CodeArts Pipeline

FAQS

Issue 01

Date 2023-11-14

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

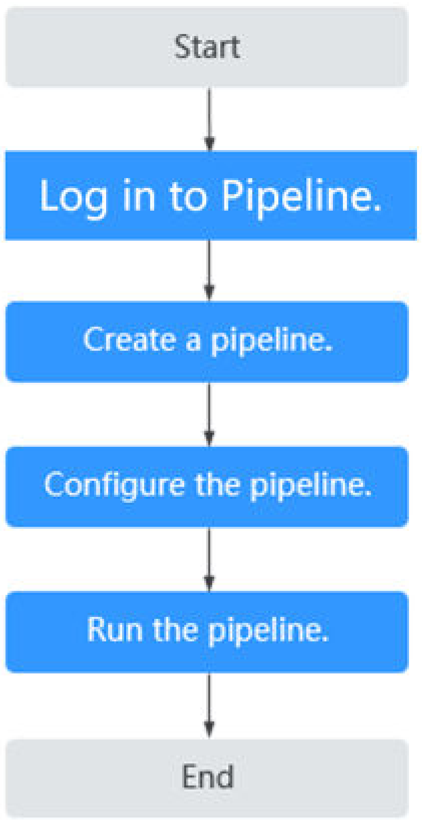
1 Functions

1.1 How Do I Use a Pipeline?

CodeArts Pipeline provides visualized continuous integration and continuous delivery (CI/CD) pipelines that can be orchestrated. It helps enterprises quickly realize continuous delivery and efficient automation in DevOps, shortens the time to market (TTM) of applications, and improves R&D efficiency.

Procedure

You can create, configure, and run a pipeline.



The following table describes the process.

Step	Description
Log in to the CodeArts Pipeline	If you have logged in to the service, skip this step. Log in to the CodeArts homepage. On the top navigation bar, choose **Services** > **Pipeline**.
Create a pipeline	Click **Create Pipeline**. You can create a pipeline using a system or custom template.
Configure the pipeline	You can configure the basic information, task orchestration (such as pipeline source, stages, tasks, and pass conditions), parameters, execution plan, permissions, and notifications.
Run the pipeline	After the setting is complete, click **Save and Run**.

1.2 What Are Pass Conditions?

CodeArts Pipeline provides unified pass condition management. Pass conditions are quality metric thresholds used for automated monitoring in a pipeline. You can configure rules and policies as pass conditions to control pipeline execution.

Currently, only pass conditions of standard policies are supported. When configuring a pipeline, click Pass Conditions under a stage. In the **Pass Conditions** dialog box, move the cursor to the pass condition card and click **Add** to add the pass condition for this stage, and configure a policy for the pass condition.

- Rules: Set the comparison relationship and threshold criteria for creating and editing policies based on output thresholds of extensions. The comparison relationship and threshold criteria are applied as pipeline pass conditions. Currently, the following extensions are supported:
  - Check: You can set thresholds for the problems found in code check results.

    If the number of problems is less than or equal to the threshold, the code check is passed and the pipeline continues to run. Otherwise, the code check fails and the pipeline execution is terminated.
  - TestPlan: You can set the test pass rate thresholds for APIs in the test suite.

    When the pass rate is greater than or equal to the threshold, the test is passed, and the pipeline continues to run. Otherwise, the test fails, and execution of the pipeline is terminated.
- Policies: a set of rules. Each rule corresponds to a condition template of the output metric value in an extension. You can select policies during pipeline orchestration as pass conditions to control pipeline execution.

NOTE

Currently, pass conditions are supported only in LA-Mexico City2.

1.3 What Tasks Can Be Run on a Pipeline?

The following tasks can be added to a pipeline:

- Build
- Check
- Deploy
- Test
- Normal (subpipeline, repository tag creation, Jenkins task, manual review, delayed execution, and pipeline suspension)

NOTE

API test tasks are supported only in LA-Mexico City2.

Third-party tasks can be scheduled through build, code check, deployment, and test tasks. Subpipeline can invoke other pipelines. Repository tags can be created for Repo code repositories and pushed for version management. Jenkins tasks enable you to schedule and execute specified tasks on Jenkins instances. Delayed execution allows a pipeline to wait for a user-defined period before being executed. Manual review allows a pipeline to continue execution only after being approved by the specified person. You can also suspend a pipeline.

1.4 Does CodeArts Pipeline Display the Execution History of Each Pipeline?

Yes.

On the CodeArts Pipeline homepage, click the target pipeline. The pipeline execution history tab page is displayed.

You can click the date picker in the upper right corner to view execution records by time. By default, executions in the last 31 days are displayed. You can switch among last 7 days, 14 days, and 31 days.

1.5 Can Multiple Packages Be Installed in a Certain Sequence?

In microservice scenarios, if a Java project contains multiple modules, multiple packages will be packed for these modules at a time. In addition, these packages need to be deployed in sequence during deployment. You can perform the following operations:

1. Create a code repository.
    a. Go to CodeArts Repo, click **New Repository**.

   b. Enter a repository name and click **OK**.
   c. Upload the Java project code to the created repository by referring to **Uploading Local Code to CodeArts Repo**.
2. Create a build task.
   a. Go to CodeArts Build, click **Create Task**.
   b. On the **Set Basic Information** page, select the code source **Repo**, the created code repository, and the branch **master**. Select the **Maven** template, and click **Next**.
   c. On the **Build Actions** tab, retain the default build actions. For details about how to configure build actions, see **Build Actions**.
   d. After the configuration is complete, click **Create and Run**. After the execution is complete, you can view the generated software package in the release repository.
3. Create a deployment task.

   Add the action **Select Deployment Source**, select the build task created in the previous step and the build package uploaded to the release repository, and download the build package to the specified path on the server. For details, see **Selecting a Deployment Source**.

   Add the action **Start/Stop Spring Boot** to start a specified JAR package. For details, see **Starting or Stopping Spring Boot**.
4. Create a pipeline. Add the build and deployment tasks to the pipeline and run the pipeline.

2 Troubleshooting

2.1 The Pipeline Parameter Referenced by the Task Does Not Exist

Background

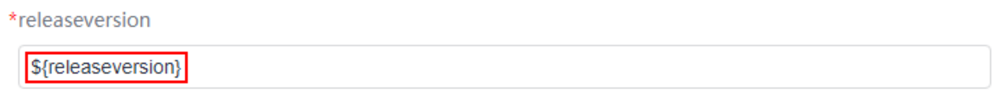
Pipeline parameters can be transferred to each task using the **${*parameter name*}** format. If a parameter associated by using **${*parameter name*}** is not on the pipeline parameter list, a message is displayed, indicating that the parameter is not specified.

Symptom

After you edit a pipeline and click **Save**, a message indicates that a pipeline parameter referenced by the task does not exist.

Root Cause

1. Click the name of the task for which the error is reported. On the task orchestration page, view the task parameter settings.

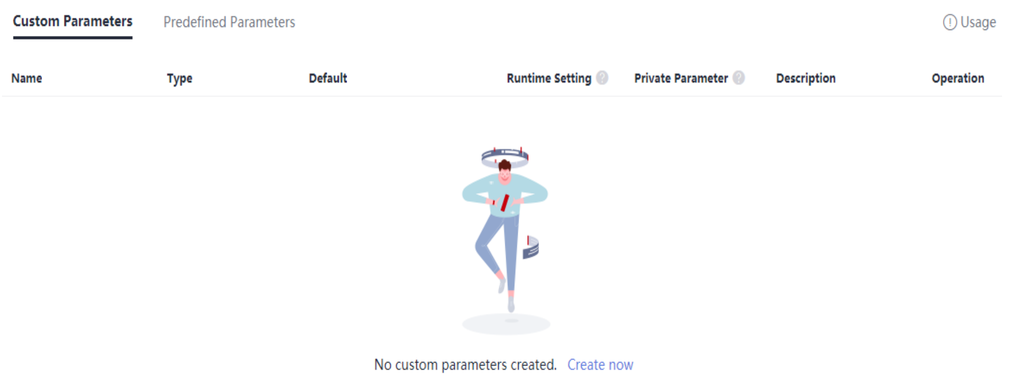
   

   *releaseversion

   ${releaseversion}

2. Access and view **Parameter Configuration** page.

   The **releaseversion** parameter referenced by the task parameter is not configured on the pipeline parameter list. As a result, this exception is reported.

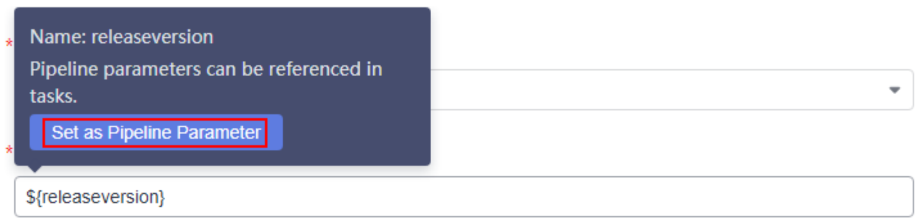


No custom parameters created.

Solution

Use either of the following methods to add a referenced parameter (for example, **releaseversion**) to the pipeline parameter list:

- On the **Parameter Configuration** page of the pipeline, click **Create now** to add a parameter.
- Click the name of the failed task. On the task orchestration page, hover over **releaseversion**, and click **Set as Pipeline Parameter** to add a parameter.



Name: releaseversion

Pipeline parameters can be referenced in tasks.

2.2 Test Suite Was Running

Symptom

A pipeline failed to run, and the message ET.00084217 was displayed, indicating that the test suite was running.

Root Cause

This exception occurs because the task is running when the pipeline triggers the execution of the task.

Solution

- Run the pipeline after the task is complete.
- Click the task name. On the displayed page, stop the task and run the pipeline again.

2.3 Failed to Obtain Tenant Information During Code Check Task Execution

Symptom

A pipeline failed to run, and the message DEV-31-50013 was displayed, indicating that tenant information cannot be obtained during code check task execution.

Root Cause

The data of the code check task associated with the project is missing.

Solution

Create a code check task and associate it with the pipeline.

Step 1 Access CodeArts Check.

Step 2 Click **Create Task** to create a code check task for the pipeline.

Step 3 Return to the failed pipeline and go to the pipeline orchestration page.

Step 4 Delete the failed code check task, associate the code check task created in **Step 2** with the pipeline, and save the changes.

Step 5 Run the pipeline again.

----End

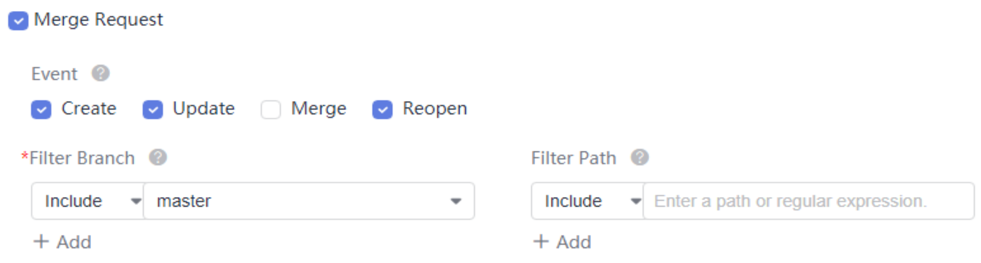
2.4 The Pipeline That Does Not Listen to the Push Event Is Triggered When the Code Is Committed

Symptom

When you modify the source branch code of an unclosed merge request and commit code, the pipeline that does not listen to the push event is triggered.

Root Cause

1. You have configured a merge request for a repository and listened to update events.



☑ Merge Request

Event

☑ Create ☑ Update ☐ Merge ☑ Reopen

*Filter Branch: Include master

Filter Path: Include Enter a path or regular expression.

2. You have created a merge request and the request is not closed.
3. When you commit code to the source branch of the merge request, an update event is generated. If the target branch is within the listening range of the merge request, the pipeline is triggered.

Conclusion

When you modify the source branch code of an unclosed merge request and commit code, an update event is generated. If the target branch of the merge request is within the listening range, the corresponding pipeline is triggered.

2.5 Scheduled Task of a Pipeline Was Not Triggered

Symptom

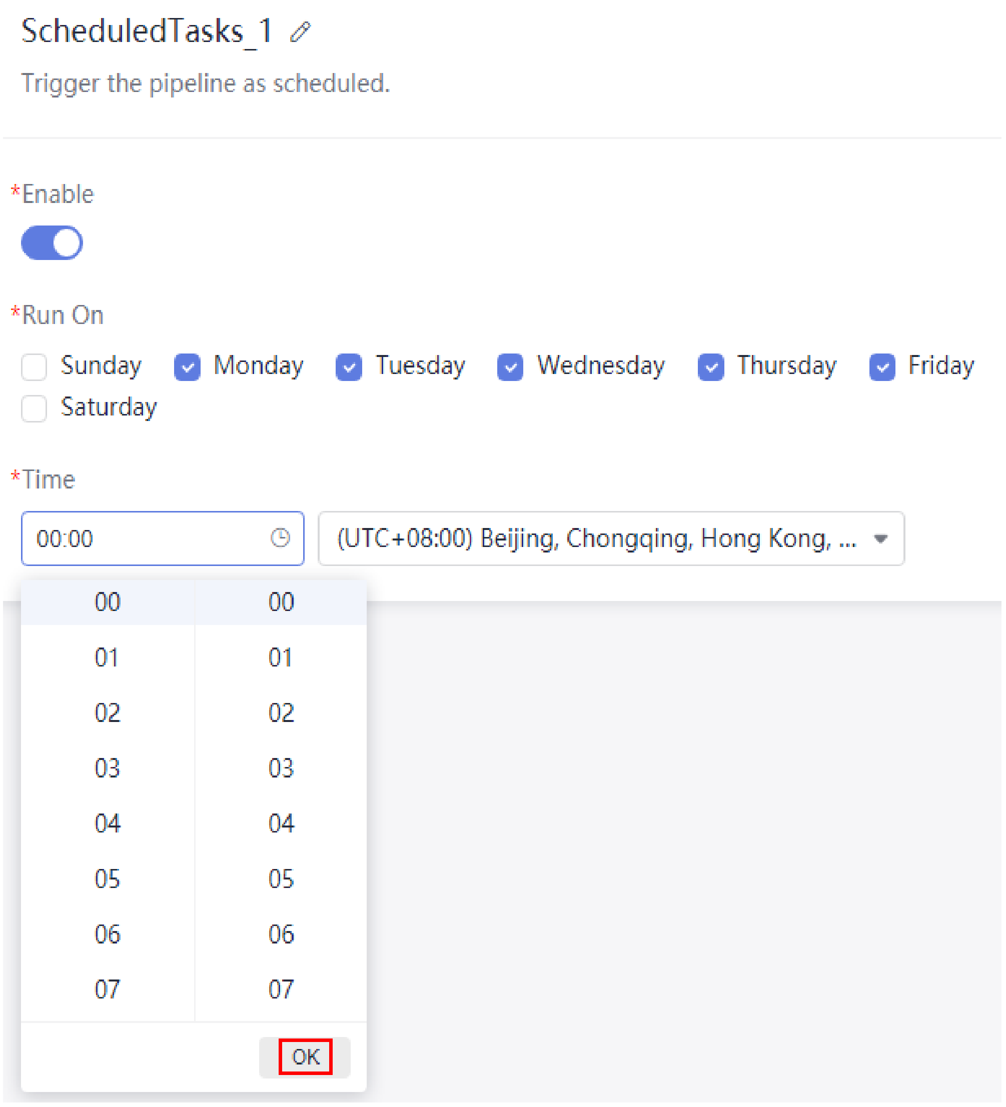
The scheduled task set for a pipeline was not triggered.

Root Cause

The scheduled task setting is not saved.

Solution

1. Find the target pipeline, go to the **Execution Plan** page, and check whether the scheduled task setting is correct.



ScheduledTasks_1

Trigger the pipeline as scheduled.

*Enable

*Run On

Sunday [✓ Monday] [✓ Tuesday] [✓ Wednesday] [✓ Thursday] [✓ Friday] Saturday

*Time

00:00 (UTC+08:00) Beijing, Chongqing, Hong Kong, ...

NOTE

After you change the execution time, click **OK** for the change to take effect.

2. Reset the scheduled task and save the pipeline.

2.6 Run Button Was Unavailable

Symptom

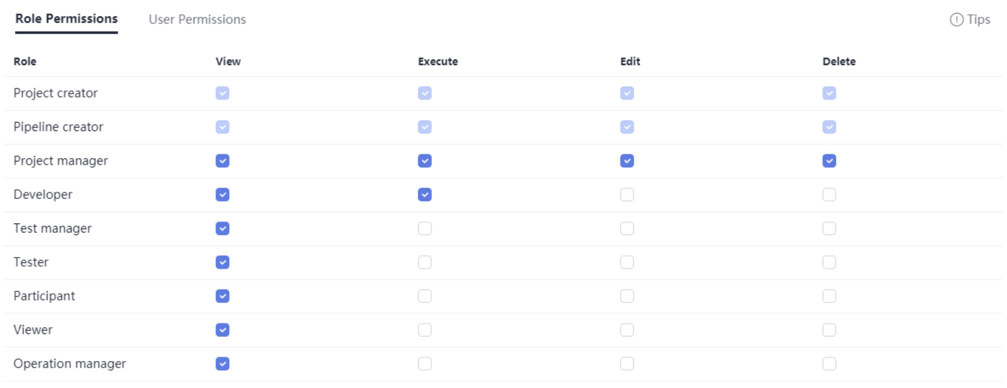
- The **Run** button next to a pipeline was unavailable.
- The **Run** button is not displayed in the upper right corner of the **Pipeline Details** tab page.

Root Cause

You do not have the permissions required to execute the pipeline.

Solution

1. Log in to CodeArts Pipeline using an account that can modify the pipeline permissions.
2. Go to the **Permissions** page of the target pipeline, and check the user permissions. By default, the project creator and pipeline creator have full permissions for the pipeline, and their permissions cannot be modified.



Role	View	Execute	Edit	Delete
Project creator	✓	✓	✓	✓
Pipeline creator	✓	✓	✓	✓
Project manager	✓	✓	✓	✓
Developer	✓	✓	☐	☐
Test manager	✓	☐	☐	☐
Tester	✓	☐	☐	☐
Participant	✓	☐	☐	☐
Viewer	✓	☐	☐	☐
Operation manager	✓	☐	☐	☐

NOTE

Pipeline permissions include role and user permissions. By default, role permissions are synchronized to users with the corresponding role, and user permissions take precedence over role permissions.

- If the role to which the user belongs does not have the run permission on the **Role Permissions** page, assign the run permission to the role. As a result, the user of the role has the run permission on the pipeline.
- If the role to which the user belongs has the run permission on the **Role Permissions** page, add the run permission to the user on the **User Permissions** page.