ServiceStage

Getting Started

Issue	01
Date	2024-09-27

HUAWEI CLOUD COMPUTING TECHNOLOGIES CO., LTD.

Huawei Cloud Computing Technologies Co., Ltd.

Address:	Huawei Cloud Data Center Jiaoxinggong Road Qianzhong Avenue Gui'an New District Gui Zhou 550029 People's Republic of China
Website:	https://www.huaweicloud.com/intl/en-us/

Contents

1 Using ServiceStage to Host Microservice Applications

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

ServiceStage provides the environment management function to manage compute resources, such as Cloud Container Engine (CCE), Elastic Cloud Server (ECS), network resources, such as Elastic Load Balance (ELB) and Elastic IP (EIP), and middleware, such as Distributed Cache Service (DCS), Relational Database Service (RDS), and Cloud Service Engine (CSE) in the same VPC. In this case, when you select an environment during application deployment, the contained resources are automatically loaded.

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

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

You can perform O&M operations, such as starting, stopping, upgrading, rolling back, and scaling application components, viewing logs, viewing events, setting access modes, and setting threshold alarms.

This example describes how to quickly create a microservice application based on the source code and ServiceComb (SpringMVC) framework to experience the ServiceStage functions.

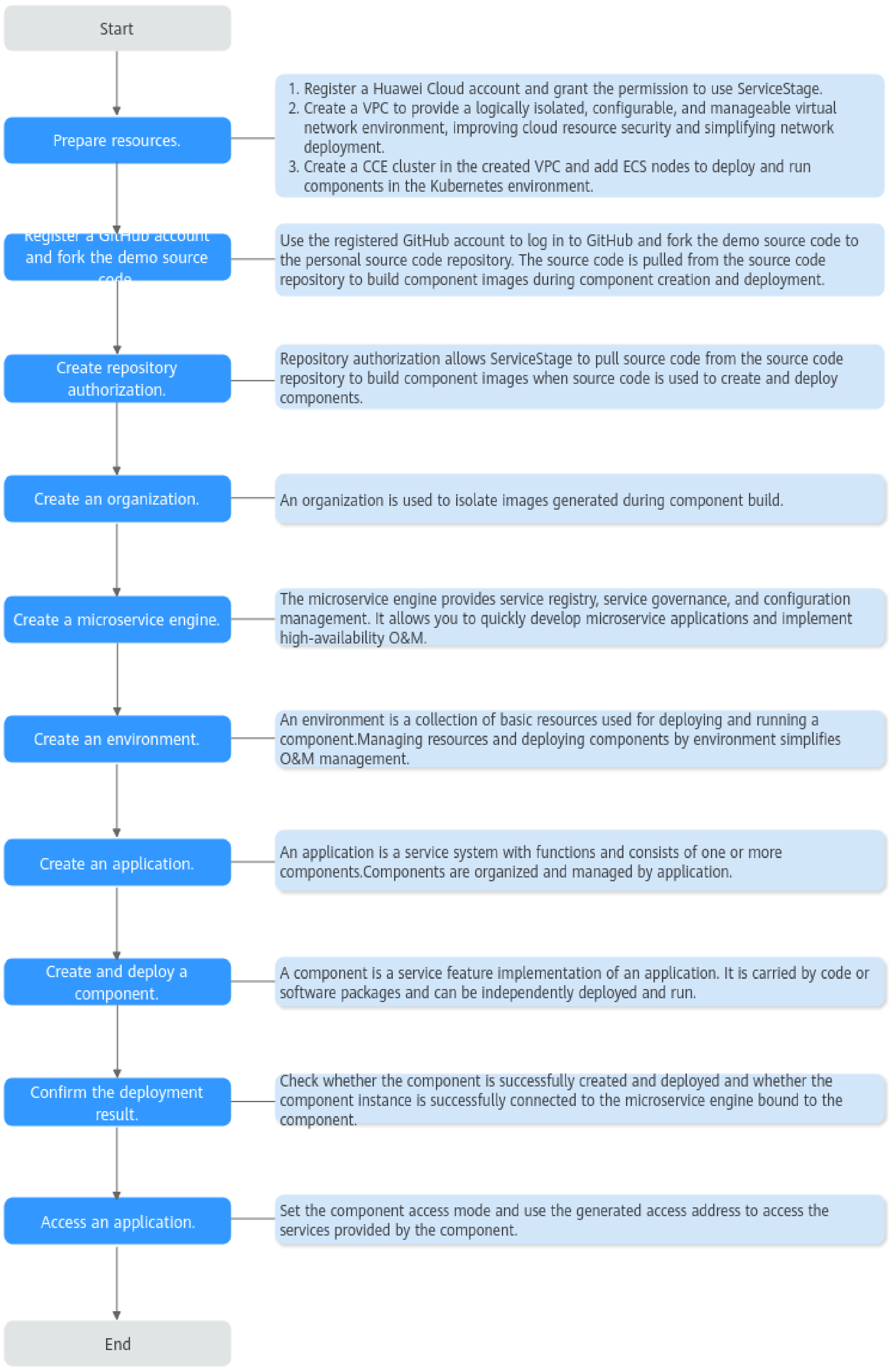
NOTE

ServiceStage provides demos in different languages based on GitHub. Experience the source code deployment function of the demo in a specific language on ServiceStage. For details, see **How Do I Use the ServiceStage Source Code Deployment Function?**

Procedure

Figure 1-1 shows the process of using ServiceStage to host microservice applications.

Figure 1-1 ServiceStage process


Start
Prepare resources.
1. Register a Huawei Cloud account and grant the permission to use ServiceStage.
2. Create a VPC to provide a logically isolated, configurable, and manageable virtual network environment, improving cloud resource security and simplifying network deployment.
3. Create a CCE cluster in the created VPC and add ECS nodes to deploy and run components in the Kubernetes environment.
Register a GitHub account and fork the demo source code
Use the registered GitHub account to log in to GitHub and fork the demo source code to the personal source code repository. The source code is pulled from the source code repository to build component images during component creation and deployment.
Create repository authorization.
Repository authorization allows ServiceStage to pull source code from the source code repository to build component images when source code is used to create and deploy components.
Create an organization.
An organization is used to isolate images generated during component build.
Create a microservice engine.
The microservice engine provides service registry, service governance, and configuration management. It allows you to quickly develop microservice applications and implement high-availability O&M.
Create an environment.
An environment is a collection of basic resources used for deploying and running a component.Managing resources and deploying components by environment simplifies O&M management.
Create an application.
An application is a service system with functions and consists of one or more components.Components are organized and managed by application.
Create and deploy a component.
A component is a service feature implementation of an application. It is carried by code or software packages and can be independently deployed and run.
Confirm the deployment result.
Check whether the component is successfully created and deployed and whether the component instance is successfully connected to the microservice engine bound to the component.
Access an application.
Set the component access mode and use the generated access address to access the services provided by the component.
End

Prerequisites

1. You have **registered a Huawei account and enabled Huawei Cloud services**.
2. The login account has the permission to use ServiceStage. For details, see **Creating a User and Granting Permissions**.
3. You have created a VPC. For details, see **Creating a VPC**.
4. You have created a CCE cluster. For details, see **Buying a CCE Cluster**.
   - The VPC to which the CCE cluster belongs is the VPC created in **3**.
   - The cluster contains at least one ECS node and is bound to an elastic IP address. (To facilitate subsequent operations, you are advised to select a node of 4 vCPUs and 8 GB memory.) For details, see **Creating a Node**.
   - The CCE cluster cannot be bound to other environments.

Registering a GitHub Account and Forking the Demo Source Code

Step 1 **Register a GitHub account**.

Step 2 **Log in to GitHub**.

Step 3 Navigate to the **demo source code repository**.

Step 4 Fork the demo source code repository to your account. For details, see **Forking a repository**.

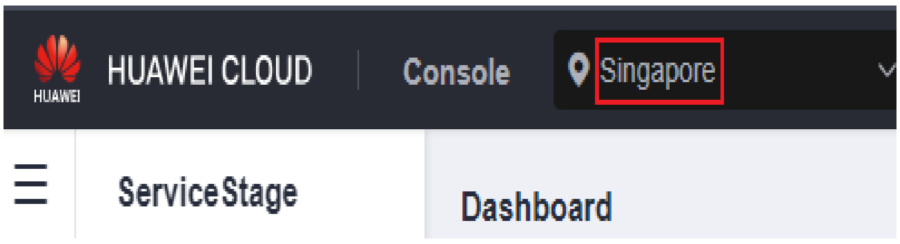
----End

Creating Repository Authorization

Step 1 **Log in to ServiceStage** using a registered Huawei Cloud account.

Step 2 In the region list, select a region where ServiceStage is to be used, for example, AP-Singapore.

Figure 1-2 Logging in to ServiceStage



Step 3 Choose **Continuous Delivery** > **Repository Authorization**.

Step 4 Click **Create Authorization**. The page for creating repository authorization is displayed.

Step 5 Retain the default authorization name.

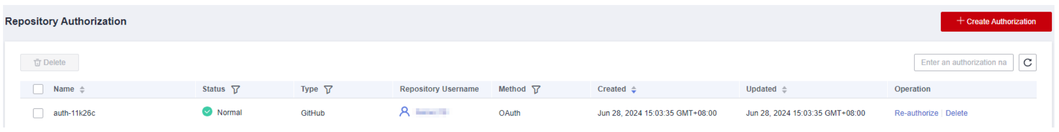
Step 6 Set **Repository Authorization**.

1. Select **GitHub**.

2. Select **OAuth**.
3. Click **Authenticate with OAuth**.
4. After reading the service statement, select **I understand that the source code building function of the ServiceStage service collects the information above and agree to authorize the collection and use of the information**.
5. Click **OK**.
6. Enter your GitHub account and password to log in to GitHub for identity authentication. Wait until the authorization is complete.

Step 7 Click **OK**. You can view the created authorization in the repository authorization list.

Figure 1-3 Authorizing a Repository



	Name	Status	Type	Repository Username	Method	Created	Updated	Operation
☐	auth-11k26c	Normal	GitHub	[redacted]	OAuth	Jun 28, 2024 15:03:35 GMT+08:00	Jun 28, 2024 15:03:35 GMT+08:00	Re-authorize | Delete

----End

Creating an Organization

Step 1 Choose **Deployment Source Management** > **Organization Management**.

Step 2 Click **Create Organization**. On the displayed page, specify **Organization Name**. For example, **ss-org**.

Step 3 Click **OK**.

----End

Creating an Exclusive Microservice Engine

NOTICE

If the engine is created using an account with the minimum permission for creating engines, for example, **cse:engine:create** in the **Fine-grained Permissions**, the default VPC security group cse-engine-default-sg needs to be preset by the primary account. For details, see **Creating a Microservice Engine**.

Step 1 Choose **Cloud Service Engine** > **Engines**.

Step 2 Click **Buy Exclusive Microservice Engine** and set the parameters by referring to the following table.

Parameter	Description
Billing Mode	Select **Pay-per-use**.

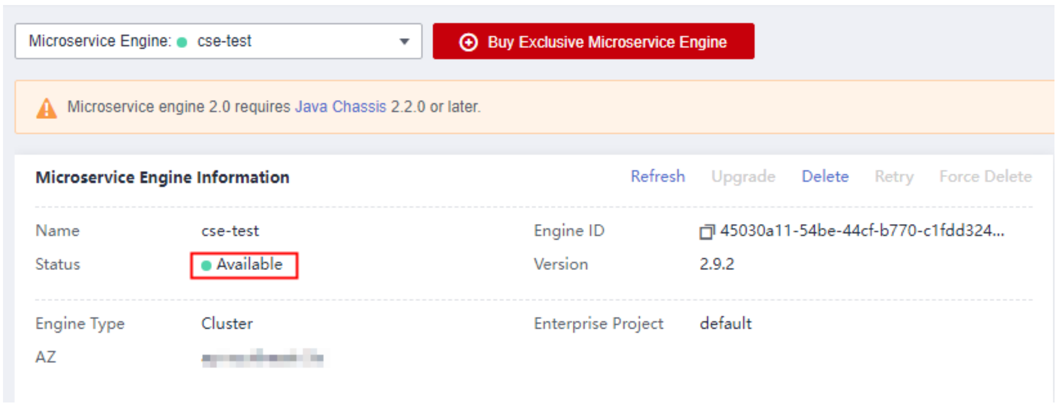
Parameter	Description
Enterprise Project	**default** is selected by default. Enterprise projects let you manage cloud resources and users by project. It is available after you **enable the enterprise project function**.
Specifications	Retain the default value.
Engine Type	Select **Cluster**.
Name	Enter a microservice engine name, for example, **cse-test**.
AZ	Select an AZ for the microservice engine.
Network	Select a VPC and its subnets created in **Prerequisites** to provision logically isolated, configurable, and manageable virtual networks for your engine.
Security Authentication	Select **Disable security authentication**.

Step 3 Click **Buy Now**.

Step 4 Confirm the parameters. Click **Submit**.

It takes about 31 minutes to create an engine. After the microservice engine is created, its status is **Available**.

Figure 1-4 Creating a microservice engine


Microservice Engine: ● cse-test
Buy Exclusive Microservice Engine

Microservice engine 2.0 requires Java Chassis 2.2.0 or later.

Microservice Engine Information

Refresh
Upgrade
Delete
Retry
Force Delete

Name	cse-test	Engine ID	45030a11-54be-44cf-b770-c1fdd324...
Status	Available	Version	2.9.2
Engine Type	Cluster	Enterprise Project	default
AZ			

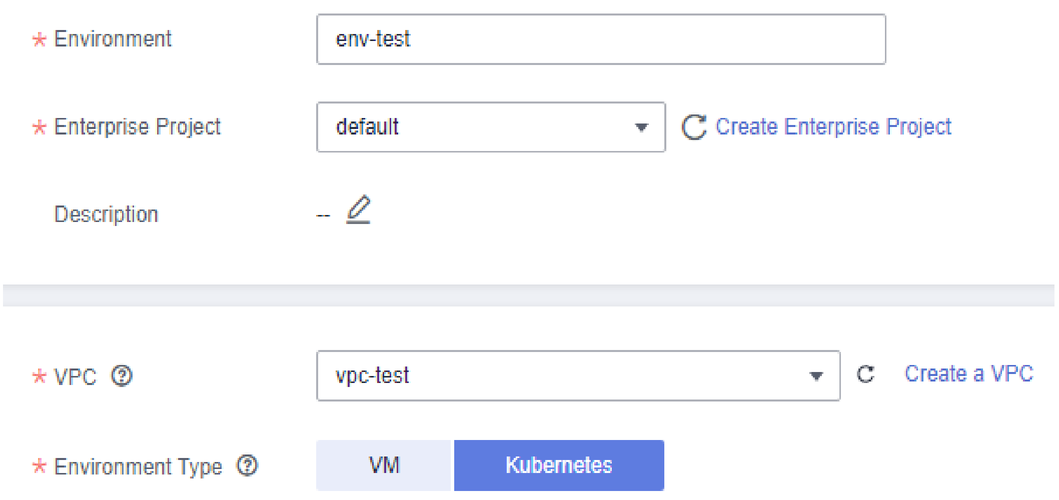
----End

Creating an Environment

Step 1 Choose **Environment Management** > **Create Environment** and set the environment information by referring to the following table.

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

Figure 1-5 Creating an environment



Step 2 Click **Create Now**.

Step 3 In the **Resource** area, choose **Cloud Container Engine** from **Compute** and click **Bind now**.

Step 4 In the dialog box that is displayed, select the CCE cluster created in **Prerequisites** and click **OK**.

Step 5 In the **Resources** area, choose **ServiceComb Engines** from **Middleware** and click **Manage Resource**.

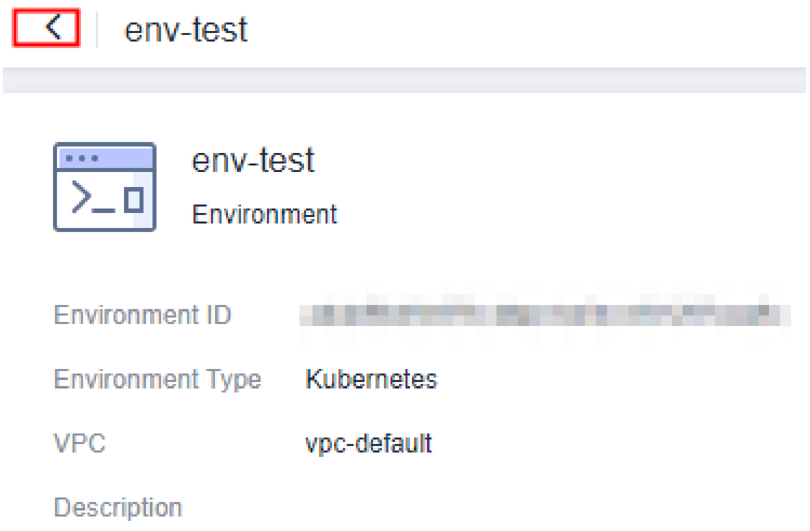
Step 6 In the dialog box that is displayed, select the ServiceComb engine created in **Creating an Exclusive Microservice Engine** and click **OK**.

----End

Creating an Application

Step 1 Click < in the upper left corner to return to the **Environment Management** page.

Figure 1-6 Backing to Environment Management


env-test

env-test
Environment

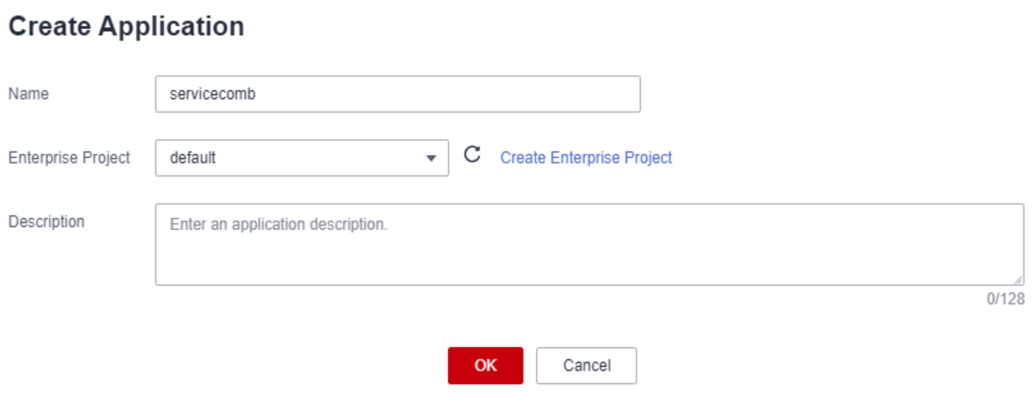
Environment ID
Environment Type Kubernetes
VPC vpc-default
Description

Step 2 Choose **Application Management** > **Create Application** and configure the application by referring to the following table.

Parameter	Description
Name	Enter an application name, for example, **servicecomb**.
Enterprise Project	**default** is selected by default. Enterprise projects let you manage cloud resources and users by project. It is available after you **enable the enterprise project function**.

Step 3 Click **OK**.

Figure 1-7 Creating an application



Create Application

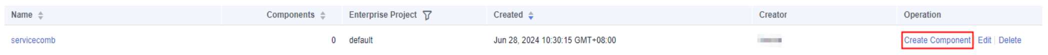
Name	servicecomb
Enterprise Project	default
Description	Enter an application description. 0/128

----End

Creating and Deploying a Component

Step 1 Select the application (for example, **servicecomb**) created in **Creating an Application** and click **Create Component** in the **Operation** column.

Figure 1-8 Creating a component

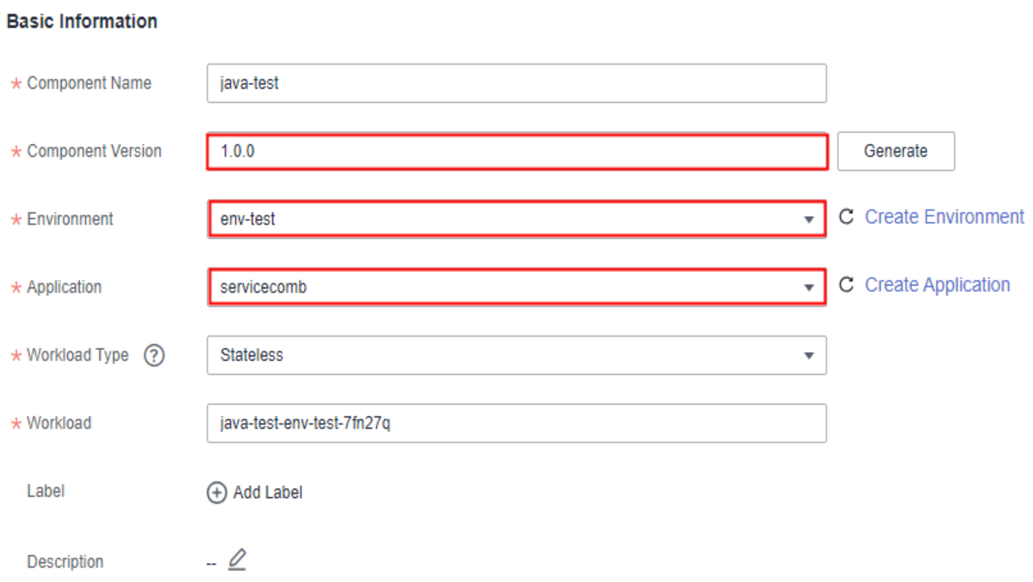


Name	Components	Enterprise Project	Created	Creator	Operation
servicecomb	0	default	Jun 28, 2024 10:30:15 GMT+08:00	[redacted]	Create Component Edit Delete

Step 2 In the **Basic Information** area, set the following mandatory parameters. Retain the default values for other parameters.

Parameter	Description
Component Name	Enter a component name, for example, **java-test**.
Component Version	Enter **1.0.0**.
Environment	Select the environment created in **Creating an Environment**, for example, **env-test**.
Application	Select the application created in **Creating an Application**, for example, **servicecomb**.

Figure 1-9 Setting the basic component information



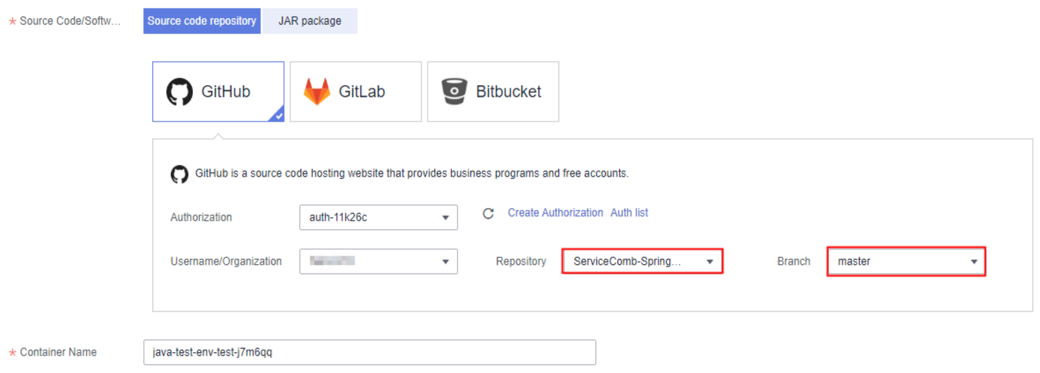
Basic Information

* Component Name	java-test
* Component Version	1.0.0
* Environment	env-test
* Application	servicecomb
* Workload Type	Stateless
* Workload	java-test-env-test-7fn27q
Label	Add Label
Description	--

Step 3 In the **Component Package** area, set the following mandatory parameters. Retain the default values for other parameters.

1. **Stack**: Select **Java**.
2. **Source Code/Software Package**: Select **Source code repository**.
3. Select **GitHub**.
4. **Authorization**: Select the repository authorization created when **Creating Repository Authorization**.
5. **Username/Organization**: Select the GitHub account created when **Registering a GitHub Account and Forking the Demo Source Code**.
6. Repository: Select demo source code repository **ServiceComb-SpringMVC** forked when **Registering a GitHub Account and Forking the Demo Source Code**.
7. **Branch**: Select **master**.

Figure 1-10 Setting the component source



* Source Code/Softw... : Source code repository

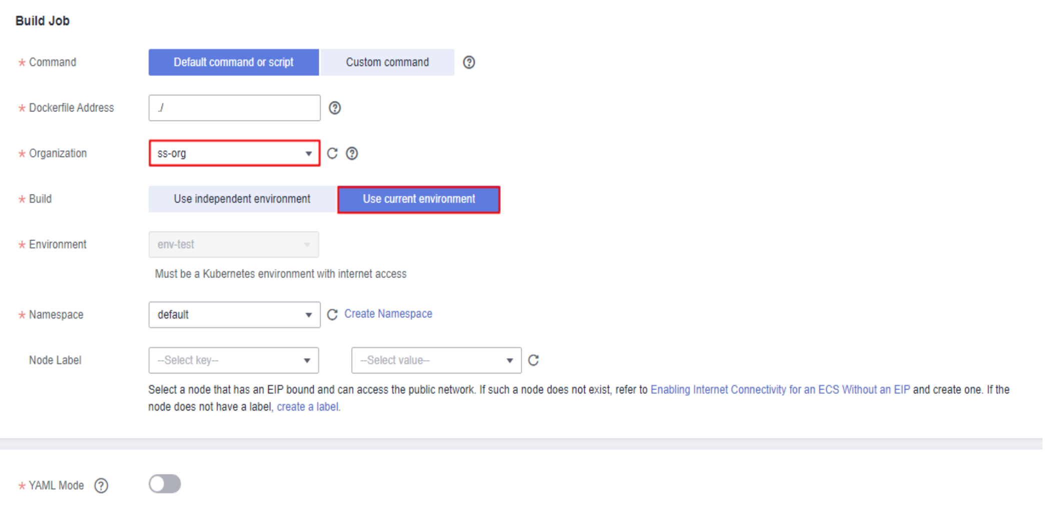
GitHub is a source code hosting website that provides business programs and free accounts.

Authorization	auth-11k26c
Username/Organization	[redacted]
Repository	ServiceComb-Spring...
Branch	master
* Container Name	java-test-env-test-j7m6qq

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

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

Figure 1-11 Configuring build parameters



Build Job

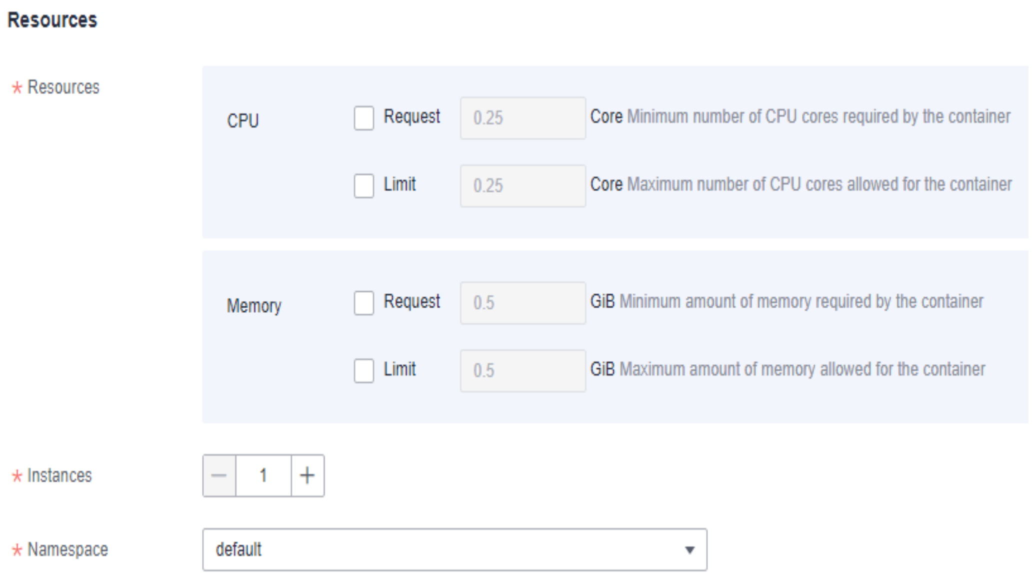
* Command	[✓ Default command or script] Custom command
* Dockerfile Address	./
* Organization	ss-org
* Build	Use independent environment [✓ Use current environment]
* Environment	env-test Must be a Kubernetes environment with internet access
* Namespace	default · Create Namespace
Node Label	--Select key-- --Select value-- Select a node that has an EIP bound and can access the public network. If such a node does not exist, refer to Enabling Internet Connectivity for an ECS Without an EIP and create one. If the node does not have a label, create a label.
* YAML Mode	

Step 5 Click **Next**.

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

Parameter	Description
Resources	Deselect **CPU** and **Memory**, indicating that the resource usage is not limited.
Instances	Set this parameter to **1**.

Figure 1-12 Setting component instance resources



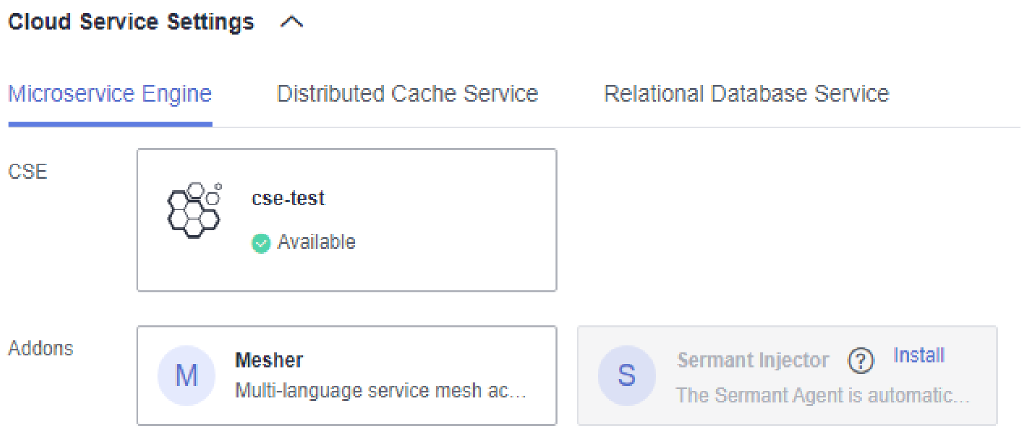
Resources

* Resources	CPU	☐ Request	0.25	Core Minimum number of CPU cores required by the container
		☐ Limit	0.25	Core Maximum number of CPU cores allowed for the container
	Memory	☐ Request	0.5	GiB Minimum amount of memory required by the container
		☐ Limit	0.5	GiB Maximum amount of memory allowed for the container
* Instances	1			
* Namespace	default			

Step 7 Bind a microservice engine.

1. Choose **Cloud Service Settings** > **Microservice Engine**.
2. Click **Bind Microservice Engine**.
3. Select the managed exclusive ServiceComb engine in the current environment.
4. Click **OK**.

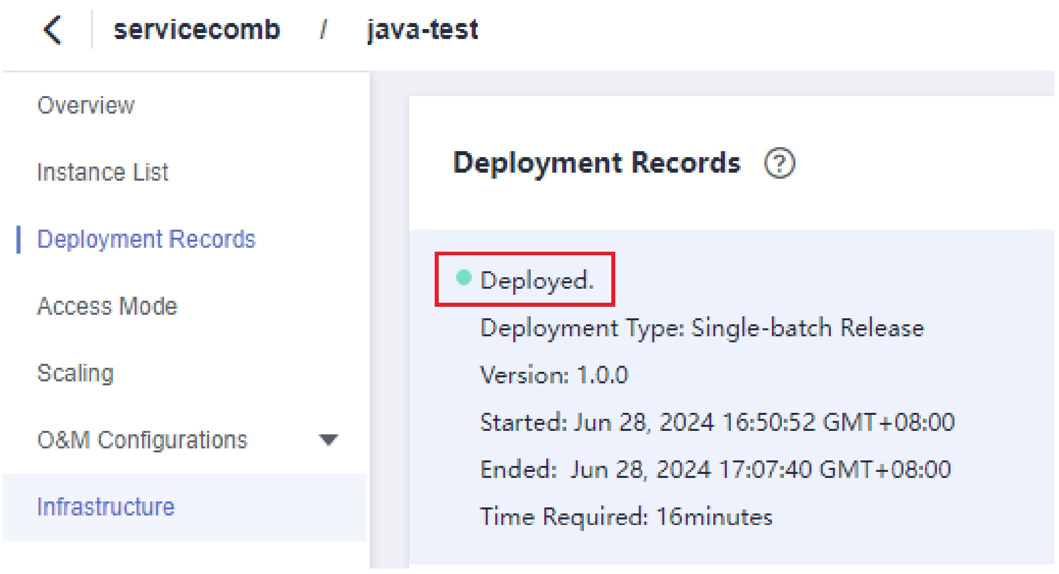
Figure 1-13 Binding a microservice engine



Cloud Service Settings
Microservice Engine
Distributed Cache Service
Relational Database Service
CSE
cse-test
Available
Addons
Mesher
Multi-language service mesh ac...
Sermant Injector
Install
The Sermant Agent is automatic...

Step 8 Click **Create and Deploy** and wait until the component is deployed.

Figure 1-14 The component is deployed.



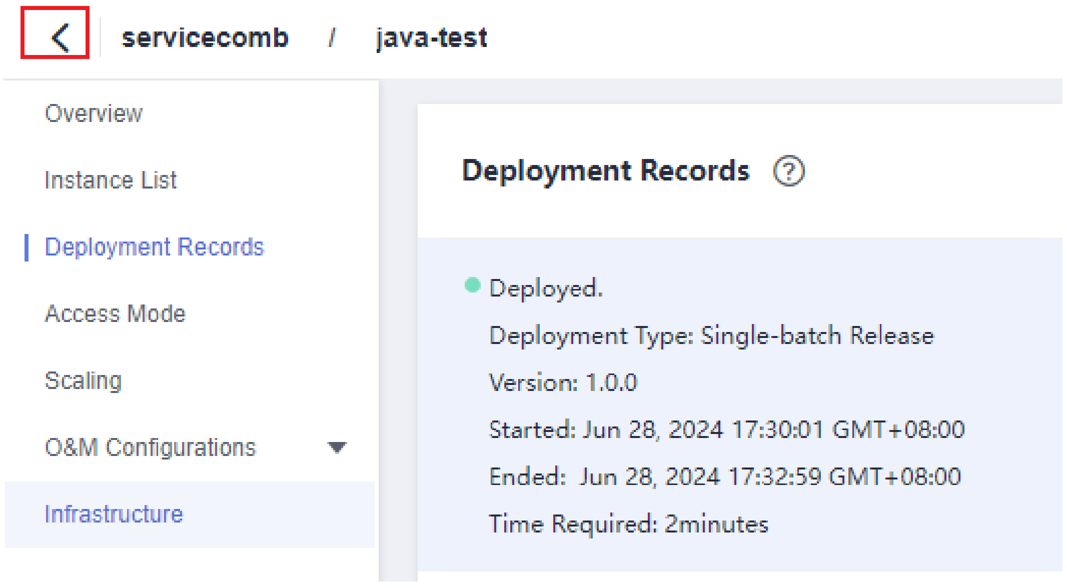
Deployed.
Deployment Type: Single-batch Release
Version: 1.0.0
Started: Jun 28, 2024 16:50:52 GMT+08:00
Ended: Jun 28, 2024 17:07:40 GMT+08:00
Time Required: 16minutes

----End

Confirming the Deployment Result

Step 1 Click < in the upper left corner to return to the **Component Management** page.

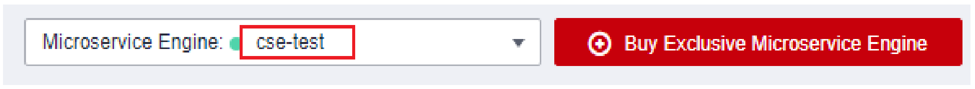
Figure 1-15 Backing to Component Management



Deployed.
Deployment Type: Single-batch Release
Version: 1.0.0
Started: Jun 28, 2024 17:30:01 GMT+08:00
Ended: Jun 28, 2024 17:32:59 GMT+08:00
Time Required: 2minutes

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

Step 3 Select a cloud service engine managed in **Creating an Environment** from the **Microservice Engine** drop-down list.

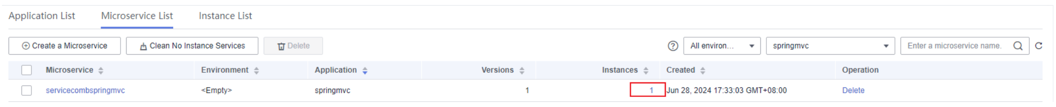


Microservice Engine: cse-test

Step 4 Select **springmvc** from **All** applications.

If the **servicecombspringmvc** microservice exists and the number of microservice instances is **1**, the component instance has been connected to the microservice engine.

Figure 1-16 Confirming the deployment result



	Microservice	Environment	Application	Versions	Instances	Created	Operation
☐	servicecombspringmvc	<Empty>	springmvc	1	1	Jun 28, 2024 17:33:03 GMT+08:00	Delete

----End

Accessing an Application

Step 1 Choose **Application Management**. The application list is displayed.

Step 2 Click the application created in **Creating an Application** (for example, **servicecomb**). The **Overview** page is displayed.

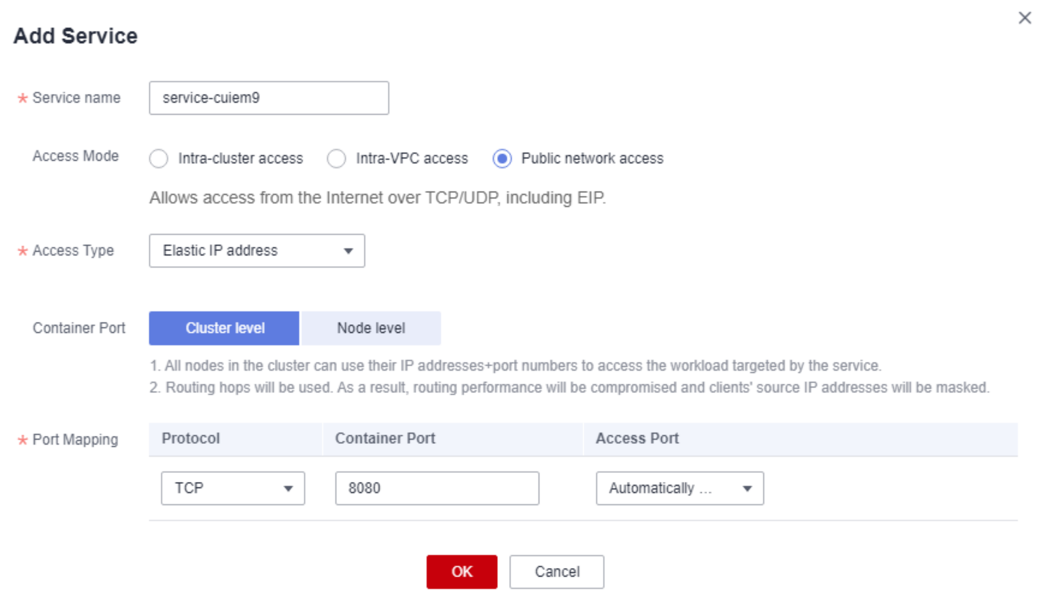
Step 3 On the **Component List** tab, click the component created in **Creating and Deploying a Component** (for example, **java-test**). The **Overview** page is displayed.

Step 4 Click **Access Mode**.

Step 5 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 setting.
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**.

Figure 1-17 Setting the access mode



Add Service

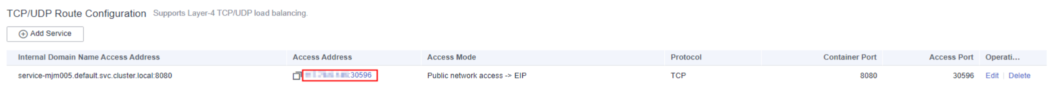
* Service name	service-cuiem9
Access Mode	○ Intra-cluster access ○ Intra-VPC access [✓ Public network access]
	Allows access from the Internet over TCP/UDP, including EIP.
* Access Type	Elastic IP address
Container Port	[✓ Cluster level] Node level
	1. All nodes in the cluster can use their IP addresses+port numbers to access the workload targeted by the service. 2. Routing hops will be used. As a result, routing performance will be compromised and clients' source IP addresses will be masked.

* Port Mapping

Protocol	Container Port	Access Port
TCP	8080	Automatically ...

Step 6 Click **OK**.

Figure 1-18 Generating an access address



TCP/UDP Route Configuration Supports Layer-4 TCP/UDP load balancing.

Internal Domain Name Access Address	Access Address	Access Mode	Protocol	Container Port	Access Port	Operati...
service-mjm005.default.svc.cluster.local:8080	[redacted]:30596	Public network access -> EIP	TCP	8080	30596	Edit | Delete

Step 7 Click the access address in the **Access Address** column to access the application, as shown in **Figure 1-18**.

The following information is displayed:

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

Step 8 Enter **http://***Access address generated in* ***Step 6*****/rest/helloworld?name=ServiceStage** in the address box of the browser to access the application again.

The following information is displayed:

```
"ServiceStage"
```

----End

2 Getting Started with Common Practices

You can use the common practices provided by ServiceStage to meet your service requirements.

Table 2-1 Common practices

Practice	Description
Using ServiceStage to Host Microservice Applications	This practice describes how to quickly create a microservice application based on the ServiceComb (SpringMVC) framework to experience the ServiceStage functions.
Enabling Security Authentication for an Exclusive Microservice Engine	The exclusive microservice engine supports security authentication based on the Role-Based Access Control (RBAC) policy and allows you to enable or disable security authentication. After security authentication is enabled for an engine, the security authentication account and password must be configured for all microservices connected to the engine. Otherwise, the microservice fails to be registered, causing service loss. This practice describes how to enable security authentication for an exclusive microservice engine and ensure that services of microservice components connected to the engine are not affected.

Practice	Description
Connecting Microservice Engine Dashboard Data to AOM through ServiceStage	The real-time monitoring data of a Java chassis application deployed on the microservice engine dashboard is retained for 5 minutes by default. To permanently store historical monitoring data for subsequent query and analysis, use the custom metric monitoring function of ServiceStage to connect the microservice data displayed on the microservice engine dashboard to AOM. This practice uses the application deployed using a software package as an example to describe how to complete the connection.
Migrating the Registered Microservice Engine Using ServiceStage Without Code Modification	This practice describes how to migrate the microservice application components that are developed using the Java chassis microservice framework and registered with the professional microservice engine to the exclusive microservice engine without any code modification.