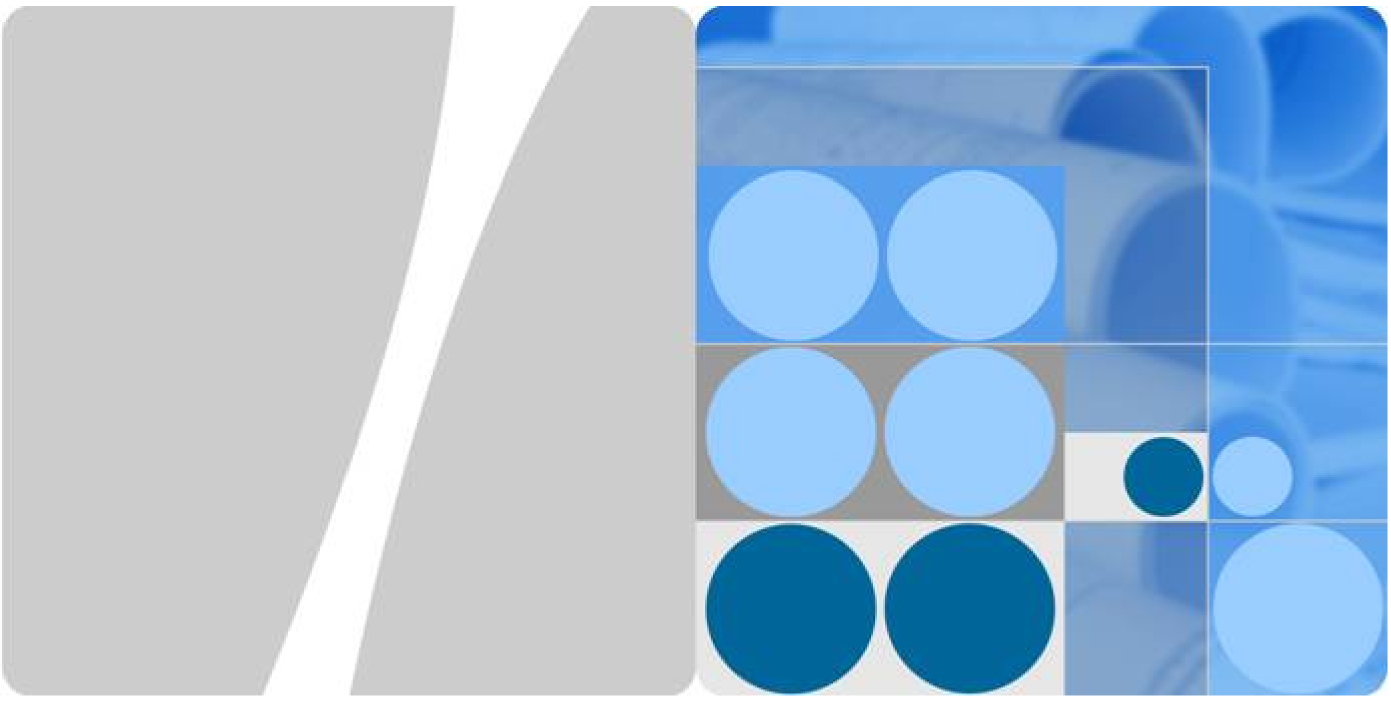


ServiceStage

User Guide

Date **2023-03-30**

Contents

1 Service Overview

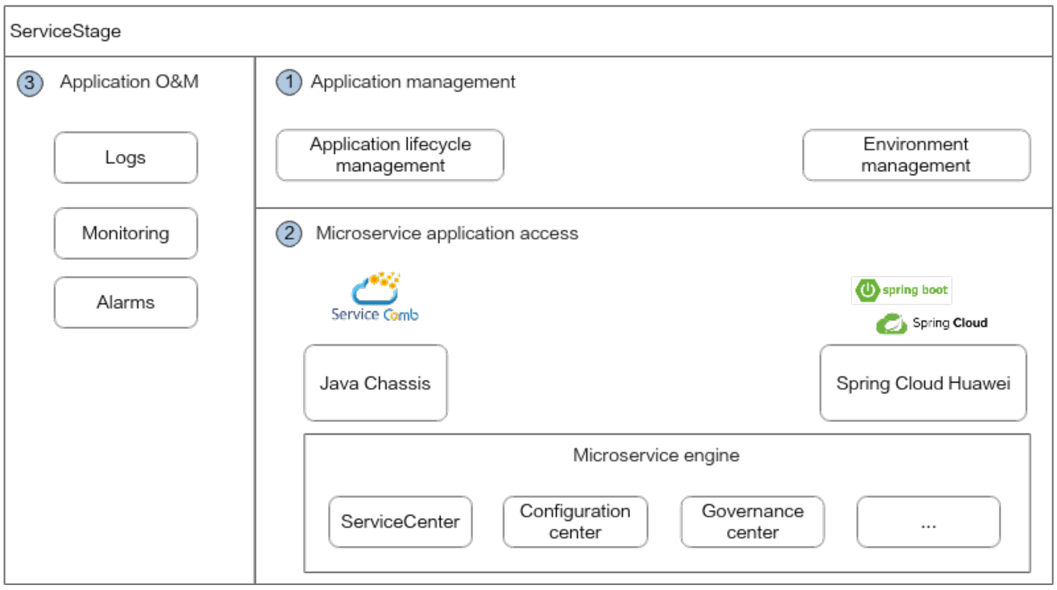
1.1 What Is ServiceStage?

ServiceStage is an application management and O&M platform that lets you deploy, roll out, monitor, and maintain applications all in one place. It supports technology stacks such as Java, Node.js, Docker, and Tomcat, and supports microservice applications such as Apache ServiceComb Java Chassis (Java chassis) and Spring Cloud, making it easier to migrate enterprise applications to the cloud.

ServiceStage provides the following capabilities:

1. Application management: application lifecycle management and environment management.
2. Microservice application access: Java chassis, and Spring Cloud. Furthermore, ServiceStage works with CSE to implement service registration and discovery, configuration management, and service governance. .
3. AOM: supports application O&M through logs, monitoring, and alarms.

Figure 1-1 ServiceStage functions



Application Management

- Application lifecycle management

  After an application is developed, it can be hosted on ServiceStage, which provides you with complete application lifecycle management:

  – Application creation by using the source code, and software packages (JAR)

  – Entire process management from application creation to logout, covering application creation, deployment, start, upgrade, rollback, scaling, stop, and deletion.

- Environment management

  An environment is a collection of compute, network, and middleware resources used for deploying and running an application component. ServiceStage combines the compute resources (such as CCE clusters), network resources (such as ELB instances and EIPs), and middleware (such as DCS instances, RDS instances, and CSE engines) into an environment, such as a development environment, testing environment, pre-production environment, or production environment.

  The resources within an environment can be networked together. Managing resources and deploying services by environment simplifies O&M.

Microservice Application Access

The microservice engine of ServiceStage supports access and governance of mainstream microservice frameworks. You can select a suitable microservice technology to quickly develop cloud applications to achieve complex and ever-changing service requirements.

- Supports the native ServiceComb microservice framework.

  Microservices developed by using the ServiceComb framework can be seamlessly connected to microservice engines.

  The microservice engine uses Apache ServiceComb Service Center, which is a RESTful-style, high-availability, and stateless service registry and discovery center and provides microservice discovery and management. Service providers can register their instance information with the registry and discovery center for users to discover and use. For details about Apache ServiceComb Service Center, see the following:

  – **https://github.com/apache/servicecomb-service-center/**

  – **https://service-center.readthedocs.io/en/latest/user-guides.html**

- Compatible with mainstream microservice open-source frameworks.

  Provides a simple access mode for microservices developed by using Spring Cloud. You only need to modify the dependencies and configurations to access microservice engines and use the unified governance policies.

- Provides microservice governance.

  After an application developed using the microservice framework is managed on ServiceStage, the microservice will be registered with the service registry and discovery center after the application instance starts. You can govern microservices by referring to "Microservice Governance" in the *User Guide*.

Application O&M

- Multi-dimensional metrics monitoring for application components, helping you understand the running status of online applications.
- GUI-based log query and search, helping you quickly locate faults.

1.2 Product Advantages

ServiceStage integrates successful experience in cloud transformation and technological innovation. As a one-stop application cloud platform, it has the following advantages over traditional platforms:

Table 1-1 Product advantages

Application Lifecycle	Traditional Platform	ServiceStage
Environment preparation	• Low resource obtaining efficiency (> 1 day) • Low resource utilization (< 30%)	• Self-service and efficient resource obtaining (minute-level) • Pay-per-use payment (auto scaling)

Application Lifecycle	Traditional Platform	ServiceStage
Service development	● Architecture coupled, a small change requires significant reconstruction ● Single technology, one technology is required to resolve all problems ● System release at a large granularity, requiring long response period	● Architecture decoupled Open API-based development makes the development, test, document, collaboration, and control activities of microservices are standardized and automated. ● Flexible access of various technologies Supports Java, PHP, Python, and Node.js. The high-performance REST/RPC microservice development framework provides out-of-the-box tools to reduce the development threshold. ● Agile The one-stop microservice governance console provides governance capabilities for microservices, such as load balancing, rate limiting, degradation, circuit breaker, fault tolerance, and fault injection. Supports microservice upgrade dark launch.
Installation and deployment	● Siloed system ● Manual deployment	Developers only need to use ServiceStage and source code repository to implement one-click automatic deployment and update.

Application Lifecycle	Traditional Platform	ServiceStage
Application upgrade	● Patch installation ● Manual upgrade ● Services interrupted	During rolling upgrades, services are evenly distributed to new and old instances; therefore, services are not interrupted.
Application O&M	● Application breakdown or crash ● Slow service response ● Insufficient system resources ● Difficult fault locating	● Real-time graphical display of application monitoring metrics CPU usage, alarms, node exceptions, running logs, and key events can be monitored in real time ● Microservice governance Supports microservice API-level SLA metrics (throughput, latency, and success rate) monitoring and governance in real time (in seconds), ensuring continuous service running.

1.3 Application Scenarios

1.3.1 Constructing Microservice Applications

Application Scenarios

Scenario

Different service modes of traditional projects in a single architecture must adopt a unified technical solution and technical platform. Each service module cannot be reused. If a module in the entire system is faulty, the entire system becomes unavailable. With the increasing complexity of enterprise services, the traditional monolithic architecture becomes more and more cumbersome, and it is difficult to adapt to flexible service requirements. Microservice applications can solve these problems.

Value

Microservice-based applications allow enterprises to divide a cumbersome system into several small service components. Among which, these components communicate with each other through lightweight protocols, decoupling the lifecycle management of each component.

Ever growing services may encounter various unexpected situations, such as instantaneous and large-scale concurrent access, service errors, and intrusion. The microservice architecture can be used to implement fine-grained service management and control to support service requirements.

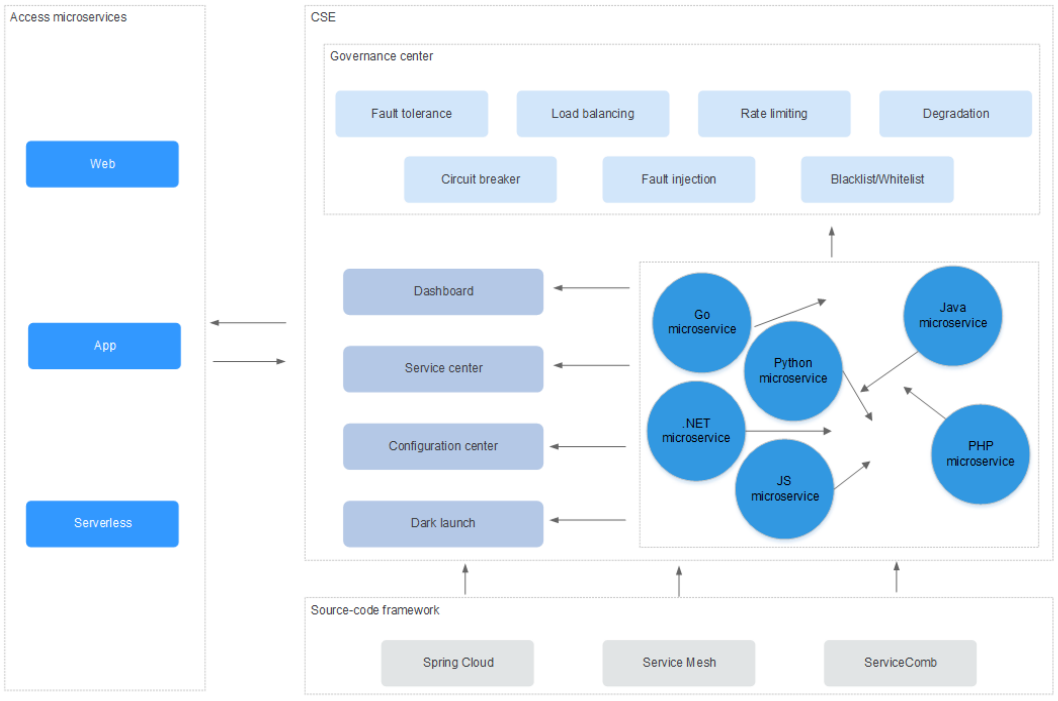
ServiceStage supports full lifecycle management of microservice applications. It supports stacks such as Java, Node.js, Docker, and Tomcat, and manages microservice applications such as Apache ServiceComb Java chassis and Spring Cloud without intrusion. In addition, it provides functions such as configuration management, monitoring and O&M, and service governance, making it easier to migrate enterprise microservice applications to the cloud.

Advantage

ServiceStage provides superior microservice application solutions and has the following advantages:

- Supports multiple microservice frameworks, such as native ServiceComb and Spring Cloud, and supports the dual-stack mode (SDK and service mesh interconnection). The service code can be directly managed on the cloud without modification.
- API First, which supports Swagger-based API management.
- Supports multiple languages, such as Java and Node.js.
- Provides functions such as service center, configuration center, dashboard, and dark launch.
- Provides complete microservice governance policies, including load balancing, fault tolerance, rate limiting, service degradation, circuit breaker, fault injection, and blacklist and whitelist. GUI-based operations can be performed for different service scenarios, greatly improving the availability of service governance.

Figure 1-2 Microservice application solution



- Implements mutual discovery between Spring Cloud and Java chassis.

Continuous Integration and Delivery

Scenario

It takes a lot of manpower and time in project creation, compilation, build, self-verification, integration verification, production environment-like verification, and rollout for a complex system, which is prone to errors caused by human factors. Continuous integration and delivery can resolve such problems due to its standardization and automation.

Value

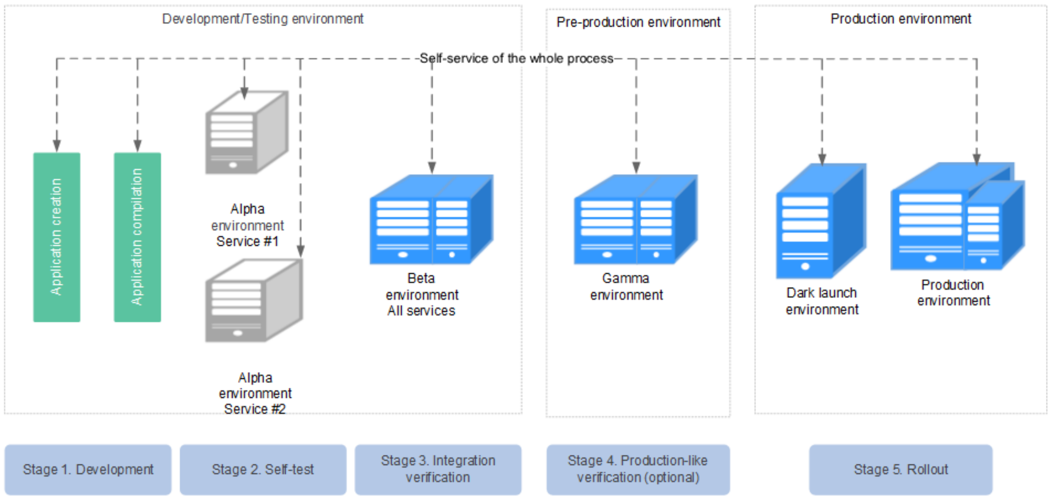
Manual execution is changed to automatic execution, which reduces errors and improves work efficiency.

The environment and process standards are unified, which facilitates service expansion and reduces upgrade and reconstruction costs.

Advantage

Based on the ServiceStage pipeline, the integration environment is unified and the delivery process is standardized. You can implement the self-service development, self-verification, integration verification, and rollout.

Figure 1-3 Continuous integration and delivery


Development/Testing environment
Pre-production environment
Production environment
Self-service of the whole process
Application creation
Application compilation
Alpha environment Service #1
Alpha environment Service #2
Beta environment All services
Gamma environment
Dark launch environment
Production environment
Stage 1. Development
Stage 2. Self-test
Stage 3. Integration verification
Stage 4. Production-like verification (optional)
Stage 5. Rollout

Dark Launch

Scenario

In dark launch, users are selected to test the beta version, ensuring smooth rollout of new features. Once new features become mature and stable, a formal version is released for all users to use.

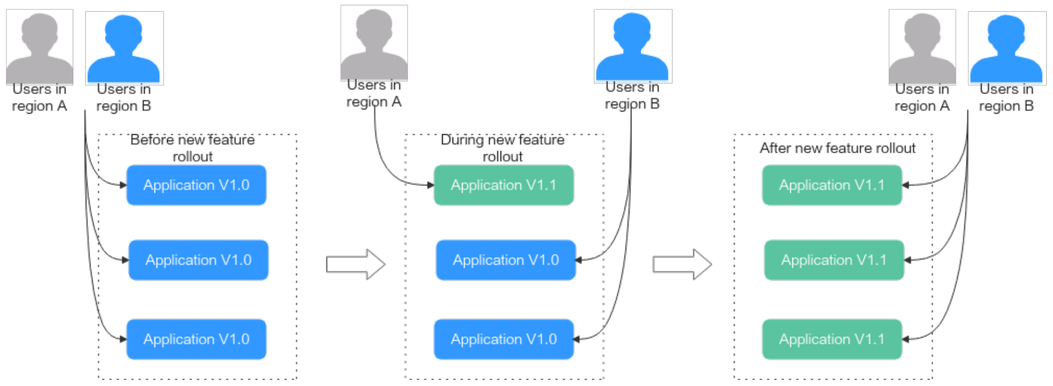
Value

Dark launch ensures stability of the entire system. During initial dark launch, problems can be detected and fixed.

Advantage

ServiceStage provides microservice-level dark launch.

Figure 1-4 Dark launch



1.3.2 Web Application Lifecycle Management

Application Scenarios

Scenario

Web applications are widely used. Enterprise service systems, online store systems, forums, blogs, Wiki systems, and online games may be web applications. Managing the lifecycle of web applications with different technical architectures is one of the main tasks of the enterprise IT department.

Value

Using a unified platform to manage various web applications can greatly reduce workload, improve efficiency, and quickly respond to complex and changeable service requirements.

Advantage

ServiceStage greatly improves the efficiency of enterprise-level web application development and O&M, making enterprises focus on service innovation. It has the following advantages:

- Deployment with a few clicks using WAR, JAR, or ZIP packages.
- One-stop O&M provides various O&M capabilities, such as upgrade, rollback, log, monitoring, and auto scaling.
- Seamless integration supports seamless integration with cloud services and applications such as ELB, RDS, and DCS.

1.4 Microservice Engine Versions

This section describes the versions supported by exclusive microservice engines.

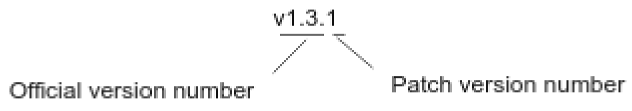
Version Description

The version number format is {major}.{minor}.{patch}.

where,

- {major}.{minor} indicates the official version number.
- {patch} indicates the patch version number.

For example, v1.3.1. 1.3 is the official version number, and 1 is the patch version number.



Version Support

- Microservice engine creation

  Only the microservice engine of the latest version can be created. The microservice engine of a specified version cannot be created.
- Microservice engine maintenance

  The latest three official versions can be maintained. For other versions, customer service will not be provided, including new functions, bug fixing, vulnerability fixing, and upgrades.

- Microservice engine upgrade
  - Official version upgrade:
    - 1.3 and 1.2 can be smoothly upgraded to 2.4 or later, and their functions are compatible.
    - Two earlier versions among the latest three official versions can be upgraded to the latest version. For example, if the latest three official versions are 2.4, 1.3, and 1.2, 1.2 and 1.3 are upgraded to 2.4.

      NOTE

      If the engine upgrade is not supported, for example, from 1.0 to 1.3, the management function of microservice engines may be unavailable. Exercise caution when performing this operation.

      You can contact customer service for risk evaluation before the upgrade.
  - Patch version upgrade: The microservice engine backend provides automatic patch version upgrade, for example, from 1.3.0 to 1.3.1.

Version Constraints

Version rollback is not supported after the microservice engine version is upgraded.

1.5 Glossary

Environment

An environment is a collection of compute, network, and middleware resources used for deploying and running an application component.

ServiceStage combines the compute resources (such as CCE clusters), network resources (such as ELB instances and EIPs), and middleware (such as DCS instances, RDS instances, and CSE engines) into an environment, such as a development environment, testing environment, pre-production environment, or production environment.

The resources within an environment can be networked together. Managing resources and deploying services by environment simplifies O&M.

Basic Resources

In ServiceStage, basic resources refer to the basic services that are required or optional to microservice application hosting and O&M, such as Cloud Container Engine (CCE).

Application

An application is a service system with functions and consists of one or more components.

Component

A component is a service feature implementation of an application. It is carried by code or software packages and can be independently deployed and run.

Stack

A technology stack includes the operating system, framework, and runtime system on which component running depends. It consists of attributes such as the technology stack name, type, status, and version. The version number complies with the **semantic versioning specifications**.

ServiceStage provides and manages the stack lifecycle. You only need to focus on service development to improve application hosting experience.

ServiceComb

Apache ServiceComb is an open-source microservice project. It is compatible with popular ecosystems and provides a one-stop open-source microservice solution, featuring out-of-the-box readiness, high performance, and multi-language programming. It aims to help enterprises, customers, and developers easily deploy enterprise applications on the cloud in the form of microservices and implement efficient O&M.

Microservice

Microservice is defined by service. If a process provides a service, it is a microservice. Each service has its own service functions. APIs that are not restricted by languages are open to other systems (HTTP frequently used). Multiple microservices form an application.

NOTE

In ServiceStage, a microservice is relative to a component.

Microservice instance

An instance is the minimum running and deployment unit of a microservice. Generally, it corresponds to an application process.

1.6 Restrictions

ServiceStage has the following restrictions, and each of them applies to every tenant in any region.

Restriction is not resource quota limit. It indicates the maximum capabilities that ServiceStage can provide for tenants. Users need to pay attention to these restrictions when selecting technologies and designing solutions.

Registry and Discovery

For details about the restrictions on professional microservice engines, see **Table 1-2**.

Table 1-2 Restrictions on professional microservice engines

Item	Restrictions
Heartbeat reporting	Every 30s at most for every microservice instance
Service discovery	Every 30s at most for every microservice instance
Microservice instance registration	10 per second

For details about the restrictions on exclusive microservice engines, see **Table 1-3**.

Table 1-3 Restrictions on exclusive microservice engines (highest specifications)

Item	Restrictions	Remarks
Heartbeat reporting	Every 20s at most for every microservice instance	Total rate limit: 2000 TPS
Service discovery	Every 20s at most for every microservice instance	-
Microservice instance registration	1000 per second	-

Exclusive Microservice Engine Types

An exclusive microservice engine of version 1.3.0 or later supports engine types listed in the following table. You can select an exclusive microservice engine of the required engine type.

Table 1-4 Exclusive microservice engine types

Engine Type	Application Scenario	AZ CPU Architecture
Cluster-deployed	Cluster-deployed engines support HA and host-level DR.	x86 or ARM. The hybrid architecture is not supported.

Engine Type	Application Scenario	AZ CPU Architecture
Single-node	Single-node-deployed engines do not support DR or HA. **NOTICE** This single-node-deployed engine will only be used for the development and test environment.	x86 or ARM.

Microservice Framework Versions

The following table lists the recommended versions of the microservice development framework.

- If you have used the microservice development framework of an earlier version to build applications, you are advised to upgrade it to the recommended version to obtain the stable and rich function experience.
- If an application has been developed using the Spring Cloud microservice development framework, you are advised to use **Spring Cloud Huawei** to access the application.
- If new microservice applications are developed based on open source and industry ecosystem components, you are advised to use the Spring Cloud framework.
- If you want to use the out-of-the-box governance capability and high-performance RPC framework provided by CSE, you are advised to use the Java chassis framework.

Framework	Recommended Versions	Description
Spring Cloud Huawei	1.10.6-2021.0.x or later	Uses **Spring Cloud Huawei** for connection. ● Spring Cloud version 2021.0.3 ● Spring Boot 2.6.7 Version description of the Spring Cloud microservice development framework: **https://github.com/huaweicloud/spring-cloud-huawei/releases**
Java Chassis	2.7.10 or later	Uses the software package provided by the open-source project for connection without introducing third-party software packages. Version description of the Java chassis microservice development framework: **https://github.com/apache/servicecomb-java-chassis/releases**.

Exclusive Microservice Engine Quota

A quota refers to the maximum number of resources that you can create in an exclusive microservice engine instance. **Table 1-5** provides the resource quota.

Table 1-5 Resource quota

Function	Resources	Quota	Modifiable	Precaution
Microservice management	Microservice versions	10,000	No	-
	Data volume of a single instance (KB)	200	Yes	Increasing quotas prolongs the microservice discovery latency.
	Number of contracts of a single microservice	500	No	-
Configuration management	Data volume of a single configuration item (KB)	128	No	-
	Data volume of an application-level configuration	2,000	No	-
Microservice governance	Application-level governance policies	1,000	No	A maximum of 1000 governance policies are supported.

NOTE

- A single governance policy contains governance rules and service scenarios. Governance rules and service scenarios occupy the same quota in the configuration center.
- Microservice version: In the microservice scenario, a version is used to mark the iteration record of a microservice to facilitate management of different iterations of a microservice.
- Microservice instance: An instance is the minimum running and deployment unit of a microservice. Generally, it corresponds to an application process. A microservice can be deployed in multiple containers or VMs to enable multiple instances to run at the same time.
- Configuration item: The configuration in the microservice scenario is to control the values of some variables in the program code. For example, dynamic configuration is to dynamically change the values of some variables during microservice running.

1.7 Edition Differences

Table 1-6 Functions

Function		Specifications
Management scale	Maximum number of application component instances in a region	5,000
	Maximum number of instances supported by a component	200
Microservice	Microservice engine	The number of microservice engines that can be created depends on the engine type and the number of VMs in the AZ where the microservice engine is located. Each AZ can contain a maximum of 50 VMs. A cluster-deployed microservice engine consumes five VMs.
	Java microservice development SDK	Supported
	Spring Cloud microservice access	
	Registry center	
	Configuration center	
	Real-time dashboard	
	Load balancing	
	Rate limiting	
	Service degradation	
	Fault tolerance	
	Circuit breaker	
	Fault injection	
	Blacklist and whitelist	

Function		Specifications
Application lifecycle management	Multi-language application management (Java, PHP, Python, Node.js, Tomcat, and Docker)	Supported
	Application lifecycle management (auto scaling, upgrade, rollback, start, stop, restart, and deletion)	
	Basic application monitoring (running status, CPU, memory, and disk usage)	
	Container-based deployment. Java, PHP, Python, Node.js, Tomcat, and Docker are supported.	
	Access control	
	Application domain name management	
	Auto scaling	
	Event analysis	
	Log analysis	
	Threshold rules	
Deployment source management	Docker image package management	Supported
	Repository permission management	
Application operations management	Log storage and search	1 TB/month; storage duration: 30 days
	Log analysis	500 GB
	API call (log data query)	10 GB/month
	Host monitoring	200 VMs; storage duration: 1 year
	Metrics in real-time monitoring	2000
	Custom metrics	500
	API call (metrics query)	5,000,000 times/month
	Events and alarms	500,000 records/moth; storage duration: 30 days

Function		Specifications
	Intelligent threshold rules	50

CSE Instance Specifications

Microservice engines have professional and exclusive editions.

- Professional microservice engine: Cloud Service Engine (CSE) is a free experience engine provided by ServiceStage. You can use a professional engine to experience all product capabilities of ServiceStage, such as service governance and configuration management. Engines are shared by all tenants; however, the performance may be affected by other tenants. A professional engine cannot be upgraded to the exclusive edition.
- Exclusive microservice engine: Exclusive engines are commercial engines that manage large-scale microservice applications. You can select different engine specifications based on service requirements, and specifications cannot be changed. Exclusive engines are exclusively used; therefore, the performance is not affected by tenants.

The following describes the maximum number of instances supported by CSE.

Table 1-7 CSE Instance Specifications

Type	Microservice Instances	Configuration Items
Professional microservice engine	1,000	-
Exclusive microservice engine	100	600
	500	3,000
	2,000	12,000

1.8 Permissions Management

If you need to assign different permissions to employees in your enterprise to access your ServiceStage resources, IAM is a good choice for fine-grained permissions management. IAM provides identity authentication, permissions management, and access control, helping you secure access to your resources.

With IAM, you can use your cloud service account to create IAM users for your employees, and assign permissions to the users to control their access to specific resource types. For example, some software developers in your enterprise need to use ServiceStage resources but must not delete them or perform any high-risk operations. To achieve this result, you can create IAM users for the software developers and grant them only the permissions required for using ServiceStage resources.

If your cloud service account does not need individual IAM users for permissions management, you may skip over this chapter.

IAM can be used free of charge. You pay only for the resources in your account. For details, see **IAM Service Overview**

ServiceStage Permissions

By default, new IAM users do not have any permissions assigned. You need to add a user to one or more groups, and assign permissions policies to these groups. The user then inherits permissions from the groups it is a member of. This process is called authorization. After authorization, the user can perform specified operations on ServiceStage based on the granted permissions policies.

ServiceStage is a project-level service deployed and accessed in specific physical regions. To assign ServiceStage permissions to a user group, specify the scope as region-specific projects and select projects for the permissions to take effect. If **All projects** is selected, the permissions will take effect for the user group in all region-specific projects. When accessing ServiceStage, the users need to switch to a region where they have been authorized to use cloud services.

You can grant users permissions by using roles and policies.

- Roles: A type of coarse-grained authorization mechanism that defines permissions related to user responsibilities. This mechanism provides only a limited number of service-level roles for authorization. When using roles to grant permissions, you also need to assign other roles on which the permissions depend to take effect. However, roles are not an ideal choice for fine-grained authorization and secure access control.
- Policies are a type of fine-grained authorization mechanism that defines permissions required to perform operations on specific cloud resources under certain conditions. This mechanism allows for more flexible policy-based authorization, meeting requirements for secure access control.

Table 1-8 lists all the system policies supported by ServiceStage. System policies are recommended. System roles are used only for compatibility with existing permission configurations.

Table 1-8 ServiceStage system permissions

Role/Policy Name	Description	Type	Depended System Permissions
ServiceStage FullAccess	Full permissions for ServiceStage.	System-defined policy	None
ServiceStage ReadOnlyAccess	Read-only permissions for ServiceStage.	System-defined policy	None

Role/Policy Name	Description	Type	Depended System Permissions
ServiceStage Developer	ServiceStage developer, who has permissions for operating applications, components, and environments, but excluding permissions for creating basic resources.	System-defined policy	None
CSE Admin	Administrator permissions for CSE.	System-defined policy	None
CSE Viewer	View permissions for CSE.	System-defined policy	None
ServiceStage Administrator	ServiceStage administrator, who has full permissions for this service.	System-defined role	CCE administrator, VPC administrator, SWR administrator, Server administrator, and OBS administrator.
ServiceStage Operator	ServiceStage operator, who has the read-only permission for this service.	System-defined role	CCE administrator, VPC administrator, SWR administrator, Server administrator, and OBS administrator.
ServiceStage Developer	ServiceStage developer, who has full permissions for this service.	System-defined role	CCE administrator, VPC administrator, SWR administrator, Server administrator, and OBS administrator.

If these policies do not meet actual requirements, you can customize policies based on **Table 1-9** and **Table 1-10**. For more information, see **Creating a Custom Policy**.

Table 1-9 Common ServiceStage operations supported by each system policy

Operation	ServiceStage ReadOnlyAccess	ServiceStage Developer	ServiceStage FullAccess
Creating an application	x	√	√
Modifying an application	x	√	√
Querying an application	√	√	√
Deleting an application	x	√	√
Creating a component	x	√	√
Searching for a component	√	√	√
Deploying a component	x	√	√
Maintaining a component	x	√	√
Deleting a component	x	√	√
Creating a build job	x	√	√
Modifying a build job	x	√	√
Querying a build job	√	√	√

Operation	ServiceStage ReadOnlyAccess	ServiceStage Developer	ServiceStage FullAccess
Starting a build job	x	√	√
Deleting a build job	x	√	√
Creating a pipeline	x	√	√
Modifying a pipeline	x	√	√
Querying a pipeline	√	√	√
Starting a pipeline	x	√	√
Cloning a pipeline	x	√	√
Deleting a pipeline	x	√	√
Creating repository authorization	x	√	√
Modifying repository authorization	x	√	√
Querying repository authorization	√	√	√
Deleting repository authorization	x	√	√

Table 1-10 Common CSE operations supported by each system policy

Operation	CSE Viewer	CSE Admin
Creating a microservice engine	x	√

Operation	CSE Viewer	CSE Admin
Maintaining a microservice engine	x	√
Querying a microservice engine	√	√
Deleting a microservice engine	x	√
Creating a microservice	x	√
Querying a microservice	√	√
Maintaining a microservice	x	√
Deleting a microservice	x	√
Creating microservice configurations	x	√
Querying microservice configurations	√	√
Editing microservice configurations	x	√
Deleting microservice configurations	x	√
Creating a microservice governance policy	x	√
Querying a microservice governance policy	√	√
Editing a microservice governance policy	x	√
Deleting a microservice governance policy	x	√

Fine-grained permissions

NOTE

- SWR does not support fine-grained permissions. Related permissions need to be authorized separately.

To use a custom fine-grained policy, log in to the IAM console as an administrator and select the desired fine-grained permissions for ServiceStage and CSE.

- **Table 1-11** describes fine-grained permission dependencies of CSE.
- **Table 1-12** describes fine-grained permission dependencies of ServiceStage.

Table 1-11 Fine-grained permission dependencies of CSE

Permission Name	Description	Permission Dependency	Application Scenario
cse:engine:list	Lists all microservice engines.	None	Engine list view
cse:engine:get	Views engine information.	cse:engine:list	Engine details view (supported by only exclusive microservice engines)
cse:engine:modify	Modifies an engine.	● cse:engine:list ● cse:engine:get	Engine modification, including enabling or disabling public access, enabling or disabling security authentication, and retrying failed tasks, supported by only exclusive microservice engines
cse:engine:upgrade	Upgrades an engine.	● cse:engine:list ● cse:engine:get	Engine upgrade, including upgrading the engine version, supported by only exclusive microservice engines.
cse:engine:delete	Deletes an engine.	● cse:engine:list ● cse:engine:get ● vpc:ports:get ● vpc:ports:delete	Engine deletion (supported by only exclusive microservice engines.)
cse:engine:create	Creates an engine.	● cse:engine:get ● cse:engine:list ● ecs:cloudServerFlavors:get ● vpc:vpcs:get ● vpc:vpcs:list ● vpc:subnets:get ● vpc:ports:get ● vpc:ports:create	Engine creation, including creating an engine and creating a backup or restoration task, supported by only exclusive microservice engines.
cse:config:modify	Modifies configuration and management functions.	● cse:engine:list ● cse:engine:get ● cse:config:get	Modification on global and governance configurations

Permission Name	Description	Permission Dependency	Application Scenario
cse:config:get	Views configuration and management functions.	● cse:engine:list ● cse:engine:get	Service configuration view
cse:governance:modify	Modifies the governance center.	● cse:engine:list ● cse:engine:get ● cse:config:get ● cse:config:modify ● cse:registry:get ● cse:registry:modify ● cse:governance:get	Service governance creation and modification
cse:governance:get	Views the governance center.	● cse:engine:list ● cse:engine:get ● cse:config:get ● cse:registry:get	Service governance view
cse:registry:modify	Modifies service registry and management.	● cse:engine:list ● cse:engine:get ● cse:registry:get	Service modification
cse:dashboard:modify	Modifies the dashboard.	● cse:engine:list ● cse:engine:get ● cse:registry:get ● cse:dashboard:get ● cse:registry:modify	Dashboard modification
cse:dashboard:get	Views the dashboard.	● cse:engine:list ● cse:engine:get ● cse:registry:get	Dashboard view
cse:registry:get	Views service registry and management.	● cse:engine:list ● cse:engine:get	Service catalog view

NOTE

The dashboard does not need to be authenticated but requires registry permissions, because it uses the service catalog function to distinguish services.

Table 1-12 Fine-grained permission dependencies of ServiceStage

Permission Name	Description	Permission Dependency	Application Scenario
servicestage:app:get	Views application information.	servicestage:app:list	Application information view
servicestage:app:create	Creates an application.	• servicestage:app:get • servicestage:app:list • servicestage:assembling:get • servicestage:assembling:list • servicestage:assembling:create	Application creation
servicestage:app:modify	Updates an application.	• servicestage:app:get • servicestage:app:list • servicestage:assembling:get • servicestage:assembling:list • servicestage:assembling:modify	Application update
servicestage:app:delete	Deletes an application.	• servicestage:app:get • servicestage:app:list • servicestage:assembling:delete	Application deletion
servicestage:app:list	Views the environment and application list.	None	Environment and application list view
servicestage:environment:create	Creates an environment.	• servicestage:app:get • servicestage:app:list	Environment creation

Permission Name	Description	Permission Dependency	Application Scenario
servicestage:environment:modify	Updates an environment.	• servicestage:app:get • servicestage:app:list	Environment update
servicestage:environment:delete	Deletes an environment.	• servicestage:app:get • servicestage:app:list	Environment deletion
servicestage:assembling:get	Views the build information.	servicestage:assembling:list	Build information view
servicestage:assembling:create	Creates a build job.	• servicestage:assembling:get • servicestage:assembling:list	Build job creation.
servicestage:assembling:modify	Modifies a build job.	• servicestage:assembling:get • servicestage:assembling:list	Build job modification
servicestage:assembling:delete	Deletes a build job.	• servicestage:assembling:get • servicestage:assembling:list	Build job deletion
servicestage:assembling:list	Views the build list.	None	Build list view

Helpful Links

- **IAM Service Overview**
- **4.1 Creating a User and Granting Permissions**

1.9 Relationship with Other Cloud Services

ServiceStage is a one-stop cloud application platform integrating knowledge and experience in cloud transformation and technology innovation. It integrates core functions of services, covering infrastructure, storage, database, software repository, monitoring and O&M, and middleware services.

ServiceStage can be used to fully experience functions of various services.

- ServiceStage implements interconnection with source code repositories, such as Gitee, GitHub, GitLab, and Bitbucket. After being bound, you can directly pull up the source code from source code repositories for building.

- ServiceStage integrates deployment source management and archives the built software packages (or image packages) to the corresponding repositories and organizations.
- ServiceStage integrates related basic resources, such as VPC, CCE, EIP, and ELB. When deploying applications, you can directly use existing or new basic resources.
- ServiceStage integrates the Cloud Service Engine (CSE). You can perform operations related to microservice governance on ServiceStage console.
- ServiceStage integrates Application Operations Management (AOM). You can perform operations related to application O&M.
- ServiceStage integrates storage, database, and cache services and implements persistent data storage through simple configuration.

2 Getting Started

Overview

This section describes the environments, applications, and components of ServiceStage.

- An environment is a combination of compute resources (such as CCE clusters), network resources (such as load balancers and EIPs), and middleware (such as DCS instances, RDS instances, and CSE engines). When deploying applications and components, you need to select an environment. After the environment is selected, the resources contained in the environment are automatically loaded.
- An application is a service system with functions and consists of one or more components. For example, a typical Enterprise Resource Planning (ERP) system is an application, which generally consists of modules such as accounting, finance, production control, logistics, procurement, distribution, and inventory. These modules are closely related to each other, and each module is a component.
- A component is an implementation of a service feature of an application, for example, a module of the preceding ERP system. In microservice application scenarios, each component has an independent software package and can be deployed and run independently. The deployed component is called a component instance. A component can have multiple component instances to form a cluster to ensure high reliability of applications and components. O&M operations are supported, such as starting, stopping, deploying (upgrading), rolling back, and scaling application component instances, viewing logs, viewing events, setting access modes, and setting threshold alarms.

This document describes how to set up an environment, create an application, create and deploy a component, confirm the deployment, access the application, and perform application O&M.

Prerequisites

1. Create a VPC. For details, see **Creating a VPC**.
2. Create a CCE cluster. For details, see **Creating a CCE Cluster**.

   The cluster must contain at least one ECS node with 8 vCPUs and 16 GB memory or two ECS nodes with 4 vCPUs and 8 GB memory and be bound to an EIP.

3. Create a bucket for storing software packages. For details, see **Creating a Bucket**.
4. For details about how to create a microservice engine with security authentication disabled, see **10.2 Creating a Microservice Engine**.
5. This example provides a microservice demo, which is compiled, built, and packaged locally. You must install the Java JDK and Maven on the local host and the local host can access the Maven central library.

   After the installation, open the Command Prompt, and run the **mvn -v** command to query the versions of Java JDK and Maven. If their versions are displayed, the installation is successful. In this example, Maven 3.6.3 and JDK 1.8.0 are used.

   ```
   Apache Maven 3.6.3 (cecedd343002696d0abb50b32b541b8a6ba2883f)
   ...
   Java version: 1.8.0_201,......
   ```

Preparing Software Packages

Step 1 **Download** the microservice demo source code package to the local host and decompress it.

Step 2 In the root directory of the project (for example, **D:\servicecomb-samples-master\servicecomb-samples-master\ServiceComb-SpringMVC**), run the **cmd** and **mvn clean package** commands to compile and package the Java project.

```
D:\servicecomb-samples-master\servicecomb-samples-master\ServiceComb-SpringMVC>mvn clean
package
......
[INFO] --- maven-jar-plugin:2.6:jar (default-jar) @ servicecomb ---
[INFO] Building jar: D:\servicecomb-samples-master\servicecomb-samples-master\ServiceComb-
SpringMVC\target\servicecomb-0.0.1-SNAPSHOT.jar
[INFO]
[INFO] --- spring-boot-maven-plugin:1.5.9.RELEASE:repackage (default) @ servicecomb ---
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
......
```

After compilation, the **servicecomb-0.0.1-SNAPSHOT.jar** software package is generated in the **target** subdirectory in the root directory of the project (for example, **D:\servicecomb-samples-master\servicecomb-samples-master\ServiceComb-SpringMVC\target**), and the message "BUILD SUCCESS" is displayed.

Step 3 Upload the generated **servicecomb-0.0.1-SNAPSHOT.jar** software package to the created OBS bucket.

For details about how to upload the software package, see "Uploading an Object" in the *Object Storage Service User Guide***Uploading a File**.

----End

Creating an Organization

Step 1 Log in to ServiceStage.

Step 2 Choose **Deployment Source Management** > **Organization Management**.

Step 3 Click **Create Organization**. On the displayed page, set **Organization Name**.

Step 4 Click **OK**.

----End

Creating an Environment

Step 1 Log in to ServiceStage.

Step 2 Choose **Environment Management** > **Create Environment** and set the environment information by referring to the following table.

Parameter	Description
Environment	Enter an environment name, for example, **test-env**.
* Enterprise Project	Enterprise project. Enterprise projects let you manage cloud resources and users by project. It is available after you **create an enterprise project**.
VPC	Select the VPC prepared in **Prerequisites**. **NOTE** The VPC cannot be modified after the environment is created.
Environment Type	Select **Kubernetes**.

Step 3 Click **Create Now**.

Step 4 In the **Resource Settings** area, choose **Cloud Container Engine** from **Compute** and click **Manage Resource**.

Step 5 In the dialog box that is displayed, select the created CCE cluster and click **OK**.

Step 6 In the **Resource Settings** area, choose **Cloud Service Engine** from **Middleware** and click **Manage Resource**.

Step 7 In the dialog box that is displayed, select the created CSE engine and click **OK**.

----End

Creating an Application

Step 1 Log in to ServiceStage.

Step 2 Choose **Application Management** > **Create Application**.

Step 3 Enter an application name, for example, **test-servicestage**. Select an enterprise project.

Step 4 Click **OK**.

----End

Creating and Deploying a Component

Step 1 Log in to ServiceStage.

Step 2 Choose **Application Management**. The application list is displayed.

Step 3 Select the application (for example, **test-servicestage**) created in **Creating an Application** and click **Create Component** in the **Operation** column.

Step 4 In the **Basic Information** area, set the following mandatory parameters. Retain the default values for other parameters.

Parameter	Description
Component Name	Enter a component name, for example, **test-cse**.
Environment	Select the environment created in **Creating an Environment**, for example, **test-env**.

Step 5 In the **Component Package** area, set the following mandatory parameters. Retain the default values for other parameters.

Parameter	Description
Stack	Select **Java**.
Upload Method	Click **Software Package** and select **servicecomb-0.0.1-SNAPSHOT.jar** uploaded in **Preparing Software Packages**.

Step 6 In the **Build** area, set the following mandatory parameters. Retain the default values for other parameters.

Parameter	Description
Organization	An organization is used to manage images generated during component build. Select the organization created in **Creating an Organization**.
Environment	Select **Use current environment** to use the CCE cluster in the deployment environment to which the component belongs to build an image. In the current environment, masters and nodes in the CCE cluster must have the same CPU architecture. Otherwise, the component build fails.

Step 7 Click **Next**.

Step 8 In the **Resources** area, retain the default settings.

Step 9 In the **Access Mode** area, retain the default settings.

Step 10 In the **Local Time** area, retain the default settings.

Step 11 In the **Advanced Settings** area, choose **Advanced Settings** > **Microservice Engine**.

1. Click **Bind Microservice Engine**.

2. Select the managed microservice engine in the current environment.
3. Click **OK**.

Step 12 Click **Create and Deploy** to deploy the component.

----End

Confirming the Deployment Result

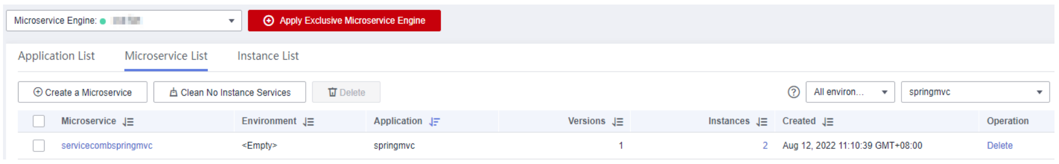
Step 1 Log in to ServiceStage.

Step 2 Choose **Cloud Service Engine** > **Microservice Catalog**.

Step 3 Select the microservice engine where the component is deployed from the microservice engine drop-down list.

Step 4 Select **springmvc** from the **All applications** drop-down list.

If the microservice servicecombspringmvc is displayed and the number of microservice instances is 2, the deployment is successful.



	Microservice	Environment	Application	Versions	Instances	Created	Operation
☐	servicecombspringmvc	<Empty>	springmvc	1	2	Aug 12, 2022 11:10:39 GMT+08:00	Delete

----End

Accessing an Application

Step 1 Log in to ServiceStage.

Step 2 Choose **Application Management**. The application list is displayed.

Step 3 Click the application created in **Creating an Application** (for example, **test-servicestage**). The **Overview** page is displayed.

Step 4 On the **Component List** tab, click the component created in **Creating and Deploying a Component** (for example, **test-cse**). The **Overview** page is displayed.

Step 5 Click **Access Mode**.

Step 6 Click **Add Service** in the **TCP/UDP Route Configuration** area and set parameters by referring to the following table.

Parameter	Description
Service Name	Retain the default value.
Access Mode	Select **Public network access**.
Access Type	Select **Elastic IP address**.
Service Affinity	Retain the default value.

Parameter	Description
Port Mapping	1. **Protocol**: Select **TCP**. 2. **Container Port**: Enter **8080**. 3. **Access Port**: Select **Automatically generated**.

Step 7 Click **OK**.

Step 8 Click the access address in the **Access Address** column to access the application.

The following information is displayed:

```
{"message":"Not Found"}
```

Step 9 Enter **http://***Access address generated in* ***Step 7*****/rest/helloworld?name=ServiceStage** in the address box of the browser to access the application again.

The following information is displayed:

```
"ServiceStage"
```

----End

Application O&M

Step 1 Log in to ServiceStage.

Step 2 Click **Application Management**.

Step 3 Click the application created in **Creating an Application** (for example, **test-servicestage**). The **Overview** page is displayed.

Step 4 On the **Component List** tab, click the component created in **Creating and Deploying a Component** (for example, **test-cse**). The **Overview** page is displayed, where you can perform O&M operations.

----End

3 Overview

ServiceStage is an application management and O&M platform that lets you deploy, roll out, monitor, and maintain applications all in one place. It supports technology stacks such as Java, Node.js, Docker, and Tomcat, and supports microservice applications such as Apache ServiceComb Java Chassis (Java chassis) and Spring Cloud, making it easier to migrate enterprise applications to the cloud.

This document describes how to use ServiceStage to create, deploy, and maintain application components and perform service governance.

Prerequisites

1. You have registered a cloud account.
2. The login account has the permission to use ServiceStage. For details, see **4.1 Creating a User and Granting Permissions**.

Logging In to ServiceStage

Step 1 Log in to the management console.

Step 2 Click ☰ in the upper left corner, and click **ServiceStage**.

- If you log in for the first time, click **Authorize** on the displayed service authorization page to authorize ServiceStage to use the services on which it depends. Then, the **ServiceStage** console is displayed.
- If this is not your first login, the **ServiceStage** console is displayed directly.

----End

Console Description

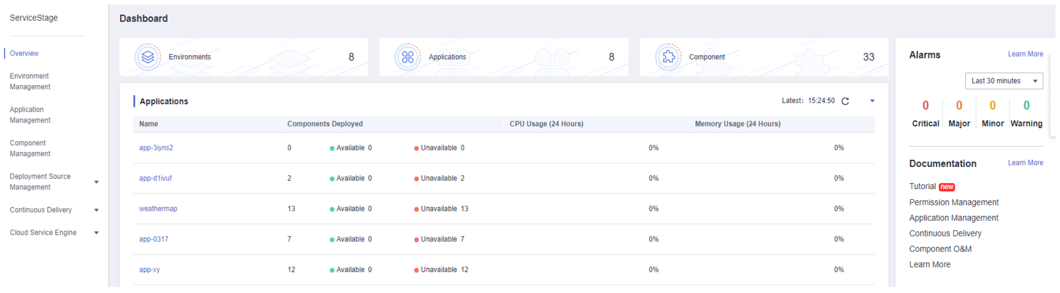
Table 3-1 describes ServiceStage console.

Table 3-1 ServiceStage console

Module	Description
Overview	The **Overview** page provides the ServiceStage overview, including the documentation help, applications, environments, components, and alarms.
Environment Management	An environment is a collection of compute, network, and middleware resources used for deploying and running a component. The **Environment Management** page allows you to create, edit, and delete environments, and configure resources (manage and remove resources). Created environments are displayed in a list.
Application Management	An application is a service system with complete functions and consists of one or more components related to features. The **Application Management** page allows you to create, edit, and delete applications. Created applications and the number of components created under them are displayed in a list, and entries for creating components under applications are available.
Component Management	A component is a service feature implementation of an application. It is carried by code or software packages and can be independently deployed and run. The **Component Management** page displays components of all applications in a list, and provides the component details page as well as the entries for component creation and O&M.
Deployment Source Management	Provides functions such as organization management and image repository. ● Organization management is used to isolate images and assign access permissions (read, write, and manage) to different users. ● Image repositories are used to store and manage Docker images.

Module	Description
Continuous Delivery	Provides functions such as viewing build projects, releasing build projects, and authorizing repositories. ● Build The software package or image package can be generated with a few clicks in a build job. In this way, the entire process of source code pull, compilation, packaging, and archiving is automatically implemented. ● Pipeline One-click deployment can be achieved through pipeline. In this way, the entire process of source code pull, compilation, packaging, archiving, and deployment is automatically implemented. This unifies the integration environment and standardizes the delivery process. ● Repository Authorization You can create repository authorization so that build projects and application components can use the authorization information to access the software repository.
Cloud Service Engine	Provides operation entries for engine instance management, dashboard usage, microservice catalog management, microservice governance, configuration management, and system management.

Figure 3-1 ServiceStage console



Applications

Latest: 15:24:50

Name	Components Deployed			CPU Usage (24 Hours)	Memory Usage (24 Hours)
app-3iyns2	0	Available 0	Unavailable 0	0%	0%
app-d1ivuf	2	Available 0	Unavailable 2	0%	0%
weathermap	13	Available 0	Unavailable 13	0%	0%
app-0317	7	Available 0	Unavailable 7	0%	0%
app-xy	12	Available 0	Unavailable 12	0%	0%

4 Permissions Management

4.1 Creating a User and Granting Permissions

Use **IAM** to implement fine-grained permissions control over your ServiceStage resources. With IAM, you can:

- Create IAM users for employees based on the organizational structure of your enterprise. Each IAM user has their own security credentials for access to ServiceStage resources.
- Grant only the permissions required for users to perform a task.
- Entrust an account or cloud service to perform professional and efficient O&M on your ServiceStage resources.

If your account does not require individual IAM users, skip this section.

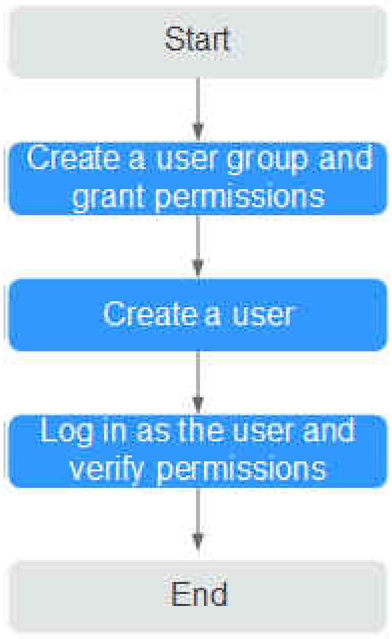
This section describes the procedure for granting permissions (see **Figure 4-1**).

Prerequisites

Before assigning permissions to user groups, you should learn about ServiceStage policies and select the policies based on service requirements. For details about system permissions supported by ServiceStage, see **1.8 Permissions Management**.

Process Flow

Figure 4-1 Process for granting ServiceStage permissions


Start
Create a user group and grant permissions
Create a user
Log in as the user and verify permissions
End

1. .

   Create a user group on the IAM console, and assign the **ServiceStage ReadOnlyAccess** policy to the group.
2. .

   Create a user on the IAM console and add the user to the group created in **1**.
3. and verify permissions.

   Log in to the ServiceStage console as the created user, and verify that it only has read permissions for ServiceStage.
   - Select **ServiceStage** from **Service List**. On the **Application Management** page, click **Create Application**. If a message appears indicating insufficient permissions to access the service, the **ServiceStage ReadOnlyAccess** policy has already taken effect.
   - Choose any other service in **Service List**. If a message appears indicating insufficient permissions to access the service, the **ServiceStage ReadOnlyAccess** policy has already taken effect.

4.2 Creating a Custom Policy

Custom policies can be created as a supplement to the system policies of ServiceStage.

You can create custom policies in either of the following ways:

- Visual editor: Select cloud services, actions, resources, and request conditions. This does not require knowledge of policy syntax.
- JSON: Create a policy in the JSON format from scratch or based on an existing policy.

For details, see **Creating a Custom Policy**. The following section contains examples of common ServiceStage custom policies.

Example Custom Policy

This procedure creates a policy that an IAM user is prohibited to create and delete a microservice engine.

```
{
        "Version": "1.1",
        "Statement": [
                {
                        "Action": [
                                "cse:*:*"
                        ],
                        "Effect": "Allow"
                },
                {
                        "Action": [
                                "cse:engine:create",
                                "cse:engine:delete"
                        ],
                        "Effect": "Deny"
                }
        ]
}
```

A deny policy must be used in conjunction with other policies to take effect. If the permissions assigned to a user contain both "Allow" and "Deny", the "Deny" permissions take precedence over the "Allow" permissions.

After authorization, users in the group can verify their permissions using the console or REST APIs.

The following uses the custom policy as an example to describe how to log in to the ServiceStage console to verify that a user is not allowed to create microservice engines.

1. Log in to the cloud service as an IAM user.
   - Tenant name: Name of the cloud service account used to create the IAM user
   - IAM username and password: Username and password specified during the IAM user creation using the tenant name
2. On the **Cloud Service Engines** page, create a microservice engine. If error 403 is returned, the permissions are correct and have taken effect.

4.3 Assigning Permissions to ServiceStage-Dependent Services

Assigning CCE Namespace Permissions

You can assign only common operation permissions on CCE cluster resources to the ServiceStage user group using IAM, excluding the namespace permissions of the clusters with Kubernetes RBAC authentication enabled. Therefore, assign the namespace permissions to the clusters separately.

For details, see **Permissions Management**.

Assigning CTS Permissions

After the permissions are assigned for ServiceStage using IAM, they do not take effect for the CTS service on which ServiceStage depends. Therefore, assign the CTS service permissions separately.

5 Environment Management

5.1 Environment Overview

An environment is a collection of compute, network, and middleware resources used for deploying and running a component. ServiceStage combines the compute resources (such as CCE clusters), network resources (such as ELB instances and EIPs), and middleware (such as DCS instances, RDS instances, and CSE engines) into an environment, such as a development environment, testing environment, pre-production environment, or production environment.

The resources within an environment can be networked together. Managing resources and deploying services by environment simplifies O&M.

A maximum of 300 environments can be created in a project.

5.2 Creating an Environment

Create an environment before deploying components.

Procedure

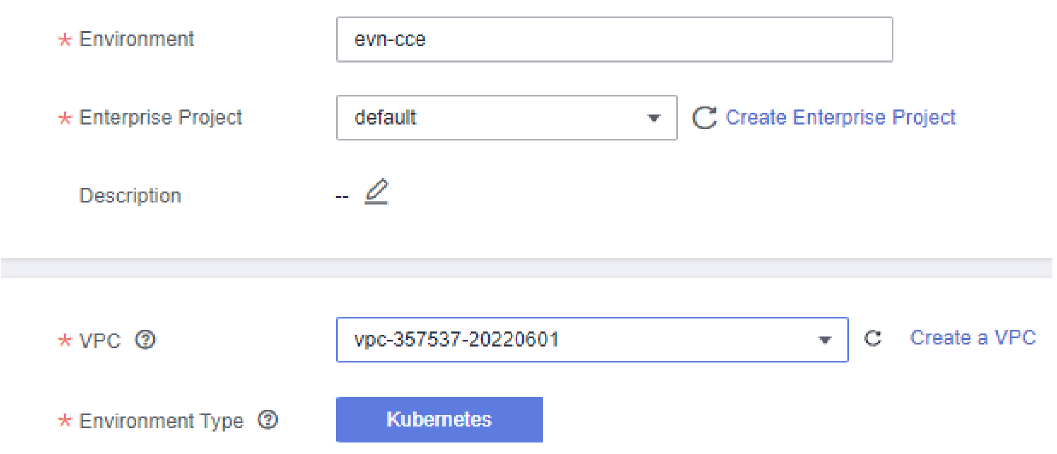
Step 1 Log in to ServiceStage.

Step 2 Choose **Environment Management** > **Create Environment** and configure the environment. Parameters marked with an asterisk (*) are mandatory.

Parameter	Description
*Environment	Environment name.
*Enterprise Project	Enterprise project. Enterprise projects let you manage cloud resources and users by project. It is available after you **create an enterprise project**.

Parameter	Description
Description	Environment description. 1. Click ✎ and enter the environment description. 2. Click ✓ to save the description.
*VPC	VPC where the environment resources are located. For details about how to create a VPC, see **Creating a VPC**. **NOTE** After the environment is created, the VPC cannot be modified during **5.6 Modifying an Environment**.
*Environment Type	Environment type. **Kubernetes**: applicable to container-based deployment (CCE). Components are deployed using container images and scheduled by Kubernetes. **NOTE** For details about the component deployment mode, see **Deploying a Component**.

Figure 5-1 Configuring an environment



* Environment	evn-cce
* Enterprise Project	default
Description	--
* VPC	vpc-357537-20220601
* Environment Type	Kubernetes

Step 3 Click **Create Now**.

After the environment is created, go to the environment details page to view the environment details and configure environment resources.

NOTE

If CCE clusters and VMs are managed in an earlier version, the environment type is **VM + Kubernetes** after the upgrade to the current version.

----End

Follow-Up Operations

- After a Kubernetes environment is created, bind a CCE cluster to the environment before using the environment to deploy components. For details, see **5.3 CCE Resource Management**.
- After an environment is created, the compute resources (excluding CCE clusters), network resources, and middleware need to be managed together to form an environment. For details, see **5.4 Managing Resources**.

5.3 CCE Resource Management

5.3.1 Binding a CCE Cluster

After a Kubernetes environment is created, bind only one CCE cluster to the environment before using the environment to deploy components.

Prerequisites

A CCE cluster has been created. The cluster must be in the same VPC as the target environment and cannot be managed by other environments.

For details about how to create a CCE cluster, see **Creating a CCE Cluster**.

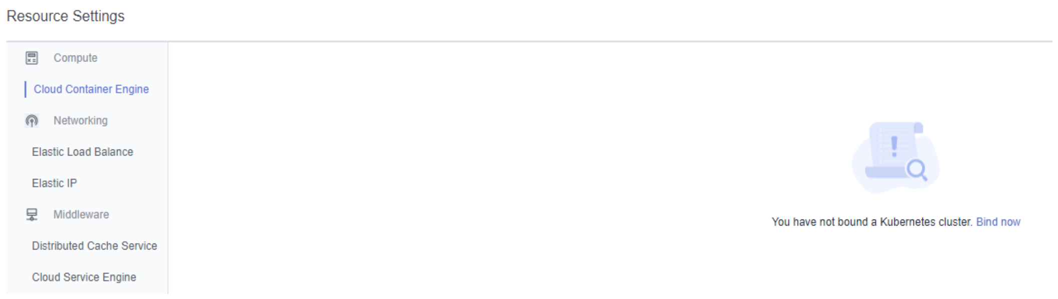
Procedure

Step 1 Log in to ServiceStage.

Step 2 On the **Environment Management** page, click the target environment.

Step 3 In the **Resource Settings** area, choose **Cloud Container Engine** from **Compute**.

Figure 5-2 Binding a CCE cluster



You have not bound a Kubernetes cluster. Bind now

Step 4 Click **Bind now**.

- If the CCE cluster to be bound has been created, select the cluster from the cluster drop-down list and click **OK**.
- If no CCE cluster is created, go to the CCE console as prompted, create a CCE cluster by referring to **Prerequisites**, and bind the CCE cluster.

NOTE

In a Kubernetes environment, if IPv6 is enabled for the selected VPC and CCE clusters are managed, select a CCE cluster with IPv6 enabled. Otherwise, the Java chassis microservice registered on the exclusive microservice engine with security authentication enabled in the VPC fails to register the discovery address using IPv6.

----End

Follow-Up Operations

- Click the **Node List** tab to view details about each node in the CCE cluster.
- Click **View Resource Details** to view CCE cluster details on the CCE console.

5.3.2 Unbinding a CCE Cluster

If a CCE cluster that has been bound to a Kubernetes environment is no longer used, you can remove it from the environment.

Prerequisites

A CCE cluster has been bound to the environment. For details, see **5.3.1 Binding a CCE Cluster**.

Procedure

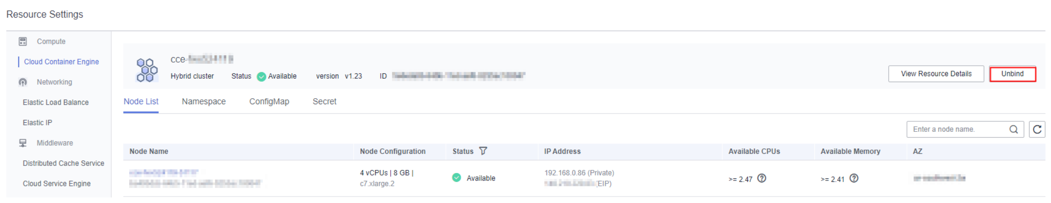
Step 1 Log in to ServiceStage.

Step 2 On the **Environment Management** page, click the target environment.

Step 3 In the **Resource Settings** area, choose **Cloud Container Engine** from **Compute**.

Step 4 Click **Unbind** to remove the CCE cluster from the environment.

Figure 5-3 Unbinding a CCE cluster



cce-[redacted]
Hybrid cluster Status Available version v1.23 ID [redacted]

Node Name	Node Configuration	Status	IP Address	Available CPUs	Available Memory	AZ
[redacted]	4 vCPUs | 8 GB | c7.xlarge.2	Available	192.168.0.86 (Private) [redacted] (EIP)	>= 2.47	>= 2.41	[redacted]

----End

5.3.3 Managing Namespaces

A namespace is a collection of resources and objects. Multiple namespaces can be created in a single CCE cluster with the data isolated from each other. This enables namespaces to share the services of the same cluster without affecting each other.

For example, you can deploy workloads in a development environment into one namespace, and deploy workloads in a testing environment into another namespace.

Table 5-1 describes the namespace types.

Table 5-1 Namespace types

Creation Type	Description
Created by a cluster by default	When a cluster is started, the **default**, **kube-public**, **kube-system**, and **kube-node-lease** namespaces are created by default. ● **default**: All objects for which no namespace is specified are allocated to this namespace. ● **kube-public**: Resources in this namespace can be accessed by all users (including unauthenticated users), such as public add-ons and container charts. ● **kube-system**: All resources created by Kubernetes are in this namespace. ● **kube-node-lease**: Each node has an associated Lease object in this namespace. The object is periodically updated by the node.
Created manually	You can create namespaces to serve separate purposes. For example, you can create three namespaces, one for a development environment, one for joint debugging environment, and one for testing environment. You can also create one namespace for login services and one for game services.

This section describes how to create and delete a namespace, and manage namespace resource quotas.

Prerequisites

A CCE cluster has been bound to the environment. For details, see **5.3.1 Binding a CCE Cluster**.

Creating a Namespace

Step 1 Log in to ServiceStage.

Step 2 On the **Environment Management** page, click the target environment.

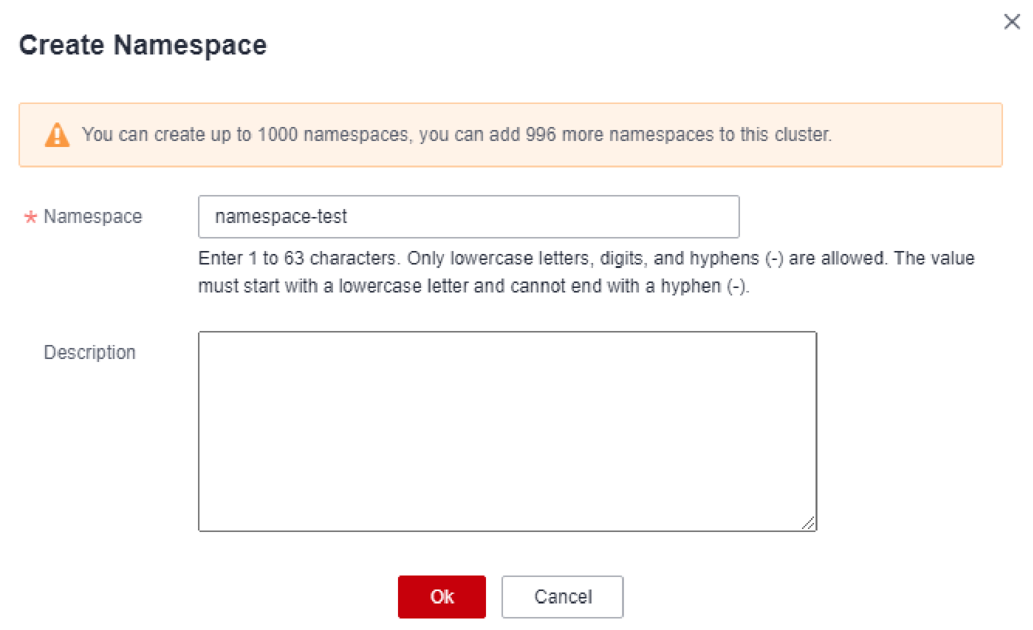
Step 3 In the **Resource Settings** area, choose **Cloud Container Engine** from **Compute**.

Step 4 Click **Namespace** > **Create Namespace**.

Step 5 Set the parameters by referring to the following table. Parameters marked with an asterisk (*) are mandatory.

Parameter	Description
*Namespace	Namespace name.
Namespace description	Description about the namespace.

Figure 5-4 Setting namespace parameters



Create Namespace

You can create up to 1000 namespaces, you can add 996 more namespaces to this cluster.

* Namespace: namespace-test

Enter 1 to 63 characters. Only lowercase letters, digits, and hyphens (-) are allowed. The value must start with a lowercase letter and cannot end with a hyphen (-).

Description

Step 6 Click **OK**.

The created namespace is displayed in the namespace list.

----End

Deleting a Namespace

NOTICE

- If a namespace is deleted, all resources (such as workloads and configuration items) in this namespace will be also deleted. Exercise caution when deleting a namespace.
- The cluster-created namespace **default** cannot be deleted.

Step 1 Log in to ServiceStage.

Step 2 On the **Environment Management** page, click the target environment.

Step 3 In the **Resource Settings** area, choose **Cloud Container Engine** from **Compute**.

Step 4 Click the **Namespace** tab.

- To delete a single namespace, locate the target namespace and click **Delete** in the **Operation** column.
- To delete namespaces in batches, select the target namespaces and click **Delete** above the namespaces.

Step 5 In the displayed dialog box, enter **DELETE** and click **OK**.

----End

Managing Namespace Resource Quotas

By default, pods running in a CCE cluster can use the CPUs and memory of a node without restrictions. This means that pods in a namespace may exhaust all resources of the cluster.

Kubernetes provides namespaces for you to group workloads in a cluster. By setting resource quotas for each namespace, you can prevent resource exhaustion and ensure cluster reliability. You can configure quotas for resources such as CPU, memory, and the number of pods in a namespace. For more information, see **Resource Quotas**.

User-created namespaces and the cluster-created namespace **default** support resource quota management.

Step 1 Log in to ServiceStage.

Step 2 On the **Environment Management** page, click the target environment.

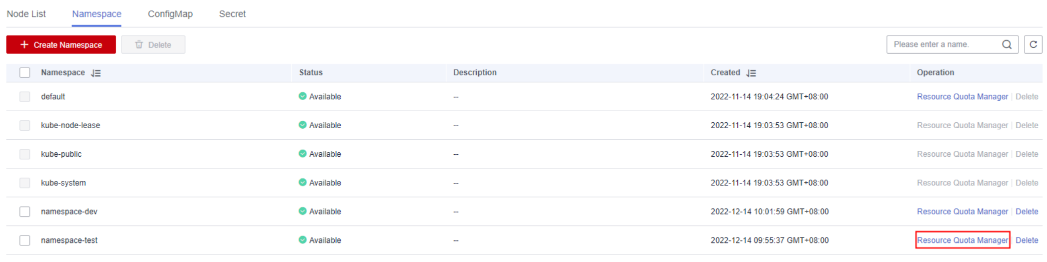
Step 3 In the **Resource Settings** area, choose **Cloud Container Engine** from **Compute**.

Step 4 Click the **Namespace** tab.

Step 5 Locate the target namespace and click **Resource Quota Manager** in the **Operation** column.

In the displayed **Resource Quota Manager** dialog box, view the resource types, total resource quotas, and accumulated quota usage in the namespace.

Figure 5-5 Accessing the Resource Quota Manager page



Namespace	Status	Description	Created	Operation
default	Available	--	2022-11-14 19:04:24 GMT+08:00	Resource Quota Manager | Delete
kube-node-lease	Available	--	2022-11-14 19:03:53 GMT+08:00	Resource Quota Manager | Delete
kube-public	Available	--	2022-11-14 19:03:53 GMT+08:00	Resource Quota Manager | Delete
kube-system	Available	--	2022-11-14 19:03:53 GMT+08:00	Resource Quota Manager | Delete
namespace-dev	Available	--	2022-12-14 10:01:59 GMT+08:00	Resource Quota Manager | Delete
namespace-test	Available	--	2022-12-14 09:55:37 GMT+08:00	Resource Quota Manager | Delete

Step 6 Click **Edit Quota** and set the total quota of each resource type.

- If the usage of a resource type is not limited, enter **0**.
- If the usage of a resource type is limited, enter the expected integer.

NOTICE

– Accumulated quota usage includes the resources used by CCE to create default components, such as the Kubernetes Services (which can be viewed using kubectl) created under the **default** namespace. Therefore, you are advised to set a resource quota greater than expected to reserve resource for creating default components.
– If the total CPU or memory quota in a namespace is limited, you must set the maximum and minimum number of CPU cores and memory (GiB) that can be used by the component when setting resources for the Kubernetes component of this namespace in **7.2 Creating and Deploying a Component** and **7.6 Upgrading a Single Component**. Otherwise, the operation will fail.
– If the total quota of other resource types in a namespace is limited and the remaining usage of this resource type does not meet requirements, the Kubernetes component of this namespace will fail to be deployed.

Step 7 Click **OK**.

----End

5.3.4 Managing Configuration Items

Configuration items (ConfigMaps) are user-defined resources that store application configurations. They can be used as files or environment variables in applications.

Configuration items allow you to decouple configuration files from images to enhance the portability of applications.

Benefits of configuration items:

- Manage configurations of different environments and services.
- Deploy applications in different environments. You can maintain configuration files in multiple versions, which makes it easy to update and roll back applications.
- Quickly import configurations in the form of files to containers.

This section describes how to create, delete, view, and update configuration items.

Prerequisites

1. A CCE cluster has been bound to the environment. For details, see **5.3.1 Binding a CCE Cluster**.
2. The namespace to which the configuration item belongs has been created. For details, see **Creating a Namespace**.

Creating a Configuration Item

Step 1 Log in to ServiceStage.

Step 2 On the **Environment Management** page, click the target environment.

Step 3 In the **Resource Settings** area, choose **Cloud Container Engine** from **Compute**.

Step 4 Click **Configuration Item** > **Create Configuration Item**.

ServiceStage allows you to create configuration items in **Visualization** or **YAML** mode.

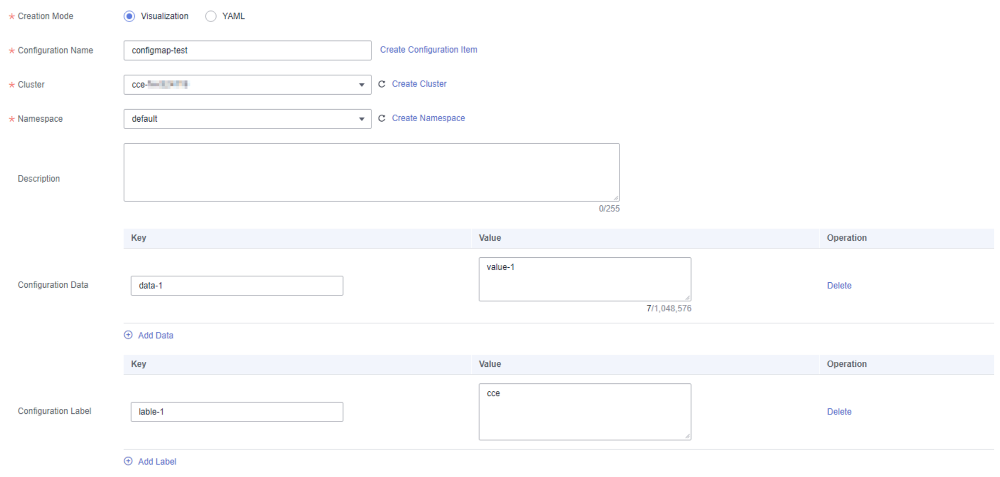
- Method 1: Visualization

  Configure the configuration item by referring to **Table 5-2**. Parameters marked with an asterisk (*) are mandatory.

Table 5-2 Parameters for creating a configuration item in visualization mode

Parameter	Description
*Configuration Name	Name of the configuration item, which must be unique in a namespace.
*Cluster	Cluster where the configuration item will be used.
*Namespace	Namespace to which the configuration item belongs. The default value is **default**.
Description	Description of the configuration item.
Configuration Data	Configuration data to be used in applications or used to store configuration data. **Key** indicates a file name, and **Value** indicates the content in the file. 1. Click **Add Data**. 2. Enter the key and value.
Configuration Labels	Labels that you want to attach to various objects (such as applications, nodes, and services) in the form of key-value pairs. Labels define the identifiable attributes of these objects and are used to manage and select the objects. 1. Click **Add Label**. 2. Enter the key and value.

Figure 5-6 Setting configuration item parameters in Visualization mode



* Creation Mode: [✓ Visualization] YAML

* Configuration Name: configmap-test

* Cluster: cce-[redacted]

* Namespace: default

Description: 0/255

Configuration Data

Key	Value	Operation
data-1	value-1 (7/1,048,576)	Delete

Configuration Label

Key	Value	Operation
lable-1	cce	Delete

- Method 2: YAML

  NOTE

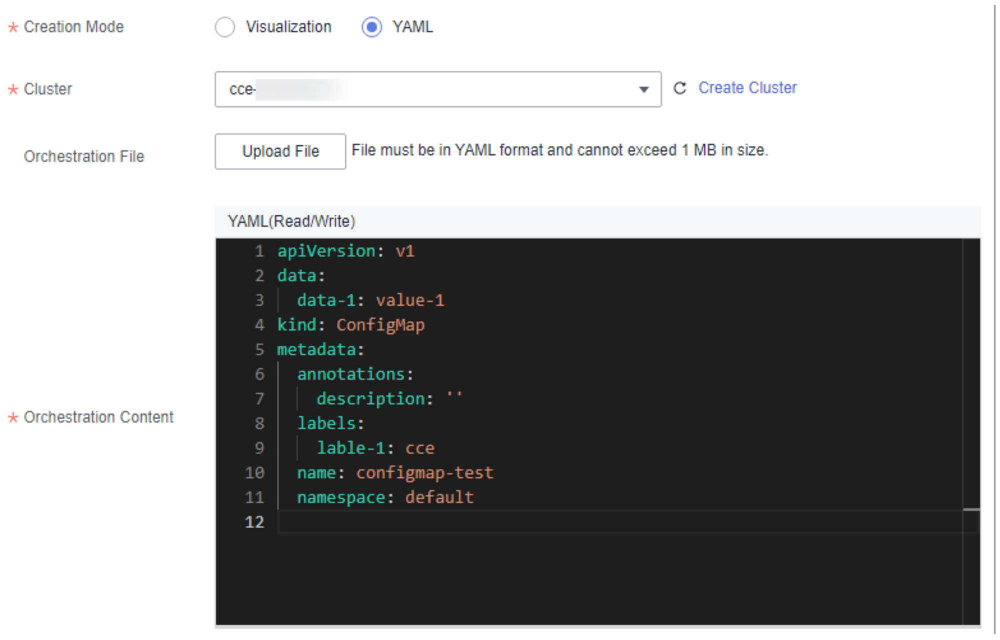
  To create a configuration item by uploading a file, ensure that a configuration item description file has been created. ServiceStage supports files in YAML format. For details, see **Configuration Item Requirements**.

  a. Select a cluster from the **Cluster** drop-down list.

  b. (Optional) Click **Upload File** and select the ConfigMap resource file created locally. Click **Open** and wait until the upload is successful.

  c. Write or modify the ConfigMap resource file in **Orchestration content**.

Figure 5-7 Setting configuration item parameters in YAML mode



* Creation Mode: Visualization [✓ YAML]

* Cluster: cce-[redacted]

Orchestration File: File must be in YAML format and cannot exceed 1 MB in size.

* Orchestration Content

YAML(Read/Write)

```yaml
apiVersion: v1
data:
  data-1: value-1
kind: ConfigMap
metadata:
  annotations:
    description: ''
  labels:
    lable-1: cce
  name: configmap-test
  namespace: default
```

Step 5 Click **Create Configuration Item**.

After the configuration item is created, it is displayed in the configuration item list.

----End

Follow-Up Operations

After a configuration item is created, you can search for, view, update, and delete the configuration item by referring to **Table 5-3**.

NOTE

- Deleted items cannot be restored. Exercise caution when performing this operation.
- The configuration item list contains system configuration items, which can only be viewed and cannot be modified or deleted.

Table 5-3 Configuration item management operations

Operation	Description
Searching for a configuration item	1. Select the namespace to which the configuration item belongs from the namespace drop-down list. 2. Enter a configuration item name in the search box.
Viewing a configuration item	Click **Show YAML** in the **Operation** column of the target configuration item to view the content of the YAML file of the configuration item.
Modifying a configuration item	1. Click **Update** in the **Operation** column of the target configuration item. 2. Modify the information according to **Table 5-2**. 3. Click **Update Configuration Item**.
Deleting a configuration item	1. Click **Delete** in the **Operation** column of the target configuration item. 2. In the displayed dialog box, click **OK**.
Deleting configuration items in batches	1. Select the configuration items to be deleted. 2. Click **Delete Configuration Item**. 3. In the displayed dialog box, click **OK**.

Configuration Item Requirements

A configuration item resource file should be in YAML format, and the file size cannot exceed 1 MB.

Example:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: configmap-test
data:
  data-1: value-1
  data-2: value-2
```

5.3.5 Managing Secrets

Secrets are user-defined resources that store authentication and sensitive information such as application keys. They can be used as files or environment variables in applications.

This section describes how to create, delete, view, and update secrets.

Prerequisites

1. A CCE cluster has been bound to the environment. For details, see **5.3.1 Binding a CCE Cluster**.
2. The namespace to which the secret belongs has been created. For details, see **Creating a Namespace**.

Creating a Secret

Step 1 Log in to ServiceStage.

Step 2 On the **Environment Management** page, click the target environment.

Step 3 In the **Resource Settings** area, choose **Cloud Container Engine** from **Compute**.

Step 4 On the **Secret** page, click **Create Secret**.

ServiceStage allows you to create secrets in **Visualization** or **YAML** mode.

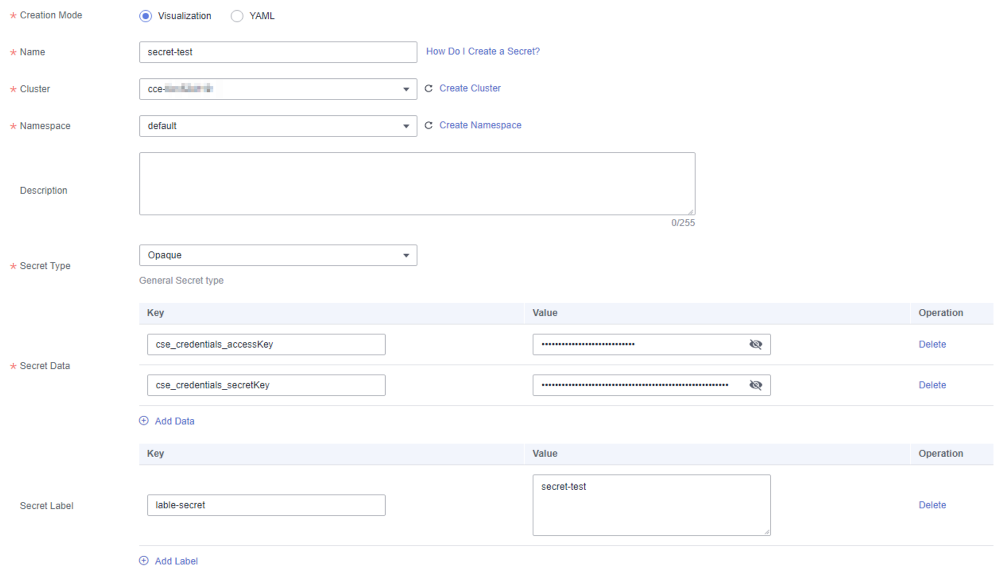
- Method 1: Visualization Configure the parameters by referring to **Table 5-4**. Parameters marked with an asterisk (*) are mandatory.

Table 5-4 Parameters for creating a secret in visualization mode

Parameter	Description
*Name	Name of a secret, which must be unique in the same namespace.
*Cluster	Cluster where the secret will be used. Click **Create Cluster** to create a cluster.
*Namespace	Namespace to which the secret belongs. If you do not specify this parameter, the value **default** is used by default.
Description	Description of a secret.

Parameter	Description
*Secret Type	Select the type of the secret to be created based on service requirements. – **Opaque**: general secret type. If the secret type is not explicitly set in the secret configuration file, the default secret type is **Opaque**. – **kubernetes.io/dockerconfigjson**: a secret that stores the authentication information required for pulling images from a private repository. – **IngressTLS**: a secret that stores the certificate required by ingresses (layer-7 load balancing services). – **Other**: Enter a secret type that is none of the above.
*Repository Address	This parameter is valid only when **Secret Type** is set to **kubernetes.io/dockerconfigjson**. Enter the address of the image repository.
*Secret data	Value of the **data** field in the application secret file. – If the secret type is **Opaque**, enter the key and value. The value must be encoded using Base64. For more information, see **Base64 Encoding**. Click **Add Data** to add secret data. – If the secret type is **kubernetes.io/dockerconfigjson**, enter the username and password. – If the secret type is **IngressTLS**, upload the certificate file and private key file. – If the secret type is **Other**, enter the key and value.
Secret Label	Labels that you want to attach to various objects (such as applications, nodes, and services) in the form of key-value pairs. Labels define the identifiable attributes of these objects and are used to manage and select the objects. 1. Click **Add Label**. 2. Enter the key and value.

Figure 5-8 Setting secret parameters in Visualization mode



* Creation Mode	[✓ Visualization] YAML
* Name	secret-test
* Cluster	cce-[redacted]
* Namespace	default
Description	0/255
* Secret Type	Opaque (General Secret type)

* Secret Data

Key	Value	Operation
cse_credentials_accessKey	••••••••••••••••••••••••••	Delete
cse_credentials_secretKey	••	Delete

Add Data

Secret Label

Key	Value	Operation
lable-secret	secret-test	Delete

Add Label

- Method 2: YAML

  NOTE

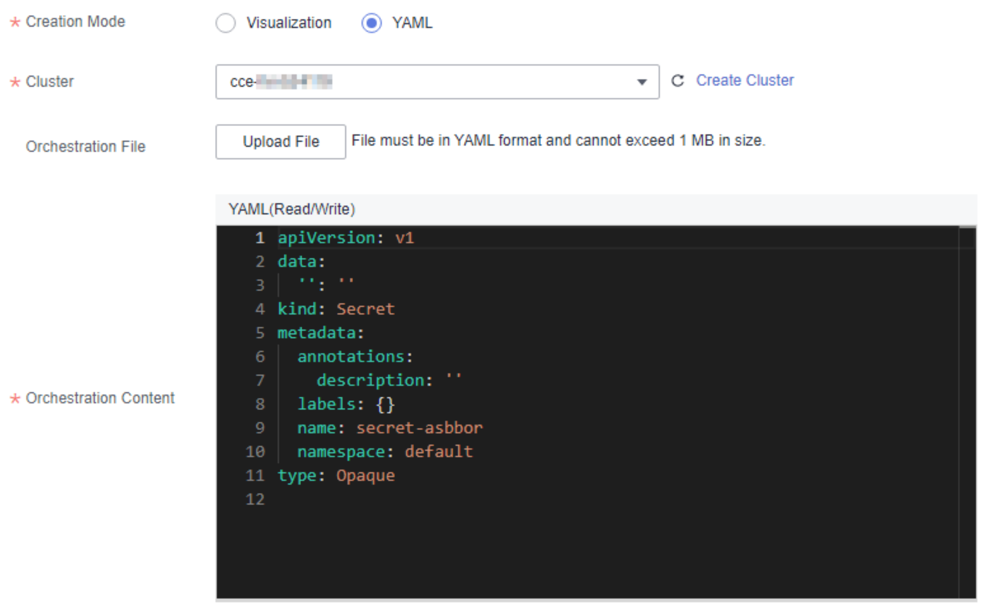
  To create a secret by uploading a file, ensure that the secret description file has been created. ServiceStage supports files in YAML format. For details, see **Secret Resource File Configuration**.

  a. Select a cluster from the **Cluster** drop-down list.

  b. (Optional) Click **Upload File**, select the created secret file, and click **Open**. Wait until the file is uploaded.

  c. Write or modify the secret resource file in **Orchestration content**.

Figure 5-9 Setting secret parameters in YAML mode



* Creation Mode	Visualization [✓ YAML]
* Cluster	cce-[redacted]
Orchestration File	Upload File — File must be in YAML format and cannot exceed 1 MB in size.
* Orchestration Content	YAML(Read/Write)

```
apiVersion: v1
data:
  '': ''
kind: Secret
metadata:
  annotations:
    description: ''
  labels: {}
  name: secret-asbbor
  namespace: default
type: Opaque

```

Step 5 Click **Create Secret**.

The new secret is displayed in the secret list.

----End

Follow-Up Operations

After a secret is created, you can search for, view, update, and delete the secret by referring to **Table 5-5**.

NOTE

- Deleted items cannot be restored. Exercise caution when performing this operation.
- The secret list contains system secrets, which can only be viewed and cannot be modified or deleted.

Table 5-5 Secret management operations

Operation	Description
Searching for a secret	1. Select the namespace to which the secret belongs from the namespace drop-down list. 2. Enter a secret name in the search box.
Viewing a secret	Click **Show YAML** in the **Operation** column of the target secret to view the content of the YAML file of the secret.
Updating a secret	1. Click **Update** in the **Operation** column of the target secret. 2. Modify the information according to **Table 5-4**. 3. Click **Modify Secret**.
Deleting a secret	1. Click **Delete** in the **Operation** column of the target secret. 2. In the displayed dialog box, click **OK**.
Deleting secrets in batches	1. Select the secrets to be deleted. 2. Click **Delete Key**. 3. In the displayed dialog box, click **OK**.

Secret Resource File Configuration

This section provides examples of configuring secret resource description files. For example, you can retrieve the username and password for an application through a secret.

username: my-username

password: my-password

The following shows the content of a secret file. The value must be encoded using Base64. For more information, see **Base64 Encoding**.

```
apiVersion: v1
kind: Secret
metadata:
  name: mysecret            #Secret name.
  namespace: default       #Namespace. The default value is default.
data:
  username: ******    #The value must be encoded using Base64.
  password: ******  #The value must be Base64-encoded.
type: Opaque     #You are advised not to change this parameter value.
```

Base64 Encoding

To encrypt a string using Base64, run the **echo -n '***Content to be encoded***' | base64** command in the local Linux environment. Example:

```
root@ubuntu:~# echo -n '3306' | base64
MzMwNg==
```

Where,

- **3306** is the content to be encoded.
- **MzMwNg==** is the encoded content.

5.4 Managing Resources

After an environment is created, the compute resources (such as CCE clusters), network resources (such as ELB instances and EIPs), and middleware (such as DCS instances, RDS instances, and CSE engines) need to be managed together to form an environment.

For details about how to manage CCE cluster resources in the Kubernetes environment, see **5.3 CCE Resource Management**.

Prerequisites

- The ELBs to be managed have been created. The ELBs must be in the same VPC as the environment.

  For details, see **Creating a Dedicated Load Balancer**.
- The EIPs to be managed have been created.

  For details, see **Assigning an EIP**.
- The DCSs to be managed have been created. The DCSs must be in the same VPC as the environment.

  For details, see **Creating an Instance**.
- The RDSs to be managed have been created. The RDSs must be in the same VPC as the environment.

  For details, see **Create a DB Instance**.
- The CSEs to be managed have been created. If the CSEs and the environment are in different VPC, correctly configure the VPC connectivity.

  For details, see **10.2 Creating a Microservice Engine**.

Procedure

Step 1 Log in to ServiceStage.

Step 2 On the **Environment Management** page, click the target environment.

Step 3 In the **Resource Settings** area, choose the resources from **Compute**, **Networking**, or **Middleware**, and click **Manage Resource**.

Step 4 In the dialog box that is displayed, select the resources to be managed and click **OK**.

NOTE

- In the same VPC, CCEs that have been managed by other environments cannot be managed again.

----End

5.5 Removing Managed Resources

You can remove a managed resource that is no longer used.

Procedure

Step 1 Log in to ServiceStage.

Step 2 On the **Environment Management** page, click the target environment.

Step 3 In the **Resource Settings** area, choose the resources from **Compute**, **Networking**, or **Middleware**.

Step 4 In the managed resource list, perform the following operations:

- To remove resources in batches, select the resources to be deleted and click **Remove Resource**.
- To remove a single resource, locate the resource to be removed and click **Remove** in the **Operation** column.

----End

5.6 Modifying an Environment

This topic describes how to modify an environment.

Procedure

Step 1 Log in to ServiceStage.

Step 2 On the **Environment Management** page, use either of the following methods to go to the **Edit Environment** page:

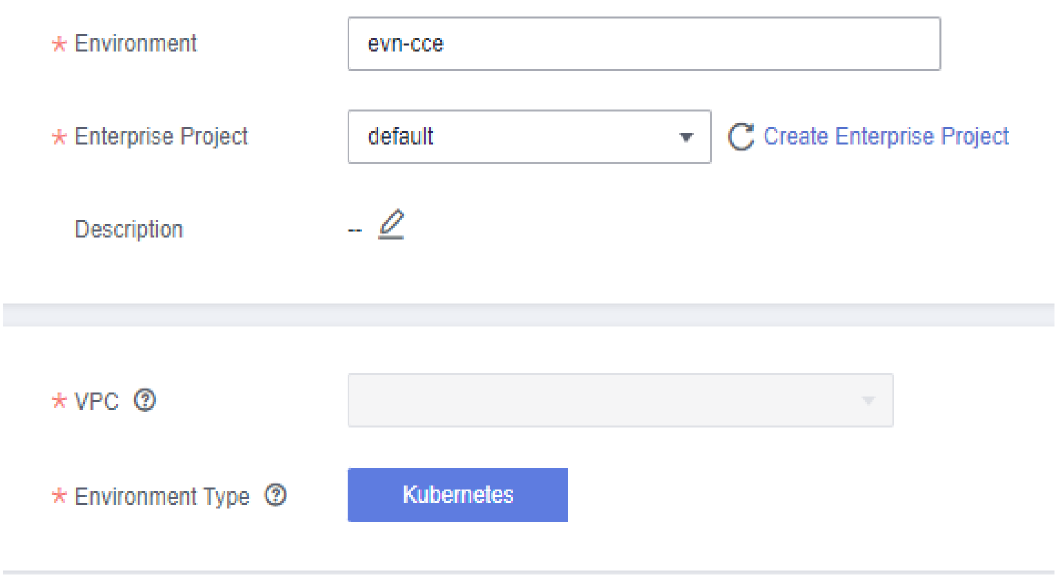
- Select the target environment and click **Edit** in the **Operation** column.
- Click the target environment. On the displayed environment details page, click **Edit**.

Step 3 Edit the environment information by referring to the following table.

Parameter	Description
Environment	Environment name.

Parameter	Description
*Enterprise Project	Enterprise projects let you manage cloud resources and users by project. It is available after you **create an enterprise project**.
Description	Environment description. 1. Click ✎ and enter the environment description. 2. Click ✓ to save the description.

Figure 5-10 Editing an environment


* Environment	evn-cce
* Enterprise Project	default
Description	--
* VPC	
* Environment Type	Kubernetes

Step 4 Click **Save**.

----End

5.7 Deleting an Environment

You can delete an environment that is no longer used.

NOTE

- Before deleting an environment, ensure that no component is deployed in the environment or the deployed components have been deleted. For details, see **7.16 Deleting a Component**.
- Deleting an environment does not delete managed resources in the environment.

Procedure

Step 1 Log in to ServiceStage.

Step 2 On the **Environment Management** page, use either of the following methods to delete an environment:

- Select the target environment and click **Delete** in the **Operation** column.
- Click the target environment. On the displayed environment details page, click **Delete**.

Step 3 Click **OK**.

----End

6 Application Management

6.1 Creating an Application

An application is a service system with complete functions and consists of one or more components related to features.

For example, the weather forecast is an application that contains the weather and forecast components. ServiceStage organizes multiple components by application, and supports quick cloning of applications in different environments.

ServiceStage allows a single user to create a maximum of 1000 applications under the same project.

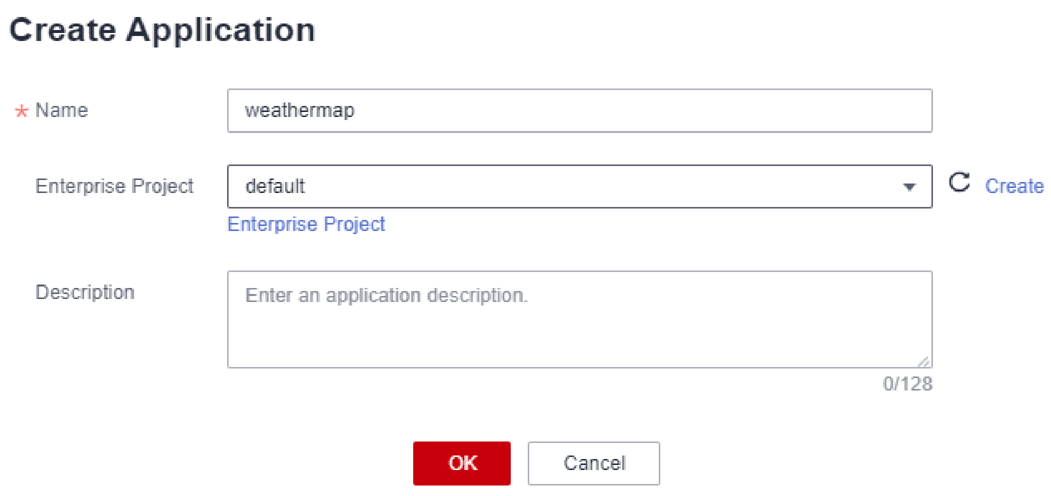
Procedure

Step 1 Log in to ServiceStage.

Step 2 Choose **Application Management** > **Create Application** and configure the application. Parameters marked with an asterisk (*) are mandatory.

Parameter	Description
*Name	Enter an application name. The application name must be unique.
Enterprise Project	Select an enterprise project. Enterprise projects let you manage cloud resources and users by project. It is available after you **create an enterprise project**.
Description	Enter the application description.

Figure 6-1 Creating an application



Create Application

* Name	weathermap
Enterprise Project	default
Description	Enter an application description.

Step 3 Click **OK**.

----End

6.2 Viewing Application Overview

After the application is created, you can go to the **Overview** page to view the application overview.

Procedure

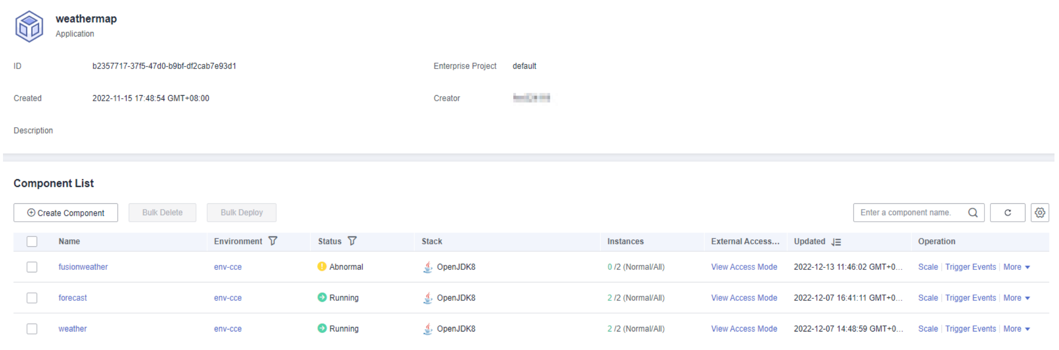
Step 1 Log in to ServiceStage.

Step 2 Choose **Application Management**.

Step 3 Click a target application. On the displayed **Overview** page, view the application details.

If a component has been created and deployed under the application, you can view the component details in the component list.

Figure 6-2 Viewing application overview



weathermap
Application

ID	b2357717-37f5-47d0-b9bf-df2cab7e93d1	Enterprise Project	default
Created	2022-11-15 17:48:54 GMT+08:00	Creator	[redacted]
Description			

Component List

Name	Environment	Status	Stack	Instances	External Access...	Updated	Operation
fusionweather	env-cce	Abnormal	OpenJDK8	0 /2 (Normal/All)	View Access Mode	2022-12-13 11:46:02 GMT+0...	Scale | Trigger Events | More
forecast	env-cce	Running	OpenJDK8	2 /2 (Normal/All)	View Access Mode	2022-12-07 16:41:11 GMT+0...	Scale | Trigger Events | More
weather	env-cce	Running	OpenJDK8	2 /2 (Normal/All)	View Access Mode	2022-12-07 14:48:59 GMT+0...	Scale | Trigger Events | More

----End

6.3 Managing Application Environment Variables

Environment variables are parameters set in the system or user applications. You can obtain the values of environment variables by calling APIs. During deployment, parameters are specified through environment variables instead of in the code, which makes the deployment flexible.

Environment variables added to an application are global environment variables and take effect for all components of the application.

For details about how to add environment variables to a specific component, see **7.17.1 Setting Component Environment Variables**.

Manually Adding an Application Environment Variable

Step 1 Log in to ServiceStage.

Step 2 Choose **Application Management**.

Step 3 Click the target application. The **Overview** page is displayed.

Step 4 In the navigation pane on the left, click **Environment Variables**.

Step 5 Select a created environment from the drop-down list.

Step 6 Click **Add Environment Variable** and set **Name** and **Variable/Variable Reference**.

Where,

- **Name** is the name of an application environment variable and must be unique.
- **Variable/Variable Reference** is the value of the application environment variable.

For example, set **Variable Name** to **User** and **Variable/Variable reference** to **admin**. That is, when the program code reads the **User** environment variable, **admin** is obtained. For example, you can start subprocesses as the admin user and read files as the admin user. The actual execution effect depends on the code.

Step 7 Click **Submit**.

----End

Importing the Application Environment Variable File

Step 1 Log in to ServiceStage.

Step 2 Choose **Application Management**.

Step 3 Click the target application. The **Overview** page is displayed.

Step 4 In the navigation pane on the left, click **Environment Variables**.

Step 5 Select a created environment from the **Environment** drop-down list.

Step 6 Click **Import** and select the environment variable file created locally.

The file to be imported must be a key-value pair mapping file in JSON or YAML format and in character string format. For example:

```
{"key1": "value1", "key2": "value2"}
```

Where,

- **key1** and **key2** are the names of application environment variables and must be unique.
- **value1** and **value2** are the values of application environment variables.

Step 7 Click **Submit**.

----End

Editing an Application Environment Variable

Step 1 Log in to ServiceStage.

Step 2 Choose **Application Management**.

Step 3 Click the target application. The **Overview** page is displayed.

Step 4 In the navigation pane on the left, click **Environment Variables**.

Step 5 Select a created environment from the **Environment** drop-down list.

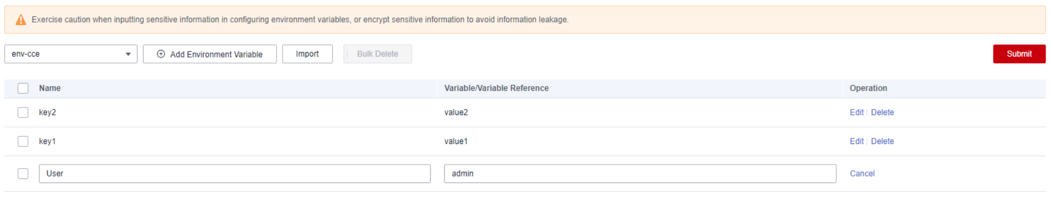
Step 6 Select the variable to be edited and click **Edit** in the **Operation** column.

Step 7 Reset **Variable Name** and **Variable/Variable Reference**.

- **Variable Name** is the name of an application environment variable and must be unique.
- **Variable/Variable Reference** is the value of the application environment variable.

Step 8 Click **Submit**.

Figure 6-3 Editing an application environment variable



Exercise caution when inputting sensitive information in configuring environment variables, or encrypt sensitive information to avoid information leakage.

Name	Variable/Variable Reference	Operation
key2	value2	Edit | Delete
key1	value1	Edit | Delete
User	admin	Cancel

----End

Deleting an Application Environment Variable

Step 1 Log in to ServiceStage.

Step 2 Choose **Application Management**.

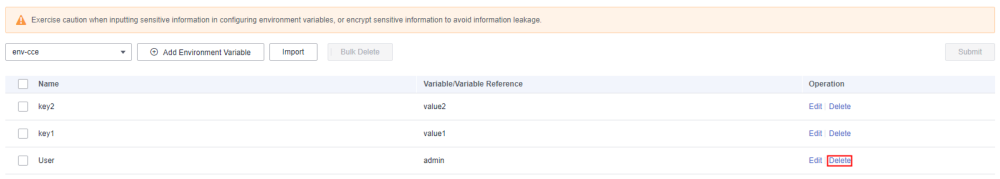
Step 3 Click the target application. The **Overview** page is displayed.

Step 4 In the navigation pane on the left, click **Environment Variables**.

Step 5 Select a created environment from the **Environment** drop-down list.

- To delete a single application environment variable, select the target variable and click **Delete** in the **Operation** column.

Figure 6-4 Deleting a single application environment variable

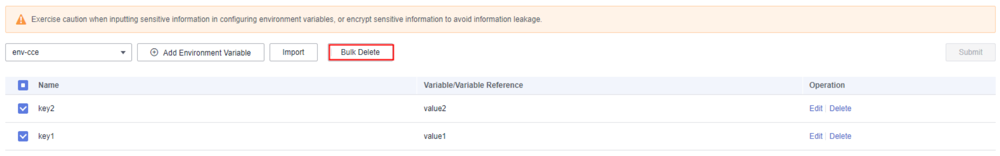


Exercise caution when inputting sensitive information in configuring environment variables, or encrypt sensitive information to avoid information leakage.

Name	Variable/Variable Reference	Operation
key2	value2	Edit | Delete
key1	value1	Edit | Delete
User	admin	Edit | Delete

- To delete application environment variables in batches, select the target variables and click **Bulk Delete**.

Figure 6-5 Deleting application environment variables in batches



Exercise caution when inputting sensitive information in configuring environment variables, or encrypt sensitive information to avoid information leakage.

Name	Variable/Variable Reference	Operation
✓ key2	value2	Edit | Delete
✓ key1	value1	Edit | Delete

Step 6 In the displayed dialog box, click **OK**.

----End

6.4 Editing an Application

You can modify the application name and description.

Procedure

Step 1 Log in to ServiceStage.

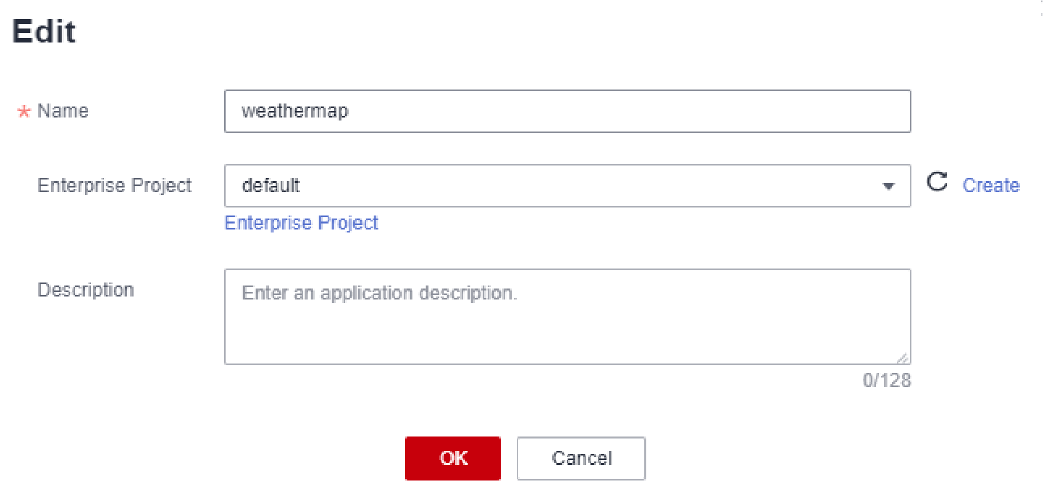
Step 2 Choose **Application Management**.

Step 3 Select the target application and click **Edit** in the **Operation** column.

Step 4 Configure the application again by referring to the following table.

Parameter	Description
Name	Enter an application name. The application name must be unique.
Enterprise Project	Select an enterprise project. Enterprise projects let you manage cloud resources and users by project. It is available after you **create an enterprise project**.
Description	Enter the application description.

Figure 6-6 Editing an application



Edit

* Name	weathermap
Enterprise Project	default
Description	Enter an application description. 0/128

Step 5 Click **OK**.

----End

6.5 Deleting an Application

You can delete an application that is no longer used.

NOTICE

Deleted applications cannot be restored. Exercise caution when performing this operation.

Prerequisites

Before deleting an application, delete all components of the application. For details, see **7.16 Deleting a Component**.

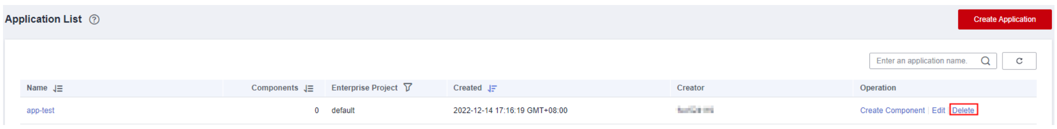
Procedure

Step 1 Log in to ServiceStage.

Step 2 Choose **Application Management**.

Step 3 Select the target application and click **Delete** in the **Operation** column.

Figure 6-7 Deleting an application



Application List

Name	Components	Enterprise Project	Created	Creator	Operation
app-test	0	default	2022-12-14 17:16:19 GMT+08:00	[redacted]	Create Component | Edit | Delete

Step 4 In the displayed dialog box, click **OK**.

----End

7 Component Management

7.1 Component Overview

A component is a service feature implementation of an application. It is carried by software packages and can be independently deployed and run.

After creating an application on ServiceStage, you can add components to the application. A maximum of 1000 components can be created for an application.

You can set the component technology stack and component source based on service requirements to create and deploy components.

Technology Stack

A technology stack includes the operating system, framework, and runtime on which component running depends. It consists of attributes such as the stack name, type, status, and version. The version number complies with the **semantic versioning specifications**.

ServiceStage provides and manages the stack lifecycle. You only need to focus on service development to improve application hosting experience.

The lifecycle phases of the technology stack are defined as follows:

- Preview: The beta version is released.
- General Availability (GA): The official version is released.
- End of Life (EOL): The lifecycle ends.

The technology stack status is defined as follows:

- Preview: The stack is in the Preview phase.
- Supported: The stack is in the GA phase.
- Deprecated: The stack is in the GA phase but the EOL announcement has been released, or the stack is not recommended by ServiceStage.

For details about the technology stack, see **Table 7-1**.

Table 7-1 Technology stack information

Technology Stack	Type	Status	Release Description	Component Source and Deployment Mode
OpenJDK8	Java	Supported	• OpenJDK-8u312 b07: **Release Note** • Image OS: EulerOS 2.9.8	The component source can be source code or JAR package, and container-based deployment is supported. For details, see **Deploying a Component**.
Tomcat8/ OpenJDK8	Tomcat	Supported	• OpenJDK-8u312 b07: **Release Note** • Tomcat-8.5.75: **Release Note** • Image OS: EulerOS 2.9.8	The component source can be source code or WAR package, and container-based deployment is supported. For details, see **Deploying a Component**.
Tomcat9/ OpenJDK8	Tomcat	Supported	• OpenJDK-8u312 b07: **Release Note** • Tomcat-9.0.58: **Release Note** • Image OS: EulerOS 2.9.8	The component source can be source code or WAR package, and container-based deployment is supported. For details, see **Deploying a Component**.
Node.js8	Node.js	Supported	• Nodejs-v8.11.3: **Release Note** • Image OS: EulerOS 2.9.8	The component source can be source code or ZIP package, and container-based deployment is supported. For details, see **Deploying a Component**.

Technology Stack	Type	Status	Release Description	Component Source and Deployment Mode
Node.js14	Node.js	Supported	• Nodejs-v14.18.1: **Release Note** • Image OS: EulerOS 2.9.8	The component source can be source code or ZIP package, and container-based deployment is supported. For details, see **Deploying a Component**.
Node.js16	Node.js	Supported	• Nodejs-v16.13.0: **Release Note** • Image OS: EulerOS 2.9.8	The component source can be source code or ZIP package, and container-based deployment is supported. For details, see **Deploying a Component**.
Docker	Docker	-	Supported by CCE.	The component source is from image package, and container-based deployment is supported. For details, see **Deploying a Component**.
Python3	Python	-	-	The component source can be source code or ZIP package, and container-based deployment is supported. For details, see **Deploying a Component**.

Technology Stack	Type	Status	Release Description	Component Source and Deployment Mode
Php7	Php	-	-	The component source can be source code or ZIP package, and container-based deployment is supported. For details, see **Deploying a Component**.

Component Source

Component Source	Description
Source Code Repository	Create authorization by referring to **9.7 Authorizing a Repository** and set the code source.
JAR package	The following upload modes are supported: Select the corresponding software package from OBS. Upload the software package to the OBS bucket in advance. For details, see **Uploading a File**.
WAR package	The following upload modes are supported: Select the corresponding software package from OBS. Upload the software package to the OBS bucket in advance. For details, see **Uploading a File**.
ZIP package	The following upload modes are supported: Select the corresponding software package from OBS. Upload the software package to the OBS bucket in advance. For details, see **Uploading a File**.

Component Source	Description
Image package	Containerized applications need to be created based on images. **My Images** (private images) and **Third-party Images** are supported. ● If you select **My Images**, upload the image to the image repository in advance. For details, see **8.1.1 Uploading an Image**. ● If you select **Third-party Images**, ensure that you have obtained the address of the third-party image. The format of the image address is as follows: *{IP address of the third-party image repository}:{Port number for accessing the third-party image repository}/{Image storage path}/{Image name}:{Image tag}* Alternatively: *{Image name}:{Image tag}* If the image tag is not specified, the latest version is used by default. Currently, only third-party public images can be obtained.

Deploying a Component

Deployment Mode	Description
Container-based deployment	Cloud Container Engine (CCE) deployment: CCE is a highly scalable, enterprise-class hosted Kubernetes service for you to run containers and applications. With CCE, you can easily deploy, manage, and scale containerized applications on the cloud platform.

7.2 Creating and Deploying a Component

This section describes how to create and deploy a component on ServiceStage.

ServiceStage allows you to create components with the same name in the same application. During component deployment, for components with the same name:

- Components with the same name in the same application cannot be deployed in the same environment.
- Components with the same name in different applications can be deployed in the same environment.

Prerequisites

1. An application has been created because components can only be added to applications. For details, see **6.1 Creating an Application**.
2. An environment has been created and resources have been managed because components need to be deployed in a specified environment. For details, see **5 Environment Management**.
3. An organization has been created because the image generated by the component deployment needs to be managed based on an organization. For details, see **Creating an Organization**.
4. (Optional) A namespace has been created, if you want to create and deploy a component in a Kubernetes environment. For details, see **Creating a Namespace**.
5. If you create a component based on a source code repository, create repository authorization first. For details, see **9.7 Authorizing a Repository**.
6. If you create a component based on a software package, the software package has been uploaded to the OBS bucket.

   Upload the software package to OBS. For details, see **Uploading a File**.

Procedure

Step 1 Log in to ServiceStage.

Step 2 Use any of the following methods to go to the **Create Component** page:

- Choose **Component Management** > **Create Component**.
- On the **Application Management** page, select the application for which you want to create a component, and click **Create Component** in the **Operation** column.
- On the **Application Management** page, click the application for which you want to create a component. On the displayed **Overview** page, click **Create Component**.

Step 3 In the **Basic Information** area, configure the component by referring to the following table. Parameters marked with an asterisk (*) are mandatory.

Parameter	Description
*Component Name	Name of a component.
*Component Version	Version number of a component. ● Automatically-generated: Click **Generate**. By default, the version number is the timestamp when you click **Generate**. The format is yyyy.mmdd.hhmms, where **s** is the ones place of the second in the timestamp. For example, if the timestamp is 2022.1214.172318, the version number is 2022.1214.17238. ● You can also customize the version number in the format of A.B.C, or A.B.C.D. A. B, C, and D are natural numbers, for example, 1.0.0 or 1.0.0.0.

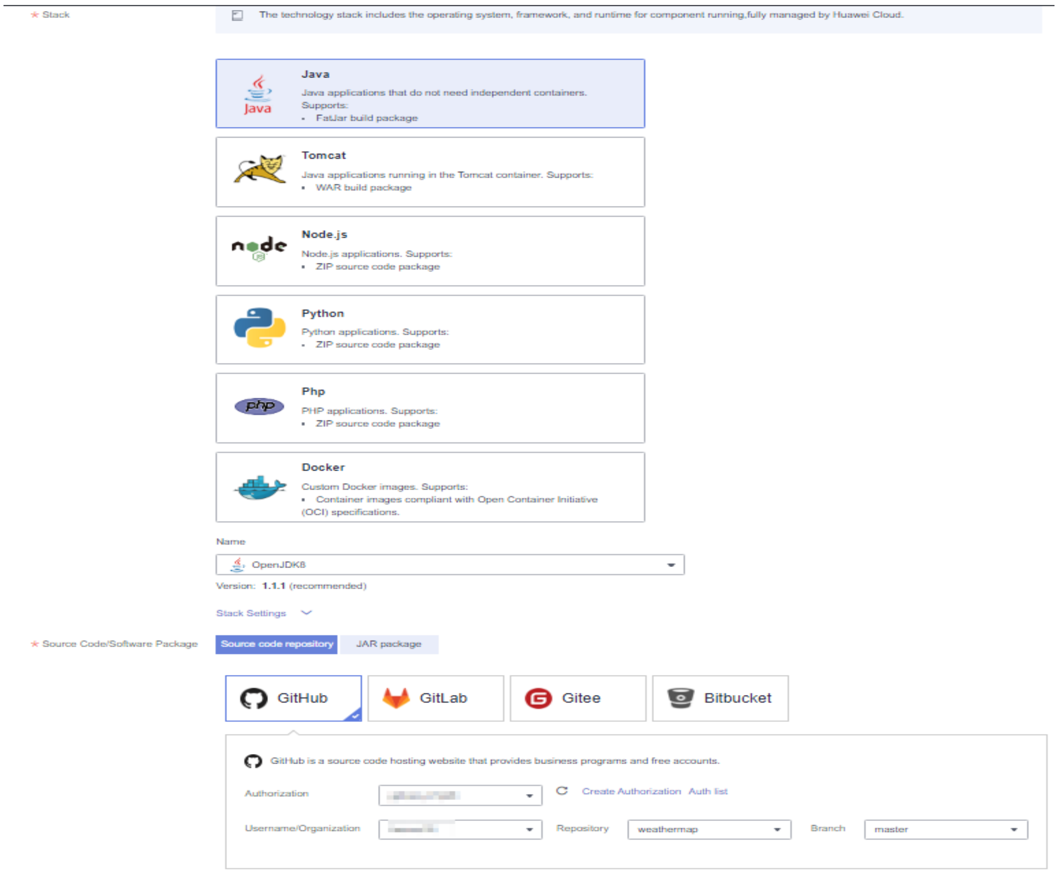
Parameter	Description
*Environment	Component deployment environment.
*Deployment Mode	Component deployment mode. This parameter is mandatory when **Environment** is set to **VM + Kubernetes**. **NOTE** If CCE clusters and VMs are managed in an earlier version, the environment type is **VM + Kubernetes** after the upgrade to the current version.
*Application	Application to which the component belongs.
Description	Component description. 1. Click ✎ to enter the component description. 2. Click ✓ to save the component description. Click ✗ to cancel the setting.

Step 4 In the **Component Package** area, configure the component package by referring to the following table. Parameters marked with an asterisk (*) are mandatory.

Parameter	Description
*Stack	1. Select a component technology stack type based on the component deployment mode. For details, see **Table 7-1**. 2. Select a technology stack from the **Name** drop-down list. 3. (Optional) Set **JVM** to configure the memory size during Java code running. This parameter is available when a Java or Tomcat technology stack is selected. Click **Stack Settings** and set **JVM**, for example, **-Xms256m -Xmx1024m**. Multiple parameters are separated by spaces. 4. (Optional) Set **Tomcat** parameters to configure the parameters such as Tomcat request path and port number. This parameter is available when a Tomcat technology stack is selected. a. Click **Stack Settings** and select **Tomcat**. The **Tomcat** dialog box is displayed. b. Click **Use Sample Code** and edit the template file based on service requirements. c. Click **OK**.
*Software Package	If you select **Source code repository**, create authorization by referring to **9.7 Authorizing a Repository** and set the code source. If you select a software package, the software package type supported by the component source is determined by the selected technology stack type. For details, see **Table 7-1**.

Parameter	Description
*Upload Method	If the component source is software package or image package, select an uploaded software package or image package. For details about the upload method, see **Component Source**.

Figure 7-1 Setting software package parameters



* Stack

The technology stack includes the operating system, framework, and runtime for component running,fully managed by Huawei Cloud.

Java
Java applications that do not need independent containers.
Supports:
- FatJar build package

Tomcat
Java applications running in the Tomcat container. Supports:
- WAR build package

Node.js
Node.js applications. Supports:
- ZIP source code package

Python
Python applications. Supports:
- ZIP source code package

Php
PHP applications. Supports:
- ZIP source code package

Docker
Custom Docker images. Supports:
- Container images compliant with Open Container Initiative (OCI) specifications.

Name: OpenJDK8
Version: **1.1.1** (recommended)

Stack Settings

* Source Code/Software Package: Source code repository | JAR package

GitHub | GitLab | Gitee | Bitbucket

GitHub is a source code hosting website that provides business programs and free accounts.

Authorization: [redacted] Create Authorization Auth list
Username/Organization: [redacted] Repository: weathermap Branch: master

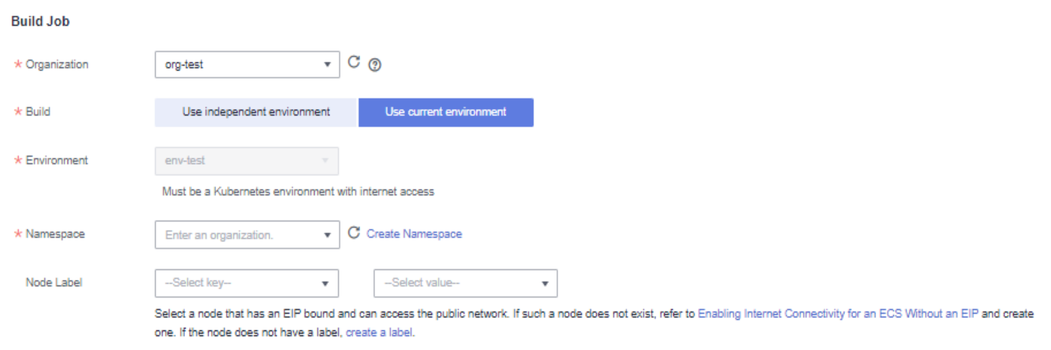
Step 5 In the **Build Job** area, set build parameters by referring to the following table. Parameters marked with an asterisk (*) are mandatory.

This area is mandatory when the component is deployed in the Kubernetes environment and the selected technology stack type is Java, Tomcat, Node.js, Python, or PHP.

Parameter	Description
*Command	If the component source is **Source code repository**, set **Command** based on service requirements. ● **Default command or script**: preferentially executes **build.sh** in the **root** directory. If **build.sh** does not exist, the code will be compiled using the common method of the selected language. Example: **mvn clean package** for Java. ● **Custom command**: Commands are customized using the selected language. Alternatively, the default command or script is used after **build.sh** is modified. **NOTICE** – If **Custom command** is selected, exercise caution when inputting sensitive information in the **echo**, **cat**, or **debug** command, or encrypt sensitive information to avoid information leakage. – To run the compilation command in the project subdirectory, you need to go to the project subdirectory and then run other script commands. For example: **cd ./weather/** **mvn clean package**
*Dockerfile Address	If the component source is **Source code repository**, set **Dockerfile Address** based on service requirements. **Dockerfile Address** is the directory where the Dockerfile is located relative to the root directory (./) of the project. The Dockerfile is used to build an image. If **Dockerfile Address** is not specified, the system searches for the Dockerfile in the root directory of the project by default. If the Dockerfile does not exist in the root directory, the system automatically generates the Dockerfile based on the selected operating environment.
*Organization	An organization is used to manage images generated during component build.

Parameter	Description
*Build	Select the type of the environment used to build an image. The build environment must be a Kubernetes environment, and can access the Internet. You are advised to select **Use current environment**. If the CCE cluster in the current environment cannot access the Internet and you have planned an independent build environment, you can select **Use independent environment**. ● Use independent environment: Use an independent build environment to build an image. The CPU architecture of the CCE cluster in the independent build environment must be the same as that of the CCE cluster in the current component deployment environment. Otherwise, the component deployment will fail. ● Use current environment: Use the deployment environment to which the component belongs to build an image. In the current environment, masters and nodes in the CCE cluster must have the same CPU architecture. Otherwise, the component build fails.
*Environment	● **Use independent environment**: Select an independent deployment environment different from that to which the component belongs. ● **Use current environment**: Select the deployment environment to which the component belongs.
*Namespace	Select the namespace of the container where the component instance is located.
Node Label	You can use a node label to deliver the build job to a fixed node bound with an EIP. For details about how to add a label, see **Managing Node Labels**.

Figure 7-2 Configuring build parameters


Build Job

* Organization: org-test

* Build: Use independent environment | [✓ Use current environment]

* Environment: env-test

Must be a Kubernetes environment with internet access

* Namespace: Enter an organization. Create Namespace

Node Label: --Select key-- --Select value--

Select a node that has an EIP bound and can access the public network. If such a node does not exist, refer to Enabling Internet Connectivity for an ECS Without an EIP and create one. If the node does not have a label, create a label.

Step 6 Click **Next**.

Step 7 In the **Resources** area, set the resources required by the component. Set the parameters by referring to the following table. Parameters marked with an asterisk (*) are mandatory.

Parameter	Description
*Resources	Components cannot be scheduled to nodes whose residual resources are fewer than the requested amount. For details about how to configure the request and limit parameters, see **Managing Resources for Containers**. You can customize **CPU** and **Memory** to set their quota, and set the maximum/minimum number of CPU cores and memory size (GiB) that can be used by components. Unselected parameters indicate no restriction.
*Instances	Set the number of component instances running in the environment. The value range is 1–200.
*Namespace	Select the namespace of the container where the component instance is located.

Step 8 (Optional) In the **Access Mode** area, enable **Public Network Access**.

Click ⬭ to enable public access and set the following parameters:

1. Set **Public Network Load Balancer**.
   - Select an Elastic Load Balance (ELB) resource that has been bound to an elastic IP address (EIP) in the selected environment.
   - If no ELB resource exists, click **Add One**. On the **Edit Environment** page that is displayed, click **Add Optional Resource** to add created ELB resources to the environment.
   - For details about how to create an ELB resource, see **Creating a Dedicated Load Balancer**.

   NOTE

   - An ELB needs to be bound to an EIP, and must be in the same VPC and subnet as the compute resources managed in the current component deployment environment.
   - Components must be bound to different ELBs in different deployment environments to avoid route errors.
2. (Optional) Set **Client Protocol**.
   - **HTTP** has security risks. You are advised to select **HTTPS**.
   - If **HTTPS** is selected, click **Use existing** to select an existing certificate.

     If no certificate exists, click **Create new** to create a server certificate. For details, see **Creating a Certificate**.
3. Set **Domain Name**.

   Enter a customize domain name.
4. Set **Listening Port**.

Enter the listening port number of the application process. Listening ports can be set only for components deployed in containers.

Step 9 (Optional) In the **Local Time** area, set the time zone of the container.

This parameter is available when the component is deployed in the Kubernetes environment.

By default, the time zone is the same as that of the region where the container node is located.

Step 10 (Optional) Set **Advanced Settings**. For details, see the following tables.

•

Operation Type	Operation	Description
Microservice Engine	Binding a Microservice Engine	Bind the microservice engine and connect the component to it. **NOTE** After a component developed based on ServiceComb 2.7.8 or later or Spring Cloud Huawei 1.10.4-2021.0.x or later is connected to a microservice engine, the following attributes are injected into the **properties** parameter of the **MicroServiceInstance** parameter when a microservice instance is created in the microservice engine: 1. **CAS_APPLICATION_ID**: ID of the application to which the component belongs. 2. **CAS_COMPONENT_NAME**: component name. 3. **CAS_ENVIRONMENT_ID**: ID of the component deployment environment. 4. **CAS_INSTANCE_ID**: component instance ID. 5. **CAS_INSTANCE_VERSION**: component instance version. 1. Choose **Advanced Settings** > **Microservice Engine**. 2. Click **Bind Microservice Engine**. 3. Select a microservice engine instance that has been bound in the environment. 4. Click **OK**. **NOTE** Move the cursor to a bound microservice engine and perform the following operations: ▪ Bind the microservice engine again: Click , select the target microservice engine again, and click **OK**. ▪ Delete a bound microservice engine: Click . 5. Click and enter the listening port number of the application process to enable multi-language access to service mesh. You can use Mesher to connect components that are not developed in the microservice framework to the microservice engine. **NOTE** ▪ For non-microservice components developed using Java, Tomcat, or Docker, you can enable Mesher and use Mesher to connect the components to CSE for microservice registry and discovery. ▪ For components developed using Python, PHP, or Node.js, forcibly enable Mesher and connect the components to CSE for microservice registry and discovery.

Operation Type	Operation	Description
Distributed Cache Service	Binding a Distributed Cache	In a distributed system, the distributed cache service provides scalable and reliable user session management. Choose **Advanced Settings** > **Distributed Cache Service** and bind a DCS instance. For details, see **7.17.4 Configuring a Distributed Cache**.
Cloud Database	Binding a Cloud Database	The cloud database is required for persistent storage of application data. Expand **Advanced Settings** > **Cloud Database** and bind cloud database. For details, see **7.17.5 Configuring Relational Databases**.
Component Configuration	Configuring Environment Variables	Environment variables are set in the container running environment and can be modified after component deployment, ensuring the flexibility of applications. Environment variables set for a component are local environment variables and take effect only for this component. Choose **Advanced Settings** > **Component Configuration** and set environment variables. For details, see **7.17.1 Setting Component Environment Variables**.
Deployment Configuration	Setting Startup Commands	A startup command is used to start a container. Choose **Advanced Settings** > **Deployment Configuration** and set the startup command. For details, see **7.17.2 Configuring the Lifecycle of a Component**.
	Configuring Data Storage	Container storage is a component that provides storage for applications. Multiple types of storage are supported. A component can use any amount of storage. Choose **Advanced Settings** > **Deployment Configuration** and configure data storage. For details, see **7.17.3 Configuring Data Storage**.
	Configuring the Lifecycle	For container-deployed components, ServiceStage provides callback functions for the lifecycle management of applications. For example, if you want an application component to perform a certain operation before stopping, you can register a hook function. Choose **Advanced Settings** > **Deployment Configuration** and configure the lifecycle. For details, see **7.17.2 Configuring the Lifecycle of a Component**.

Operation Type	Operation	Description
	Configuring the Scheduling Policy	For container-based deployment components, ServiceStage splits the components into minimum deployment instances based on the deployment features of the components. The application scheduler monitors application instance information in real time. When detecting that a new pod needs to be scheduled, the application scheduler calculates all remaining resources (compute, storage, and network resources) in the cluster to obtain the most appropriate scheduling target node. Choose **Advanced Settings** > **Deployment Configuration** and configure the scheduling policy. For details, see **7.17.6 Configuring a Scheduling Policy of a Component Instance**.
O&M Monitoring	Configuring Log Collection	For container-based deployment components, ServiceStage supports setting of application log policies. You can view related logs on the AOM console. You can configure a log policy during or after component deployment. If no configuration is performed, the system collects standard application output logs by default. Choose **Advanced Settings** > **O&M Monitoring** and configure log collection. For details, see **7.17.7 Configuring a Log Policy of an Application**.
	Configuring Health Check	Health check periodically checks application health status during running of container-based deployment components. Choose **Advanced Settings** > **O&M Monitoring** and configure health check. For details, see **7.17.9 Configuring Health Check**.
	Configuring Custom Monitoring	ServiceStage allows you to obtain monitoring data based on custom metrics. You can set custom metric monitoring during or after component deployment. Components deployed using CCE support custom monitoring. Choose **Advanced Settings** > **O&M Monitoring** and configure custom monitoring. For details, see **7.17.8 Configuring Custom Monitoring of a Component**.

Step 11 Click **Create and Deploy**.

On the **Deployment Records** page, view the deployment logs and wait until the component deployment is complete.

----End

7.3 Viewing Component Details

After the component is created and deployed, you can view the component details on the component **Overview** page.

Procedure

Step 1 Log in to ServiceStage.

Step 2 Use either of the following methods to go to the **Overview** page and view component details.

- On the **Application Management** page, click the application to which the component belongs, and click the name of the target component in the **Component List**.
- On the **Component Management** page, click the target component.

NOTE

ServiceStage periodically syncs component information from compute resources (CCE) that run component instances. To manually update and sync component information, perform the following operations:

1. Select the target component:
   - On the **Application Management** page, click the application to which the component belongs to go to the **Application Overview** page. In the **Component List** area, select the target component.
   - On the **Component Management** page, select the target component.
2. In the **Operation** column, select **Operation** > **Sync Status** to manually sync the updated component information.

----End

7.4 Managing Component Labels

Labels are key-value pairs and can be attached to workloads. Workload labels are often used for affinity and anti-affinity scheduling. You can add labels to multiple workloads or a specified workload.

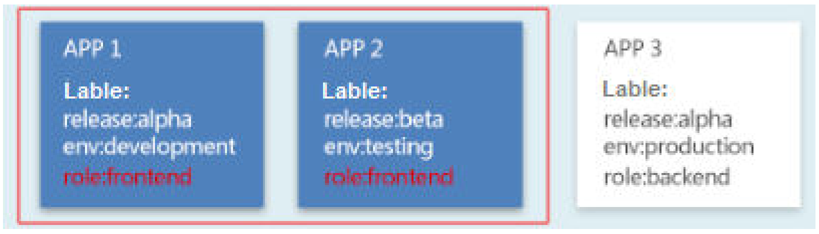
You can manage the labels of stateless workloads, stateful workloads, and Daemon sets based on service requirements. This section uses Deployments as an example to describe how to manage labels.

In the following figure, three labels (release, env, and role) are defined for workload APP 1, APP 2, and APP 3. The values of these labels vary with workload.

- APP 1: [release:alpha;env:development;role:frontend]
- APP 2: [release:beta;env:testing;role:frontend]
- APP 3: [release:alpha;env:production;role:backend]

If you set **key** to **role** and **value** to **frontend** when using workload scheduling or another function, APP 1 and APP 2 will be selected.

Figure 7-3 Label example



NOTE

Labels cannot be added to components that are abnormal.

Adding Component Labels

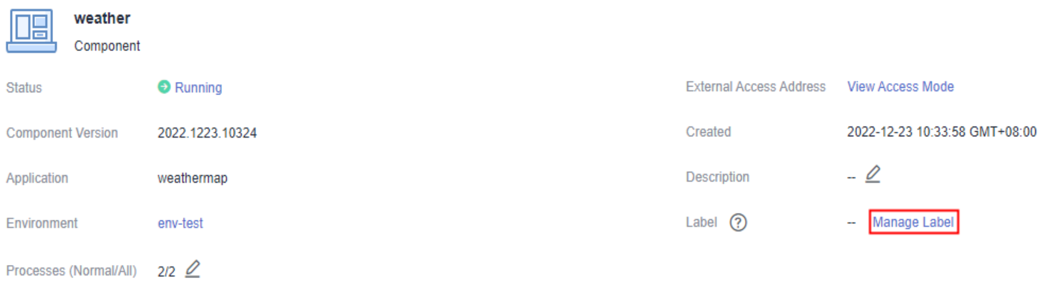
Step 1 Log in to ServiceStage.

Step 2 Use either of the following methods to go to the component **Overview** page.

- On the **Application Management** page, click the application to which the target component belongs, and click the component in **Component List**.
- On the **Component Management** page, click the target component.

Step 3 Click **Manage Label**.

Figure 7-4 Managing labels



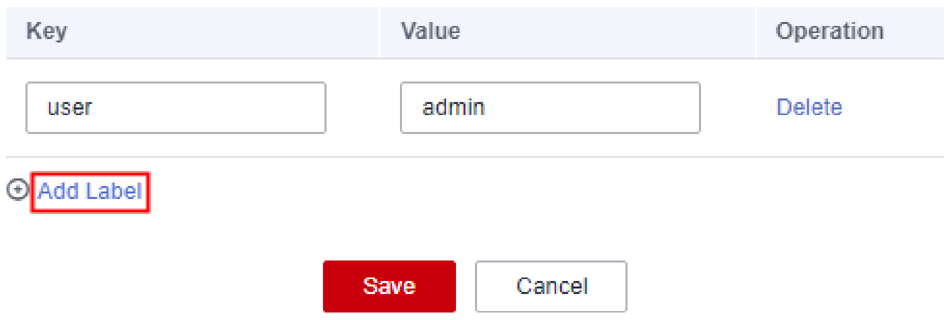
weather
Component

Status	Running	External Access Address	View Access Mode
Component Version	2022.1223.10324	Created	2022-12-23 10:33:58 GMT+08:00
Application	weathermap	Description	--
Environment	env-test	Label	-- Manage Label
Processes (Normal/All)	2/2		

Step 4 Click **Add Label**.

1. Enter the **key** and **value**.
   The key must be unique.
2. Click **Save**.

Figure 7-5 Adding a label



----End

Deleting a Component Label

Step 1 Log in to ServiceStage.

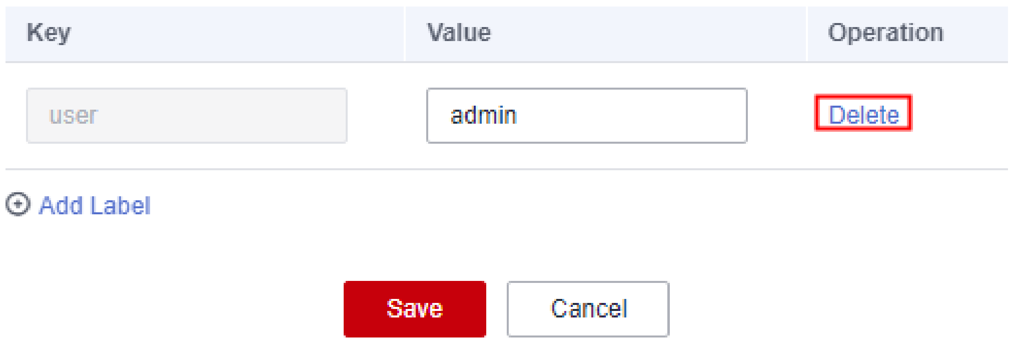
Step 2 Use either of the following methods to go to the component **Overview** page.

- On the **Application Management** page, click the application to which the target component belongs, and click the component in **Component List**.
- On the **Component Management** page, click the target component.

Step 3 Click **Manage Label**.

Step 4 Select the label to be deleted and click **Delete** in the **Operation** column.

Figure 7-6 Deleting a label



Step 5 Click **Save**.

----End

7.5 Managing Component Instances

After a component is created and deployed, you can manage component instances on the component **Instance** page.

Procedure

Step 1 Log in to ServiceStage.

Step 2 Use either of the following methods to go to the **Instance List** page.

- On the **Application Management** page, click the application to which the component belongs, and click the target component in **Component List**. In the left navigation pane, choose **Instance List**.
- On the **Component Management** page, click the target component. In the left navigation pane, choose **Instance List**.

Step 3 On the **Instance List** page, you can perform the operations listed in the following table.

Operation	Description
Restart a single instance	If an instance of a component deployed in the Kubernetes environment is abnormal, you can delete the instance to restart it. 1. Select the instance to be deleted and click **Delete** in the **Operation** column. 2. In the displayed dialog box, click **OK**.
View instance running monitoring information	By viewing the instance running monitoring information, you can learn about the CPU and memory usage of a single running instance. 1. In the instance list, click ∨ next to the target instance. 2. Click the **Monitor** tab to view the running monitoring information about the instance.
View instance running events	ServiceStage allows you to view details about events that occur during the running of a specified instance. 1. In the instance list, click ∨ next to the target instance. 2. Click the **Events** tab to view details about the events that occur during the running of the instance.
View running instance containers	For components deployed in the Kubernetes environment, ServiceStage allows you to view information about the container where a specified instance runs, including the container name, running status, and mounted image. 1. In the instance list, click ∨ next to the target instance. 2. Click the **Container** tab to view the information about the container where the instance runs.

----End

7.6 Upgrading a Single Component

7.6.1 Single-batch Release

After a component is created and deployed, you can upgrade a **Running** or **Not ready** component in single-batch release mode.

In single-batch release mode, all instances are upgraded at a time. During the upgrade, component services will be interrupted. This is applicable to the test upgrade scenario or the upgrade scenario where services are to be stopped. The upgrade takes a short time.

NOTE

Only components deployed in the Kubernetes environment can be upgraded in single-batch release mode.

For details about how to upgrade multiple components of the same application in batches, see **7.7 Upgrading Components in Batches**.

Prerequisites

You have created and deployed a component. For details, see **7.2 Creating and Deploying a Component**.

Procedure

Step 1 Log in to ServiceStage.

Step 2 Use either of the following methods to go to the component **Overview** page.

- On the **Application Management** page, click the application to which the target component belongs, and click the component in **Component List**.
- On the **Component Management** page, click the target component.

Step 3 Click **Upgrade** in the upper right corner of the page.

Step 4 Select **Single-batch Release** for **Upgrade Type**.

Step 5 Click **Next** and set the component version configuration information by referring to the following table. Parameters marked with an asterisk (*) are mandatory.

Parameter	Description
Stack	The value is fixed to the technology stack selected during component creation and deployment.
*Software Package/ Image	The value is fixed to the component source selected during component creation and deployment. If you select **Source code repository**, create authorization by referring to **9.7 Authorizing a Repository** and set the code source. If you select a software package, the software package type supported by the component source is determined by the selected technology stack type. For details, see **Table 7-1**.

Parameter	Description
*Upload Method	If the component source is software package or image package, select an uploaded software package or image package. For details about the upload method, see **Component Source**.
*Command	This parameter is mandatory if the component source is source code, the component is deployed in the Kubernetes environment, and the selected technology stack type is Java, Tomcat, Node.js, Python, or PHP. ● **Default command or script**: preferentially executes **build.sh** in the **root** directory. If **build.sh** does not exist, the code will be compiled using the common method of the selected language. Example: **mvn clean package** for Java. ● **Custom command**: Commands are customized using the selected language. Alternatively, the default command or script is used after **build.sh** is modified. **NOTICE** – If **Custom command** is selected, exercise caution when inputting sensitive information in the **echo**, **cat**, or **debug** command, or encrypt sensitive information to avoid information leakage. – To run the compilation command in the project subdirectory, you need to go to the project subdirectory and then run other script commands. For example: **cd ./weather/** **mvn clean package**
*Dockerfile Address	This parameter is mandatory if the component source is **Source code repository**, the component is deployed in the Kubernetes environment, and the selected technology stack type is Java, Tomcat, Node.js, Python, or PHP. **Dockerfile Address** is the directory where the Dockerfile is located relative to the root directory (./) of the project. The Dockerfile is used to build an image. If **Dockerfile Address** is not specified, the system searches for the Dockerfile in the root directory of the project by default. If the Dockerfile does not exist in the root directory, the system automatically generates the Dockerfile based on the selected operating environment.

Parameter	Description
*Component Version	Version number of a component. ● Automatically-generated: Click **Generate**. By default, the version number is the timestamp when you click **Generate**. The format is yyyy.mmdd.hhmms, where **s** is the ones place of the second in the timestamp. For example, if the timestamp is 2022.1214.172318, the version number is 2022.1214.17238. ● You can also customize the version number in the format of A.B.C, or A.B.C.D. A. B, C, and D are natural numbers, for example, 1.0.0 or 1.0.0.0. **NOTICE** – The customized version number must be unique and cannot be the same as any historical version number of the component.
Resources	Components cannot be scheduled to nodes whose residual resources are fewer than the requested amount. For details about how to configure the request and limit parameters, see **Managing Resources for Containers**. You can customize **CPU** and **Memory** to set their quota, and change the maximum/minimum number of CPU cores and memory size (GiB) that can be used by components. To modify them, select the item to be changed and enter a new value. Unselected parameters indicate no restriction.
JVM Parameters	This parameter is available when the technology stack type is Java or Tomcat. It configures the memory size during Java code running. Enter the JVM parameter, for example, **-Xms256m -Xmx1024m**. Multiple parameters are separated by spaces.
Tomcat	This parameter is available when the technology stack type is Tomcat. It configures parameters such as the request path and port number of Tomcat. 1. Select **Tomcat**. The **Tomcat** dialog box is displayed. 2. Click **Use Sample Code** and edit the template file based on service requirements. 3. Click **OK**.
Advanced Settings	Set **Component Configuration**, **Deployment Configuration**, and **O&M Monitoring** by referring to **Step 10**.

Step 6 Click **Upgrade**.

Wait until the component status changes from **Upgrading/Rolling back** to **Running**, indicating that the component version configuration is successfully upgraded.

----End

Follow-Up Operations

Operation	Description
Redeploy a component	You can select a historical version configuration from the deployment record list and use the version configuration as a template to redeploy components. For details, see **7.9 Redeploying a Component**.

7.6.2 Rolling Release

After a component is created and deployed, you can upgrade a **Running** or **Not ready** component in rolling release mode.

In rolling release mode, only one or more instances are upgraded at a time and then added to the production environment. This process repeats until all old instances are upgraded. Services will not be interrupted during the upgrade.

For details about how to upgrade multiple components of the same application in batches, see **7.7 Upgrading Components in Batches**.

Prerequisites

You have created and deployed a component. For details, see **7.2 Creating and Deploying a Component**.

Procedure

Step 1 Log in to ServiceStage.

Step 2 Use either of the following methods to go to the component **Overview** page.

- On the **Application Management** page, click the application to which the target component belongs, and click the component in **Component List**.
- On the **Component Management** page, click the target component.

Step 3 Click **Upgrade** in the upper right corner of the page.

Step 4 Select **Rolling Release** for **UpgradeType**.

Step 5 Click **Next** and set the component version configuration information by referring to the following table. Parameters marked with an asterisk (*) are mandatory.

Parameter	Description
Stack	The value is fixed to the technology stack selected during component creation and deployment.

Parameter	Description
*Software Package/ Image	The value is fixed to the component source selected during component creation and deployment. If you select **Source code repository**, create authorization by referring to **9.7 Authorizing a Repository** and set the code source. If you select a software package, the software package type supported by the component source is determined by the selected technology stack type. For details, see **Table 7-1**.
*Upload Method	If the component source is software package or image package, select an uploaded software package or image package. For details about the upload method, see **Component Source**.
*Command	This parameter is mandatory if the component source is source code, the component is deployed in the Kubernetes environment, and the selected technology stack type is Java, Tomcat, Node.js, Python, or PHP. ● **Default command or script**: preferentially executes **build.sh** in the **root** directory. If **build.sh** does not exist, the code will be compiled using the common method of the selected language. Example: **mvn clean package** for Java. ● **Custom command**: Commands are customized using the selected language. Alternatively, the default command or script is used after **build.sh** is modified. **NOTICE** – If **Custom command** is selected, exercise caution when inputting sensitive information in the **echo**, **cat**, or **debug** command, or encrypt sensitive information to avoid information leakage. – To run the compilation command in the project subdirectory, you need to go to the project subdirectory and then run other script commands. For example: **cd ./weather/** **mvn clean package**
*Dockerfile Address	This parameter is mandatory if the component source is **Source code repository**, the component is deployed in the Kubernetes environment, and the selected technology stack type is Java, Tomcat, Node.js, Python, or PHP. **Dockerfile Address** is the directory where the Dockerfile is located relative to the root directory (./) of the project. The Dockerfile is used to build an image. If **Dockerfile Address** is not specified, the system searches for the Dockerfile in the root directory of the project by default. If the Dockerfile does not exist in the root directory, the system automatically generates the Dockerfile based on the selected operating environment.

Parameter	Description
*Component Version	Version number of a component. ● Automatically-generated: Click **Generate**. By default, the version number is the timestamp when you click **Generate**. The format is yyyy.mmdd.hhmms, where **s** is the ones place of the second in the timestamp. For example, if the timestamp is 2022.1214.172318, the version number is 2022.1214.17238. ● You can also customize the version number in the format of A.B.C, or A.B.C.D. A. B, C, and D are natural numbers, for example, 1.0.0 or 1.0.0.0. **NOTICE** – The customized version number must be unique and cannot be the same as any historical version number of the component.
Resources	This parameter is available when the component is deployed in the Kubernetes environment. Components cannot be scheduled to nodes whose residual resources are fewer than the requested amount. For details about how to configure the request and limit parameters, see **Managing Resources for Containers**. You can customize **CPU** and **Memory** to set their quota, and change the maximum/minimum number of CPU cores and memory size (GiB) that can be used by components. To modify them, select the item to be changed and enter a new value. Unselected parameters indicate no restriction.
JVM Parameters	This parameter is available when the technology stack type is Java or Tomcat. It configures the memory size during Java code running. Enter the JVM parameter, for example, **-Xms256m -Xmx1024m**. Multiple parameters are separated by spaces.
Tomcat	This parameter is available when the technology stack type is Tomcat. It configures parameters such as the request path and port number of Tomcat. 1. Select **Tomcat**. The **Tomcat** dialog box is displayed. 2. Click **Use Sample Code** and edit the template file based on service requirements. 3. Click **OK**.
Advanced Settings	This parameter is available when the component is deployed in the Kubernetes environment. Set **Component Configuration**, **Deployment Configuration**, and **O&M Monitoring** by referring to **Step 10**.

Parameter	Description
*Deploymen t Batches	This parameter is available when the component is deployed in the Kubernetes environment. Number of batches in which component instances are upgraded. The value range is [1, Total number of instances]. Total number of instances refers to the number of running instances of the component. For example, if there are 4 component instances and **Deployment Batches** is set to **2**, these component instances are upgraded in two batches, and each batch involves two component instances.

Step 6 Click **Upgrade**.

Wait until the component status changes from **Upgrading/Rolling back** to **Running**, indicating that the component version configuration is successfully upgraded.

----End

Follow-Up Operations

Operation	Description
Redeploy a component	You can select a historical version configuration from the deployment record list and use the version configuration as a template to redeploy components. For details, see **7.9 Redeploying a Component**.

7.6.3 Dark Launch (Canary)

After a component is created and deployed, you can upgrade a **Running** or **Not ready** component in dark launch (canary) mode.

In dark launch (canary) mode, a certain proportion of instances are upgraded, and traffic is directed to the new version to verify whether functions of the new version are normal. Then, the remaining instances will be upgraded in rolling mode. Dark launch ensures stability of the entire system. During initial dark launch, problems can be detected and fixed.

For details about dark launch (canary) types and details, see **Table 7-2**.

Table 7-2 Dark launch (canary) types and description

Type	Description
Microservice Dark Launch	Applies to ServiceComb and Spring Cloud applications. Dark launch tasks function on microservices. Multiple microservices can work together to roll out new features. 1. The Java, Tomcat, or Docker technology stack must be selected for the component. 2. The component must be bound to a microservice engine with security authentication disabled and multi-language access to service mesh disabled. 3. ServiceComb 2.7.8 or later is required. Spring Cloud Huawei 1.10.4-2021.0.x or later is required.
ELB Dark Launch	Applies to ELB traffic-based components. Dark launch tasks function on ELB. The component must be accessible from the public network and bound to an ELB.

NOTE

Dark launch (canary) is supported only when the deployment environment is Kubernetes and there are two or more component instances.

For details about how to upgrade multiple components of the same application in batches, see **7.7 Upgrading Components in Batches**.

Prerequisites

You have created and deployed a component. For details, see **7.2 Creating and Deploying a Component**.

Procedure

Step 1 Log in to ServiceStage.

Step 2 Use either of the following methods to go to the component **Overview** page.

- On the **Application Management** page, click the application to which the target component belongs, and click the component in **Component List**.
- On the **Component Management** page, click the target component.

Step 3 Click **Upgrade** in the upper right corner of the page.

Step 4 Select **Dark Launch (Canary)** for **Upgrade Type**.

Step 5 Click **Next** and set the component version configuration information by referring to the following table. Parameters marked with an asterisk (*) are mandatory.

Parameter	Description
Stack	The value is fixed to the technology stack selected during component creation and deployment.
*Software Package/ Image	The value is fixed to the component source selected during component creation and deployment. If you select **Source code repository**, create authorization by referring to **9.7 Authorizing a Repository** and set the code source. If you select a software package, the software package type supported by the component source is determined by the selected technology stack type. For details, see **Table 7-1**.
*Upload Method	If the component source is software package or image package, select an uploaded software package or image package. For details about the upload method, see **Component Source**.
*Command	This parameter is mandatory if the component source is source code, the component is deployed in the Kubernetes environment, and the selected technology stack type is Java, Tomcat, Node.js, Python, or PHP. ● **Default command or script**: preferentially executes **build.sh** in the **root** directory. If **build.sh** does not exist, the code will be compiled using the common method of the selected language. Example: **mvn clean package** for Java. ● **Custom command**: Commands are customized using the selected language. Alternatively, the default command or script is used after **build.sh** is modified. **NOTICE** – If **Custom command** is selected, exercise caution when inputting sensitive information in the **echo**, **cat**, or **debug** command, or encrypt sensitive information to avoid information leakage. – To run the compilation command in the project subdirectory, you need to go to the project subdirectory and then run other script commands. For example: **cd ./weather/** **mvn clean package**
*Dockerfile Address	This parameter is mandatory if the component source is **Source code repository**, the component is deployed in the Kubernetes environment, and the selected technology stack type is Java, Tomcat, Node.js, Python, or PHP. **Dockerfile Address** is the directory where the Dockerfile is located relative to the root directory (./) of the project. The Dockerfile is used to build an image. If **Dockerfile Address** is not specified, the system searches for the Dockerfile in the root directory of the project by default. If the Dockerfile does not exist in the root directory, the system automatically generates the Dockerfile based on the selected operating environment.

Parameter	Description
*Componen t Version	Version number of a component. ● Automatically-generated: Click **Generate**. By default, the version number is the timestamp when you click **Generate**. The format is yyyy.mmdd.hhmms, where **s** is the ones place of the second in the timestamp. For example, if the timestamp is 2022.1214.172318, the version number is 2022.1214.17238. ● You can also customize the version number in the format of A.B.C, or A.B.C.D. A. B, C, and D are natural numbers, for example, 1.0.0 or 1.0.0.0. **NOTICE** – The customized version number must be unique and cannot be the same as any historical version number of the component.
Resources	This parameter is available when the component is deployed in the Kubernetes environment. Components cannot be scheduled to nodes whose residual resources are fewer than the requested amount. For details about how to configure the request and limit parameters, see **Managing Resources for Containers**. You can customize **CPU** and **Memory** to set their quota, and change the maximum/minimum number of CPU cores and memory size (GiB) that can be used by components. To modify them, select the item to be changed and enter a new value. Unselected parameters indicate no restriction.
JVM Parameters	This parameter is available when the technology stack type is Java or Tomcat. It configures the memory size during Java code running. Enter the JVM parameter, for example, **-Xms256m -Xmx1024m**. Multiple parameters are separated by spaces.
Tomcat	This parameter is available when the technology stack type is Tomcat. It configures parameters such as the request path and port number of Tomcat. 1. Select **Tomcat**. The **Tomcat** dialog box is displayed. 2. Click **Use Sample Code** and edit the template file based on service requirements. 3. Click **OK**.
Advanced Settings	Set **Component Configuration**, **Deployment Configuration**, and **O&M Monitoring** by referring to **Step 10**.
Dark Launch Policy	● **Traffic Ratio**: percentage of traffic directed to the new version. ● **Current Traffic Ratio**: percentage of traffic directed to the current version.

Parameter	Description
*First Batch of Dark Launch Instances	Number of instances for dark launch in the first batch. The value range is [1, Total number of instances – 1]. Total number of instances refers to the number of running instances of the component. For example, if there are 6 component instances and **First Batch of Dark Launch Instances** is set to **1**, 1 instance will be upgraded in the first batch.
Deployment Batch with Remaining Instances	Number of batches whose remaining instances will be upgraded. For example, if there are 6 component instances, **First Batch of Dark Launch Instances** is set to **1**, and **Deployment Batch with Remaining Instances** is set to **3**, there are 5 instances remaining to be deployed in 3 batches, and these 5 instances will be upgraded in the sequence 2:2:1.

Step 6 Click **Upgrade**.

Wait until the component status changes from **Upgrading/Rolling back** to **Running**, indicating that the component version configuration is successfully upgraded.

----End

Follow-Up Operations

Operation	Description
View system monitoring	If the component is upgraded through dark launch (canary), choose **System Monitoring** to monitor the CPU and memory usage of instances of the dark launch version and the current version after the first batch of dark launch is complete.
Upgrade remaining instances in rolling mode	If the component is upgraded through dark launch (canary) after the first batch of dark launch is successful and the functions of the new version are normal, you can perform the following operations to upgrade the remaining component instances: 1. Select the deployment record of the **Dark Launch (Canary)** type. 2. Click **Upgrade Remaining Instances in Rolling Mode**. 3. In the displayed dialog box, click **OK**. The remaining instances are upgraded to the new version based on the upgrade policy set in **Step 5**.

Operation	Description
Roll back a component	The component version can be rolled back in the following scenarios: ● The first batch of dark launch is complete if the component is upgraded through dark launch (canary). ● All component instances have been upgraded to the new version. To roll back the component configuration to the source version, refer to **7.8 Rolling Back a Component**.
Redeploy a component	You can select a historical version configuration from the deployment record list and use the version configuration as a template to redeploy components. For details, see **7.9 Redeploying a Component**.

7.7 Upgrading Components in Batches

After components are created and deployed, you can reconfigure and deploy multiple **Running** and **Not ready** components of the same application in rolling release mode. In rolling release mode, only one or more instances are upgraded at a time. After the upgrade is complete, the instances are added to the production environment. This process will be repeated until all old versions are upgraded to the new version. During the upgrade, services are not interrupted.

For details about how to upgrade the version configuration of a single component, see **7.6 Upgrading a Single Component**.

Procedure

Step 1 Log in to ServiceStage.

Step 2 Choose **Application Management**.

Step 3 Click the application where the target component is located. The **Overview** page of the application is displayed.

Step 4 Select the components to be upgraded in batches in **Component List** and click **Bulk Upgrade**.

Step 5 Set the version configuration information of the components to be upgraded by referring to the following table.

Parameter	Description
Target Version	Target version of the upgraded component. ● Automatically-generated: Click **Generate**. By default, the version number is the timestamp when you click **Generate**. The format is yyyy.mmdd.hhmms, where **s** is the ones place of the second in the timestamp. For example, if the timestamp is 2022.1214.172318, the version number is 2022.1214.17238. ● You can also customize the version number in the format of A.B.C, or A.B.C.D. A. B, C, and D are natural numbers, for example, 1.0.0 or 1.0.0.0. **NOTICE** – The customized version number must be unique and cannot be the same as any historical version number of the component.
Software Package/ Image Package/ Source Code Repository	Click and select the software package or image packagesource code repository again. For details, see **Component Source**.
Deployment Batches	Number of batches in which component instances are upgraded. The value range is [1, Total number of instances]. Total number of instances refers to the number of running instances of the component. For example, if there are 4 component instances and **Deployment Batches** is set to **2**, these component instances are upgraded in two batches, and each batch involves two component instances.

Click in the **Operation** column of a component to deselect the component to be upgraded.

Step 6 Click **OK**.

Wait until the component status changes from **Upgrading/Rolling back** to **Running**, indicating that the component version configuration is successfully upgraded.

----End

Follow-Up Operations

Operation	Description
Roll back a component	After the version configuration of all component instances is upgraded to the new version, if you need to roll back the component to the source version, see **7.8 Rolling Back a Component**.

Operation	Description
Redeploy a component	You can select a historical version configuration from the deployment record list and use the version configuration as a template to redeploy components. For details, see **7.9 Redeploying a Component**.

7.8 Rolling Back a Component

Based on service requirements, you can roll back a component from the latest version to the version before the upgrade or redeployment.

A component that has been rolled back cannot be rolled back again.

Procedure

Step 1 Log in to ServiceStage.

Step 2 Use either of the following methods to go to the **deployment record** page.

- On the **Application Management** page, click the application to which the component belongs, and click the target component in **Component List**. In the left navigation pane, choose **deployment record**.
- On the **Component Management** page, click the target component. In the left navigation pane, choose **deployment record**.

Step 3 In the **deployment record** list, select the deployment record of the latest version.

Step 4 Click **Roll Back**.

Step 5 In the displayed dialog box, click **OK**.

After the rollback is complete, the component version will be rolled back to the version before upgrade.

----End

7.9 Redeploying a Component

7.9.1 Single-batch Release

You can select a historical version configuration from the deployment record list and use the version configuration as a template to redeploy components in single-batch release mode.

In single-batch release mode, all instances are redeployed at a time. During the deployment, component services will be interrupted. This is applicable to the test deployment scenario or the deployment scenario where services are to be stopped. The deployment takes a short time.

NOTE

Only components deployed in the Kubernetes environment can be redeployed in single-batch release mode.

The component version configuration that has been rolled back by referring to **7.8 Rolling Back a Component** cannot be used as a template to redeploy the component.

Prerequisites

You have upgraded a component. For details, see **7.6 Upgrading a Single Component** or **7.7 Upgrading Components in Batches**.

Procedure

Step 1 Log in to ServiceStage.

Step 2 Use either of the following methods to go to the **Deployment Records** page.

- On the **Application Management** page, click the application to which the component belongs, and click the target component in **Component List**. In the left navigation pane, choose **Deployment Records**.
- On the **Component Management** page, click the target component. In the left navigation pane, choose **Deployment Records**.

Step 3 In the **Deployment Records** list, select the deployment record of the historical version to be used as the configuration template.

Step 4 Click **Redeploy** in the upper right corner of the page. The **Redeploy** dialog box is displayed.

Step 5 Select **Single-batch Release** for **Deployment Type** and click **OK**.

Step 6 Configure the component version by referring to the following table. Parameters marked with an asterisk (*) are mandatory.

Parameter	Description
Stack	The value is fixed to the configuration of the selected historical version and cannot be changed.
*Software Package/ Image	The value is fixed to the component source selected during component creation and deployment. If you select **Source code repository**, create authorization by referring to **9.7 Authorizing a Repository** and set the code source. If you select a software package, the software package type supported by the component source is determined by the selected technology stack type. For details, see **Table 7-1**.
*Upload Method	If the component source is software package or image package, select an uploaded software package or image package. For details about the upload method, see **Component Source**.

Parameter	Description
*Command	This parameter is mandatory if the component source is **Source code repository**, the component is deployed in the Kubernetes environment, and the selected technology stack type is Java, Tomcat, Node.js, Python, or PHP. ● **Default command or script**: preferentially executes **build.sh** in the **root** directory. If **build.sh** does not exist, the code will be compiled using the common method of the selected language. Example: **mvn clean package** for Java. ● **Custom command**: Commands are customized using the selected language. Alternatively, the default command or script is used after **build.sh** is modified. **NOTICE** – If **Custom command** is selected, exercise caution when inputting sensitive information in the **echo**, **cat**, or **debug** command, or encrypt sensitive information to avoid information leakage. – To run the compilation command in the project subdirectory, you need to go to the project subdirectory and then run other script commands. For example: **cd ./weather**/ **mvn clean package**
*Dockerfile Address	This parameter is mandatory if the component source is **Source code repository**, the component is deployed in the Kubernetes environment, and the selected technology stack type is Java, Tomcat, Node.js, Python, or PHP. **Dockerfile Address** is the directory where the Dockerfile is located relative to the root directory (./) of the project. The Dockerfile is used to build an image. If **Dockerfile Address** is not specified, the system searches for the Dockerfile in the root directory of the project by default. If the Dockerfile does not exist in the root directory, the system automatically generates the Dockerfile based on the selected operating environment.
*Component Version	Version number of a component. ● Automatically-generated: Click **Generate**. By default, the version number is the timestamp when you click **Generate**. The format is yyyy.mmdd.hhmms, where **s** is the ones place of the second in the timestamp. For example, if the timestamp is 2022.1214.172318, the version number is 2022.1214.17238. ● You can also customize the version number in the format of A.B.C, or A.B.C.D. A. B, C, and D are natural numbers, for example, 1.0.0 or 1.0.0.0. **NOTICE** – The customized version number must be unique and cannot be the same as any historical version number of the component.
Resources	The value is fixed to the configuration of the selected historical version and cannot be changed.

Parameter	Description
JVM Parameters	The value is fixed to the configuration of the selected historical version and cannot be changed. This parameter is available when the technology stack type is Java or Tomcat. It configures the memory size during Java code running.
Tomcat	The value is fixed to the configuration of the selected historical version and cannot be changed. This parameter is available when the technology stack type is Tomcat. It configures parameters such as the request path and port number of Tomcat.
Advanced Settings	The value is fixed to the configuration of the selected historical version and cannot be changed.

Step 7 Click **Upgrade**.

In the **Deployment Records** area, you can view the deployment progress and wait until the deployment is complete.

----End

7.9.2 Rolling Release

You can select a historical version configuration from the deployment record list and use the version configuration as a template to redeploy components in rolling release mode.

In rolling release mode, only one or more instances are deployed at a time and then added to the production environment. This process repeats until all old instances are upgraded. Services will not be interrupted during the deployment.

The component version configuration that has been rolled back by referring to **7.8 Rolling Back a Component** cannot be used as a template to redeploy the component.

Prerequisites

You have upgraded a component. For details, see **7.6 Upgrading a Single Component** or **7.7 Upgrading Components in Batches**.

Procedure

Step 1 Log in to ServiceStage.

Step 2 Use either of the following methods to go to the **Deployment Records** page.

- On the **Application Management** page, click the application to which the component belongs, and click the target component in **Component List**. In the left navigation pane, choose **Deployment Records**.
- On the **Component Management** page, click the target component. In the left navigation pane, choose **Deployment Records**.

Step 3 In the **Deployment Records** list, select the deployment record of the historical version to be used as the configuration template.

Step 4 Click **Redeploy** in the upper right corner of the page. The **Redeploy** dialog box is displayed.

Step 5 Select **Rolling Release** for **Deployment Type** and click **OK**.

Step 6 Configure the component version by referring to the following table. Parameters marked with an asterisk (*) are mandatory.

Parameter	Description
Stack	The value is fixed to the configuration of the selected historical version and cannot be changed.
*Software Package/ Image	The value is fixed to the component source selected during component creation and deployment. If you select **Source code repository**, create authorization by referring to **9.7 Authorizing a Repository** and set the code source. If you select a software package, the software package type supported by the component source is determined by the selected technology stack type. For details, see **Table 7-1**.
*Upload Method	If the component source is software package or image package, select an uploaded software package or image package. For details about the upload method, see **Component Source**.
*Command	This parameter is mandatory if the component source is **Source code repository**, the component is deployed in the Kubernetes environment, and the selected technology stack type is Java, Tomcat, Node.js, Python, or PHP. ● **Default command or script**: preferentially executes **build.sh** in the **root** directory. If **build.sh** does not exist, the code will be compiled using the common method of the selected language. Example: **mvn clean package** for Java. ● **Custom command**: Commands are customized using the selected language. Alternatively, the default command or script is used after **build.sh** is modified. **NOTICE** – If **Custom command** is selected, exercise caution when inputting sensitive information in the **echo**, **cat**, or **debug** command, or encrypt sensitive information to avoid information leakage. – To run the compilation command in the project subdirectory, you need to go to the project subdirectory and then run other script commands. For example: **cd ./weather/** **mvn clean package**

Parameter	Description
*Dockerfile Address	This parameter is mandatory if the component source is **Source code repository**, the component is deployed in the Kubernetes environment, and the selected technology stack type is Java, Tomcat, Node.js, Python, or PHP. **Dockerfile Address** is the directory where the Dockerfile is located relative to the root directory (./) of the project. The Dockerfile is used to build an image. If **Dockerfile Address** is not specified, the system searches for the Dockerfile in the root directory of the project by default. If the Dockerfile does not exist in the root directory, the system automatically generates the Dockerfile based on the selected operating environment.
*Component Version	Version number of a component. ● Automatically-generated: Click **Generate**. By default, the version number is the timestamp when you click **Generate**. The format is yyyy.mmdd.hhmms, where **s** is the ones place of the second in the timestamp. For example, if the timestamp is 2022.1214.172318, the version number is 2022.1214.17238. ● You can also customize the version number in the format of A.B.C, or A.B.C.D. A. B, C, and D are natural numbers, for example, 1.0.0 or 1.0.0.0. **NOTICE** – The customized version number must be unique and cannot be the same as any historical version number of the component.
Resources	The value is fixed to the configuration of the selected historical version and cannot be changed. This parameter is available when the component is deployed in the Kubernetes environment.
JVM Parameters	The value is fixed to the configuration of the selected historical version and cannot be changed. This parameter is available when the technology stack type is Java or Tomcat. It configures the memory size during Java code running.
Tomcat	The value is fixed to the configuration of the selected historical version and cannot be changed. This parameter is available when the technology stack type is Tomcat. It configures parameters such as the request path and port number of Tomcat.
Advanced Settings	The value is fixed to the configuration of the selected historical version and cannot be changed. This parameter is available when the component is deployed in the Kubernetes environment.

Parameter	Description
*Deploymen t Batches	Number of batches in which component instances are upgraded. The value range is [1, Total number of instances]. Total number of instances refers to the number of running instances of the component. For example, if there are 4 component instances and **Deployment Batches** is set to **2**, these component instances are upgraded in two batches, and each batch involves two component instances. This parameter is available when the component is deployed in the Kubernetes environment.

Step 7 Click **Upgrade**.

In the **Deployment Records** area, you can view the deployment progress and wait until the deployment is complete.

----End

7.9.3 Dark Launch (Canary)

You can select a historical version configuration from the deployment record list and use the version configuration as a template to redeploy components in dark launch (canary) mode.

In dark launch (canary) mode, a certain proportion of instances are upgraded, and traffic is directed to the new version to verify whether functions of the new version are normal. Then, the remaining instances will be upgraded in rolling mode. Dark launch ensures stability of the entire system. During initial dark launch, problems can be detected and fixed.

For details about dark launch (canary) types and details, see **Table 7-3**.

Table 7-3 Dark launch (canary) types and description

Type	Description
Microservice Dark Launch	Applies to ServiceComb and Spring Cloud applications. Dark launch tasks function on microservices. Multiple microservices can work together to roll out new features. 1. The Java, Tomcat, or Docker technology stack must be selected for the component. 2. The component must be bound to a microservice engine with security authentication disabled and multi-language access to service mesh disabled. 3. ServiceComb 2.7.8 or later is required. Spring Cloud Huawei 1.10.4-2021.0.x or later is required.

Type	Description
ELB Dark Launch	Applies to ELB traffic-based components. Dark launch tasks function on ELB. The component must be accessible from the public network and bound to an ELB.

NOTE

Component redeployment in dark launch (canary) mode is supported only when the deployment environment is Kubernetes and there are two or more component instances.

The component version configuration that has been rolled back by referring to **7.8 Rolling Back a Component** cannot be used as a template to redeploy the component.

Procedure

Step 1 Log in to ServiceStage.

Step 2 Use either of the following methods to go to the **Deployment Records** page.

- On the **Application Management** page, click the application to which the component belongs, and click the target component in **Component List**. In the left navigation pane, choose **Deployment Records**.
- On the **Component Management** page, click the target component. In the left navigation pane, choose **Deployment Records**.

Step 3 In the **Deployment Records** list, select the deployment record of the historical version to be used as the configuration template.

Step 4 Click **Redeploy** in the upper right corner of the page. The **Redeploy** dialog box is displayed.

Step 5 Select **Dark Launch (Canary)** for **Type** and click **OK**.

Step 6 Configure the component version by referring to the following table. Parameters marked with an asterisk (*) are mandatory.

NOTICE

During the dark launch upgrade of a component microservice in this operation, do not use CSE to perform dark launch of the component microservice at the same time. Otherwise, this operation will fail.

For details about how to perform dark launch of a component microservice through CSE, see **Dark Launch**.

Parameter	Description
Stack	The value is fixed to the configuration of the selected historical version and cannot be changed.

Parameter	Description
*Software Package/ Image	The value is fixed to the component source selected during component creation and deployment.
*Upload Method	If the component source is software package or image package, select an uploaded software package or image package. For details about the upload method, see **Component Source**.
*Command	This parameter is mandatory if the component source is **Source code repository**, the component is deployed in the Kubernetes environment, and the selected technology stack type is Java, Tomcat, Node.js, Python, or PHP. ● **Default command or script**: preferentially executes **build.sh** in the **root** directory. If **build.sh** does not exist, the code will be compiled using the common method of the selected language. Example: **mvn clean package** for Java. ● **Custom command**: Commands are customized using the selected language. Alternatively, the default command or script is used after **build.sh** is modified. **NOTICE** – If **Custom command** is selected, exercise caution when inputting sensitive information in the **echo**, **cat**, or **debug** command, or encrypt sensitive information to avoid information leakage. – To run the compilation command in the project subdirectory, you need to go to the project subdirectory and then run other script commands. For example: **cd ./weather/** **mvn clean package**
*Dockerfile Address	This parameter is mandatory if the component source is **Source code repository**, the component is deployed in the Kubernetes environment, and the selected technology stack type is Java, Tomcat, Node.js, Python, or PHP. **Dockerfile Address** is the directory where the Dockerfile is located relative to the root directory (./) of the project. The Dockerfile is used to build an image. If **Dockerfile Address** is not specified, the system searches for the Dockerfile in the root directory of the project by default. If the Dockerfile does not exist in the root directory, the system automatically generates the Dockerfile based on the selected operating environment.

Parameter	Description
*Component Version	Version number of a component. ● Automatically-generated: Click **Generate**. By default, the version number is the timestamp when you click **Generate**. The format is yyyy.mmdd.hhmms, where **s** is the ones place of the second in the timestamp. For example, if the timestamp is 2022.1214.172318, the version number is 2022.1214.17238. ● You can also customize the version number in the format of A.B.C, or A.B.C.D. A. B, C, and D are natural numbers, for example, 1.0.0 or 1.0.0.0. **NOTICE** – The customized version number must be unique and cannot be the same as any historical version number of the component.
Resources	The value is fixed to the configuration of the selected historical version and cannot be changed.
JVM Parameters	The value is fixed to the configuration of the selected historical version and cannot be changed. This parameter is available when the technology stack type is Java or Tomcat. It configures the memory size during Java code running.
Tomcat	The value is fixed to the configuration of the selected historical version and cannot be changed. This parameter is available when the technology stack type is Tomcat. It configures parameters such as the request path and port number of Tomcat.
Advanced Settings	The value is fixed to the configuration of the selected historical version and cannot be changed.
Dark Launch Policy	● **Traffic Ratio**: percentage of traffic directed to the new version. ● **Current Traffic Ratio**: percentage of traffic directed to the current version.
*First Batch of Dark Launch Instances	Number of instances for dark launch in the first batch. The value range is [1, Total number of instances – 1]. Total number of instances refers to the number of running instances of the component. For example, if there are 6 component instances and **First Batch of Dark Launch Instances** is set to **1**, 1 instance will be upgraded in the first batch.
Deployment Batch with Remaining Instances	Number of batches whose remaining instances will be upgraded. For example, if there are 6 component instances, **First Batch of Dark Launch Instances** is set to **1**, and **Deployment Batch with Remaining Instances** is set to **3**, there are 5 instances remaining to be deployed in 3 batches, and these 5 instances will be upgraded in the sequence 2:2:1.

Step 7 Click **Upgrade**.

In the **Deployment Records** area, you can view the deployment progress and wait until the deployment is complete.

----End

7.10 Setting the Access Mode of Components

This section will guide you on how to set the access mode of components. After the setting is successful, you can access the services provided by the component in the set mode.

You can only set the component access mode for components that are deployed in the Kubernetes environment and are in the **Running** state.

Procedure

Step 1 Log in to ServiceStage.

Step 2 Use either of the following methods to go to the **Access Mode** page.

- On the **Application Management** page, click the application to which the component belongs, and click the target component in **Component List**. In the left navigation pane, choose **Access Mode**.
- On the **Component Management** page, click the target component. In the left navigation pane, choose **Access Mode**.

Step 3 Click **Add services**. Set the following parameters. Parameters marked with an asterisk (*) are mandatory.

Parameter	Description
*Service Name	Service Name: Specify a service name. You can use the component name as the service name.
Access Type	Sets the service access mode. The options are as follows: ● **Intra-cluster access**: allows access from other services in the same cluster over TCP/UDP. ● **Intra-VPC access**: allows access from other services in the same VPC over TCP/UDP. ● **Public network access**: allows access from the Internet over TCP/UDP, including EIP.
Intra-VPC Load Balancing	This parameter is valid when **Access Mode** is set to **Intra-VPC access**.
*Access Type	● This parameter is valid when **Access Mode** is set to **Intra-VPC access** and **Intra-VPC load balancing** is enabled. ● This parameter is valid when **Access Mode** is set to **Public network access**.
Service Affinity	This parameter is valid when **Access Mode** is set to **Intra-VPC access** or **Public network access**.

Parameter	Description
*Port Mapping	Set the **protocol**, **container port**, and **access port** for accessing the service.

Step 4 Click **OK**.

----End

7.11 Changing the Component Access Domain Name

For components with enabled public network access and set access domain names, you can change the domain names after the components are deployed.

Prerequisites

- You can change the domain name only when the component is in the **Running** state.
- You have obtained the domain name from the domain name provider.
- You have obtained the elastic public IP address of the ELB bound to the component.

Procedure

Step 1 Log in to ServiceStage.

Step 2 Use either of the following methods to go to the **Access Mode** page.

- On the **Application Management** page, click the application to which the component belongs, and click the target component in **Component List**. In the left navigation pane, choose **Access Mode**.
- On the **Component Management** page, click the target component. In the left navigation pane, choose **Access Mode**.

Step 3 Click **Set domain**.

1. Enter the obtained **Domain Name**.
2. Enter the listening port number. Only components deployed in containers support listening port numbers.
3. (Optional) Enable **HTTPS**.
   - Click **Use existing** to select an existing certificate.
   - Click **Create new** to create a server certificate. For details, see **Creating a Certificate**.

----End

7.12 Configuring a Scaling Policy of a Component Instance

After scaling policies are configured, instances can be automatically added or deleted based on resource changes or a specified schedule. This reduces manual resource adjustment to cope with service changes and service peak, helping you save resources and labor costs.

- Graceful scaling-in

  You can configure graceful scale-in policies only for the components deployed in the Kubernetes environment.

  You can set a graceful scale-in time window to save important data before a component instance stops. The value ranges from 0 to 9999, in seconds. The default value is **30**. For example, if an application has two instances and only one instance will be kept after the scale-in operation, you can still perform certain operations on the instance to be stopped in the specified time window.

  You can also set the maximum number of unavailable instances allowed during the rolling upgrade every day.
- Manual scaling

  The number of instances will be increased or decreased immediately after the configuration is complete.
- Auto scaling–HPA

  Only CCE clusters of 1.15 or later support HPA.

  HPA is a built-in component of Kubernetes, which enables horizontal scaling of pods. It supports the application-level cooldown time window and scaling threshold functions based on the Kubernetes HPA.

Configuring a Graceful Scale-In Policy

Step 1 Log in to ServiceStage.

Step 2 Use either of the following methods to go to the **Scaling** page.

- On the **Application Management** page, click the application to which the component belongs, and click the target component in **Component List**. In the left navigation pane, choose **Scaling**.
- On the **Component Management** page, click the target component. In the left navigation pane, choose **Scaling**.

Step 3 On the **Scaling** page, configure a graceful scale-in policy.

Set **Graceful Time Window (s)**. Specifically, click ✎, enter a value, and click ✓.

----End

Configuring a Manual Scaling Policy

Step 1 Log in to ServiceStage.

Step 2 Use either of the following methods to go to the **Scaling** page.

- On the **Application Management** page, click the application to which the component belongs, and click the target component in **Component List**. In the left navigation pane, choose **Scaling**.
- On the **Component Management** page, click the target component. In the left navigation pane, choose **Scaling**.

Step 3 In the **Manual Scaling** area on the **Scaling** page, configure a manual scaling policy.

- For components deployed in the Kubernetes environment, perform the following operations:
  a. Click ✎ and change the number of instances.
  b. Click ✓ for the instance scaling to take effect.

----End

Configuring an HPA Policy

Step 1 Log in to ServiceStage.

Step 2 Use either of the following methods to go to the **Scaling** page.

- On the **Application Management** page, click the application to which the component belongs, and click the target component in **Component List**. In the left navigation pane, choose **Scaling**.
- On the **Component Management** page, click the target component. In the left navigation pane, choose **Scaling**.

Step 3 On the **Scaling** page, click ⬭ next to **Auto Scaling by HPA** to enable auto scaling policy configuration. The **Policy** page is displayed.

- If metrics-server has not been installed in the CCE cluster, go to **Step 4**.
- If metrics-server has been installed in the CCE cluster, go to **Step 6**.

Step 4 Click **Configure Now** to install the metrics-server add-on on the CCE console.

Install the metrics-server add-on for the CCE cluster. For details, see **metrics-server**.

Step 5 After the add-on is installed, return to the **Policy** page and click **refresh**.

Step 6 Configure the scaling policy.

1. Policy Name

   Enter a policy name. After an auto scaling policy is configured, its name cannot be changed.
2. Cooldown Period

   Enter a scale-out/scale-in cooldown period.

   The same scaling operation will not be triggered again within the specified period.
3. Pod Range

   Enter the minimum and maximum numbers of instances.

   After the policy is triggered, the workload pods are scaled within this range.
4. Trigger Condition

   You can change trigger condition on the GUI or by editing the YAML file.
   - GUI

     Set **Desired Value** and **Threshold** (scale-in and scale-out thresholds) of **CPU usage** and **Memory usage**.

After the policy is triggered, the number of instances to be scaled is calculated by rounding up the value of (Current CPU or memory usage/ Expected value x Number of running instances).

- Scale-in is triggered when the current CPU or memory usage is less than the scale-in threshold.
- Scale-out is triggered when the current CPU or memory usage is greater than the scale-out threshold.

– YAML

```
metrics:
  - type: Resource
    resource:
      name: cpu
      target:
        type: Utilization
        averageUtilization: 50
  - type: Resource
    resource:
      name: memory
      target:
        type: Utilization
        averageUtilization: 50
  - type: Pods
    pods:
      metric:
        name: packets-per-second
      target:
        type: AverageValue
        averageValue: 1k
  - type: Object
    object:
      metric:
        name: requests-per-second
      describedObject:
        apiVersion: networking.k8s.io/v1beta1
        kind: Ingress
        name: main-route
      target:
        type: Value
        value: 10k
```

As shown in the preceding example, in addition to using the CPU and memory usage as metrics, you can use the YAML format to customize metric parameters and support more metrics such as pods, object, and external.

NOTE

To configure custom metric parameters by using **YAML**, ensure that the prometheus add-on has been installed for the CCE cluster.

Install the prometheus add-on for the CCE cluster. For details, see **prometheus**.

Step 7 Click **OK**.

NOTE

After the HPA policy is configured, you can perform the following operations based on service requirements:

- **Modifying an HPA Policy**
- **Viewing the Running Status of the HPA Policy**
- **Deleting an HPA Policy**

----End

Modifying an HPA Policy

You can edit an existing HPA policy and reconfigure policy parameters.

Step 1 Log in to ServiceStage.

Step 2 Use either of the following methods to go to the **Scaling** page.

- On the **Application Management** page, click the application to which the component belongs, and click the target component in **Component List**. In the left navigation pane, choose **Scaling**.
- On the **Component Management** page, click the target component. In the left navigation pane, choose **Scaling**.

Step 3 On the **Scaling** page, choose **Policy**, click **Edit Policy**, and reconfigure parameters.

1. Cooldown Period

   Change the scale-out/scale-in cooldown period.
2. Pod Range

   Change the minimum and maximum numbers of instances.
3. Trigger Condition

   You can change trigger condition on the GUI or by editing the YAML file.
   - GUI

     Change **Desired Value** and **Threshold** (scale-in and scale-out thresholds) of **CPU usage** and **Memory usage**.
   - YAML

     You can use the YAML format to customize metric parameters and support more metrics such as pods, objects, and external.

     NOTE

     To configure custom metric parameters by using **YAML**, ensure that the prometheus add-on has been installed for the CCE cluster.

     Install the prometheus add-on for the CCE cluster. For details, see **prometheus**.

Step 4 Click **OK**.

----End

Viewing the Running Status of the HPA Policy

ServiceStage allows you to view the running status and events of a configured HPA policy.

Step 1 Log in to ServiceStage.

Step 2 Use either of the following methods to go to the **Scaling** page.

- On the **Application Management** page, click the application to which the component belongs, and click the target component in **Component List**. In the left navigation pane, choose **Scaling**.
- On the **Component Management** page, click the target component. In the left navigation pane, choose **Scaling**.

Step 3 On the **Scaling** page:

- Click the **Status** tab to view the policy running status.
- Click the **Event** tab to view events that occur during policy running.

----End

Deleting an HPA Policy

You can delete an HPA policy that is no longer used.

NOTICE

Deleted policies cannot be recovered. Exercise caution when performing this operation.

Step 1 Log in to ServiceStage.

Step 2 Use either of the following methods to go to the **Scaling** page.

- On the **Application Management** page, click the application to which the component belongs, and click the target component in **Component List**. In the left navigation pane, choose **Scaling**.
- On the **Component Management** page, click the target component. In the left navigation pane, choose **Scaling**.

Step 3 On the **Scaling** page, click on the right of **Auto Scaling by HPA**.

Step 4 Click **OK**.

----End

7.13 Component O&M

7.13.1 Viewing the Component Running Metrics

After a component is created and deployed, you can go to the Component **Metric Graph** page to view the statistical results of component running metrics.

Procedure

Step 1 Log in to ServiceStage.

Step 2 Use either of the following methods to go to the **Metric Graph** page.

- On the **Application Management** page, click the application to which the component belongs, and click the target component in **Component List**. In the navigation pane on the left, go to **O&M** > **Monitoring Overview**.
- On the **Component Management** page, click the name of the target component, and choose **O&M** > **Monitoring Overview** in the navigation pane on the left.

Step 3 On the **Metric Graph** page, view the statistics of component running metrics in the last hour at an interval of 1 minute.

- Click on the component running metric page for which you want to suspend statistics collection.
- Click on the component running metric page to continue collecting statistics on the running metric.

----End

7.13.2 Customizing Component Running Metrics

If the deployment environment is Kubernetes, after creating and deploying a component, you can go to the component **Metric Graph** page to customize the component running metrics to be viewed.

Procedure

Step 1 Log in to ServiceStage.

Step 2 Use either of the following methods to go to the **Metric Graph** page.

- On the **Application Management** page, click the application to which the component belongs, and click the target component in **Component List**. In the navigation pane on the left, go to **O&M** > **Monitoring Overview**.
- On the **Component Management** page, click the name of the target component, and choose **O&M** > **Monitoring Overview** in the navigation pane on the left.

Step 3 On the **Metric Graph** page, click **Set metric graph**.

- Select the system metrics to be viewed and select the **Statistical method**.
- Deselect the system metrics that you do not need to view.

  You can click **Clear** next to the **Selected objects** to clear all selected system metrics.

Step 4 Click **OK**.

----End

7.13.3 Managing Component Running Logs

ServiceStage supports viewing, searching for, and exporting logs to locate and rectify faults that occur during component running.

Procedure

Step 1 Log in to ServiceStage.

Step 2 Use either of the following methods to go to the **Logs** page.

- On the **Application Management** page, click the application to which the component belongs, and click the target component in **Component List**. In the left navigation pane, choose **O&M Configurations** > **Logs**.
- On the **Component Management** page, click the target component. In the left navigation pane, choose **O&M Configurations** > **Logs**.

Step 3 On the **Logs** page, manage component run logs by referring to the following table.

Operation	Description
View Logs	1. Select the instance whose logs you want to view. 2. Select the log file you want to view. 3. Select the time range of the logs you want to view. If you select **Custom**, the time range cannot exceed 30 days. 4. Enter keywords in the search box and click 🔍 to view details about the specified logs.
Export Logs	1. Click **Last Records**. 2. Select the number of log records to be exported. 3. Open the **log.txt** file exported locally and view the exported log records.
View AOM Logs	Click **View AOM Logs**. You can view the component run logs on the AOM console. For details, see **Viewing Log Files**.

----End

7.13.4 Configuring Alarm Thresholds for Resource Monitoring

When a component is deployed in the Kubernetes environment, if you need to monitor some resources and respond to exceptions in a timely manner, you can create threshold rules for metrics of these key resources, so that you can find and handle exceptions in time.

- If the metric meets the threshold conditions within a specified period, the system sends a threshold alarm.
- If no metric is reported within a specified period, the system sends a data insufficiency event.
- If you cannot query the change information about the threshold rule status on the ServiceStage console due to non-business hours or business trips, you can enable the notification function to send the change information to related personnel through SMS messages or emails.

Procedure

Step 1 Log in to ServiceStage.

Step 2 Use either of the following methods to go to the **Threshold Alarms** page.

- On the **Application Management** page, click the application to which the component belongs, and click the target component in **Component List**. In the left navigation pane, choose **O&M Configurations** > **Threshold Alarms**.
- On the **Component Management** page, click the target component. In the left navigation pane, choose **O&M Configurations** > **Threshold Alarms**.

Step 3 Click **Set Threshold Rule** and set threshold rule parameters by referring to **Table 7-4**. Parameters marked with an asterisk (*) are mandatory.

Table 7-4 Threshold rule parameters

Parameter	Description
*Threshold Name	Name of the threshold rule to be added. **NOTE** The name must be unique and cannot be modified once specified.
Description	Description about the threshold rule.
Statistic Method	Method used to measure metrics.
Statistical Periods	Interval at which metric data is collected.
Metric	Select the metrics to be monitored.
*Threshold Condition	Trigger of a threshold alarm. A threshold condition consists of two parts: operators (≥, ≤, >, and <) and threshold value. For example, if this parameter is set to **≥ 80**, the system generates a threshold alarm when the metric is greater than or equal to 80.
Consecutive Periods	When the metric meets the threshold condition for a specified number of consecutive periods, a threshold alarm will be generated.
Alarm Severity	Severity of the threshold alarm.
Send Notifications	Whether to send notifications. ● If you select **Yes** (recommended), the system sends an email or SMS message to the user. ● If you select **No**, the system does not send an email or SMS message to the user.
*Topic Name	If you select **Yes** for **Send Notifications**, select a topic and click ▲. For details about how to create a topic, see **Creating a Topic**.
*Trigger Condition	Trigger condition for sending a notification when **Send Notification** is set to **Yes**. ● **An alarm occurred**: When a threshold alarm is generated, the system sends a notification to a specified user by email or SMS message. ● **The alarm is cleared**: When the alarm is cleared, the system sends a notification to a specified user by email or SMS message. ● **Insufficient**: When no metric is reported, the system sends a notification to a specified user by email or SMS message.

Step 4 Click **OK**.

----End

Follow-Up Operations

After a threshold rule is created, you can manage threshold alarms by referring to **Table 7-5**.

Table 7-5 Operations related to threshold alarm management

Operation	Description
Modify a Threshold Alarm	When you find that the current threshold rule is not properly set, you can perform the following operations to modify the threshold rule to better meet your service requirements. 1. Click **Modify** in the **Operation** column of the threshold alarm list. 2. On the **Modify Threshold Rule** page, modify the parameters of the threshold rule as prompted. 3. Click **Modify**.
Delete a Threshold Alarm	When you find that the current threshold rule is no longer needed, you can perform the following operations to delete the threshold rule to release more threshold rule resources. 1. Delete one or multiple threshold rules. ● To delete a single threshold, click **Delete** in the **Operation** column of the threshold rule list. ● To delete threshold rules in bathes, select one or more threshold rules and click **Delete** on the upper part of the page. 2. In the displayed dialog box, click **OK**.
Search for Threshold Alarms	1. Select a time segment from the drop-down list. 2. Enter the keyword of the alarm name or description in the search box on the upper right corner of the page. 3. Click 🔍 or press **Enter**. Alternatively, click **Advanced Search**, set the search criteria, and click **Search**.
View Threshold-Crossing Alarms	If the metric meets the threshold conditions within a specified period, the system sends a threshold alarm. View the alarm in the threshold alarm list.
View Alarm History	Click **History** in the **Operation** column of the threshold rule list to view historical alarms.

Operation	Description
Check the data insufficiency event.	If no metric is reported within a specified period, the system sends a data insufficiency event. You can view the event on the **Event** page. For details, see **7.13.5 Viewing the Component Running Events**.

7.13.5 Viewing the Component Running Events

If a component is deployed in the Kubernetes environment, you can view events that occur during component running to locate and rectify faults that occur during component running.

Procedure

Step 1 Log in to ServiceStage.

Step 2 Use either of the following methods to go to the **Event** page.

- On the **Application Management** page, click the application to which the component belongs, and click the target component in **Component List**. In the navigation pane on the left, go to **O&M** > **Event**.
- On the **Component Management** page, click the name of the target component, and choose **O&M** > **Event** in the navigation pane on the left.

Step 3 On the **Event** page, view the component running events.

- You can select the query time to view the component running events within a specified time range.
- You can enter an event keyword to search for and view specific component running events.

----End

7.14 Viewing the Component Running Environment

After a component is successfully deployed, you can view the resources (such as CCE clusters and microservice engines) on which the component depends and the resource status and usage on the component **Infrastructure** page.

Procedure

Step 1 Log in to ServiceStage.

Step 2 Use either of the following methods to go to the **Infrastructure** page.

- On the **Application Management** page, click the application to which the component belongs, and click the target component in **Component List**. In the left navigation pane, choose **Infrastructure**.
- On the **Component Management** page, click the target component. In the left navigation pane, choose **Infrastructure**.

Step 3 On the component **Infrastructure** page, select component running resources and view the resource status and usage.

----End

7.15 Component Instance Start and Stop

After a component is successfully deployed, you can restart or stop the component as required.

Procedure

Step 1 Log in to ServiceStage.

Step 2 Use either of the following methods to go to the **Overview** page.

- On the **Application Management** page, click the application to which the target component belongs, and click the component in **Component List**.
- On the **Component Management** page, click the target component.

Step 3 Start or stop a component.

- Click **Stop** to stop an application component in the **Running** or **Not ready** state.
- Click **Start** to start an application component in the **Stopped** state.
- Click **Restart** to restart an application component in the **Running** or **Not ready** state.

----End

7.16 Deleting a Component

This topic describes how to delete a component that is no longer used.

NOTICE

Deleted application components cannot be restored. Exercise caution when performing this operation.

Procedure

Step 1 Log in to ServiceStage.

Step 2 On the **Application Management** page, click the application to which the component belongs.

Step 3 In **Component List**:

- Single deletion: Locate the component to be deleted and choose **Delete** in the **Operation** column.
- Batch deletion: Select the components to be deleted and click **Bulk Delete**.

Step 4 In the displayed dialog box, click **OK**.

----End

7.17 Component Advanced Setting

7.17.1 Setting Component Environment Variables

Environment variables are set in the container running environment and can be modified after component deployment, ensuring the flexibility of applications.

Environment variables set for an application component are local environment variables and take effect only for this application component.

If you add an application environment variable to the application where the component is located and the name of the application environment variable is the same as that of the component environment variable in the application, the application environment variable is shielded by the component environment variable and does not take effect for the component. For details about how to add application environment variables, see **6.3 Managing Application Environment Variables**.

This topic describes how to configure component environment variables in different deployment modes during component deployment. For details about the component deployment mode, see **Deploying a Component**.

Deploying a Component Using CCE

During component deployment, add environment variables on the **Advanced Settings** page by referring to the following steps.

Step 1 Choose **Advanced Settings** > **Component Configuration**.

Step 2 Add environment variables by referring to **Table 7-6**.

Currently, environment variables can be added using any of the following methods:

Table 7-6 Environment variable types

Environment Variable Type	Procedure
Add manually	1. Click **Add Environment Variable** and select **Add manually**. 2. Set **Name** and **Variable/Variable Reference** to add an environment variable.

Environment Variable Type	Procedure
Import from secret	1. Create a secret. For details, see **Creating a Secret**. 2. Click **Add Environment Variable** and select **Add from secret**. 3. Set **Name**. 4. Select a secret from the **Variable/Variable Reference** drop-down list.
Import from configuration items	1. Create a configuration item. For details, see **Creating a Configuration Item**. 2. Click **Add Environment Variable** and select **Add from ConfigMap**. 3. Set **Name**. 4. Select a configuration item from the **Variable/Variable Reference** drop-down list.
Import from a file	Click **Import** and select a local configuration file. The imported file must be a key-value pair mapping file in JSON or YAML format. For example: {"key1":"value1","key2":"value2"}

----End

7.17.2 Configuring the Lifecycle of a Component

For container-deployed components, ServiceStage provides callback functions for the lifecycle management of applications. For example, if you want an application component to perform a certain operation before stopping, you can register a hook function.

ServiceStage provides the following lifecycle callback functions:

- Startup command: used to start a container.
- Post-start processing: triggered after an application is started.
- Pre-stop processing: triggered before an application is stopped.

Procedure

Step 1 During component deployment, choose **Advanced Settings** > **Deployment Configuration** in the **Advanced Settings** area.

Step 2 Click **Startup Command** to set **Command** and **Parameter** for the container.

A Docker image has metadata that stores image information. If no **Lifecycle** command or parameter is set, the container runs the default command and parameter provided during image creation. The Docker defines the default command and parameter as **CMD** and **Entrypoint**. For details about the two fields, see ***Entrypoint Description*** and ***CMD Description***.

If the running command and parameter of the application are set during application component deployment, the default **Entrypoint** and **CMD** will be overwritten during image building. **Table 7-7** describes the rules.

Table 7-7 Startup command parameters

Image Entrypoint	Image CMD	Application Running Command	Application Running Parameter	Final Execution
[touch]	[/root/test]	Not set	Not set	[touch /root/ test]
[touch]	[/root/test]	[mkdir]	Not set	[mkdir]
[touch]	[/root/test]	Not set	[/opt/test]	[touch /opt/ test]
[touch]	[/root/test]	[mkdir]	[/opt/test]	[mkdir /opt/ test]

Step 3 Click **Lifecycle** and set **Post-Start** and **Pre-Stop** parameters. **Table 7-8** describes the parameters. Select one of the parameters.

Table 7-8 Container lifecycle parameters

Paramet er	Description

CLI mode	Command to be executed in the component instance. The command format is **Command** *Args[1] Args[2]…*. **Command** is a system command or a user-defined executable program. If no path is specified, an executable program in the default path will be selected. If multiple commands need to be executed, write the commands into a script for execution.
	For example, the following commands need to be executed:
	`exec:`
	`  command:`
	`  - /install.sh`
	`  - install_agent`
	Write **/install.sh install_agent** in the script.
	This command indicates that the agent will be installed after the component is deployed.

HTTP request mode	HTTP call request. The related parameters are described as follows: ● **Path**: (optional) URL of a request. ● **Port**: (mandatory) request port. ● **Host Address**: (optional) IP address of the request. The default value is the IP address of the node where the application is located.

----End

7.17.3 Configuring Data Storage

Container storage is a component that provides storage for applications. Multiple types of storage are supported. An application component can use any amount of storage.

Components deployed using CCE support data storage settings.

Scenario

Table 7-9 Storage scenarios

Storage Type	Scenario
EVS Disks	EVS supports three specifications: common I/O, high I/O, and ultra-high I/O. ● Common I/O: The backend storage is provided by the Serial Advanced Technology Attachment (SATA) storage media. Common I/O is applicable to scenarios where large capacity is needed but high read/write rate is not required, and the volume of transactions is low. Examples include development testing and enterprise office applications. ● High I/O: The backend storage is provided by the Serial Attached SCSI (SAS) storage media. High I/O is applicable to scenarios where relatively high performance, high read/write rate, and real-time data storage are required. Examples include creating file systems and sharing distributed files. ● Ultra-high I/O: The backend storage is provided by the Solid-State Drive (SSD) storage media. Ultra-high I/O is applicable to scenarios where high performance, high read/write rate, and data-intensive applications are required. Examples include NoSQL and data warehouse (such as Oracle RAC and SAP HANA).
SFS File Systems	SFS applies to a wide range of scenarios, including media processing, content management, big data, and workload analysis.
OBS Buckets	● Standard OBS buckets: This type of OBS buckets applies to scenarios where a large number of hotspot files or small-sized files need to be accessed frequently (multiple times per month on average) and data can be quickly obtained. For example, cloud applications, data analysis, content analysis, and hotspot objects. ● Infrequent access OBS buckets: This type of OBS buckets applies to scenarios where data is not frequently accessed (less than 12 times per year on average) but fast access response is required. For example, static website hosting, backup/active archiving, storage resource pools or backup storage for cloud services.

Storage Type	Scenario
HostPath	The file directory of the host where the application component is located is mounted to the specified mounting point of the application. If the application component needs to access **/etc/hosts**, use **HostPath** to map **/etc/hosts**. **NOTICE** Do not mount the file directory to a system directory such as **/** or **/var/run**. Otherwise, an exception occurs. An empty directory is recommended. If the directory is not empty, ensure that the directory does not contain any file that affects component instance startup. Otherwise, the files will be replaced, causing startup exceptions.
EmptyDir	Used for temporary storage. The lifecycle of temporary storage is the same as that of an application component instance. When an application instance disappears, **EmptyDir** will be deleted and the data is permanently lost.
ConfigMap	Keys in a configuration item are mapped to an application so that configuration files can be mounted to the specified application component directory.
Secret	Sensitive information such as application authentication and application keys is stored in a secret, and the secret is mounted to a specified path of the application component.

EVS Disks

Step 1 During component deployment, choose **Advanced Settings** > **Deployment Configuration** in the **Advanced Settings** area.

Step 2 Choose **Data Storage** > **Cloud Storage** > **Add Cloud Storage** and set parameters by referring to **Table 7-10**.

Table 7-10 EVS disks

Parameter	Description
Storage Type	Select **EVS disk**. The method of using an EVS disk is the same as that of using a traditional disk. However, EVS disks have higher data reliability and I/O throughput and are easier to use. They apply to file systems, databases, or other system software or workloads that require block storage devices.

Parameter	Description
Storage Allocation Mode	● **Manual** Select a created storage. ● **Automatic** A storage is created automatically. You need to enter the storage capacity. 1. If **Storage Class** is set to **EVS Disk**, select an AZ for creating the EVS disk first. 2. Select a storage sub-type. High I/O: EVS disks that have high I/O and use SAS. Common I/O: EVS disks that use SATA. Ultra-high I/O: EVS disks that have ultra-high I/O and use SSD. 3. Enter the storage capacity, in the unit of GB. Ensure that the storage capacity quota is not exceeded; otherwise, creation will fail.
Add Docker Mounting	1. Set **Sub-path** and **Container Path** to the path to which the data volume is mounted. **NOTICE** – Do not mount a data volume to a system directory such as **/** or **/var/run**. Otherwise, an exception occurs. An empty directory is recommended. If the directory is not empty, ensure that the directory does not contain any file that affects application startup. Otherwise, the files will be replaced, causing application startup exceptions. As a result, the application fails to be created. – When the data volume is mounted to a high-risk directory, you are advised to use a low-permission account to start the application; otherwise, high-risk files on the host may be damaged. 2. Set **Permission**. – Read-only: allows you only to read data volumes in the application path. – Read/Write: allows you to modify data volumes in the application path. To prevent data loss, newly written data will not be migrated during application migration.

Step 3 Click **OK**.

----End

SFS File Systems

Step 1 During component deployment, choose **Advanced Settings** > **Deployment Configuration** in the **Advanced Settings** area.

Step 2 Choose **Data Storage** > **Cloud Storage** > **Add Cloud Storage** and set parameters by referring to **Table 7-11**.

Table 7-11 SFS file systems

Parameter	Description
Storage Type	Select **SFS**. SFS applies to a wide range of scenarios, including media processing, content management, big data, and application analysis.
Storage Allocation Mode	● **Manual** Select a created storage. ● **Automatic** A storage is created automatically. You need to enter the storage capacity. 1. Select a storage sub-type. Set the sub-type to NFS. 2. Enter the storage capacity, in the unit of GB. Ensure that the storage capacity quota is not exceeded; otherwise, creation will fail.
Add Docker Mounting	1. Set **Sub-path** and **Container Path** to the path to which the data volume is mounted. NOTICE – Do not mount a data volume to a system directory such as **/** or **/var/run**. Otherwise, an exception occurs. An empty directory is recommended. If the directory is not empty, ensure that the directory does not contain any file that affects application startup. Otherwise, the files will be replaced, causing application startup exceptions. As a result, the application fails to be created. – When the data volume is mounted to a high-risk directory, you are advised to use a low-permission account to start the application; otherwise, high-risk files on the host may be damaged. 2. Set **Permission**. – Read-only: allows you only to read data volumes in the application path. – Read/Write: allows you to modify data volumes in the application path. To prevent data loss, newly written data will not be migrated during application migration.

Step 3 Click **OK**.

----End

OBS Buckets

Step 1 During component deployment, choose **Advanced Settings** > **Deployment Configuration** in the **Advanced Settings** area.

Step 2 Choose **Data Storage** > **Cloud Storage** > **Add Cloud Storage** and set parameters by referring to Table 7-12.

Table 7-12 OBS buckets

Parameter	Description
Storage Type	Select **OBS**. Standard and Infrequent Access OBS classes are supported. OBS buckets apply to scenarios such as big data analytics, cloud native application data, static website hosting, and backup/active archiving.
Storage Allocation Mode	● **Manual** Select a created storage. ● **Automatic** 1. Set **Secret**. **Namespace** is the namespace of the container where the component instance is deployed when **creating and deploying a component**. It cannot be changed. Click **Use Existing Secret** to select the secret in the namespace of the container where the component instance is located. You can also create a secret: Enter a new secret name, click **Add Key File**, and upload the obtained local secret file. 2. Select a storage sub-type. You can select **Standard** or **Infrequent Access**.
Add Docker Mounting	1. Set **Sub-path** and **Container Path** to the path to which the data volume is mounted. NOTICE – Do not mount a data volume to a system directory such as **/** or **/var/run**. Otherwise, an exception occurs. An empty directory is recommended. If the directory is not empty, ensure that the directory does not contain any file that affects application startup. Otherwise, the files will be replaced, causing application startup exceptions. As a result, the application fails to be created. – When the data volume is mounted to a high-risk directory, you are advised to use a low-permission account to start the application; otherwise, high-risk files on the host may be damaged. 2. Set **Permission**. – Read-only: allows you only to read data volumes in the application path. – Read/Write: allows you to modify data volumes in the application path. To prevent data loss, newly written data will not be migrated during application migration.

Step 3 Click **OK**.

----End

HostPath

The file or directory of the host is mounted to the component.

Step 1 During component deployment, choose **Advanced Settings** > **Deployment Configuration** in the **Advanced Settings** area.

Step 2 Choose **Data Storage** > **Local Disk** > **Add Local Disk** and set parameters by referring to **Table 7-13**.

Table 7-13 HostPath

Parameter	Description
Local Disk Type	Select **HostPath**.
Host Path	Enter the host path, for example, **/etc/hosts**.
Docker Mounting	1. Set **Sub-path** and **Container Path** to the path to which the data volume is mounted. **NOTICE** – Do not mount a data volume to a system directory such as **/** or **/var/run**. Otherwise, an exception occurs. An empty directory is recommended. If the directory is not empty, ensure that the directory does not contain any file that affects application startup. Otherwise, the files will be replaced, causing application startup exceptions. As a result, the application fails to be created. – When the data volume is mounted to a high-risk directory, you are advised to use a low-permission account to start the application; otherwise, high-risk files on the host may be damaged. 2. Set **Permission**. – Read-only: allows you only to read data volumes in the application path. – Read/Write: allows you to modify data volumes in the application path. To prevent data loss, newly written data will not be migrated during application migration.

Step 3 Click **OK**.

----End

EmptyDir

EmptyDir applies to temporary data storage, disaster recovery, and shared running. It will be deleted upon deletion or transfer of application component instances.

Step 1 During component deployment, choose **Advanced Settings** > **Deployment Configuration** in the **Advanced Settings** area.

Step 2 Choose **Data Storage** > **Local Disk** > **Add Local Disk** and set parameters by referring to **Table 7-14**.

Table 7-14 EmptyDir

Parameter	Description
Local Disk Type	Select **EmptyDir**.
Disk Media	● If you select **Memory**, the running speed is improved, but the storage capacity is limited by the memory size. This mode applies to a small amount of data with high requirements on reading and writing efficiency. ● If **Memory** is not selected, data is stored in disks, which is applicable to a large amount of data with low requirements on reading and writing efficiency.
Docker Mounting	1. Set **Sub-path** and **Container Path** to the path to which the data volume is mounted. **NOTICE** – Do not mount a data volume to a system directory such as **/** or **/var/run**. Otherwise, an exception occurs. An empty directory is recommended. If the directory is not empty, ensure that the directory does not contain any file that affects application startup. Otherwise, the files will be replaced, causing application startup exceptions. As a result, the application fails to be created. – When the data volume is mounted to a high-risk directory, you are advised to use a low-permission account to start the application; otherwise, high-risk files on the host may be damaged. 2. Set **Permission**. – Read-only: allows you only to read data volumes in the application path. – Read/Write: allows you to modify data volumes in the application path. To prevent data loss, newly written data will not be migrated during application migration.

Step 3 Click **OK**.

----End

ConfigMap

ServiceStage separates the application codes from configuration files. **ConfigMap** is used to process application component configuration parameters.

Step 1 During component deployment, choose **Advanced Settings** > **Deployment Configuration** in the **Advanced Settings** area.

Step 2 Choose **Data Storage** > **Local Disk** > **Add Local Disk** and set parameters by referring to **Table 7-15**.

Table 7-15 ConfigMap

Parameter	Description
Local Disk Type	Select **ConfigMap**.
Configuration Item	Select the desired configuration item name. Create a configuration item. For details, see **Creating a Configuration Item**.
Docker Mounting	1. Set **Sub-path** and **Container Path** to the path to which the data volume is mounted. **NOTICE** – Do not mount a data volume to a system directory such as **/** or **/var/run**. Otherwise, an exception occurs. An empty directory is recommended. If the directory is not empty, ensure that the directory does not contain any file that affects application startup. Otherwise, the files will be replaced, causing application startup exceptions. As a result, the application fails to be created. – When the data volume is mounted to a high-risk directory, you are advised to use a low-permission account to start the application; otherwise, high-risk files on the host may be damaged. 2. Set **Permission**. – Read-only: allows you only to read data volumes in the application path. – Read/Write: allows you to modify data volumes in the application path. To prevent data loss, newly written data will not be migrated during application migration.

Step 3 Click **OK**.

----End

Secret

The data in the secret is mounted to the specified application component. The content of the secret is user-defined.

Step 1 During component deployment, choose **Advanced Settings** > **Deployment Configuration** in the **Advanced Settings** area.

Step 2 Choose **Data Storage** > **Local Disk** > **Add Local Disk** and set parameters by referring to **Table 7-16**.

Table 7-16 Secret

Parameter	Description
Local Disk Type	Select **Secret**.

Parameter	Description
Secret Item	Select the desired secret name. For details about how to create a secret, see **Creating a Secret**.
Docker Mounting	1. Set **Sub-path** and **Container Path** to the path to which the data volume is mounted. **NOTICE** – Do not mount a data volume to a system directory such as **/** or **/var/run**. Otherwise, an exception occurs. An empty directory is recommended. If the directory is not empty, ensure that the directory does not contain any file that affects application startup. Otherwise, the files will be replaced, causing application startup exceptions. As a result, the application fails to be created. – When the data volume is mounted to a high-risk directory, you are advised to use a low-permission account to start the application; otherwise, high-risk files on the host may be damaged. 2. Set **Permission**. – Read-only: allows you only to read data volumes in the application path. – Read/Write: allows you to modify data volumes in the application path. To prevent data loss, newly written data will not be migrated during application migration.

Step 3 Click **OK**.

----End

7.17.4 Configuring a Distributed Cache

Traditional single-instance applications use local session management. Session contexts generated by user requests are stored in the process memory. After the load balancing module is added, multi-instance sessions need to be shared using distributed storage.

ServiceStage provides the out-of-the-box distributed session function. It uses the Distributed Cache Service as the session persistence layer. Without code modification, ServiceStage supports distributed session management for Tomcat applications, Node.js applications that use express-session, and PHP applications that use session handle.

During component deployment, you can bind the distributed cache when configuring **Advanced Settings**.

Prerequisites

A distributed cache has been created. For details, see **Creating a DCS Redis Instance**.

Procedure

Step 1 Choose **Advanced Settings** > **Distributed Cache**.

Step 2 Click **Bind Distributed Cache**.

Step 3 Select a distributed cache instance that has been bound in the environment.

If no distributed cache instance is bound to the environment, click **Add One**. On the **Edit Environment** page that is displayed, click **Add Optional Resource** to add created DCS resources to the environment.

Step 4 If the DCS instance must be accessed using a password, enter the access password.

Step 5 Click **OK**.

----End

7.17.5 Configuring Relational Databases

To store application data permanently, you need to use Relational Database Service (RDS). Based on the cloud computing platform, ServiceStage provides RDS for MySQL which is reliable, scalable, easy to manage, and ready for use. RDS for MySQL enables you to easily set and scale relational databases on the cloud. Using the RDS service, you can perform nearly all necessary tasks without programming. This service simplifies operation procedures and reduces routine O&M workloads, so that you can focus on application and service development.

During component deployment, you can bind relational databases in **Database**. The procedure is as follows:

Prerequisites

An RDS DB instance has been created. For details, see **Create a DB Instance**.

Procedure

Step 1 Choose **Advanced Settings** > **Cloud Database**.

Step 2 Click **Bind Cloud Database**.

Step 3 Select a cloud database instance that has been bound in the environment and click **OK**.

If no cloud database instance is bound to the environment, click **Add One**. On the **Edit Environment** page that is displayed, click **Add Optional Resource** to add created RDS resources to the environment.

Step 4 Set the parameters by referring to the following table. Parameters marked with an asterisk (*) are mandatory.

Parameter	Description
*Connection Type	Select a connection type. ● **JNDI**: standard Java connection mode. ● **Spring Cloud Connector**: Spring connection mode.
*Database Name	Enter a database name.

Parameter	Description
*Database Account	Enter a database account.
*Database Password	Enter the database password.

Step 5 Click **OK**.

----End

7.17.6 Configuring a Scheduling Policy of a Component Instance

Based on features of components deployed using CCE, ServiceStage divides application components into the minimum deployment instances. The application scheduler monitors application instance information in real time. When detecting that a new pod needs to be scheduled, the application scheduler calculates all remaining resources (compute, network resources, and middleware) in the cluster to obtain the most appropriate scheduling target node.

ServiceStage supports multiple scheduling algorithms, including affinity scheduling between applications and AZs, between applications and nodes, and between applications.

You can freely combine these policies to meet your requirements.

Affinity

If an application is not containerized, multiple components of the application may run on the same virtual machine, and processes communicate with each other.

However, during containerization splitting, containers are usually split by process. For example, service processes are stored in a container, monitoring log processing or local data is stored in another container, and there is an independent life cycle. If closely related container processes run on distant nodes, routing between them will be costly and slow.

Affinity: Containers are scheduled onto the nearest node. This makes routing paths between containers as short as possible, which in turn reduces network overhead.

Anti-affinity: Instances of the same application are spread across different nodes to achieve higher availability. Once a node is down, instances on other nodes are not affected.

- Application-AZ Affinity and Anti-Affinity
  - **Affinity with AZs**: Application components can be deployed in specific AZs.
  - **Non-affinity with AZs**: Application components cannot be deployed in specific AZs.
- Application-Node Affinity and Anti-Affinity

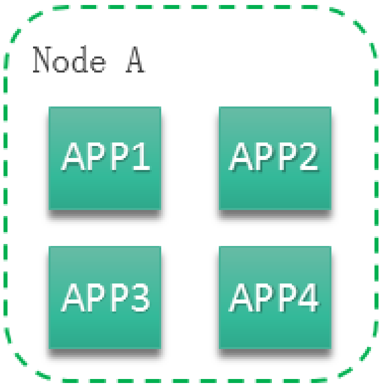
- **Affinity with Nodes**: Application components can be deployed on specific nodes.
- **Non-affinity with Nodes**: Application components cannot be deployed on specific nodes.

- Application Affinity

  It determines whether application components are deployed on the same node or different nodes.

  - **Affinity with Applications**: Application components are deployed on the same node. You can deploy application components based on service requirements. The nearest route between application components is used to reduce network consumption. For example, **Figure 7-7** shows affinity deployment, in which all applications are deployed on the same node.

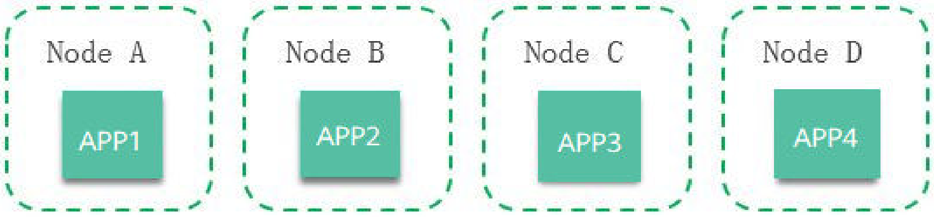
Figure 7-7 Application affinity



  - **Anti-affinity with Applications**: Different applications or multiple instances of the same application component are deployed on different nodes. Anti-affinity deployment for multiple instances of the same application reduces the impact of system breakdowns. Anti-affinity deployment for applications can prevent interference between the applications.

    As shown in **Figure 7-8**, four applications are deployed on four different nodes. The four applications are deployed in anti-affinity mode.

Figure 7-8 Application anti-affinity



Precautions

When setting application component-node affinity and application component-application component affinity, ensure that the affinity relationships are not mutually exclusive; otherwise, application deployment will fail. For example, application deployment will fail when the following conditions are met:

- Anti-affinity is configured for two application components APP 1 and APP 2. For example, APP 1 is deployed on node A and APP 2 is deployed on node B.
- When APP 3 is deployed on node C and goes online, affinity is configured between APP 3 and APP 2. As a result, affinity relationships are mutually exclusive, and APP 3 fails to be deployed.

Procedure

Step 1 Choose **Advanced Settings** > **Deployment Configuration**.

Step 2 On the **Scheduling Policies** tab page, set the component instance scheduling policy by referring to the following table.

Purpose	Procedure
Setting application component-AZ affinity	1. Click **Add Affinity Object**. 2. Set the object type to **AZ**, and select the desired AZ. 3. Click **OK**.
Setting application component-AZ anti-affinity	1. Click **Add Anti-affinity Object**. 2. Set the object type to **AZ**, and select the desired AZ. 3. Click **OK**.
Setting application component-node affinity	1. Click **Add Affinity Object**. 2. Set the object type to **Node**, and select the desired node. 3. Click **OK**.
Setting application component-node non-affinity	1. Click **Add Anti-affinity Object**. 2. Set the object type to **Node**, and select the desired node. 3. Click **OK**.
Setting application component-application component affinity	1. Click **Add Affinity Object**. 2. Set the object type to **Component**, and select the desired application components. 3. Click **OK**. The selected application components are deployed on the same node.
Setting application component-application component anti-affinity	1. Click **Add Anti-affinity Object**. 2. Set the object type to **Component**, and select the desired application components. 3. Click **OK**. The selected application components are deployed on different nodes.

----End

7.17.7 Configuring a Log Policy of an Application

ServiceStage allows you to configure application log policies for application components deployed in containers. You can view related logs on the AOM console.

You can configure log policies during component deployment. If no configuration is performed, the system collects standard application output logs by default.

Procedure

Step 1 Choose **Advanced Settings** > **O&M Monitoring**.

Step 2 On the **Log Collection** tab, click **Add Log Policy** and set the parameters by referring to the following table. Parameters marked with an asterisk (*) are mandatory.

Parameter	Description
Storage Type	Select a storage type. ● **HostPath**: Mount a host path to a specified container path. ● **Mounting Path**: Logs are exported only to the container path. You do not need to mount the host path.
*Host Path	This parameter is mandatory when **Storage Type** is set to **HostPath**. Enter the log storage path on the host.

Parameter	Description
*Docker Mounting	1. Set **Mounting Path**: Enter the application path to which the data volume is mounted. **NOTICE** – Do not mount a data volume to a system directory such as **/** or **/var/run**. Otherwise, an exception occurs. An empty directory is recommended. If the directory is not empty, ensure that the directory does not contain any file that affects application startup. Otherwise, the files will be replaced, causing application startup exceptions. As a result, the application fails to be created. – When the data volume is mounted to a high-risk directory, you are advised to use a low-permission account to start the application; otherwise, high-risk files on the host may be damaged. 2. Set **Extended Host Path**. This parameter is mandatory when **Storage Type** is set to **HostPath**. – **None**: No extended path is configured. – **PodUID**: Pod ID. – **PodName**: Pod name. – **PodUID/ContainerName**: Pod ID or container name. – **PodName/ContainerName**: Pod name or container name. 3. Set **Collection Path**. After specifying the collection path, you can specify the content to collect. The collection path can be specified if the ICAgent version of the collector is 5.12.22 or later. The following modes are supported: – If no collection path is specified, log files in **.log**, **.trace**, and **.out** formats will be collected from the current path by default. – If a collection path contains double asterisks (**), log files in **.log**, **.trace**, and **.out** formats will be collected from 5 levels of subdirectories. – If a collection path contains an asterisk (*), a fuzzy match is performed. Example: If the collection path is **/tmp/**/test*.log**, all **.log** files prefixed with **test** will be collected from **/tmp** and its 5 levels of subdirectories. 4. Set **Aging Period**. – **Hourly**: Log files are scanned every hour. If the size of a log file exceeds 20 MB, the system compresses the log file to a historical file, dumps the historical file to the directory where the log file is stored, and clears the original log file. – **Daily**: Log files are scanned once a day. If the size of a log file exceeds 20 MB, the system compresses the log file to a historical file, dumps the historical file to the directory where the log file is stored, and clears the original log file. – **Weekly**: Log files are scanned once a week. If the size of a log file exceeds 20 MB, the system compresses the log file

Parameter	Description
	to a historical file, dumps the historical file to the directory where the log file is stored, and clears the original log file. 5. (Optional) Click ⌄ to set **Log Format**. By default, the system collects and displays the logs printed by the program by line. If a complete log occupies multiple lines and you want to collect and display the entire log, you can set **Log Format** to enable multi-line logs. – **Single-line**: The system collects logs by line. – **Multi-line**: The system collects logs based on the configured matching rules. **Log Segmentation**: This parameter is mandatory when **Log Format** is set to **Multi-line**. It is used to match the beginning of each log. If you select **Log Time**, the time matching mode is used. If you select **Regular pattern**, the regular expression matching mode is used. **Time Wildcard**: Enter the time wildcard when **Log Segmentation** is set to **Log Time**. For example, if the time format of each log is YYYY-MM-DD hh:mm:ss, set the time wildcard to YYYY-MM-DD hh:mm:ss. **Regularization**: If **Log Segmentation** is set to **Regular pattern**, enter the regular expression based on the format of the beginning of each log.

NOTE

After the component configuration and deployment are complete, you can view run logs on the AOM console. For details, see **Viewing Log Files**.

----End

7.17.8 Configuring Custom Monitoring of a Component

ServiceStage allows you to obtain custom metrics when components are deployed using CCE. You can use this method to report custom component running metrics.

Precautions

- Currently, only **Gauge metrics** of Prometheus can be obtained.
- Before setting custom metric monitoring for an application component, you must understand **Prometheus** and provide the GET API for obtaining custom metric data in your application component so that ServiceStage can obtain custom metric data using this API.

Procedure

Step 1 Choose **Advanced Settings** > **O&M Monitoring**.

Step 2 On the **O&M Policy** tab, configure the custom monitoring by referring to the following table.

Parameter	Description	Mandatory
Report Path	URL provided by the exporter for ServiceStage to obtain custom metric data. Example: **/metrics**	Yes
Report Port	Port provided by the exporter for ServiceStage to obtain custom metric data. Example: **8080**	Yes
Monitoring Metrics	Name of the custom metric provided by the exporter. Example: **["cpu_usage","mem_usage"]** ● If this parameter is not set, ServiceStage collects data of all custom metrics. ● If you set this parameter, for example, to **["cpu_usage","mem_usage"]**, ServiceStage collects the data of the specified cpu_usage and mem_usage metrics.	No

NOTE

After the configuration and deployment are complete, you can view the monitoring metric data on the AOM console. For details, see **Metric Monitoring**.

----End

7.17.9 Configuring Health Check

Health check periodically checks health status during component running according to your needs.

ServiceStage provides the following health check methods:

- **Component Liveness Probe**: checks whether an application component exists. It is similar to the **ps** command that checks whether a process exists. If the liveness check of an application component fails, the cluster restarts the application component. If the liveness check is successful, no operation is executed.
- **Component Service Probe**: checks whether an application component is ready to process user requests. It may take a long time for some applications to start before they can provide services. This is because that they need to load disk data or rely on startup of an external module. In this case, the application process exists, but the application cannot provide services. This check method is useful in this scenario. If the application component readiness check fails, the cluster masks all requests sent to the application component. If the application component readiness check is successful, the application component can be accessed.

Health Check Modes

- HTTP request-based check

  This health check mode is applicable to application components that provide HTTP/HTTPS services. The cluster periodically sends an HTTP/HTTPS GET request to such application components. If the return code of the HTTP/HTTPS response is within 200–399, the check is successful. Otherwise, the check fails. In this health check mode, you must specify an application listening port and an HTTP/HTTPS request path.

  For example, if the application component provides the HTTP service, the port number is 80, the HTTP check path is **/health-check**, and the host address is **containerIP**, the cluster periodically initiates the following request to the application:

  ```
  GET http://containerIP:80/health-check
  ```

  NOTE

  If the host address is not set, the instance IP address is used by default.

- TCP port-based check

  For applications that provide a TCP communication service, the cluster periodically establishes a TCP connection to the application. If the connection is successful, the probe is successful. Otherwise, the probe fails. In this health check mode, you must specify an application listening port. For example, if you have a Nginx application component with service port 80, after you configure a TCP port-based check for the application component and specify port 80 for the check, the cluster periodically establishes a TCP connection with port 80 of the application component. If the connection is successful, the check is successful. Otherwise, the check fails.

- CLI-based check

  In this mode, you must specify an executable command in an application component. The cluster will periodically execute the command in the application component. If the command output is **0**, the health check is successful. Otherwise, the health check fails.

  The CLI mode can be used to replace the following modes:

  – TCP port-based check: Write a program script to connect to an application component port. If the connection is successful, the script returns **0**. Otherwise, the script returns **–1**.

  – HTTP request-based check: Write a program script to run the **wget** command for an application component.

    wget http://127.0.0.1:80/health-check

    Check the return code of the response. If the return code is within 200–399, the script returns **0**. Otherwise, the script returns **–1**.

NOTICE

- Put the program to be executed in the application component image so that the program can be executed.
- If the command to be executed is a shell script, add a script interpreter instead of specifying the script as the command. For example, if the script is **/data/scripts/health_check.sh**, you must specify **sh/data/scripts/health_check.sh** for command execution. The reason is that the cluster is not in the terminal environment when executing programs in an application component.

Common Parameter Description

Table 7-17 Common parameter description

Parameter	Description
Latency (s)	Check delay time. Unit: second. Set this parameter according to the normal startup time of services. For example, if this parameter is set to 30, the health check will be started 30 seconds after the application starts. The time is reserved for containerized services to start.
Timeout Period (s)	Timeout duration. Unit: second. If the time exceeds this value, the health check fails. For example, setting this parameter to 10 indicates that the health check timeout period is 10s. If the parameter is left blank or set to **0**, the default timeout time is 1s.

Procedure

Step 1 Choose **Advanced Settings** > **O&M Monitoring**.

Step 2 Click **Health Check**, and set health check parameters based on service requirements.

For details about common parameters, see **Table 7-17**.

----End

8 Deployment Source Management

8.1 Image Repository

8.1.1 Uploading an Image

After an organization is created, you can upload an image to it through the page or client.

- **Uploading an Image Through the Browser**: Upload an image to SWR through the page.
- **Uploading an Image Through the Client**: Upload an image to an image repository of SWR by running commands on the client.

Container repositories are used to easily store, deploy, and manage Docker images.

Prerequisites

- An organization has been created. For details, see **Creating an Organization**.
- The image has been saved as a .tar or .tar.gz file. For details, see **Creating an Image Package**.
- The image package is created using Docker 1.11.2 or later.
- If the image is uploaded through a client, the version of the container engine client to which the image is uploaded must be 1.11.2 or later.

Uploading an Image Through the Browser

NOTE

- A maximum of 10 files can be uploaded at a time. The size of a single file (including the decompressed files) cannot exceed 2 GB.
- The file name should be a string starting with a letter or digit, containing 255 characters at most, and including letters, digits, underscores (_), and hyphens (-).

Step 1 Log in to ServiceStage.

Step 2 Choose **Deployment Source Management** > **Image Repository**.

Step 3 On the **Image Repository** page, click **Upload Through SWR**.

Step 4 In the displayed dialog box, specify **Organization** to which the image is to be uploaded, click **Add File**, and select the image file to be uploaded.

NOTE

If you select multiple images to upload, the system uploads them one by one. Concurrent upload is not supported.

Step 5 In the displayed dialog box, click **Upload**.

If **Upload completed** is displayed, the image is successfully uploaded.

NOTE

If the image fails to be uploaded, the possible causes are as follows:

- The network is abnormal. In this case, check network connectivity.
- The HTTPS certificate has errors. Press **F12** to copy the URL that fails to be requested to the address bar of the browser, open the URL again, agree to continue the access, and return to the upload page to upload the certificate again.

----End

Uploading an Image Through the Client

NOTE

If you use the client to upload an image, each image layer cannot exceed 10 GB.

Step 1 Log in to ServiceStage.

Step 2 Choose **Deployment Source Management** > **Image Repository**.

Step 3 On the **Image Repository** page, click **Upload Through Client**.

Step 4 Upload the image as prompted.

----End

8.1.2 Managing Images

Obtaining an Image Pull Address

Step 1 Log in to ServiceStage.

Step 2 Choose **Deployment Source Management** > **Image Repository** > **My Images**.

Step 3 Select an organization from the drop-down list on the right of **Organization Management**.

Step 4 In the image repository list, click an image repository name to go to the details page.

Step 5 Click the **Image Tags** tab and obtain the command for pulling an image.

Click ⧉ on the right of the command of the image version to be downloaded to copy the command.

----End

Setting Image Repository Attributes

Step 1 Log in to ServiceStage.

Step 2 Choose **Deployment Source Management** > **Image Repository** > **My Images**.

Step 3 Select an organization from the drop-down list on the right of **Organization Management**.

Step 4 In the image repository list, click an image repository name to go to the details page.

Step 5 Click **Edit** in the upper-right corner. In the displayed dialog box, perform the following operations:

- Set **Sharing Type** to **Public** or **Private**.

  NOTE

  Public images can be downloaded and used by all users.

  - If your node and the image repository are in the same region, you can access the image repository over private networks.
  - If your node and the image repository are in different regions, the node must have access to public networks to pull images from the image repository.

- Set **Category** to set the repository category.
- Set **Description** to update the description of the image repository.

Step 6 Click **OK**.

----End

Setting Automatic Image Synchronization

If image synchronization is enabled, the latest images are automatically synchronized to image repositories in other regions. Only accounts and users with administrator permissions can configure automatic image synchronization.

NOTE

After you configure automatic image synchronization, image updates will also be synchronized to target repositories. However, images that were pushed to repositories before automatic image synchronization was enabled will not be automatically synchronized.

For details on how to synchronize images pushed before you set the automatic synchronization, see .

Step 1 Log in to ServiceStage and choose **Deployment Source Management** > **Image Repository** > **My Images**.

Step 2 Select an organization from the drop-down list on the right of **Organization Management**.

Step 3 In the image repository list, click an image repository name to go to the details page.

Step 4 Click **Set Image Synchronization** in the upper-right corner.

Step 5 In the displayed dialog box, click **Add**, set the following parameters, and click **OK** in the **Operation** column.

- **Target Region**: Select the target region for synchronization.
- **Target Organization**: Select the target organization for synchronization.
- **Overwrite Existing Image**: Select this option if you want to overwrite any nonidentical images that have the same name in the target organization. Deselect this option if you do not want any nonidentical images having the same name in the target organization to be overwritten and you want to receive a notification of the existence of such images.

Step 6 Click **OK**.

On the **Synchronization Records** tab of image details page, you can view the details of each synchronization task, including the start time, image tag, task status, type, duration, target region and organization, and task operator.

----End

Adding Image Permissions

To allow IAM users of your account to read, write, and manage a specific image, add the required permissions to the IAMusers on the details page of this image.

Step 1 Log in to ServiceStage and choose **Deployment Source Management** > **Image Repository** > **My Images**.

Step 2 Select an organization from the drop-down list on the right of **Organization Management**.

Step 3 In the image repository list, click an image repository name to go to the details page.

Step 4 Click the **Permission Management** tab, click **Add Permission**, select an IAM user, add the **Read**, **Write**, or **Manage** permission, and click **OK**.

Then, this IAM user has the corresponding permission.

----End

Deleting an Image

NOTICE

Deleted images cannot be recovered.

Step 1 Log in to ServiceStage and choose **Deployment Source Management** > **Image Repository** > **My Images**.

Step 2 Select an organization from the drop-down list on the right of **Organization Management**.

Step 3 In the image repository list, click an image repository name to go to the details page.

- Deleting an image repository

  Click **Delete** in the upper-right corner of the page and delete the image repository as prompted.
- Deleting an image tag

  In the **Operation** column of the target image tag, click **Delete** to delete the image tag as prompted.
- Deleting image tags in batches

  Select the target image tags, click **Delete** above the tag list, and delete the image tags as prompted.

----End

8.2 Organization Management

Overview

Organizations are used to isolate image repositories. With each organization being limited to one company or department, software can be managed in a centralized manner. A software name only needs to be unique within an organization. The same IAM user can join different organizations. Different permissions, namely read, write, and manage, can be assigned to different IAM users in the same account.

Creating an Organization

Step 1 Log in to ServiceStage and choose **Deployment Source Management** > **Organization**.

Step 2 Click **Create Organization**. On the displayed page, specify **Organization Name** and click **OK**.

----End

Adding Permissions

Grant permissions to users in an organization so that they can read, edit, and manage all images in the organization.

Only users with the **Management** permission can grant permissions.

User permissions include:

- **Read-only**: Users can only download software but cannot upload software.
- **Read/write**: Users can download software, upload software, and edit software attributes.
- **Management**: Users can download and upload software, delete software or versions, edit software attributes, grant permission, and share images.

Step 1 Log in to ServiceStage and choose **Deployment Source Management** > **Organization**.

Step 2 Click **Add Permission** on the right of an organization.

Step 3 In the displayed dialog box, specify **Permission** and click **OK**.

----End

Deleting an Organization

Step 1 Log in to ServiceStage and choose **Deployment Source Management** > **Organization**.

Step 2 Click **Delete** on the right of an organization.

Before deleting an organization, delete the image repositories of the organization.

For details about how to delete an image repository, see **Deleting an Image**.

Step 3 Click **OK**.

----End

9 Continuous Delivery

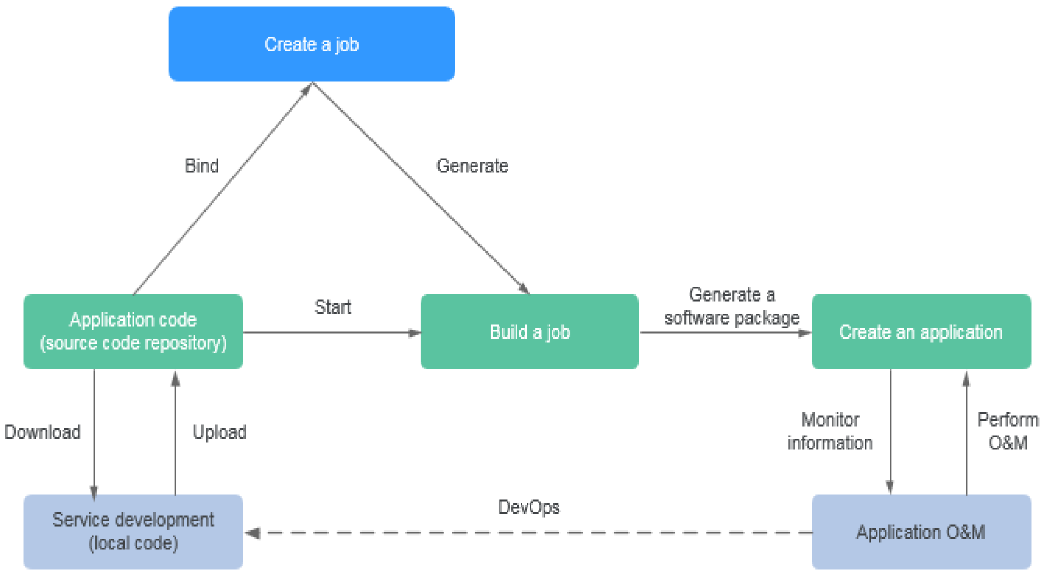
9.1 Overview

Continuous delivery provides functions such as project build and release.

Creating a Build Job

Based on the existing service code, you can create a build job, start the build job, package the service code, and archive the package to the deployment source. Then, you can use the package when deploying an application component.

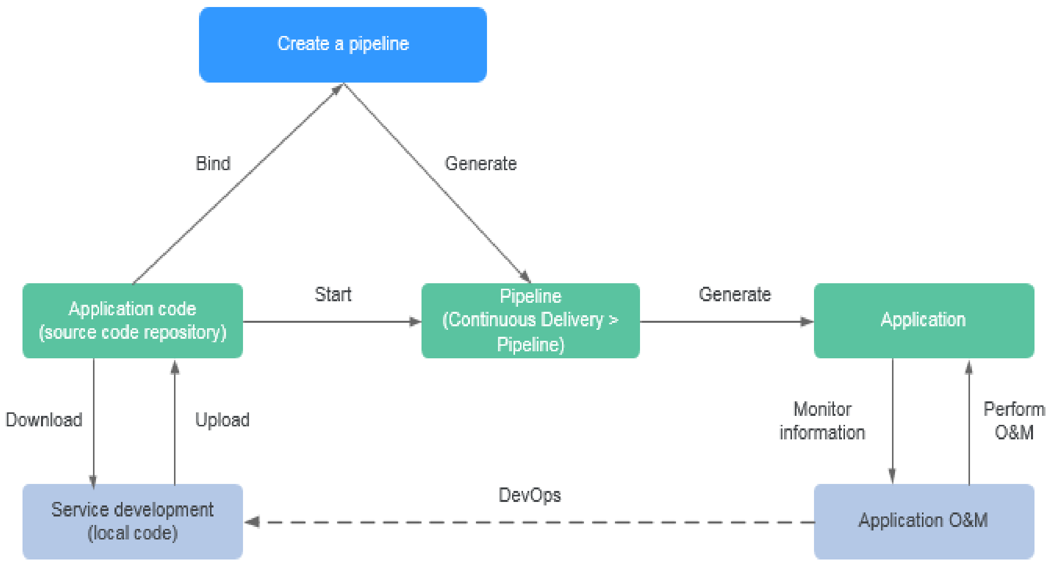
Figure 9-1 Creating a build job



Creating a Pipeline

Based on the existing service code, you can create a pipeline and then start the pipeline to complete service code building and deployment. Application O&M can also be completed on ServiceStage.

Figure 9-2 Creating a pipeline



9.2 Viewing Build Jobs

For components deployed in the Kubernetes environment, you can view the build records and logs of a specified build job in the build list to locate and rectify faults that occur during component deployment.

Procedure

Step 1 Log in to ServiceStage.

Step 2 Choose **Continuous Delivery** > **Build**.

Step 3 On the **Build** page, use either of the following methods to search for a build job:

- Select the CCE cluster where the component is deployed and the build job status, and select the specified build job in the build list.
- Search for the build job in the search box.

Step 4 Click the build task name. The build task details page is displayed.

- View **Basic Information** and **Build Record** of the build job.
- Click **View Log** next to a build record to view **Build Details**, **CodeCheck**, and **Log**.

  NOTE

  Only the Maven build project supports code check. Currently, the following code check plug-ins are supported: Checkstyle, FindBugs, and PMD.

----End

9.3 Creating a Source Code Job

The software package or image package can be generated with a few clicks in a build job. In this way, the entire process of source code pull, compilation, packaging, and archiving is automatically implemented.

- Images built in the x86-system jobs are ones of the x86 system.
- Images built in the Arm-system jobs are ones of the Arm system.

Prerequisites

1. A cluster has been created. For details, see **Creating a CCE Cluster**.

   NOTICE

   - The build job depends on the JDK, Golang, Maven, Gradle, Ant, or Node.js compilation tool preconfigured in the build container.
   - Different IAM users under the same account can perform operations on the same cluster. To cancel the build permission from a specific IAM user, set the **servicestage:assembling:create**, **servicestage:assembling:modify**, and **servicestage:assembling:delete** permissions to **Deny** by referring to **4.2 Creating a Custom Policy**.

2. An EIP has been bound to the build node. For details, see **Assigning an EIP and Binding It to an ECS**.

Procedure

Step 1 Log in to ServiceStage, choose **Continuous Delivery** > **Build**, and click **Create Source Code Job**.

Step 2 Configure basic information.

1. Enter **Name**.
2. Enter **Enterprise Project**.

   Enterprise projects let you manage cloud resources and users by project.

   It is available after you **create an enterprise project**.
3. (Optional) Enter **Description**.
4. Set **Code Source**.
   - Create authorization by referring to **9.7 Authorizing a Repository** and set the code source.
   - Click **Samples** and select a required sample.
5. Select a cluster from the **Cluster** drop-down list.
6. (Optional) Specify **Node Label** to deliver the build job to a fixed node based on the node label. For details about how to add a label, see **Adding a Node Label**.
7. Click **Next**.

Step 3 Select a build template.

- If you select **Maven**, **Ant**, **Gradle**, **Go**, or **Docker**, you can compile and archive binary packages or Docker images at the same time. Go to **Step 4**.
- If you select **Custom**, you can customize the build mode. Go to **Step 6**.

Step 4 Select an archive mode.

- **Not archived**: No Docker build job is added or archived.
- **Archive binary package**: No Docker build job is added and binary packages are archived.
- **Archive image compilation**: Docker build job is added and Docker images are archived.

Step 5 Set mandatory parameters.

To delete a parameter setting, click on the parameter setting page.

- Build parameters

  Compilation parameters are set with different values. For details about parameter description, click a text box or next to it.
- Image parameters

  On the page, enter **Job Name**, **Dockerfile Path**, **Image Name**, and **Image Tag**.
- Image archiving parameters

  On the page, enter **Job Name**, **Archive Image**, **Repository Organization**, and **Type** of the corresponding image to archive the image.
- Binary parameters

  On the page, set the following parameters.

Parameter	Description
Task Name	Job name.
Sharing Type	Repositories are classified into public repositories and private repositories. – Public repositories are isolated from each other. Tenants in the same system can resources. – Private repositories are isolated by tenants. Users under the current tenant share resources. Other tenants cannot access resources of the current tenant.
Repository Organization	Namespace of a repository.
Software Repository	Name of a software repository.
Name	Name of the archived software package after the build completes.

Parameter	Description
Software Package Version	Version of the archived software package.
Build Package Path	Address of the binary software package generated after the compilation and build are complete. For example, **./target/xxx.jar** in the Java project.

Step 6 (Optional) Click **Advanced Configuration** to set the environment.

To add multiple tasks, you can customize them in **Advanced Configuration**.

1. Click **Add Plug-in** in the corresponding stage on the left. The **Select Job Type** page is displayed.
2. Click **Select** of the target task type to add a task type. Then, configure task parameters in the right pane of the **Environment Configurations** page.

NOTICE

When the Build Common Cmd plug-in is added to the compilation process, pay attention to the following:

– Exercise caution when inputting sensitive information in the **echo**, **cat**, or **debug** command, or encrypt sensitive information to avoid information leakage.

– When **Language** is set to **Python** and **Python Framework Type** is set to a Python project that complies with the **WSGI** standard, you need to set **Main Python Module** and **Function of the Main Python Module**. The following is an example of the main Python module and main function:

  Main Python Module: If the entry point file of the Python project is **server.py**, the main module name is **server**.

  Function of the Main Python Module: If the application function name of the Python project entry point file **server.py** is **app=get_wsgi_application()**, the function name of the main module is **app**.

Step 7 Click **Build** to save the settings and start the build.

Click **Save** to save the settings (not to start the build).

----End

Follow-Up Operations

After an application component is successfully built, you can manage it on ServiceStage. For details, see **Deploying a Component**.

9.4 Creating a Package Job

The image package can be generated with a few clicks in a build job. In this way, the entire process of package obtainment, and image compilation and archiving is automatically implemented.

Prerequisites

1. A cluster has been created. For details, see **Creating a CCE Cluster**.

   NOTICE

   - The build job starts a build container on the cluster node to perform build-related operations. To ensure build security, you are advised to perform security hardening on CCE cluster nodes.
   - The build job depends on the JDK, Golang, Maven, Gradle, Ant, or Node.js compilation tool preconfigured in the build container.
   - Different IAM users under the same account can perform operations on the same cluster. To cancel the build permission from a specific IAM user, set the **servicestage:assembling:create**, **servicestage:assembling:modify**, and **servicestage:assembling:delete** permissions to **Deny** by referring to **4.2 Creating a Custom Policy**.

2. An EIP has been bound to the build node. For details, see **Assigning an EIP and Binding It to an ECS**.

Procedure

Step 1 Log in to ServiceStage, choose **Continuous Delivery** > **Build**, and click **Create Package Job**.

Step 2 Enter **Job Name**.

Step 3 Enter **Enterprise Project**.

Enterprise projects let you manage cloud resources and users by project.

It is available after you **create an enterprise project**.

Step 4 (Optional) Enter **Description**.

Step 5 Set **Package Source**.

The following upload modes are supported:

- Select the corresponding software package from OBS. Upload the software package to the OBS bucket in advance. For details, see **Uploading a File**.

Click **Select Software Package** and select the corresponding software package.

Step 6 Select a build type.

- System default

a. Select a basic image, which must be the same as the software package compilation language selected in **Step 5**.

b. Set **Basic Image Tag**.

- Custom Dockerfile

  Enter custom commands in the compilation box.

  NOTICE

  Exercise caution when inputting sensitive information in the **echo**, **cat**, or **debug** command, or encrypt sensitive information to avoid information leakage.

- Image

  Select a basic image, which must be the same as the software package compilation language selected in **Step 5**.

Step 7 Set **Image Class**.

- Public: This is a widely used standard image that contains an OS and pre-installed public applications and is visible to all users. You can configure the applications or software in the public image as needed.
- Private: A private image contains an OS or service data, pre-installed public applications, and private applications. It is available only to the user who created it.

Step 8 Set **Archived Image Address**.

Step 9 Select **Cluster**.

If you use the selected cluster to perform a build job, you can deliver the build job to a fixed node through node labels. For details about how to add a label, see **Adding a Node Label**.

Step 10 Click **Build Now** to start the build.

Click **Save** to save the settings (not to start the build).

----End

Follow-Up Operations

After an application component is successfully built, you can manage it on ServiceStage. For details, see **Deploying a Component**.

9.5 Maintaining Build Jobs

For components deployed in the Kubernetes environment, you can maintain build jobs in the build job list.

Maintenance

Table 9-1 Maintenance

Operation	Description
Editing a Build Job	See **Editing a Package Job** or **Editing a Source Code Job**.
Starting a Build Job	See **Starting a Build Job**.
Viewing Details/Build History	See **9.2 Viewing Build Jobs**.
Branch/Tag	See **Branch/Tag**. Source code jobs support this operation.
Deleting a Build Job	See **Deleting a Build Job**. User-created jobs support this operation.

Editing a Package Job

Step 1 Log in to ServiceStage.

Step 2 Choose **Continuous Delivery** > **Build**.

Step 3 On the **Build** page, use either of the following methods to search for a build job:

- Select the CCE cluster where the component is deployed and the build job status, and select the specified build job in the build list.
- Search for the build job created by the specified user in the search box.

Step 4 Click **More** > **Edit**. The build job configuration page is displayed.

Step 5 Enter a job name.

Step 6 (Optional) Enter the description.

Step 7 Set **Package Source**.

The following upload modes are supported:

Select the corresponding software package from OBS. Upload the software package to OBS. For details, see **Uploading a File**.

Step 8 Select a build type.

- System default
  a. Select the language of the basic image, which must be the same as that of the software package.
  b. Set **Basic Image Tag**.
     The build node can download basic images only when it can access the public network.

- Custom Dockerfile
  Enter custom commands in the compilation box.
- Image
  Set **Basic Image**.

Step 9 Set **Image Class**.

- **Public**: This is a widely used standard image that contains an OS and pre-installed public applications and is visible to all users. You can configure the applications or software in the public image as needed.
- **Private**: A private image contains an OS or service data, pre-installed public applications, and private applications. It is available only to the user who created it.

Step 10 Set **Archived Image Address**.

Step 11 Select **Cluster**.

Step 12 (Optional) Specify **Node Label** to deliver the build job to a fixed node based on the node label.

For details about how to add a label, see **Managing Node Labels**.

Step 13 Click **Build Now** to start the build.

Click **Save** to save the settings (not to start the build).

----End

Editing a Source Code Job

Step 1 Log in to ServiceStage.

Step 2 Choose **Continuous Delivery** > **Build**.

Step 3 On the **Build** page, use either of the following methods to search for a build job:

- Select **User created** and select the CCE cluster and build job status. Then, select the target build job in the build list.
- Search for the build job created by the specified user in the search box.

Step 4 Click **More** > **Edit**. The build job configuration page is displayed.

Step 5 Enter a job name.

Step 6 (Optional) Enter the description.

Step 7 Click **Modify** and set **Code Source**.

You need to create a repository authorization first. For details, see **9.7 Authorizing a Repository**.

Step 8 Select **Cluster**.

1. (Optional) Specify **Node Label** to deliver the build job to a fixed node based on the node label.
2. Click **Next**.

Step 9 Set the environment.

1. Edit a build template.

   Select **Maven**, **Ant**, **Gradle**, **Go**, **Docker**, or **Build Common Cmd**. You can compile and archive binary packages or Docker images at the same time.

2. Select an archive mode.
   - Publish Build Artifact: Binary package archive plug-in, archived to the SWR software repository.
   - Publish Build Image: Image archive plug-in, archived to the SWR image repository.

Step 10 Click **Build** to save the settings and start the build.

Click **Save** to save the settings (not to start the build).

----End

Starting a Build Job

Step 1 Log in to ServiceStage.

Step 2 Choose **Continuous Delivery** > **Build**.

Step 3 On the **Build** page, use either of the following methods to search for a build job:

- Select the CCE cluster where the component is deployed and the build job status, and select the specified build job in the build list.
- Search for the build job in the search box.

Step 4 Click **Build Now** to start the build.

----End

Branch/Tag

Step 1 Log in to ServiceStage.

Step 2 Choose **Continuous Delivery** > **Build**.

Step 3 On the **Build** page, use either of the following methods to search for a build job:

- Select the CCE cluster where the component is deployed and the build job status, and select the specified build job in the build list.
- Search for the build job in the search box.

Step 4 Choose **More** > **Branch/Tag** and set build parameters.

1. Select **Branch/Tag**.
2. Select the corresponding branch or tag from the drop-down list.
3. Specify **Commit ID** for the branch or tag.

Step 5 Click **OK**.

----End

Deleting a Build Job

Step 1 Log in to ServiceStage.

Step 2 Choose **Continuous Delivery** > **Build**.

Step 3 On the **Build** page, use either of the following methods to search for a build job:

- Select **User created** and select the CCE cluster and build job status. Then, select the target build job in the build list.
- Search for the build job created by the specified user in the search box.

Step 4 Click **More** > **Delete**.

Step 5 Click **OK**.

----End

9.6 Managing Pipelines

One-click deployment can be achieved through pipeline. In this way, the entire process of source code pull, compilation, packaging, archiving, and deployment is automatically implemented. This unifies the integration environment and standardizes the delivery process.

In the new pipeline, the "phase/task" model is optimized to the "build/ environment" model. Each pipeline includes a group of build jobs and one or more groups of environment (such as development environment, production-like environment, and production environment) tasks, each group of environment tasks contains one or more subtasks (such as deployment and test tasks) and provides templates.

ServiceStage allows a single user to create a maximum of 100+N pipelines in a . N indicates the total number of components created by the user.

Creating a Pipeline

Step 1 Log in to ServiceStage, choose **Continuous Delivery** > **Pipeline**, and click **Create Pipeline**.

Step 2 Enter the basic pipeline information.

1. Enter **Pipeline**.
2. Enter **Enterprise Project**.

   Enterprise projects let you manage cloud resources and users by project.

   It is available after you **create an enterprise project**.
3. (Optional) Enter **Description**.

Step 3 Select a pipeline template.

ServiceStage provides built-in pipeline templates in typical scenarios. After you select a pipeline template, the Build/Environment model is automatically generated. You can directly use the model.

Table 9-2 Template description

Template	Description	Description
Empty template	You need to add the build/environment model.	Set this parameter as required. For details, see **Step 3.1** to **Step 3.3**.
Simple template	The "build" model is automatically added to compile and build the source code of the code library.	For details, see **Step 3.1**.
Common template	The "build/environment" model is automatically added to compile and build the source code in the code library, and the generated software package or image is continuously released to the production environment.	For details, see **Step 3.1** to **Step 3.3**.

1. Add a build job.

   Click **Select Build Job**, select a created build job, and click **OK**.

   If no build job is available, choose **Select Build Job** > **New build task** to create a source code build job or package build job. For details, see **9.3 Creating a Source Code Job** or **9.4 Creating a Package Job**.

   Repeat this step to add more build jobs.
2. Add a deploy job.

   Click **Add Environment** and enter an environment name. Select a deployed application component.

   If no application component is available, create and deploy an application component. For details, see **7.2 Creating and Deploying a Component**.

   Select the build job added in **Step 3.1** from the **Select Build Job** drop-down list box.

   Select build output.

   Repeat this step to add more environments.
3. Set pipeline approval.

   Click in the environment area to set the approval mode and approver.

   – **Approval Mode**: **By all** and **By one person** are now supported.

   – **Approved By**: You can select multiple accounts as approvers. The system automatically loads all subaccounts of the account.

Step 4 Click **Create and Start** to start the pipeline.

Click **Create** to save the settings and do not execute the pipeline.

----End

Configuring the Pipeline Triggering Policy

Choose **Continuous Delivery** > **Pipeline**. On the **Pipeline** page that is displayed, set the pipeline triggering policy as follows.

Table 9-3 Triggering policies

Policy	Mode	Description
Manual	-	Select the pipeline task to be triggered and click **Start** to manually start the pipeline.
Automatic	-	Set the code source, corresponding namespace, repository name, and branch. When code is submitted to the corresponding branch of the source code repository, the pipeline is automatically triggered. You can set a maximum of eight trigger sources. The procedure is as follows: 1. Select a pipeline and choose **More** > **Triggering Policy**. 2. Set **Type** to **Automatic**. 3. Select **Source Code Repository** to push the code to the selected source code repository. 4. Click **OK**.

Policy	Mode	Description
Scheduled	Single-time	Set the triggering time to trigger a single-time pipeline. The procedure is as follows: 1. Select a pipeline and choose **More** > **Triggering Policy**. 2. Set **Type** to **Scheduled**. 3. Specify **Triggered**. 4. Click **OK**.
	Periodic	Set the triggering time segment, interval, and period to implement periodic pipeline triggering. The procedure is as follows: 1. Select a pipeline and choose **More** > **Triggering Policy**. 2. Set **Type** to **Scheduled**. 3. Enable **Periodic Triggering**. 4. Specify **Period**, **Triggered**, **Effective Time**, and **Period**. 5. Click **OK**.

Cloning a Pipeline

You can clone a pipeline to generate a new pipeline based on the existing pipeline configuration.

Step 1 Log in to ServiceStage and choose **Continuous Delivery** > **Pipeline**.

Step 2 Select a pipeline and choose **More** > **Clone**.

Step 3 ServiceStage automatically loads configurations of the clone pipeline. You can then modify the configurations as required by referring to **Creating a Pipeline**.

Step 4 Click **Create and Start** to start the pipeline.

Click **Create** to save the settings and do not execute the pipeline.

----End

Follow-Up Operations

After the pipeline is started, you can build and deploy applications in one-click mode. For details about maintenance operations after application components are deployed, see **7.13 Component O&M**.

9.7 Authorizing a Repository

You can create repository authorization so that build projects and application components can use the authorization information to access the software repository.

Step 1 Log in to ServiceStage, choose **Continuous Delivery** > **Repository Authorization**, and click **Create Authorization**.

Step 2 Configure authorization information by referring to the following table. Parameters marked with an asterisk (*) are mandatory.

Table 9-4 Authorization information

Parameter	Description
*Name	Authorization name, which cannot be changed after being created.

Parameter	Description
*Repository Type	The following official repositories are supported: ● GitHub (**https://github.com**) Authorization mode: OAuth or private token. ● Gitee (**https://gitee.com**) Authorization mode: OAuth or private token. ● Bitbucket (**https://bitbucket.org**) Authorization mode: OAuth or private Bitbucket. ● GitLab (**https://gitlab.com**) Authorization mode: OAuth or private token. **NOTE** ServiceStage allows you to access official and private GitLab source code repositories using private tokens. – Visit the official GitLab source code repository, obtain and enter a private token as prompted, and select **Verify token (access the repository address from the public network)**. – Visit the private GitLab source code repository, enter the correct private GitLab source code repository address and private token as prompted. You do not need to select **Verify token (access the repository address from the public network)**.

Step 3 Click **Create**.

----End

10 Microservice Engine

10.1 Cloud Service Engine Overview

Cloud Service Engine (CSE) provides service registry, service governance, and configuration management. It allows you to quickly develop microservice applications and implement high-availability O&M, and supports multiple languages, multiple runtime systems, and Spring Cloud and Apache ServiceComb Java Chassis (Java chassis) frameworks.

You can use the professional microservice engine named "Cloud Service Engine" or create an exclusive microservice engine.

- An exclusive microservice engine is physically isolated. A tenant exclusively uses an exclusive microservice engine.
- The professional microservice engine does not support multiple AZs.
- You can configure multiple AZs when creating an exclusive engine.
- After a microservice engine is created, the AZ cannot be modified. Select a suitable AZ when creating a microservice engine.
- Exclusive microservice engines cannot run across CPU architectures.

10.2 Creating a Microservice Engine

This section describes how to create a microservice engine.

Prerequisites

- A microservice engine runs on a VPC. Before creating a microservice engine, ensure that VPCs and subnets are available.

  For details about how to create a VPC and subnet, see **Creating a VPC**.

  For details about how to add a security group rule, see **Adding a Security Group Rule**.

Table 10-1 cse-engine-default-sg rules

Direction	Priority	Policy	Protocol and Port	Type	Source Address
Inbound	1	Allow	ICMP: all	IPv6	::/0
	1	Allow	TCP: 30100–30130	IPv6	::/0
	1	Allow	All	IPv6	cse-engine-default-sg
	1	Allow	TCP: 30100–30130	IPv4	0.0.0.0/0
	1	Allow	ICMP: all	IPv4	0.0.0.0/0
Outbound	100	Allow	All	IPv4	0.0.0.0/0
	100	Allow	All	IPv6	::/0

Procedure

Step 1 Log in to ServiceStage and choose **Cloud Service Engine** > **Engines**.

Step 2 Click **Create Microservice Engine**.

Step 3 Set parameters according to the following table. Parameters marked with an asterisk (*) are mandatory.

Parameter	Description
*Billing Mode	Billing mode. Currently, **Pay-per-use** is supported.
*Enterprise Project	Enterprise project where the microservice engine locates. Enterprise projects let you manage cloud resources and users by project. It is available after you **create an enterprise project**. **NOTE** The enterprise project cannot be changed once the microservice engine is created.
*Specification	Specifications of the microservice engine.
*Engine Type	Microservice engine type. If the engine type is cluster, the engine is deployed in cluster mode and supports host-level DR.
*Name	Name of the microservice engine. The name cannot be changed once the microservice engine is created.

Parameter	Description
*AZ	Availability zone. Select one or three AZs for the engine based on the number of AZs in the environment. ● Select one AZ to provide host-level DR. ● Select three AZs to provide AZ-level DR. **NOTE** ● The AZ of a created microservice engine cannot be changed. ● The AZs in one region can communicate with each other over an intranet. ● Multiple AZs enhance DR capabilities.
*Network	You can select a created VPC and its subnets to provision logically isolated, configurable, and manageable virtual networks for your engine.
Description	Click 🖉 and enter the engine description.
Authentication Mode	The exclusive microservice engine with security authentication enabled provides the system management function using the role-based access control (RBAC) through the microservice console. ● Select **Enable security authentication**: 1. Determine whether to enable **Authenticate Programming Interface**. After it is enabled, you need to add the corresponding account and password to the microservice configuration file. Otherwise, the service cannot be registered with the engine. After it is disabled, you can register the service with the engine without configuring the account and password in the microservice configuration file, which improves the efficiency. You are advised to disable this function when accessing the service in a VPC. 2. Enter and confirm the password of user **root**. Keep the password secure. ● Select **Disable security authentication**: Disable security authentication. You can enable it after the instance is created.

Step 4 Click **Create**. The page for confirming the engine information is displayed.

Step 5 Click **Submit** and wait until the engine is created.

NOTE

- It takes about 31 minutes to create a microservice engine.
- After the microservice engine is created, its status is **Available**. For details about how to view the microservice engine status, see **10.3.1 Viewing Microservice Engine Information**.
- If the microservice engine fails to be created, view the failure cause on the **Operation** page and rectify the fault. Then, you can perform the following operations:
  - In the **Microservice Engine Information** area, click **Retry** to create a microservice engine again.
  - If the retry fails, delete the microservice engine that fails to be created. For details, see **10.3.10 Deleting a Microservice Engine**.

----End

10.3 Managing Microservice Engines

10.3.1 Viewing Microservice Engine Information

In the **Microservice Engine Information** area, you can view the microservice engine information as shown in **Table 10-2**.

Procedure

Step 1 Select the target microservice engine from the **Microservice Engine** drop-down list in the upper part of the page.

Step 2 In the **Microservice Engine Information** area, view the microservice engine information in **Table 10-2**.

Table 10-2 Engine details

Item	Description
Name	Engine name entered when **creating the microservice engine**.
Engine ID	Engine ID. You can click ⧉ to copy it.

Item	Description
Status	Engine status, which can be: ● Creating ● Available ● Unavailable ● Deleting ● Upgrading ● Resizing ● Creation failed ● Deletion failed ● Upgrade failed ● Resizing failed
Version	Engine version.
Specification	Engine specification selected when **creating the microservice engine**.
Enterprise Project	Enterprise project selected when **creating the microservice engine**. It is available after you **create an enterprise project**.
AZ	AZ selected when **creating the microservice engine**.
Description	Engine description entered when **creating the microservice engine**.

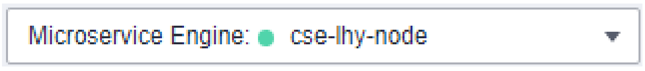
----End

10.3.2 Obtaining the Service Center Address of a Microservice Engine

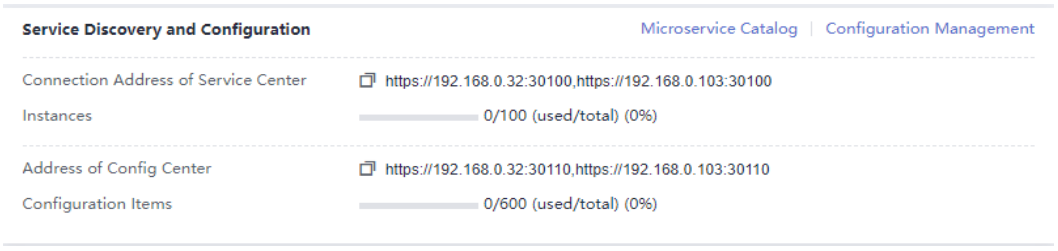
This section describes how to obtain the service center address of a microservice engine.

Procedure

Step 1 Select the target microservice engine from the **Microservice Engine** drop-down list in the upper part of the page.


Microservice Engine: ● cse-lhy-node

Step 2 In the **Service Discovery and Configuration** area, view the service center address of the microservice engine.


Service Discovery and Configuration
Microservice Catalog | Configuration Management
Connection Address of Service Center https://192.168.0.32:30100,https://192.168.0.103:30100
Instances 0/100 (used/total) (0%)
Address of Config Center https://192.168.0.32:30110,https://192.168.0.103:30110
Configuration Items 0/600 (used/total) (0%)

----End

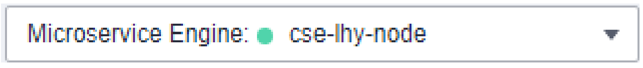
10.3.3 Obtaining the Configuration Center Address of a Microservice Engine

This section describes how to obtain the configuration center address of a microservice engine.

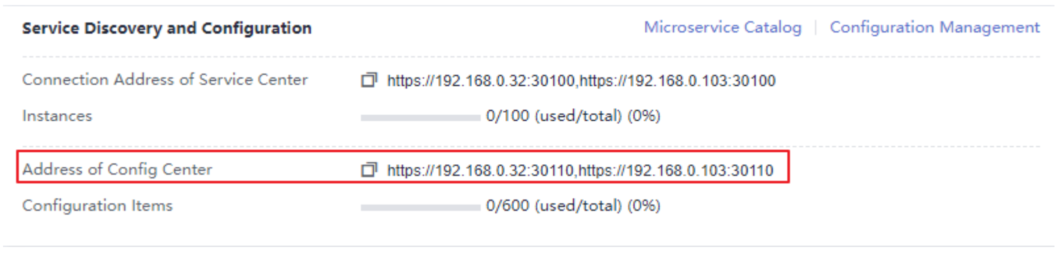
Procedure

Step 1 Log in to ServiceStage and choose **Cloud Service Engine** > **Engines**.

Step 2 Select the target microservice engine from the **Microservice Engine** drop-down list in the upper part of the page.


Microservice Engine: ● cse-lhy-node

Step 3 In the **Service Discovery and Configuration** area, view the configuration center address of the microservice engine.


Service Discovery and Configuration
Microservice Catalog | Configuration Management
Connection Address of Service Center https://192.168.0.32:30100,https://192.168.0.103:30100
Instances 0/100 (used/total) (0%)
Address of Config Center https://192.168.0.32:30110,https://192.168.0.103:30110
Configuration Items 0/600 (used/total) (0%)

NOTE

- For microservice engine 1.x, the port number of the configuration center address is 30103.
- For microservice engine 2.x, the port number of the configuration center address is 30110.

----End

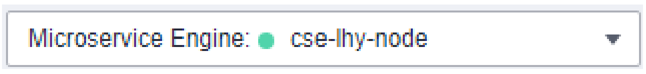
10.3.4 Viewing the Quota of Microservice Engine Instances

This section describes how to view the instance quota and quota usage of a microservice engine.

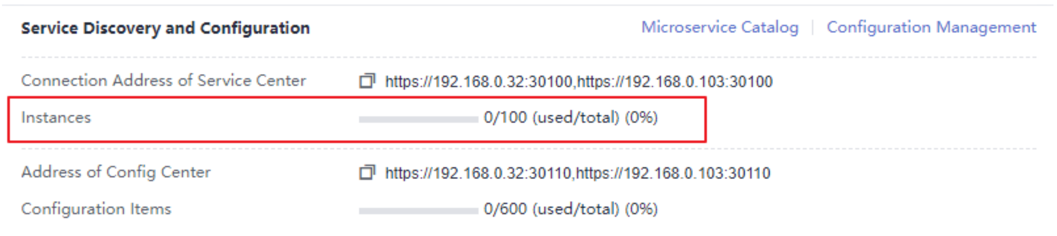
Procedure

Step 1 Log in to ServiceStage and choose **Cloud Service Engine** > **Engines**.

Step 2 Select the target microservice engine from the **Microservice Engine** drop-down list in the upper part of the page.


Microservice Engine: ● cse-lhy-node

Step 3 In the **Service Discovery and Configuration** area, view the instance quota and quota usage of the microservice engine.


Service Discovery and Configuration

Connection Address of Service Center	https://192.168.0.32:30100,https://192.168.0.103:30100
Instances	0/100 (used/total) (0%)
Address of Config Center	https://192.168.0.32:30110,https://192.168.0.103:30110
Configuration Items	0/600 (used/total) (0%)

----End

10.3.5 Viewing the Quota of Microservice Engine Configuration Items

This section describes how to view the configuration item quota and quota usage of a microservice engine.

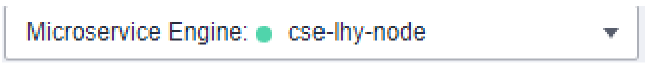
NOTE

This section applies only to microservice engine 2.x.

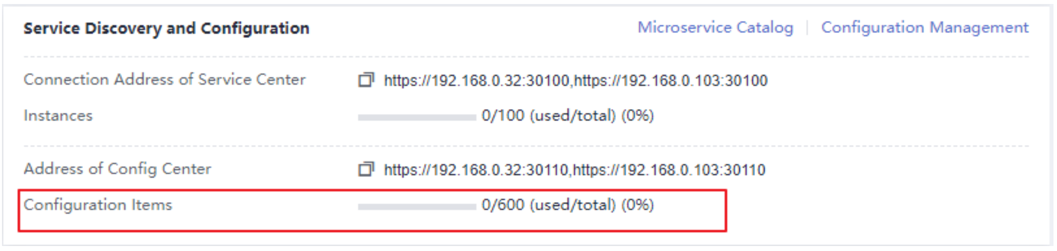
Procedure

Step 1 Log in to ServiceStage and choose **Cloud Service Engine** > **Engines**.

Step 2 Select the target microservice engine from the **Microservice Engine** drop-down list in the upper part of the page.


Microservice Engine: ● cse-lhy-node

Step 3 In the **Service Discovery and Configuration** area, view the configuration item quota and quota usage of the microservice engine.


Service Discovery and Configuration

Connection Address of Service Center	https://192.168.0.32:30100,https://192.168.0.103:30100
Instances	0/100 (used/total) (0%)
Address of Config Center	https://192.168.0.32:30110,https://192.168.0.103:30110
Configuration Items	0/600 (used/total) (0%)

----End

10.3.6 Configuring Backup and Restoration of a Microservice Engine

The ServiceStage console provides the backup and restoration functions. You can back up and restore CSE data, including microservices, contracts, configurations, and account role.

You can customize backup policies to periodically back up microservice engines or manually back up microservice engines.

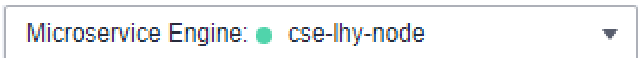
Background

- Each exclusive microservice engine supports a maximum of 15 successful backups, including a maximum of 10 manual backups and a maximum of 5 automatic backups.
- The backup data will be stored for 10 days. Expired backup data will be deleted.

Automatic Backup

Step 1 Log in to ServiceStage and choose **Cloud Service Engine** > **Engines**.

Step 2 Select the target microservice engine from the **Microservice Engine** drop-down list in the upper part of the page.


Microservice Engine: ● cse-lhy-node

Step 3 In the **Backup and Restoration** area, click **Automatic backup settings** and set backup parameters.

Table 10-3 Automatic backup parameters

Parameter	Description
Automatic Backup	After automatic backup is disabled, the previously set backup policy will be deleted. In this case, you need to set the backup policy again.
Backup Interval	Backup period. This parameter takes effect after **Automatic Backup** is enabled.
Trigger Time	Time at which a backup task starts. Only the hour is supported. This parameter takes effect after **Automatic Backup** is enabled.

Step 4 Click **OK**.

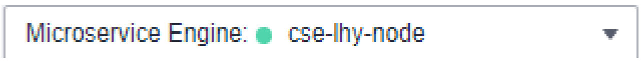
Once the backup policy is set, the backup task is triggered within one hour after the preset time.

----End

Manual Backup

Step 1 Log in to ServiceStage and choose **Cloud Service Engine** > **Engines**.

Step 2 Select the target microservice engine from the **Microservice Engine** drop-down list in the upper part of the page.


Microservice Engine: ● cse-lhy-node

Step 3 In the **Backup and Restoration** area, click **Create Manual Backup** and set backup parameters.

Table 10-4 Manual backup parameters

Parameter	Description
Name	Name of a backup task. The name cannot be changed after the backup task is created.
Remarks	(Optional) Description about the backup task.

Step 4 Click **OK**.

----End

Restoring Backup Data

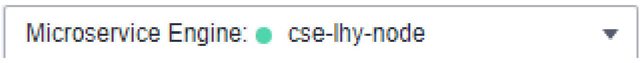
NOTICE

The backup data will overwrite the current data of the microservice engine. As a result, the microservice and service instances may be messed, and dynamic configurations may be lost. Exercise caution when performing this operation.

If security authentication is enabled, the backup data contains the account information. You are advised to disable security authentication before restoring the backup data. Otherwise, the authentication for accessing the microservice engine may fail after the restoration.

Step 1 Log in to ServiceStage and choose **Cloud Service Engine** > **Engines**.

Step 2 Select the target microservice engine from the **Microservice Engine** drop-down list in the upper part of the page.


Microservice Engine: ● cse-lhy-node

Step 3 In the **Backup and Restoration** area, click **Restore** in the **Operation** column of the row that contains the specified backup data.

1. Select **I have read and fully understand the risks**.

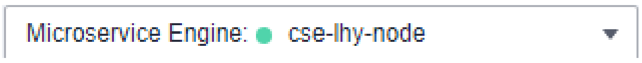
2. Click **OK**. To view the restoration status, click **Restoration History** in the **Backup and Restoration** area.

----End

Deleting Backup Data

Step 1 Log in to ServiceStage and choose **Cloud Service Engine** > **Engines**.

Step 2 Select the target microservice engine from the **Microservice Engine** drop-down list in the upper part of the page.



Step 3 In the **Backup and Restoration** area, click **Delete** in the **Operation** column of the target backup data. In the displayed dialog box, enter **DELETE** and click **OK**.

----End

10.3.7 Managing Public Network Access for a Microservice Engine

10.3.7.1 Binding an EIP

Exclusive microservice engines that are bound with EIPs can be accessed from the public network.

NOTICE

Microservice engines that do not have security authentication enabled do not have the authentication and authorization capabilities. Opening those microservice engines to the public network may cause security risks and increases the system vulnerability. For example, data assets such as configurations and service information may be stolen.

Do not use this function in a production environment or a network environment with high security requirements.

Prerequisites

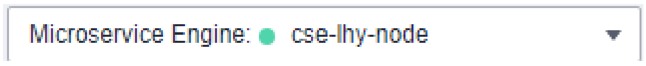
An EIP has been created.

For details about how to create an EIP, see **Assigning an EIP and Binding It to an ECS**.

Procedure

Step 1 Log in to ServiceStage and choose **Cloud Service Engine** > **Engines**.

Step 2 Select the target microservice engine from the **Microservice Engine** drop-down list in the upper part of the page.


Microservice Engine: ● cse-lhy-node

Step 3 In the **Network Configuration and Security** area, click **Bind EIP**.

Step 4 Read the security risk prompt in the displayed dialog box and select **I understand the security risks**.

Step 5 In the **EIP** drop-down list, select the EIP to be bound.

Step 6 Click **OK**.

----End

10.3.7.2 Unbinding an EIP

If an EIP has been bound to an exclusive microservice engine, you can unbind the EIP from the exclusive microservice engine to disable the public network access to the engine.

Procedure

Step 1 Log in to ServiceStage and choose **Cloud Service Engine** > **Engines**.

Step 2 Select the target microservice engine from the **Microservice Engine** drop-down list in the upper part of the page.

Microservice Engine: ● cse-lhy-node

Step 3 In the **Network Configuration and Security** area, click **Unbind EIP**.

Step 4 In the displayed dialog box, click **OK**.

----End

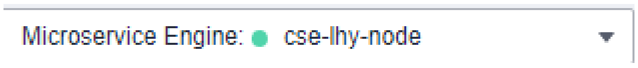
10.3.8 Viewing Microservice Engine Operation Logs

In the **Operation** area, you can view the operation logs of a microservice engine.

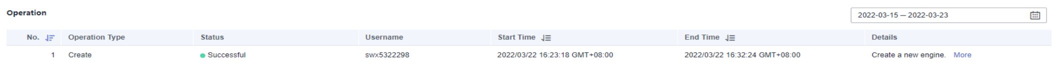
Procedure

Step 1 Log in to ServiceStage and choose **Cloud Service Engine** > **Engines**.

Step 2 Select the target microservice engine from the **Microservice Engine** drop-down list in the upper part of the page.


Microservice Engine: ● cse-lhy-node

Step 3 In the **Operation** area, view the operation logs of the microservice engine.


Operation

2022-03-15 — 2022-03-23

No.	Operation Type	Status	Username	Start Time	End Time	Details
1	Create	● Successful	swx5322298	2022/03/22 16:23:18 GMT+08:00	2022/03/22 16:32:24 GMT+08:00	Create a new engine. More

- Click in the upper right corner to view operation logs in a specified period.
- Click **More** in the **Details** column of a specified operation log to view details about the operation log.

----End

10.3.9 Upgrading a Microservice Engine Version

Microservice engines are created using the latest engine version. When a later version is released, you can upgrade your microservice engine.

NOTICE

Only exclusive microservice engines can be upgraded. Version rollback is not supported after the upgrade.

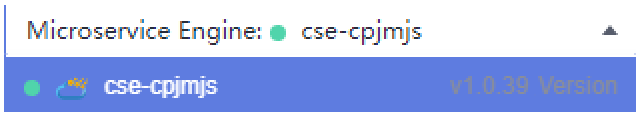
Background

During upgrade, two instances are upgraded in rolling mode without service interruptions. However, one of the two access addresses may be unavailable. In this case, you need to quickly switch to the other instance. Currently, ServiceComb SDK and Mesher support instance switching. If you call the APIs of the service center and configuration center for service registry and discovery, instance switching is required.

Procedure

Step 1 Log in to ServiceStage and choose **Cloud Service Engine** > **Engines**.

Step 2 Select the target microservice engine from the **Microservice Engine** drop-down list in the upper part of the page.



Step 3 In the **Microservice Engine Information** area, click **Upgrade**.

Step 4 Select **Target Version** and view the version description. Determine whether to upgrade the software to this version.

Step 5 Click **OK**.

If the upgrade fails, click **Retry** to perform the upgrade again.

----End

10.3.10 Deleting a Microservice Engine

You can delete an exclusive microservice engine if it is no longer used.

NOTICE

Deleted engines cannot be recovered. Exercise caution when performing this operation.

Background

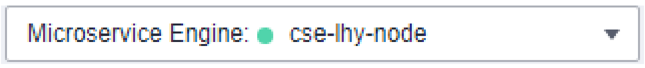
You can delete exclusive microservice engines in the following states:

- Available
- Unavailable
- Creation failed
- Resizing failed
- Upgrade failed

Procedure

Step 1 Log in to ServiceStage and choose **Cloud Service Engine** > **Engines**.

Step 2 Select the target microservice engine from the **Microservice Engine** drop-down list in the upper part of the page.



Step 3 In the **Microservice Engine Information** area, click **Delete**. In the displayed dialog box, enter **DELETE** and click **OK**.

----End

10.3.11 Managing Security Authentication of a Microservice Engine

A microservice engine may be used by multiple users. Different users must have different microservice engine access and operation permissions based on their responsibilities and permissions. If security authentication is enabled for an exclusive microservice engine, grant different access and operation permissions to users based on the roles associated with the accounts used by the users to access the microservice engine.

For details about security authentication, see **10.4.6 System Management**.

Currently, Java chassis and Spring Cloud support security authentication for microservices. The Java chassis version must be 2.3.5 or later, and Spring Cloud must integrate Spring Cloud Huawei 1.6.1 or later.

You can enable or disable security authentication for the exclusive microservice engine based on service requirements.

- **Enabling Security Authentication**

  If a microservice engine is available with security authentication disabled, you can enable security authentication based on service requirements.

  After security authentication is enabled and programming interface authentication is also enabled, if security authentication parameters are not configured for the microservice components connected to the engine, or the security authentication account and password configured for the microservice components are incorrect, the heartbeat of the microservice components fails and the service is forced to go offline.

- **Disabling Security Authentication**

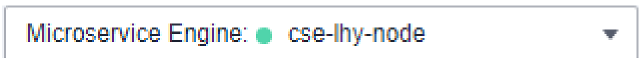
  If a microservice engine is available with security authentication enabled, you can disable security authentication based on service requirements.

  After security authentication is disabled for a microservice component, service functions of the microservice component are not affected no matter whether security authentication parameters are configured for the microservice component.

Enabling Security Authentication

Step 1 Log in to ServiceStage and choose **Cloud Service Engine** > **Engines**.

Step 2 Select the target microservice engine from the **Microservice Engine** drop-down list in the upper part of the page.



Step 3 In the **Network Configuration and Security** area, click **Enable security authentication**.

- If the engine version is earlier than 1.2.0, go to **Step 4**.
- If the engine version is 1.2.0 or later, go to **Step 5**.

Step 4 Upgrade the engine to 1.2.0 or later.

1. Click **Upgrade**.
2. Select **Target Version** and view the version description. Determine whether to upgrade the software to this version. Then, click **OK**.
3. After the engine is upgraded, click **Back to CSE**.
4. Select the upgraded microservice engine. In the **Network Configuration and Security** area, click **Enable security authentication**.

Step 5 On the **Cloud Service Engine** > **System Management** page, enable security authentication.

- To enable security authentication for the first time, click **Enable security authentication**.

  You need to create user **root** first. Enter and confirm the password of user **root**. Then, click **Create Now**.

- Enable security authentication again and enter the name and password of the account associated with the **admin** role in the engine.

Step 6 (Optional) Create a role based on service requirements. For details, see **10.4.6.3 Roles**.

Step 7 (Optional) Create an account based on service requirements. For details, see **10.4.6.2 Accounts**.

Step 8 click **Enable security authenticationSet Authentication**, and set security configurations based on service requirements.

- If you enable **Authenticate Console**, go to **Step 10**.

  After **Authenticate Console** is enabled, you need to use an account and password to log in to the CSE console. The login user can only view and configure services on which the user has permission.
- If you enable **Authenticate Programming Interface**, go to **Step 9**.

  After **Authenticate Programming Interface** is enabled, **Authenticate Console** is automatically enabled.

  After it is enabled, you need to add the corresponding account and password to the microservice configuration file. Otherwise, the service cannot be registered with the engine.

  After it is disabled, you can register the service with the engine without configuring the account and password in the microservice configuration file, which improves the efficiency. You are advised to disable this function when accessing the service in a VPC.

Step 9 Configure the SDK. For microservice components that have been deployed but not configured with security authentication parameters, configure the account name and password for security authentication and then upgrade the component.

Step 10 Click **OK**.

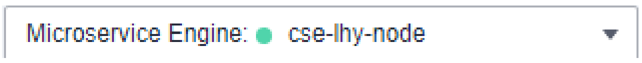
After the microservice engine is updated and the engine status changes from **Configuring** to **Available**, security authentication is enabled successfully.

----End

Disabling Security Authentication

Step 1 Log in to ServiceStage and choose **Cloud Service Engine** > **Engines**.

Step 2 Select the target microservice engine from the **Microservice Engine** drop-down list in the upper part of the page.



Step 3 In the **Network Configuration and Security** area, click **Set Authentication**.

Step 4 On the **System Management** page, click **Set Authentication**.

Step 5 On the **Security Settings** page, disable **Authenticate Console**.

Step 6 Click **OK**. After the microservice engine is updated and the engine status changes from **Configuring** to **Available**, security authentication is disabled successfully.

NOTE

After security authentication is disabled, accounts created on the engine will not be deleted.

----End

10.4 Using Microservice Engines

10.4.1 Using the Microservice Dashboard

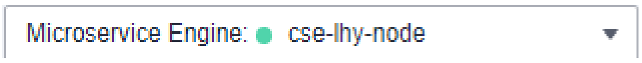
You can view metrics related to microservices through the dashboard in real time. Based on abundant and real-time dashboard data, you can take corresponding governance actions for microservices.

Background

- If a microservice application is deployed on ServiceStage, you need to configure the microservice engine during application deployment. The application automatically obtains the service registry and discovery address, configuration center address, and dashboard address. You do not need to configure the monitor address.
- If the microservice application is locally started and registered with the microservice engine, manually configure the monitor address before using the dashboard.

Procedure

Step 1 Select the target microservice engine from the **Microservice Engine** drop-down list in the upper part of the page.


Microservice Engine: ● cse-lhy-node

Step 2 Choose **Dashboard**.

- For microservice engines with security authentication disabled, go to **Step 4**.
- For microservice engines with security authentication enabled, go to **Step 3**.

Step 3 In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.

NOTE

- If you connect to the microservice engine for the first time, enter the account name **root** and the password entered when **creating the microservice engine**.
- For details about how to create an account, see **Adding an Account**.

Step 4 On the **Dashboard** page, select an application from the drop-down list box and enter a microservice name in the search box. The operating metrics of the microservice are displayed.

Click **View Diagram** to view the description of operating metrics.

Step 5 Select a sorting order to sort the filtered microservices.

----End

10.4.2 Managing Microservices

You can use the microservice catalog to view microservice details and search for target microservices to maintain microservices. The **Microservice Catalog** page contains the following tabs:

- **Application List**: displays all applications of the current microservice engine. You can search for the target application by application name, or filter applications by environment. For details, see **Viewing the Application List**.

- **Microservice List**: For details about the operations supported by in **Microservice List**, see the following table.

Operation	Description
Viewing the Microservice List	Displays all microservices of the current microservice engine. You can search for the target microservice by microservice name, or filter microservices by environment and application.
Viewing Microservice Details	On the microservice details page, you can view the instance list, called services, calling services, dynamic configuration, and service contract.
Creating a Microservice	Creates a microservice.
Cleaning Versions Without Instances	Cleans microservice versions that have no instances.
Deleting a Microservice	Deletes a microservice that is no longer used.
Dynamic Configuration	Creates a microservice-level configuration.
Dark Launch	In dark launch, new features are tested in a selected group of users. When the features become mature and stable, they are released to all users.

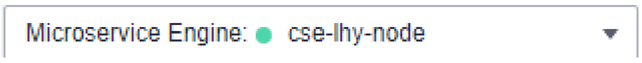
- **Instance List**: For details about the operations supported by in **Instance List**, see the following table.

Operation	Description
Viewing the Instance List	Displays all instances of the current microservice engine. You can search for the target instance by microservice name, or filter instances by environment and application.
Changing the Instance Status	**Status** indicates the status of a microservice instance.

Viewing the Application List

Step 1 Log in to ServiceStage and choose **Cloud Service Engine** > **Engines**.

Step 2 Select the target microservice engine from the **Microservice Engine** drop-down list in the upper part of the page.



Step 3 Choose **Microservice Catalog**.

- For microservice engines with security authentication disabled, go to **Step 5**.
- For microservice engines with security authentication enabled, go to **Step 4**.

Step 4 In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.

NOTE

- If you connect to the microservice engine for the first time, enter the account name **root** and the password entered when **creating the microservice engine**.
- For details about how to create an account, see **Adding an Account**.

Step 5 Click **Application List** to view details about all applications of the current account under the engine.

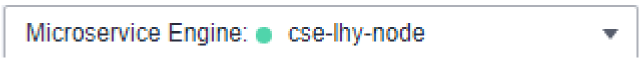
You can search for the target application by application name, or filter applications by environment.

----End

Viewing the Microservice List

Step 1 Log in to ServiceStage and choose **Cloud Service Engine** > **Engines**.

Step 2 Select the target microservice engine from the **Microservice Engine** drop-down list in the upper part of the page.



Step 3 Choose **Microservice Catalog**.

- For microservice engines with security authentication disabled, go to **Step 5**.
- For microservice engines with security authentication enabled, go to **Step 4**.

Step 4 In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.

NOTE

- If you connect to the microservice engine for the first time, enter the account name **root** and the password entered when **creating the microservice engine**.
- For details about how to create an account, see **Adding an Account**.

Step 5 Click **Microservice List** to view all microservices of the current account under the engine.

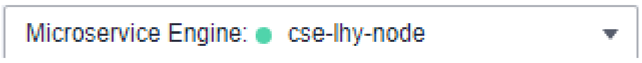
You can search for the target microservice by microservice name, or filter microservices by environment and application.

----End

Viewing Microservice Details

Step 1 Log in to ServiceStage and choose **Cloud Service Engine** > **Engines**.

Step 2 Select the target microservice engine from the **Microservice Engine** drop-down list in the upper part of the page.



Step 3 Choose **Microservice Catalog**.

- For microservice engines with security authentication disabled, go to **Step 5**.
- For microservice engines with security authentication enabled, go to **Step 4**.

Step 4 In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.

NOTE

- If you connect to the microservice engine for the first time, enter the account name **root** and the password entered when **creating the microservice engine**.
- For details about how to create an account, see **Adding an Account**.

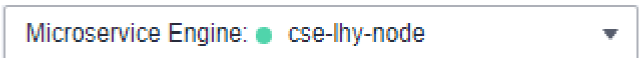
Step 5 Click the microservice to be viewed in **Microservice List**. On the displayed page, view the instance list, called services, calling services, configurations, and service contract.

----End

Creating a Microservice

Step 1 Log in to ServiceStage and choose **Cloud Service Engine** > **Engines**.

Step 2 Select the target microservice engine from the **Microservice Engine** drop-down list in the upper part of the page.



Step 3 Choose **Microservice Catalog**.

- For microservice engines with security authentication disabled, go to **Step 5**.
- For microservice engines with security authentication enabled, go to **Step 4**.

Step 4 In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.

NOTE

- If you connect to the microservice engine for the first time, enter the account name **root** and the password entered when **creating the microservice engine**.
- For details about how to create an account, see **Adding an Account**.

Step 5 Choose **Microservice List** > **Create a Microservice** and set microservice parameters by referring to the following table. Parameters marked with an asterisk (*) are mandatory.

Parameter	Description
*Microservice	Microservice name, for example, **myServiceName**.
*Application	Name of the application to which the microservice belongs. Microservices are isolated by applications.
*Version	Microservice version. The default value is **1.0.0**. **NOTE** The microservice version is in the format of X.Y.Z or X.Y.Z.B, where X, Y, Z, and B are digits and range from 0 to 32767. The value contains 3 to 46 characters.
*Environment	Environment where the microservice is located to isolate microservice data, including the version and instance.
Detail	Microservice description.

Step 6 Click **OK**.

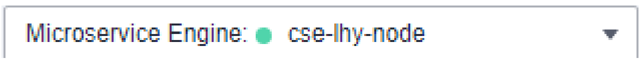
Once the microservice is created, it will be displayed in **Microservice List**.

----End

Cleaning Versions Without Instances

Step 1 Log in to ServiceStage and choose **Cloud Service Engine** > **Engines**.

Step 2 Select the target microservice engine from the **Microservice Engine** drop-down list in the upper part of the page.


Microservice Engine: ● cse-lhy-node

Step 3 Choose **Microservice Catalog**.

- For microservice engines with security authentication disabled, go to **Step 5**.
- For microservice engines with security authentication enabled, go to **Step 4**.

Step 4 In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.

NOTE

- If you connect to the microservice engine for the first time, enter the account name **root** and the password entered when **creating the microservice engine**.
- For details about how to create an account, see **Adding an Account**.

Step 5 Choose **Microservice List** > **Clean No Instance Services**. Select the microservice version without instances to be cleaned.

You can search for the target microservice by microservice name, or filter microservices by environment and application.

Step 6 Click **OK**.

----End

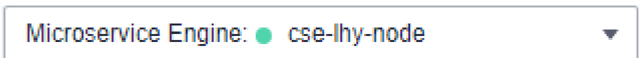
Deleting a Microservice

NOTICE

- After a microservice is deleted, you can restore it by referring to **Restoring Backup Data**.
- If the service to be deleted has instances, delete the instances first. Otherwise, the service will be registered again.

Step 1 Log in to ServiceStage and choose **Cloud Service Engine** > **Engines**.

Step 2 Select the target microservice engine from the **Microservice Engine** drop-down list in the upper part of the page.



Step 3 Choose **Microservice Catalog**.

- For microservice engines with security authentication disabled, go to **Step 5**.
- For microservice engines with security authentication enabled, go to **Step 4**.

Step 4 In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.

NOTE

- If you connect to the microservice engine for the first time, enter the account name **root** and the password entered when **creating the microservice engine**.
- For details about how to create an account, see **Adding an Account**.

Step 5 Click **Microservice List**.

- To delete microservices in batches, select the microservices to be deleted and click **Delete** above the microservices.
- To delete one microservice, locate the row that contains the microservice to be deleted and click **Delete** in the **Operation** column.

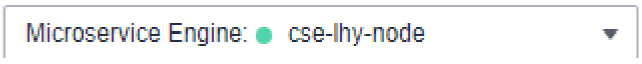
Step 6 In the displayed dialog box, enter **DELETE** to confirm the deletion and click **OK**.

----End

Dynamic Configuration

Step 1 Log in to ServiceStage and choose **Cloud Service Engine** > **Engines**.

Step 2 Select the target microservice engine from the **Microservice Engine** drop-down list in the upper part of the page.



Step 3 Choose **Microservice Catalog**.

- For microservice engines with security authentication disabled, go to **Step 5**.
- For microservice engines with security authentication enabled, go to **Step 4**.

Step 4 In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.

NOTE

- If you connect to the microservice engine for the first time, enter the account name **root** and the password entered when **creating the microservice engine**.
- For details about how to create an account, see **Adding an Account**.

Step 5 Click **Microservice List**.

Step 6 Click the target microservice.

Step 7 Choose **Dynamic Configuration**. On the **Dynamic Configuration** tab, perform the following operations.

NOTICE

Configuration items are stored in plaintext. Do not include sensitive data.

Operation	Procedure
Creating a configuration item	See **Creating a Microservice-Level Configuration**. **Microservice-level** is selected for **Configuration Range** and **Microservices** is set to the current microservice.
Viewing historical versions	Click **View Historical Version** in the **Operation** column of the target configuration item.
Disabling a configuration item	1. Click **Disable** in the **Operation** column of the target configuration item. 2. Click **OK**.
Modifying a configuration item	1. Click **More** > **Edit** in the **Operation** column of the target configuration item. 2. On the configuration details page, click **Edit**. 3. On the **Configuration Details** tab, enter the new configuration. 4. Click **Save**.
Deleting a configuration item	1. Click **More** > **Delete** in the **Operation** column of the target configuration item. 2. Click **OK**.

----End

Dark Launch

In dark launch, new features are tested in a selected group of users. When the features become mature and stable, they are released to all users. This ensures the smooth feature rollout.

Step 1 Log in to ServiceStage and choose **Cloud Service Engine** > **Engines**.

Step 2 Choose **Microservice Catalog**.

- For microservice engines with security authentication disabled, go to **Step 4**.
- For microservice engines with security authentication enabled, go to **Step 3**.

Step 3 In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.

NOTE

- If you connect to the microservice engine for the first time, enter the account name **root** and the password entered when **creating the microservice engine**.
- For details about how to create an account, see **Adding an Account**.

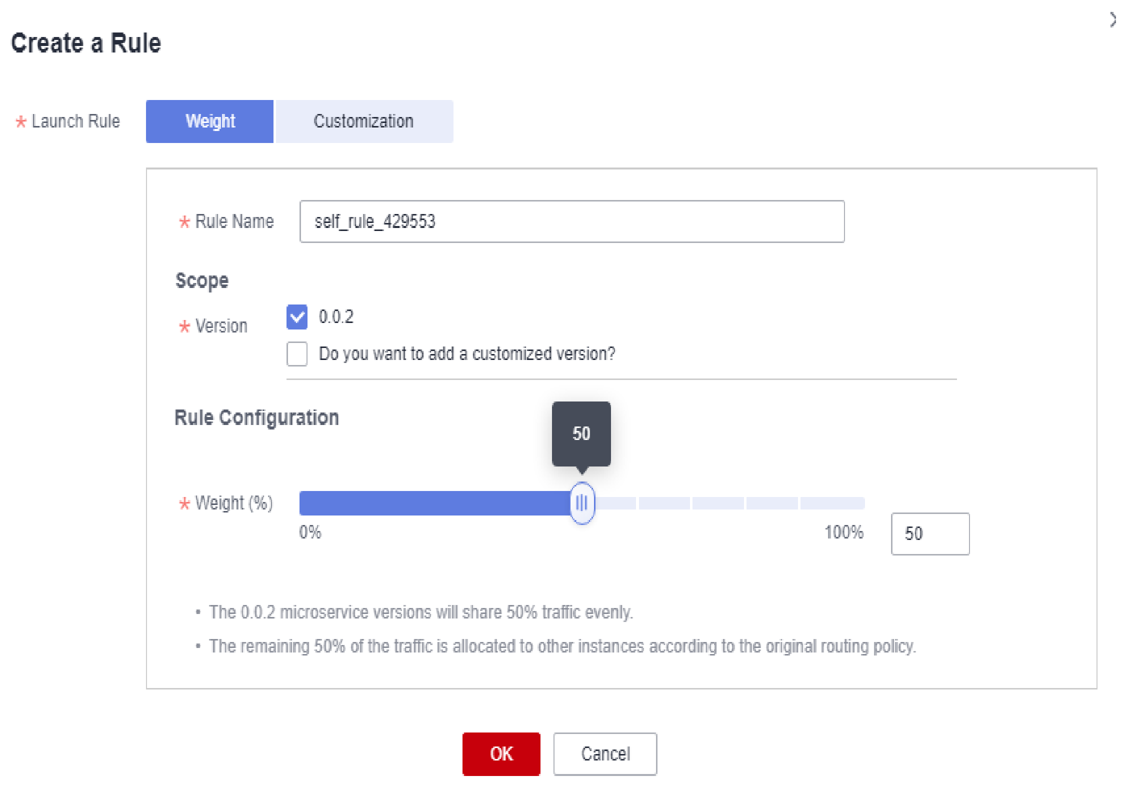
Step 4 In the microservice list, click a microservice. On the displayed page, choose **Dark Launch**.

Step 5 Click **Add a Launch Rule**.

- To add a launch rule by **Weight**:

  a. Click **Weight**.

Figure 10-1 Adding a launch rule by weight


Create a Rule

* Launch Rule: Weight | Customization

* Rule Name: self_rule_429553

Scope

* Version: ☑ 0.0.2 ☐ Do you want to add a customized version?

Rule Configuration

* Weight (%): 50 (0% – 100%) 50

- The 0.0.2 microservice versions will share 50% traffic evenly.
- The remaining 50% of the traffic is allocated to other instances according to the original routing policy.

b. Set the following parameters.

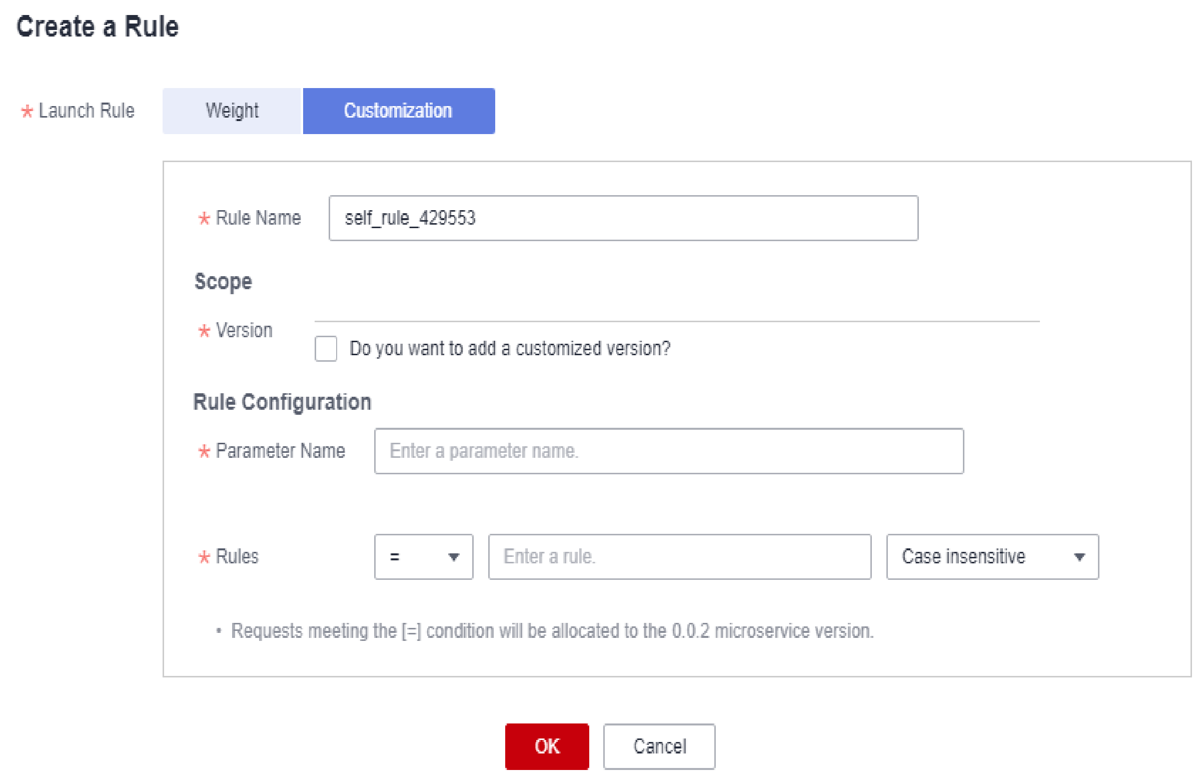
Item	Description
Rule Name	Name of the rule.
Scope	■ Microservice version to which the rule applies. ■ Select **Do you want to add a customized version?** and add a new version as prompted.
Rule Configuration	Traffic allocation rate for the selected version. Traffic is evenly allocated to the selected service versions based on the configured value.

c. Click **OK** to complete the weight rule configuration and dark launch.

- To add a launch rule by **Customization**:

a. Click **Customization**.

Figure 10-2 Customizing a launch rule



Create a Rule

* Launch Rule: Weight | Customization

* Rule Name: self_rule_429553

Scope

* Version: Do you want to add a customized version?

Rule Configuration

* Parameter Name: Enter a parameter name.

* Rules: = | Enter a rule. | Case insensitive

• Requests meeting the [=] condition will be allocated to the 0.0.2 microservice version.

b. Set the following parameters.

Item	Description
Rule Name	Name of the rule.

Item	Description
Scope	■ Microservice version to which the rule applies. ■ Select **Do you want to add a customized version?** and add a new version as prompted.
Rule Configuration	■ Parameter Name Name customized according to the key field provided by the service contract. This key must exist in the contract. It is possible that the server API is **String paramA**, but **paramB** is actually generated after the annotation is added. Therefore, **paramB** should be set here. ■ Rules Value corresponding to the key of a contract. **NOTE** ○ If ~ is selected from the drop-down list next to **Rules**, the asterisk (*) and question mark (?) in the **Rules** value can be used for fuzzy matching. The asterisk (*) indicates that the character length is not limited, and the question mark (?) indicates that only one character is allowed. For example, if the rule value of **Name** is set to ***1000**, all **Name** fields ending with 1000 can be matched. ○ If ~ is not selected from the drop-down list next to **Rules**, the asterisk (*) and question mark (?) in the **Rules** value cannot be used for fuzzy matching.

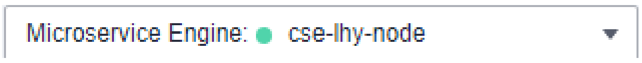
c. Click **OK** to complete the customization rule configuration and dark launch.

----End

Viewing the Instance List

Step 1 Log in to ServiceStage and choose **Cloud Service Engine** > **Engines**.

Step 2 Select the target microservice engine from the **Microservice Engine** drop-down list in the upper part of the page.



Step 3 Choose **Microservice Catalog**.

- For microservice engines with security authentication disabled, go to **Step 5**.
- For microservice engines with security authentication enabled, go to **Step 4**.

Step 4 In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.

NOTE

- If you connect to the microservice engine for the first time, enter the account name **root** and the password entered when **creating the microservice engine**.
- For details about how to create an account, see **Adding an Account**.

Step 5 Click **Instance List** to view all instances of the engine.

You can search for the target instance by microservice name, or filter instances by environment and application.

----End

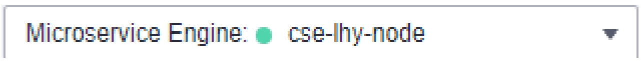
Changing the Instance Status

Status indicates the status of a microservice instance. The following table describes the microservice instance statuses.

Status	Description
Online	The instance is running and can provide services.
Offline	Before the instance process ends, the instance is marked as not providing services externally.
Out of Service	The instance has been registered with the microservice engine and does not provide services.
Testing	The instance is in the internal joint commissioning state and does not provide services.

Step 1 Log in to ServiceStage and choose **Cloud Service Engine** > **Engines**.

Step 2 Select the target microservice engine from the **Microservice Engine** drop-down list in the upper part of the page.


Microservice Engine: ● cse-lhy-node

Step 3 Choose **Microservice Catalog**.

- For microservice engines with security authentication disabled, go to **Step 5**.
- For microservice engines with security authentication enabled, go to **Step 4**.

Step 4 In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.

NOTE

- If you connect to the microservice engine for the first time, enter the account name **root** and the password entered when **creating the microservice engine**.
- For details about how to create an account, see **Adding an Account**.

Step 5 Click **Instance List**, select the target instance, and change the instance status.

- Offline

  In the **Operation** column, click **Offline**.
- Online

  In the **Operation** column, choose **More** > **Online**.
- Out of Service

  In the **Operation** column, choose **More** > **Out of Service**.
- Testing

  In the **Operation** column, choose **More** > **Testing**.

----End

10.4.3 Microservice Governance

10.4.3.1 Overview

If an application is developed using the microservice framework, the microservice is automatically registered with the corresponding microservice engine after the application is managed and started. You can perform service governance on the microservice engine console by referring to **10.4.3.2 Governing Microservices**.

NOTE

This function is supported by only microservice engine 1.x, and 2.4.0 and later versions.

10.4.3.2 Governing Microservices

After a microservice is deployed, you can govern it based on its running statuses.

Prerequisites

- You can create a microservice in **Microservice List** from **Service Catalog** and start the microservice. After the microservice starts, the service instance is registered under the corresponding service based on configurations in the **.yaml** file.
- If the microservice is not created in advance or has been deleted, the microservice is automatically created when the service instance is registered.
- After a microservice is created, register the service instance before performing the corresponding operation.

Governance Policies

You can configure the following policies: Load Balancing, Rate Limiting, Fault Tolerance, Service Degradation, Circuit Breaker, Fault Injection, and Blacklist and Whitelist. For details, see the following table.

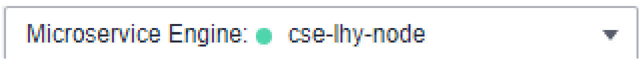
Name	Description
Load Balancing	● Application scenario Generally, multiple instances are deployed for a microservice. Load balancing controls the policy for a microservice consumer to access multiple instances of a microservice provider to balance traffic. It includes polling, random, response time weight, and session stickiness.
Rate Limiting	● Application scenario This policy controls the number of requests for accessing microservices to prevent the system from being damaged due to traffic impact.
Service Degradatio n	● Application scenario When a microservice invokes other microservices, the default value is forcibly returned or an exception is thrown instead of sending the request to the target microservice. In this way, the access to the target microservice is shielded and the pressure on the target microservice is reduced.
Fault Tolerance	● Application scenario If an exception occurs when a microservice consumer accesses a provider, for example, the instance network is disconnected, the request needs to be forwarded to another available instance. Fault tolerance is often referred to as retry.
Circuit Breaker	● Application scenario If an exception occurs when a microservice consumer accesses a provider, for example, the instance network is disconnected or the request times out, and the exception accumulates to a certain extent, the consumer needs to stop accessing the provider and return an exception or a default value to prevent the avalanche effect. Circuit breaker provides automatic circuit breaker. Automatic circuit breaker determines whether to trigger circuit breaker based on the error rate.
Fault Injection	● Application scenario Fault injection can simulate an invoking failure, which is mainly used for function verification and fault scenario demonstration. ● Governance of microservices accessed through Java chassis. **NOTE** This policy applies only to microservices accessed through Java chassis.

Name	Description
Blacklist and Whitelist	● Application scenario Based on the public key authentication mechanism, CSE provides the blacklist and whitelist functions. The blacklist and whitelist can be used to control which services can be accessed by microservices. ● Governance of microservices accessed through Java chassis The blacklist and whitelist take effect only after public key authentication is enabled. For details, see **Configuring Public Key Authentication**. NOTE This policy applies only to microservices accessed through Java chassis.

Configuring Load Balancing

Step 1 Log in to ServiceStage and choose **Cloud Service Engine** > **Engines**.

Step 2 Select the target microservice engine from the **Microservice Engine** drop-down list in the upper part of the page.


Microservice Engine: ● cse-lhy-node

Step 3 Choose **Microservice Governance**.

- For microservice engines with security authentication disabled, go to **Step 5**.
- For microservice engines with security authentication enabled, go to **Step 4**.

Step 4 In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.

NOTE

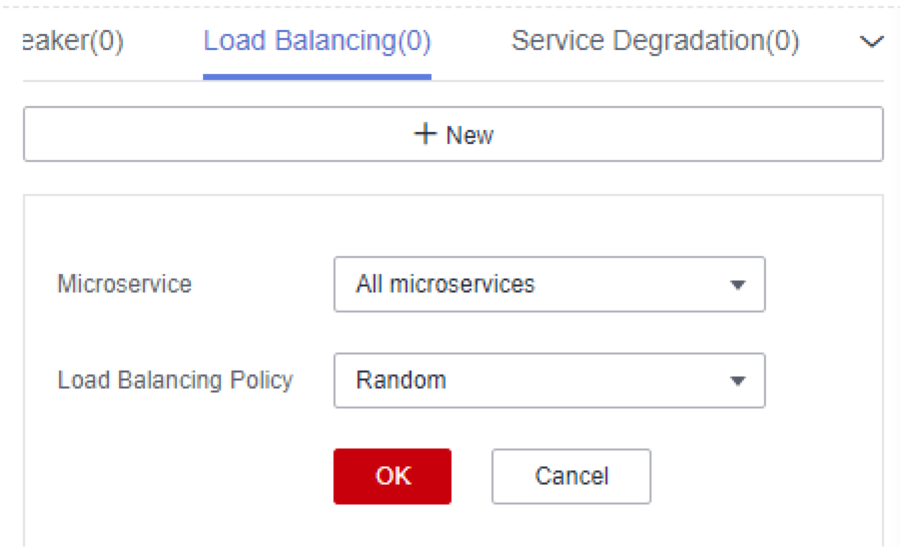
- If you connect to the microservice engine for the first time, enter the account name **root** and the password entered when **creating the microservice engine**.
- For details about how to create an account, see **Adding an Account**.

Step 5 Click the microservice to be governed.

Step 6 Choose **Load Balancing**.

Step 7 Click **New**. Select the microservices to be governed and select a proper load balancing policy. For details, see the following table.

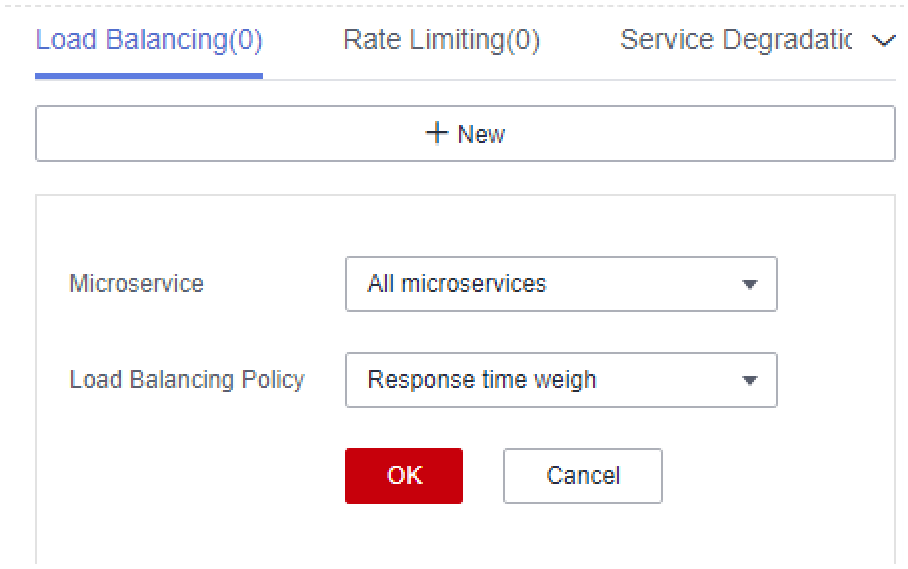
Figure 10-3 Configuring load balancing (for microservices accessed through Spring Cloud)



Microservice: All microservices

Load Balancing Policy: Random

Figure 10-4 Configuring load balancing (for microservices accessed through Java chassis)



Microservice: All microservices

Load Balancing Policy: Response time weigh

Policy	Description
Round robin	Supports routes according to the location information about service instances.
Random	Provides random routes for service instances.

Policy	Description
Response time weigh **NOTE** This policy applies to microservices accessed through Java chassis.	Provides weight routes with the minimum active number (latency) and supports service instances with slow service processing in receiving a small number of requests to prevent the system from stopping response. This load balancing policy is suitable for applications with low and stable service requests.
Session stickiness **NOTE** This policy applies to microservices accessed through Java chassis.	Provides a mechanism on the load balancer. In the specified session stickiness duration, this mechanism allocates the access requests related to the same user to the same instance. ● **Stickiness Duration**: time limit for keeping a session. The value ranges from 0 to 86400, in seconds. ● **Failures**: number of access failures. The value ranges from 0 to 10. If the upper limit of failures or the session stickiness duration exceeds the specified values, the microservice stops accessing this instance.

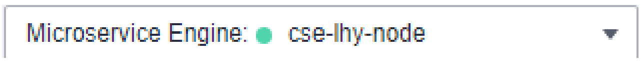
Step 8 Click **OK**.

----End

Configuring Rate Limiting

Step 1 Log in to ServiceStage and choose **Cloud Service Engine** > **Engines**.

Step 2 Select the target microservice engine from the **Microservice Engine** drop-down list in the upper part of the page.


Microservice Engine: ● cse-lhy-node

Step 3 Choose **Microservice Governance**.

- For microservice engines with security authentication disabled, go to **Step 5**.
- For microservice engines with security authentication enabled, go to **Step 4**.

Step 4 In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.

NOTE

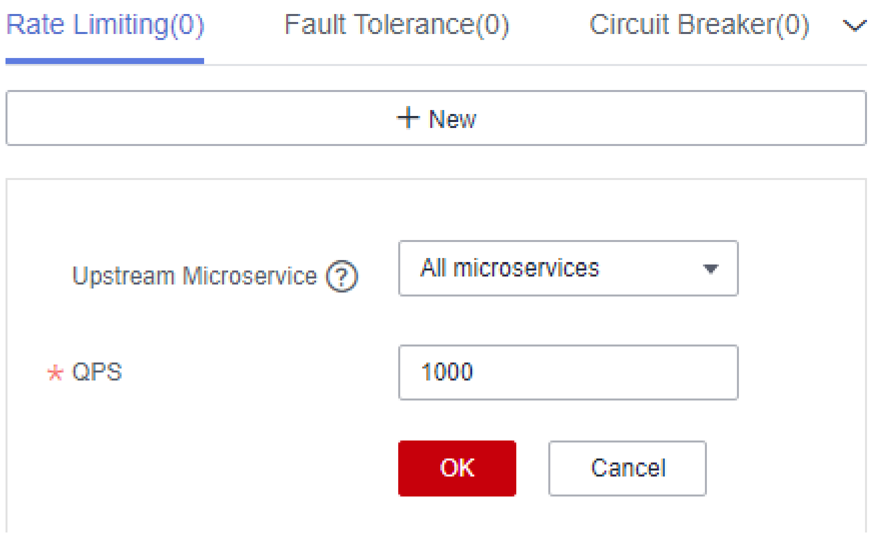
- If you connect to the microservice engine for the first time, enter the account name **root** and the password entered when **creating the microservice engine**.
- For details about how to create an account, see **Adding an Account**.

Step 5 Click the microservice to be governed.

Step 6 Click **Rate Limiting**.

Step 7 Click **New**. The following table describes configuration items of rate limiting.

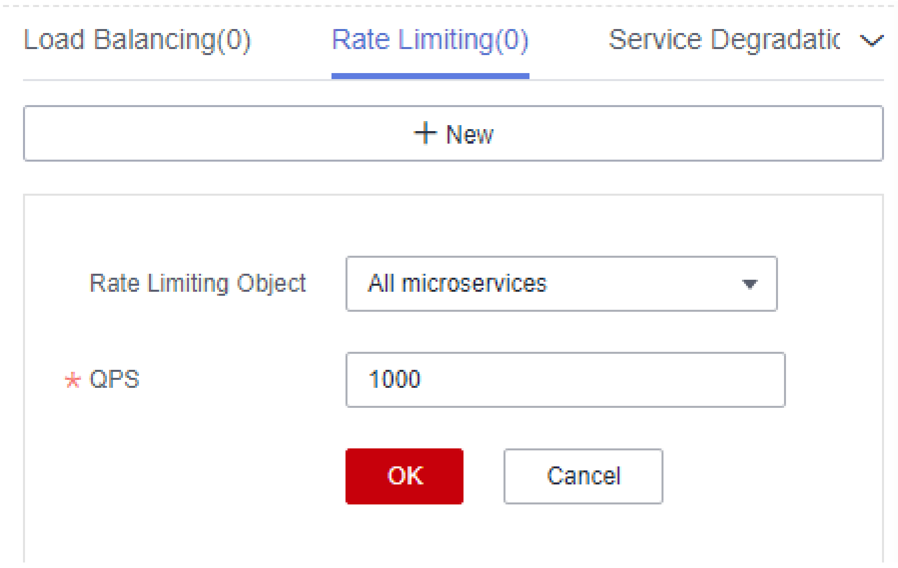
Figure 10-5 Configuring rate limiting (for microservices accessed through Spring Cloud)



Upstream Microservice: All microservices

* QPS: 1000

Figure 10-6 Configuring rate limiting (for microservices accessed through Java chassis)



Rate Limiting Object: All microservices

* QPS: 1000

Configuration Item	Description	Value Range
Rate Limiting Object **NOTE** This configuration applies to microservices accessed through Java chassis.	Other microservices that access the microservice.	Select an item from the drop-down list next to **Rate Limiting Object**.
Upstream Microservice **NOTE** This configuration applies to microservices accessed through Spring Cloud.	Configure rate limiting for the upstream microservice to invoke the service.	Select an item from the drop-down list next to **Upstream Microservice**.
QPS	Requests generated per second. When the number of requests sent by the rate limiting object to the current service instance exceeds the specified value, the current service instance no longer accepts requests from the rate limiting object.	Enter an integer ranging from 1 to 99999.

NOTE

If a microservice has three instances, the rate limiting of each instance is set to 2700 QPS, then the total QPS is 8100, and rate limiting is triggered only when the QPS exceeds 8100.

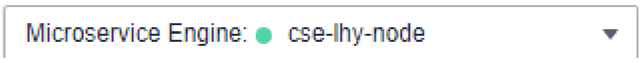
Step 8 Click **OK**.

----End

Configuring Service Degradation

Step 1 Log in to ServiceStage and choose **Cloud Service Engine** > **Engines**.

Step 2 Select the target microservice engine from the **Microservice Engine** drop-down list in the upper part of the page.



Step 3 Choose **Microservice Governance**.

- For microservice engines with security authentication disabled, go to **Step 5**.
- For microservice engines with security authentication enabled, go to **Step 4**.

Step 4 In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.

NOTE

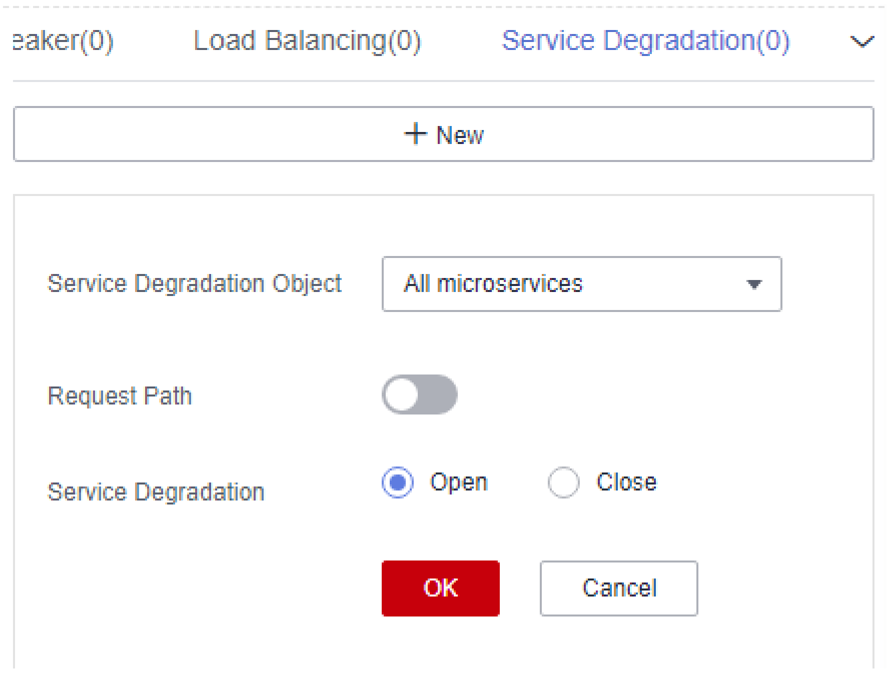
- If you connect to the microservice engine for the first time, enter the account name **root** and the password entered when **creating the microservice engine**.
- For details about how to create an account, see **Adding an Account**.

Step 5 Click the microservice to be governed.

Step 6 Click **Service Degradation**.

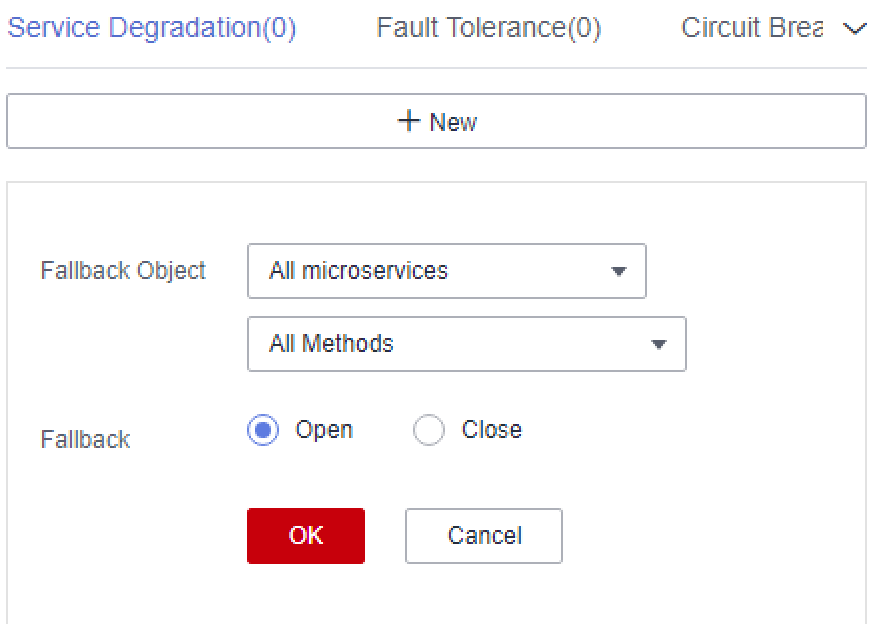
Step 7 Click **New** and select a proper policy. The following table describes the configuration items of service degradation.

Figure 10-7 Configuring service degradation (for microservices accessed through Spring Cloud)



Service Degradation Object: All microservices
Request Path
Service Degradation: [✓ Open] Close

Figure 10-8 Configuring service degradation (for microservices accessed through Java chassis)



Fallback Object: All microservices, All Methods
Fallback: [✓ Open] Close

Configuration Item	Description
Fallback Object	Microservice to be degraded.
Request Path **NOTE** This configuration applies to microservices accessed through Spring Cloud.	Click ◯ and set **Method**, **Path**, and **Headers** to specify the request path.
Fallback	• Open • Close

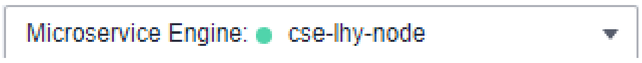
Step 8 Click **OK**.

----End

Configuring Fault Tolerance

Step 1 Log in to ServiceStage and choose **Cloud Service Engine** > **Engines**.

Step 2 Select the target microservice engine from the **Microservice Engine** drop-down list in the upper part of the page.



Step 3 Choose **Microservice Governance**.

- For microservice engines with security authentication disabled, go to **Step 5**.
- For microservice engines with security authentication enabled, go to **Step 4**.

Step 4 In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.

NOTE

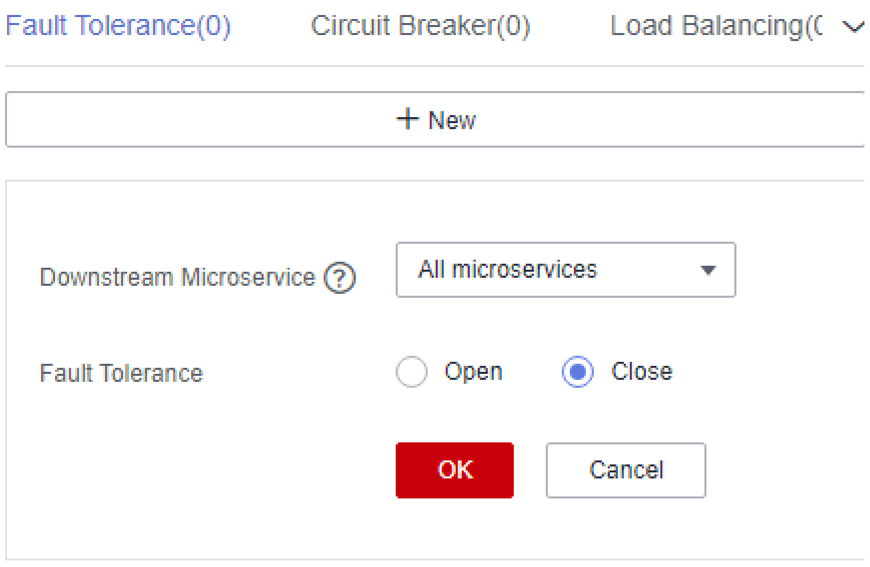
- If you connect to the microservice engine for the first time, enter the account name **root** and the password entered when **creating the microservice engine**.
- For details about how to create an account, see **Adding an Account**.

Step 5 Click the microservice to be governed.

Step 6 Click **Fault Tolerance**.

Step 7 Click **New** and select a proper policy. The following table describes the configuration items of fault tolerance.

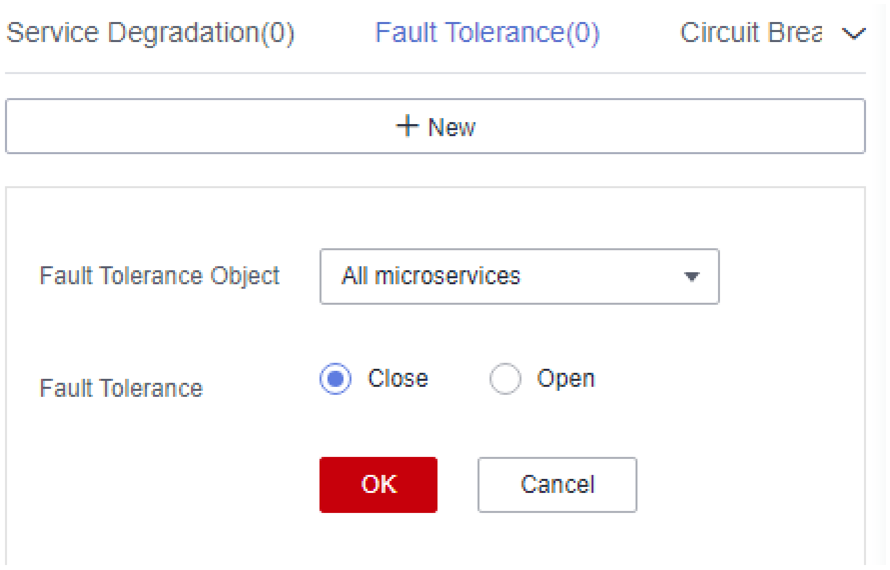
Figure 10-9 Configuring fault tolerance (for microservices accessed through Spring Cloud)



Downstream Microservice: All microservices

Fault Tolerance: Open [✓ Close]

Figure 10-10 Configuring fault tolerance (for microservices accessed through Java chassis)



Fault Tolerance Object: All microservices

Fault Tolerance: [✓ Close] Open

Configuration Item	Description
Downstream Microservice **NOTE** This configuration applies to microservices accessed through Spring Cloud.	Configure fault tolerance for the microservice to invoke the downstream microservice. You can select a value from the drop-down list.
Fault Tolerance Object **NOTE** This configuration applies to microservices accessed through Java chassis.	Microservice or method that the application relies on.
Fault Tolerance	**Open**: The system processes a request sent to the fault tolerance object based on the selected fault tolerance policy when the request encounters an error. **Close**: The system waits until the timeout interval expires and then returns the failure result even though the service request fails to be implemented.

Configuration Item	Description
FT Policy	This parameter is mandatory when **Fault Tolerance** is set to **Open**. For microservices accessed through Spring Cloud, set the following parameters: ● Number of attempts to the same microservice instance ● Number of attempts to the new microservice instance For microservices accessed through Java chassis, set the following parameters: ● Failover The system attempts to reestablish connections on different servers. ● Failfast The system does not attempt to reestablish a connection. After a request fails, a failure result is returned immediately. ● Failback The system attempts to reestablish connections on the same server. ● custom – Number of attempts to reestablish connections on the same server – Number of attempts to reestablish connections on new servers

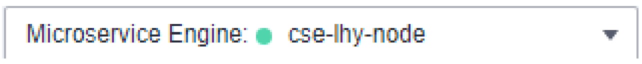
Step 8 Click **OK**.

----End

Configuring Circuit Breaker

Step 1 Log in to ServiceStage and choose **Cloud Service Engine** > **Engines**.

Step 2 Select the target microservice engine from the **Microservice Engine** drop-down list in the upper part of the page.


Microservice Engine: ● cse-lhy-node

Step 3 Choose **Microservice Governance**.

- For microservice engines with security authentication disabled, go to **Step 5**.
- For microservice engines with security authentication enabled, go to **Step 4**.

Step 4 In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.

NOTE

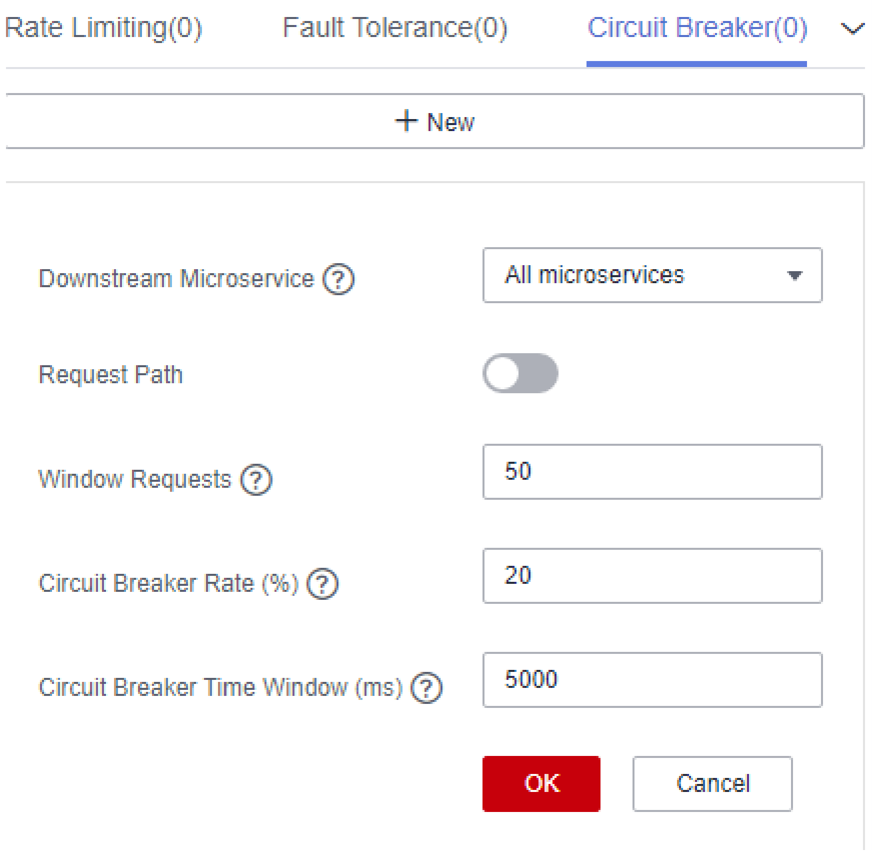
- If you connect to the microservice engine for the first time, enter the account name **root** and the password entered when **creating the microservice engine**.
- For details about how to create an account, see **Adding an Account**.

Step 5 Click the microservice to be governed.

Step 6 Click **Circuit Breaker**.

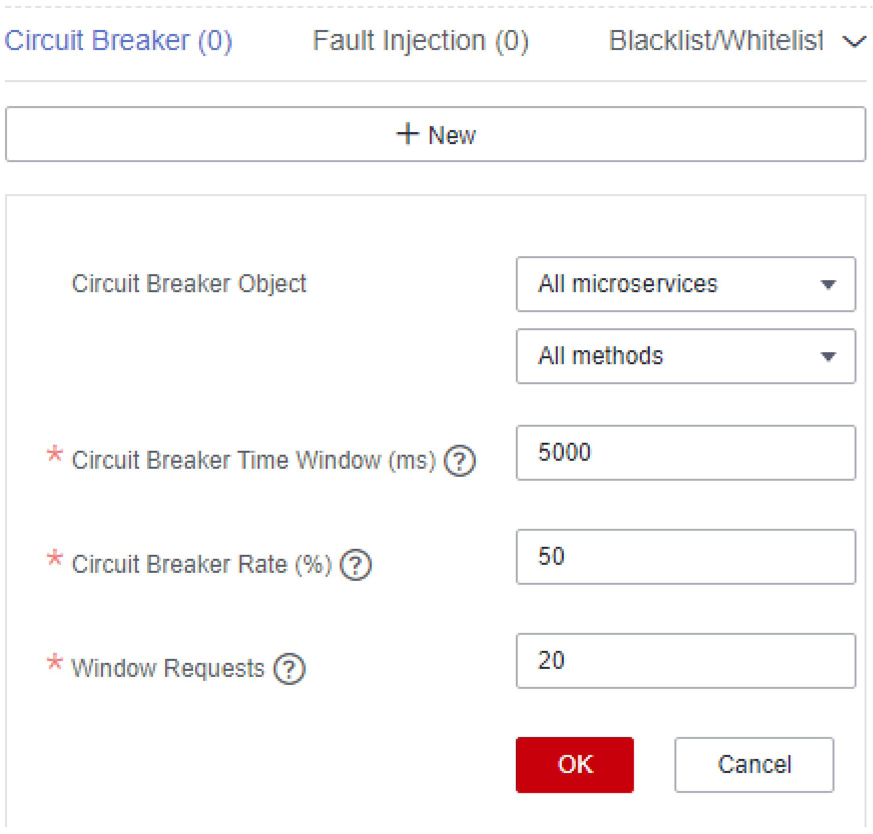
Step 7 Click **New** and select a proper policy. The following table describes the configuration items of circuit breaker.

Figure 10-11 Configuring circuit breaker (for microservices accessed through Spring Cloud)



Downstream Microservice	All microservices
Request Path	
Window Requests	50
Circuit Breaker Rate (%)	20
Circuit Breaker Time Window (ms)	5000

Figure 10-12 Configuring circuit breaker (for microservices accessed through Java chassis)



Circuit Breaker Object: All microservices / All methods

* Circuit Breaker Time Window (ms): 5000

* Circuit Breaker Rate (%): 50

* Window Requests: 20

Configuration Item	Description
Downstream Microservice **NOTE** This configuration applies to microservices accessed through Spring Cloud.	Configure circuit breaker for the microservice to invoke the downstream microservice.

Configuration Item	Description
Fallbreak Object **NOTE** This configuration applies to microservices accessed through Java chassis.	Microservice or method invoked by the application.
Request Path **NOTE** This configuration applies to microservices accessed through Spring Cloud.	Click and set **Method**, **Path**, and **Headers** to specify the request path.
Triggering Condition	• **Circuit Breaker Time Window**: circuit breaker duration. The system does not respond to requests within this time window. • **Request Failure Rate**: failure rate of window requests. • **Window Requests**: number of requests received by the window. Circuit breaker is triggered only when **Request Failure Rate** and **Window Requests** both reach their thresholds.

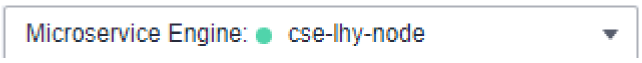
Step 8 Click **OK**.

----End

Configuring Fault Injection

Step 1 Log in to ServiceStage and choose **Cloud Service Engine** > **Engines**.

Step 2 Select the target microservice engine from the **Microservice Engine** drop-down list in the upper part of the page.



Microservice Engine: ● cse-lhy-node

Step 3 Choose **Microservice Governance**.

- For microservice engines with security authentication disabled, go to **Step 5**.
- For microservice engines with security authentication enabled, go to **Step 4**.

Step 4 In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.

NOTE

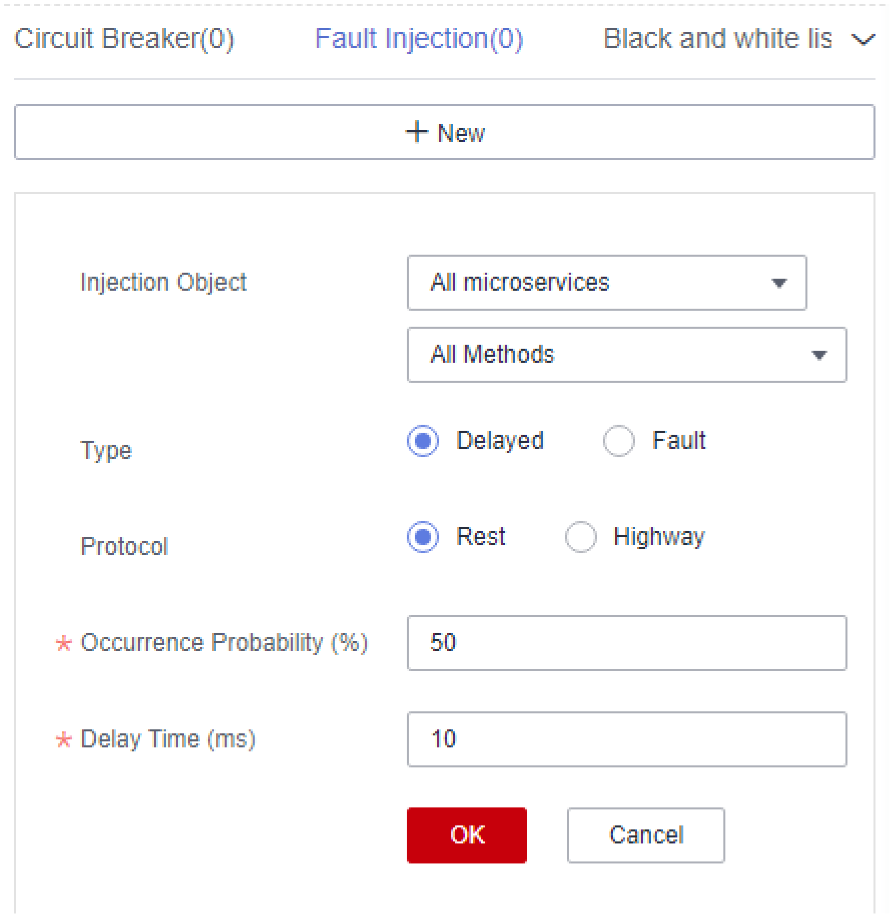
- If you connect to the microservice engine for the first time, enter the account name **root** and the password entered when **creating the microservice engine**.
- For details about how to create an account, see **Adding an Account**.

Step 5 Click the microservice to be governed.

Step 6 Click **Fault Injection**.

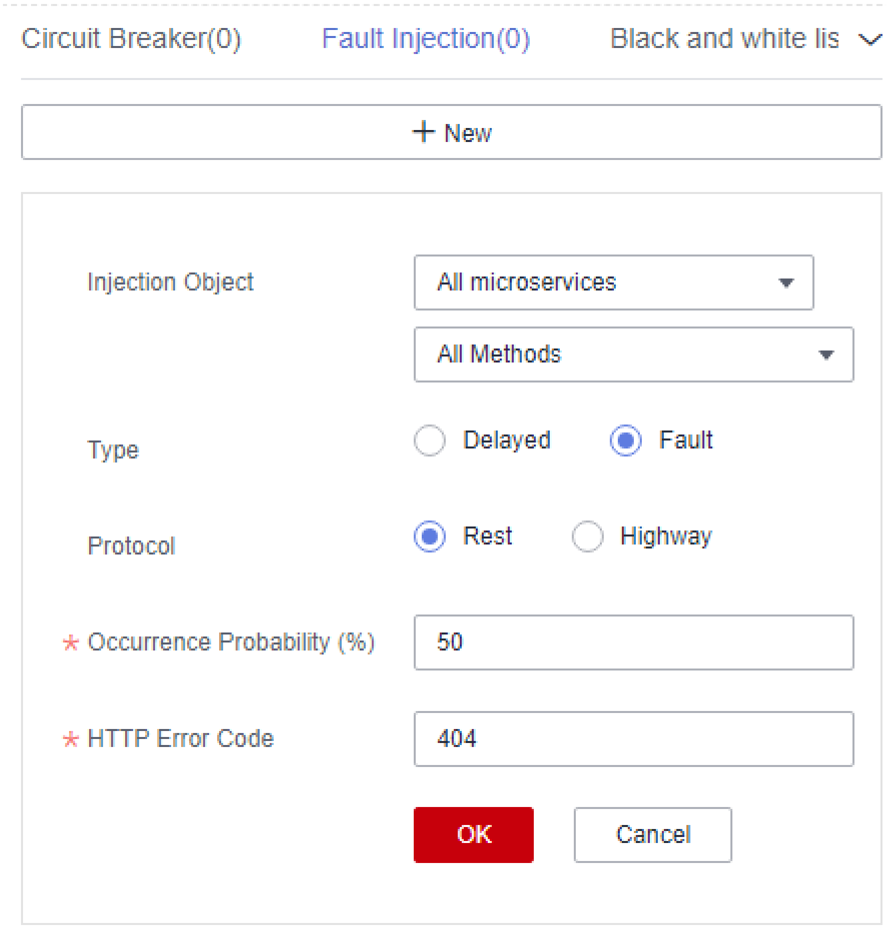
Step 7 Click **New** and select a proper policy. The following table describes the configuration items of fault injection.

Figure 10-13 Configuring fault injection (delayed)



Injection Object: All microservices; All Methods
Type: [✓ Delayed] Fault
Protocol: [✓ Rest] Highway
* Occurrence Probability (%): 50
* Delay Time (ms): 10

Figure 10-14 Configuring fault injection (fault)



Injection Object: All microservices; All Methods
Type: Delayed [✓ Fault]
Protocol: [✓ Rest] Highway
* Occurrence Probability (%): 50
* HTTP Error Code: 404

Configuration Item	Description
Injection Object	Microservices for which fault injection is required. You can specify a method for this configuration item.
Type	Type of the fault injected to the microservice. ● Delayed ● Fault
Protocol	Protocol for accessing the microservice when latency or fault occurs. ● Rest ● Highway
Occurrence Probability	Probability of latency or fault occurrence.
Delay Time	Duration of the latency during microservice access. This parameter is required when **Type** is set to **Delayed**.

Configuration Item	Description
HTTP Error Code	HTTP error code during microservice access. This parameter is required when **Type** is set to **Fault**. This error code is an HTTP error code.

Step 8 Click **OK**.

----End

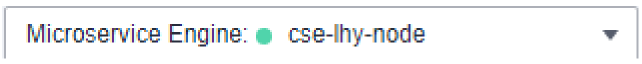
Configuring Blacklist and Whitelist

Based on the public key authentication mechanism, CSE provides the blacklist and whitelist functions. The blacklist and whitelist can be used to control which services can be accessed by microservices.

The blacklist and whitelist take effect only after public key authentication is enabled. For details, see **Configuring Public Key Authentication**.

Step 1 Log in to ServiceStage and choose **Cloud Service Engine** > **Engines**.

Step 2 Select the target microservice engine from the **Microservice Engine** drop-down list in the upper part of the page.


Microservice Engine: ● cse-lhy-node

Step 3 Choose **Microservice Governance**.

- For microservice engines with security authentication disabled, go to **Step 5**.
- For microservice engines with security authentication enabled, go to **Step 4**.

Step 4 In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.

NOTE

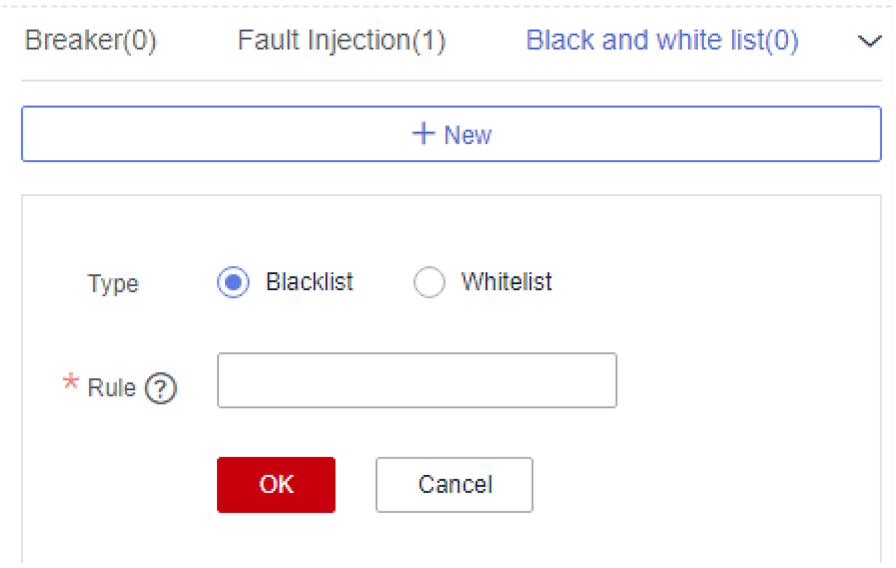
- If you connect to the microservice engine for the first time, enter the account name **root** and the password entered when **creating the microservice engine**.
- For details about how to create an account, see **Adding an Account**.

Step 5 Click the microservice to be governed.

Step 6 Click **Black and white list**.

Step 7 Click **New** to add a blacklist or whitelist for the application. The following table describes configuration items of blacklist and whitelist.

Figure 10-15 Configuring blacklist and whitelist



Configuration Item	Description
Type	• **Blacklist**: Microservices that match the matching rule are not allowed to access the current service. • **Whitelist**: Microservices that match the matching rule are allowed to access the current service.
Rule	Use a regular expression. For example, if **Rule** is set to **data***, services whose names start with **data** in the blacklist are not allowed to access the current service, or services whose names start with **data** in the whitelist are allowed to access the current service.

Step 8 Click **OK**.

----End

Configuring Public Key Authentication

Public key authentication is a simple and efficient authentication mechanism between microservices provided by CSE. Its security is based on the reliable interaction between microservices and the service center. That is, the authentication mechanism must be enabled between microservices and the service center. The procedure is as follows:

1. When the microservice starts, a key pair is generated and the public key is registered with the service center.
2. Before accessing the provider, the consumer uses its own private key to sign a message.
3. The provider obtains the public key of the consumer from the service center and verifies the signed message.

To enable public key authentication, perform the following steps:

1. Enable public key authentication for both the consumer and provider.
```
servicecomb:
  handler:
    chain:
      Consumer:
        default: auth-consumer
      Provider:
        default: auth-provider
```
2. Add the following dependency to the **pom.xml** file:
```
<dependency>
    <groupId>org.apache.servicecomb</groupId>
    <artifactId>handler-publickey-auth</artifactId>
  </dependency>
```

10.4.4 Configuration Management (Applicable to Microservice Engine 2.x)

CSE defines a configuration mechanism that is irrelevant to development frameworks. A configuration item consists of a key, label, and value. The label is used to identify whether a configuration item belongs to global configuration or microservice configuration. The label can also indicate the value type.

You can refer to the following table to select the operations to be performed.

Operation	Description
Creating an Application-Level Configuration	Associates the new configuration with an application, and adds the application name and environment label.
Creating a Microservice-Level Configuration	Associates the new configuration with a microservice, and adds the microservice name, application name, and environment.
Creating a Customized Configuration Item	If application-level and microservice-level configurations cannot meet service requirements, you can customize configuration files.
Importing Configurations	Imports the local configuration file.

Operation	Description
Exporting Configurations	Exports the selected configuration file to the local host.
Comparing Configuration Versions	Compares differences between historical versions.
Rolling Back a Version	Rolls back to the selected historical version.
Viewing Historical Versions	Displays configurations of different historical versions.
Editing a Configuration Item	Edits a configuration item.
Disabling a Configuration Item	Disables a configuration item.
Deleting a Configuration Item	Deletes a configuration item.

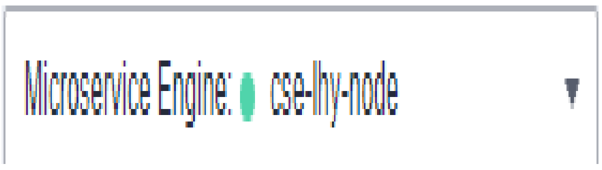
NOTE

When the configuration item quota specified by the engine specifications is about to be used up, the engine allows new configuration items that exceed the remaining quota to be created at the same time to ensure availability. Expand the capacity of the engine as soon as possible to prevent configuration creation failures.

Creating an Application-Level Configuration

Step 1 Log in to ServiceStage and choose **Cloud Service Engine** > **Engines**.

Step 2 Select the microservice engine where the application has been deployed but is to be governed from the **Microservice Engine** drop-down list in the upper part of the page.



Step 3 Click **Configuration Management**.

- For microservice engines with security authentication disabled, go to **Step 5**.
- For microservice engines with security authentication enabled, go to **Step 4**.

Step 4 In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.

NOTE

- If you connect to the microservice engine for the first time, enter the account name **root** and the password entered when **creating the microservice engine**.
- For details about how to create an account, see **Adding an Account**.

Step 5 Click **Create Configuration Item** and set parameters based on the following table. Parameters marked with an asterisk (*) are mandatory.

Parameter	Description
*Configuration Item	Enter a configuration item. The configuration item is the global ID of the configuration. In the coding phase, the configuration item is used to index and operate the configuration. You are advised to use the Java package naming rule (for example, **cse.service.registry.address**) to ensure the readability and uniqueness of the configuration.
Configuration Range	Select **Application-level**.
*Application	1. Select or enter an application name. 2. Select an environment.
Configuration Format	Select a configuration format.
*Configuration Content	Enter the configuration content.
Enable Configuration	Determine whether to enable the configuration item. ● **Enable now**: The configuration item takes effect immediately once created. ● **Not Enabled**: The configuration item does not take effect.

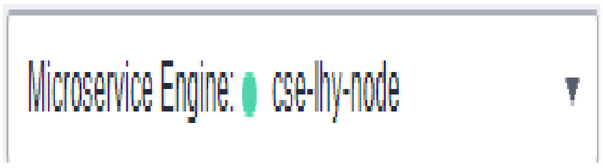
Step 6 Click **Create Now** to enable the configuration item.

----End

Creating a Microservice-Level Configuration

Step 1 Log in to ServiceStage and choose **Cloud Service Engine** > **Engines**.

Step 2 Select the microservice engine where the microservice has been deployed but is to be governed from the **Microservice Engine** drop-down list in the upper part of the page.



Step 3 Click **Configuration Management**.

- For microservice engines with security authentication disabled, go to **Step 5**.
- For microservice engines with security authentication enabled, go to **Step 4**.

Step 4 In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.

NOTE

- If you connect to the microservice engine for the first time, enter the account name **root** and the password entered when **creating the microservice engine**.
- For details about how to create an account, see **Adding an Account**.

Step 5 Click **Create Configuration Item** and set parameters based on the following table. Parameters marked with an asterisk (*) are mandatory.

Parameter	Description
*Configuration Item	Enter a configuration item. The configuration item is the global ID of the configuration. In the coding phase, the configuration item is used to index and operate the configuration. You are advised to use the Java package naming rule (for example, **cse.service.registry.address**) to ensure the readability and uniqueness of the configuration.
Configuration Range	Select **Microservice-level**.
*Microservice	1. Select or enter a microservice name. 2. Select or enter an application name. 3. Select an environment.
Configuration Format	Select a configuration format.
*Configuration Content	Enter the configuration content.
Enable Configuration	Determine whether to enable the configuration item. ● **Enable now**: The configuration item takes effect immediately once created. ● **Not Enabled**: The configuration item does not take effect.

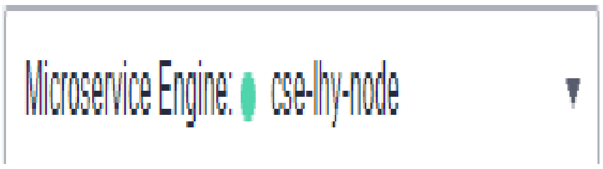
Step 6 Click **Create Now** to enable the configuration item.

----End

Creating a Customized Configuration Item

Step 1 Log in to ServiceStage and choose **Cloud Service Engine** > **Engines**.

Step 2 Select the target microservice engine from the **Microservice Engine** drop-down list in the upper part of the page.



Step 3 Click **Configuration Management**.

- For microservice engines with security authentication disabled, go to **Step 5**.
- For microservice engines with security authentication enabled, go to **Step 4**.

Step 4 In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.

NOTE

- If you connect to the microservice engine for the first time, enter the account name **root** and the password entered when **creating the microservice engine**.
- For details about how to create an account, see **Adding an Account**.

Step 5 Click **Create Configuration Item** and set parameters based on the following table. Parameters marked with an asterisk (*) are mandatory.

Parameter	Description
*Configuration Item	Enter a configuration item. The configuration item is the global ID of the configuration. In the coding phase, the configuration item is used to index and operate the configuration. You are advised to use the Java package naming rule (for example, **cse.service.registry.address**) to ensure the readability and uniqueness of the configuration.
Configuration Range	Select **Customize**.
*Labels	If application-level and microservice-level configurations cannot meet service requirements, you can use labels to customize configurations.
Configuration Format	Select a configuration format.
*Configuration Content	Enter the configuration content.

Parameter	Description
Enable Configuration	Determine whether to enable the configuration item. ● **Enable now**: The configuration item takes effect immediately once created. ● **Not Enabled**: The configuration item does not take effect.

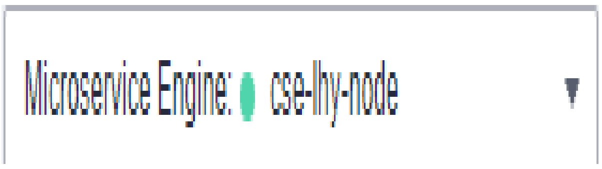
Step 6 Click **Create Now** to enable the configuration item.

----End

Importing Configurations

Step 1 Log in to ServiceStage and choose **Cloud Service Engine** > **Engines**.

Step 2 Select the target microservice engine from the **Microservice Engine** drop-down list in the upper part of the page.



Step 3 Click **Configuration Management**.

- For microservice engines with security authentication disabled, go to **Step 5**.
- For microservice engines with security authentication enabled, go to **Step 4**.

Step 4 In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.

NOTE

- If you connect to the microservice engine for the first time, enter the account name **root** and the password entered when **creating the microservice engine**.
- For details about how to create an account, see **Adding an Account**.

Step 5 Click **Import** in the upper right corner and set parameters by referring to the following table.

Parameter	Description
Import to a specific environment	● Disabled: The imported configuration does not change the environment label. ● Enabled: Importing the configuration to a specific environment will change the environment label.

Parameter	Description
Same Configuration	● **Terminate**: If a configuration is the same as that in the system, the import terminates. ● **Skip**: During import, if a configuration is the same as that in the system, the configuration is skipped and other configurations are imported. ● **Overwrite**: During import, if a configuration is the same as that in the system, the value of the configuration will be replaced.
Configuration File	Click **Import** and select the target file. **NOTE** The file size cannot exceed 2 MB.

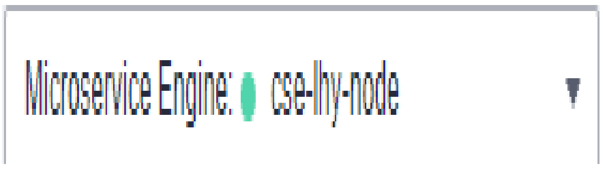
Step 6 Click **Close**.

----End

Exporting Configurations

Step 1 Log in to ServiceStage and choose **Cloud Service Engine** > **Engines**.

Step 2 Select the target microservice engine from the **Microservice Engine** drop-down list in the upper part of the page.



Step 3 Click **Configuration Management**.

- For microservice engines with security authentication disabled, go to **Step 5**.
- For microservice engines with security authentication enabled, go to **Step 4**.

Step 4 In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.

NOTE

- If you connect to the microservice engine for the first time, enter the account name **root** and the password entered when **creating the microservice engine**.
- For details about how to create an account, see **Adding an Account**.

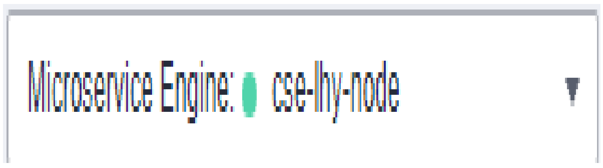
Step 5 Select the configuration items to be exported and click **Export**.

----End

Comparing Configuration Versions

Step 1 Log in to ServiceStage and choose **Cloud Service Engine** > **Engines**.

Step 2 Select the target microservice engine from the **Microservice Engine** drop-down list in the upper part of the page.



Step 3 Click **Configuration Management**.

- For microservice engines with security authentication disabled, go to **Step 5**.
- For microservice engines with security authentication enabled, go to **Step 4**.

Step 4 In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.

NOTE

- If you connect to the microservice engine for the first time, enter the account name **root** and the password entered when **creating the microservice engine**.
- For details about how to create an account, see **Adding an Account**.

Step 5 Click the configuration item to be compared.

Step 6 Click **View Historical Version**.

Step 7 In **Historical Versions** on the left, select the historical version to be viewed.

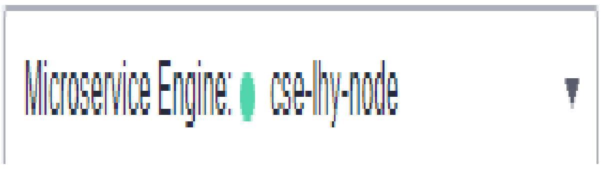
In the **Configuration file** on the right, you can view the differences between the current and historical versions.

----End

Rolling Back a Version

Step 1 Log in to ServiceStage and choose **Cloud Service Engine** > **Engines**.

Step 2 Select the target microservice engine from the **Microservice Engine** drop-down list in the upper part of the page.



Step 3 Click **Configuration Management**.

- For microservice engines with security authentication disabled, go to **Step 5**.
- For microservice engines with security authentication enabled, go to **Step 4**.

Step 4 In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.

NOTE

- If you connect to the microservice engine for the first time, enter the account name **root** and the password entered when **creating the microservice engine**.
- For details about how to create an account, see **Adding an Account**.

Step 5 Click the target configuration item.

Step 6 Click **View Historical Version**.

Step 7 In **Historical Versions** on the left, select the target historical version.

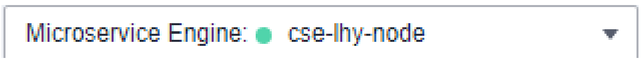
Step 8 In **Configuration file** on the right, click **Roll Back to the Selected Version**.

----End

Viewing Historical Versions

Step 1 Log in to ServiceStage and choose **Cloud Service Engine** > **Engines**.

Step 2 Select the target microservice engine from the **Microservice Engine** drop-down list in the upper part of the page.



Step 3 Click **Configuration Management**.

- For microservice engines with security authentication disabled, go to **Step 5**.
- For microservice engines with security authentication enabled, go to **Step 4**.

Step 4 In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.

NOTE

- If you connect to the microservice engine for the first time, enter the account name **root** and the password entered when **creating the microservice engine**.
- For details about how to create an account, see **Adding an Account**.

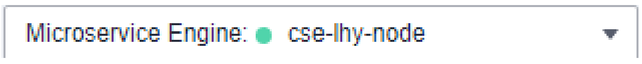
Step 5 Click **View Historical Version** in the **Operation** column of a configuration item. On the **Historical Versions** page that is displayed, you can view the historical versions of the configuration item. On this page, you can compare the configuration version with the rollback version.

----End

Editing a Configuration Item

Step 1 Log in to ServiceStage and choose **Cloud Service Engine** > **Engines**.

Step 2 Select the target microservice engine from the **Microservice Engine** drop-down list in the upper part of the page.



Step 3 Click **Configuration Management**.

- For microservice engines with security authentication disabled, go to **Step 5**.
- For microservice engines with security authentication enabled, go to **Step 4**.

Step 4 In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.

NOTE

- If you connect to the microservice engine for the first time, enter the account name **root** and the password entered when **creating the microservice engine**.
- For details about how to create an account, see **Adding an Account**.

Step 5 Click **Edit** in the **Operation** column of the target configuration item. You can also click the target configuration item and then click **Edit** on the displayed configuration details page.

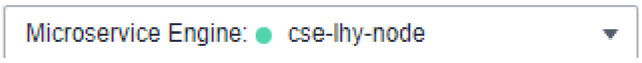
Step 6 Enter the configuration information in the **Configuration Content** text box and click **Save**.

----End

Disabling a Configuration Item

Step 1 Log in to ServiceStage and choose **Cloud Service Engine** > **Engines**.

Step 2 Select the target microservice engine from the **Microservice Engine** drop-down list in the upper part of the page.



Step 3 Click **Configuration Management**.

- For microservice engines with security authentication disabled, go to **Step 5**.
- For microservice engines with security authentication enabled, go to **Step 4**.

Step 4 In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.

NOTE

- If you connect to the microservice engine for the first time, enter the account name **root** and the password entered when **creating the microservice engine**.
- For details about how to create an account, see **Adding an Account**.

Step 5 In the **Operation** column of the target configuration item, click **More** > **Disable**.

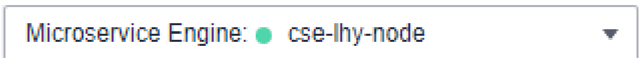
Step 6 Click **OK**.

----End

Deleting a Configuration Item

Step 1 Log in to ServiceStage and choose **Cloud Service Engine** > **Engines**.

Step 2 Select the target microservice engine from the **Microservice Engine** drop-down list in the upper part of the page.



Step 3 Click **Configuration Management**.

- For microservice engines with security authentication disabled, go to **Step 5**.
- For microservice engines with security authentication enabled, go to **Step 4**.

Step 4 In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.

NOTE

- If you connect to the microservice engine for the first time, enter the account name **root** and the password entered when **creating the microservice engine**.
- For details about how to create an account, see **Adding an Account**.

Step 5 Click **Delete** in the **Operation** column of the target configuration item. You can also click the target configuration item and then click **Delete** on the displayed configuration details page.

Step 6 Click **OK**.

----End

10.4.5 Configuration Management (Applicable to Microservice Engine 1.x)

The configuration added here is a global configuration. After being added, the configuration takes effect immediately if all microservices registered with the engine use it.

If dynamic configuration is set for a single microservice, the dynamic configuration overwrites the global configuration. For details about how to set dynamic configuration, see **Dynamic Configuration**.

Creating a Configuration

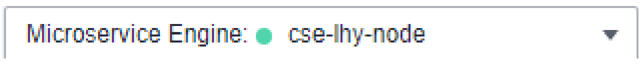
Configuration management provides common configurations for microservices, such as log levels and running parameters. After being added, the configuration item is used as the default one if no same configuration items are defined for microservices.

NOTICE

Configuration items are stored in plaintext. Do not include sensitive data.

Step 1 Log in to ServiceStage and choose **Cloud Service Engine** > **Engines**.

Step 2 Select the microservice engine where the application has been deployed but is to be governed from the **Microservice Engine** drop-down list in the upper part of the page.



Step 3 Click **Configuration Management**.

- For microservice engines with security authentication disabled, go to **Step 5**.
- For microservice engines with security authentication enabled, go to **Step 4**.

Step 4 In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.

NOTE

- If you connect to the microservice engine for the first time, enter the account name **root** and the password entered when **creating the microservice engine**.
- For details about how to create an account, see **Adding an Account**.

Step 5 Click **Create Configuration Item**.

Step 6 On the **Create Configuration Item** page, select a microservice environment and enter **Configuration Item** and **Value**.

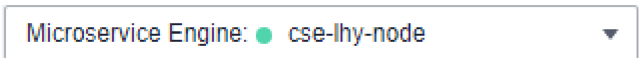
Step 7 Click **OK**.

----End

Importing Configurations

Step 1 Log in to ServiceStage and choose **Cloud Service Engine** > **Engines**.

Step 2 Select the microservice engine where the application has been deployed but is to be governed from the **Microservice Engine** drop-down list in the upper part of the page.



Step 3 Click **Configuration Management**.

- For microservice engines with security authentication disabled, go to **Step 5**.
- For microservice engines with security authentication enabled, go to **Step 4**.

Step 4 In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.

NOTE

- If you connect to the microservice engine for the first time, enter the account name **root** and the password entered when **creating the microservice engine**.
- For details about how to create an account, see **Adding an Account**.

Step 5 Click **Import**.

Step 6 Select a microservice environment, click **Import**, and select the target file.

NOTE

A maximum of 150 configuration items can be imported at a time.

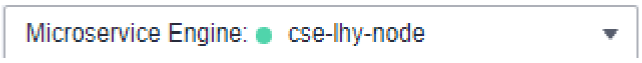
Step 7 Click **Close**.

----End

Exporting Configurations

Step 1 Log in to ServiceStage and choose **Cloud Service Engine** > **Engines**.

Step 2 Select the microservice engine where the application has been deployed but is to be governed from the **Microservice Engine** drop-down list in the upper part of the page.


Microservice Engine: cse-lhy-node

Step 3 Click **Configuration Management**.

- For microservice engines with security authentication disabled, go to **Step 5**.
- For microservice engines with security authentication enabled, go to **Step 4**.

Step 4 In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.

NOTE

- If you connect to the microservice engine for the first time, enter the account name **root** and the password entered when **creating the microservice engine**.
- For details about how to create an account, see **Adding an Account**.

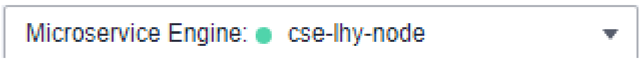
Step 5 Click **Export All**.

----End

Deleting a Configuration

Step 1 Log in to ServiceStage and choose **Cloud Service Engine** > **Engines**.

Step 2 Select the microservice engine where the application has been deployed but is to be governed from the **Microservice Engine** drop-down list in the upper part of the page.


Microservice Engine: cse-lhy-node

Step 3 Click **Configuration Management**.

- For microservice engines with security authentication disabled, go to **Step 5**.
- For microservice engines with security authentication enabled, go to **Step 4**.

Step 4 In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.

NOTE

- If you connect to the microservice engine for the first time, enter the account name **root** and the password entered when **creating the microservice engine**.
- For details about how to create an account, see **Adding an Account**.

Step 5 Select the target configuration item and click **Delete**. You can also click **Delete** in the **Operation** column of the target configuration item.

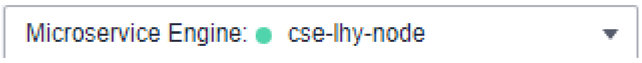
Step 6 Click **OK**.

----End

Editing a Configuration

Step 1 Log in to ServiceStage and choose **Cloud Service Engine** > **Engines**.

Step 2 Select the microservice engine where the application has been deployed but is to be governed from the **Microservice Engine** drop-down list in the upper part of the page.



Step 3 Click **Configuration Management**.

- For microservice engines with security authentication disabled, go to **Step 5**.
- For microservice engines with security authentication enabled, go to **Step 4**.

Step 4 In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.

NOTE

- If you connect to the microservice engine for the first time, enter the account name **root** and the password entered when **creating the microservice engine**.
- For details about how to create an account, see **Adding an Account**.

Step 5 Click **Edit** in the **Operation** column of the target configuration item and edit the values of the configuration item.

Step 6 Click **OK**.

----End

10.4.6 System Management

10.4.6.1 Overview

A microservice engine may be used by multiple users. Different users must have different microservice engine access and operation permissions based on their responsibilities and permissions.

The exclusive microservice engine with security authentication enabled provides the system management function using the role-based access control (RBAC) through the microservice console.

The exclusive microservice engine with security authentication enabled supports the access of Spring Cloud and Java chassis microservice frameworks.

NOTE

- The RBAC-based system management function is irrelevant to IAM permission management. It is only an internal permission management mechanism of CSE.
- To operate a microservice engine on CSE, you must have both the IAM and RBAC permissions, and the IAM permission takes precedence over the RBAC permission.
- If you perform operations on a microservice engine through APIs or the microservice framework, you only need to have the RBAC permissions.

1. You can use an account associated with the **admin** role to create an account and associate a proper role with the account based on service requirements. The user who uses this account has the access and operation permissions on the microservice engine.
   – When you create an exclusive microservice engine with security authentication enabled, the system automatically creates the **root** account associated with the **admin** role. The **root** account cannot be edited or deleted.

  - You can create an account using the **root** account of the microservice engine or an account associated with the **admin** role of the microservice engine. For details about how to create and manage an account, see **10.4.6.2 Accounts**.

2. You can create a custom role using an account associated with the **admin** role and grant proper microservice engine access and operation permissions to the role based on service requirements.
   - The system provides two default roles: administrator (**admin**) and developer (**developer**). Default roles cannot be edited or deleted.
   - You can create a custom role using the **root** account of the microservice engine or an account associated with the **admin** role of the microservice engine. For details about how to create and manage a role, see **10.4.6.3 Roles**.
   - For details about role permissions, see **Table 10-5**.

Table 10-5 Role permissions

Role	Permission Description
Admin	Full permissions for all microservices, accounts, and roles of the microservice engine.
Developer	Full permissions for all microservices of the microservice engine.
Custom role	You can create roles based on service requirements and grant microservice operation permissions to the roles.

10.4.6.2 Accounts

You can use an account associated with the **admin** role to log in to the microservice engine console and create an account or manage a specified account created in the engine based on service requirements.

Table 10-6 Account management operations

Operation	Description
Adding an Account	Creates an account and associates a proper role with the account. Users who use the account have the access and operation permissions on the microservice engine. You can create up to 1000 accounts.
Viewing Role Permissions	Displays the permissions of the role associated with a specified account.

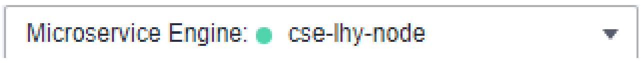
Operation	Description
Editing an Account	Adds or deletes roles for an account. The **root** account cannot be edited.
Changing the Password	Changes the password of an account that has logged in to the microservice engine based on service requirements or security regulations. NOTICE ● If the account and password are used to register a microservice in the SDK, changing the account and password may affect the service running of the microservice (the microservice cannot be registered with the microservice engine). As a result, the service system will be damaged. Exercise caution when performing this operation. ● After the password is changed, update the microservice authentication configuration in a timely manner. ● After the password is changed, the account may be locked due to three consecutive incorrect password attempts. The account will be unlocked after 15 minutes.
Deleting an Account	Deletes an account that is no longer used. The **root** account cannot be deleted. NOTICE If the account and password are used to register a service in the SDK, deleting the account will affect the service running (the account cannot be registered with the engine) and damage the service system. Exercise caution when performing this operation.

Adding an Account

Before adding an account, you can create a role based on service requirements. For details, see **Creating a Role**.

Step 1 Log in to ServiceStage and choose **Cloud Service Engine** > **Engines**.

Step 2 Select the target microservice engine with security authentication enabled from the **Microservice Engine** drop-down list in the upper part of the page.



Step 3 Choose **System Management**.

Step 4 In the displayed **Security Authentication** dialog box, enter the name and password of the account associated with the **admin** role under the microservice engine, and click **OK**.

NOTE

- If you connect to the microservice engine for the first time, enter the account name **root** and the password entered when **creating the microservice engine**.
- For details about how to create an account, see **Adding an Account**.

Step 5 Choose **Accounts** > **Create Account** and configure account parameters by referring to the following table:

Parameter	Description
Account	Enter an account name. **NOTE** The account name cannot be changed after the account is created.
Role	Select a role based on service requirements. **NOTE** An account can be associated with up to five roles.
Password	Enter the password.
Confirm Password	Enter the password again.

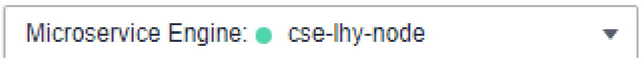
Step 6 Click **OK**.

----End

Viewing Role Permissions

Step 1 Log in to ServiceStage and choose **Cloud Service Engine** > **Engines**.

Step 2 Select the target microservice engine with security authentication enabled from the **Microservice Engine** drop-down list in the upper part of the page.



Step 3 Choose **System Management**.

Step 4 In the displayed **Security Authentication** dialog box, enter the name and password of the account associated with the **admin** role under the microservice engine, and click **OK**.

NOTE

- If you connect to the microservice engine for the first time, enter the account name **root** and the password entered when **creating the microservice engine**.
- For details about how to create an account, see **Adding an Account**.

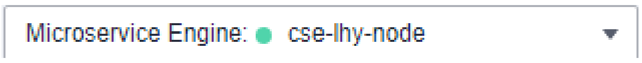
Step 5 Click the role in the **Role** column of the account to be viewed in the account list. On the displayed page, view the role and permission configuration associated with the account.

----End

Editing an Account

Step 1 Log in to ServiceStage and choose **Cloud Service Engine** > **Engines**.

Step 2 Select the target microservice engine with security authentication enabled from the **Microservice Engine** drop-down list in the upper part of the page.



Step 3 Choose **System Management**.

Step 4 In the displayed **Security Authentication** dialog box, enter the name and password of the account associated with the **admin** role under the microservice engine, and click **OK**.

NOTE

- If you connect to the microservice engine for the first time, enter the account name **root** and the password entered when **creating the microservice engine**.
- For details about how to create an account, see **Adding an Account**.

Step 5 On the **Accounts** tab page, click **Edit Account** in the **Operation** column of the account to be edited.

Step 6 Select a role based on service requirements.

NOTE

An account can be associated with up to five roles.

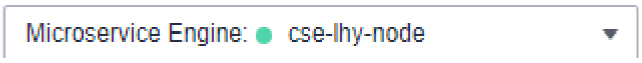
Step 7 Click **Save**.

----End

Changing the Password

Step 1 Log in to ServiceStage and choose **Cloud Service Engine** > **Engines**.

Step 2 Select the target microservice engine with security authentication enabled from the **Microservice Engine** drop-down list in the upper part of the page.



Step 3 Choose **System Management**.

Step 4 In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.

NOTE

- If you connect to the microservice engine for the first time, enter the account name **root** and the password entered when **creating the microservice engine**.
- The account for connecting to the microservice engine is not associated with the admin role. You can only change the password of the current login account.
- The account for connecting to the microservice engine is associated with the admin role. You can change the passwords of all accounts of the microservice engine.
- For details about how to create an account, see **Adding an Account**.

Step 5 On the **Accounts** tab page, select the account for logging in to the microservice engine and click **Reset Own Password** in the **Operation** column.

1. Enter the old password and a new password, and confirm the password.

2. After confirming that the password needs to be changed, select **I Understand**.

NOTE

You can also click **Reset Own Password** in the upper right corner of the **System Management** page to change the password of the current login account.

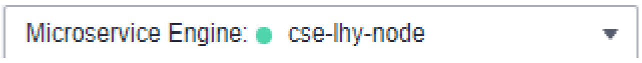
Step 6 Click **Save**.

----End

Deleting an Account

Step 1 Log in to ServiceStage and choose **Cloud Service Engine** > **Engines**.

Step 2 Select the target microservice engine with security authentication enabled from the **Microservice Engine** drop-down list in the upper part of the page.



Step 3 Choose **System Management**.

Step 4 In the displayed **Security Authentication** dialog box, enter the name and password of the account associated with the **admin** role under the microservice engine, and click **OK**.

NOTE

- If you connect to the microservice engine for the first time, enter the account name **root** and the password entered when **creating the microservice engine**.
- For details about how to create an account, see **Adding an Account**.

Step 5 On the **Accounts** tab page, click **Delete** in the **Operation** column of the account to be deleted.

Step 6 Click **OK**.

----End

10.4.6.3 Roles

In addition to the default roles **admin** and **developer**, you can use an account associated with the **admin** role to log in to the CSE console and perform operations listed in **Table 10-7** based on service requirements.

Table 10-7 Role management operations

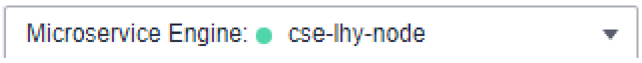
Operation	Description
Creating a Role	Creates a role and configures permission actions for the role in different service groups. A maximum of 100 roles can be created.
Editing a Role	Modifies the permissions of the created role.

Operation	Description
Deleting a Role	Deletes a role that is no longer used. **NOTE** ● Deleted roles cannot be restored. Exercise caution when performing this operation. ● Before deleting a role, ensure that the role is not associated with any account. For details about how to cancel the association between a role and an account, see **Editing an Account**.
Viewing a Role	Displays the created roles of the microservice engine based on the keyword of the role name.

Creating a Role

Step 1 Log in to ServiceStage and choose **Cloud Service Engine** > **Engines**.

Step 2 Select the target microservice engine with security authentication enabled from the **Microservice Engine** drop-down list in the upper part of the page.


Microservice Engine: ● cse-lhy-node

Step 3 Choose **System Management**.

Step 4 In the displayed **Security Authentication** dialog box, enter the name and password of the account associated with the **admin** role under the microservice engine, and click **OK**.

NOTE

- If you connect to the microservice engine for the first time, enter the account name **root** and the password entered when **creating the microservice engine**.
- For details about how to create an account, see **Adding an Account**.

Step 5 On the **Roles** tab page, click **Create Role**.

Step 6 Enter a role name.

NOTE

The role name cannot be changed after the role is created.

Step 7 Configure permissions.

1. Set **Service Group**.
   - If you select **All Services**:

     You can perform corresponding permission actions on all microservices of the microservice engine.
   - If you select **Custom Service Groups**, set the parameters according to **Table 10-8**.

Table 10-8 Custom service group operations

Operation	Description
Adding a Matching Rule	Click **Add Service Group Matching Rule**. Select **Application**, **Environment**, and **Service** based on service requirements to filter the microservices on which the role can perform permission actions. **NOTE** **Application**, **Environment**, and **Service** are three parameters of a microservice: ■ If only one parameter is set for a single matching rule, the role has the operation permission on the microservice that matches the parameter value. For example, if you add **Environment: production**, the role has the operation permission only on the microservice whose environment name is **production**. ■ If more than one parameter is set for a single matching rule, the role has the operation permission on the microservices that match all parameter values. For example, if you add **Environment: production Application: abc**, the role has the operation permission on the microservice whose environment name is **production** and application name is **abc**. ■ When automatic discovery is enabled, microservices query the instance addresses of services such as the registry center, configuration center, and dashboard through the registry center. When you grant the query permission to a microservice, the permission of the default application must be included. In this case, add the matching rule **Application: default**. After the microservice matching rule is set, click **OK**.
Editing a Matching Rule	Click ✎ next to the matching rule to be edited. You can reconfigure **Service Group** and **Action** of the matching rule based on service requirements. After the service group matching rule is configured, click **OK**.
Deleting a Matching Rule	Click 🗑 next to the matching rule to be deleted. You can delete the matching rule based on service requirements.

NOTE

A maximum of 20 microservice matching rules can be set for a custom service group.

If multiple matching rules are set for a custom service group, the role has the operation permission on the microservice as long as the microservice meets any of the matching rules.

2. Set **Action**.

Configure the permission actions that can be performed by the role on the selected service group based on service requirements. You can select multiple permission actions.

- **All**: Add, delete, modify, and query resources in the service group.
- **Add**: Add resources to the service group.
- **Delete**: Delete resources from the service group.

  NOTE

  If only **Delete** is selected, you cannot delete resources in the service group. You must select **View** at the same time.
- **Modify**: Modify resources in the service group.

  NOTE

  If only **Modify** is selected, you cannot modify resources in the service group. You must select **View** at the same time.
- **View**: View resources in the service group.

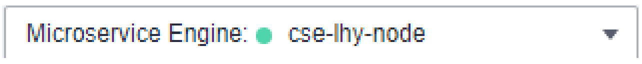
Step 8 Click **Create**.

----End

Editing a Role

Step 1 Log in to ServiceStage and choose **Cloud Service Engine** > **Engines**.

Step 2 Select the target microservice engine with security authentication enabled from the **Microservice Engine** drop-down list in the upper part of the page.



Step 3 Choose **System Management**.

Step 4 In the displayed **Security Authentication** dialog box, enter the name and password of the account associated with the **admin** role under the microservice engine, and click **OK**.

NOTE

- If you connect to the microservice engine for the first time, enter the account name **root** and the password entered when **creating the microservice engine**.
- For details about how to create an account, see **Adding an Account**.

Step 5 On the **Roles** tab page, click **Edit** in the **Operation** column of the role to be edited.

Step 6 Modify **Service Group** and **Action** based on service requirements.

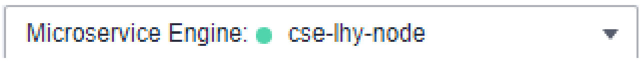
Step 7 Click **Save**.

----End

Deleting a Role

Step 1 Log in to ServiceStage and choose **Cloud Service Engine** > **Engines**.

Step 2 Select the target microservice engine with security authentication enabled from the **Microservice Engine** drop-down list in the upper part of the page.



Step 3 Choose **System Management**.

Step 4 In the displayed **Security Authentication** dialog box, enter the name and password of the account associated with the **admin** role under the microservice engine, and click **OK**.

NOTE

- If you connect to the microservice engine for the first time, enter the account name **root** and the password entered when **creating the microservice engine**.
- For details about how to create an account, see **Adding an Account**.

Step 5 On the **Roles** tab page, click **Delete** in the **Operation** column of the role to be deleted. In the displayed dialog box, enter **DELETE** and click **OK**.

NOTE

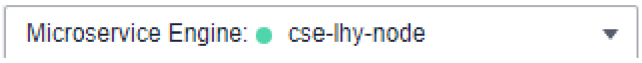
- Deleted roles cannot be restored. Exercise caution when performing this operation.
- Before deleting a role, ensure that the role is not associated with any account. For details about how to cancel the association between a role and an account, see **Editing an Account**.

----End

Viewing a Role

Step 1 Log in to ServiceStage and choose **Cloud Service Engine** > **Engines**.

Step 2 Select the target microservice engine with security authentication enabled from the **Microservice Engine** drop-down list in the upper part of the page.



Step 3 Choose **System Management**.

Step 4 In the displayed **Security Authentication** dialog box, enter the name and password of the account associated with the **admin** role under the microservice engine, and click **OK**.

NOTE

- If you connect to the microservice engine for the first time, enter the account name **root** and the password entered when **creating the microservice engine**.
- For details about how to create an account, see **Adding an Account**.

Step 5 On the **Roles** tab page, click ⌄ next to the role to be viewed to expand the role details.

Service Group and **Action** of the role are displayed.

----End

11 Key Operations Recorded by CTS

11.1 ServiceStage Operations That Can Be Recorded by CTS

Cloud Trace Service (CTS) records ServiceStage application management operations, enabling you to query, audit, and review operations.

After CTS is **enabled**, the system starts recording operations on ServiceStage resources. You can view the operation records of the last seven days on the CTS console.

Table 11-1 ServiceStage operations that can be recorded by CTS

Operation	Resource Type	Event Name
Creating a component	component	createComponent
Deleting a component	component	deleteComponent
Upgrading a component	component	updateComponent
Starting a component	component	startComponent
Stopping a component	component	stopComponent
Restarting a component	component	restartComponent
Scaling a component	component	scaleComponent

Operation	Resource Type	Event Name
Rolling back a component	component	rollbackComponent
Deploying a component	component	provisionComponent
Uninstalling a component	component	deprovisionComponent
Creating an application	application	createApplication
Deleting an application	application	deleteApplication
Updating an application	application	updateApplication
Creating an environment	environment	createEnvironment
Deleting an environment	environment	deleteEnvironment

11.2 Querying Audit Logs

For details about how to view audit logs, see **Querying Real-Time Traces**.

12 Viewing Monitoring Metrics and Alarms

Introduction

Application Operations Management (AOM) monitors and displays the running status of ServiceStage and the usage of each metric, and creates alarm rules for monitoring items.

After you use ServiceStage to deploy components, AOM can associate monitoring metrics of the components to help you master the performance metrics of the components in real time and accurately master the running status of the components.

Setting Monitoring and Alarms

CCE works with AOM to comprehensively monitor clusters. When a node is created, the ICAgent (the DaemonSet named **icagent** in the kube-system namespace of the cluster) of AOM is installed by default. The ICAgent collects monitoring data of underlying resources and workloads running on the cluster, and uploads the data to AOM. In addition, after **Customizing Component Running Metrics**, the ICAgent can collect monitoring data of user-defined load metrics and upload the data to AOM.

After **7.13.4 Configuring Alarm Thresholds for Resource Monitoring**, alarms generated during component running are reported to AOM.

Supported Metrics

Metrics reflect the resource performance or status.

Basic resource monitoring includes CPU, memory, and disk monitoring. For details, see **Table 12-1**.

- **Table 12-1** Resource metrics

Metric	Description	Value Range	Unit
Total CPU cores (cpuCoreLimit)	Total number of CPU cores that have been applied for a measured object	≥1	Cores
Used CPU cores (cpuCoreUsed)	Number of CPU cores used by a measured object	≥0	Cores
CPU usage (cpuUsage)	CPU usage of a measured object, that is, the ratio of the used CPU cores to the total CPU cores.	0%–100%	%
Total physical memory (memCapacity)	Total physical memory that has been applied for a measured object	≥0	MB
Physical memory usage (memUsage)	Percentage of the used physical memory to the total physical memory	0%–100%	%
Used physical memory (memUsed)	Used physical memory of a measured object	≥0	MB
Disk read rate (diskReadRate)	Volume of data read from a disk per second	≥0	KB/s
Disk write rate (diskWriteRate)	Volume of data written into a disk per second	≥0	KB/s
Downlink rate (recvPackRate)	Number of data packets received by the NIC per second	≥0	Packets per second (PPS)

Metric	Description	Value Range	Unit
Total file system (filesystemCapacity)	Total file system capacity of a measured object. This metric is available only for containers using the Device Mapper storage drive in the Kubernetes cluster of version 1.11 or later.	≥0	MB
Downlink rate (recvBytesRate)	Inbound traffic rate of a measured object	≥0	Byte per second (BPS)
Downlink error rate (recvErrPackRate)	Number of error packets received by an NIC per second	≥0	PPS
Uplink rate (sendPackRate)	Outbound traffic rate of a measured object	≥0	BPS
Uplink error rate (sendErrPackRate)	Number of error packets sent by the NIC per second	≥0	PPS
Uplink rate (sendBytesRate)	Outbound traffic rate of a measured object	≥0	BPS
Error packets (rxPackErrors)	Number of error packets received by a measured object	≥0	Packets
Threads (threadsCount)	Number of threads created on a host	≥0	N/A
Available file system (filesystemAvailable)	Available file system capacity of a measured object. This metric is available only for containers using the Device Mapper storage drive in the Kubernetes cluster of version 1.11 or later.	≥0	MB

Metric	Description	Value Range	Unit
File system usage (filesystemUsage)	File system usage of a measured object, that is, the ratio of the used file system to the total file system. This metric is available only for containers using the Device Mapper storage drive in the Kubernetes cluster of version 1.11 or later.	≥0	%
Handles (handleCount)	Number of handles used by a measured object	≥0	N/A
Component status (status)	Status of an application group	● 0: normal ● 1: abnormal	N/A
Total virtual memory (virMemCapacity)	Total virtual memory that has been applied for a measured object	≥0	MB

13 FAQs

13.1 Application Development FAQs

Key Information

To facilitate quick fault locating, provide detailed key information when posting an issue in the community. You are advised to provide a demo that can reproduce the fault.

The following uses ServiceComb Java chassis as an example:

1. Framework logs: By default, framework logs are printed with service logs, and the **cse.log** file is generated in the root directory. If the log framework such as Log4j2 or Logback is used on the service side, search for the key information based on the customized log policy.
   a. Key information about service startup:

Table 13-1 Key information about service startup

Keyword	Description
choose org.apache.servicecomb	ServiceComb Java chassis supports two types of REST communication channels. You need to determine the communication channel to be used based on logs. The **choose org.apache.servicecomb.transport.rest.vertx.VertxRestTransport** framework uses the REST over Vertx communication channel by default. That is, Vertx is used as the HTTP server. The **choose org.apache.servicecomb.transport.rest.servlet.ServletRestTransport** framework also supports the REST over Servlet communication channel. That is, other HTTP servers, such as Tomcat, are used.
endpoint to publish	Microservice release address.
Register microservice instance success	Flag indicating a successful service instance registration.

b. Key information about service calling:

Table 13-2 Key information about service calling

Keyword	Description
find instances	Before calling the server (called service), the consumer (calling service) queries the server instance from the service center of the microservice engine.
accesslog	The **access.log** file records the request sources, such as APIs and status codes for calling the service. By default, this function is disabled.

The **access.log** file printing is affected by the communication channel and log framework. If the REST over Vertx communication channel is used, the **access.log** file is recorded by Vertx.

The recommended format of the **access.log** file is as follows:

```
servicecomb.accesslog.pattern: "%h - - %t cs-uri %s %B %D %H %SCB-traceId"
```

By default, the **access.log** file is generated in the root directory. If the log framework such as Log4j2 or Logback is used on the service side, you can switch the log framework.

If the REST over Servlet communication channel is used, the **access.log** file is recorded by the HTTP server. To use the **access.log** file, find the related reference.

For example, enable the built-in Tomcat of Spring Boot as follows:

```
server:
  tomcat:
    accesslog:
      enabled: true
      pattern: '%h %l %u %t "%r" %s %b %D'
      directory: accesslogs
      buffered: false
    basedir: ./logs
```

2. Versions of the microservice engine and SDK. You can click the engine name to view the microservice engine version. For the SDK version, search for the dependency whose **groupId** is **org.apache.serivcecomb**.

13.2 Environment Management

13.2.1 What Are the Differences Between the Microservice and Platform Service?

The microservice is an architecture model used to build an application system. The platform service is the middleware service provided by the cloud.

You must purchase a platform service to use it To use a microservice, first develop it and release it on the cloud through the service discovery capability provided by the cloud.

13.3 Application Management

13.3.1 How Do I View the Causes of Application Component Deployment Failures?

Symptom

After the application component is deployed, the status is displayed as **Not Ready**, indicating that the application component fails to be deployed.

Solution

Step 1 Log in to ServiceStage.

Step 2 Use either of the following methods to go to the **Instance List** page.

- On the **Application Management** page, click the application to which the component belongs, and click the target component in **Component List**. In the left navigation pane, choose **Instance List**.
- On the **Component Management** page, click the target component. In the left navigation pane, choose **Instance List**.

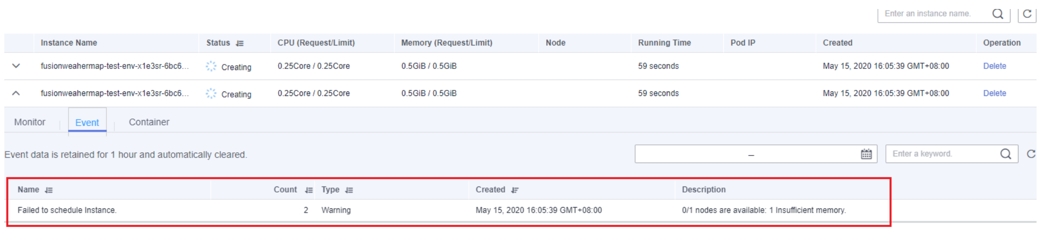
Step 3 In the instance list, click ﹀ next to the target instance.

Step 4 On the **Events** tab page, view a failure event and determine its cause.

----End

13.3.2 What If an Instance Is Being Created for a Long Time?

After an application component is created, if the service instance is in the **Not ready** state for a long time, go to the service instance list and check the instance details. On the **Event** tab page, you can see that the memory is insufficient.



	Instance Name	Status	CPU (Request/Limit)	Memory (Request/Limit)	Node	Running Time	Pod IP	Created	Operation
﹀	fusionweahermap-test-env-x1e3sr-6bc6...	Creating	0.25Core / 0.25Core	0.5GiB / 0.5GiB		59 seconds		May 15, 2020 16:05:39 GMT+08:00	Delete
︿	fusionweahermap-test-env-x1e3sr-6bc6...	Creating	0.25Core / 0.25Core	0.5GiB / 0.5GiB		59 seconds		May 15, 2020 16:05:39 GMT+08:00	Delete

Event data is retained for 1 hour and automatically cleared.

Name	Count	Type	Created	Description
Failed to schedule Instance.	2	Warning	May 15, 2020 16:05:39 GMT+08:00	0/1 nodes are available: 1 Insufficient memory.

You can solve this problem by adding nodes. For details, see **Creating a Node**.

13.3.3 How Do I Solve the Dependency Problem When a Node Program Runs in Docker?

Symptom

A node program depends on node-gyp when running in the microservice docker. How can I install the dependency before the program runs?

Solution

Customize a Dockerfile and add the node-gyp dependency to the Dockerfile.

13.3.4 How Do I Customize a Tomcat Context Path?

When creating and deploying a Tomcat application, Tomcat configurations are required. Specifically, the default **server.xml** configuration is used, the context path is **/**, and no application path is specified.

- If **Public Network Access** is enabled, the application access address is **http://***${Public domain name of the application}*:*${Application access port}*, for example, **http://example_domain.com:30317**.
- If **Public Network Access** is not enabled, the application access address is **http://***${Intranet access address of the VPC}*:*${Application access port}*, for example, **http://192.168.0.168:30317**.

During the component configuration for component deployment, you can customize the application path based on the actual service when configuring Tomcat parameters.

1. Select **Parameter settings**.
2. Click **Use Sample Code** and edit the template file based on service requirements.

3. Modify the value of **Context path** by referring to the following example. For example, after you change the value to **app-path**, the application access address is changed to **http://example_domain.com:30317/app-path** or **http://192.168.0.168:30317/app-path**.

```
<Host name="localhost"  appBase="webapps"
      unpackWARs="true" autoDeploy="true" >
   <Context path="app-path" docBase="ROOT.war"/>
```

13.3.5 How Do I Use a Fixed Application Component IP?

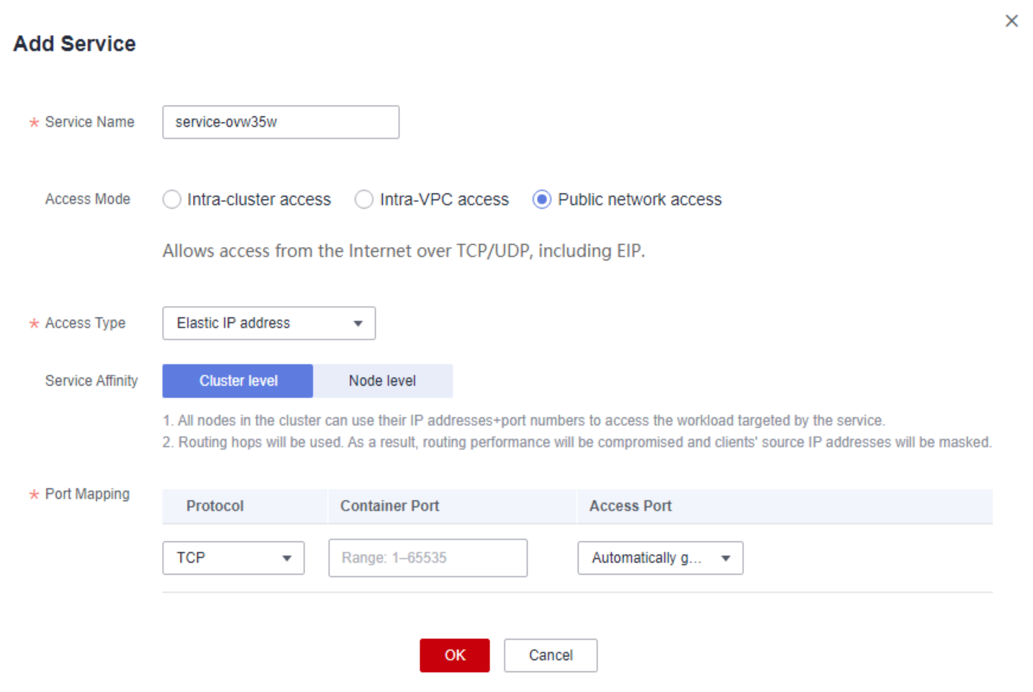
Symptom

If **TCP/UDP Route Configuration** is not set during application component deployment, the access IP address of the application changes when the container restarts. This may create difficulties in your configuration.

Solution

Set **TCP/UDP Route Configuration** when creating or deploying an application component. You can solve the problem using any of the following methods:

- Intra-cluster access: An application can be accessed by other applications in the same cluster using an internal domain name.
- Intra-VPC access: An application can be accessed by other applications in the same VPC using the IP address of a cluster node or the IP address of an ELB service in a private network.
- External access: An EIP is used to access applications from a public network. This access mode is applicable to services that need to be exposed to a public network in the system. In this access mode, EIP must be bound to any node in the cluster and a port mapped to the node must be configured.


Add Service

* Service Name: service-ovw35w

Access Mode: Intra-cluster access Intra-VPC access [✓ Public network access]

Allows access from the Internet over TCP/UDP, including EIP.

* Access Type: Elastic IP address

Service Affinity: [✓ Cluster level] Node level

1. All nodes in the cluster can use their IP addresses+port numbers to access the workload targeted by the service.
2. Routing hops will be used. As a result, routing performance will be compromised and clients' source IP addresses will be masked.

* Port Mapping

Protocol	Container Port	Access Port
TCP	Range: 1–65535	Automatically g...

13.3.6 How Do I Use the ServiceStage Source Code Deployment Function?

ServiceStage provides GitHub demos in different languages, as shown in **Table 13-3**.

You can fork the demo of a specific language to your GitHub code repository and experience the source code deployment function of ServiceStage by referring to **7.2 Creating and Deploying a Component**.

Table 13-3 Demos provided by ServiceStage and GitHub addresses

Demo	Language	GitHub Repository Address
ServiceComb-SpringMVC	Java	**https://github.com/servicestage-template/ServiceComb-SpringMVC**
ServiceComb-JAX-RS	Java	**https://github.com/servicestage-template/ServiceComb-JAX-RS**
ServiceComb-POJO	Java	**https://github.com/servicestage-template/ServiceComb-POJO**
SpringBoot-WebService	Java	**https://github.com/servicestage-template/SpringBoot-WebService**
SpringBoot-Webapp-Tomcat	Java	**https://github.com/servicestage-template/SpringBoot-Webapp-Tomcat**
nodejs-express	Node.js	**https://github.com/servicestage-template/nodejs-express-4-16**
nodejs-koa	Node.js	**https://github.com/servicestage-template/nodejs-koa-2-5-2**
php-laravel	PHP	**https://github.com/servicestage-template/php-laravel-v5-6-28**
php-slim	PHP	**https://github.com/servicestage-template/php-slim-3-10-0**

13.4 Continuous Delivery

13.4.1 How Does ServiceStage Manage Code on IDEA?

IDEA is a local IDE. You can encode on the IDE, upload the code to a code library, and select the source code repository for deployment.

If applications are developed based on the ServiceComb framework, select the source code repository for deployment and specify an engine to manage the applications.

13.4.2 How Do I Add the Build Server Address to the GitLab Server Security Group?

Background

If your GitLab service is built on the intranet of a cloud, and the public network cannot be accessed directly, add the address of the build service to your GitLab server's security group to ensure that the build task can run.

Procedure

Step 1 Add the network segment where ServiceStage is located to the security group of the node where the GitLab private repository is located. The build service uses this IP address segment to access the GitLab service API.

For details about how to set a security group, see **Adding a Security Group Rule**.

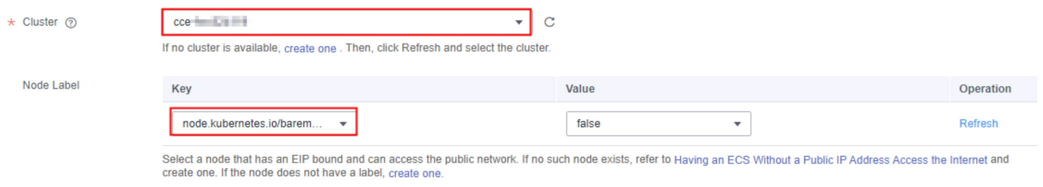
NOTE

For details about the network segment where ServiceStage is located, contact technical support.

Step 2 Obtain the cluster name and node label for creating an image.

- For application component building, obtain **Cluster** and **Node Label** by referring to **Editing a Source Code Job**.
- For build job building, obtain **Cluster** and **Node Label** by referring to **9.3 Creating a Source Code Job**.

Figure 13-1 Obtaining the cluster name and node label


* Cluster: cce[redacted]

If no cluster is available, create one . Then, click Refresh and select the cluster.

Node Label

Key	Value	Operation
node.kubernetes.io/barem...	false	Refresh

Select a node that has an EIP bound and can access the public network. If no such node exists, refer to Having an ECS Without a Public IP Address Access the Internet and create one. If the node does not have a label, create one.

Step 3 Obtain the EIP of the node in the cluster.

- Application component building
  a. Log in to ServiceStage and choose **Continuous Delivery** > **Build**.
  b. Click the name of the target cluster to enter its details page.
  c. Click **Nodes** to obtain the EIP of the node in the cluster.
- Job building
  a. Log in to ServiceStage and choose **Continuous Delivery** > **Build**.
  b. Select a build job and click a cluster to enter its details page.
  c. Click **Nodes** to obtain the EIP of the node in the cluster.

Step 4 Add the running node of the build image obtained in **Step 3** to the security group of the node where the GitLab private repository is located. During the build, the build service accesses the GitLab service to pull the code.

For details about how to set a security group, see **Adding a Security Group Rule**.

----End

13.4.3 How Do I Add the Build Server Address to the Maven Server Security Group?

Background

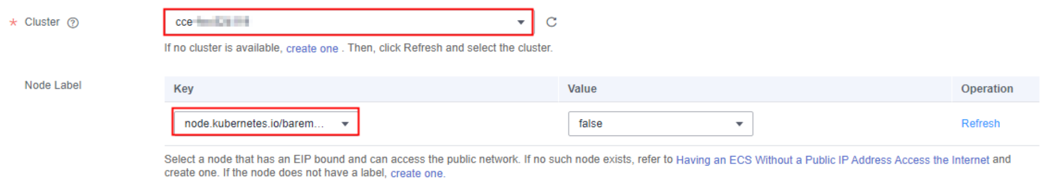
Add the EIP of the build node in the build cluster to the security group of the node where the private Maven service is located to enable the build service to access the Maven server to download the dependency package.

Procedure

Step 1 Obtain the cluster name and node label for creating an image.

- For application component building, obtain **Cluster** and **Node Label** by referring to **Editing a Source Code Job**.
- For build job building, obtain **Cluster** and **Node Label** by referring to **9.3 Creating a Source Code Job**.

Figure 13-2 Obtaining the cluster name and node label



* Cluster: cce[redacted]

If no cluster is available, create one . Then, click Refresh and select the cluster.

Node Label

Key	Value	Operation
node.kubernetes.io/barem...	false	Refresh

Select a node that has an EIP bound and can access the public network. If no such node exists, refer to Having an ECS Without a Public IP Address Access the Internet and create one. If the node does not have a label, create one.

Step 2 Obtain the EIP of the node in the cluster.

- Application component building
  a. Log in to ServiceStage and choose **Continuous Delivery** > **Build**.
  b. Click the name of the target cluster to enter its details page.
  c. Click **Nodes** to obtain the EIP of the node in the cluster.
- Job building
  a. Log in to ServiceStage and choose **Continuous Delivery** > **Build**.
  b. Select a build job and click a cluster to enter its details page.
  c. Click **Nodes** to obtain the EIP of the node in the cluster.

Step 3 Add the EIP of the build node in the build cluster to the security group of the node where the private Maven service is located.

For details about how to set a security group, see **Adding a Security Group Rule**.

----End

13.5 Infrastructure

13.5.1 What Should I Do If the Service Registration Fails After IPv6 Is Enabled for the Exclusive Microservice Engine with Security Authentication Enabled?

Symptom

A microservice developed based on Java chassis is registered with the exclusive microservice engine with security authentication enabled. The microservice registry center address is the IPv4 address of the microservice engine registry center. The microservice can be successfully registered and started.

If the microservice registry center address is changed to the IPv6 address of the microservice engine registry center, the registration fails and the error "java.net.SocketException: Protocol family unavailable" is reported.

Possible Cause

If you select the VPC network with IPv6 enabled when creating an exclusive microservice engine, the engine supports the IPv6 network. If the service is deployed using a container through an IPv6 network segment, the IPv6 dual-stack function must be enabled for the selected CCE cluster.

If IPv6 is not enabled for the selected cluster, the service network is disconnected, and the error "java.net.SocketException: Protocol family unavailable" is reported.

Solution

Step 1 Modify the environment where the microservice application is deployed by adding a CCE cluster with the IPv6 dual-stack function enabled.

Modify the environment. For details, see **5.6 Modifying an Environment**.

Step 2 Redeploy the application. For details, see **7.2 Creating and Deploying a Component**.

----End

13.5.2 What Should I Do If a Non-Microservice Engine Error Occurs When I Operate an Exclusive Microservice Engine?

Symptom

When you create, delete, or upgrade an exclusive microservice engine, a non-microservice engine error may occur.

For example, when you create an exclusive microservice engine, the cluster fails to be deployed and the following error message is displayed:

```
{"error_code":"SVCSTG.00500400","error_message":"{\"kind\":\"Status\",\"apiVersion\":\"v1\",\"metadata\":
{},\"status\":\"Failure\",\"code\":400,\"errorCode\":\"CCE.01400013\",\"errorMessage\":\"Insufficient volume
quota.\",\"error_code\":\"CCE_CM.0307\",\"error_msg\":\"Volume quota is not enough\",\"message
\":\"volume quota checking failed as [60/240] insufficient volume size quota\",\"reason\":\"QuotaInsufficient
\"}"}
```

Solution

The displayed error information contains the error code of the corresponding service. Contact the corresponding technical support.

13.6 Application O&M

13.6.1 Why Can't I View ServiceStage Logs?

Possible reasons why logs cannot be viewed on ServiceStage are:

- ICAgent was not installed on the host you are trying to view logs on.
- User service logs were output to a non-standard location.

Solution

- If ICAgent is not installed on your host:

  ServiceStage log viewing is provided by the Application Operations Management (AOM) service and requires ICAgent because it is the AOM collector running on each host to collect metrics, logs, and application performance data in real time.

  For details, see **Installing an ICAgent**.
- If logs are output to a non-standard location:

  Since log policies are user-defined, the service logs of the user program are not output to the standard output location. Perform the following steps:
  - VM-based deployment

    Check whether the configured log policy writes the user application service logs to the default VM log directory specified by ServiceStage: /var/log/application/${Component name}-${Environment name}-${Random character string}/${Version number}/${Instance ID}/start_app.log.

    Query the service code and adjust the log policy.
  - Container-based deployment

    Check the configured log policy to determine where service logs are output to. For details, see **7.17.7 Configuring a Log Policy of an Application**.

13.6.2 What Should I Do If Application Access Mode Becomes Invalid When EIP Is Replaced?

Symptom

When I bind a load balancer to an application and replace its EIP, the application access mode cannot be automatically updated.

Solution

Manually delete the original EIP, add a new EIP, and use the new EIP for ELB access.

13.6.3 What Should I Do If the Memory Usage of a Node Becomes Too High After a New Service Is Started?

Symptom

What should I do if the memory usage of a node is too high after a new service is started?

Solution

Set affinity by referring to **Setting Scheduling Policies for Component Instances** so that service instances will be deployed based on affinity nodes.