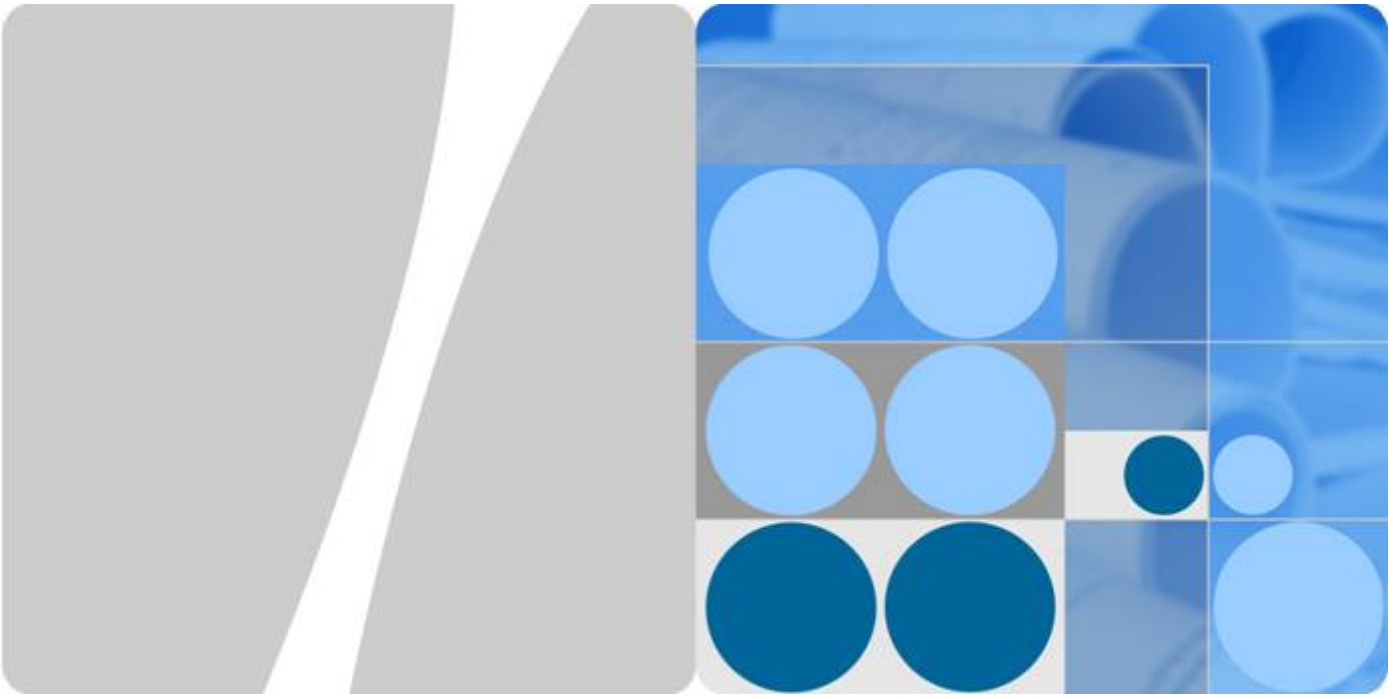


ServiceStage

User Guide

Date **2023-06-25**

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. Java, Go, PHP, Node.js, Python, Docker, and Tomcat are supported. Web applications, microservice applications such as Apache ServiceComb, Spring Cloud, Dubbo, and service mesh, and common applications make it easier to migrate enterprise applications to the cloud.

ServiceStage provides the following capabilities:

- Application management: supports application lifecycle management and environment management.
- Microservice application access: supports Java Chassis, Go Chassis, Spring Cloud and Dubbo microservice frameworks and ServiceComb Mesher; works with the microservice engine to implement service registry and discovery, configuration management, and service governance.
- AOM: supports application O&M through logs, monitoring, and alarms.

Application Development

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.

- Compatible with mainstream microservice open-source frameworks.

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

Application 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, software packages (JAR, WAR, or ZIP), and container images, achieving application deployment.
- Entire process management from application creation to logout, covering application creation, deployment, start, upgrade, rollback, scaling, stop, and deletion.
- Multi-dimensional metrics monitoring for application components, helping you understand the running status of online applications.
- GUI-based log query and search, help you quickly locate faults.

Microservice Governance

After an application developed using the microservice framework is managed on ServiceStage, the microservice will be registered with the service center after the application instance starts. You can govern microservices. The supported service governance policies are described in **Table 1-1**.

Table 1-1 Service governance policies

Name	Description
Load Balancing	When the access traffic and traffic volume are large and one server cannot be loaded, you can configure load balancing to distribute traffic to multiple servers for load balancing. This reduces latency and prevents server overload.
Rate Limiting	When the number of requests sent per second 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.
Fault Tolerance	Fault tolerance is a processing policy when an exception occurs in a service instance. After an exception occurs, the system retries or accesses a new service instance based on the defined policy.

Name	Description
Service Degradation	Service degradation is a special form of fault tolerance. When the service throughput is large and resources are insufficient, you can use service degradation to disable some services that are not important and have poor performance to avoid occupying resources and ensure that the main services are normal.
Circuit Breaker	If the service is overloaded due to certain reasons, you can use circuit breaker to protect the system from being faulty.
Fault Injection	Fault injection is used to test the fault tolerance capability of microservices. This helps the user ensure whether the system can run properly when latency or fault occurs.
Black and white list	Blacklist and whitelist allow you to set the information that is associated with routes.

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 1-2 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)

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, Go, 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 configuration	• Complex configuration items • All involved environments configured separately • Error-prone	• Configuration files can be imported to applications. • The configuration file is decoupled from the environment so that the configuration file can be separately maintained but shared by multiple environments. • The configuration file supports multiple versions, which facilitates system update and rollback.
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

Service Scenarios

Application Scenarios

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 runtime environments such as Java, Go, PHP, Node.js, Docker and Tomcat, and manages microservice applications such as Apache ServiceComb, Spring Cloud, Dubbo, and Service Mesh 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.

Advantages

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

- Supports multiple microservice frameworks, such as native ServiceComb, Spring Cloud, Dubbo, and Service Mesh, 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, Go, .NET, Node.js, PHP, and Python.
- Provides functions such as the service center, configuration center, dark launch, and dashboard.
- Provides complete microservice governance policies, including 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.
- Implements mutual discovery between Spring Cloud, ServiceComb, Java Chassis, and Go Chassis.

Continuous Integration and Delivery

Application Scenarios

It takes a lot of manpower and time in project creation, compilation, building, 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

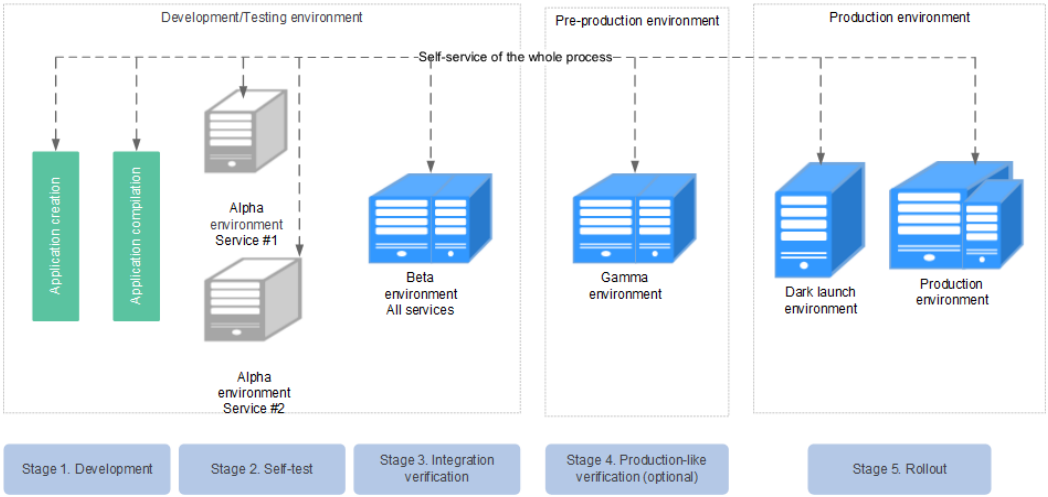
Manual execution is changed to automatic execution, which reduces errors and improves work efficiency.

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

Advantages

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 1-1 Continuous Integration and Delivery



Dark Launch

Application Scenarios

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.

Advantages

ServiceStage provides microservice-level dark launch.

1.3.2 Web Application Lifecycle Management

Service Scenarios

Application Scenarios

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.

Advantages

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 and DCS.

1.4 Glossary

Environment

Environment is a collection of infrastructures, covering computing, storage, and networks, used for application deployment and running. ServiceStage combines basic resources, such as Cloud Container Engine (CCE) and Elastic Cloud Server (ECS) and optional resources, such as Elastic Load Balance (ELB) and Distributed Cache Service (DCS) in the same VPC into an environment, such as the development environment, testing environment, pre-production environment, and production environment. Networks in an environment can communicate with each other. You can manage resources and deploy services by environment, simplifying infrastructure O&M.

Infrastructure

In ServiceStage, infrastructure refers 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 application components.

Application Component

An application component implements a service feature of an application. It is in the form of code or software packages and can be deployed independently.

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 an application component.

Microservice instance

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

1.5 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 1-3**.

Table 1-3 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

Item	Restrictions
Microservice instance registration	10 per second

For details about the restrictions on exclusive microservice engines, see **Table 1-4**.

Table 1-4 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	-

1.6 Specifications

Restrictions on the Maximum Number of Microservice Engine Services

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 engines can be added. Exclusive engines are exclusively used; therefore, the performance is not affected by tenants.

The maximum number of service instances supported is described as follows:

Table 1-5 Restrictions on the maximum number of microservice engine services

Type	Instances
Professional microservice engine	1000
Exclusive microservice engine	100
	500
	2000

1.7 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. For more information about IAM, 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 1-6 lists all the system policies supported by ServiceStage.

Table 1-6 Permissions description

Role/Policy Name	Description	Type	Dependencies
ServiceStage FullAccess	Full permissions for ServiceStage.	System-defined policy	None
ServiceStage Developer	ServiceStage developer, who has permissions to perform operations on applications, components, and environments, but does not have permissions to approve applications or create infrastructure. .	System-defined policy	None
ServiceStage ReadOnlyAccess	Read-only permissions for ServiceStage.	System-defined policy	None
CSE Admin	Administrator permissions for CSE.	System-defined policy	None
CSE Viewer	View permissions for CSE.	System-defined policy	None

If these policies do not meet actual requirements, you can customize policies. For more information, see **Creating a Custom Policy**.

Table 1-7 lists the common operations supported by each system-defined policy of ServiceStage. Please choose proper system-defined policies according to this table.

Table 1-7 Common operations supported by each system policy

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

Operation	ServiceStage ReadOnlyAccess	ServiceStage Developer	ServiceStage FullAccess	CSE Viewer	CSE Admin
Querying a pipeline	√	√	√	x	x
Starting a pipeline	x	√	√	x	x
Cloning a pipeline	x	√	√	x	x
Deleting a pipeline	x	√	√	x	x
Creating repository authorization	x	√	√	x	x
Modifying repository authorization	x	√	√	x	x
Querying repository authorization	√	√	√	x	x
Deleting repository authorization	x	√	√	x	x
Creating a microservice engine	x	x	√	x	√
Maintaining a microservice engine	x	x	√	x	√
Querying a microservice engine	√	√	√	√	√
Deleting a microservice engine	x	x	√	x	√

Operation	ServiceStage ReadOnlyAccess	ServiceStage Developer	ServiceStage FullAccess	CSE Viewer	CSE Admin
Registering a microservice	√	√	√	x	√
Configuring a microservice	√	√	√	x	√
Governing a microservice	x	√	√	x	√

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 1-8** describes fine-grained permission dependencies of CSE.
- **Table 1-9** describes fine-grained permission dependencies of ServiceStage.

Table 1-8 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)

Permission Name	Description	Permission Dependency	Application Scenario
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	Engine deletion (supported by only exclusive microservice engines.)
cse:engine:create	Creates an engine.	• cse:engine:get • cse:engine:list	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
cse:config:get	Views configuration and management functions.	• cse:engine:list • cse:engine:get	Service configuration view

Permission Name	Description	Permission Dependency	Application Scenario
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 1-9 Fine-grained permission dependencies of ServiceStage

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

Permission Name	Description	Permission Dependency	Application Scenario
servicestage:assembling:modify	Modifies a build job.	None	Build job modification
servicestage:assembling:delete	Deletes a build job.	None	Build job deletion
servicestage:assembling:list	Views the build list.	None	Build list view

2 Getting Started

Overview

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

An environment consists of basic resources (such as CCE clusters and ECS) and optional resources (such as ELB and DCS) in the same VPC. When deploying applications or components, you need to select an environment. After you select an environment, the resources 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, upgrading, rolling back, and scaling application component instances, viewing logs, viewing events, setting access modes, and setting threshold alarms.

The following describes how to set up an environment and create microservice applications and components in ServiceStage.

Prerequisites

- Create a VPC. For details, see **Creating a VPC**.
- 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.
- Create a bucket for storing JAR packages. For details, see
- Create an exclusive microservice engine. For details, see **Creating an Exclusive Microservice Engine**.

- 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.

Upload an object. For details, see **Uploading a File**.

----End

Creating an Organization

Step 1 Log in to ServiceStage and choose **Software Center** > **Organization**.

Step 2 Click **Create Organization**. On the displayed page, set **Organization Name**.

Step 3 Click **OK**.

----End

Creating an Environment

Step 1 Log in to ServiceStage, choose **Environment Management**, and click **Create Environment**.

Step 2 Configure the environment parameters by referring to the following table.

Parameter	Description
Environment	Enter an environment name, for example, **test-env**.
VPC	Select the VPC prepared in **Prerequisites**. **NOTE** After a VPC is selected, basic resources and optional resources in the VPC are loaded. Resources that are not in the VPC cannot be selected.
Basic Resources	Select the basic resources in the VPC. In this example, the CCE created in **Prerequisites** is used.
Optional Resources	Select optional resources in the VPC. In this example, the exclusive microservice engine created in **Prerequisites** is used.

Step 3 Click **Create Now**.

----End

Creating an Application

Step 1 Log in to ServiceStage, choose **Application Management** > **Application List**, and click **Create Application**.

Step 2 Set basic application information.

1. **Name**: Enter an application name, for example, **test-servicestage**.
2. **Description**: (Optional) Enter an application description.

Step 3 Click **OK**.

----End

Creating a Component

Step 1 Log in to ServiceStage and choose **Application Management** > **Application List**.

Step 2 Select application **test-servicestage** and click **Create Component** in the **Operation** column.

Step 3 Select **Template** for **Configuration Method** and select the **ServiceComb** template. Click **Next**.

Step 4 Set basic component information.

1. **Name**: Enter a component name, for example, **test-cse**.
2. **Upload Method**: Click **Software Package** and select the **servicecomb-0.0.1-SNAPSHOT.jar** software package that has been uploaded to the OBS bucket in **Preparing Software Packages**.

Step 5 Set **Build** parameters.

1. Select the organization created in **Creating an Organization**.

2. Select the CCE cluster used in **Creating an Environment**.
3. Retain the default values for other parameters.

Step 6 Click **Create Now** to create a static component.

----End

Deploying a Component

Step 1 Log in to ServiceStage and choose **Application Management** > **Application List**.

Step 2 Click the created application **test-servicestage**. The **Overview** page is displayed.

Step 3 On the **Component List** page, select the created component **test-cse** and click **Deploy** in the **Operation** column.

Step 4 Configure basic information according to the following table. Parameters marked with an asterisk (*) are mandatory.

Parameter	Description
*Environment	Select the created environment **test-env**.
*Version	The default version is **1.0.0**.
*Deployment System	Select **Cloud Container Engine**.
*Basic Resources	Use CCE resources (automatically loaded) in the **test-env** environment.
*Instances	Set this parameter to **1**.
Resource Quota	Retain the default value.

Step 5 Click **Next** to configure the component.

1. Select **Cloud Service Engine**. The exclusive microservice engine in the **test-env** environment is used by default.

   NOTE

   - After an application component is deployed, the microservice is registered with the selected microservice engine.
   - All application components must be registered with the same microservice engine so that they can discover each other.
2. Retain the default values for other parameters.

Step 6 Click **Next** to confirm the specifications.

Step 7 Click **Deploy** to deploy the component.

----End

Confirming the Deployment Result

Step 1 Log in to ServiceStage and choose **Infrastructure** > **Cloud Service Engines**.

Step 2 Select the exclusive microservice engine used in **Deploying a Component** and click **Console** to access the microservice console.

Step 3 Choose **Service Catalog** > **Microservice List** and select **springmvc** from the **All applications** drop-down list.

If the microservice servicecombspringmvc is displayed and the number of microservice instances is the same as that set in **Deploying a Component**, the deployment is successful.

----End

Accessing an Application

Step 1 Log in to ServiceStage and choose **Application Management** > **Application List**.

Step 2 Click the created application **test-servicestage**. The **Overview** page is displayed.

Step 3 On the **Environment View** tab, select **test-env** for **Environment** to view the deployed application components.

Step 4 Click the component **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.
Port Mapping	1. **Protocol**: Select **TCP**. 2. **Container Port**: Enter **8080**. 3. **Access Port**: Select **Automatically generated**.

Step 7 Click **OK**.

Figure 2-1 Access address

Step 8 Click the access address in the **Access Address** column to access the application, as shown in **Figure 2-1**.

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, choose **Application Management** > **Application List**, and click application **test-servicestage**. The **Overview** page is displayed.

Step 2 On the **Component List** page, click component **test-cse**. The **Overview** page is displayed. You can view the component version in the **Environment** area.

Step 3 Click **View Details** of an environment. On the component details page that is displayed, you can perform O&M operations, such as starting, stopping, upgrading, rolling back, and scaling application component instances, viewing logs, viewing events, setting the access mode, and setting threshold alarms.

----End

3 User Guide

3.1 Overview

ServiceStage is an application management and O&M platform that lets you deploy, roll out, monitor, and maintain applications all in one place. Java, Go, PHP, Node.js, Python, Docker, and Tomcat are supported. Web applications, microservice applications such as Apache ServiceComb, Spring Cloud, Dubbo, and service mesh, and common applications make it easier to migrate enterprise applications to the cloud.

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

Console Description

Table 3-1 describes ServiceStage console.

Table 3-1 ServiceStage console

Module	Description
Overview	Provides ServiceStage overview, including the tutorials, applications, environments, and components.
Application Management	• Application List Provides application lifecycle management, such as application creation, component addition, component list, environment view, component deployment, component details, and O&M. • Application Configuration Supports configuration item and secret management.

Module	Description
Environment Management	An environment is a collection of compute, storage, and network resources used for deploying and running an application. Provides environment creation, editing, and deletion, and displays resource information in an existing environment.
Continuous Delivery	Supports project build and release. • 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.
Software Center	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.
Infrastructure	Provides application infrastructure management, such as CSE and VM agent. On the CSE page, go to its console to perform microservice governance.

NOTE

The VM agent management function depends on the ECS and AOM services. If these services are not installed, the VM agent management function is unavailable.

3.2 Permissions Management

3.2.1 Creating a User and Granting Permissions

This topic describes how to use IAM to implement fine-grained permissions control for 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 a account or cloud service to perform efficient O&M on your ServiceStage resources.

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

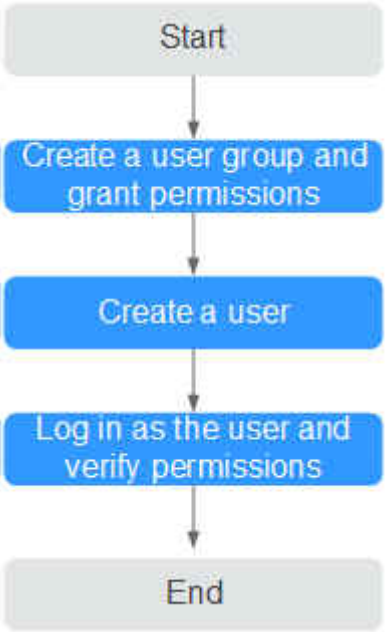
This section describes the procedure for granting permissions (see **Figure 3-1**).

Prerequisites

Before assigning permissions to user groups, you should learn about the ServiceStage permissions listed in **Permissions Management**.

Process Flow

Figure 3-1 Process for granting ServiceStage permissions



1. Create a user group and grant permissions to it.

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

2. Create a user.

   Create a user on the IAM console and add the user to the group created in 1.

3. Log in 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**. Choose **Application Management** > **Application List** from the navigation tree. On the page that is displayed, 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.

3.2.2 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: Edit JSON policies 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 modify 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 console 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 ServiceStage console, choose **Infrastructure** > **Cloud Service Engines**, and create a microservice engine. If error 403 is returned, the permissions are correct and have already taken effect.

3.2.3 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 about how to set CCE namespace permissions, 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.

3.3 Application Management

3.3.1 Creating an Application

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

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.

Creating an Application

Step 1 Log in to ServiceStage and choose **Application Management** > **Application List**.

Step 2 Click **Create Application** and set basic application information.

1. **Name**: Enter an application name. This name cannot be changed after the application is created.
2. **Description**: (Optional) Enter an application description.

Step 3 Click **OK**.

----End

Adding Environment Variables

An environment is a collection of compute, storage, and network resources used for deploying and running an application. ServiceStage combines basic resources (such as CCE clusters and ECSs) and optional resources (such as ELB instances and DCS instances) in the same VPC 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.

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.

Step 1 Log in to ServiceStage and choose **Application Management** > **Application List**.

Step 2 Click an application. The **Overview** page is displayed.

Step 3 Click **Environment Variables** and select a created environment from the **Environment** drop-down list.

Step 4 Click **Add Environment Variable** and enter the values in **Key** and **Value**.

Key indicates the name of the environment variable, and **Value** indicates the value of the environment variable. Click **Submit**.

For example, set **Key** to **User** and **Value** 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 5 Click **Submit**.

----End

3.3.2 Creating an Application Component

3.3.2.1 Application Components

An application component implements a service feature of an application. It is in the form of code or software packages and can be deployed independently.

After creating an application on ServiceStage, you can add components to the application. Currently, microservice, web, and common application components are supported. A maximum of 1000 application components can be created for an application.

You can create a static component by setting the component type, framework, runtime system, and component source, and then deploy this component.

In the process of adding a component, you can configure the component using a template (**Template**) or customize the configuration (**Custom**).

- **Template** provides default configurations of the component type, runtime system, and framework to help you quickly create components.

- **Custom** allows you to select the desired component type, runtime system, and proper framework/service mesh.

Existing Templates

Table 3-2 Existing templates

Type	Runtime System	Framework
ServiceComb MicroService	Java8	Java chassis
SpringCloud MicroService	Java8	Spring Cloud
Web(Tomcat) WebApp	Tomcat8	Web

Microservice Components

Supported Runtime System	Supported Framework/ Service Mesh	Supported Software Package
Java8	Java chassis	Source code repository, template, and JAR package
Tomcat8		Source code repository, template, and WAR package
Docker		This parameter does not need to be set.
Java8	Mesher	Source code repository and JAR package
Tomcat8		Source code repository and WAR package
Node.js8		Source code repository and ZIP package
Php7		Source code repository and ZIP package
Docker		This parameter does not need to be set.
Python3		Source code repository and ZIP package
Docker	Go Chassis	This parameter does not need to be set.

Supported Runtime System	Supported Framework/ Service Mesh	Supported Software Package
Java8	Spring Cloud	Source code repository and JAR package
Tomcat8		Source code repository and WAR package
Docker		This parameter does not need to be set.
Java8	Dubbo	Source code repository, template, and JAR package
Tomcat8		Source code repository, template, and WAR package
Docker		This parameter does not need to be set.

Web Application Components

Supported Runtime System	Supported Software Package
Java8	Source code repository, template, and JAR package
Nodejs8	Source code repository, template, and ZIP package
Php7	Source code repository, template, and ZIP package
Tomcat8	Source code repository, template, and WAR package
Docker	This parameter does not need to be set.
Python3	Source code repository and ZIP package

Common Components

Supported Runtime System	Supported Software Package
Java8	Source code repository, template, and JAR package
Tomcat8	Source code repository, template, and WAR package

Supported Runtime System	Supported Software Package
Node.js8	Source code repository, template, and ZIP package
Php7	Source code repository, template, and ZIP package
Docker	This parameter does not need to be set.
Python3	Source code repository and ZIP package

Component Source

Component Source	Description
Source code repository	Create authorization by referring to **Authorizing a Repository** and set the code source.
JAR package	Supports the following uploading modes: 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	Supports the following uploading modes: 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	Supports the following uploading modes: 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	If you use a private image to create your containerized application, upload the private image to the image repository. Choose **Software Center** > **Image Repository** and upload the image to the image repository by referring to **Managing Images**.
Template	Create authorization by referring to **Authorizing a Repository** and set the organization and repository names. ServiceStage provides component templates. You can select a template to quickly create an application and generate a development project in the configured code repository. For details, see **Template Framework**.

Template Framework

Runtime System	Framework Provided by the Template	Framework Description
Java8	CSE-Java (SpringMVC)	Based on the ServiceComb microservice development framework, supports SpringMVC annotations and uses the SpringMVC style to develop microservices.
	CSE-Java (JAX-RS)	Based on the ServiceComb microservice development framework, supports JAX-RS annotations and uses the JAX-RS mode to develop microservices.
	CSE-Java (POJO)	Based on the ServiceComb microservice development framework, supports APIs and API implementation, and uses transparent RPC to develop microservices.
Tomcat8	SpringBoot-Webapp-Tomcat	Web applications, running on an independent web server.
	SpringBoot-WebService-Tomcat	Web services, running on an independent web server.

Runtime System	Framework Provided by the Template	Framework Description
Nodejs8	Express	A Node.js web framework that supports high compatibility, and fast and simple deployment.
	Koa	Next-generation web development framework based on the Node.js platform.
Php7	Laravel	A PHP development framework for web developers.
	Slim	A lightweight micro-PHP framework.

3.3.2.2 Quickly Creating a Component

ServiceStage provides three default templates. For details, see **Existing Templates**.

A template provides default configurations of the component type, language/runtime system, and framework/service mesh to help you quickly create a component.

Prerequisites

1. An application has been created because components can only be added to applications. For details, see **Creating an Application**.
2. If you create a microservice component based on a source code repository or template, create repository authorization first. For details, see **Authorizing a Repository**.
3. If you create a microservice component based on a software package, upload the software package to the OBS bucket.

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

Procedure

Step 1 Log in to ServiceStage and choose **Application Management** > **Application List**.

Step 2 Select the created application and click **Create Component** in the **Operation** column.

Step 3 Select **Template** for **Configuration Method**, select a template, and click **Next**.

Step 4 Configure component information according to the following table. Parameters marked with an asterisk (*) are mandatory.

Table 3-3 Basic component information

Parameter	Description
*Name	Component name.
*Software Package	• Select **Source code repository**. – Create authorization by referring to **Authorizing a Repository** and set the code source. – Set **Build** parameters to build the application component. Set **Command** and **Organization**, and select a cluster based on service requirements. You can also specify **Node Label** to deliver the build job to a fixed node based on the node label. For details about how to add a node label, see **Adding a Node Label**. **NOTICE** If **Custom command** is selected for **Command**: Exercise caution when inputting sensitive information in the **echo**, **cat**, or **debug** command, or encrypt sensitive information to avoid information leakage. • Select **JAR package** or **WAR package**. **NOTE** – Select **JAR package** if **Java8** is selected for **Select Runtime System**. – Select **WAR package** if **Tomcat8** is selected for **Select Runtime System**. 1. Select **Upload Method**. Upload the software package to the OBS bucket. For details, see **Uploading a File**. 2. (Optional) Set **Build** parameters to build the application component. Set **Organization** and select a cluster based on service requirements. You can also specify **Node Label** to deliver the build job to a fixed node based on the node label. For details about how to add a node label, see **Adding a Node Label**. • Set the following parameters if **Template** is selected. 1. Select the template framework. ServiceStage provides template frameworks. You can select one to quickly create an application component. 2. Set **Code Archive**. See **Authorizing a Repository** to create authorization and set **Username/Organization** and **Repository**.

Step 5 Complete component creation.

- Click **Create Now** to create a static component.

- Click **Create and Deploy**. The deployment page is displayed. For details, see **Deploying a Component**.

After the component is created, you can view the component status in the **Component List** on the **Overview** page.

----End

3.3.2.3 Creating a Microservice Component

ServiceStage provides a microservice framework that enables you to develop and deploy applications on the cloud. It provides code framework generation, service registry and discovery, load balancing, and service reliability including fault tolerance, circuit breaker, rate limiting, and service degradation. This section describes how to create a static microservice application component using ServiceStage. For details about how to deploy a component, see **Deploying a Component**.

Prerequisites

1. An application has been created because components can only be added to applications. For details, see **Creating an Application**.
2. If you create a microservice component based on a source code repository or template, create repository authorization first. For details, see **Authorizing a Repository**.
3. If you create a microservice component based on a software package, upload the software package to the OBS bucket.

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

Procedure

Step 1 Log in to ServiceStage and choose **Application Management** > **Application List**.

Step 2 Select the created application and click **Create Component** in the **Operation** column.

Step 3 Select **Custom** for **Configuration Method** and select **Microservice** as the component type, and click **Next**.

Step 4 Select a runtime system and click **Next**.

Different frameworks support different runtime systems. For details, see **Microservice Components**.

Step 5 Selecting a framework/service mesh

For details about the framework/service mesh, see **Microservice Components**.

Step 6 Select whether you want to save the preceding configurations as a template for future use.

- If you select this function, enter a template name. Then, go to **Step 7**.
- If you do not select this function, go to **Step 7**.

Step 7 Check whether **Docker** is selected in **Step 4**.

- If yes, click **Next** and go to **Step 8**.
- If no, click **Next** and go to **Step 9**.

Step 8 Create a Docker component.

1. Enter a component name.
2. Create a component.
   - Click **Create Now** to create a static component.
   - Click **Create and Deploy**. The deployment page is displayed. For details, see **Deploying a Component**.
3. No further action is required.

   After the component is created, you can view the component status in the **Component List** on the **Overview** page.

Step 9 Configure component information according to the following table. Parameters marked with an asterisk (*) are mandatory.

Table 3-4 Basic component information

Parameter	Description
*Name	Component name.
*Software Package	• If you select **Source code repository**, create authorization by referring to **Authorizing a Repository** and set the code source. • If you select **JAR package**/**WAR package**/**ZIP package**, set **Upload Method**. Upload the software package to the OBS bucket. For details, see **Uploading a File**. **NOTE** – Select **JAR package** if **Java8** is selected for **Select Runtime System**. – Select **WAR package** if **Tomcat8** is selected for **Select Runtime System**. – Select **ZIP package** if **Nodejs8**, **Php7**, or **Python3** is selected for **Select Runtime System**. • Set the following parameters if **Template** is selected. 1. ServiceStage provides template frameworks. You can select one to quickly create an application component. 2. Set **Code Archive**. See **Authorizing a Repository** to create authorization and set **Username/Organization** and **Repository**. **NOTE** This parameter is invalid if you select **Mesher** or **Spring Cloud** for **Select Framework/Service Mesh** in **Step 5**.

Parameter	Description
*Python Framework	This parameter is mandatory if you select **Python3** for **Select Runtime System** in **Step 4**. Set **Module Name** and **Variable Name** for all frameworks except **Python3-Django**. • If the entry point file of the Python project is **server.py**, **Module Name** is **server**. • If the application function of the **server.py** entry point file of the Python project is **app=get_wsgi_application()**, **Variable Name** is **app**.
Build	• If **Source code repository** or **Template** is selected for **Software Package**, set **Build** parameters to build the application component. Set **Command** and **Organization**, and select a cluster based on service requirements. You can also specify **Node Label** to deliver the build job to a fixed node based on the node label. For details about how to add a node label, see **Adding a Node Label**. **NOTICE** If **Custom command** is selected for **Command**: Exercise caution when inputting sensitive information in the **echo**, **cat**, or **debug** command, or encrypt sensitive information to avoid information leakage. • (Optional) If **JAR package**, **WAR package**, or **ZIP package** is selected for **Software Package**, set **Build** parameters to build the application component. Set **Organization** and **CPU Architecture**, and select a cluster based on service requirements. You can also specify **Node Label** to deliver the build job to a fixed node based on the node label. For details about how to add a node label, see **Adding a Node Label**.

Step 10 Create a component.

- Click **Create Now** to create a static component.
- Click **Create and Deploy**. The deployment page is displayed. For details, see **Deploying a Component**.

After the component is created, you can view the component status in the **Component List** on the **Overview** page.

----End

3.3.2.4 Creating a Web Component

This section describes how to create a static web application component using ServiceStage. For details, see **Deploying a Component**.

Prerequisites

1. An application has been created because components can only be added to applications. For details, see **Creating an Application**.
2. If you create a microservice component based on a source code repository or template, create repository authorization first. For details, see **Authorizing a Repository**.
3. If you create a microservice component based on a software package, upload the software package to the OBS bucket.

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

Procedure

Step 1 Log in to ServiceStage and choose **Application Management** > **Application List**.

Step 2 Select the created application and click **Create Component** in the **Operation** column.

Step 3 Select **Custom** for **Configuration Method** and select **Web** as the component type, and click **Next**.

Step 4 Select a runtime system and click **Next**.

Different frameworks support different runtime systems. For details, see **Microservice Components**.

Step 5 Select whether you want to save the preceding configurations as a template for future use.

- If you select this function, enter a template name. Then, go to **Step 6**.
- If you do not select this function, go to **Step 6**.

Step 6 Check whether **Docker** is selected in **Step 4**.

- If yes, click **Next** and go to **Step 7**.
- If no, click **Next** and go to **Step 8**.

Step 7 Create a Docker component.

1. Enter a component name.
2. Create a component.
   - Click **Create Now** to create a static component.
   - Click **Create and Deploy**. The deployment page is displayed. For details, see **Deploying a Component**.
3. No further action is required.

   After the component is created, you can view the component status in the **Component List** on the **Overview** page.

Step 8 Configure component information according to the following table. Parameters marked with an asterisk (*) are mandatory.

Table 3-5 Basic component information

Parameter	Description
*Name	Component name.
*Software Package	• If you select **Source code repository**, create authorization by referring to **Authorizing a Repository** and set the code source. • If you select **JAR package**/**WAR package**/**ZIP package**, set **Upload Method**. Upload the software package to the OBS bucket. For details, see **Uploading a File**. **NOTE** – Select **JAR package** if **Java8** is selected for **Select Runtime System**. – Select **WAR package** if **Tomcat8** is selected for **Select Runtime System**. – Select **ZIP package** if **Nodejs8**, **Php7**, or **Python3** is selected for **Select Runtime System**. • Set the following parameters if **Template** is selected. 1. ServiceStage provides template frameworks. You can select one to quickly create an application component. 2. Set **Code Archive**. See **Authorizing a Repository** to create authorization and set **Username/Organization** and **Repository**. **NOTE** This parameter is invalid if you select **Python3** for **Select Runtime System** in **Step 4**.
*Python Framework	This parameter is mandatory if you select **Python3** for **Select Runtime System** in **Step 4**. Set **Module Name** and **Variable Name** for all frameworks except **Python3-Django**. • If the entry point file of the Python project is **server.py**, **Module Name** is **server**. • If the application function of the **server.py** entry point file of the Python project is **app=get_wsgi_application()**, **Variable Name** is **app**.

Parameter	Description
Build	• If **Source code repository** is selected for **Software Package**, set **Build** parameters to build the application component. Set **Command** and **Organization**, and select a cluster based on service requirements. You can also specify **Node Label** to deliver the build job to a fixed node based on the node label. For details about how to add a node label, see **Adding a Node Label**. **NOTICE** If **Custom command** is selected for **Command**: Exercise caution when inputting sensitive information in the **echo**, **cat**, or **debug** command, or encrypt sensitive information to avoid information leakage. • (Optional) If **JAR package**, **WAR package**, or **ZIP package** is selected for **Software Package**, set **Build** parameters to build the application component. Set **Organization** and **CPU Architecture**, and select a cluster based on service requirements. You can also specify **Node Label** to deliver the build job to a fixed node based on the node label. For details about how to add a node label, see **Adding a Node Label**.

Step 9 Create a component.

- Click **Create Now** to create a static component.
- Click **Create and Deploy**. The deployment page is displayed. For details, see **Deploying a Component**.

After the component is created, you can view the component status in the **Component List** on the **Overview** page.

----End

3.3.2.5 Creating a Common Component

This section describes how to create a static common application component using ServiceStage. For details about how to deploy a component, see **Deploying a Component**.

Prerequisites

1. An application has been created because components can only be added to applications. For details, see **Creating an Application**.
2. If you create a microservice component based on a source code repository or template, create repository authorization first. For details, see **Authorizing a Repository**.
3. If you create a microservice component based on a software package, upload the software package to the OBS bucket.

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

Procedure

Step 1 Log in to ServiceStage and choose **Application Management** > **Application List**.

Step 2 Select the created application and click **Create Component** in the **Operation** column.

Step 3 Select **Custom** for **Configuration Method** and select **Common** as the component type, and click **Next**.

Step 4 Select a runtime system and click **Next**.

Different frameworks support different runtime systems. For details, see **Microservice Components**.

Step 5 Select whether you want to save the preceding configurations as a template for future use.

- If you select this function, enter a template name. Then, go to **Step 6**.
- If you do not select this function, go to **Step 6**.

Step 6 Check whether **Docker** is selected in **Step 4**.

- If yes, click **Next** and go to **Step 7**.
- If no, click **Next** and go to **Step 8**.

Step 7 Create a Docker component.

1. Enter a component name.
2. Create a component.
   - Click **Create Now** to create a static component.
   - Click **Create and Deploy**. The deployment page is displayed. For details, see **Deploying a Component**.
3. No further action is required.

   After the component is created, you can view the component status in the **Component List** on the **Overview** page.

Step 8 Configure component information according to the following table. Parameters marked with an asterisk (*) are mandatory.

Parameter	Description
*Name	Component name.

Parameter	Description
*Software Package	• If you select **Source code repository**, create authorization by referring to **Authorizing a Repository** and set the code source. • If you select **JAR package**/**WAR package**/**ZIP package**, set **Upload Method**. Upload the software package to the OBS bucket. For details, see **Uploading a File**. **NOTE** – Select **JAR package** if **Java8** is selected for **Select Runtime System**. – Select **WAR package** if **Tomcat8** is selected for **Select Runtime System**. – Select **ZIP package** if **Nodejs8**, **Php7**, or **Python3** is selected for **Select Runtime System**. • Set the following parameters if **Template** is selected. 1. ServiceStage provides template frameworks. You can select one to quickly create an application component. 2. Set **Code Archive**. See **Authorizing a Repository** to create authorization and set **Username/Organization** and **Repository**. **NOTE** This parameter is invalid if you select **Python3** for **Select Runtime System** in **Step 4**.
*Python Framework	This parameter is mandatory if you select **Python3** for **Select Runtime System** in **Step 4**. Set **Module Name** and **Variable Name** for all frameworks except **Python3-Django**. • If the entry point file of the Python project is **server.py**, **Module Name** is **server**. • If the application function of the **server.py** entry point file of the Python project is **app=get_wsgi_application()**, **Variable Name** is **app**.

Parameter	Description
Build	• If **Source code repository** is selected for **Software Package**, set **Build** parameters to build the application component. Set **Command** and **Organization**, and select a cluster based on service requirements. You can also specify **Node Label** to deliver the build job to a fixed node based on the node label. For details about how to add a node label, see **Adding a Node Label**. **NOTICE** If **Custom command** is selected for **Command**: Exercise caution when inputting sensitive information in the **echo**, **cat**, or **debug** command, or encrypt sensitive information to avoid information leakage. • (Optional) If **JAR package**, **WAR package**, or **ZIP package** is selected for **Software Package**, set **Build** parameters to build the application component. Set **Organization** and **CPU Architecture**, and select a cluster based on service requirements. You can also specify **Node Label** to deliver the build job to a fixed node based on the node label. For details about how to add a node label, see **Adding a Node Label**.

Step 9 Create a component.

- Click **Create Now** to create a static component.
- Click **Create and Deploy**. The deployment page is displayed. For details, see **Deploying a Component**.

After the component is created, you can view the component status in the **Component List** on the **Overview** page.

----End

3.3.3 Deploying an Application Component

3.3.3.1 Deployment Mode

Deploying a Component Using CCE

Cloud Container Engine (CCE) provides highly scalable, high-performance, enterprise-class Kubernetes clusters and supports Docker containers. With CCE, you can easily deploy, manage, and scale containerized applications on the cloud platform.

If the build function is not enabled for the created component, the component cannot be deployed using a container.

Deploying a Component Using VM

The created component can be deployed using a VM only when **Java8**, **Tomcat8**, or **Nodejs8** is selected for **Select Runtime System**.

3.3.3.2 Deploying a Component

This section describes how to deploy static components in the corresponding environment.

When creating an application component, you can also select **Create and Deploy**. The deployment procedure is the same as that described in this section.

Prerequisites

1. An application component has been created or is being created, and has been configured. For details, see **Creating an Application Component**.
2. The environment has been created. For details, see **Environment Management**.
3. If you deploy components based on software packages or image packages, you need to upload the software packages or image packages.
   - Upload the software package to the OBS bucket. For details, see **Uploading a File**.
   - Upload the image package to the image repository. For details, see **Uploading an Image**.

Procedure

Step 1 Log in to ServiceStage and choose **Application Management** > **Application List**.

Step 2 Click an application. The **Overview** page is displayed.

Step 3 On the **Component List** tab, select a component and click **Deploy** in the **Operation** column.

Step 4 Set basic parameters. Parameters marked with an asterisk (*) are mandatory.

Parameter	Description
*Environment	Select an environment.
*Version	Component version number, for example, 1.0.0.
Description	Description of the component.
*Deployment System	Supports Cloud Container Engine and VM. For details, see **Deployment Mode**.
*Resource Type	This parameter takes effect only when **VM** is selected for **Deployment System**. You can select **Elastic cloud server**.
*Basic Resource	The basic resources contained in the selected environment are automatically loaded. Select the resources as required.

Parameter	Description
*Instances	Number of instances in an application component. An application component can have one or more instances. You can specify the number of instances as required. Configuring multiple instances for an application component ensures high reliability of the application component. For such a component, if an instance is faulty, the component can still run properly. **NOTE** • This parameter is invalid when **VM** is selected for **Deployment System**. • The number of component instances is determined by the number of **Basic Resources**. • When the quota for the number of microservice instances specified by the engine specifications is about to be used up, the engine allows component deployment operations that exceed the remaining quota to succeed at the same time to ensure availability. Expand the capacity of the engine as soon as possible to prevent component deployment failures.
*Resource Quota	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** as required. **NOTE** • This parameter is not displayed when the component type is **Common** and the runtime system is **Docker**. • This parameter is invalid when **VM** is selected for **Deployment System**.
Component Status	Sets the component status as required. **NOTE** This parameter is available when the component type is **Common**, the runtime system is **Docker**, and **Cloud Container Engine** is selected for **Deployment System**.

Step 5 Click **Next** to configure the component.

- When the component type is **Common** and the runtime system is **Docker**, perform the following operations:
  a. Select an image. Multiple containers are supported. You can click **Add Container** to add an image.
  b. Select an image version.
  c. Enter a container name.
  d. (Optional) Set **Resource Quota**. 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** as required.

e. (Optional) Set **Advanced Settings**.

- Choose **Advanced Settings** > **Component Configuration** and set environment variables. For details, see **Configuring Environment Variables of a Component**.
- Choose **Advanced Settings** > **Deployment Configuration**.
  - Set **Startup Command** and **Lifecycle**. For details, see **Configuring the Lifecycle of an Application**.
  - Set **Data Storage**. For details, see **Configuring Data Storage**.
- Choose **Advanced Settings** > **O&M Monitoring**.
  - Set **Log Collection**. For details, see **Configuring a Log Policy of an Application**.
  - Set **Health Check**. For details, see **Configuring Health Check**.

f. (Optional) Enable **Public Network Access**.

i. Set **Public Network Load Balancer**.

Select an existing ELB resource in the 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 Load Balancer**.

ii. (Optional) Set **HTTPS**.

If **HTTPS** is enabled, click **Use existing** to select an existing certificate.

If no certificate exists, click **Create new** to create a server certificate. For details about how to create a certificate, see **Creating a Certificate**.

iii. Set **Domain Name**.

Enter a customize domain name. For details, see **Configuring Domain Name Mappings**.

iv. Set **Listening Port**.

Set the listening port of the application process.

g. (Optional) Set **Database**.

Select **Distributed session**. For details, see **Configuring Distributed Sessions**.

h. (Optional) Set **Local Time**.

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

i. (Optional) Set **Scheduling Policies**. For details, see **Configuring a Scheduling Policy of an Application Component Instance**.

j. (Optional) Set **Upgrade Policies**. For details, see **Configuring an Upgrade Policy of an Application Component Instance**.

- For other types of components for which **Cloud Container Engine** is selected for **Deployment System** in **4**, perform the following operations:

a. Set **Image**.

- If the application source is a software package, source code, or template, the configured static component information will be loaded.
- If **Runtime System** is set to **Docker**, select an image package from the SWR image repository.

b. (Optional) Enable **Public Network Access**.

i. Set **Public Network Load Balancer**.

Select an existing ELB resource in the 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 Load Balancer**.

ii. (Optional) Set **HTTPS**.

If **HTTPS** is enabled, click **Use existing** to select an existing certificate.

If no certificate exists, click **Create new** to create a server certificate. For details about how to create a certificate, see **Creating a Certificate**.

iii. Set **Domain Name**.

Enter a customize domain name. For details, see **Configuring Domain Name Mappings**.

iv. (Optional) Set **Listening Port**.

Listening port of an application process. If **Tomcat8** is selected as the **Runtime System**, this port is set to **8080** by default. You can customize this port.

c. Set **Cloud Service Engine**.

This parameter is mandatory for microservice components.

By default, the microservice engine added in the environment is selected. For details about how to create a microservice engine, see **Creating an Exclusive Microservice Engine**.

d. (Optional) Set **JVM**.

This parameter is mandatory when **Runtime System** is set to **Java8** or **Tomcat8**.

Enter the JVM parameter, for example, **-Xms256m -Xmx1024m**. Multiple parameters are separated by spaces. If the parameter is left blank, the default value is used.

e. (Optional) Configure Tomcat parameters.

This parameter is mandatory when **Runtime System** is set to **Tomcat8**.

i. Select **Parameter settings**. The **Tomcat** dialog box is displayed.

ii. Click **Use Sample Code** and edit the template file based on service requirements.

iii. Click **OK**.

f. (Optional) Set **Database**.

Select **Distributed session**. For details, see **Configuring Distributed Sessions**.

g. (Optional) Set **Local Time**.

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

h. (Optional) Set **Advanced Settings**.

- Choose **Advanced Settings** > **Component Configuration** and set environment variables. For details, see **Configuring Environment Variables of a Component**.
- Choose **Advanced Settings** > **Deployment Configuration**.
  - Set **Startup Command** and **Lifecycle**. For details, see **Configuring the Lifecycle of an Application**.
  - Set **Data Storage**. For details, see **Configuring Data Storage**.
  - Set **Scheduling Policy**. For details, see **Configuring a Scheduling Policy of an Application Component Instance**.
  - Set **Upgrade Policy**. For details, see **Configuring an Upgrade Policy of an Application Component Instance**.
- Choose **Advanced Settings** > **O&M Monitoring**.
  - Set **Log Collection**. For details, see **Configuring a Log Policy of an Application**.
  - Set **Health Check**. For details, see **Configuring Health Check**.
  - Set **O&M Policy**. For details, see **Configuring Custom Monitoring of an Application Component**.

- For other types of components for which **VM** is selected for **Deployment System** in **4**, perform the following operations:

a. (Optional) Enable **Public Network Access**.

i. Set **Public Network Load Balancer**.

Select the created load balancer.

If no load balancer exists, create one. For details, see **Creating a Load Balancer**.

ii. (Optional) Set **HTTPS**.

If **HTTPS** is enabled, click **Use existing** to select an existing certificate.

If no certificate exists, click **Create new** to create a server certificate. For details about how to create a certificate, see **Creating a Certificate**.

iii. Set **Domain Name**.

Enter a customize domain name. For details, see **Configuring Domain Name Mappings**.

b. (Optional) Set **Cloud Service Engine**.

This parameter is mandatory for microservice components.

By default, the microservice engine added in the environment is selected. For details about how to create a microservice engine, see **Creating an Exclusive Microservice Engine**.

c. (Optional) Set **Environment Variables**.

For details, see **Configuring Environment Variables of a Component**.

d. (Optional) Set **JVM**.

This parameter is mandatory when **Runtime System** is set to **Java8** or **Tomcat8**.

Enter the JVM parameter, for example, **-Xms256m -Xmx1024m**. Multiple parameters are separated by spaces. If the parameter is left blank, the default value is used.

e. (Optional) Configure Tomcat parameters.

This parameter is mandatory when **Runtime System** is set to **Tomcat8**.

i. Select **Parameter settings**. The **Tomcat** dialog box is displayed.

ii. Click **Use Sample Code** and edit the template file based on service requirements.

iii. Click **OK**.

Step 6 Click **Next**, confirm the specifications, and click **Deploy**.

After the component is deployed, you can view the component status in the **Environment View** on the **Overview** page.

----End

3.3.4 Managing Application Components

After a component is created or deployed, you can perform the following management operations:

- **Viewing Application Components**: View the list of components created under the application.
- **Deploying a Component**: Deploy the created static components.
- **Updating Component Source**: Update the software package, version, and environment configuration of the components. Components whose runtime system is **Docker** do not support this operation.
- **Deleting Components**: Delete the created components.
- **Creating a Pipeline for a Component**: 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.
- **Viewing Application Component Building**: View the status of the application component build job.
- **Maintaining Component Instances**: Maintain the deployed application component instances.
- **Managing Component Instance Access Mode**: Set the access mode of the component instances.

Viewing Application Components

Step 1 Log in to ServiceStage and choose **Application Management** > **Application List**.

Step 2 Click an application. The **Overview** page is displayed.

Step 3 Click the **Component List** tab to view the list of components created for the application.

----End

Deploying a Component

For details about how to deploy a component, see **Deploying a Component**.

Updating Component Source

After a component is created, you can update the software package, version, and environment configuration of the component.

Components whose runtime system is **Docker** do not support this operation.

Step 1 Log in to ServiceStage and choose **Application Management** > **Application List**.

Step 2 Click an application. The **Overview** page is displayed.

Step 3 Click the **Component List** tab.

- To update the source of a single component, select the component and click **Update Source** in the **Operation** column.
- To update the component sources in batches, select multiple components and click **Update Component Source**.

Step 4 Set **Software Package**.

- **Source code repository**: Create authorization by referring to **Authorizing a Repository** and set the code source.
- **Software Package**:
  - Click **Replace Software Package** and select the corresponding software package from OBS. Upload the software package to the OBS bucket in advance. For details, see **Uploading a File**.

Step 5 Set the target version and choose the environment to be upgraded.

Step 6 Click **Confirm**.

----End

Deleting Components

Step 1 Log in to ServiceStage and choose **Application Management** > **Application List**.

Step 2 Click an application. The **Overview** page is displayed.

Step 3 Click the **Component List** tab.

- Delete a single component.

  Select the component to be deleted and click **Delete** in the **Operation** column. In the displayed dialog box, click **OK**.

- Delete components in batches.

  Select the components to be deleted and click **Delete Component**. In the displayed dialog box, click **OK**.

----End

Creating a Pipeline for a Component

Step 1 Log in to ServiceStage and choose **Application Management** > **Application List**.

Step 2 Click an application. The **Overview** page is displayed.

Step 3 On the **Component List** tab, click a component to go to the **Overview** page.

Step 4 Choose **Pipeline** > **Create Pipeline** to create a pipeline. For details, see **Managing Pipelines**.

NOTE

- Pipelines cannot be created for component instances deployed on VMs.
- Pipelines cannot be created for components whose runtime system is **Docker**.

----End

Viewing Application Component Building

Step 1 Log in to ServiceStage and choose **Application Management** > **Application List**.

Step 2 Click an application. The **Overview** page is displayed.

Step 3 On the **Component List** tab, click a component to go to the **Overview** page.

Step 4 Click the **Build Job** tab to view the status of the application component build job. For details, see **Creating a Source Code Job**.

----End

Maintaining Component Instances

Step 1 Log in to ServiceStage and choose **Application Management** > **Application List**.

Step 2 Click an application. The **Overview** page is displayed.

Step 3 On the **Environment View** tab, select an environment.

- You can view the deployment of the application component in each environment.
- (Optional) Select an application component version whose type is **Microservice** and click **Console** to go to the microservice console for service governance. For details, see **Microservice Governance**.
- Select an application component version and click **Perform O&M**. The **Overview** page is displayed, where you can view the component instance details.
- Select an application component version and click **Operation**. You can perform O&M operations such as component upgrade, scaling, event viewing, start/stop, restart, rollback, and deletion. For details, see **Application O&M**.

- Select **All** or the corresponding application component, and click **Upgrade Component** to change the version number, software package, or image package of the component.

----End

Managing Component Instance Access Mode

Step 1 Log in to ServiceStage and choose **Application Management** > **Application List**.

Step 2 Click an application. The **Overview** page is displayed.

Step 3 On the **Component List** tab, click a component to go to the **Overview** page.

You can view the component version on the card of the corresponding environment.

Step 4 Select a component whose status is **Running** and click **Set**. On the **Access Mode** page that is displayed, click **Add Service**.

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

Parameter	Description
*Service Name	Sets the name of the service to be accessed.
Access Mode	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**.
Port Mapping	Sets **Protocol**, **Container Port**, and **Access Port** for accessing the service.

Step 6 Click **OK**.

----End

3.3.5 Performing Advanced Settings of an Application

3.3.5.1 Configuring Environment Variables of a Component

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

This section describes how to configure environment variables during deployment using CCE and VM.

CCE

If **Cloud Container Engine** is selected for **Deployment System** on the **Configure Basic Settings** page during component deployment, add environment variables by referring to the following steps.

Step 1 On the **Configure Component** page, choose **Advanced Settings** > **Component Configuration**.

Step 2 Add environment variables by referring to **Table 3-6**.

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

Table 3-6 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.
Add 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. **Optional:** Select a secret from the **Variable/Variable Reference** drop-down list.
Add from ConfigMap	1. Create a configuration item. For details, see **Creating a Configuration Item**. 2. Click **Add Environment Variable** and select **Add from ConfigMap**. 3. Enter **Name**. 4. Select a configuration item from the **Variable/Variable Reference** drop-down list.
Import	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

VM

If **VM** is selected for **Deployment System** on the **Configure Basic Settings** page during component deployment, add environment variables by referring to the following steps.

Step 1 On the **Configure Component** page, click **Add Environment Variable**.

Step 2 Set **Key** and **Value**, and click **OK**.

----End

3.3.5.2 Configuring the Lifecycle of an Application

If **Cloud Container Engine** or is selected for **Deployment System** on the **Configure Basic Settings** page during component deployment, ServiceStage provides callback functions that can be invoked in specific phases of the application lifecycle. For example, if an operation needs to be performed on an application component before the component is stopped, you can register the corresponding 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 When deploying an application component, choose **Advanced Settings** > **Component Configuration** on the **Configure Component** page.

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 3-7** describes the rules.

Table 3-7 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 3-8** describes the parameters. Select one of the parameters.

Table 3-8 Container lifecycle parameters

Parameter	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

3.3.5.3 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.

If **Cloud Container Engine** is selected for **Deployment System** on the **Configure Basic Settings** page during component deployment, you can set data storage.

Scenario

Table 3-9 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.
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.

Storage Type	Scenario
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 When deploying an application component, choose **Advanced Settings** > **Component Configuration** on the **Configure Component** page.

Step 2 Choose **Data Storage** > **Cloud Storage** > **Add Cloud Storage** and set parameters by referring to **Table 3-10**.

Table 3-10 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.
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.

Parameter	Description
Add 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 **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 application component.

Step 1 When deploying an application component, choose **Advanced Settings** > **Component Configuration** on the **Configure Component** page.

Step 2 Choose **Data Storage** > **Local Disk** > **Add Local Disk** and set parameters by referring to **Table 3-11**.

Table 3-11 HostPath

Parameter	Description
Local Disk Type	Select **HostPath**.
Host Path	Enter the host path, for example, **/etc/hosts**.

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 **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 When deploying an application component, choose **Advanced Settings** > **Component Configuration** on the **Configure Component** page.

Step 2 Choose **Data Storage** > **Local Disk** > **Add Local Disk** and set parameters by referring to **Table 3-12**.

Table 3-12 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.

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 **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 When deploying an application component, choose **Advanced Settings** > **Component Configuration** on the **Configure Component** page.

Step 2 Choose **Data Storage** > **Local Disk** > **Add Local Disk** and set parameters by referring to **Table 3-13**.

Table 3-13 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**.

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 **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 When deploying an application component, choose **Advanced Settings** > **Component Configuration** on the **Configure Component** page.

Step 2 Choose **Data Storage** > **Local Disk** > **Add Local Disk** and set parameters by referring to **Table 3-14**.

Table 3-14 Secret

Parameter	Description
Local Disk Type	Select **Secret**.
Secret Item	Select the desired secret name. For details about how to create a secret, see **Creating a Secret**.

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 **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

3.3.5.4 Configuring Distributed Sessions

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 **Deploying a Component**, you can bind distributed sessions in **Database**.

Prerequisites

Before setting a distributed session, create a distributed session. For details, see **Creating a DCS Redis Instance**.

Procedure

Step 1 During component deployment, select **Distributed session** on the **Configure Component** page.

Step 2 Click **Add One**. On the **Edit Environment** page that is displayed, click **Add Optional Resource** to add created DCS resources to the environment.

----End

3.3.6 Building an Application Component

Before deploying ServiceStage, you need to build the software package into an image or build the source code into a software package. Therefore, after you set the application source when creating an application component, ServiceStage generates a build job for the application component.

NOTE

Components whose runtime system is **Docker** cannot be built.

Viewing Application Component Building

Step 1 Log in to ServiceStage, choose **Application Management** > **Application List**, and click an application to go to the **Overview** page.

Step 2 On the **Component List** tab, click a component to go to the **Overview** page.

Step 3 Click the **Build Job** tab to view the status of the application component build job.

----End

Maintaining a Build Job

Table 3-15 Maintenance

Operation	Operation Description
Edit	Click **Edit** to go to the build job configuration page. For details, see **Editing a Package Build Job** or **Editing a Source Code Job**.
Build Now	Click **Build Now** to start a build job.
Query details/build history	• Click **View Other Build Records** and view the build history. • Click **Logs** to view the build log. • Click **Code Check** to view the code check overview and details. Currently, the following code check plug-ins are supported: Checkstyle, FindBugs, and PMD. **NOTE** Only the Maven build project supports code check.

Editing a Package Build Job

Step 1 Log in to ServiceStage, choose **Application Management** > **Application List**, and click an application to go to the **Overview** page.

Step 2 On the **Component List** tab, click a component to go to the **Overview** page.

Step 3 Click the **Build Job** tab and click **Edit**. The build job configuration page is displayed.

Step 4 (Optional) Set **Description**.

Step 5 Set **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 6 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 7 Set **Archived Image Address**.

Step 8 Select **Cluster**.

Step 9 (Optional) Specify **Node Label** to deliver the build job to a fixed node based on the node label.

Step 10 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, choose **Application Management** > **Application List**, and click an application to go to the **Overview** page.

Step 2 On the **Component List** tab, click a component to go to the **Overview** page.

Step 3 Click the **Build Job** tab and click **Edit**. The build job configuration page is displayed.

1. (Optional) Set **Description**.
2. Select **Cluster**.
3. (Optional) Specify **Node Label** to deliver the build job to a fixed node based on the node label.
4. Click **Next**.

Step 4 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 5 Click **Build** to save the settings and start the build.

Click **Save** to save the settings (not to start the build).

----End

3.3.7 Pipelining an Application Component

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.

NOTE

- Pipelines cannot be created for instances deployed on VMs.
- Pipelines cannot be created for components whose runtime system is **Docker**.

Creating a Pipeline

Step 1 Log in to ServiceStage, choose **Application Management** > **Application List**, and click an application to go to the **Overview** page.

Step 2 On the **Component List** tab, click a component to go to the **Overview** page.

Step 3 Click the **Pipeline** tab and click **Create Pipeline**.

Step 4 Enter the basic pipeline information.

1. Enter a pipeline name.
2. (Optional) Enter **Description**.

Step 5 Set pipeline.

1. Add a build job.

   The build job of the component is automatically loaded.
2. Add a deploy job.

   Click **Add Environment**. The deployed components are automatically loaded.

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 6 Click **Create and Start** to start the pipeline.

Click **Create** to save the settings and do not execute the pipeline.

----End

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, choose **Application List**, and select an application and an application component to go to the application component details page. On the page that is displayed, click **Pipeline**.

Step 2 Select a pipeline and click **Clone**.

Step 3 ServiceStage automatically loads configurations of the clone pipeline. You can then modify the configurations as required.

Step 4 Click **Create and Start** to start the pipeline.

Click **Create** to save the settings and do not execute the pipeline.

----End

Related Operations

After the pipeline is started, you can build and upgrade applications in one-click mode. For details about maintenance after application components are upgraded, see **Application O&M**.

3.3.8 Application Configuration

3.3.8.1 Creating a Secret

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.

Prerequisites

- You have created a CCE cluster that requires a secret.

  For details about how to create a CCE cluster, see **Creating a CCE Cluster**.
- You have created a namespace for the secret. For details, see **Creating a Namespace**.

Creating a Secret

Step 1 Log in to ServiceStage and choose **Application Management** > **Application Configuration** > **Secret**

Step 2 Click **Create**.

Step 3 Create a secret by **Visualization** or **YAML**.

- Method 1: **Visualization**. On the displayed page, set the parameters listed in the following table. Parameters marked with an asterisk (*) are mandatory.

Table 3-16 Parameters for creating a secret

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. The default value is **default**.
Description	Description of the secret. Click **Create Namespace** to create a namespace.
*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.

Parameter	Description
*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 image repository address, 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 secret type, key, and value.
Secret Label	Labels are attached to 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. Set keys and values.

- Method 2: **YAML**.

  NOTE

  To create a secret by uploading a file, ensure that the resource description file has been created. ServiceStage supports resource description files in YAML format. For more information, 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**.
  c. Write or modify the secret file in **Orchestration content**.

Step 4 Click **Create**.

The new secret is displayed in the secret list.

----End

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 content in the secret file **secret.yaml** is as follows. 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 encoded using Base64.
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.

Managing Secrets

Operation	Description
Modifying a secret	1. Click **Modify** in the **Operation** column of the target secret. 2. Modify the secret information according to **Table 3-16**. 3. Click **Modify Secret**.
Deleting a secret	Click **Delete** in the **Operation** column of the secret to be deleted, and follow the system prompts to delete the secret.
Deleting secrets in batches	1. Select the secrets to be deleted. 2. Click **Delete** in the upper left of the page, and follow the system prompts to delete the secrets.
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.

NOTE

The secret list contains system secrets, which can only be viewed and cannot be modified or deleted.

3.3.8.2 Creating a Configuration Item

Configuration items 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.

Prerequisites

- You have created a CCE cluster that requires a configuration item.

  For details about how to create a CCE cluster, see **Creating a CCE Cluster**.
- You have created a namespace for the configuration item. For details, see **Creating a Namespace**.

Creating a Configuration Item

Step 1 Log in to ServiceStage and choose **Application Management** > **Application Configuration** > **ConfigMap**

Step 2 Click **Create**.

Step 3 Create a configuration item by **Visualization** or **YAML**.

- Method 1: **Visualization**. On the displayed page, set the parameters listed in the following table. Parameters marked with an asterisk (*) are mandatory.

Table 3-17 Parameters for creating a configuration item

Parameter	Description
*Configuration Name	Name of a configuration item, which must be unique in the same namespace.
*Cluster	Cluster where the configuration item will be used.
*Namespace	Namespace to which the configuration item belongs. If you do not specify this parameter, the value **default** is used by default.
Description	Description of the configuration item.
Configuration Data	Used in applications or used to store configuration data. **Key** is a file name, and **Value** is the content of the file. 1. Click **Add Label**. 2. Set keys and values.
Configuration Labels	Labels are attached to 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. Set keys and values.

- Method 2: **YAML**.

  NOTE

  To create a configuration item by uploading a file, ensure that a resource description file has been created. ServiceStage supports resource description 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**, select the created configuration item resource file, and then click **Open**.
  c. Write or modify the configuration item resource file in **Orchestration content**.

Step 4 Click **Create**.

The new configuration item is displayed in the configuration item list.

----End

Configuration Item Requirements

A configuration item resource file should be in YAML format, and the file size cannot exceed 2 MB.

The following shows an example of a configuration item resource file named **configmap.yaml**:

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

Managing Configuration Items

Operation	Description
Modifying a configuration item	1. Click **Modify** in the **Operation** column of the target configuration item. 2. Modify the configuration item information according to **Table 3-17**. 3. Click **Modify**.
Deleting a configuration item	Click **Delete** in the **Operation** column of the configuration item to be deleted, and follow the system prompts to delete this configuration item.
Deleting configuration items in batches	1. Select the configuration items to be deleted. 2. Click **Delete** in the upper left of the page, and follow the system prompts to delete the configuration items.

Operation	Description
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.

NOTE

The configuration item list contains system configuration items, which can only be viewed and cannot be modified or deleted.

3.4 Environment Management

An environment is a collection of compute, storage, and network resources used for deploying and running an application. ServiceStage combines basic resources (such as CCE clusters and ECSs) and optional resources (such as ELB instances and DCS instances) in the same VPC 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.

ServiceStage allows a single user to create a maximum of 1000 environments.

Creating an Environment

Step 1 Log in to ServiceStage, choose **Environment Management**, and click **Create Environment**.

Step 2 Set basic parameters. Parameters marked with an asterisk (*) are mandatory.

Parameter	Description
*Environment	Environment name.
Description	Environment description.
*VPC	Select the VPC to which the infrastructure belongs. For details about how to create a VPC, see **Creating a VPC**. **NOTE** After a VPC is selected, infrastructure resources in the VPC are loaded for selection. Resources that are not in the VPC cannot be selected.
*Basic Resource	Select at least one of the following infrastructures: CCE, ECS. You can select multiple infrastructures.
Optional Resource	You can select ELB, EIP, DCS, or CSE as required.

Step 3 Click **Create Now**.

After the environment is created, you can view the environment information on the **Environment Management** page.

----End

Modifying an Environment

Step 1 Log in to ServiceStage, choose **Environment Management**, and click **Edit** of the environment to be modified.

Step 2 Set basic parameters. Parameters marked with an asterisk (*) are mandatory.

Parameter	Description
*Environment	Enter a new environment name.
Description	Environment description.
*VPC	The VPC cannot be modified. You can only add resources in the selected VPC. Resources that are not in the VPC cannot be selected.
*Basic Resource	You can add or delete basic resources, including CCE and ECS.
Optional Resource	You can add or delete optional resources, including ELB, EIP, DCS, or CSE as required.

Step 3 Click **Save**.

After the environment is modified, you can view the environment information on the **Environment Management** page.

----End

Deleting an Environment

NOTE

- Before deleting an environment, ensure that no application component is deployed in the environment or the deployed application components have been deleted. For details, see **Managing Application Components**.
- Deleting an environment does not delete resources in the environment.

Step 1 Log in to ServiceStage, choose **Environment Management**, and click **Delete** on an existing environment.

Step 2 In the dialog box that is displayed, click **OK**.

----End

3.5 Application O&M

3.5.1 Maintaining Application Component Instances

Scaling Application Components

You can define auto-scaling policies as required, releasing you from the workload of repeatedly adjusting resources in response to service changes and heavy burden during peak hours and saving resource and labor costs. For details, see **Configuring a Scaling Policy of an Application Component Instance**.

Starting and Stopping an Application Component

After an application component is deployed, you can start or stop it as required.

Step 1 Log in to ServiceStage and choose **Application Management** > **Application List** to view all applications.

Step 2 Click an application. The **Overview** page is displayed.

Step 3 On the **Environment View** tab, select an environment to view the application components that have been deployed in the environment.

Step 4 Start or stop a component.

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

----End

Upgrading an Application Component

After an application component is deployed, you can re-deploy software packages and modify component configurations as required.

Step 1 Log in to ServiceStage and choose **Application Management** > **Application List** to view all applications.

Step 2 Click an application. The **Overview** page is displayed.

Step 3 On the **Environment View** tab, select an environment to view the application components that have been deployed in the environment.

Step 4 Click **Upgrade** in the **Operation** column to upgrade an application component.

Step 5 Update the configuration and version based on service requirements.

NOTICE

The component can be rolled back to the source version only if the component version has been changed.

Step 6 Configure advanced settings. For details, see **Performing Advanced Settings of an Application**.

Step 7 Click **Re-deployment**.

Step 8 Click **OK** and wait until the component upgrade is complete.

----End

Viewing Instance Details

After an application component is deployed, you can view the overview, instance list, and access mode on the application component details page.

Step 1 Log in to ServiceStage and choose **Application Management** > **Application List** to view all applications.

Step 2 Click an application. The **Overview** page is displayed.

Step 3 On the **Environment View** tab, select an environment to view the application components that have been deployed in the environment.

Step 4 Click an application component. The **Overview** page is displayed. You can view the instance details.

----End

Querying Application Logs

After an application component starts, you can query application logs to learn about its running status.

Step 1 Log in to ServiceStage and choose **Application Management** > **Application List**. to view all applications.

Step 2 Click an application. The **Overview** page is displayed.

Step 3 On the **Environment View** tab, select an environment to view the application components that have been deployed in the environment.

Step 4 Click an application component. The **Overview** page is displayed.

Step 5 Click **Logs** to view application component logs.

- Select an instance, log file name, and time granularity to view logs of the specified instance in a specified period.
- Enter a keyword in the text box to search for logs.

----End

Rolling Back an Application Component

After an application component is upgraded, you can roll it back to its target version.

Step 1 Log in to ServiceStage and choose **Application Management** > **Application List** to view all applications.

Step 2 Click an application. The **Overview** page is displayed.

Step 3 On the **Environment View** tab, select an environment to view the application components that have been deployed in the environment.

Step 4 Click **Roll Back** in the **Operation** column to roll back an application component.

Step 5 Locate the target version and click **Roll back to this version** in the **Operation** column.

----End

Deleting an Application Component

NOTICE

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

Step 1 Log in to ServiceStage and choose **Application Management** > **Application List** to view all applications.

Step 2 Click an application. The **Overview** page is displayed.

Step 3 On the **Environment View** tab, select an environment to view the application components that have been deployed in the environment.

Step 4 Click **Delete** in the **Operation** column delete an application component.

Click **OK** as prompted.

----End

3.5.2 Adding Labels for Application Component Instances

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 3-2 Label example

Procedure

Step 1 Log in to ServiceStage and choose **Application Management** > **Application List** to view all applications.

Step 2 Click an application. The **Overview** page is displayed.

Step 3 On the **Environment View** tab, select an environment to view the application components that have been deployed in the environment.

Step 4 Click an application component. The **Overview** page is displayed.

Step 5 Click **Manage Label**.

Step 6 Click **Add Label**, set **Key** and **Value**, and click **Save**.

NOTE

- The key name must be unique.
- Labels cannot be added to application component instances that are abnormal or deployed on VMs.

----End

3.5.3 Configuring Domain Name Mappings

For application components that have Internet access enabled, you must define a domain name on ServiceStage and configure the domain name mapping in the domain name provider.

Prerequisites

- You can change the domain name only when the application 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 application component.

Procedure

Step 1 Log in to ServiceStage and choose **Application Management** > **Application List** to view all applications.

Step 2 Click an application. The **Overview** page is displayed.

Step 3 On the **Environment View** tab, select an environment to view the application components that have been deployed in the environment.

Step 4 Click an application component. The **Overview** page is displayed.

Step 5 Set a domain name.

1. Choose **Access Mode** > **Set domain**, and enter the obtained domain name.
2. Enable **HTTPS**.

   If **HTTPS** is enabled, click **Use existing** to select an existing certificate.

   If no certificate exists, click **Create new** to create a server certificate. For details about how to create a certificate, see **Creating a Certificate**.

Step 6 Configure domain name mapping in the domain name provider.

----End

3.5.4 Configuring Alarm Thresholds for Resource Monitoring

When 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 in routine O&M 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 and choose **Application Management** > **Application List** to view all applications.

Step 2 Click an application. The **Overview** page is displayed.

Step 3 On the **Environment View** tab, select an environment to view the application components that have been deployed in the environment.

Step 4 Click an application component. The **Overview** page is displayed.

Step 5 Choose **Threshold Alarms** > **Set Threshold Rule** and set threshold rule parameters based on **Table 3-18**. The parameters marked with * are mandatory.

Table 3-18 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.
Statistical Mode	Method used to measure metrics.
Statistics Cycle	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.
Continuous Cycle	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 Notification	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.

Step 6 Click **OK**.

----End

Managing Threshold Alarms

After a threshold rule is created, you can manage threshold alarms by referring to **Table 3-19**.

Table 3-19 Related operations

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 one or more threshold rules, 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**. You can also click **Advanced Search** to set the search criteria and click **Search** to query.
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.
Check the data insufficiency event.	If no metric is reported within a specified period, the system sends a data insufficiency event. You can click the **Event** tab to view the event in the event list.

3.5.5 Configuring a Scaling Policy of an Application Component Instance

After scaling policies are set, 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.

- Auto scaling: Metric-based, scheduled, and periodic policies are supported. After configuration, instances can be automatically added or deleted based on resource changes or a specified schedule.
- Manual scaling: The number of instances is increased or decreased immediately after the configuration is complete.

Graceful Scaling-In

You can set a graceful scaling-in time window to save important data before an application 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.

Step 1 Log in to ServiceStage and choose **Application Management** > **Application List** to view all applications.

Step 2 Click an application. The **Overview** page is displayed.

Step 3 On the **Environment View** tab, select an environment to view the application components that have been deployed in the environment.

Step 4 Click an application component. The **Overview** page is displayed.

Step 5 Choose **Scaling**.

- Set **Graceful Time Window (s)**. Specifically, click  , enter a value, and click .
- Set **Maximum number of unavailable instances**. Specifically, click  , enter the maximum number (or select **percentage** and enter the maximum percentage), and click **Save**.

Click **View Component Details**. The instance is displayed in the **Upgrading/ Rolling back the component** state. When the status changes to **Running**, the scaling is complete.

----End

Manual Scaling

Step 1 Log in to ServiceStage and choose **Application Management** > **Application List** to view all applications.

Step 2 Click an application. The **Overview** page is displayed.

Step 3 On the **Environment View** tab, select an environment to view the application components that have been deployed in the environment.

Step 4 Click an application component. The **Overview** page is displayed.

Step 5 Choose **Scaling**. In the **Manual Scaling** area:

1. Click  and change the number of instances.
2. Click .

Click **View Component Details**. The instance is displayed in the **Upgrading/ Rolling back the component** state. When the status changes to **Running**, the scaling is complete.

----End

Auto Scaling

NOTE

- CCE clusters of versions later than 1.15 do not support auto scaling.
- VM deployment does not support auto scaling.

Step 1 Log in to ServiceStage and choose **Application Management** > **Application List** to view all applications.

Step 2 Click an application. The **Overview** page is displayed.

Step 3 On the **Environment View** tab, select an environment to view the application components that have been deployed in the environment.

Step 4 Click an application component. The **Overview** page is displayed. Choose **Scaling**.

Step 5 In the **Auto Scaling** area, click **Edit Scaling Rule**.

1. Set **Cooling Time**, **Maximum Instances**, and **Minimum Instances** based on service requirements.
2. Click **Save**.

Step 6 In the **Auto Scaling** area, click **Add Scaling Policy**.

Currently, ServiceStage supports the following types of auto scaling policies:

- **Alarm Policy**: scaling based on the CPU or memory settings. After an application component is deployed, instances in this application can be automatically scaled in or out when the number of CPU cores or memory amount exceeds or is less than a specified value.

Table 3-20 Parameters for adding a metric-based policy

Parameter	Description
Policy Name	Name of the scaling policy.
Policy Type	Choose **Alarm Policy**.

Parameter	Description
Metric	Select a metric. Metrics reflect the resource performance or status. – CPU usage of the measured object. This metric indicates the percentage of the CPU cores actually used by the measured object to the total CPU cores that the measured object has applied for. – Physical memory usage. This metric indicates the percentage of the physical memory size used by the measured object to the physical memory size that the measured object has applied for. – Disk read rate, which indicates the data volume read from the disk per second. – Physical memory size that the measured object has applied for. – Data receiving rate, which indicates the data volume received by the measured object per second. – Disk write rate, which indicates the data volume written into the disk per second. – Size of the physical memory used by the measured object. – Total number of CPU cores that the measured object has applied for. – Data sending rate, which indicates the data volume sent by the measured object per second. – Number of error packets received by the measured object. – Number of CPU cores used by the measured object.
Trigger Condition	Condition based on which the scaling policy is triggered.
Duration	Metric statistics period. For example, if the parameter is set to 20s, metric statistics is collected every 20s.
Continuous Cycle	Number of consecutive times that the threshold is triggered. For example, if the parameter is set to **3**, the action is triggered if the threshold is reached for three consecutive measurement periods.
Action	Select **Add** or **Reduce** to set the action to be executed after the policy is triggered.

NOTE

Click **Show/Hide Preview** to set **Triggering Condition**, **Duration**, **Continuous Cycle**, and **Action**.

- **Scheduled Policy**: Instances in an application can be automatically scaled in or out at a specified time.

Table 3-21 Parameters for adding a scheduled policy

Parameter	Description
Policy Name	Name of the scaling policy.
Policy Type	Choose **Scheduled Policy**.
Trigger Time	Set the time at which the policy is enforced.
Action	Select **Add**, **Reduce**, or **Set** to set the action to be executed after the policy is triggered.

- **Periodic Policy**: Scaling policies can be executed daily, weekly, or monthly. This policy is applicable to scenarios where traffic changes periodically.

  Table 3-22 Parameters for adding a periodic policy

Parameter	Description
Policy Name	Name of the scaling policy.
Policy Type	Choose **Periodic Policy**.
Trigger Time	Set the time at which the policy is enforced.
Action	Select **Add**, **Reduce**, or **Set** to set the action to be executed after the policy is triggered.

Step 7 Click **OK**.

In the **Auto Scaling** area, check that the policy has been started. When the trigger is met, the auto scaling policy immediately takes effect.

----End

3.5.6 Configuring a Scheduling Policy of an Application Component Instance

ServiceStage provides a variety of scheduling policies, including static global scheduling policies and dynamic runtime scheduling policies. You can select or combine these policies as required.

Concepts

- Application-AZ Affinity and Anti-Affinity

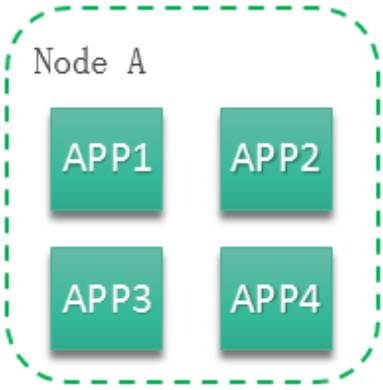
  - **Affinity with AZs**: Application components can be deployed in specific AZs.
  - **Non-affinity with AZs**: Application components cannot be deployed in specific AZs.

- 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 3-3** shows affinity deployment, in which all applications are deployed on the same node.

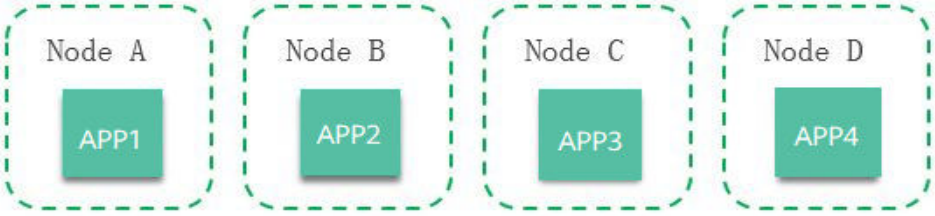
    Figure 3-3 Application affinity

  - **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 3-4**, four applications are deployed on four different nodes. The four applications are deployed in anti-affinity mode.

    Figure 3-4 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

When the component type is **Common** and the runtime system is **Docker**, perform the following operations:

Step 1 Access the page for setting a scheduling policy of an application component instance.

- To set a scheduling policy during **component configuration** in the application component deployment, go to **Step 2**.
- To set a scheduling policy after an application component is deployed, go to **Step 3**.

Step 2 On the **Configure Component** page:

1. Configure the scheduling policy for the application component instance based on 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**.

Purpose	Procedure
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.

2. Click **Next** to complete the component deployment.

Step 3 Log in to ServiceStage and choose **Application Management** > **Application List** to view all applications.

Step 4 Click an application. The **Overview** page is displayed.

Step 5 On the **Environment View** tab, select an environment to view the application components that have been deployed in the environment.

Step 6 Click an application component. The **Overview** page is displayed.

Step 7 Choose **Scheduling Policy** and set the following parameters:

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**.

Purpose	Procedure
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.

Step 8 Click **Re-deployment** to complete the setting.

----End

For other types of components, perform the following operations:

Step 1 Access the page for setting a scheduling policy of an application component instance.

- To set a scheduling policy during **component configuration** in the application component deployment, go to **Step 6**.
- To set a scheduling policy after an application component is deployed, go to **Step 2**.

Step 2 Log in to ServiceStage and choose **Application Management** > **Application List** to view all applications.

Step 3 Click an application. The **Overview** page is displayed.

Step 4 On the **Environment View** tab, select an environment to view the application components that have been deployed in the environment.

Step 5 Click an application component. The **Overview** page is displayed. Choose **Upgrade**.

Step 6 Choose **Advanced Settings** > **Deployment Configuration** and set the following parameters on the **Scheduling Policy** tab.

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**.

Purpose	Procedure
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.

Step 7 Complete the settings of the scheduling policy.

- If the scheduling policy is set during **component configuration**, click **Next**.
- If the scheduling policy is set after the application component is deployed, click **Re-deployment**.

----End

3.5.7 Configuring an Upgrade Policy of an Application Component Instance

You can configure an upgrade policy during or after application component deployment.

Procedure

When the component type is **Common** and the runtime system is **Docker**, perform the following operations:

Step 1 Access the page for setting an upgrade policy of an application component instance.

- To set an upgrade policy during **component configuration** in the application component deployment, go to **Step 2**.
- To set an upgrade policy after an application component is deployed, go to **Step 3**.

Step 2 On the **Configure Component** page, configure the upgrade policy for the application component instance.

1. Select an upgrade mode for the application component instance.

   NOTE

   The default upgrade mode is **Rolling upgrade**.

   - Rolling upgrade

     Install a new instance and then remove the old one. In this pattern, services are evenly distributed to new and old instances during the upgrade, so services are not interrupted.

   - In-place upgrade

     Delete the old instance and then create a new one. Services are interrupted during the upgrade.

2. Click **Next** to complete the component deployment.

Step 3 Log in to ServiceStage and choose **Application Management** > **Application List** to view all applications.

Step 4 Click an application. The **Overview** page is displayed.

Step 5 On the **Environment View** tab, select an environment to view the application components that have been deployed in the environment.

Step 6 Click an application component. The **Overview** page is displayed.

Step 7 Choose **Upgrade** to configure the upgrade policy.

1. Select an upgrade mode for the application component instance.

   NOTE

   The default upgrade mode is **Rolling upgrade**.

   - Rolling upgrade

     Install a new instance and then remove the old one. In this pattern, services are evenly distributed to new and old instances during the upgrade, so services are not interrupted.

   - In-place upgrade

     Delete the old instance and then create a new one. Services are interrupted during the upgrade.

2. Click **Re-deployment** to complete the setting.

----End

For other types of components, perform the following operations:

Step 1 Access the page for setting an upgrade policy of an application component instance.

- To set an upgrade policy during **component configuration** in the application component deployment, go to **Step 6**.

- To set an upgrade policy after an application component is deployed, go to **Step 2**.

Step 2 Log in to ServiceStage and choose **Application Management** > **Application List** to view all applications.

Step 3 Click an application. The **Overview** page is displayed.

Step 4 On the **Environment View** tab, select an environment to view the application components that have been deployed in the environment.

Step 5 Click an application component. The **Overview** page is displayed. Choose **Upgrade**.

Step 6 Choose **Advanced Settings** > **Deployment Configuration**. On the **Upgrade Policy** tab page, select the upgrade mode.

NOTE

The default upgrade mode is **Rolling upgrade**.

- Rolling upgrade

  Install a new instance and then remove the old one. In this pattern, services are evenly distributed to new and old instances during the upgrade, so services are not interrupted.

- In-place upgrade

  Delete the old instance and then create a new one. Services are interrupted during the upgrade.

Step 7 Complete the setting of the upgrade policy.

- If the upgrade policy is set during **component configuration**, click **Next**.
- If the upgrade policy is set after the application component is deployed, click **Re-deployment**.

----End

3.5.8 Configuring Custom Monitoring of an Application Component

ServiceStage allows you to obtain monitoring data based on custom metrics.

You can configure custom metric monitoring during or after application component deployment.

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

When the component type is **Common** and the runtime system is **Docker**, perform the following operations:

Step 1 Access the page for setting custom metric monitoring for an application component.

- To set custom metric monitoring during **component configuration** in the application component deployment, go to **Step 2**.
- To set custom metric monitoring after an application component is deployed, go to **Step 3**.

Step 2 On the **Configure Component** page:

1. Specify the following parameters to set custom metric monitoring for the application component.

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

2. Click **Next** to complete the component deployment.

Step 3 Log in to ServiceStage and choose **Application Management** > **Application List** to view all applications.

Step 4 Click an application. The **Overview** page is displayed.

Step 5 On the **Environment View** tab, select an environment to view the application components that have been deployed in the environment.

Step 6 Click an application component. The **Overview** page is displayed.

Step 7 Choose **O&M Configurations** and set the following parameters.

Parameter	Description	Mandatory
Report Path	URL provided by the exporter for ServiceStage to obtain custom metric data. Example: **/metrics**	Yes

Parameter	Description	Mandatory
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

Step 8 Click **Re-deployment** to complete the setting.

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

For other types of components, perform the following operations:

Step 1 Access the page for setting custom metric monitoring for an application component.

- To set custom metric monitoring during **component configuration** in the application component deployment, go to **Step 6**.
- To set custom metric monitoring after an application component is deployed, go to **Step 2**.

Step 2 Log in to ServiceStage and choose **Application Management** > **Application List** to view all applications.

Step 3 Click an application. The **Overview** page is displayed.

Step 4 On the **Environment View** tab, select an environment to view the application components that have been deployed in the environment.

Step 5 Click an application component. The **Overview** page is displayed. Choose **Upgrade**.

Step 6 Choose **Advanced Settings** > **O&M Monitoring**. On the **O&M Policy** tab, set the following parameters.

Parameter	Description	Mandatory
Report Path	URL provided by the exporter for ServiceStage to obtain custom metric data. Example: **/metrics**	Yes

Parameter	Description	Mandatory
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

Step 7 Complete the setting of the custom metric monitoring.

- If the custom metric monitoring policy is set during **component configuration**, and click **Next**.
- If the custom metric monitoring policy is set after the application component is deployed, and click **Re-deployment**.

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

3.5.9 Configuring a Log Policy of an Application

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 application component deployment.

If no configuration is performed, the system collects standard application output logs by default.

Procedure

Step 1 Go to the log policy configuration page.

- To configure the application log policy during **component configuration** in the component deployment, go to **Step 6**.
- To configure the application log policy after the component is deployed, go to **Step 2**.

Step 2 Log in to ServiceStage and choose **Application Management** > **Application List** to view all applications.

Step 3 Click an application. The **Overview** page is displayed.

Step 4 On the **Environment View** tab, select an environment to view the application components that have been deployed in the environment.

Step 5 Click an application component. The **Overview** page is displayed. Choose **Upgrade**.

Step 6 Choose **Advanced Settings** > **O&M Monitoring** > **Log Collection**, click **Add Log Policy**, and set the parameters listed in the following table.

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.
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**. – **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 **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 to a historical file, dumps the historical file to the directory where the log file is stored, and clears the original log file.

Step 7 Click **Confirm**.

Step 8 Complete the application log policy configuration.

- If the log policy is set during **component configuration** in the component deployment, click **Next**.
- If the log policy is set after the component is deployed, click **Re-deployment**.

NOTE

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

----End

3.5.10 Configuring Health Check

Health check periodically checks application health status during application 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.

Procedure

Step 1 Go to the application health check configuration page.

- To configure health check during **component configuration** in the application component deployment, go to **Step 6**.
- To configure health check after the component is deployed, go to **Step 2**.

Step 2 Log in to ServiceStage and choose **Application Management** > **Application List** to view all applications.

Step 3 Click an application. The **Overview** page is displayed.

Step 4 On the **Environment View** tab, select an environment to view the application components that have been deployed in the environment.

Step 5 Click an application component. The **Overview** page is displayed. Choose **Upgrade**.

Step 6 Choose **Advanced Settings** > **O&M Monitoring**, click **Health Check**, and configure health check.

Step 7 Complete the application health check configuration.

- If the health check is configured during **component configuration** in the component deployment, click **Next**.
- If the health check is configured after the component is deployed, click **Re-deployment**.

----End

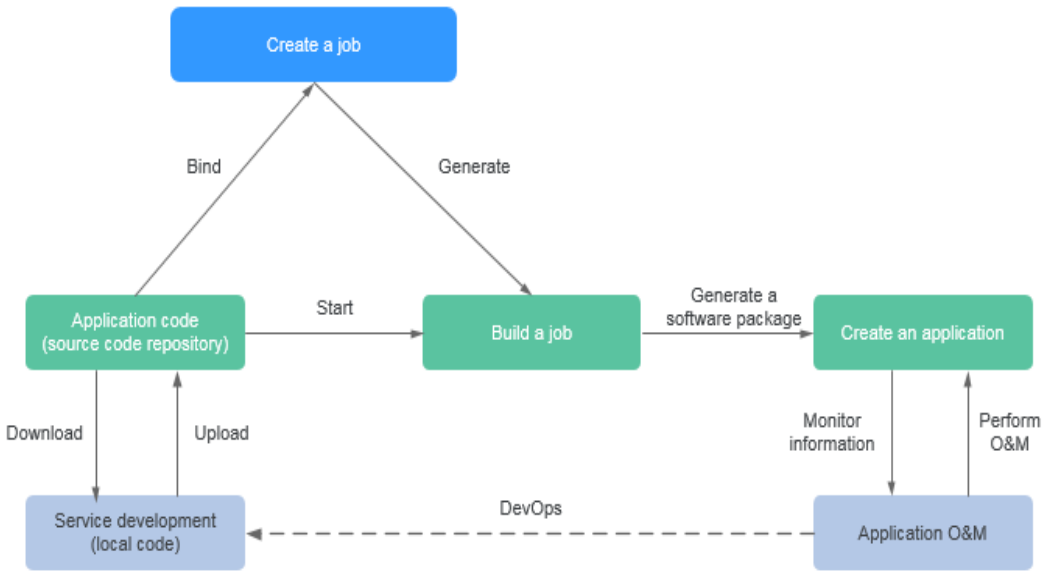
3.6 Continuous Delivery

3.6.1 Overview

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 software center. Then, you can use the package when deploying an application component.

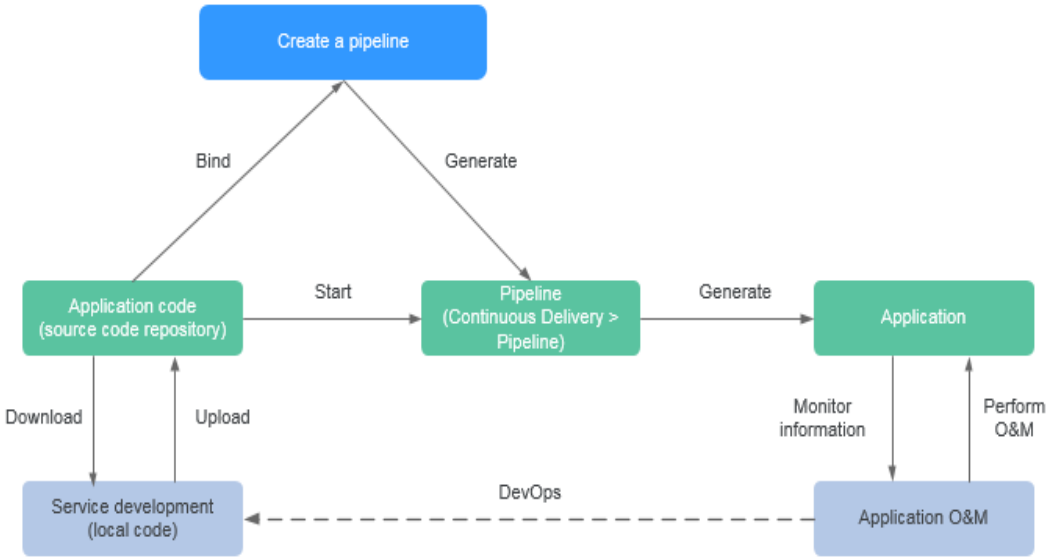
Figure 3-5 Creating a build job



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 3-6 Creating a pipeline



3.6.2 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**.
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. (Optional) Enter **Description**.
3. Set **Code Source**.

- Create authorization by referring to **Authorizing a Repository** and set the code source.
- Click **Samples** and select a required sample.

4. Select a cluster from the **Cluster** drop-down list.
5. (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 node label, see **Adding a Node Label**.
6. 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	Task name.

Parameter	Description
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.
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

Related Operations

After an application component is successfully built, you can manage it on ServiceStage. For details, see **Deployment Mode**.

Maintenance

Table 3-23 Maintenance

Operation	Description
Query details/build history	1. Click the target build project and view the build history under **Build Record**. 2. Click a record to view the record. 3. Click **Code Check** to view the code check overview and details. Currently, the following code check plug-ins are supported: Checkstyle, FindBugs, and PMD. **NOTE** Only the Maven build project supports code check.
Build Now	Select the target build project and click **Build Now** in the **Operation** column.

Operation	Description
Branch/Tag	Select the target build project created based on source code and click **Branch/Tag** in the **Operation** column. 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. 4. Click **OK**.
Edit	Select the target build project you create and choose **More** > **Edit** in the **Operation** column to edit the build project.
Delete	1. Select the target build project you create and choose **More** > **Delete** in the **Operation** column. 2. Click **OK**.

3.6.3 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**.
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 (Optional) Enter **Description**.

Step 4 Set **Package Source**.

The following upload modes are supported:

- Select the corresponding software package from the OBS bucket. You need to 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 5 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 4**.

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 4**.

Step 6 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 7 Set **Archived Image Address**.

Step 8 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 node label, see **Adding a Node Label**.

Step 9 Click **Build Now** to start the build.

Click **Save** to save the settings (not to start the build).

----End

Related Operations

After an application component is successfully built, you can manage it on ServiceStage. For details, see **Deploying an Application Component**.

Maintenance

Table 3-24 Maintenance

Operation	Description
Query details/build history	1. Click the target build project and view the build history under **Build Record**. 2. Click a record to view the record.

Operation	Description
Build Now	Select the target build project and click **Build Now** in the **Operation** column.
Edit	Select the target build project and choose **More** > **Edit** in the **Operation** column to edit the build project.
Delete	1. Select the target build project and choose **More** > **Delete** in the **Operation** column. 2. Click **OK**.

3.6.4 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.

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 a pipeline name.
2. (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 3-25 Template description

Template	Description	Operation 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**.

Template	Description	Operation Description
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**.

   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 **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 3-26 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**.
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**.

Policy	Mode	Description
	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 the configuration information. Modify the configuration parameters 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

Related 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 **Application O&M**.

3.6.5 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 3-27 Authorization information

Parameter	Description
*Name	Authorization name, which cannot be changed after being created.
*Repository Type	The following repositories are supported: • GitHub Authorization mode: OAuth or private token. • Gitee Authorization mode: OAuth or private token. • Bitbucket Authorization mode: OAuth, password, or private Bitbucket. • GitLab Authorization mode: OAuth or private token.

Step 3 Click **Create**.

----End

3.7 Software Center

3.7.1 Image Repository

3.7.1.1 Uploading an Image

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

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

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.

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 **Basics of the Container Engine**.
- The image package is created using Docker 1.11.2 or later.

Procedure

Step 1 Log in to ServiceStage and choose **Software Center** > **Image Repository**.

Step 2 On the **My Images** page, click **Upload Through SWR**.

Step 3 Click **Upload Through SWR**. In the dialog box that is displayed, specify **Organization** to which the image is to be uploaded, click **Select 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 4 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.

Prerequisites

- An organization has been created. For details, see **Creating an Organization**.
- Your container engine client version must be 1.11.2 or later.

Procedure

Step 1 Log in to ServiceStage and choose **Software Center** > **Image Repository**.

Step 2 On the **My Images** page, click **Upload Through Docker Client**.

Step 3 Upload the image as prompted.

----End

3.7.1.2 Managing Images

Obtaining an Image Pull Address

Step 1 Log in to ServiceStage and choose **Software Center** > **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 **Image Tags** tab and obtain the command for pulling an image.

Click on the right of the command to copy the command.

----End

Setting Image Repository Attributes

Step 1 Log in to ServiceStage and choose **Software Center** > **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 **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**.

Step 5 Click **OK**.

----End

Adding Image Permissions

To allow users of your account to read, write, and manage a specific image, add the required permissions to the users on the details page of this image.

Step 1 Log in to ServiceStage and choose **Software Center** > **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 user, add the **Read**, **Write**, or **Manage** permission, and click **OK**.

Then, this user has the corresponding permission.

----End

Deleting an Image

Step 1 Log in to ServiceStage and choose **Software Center** > **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.

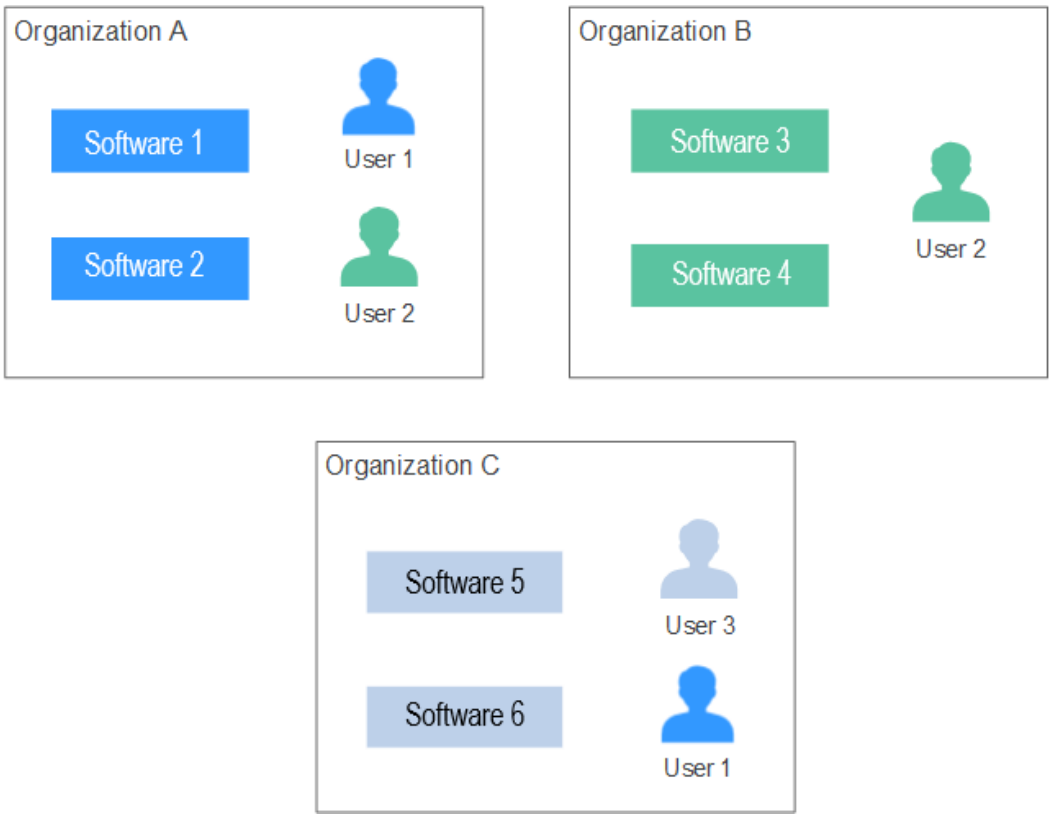
----End

3.7.2 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 user can join different organizations. Different permissions, namely read, write, and manage, can be assigned to different users in the same account.

Figure 3-7 Organization



Creating an Organization

Step 1 Log in to ServiceStage and choose **Software Center** > **Organization**.

Step 2 Click **Create Organization**, enter **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 **Software Center** > **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 **Software Center** > **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 **Managing Images**.

Step 3 Click **OK**.

----End

3.8 Infrastructure

3.8.1 Cloud Service Engine (CSE)

3.8.1.1 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. Furthermore, it supports multiple languages and runtime systems, and unified access and governance of intrusive frameworks such as Spring Cloud, Apache ServiceComb (Java chassis/Go chassis), and Dubbo, and non-intrusive service meshes.

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. You can customize specifications and features, and create a microservice engine with a specific instance quantity.
- 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.

3.8.1.2 Creating an Exclusive Microservice Engine

The exclusive microservice engine is deployed in physical isolation mode. Tenants exclusively use the microservice engine. You can create an exclusive microservice engine based on service requirements.

Prerequisites

An exclusive microservice engine runs on a VPC. Before creating a microservice engine, ensure that VPCs and subnets are available.

Create a VPC and subnet. For details, see .

Procedure

Step 1 Log in to ServiceStage and choose **Infrastructure** > **Cloud Service Engines**.

Step 2 Click **Create Microservice Engine** in the upper part of the page.

Step 3 Set the parameters. For details, see **Table 3-28**.

Table 3-28 Description

Parameter	Description
Microservice Engine Name	Name of the microservice engine. The name cannot be changed after the engine is created.
(Optional) Description	Description of the microservice engine.
Specification	Select the engine instance specifications. • **HA** You can select up to two AZs with the same CPU architecture. The service is more reliable and can be used in the production environment. • **Non-HA** You can select only one AZ. If a node is faulty, the non-HA engine is unavailable, affecting some service functions. After a non-HA engine is created, it cannot be switched to an HA engine. It is recommended that this option be used only in the service development and test phases.
AZ	Availability zone. • The AZ of a created microservice engine cannot be changed. • The AZs in one region can communicate with each other over an intranet.
Instances	Maximum number of microservice instances supported by a microservice engine.
VPC	VPC in which the microservice engine locates. A VPC enables you to provision logically isolated, configurable, and manageable virtual networks for your engine.
Subnet	Subnet.

Step 4 Click **Next**. Confirm the specifications and click **Submit** to create the microservice engine.

- It takes 10 to 30 minutes to create an exclusive microservice engine.

- After the microservice engine is created, the microservice engine status changes to **Available**.
- If the microservice engine fails to be created, click and choose **Retry**.

NOTE

If the retry fails, delete the failed microservice engine. For details about how to delete an exclusive microservice engine, see **Deleting an Exclusive Microservice Engine**.

If the failed microservice engine is not deleted in a timely manner, it will occupy compute resources. The system clears resources at 18:00 (UTC time) every day.

----End

3.8.1.3 Microservice Engine Management

3.8.1.3.1 Configuring Backup and Restoration of an Exclusive Microservice Engine

You can customize backup policies to periodically back up microservice engines or manually back up microservice engines.

Context

- This function applies only to exclusive microservice engines.
- 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 **Infrastructure** > **Cloud Service Engines**.

Step 2 Click an exclusive microservice engine. The **Basic Information** page is displayed.

Step 3 Click the **Backup and Restoration** tab, click **Backup Policy**, and set backup parameters.

Table 3-29 Backup parameter description

Parameter	Description
Automatic Backup	After automatic backup is disabled, the previous backup policy will be deleted. In this case, set the backup policy again.
Backup Cycle	Backup period.
Start Time	Time at which a backup task starts. Only the hour is supported.

Step 4 Click **OK** to complete the configuration of the backup policy.

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 **Infrastructure** > **Cloud Service Engines**.

Step 2 Click an exclusive microservice engine. The **Basic Information** page is displayed.

Step 3 Click the **Backup and Restoration** tab, click **Manual Backup**, and set backup parameters.

Table 3-30 Backup parameter description

Parameter	Description
Name	Name of a backup task. The name cannot be changed after the backup task is created.
Remarks	Description about the backup task. This field is optional.

Step 4 Click **OK** to execute the backup task immediately.

----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.

Step 1 Log in to ServiceStage and choose **Infrastructure** > **Cloud Service Engines**.

Step 2 Click an exclusive microservice engine. The **Basic Information** page is displayed.

Step 3 Click the **Backup and Restoration** tab, and click **Restore** in the **Operation** column next to the specified backup data.

Step 4 Select **I have read and fully understood the risk** and click **OK** to restore the backup data.

To view the restoration status, click **Restoration History**.

----End

3.8.1.3.2 Configuring Public Network Access of an Exclusive Microservice Engine

Exclusive microservice engines that are in the **Available** state 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.

Enabling Public Network Access

Step 1 Log in to ServiceStage and choose **Infrastructure** > **Cloud Service Engines**.

Step 2 Click the microservice engine for which public network access is to be enabled. The **Basic Information** page is displayed.

Step 3 Enable **Public Network Access**.

Step 4 In the displayed dialog box, select the warning information and click **OK**.

Step 5 Click the drop-down list next to **EIP**. Select an EIP from the drop-down list and click √.

If no EIP is available, click **Create EIP** to create one.

----End

Disabling Public Network Access

Step 1 Log in to ServiceStage and choose **Infrastructure** > **Cloud Service Engines**.

Step 2 Click the microservice engine for which public network access is to be disabled. The **Basic Information** page is displayed.

Step 3 Disable **Public Network Access**.

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

----End

3.8.1.3.3 Viewing the Access Address of a Microservice Engine

You can view the access address only after an exclusive microservice engine is created. To view the access address of a microservice engine, perform the following steps:

Procedure

Step 1 Log in to ServiceStage and choose **Infrastructure** > **Cloud Service Engines**.

Step 2 Locate the target exclusive microservice engine, and view or click to copy the service center address.

----End

3.8.1.3.4 Viewing Operation Logs of an Exclusive Microservice Engine

Operations such as creation, upgrade, deletion, and change on the microservice engine will be performed in the backend. You can view the task execution status in the list.

Procedure

Step 1 Log in to ServiceStage and choose **Infrastructure** > **Cloud Service Engines**.

Step 2 Click an exclusive microservice engine. The **Basic Information** page is displayed.

Step 3 Click the **Other Operations** tab and click an operation type to view the operation logs.

----End

3.8.1.3.5 Upgrading an Exclusive Microservice Engine

Exclusive 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.

Context

During the upgrade, the performance of HA engines is different from that of non-HA engines.

- For HA engines, 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, Go chassis, 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.
- A non-HA engine has only one service instance. During the upgrade, services are interrupted and cannot be registered, discovered, or modified. Therefore, you need to evaluate whether the service is affected during an upgrade. The reliability of ServiceComb SDK, Go chassis, and Mesher is enhanced. During service interruption, the client caches data. If an empty instance is found, the

client uses the local cache. If you call the APIs of the service center and configuration center for service registry and discovery, you need to use cache.

Procedure

Step 1 Log in to ServiceStage and choose **Infrastructure** > **Cloud Service Engines**.

Step 2 Select an available microservice engine that can be upgraded, click , and choose **Upgrade**.

You can also click the target microservice engine and click **Upgrade** in the upper right corner of the details page.

Step 3 Select the target version and view the version description.

Step 4 Click **OK**.

If the upgrade fails, click and choose **Retry**.

----End

3.8.1.3.6 Deleting an Exclusive 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.

Context

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 **Infrastructure** > **Cloud Service Engines**.

Step 2 Click the target exclusive microservice engine to go to the **Basic Information** page.

1. In the upper right corner of the page, click **Delete**.
2. In the displayed dialog box, enter **DELETE** and click **OK**.

----End

3.8.1.4 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 center address, configuration center address, and dashboard address. You do not need to configure the monitor address. Currently, only Java chassis and Go chassis support automatic discovery of a dashboard 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 Log in to ServiceStage and choose **Infrastructure** > **Cloud Service Engines**.

Step 2 Select a microservice engine and click **Console**.

Step 3 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 4 Select a sorting order to sort the filtered microservices.

----End

3.8.1.5 Microservice Governance

3.8.1.5.1 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. Service governance applies only to Java Chassis and Go Chassis development frameworks.

Currently, ServiceStage provides the Cloud Service Engine (professional edition). You can use it directly. You can also create exclusive microservice engines. For details, see **Creating an Exclusive Microservice Engine**.

3.8.1.5.2 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, you need to register the service instance before performing the corresponding operation.

Governance Policies

Supports the configuration of policies such as load balancing, rate limiting, fault tolerance, service degradation, circuit breaker, and fault injection. For details, see the following table.

Name	Description
Load Balancing	When the access traffic and traffic volume are large and one server cannot handle the load, you can configure load balancing to distribute traffic to multiple servers for balancing. In this way, the response duration is reduced and server overload can be prevented. You can configure load balancing policies by adding a rule. The rule parameters include **Polling**, **Random**, **Response Time Weight**, and **Session Stickiness**.
Rate Limiting	Rate limiting is used to solve the problem of traffic distribution across microservices. This ensures that microservices run in their own resource pools without affecting each other. ● 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. ● Common detection methods include request timeout and excessive traffic. ● The parameters include **Flow Control Object** and **QPS**.
Service Degradation	Service degradation is a special form of fault tolerance. When the service throughput is large and resources are insufficient, you can use service degradation to disable some services that are not important and have poor performance to avoid occupying resources and ensure that the main services are normal.
Fault Tolerance	Fault tolerance is used when an exception occurs in a service instance after you access that instance. After the exception occurs, you can retry to access the instance, or access another instance based on the configured policy.

Name	Description
Circuit Breaker	If the service is overloaded, you can use circuit breaker to protect the system from breaking down. Circuit breaker is triggered when a service request is handled abnormally. After circuit breaker is triggered, Hystrix considers that the requested service cannot process requests, so it immediately rejects requests and returns an error message to the caller. Hystrix attempts to access backend services at a specified interval. If the services are restored, they will exit the circuit breaker state and resume to accept requests.
Fault Injection	Fault injection is used to test the fault tolerance capability of microservices. This helps the user determine whether the system can run properly when latency or fault occurs. Fault injection allows you to test fault tolerance of microservices with latency or faults.
Routing Policy	Based on the public key authentication mechanism, CSE provides the blacklist and whitelist functions to control the services that can access microservices. The blacklist and whitelist take effect only after public key authentication is enabled. For details, see **Configuring Public Key Authentication**.

Configuring Load Balancing

Step 1 Log in to ServiceStage and choose **Infrastructure** > **Cloud Service Engines**.

Step 2 Click **Console** of a microservice engine.

Step 3 Choose **Service Governance**.

Step 4 Click the microservice to be governed.

Step 5 Click **Load Balancing**.

Step 6 Click **Add**. Select the microservices to be governed and select a proper load balancing policy. For details, see the following table.

Policy	Description
Round robin	Supports routes according to the location information about service instances.
Random	Provides random routes for service instances.
Response time weight	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.

Policy	Description
Session stickiness	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. • **Session Stickiness Duration**: session hold time. 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 7 Click **OK** to save the settings.

----End

Configuring Rate Limiting

Step 1 Log in to ServiceStage and choose **Infrastructure** > **Cloud Service Engines**.

Step 2 Click **Console** of a microservice engine.

Step 3 Choose **Service Governance**.

Step 4 Click the microservice to be governed.

Step 5 Click **Rate Limiting**.

Step 6 Click **Add**. The following table describes the configuration items of rate limiting.

Configuration Item	Description	Value Range
Object	Other microservices that access the microservice.	You can set the item in the drop-down list box next to **Object**.
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.	An integer ranging from 0 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 7 Click **OK** to save the settings.

----End

Configuring Service Degradation

Step 1 Log in to ServiceStage and choose **Infrastructure** > **Cloud Service Engines**.

Step 2 Click **Console** of a microservice engine.

Step 3 Choose **Service Governance**.

Step 4 Click the microservice to be governed.

Step 5 Click **Service Degradation**.

Step 6 Click **Add**. Select a proper policy. The following table describes the configuration items of service degradation.

Configuration Item	Description
Object	Microservice to be degraded and the corresponding degradation method.
Policy	• Enable • Disable

Step 7 Click **OK** to save the settings.

----End

Configuring Fault Tolerance

Step 1 Log in to ServiceStage and choose **Infrastructure** > **Cloud Service Engines**.

Step 2 Click **Console** of a microservice engine.

Step 3 Choose **Service Governance**.

Step 4 Click the microservice to be governed.

Step 5 Click **Fault Tolerance**.

Step 6 Click **Add**. Select a proper policy. The following table describes the configuration items of fault tolerance.

Configuration Item	Description
Object	Microservice or method that the application relies on. You can select it from the drop-down list.

Configuration Item	Description
Degradation Mode	**Enable**: The system processes the service request based on the selected fault tolerance policy when the request sent to the fault tolerance object encounters an error. **Disable**: The system waits until the timeout interval expires and then returns the failure result even though the service request fails to be implemented.
Policy **NOTE** Set this parameter when **Degradation Mode** is set to **Enable**.	• 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 7 Click **OK** to save the settings.

----End

Configuring Circuit Breaker

Step 1 Log in to ServiceStage and choose **Infrastructure** > **Cloud Service Engines**.

Step 2 Click **Console** of a microservice engine.

Step 3 Choose **Service Governance**.

Step 4 Click the microservice to be governed.

Step 5 Click **Circuit Breaker**.

Step 6 Click **Add**. Select a proper policy. The following table describes the configuration items of circuit breaker.

Configuration Item	Description
Object	Microservice or method invoked by the application.

Configuration Item	Description
Trigger Condition	• **Manual** Circuit breaker is triggered immediately and microservice instances are not called. • **Cancel** Circuit breaker taking effect on the microservice instance is canceled and the microservice instance can be called. • **Automatic** – **Circuit Breaker Time Window**: circuit breaker duration. No response is sent within the time window. – **Failure Rate**: failure rate of window requests, which is a triggering condition. – **Window Requests**: number of requests received by the window, which is a triggering condition. Circuit breaker is triggered only when **Failure Rate** and **Window Requests** both reach their thresholds.

Step 7 Click **OK** to save the settings.

----End

Configuring Fault Injection

Step 1 Log in to ServiceStage and choose **Infrastructure** > **Cloud Service Engines**.

Step 2 Click **Console** of a microservice engine.

Step 3 Choose **Service Governance**.

Step 4 Click the microservice to be governed.

Step 5 Click **Fault Injection**.

Step 6 Click **Add**. Select a proper policy. The following table describes the configuration items of fault injection.

Configuration Item	Description
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. • **Latency** • **Error**

Configuration Item	Description
Protocol	Protocol for accessing the microservice when latency or fault occurs. • Rest • Highway
Latency	Latency for accessing a microservice. This parameter is required when **Type** is set to **Latency**.
HTTP Error Code	HTTP error code displayed during microservice access. This parameter is required when **Type** is set to **Error**. This error code is an HTTP error code.
Trigger Probability	Probability of latency or fault occurrence.

Step 7 Click **OK** to save the settings.

----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 **Infrastructure** > **Cloud Service Engines**.

Step 2 Click **Console** of a microservice engine.

Step 3 Choose **Service Governance**.

Step 4 Click the microservice to be governed.

Step 5 Click **Blacklist and Whitelist**.

Step 6 Click **Add** to add a blacklist or whitelist for the application. The following table describes the configuration items of blacklist and whitelist.

Configuration Item	Description
Type	• **Blacklist**: Microservices matching the matching rule are not allowed to access the current service. • **Whitelist**: Microservices matching the matching rule are allowed to access the current service.

Configuration Item	Description
Matching Rule	Expressed by a regular expression. For example, if **Matching Rule** is set to **data***, services whose name starts with **data** in the blacklist cannot access the current service, or services whose name starts with **data** in the whitelist can access the current service.

Step 7 Click **OK** to save the settings.

----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 a 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>
```

3.8.1.6 Configuring Microservices

You can use the global configuration function provided by ServiceStage to configure microservices.

After the global configuration is added, the configuration takes effect immediately if it is used by all microservices registered with the engine.

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**.

Global Configuration

Global configuration provides common configurations for microservices, such as log levels and running parameters. After being added, the global configuration is used as the default configuration 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 **Infrastructure** > **Cloud Service Engines**.

Step 2 Click **Console** of a microservice engine.

Step 3 Choose **Global Configuration**. Then, you can perform the following operations:

Operation	Procedure
Export	Click **Export** to export all global configuration items.
Import	1. Click **Import**. 2. Click ··· to select a configuration item file. 3. Click **Upload** to import configuration items in batches.
Creation	1. Click **Create Configuration**. The **Create Configuration** dialog box is displayed. 2. Select a microservice environment and enter **Configuration Item** and **Value**. 3. Click **OK** to save the settings.
Modification	1. Click **Edit** in the **Operation** column corresponding to the target configuration item. 2. Enter **Value**. 3. Click **OK** to save the settings.
Deletion	1. Select the configuration item to be deleted. 2. Click **Delete** in the **Operation** column. 3. In the **Delete Configuration** dialog box, click **OK** to delete the global configuration.
Batch deletion	1. Select the configuration items to be deleted. 2. Click **Delete** above the configuration item list to delete global configuration items in batches.

----End

3.8.1.7 Maintaining Microservices

You can use service catalogs to view microservice details and search for target microservices to maintain microservices.

The following information is displayed on the **Service Catalog** page:

- **Application List**: displays all applications of the current user. You can search for the target application by application name, or filter applications by environment.
- **Microservice List**: displays all microservices of the current user. You can search for the target microservice by microservice name, or filter microservices by environment and application.
- **Instance List**: displays all instances of the current user. You can search for the target instance by microservice name, or filter instances by environment or application.

Dynamic Configuration

Step 1 Log in to ServiceStage and choose **Infrastructure** > **Cloud Service Engines**.

Step 2 Click **Console** of a microservice engine.

Step 3 Choose **Service Catalog**.

Step 4 Click a microservice.

Step 5 Choose **Dynamic Configuration**. The **Dynamic Configuration** page is displayed. On the **Dynamic Configuration** tab, perform the following operations.

NOTICE

Configuration items are stored in plaintext. Do not include sensitive data.

Operation	Procedure
Export configurations.	Select a scope from the **All** drop-down list and click **Export** to export the JSON configuration file of the current scope. The scope format is as follows: • *Microservice name*@*Application to which the microservice belongs* • *Microservice name*@*Application to which the microservice belongs*#*Version number*
Import configurations.	1. Click **Import** and select a scope. The scope format is as follows: – *Microservice name*@*Application to which the microservice belongs* – *Microservice name*@*Application to which the microservice belongs*#*Version number* 2. Click ··· to select a configuration item file. 3. Click **Upload** to import configuration items in batches. 4. Click **Close**.

Operation	Procedure
Create configurations.	1. Click **Create Configuration** and select a scope. 2. Enter **Configuration Item**. 3. Enter **Value**. 4. Click **OK** to save the settings.
Modify configurations.	1. Click **Edit** in the **Operation** column corresponding to the target configuration item. 2. Enter a new value in the **Value** text box. 3. Click **OK** to save the settings.
Delete configurations.	1. Click **Delete** in the **Operation** column corresponding to the target configuration item. 2. Click **OK** to delete the configuration.

----End

Dark Launch

Step 1 Log in to ServiceStage and choose **Infrastructure** > **Cloud Service Engines**.

Step 2 Click **Console** of a microservice engine.

Step 3 Choose **Service Catalog**.

Step 4 Click a microservice. On the displayed page, click **Dark Launch**.

Step 5 Click **Add Launch Rule**.

- To add a launch rule by weight:
  a. Click **Weight**.
  b. Set the following parameters.

Configuration Item	Description
Rule Name	Name of a customized rule.
Scope	▪ Version of the microservice to which the rule applies. ▪ **Add custom version**: adds 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 customize a launch rule:

a. Click **Customize**.

b. Set the following parameters.

Configuration Item	Description
Rule Name	Name of a customized rule.
Scope	▪ Version of the microservice to which the rule applies. ▪ **Add custom version**: adds a new version as prompted.
Rule Configuration	▪ **Parameter Name** This name is 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 (?) can be used for fuzzy matching when you specify the value of **Rules**. The asterisk (*) represents an unlimited number of characters, and the question mark (?) represents only one character. 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 (?) cannot be used for fuzzy matching.

c. Click **OK** to complete the customized rule configuration and dark launch.

----End

Delete a microservice.

Step 1 Log in to ServiceStage and choose **Infrastructure** > **Cloud Service Engines**.

Step 2 Click **Console** of a microservice engine.

Step 3 Choose **Service Catalog**.

Step 4 Select the microservice to be deleted, click **Delete**, and delete the microservice as prompted.

NOTE

- If the number of microservice instances is 0, you can directly delete the microservice.
- If the number of microservice instances is not 0, the microservice will be re-registered with the service center after being deleted for a period of time.

----End

Viewing Microservice Details

Step 1 Log in to ServiceStage and choose **Infrastructure** > **Cloud Service Engines**.

Step 2 Click **Console** of a microservice engine.

Step 3 Choose **Service Catalog**.

Step 4 Click a microservice. The microservice details page is displayed.

On the microservice details page, you can view the instance list, called services, calling services, dynamic configuration, dark launch, and service contract.

----End

Viewing a Service Contract

Step 1 Log in to ServiceStage and choose **Infrastructure** > **Cloud Service Engines**.

Step 2 Click **Console** of a microservice engine.

Step 3 Choose **Service Catalog**.

Step 4 Click a microservice. The microservice details page is displayed.

Step 5 Click **Service Contract** to view the service contract.

----End

Adding a Label

Step 1 Log in to ServiceStage and choose **Infrastructure** > **Cloud Service Engines**.

Step 2 Click **Console** of a microservice engine.

Step 3 Choose **Service Catalog**.

Step 4 Click a microservice. On the displayed page, click **Manage Tag** next to **Label**.

Step 5 Click **Add Label**, and enter **Key** and **Value**.

Step 6 Click **OK** to save the settings.

----End

Modifying Microservice 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.
Test	The instance is in the internal joint commissioning state and does not provide services.

Step 1 Log in to ServiceStage and choose **Infrastructure** > **Cloud Service Engines**.

Step 2 Click **Console** of a microservice engine.

Step 3 Go to the microservice console.

Step 4 Choose **Service Catalog** > **Instance List**.

Step 5 Select the target instance and change the microservice instance status.

- Offline
  In the **Operation** column, click **Offline**.
- Online
  In the **Operation** column, click **Online**.
- Out of Service
  In the **Operation** column, choose **More** > **Out of Service**.
- Test
  In the **Operation** column, choose **More** > **Test**.

----End

3.8.2 Installing VM Agent on a Single VM

To deploy a component to a virtual machine (VM), you need to install the agent. After the host is managed, the backend can communicate with the host.

For details about the VM agent status and description, see **Table 3-31**.

Table 3-31 VM agent status description

Agent Status	Description
Agent missing. Install it first.	The VM agent is not installed on the ECS node. You need to install the VM agent.

Agent Status	Description
a.b.c	The VM agent has been installed and is running properly. *a.b.c* indicates the version of the VM agent. Example: **1.3.15**.
Agent offline	The agent has been installed but is offline and cannot work properly.
ECSs in the CCE cluster are managed in the environment with the cluster and do not need to be managed separately.	The ECS node is a node in the CCE cluster. Therefore, you cannot install the VM agent on the ECS node.

This section describes how to install the VM agent on a single VM.

Procedure

Step 1 Log in to ServiceStage, and choose **Infrastructure** > **VMAgent Manager**.

Step 2 Locate the VM where the agent is to be installed and click **Install Agent**.

Step 3 Select an authorization mode.

Authorize the agent to use your authentication information to obtain the deployment, upgrade, start, and stop tasks of an application and execute the task.

You can use AK/SK to perform authorization.

Select **AKSK** for **Authorization Model** and enter the AK and SK. For details about how to obtain the AK/SK, see **How Do I Obtain an AK/SK Pair?**.

Step 4 Select **Add application access port automatically** based on service requirements.

Step 5 Copy the command automatically generated in the lower part of the window, that is, the agent installation command.

Example command for the **AKSK** model:

```
export AGENT_INSTALL_URL=https://${Region_Name}-servicestage-vmapp.obs.${Region_Name}.${Domain_Name}/vmapp/agent/agent-install.sh;if [ -f `which curl` ];then curl -# -O -k ${AGENT_INSTALL_URL};else wget --no-check-certificate ${AGENT_INSTALL_URL};fi;bash agent-install.sh ${AK}${SK} ${Project_ID} ${Version} ${Region_Name} ${Flag}
```

- In the preceding command, **AGENT_INSTALL_URL** indicates the installation address of the Agent.
- **${AK}/${SK}** indicates an access key.
- **${Region_Name}** indicates a region name.
- **${Domain_Name}** indicates the global domain name.
- **${Project_ID}** indicates a project ID. For details about how to obtain a project ID, see **How Do I Obtain a Project Name?**.
- **${Version}** is the version number. Use **latest** to automatically download the latest version.

- **${Flag}** is a Boolean value, indicating whether to automatically add the application access port. **true** indicates yes and **false** indicates no.

Step 6 Log in to the VM and run the installation command.

----End

4 FAQs

4.1 How Do I Obtain an AK/SK Pair?

NOTE

Log in to the ServiceStage console as the user with the required permissions.

Step 1 Log in to ServiceStage.

Step 2 Move the pointer to the username and choose **My Credentials** from the drop-down list.

Step 3 In the navigation pane, choose **Access Keys**.

Step 4 Click **Create Access Key**. After authentication, an AK/SK pair is created.

Step 5 Click **Download**.

Obtain the AK and SK information from the **credentials** file.

- The value of **Access Key Id** is the AK.
- The value of **Secret Access Key** is the SK.

NOTICE

- Each user can retain only two valid access keys.
- For security purposes, access keys are automatically downloaded only when they are generated for the first time and cannot be obtained from the console later. Keep the access key secure.

----End

4.2 What Should I Do If an Error Occurs When I Change the Name of a Project ?

For the professional edition of microservice engines, projects are used for resource isolation. When their names are changed, the changes need to be updated globally.

It means that you need to update all microservices that use the professional service center for registration and discovery in the project. Otherwise, the "Project id or name is not existed" error may occur.

You can perform the following operations to resolve this issue:

- If a microservice is deployed through ServiceStage, update the microservice components by referring to **Upgrading an Application Component** (you do not need to modify any parameter).
- If a microservice is not deployed through ServiceStage, change the name of the project in the microservice configuration to the actual value and update the microservice.
  - For details about the configuration items of a project, see **Table 4-1**.
  - For details about how to obtain the name of a project, see **How Do I Obtain a Project Name?**.

Table 4-1 Configuration items of project

Microservice Framework	Configuration Item	Configuration File
Java Chassis	servicecomb.credentials.project	microservice.yml
Spring Cloud Huawei	spring.cloud.servicecomb.credentials.project	application.yml
Dubbo ServiceComb	dubbo.servicecomb.credentials.project	dubbo.properties
Go Chassis	servicecomb.credentials.project	chassis.yaml or auth.yaml

4.3 What Are the Differences Between Microservices and Common Applications?

Microservice is an architecture mode, in which, a monolithic application is divided into multiple parts for development. Therefore, an application using the microservice architecture is essentially a distributed application.

Applications built based on the microservice architecture can make services change faster and the overall system reliability higher.

Service Type	Microservice	Common Application
Developer	The workload of a microservice is light. A Two Pizza team can rewrite all the code of a microservice in two weeks. This can be used as a symbol of microservices. When developing a microservice, its APIs need to be interconnected with other microservices. Therefore, the API definition-based development mode is highly recommended.	Complex logic, coupled modules, bloated code, difficult modification, and low version iteration efficiency.
Confirming Deployment	An application consisting of multiple microservices is complex. Deployment orchestration is required when the application is deployed.	Possible large, and long build and deployment time, which is not conducive to frequent deployment and hinders continuous delivery. This problem is especially serious in mobile application development.
Operation	Governance takes the notice in addition to metrics monitoring and log collection. The core concept of microservice governance is to maintain the system performance through modifications while the system is running.	It takes a long time to rectify common online problems. To rectify any online problem, the entire application system must be upgraded.

4.4 How Do I View the Causes for 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 and choose **Application Management** > **Application List**.

Step 2 Click the application name. The **Overview** page is displayed.

Step 3 On the **Environment View** tab, select an environment to view the application components that have been deployed in the environment.

Step 4 Click the name of a component that is not ready. The **Overview** page of the component instance is displayed.

Step 5 Choose **Instance** and click before the instance name.

Step 6 Click the **Event** tab to view abnormal events.

For example, if the event type is **Warning**, check the **Description** column to find the cause of the application failure.

----End

4.5 What Should I Do If a VM Component Fails to Be Deployed or Updated?

If a component fails to be deployed or upgraded on a VM, perform the following steps to locate and rectify the fault.

Step 1 Log in to ServiceStage and choose **Application Management** > **Application List**.

Step 2 Click the application name. The **Overview** page is displayed.

Step 3 On the **Environment View** tab, select an environment to view the application components that have been deployed in the environment.

Step 4 Click the name of a component that is not ready. On the **Overview** page of the component instance that is displayed, obtain the name of the component instance.

Step 5 Choose **Update**. On the **Update** page, obtain the version number of the component instance.

Step 6 Choose **Instance** and click before the instance which fails to be started or installed.

Step 7 Click the **Events** tab to view the error information. If the system displays a message indicating that the port is occupied, you can deploy a port again.

Step 8 If the log information obtained in **Step 6** is not clear, obtain the logs of the component instance from the ECS.

1. Click the node where the instance resides to go to the host details page. If an EIP has been bound to the node, click the **EIPs** tab to obtain the EIP of the node and use the SSH client tool to connect to the node. Alternatively, click **Remote Login** in the upper-left corner to log in to the node and run the following command:

   ```
   cd /var/log/application/{component_instance_name}/{version}
   ```

   Replace **{component_instance_name}** with the component instance name obtained in **Step 4** and **{version}** with the version number obtained in **Step 5**.

   Run the **ls** command, and then run the **cd** command to access the sub-directory displayed after the **ls** command is run.

2. Obtain the detailed error logs from the sub-directory.

   If the component is a Tomcat8 component, view the **catalina.out** log to check the errors reported during Tomcat startup.

For other types of applications, view the **start_app.log** file to check the errors reported during the startup.

----End

4.6 What Are the Constraints on Packaging a Node.js 8 Software Package?

ServiceStage supports the installation of Node.js 8 software packages (**.zip** format) on virtual machines (VMs). You only need to upload and deploy the **.zip** package of the service. ServiceStage automatically installs the basic software for running Node.js 8 on the VM and automatically runs the **npm run** command to start the Node.js 8 component.

The **.zip** package cannot contain the **start.sh**, **stop.sh**, **install.sh**, **pre-install.sh**, **uninstall.sh**, **appspec.node.yml**, or **agent_template.yml** file. Otherwise, the files in the **.zip** package may conflict with the files in the basic software package.

4.7 What Should I Do If the Agent Fails to Be Installed?

Symptom

The system displays the following message, indicating that the agent installation fails.

```
auth token valid fail: User does not have valid roles to write!
```

Solution

Step 1 Obtain the AK/SK pair. For details, see **How Do I Obtain an AK/SK Pair?**.

Step 2 Check whether the AK/SK pair in the installation command is the same as that obtained in **Step 1**.

----End

4.8 What Should I Do If the Agent Is Offline?

Obtain a new AK/SK pair, update the AK/SK pair configured for the VM agent, and restart the VM agent.

Perform the following steps:

Step 1 Log in to the ECS node using VNC on the ECS console, or using the SSH client after binding an EIP to the ECS node.

Step 2 Run the **cd /opt/servicestage-agent/** command, and edit the **servicestage-agent.conf** file to change the AK/SK pair in the file, and save and exit the file.

Step 3 Run the following commands to restart the agent. Replace *x.x.x* with the latest version of **servicestage-agent** in the actual environment.

```
cd /opt/servicestage-agent/servicestage-agent-x.x.x
su agent
./servicestage-agent.sh restart
```

----End

4.9 Which Directories Do I Use to Write Files for VM-based Application Components?

For such components, only their running directory is available for writing files. Such files include log files or zip packages.

This directory is the **/opt/application/***${appName}*/*${appVersion}*/*${instanceId}* directory on the application ECS, where

- *${appName}* indicates the component instance name.
- *${appVersion}* indicates the version number of the component instance.
- *${instanceId}* indicates the instance ID.

This rule takes effect only for new and upgraded component instances. Directories of component instances that have been deployed retain the original permissions.

4.10 What Should I Do If "host status is not active" Is Reported When a VM-based Component Fails to Be Deleted?

Symptom

The component deployed on a VM fails to be deleted. Task details show the error information:

```
{
    "statusCode": 400,
    "jsonBody": {
        "error_code": "SVCSTG.VMAPP.4001020",
        "error_msg": "4001020",
        "error_detail": "host status is not active: abb3d0a4-f715-4932-
b7ec-6dd917f65778,4f68e35b-6e08-48d0-bd3a-1151be19efa5"
    }
}
```

where

- The error code is **SVCSTG.VMAPP.4001020**.
- The detailed error information is: host status is not active: abb3d0a4-f715-4932-b7ec-6dd917f65778. **abb3d0a4-f715-4932-b7ec-6dd917f65778** and **4f68e35b-6e08-48d0-bd3a-1151be19efc6** indicate the IDs of the two ECSs where components are deployed.

Solution

Step 1 Log in to the ECS console and click **Elastic Cloud Server**.

Step 2 In the ECS list, search the ECS IDs in the error information to find the ECSs where the components are deployed.

Step 3 Check whether each of their status is **Running**.

- If yes, go to **Step 2** and search for the next ECS.
- If no, go to **Step 4**.

Step 4 Either restore the status or delete the ECS.

- To restore the ECS status to **Running**: In the **Operation** column, choose **More** > **Start** or **More** > **Restart**.
- To delete an ECS you no longer use: In the **Operation** column, choose **More** > **Delete**.

Step 5 After you have performed **Step 2** to **Step 4** on all ECSs displayed in the error information, delete the components on the ServiceStage page again.

----End

4.11 How Do I Handle Docker Application Dependency?

Symptom

If I want to Run a node program in the microservice docker. The program depends on node-gyp. How do I install the dependency before compiling the program?

NOTE

The environment does not have the permission to access the external network. Therefore, the dependency cannot be obtained from the external network. You need to compile the node program in advance.

Solution

Step 1 Compile your project.

Step 2 Compress the compiled project into a .zip package.

Step 3 Upload the package to OBS and use the application management function of ServiceStage.

NOTE

When creating a component, select Node.js 8 during running.

----End

4.12 What Should I Do If Docker Client Fails to Push Images?

Symptom

The logged-in Docker client is used to upload an image package. For example, the following command is executed:

docker push xxx/*test*/*busybox*:latest

NOTE

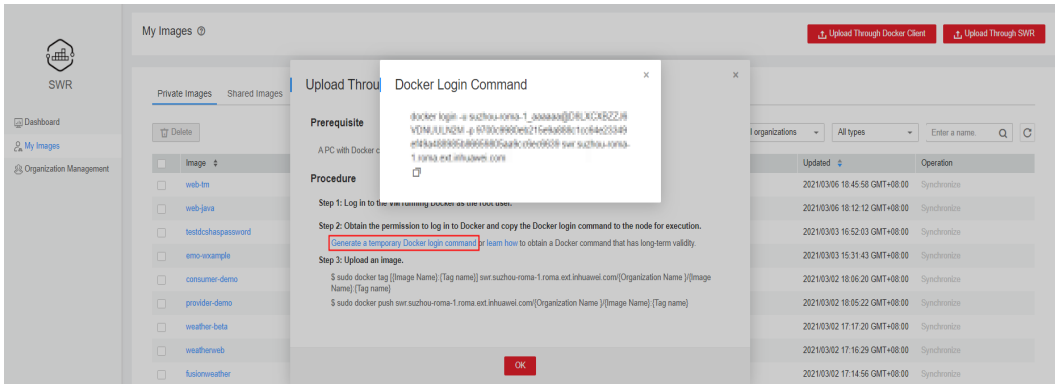
- ***xxx*** indicates the address of the repository to which the tenant or user is to upload the image.
- **test** indicates the organization.
- **busybox** is the image name.
- **latest** indicates the latest image version.

However, the upload fails, and the following information is displayed on the Docker client:

```
unauthorized: authentication required
```

Solution

Step 1 On the **SWR** page, choose **My Images** > **Upload Through Docker Client** > **Generate a temporary Docker login command**, and copy the command.



Step 2 Click **Organization Management** and create an organization, for example, **test**.

Step 3 Log in to a data plane node and run the command obtained in **Step 1**.

Step 4 Run the following command to upload an image to an organization on which the current tenant or user has the operation permission. For example, upload an image named **busybox** to the organization **test** created in **Step 2**.

docker push *xxx/test*/**busybox:latest**

NOTE

- ***xxx*** indicates the address of the repository to which the tenant or user is to upload the image. It is the tail of the command copied in **Step 1**, for example, **swr.{regionID}.{external_global_domain_name}**.
- **test** is the organization on which the tenant or user has the operation permission.
- **latest** indicates the latest image version.

Step 5 After the image is uploaded, the following information is displayed.

```
The push refers to a repository [xxx/test/busybox]
6a749002dd6a: Pushed
latest: digest: sha256:ecb3f3e96e003af6e02f0f47ac4d25a3b0585db54de0a82bb070f8cb78a79bc7 size: 527
```

If an exception occurs, contact technical support.

----End

4.13 How Do I Obtain a Project Name?

Step 1 Log in to ServiceStage.

Step 2 Click the username and select **My Credentials** from the drop-down list.

Step 3 Choose **API Credentials** and view **Project Name** and **Project ID** in **Projects**.

----End

4.14 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 registry center address of the microservice is the IPv4 address of the microservice engine registry center. The microservice can be successfully registered and started.

If the registry center address of the microservice 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 **Modifying an Environment**.

Step 2 Deploy the application again. For details, see **Deploying a Component**.

----End

4.15 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, or modify specifications, 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.

4.16 What Should I Do I Get an ECS Error When Deploying VM-based Components?

Symptom

During application component deployment on ServiceStage, an error is reported on the deployment task details page:

```
{
   "Message": "ECF0006",
   "Detail": "io.vertx.core.VertxException:Connection was closed"
}
```

Solution

The connection was closed before the request was sent, possibly due to network fluctuations. Try waiting for a while and redeploying the connection.

If redeployment fails, contact technical support.

4.17 What Should I Do If an ECS Error Occurs During VM-based Component Deployment?

Symptom

During basic configuration of component deployment, selecting **VM** for **Deployment System** and **Elastic cloud server** for **Resource Type** may cause the ECS service to be unavailable.

For example, calling the ECS interface times out during component deployment, and the following error information is displayed in log details:

```
{
   "statusCode": 500,
   "jsonBody": {
      "error_code": "SVCSTG.VMAPP.5001002",
      "error_msg": "read ECS host 471ff77a-c827-41d5-941d-4fea8aaa56ef fail TIMEOUT."
   }
}
```

Solution

Step 1 Redeploy the component and check whether the deployment is successful.

- If yes, no further action is required.
- If no, go to **Step 2**.

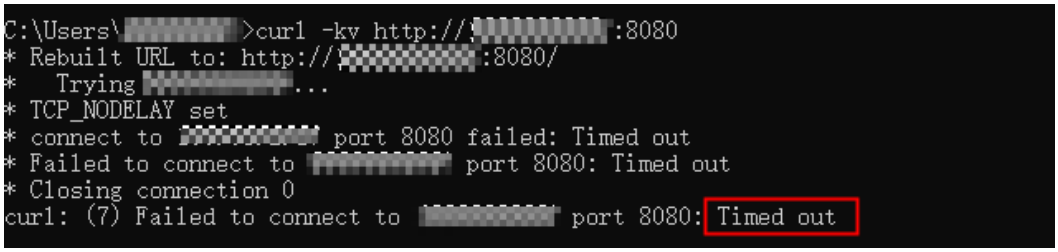
Step 2 Contact technical support.

----End

4.18 What Should I Do If I Cannot Access the Port During VM-based Deployment?

Symptom

During basic configuration of component deployment, selecting **VM** for **Deployment System** and **Elastic cloud server** for **Resource Type** may prevent access to the container port. When the **curl -kv http://***${IP address of the ECS node where the application component is deployed}*:*${Container port}* command is run to access the container port, the system displays a message indicating that the access timed out.



Solution

Step 1 Log in to the ECS console and click **Elastic Cloud Server**.

Step 2 In the ECS list, find the target ECS for deploying the component and click it to open its details page.

Step 3 On the **Security Groups** tab, click **Change Security Group** and check whether the port rule exists in an existing group.

- If yes, select the security group.
- If no, click **Create Security Group** to create one and configure its rules. Next, select the created group.

Step 4 Run the **curl -kv http://***${IP address of the ECS node where the application component is deployed}*:*${Container port}* command again to access the container port and check whether the fault is rectified.

----End

4.19 What Should I Do If the Microservice Application Name Is Different from the Component Application Name?

Symptom

After a microservice component is created and deployed on ServiceStage, when you choose **Service Catalog** > **Microservice List** on the engine console where the microservice component is deployed to view the microservice application name (as shown in **Figure 4-2**), this microservice application name may be different from that of the application where the microservice component is located (as shown in **Figure 4-1**).

Figure 4-1 Application name on the Application Management page

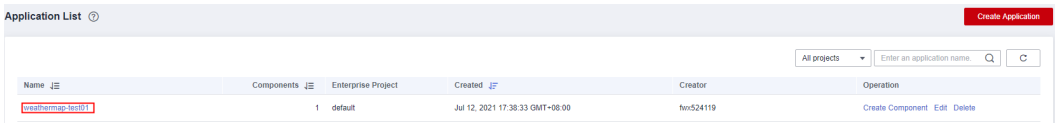
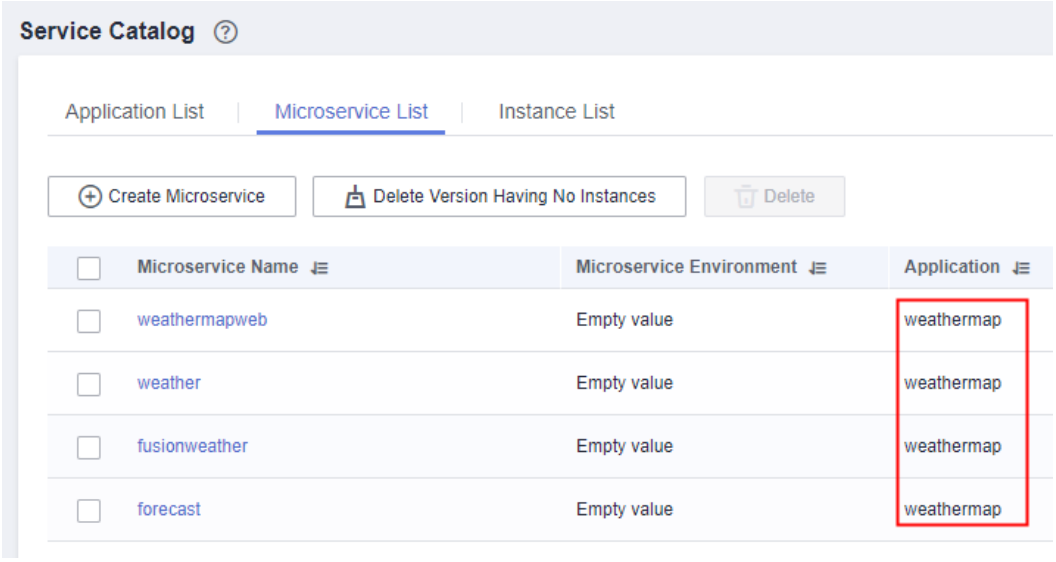


Figure 4-2 Application name on the Service Catalog page



Solution

Step 1 Log in to ServiceStage and choose **Application Management** > **Application List**.

Step 2 Click the application where the faulty microservice component is located to go to its details page.

Step 3 Click **Environment Variables** and select a created environment from the **Environment** drop-down list.

Step 4 Choose **Environment Variable** > **Add Environment Variable** to configure global environment variables.

- Spring Cloud framework: Enter a key and value by referring to **Table 4-2**.

  Table 4-2 Global environment variables of the Spring Cloud framework

Key	Value
spring_cloud_servicecomb_discovery_appName	Enter the name of the application where the microservice component is located.

- Java Chassis framework: Enter a key and value by referring to **Table 4-3**.

  Table 4-3 Global environment variables of the Java Chassis framework

Key	Value
servicecomb_service_application	Enter the name of the application where the microservice component is located.

Step 5 Deploy the created component for the application again. For details, see **Deploying an Application Component**.

Wait until the component deployment is complete and the component status changes to **Running**.

Step 6 Choose **Infrastructure** > **Cloud Service Engines**.

Step 7 Select the microservice engine where the microservice component is deployed and click **Console**.

Step 8 Choose **Service Catalog** > **Microservice List**. The application name on the displayed page is the same as the application name on the **Application Management** page.

----End

4.20 Why the Microservice Name Is Different from the Component Name?

Symptom

After a microservice component is created and deployed on the ServiceStage console, choose **Service Catalog** > **Microservice List** on the microservice engine console where the microservice component is deployed to view the microservice name. The microservice name may be different from the microservice component name.

Cause Analysis

The microservice name displayed in the microservice engine is configured in the microservice configuration file during microservice development, which is related to service calling. The component name is user-defined during microservice component creation and deployment on the ServiceStage console.

This difference does not affect services, and no action is required.

4.21 Failed to Restore Data of an Exclusive Microservice Engine

Symptom

Data of an exclusive microservice engine fails to be restored.

Solution

If data restoration fails, the engine may be unavailable, causing service faults. Contact customer service to rectify the fault by following the troubleshooting procedure and restore the engine functions first.