CodeArts Build

FAQs

Issue 01

Date 2023-11-30

HUAWEI TECHNOLOGIES CO., LTD.

Security Declaration

Vulnerability

Huawei's regulations on product vulnerability management are subject to "Vul. Response Process". For details about the policy, see the following website:https://www.huawei.com/en/psirt/vul-response-process
For enterprise customers who need to obtain vulnerability information, visit:https://securitybulletin.huawei.com/enterprise/en/security-advisory

Contents

1 General FAQs....... 1
1.1 What Is CodeArts Build?....... 1
1.2 Can I Specify a Server or Server Configuration for Running a Build Task?....... 1
1.3 Does CodeArts Build Support iOS?....... 2
1.4 Is There a Limit on the Size of the Build Package to Be Uploaded?....... 2
1.5 Project Files Not Found During Builds....... 2
1.6 Files Not Found During Software Package Upload....... 3
1.7 Insufficient Permissions....... 4
1.8 Task Not Found....... 4
1.9 Task Aborted....... 5
1.10 Migrating Common Java Projects to Cloud by Eclipse....... 5
2 Fetching Code....... 11
2.1 Failing to Pull Code of a Submodule....... 11
2.2 Failing to Pull a Submodule and Obtain Its Revision Version By Git....... 12
2.3 Git Does Not Pull Submodules....... 13
3 Using Maven for Build....... 14
3.1 Failed License Check....... 14
3.2 Failed to Upload a Package Using the maven deploy Command....... 14
3.3 POM Not Found....... 15
3.4 Package or Symbol Not Found....... 17
3.5 Incomplete JAR Due to Parallel Build Tasks....... 19
3.6 Using the exec-maven-plugin Extension for Maven and NPM Hybrid Builds....... 20
3.7 Referencing Between Parent and Child POMs....... 22
3.8 Configuring and Clearing Maven Build Cache....... 24
3.9 Finding the Correct Build Package Directory....... 25
3.10 Using the jib-maven-plugin Extension to Build a Maven Project and Create an Image....... 26
3.11 Package Remains Old After Code Update....... 27
3.12 Service Endpoint Did Not Exist....... 28
4 Using Android for Build....... 29
4.1 Jcenter() Configured for the Project Is Unstable....... 29
4.2 Failing to Execute a Task Due to the lint Check Error....... 30
4.3 Failing to Download com.android.tools.build:gradle:3.0.1....... 30

4.4 Javadoc generation failed........ 31
4.5 Could not find method google()........ 31
4.6 Gradle Version Too Early........ 32
4.7 Signature Failures During a Build with Android........ 32

5 Using Gradle for Build........ 35

5.1 Failing to Find Gradle of the Specified Version........ 35

6 Using npm for Build........ 37

6.1 JavaScript heap out of memory........ 37
6.2 enoent ENOENT: no such file or directory........ 38
6.3 Module not found: Error: Can't resolve 39
6.4 No Error Displayed for Failed Build with npm........ 39
6.5 npm cb() never called........ 40
6.6 gyp ERR! stack Error: EACCES: permission denied........ 41
6.7 eslint: error 'CLODOP' is not defined........ 41
6.8 Failed to Download node-sass........ 42
6.9 error: could not write config file........ 43
6.10 Low Efficiency and Slow Dependency Installation During Build with npm........ 44

7 Using Docker for Build........ 45

7.1 Failed to Create an Image Using Dockerfile........ 45
7.2 Failed to Push Images to SWR........ 47
7.3 Failed to Pull the Image........ 49
7.4 No Permission to Pull Images When SWR Public Images Are Used........ 50
7.5 Failed to Log In to the Image Repository........ 51

8 Creating Images and Pushing to SWR........ 52

8.1 How Do I Push Images to Other Tenants?........ 52
8.2 Pulling Docker Hub Images Times Out or Exceeds the Max. Attempts........ 54

1 General FAQs

1.1 What Is CodeArts Build?

Background

CodeArts Build is seamlessly interconnected with CodeArts Repo to provide cloud-based builds. With CodeArts Build, you can create, configure, and run build tasks with a few clicks. CodeArts Build also supports automation of code retrieval, build, and packaging, as well as real-time status monitoring.

Programming Languages

CodeArts Build supports mainstream programming languages, such as Java, C, C++, PHP, Groovy, C#, JavaScript, Python, Go, build standards, as well as embedded applications. In addition, CodeArts Build supports a wide range of environments and custom templates.

Code Source

- Repo
- GitHub
- Git
- Pipeline

Procedure

Quickly get started with CodeArts Build by following the procedure in **this document**.

1.2 Can I Specify a Server or Server Configuration for Running a Build Task?

No.

Currently, CodeArts Build randomly allocates idle servers for running build tasks. You cannot specify a server for running a build task.

1.3 Does CodeArts Build Support iOS?

Yes. CodeArts Build supports iOS builds using a custom macOS executor.

1.4 Is There a Limit on the Size of the Build Package to Be Uploaded?

Yes.

For security purposes, the size of a build package for a single build is limited.

Scenario	Package Size
Maximum upload from CodeArts Build to the release repo	5 GB
Maximum upload from CodeArts Build to the self-hosted repo	300 MB

1.5 Project Files Not Found During Builds

Symptoms

Building with Maven depends on build files such as **pom.xml**. If these files are not found, the build will fail and return an error message indicating that xxx project cannot find the xxx file. The following table lists common error messages.

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

Cause Analysis

The possible causes are as follows:

- No build file exists in the code project directory.
- The code project directory is nested. The directory where the build commands are executed does not contain the required file.

Solution

- Check whether the required build file is lost in the project.
- Check whether the build file is in the root directory of the project (or whether the build file path is specified in the build command). If necessary, run the **cd** command to go to the subdirectory.

  Example:

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

1.6 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 archiced
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**.

1.7 Insufficient Permissions

Symptoms

A build task fails to be executed and an error message is returned indicating that you do not have the required permissions.

Cause Analysis

If a user does not know his/her role or the role is modified, the user does not have the permission for builds. As a result, the user cannot perform 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.

----End

1.8 Task Not Found

Symptoms

Failed to execute a pipeline. The build task mounted to the pipeline reports an error, indicating that the task does not exist.

Cause Analysis

The build task is deleted. As a result, the pipeline fails to be executed.

Solution

Step 1 Check whether the task has been manually deleted and cannot be restored by users.

Step 2 Reconfigure the build task and the pipeline.

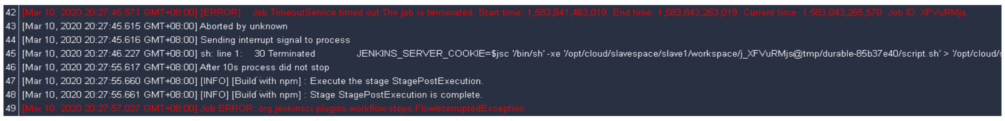
Step 3 If the problem persists, contact technical support.

----End

1.9 Task Aborted

Symptoms

The build task is aborted and the error message is displayed as follows:



```
[Mar 10, 2020 20:27:45.571 GMT+08:00] [ERROR]  : Job TimeoutService timed out.The job is terminated. Start time: 1,583,841,463,019. End time: 1,583,843,263,019. Current time: 1,583,843,265,570. Job ID: XFVuRMjs.
[Mar 10, 2020 20:27:45.615 GMT+08:00] Aborted by unknown
[Mar 10, 2020 20:27:45.616 GMT+08:00] Sending interrupt signal to process
[Mar 10, 2020 20:27:46.227 GMT+08:00] sh: line 1:    30 Terminated            JENKINS_SERVER_COOKIE=$jsc '/bin/sh' -xe '/opt/cloud/slavespace/slave1/workspace/j_XFVuRMjs@tmp/durable-85b37e40/script.sh' > '/opt/cloud/s
[Mar 10, 2020 20:27:55.617 GMT+08:00] After 10s process did not stop
[Mar 10, 2020 20:27:55.660 GMT+08:00] [INFO] [Build with npm] : Execute the stage StagePostExecution.
[Mar 10, 2020 20:27:55.661 GMT+08:00] [INFO] [Build with npm] : Stage StagePostExecution is complete.
[Mar 10, 2020 20:27:57.027 GMT+08:00] Job ERROR: org.jenkinsci.plugins.workflow.steps.FlowInterruptedException
```

Cause Analysis

The maximum build duration of a single build task is 1 hour (for non-paid users) or 4 hours (for paid users). If the build duration exceeds the system limit, the system forcibly stops the task.

1.10 Migrating Common Java Projects to Cloud by Eclipse

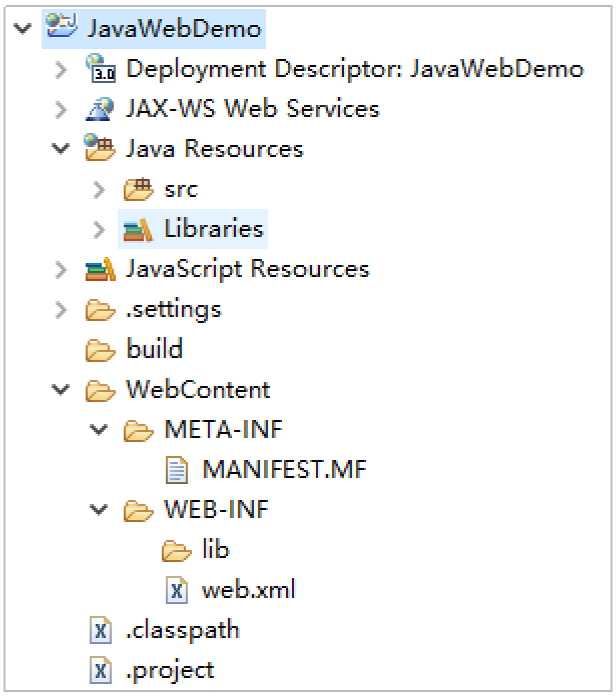
Background

Java web projects developed by Eclipse cannot generate build packages in CodeArts. Therefore, the project needs to be converted. The following procedure describes how to reconstruct a project into an Ant project and use the Ant tool to build a package in CodeArts.

Procedure

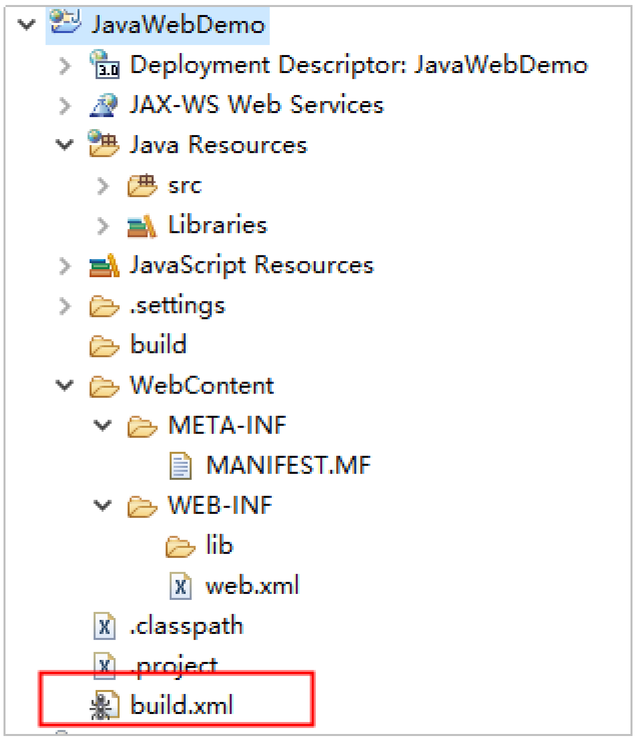
Step 1 Create a **build.xml** file in the project.

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



```
JavaWebDemo
  Deployment Descriptor: JavaWebDemo
  JAX-WS Web Services
  Java Resources
    src
    Libraries
  JavaScript Resources
  .settings
  build
  WebContent
    META-INF
      MANIFEST.MF
    WEB-INF
      lib
      web.xml
  .classpath
  .project
```

Create a **build.xml** file in the root directory. The following figure shows the new directory structure.



```
JavaWebDemo
  Deployment Descriptor: JavaWebDemo
  JAX-WS Web Services
  Java Resources
    src
    Libraries
  JavaScript Resources
  .settings
  build
  WebContent
    META-INF
      MANIFEST.MF
    WEB-INF
      lib
      web.xml
  .classpath
  .project
  build.xml
```

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 -->
    <pathelement 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 JavaScript 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 file">
<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 file">
<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 software release repo.

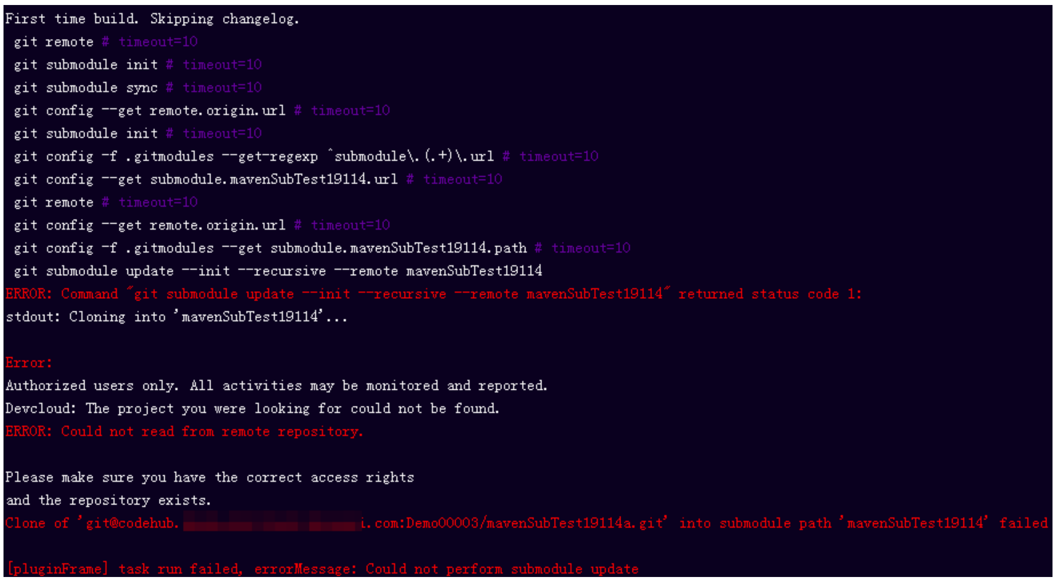
----End

2 Fetching Code

2.1 Failing to Pull Code of a Submodule

Symptoms

When a build task is executed, the following error information is displayed.



```
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.[redacted]i.com:Demo00003/mavenSubTest19114a.git' into submodule path 'mavenSubTest19114' failed

[pluginFrame] task run failed, errorMessage: Could not perform submodule update
```

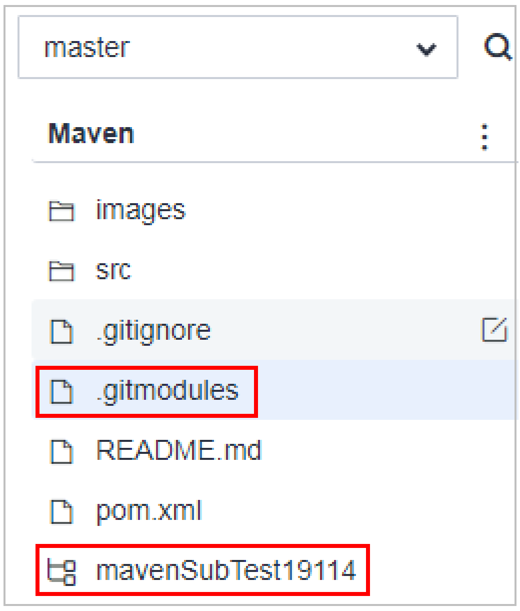
Cause Analysis

If the error **Could not read from remote repository** is displayed when Git pulls sub-modules from CodeArts Repo, the possible cause is that the user does not have the required permissions or the **.gitmodules** file is incorrectly configured.

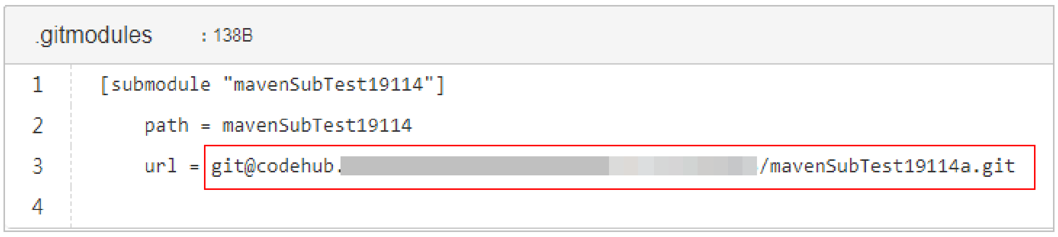
Solution

Step 1 Open the main code repository and choose **Settings** > **Submodules**. Click the synchronization icon to synchronize the deployment key. Then, run the build task again.

Step 2 If the deployment key has been synchronized in **Step 1**, the **.gitmodules** file in the main code repository may be incorrectly configured. Check whether the **.gitmodules** file exists and the submodule is **mavenSubTest19114**.



Step 3 Open the **.gitmodules** file and modify the submodule configuration to **mavenSubTest19114a.git**.



.gitmodules : 138B

```
[submodule "mavenSubTest19114"]
    path = mavenSubTest19114
    url = git@codehub.[redacted]/mavenSubTest19114a.git

```

Step 4 Modify the **.gitmodules** file and run the build task again.

----End

2.2 Failing to Pull a Submodule and Obtain Its Revision Version By Git

Symptoms

The following error is displayed:

```
[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]
[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]: StagePostExecution started
[2019-07-02 08:29:23.251] [INFO] [Code checkout]: StagePostExecution finished
```

Cause Analysis

The original directory is incorrect. In this example, the directory is **asae-feign**. This is a Git bug.

Solution

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

2.3 Git Does Not Pull Submodules

Symptoms

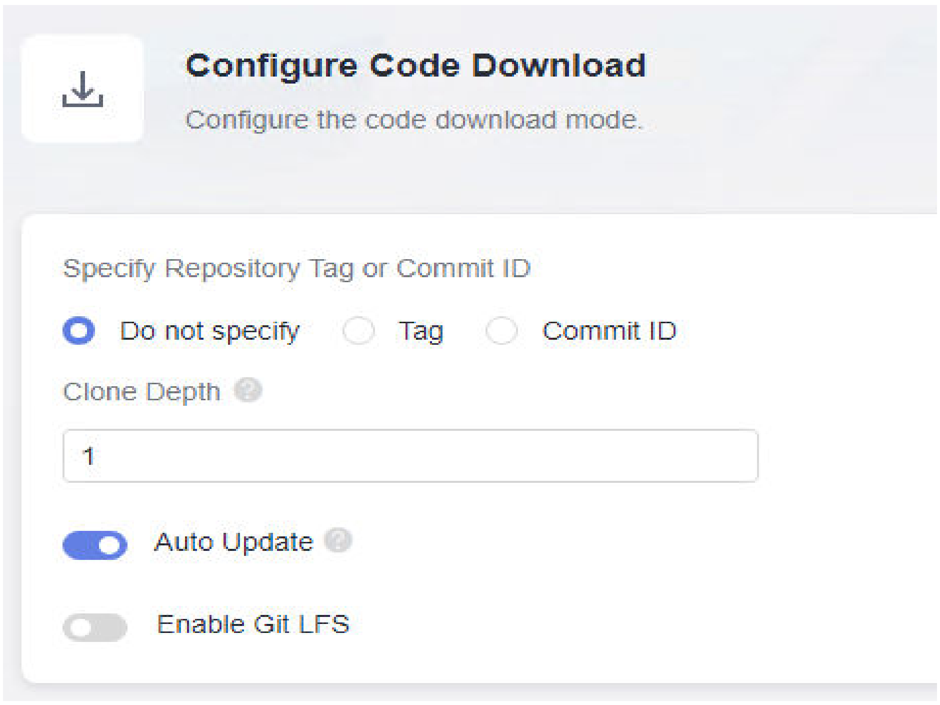
When CodeArts Build pulls code from Repo, GitHub, and other sources, the **.gitmodules** file exists and the configuration is correct, but the sub-module is not pulled.

Cause Analysis

Generally, this problem occurs because the automatic update function of the submodule is disabled.

Solution

Edit the build task file, select the **Configure Code Download** build action, and enable **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
1
Auto Update: enabled
Enable Git LFS: disabled

3 Using Maven for Build

3.1 Failed License Check

Symptoms

The following error is displayed:

```
[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
```

3.2 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:

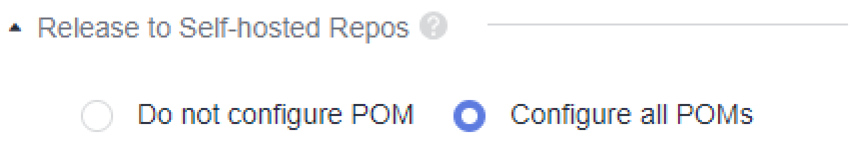
```
[ERROR] Failed to execute goal org.apache.maven.plugins:maven-deploy-plugin:2.7:deploy (default-deploy)
on project javaMavenDemo: Deployment failed: repository element was not specified in the POM inside
distributionManagement element or in -DaltDeploymentRepository=id::layout::url parameter -&gt;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

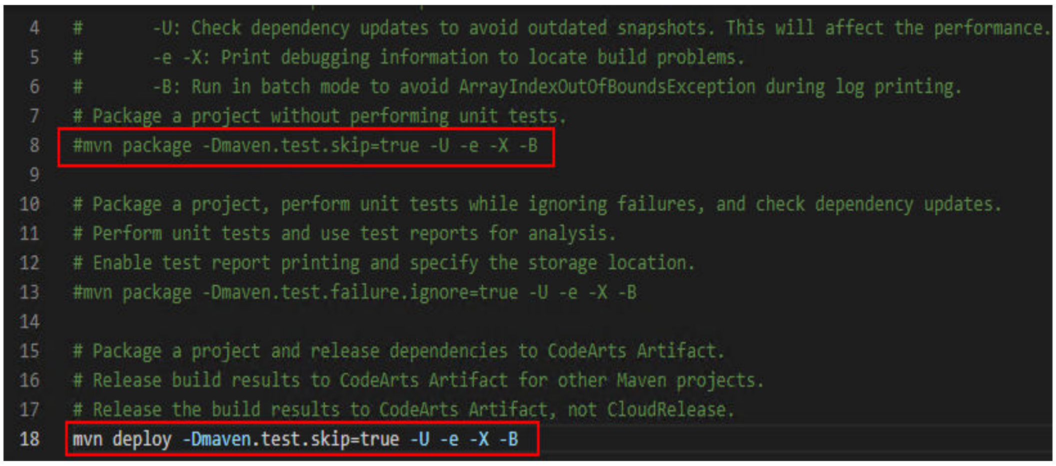


Release to Self-hosted Repos

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

3.3 POM Not Found

Symptoms

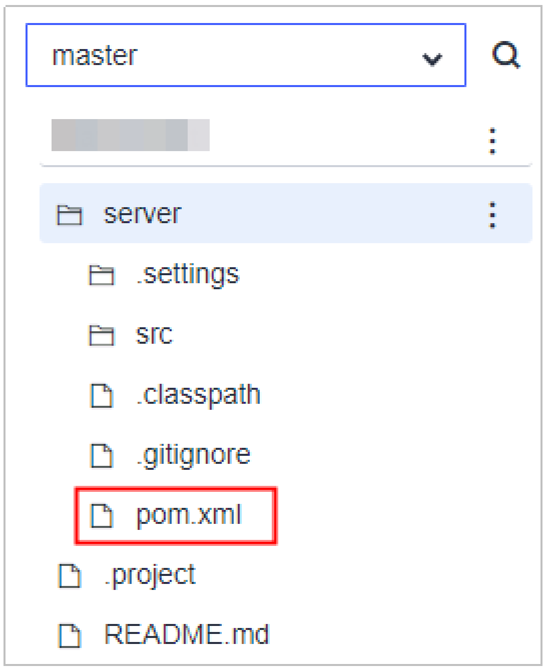
The following error is displayed:

```
[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.

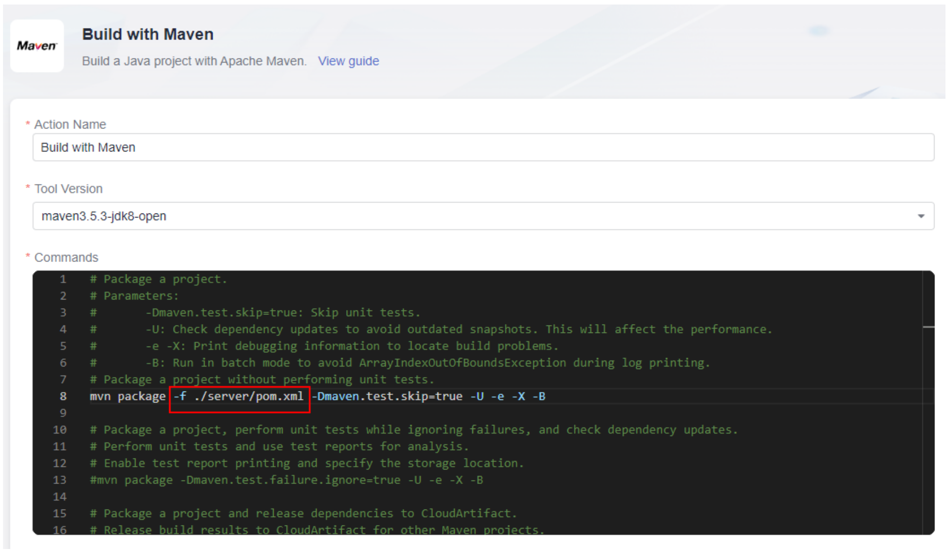
For example, the following screenshot shows that POM does not exist in the root directory of the source code. Instead, it is in the **server** directory.



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.



Action Name: Build with Maven

Tool Version: maven3.5.3-jdk8-open

Commands

```
# Package a project.
# Parameters:
#       -Dmaven.test.skip=true: Skip unit tests.
#       -U: Check dependency updates to avoid outdated snapshots. This will affect the performance.
#       -e -X: Print debugging information to locate build problems.
#       -B: Run in batch mode to avoid ArrayIndexOutOfBoundsException during log printing.
# Package a project without performing unit tests.
mvn package -f ./server/pom.xml -Dmaven.test.skip=true -U -e -X -B

# Package a project, perform unit tests while ignoring failures, and check dependency updates.
# Perform unit tests and use test reports for analysis.
# Enable test report printing and specify the storage location.
#mvn package -Dmaven.test.failure.ignore=true -U -e -X -B

# Package a project and release dependencies to CloudArtifact.
# Release build results to CloudArtifact for other Maven projects.
```

3.4 Package or Symbol Not Found

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 the groupId, artifactId, and version (GAV)**
  - **Damaged dependency package**
  - Others

Dependency Package Conflict

Sometimes, multiple versions of the same artifact 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 artifact version is not the required version, import the required dependency at the outermost layer of the POM and try again.

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

----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 the groupId, artifactId, and version (GAV)

- When uploading a dependency package to self-hosted repos using the 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 artifact A, which is built through project A, for building project D. Artifact A contains a third-party tool, B. The dependency relationships are: D > A > B. When parsing artifact A, Maven will not be able to identify artifact B. As a result, project D cannot find contents in artifact 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 artifact A from 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 artifact A and upload it again using either of the following methods:

- Build project A in CodeArts Build. Run the **deploy** command to upload artifact A to the private Maven repository. You can use the pipeline for automation.
- Upload artifact 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**.

3.5 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

3.6 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.

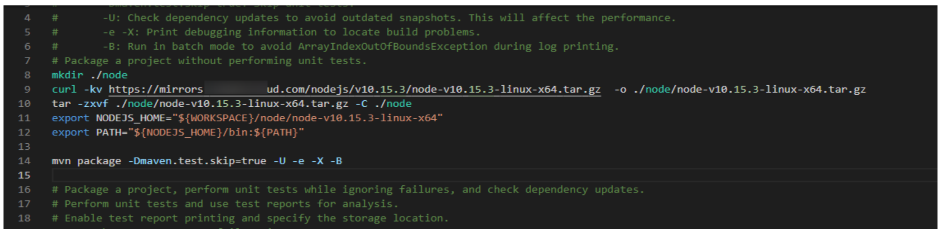
```
curl -kv https://mirrors.xxxcloud.com/nodejs/v10.15.3/node-v10.15.3-linux-x64.tar.gz  -o ./node/node-v10.15.3-linux-x64.tar.gz
```

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}"
```



```
#       -U: Check dependency updates to avoid outdated snapshots. This will affect the performance.
#       -e -X: Print debugging information to locate build problems.
#       -B: Run in batch mode to avoid ArrayIndexOutOfBoundsException during log printing.
# Package a project without performing unit tests.
mkdir ./node
curl -kv https://mirrors[redacted]ud.com/nodejs/v10.15.3/node-v10.15.3-linux-x64.tar.gz  -o ./node/node-v10.15.3-linux-x64.tar.gz
tar -zxvf ./node/node-v10.15.3-linux-x64.tar.gz -C ./node
export NODEJS_HOME="${WORKSPACE}/node/node-v10.15.3-linux-x64"
export PATH="${NODEJS_HOME}/bin:${PATH}"

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

# Package a project, perform unit tests while ignoring failures, and check dependency updates.
# Perform unit tests and use test reports for analysis.
# Enable test report printing and specify the storage location.
```

Step 4 Save the settings and verify the build.

----End

3.7 Referencing Between Parent and Child POMs

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 reference of the project. 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:

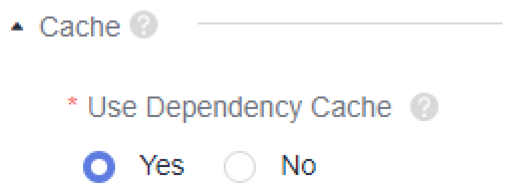
```
<modelVersion>4.0.0</modelVersion>
<groupId>com.demo</groupId>
<artifactId>childA</artifactId>
<version>1.0</version>
<parent>
  <groupId>com.demo</groupId>
  <artifactId>childA</artifactId>
  <version>1.0</version>
</parent>
```

3.8 Configuring and Clearing Maven Build Cache

CodeArts Build allows you to cache the dependencies in your private storage space. Once cached, the dependencies will not have to be pulled for future builds. It greatly accelerates builds.

Configuring Cache Setting

When a build task is created, the cache acceleration build is selected by default. You can determine whether to use the cache when configuring the Maven build action.



Clearing Cache

Network jitter, concurrent builds, or other extreme conditions may result in abnormal cache. Consequently, build errors may occur. The following describes how to clear the abnormal cache.

NOTICE

Before clearing cache, make sure you are well aware of the following precautions:

- The cache directory is shared by multiple users of the same tenant. If the cache is frequently cleared, exceptions (usually xxx file does not exist) may occur during other users' builds. Therefore, this operation can be performed only when the cache is abnormal. After the task succeeds, edit the task to delete the cache clearing command. While clearing the cache, do not run other build tasks that use the 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 be executed only in the **Build with Maven** action. If this command is executed in other actions, the clearing operation may not succeed or an error will be displayed (for example the directory does not exist).

Step 1 Click ✏ on the build tasks page.

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

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

The format of the cache clearing command is **rm -rf /repository/local/maven/{groupId}/{artifactId}/{version}**. The parameters to be entered correspond to **groupId**, **artifactId**, and **version** in the dependency package coordinates. The dots in **groupId** are automatically divided into hierarchical directories.

For example, if the dependency package is as follows:

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

The command for clearing the package 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

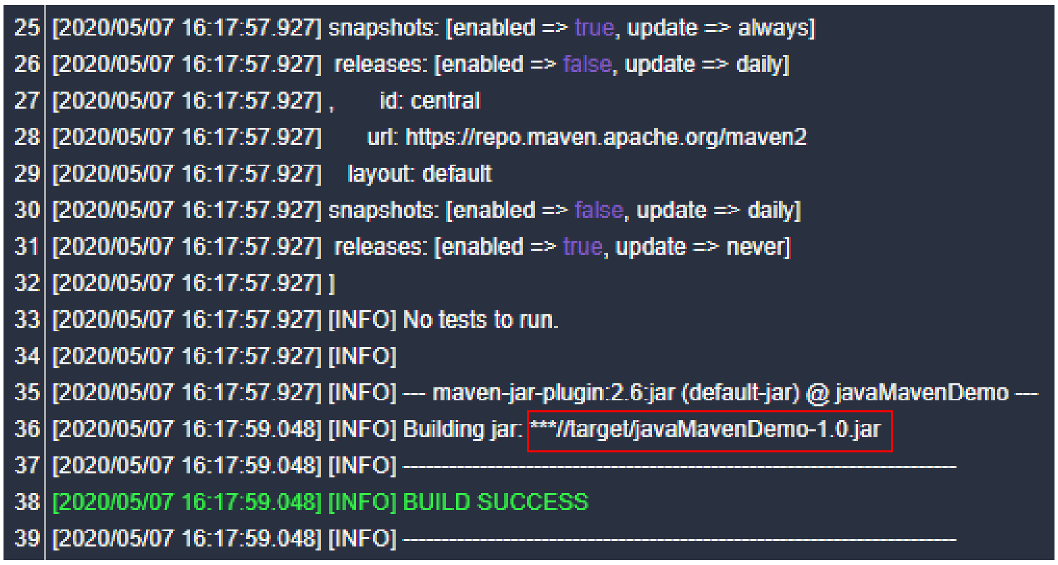
3.9 Finding the Correct Build Package Directory

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.



```
[2020/05/07 16:17:57.927] snapshots: [enabled => true, update => always]
[2020/05/07 16:17:57.927]  releases: [enabled => false, update => daily]
[2020/05/07 16:17:57.927] ,      id: central
[2020/05/07 16:17:57.927]      url: https://repo.maven.apache.org/maven2
[2020/05/07 16:17:57.927]   layout: default
[2020/05/07 16:17:57.927] snapshots: [enabled => false, update => daily]
[2020/05/07 16:17:57.927]  releases: [enabled => true, update => never]
[2020/05/07 16:17:57.927] ]
[2020/05/07 16:17:57.927] [INFO] No tests to run.
[2020/05/07 16:17:57.927] [INFO]
[2020/05/07 16:17:57.927] [INFO] --- maven-jar-plugin:2.6:jar (default-jar) @ javaMavenDemo ---
[2020/05/07 16:17:59.048] [INFO] Building jar: ***/target/javaMavenDemo-1.0.jar
[2020/05/07 16:17:59.048] [INFO] ------------------------------------------------------------------------
[2020/05/07 16:17:59.048] [INFO] BUILD SUCCESS
[2020/05/07 16:17:59.048] [INFO] ------------------------------------------------------------------------
```

----End

3.10 Using the jib-maven-plugin Extension to Build a Maven Project and Create an Image

Background

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.

Procedure

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.cn-north-5.myxxcloud.com/xxxx/xxx:xxxxx</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>
                    <!--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:

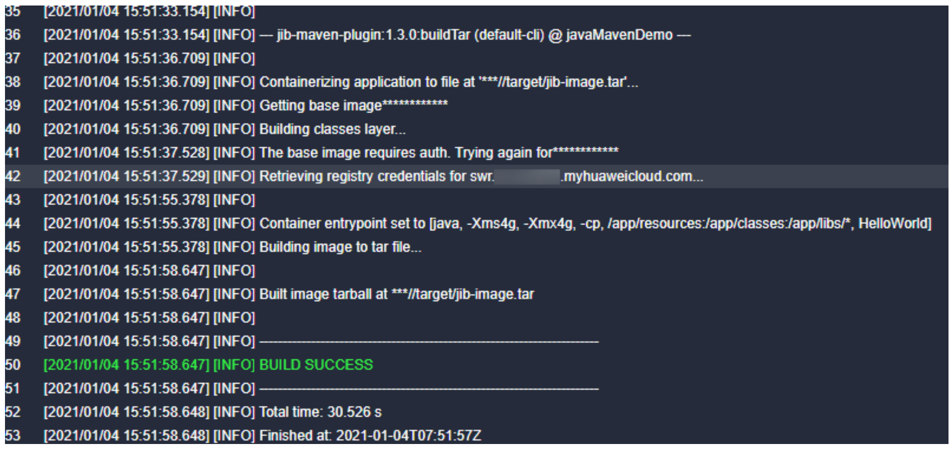
   ```
   mvn compile jib:buildTar -Dmaven.test.skip=true -U -e -X –B
   ```

   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:

   ```
   [2021/01/04 15:51:33.154] [INFO]
   [2021/01/04 15:51:33.154] [INFO] --- jib-maven-plugin:1.3.0:buildTar (default-cli) @ javaMavenDemo ---
   [2021/01/04 15:51:36.709] [INFO]
   [2021/01/04 15:51:36.709] [INFO] Containerizing application to file at '***//target/jib-image.tar'...
   [2021/01/04 15:51:36.709] [INFO] Getting base image************
   [2021/01/04 15:51:36.709] [INFO] Building classes layer...
   [2021/01/04 15:51:37.528] [INFO] The base image requires auth. Trying again for************
   [2021/01/04 15:51:37.529] [INFO] Retrieving registry credentials for swr.[redacted].myhuaweicloud.com...
   [2021/01/04 15:51:55.378] [INFO]
   [2021/01/04 15:51:55.378] [INFO] Container entrypoint set to [java, -Xms4g, -Xmx4g, -cp, /app/resources:/app/classes:/app/libs/*, HelloWorld]
   [2021/01/04 15:51:55.378] [INFO] Building image to tar file...
   [2021/01/04 15:51:58.647] [INFO]
   [2021/01/04 15:51:58.647] [INFO] Built image tarball at ***//target/jib-image.tar
   [2021/01/04 15:51:58.647] [INFO]
   [2021/01/04 15:51:58.647] [INFO] ------------------------------------------------------------------------
   [2021/01/04 15:51:58.647] [INFO] BUILD SUCCESS
   [2021/01/04 15:51:58.647] [INFO] ------------------------------------------------------------------------
   [2021/01/04 15:51:58.648] [INFO] Total time: 30.526 s
   [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 o 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.

3.11 Package Remains Old After Code Update

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
  ```

3.12 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.

4 Using Android for Build

4.1 Jcenter() Configured for the Project Is Unstable

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

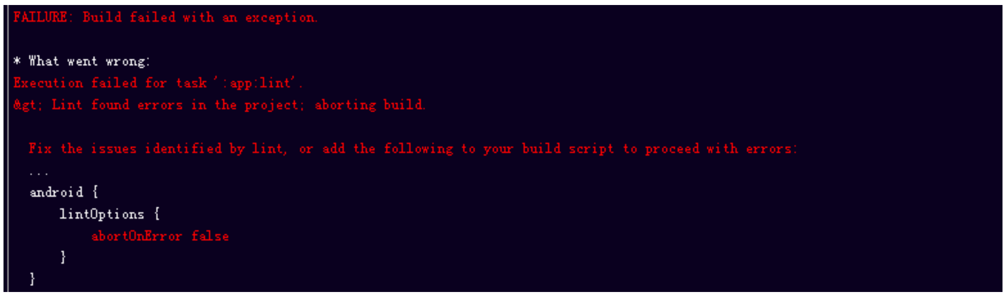
You are advised to configure an open-source image site, which is stable and fast. The configuration method is as follows:

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()
}
}
```

4.2 Failing to Execute a Task Due to the lint Check Error

Symptoms



```
FAILURE: Build failed with an exception.

* What went wrong:
Execution failed for task ':app:lint'.
&gt; 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

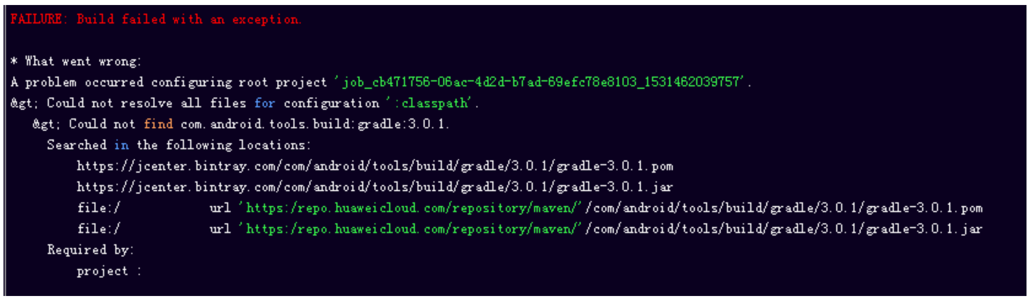
```
gradle assembleDebug -Dorg.gradle.daemon=false -d --stacktrace --init-script /root/.gradle/init.gradle -xlint
```

4.3 Failing to Download com.android.tools.build:gradle:3.0.1

Symptoms

The error information is as follows:

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



```
FAILURE: Build failed with an exception.

* What went wrong:
A problem occurred configuring root project 'job_cb471756-06ac-4d2d-b7ad-69efc78e8103_1531462039757'.
&gt; Could not resolve all files for configuration ':classpath'.
   &gt; Could not find com.android.tools.build:gradle:3.0.1.
     Searched in the following locations:
         https://jcenter.bintray.com/com/android/tools/build/gradle/3.0.1/gradle-3.0.1.pom
         https://jcenter.bintray.com/com/android/tools/build/gradle/3.0.1/gradle-3.0.1.jar
         file:/           url 'https:/repo.huaweicloud.com/repository/maven/'/com/android/tools/build/gradle/3.0.1/gradle-3.0.1.pom
         file:/           url 'https:/repo.huaweicloud.com/repository/maven/'/com/android/tools/build/gradle/3.0.1/gradle-3.0.1.jar
     Required by:
         project :
```

Solution

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

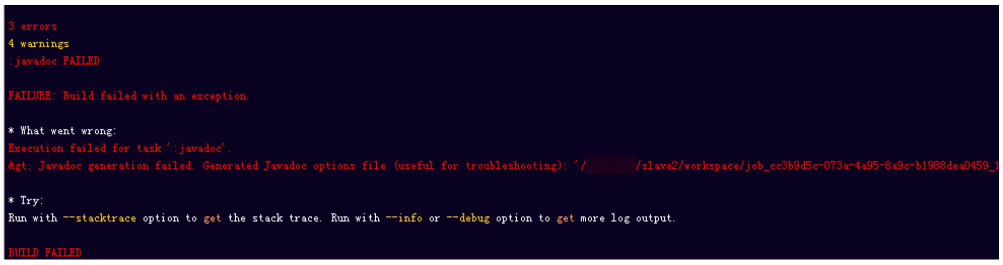
```
allprojects {
repositories {
```

```
google()
jcenter()
}
}
```

4.4 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'.
&gt; Javadoc generation failed. Generated Javadoc options file (useful for troubleshooting): '/[redacted]/slave2/workspace/job_cc3b9d5c-073a-4a95-8a9c-b1988dea0459_

* 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')
}
}
```

4.5 Could not find method google()

Symptoms

After the Gradle plug-in is upgraded to v3.0, the corresponding Gradle needs to be upgraded to v4.1. If Gradle v4.1 is not found, the following error is displayed:

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

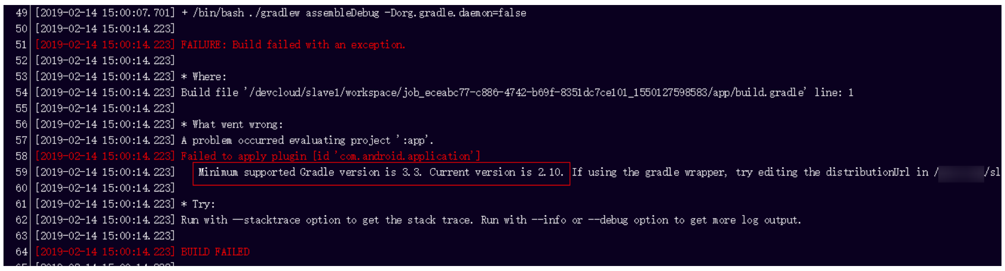
Solution

Use Gradle v4.1 or later versions.

4.6 Gradle Version 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.



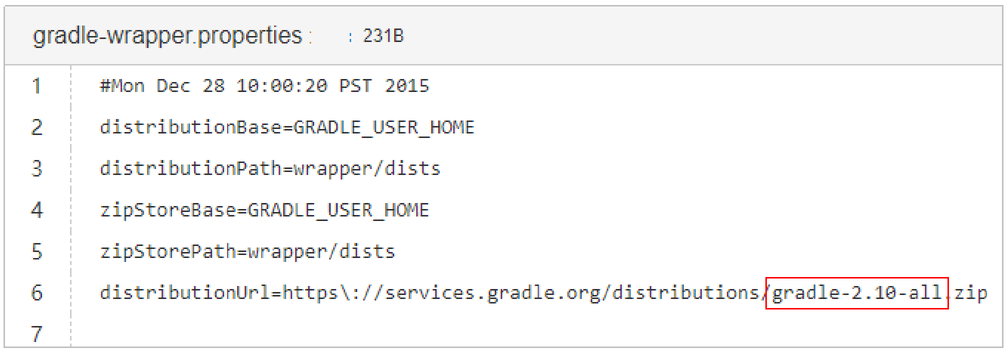
```
[2019-02-14 15:00:07.701] + /bin/bash ./gradlew assembleDebug -Dorg.gradle.daemon=false
[2019-02-14 15:00:14.223]
[2019-02-14 15:00:14.223] FAILURE: Build failed with an exception.
[2019-02-14 15:00:14.223]
[2019-02-14 15:00:14.223] * Where:
[2019-02-14 15:00:14.223] Build file '/devcloud/slave1/workspace/job_eceabc77-c886-4742-b69f-8351dc7ce101_1550127598583/app/build.gradle' line: 1
[2019-02-14 15:00:14.223]
[2019-02-14 15:00:14.223] * What went wrong:
[2019-02-14 15:00:14.223] A problem occurred evaluating project ':app'.
[2019-02-14 15:00:14.223] Failed to apply plugin [id 'com.android.application']
[2019-02-14 15:00:14.223]    Minimum supported Gradle version is 3.3. Current version is 2.10. If using the gradle wrapper, try editing the distributionUrl in /[redacted]/sl
[2019-02-14 15:00:14.223]
[2019-02-14 15:00:14.223] * Try:
[2019-02-14 15:00:14.223] Run with --stacktrace option to get the stack trace. Run with --info or --debug option to get more log output.
[2019-02-14 15:00:14.223]
[2019-02-14 15:00:14.223] BUILD FAILED
```

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

```
#Mon Dec 28 10:00:20 PST 2015
distributionBase=GRADLE_USER_HOME
distributionPath=wrapper/dists
zipStoreBase=GRADLE_USER_HOME
zipStorePath=wrapper/dists
distributionUrl=https\://services.gradle.org/distributions/gradle-2.10-all.zip

```

4.7 Signature Failures During a Build with Android

Symptoms

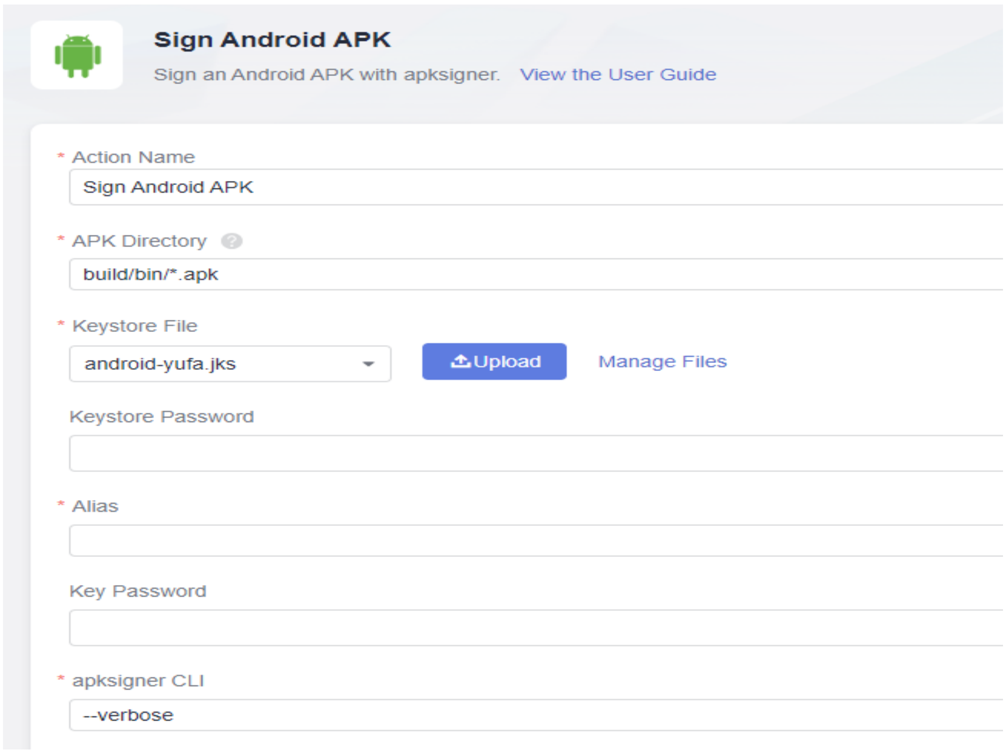
A signature error is reported during a build with Android.

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

Keystore Password:

* Alias:

Key Password:

* apksigner CLI: --verbose

Parameters:

Parameter	Description
APK Location	Location of the APK file to be signed generated after Android building. Regular expressions are supported. For example, **build/bin/*.apk** can be used to match the built APK package.
Keystore File	Keystore file used for signature. Click the drop-down list box to display the Keystore files uploaded on the **File Management** page. Select a file as required.
keystore password	Keystore Password
Alias	Alias of the keystore file.
key password	Password of the key.
apksigner Command	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.

5 Using Gradle for Build

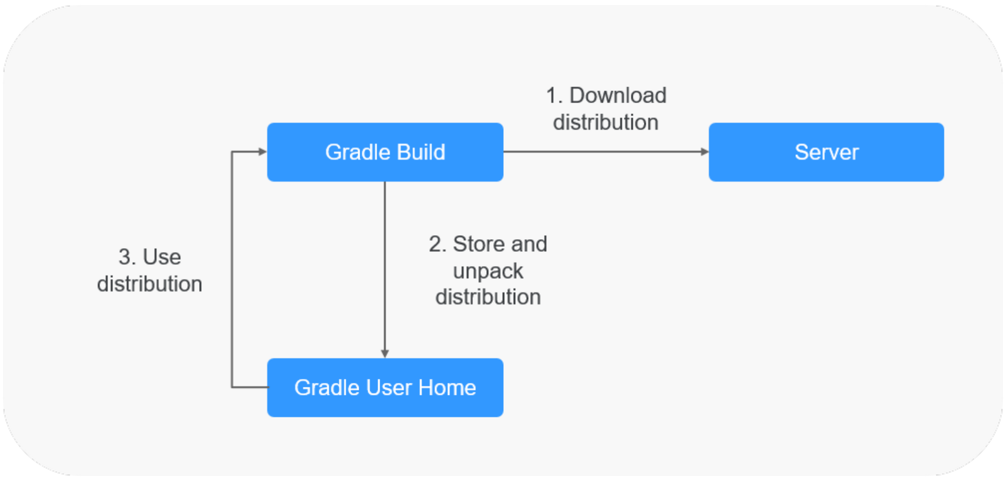
5.1 Failing to Find Gradle of the Specified Version

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 the 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

6 Using npm for Build

6.1 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 as follows:

Frame	Solution
Vue	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: `"build": "node --max_old_space_size=4096 build/build.js"`

Frame	Solution

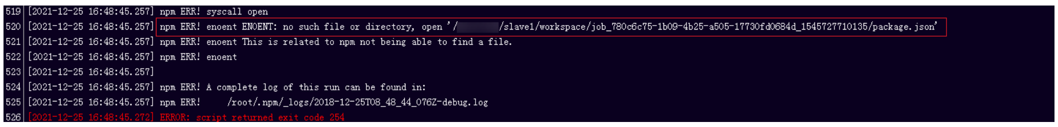
React	Assume that the content of the **scripts** field in the **package.json** file is as follows:
	`"scripts": {`
	`"start": "react-scripts start",`
	`"build": "react-scripts build",`
	`"test": "react-scripts test --env=jsdom",`
	`"eject": "react-scripts eject"`
	`}`
	The actual code that works in the **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**:
	`#!/usr/bin/env node --max_old_space_size=4096`

Angular	Assume that the content of the **scripts** field in the **package.json** file is as follows:
	`"scripts": {`
	`"ng": "ng",`
	`"start": "ng serve",`
	`"build": "ng build",`
	`"test": "ng test",`
	`"lint": "ng lint",`
	`"e2e": "ng e2e"`
	`}`
	Similar to **React**, the **ng** file exists in the **.bin** directory in the **mode_modules** folder in the root directory of a project. Modify the first line of the file.
	`#!/usr/bin/env  node --max_old_space_size=4096`

6.2 enoent ENOENT: no such file or directory

Symptoms

The following error is displayed:



```
519 [2021-12-25 16:48:45.257] npm ERR! syscall open
520 [2021-12-25 16:48:45.257] npm ERR! enoent ENOENT: no such file or directory, open '/[redacted]/slave1/workspace/job_780c6c75-1b09-4b25-a505-17730fd0684d_1545727710135/package.json'
521 [2021-12-25 16:48:45.257] npm ERR! enoent This is related to npm not being able to find a file.
522 [2021-12-25 16:48:45.257] npm ERR! enoent
523 [2021-12-25 16:48:45.257]
524 [2021-12-25 16:48:45.257] npm ERR! A complete log of this run can be found in:
525 [2021-12-25 16:48:45.257] npm ERR!     /root/.npm/_logs/2018-12-25T08_48_44_076Z-debug.log
526 [2021-12-25 16:48:45.272] ERROR: script returned exit code 254
```

Cause Analysis

The project lacks key files.

In the preceding figure, error **npm ERR! enoent ENOENT: no such file or directory, open '/xxx/slave1/workspace/job_780c6c75-1b09-4b25-a505-17730fd0684d_1545727710135/package.json'** in line 520 indicates that the **package.json** file is missing.

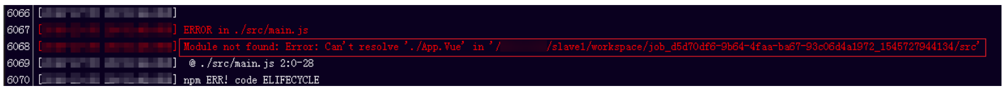
Solution

Add missing files as indicated in the error log. For example, if the **package.json** file is missing, add the **package.json** file to the root directory of the code.

6.3 Module not found: Error: Can't resolve ...

Symptoms

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



```
6066 [[redacted]]
6067 [[redacted]] ERROR in ./src/main.js
6068 [[redacted]] Module not found: Error: Can't resolve './App.Vue' in '/[redacted]/slave1/workspace/job_d5d70df6-9b64-4faa-ba67-93c06d4a1972_1545727944134/src'
6069 [[redacted]]  @ ./src/main.js 2:0-28
6070 [[redacted]] npm ERR! code ELIFECYCLE
```

Cause Analysis

The required files failed to be found.

Error **Module not found: Error: Can't resolve './App.Vue' in '/xxx/slave1/workspace/job_d5d70df6-9b64-4faa-ba67-93c06d4a1972_1545727944134/src'** in line 6068 in the preceding figure indicates that the **./App.Vue** file cannot be found in the **src** folder. Possible causes are as follows:

- The required file does not exist in the corresponding folder.
- The casing of the file path is incorrect. In the preceding figure, file name **'./App.vue'** is misspelled as **'./App.Vue'** in the code. As a result, this file cannot be found. The Windows OS is case insensitive, but the Linux OS is case sensitive. Therefore, the local build may be successful, but the build may fail in CodeArts Build.

Solution

Step 1 In the corresponding folder of the code project, add the missing files as indicated by the error.

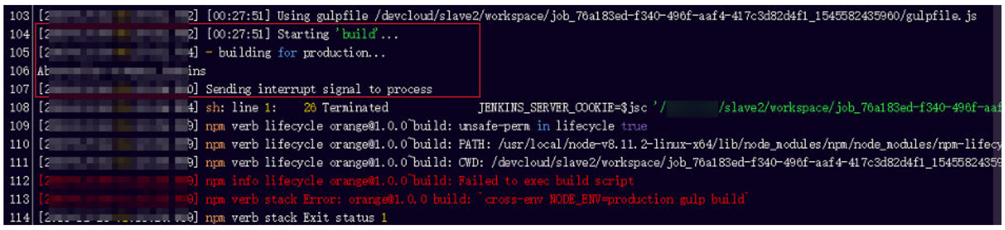
Step 2 Modify the file path configured in the code.

----End

6.4 No Error Displayed for Failed Build with npm

Symptoms

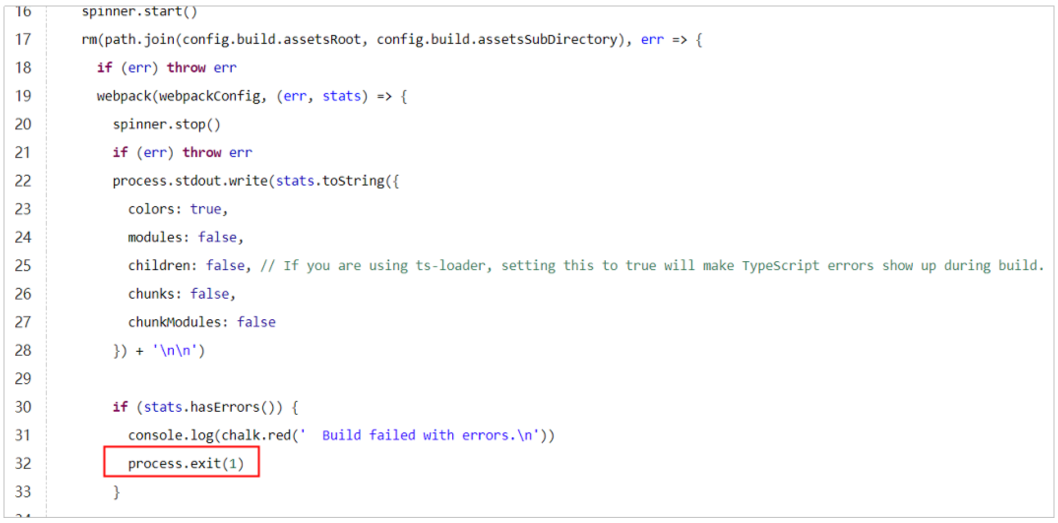
Build with npm fails, but no error log is displayed. The error is as follows:



```
[2[redacted]2] [00:27:51] Using gulpfile /devcloud/slave2/workspace/job_76a183ed-f340-496f-aaf4-417c3d82d4f1_1545582435960/gulpfile.js
[2[redacted]2] [00:27:51] Starting 'build'...
[2[redacted]4] - building for production...
Ab[redacted]ins
[2[redacted]0] Sending interrupt signal to process
[2[redacted]4] sh: line 1:    26 Terminated              JENKINS_SERVER_COOKIE=$jsc '/[redacted]/slave2/workspace/job_76a183ed-f340-496f-aaf
[2[redacted]9] npm verb lifecycle orange@1.0.0~build: unsafe-perm in lifecycle true
[2[redacted]9] npm verb lifecycle orange@1.0.0~build: PATH: /usr/local/node-v8.11.2-linux-x64/lib/node_modules/npm/node_modules/npm-lifecy
[2[redacted]9] npm verb lifecycle orange@1.0.0~build: CWD: /devcloud/slave2/workspace/job_76a183ed-f340-496f-aaf4-417c3d82d4f1_154558243591
[2[redacted]9] npm info lifecycle orange@1.0.0~build: Failed to exec build script
[2[redacted]9] npm verb stack Error: orange@1.0.0 build: `cross-env NODE_ENV=production gulp build`
[2[redacted]9] npm verb stack Exit status 1
```

Cause Analysis

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



```
    spinner.start()
    rm(path.join(config.build.assetsRoot, config.build.assetsSubDirectory), err => {
      if (err) throw err
      webpack(webpackConfig, (err, stats) => {
        spinner.stop()
        if (err) throw err
        process.stdout.write(stats.toString({
          colors: true,
          modules: false,
          children: false, // If you are using ts-loader, setting this to true will make TypeScript errors show up during build.
          chunks: false,
          chunkModules: false
        }) + '\n\n')

        if (stats.hasErrors()) {
          console.log(chalk.red('  Build failed with errors.\n'))
          process.exit(1)
        }
```

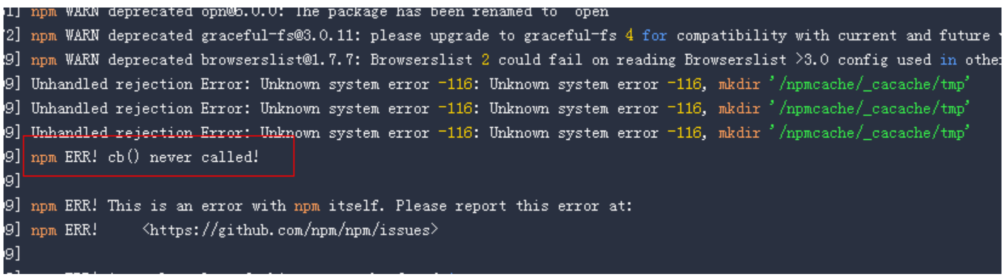
Solution

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

6.5 npm cb() never called

Symptoms

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



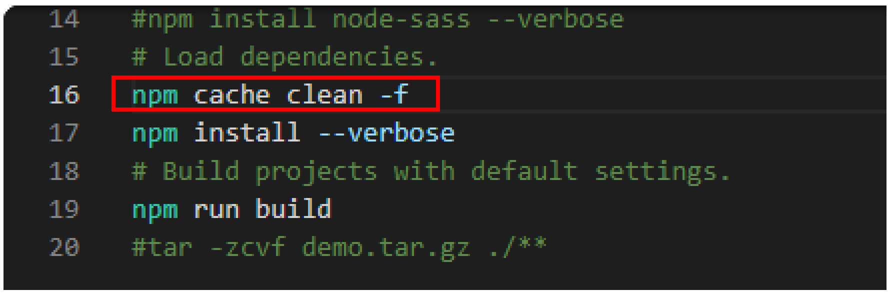
```
1] npm WARN deprecated opn@5.0.0: The package has been renamed to `open`
2] npm WARN deprecated graceful-fs@3.0.11: please upgrade to graceful-fs 4 for compatibility with current and future
9] npm WARN deprecated browserslist@1.7.7: Browserslist 2 could fail on reading Browserslist >3.0 config used in othe
9] Unhandled rejection Error: Unknown system error -116: Unknown system error -116, mkdir '/npmcache/_cacache/tmp'
9] Unhandled rejection Error: Unknown system error -116: Unknown system error -116, mkdir '/npmcache/_cacache/tmp'
9] Unhandled rejection Error: Unknown system error -116: Unknown system error -116, mkdir '/npmcache/_cacache/tmp'
9] npm ERR! cb() never called!
9]
9] npm ERR! This is an error with npm itself. Please report this error at:
9] npm ERR!     <https://github.com/npm/npm/issues>
9]
```

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.



```
#npm install node-sass --verbose
# Load dependencies.
npm cache clean -f
npm install --verbose
# Build projects with default settings.
npm run build
#tar -zcvf demo.tar.gz ./**
```

6.6 gyp ERR! stack Error: EACCES: permission denied

Symptoms

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

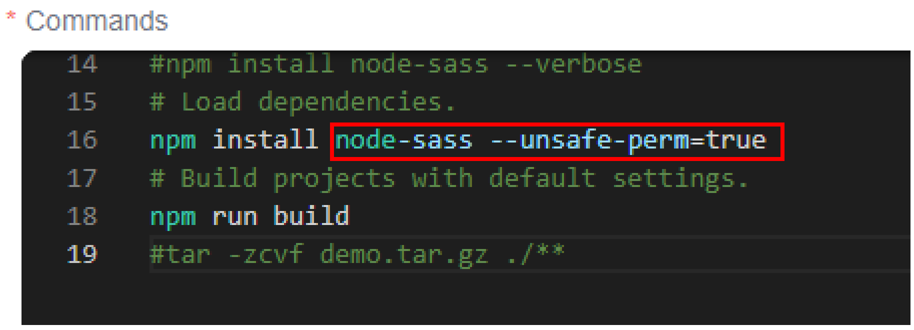
```
gyp ERR! stack Error: EACCES: permission denied, mkdir '******/
job_1451ba57-0c35-4daa-99c2-21425404f61c_1564043318112/saas_shop/node_modules/node-sass/.node-
gyp'
```

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.



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

6.7 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 */
   ```

6.8 Failed to Download node-sass

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 image source of Huawei Cloud and perform the build again.

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

6.9 error: could not write config file

Symptoms

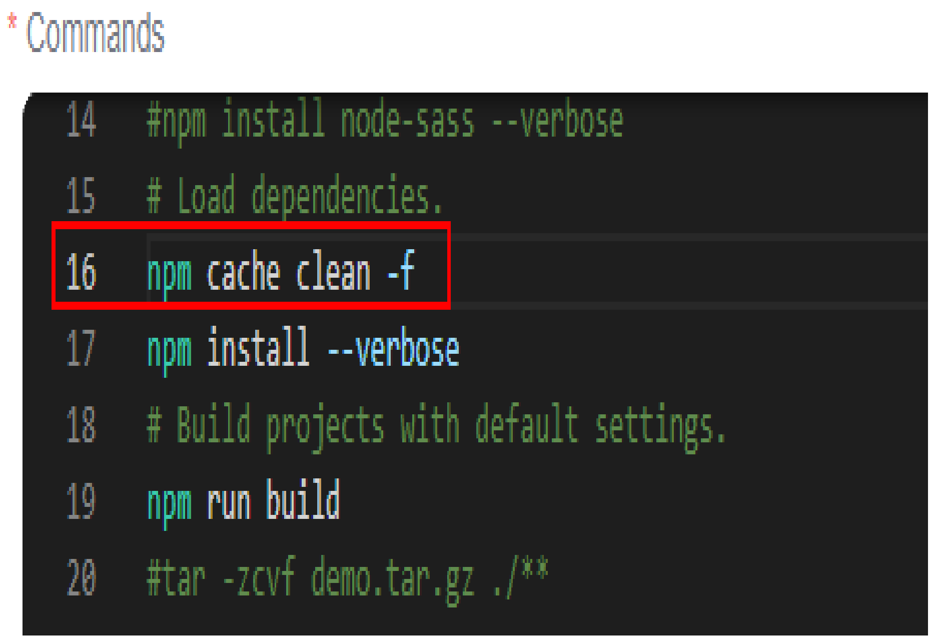
When an npm build task is executed, the following error information is reported in the log: **error: could not write config file /npmcache/_cacache/tmp/git-clone-b0ba91a1/.git/config: Disk quota exceeded**

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.



```
#npm install node-sass --verbose
# Load dependencies.
npm cache clean -f
npm install --verbose
# Build projects with default settings.
npm run build
#tar -zcvf demo.tar.gz ./**
```

6.10 Low Efficiency and Slow Dependency Installation During Build with npm

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**.

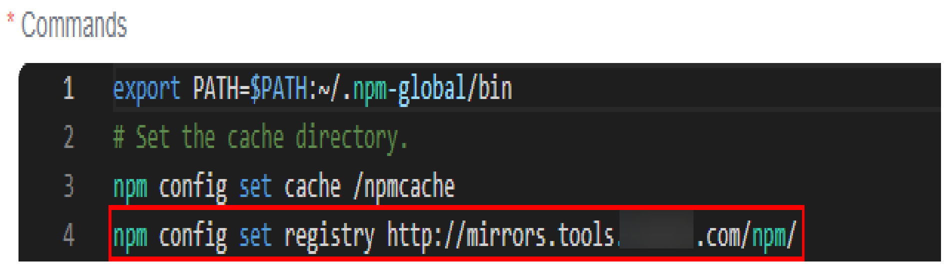
Step 3 On the **Build Actions** page, modify the **Build with npm** action.

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
```



```
export PATH=$PATH:~/.npm-global/bin
# Set the cache directory.
npm config set cache /npmcache
npm config set registry http://mirrors.tools.[redacted].com/npm/
```

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

----End

7 Using Docker for Build

7.1 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

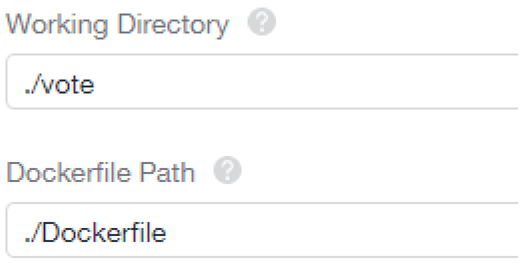
Assume that the structure of the current directory is as follows:

```
+ 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
  ./vote

  Dockerfile Path
  ./Dockerfile

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 example, 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.

7.2 Failed to Push Images to SWR

When you execute the action **Build and Push Images 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 login 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 valid login information 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.

7.3 Failed to Pull the Image

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.

7.4 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
```

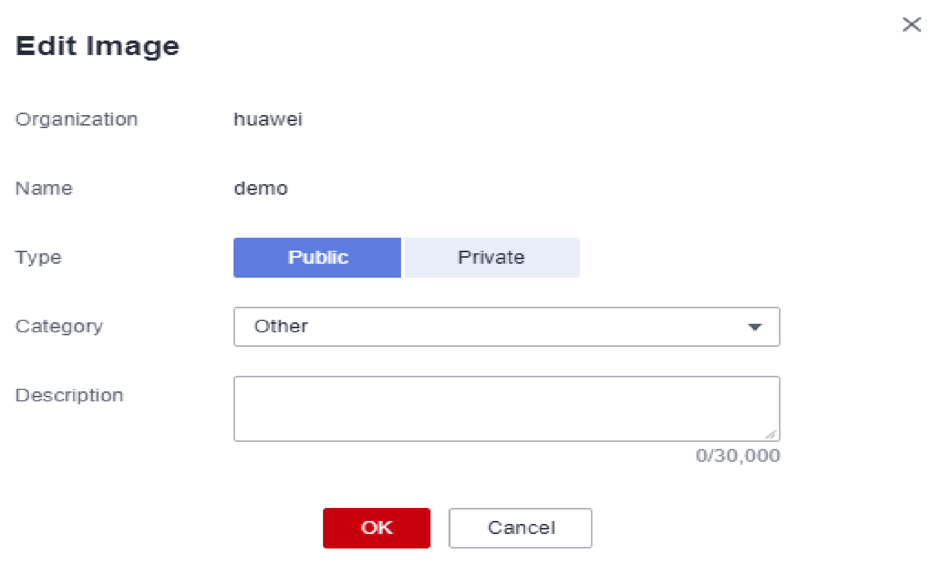
Cause Analysis

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, 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 huawei

Name demo

Type [✓ Public] Private

Category Other

Description

0/30,000

7.5 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.

8 Creating Images and Pushing to SWR

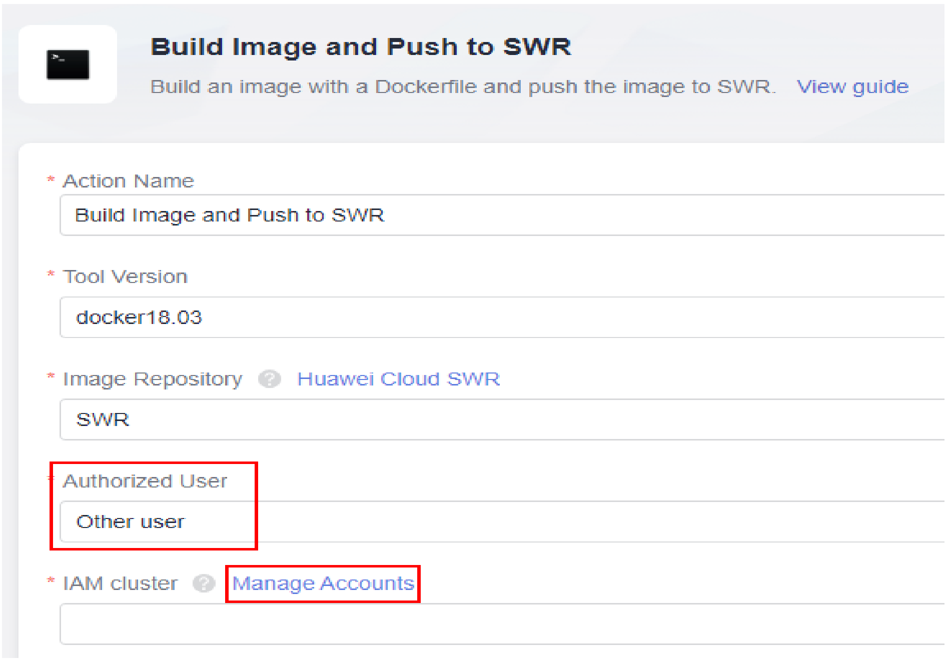
8.1 How Do I Push Images to Other Tenants?

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

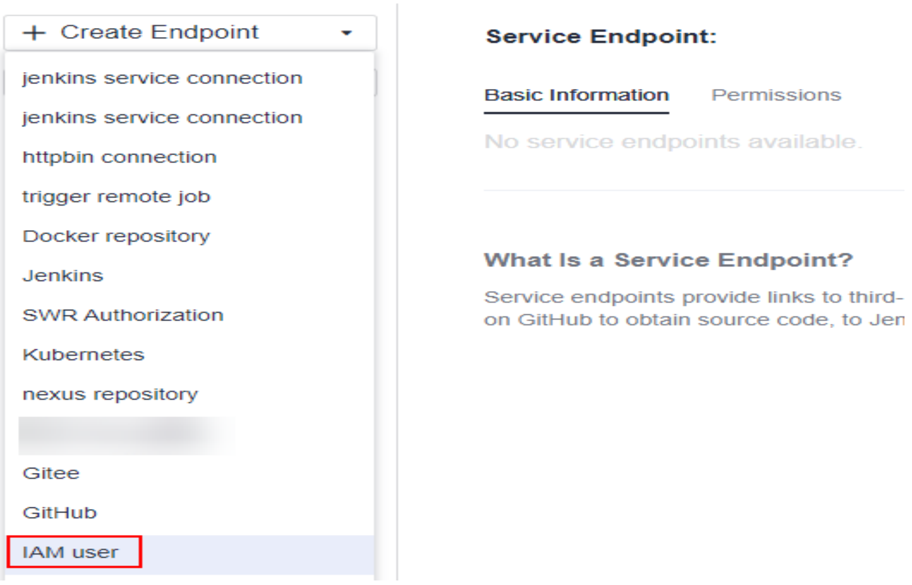
* Tool Version: docker18.03

* Image Repository: SWR

Authorized User: Other user

* IAM cluster

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



Create Endpoint

- jenkins service connection
- jenkins service connection
- httpbin connection
- trigger remote job
- Docker repository
- Jenkins
- SWR Authorization
- Kubernetes
- nexus repository
- Gitee
- GitHub
- IAM user

Service Endpoint:

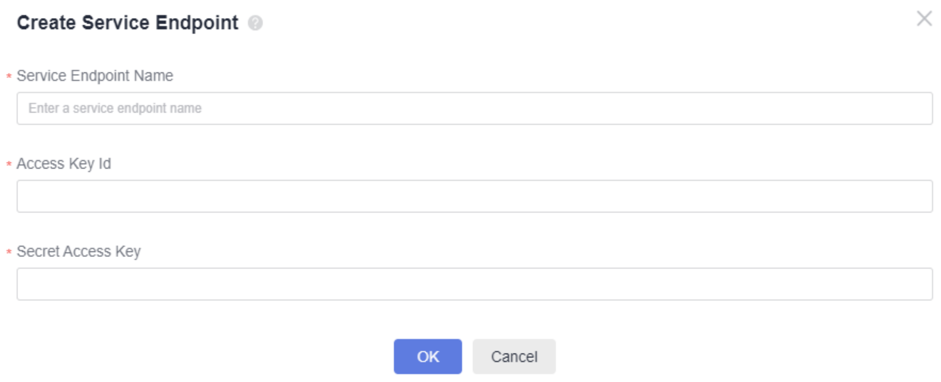
Basic Information | Permissions

No service endpoints available.

What Is a Service Endpoint?

Service endpoints provide links to third-
on GitHub to obtain source code, to Jen

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



Create Service Endpoint

* Service Endpoint Name: Enter a service endpoint name

* Access Key Id

* Secret Access Key

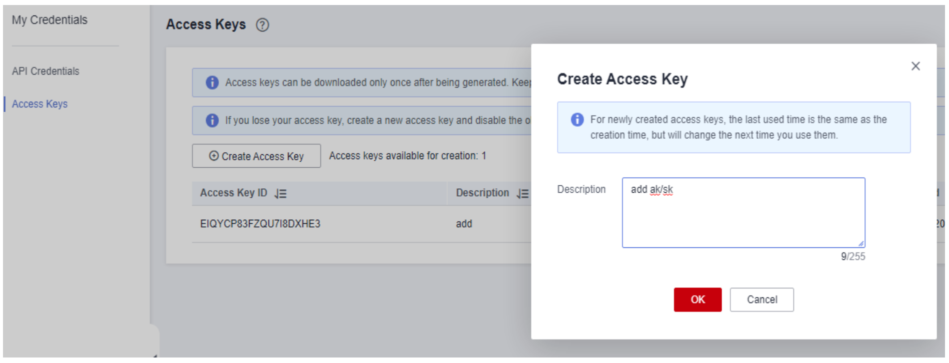
To obtain the access key ID and secret access key, perform the following steps:

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**.



My Credentials
- API Credentials
- Access Keys

Access Keys

Access keys can be downloaded only once after being generated. Keep

If you lose your access key, create a new access key and disable the o

Create Access Key | Access keys available for creation: 1

Access Key ID	Description
EIQYCP83FZQU7I8DXHE3	add

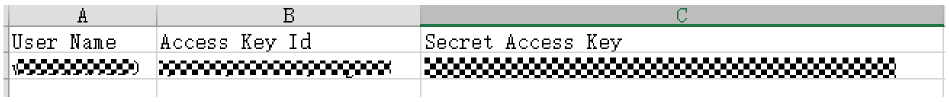
Create Access Key

For newly created access keys, the last used time is the same as the creation time, but will change the next time you use them.

Description: add ak/sk

9/255

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
[redacted]	[redacted]	[redacted]

7. Select the service endpoint created in **Step 6** for the IAM account in **Step 4**.

8.2 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

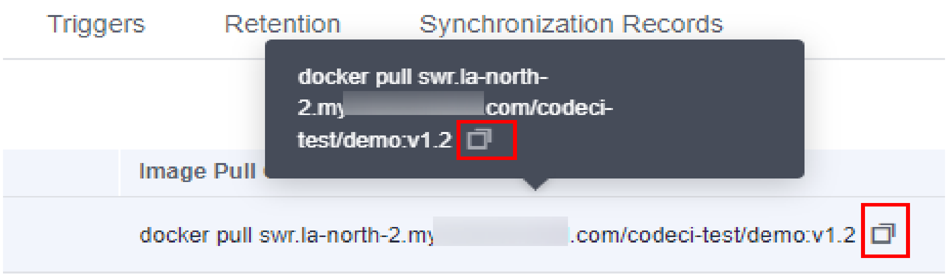
The network of the Docker Hub is unstable and the frequency is limited. As a result, the image pull may time out or fail. You can migrate the Docker Hub image to SWR and then pull the image.

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**.

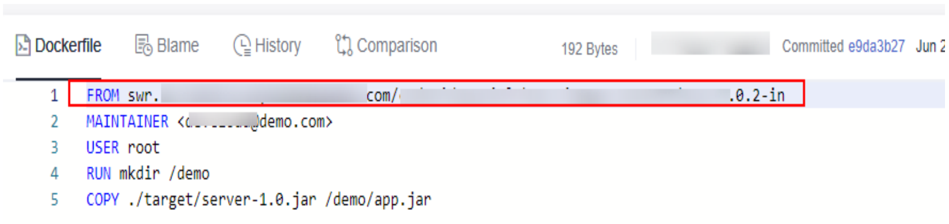
Step 3 On the **Tags** tab page, in the same row as the target image tag, click ⧉ in the **Image Pull Command** column to copy the command.



docker pull swr.la-north-2.my[redacted].com/codeci-test/demo:v1.2

docker pull swr.la-north-2.my[redacted].com/codeci-test/demo:v1.2

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



```
FROM swr.[redacted]com/[redacted].0.2-in
MAINTAINER <[redacted]demo.com>
USER root
RUN mkdir /demo
COPY ./target/server-1.0.jar /demo/app.jar
```

----End