

CodeArts Check

Best Practices

Issue 01
Date 2024-01-16



Copyright © Huawei Technologies Co., Ltd. 2024. All rights reserved.

No part of this document may be reproduced or transmitted in any form or by any means without prior written consent of Huawei Technologies Co., Ltd.

Trademarks and Permissions



HUAWEI and other Huawei trademarks are trademarks of Huawei Technologies Co., Ltd.

All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

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

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

Security Declaration

Vulnerability

Huawei's regulations on product vulnerability management are subject to the *Vul. Response Process*. For details about this process, visit the following web page:

<https://www.huawei.com/en/psirt/vul-response-process>

For vulnerability information, enterprise customers can visit the following web page:

<https://securitybulletin.huawei.com/enterprise/en/security-advisory>

Contents

1 Performing a Check Task from CodeArts Repo.....	1
2 Performing a Check Task from Git.....	4
3 Huawei E2E DevOps Practice: Checking Code.....	7

1 Performing a Check Task from CodeArts Repo

By checking code vulnerabilities and non-standard coding, CodeArts Check improves code quality and instructs developers to form proper coding habits.

This practice describes how to create, execute, and view a check task from CodeArts Repo and how to configure rule sets and tasks.

A common process is as follows:

- [Step 1: Create a task to check code from Repo.](#)
- [Step 2: \(Optional\) Create a custom rule set.](#)
- [Step 3: \(Optional\) Configure a rule set.](#)
- [Step 4: Execute a check task.](#)
- [Step 5: View check details.](#)

Preparations

- [Create a Project](#) (using a Scrum project as example).
- [Create a code repository](#) in the project. (Do not select **Automatically create Check task** in this example.)

Step 1: Create a task to check code from Repo.

After the [preparations](#) are complete, you can create a task.

Step 1 [Log in to CodeArts.](#)

Step 2 Search for a project created in [preparations](#) and click the project name.

Step 3 In the navigation tree, choose **Code > Check**.

Step 4 Click **Create Task**.

The code source is Repo by default. Select the code repository created in [preparations](#).

Step 5 Click **Create Task**.

----End

Step 2: (Optional) Create a custom rule set.

After creating a check task, you can create a custom rule set for the task.

Step 1 Choose **Code > Check**.

Step 2 Click **Rule set**.

Step 3 Click **Create Rule Set**. In the displayed dialog box, enter a rule set name and select a language.

Step 4 Click **OK**.

Step 5 Select the required rule names, set issue levels, and click **Save** in the upper right corner.

----End

Step 3: (Optional) Configure a rule set.

Before executing a task, change the rule set to a custom rule set as required.

Step 1 Choose **Code > Check**.

Step 2 In the **Tasks** page, click the task name created in [step 1](#).

Step 3 Choose **Settings > Rule Sets**.

Step 4 In the **Languages Included** area, enable the target language. In the **Enable Rule Set** area, click the rule set created in [step 2](#).

Step 5 In the displayed dialog box, click **OK** to change the language rule set.

----End


Step 4: Execute a check task.

After the rule set is configured, the check task is executed based on the custom rule set.

Step 1 Choose **Code > Check**.

Step 2 In the **Tasks** page, click **Execute immediately** in the **Last Check** column.

 **NOTE**

If the check task needs to be executed again, click  in the row where the task is located to execute the task again.

Step 3 Wait until the task is complete as prompted. For details about the check result, see [Step 5: View check details..](#)

----End

Step 5: View check details.

After the check task is executed, you can view check details.

Step 1 Choose **Code > Check**.

Step 2 In the **Tasks** page, search for the task created in **step 1**.

Step 3 Click the task name to view the check details, including overview, issues, metrics, logs, and settings.

----End

2 Performing a Check Task from Git

By checking code vulnerabilities and non-standard coding, CodeArts Check improves code quality and instructs developers to form proper coding habits.

This practice describes how to create, execute, and view a check task from Git and how to configure rule sets and tasks.

A common process is as follows:

- **Step 1: Create a task to check code from Git.**
- **Step 2: (Optional) Create a custom rule set.**
- **Step 3: (Optional) Configure a rule set.**
- **Step 4: Execute a check task.**
- **Step 5: View check details.**

Preparations

- **Create a Project** (using a Scrum project as example).
- **Creating a Git Service Endpoint.**

Step 1: Create a task to check code from Git.

After the **preparations** are complete, you can create a task.

Step 1 **Log in to CodeArts.**

Step 2 Search for a project created in **preparations** and click the project name.

Step 3 In the navigation tree, choose **Code > Check**.

Step 4 Click **Create Task**.

Select the **Git** code source as well as the endpoint created in **preparations** and configure the **Repository**, **Branch**, and **Language**.

Step 5 Click **Create Task**.

----End

Step 2: (Optional) Create a custom rule set.

After creating a check task, you can create a custom rule set for the task.

Step 1 Choose **Code > Check**.

Step 2 Click **Rule set**.

Step 3 Click **Create Rule Set**. In the displayed dialog box, enter a rule set name and select a language.

Step 4 Click **OK**.

Step 5 Select the required rule names, set issue levels, and click **Save** in the upper right corner.

----End

Step 3: (Optional) Configure a rule set.

Before executing a task, change the rule set to a custom rule set as required.

Step 1 Choose **Code > Check**.

Step 2 In the **Tasks** page, click the task name created in [step 1](#).

Step 3 Choose **Settings > Rule Sets**.

Step 4 In the **Languages Included** area, enable the target language. In the **Enable Rule Set** area, click the rule set created in [step 2](#).

Step 5 In the displayed dialog box, click **OK** to change the language rule set.

----End


Step 4: Execute a check task.

After the rule set is configured, the check task is executed based on the custom rule set.

Step 1 Choose **Code > Check**.

Step 2 In the **Tasks** page, click **Execute immediately** in the **Last Check** column.

 **NOTE**

If the check task needs to be executed again, click  in the row where the task is located to execute the task again.

Step 3 Wait until the task is complete as prompted. For details about the check result, see [Step 5: View check details..](#)

----End

Step 5: View check details.

After the check task is executed, you can view check details.

Step 1 Choose **Code > Check**.

Step 2 In the **Tasks** page, search for the task created in **step 1**.

Step 3 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 task corresponding to the Worker function code.
phoenix-codecheck-result	Checks the task corresponding to the Result function code.
phoenix-codecheck-vote	Checks the task corresponding to the Vote function code.
phoenix-sample-javas	Checks the task corresponding to the JavaScript function code.


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** 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 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 fixing 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