CodeArts Check

Best Practices

Issue	01
Date	2025-08-01

HUAWEI TECHNOLOGIES CO., LTD.

Security Declaration

Vulnerability

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

Contents

1 Checking Code from Git with Preset Rules

Scenario

Check Java code from Git to protect quality.

Prerequisites

- You have obtained permissions of CodeArts Check.
- There is Java code in the Git repository.

Procedure

Table 1-1 Steps

No.	Step	Description
1	**Creating a Project**	Create a project.
2	**Creating a Git Service Endpoint**	Use a service endpoint to connect to a third-party repository.
3	**Creating a Task to Check Code from Git**	Create a task.
4	**Executing the Task**	Execute a task.
5	**Viewing Check Results**	View check results.

Creating a Project

Step 1 Log in to the Huawei Cloud console.

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

Step 3 Click **Access Service**.

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

Step 5 Click **OK** to access the project.

----End

Creating a Git Service Endpoint

A service endpoint is an extension to CodeArts and supports connection to third-party repositories.

With a service endpoint, CodeArts Check supports repositories either of CodeArts Repo and third-parties.

Step 1 Enter a task through a project. In the navigation pane, choose **Settings** > **General** > **Service Endpoints**.

Step 2 Click **Create Endpoint** and choose **Git repository** from the drop-down list.

Step 3 Configure the following information and click **Confirm**.

Table 1-2 Creating a Git Service Endpoint

Parameter	Description
Service Endpoint Name	Enter a maximum of 256 characters, including letters, digits, hyphens (-), underscores (_), periods (.), and spaces. For example, **Endpoint01**.
Git Repository URL	Enter the HTTPS address of the Git repository to connect.
Username	Enter the username of the Git repository to connect (max. 300 characters).
Password or Access Token	Enter the password of the Git repository to connect (max. 300 characters).

----End

Creating a Task to Check Code from Git

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

Step 2 Click **Create Task**. Set parameters by referring to the following table.

Table 1-3 Task parameters

Parameter	Description
Project	Project that the task belongs to. Retain the default value (the **Scrum01** project created in **Creating a Project**).
Code Source	Select **Git**.
Name	Enter a task name, for example, **CheckTask01**.
Endpoint	Select the **Endpoint01** service endpoint created in **Creating a Git Service Endpoint**.
Repository	Retain the default value.
Branch	Retain the default value **master**.
Language	Select the code language to be checked, for example, **Java**.

Step 3 Click **Create Task**.

----End

Executing the Task

Step 1 In the **Tasks** page, click ▷ to execute the task.

Step 2 Wait until the task is complete as prompted.

----End

Viewing Check Results

Step 1 In the **Tasks** page, search for the **CheckTask01** task created in **Creating a Task to Check Code from Git**.

Step 2 Click the task name to view the check details, including overview, issues, metrics, logs, and settings.

So far, we have completed a basic common check process for Git code sources.

----End

Related Operations

- For more configurations, see **Configuring a Task**.

- For general issues about executing tasks, see **General Issues**.

2 Checking Code from CodeArts Repo with Custom Rules

Scenario

As the code and development framework expand, the static analysis needs to cover additional scenarios. However, the following questions have also arisen:

- The traditional static analysis engines cannot offer real-time scenario-based code checks by relying solely on general rules.
- Users may not be familiar with all scenarios covered by general rules, making finding applicable rules for a newly developed service time-consuming.
- It is challenging to develop comprehensive and effective rules to fit different users and services.

This section describes how to use custom rules to check code.

Prerequisites

- You have **obtained permissions of CodeArts Check**.
- There is Java code in the Git repository.

Procedure

Table 2-1 Steps

No.	Step	Description
1	**Creating a Project**	Create a project.
2	**Creating a Code Repository in CodeArts Repo**	Create a code repository.

No.	Step	Description
3	Creating a Rule File	Create a rule file to be uploaded when a custom rule is created.
4	Creating a Custom Rule	Create a custom rule.
5	Creating a Custom Rule Set	Create a custom rule set to use custom rules.
6	Creating a Task	Create a task that uses custom rules.
7	Checking Code by Using a Custom Rule Set	Configure the task with the custom rule set.
8	Viewing Check Results	View the check results to check whether the rule takes effect.

Creating a Project

Step 1 Log in to the Huawei Cloud console.

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

Step 3 Click **Access Service**.

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

Step 5 Click **OK** to access the project.

----End

Creating a Code Repository in CodeArts Repo

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

Step 2 On the CodeArts Repo homepage, click **Create Repository**.

Step 3 On the displayed page, select **Template**.

Step 4 Click **Next** and select the **Java Maven Demo** template.

Step 5 Click **Next**. Set the repository name to **Repo01** and deselect **Automatically create check task**. Retain the default values for other parameters.

Step 6 Click **OK**.

Step 7 Modify the code information in the **HelloWorld.java** file in the **com/huawei** directory as follows:

```
package com.huawei;
/**
 * Generate a unique number
 *
 */
public class HelloWorld
{
//Used to print logs
  public void debugLog(List<String> msg) {
    for (String msg0 : msg) {
      System.out.println("DEBUG:"+ msg0);
    }
}
    public static void main( String[] args )
    {
      System.out.println("Hello World!");
    }
}
```

----End

Creating a Rule File

Step 1 Download and install the **Visual Studio Code IDE editor** (version 1.67.0 or later).

Step 2 On the IDE editor page, click ⊞ on the left and search for **Huawei Cloud CodeNavi** in the displayed window.

Step 3 Click **Install** to install this plug-in.

Step 4 Create a **.kirin** file in the editor workspace, for example, **CheckDebugCode.kirin**. The file content is as follows:

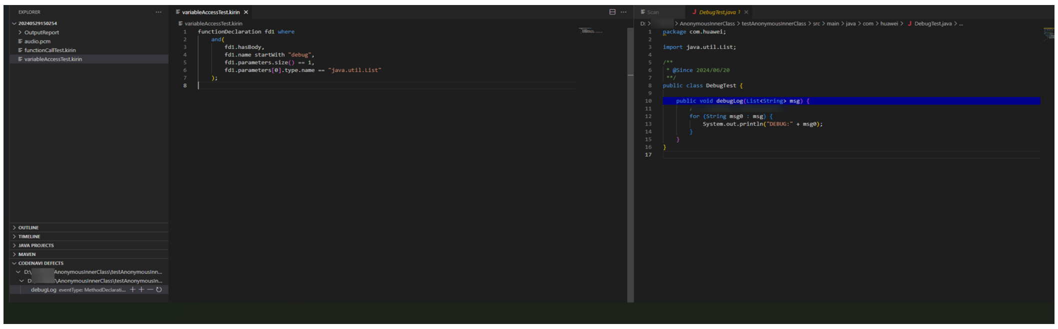
```
functionDeclaration fd1 where
    and(
        fd1.hasBody,
        fd1.name startWith "debug",
        fd1.parameters.size() == 1,
        fd1.parameters[0].type.name == "java.util.List"
    );
```

Step 5 Right-click the rule file and choose **CodeNavi** > **Format** to verify the syntax.

Step 6 Right-click the rule file and choose **CodeNavi** > **Scan**.

Step 7 In the displayed dialog box, select the file or directory to be checked and click **Scan**.

Step 8 After the scanning is complete, click the defects in the lower left corner of the page to display the specific code snippet. In addition, a rule file in **.json** format is generated in the **OutputReport** file in the same directory.



```
functionDeclaration fd1 where
    and(
        fd1.hasBody,
        fd1.name startWith "debug",
        fd1.parameters.size() == 1,
        fd1.parameters[0].type.name == "java.util.List"
    );
```

```java
package com.huawei;

import java.util.List;

/**
 * @Since 2024/06/20
 **/
public class DebugTest {

    public void debugLog(List<String> msg) {
        [redacted]
        for (String msg0 : msg) {
            System.out.println("DEBUG:" + msg0);
        }
    }
}
```

----End

Creating a Custom Rule

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

Step 2 Click the **Rules** tab.

Step 3 Click **Create Rule**. Set parameters by referring to **Table 2-2**.

Table 2-2 Rule parameters

Parameter	Description
Rule Name	Custom rule name. It can be customized. For example, **CheckDebugCode**.
Tool Rule Name	Rule source code file (by default).
Tool	Check tool used by a custom rule. Currently, only SecBrella is supported.
Language	Language checked by a custom rule. Currently, Java and ArkTS are supported.
Source Code	Rule source code file. Upload the file generated in **Creating a Rule File**.
Severity	Severity of a code issue detected by a rule. The value can be **Critical**, **Major**, **Minor**, or **Suggestion**. Set this parameter to **Suggestion**.
Tag	(Optional) Rule tag for different scenarios. **NOTE** Use commas (,) to separate multiple tags.
Description	Rule description. The content contains code in Markdown. Max. 10,000 characters. For example, check whether debugging code exists.
Compliant Example	(Optional) Compliant code example. The content contains code in Markdown. Max. 10,000 characters.

Parameter	Description
Noncompliant Example	(Optional) Noncompliant code example. The content contains code in Markdown. Max. 10,000 characters.
Fix Suggestions	(Optional) Issue fixing suggestions. The content contains code in Markdown. Max. 10,000 characters.

Step 4 Click **OK**.

----End

Creating a Custom Rule Set

Step 1 On the task list, click the **Rule Sets** tab.

Step 2 Click **Create Rule Set**. In the displayed window, set **Rule Set** to **RuleList** and **Language** to **JAVA**.

Step 3 Click **Confirm**.

Step 4 Select the rule created in **Creating a Custom Rule** and click **Save** in the upper right corner.

----End

Creating a Task

Step 1 On the task list page, click **Create Task** and set parameters by referring to the following table.

Table 2-3 Task parameters

Parameter	Description
Project	Retain the default value (the **Scrum01** project created in **Creating a Project**).
Code Source	Source of code. Select **Repo**.
Name	Enter a task name, for example, **CheckTask01**.
Repository	Select the **Repo01** code repository created in **Creating a Code Repository in CodeArts Repo**.
Branch	Retain the default value **master**.

Parameter	Description
Language	Select **Java**.

Step 2 Click **Confirm**.

----End

Checking Code by Using a Custom Rule Set

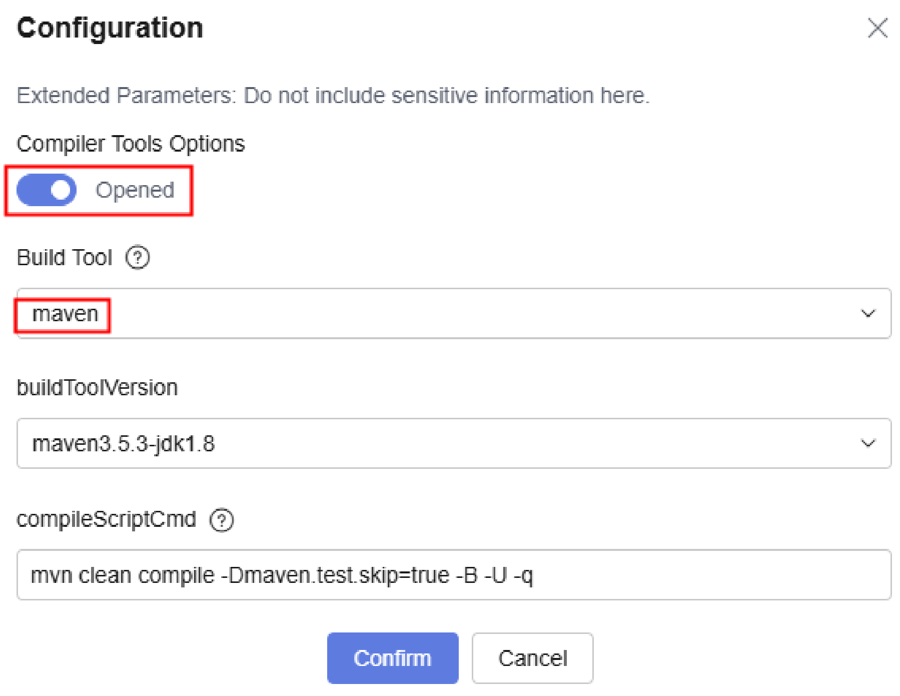
Step 1 In the **Tasks** page, click the task name.

Step 2 Click **Settings**.

Step 3 Click **Rule Sets**. In the right pane, click ○ to select the **RuleList** rule set created in **Creating a Custom Rule Set**.

Step 4 Click **Configuration**, set **Compiler Tools Options** to the enabled toggle, and set **Build Tool** to **maven**. Retain the default values for other parameters and click **Confirm**.

Figure 2-1 Configuration


Configuration

Extended Parameters: Do not include sensitive information here.

Compiler Tools Options

Opened

Build Tool

maven

buildToolVersion

maven3.5.3-jdk1.8

compileScriptCmd

mvn clean compile -Dmaven.test.skip=true -B -U -q

Step 5 Click **Start Check** in the upper right corner.

----End

Viewing Check Results

Step 1 In the **Tasks** page, search for the **CheckTask01** task created in **Creating a Task**.

Step 2 Click the task name to view the check details, including overview, issues, metrics, logs, and settings.

----End

Related Operations

- For more configurations, see **Configuring a Task**.
- For general issues about executing tasks, see **General Issues**.

3 Checking Code with Custom Executors

Scenario

You can register your own executors with CodeArts Check to schedule and execute check tasks. This section describes how to use a custom executor to check code from CodeArts Repo.

This practice depends on **CodeArts Repo** to store the code.

Constraints

- To use your custom executors, contact technical support.
- You have **permissions for CodeArts Repo**.

Resource and Cost Planning

In this practice, you need to purchase an ECS as a custom executor. For details about the price of an ECS, see **Price Calculator**.

Prerequisites

- You have **purchased an ECS**.

  NOTE

  Only EulerOS 2.5 is supported for custom executors.

- You have installed Git-LFS on the custom executor. If not, install it by referring to the following command-based example.

  Run the following commands on the executor:
  ```
  # Download
  wget -O  git-lfs.tar.gz https://github.com/git-lfs/git-lfs/releases/download/v3.4.1/git-lfs-linux-amd64-v3.4.1.tar.gz
  # Decompress the package
  tar -zxvf git-lfs.tar.gz
  # Open the directory generated after the decompression
  cd  git-lfs-3.4.1
  # Run the installation script
  sh install.sh
  # Verify
  git lfs version
  ```

- You have **attached an EVS disk**.

Procedure

Table 3-1 Steps

Step	Description
Creating a Project	Create a project.
Creating an Agent Pool	Create a pool of custom executors (agent pool).
Creating a CodeArts Repo Repository	Create a repository to store code.
Configuring and Executing a Check Task	Configure the task to use the custom executor.
Viewing Check Results	View the check logs to verify the executor used for the task.

Creating a Project

Step 1 Log in to the Huawei Cloud console.

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

Step 3 Click **Go to CodeArts Check** to go to the CodeArts Check homepage.

Step 4 In the navigation pane, choose **Homepage**. Click **Create** > **Create Project**, and select the **Scrum** template.

Step 5 Enter the project name, for example, **check-bestpractice**. Retain the other parameters as default.

Step 6 Click **OK** to access the project.

----End

Creating an Agent Pool

Step 1 On the navigation bar, click the username and choose **All Account Settings**.

Step 2 Choose **Agent Management** > **Agent Pool**.

Step 3 Click **Create Pool**. In the displayed dialog box, set parameters according to **Table 3-2** and click **Save**.

Table 3-2 Agent pool configuration

Parameter	Description
Pool Name	Assign a custom name to the pool, for example, **custom_pool**.
Pool Type	Select **LINUX_DOCKER**. When a task is initiated, a Linux Docker container will be started to run the task.
Description	(Optional) Enter additional information to describe the pool.
This pool can be used by all users of the current account.	(Optional) Selecting this option allows all users within the current account to use the pool.

Step 4 Click the name of the new pool (**custom_pool** is used in this practice). The pool configuration page is displayed.

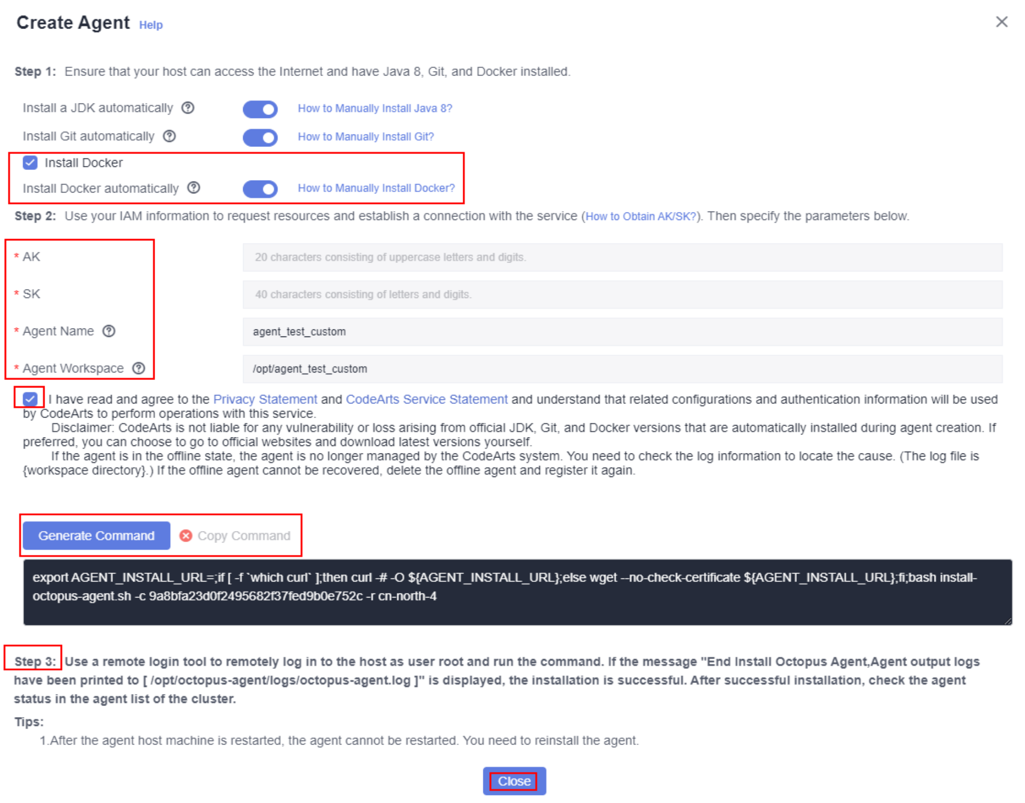
Step 5 Click **Create Agent**. In the displayed dialog box, configure the agent according to **Table 3-3** and leave the other parameters as default.

Table 3-3 Parameters for creating an agent

Parameter	Description
Install Docker	Selecting this option mandates Docker installation.
Install Docker automatically	Toggling on the switch will automatically install Docker.
AK	**Obtain an AK**.
SK	**Obtain an SK**.
Agent Name	Assign a custom name to the agent, for example, **agent_test_custom**.
Agent Workspace	Enter an agent workspace that follows the standard Linux directory structure. For example, **/opt/agent_test_custom**.

Step 6 Select the check box to confirm that you have read and accept the agreements. Then click **Generate Command** and **Copy Command**. Click **Close**.

Figure 3-1 Creating an agent



Step 1: Ensure that your host can access the Internet and have Java 8, Git, and Docker installed.

Install a JDK automatically
How to Manually Install Java 8?
Install Git automatically
How to Manually Install Git?
Install Docker
Install Docker automatically
How to Manually Install Docker?

Step 2: Use your IAM information to request resources and establish a connection with the service (How to Obtain AK/SK?). Then specify the parameters below.

* AK	20 characters consisting of uppercase letters and digits.
* SK	40 characters consisting of letters and digits.
* Agent Name	agent_test_custom
* Agent Workspace	/opt/agent_test_custom

I have read and agree to the Privacy Statement and CodeArts Service Statement and understand that related configurations and authentication information will be used by CodeArts to perform operations with this service.
Disclaimer: CodeArts is not liable for any vulnerability or loss arising from official JDK, Git, and Docker versions that are automatically installed during agent creation. If preferred, you can choose to go to official websites and download latest versions yourself.
If the agent is in the offline state, the agent is no longer managed by the CodeArts system. You need to check the log information to locate the cause. (The log file is {workspace directory}.) If the offline agent cannot be recovered, delete the offline agent and register it again.

```
export AGENT_INSTALL_URL=;if [ -f `which curl` ];then curl -# -O ${AGENT_INSTALL_URL};else wget --no-check-certificate ${AGENT_INSTALL_URL};fi;bash install-octopus-agent.sh -c 9a8bfa23d0f2495682f37fed9b0e752c -r cn-north-4
```

Step 3: Use a remote login tool to remotely log in to the host as user root and run the command. If the message "End Install Octopus Agent,Agent output logs have been printed to [/opt/octopus-agent/logs/octopus-agent.log]" is displayed, the installation is successful. After successful installation, check the agent status in the agent list of the cluster.

Tips:
1.After the agent host machine is restarted, the agent cannot be restarted. You need to reinstall the agent.

Step 7 Go to the ECS list page, find the row of the ECS purchased to meet **prerequisites**, click **Remote Login**, and run the command copied in **Step 6**, as instructed by **Step 3**.

Step 8 On the agent list page, click **Refresh List**. After the information is automatically synchronized in the background, a new item will be added to the list. The agent alias is **agent_test_custom-mwlye1NlLG**.

----End

Creating a CodeArts Repo Repository

Step 1 In the navigation pane, choose **Code** > **Repo** to go to the CodeArts Repo page of the **check-bestpractice** project.

Step 2 Click **Create Repository** and select **CR**.

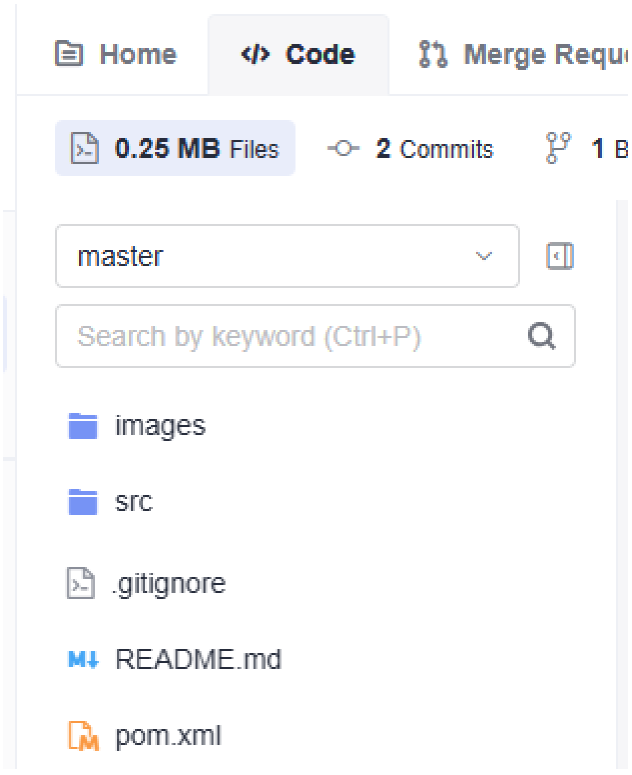
Step 3 On the displayed page, select **Template** and click **Next**.

Step 4 On the template selection page, select the **Java Maven Demo** template and click **Next**.

Step 5 On the displayed page, set **Repository Name** to **custom_repo**, select **Automatically create Check task**, and leave the other parameters as default. Click **OK**.

Figure 3-2 shows the directory that stores files of the code repository.

Figure 3-2 Directory that stores files of the code repository



----End

Configuring and Executing a Check Task

Step 1 In the navigation pane, choose **Code** > **Check**. The task is displayed on the task list page, because you have selected **Automatically create Check task** when creating a cloud repository in CodeArts Repo.

Step 2 Click the task name. On the displayed page, choose **Settings** > **Custom Environment**.

Step 3 In the **Execution Host** area, select **Self-Hosted**.

Step 4 Expand the drop-down list and select **custom_pool** you created in **Creating an Agent Pool**.

Step 5 Click **Save** and click **Start Check**.

----End

Viewing Check Results

Click the task name. On the displayed task details page, click **Logs**. If the logs contain "Find available executor node:agent_test_custom-mwlye1NlLG," the task is executed by a custom executor. **agent_test_custom-mwlye1NlLG** is the agent alias mentioned in **Step 8**.

Related Operations

- For more configurations, see **Configuring a Task**.
- For general issues about executing tasks, see **General Issues**.

4 Executing a Task Securely

Scenario

The enhanced package offers a robust security check feature that thoroughly detects code risks and vulnerabilities. It also covers unique risk scenarios not available in the edition packages, for example, value errors, encryption issues, and data verification issues. Moreover, it strengthens vulnerability analyses for detection items (such as cross-function check, cross-file check, taint analysis, semantic analysis).

Resource and Cost Planning

Purchase the CodeArts Check enhanced package by referring to **Purchasing a Value-Added Feature**. For details about the price, see **Price Calculator**.

Procedure

Table 4-1 Procedure

Step	Description
Creating a Project	Create a project.
Creating a CodeArts Repo Repository	Create a code repository.
Configuring a Rule Set to Execute a Task	Configure a rule set with the security enhanced package to a task.
Viewing Check Results	View the check results to check whether the rule takes effect.

Creating a Project

Step 1 Log in to the Huawei Cloud console.

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

Step 3 Click **Go to CodeArts Check** to go to the CodeArts Check homepage.

Step 4 In the navigation pane, choose **Homepage**. Click **Create** > **Create Project**, and select the **Scrum** template.

Step 5 Enter the project name, for example, **check-bestpractice**. Retain the other parameters as default.

Step 6 Click **OK** to access the project.

----End

Creating a CodeArts Repo Repository

Step 1 In the navigation pane, choose **Code** > **Repo** to go to the CodeArts Repo page of the **check-bestpractice** project.

Step 2 Click **Create Repository**.

Step 3 On the displayed page, select **Template** and click **Next**.

Step 4 On the template selection page, select the **Java Maven Demo** template and click **Next**.

Step 5 Set **Repository Name** to **Repo01**, select **Automatically create check task**, and retain the other parameters as default. Click **OK**.

----End

Configuring a Rule Set to Execute a Task

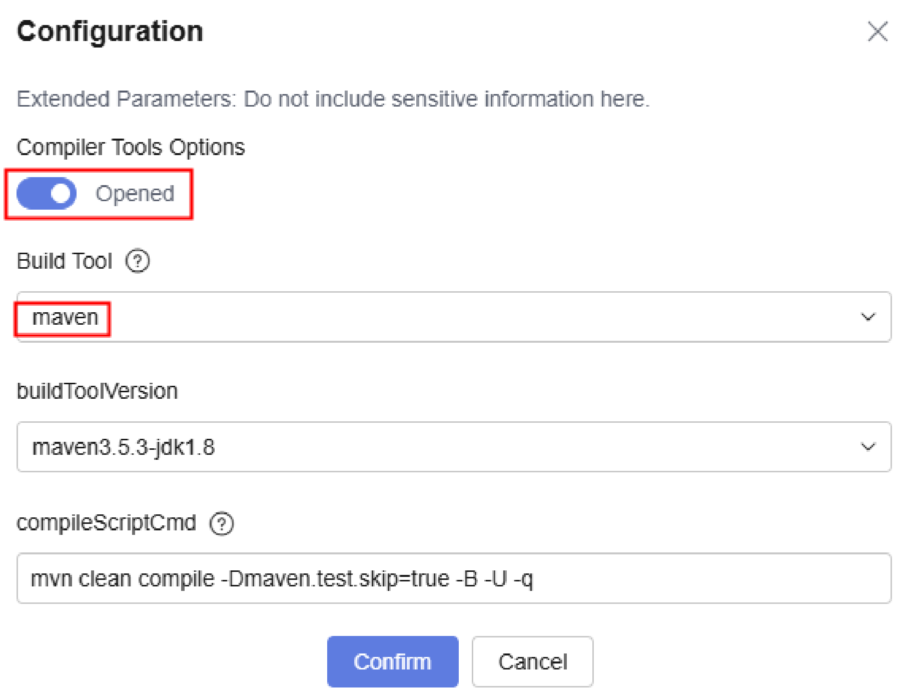
Step 1 The task is displayed on the task list page, because you have selected **Automatically create Check task** when creating a cloud repository in CodeArts Repo. On the **Tasks** page, click the task name.

Step 2 Click **Settings**.

Step 3 Click **Rule Sets** and click ○ in the upper right corner of **Huawei Java Enhanced Coding Standard Rule Set**.

Step 4 Click **Configuration**, set **Compiler Tools Options** to ●, and set **Build Tool** to **maven**. Retain the default values for other parameters and click **Confirm**.

Figure 4-1 Configuration


Configuration

Extended Parameters: Do not include sensitive information here.

Compiler Tools Options

Opened

Build Tool

maven

buildToolVersion

maven3.5.3-jdk1.8

compileScriptCmd

```
mvn clean compile -Dmaven.test.skip=true -B -U -q
```

Step 5 Click **Start Check** in the upper right corner.

----End

Viewing Check Results

If issues checked by **Huawei Java Enhanced Programming Rule Set** are displayed, this rule set is used for the task.

Related Operations

For more rule set configurations, see **Configuring a Rule Set**.

5 Huawei E2E DevOps Practice: Checking Code

This section takes a DevOps full-process sample project as an example to describe how to configure a check task in a project.

Preset Tasks

The sample project presets four code check tasks.

Table 5-1 Preset tasks

Preset Task	Description
phoenix-codecheck-worker	Checks the Worker function code.
phoenix-codecheck-result	Checks the Result function code.
phoenix-codecheck-vote	Checks the Vote function code.
phoenix-sample-javas	Checks the JavaScript code of the entire code repository.

This section uses the **phoenix-codecheck-worker** task as an example.

Configuring and Executing a Task

Developers can slightly adjust preset tasks in the sample project to make the check more comprehensive.

This practice uses the Python check rule set as an example.

Step 1 Go to the **Phoenix Mall** project, and choose **Code** > **Check**. The preset four tasks are displayed.

Step 2 Find the **phoenix-codecheck-worker** task in the list, click ··· in the **Operation** column, and choose **Settings**.

Step 3 In the navigation pane, choose **Rule Sets**. The default language of each rule set is Java.

Step 4 Add the Python rule set.

1. Click next to **Languages Included** to refresh the language list.
2. Enable Python by setting the switch to the status.

The rule set is configured.

Step 5 Click **Start Check** to start the task.

If Success is displayed, the task is successfully executed.

If the task fails, rectify the fault by referring to **CodeArts Check FAQs**.

----End

Viewing the Code Check Result

CodeArts Check collects check results and provides fix suggestions for detected issues. Optimize the project code based on the suggestions.

Step 1 On the task details page, click the **Overview** tab to view the result statistics.

Step 2 Click the **Issues** tab to view the issue list.

Click **Help** in the question box to view fix suggestions. You can find the corresponding file and code location in the code repository, and optimize the code based on the fix suggestions.

Figure 5-1 Viewing help information

----End