR package of the web server -->
<!-- <fileset dir="${localWebServer.home}/lib">
    <include name="**/*.jar" />
</fileset> -->
</path>
```

3. Defining the build process

- Initialization (init), including clearing the output directory of the WAR package and creating the **classes** directory.

Attribute	Description
<delete> tag	Deletion action. The dir attribute indicates the directory to be deleted.
<mkdir> tag	Directory creation action. The dir attribute indicates the directory to be created.
<echo> tag	Printing action. The message attribute indicates the content to be printed.

In this step, create the following directories which are not included in the original project: WAR package directory (**dist.war.dir**), CLASS file directory (**webcontent.webinf.classes.dir**) for compilation, and lib directory (**webcontent.webinf.lib.dir**) for storing dependency packages.

```
<target name="init">
    <echo message="Delete the targets directory (output directory of the WAR package)" />
    <delete dir="${dist.war.dir}" />
    <echo message="Create the targets directory (output directory of the WAR package)." />
    <mkdir dir="${dist.war.dir}" />
    <echo message="Create the classes directory." />
    <!-- classes directory under the WebContent directory -->
    <mkdir dir="${webcontent.webinf.classes.dir}" />
</target>
```

- Compile the JAVA file. Use the **<javac>** tag to compile the JAVA file to **dist.classes**. The compilation procedure depends on the **classes** directory created in the init step.

Attribute	Description
depends	depends="init" indicates that the current step must be used after the init step.
srcdir	Specifies the path attribute src.dir of the defined Java code.
destdir	Specifies the classes directory that is defined to store the compiled CLASS file.
source, target	Specifies the JDK version used during compilation.

```
<target name="compile" depends="init">
    <echo message="Use the specified classpath to compile the source code and output the code to
```

```

the classes directory." />
<!-- The JDK version is 1.8. -->
<javac encoding="utf-8" listfiles="true" srcdir="${src.dir}"
  destdir="${webcontent.webinf.classes.dir}" debug="on" deprecation="false"
  optimize="true" failonerror="true" source="${source.version}" target="${target.version}">
  <classpath refid="classpath" />
</javac>
</target>

```

- Compress the compiled project into a WAR package. Use the **<delete>** tag to delete the original WAR package, and then use the **<war>** tag to package the project.

Attribute	Description
warfile	Name of the WAR package, including the package path.
webxml	Specifies the path of the web.xml file.
<fileset> subtag	Specifies the WebContent path.

```

<target name="war" depends="compile" description="Package a project into a WAR package">
  <echo message="Generate a WAR package" />
  <delete file="${dist.war.dir}/${package.name}" />
  <war warfile="${dist.war.dir}/${package.name}" webxml="${webxml.path}">
    <fileset dir="${webcontent.dir}">
      </fileset>
  </war>
</target>

```

NOTE

If you want to pack a JAR package, use the **<jar>** tag instead of the **<war>** tag. The following is an example:

- The **jarfile** attribute is similar to the WAR package attribute. It is the path for storing the packaged JAR package and needs to be defined in the attribute definition phase. The **basedir** attribute is the directory of the compiled **.class** directory, that is, the **webcontent.webinf.classes.dir** attribute defined in the WAR package.
- The JAR package does not require the **webxml** attribute.
- If the JAR package to be packed is an executable JAR package that needs to be executed using **java -jar**, you need to define the manifest attribute. If the JAR package is only a functional one, the depended JAR package is not required.

Main-Class specifies the class to which the **main** function belongs.

```

<target name="jar" depends="compile" description="make jar file">
  <!--JAR operation. jarfile specifies the path for storing the JAR package. basedir is the directory
of the compiled CLASS file. -->
  <jar jarfile="${jarfilename}" basedir="${classes}">
    <!--Specify the manifest file for the JAR package. If the JAR package does not need to be
packed in runnable format, the manifest file is optional.-->
    <manifest>
      <!--Specify main-class-->
      <attribute name="Main-Class" value="demo.SayHello" />
    </manifest>
  </jar>
</target>

```

The following is a complete example **build.xml** for your reference. Generally, you only need to fill in the paths in the attribute definition phase based on the actual project paths.

```
<?xml version="1.0"?>
<project default="war" basedir=".">
  <echo message="pulling in property files" /> <property file="build.properties" />
  <property name="project.name" value="JavaWebDemo" />
  <property name="package.name" value="${project.name}.war" />
  <property name="dist.war.dir" value="./targets" />
  <property name="src.dir" value="src" />
  <property name="webcontent.dir" value="./WebContent" />
  <property name="webcontent.webinf.dir" value="${webcontent.dir}/WEB-INF" />
  <property name="webcontent.webinf.classes.dir" value="${webcontent.webinf.dir}/classes" />
  <property name="webcontent.webinf.lib.dir" value="${webcontent.webinf.dir}/lib" />
  <property name="source.version" value="1.8" />
  <property name="target.version" value="1.8" />
  <property name="webxml.path" value="${webcontent.webinf.dir}/web.xml" />

  <path id="classpath">
    <fileset dir="${webcontent.webinf.lib.dir}">
      <include name="**/*.jar" />
    </fileset>
    <pathelement location="${webcontent.webinf.classes.dir}" />
  </path>

  <target name="init">
    <echo message="Delete the targets directory (output directory of the WAR package)." />
    <delete dir="${dist.war.dir}" />
    <echo message="Create the targets directory (output directory of the WAR package)." />
    <mkdir dir="${dist.war.dir}" />
    <echo message="Create the classes directory." />
    <mkdir dir="${webcontent.webinf.classes.dir}" />
  </target>

  <target name="compile" depends="init">
    <echo message="Use the specified classpath to compile the source code and output the code to the
classes directory." />
    <javac encoding="utf-8" listfiles="true" srcdir="${src.dir}"
      destdir="${webcontent.webinf.classes.dir}" debug="on" deprecation="false"
      optimize="true" failonerror="true" source="${source.version}" target="${target.version}">
      <classpath refid="classpath" />
    </javac>
  </target>

  <target name="war" depends="compile" description="Package a project into a WAR package">
    <echo message="Generate a WAR package" />
    <delete file="${dist.war.dir}/${package.name}" />
    <war warfile="${dist.war.dir}/${package.name}" webxml="${webxml.path}">
      <fileset dir="${webcontent.dir}">
        </fileset>
    </war>
  </target>
</project>
```

Step 2 Submit the modified **build.xml** file to the code repository and create an **Ant** build task.

Upload the software package to the build package path in the software release repos. Enter the WAR package output path and package name in the format described in the **build.xml** file.

Step 3 Save and run the build task. After the build is successful, you can view the compiled WAR package in the release repo.

----End

1.8 Service Endpoint Did Not Exist

Symptoms

A build task fails to be executed, and the log indicates that the corresponding service extension point does not exist.

Cause Analysis

The service endpoint data is lost. Build tasks associated with this service endpoint will fail.

Solution

Create a service endpoint and associate it with the build task. The following procedure uses Git as an example.

1. On the navigation bar, choose **Settings > General > Service Endpoints**.
2. Create a Git service endpoint.
3. Return to the build task that fails to be executed, edit the task, and re-associate the new Git service extension on the source code tab.
4. Run the build task again.

1.9 Incomplete JAR Due to Parallel Build Tasks

Symptoms

If the build environment is abnormal or the build mode is improper, the generated JAR package may be missing, but the build result is successful. As a result, the problem is difficult to locate.

- Prerequisites: Project A depends on project B. Projects A and B are built concurrently. Dependencies A and B are uploaded at the same time. This scenario occurs when different users run different build tasks together or the build tasks are configured to run in parallel in the pipeline.
- Build results: Both build task A and build task B succeeded.
- Problem: Artifact B is normal, but artifact A is occasionally incomplete.

Cause Analysis

If project A depends on project B and projects A and B are built at the same time, there is a possibility that when A starts to download dependency B, B is still being uploaded. As a result, A cannot obtain the full content of artifact B.

Solution

- Step 1** Find out all self-developed projects that A depends on: B1, B2, ... Bn.

- Step 2** Check the pipeline to see whether project A and project Bn are configured to run in parallel.
- Step 3** If they run in parallel, edit the pipeline configurations to make A and B run in serial.
- Step 4** If they do not run in serial, check the build history or build time of A and B to confirm whether they had run concurrently.
- End

1.10 How Do I Resolve the Error Occurred When Pulling the Sub-module During My Build?

Symptoms

The following error message is displayed during a build process.

```
First time build. Skipping changelog.
git remote # timeout=10
git submodule init # timeout=10
git submodule sync # timeout=10
git config --get remote.origin.url # timeout=10
git submodule init # timeout=10
git config -f .gitmodules --get-regexp ^submodule\.(.+)\.url # timeout=10
git config --get submodule.mavenSubTest19114.url # timeout=10
git remote # timeout=10
git config --get remote.origin.url # timeout=10
git config -f .gitmodules --get submodule.mavenSubTest19114.path # timeout=10
git submodule update --init --recursive --remote mavenSubTest19114
ERROR: Command "git submodule update --init --recursive --remote mavenSubTest19114" returned status code 1:
stdout: Cloning into 'mavenSubTest19114'...

Error:
Authorized users only. All activities may be monitored and reported.
Devcloud: The project you were looking for could not be found.
ERROR: Could not read from remote repository.

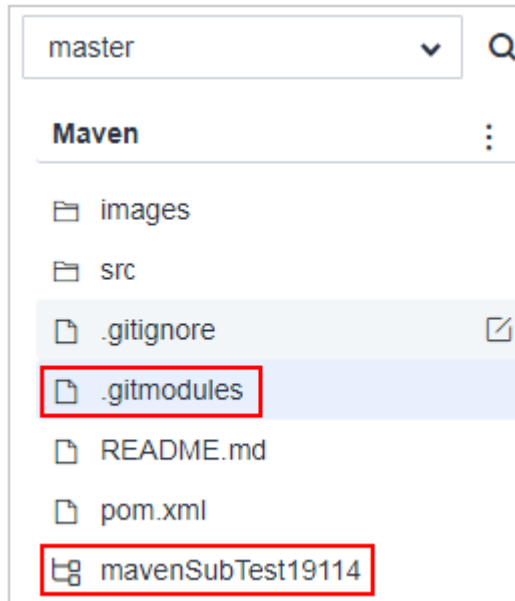
Please make sure you have the correct access rights
and the repository exists.
Clone of 'git@codehub.*****.i.com:Demo00003/mavenSubTest19114a.git' into submodule path 'mavenSubTest19114' failed
[pluginFrame] task run failed, errorMessage: Could not perform submodule update
```

Cause Analysis

If you get the **Could not read from remote repository** error when using Git to pull the sub-module from CodeArts Repo, it could be due to either insufficient permissions or incorrect configuration of the **.gitmodules** file.

Solution

- Step 1** Check if the deploy key is synchronized. Go to the main code repository and choose **Settings > Submodules**. If the deploy key is not synchronized, click the synchronization button to synchronize the deploy key and then run the build task again.
- Step 2** If the deploy key has been synchronized in **Step 1**, the problem may be with the **.gitmodules** file in the main repository, specifically with the submodule **mavenSubTest19114**. Check if the **.gitmodules** file exists and if it is configured correctly.



Step 3 Change the submodule configuration in the **.gitmodules** file to **mavenSubTest19114a.git**.

```
.gitmodules      : 138B
1  [submodule "mavenSubTest19114"]
2      path = mavenSubTest19114
3      url = git@codehub.█/mavenSubTest19114a.git
4
```

Step 4 Rerun the build task.

----End

1.11 How Do I Fix the Submodules Checkout Error Due to a Failure to Obtain Its Revision?

Symptoms

The error message is as follows:

```
[2019-07-02 08:29:23.179] ERROR: Command "git submodule update --init --recursive --remote asae-feign"
returned status code 1:
[2019-07-02 08:29:23.179] stdout: Cloning into 'asae-feign'...
[2019-07-02 08:29:23.179] Error: ERROR: Needed a single revision
[2019-07-02 08:29:23.179] Unable to find current origin/develop revision in submodule path 'asae-feign'
[2019-07-02 08:29:23.179]
[2019-07-02 08:29:23.202] [INTERNAL] : [pluginFrame] step run failed, errorMessage: Could not perform
submodule update
[2019-07-02 08:29:23.250] [INFO] [Code checkout]: StagePostExecute started
[2019-07-02 08:29:23.251] [INFO] [Code checkout]: StagePostExecute finished
```

Cause Analysis

The directory (**asae-feign** is used in this example.) that was checked out is incorrect. This is a Git bug.

Solution

Delete the directory from the code repository, run the **git submodule update** command again, and rerun the build task.

1.12 Why Are the Submodules Not Being Pulled During My Build?

Symptoms


When pulling code from CodeArts Repo, GitHub, or other sources, CodeArts Build is not pulling the submodules even though the **.gitmodules** file exists and the configuration is correct.

Cause Analysis

This problem typically occurs when the submodules are not set up for automatic updates.

Solution

Edit your build task. Click the **Configure Code Download** action, and toggle on **Auto Update** for submodules.



Configure Code Download

Configure the code download mode.

Specify Repository Tag or Commit ID

Do not specify Tag Commit ID

Clone Depth ⓘ

Auto Update ⓘ

Enable Git LFS

2 Maven Builds

2.1 What Can I Do If License Check Fails?

Symptoms

The error message is as follows:

```
[ERROR] Failed to execute goal org.apache.rat:apache-rat-plugin:0.12:check (rat-check) on project maven:
Too many files with unapproved license: 7 See RAT report in: /xxx/slave1/workspace/
job_4f1501a3-542c-4f3d-a2bb-8fdbd4d76678_1534924094266/target/rat.txt -&gt; [Help 1]
```

Cause Analysis

The license check of the file fails.

Solution

Configure the following parameters in the **mvn** command:
`apache-rat:check -Drat.numUnapprovedLicenses=600`

2.2 Failed to Upload a Package Using the maven deploy Command

Symptoms

When a Maven build task is executed to upload dependencies to the self-hosted repo, the following error information is recorded in logs:

```
[ERROR] Failed to execute goal org.apache.maven.plugins:maven-deploy-plugin:2.7:deploy (default-deploy)
on project javaMavenDemo: Deployment failed: repository element was not specified in the POM inside
distributionManagement element or in -DaltDeploymentRepository=id::layout::url parameter -&gt; [Help 1]
```

Cause Analysis

distributionManagement is incorrectly configured in the **pom.xml** file.

Solution

Step 1 Configure **Build with Maven**, expand **Release to Self-hosted Repos**, and select **Configure all POMs**.

▲ **Release to Self-hosted Repos** ?

Do not configure POM **Configure all POMs**

- **Do not configure POM:** private dependency packages do not need to be released to the CodeArts self-hosted repo.
- **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 CodeArts Artifact.

Step 2 In the command window, use **#** to comment out the command in line 8 and delete **#** before the command in line 18.

```
4 # -U: Check dependency updates to avoid outdated snapshots. This will affect the performance.
5 # -e -X: Print debugging information to locate build problems.
6 # -B: Run in batch mode to avoid ArrayIndexOutOfBoundsException during log printing.
7 # Package a project without performing unit tests.
8 #mvn package -Dmaven.test.skip=true -U -e -X -B
9
10 # Package a project, perform unit tests while ignoring failures, and check dependency updates.
11 # Perform unit tests and use test reports for analysis.
12 # Enable test report printing and specify the storage location.
13 #mvn package -Dmaven.test.failure.ignore=true -U -e -X -B
14
15 # Package a project and release dependencies to CodeArts Artifact.
16 # Release build results to CodeArts Artifact for other Maven projects.
17 # Release the build results to CodeArts Artifact, not CloudRelease.
18 mvn deploy -Dmaven.test.skip=true -U -e -X -B
```

Step 3 Run the build task. After the execution is successful, the dependency package is released to the self-hosted repo.

----End

2.3 How Do I Deal With the Missing POM File?

Symptoms

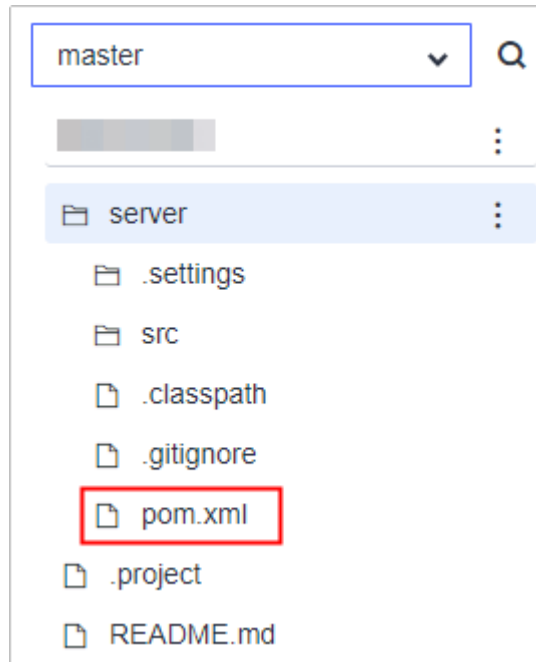
The error message is as follows:

```
[ERROR] The goal you specified requires a project to execute but there is no POM in this directory (/xxx/slavespace/slave3/workspace/job_4a1d5be4-b273-4ac8-8d5d-2ee583e71832_1544498089095). Please verify you invoked Maven from the correct directory.
```

Cause Analysis

The error information shows that the POM file cannot be found. The default build command attempts to find POM in the root directory of the source code. This error indicates that POM is not in the root directory.

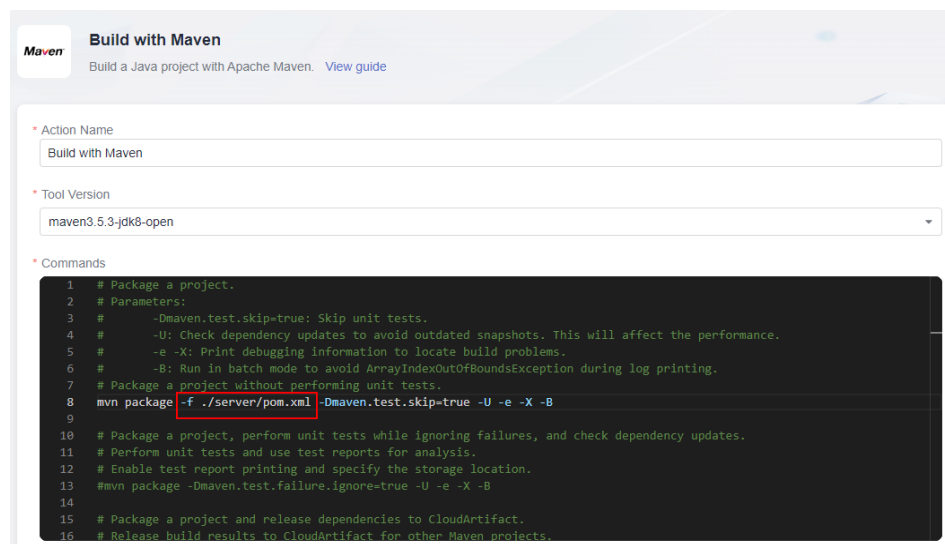
For example, the following screenshot shows that POM does not exist in the root directory of the source code. Instead, it is in the **server** directory.



Solution

Modify the default build command. The following takes the preceding source code structure as an example. Select either one of the following solutions.

- Run the **cd server** command to go to the **server** directory, and then run the **mvn** command.
- Add **-f ./server/pom.xml** to the end of the **mvn** command to specify the path of POM.



2.4 How Do I Deal With the Missing Package or Symbol?

Symptoms

When Maven build task is executed, an error message is displayed, indicating that the package or symbol cannot be found. For example:

```
com/xxx/xxx/configserver/encryptor/xxx.java:[11,40] package com.sun.jersey.api.client.config does not exist
```

Cause Analysis

According to the log, the project references the **com.sun.jersey.api.client.config** package. However, the package cannot be found in the project or all parsed dependency packages. The problem lies either in the code or the environment/component:

- Code problem: The package reference in the code is incorrect. In this case, check the code to solve the problem.
- Environment/Component problems: The dependency package is damaged or inconsistent. This type of problem is usually accompanied by successful native compilation with failed build in the cloud. The following sections provide solutions to solve different types of environment or component problems:
 - [Dependency package conflict](#)
 - [Incorrect dependency scope](#)
 - [Dependency package uploaded using GAV](#)
 - [Damaged dependency package](#)
 - Others

Dependency Package Conflict

Sometimes, multiple versions of the same dependency exist in a project due to misoperation or third-party dependency imports. This will make the used version different from the required version, and as a result the specified package cannot be found. To solve this problem, perform the following steps:

Step 1 Use either of the following methods to check the version of the dependency package:

1. Maven resolves dependency conflicts with the following strategies:
 - Nearest wins: For example, when building A, two dependencies exist: A > B > C > X 1.0 and A > D > X 2.0. Then X 2.0 will be used because the path from A to X through D is shorter.
 - First declaration wins: If two dependency versions are at the same depth, the first declaration wins.

2. Use the Maven Dependency Plugin and run the **mvn dependency:tree** command in the build task.

Step 2 If the build fails because the dependency version is not the required version, import the required dependency at the outermost layer of the POM and try again.

```
<dependencies>
  <dependency>
    <groupId></groupId>
    <artifactId></artifactId>
    <version></version>
  </dependency>
</dependencies>
```

----End

Incorrect Dependency Range

In Maven, the dependency scope attribute specifies the visibility of a dependency. If the dependency scope is incorrectly specified, the dependency will be invalid during compilation. If the package in the dependency is used in the main code of the project, a compilation error will occur. Perform the following steps to solve the problem.

- Step 1** Run the `mvn dependency:tree` command to view the dependency and dependency scope used by the project.
- Step 2** Compare the dependency scope and the location where the dependency is used in the project.

If the package in the dependency is used in the main code and the dependency must be valid during builds, the dependency scope must be one of the following options:

- compile
- provided
- system: The location of the dependency file is specified in `systemPath`. The package must exist in the specified directory.

----End

Dependency Package Uploaded Using GAV

- When uploading a dependency package to self-hosted repos using the groupId, artifactId, and version (GAV), you only need to upload the JAR package. The POM file will be generated automatically, but only contains the identifiers of the dependency. The original `<dependencies>` details will be lost.
- For example, assume that you use dependency A, which is built through project A, for building project D. Dependency A contains a third-party dependency, B. The dependency relationships are: $D > A > B$. When parsing dependency A, Maven will not be able to identify dependency B. As a result, project D cannot find contents in dependency B. In this case, perform the following steps:

- Step 1** Check the dependency tree of project D and check whether the missing content is introduced by the POM file of project A. If yes, go to the next step. If no, try other solutions.
- Step 2** Download the POM file of dependency A from the self-hosted repo of CodeArts Artifact and compare it with the POM file of project A. If the downloaded file does not contain the content introduced by B, go to the next step. Otherwise, try other solutions.

Step 3 Update the version of dependency A and upload it again using either of the following methods:

- Build project A in CodeArts Build. Run the **deploy** command to upload dependency A to the private Maven repository. You can use the pipeline for automation.
- Upload dependency A to the private Maven repository again. This time, upload the JAR packages and POM files separately.

Step 4 Build project D again.

----End

Damaged Dependency Package

If the dependency package is damaged, some files in the package may be missing. As a result, the required dependency package can be found during a build, but the CLASS file or package cannot be found. The solution varies by the package type:

- Third-party dependency package: Contact technical support.
- Self-developed package (manually uploaded to the private Maven repository): Perform the following procedure.
 - a. Download the dependency package from the private Maven repository.
 - b. Decompress the package and check whether the content is normal.
 - c. If the content of the dependency package is abnormal, perform either of the following steps:
 - If the package is provided by a third party and manually uploaded to the private Maven repository, ensure that the package file is correct and upload the package again. (Note that both the POM and JAR files must be uploaded.)
 - If the dependency package is built by yourself (on premises or in the cloud) and the code is correct, check whether **the JAR package is incomplete because multiple build tasks run in parallel**.

2.5 Using the exec-maven-plugin Extension for Maven and NPM Hybrid Builds

Symptoms

The Maven project contains front-end code and requires a build with npm. However, the provided Maven image does not contain the npm build environment.

Solution

You can use exec-maven-plugin for hybrid compilation. First configure the plug-in, and then configure the npm environment, and finally run the build task.

Step 1 Configure the POM file.

Each npm command is an `<execution>` in the `<executions>` tag. You are advised not to configure a proxy or a private npm image repository. Instead, use CodeArts Mirror. The configuration is as follows:

```
<build>
  <plugins>
    <plugin>
      <groupId>org.codehaus.mojo</groupId>
      <artifactId>exec-maven-plugin</artifactId>
      <executions>
        <execution>
          <id>exec-npm-config</id>
          <phase>prepare-package</phase>
          <goals>
            <goal>exec</goal>
          </goals>
          <configuration>
            <executable>npm</executable>
            <arguments>
              <argument>config</argument>
              <argument>set</argument>
              <argument>registry</argument>
              <argument>https://mirrors.xxcloud.com/repository/npm/</argument>
            </arguments>
            <!-- <workingDirectory>${basedir}</workingDirectory-->
          </configuration>
        </execution>

        <execution>
          <id>exec-npm-config-4</id>
          <phase>prepare-package</phase>
          <goals>
            <goal>exec</goal>
          </goals>
          <configuration>
            <executable>npm</executable>
            <arguments>
              <argument>config</argument>
              <argument>set</argument>
              <argument>sass_binary_site</argument>
              <argument>https://repo.huaweicloud.com/node-sass/</argument>
            </arguments>
            <!-- <workingDirectory>${basedir}</workingDirectory-->
          </configuration>
        </execution>

        <execution>
          <id>exec-npm-install</id>
          <phase>prepare-package</phase>
          <goals>
            <goal>exec</goal>
          </goals>
          <configuration>
            <executable>npm</executable>
            <arguments>
              <argument>install</argument>
            </arguments>
            <workingDirectory>${basedir}</workingDirectory>
          </configuration>
        </execution>

        <execution>
          <id>exec-npm-run-build</id>
          <phase>prepare-package</phase>
          <goals>
            <goal>exec</goal>
          </goals>
          <configuration>
            <executable>npm</executable>
```

```
<arguments>
  <argument>run</argument>
  <argument>build</argument>
</arguments>
<workingDirectory>${basedir}</workingDirectory>
</configuration>
</execution>
</executions>
</plugin>
</plugins>
</build>
```

Step 2 Create a Maven build task.

Step 3 Add the following npm installation and environment configuration commands to the Maven build task:

Create a directory.

```
mkdir ./node
```

Run the **curl** command to download the **Node.js** software package.

```
curl -kv https://mirrors.xxxcloud.com/nodejs/v10.15.3/node-v10.15.3-linux-x64.tar.gz -o ./node/node-v10.15.3-linux-x64.tar.gz
```

Run the **tar** command to decompress the package:

```
tar -zxvf ./node/node-v10.15.3-linux-x64.tar.gz -C ./node
```

Configure environment variables.

```
export NODEJS_HOME="${WORKSPACE}/node/node-v10.15.3-linux-x64"
export PATH="${NODEJS_HOME}/bin:${PATH}"
```

```
4 # -U: Check dependency updates to avoid outdated snapshots. This will affect the performance.
5 # -e -X: Print debugging information to locate build problems.
6 # -B: Run in batch mode to avoid ArrayIndexOutOfBoundsException during log printing.
7 # Package a project without performing unit tests.
8 mkdir ./node
9 curl -kv https://mirrors.xxxcloud.com/nodejs/v10.15.3/node-v10.15.3-linux-x64.tar.gz -o ./node/node-v10.15.3-linux-x64.tar.gz
10 tar -zxvf ./node/node-v10.15.3-linux-x64.tar.gz -C ./node
11 export NODEJS_HOME="${WORKSPACE}/node/node-v10.15.3-linux-x64"
12 export PATH="${NODEJS_HOME}/bin:${PATH}"
13
14 mvn package -Dmaven.test.skip=true -U -e -X -B
15
16 # Package a project, perform unit tests while ignoring failures, and check dependency updates.
17 # Perform unit tests and use test reports for analysis.
18 # Enable test report printing and specify the storage location.
```

Step 4 Save the settings and verify the build.

----End

2.6 How Do I Fix the Errors Related to References Among Multiple Child and Parent Projects?

Symptoms

In Maven build task, the POM file contains multiple references between child and parent projects. During task execution, the following error information is recorded in logs:

```
[ERROR] Project 'xxx.xxx:xxx1:1.0-SNAPSHOT' is duplicated in the reactor @
[2022-03-02 14:02:52.656] [ERROR] Project 'xxx.xxx:xxx2:1.0-SNAPSHOT' is duplicated in the reactor ->
[Help 1]
[2022-03-02 14:02:52.656] [ERROR]
[2022-03-02 14:02:52.656] [ERROR] To see the full stack trace of the errors, re-run Maven with the -e
```

```
switch.  
[2022-03-02 14:02:52.656] [ERROR] Re-run Maven using the -X switch to enable full debug logging.
```

Cause Analysis

In Maven, the parent can specify its modules, such as childA and childB. This is called aggregation. To compile multiple modules together, perform the following steps:

1. Add the following configurations to the parent POM:

```
<modelVersion>4.0.0</modelVersion>  
<groupId>com.demo</groupId>  
<artifactId>parent</artifactId>  
<version>1.0</version>  
<modules>  
  <module>childA</module>  
  <module>childB</module>  
</modules>
```

2. Add the following configurations to the childA POM and childB POM to specify their parent:

```
- childA:  
<modelVersion>4.0.0</modelVersion>  
<groupId>com.demo</groupId>  
<artifactId>childA</artifactId>  
<version>1.0</version>  
<parent>  
  <groupId>com.demo</groupId>  
  <artifactId>parent</artifactId>  
  <version>1.0</version>  
</parent>
```

```
- childB:  
<modelVersion>4.0.0</modelVersion>  
<groupId>com.demo</groupId>  
<artifactId>childB</artifactId>  
<version>1.0</version>  
<parent>  
  <groupId>com.demo</groupId>  
  <artifactId>parent</artifactId>  
  <version>1.0</version>  
</parent>
```

In these configurations, a parent project is specified for same-level children, childA and childB. The error shown at the beginning is displayed because a conflict occurs when childA POM references project B as its child or takes the parent project as its child.

Solution

Check the POM references of the projects. If you want project B to be a child project A, remove the reference of project B from the parent POM and point the parent tag of project B to project A.

- Parent project:

```
<modelVersion>4.0.0</modelVersion>  
<groupId>com.demo</groupId>  
<artifactId>parent</artifactId>  
<version>1.0</version>  
<modules>  
  <module>childA</module>  
</modules>
```
- Project A:

```
<modelVersion>4.0.0</modelVersion>
<groupId>com.demo</groupId>
<artifactId>childA</artifactId>
<version>1.0</version>
<parent>
  <groupId>com.demo</groupId>
  <artifactId>parent</artifactId>
  <version>1.0</version>
</parent>
<modules>
  <module>childB</module>
</modules>
```

- Project B:

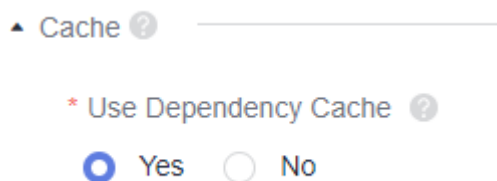
```
<modelVersion>4.0.0</modelVersion>
<groupId>com.demo</groupId>
<artifactId>childA</artifactId>
<version>1.0</version>
<parent>
  <groupId>com.demo</groupId>
  <artifactId>childA</artifactId>
  <version>1.0</version>
</parent>
```

2.7 How Do I Configure and Clear the Cache for a Maven Build?

CodeArts Build allows you to cache the dependencies in your private storage space, eliminating the need to download them repeatedly for future builds. This speeds up the build process by allowing the dependencies to be used directly from the cache.

Configuring Build Cache

By default, the option to use cache is enabled when you create a build task. You can specify whether to use the cache when configuring the **Build with Maven** action.



Clearing Cache

Under extreme conditions such as network jitter or concurrent builds, abnormal cache may occur, leading to build errors. In the following section, you will find the steps to clear the abnormal cache.

NOTICE

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.
- An accurate file path is required for clearing the cache. For example, to clear the demo 1.0.0 of vendor XXX, run the **rm -rf /path/com/xxx/demo/1.0.0** command. If the level of the directory entered in the cache clearing command is too high, the next build will be slow or the dependency will be abnormal due to network problems.
- For security purposes, the cache clearing command can only be run in the **Build with Maven** action. Running this command in other actions may encounter an error (for example, the clearing operation failed or the directory not found).

Step 1 Click  in the **Operation** column of the build task.

Step 2 Choose **Build Actions > Build with Maven**. Locate the line starting with **mvn xxx** in the commands.

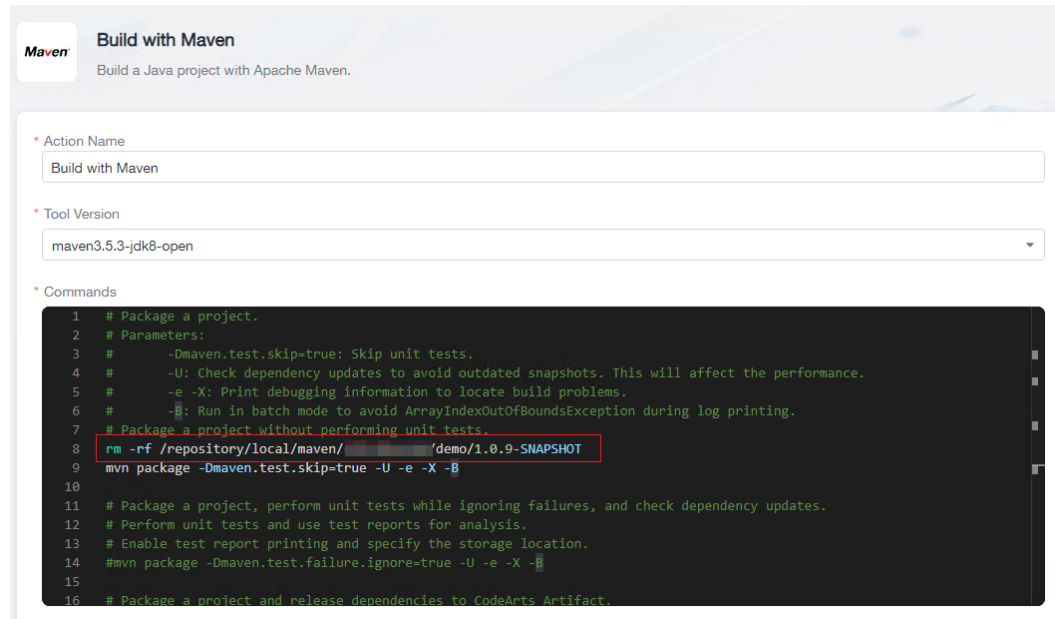
Step 3 Enter the cache clearing command in front of **mvn xxx** and click **Save**.

The cache clearing command follows this format: **rm -rf /repository/local/maven/{groupId}/{artifactId}/{version}**. Enter the parameters for **groupId**, **artifactId**, and **version** based on the dependency coordinates. The periods in **groupId** are automatically converted into directory separators (/) and form a directory structure.

For example, assume that the dependency coordinates are as follows:

```
<dependency>
<groupId>com.xxx.xxx</groupId>
<artifactId>demo</artifactId>
<version>1.0.9-SNAPSHOT</version>
</dependency>
```

The command for clearing the dependency is **rm -rf /repository/local/maven/com/xxx/xxx/demo/1.0.9-SNAPSHOT**.



Step 4 Run the build task again. After the task succeeds, edit the task again and remove the cache clearing command.

----End

2.8 How Do I Find the Correct Build Package Path?

Step 1 Create a Maven build task and add the **Upload to Release Repos** action below the Maven build action.

Step 2 Configure the build package path, enter any path, and save the configuration.

Step 3 Run the build task and search for **BUILD SUCCESS** in the log.

Find the information similar to **/target/****.war** in the preceding lines, which is the correct path of the build package.

```
25 [2020/05/07 16:17:57.927] snapshots: [enabled => true, update => always]
26 [2020/05/07 16:17:57.927] releases: [enabled => false, update => daily]
27 [2020/05/07 16:17:57.927] , id: central
28 [2020/05/07 16:17:57.927] url: https://repo.maven.apache.org/maven2
29 [2020/05/07 16:17:57.927] layout: default
30 [2020/05/07 16:17:57.927] snapshots: [enabled => false, update => daily]
31 [2020/05/07 16:17:57.927] releases: [enabled => true, update => never]
32 [2020/05/07 16:17:57.927] ]
33 [2020/05/07 16:17:57.927] [INFO] No tests to run.
34 [2020/05/07 16:17:57.927] [INFO]
35 [2020/05/07 16:17:57.927] [INFO] --- maven-jar-plugin:2.6:jar (default-jar) @ javaMavenDemo ---
36 [2020/05/07 16:17:59.048] [INFO] Building jar: ***/target/javaMavenDemo-1.0.jar
37 [2020/05/07 16:17:59.048] [INFO]
38 [2020/05/07 16:17:59.048] [INFO] BUILD SUCCESS
39 [2020/05/07 16:17:59.048] [INFO]
```

----End

2.9 How Do I Use jib-maven-plugin to Build a Maven Project and Create an Image?

Symptoms

The Maven image provided by CodeArts does not contain the Docker environment. An error is reported when an image is created using CodeArts Build and the **docker-maven-plugin** extension. For example:

```
INFO: I/O exception (java.io.IOException) caught when processing request to {}->unix://localhost:80: No such file or directory
```

This document describes how to use the **jib-maven-plugin** extension to create a TAR file with an image in a Maven environment without Docker.

Solution

1. Modify the code of the project for which you want to create an image. Find the POM file and import the extension. The content is as follows:

```
<!--Use the jib extension.-->
<plugin>
  <groupId>com.google.cloud.tools</groupId>
  <artifactId>jib-maven-plugin</artifactId>
  <version>1.3.0</version>
  <configuration>
    <!--The from node is used to set the base image, which is equivalent to the FROM
keyword in a Dockerfile.-->
    <from>
      <!--You are advised to use SWR public images for fast download and more stable
download.-->
      <image>swr.example.myxxcloud.com/xxxx/xxx:xxxx</image>
    </from>
    <to>
      <!--Image name and tag. The mvn built-in variable ${project.version} is used,
indicating the version of the current project.-->
      <image>hellojib:${project.version}</image>
    </to>
  </configuration>
</plugin>
```



```
<!--Attribute related to containers-->
<container>
  <!--JVM memory parameter-->
  <jvmFlags>
    <jvmFlag>-Xms4g</jvmFlag>
    <jvmFlag>-Xmx4g</jvmFlag>
  </jvmFlags>
  <!--Port to be exposed-->
  <ports>
    <port>8080</port>
  </ports>
</container>
</configuration>
</plugin>
```

- From tag: Set the base image, which is equivalent to the FROM keyword in a Dockerfile. You are advised to use the image in SWR because the download speed is fast and stable.
- To tag: Set the name and tag of the created image.
- Container tag: Set the container attribute, JVM memory parameter, port, and so on.
- mainClass tag: Set the main program for starting the project, that is, the **Application** class of Spring Boot.

2. Create build task and execute it.

- a. Add two build actions: **Build with Maven** and **Upload to Release Repos.** Change the default Maven build command to the following command:
mvn compile jib:buildTar -Dmaven.test.skip=true -U -e -X -B

NOTE

The Jib build tool provides four powerful functions. The build and **dockerBuild** commands cannot be used to create images because the build tool is built without the Docker environment. You can only run the **buildTar** command to create a .tar file that contains images.

- **build** allows you to create images and push them to a remote repository.
- **buildTar** provides the function of creating a .tar file that contains images.
- **dockerBuild** provides the function of creating Docker images to a local PC.
- **exportDockerContext** provides the Dockerfile creation function.

After the building is successful, the following information is displayed in the log:

```
35 [2021/01/04 15:51:33.154] [INFO]
36 [2021/01/04 15:51:33.154] [INFO] — jib-maven-plugin:1.3.0:buildTar (default-cli) @ javaMavenDemo —
37 [2021/01/04 15:51:36.709] [INFO]
38 [2021/01/04 15:51:36.709] [INFO] Containerizing application to file at ****/target/jib-image.tar...
39 [2021/01/04 15:51:36.709] [INFO] Getting base image*****
40 [2021/01/04 15:51:36.709] [INFO] Building classes layer...
41 [2021/01/04 15:51:37.528] [INFO] The base image requires auth. Trying again for*****
42 [2021/01/04 15:51:37.529] [INFO] Retrieving registry credentials for swr. .my cloud.com...
43 [2021/01/04 15:51:55.378] [INFO]
44 [2021/01/04 15:51:55.378] [INFO] Container endpoint set to [java, -Xms4g, -Xmx4g, -cp, /app/resources:/app/classes:/app/libs/, HelloWorld]
45 [2021/01/04 15:51:55.378] [INFO] Building image to tar file...
46 [2021/01/04 15:51:58.647] [INFO]
47 [2021/01/04 15:51:58.647] [INFO] Built image tarball at ***/target/jib-image.tar
48 [2021/01/04 15:51:58.647] [INFO]
49 [2021/01/04 15:51:58.647] [INFO] _____
50 [2021/01/04 15:51:58.647] [INFO] BUILD SUCCESS
51 [2021/01/04 15:51:58.647] [INFO] _____
52 [2021/01/04 15:51:58.648] [INFO] Total time: 30.526 s
53 [2021/01/04 15:51:58.648] [INFO] Finished at: 2021-01-04T07:51:57Z
```

- b. In the target directory of the Java project, the **jib-image.tar** file is generated. In addition, the task uploads the software to release repos.

3. Use the tar image.

Run the script or download command to download the .tar file from the release repo on the server where the application is to be deployed. Run the docker load command to load the image of the .tar file to the local image repository, and run the docker run command to start the image.

2.10 Why Is the New Package Still Built from the Old Code Even After the Code Has Been Updated?

Symptoms

The local code is committed to a remote repository and the code in the remote repository has been updated. However, the code in the package generated after the build is decompressed and decompiled is still the old code.

Cause Analysis

Generally, this problem occurs because the user accidentally uploads the locally compiled file (in the **target** directory) to the remote repository and does not perform the clean operation before packaging.

Solution

- Method 1: Delete the **target** directory of the remote repository.
- Method 2: Add the **clean** parameter to the packaging command. For example, if the original packaging command is **mvn package -Dmaven.test.skip=true -U -e -X -B**, the following information is displayed after the **clean** parameter is added:

```
mvn clean package -Dmaven.test.skip=true -U -e -X -B
```

2.11 Why Is My Download of Maven Components Slow?

Symptoms

Maven components are downloaded slowly during the build process.

Cause Analysis

The Maven build action generates a default **settings** file with a public network address for the mirror of a Maven repository. This can cause some Maven components to be downloaded from the public network. When public network access times out, the components are then downloaded from the private repository. As a result, downloading all Maven components may take a long time.

Solution

(Recommended) Method 1:

1. [Visit the service homepage.](#)
2. Upload the modified **settings.xml** file to the root directory of the code repository. For details, see [Customizing the settings.xml File](#).
3. On the build task list page, click the name of the target build task.
4. Click **Edit**. In the **Commands** window of the **Build with Maven** action, add **--settings settings.xml** to use the added **settings.xml** file for the Maven build.

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

Method 2:

1. [Visit the service homepage.](#)
2. On the build task list page, click the name of the target build task.
3. Click **Edit** and add the **Download File from File Manager** action before the **Build with Maven** action.
4. Click **Upload** to upload the custom **settings.xml** file. Leave the other parameters as default. For details about how to customize the **settings.xml** file, see [Customizing the settings.xml File](#).
5. In the **Commands** window of the **Build with Maven** action, add **--settings settings.xml** to use the added **settings.xml** file for the Maven build.

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

Customizing the settings.xml File

Step 1 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.

Step 2 Customize a new **settings.xml** file according to the information from the **settings.xml** file in the build logs.

----End

3 Android Builds

3.1 How Do I Deal With the Unstable Jcenter () Configured for a Project?

Symptoms

The following error information is recorded in the build task file:

```
Caused by: org.gradle.internal.resource.transport.http.HttpErrorStatusCodeException: Could not GET 'https://jcenter.bintray.com/org/apache/commons/commons-compress/1.8.1/commons-compress-1.8.1.pom'.  
Received status code 504 from server: Gateway Time-out
```

Cause Analysis

- The network is abnormal and the dependent image repository cannot be connected.
- The dependent image repository is abnormal.

Solution

You are advised to configure an open-source image site, which is stable and fast. The configuration method is as follows:

Go to the code repository on which build task depends and add the following code to the **build.gradle** file to configure the open-source image repository:

```
allprojects {  
    repositories {  
        maven {  
            url 'https://repo.xxcloud.com/repository/maven/'  
        }  
        jcenter()  
    }  
}
```

3.2 What Should I Do If an Error Occurs During the Lint Check and the Task Stops?

Symptoms

```
FAILURE: Build failed with an exception.

* What went wrong:
Execution failed for task ':app:lint'.
> Lint found errors in the project; aborting build.

Fix the issues identified by lint, or add the following to your build script to proceed with errors:
...
android {
    lintOptions {
        abortOnError false
    }
}
```

Solution

Add **-xlint** after the **gradle** command in the command line to skip the lint check.
For example:

```
/bin/bash ./gradlew assembleDebug -Dorg.gradle.daemon=false -d --stacktrace -xlint
```

or

```
gradle assembleDebug -Dorg.gradle.daemon=false -d --stacktrace --init-script /root/.gradle/init.gradle -xlint
```

3.3 What Should I Do When Dependency com.android.tools.build:gradle:3.0.1 Cannot Be Downloaded?

Symptoms

The error information is as follows:

```
Could not find com.android.tools.build:gradle:3.0.1
```

Solution

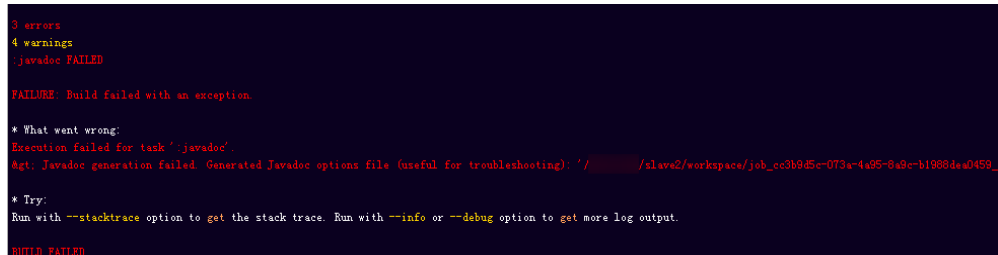
According to the log, add the **google()** repository to the **build.gradle** file in the **app** directory as shown below:

```
allprojects {
    repositories {
        google()
        jcenter()
    }
}
```

3.4 Why Am I Seeing the Error Message "Javadoc generation failed"?

Symptoms

The Gradle performs the **javadoc** check during a build. Therefore, the error **Javadoc generation failed** may be reported.



```
3 errors
4 warnings
:javadoc FAILED

FAILURE: Build failed with an exception.

* What went wrong:
Execution failed for task ':javadoc'.
> Agt. Javadoc generation failed. Generated Javadoc options file (useful for troubleshooting): '/.../slave2/workspace/job_cc3b985c-073a-4495-849--b19884e40459_1

* Try:
Run with --stacktrace option to get the stack trace. Run with --info or --debug option to get more log output.

BUILD FAILED
```

Solution

To avoid **javadoc** check, add the following configuration to Gradle in the root directory of the project:

```
allprojects {
    repositories {
        jcenter()
    }
    tasks.withType(Javadoc) {
        options.addStringOption('Xdoclint:none', '-quiet')
        options.addStringOption('encoding', 'UTF-8')
    }
}
```

3.5 Why Am I Seeing the Error Message "Could not find method google()"?

Symptoms

After the Gradle plug-in is upgraded to version 3.0, the corresponding Gradle needs to be upgraded to version 4.1. If Gradle 4.1 is not found, the following error is displayed:

```
Could not find method google() for arguments [] on repository container
```

Solution

Use Gradle v4.1 or later versions.

3.6 What Can I Do If the Gradle Version Is Too Early?

Symptoms

After a build with Android is run, the following information is displayed, indicating that the Gradle version must be 3.3 or later, but the current version is 2.10.

```
49 [2019-02-14 15:00:07.701] + /bin/bash ./gradlew assembleDebug -Dorg.gradle.daemon=false
50 [2019-02-14 15:00:14.228]
51 [2019-02-14 15:00:14.228] FAILURE: Build failed with an exception.
52 [2019-02-14 15:00:14.228]
53 [2019-02-14 15:00:14.228] * Where:
54 [2019-02-14 15:00:14.228] Build file '/devcloud/slave1/workspace/job_eceabc77-c886-4742-b69f-8351d7ce101_1550127998983/app/build.gradle' line: 1
55 [2019-02-14 15:00:14.228]
56 [2019-02-14 15:00:14.228] * What went wrong:
57 [2019-02-14 15:00:14.228] A problem occurred evaluating project ':app'.
58 [2019-02-14 15:00:14.228] Failed to apply plugin [id 'com.android.application']
59 [2019-02-14 15:00:14.228] Minimum supported Gradle version is 3.3. Current version is 2.10. If using the gradle wrapper, try editing the distributionUrl in /.../sl
60 [2019-02-14 15:00:14.228]
61 [2019-02-14 15:00:14.228] * Try:
62 [2019-02-14 15:00:14.228] Run with --stacktrace option to get the stack trace. Run with --info or --debug option to get more log output.
63 [2019-02-14 15:00:14.228]
64 [2019-02-14 15:00:14.228] BUILD FAILED
65 [2019-02-14 15:00:14.228]
```

Cause Analysis

The Gradle version in the compilation environment is too early to meet the compilation requirements.

Solution

- For a build with Gradle, select a Gradle version that meets the requirements.
- For a build with Gradlew, modify the **gradle/wrapper/gradle-wrapper.properties** file and change the version of **gradle-*. *-all.zip**.

```
gradle-wrapper.properties : 231B
1 #Mon Dec 28 10:00:20 PST 2015
2 distributionBase=GRADLE_USER_HOME
3 distributionPath=wrapper/dists
4 zipStoreBase=GRADLE_USER_HOME
5 zipStorePath=wrapper/dists
6 distributionUrl=https\://services.gradle.org/distributions/gradle-2.10-all.zip
7
```

3.7 How Do I Fix the Android APK Signing Failure?

Symptoms

A signature error is reported during a build with Android.

The error information includes **Execution failed for task ':app:validateSigningDebug** or **Execution failed for task ':app:validateSigningRelease**.


Solution


During Android builds, you are advised to use the Android APK signature build action to sign the APK. CodeArts Build provides the Android APK signature build action. The configuration method is as follows:

1. Add the action **Sign Android APK** after **Build with Android**.

Sign Android APK
Sign an Android APK with apksigner. [View the User Guide](#)

* Action Name
Sign Android APK

* APK Directory 
build/bin/*.apk

* Keystore File
android-yufa.jks  [Upload](#) [Manage Files](#)

Keystore Password

* Alias

Key Password

* apksigner CLI
--verbose

The parameters are described in the following table.

Parameter	Description
APK Directory	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	Keystore file used for signature. Click the drop-down list box to display the Keystore files uploaded on the Files page. Select a file as required.
Keystore Password	Keystore password.

Parameter	Description
Alias	Alias of the keystore file.
Key Password	Key password.
apksigner CLI	Custom signature parameter. The default value is --verbose .

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.

4 Using Gradle for Build

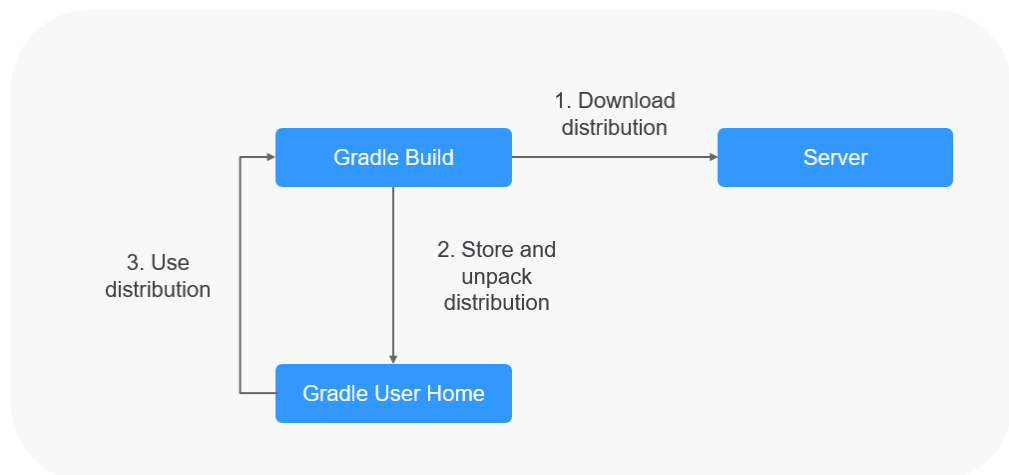
4.1 Failing to Find Gradle of the Specified Version

Symptoms

Failed to find the required Gradle version during action editing.

Cause Analysis

- If the Gradle version on which a project to be compiled depends is not in the list, you can use **gradlew(gradle wrapper)** to encapsulate Gradle commands.
- The Gradle commands are encapsulated in Gradlew. The Gradle of the specified version will be installed before the build commands are executed.
- Gradle recommends that **Wrapper** files be created in all Gradle projects to facilitate users who have not installed Gradle.



Using Gradle Wrapper

Step 1 In the local environment, go to the root directory of the code and run the **Gradle Wrapper** command. After the command is executed, the following files are added to the code repository:

- **gradlew** (Unix Shell script)
- **gradlew.bat** (Windows batch processing file)
- **gradle/wrapper/gradle-wrapper.jar** (Wrapper JAR file)
- **gradle/wrapper/gradle-wrapper.properties** (Wrapper file)

Step 2 Commit the code to the code repository.

Step 3 Modify the statement in the command line of the build task and replace **gradle** with **./gradlew**. For example, replace **gradle build** with **./gradlew build**.

----End

5 npm Builds

5.1 Why Am I Seeing the Error Message "JavaScript heap out of memory"?

Symptoms

When the npm build task is executed, the following error information is displayed in the log.

```
FATAL ERROR: CALL_AND_RETRY_LAST Allocation failed - JavaScript heap out of memory.
```

Cause Analysis

The memory used by the Node.js runtime is limited: 1.4 GB for a 64-bit system and 0.7 GB for a 32-bit system. The memory used in the current build has exceeded the default-allowed size.

Solution

Method 1: Upgrade the Node.js version.

Method 2: Set the `--max_old_space_size` or `--max_new_space_size` parameter when starting the node to adjust the memory usage limit.

```
node --max_old_space_size=1700 test.js // The unit is MB. Modifies the old memory limit.  
node --max_new_space_size=1024 test.js // The unit is KB. Modifies the new memory limit.
```

The solutions for the three major front-end frameworks are described in the following table.

Frame	Solution
Vue	<p>You only need to change the value of build in the package.json file and add the node command with parameters to the command. For example:</p> <pre>"build": "node --max_old_space_size=4096 ./node_modules/vite/bin/vite.js build"</pre> <p>or</p> <pre>"build": "node --max_old_space_size=4096 ./node_modules/@vue/cli-service/bin/vue-cli-service.js build"</pre>
React	<p>Assume that the content of the scripts field in the package.json file is as follows:</p> <pre>"scripts": { "start": "react-scripts start", "build": "react-scripts build", "test": "react-scripts test --env=jsdom", "eject": "react-scripts eject" }</pre> <p>The actual code that works in the npm run build command is react-scripts build. Open the node_modules folder in the root directory of a project, find and open the .bin directory, find and open the react-scripts file, and add max_old_space_size=4096 to #!/usr/bin/env node:</p> <pre>#!/usr/bin/env node --max_old_space_size=4096</pre>
Angular	<p>Assume that the content of the scripts field in the package.json file is as follows:</p> <pre>"scripts": { "ng": "ng", "start": "ng serve", "build": "ng build", "test": "ng test", "lint": "ng lint", "e2e": "ng e2e" }</pre> <p>Similar to React, the ng file exists in the .bin directory in the node_modules folder in the root directory of a project. Modify the first line of the file.</p> <pre>#!/usr/bin/env node --max_old_space_size=4096</pre>

5.2 Why Am I Seeing the Error Message "enoent ENOENT: no such file or directory"?

Symptoms

The error message is as follows:

```
619 [2021-12-25 16:48:45.257] npm ERR! syscall open
620 [2021-12-25 16:48:45.257] npm ERR! enoent ENOENT: no such file or directory, open '/.../slave1/workspace/job_780c5c75-1b09-4b25-a506-17730f40684d_1545727710135/package.json'
621 [2021-12-25 16:48:45.257] npm ERR! enoent This is related to npm not being able to find a file.
622 [2021-12-25 16:48:45.257] npm ERR! enoent
623 [2021-12-25 16:48:45.257]
624 [2021-12-25 16:48:45.257] npm ERR! A complete log of this run can be found in:
625 [2021-12-25 16:48:45.257] npm ERR! /root/.npm/_logs/2018-12-26T08_48_44_076Z-debug.log
626 [2021-12-25 16:48:45.257] npm ERR! script returned exit code 254
```


5.4 Why Did My Build Fail Without Displaying Any Errors?

Symptoms

Build with npm fails, but no error log is displayed. The error is shown in the following figure.

```
103 [2] [00:27:51] Using gulpfile /devcloud/slave2/workspace/job_76a183ed-f340-496f-aaf4-417c3d82d4f1_1545582435960/gulpfile.js
104 [2] [00:27:51] Starting 'build'...
105 [2] [00:27:51] - building for production...
106 At:
107 [2] [00:27:51] Sending interrupt signal to process
108 [2] [00:27:51] sh: line 1: 26 Terminated                  JENKINS_SERVER_COOKIE=$jsc /devcloud/slave2/workspace/job_76a183ed-f340-496f-aaf4-417c3d82d4f1_1545582435960
109 [2] [00:27:51] npm verb lifecycle orange@1.0.0 build: unsafe-perm in lifecycle true
110 [2] [00:27:51] npm verb lifecycle orange@1.0.0 build: PATH: /usr/local/node-v8.11.2-linux-x64/lib/node_modules/npm/node_modules/npm-lifecycle
111 [2] [00:27:51] npm verb lifecycle orange@1.0.0 build: CWD: /devcloud/slave2/workspace/job_76a183ed-f340-496f-aaf4-417c3d82d4f1_1545582435960
112 [2] [00:27:51] npm info lifecycle orange@1.0.0 build: Failed to exec build script
113 [2] [00:27:51] npm verb stack Error: orange@1.0.0 build: `cross-env NODE_ENV=production gulp build`
114 [2] [00:27:51] npm verb stack Exit status 1
```

Cause Analysis

In the build script, it has been configured that the build stops when an error occurs.

```
16 spinner.start()
17 rm(path.join(config.build.assetsRoot, config.build.assetsSubDirectory), err => {
18   if (err) throw err
19   webpack(webpackConfig, (err, stats) => {
20     spinner.stop()
21     if (err) throw err
22     process.stdout.write(stats.toString({
23       colors: true,
24       modules: false,
25       children: false, // If you are using ts-loader, setting this to true will make TypeScript errors show up during build.
26       chunks: false,
27       chunkModules: false
28     }) + '\n\n')
29
30     if (stats.hasErrors()) {
31       console.log(chalk.red(' Build failed with errors.\n'))
32       process.exit(1)
33     }
34   })
35 }
```

Solution

Check the build script and delete the configurations that may cause the sudden stops when an error occurs, such as **process.exit(1)**.

5.5 Why Am I Seeing the Error Message "npm cb() never called"?

Symptoms

When the npm build task is executed, the following error information is displayed in the log.

```
1] npm WARN deprecated opn@6.0.0: The package has been renamed to open
2] npm WARN deprecated graceful-fs@3.0.11: please upgrade to graceful-fs 4 for compatibility with current and future
3] npm WARN deprecated browserslist@1.7.7: Browserslist 2 could fail on reading Browserslist >3.0 config used in other
4] Unhandled rejection Error: Unknown system error -116: Unknown system error -116, mkdir '/npmcache/_cacache/tmp'
5] Unhandled rejection Error: Unknown system error -116: Unknown system error -116, mkdir '/npmcache/_cacache/tmp'
6] Unhandled rejection Error: Unknown system error -116: Unknown system error -116, mkdir '/npmcache/_cacache/tmp'
7] npm ERR! cb() never called!
8]
9] npm ERR! This is an error with npm itself. Please report this error at:
10] npm ERR! <https://github.com/npm/npm/issues>
11]
```

Cause Analysis

The npm cache is abnormal and needs to be cleared.

Solution

Edit the task, add the **npm cache clean -f** command before the **npm install** command, save the task, and execute the task again.

```
14 #npm install node-sass --verbose
15 # Load dependencies.
16 npm cache clean -f
17 npm install --verbose
18 # Build projects with default settings.
19 npm run build
20 #tar -zcvf demo.tar.gz ./**
```

5.6 Why Am I Seeing the Error Message "gyp ERR!"? stack Error: EACCES: permission denied

Symptoms

When the npm build task is executed, the following error information is displayed in the log.

```
gyp ERR! stack Error: EACCES: permission denied, mkdir '*****/
job_1451ba57-0c35-4daa-99c2-21425404f61c_1564043318112/saas_shop/node_modules/node-sass/.node-
gyp'
```

Cause Analysis

The current directory does not have the read and write permissions.

Solution

Edit the task, add **node-sass --unsafe-perm=true** after the **npm install** command, save the task, and execute the task again.

*** Commands**

```
14 #npm install node-sass --verbose
15 # Load dependencies.
16 npm install node-sass --unsafe-perm=true
17 # Build projects with default settings.
18 npm run build
19 #tar -zcvf demo.tar.gz ./**
```

5.7 Why Am I Seeing the Error Message "eslint: error 'CLODOP' is not defined"?

Symptoms

When the npm build task is executed, the following error information is displayed in the log.

```
Module Error (from ./node_modules/@vue/cli-plugin-eslint/node_modules/eslint-loader/index.js):
```

```
***//public/LodopFuncs.js
79:25 error 'getCLodop' is not defined no-undef
80:27 error Empty block statement no-empty
89:21 error 'CLODOP' is not defined no-undef
```

Cause Analysis

If the function in the **LodopFuncs.js** file is not defined, check the file first. If the file is normal, the error may be caused by non-compliance with the ESLint specifications.

Solution

1. Check whether the *getCLodop* function is defined in the **LodopFuncs.js** file.
2. If the file is normal, add the following command to the header of the file that fails to pass the ESLint check to ignore the ESLint check:

```
/* eslint-disable */
```

5.8 How Do I Fix the node-sass Download Failure?

Symptoms

When the npm build task is executed, the following error information is displayed in the log.

```
Downloading binary from https://github.com/sass/node-sass/releases/download/v4.14.1/linux-x64-72_binding.node
Cannot download "https://github.com/sass/node-sass/releases/download/v4.14.1/linux-x64-72_binding.node":
read ECONNRESET
```

```
...  
npm ERR! code ELIFECYCLE  
npm ERR! errno 1  
npm ERR! node-sass@4.14.1 postinstall: `node scripts/build.js`  
npm ERR! Exit status 1  
npm ERR!  
npm ERR! Failed at the node-sass@4.14.1 postinstall script.  
npm ERR! This is probably not a problem with npm. There is likely additional logging output above.
```

Cause Analysis

The image source of node-sass needs to be set separately. If the mirror source is not set, npm downloads the mirror source from GitHub by default. The network between CodeArts and GitHub is unstable, and the download may fail.

Solution

Add the following command before the default command **npm install** to use the mirror source of Huawei Cloud and perform the build again.

```
npm config set sass_binary_site https://repo.xxcloud.com/node-sass/
```

5.9 Why Am I Seeing the Error Message "error: Could Not Write Config File"?

Symptoms

When an npm build task is executed, the following error information is reported in the log: **error: could not write config file /npmcache/_cacache/tmp/git-clone-b0ba91a1/.git/config: Disk quota exceeded**

Cause Analysis

The npm cache space is insufficient and needs to be cleared.

Solution

1. Log in to CodeArts Build.
2. Select the corresponding build task, click **...** in the row of the task, and click **Edit**.
3. On the **Build Actions** page, modify the **Build with npm** action.
4. Add the **npm cache clean -f** command before the **npm install** command, save the task, and execute the task again.

* Commands

```
14 #npm install node-sass --verbose
15 # Load dependencies.
16 npm cache clean -f
17 npm install --verbose
18 # Build projects with default settings.
19 npm run build
20 #tar -zcvf demo.tar.gz ./**
```

5.10 Why Does It Take Long to Install Dependencies and Build?

Cause Analysis

The default image repository address may cause low efficiency of downstream traffic on the network side due to network problems.

Solution

Step 1 Log in to CodeArts Build.

Step 2 Select the corresponding build task, click ******* in the row of the task, and click **Edit**.

Step 3 On the **Build Actions** page, modify the **Build with npm** action.

Step 4 In the **Build with npm** action, add the following command to change the address of the npm image repository:

```
npm config set registry https://repo.xxcloud.com/repository/npm/
```

or

```
npm config set registry https://registry.npm.taobao.org
```

* Commands

```
1 export PATH=$PATH:~/.npm-global/bin
2 # Set the cache directory.
3 npm config set cache /npmcache
4 npm config set registry http://mirrors.tools. .com/npm/
```

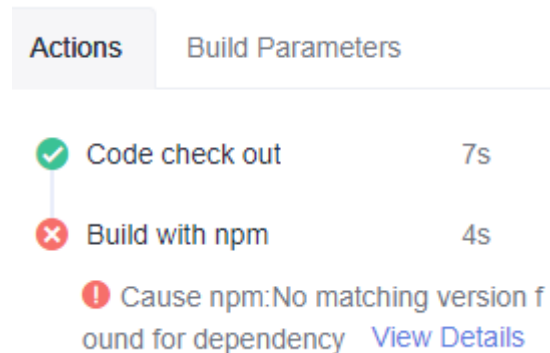
Step 5 Click **Save and Run** to execute the build task again.

----End

5.11 What Should I Do When the Dependency Version Cannot Be Found?

Symptoms

npm cannot find a matching version for the dependency. The error is displayed in the following figure.



Cause Analysis

npm cannot find a matching version for the dependency.

Solution

1. Check whether the dependency versions configured in **package.json** and **package-lock.json** are correct. If not, correct them.
2. Check whether the version exists in the image origin.

6 Images

6.1 Failed to Create an Image Using Dockerfile

When you create an image by using the action Build Image and Push to SWR or Run Docker Commands, the image may fail to be created in the docker build stage. In this case, rectify the fault by referring to the solution for each scenario.

- [Failed to Find a File by Running the COPY or ADD Command](#)
- [Failed to Pull the Base Image During Image Creation](#)
- [Failed to Execute the Command](#)

Failed to Find a File by Running the COPY or ADD Command

Symptoms

The build task contains the Build Image and Push to SWR or Run Docker Commands action. When a task is executed, the following error information is displayed in the log:

```
ADD failed: stat /var/lib/docker/tmp/docker-builder154037010/temp: no such file or directory
[ERROR] [Build Image and Push to SWR]: Error information: DEV.CB.0210043, Docker image creation failed.
COPY failed: stat /var/lib/docker/tmp/docker-builder076130522/test.txt: no such file or directory
```

Cause Analysis

The source file of the **ADD** command is **./temp**, but no temp file is available in the current directory.

Solution


Assume that the structure of the current directory is as follows:

```
+ target
- temp
- Dockerfile
```

The **target** directory contains **temp** files, and the **Dockerfile** file is at the same level as the **target** directory.

- Method 1: Change the source file of the **ADD** command to **./target/temp**.

- Method 2: Use the **target** directory as the working directory. Change the working directory for the Build Image and Push to SWR action to **target** and the Dockerfile path to **../Dockerfile**.

Working Directory 

Dockerfile Path 

Failed to Pull the Base Image During Image Creation

When you use **Dockerfile** to create an image and specified base image parameters are incorrect, the image will not be pulled. The scenarios are as follows:

- No specified images or no permission

Error Log

```
pull access denied for java1, repository does not exist or may require 'docker login'
```

Analysis and Solution

This error occurs when the specified image cannot be found in the image repository or the current user does not have the pull permission on the image.

In this example, the image **java1** specified by the **FROM java1:8ull-jdk-alpine** command cannot be found in the image repository. Therefore, this error occurs. Check and modify the image name and try again.

- No specified image tags

Error Log

```
manifest for java:8ull-jdk-alpine not found
```

Analysis and Solution

If the specified image exists in the image repository but the corresponding version or tag of the image does not exist, the error **manifest not found** is displayed. In this Dockerfile, the image **java:8ull-jdk-alpine** is specified by the **FROM java:8ull-jdk-alpine** command. The Java image exists in the image repository but does not have the corresponding version or tag (8ull-jdk-alpine). As a result, this error occurs. Check and modify the image version and try again.

Failed to Execute the Command

Symptoms

When Dockerfile is used to create an image, the following error message is displayed in the docker build stage:

```
exec user process caused "exec format error"
```

Cause Analysis

The possible causes are as follows:

- The base image used for creating an image does not match the executor. For example, the image is an Arm image, but the executor is an x86 image.
- An error occurs when the Dockerfile file content is copied from another place.

Solution

1. Check whether the image matches the executor. If the image is an x86 image, only the x86 executor can be used.
2. Perform the build again and check whether the build is successful. If the build fails, manually enter the Dockerfile and perform the build again.

6.2 Failed to Push Images to SWR

When you execute the action **Build Image and Push to SWR** or the action **Run Docker Commands**, an image may fail to be pushed due to incorrect parameters or environment problems. In this case, rectify the fault by referring to the following solutions provided for different scenarios.

- **Insufficient Permissions** (denied: you do not have the permission)
- **Number of Organizations Reaches the Upper Limit** (denied: The number of namespaces exceeds the upper limit)
- **You Have Not Logged In** (denied: You may not log in yet)
- **Authentication Failure** (denied: Authenticate Error)
- **Invalid Organization Name** (invalid reference format)
- **The Local Image Does Not Exist** (An image does not exist locally with the tag: ***)
- **Invalid Abstract** (digest invalid: Invalid digest)

Insufficient Permissions

Error Log

When an image is uploaded to SWR, the following error message is displayed:

```
denied: you do not have the permission
```

```
[ERROR] : [pluginFrame] step run failed, errorMessage: DEV.CB.0210044, Docker push failed
```

Analysis and Solution

This error indicates that the current user does not have the permission on the target organization. Check the following possible causes:

1. Edit the build task file, click Create Image and Push to SWR, and view the organization name.
2. Log in to SWR and check whether the organization exists on the organization management page.
 - The organization does not exist. Please **create an organization**. (The number of organizations cannot exceed the upper limit.)
 - If the current user does not have the editing permission for the organization or image that exists in SWR, the preceding error will occur when the image is pushed. The administrator can authorize the current user by referring to **User Permissions**.

- If the organization exists and the user has the edit permission on the organization, log in to the Identity and Access Management (IAM) console and check whether the user is in a user group with the read-only permission. If yes, remove the user.

Number of Organizations Reaches the Upper Limit

Error Log

```
denied: The number of namespaces exceeds the upper limit
```

```
[ERROR] : [pluginFrame] step run failed, errorMessage: DEV.CB.0210044, Docker push failed
```

Analysis and Solution

By default, if an unused new organization name is specified for pushing an image, SWR automatically creates an organization with the name for the current tenant. SWR limits the number of organizations that can be created by each tenant. If the number of organizations exceeds the limit, the preceding error occurs.

If this error occurs, use the administrator account (or any account with the SWR organization management permissions) to go to the [Organization Management](#) page, switch to the corresponding region, view the existing organization list, and select an existing organization or delete unnecessary organizations.

You Have Not Logged In

Error Log

```
denied: You may not login yet
```

```
[ERROR] : [pluginFrame] step run failed, errorMessage: fail to execute docker command
```

Analysis and Solution

The two possible causes are as follows:

- If you have not run the **docker login** command to log in before pushing an image, add the corresponding login command.
- The login command is executed, but the SWR address in the login command is incorrect. As a result, no error is reported but the login does not take effect. Check whether the login command is correct.

Authentication Failure

Error Log

```
Error response from daemon: Get https://swr.example.example.com/v2/: denied: Authenticate Error
```

```
[ERROR] : [pluginFrame] step run failed, errorMessage: fail to execute docker command.
```

Analysis and Solution

The possible cause is that the account or password in the login command is incorrect or the temporary login information has expired. Obtain a valid login command by referring to [Obtaining a Valid Login Command](#) and try again.

Invalid Organization Name

Error Log


```
invalid reference format  
[ERROR] : [pluginFrame] step run failed, errorMessage: fail to execute docker command.
```

Analysis and Solution

The SWR service has requirements on the **organization** name format. This error occurs when the organization name used to push an image does not meet the format requirements.

If this error occurs, go to the [Organization Management](#) page, switch to the corresponding region, and check whether the entered organization name is correct. If yes, manually create an organization with a valid name and try again.

Local Image Does Not Exist

Error Log

```
[2022-03-05 17:01:05.816] An image does not exist locally with the tag:  
swr.example.example.com/demo/faqdemo1  
[ERROR] : [pluginFrame] step run failed, errorMessage: fail to execute docker command.
```

Analysis and Solution

The possible cause is that the image fails to be created or the image name and tag in the push command are incorrect. As a result, the expected image in the push command is inconsistent with the image generated from the build or tag command. Check whether the image creation process or push parameters are correct.

In this example, **faqdemo1** in the image **docker push swr.example.example.com/demo/faqdemo1:v1.1** is incorrect. The image name specified in the build parameter is **faqdemo**. Modify the push parameter and try again.

Invalid Abstract

Error Log

```
digest invalid: Invalid digest
```

Analysis and Solution

This problem is usually caused by unstable SWR network. Try again for several times.

6.3 How Do I Resolve the Image Pull Failure During a Build Task?

Symptoms

When a build task is executed, the following error information is displayed in the log:

```
ERROR: docker pull image failed, dockerImage
```

Cause Analysis

The possible causes are as follows:

- The network is abnormal. As a result, the pull times out.
- The image to be pulled does not exist.
- The pulled image is private.

Solution

- If the fault is caused by a network exception, perform the following operations to rectify the fault:
 - a. Try again to check whether the problem can be solved.
 - b. If the problem occurs frequently or the retry still fails, contact customer service.
- The image does not exist. Ensure that the image has been uploaded to the image repository and the image name and version are correct.
- If the image is private, set the image to be public, or perform the *docker login* authentication, and perform the *docker pull* operation.

6.4 No Permission to Pull Images When SWR Public Images Are Used

Symptoms

When a build task is executed, the following error information is displayed in the log:

```
Get https://swr.example.example.com/v2/codeciexample-test/demo/manifests/v1.1: denied: You may not login yet
```

Cause Analysis

When the build task contains the action **Use SWR Public Image**, an error is reported because the permission of the Docker image invoked for building is not set to public.

Solution

Log in to SWR, find the image used during the build, edit the image, and set the image type to be public. The procedure is as follows:

1. Log in to SWR.
2. In the navigation pane on the left, 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 edit box, set **Type** to **Public**.

Edit Image



Organization

Name

Type Public Private

Category

Description
0/30,000

6.5 Failed to Log In to the Image Repository

Symptoms

The following error is displayed.

```
Error response from daemon: login attempt to https://hub.docker.com/v2/ failed with status: 404 Not Found
```

Cause Analysis

The image repository address is incorrect. The image repository for HTTPS requests cannot be customized.

Solution

Use the default image repository address provided by the system.

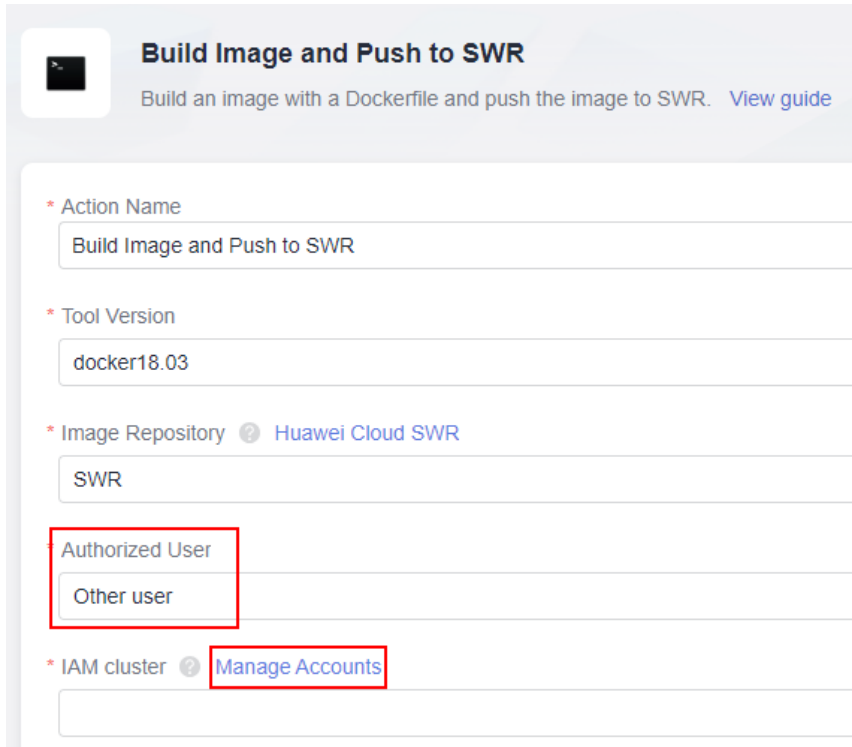
6.6 How Do I Push an Image to Other Tenants?

Symptoms

When an image is created and pushed to the SWR repository, error message **DEV.CB.0210043** is displayed, indicating that the Docker image fails to be created.

Solution

1. Log in to CodeArts Build.
2. Select the corresponding build task, click **...** in the row of the task, and click **Edit**.
3. On the **Build Actions** page, configure **Build Image and Push to SWR**.
4. Click **Manage Accounts**.



Build Image and Push to SWR
Build an image with a Dockerfile and push the image to SWR. [View guide](#)

* Action Name
Build Image and Push to SWR

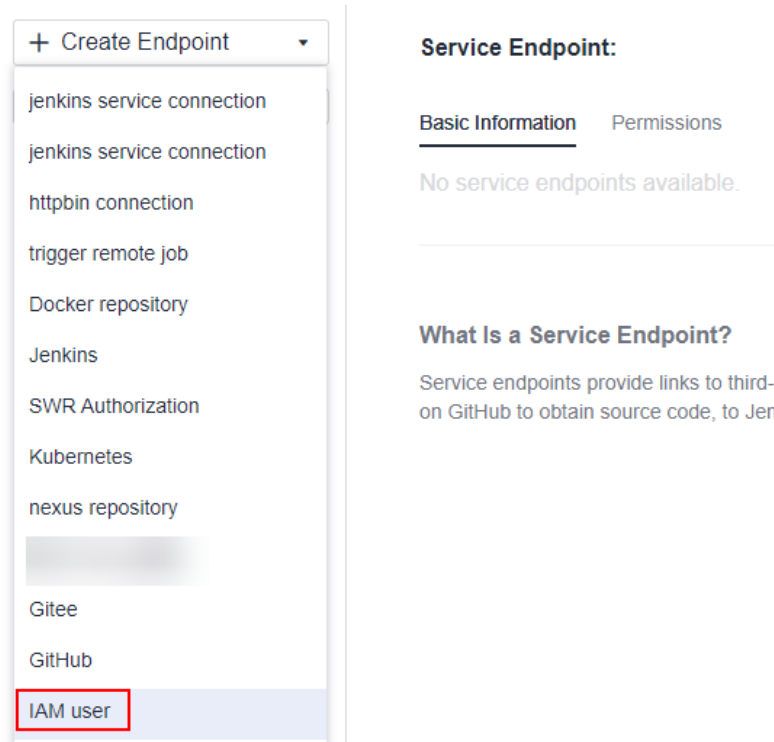
* Tool Version
docker18.03

* Image Repository ⓘ Huawei Cloud SWR
SWR

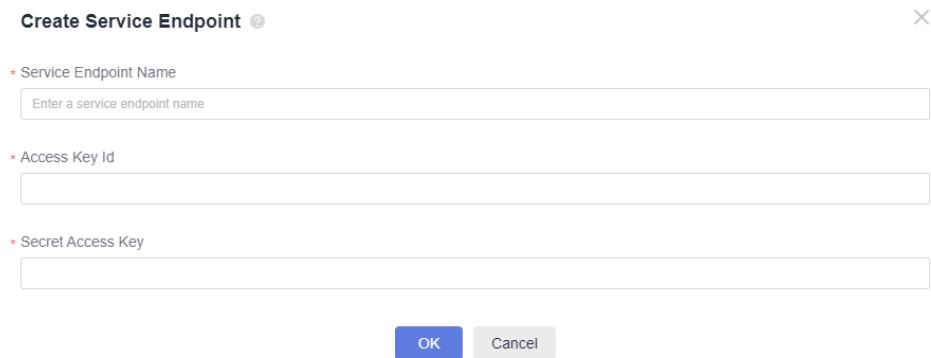
Authorized User
Other user

* IAM cluster ⓘ **Manage Accounts**

5. Click **Create Endpoint** and select **IAM user**.

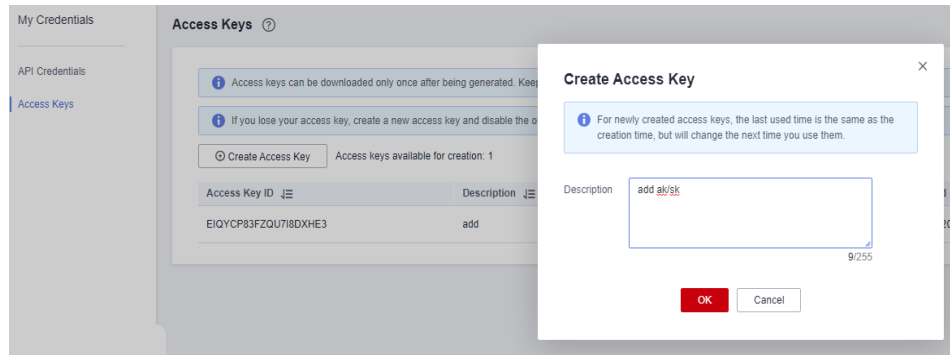


6. In the dialog box displayed, set parameters as shown in the following figure.



To obtain the access key ID and secret access key, perform the following steps:

- a. Click **Console** in the upper left corner of the page.
- b. In the upper right corner of the page, click the username and select **My Credentials**.
- c. Click **Access Keys** in the navigation pane.
- d. Click **Create Access Key**, enter a description, and click **OK**.



- e. In the dialog box that is displayed, click **Download** to download the key information to the local PC.

A	B	C
User Name	Access Key Id	Secret Access Key
XXXXXXXXXXXXXXXXXXXX	XXXXXXXXXXXXXXXXXXXX	XXXXXXXXXXXXXXXXXXXX

7. Select the service endpoint created in [Step 6](#) for the IAM account in [Step 4](#).

6.7 Pulling Docker Hub Images Times Out or Exceeds the Max. Attempts

Symptoms

When the build task is executed, the following error information is displayed in a log, indicating that the image pull from Docker Hub times out or reaches the pull limit:

```
Error response from daemon: Get https://registry.docker-cn.com/v2/: net/http: request canceled while waiting for connection (Client.Timeout exceeded while awaiting headers)
```


or

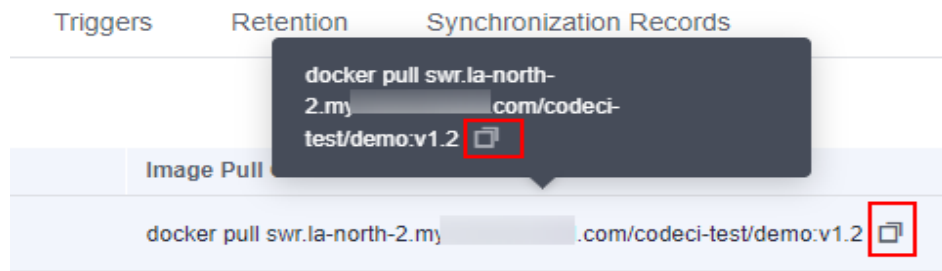
```
toomanyrequests: You have reached your pull rate limit. You may increase the limit by authenticating and upgrading: https://www.docker.com/increase-rate-limit
```

Cause Analysis

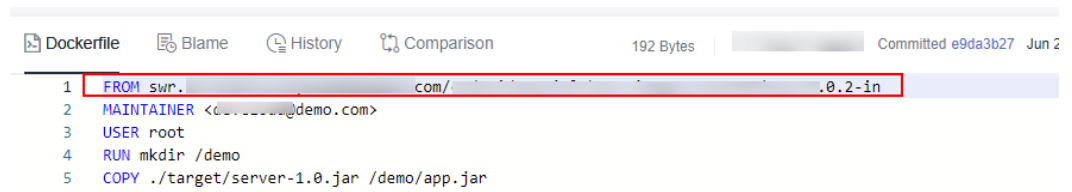
The network of the Docker Hub is unstable and the frequency is limited. As a result, the image pull may time out or fail. You can migrate the Docker Hub image to SWR and then pull the image.

Solution

- Step 1** Download the image of Docker Hub to the local PC.
- Step 2** Upload the image to SWR. For details, see [Uploading an Image Through SWR Console](#).
- Step 3** On the **Tags** tab page, in the same row as the target image tag, click  in the **Image Pull Command** column to copy the command.



Step 4 Modify the **Dockerfile** in the code repository and change the image path in the file to the address copied in **Step 3**.



----End