CodeArts Check

Best Practices

Issue	01
Date	2024-01-16

HUAWEI TECHNOLOGIES CO., LTD.

Trademarks and Permissions

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

Notice

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

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

Security Declaration

Vulnerability

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

Contents

1 Checking Code from Git with Preset Rules

Scenario

Check Java code from Git to protect quality.

Preparation

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

Process

Table 1-1 Process

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	Customize 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.

----End

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, which makes 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.

Preparation

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

Process

Table 2-1 Process

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	**Customizing a Rule**	Create a custom rule.
5	**Customizing a 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 **New Repository** and select **Template**.

Step 3 Click **Next**, and search for and select the **Java Ant Demo** template.

Step 4 Click **Next**. Set the repository name to **Repo01** and deselect **Automatically create Check task**. Retain the default values for other parameters.

Step 5 Click **OK**.

Step 6 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:

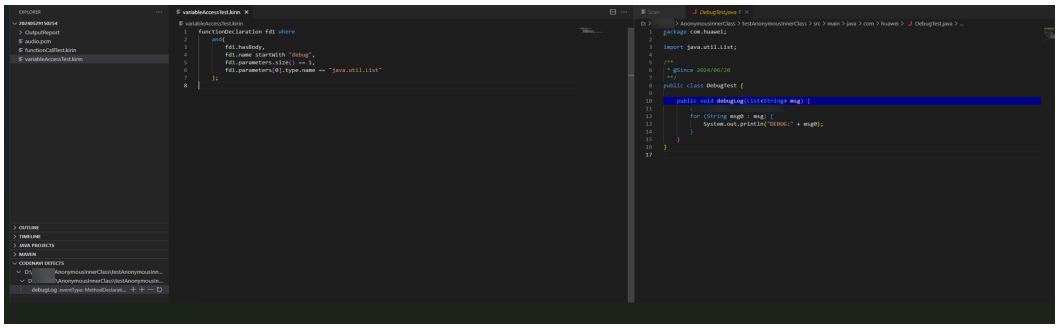
```
functionDeclaration fd1 where
    and(
        fd1.hasBody,
        fd1.name startWith "debug",
        fd1.parameters.size() == 1,
        fd1.parameters[0].type.name == "java.util.List"
    );
```

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.



----End

Customizing a 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, only Java is 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.
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 **Create Rule**.

----End

Customizing a 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 **OK**.

Step 4 Select the rule created in **Customizing a 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	Customize 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**.
Language	Select **Java**.

Step 2 Click **Create Task**.

----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 **Customizing a Rule Set**.

Step 4 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

3 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 has four preset code check tasks.

Table 3-1 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

For comprehensive checks, developers can add some simple configurations (for example, a Python check rule set) to the preset code check task.

Step 1 Edit a task.

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

2. Find the **phoenix-codecheck-worker** task in the list.
3. Click the task name to go to the details page and click the **Settings** tab.
4. In the navigation pane, choose **Rule Sets**. The default language of a rule set is Java.
5. Add the Python language check rule set.
   a. Click next to **Languages Included** to refresh the language list.

      NOTE

      If Python is displayed on the page, skip this step.

   b. Click to enable the Python language.
   c. In the dialog box that is displayed, click **OK**.

Step 2 Execute the task.

1. Click **Start Check** to start the task.
2. If Success is displayed on the page, the task is successfully executed.

   If the task fails, check and fix errors based on the message displayed on the page.

----End

Viewing Check Results

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 as required and optimize the code based on the fix suggestions.

----End

