

CodeArts

Best Practices

Issue 01
Date 2026-01-12



Copyright © Huawei Cloud Computing Technologies Co., Ltd. 2026. 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 Cloud Computing Technologies Co., Ltd.

Trademarks and Permissions



HUAWEI and other Huawei trademarks are the property 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 Cloud 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.

Huawei Cloud Computing Technologies Co., Ltd.

Address: Huawei Cloud Data Center Jiaoxinggong Road
Qianzhong Avenue
Gui'an New District
Gui Zhou 550029
People's Republic of China

Website: <https://www.huaweicloud.com/intl/en-us/>

Contents

1 Developing an E-mail Project with CodeArts.....	1
1.1 Overview.....	1
1.2 Resource Planning.....	5
1.3 Process.....	5
1.4 Implementation Procedure.....	7
1.4.1 Preparation.....	7
1.4.2 Step 1: Managing Project Plans.....	9
1.4.3 Step 2: Managing Project Configurations.....	12
1.4.4 Step 3: Writing Code.....	14
1.4.5 Step 4: Checking Code.....	17
1.4.6 Step 5: Building an Application.....	18
1.4.7 Step 6: Deploying an Application (CCE).....	23
1.4.8 Step 6: Deploying an Application (ECS).....	26
1.4.9 Step 7: Managing Project Tests.....	31
1.4.10 Step 8: Configuring a Pipeline for Continuous Delivery.....	34
1.4.11 Releasing Resources.....	37

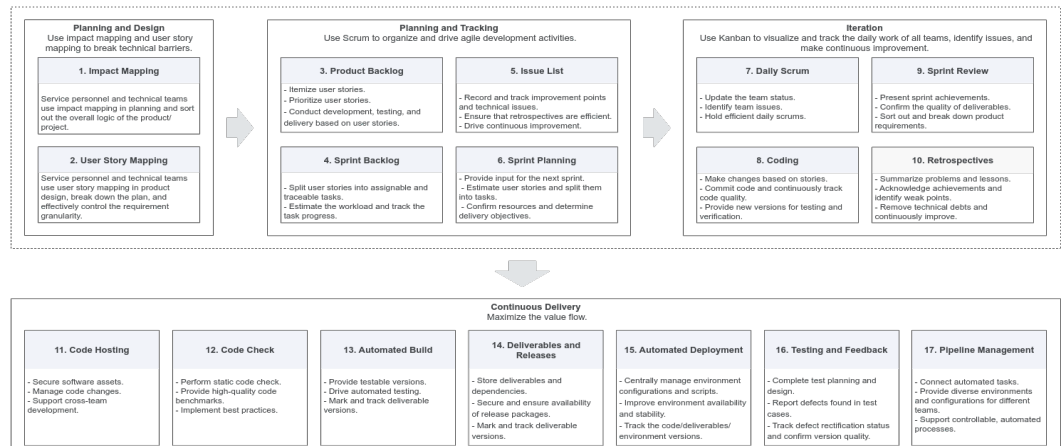
1 Developing an E-mall Project with CodeArts

1.1 Overview

Background

CodeArts proposes a feasible agile development methodology — HE2E DevOps implementation framework by using industry-leading practices.

Figure 1-1 HE2E DevOps implementation framework



- **Planning and Design**

The first two steps in the diagram depict the product planning process where service personnel (or customers) collaborate with technical personnel to establish the product's overall logic, plan and design the product, and break down requirements at a specific granularity.

- Software development solves issues and delivers value, not simply provides functions. Impact mapping identifies user requirements and root causes.

- User stories convey objectives and requirements in specific scenarios, facilitating communication between customers, service personnel, and developers. If you only view separate requirement items, you cannot consider them from the entire solution's perspective. User stories focus on scenarios, sort out and display stages and activities in a tree structure. In this way, you will view both requirement items and overall requirement scenarios.
- Planning, Tracking, and Iteration
Steps 3 to 10 are the main management practices in the Scrum framework.
 - Scrum defines a relatively complete framework for agile process management. CodeArts well integrates the Scrum framework with daily activities of development teams. Main process deliverables include the product backlog, sprint backlog, potential deliverable product increments, and issue list. Core team activities include sprint planning meetings, daily Scrums, sprint reviews, sprint retrospective meetings, and daily updates.
 - In addition, the combination of Kanban with the Scrum framework enables teams to learn from Kanban Lean, visualize value streams, identify and resolve blockage and bottlenecks, accelerate delivery of value streams, accelerate the feedback loop, and make continuous improvement.
- Continuous Delivery
Starting from step 11 is the engineering practice, that is, the CI/CD process.
 - Continuous delivery is based on code configuration management. It not only covers traditional security control of code assets, concurrent development, and version and baseline management, but also reflects team collaboration and communication.
 - The pipeline connects code check (or static scanning), automated build, automated testing in all stages, and automated deployment.
 - Continuous delivery also covers continuous artifact management and environment management at different levels, including development, test, quasi-production, and production environments.
 - The continuous delivery pipeline manages stages, environments, activities, entry and pass quality gates, and input and output artifacts in each stage.

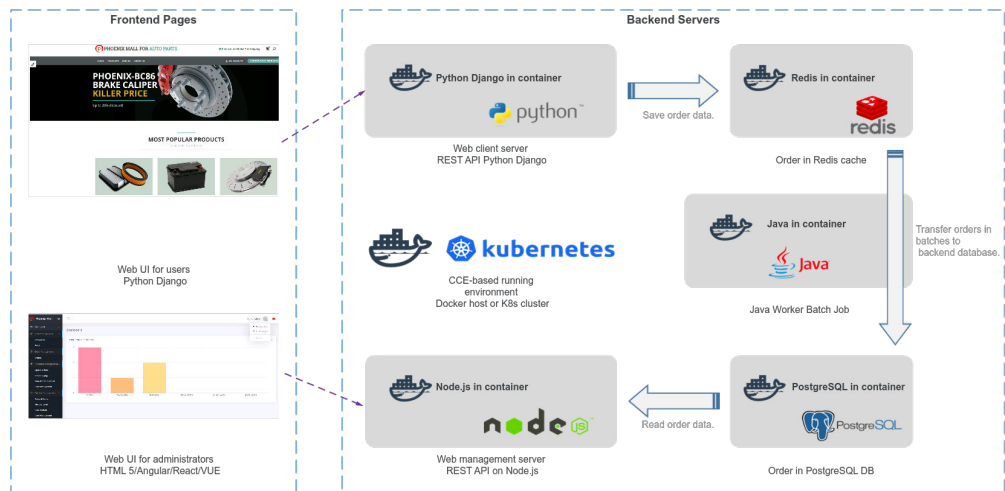
Application Scenarios

We will use the sample code for an auto part e-mall named "Phoenix Mall" and the "DevOps Full-Process" sample project to describe how to use CodeArts to implement the HE2E DevOps framework. This solution is applicable to Scrum R&D projects.

Solution Architecture

- Phoenix Mall Architecture
Figure 1-2 shows the architecture of the Phoenix Mall sample program.

Figure 1-2 Technical architecture of Phoenix Mall



The sample program consists of five microservice components that can be independently developed, tested, and deployed, as shown in [Table 1-1](#).

Table 1-1 Microservice components of Phoenix Mall

Microservice Component	Description
Web client server (corresponding to the Vote function in the sample code)	<ul style="list-style-type: none"> • Service logic: Users can use a browser to access the web UI of this service. When a user clicks Like on a specific offering, the service saves the record of the selected offering in the Redis cache. • Technology stack: Python and Flask frameworks • Application server: Gunicorn
Web management server (corresponding to the Result function in the sample code)	<ul style="list-style-type: none"> • Service logic: Users can use a browser to access the web UI of this service. The statistics about Like clicked by users on the UI are dynamically displayed. The data is obtained from the PostgreSQL database. • Technology stack: Node.js and Express frameworks • Application server: server.js

Microservice Component	Description
Background order batch processing program (corresponding to the Worker function in the sample code)	<ul style="list-style-type: none"> Service logic: This service is a background process. It monitors item records in the Redis cache, obtains new records, and saves them in the PostgreSQL database so that the management UI can extract data for statistics display. Technology stack: .NET Core or Java (This service provides two technology stacks to implement the same function. You can modify the configuration and select one of the technology stacks as the runtime process.)
Order cache	<ul style="list-style-type: none"> Service logic: Persists data for the client UI. Technology stack: Redis
Order database	<ul style="list-style-type: none"> Service logic: Serves as the data source of the management UI. Technology stack: PostgreSQL

- Composition of the "DevOps Full-Process" Sample Project

This project uses the Scrum template and presets some service resources. The following table lists the products and services involved in this project.

Table 1-2 List of involved products/services

Service	Description
CodeArts Req	Presets three planned and completed sprints, project module settings, and several statistical reports.
CodeArts Repo	Presets the code repository phoenix-sample for storing sample code.
CodeArts Check	Presets four tasks. For details, see Step 4: Checking Code .
CodeArts Build	Presets five tasks. For details, see Step 5: Building an Application .
CodeArts Artifact	Stores software packages generated by build tasks.
CodeArts Deploy	Presets three applications. For details, see Step 6: Deploying an Application (CCE) .
CodeArts TestPlan	Presets more than 10 test cases in the test case library.
CodeArts Pipeline	Presets five pipelines. For details, see Step 8: Configuring a Pipeline for Continuous Delivery .

Service		Description
Other components and services	Identity and Access Management (IAM)	Manages accounts.
	SoftWare Repository for Container (SWR)	Stores Docker images generated by build tasks.
	Cloud Container Engine (CCE)	Deploys software packages in a different way from ECS-based deployment.
	Elastic Cloud Server (ECS)	Deploys software packages in a different way from CCE-based deployment.

Advantages

- We provide a one-stop cloud platform to manage the entire software development process, to address R&D pain points such as frequent requirement changes, complex development and test environments, difficult multi-version maintenance, and failure to effectively monitor the progress and quality.
- This service provides visualized and customizable pipeline services for continuous delivery, doubling the software rollout speed.

1.2 Resource Planning

The following table lists the resources required for completing this practice. The practice may take 2 to 3 hours.

Table 1-3 Resource planning

Service Name	Quantity
CodeArts	-
Cloud Container Engine (CCE)	1
Elastic Cloud Server (ECS)	1

1.3 Process

This document describes the operation process of the HE2E DevOps practice.

Figure 1-3 HE2E DevOps practice process

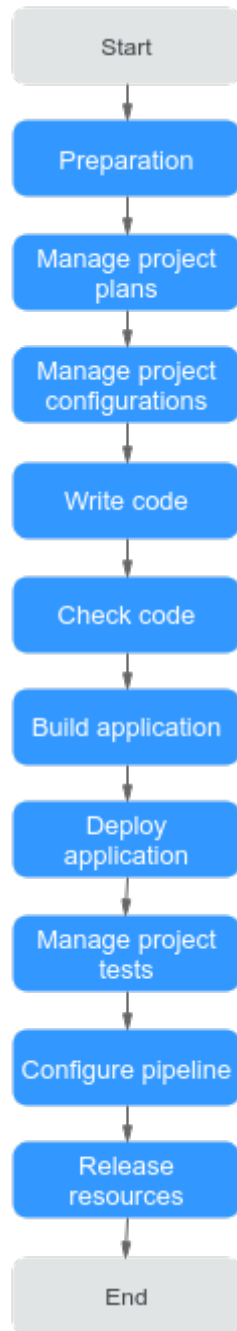


Table 1-4 HE2E DevOps practice process description

Step	Description
Preparation	Complete preparations before the practice, including creating a project and adding project members.
Manage project plans	Complete the overall planning of the project, including project and sprint requirements.

Step	Description
Manage project configurations	Customize the work item change notification and status transition modes based on project requirements.
Write code	Use branches to compile code, including creating branches, committing code, and merging branches.
Check code	Perform static scanning on the code and optimize the code based on fix suggestions to improve the code quality.
Build application	Build environment images and compile and package code into software packages.
Deploy application	Install and run the built environment image and software package in the environment. This document describes deployment in two environments: CCE and ECS.
Manage project tests	Create test plans for sprints, design test cases, and execute test cases as planned.
Configure pipeline	Orchestrate code check, build, and deployment tasks into a pipeline. When code is updated, the pipeline is automatically triggered for continuous delivery.
Release resources	After completing the practice, release the resources on CodeArts, CCE, and other services.

1.4 Implementation Procedure

1.4.1 Preparation

Before performing a specific task, you need to complete the following preparations:

Enabling Pay-per-Use Billing

- Step 1** Log in to the console, choose **Overview**, and click **Pay-per-Use Package**.
- Step 2** View the service specifications and click **Next**.
- Step 3** View and confirm the package details, and click **Submit**. In the dialog box that is displayed, click **Yes**.
- Step 4** Check the enabling status.

Return to the **Overview** page, and check that the pay-per-use package is enabled on the **My Services** tab.

----End

Creating a Project

Before starting the practice, the product owner Sarah creates a project.

- Step 1** Log in to the CodeArts homepage.
- Step 2** Click **Create Project**, and select **DevOps Full-Process**.
- Step 3** Enter the project name **Phoenix Mall** and click **OK**.

----End

Adding Project Members

Sarah creates accounts for team members and adds them to the project.

This sample project involves four project roles. To facilitate introduction, each role in this document corresponds to a person, as shown in [Table 1-5](#).

Table 1-5 Project role list

Project Member	Project Role	Responsibility
Sarah	Product owner (project creator)	Responsible for the overall product planning and product team setup.
Maggie	Project manager	Manages project delivery plans.
Chris	Developer	Develops, compiles, deploys, and verifies project code.
Billy	Tester	Writes and executes test cases.

- Step 1** Go to the **Phoenix Mall** project, and choose **Settings > General > Service Permissions > Member**.
- Step 2** On the **Member** tab, choose **Add Members > Users from My Enterprise**.
- Step 3** In the dialog box that is displayed, click **Create User**. The **Users** page is displayed.
- Step 4** Click **Create User**, and create three users named **Maggie**, **Chris**, and **Billy** in sequence.
- Step 5** Return to the CodeArts page, refresh the browser, click **Add Members** above the member list, and choose **Users from My Enterprises**. Select **Maggie**, **Chris**, and **Billy**, and click **Next**.
- Step 6** Click the **Role** drop-down list in each row, select **Project manager** for Maggie, **Developer** for Chris, and **Tester** for Billy, and then click **Save**.

----End

1.4.2 Step 1: Managing Project Plans

CodeArts Req provides simple and efficient team collaboration services, including multi-project management, agile iteration, and task management.

This sample project uses the Scrum mode for iterative development. Each sprint lasts for two weeks. The Phoenix Mall has been developed in the first three sprints, and sprint 4 is being planned.

According to the project plan, time-limited discount and group buying activity management functions need to be implemented in sprint 4.

Due to business and market changes, store network query is added as an urgent requirement. Therefore, this function will be developed in sprint 4.

This section describes how Sarah (product owner) and Maggie (project manager) manage requirements and sprints, and track the project progress.

Managing Requirements

Project requirements are managed using mind maps, which present work items in the hierarchical structure of "Epic > Feature > Story > Task". These work item types are described in [Table 1-6](#).

Table 1-6 Work item types

Type	Description
Epic	An important strategic measure of a company. For example, Phoenix Mall in this sample project is a key strategic measure related to the survival of the company.
Feature	A function valuable to customers. You can use features to fulfil customer requirements. For example, the store network query function in Phoenix Mall is continuously delivered in multiple sprints.
Story	Generated by breaking down a function based on user scenarios. A story can be completed in one sprint.
Task	Generated by breaking down a user story. Preparing environments and test cases can be tasks for a story.

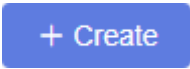

Step 1 Create a work item for the new requirement.

The store network query function is a new requirement. Therefore, Sarah needs to add it to the requirement planning view.

1. Go to the **Phoenix Mall** project and choose **Work** from the navigation pane.
2. Click the **Plans** tab and click **Phoenix-MindMap**.

 NOTE

If the **Plans** tab page is empty, create a mind map.

1. Click  and choose **Mind Maps** from the drop-down list.
In the dialog box that is displayed, enter the name **Requirement_Planning**, and click **OK**. The mind map details page is displayed.
2. Click **Add to Existing Epic**. In the displayed dialog box, select **Phoenix** and click **Add**.
3. Create a feature named **Store Network**.
 - a. Click  under Epic **Phoenix**.
 - b. Enter the name **Store Network** and press **Enter** to save the settings.
4. Use the same method to add story **User can query network of all stores** to feature **Store Network**.

Step 2 Edit the story.

1. Click story **User can query network of all stores** and edit it by referring to the following table.

Table 1-7 Story configurations

Configuration Item	Suggestion
Description	Enter As a user, I want to query all stores so that I can select a proper store to obtain the service.
Priority	Select High .
Severity	Select Critical .

2. To facilitate understanding, prepare a local file named **Store List** that includes the following content.

Table 1-8 Store list

Branch Name	Branch Address
Branch A	123 meters to the departure floor, Terminal 1, Airport E
Branch B	No. 456, Street G, Area F
Branch C	No. 789, Street J, Area H
Branch D	West side of Building K, Avenue L, Area K

3. Return to the story editing page, click **Select or Drag & Drop File**, choose **Upload** from the drop-down list, and upload the list file to the work item as an attachment.

4. Click **Save**. The story details are edited.

----End

Managing Sprints

Before a sprint starts, the project manager Maggie organizes a plan meeting to add the new story to the sprint, break down the story into tasks, and assign the tasks to developers.

This section describes how to plan sprint 4.

Step 1 Create a sprint.


1. Go to the **Phoenix Mall** project, choose **Work** from the navigation pane, and click the **Sprints** tab.
2. Click  next to **Sprint** in the upper left corner of the page. In the dialog box that is displayed, configure the sprint by following [Table 1-9](#). Then click **OK**.

Table 1-9 Sprint information configurations

Configuration Item	Suggestion
Sprint Name	Enter Sprint4 .
Planned Duration	Set the duration to 2 weeks.

Step 2 Plan the sprint.

1. From the left navigation pane, choose **Unplanned Work Items**.
2. Select the following three stories in the list as planned:
 - User can query network of all stores
 - Admin adds group buying activities
 - Admin adds time-limited discounts
3. Click **Edit** at the bottom of the page.
4. Click **Add Field**.
5. Choose **Sprint** from the **Field Name** drop-down list box, select **Sprint4** from the **Field Value** drop-down list box, and click **Save**.

Step 3 Assign the stories.

1. Choose **Sprint4** from the left navigation pane.
2. Select all stories and set the **Assign To** field to **Chris** by following the instructions in [Planning a Sprint](#).


Step 4 Break down the stories.

1. Find the story **User can query network of all stores**. Click the story name.
2. In the displayed window, click the **Child Work Items** tab.
3. Click **Fast Create Child**. Enter the title **Frontend display - Add store network menu**, assign it to **Chris**, and click **OK**.

4. Use the same method to add task **Background management - Add store network management and maintenance module**.

----End

Tracking Project Status

- Tracking task progress via daily stand-ups
After the sprint starts, the project team communicates the current progress of each work item through daily stand-up meetings and updates the status. You can view the status of work items in the sprint in the card mode.
Go to the **Sprints** tab page and click  to switch to the card mode. This page displays work item cards in each status. You can drag a work item card to update its status.
- Accepting iteration results via a review meeting
Before the expected end time of the sprint, the project team holds a review meeting to present work achievements of the current sprint.
The **Sprints** tab page provides sprint statistics and charts. The team can easily collect statistics on the progress of the current sprint, including the requirement completion status, sprint burndown chart, and workload.
Go to the **Sprints** tab page and click **Statistics** to display the progress view.

1.4.3 Step 2: Managing Project Configurations

Managing Project Notifications

The project manager Maggie wants team members to be notified of new tasks (work items) so that the members can handle the tasks (work items) in time.


Step 1 Go to the **Phoenix Mall** project, and choose **Settings > Work > Notifications** from the navigation pane.

Step 2 View default settings of the sample project displayed on the page.

We will keep default settings unchanged because they can meet requirements. If necessary, modify the settings. Then they will be automatically saved.

Step 3 Verify the configuration result.

When the project manager is done with breaking down the story, the developer Chris will receive the following two types of notifications.

- Direct messages: After Chris logs in to the homepage, they will see a number in the upper right corner. They can click  to view the notification.
- Email: The project member also receives an email if an email address has been configured for the corresponding user and the **Email Notifications** option has been enabled on the **This Account Settings** page.

 **NOTE**

All members can determine whether to receive email notifications. To enable email notification, perform the following steps:

1. Click the username in the upper right corner of the page and choose **This Account Settings** from the drop-down list.
2. Find **Email Notifications** on the **Notifications** page and click **Enable**. You can click **Edit Settings** to change the email address.

----End

Customizing a Project Workflow

In the sprint review meeting, the team demonstrates the product to the product owner and presents the test report. The product owner confirms whether the story is complete. However, the current story status does not show that the test is complete. Therefore, the tester suggests adding a status named **Accepting**.

The project manager Maggie performs the following operations to add a status to a story.

- Step 1** Go to the **Phoenix Mall** project, and choose **Settings > Work** from the navigation pane.
- Step 2** Choose **Common Statuses** from the middle navigation pane. View the default work item statuses of the sample project.
- Step 3** Click **Add Status**. In the dialog box that is displayed, edit the status information by referring to [Table 1-10](#), and click **Add**.

Table 1-10 Status configurations

Configuration Item	Suggestion
Status	Enter Accepting .
Status Category	Select Doing .

- Step 4** Choose **Stories > Statuses and Transitions** from the navigation pane. View the default story statuses of the sample project.
- Step 5** Click **Add Existing Status**. In the displayed dialog box, select **Accepting** and click **OK**.
- Step 6** Drag and place **Accepting** below **Testing**.
- Step 7** Verify the configuration result.
 1. Choose **Work** from the navigation pane, and click the **Work Items** tab.
 2. Click any story name in the list to view story details.
 3. Click the value in the **Status** column. In the drop-down list, you can see that the **Accepting** option is displayed.

----End

1.4.4 Step 3: Writing Code

CodeArts Repo provides Git-based online code management service, including code cloning/commit and branch management.

This section uses store network query as an example to describe how to manage and develop source code. Store network query is a high-priority story.

In this sample project, code is developed in a branch. Developer Chris creates a branch in a code repository, develops code, and submits a merge request. Project manager Maggie approves the request and then merges it to the master branch.

Managing Code with a Branch

A branch is a tool for parallel feature development. Branches enable you to diverge from the main line of development and continue to do work without messing with that main line.

When a code repository is created, there is a default branch named **master**, that is, the main line. To ensure stable running of the Phoenix Mall, a stable and continuously available **master** is required. The project manager suggests "function branches + merge requests". Each function branch must be reviewed by other members in the team before merge.

Step 1 Set **master** as a protected branch (this operation is performed by project manager Maggie in this document).

1. Go to the **Phoenix Mall** project, choose **Code > Repo** from the navigation pane, and locate the code repository **phoenix-sample**.
2. Go to the code repository by clicking its name and click the **Settings** tab. Choose **Policy Settings > Protected Branches** from the navigation pane.
3. Click **Create Protected Branch**. Complete the settings by referring to the following table and click **OK**.

NOTE


If the protected branch **master** already exists on the page, click  and modify the branch configuration as required.

Table 1-11 Creating a protected branch

Configuration Item	Suggestion
Branch	Select master .
Push	Retain the default value.
Merge	Retain the default value.
Members	Retain the default value.

Step 2 Create a function branch (this operation is performed by developer Chris in this document).

1. Go to the **Phoenix Mall** project. Choose **Code > Repo**. Find the **phoenix-sample** in the repository.

2. Click the repository name to access the code repository. On the **Code** tab, click **Branches**.
3. Click **Create Branch**, set the branch information by referring to [Table 1-12](#), and click **OK**.

Table 1-12 Creating a branch

Configuration Item	Suggestion
Based On	Select master .
Branch Name	Enter Feature-Store .
Work Items to Associate	Select User can query network of all stores .

----End

Modifying and Committing Code

The store network query function is divided into frontend display and backend management tasks during [sprint planning](#). This section uses task **Frontend display - Add store network menu** as an example to describe how to modify and commit code.

Step 1 Choose **Work** from the navigation pane, and click the **Sprints** tab.


In sprint 4, find the task **Frontend display - Add store network menu**, and change the task status to **Developing**.

Step 2 Choose **Code > Repo**, and find the **phoenix-sample** repository.

Step 3 Go to the code repository by clicking its name and click the **Code** tab.

Step 4 Click **master** above the file list and select the branch **Feature-Store** from the drop-down list.

Step 5 Find **vote/templates/store-network.html** in the file list and open it.

Step 6 Click , add the store address specified in the story, enter the commit message **Store list added** in the text box at the bottom of the page, and click **OK**.

```
<ul>
  <li>Branch A: 123 meters to the departure floor, Terminal 1, Airport E</li>
  <li>Branch B: No. 456, Street G, Area F</li>
  <li>Branch C: No. 789, Street J, Area H</li>
  <li>Branch D: West side of Building K, Avenue L, Area K</li>
</ul>
```

Step 7 Open and edit the **/vote/templates/index.html** file in the same way.

Add the menu **Store Network** in line 179, enter the commit message **fix #xxxxxx Frontend display - Add store network menu**, and click **OK**.

In the preceding message, **#xxxxxx** indicates the ID of task **Frontend display - Add store network menu**, which is obtained from the work item list.

```
<li class="nav-item"> <a href="store-network" class="nav-link">Store network</a> </li>
```

Step 8 Choose **Work** from the navigation pane, and click the **Sprints** tab.

In sprint 4, find the task **Frontend display - Add store network menu**.

- Click the task name. The task status automatically changes to **Resolved** on the details page.
- Click the **Associations** tab. Under **Code Commits**, a record is displayed with the commit message configured in the [previous step](#).

----End



Reviewing Code and Merging Branches

Step 1 The developer initiates a merge request.

After coding, developer Chris initiates a merge request to merge their function branch into the **master**.

1. Go to the code repository, click the **Merge Requests** tab, and click **Create MR**.
2. Set the source branch to **Feature-Store** and the target branch to **master**, and click **Next**.
3. Edit the merge request details by referring to [Table 1-13](#).

Table 1-13 Merge request configurations

Configuration Item	Suggestion
Title	Enter Add store list .
Mergers	Click  . In the dialog box that is displayed, select Maggie and click OK .
Approvers	Click  . In the dialog box that is displayed, select Maggie and click OK .

4. Click **Create Merge Request** to complete the creation.

Step 2 The project manager reviews the code and merges the commit request.

In this document, project manager Maggie is both the approver and merger. Maggie approves the request content and merges the request.

1. Go to the code repository, click the **Merge Requests** tab, and find the merge request created by developer Chris.
2. Click the request to view details.
3. Leave comments on the page. Click **Approve** in **Approval Gate** to complete the approval.
4. Click **Merge** to merge the branch into **master**.

 NOTE

If you have selected **Delete source branch after merged** when creating a merge request, the branch **Feature-Store** will be deleted after the branch merge is complete.

----End

1.4.5 Step 4: Checking Code

CodeArts Check provides cloud-based code quality management, static code check (quality and style), security check, issue fixing suggestions, and trend analysis.

As Phoenix Mall becomes increasingly larger, more issues have occurred, involving high fixing costs. However, no unified coding standards are available. The project manager suggests some basic standards, continuous static code scanning, and issue fixing within sprints.

This section describes how developer Chris scans static code and identify and fix issues for different technology stacks.

Introduction to Preset Tasks

The sample project has four preset code check tasks.

Table 1-14 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.

Step 2 Execute the task.

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

1.4.6 Step 5: Building an Application

CodeArts Build provides a hybrid language build platform with simple configurations, supports one-click task creation, configuration, and execution, and automates activities such as code obtaining, building, and packaging.

During project deployment, failures often occur due to environment inconsistency. For example, after the JDK in the R&D debugging environment is upgraded, the JDK is not marked in the environment list. As a result, the production environment is not upgraded, causing failures. To avoid problems caused by environment inconsistency, we package microservice applications and environments into images in this example to ensure that the environments (development and commissioning, test, QA, and production) are consistent.

This section describes how Chris builds and archives images and software packages.

Introduction to Preset Tasks

There are five build tasks preset in the sample project.

Table 1-15 Preset tasks

Preset Task	Description
phoenix-sample-ci	Basic build task
phoenix-sample-ci-test	Task for building images in the test environment
phoenix-sample-ci-worker	Task for creating a Worker function image
phoenix-sample-ci-result	Task for creating a Result function image
phoenix-sample-ci-vote	Task for creating a Vote function image

This section uses the **phoenix-sample-ci** task as an example. The following table shows the steps involved in this task.

Table 1-16 Build actions

Build Action	Description
Build Vote Image and Push It to SWR	Create a Vote image using the vote/Dockerfile file in the code repository, and push the image to SWR.
Build Result Image and Push It to SWR	Create a Result image using the result/Dockerfile file in the code repository, and push the image to SWR.
Install Worker Dependency Using Maven	Use Maven to install the dependencies required by the Worker function.
Build Worker Image and Push It to SWR	Create a Worker image using the worker/Dockerfile file in the code repository, and push the image to SWR.
Create Postgres and Redis Dockerfiles	Use shell commands to generate a Dockerfile for creating Postgres (database) and Redis (cache) images.
Build Postgres Image and Push It to SWR	Create a Postgres image using the Dockerfile generated in action Create Postgres and Redis Dockerfiles , and push the image to SWR.
Build Redis Image and Push It to SWR	Create a Redis image using the Dockerfile generated in action Create Postgres and Redis Dockerfiles , and push the image to SWR.

Build Action	Description
Replace Image Version of Docker-Compose Deployment File	To ensure that the correct image can be pulled when the image is deployed on ECS, run the shell commands to perform the following operations: <ol style="list-style-type: none">1. Run sed commands to replace the parameters in the docker-compose-standalone.yml file with the dockerServer, dockerOrg, and BUILDNUMBER parameters of the build task in sequence.2. Run the tar command to compress the docker-compose-standalone.yml file into the docker-stack.tar.gz file. Compress the files required for deployment so that the files can be uploaded and archived in subsequent steps.
Replace Image Version of Kubernetes Deployment File	To ensure that the correct image can be pulled when the image is deployed on CCE, run the shell commands to perform the following operations: <ol style="list-style-type: none">1. Run the sed commands to replace the docker-server and docker-org parameters in all the files whose names end with deployment in the kompose directory of the code repository with the dockerServer and dockerOrg parameters of the build task.2. Run the sed commands to replace image-version in the result-deployment.yaml, vote-deployment.yaml, and worker-deployment.yaml files of the code repository with BUILDNUMBER.
Upload Kubernetes Deployment File to Release Repo	Upload all .yaml files modified in action Replace Image Version of Kubernetes Deployment File to the release repo for archiving.
Upload Docker-Compose Deployment File to Release Repo	Upload the docker-stack.tar.gz package generated in action Replace Image Version of Docker-Compose Deployment File to the release repo for archiving.

Configuring SWR

This example uses SWR to store environment images. Before executing tasks, obtain the SWR login command and create an organization.

Step 1 Log in to the SWR console.

Step 2 Click **Generate Login Command**. The login command is displayed in a dialog box.

In this command,

- The character string following **-u** is the username.
- The character string following **-p** is the password.

- The last character string is the address of the SWR server, which is the value of **dockerServer** for configuring and executing tasks later.

Step 3 Click **Create Organization**. In the displayed dialog box, enter **phoenix** as the organization name and click **OK**. (If a message is displayed indicating that the organization already exists, enter another name.)

The organization name is the value of **dockerOrg**, which will be used for configuring and executing tasks later.

----End

Configuring and Executing a Task

Step 1 Configure a task.

1. Go to the **Phoenix Mall** project and choose **CICD > Build**. The built-in build tasks of the sample project are displayed on the page.
2. Find the **phoenix-sample-ci** task in the list. Click ******* and choose **Edit**.
3. Click the **Parameters** tab and set parameters by referring to [Table 1-17](#).



Table 1-17 Setting parameters

Name	Default Value
codeBranch	master
dockerOrg	phoenix (name of the organization created in SWR)
version	1.0.0
dockerServer	SWR server address obtained from SWR.

NOTE

Ensure that the values of **dockerOrg** and **dockerServer** are correct. Otherwise, the task will fail.

Step 2 Click **Save and Run**. In the dialog box that is displayed, click **OK** and start the build task.

If  is displayed, the task is successfully executed. Record the character string starting with **#** (for example,  **#20230401.1**).

If the build fails, rectify the fault based on the failed action and the error information in logs.

Step 3 Check the release file.

1. Choose **Artifact** from the navigation pane and click the **Release Repos** tab.
2. In the repository named after the project, find the **docker-stack** and **phoenix-sample-ci** folders.

- In the **docker-stack** folder, find the folder named after the character string recorded in **Step 2**. Then find the release file **docker-stack.tar.gz** in this folder.
 - In the **phoenix-sample-ci/1.0.0** folder, find the 10 archived **.yaml** files.
3. Go to SWR. In the navigation pane, choose **Organizations**, and click the organization with the same name as the value of the build task parameter **dockerOrg**.
Click the **Images** tab. Then find the five images (**redis**, **postgres**, **worker**, **result**, and **vote**) in the list.
4. Click the names of the five images in sequence to go to the image details page. View the image tag on the **Tags** tab page.
- The tag of **redis** is **alpine**.
 - The tag of **postgres** is **9.4**.
 - The tags of **worker**, **result**, and **vote** are the same as those recorded in **Step 2**.

----End

Configuring Auto Compilation upon Code Commits

After the following configuration, code changes will automatically trigger the application build task, achieving continuous integration.

Step 1 On the details page of task **phoenix-sample-ci**, click **Edit**.

Step 2 Click the **Schedule** tab.

Step 3 Enable **Run upon Code Commit**. When  is displayed, click **Save**.

The default value of **codeBranch** on the **Parameters** tab is **master**. Therefore, the build is automatically run when the code in **master** code is changed.

Step 4 Modify the project code and submit it to **master** to check whether the build task is automatically executed.

----End


Configuring Scheduled Execution

To prevent code bugs from entering the production environment and ensure that the application is always deployable, the team recommends continuous verification of the application.

You can perform the following operations to execute the build task at a scheduled time:

Step 1 On the details page of task **phoenix-sample-ci**, click **Edit**.

Step 2 Click the **Schedule** tab.

Step 3 Enable **Scheduled Execution** ( means enabled), select days and time, disable **Upon Code Change**, and click **Save**.

For this example, select **All**, and set the execution time to **12:00**. (This example uses the default time zone. You can change it to the one you use.)

Step 4 Check whether the build task is automatically executed at the configured time.

----End

1.4.7 Step 6: Deploying an Application (CCE)

CodeArts Deploy provides visualized and automated deployment. Various deployment actions are provided to help you formulate a standard deployment process, reducing deployment costs and improving release efficiency.

To deliver software more quickly and stably, the development team needs some self-service deployment service capabilities to reduce subsequent maintenance workload.

This section describes how Chris deploys a release package on CCE. For details about ECS-based deployment, see [Step 6: Deploying an Application \(ECS\)](#).

Preset Applications

There are three deployment applications preset in the sample project.

Table 1-18 Preset applications

Preset Application	Description
phoenix-cd-cce	For deployment to CCE
phoenix-sample-standalone	For deployment to ECS
phoenix-sample-predeploy	For installing dependency tools on ECS

This section uses the **phoenix-cd-cce** application as an example.

Buying and Configuring CCE

This section uses Cloud Container Engine (CCE) for deployment.

You can buy a standard or turbo cluster on the console.

For details about the mandatory configurations of clusters and nodes, see [Table 1-19](#) and [Table 1-20](#). You can select the configurations that are not listed in the table based on the site requirements.

Table 1-19 Buying a CCE cluster

Category	Configuration Item	Suggestion
Basic Settings	Type	Select CCE Standard Cluster .
	Billing Mode	Select Pay-per-use .
	Cluster Name	Enter a name.
	Cluster Version	Select a version as required. You are advised to select the latest version.
Network Settings	Network Model	Select Tunnel network .
	VPC	Select an existing VPC. If no proper VPC is available in the list, click Create VPC .
	Default Node Subnet	Select an existing subnet. If no proper subnet is available in the list, click Create Subnet .
	Container CIDR Block	Click Auto select .

Table 1-20 Configuring a node

Category	Configuration Item	Suggestion
Node Configuration	Billing Mode	Select Pay-per-use .
	Node Type	Select Elastic Cloud Server (VM) .
	Specifications	Select 2 vCPUs and 8 GiB memory or higher.
	OS	Click Public image and select an Euler image.
	Node Name	Enter a custom name.
	Login Mode	Select Password .
	Password	Enter a password.

Category	Configuration Item	Suggestion
Network Settings	Node IP	Select Automatic .
	EIP	Select Auto create .

Configuring and Executing an Application

Deploy the `.yaml` files generated in [Step 5: Building an Application](#) in the CCE cluster one by one.

Step 1 Configure the application.

1. Go to the **Phoenix Mall** project and choose **CICD > Deploy**. The built-in deployment applications of the sample project are displayed on the page.
2. Find application **phoenix-cd-cc**. Click ******* and choose **Edit**.
3. On the **Deployment Actions** tab page, complete the following configurations in each action.

Table 1-21 Configuring deployment actions

Configuration Item	Suggestion
Cluster Name	Use the cluster name set when buying a CCE cluster.
Namespace	In this document, select default .

4. Click the **Parameters** tab and set parameters.

Table 1-22 Parameter configuration


Parameter	Example Value
ci_task_name	Enter phoenix-sample-ci .
version	Use the value of version of the phoenix-sample-ci task.


5. Click **Save**.

Step 2 Go to the CCE console. Locate the target cluster and click its name to go to the **Overview** page.

Choose **Workloads** from the navigation pane, click the **Deployments** tab, and verify that no record exists in the list.

If there are records in the list, select all records, click **Delete**, select all resource release options, and click **Yes** to clear the records in the list.

Step 3 Return to the application list page, click  in the row of the **phoenix-cd-cc** application, and click **OK** in the dialog box that is displayed to start deployment.

If  is displayed on the page, the deployment is successful. If the deployment fails, rectify the fault based on the failed action and the error information in logs.

Step 4 Verify the deployment result.

1. Go to the CCE console.
2. Locate the target cluster and click its name to go to the **Overview** page.
3. Choose **Workloads** from the navigation pane, and click the **Deployments** tab. Five records are displayed on the page. All of them are in the **Running** state.
4. Click **vote** to go to the details page. On the **Access Mode** tab page, choose **More > Update**.

Set the parameters by referring to [Table 1-23](#), and click **OK**.

Table 1-23 Updating a service

Parameter	Example Value
Service Type	Select NodePort .
Service Affinity	Select Cluster-level .
Service Port	Enter 5000 .
Container Port	Enter 80 .
Node Port	Set the port number based on the site requirements. In this document, this parameter is set to Auto .

5. Return to the list. Record the port number with a dotted line below the text in the **Access Port:Container Port/Protocol** column.
6. Open a new browser page and enter **http://IP.port number** in the address box. The Phoenix Mall homepage is displayed.
IP indicates the elastic IP address bound to the node in [Buying and Configuring CCE](#), and *port number* indicates the value recorded in [Step 4.5](#).
7. Return to the **Deployments** page and update **result** (the service port is **5001**) by referring to [Step 4.4](#).

After the creation is successful, enter the node IP address and service port number in the address box of a new browser. The dashboard of Phoenix Mall is displayed.

----End

1.4.8 Step 6: Deploying an Application (ECS)

This section uses **phoenix-sample-standalone** as an example to describe how to deploy the release package to a host. For details about CCE-based deployment, see [Step 6: Deploying an Application \(CCE\)](#).

Purchasing and Configuring an ECS

This section uses Elastic Cloud Server (ECS). You can also use your own Linux host that runs Ubuntu 16.04 OS.

Step 1 "Customizing an ECS" chapter in the *Elastic Cloud Server User Guide*.

The following table lists mandatory configurations. You can also select other configurations as necessary.

Table 1-24 Configuring an ECS purchase

Category	Configuration Item	Suggestion
Configure Basic Settings	CPU Architecture	Select x86 .
	Specifications	Select General-purpose with 2 vCPUs and 8 GB memory or higher.
	Image	Choose Public image > Ubuntu > Ubuntu 16.04 server 64bit .
Configure Network	EIP	Select Auto assign .
	Billed By	Select Bandwidth .
Configure Advanced Settings	Login Mode	Select Password .
	Password	Enter a password.

Step 2 Configure security group rules.

Use ports 5000 and 5001 to verify the sample project. Therefore, add an inbound rule that allows access over ports 5000 and 5001.

The procedure is as follows:

1. Log in to the ECS list page, locate the ECS purchased in step **Step 1**, and click the ECS name.
2. Click the **Security Group** tab and add an inbound rule in which **Protocol** is set to **TCP** and **Port Range** is set to **5000-5001** by referring to "Configuring Security Group Rules" chapter in the *Elastic Cloud Server User Guide*.

----End

Adding a Target Host to the Project

Before deploying applications to ECSs, add the target hosts to the basic resources of the project.

Step 1 Go to the **Phoenix Mall** project and choose **Settings > General > Basic Resources** from the navigation pane.

Step 2 Click **Create Host Cluster**, configure the following information, and click **Save**.

Table 1-25 Creating a host cluster

Parameter	Suggestion
Cluster Name	Enter host-group .
OS	Select Linux .
Host Connection Mode	Select Direct Connection .
Execution Resource Pool	Select Official .

Step 3 Click **Add Host**, configure the following information, and click **OK**.

Table 1-26 Adding a host

Configuration Item	Suggestion
Add Hosts by	Select Adding IP .
Host Name	Keep this name same as the name of the purchased ECS.
IP	Enter the EIP generated when buying the ECS.
Authorization	Select Password .
Username	Enter root .
Password	Enter the password set when buying the ECS.
SSH Port	Enter 22 .

Step 4 A host record is displayed on the page. If **Succeed** is displayed in the **Verification Result** column, the host is added successfully.

If the host fails to be added, check the host configuration based on the failure details.

----End

Installing Dependency Tools on ECS

The sample program depends on Docker and Docker-Compose, which must be installed on the target ECS.

Step 1 Go to the **Phoenix Mall** project, choose **CICD > Deploy**, and find the **phoenix-sample-predeploy** application in the list.

Step 2 Click ******* and choose **Edit** from the drop-down list.

Step 3 Click the **Environment Management** tab and configure the host environment.

1. Click **Create Environment**, configure the following information, and click **Save**.

Table 1-27 Creating an environment

Parameter	Value
Environment	Enter phoenix-env .
Resource Type	Select Host .
OS	Select Linux .

 **NOTE**

If you do not have permission to create environments, contact the administrator to grant you permissions on the permission management page of the application.

2. Click **Import Host**. In the displayed dialog box, select the configured host cluster and host and click **Import**.
3. When a message indicating that the import is successful is displayed, close the window.

Step 4 On the **Deployment Actions** tab page, edit the actions of the application.

In action **Install Docker**, select **phoenix-env** from the **Environment** drop-down list. If a dialog box is displayed, asking you to confirm whether you want to change the environment to **phoenix-env** for the subsequent actions, click **OK**.

Step 5 Click **Save & Deploy** to start the deployment task.

If a message is displayed indicating successful deployment, the task is successfully executed.

Step 6 Log in to the ECS and check whether the dependency tools are successfully installed:

- Run the **docker -v** command to check the Docker image version.
- Run the **docker-compose -v** command to check the Docker-Compose version.

If the command output similar to **Figure 1-4** is displayed, the installation is successful.

Figure 1-4 Checking the Docker and Docker-Compose versions

```
root@ecs-he2e:~# docker -v
Docker version 18.09.0, build 4d60db4
root@ecs-he2e:~# docker-compose -v
docker-compose version 1.17.1, build 6d101fb
root@ecs-he2e:~#
```

----End

Configuring and Executing an Application

During deployment, configure the ECS in the environment list of the application and set the build task **phoenix-sample-ci** as the deployment source.

- Step 1** Go to the **Phoenix Mall** project, choose **CICD > Deploy**, and find the **phoenix-sample-standalone** application in the list.
- Step 2** Click ******* and choose **Edit** from the drop-down list.
- Step 3** Click the **Environment Management** tab and configure the host environment.
1. Click **Create Environment**, configure the following information, and click **Save**.

Table 1-28 Creating an environment

Parameter	Value
Environment	Enter phoenix-env .
Resource Type	Select Host .
OS	Select Linux .

2. Click **Import Host**. In the displayed dialog box, select the configured host cluster and host and click **Import**.
3. When a message indicating that the import is successful is displayed, close the window.

- Step 4** On the **Deployment Actions** tab page, edit the actions of the application.
1. Click **Select a Deployment Source**. Set the deployment source by referring to [Table 1-29](#).

Table 1-29 Configuring the deployment source

Configuration Item	Suggestion
Source	Select Build task .
Build Task	Select phoenix-sample-ci .
Environment	Select phoenix-env . If a dialog box is displayed, asking you to confirm whether you want to change the environment to phoenix-env for the subsequent actions, click OK .

2. Retain the default settings in actions **Decompress Files** and **Run Shell Commands**.

- Step 5** Click the **Parameters** tab and set parameters based on the SWR login command.
- You can obtain the login command from the console. For details, see [Configuring SWR](#).

Step 6 Click **Save & Deploy** to start deployment.

If a message is displayed indicating that the deployment is successful, continue with the next step. If the deployment fails, rectify the fault based on the failed action and the error information in logs.

Step 7 Verify the deployment result.

1. Open a browser, enter **http://IP:5000** in the address box, and press **Enter**. *IP* indicates the elastic IP address of the ECS. The Phoenix Mall homepage is displayed.
2. Enter **http://IP:5001** and press **Enter**. *IP* indicates the elastic IP address of the ECS. The Phoenix Mall dashboard is displayed.

----End

1.4.9 Step 7: Managing Project Tests

CodeArts TestPlan provides a one-stop cloud test platform that integrates DevOps agile test concepts, helping to efficiently manage test activities and ensure high-quality product delivery.

This section describes how Billy manages the test period of a project, including creating and executing test cases and tracking the test progress.

Creating a Sprint Test Plan

After the requirements (stories) to be implemented in sprint 4 are determined (that is, [Step 1: Managing Project Plans](#) is complete), the tester can write test cases while the developer codes.

Step 1 Create a test plan.

1. Go to the **Phoenix Mall** project and choose **Testing > Testing Plan**.
2. Click **Create Plan** and configure test plan information.
 - a. Basic information: Configure the following information and click **Next**.

Table 1-30 Basic information about a test plan

Sub-Configuration Item	Suggestion
Name	Enter Sprint4 .
Processor	Select Billy .
Plan Period	Use the same period as Sprint4 created in Req.
Associated Sprint	Select Sprint4 .

- b. Advanced settings: Select **Manual Test**. Confirm that the requirements in the list are the same as those of **Sprint4** in CodeArts Req, and then click **Save and Use**.

3. Return to the **Testing Plan** page. You can find the newly created test plan **Sprint4** in the list. The test plan is in the new state.

Step 2 Design test cases.

1. In the test plan **Sprint4**, click **Design**.
2. Expand the **Requirements** on the left of the page and find the story **User can query network of all stores**.
Click **...** and choose **Create Test Case**.
3. Enter **Store Network Query**, edit the test steps and expected results by referring to [Table 1-31](#), and click **Save**.

Table 1-31 Test steps

Test Step	Expected Result
Opening the homepage of Phoenix Mall	The page is displayed.
Clicking the Store Network menu	The Store Network page is displayed. Location filtering is available on the page, and information about recommended stores is displayed at the bottom of the page.
Selecting City A	The store information list of city A is displayed.

4. Create test cases for the other two stories in the same way.
5. Choose **Testing** > **Testing Plan** from the navigation pane to return to the test plan list.
In the list, the status of the test plan **Sprint4** is **Designing**.

----End

Executing a Test Plan

After developing story code and deploying applications on the test environment (see [Step 6: Deploying an Application \(CCE\)](#) or [Step 6: Deploying an Application \(ECS\)](#)), the developer can set the story status to **Resolved** and set the story handler to the tester.

In this case, the tester can execute the test cases corresponding to the story.

This section uses the store network query function as an example to describe how to execute test cases and how to report bugs when test case execution fails.

- Step 1** In the **Phoenix Mall** project, choose **Work** from the navigation pane, and click the **Sprints** tab.

In sprint 4, find the story **User can query network of all stores** and change the story status to **Testing**.

- Step 2** Go to the **Testing** > **Testing Case** page and select **Sprint4** in the upper part of the page.



- Step 3** In the list, click the case **Store Network Query**, change the status to **Testing**, and click **Save**.
- Step 4** Click the **Manual Test** tab and click  in the **Operation** column of the row where **Store Network Query** is located. The **Execute** window is displayed on the right.
- Step 5** Execute the steps one by one in the test environment.
- If the operation is successful, go to **Step 6** to continue the operation.
 - If the execution fails, for example, when you perform step 2, the page fails to be redirected and 404 is displayed, go to **Step 7**.
- Step 6** Return to the test case execution window and record the execution result.
1. In the table, set the actual result of all steps to **Successful**.
 2. In the upper part of the table, set the test case result to **Successful**.
 3. Select **Set the case status to Completed**.
 4. Click **Save** in the upper right corner of the page.
- The status of the test case automatically changes to **Completed**. Go to **Step 13** to continue the operation.
- Step 7** Return to the test case execution window and record the execution result.
1. In the table, set the actual result of step 1 to **Successful**.
 2. Set the actual result of step 2 to **Failed** and enter **Redirection failure. 404 is displayed on the page**.
 3. In the upper part of the table, set the test case result to **Failed**.
 4. Click **Save** in the upper right corner of the page.
- Step 8** Click  in the upper right corner of the page and choose **Create Defect**. The defect creation (work item creation) page is displayed.
- Step 9** At the end of the text box in the lower left corner of the page, view the reproduction procedure for automatically filling in the defect.
- Edit defect details by referring to **Table 1-32** and click **OK**. The work item list page is displayed.

Table 1-32 Defect details configuration

Configuration Item	Suggestion
Title	Enter Store Network 404 .
Assigned To	Select Chris .
Sprint	Select Sprint4 .

- Step 10** In the work item list, find bug **Store Network 404**. Click the name and click the **Associated** tab. You can find the test case **Store Network Query** by expanding **Associate with Test Case**.
- Step 11** Click the ID of an associated case to go to the case details page.

Click the **Bug** tab. A bug is displayed, that is, the defect created in [Step 9](#).

Step 12 After the developer fixes the defect and verifies it, they set the case result by referring to [Step 6](#) and set the corresponding defect status to **Closed**.

Step 13 Execute other test cases.

Step 14 When the status of all cases is **Completed**, choose **Testing > Testing Plan** to return to the test plan list where the status of **Sprint4** is displayed as **Completed**.

----End

Tracking a Test Plan

- Viewing the quality report

Through the quality report, the team can intuitively check the current progress of the test plan, including the requirement coverage rate, defects, case pass rate, and case completion rate.

On the **Testing > Testing Plan** page, click **Report** on the card of **Sprint4** to view the quality report.

- Customizing reports

In addition to built-in quality reports, the team can customize statistical reports as needed.

The following describes how to customize a statistical report by taking test case execution statistics as an example.

- a. On the **Quality Dashboard** page, click **Add Report** in the lower part of the page. In the dialog box that is displayed, select **Custom Reports**.
- b. Edit the report information by referring to [Table 1-33](#) and click **Save**.

Table 1-33 Report configuration

Configurati on Item	Suggestion
Report Title	Enter Test Case Execution Statistics .
Data Type	Select Test Cases .
Analysis Dimension	Select Result .

- c. The **Quality Dashboard** page is displayed. Check the new report at the bottom of the page.

1.4.10 Step 8: Configuring a Pipeline for Continuous Delivery

CodeArts Pipeline provides a visualized and customizable software pipeline for automatic delivery. It supports multiple task types, such as code check, build, and deployment tasks.

As the project progresses, each stage (build, release, and deployment) becomes more and more standardized. However, each stage is relatively independent and incomplete and cannot deliver business value directly. Only by effectively

connecting each stage to form a complete continuous delivery pipeline can we truly improve the efficiency and quality of software release and continuously create business value.

This section describes how Chris, a developer, connects code check, build, and deployment for continuous delivery.

Introduction to Preset Pipelines

There are five pipeline tasks preset in the sample project. You can view and use them as needed.



Table 1-34 Preset pipeline tasks

Preset Pipeline Task	Description
phoenix-workflow	Implements basic functions.
phoenix-workflow-test	Runs in the test environment.
phoenix-workflow-work	Implements the Worker function.
phoenix-workflow-result	Implements the Result function.
phoenix-workflow-vote	Implements the Vote function.

Configuring and Executing a Pipeline

A pipeline usually consists of multiple stages. You can add multiple jobs to each stage.

Step 1 Configure a pipeline.

1. Go to the **Phoenix Mall** project and choose **CICD > Pipeline**.
2. Find pipeline **phoenix-workflow**. Click ******* and choose **Edit**.
3. Add a code check stage.
 - a. Click  between **Code Source** and **Build** to add a stage.
 - b. Click  next to **Stage_1**. In the **Edit Stage** window, enter the stage name **Check** and click **Confirm**.
 - c. Click **Job**.
In the **New Job** window, click **Add** next to the **Check** extension.
 - d. Select the **phoenix-codecheck-worker** task and click **OK**.

 NOTE

The check task has three modes. This procedure uses the default mode **Full**. You can change the mode as required.

- **Full**: All files in the code repository are scanned.
- **Incremental (last commit)**: Incremental check is performed based on the latest commit file.
- **Incremental (last success)**: Incremental check is performed based on the changed files since the latest access control was passed.

4. Configure a deployment task.

Click the deployment task name, select the associated build task **phoenix-sample-ci**, and check the values of configuration items.

- The configurations of task **phoenix-sample-standalone** must be the same as those on the **Parameters** page of the task with the same name in CodeArts Deploy.
- The configurations of task **phoenix-cd-cce** must be the same as those on the **Parameters** page of the task with the same name in CodeArts Deploy.

 NOTE

Two deployment tasks are added in this example. If you selected only one deployment mode in preceding steps, keep the corresponding task and delete the other one.

5. Click **Save**.

Step 2 Go to the CCE console if you have configured deployment task **phoenix-cd-cce** in **Step 1**. Locate the target cluster and click its name to go to the **Overview** page.

Choose **Workloads** from the navigation pane, click the **Deployments** tab, and verify that no record exists in the list.

If there are records in the list, select all records, click **Delete**, select all options, and click **Yes** to clear the records in the list.

Step 3 Return to the pipeline list page. Click  in the row where **phoenix-workflow** is located, and click **Execute** in the window that is displayed to start the pipeline.

If  is displayed, the task is successfully executed.

If the task fails to be executed, check the failure cause in the failed task. You can open the step details page to view the task logs and rectify the faults based on the logs.

----End

Configuring Pass Conditions

To control the code quality, the code must be scanned and the number of errors must be within a reasonable range before being released. By adding quality gates, you can effectively automate the control process.

Step 1 On the details page of pipeline task **phoenix-workflow**, click **Edit**.

Step 2 In the **Check** stage, click **Pass Conditions**.

Step 3 In the **Pass Conditions** dialog box, click **Add** next to **Pass-Conditions-of-Standard-Policies**.

Step 4 Select **SystemPolicy** and click **OK**.

Step 5 Click **Save and Execute**.

If the number of check issues does not meet the pass condition, the pipeline task fails to be executed.

----End

Configuring Code Changes to Automatically Trigger a Pipeline

Through the following configuration, code changes can automatically trigger pipeline execution, implementing continuous project delivery.

Step 1 On the details page of pipeline task **phoenix-workflow**, click **Edit**.

Step 2 Click the **Execution Plan** tab, select **Code commit** under **Trigger Events**, select the **master** branch, and click **Save**.

Step 3 Modify the code and push it to the **master** branch to check whether the pipeline task is automatically executed.

----End

1.4.11 Releasing Resources

To avoid occupying too many resources, Sarah can release unused resources after trying the sample project.

The following resources can be released.

Table 1-35 Releasing resources

Resource	Procedure
CodeArts project	Choose Settings > General > Basic Information , click Delete Project , and follow the prompts to delete the project.
SWR organization and image	<ol style="list-style-type: none"> 1. Log in to the SWR console. 2. On the My Images page, select the image created in this example, click Delete, and follow the prompts to delete the image. 3. On the Organizations page, click the name of the organization to be deleted. Click Delete and follow the prompts to delete the organization.
CCE cluster	Log in to the CCE console. Locate the target cluster in the list, click ... , select Delete Cluster , and follow the prompts to delete it.

Resource	Procedure
ECS	Log in to the ECS console. Locate the ECS to be deleted in the list, choose More > Delete , and follow the prompts to delete it.

NOTICE

Released resources cannot be recovered. Exercise caution when performing these operations.
