

ServiceStage

Getting Started

Issue 01
Date 2025-07-07



Copyright © Huawei Cloud Computing Technologies Co., Ltd. 2025. All rights reserved.

No part of this document may be reproduced or transmitted in any form or by any means without prior written consent of Huawei Cloud Computing Technologies Co., Ltd.

Trademarks and Permissions



HUAWEI and other Huawei trademarks are the property of Huawei Technologies Co., Ltd.

All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between Huawei Cloud and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Huawei Cloud Computing Technologies Co., Ltd.

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

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

Contents

1 Using ServiceStage to Host Microservice Applications..... 1

2 Getting Started with Common Practices..... 15

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.

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

1. **I. Registering a GitHub Account and Forking the Demo Source Code**
2. **II. Creating Repository Authorization**

3. [III. Creating an Organization](#)
4. [IV. Creating an Exclusive Microservice Engine](#)
5. [V. Creating an Environment](#)
6. [VI. Creating an Application](#)
7. [VII. Creating and Deploying a Component](#)
8. [VIII. Confirming the Deployment Result](#)
9. [IX. Accessing the Application](#)

Prerequisites

- You have [registered a Huawei account and enabled Huawei Cloud services](#).
- The login account has the permission to use ServiceStage. For details, see [Creating a User and Granting Permissions](#).
- You have created a VPC. For details, see [Creating a VPC](#).
- You have created a CCE cluster. For details, see [Buying a CCE Cluster](#).
 - The CCE cluster is in a VPC that has been created.
 - The cluster contains at least one ECS node with 4 vCPUs and 8 GB memory, and is bound to an EIP. For details, see [Creating a Node](#).

I. 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](#).

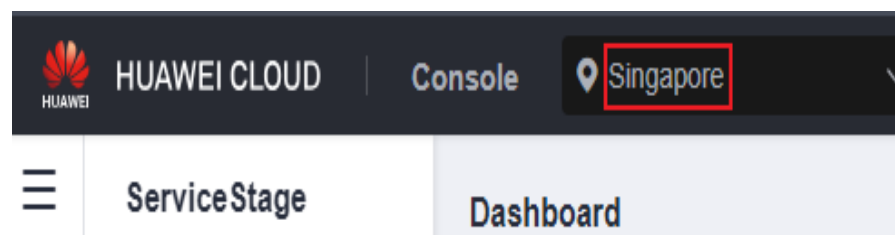
----End

II. 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 Selecting the service region



Step 3 In the navigation tree on the left, 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.
- Step 7** In the displayed dialog box, click **OK**. You can view the created authorization in the repository authorization list.
- End

III. Creating an Organization

- Step 1** In the navigation tree on the left, 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**.
- You can view the created organization on the **Organization Management** page.
- End

IV. 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** In the navigation tree on the left, choose **Cloud Service Engine > Engines**.
- Step 2** Click **Buy Exclusive Microservice Engine** and set the parameters by referring to the following table.

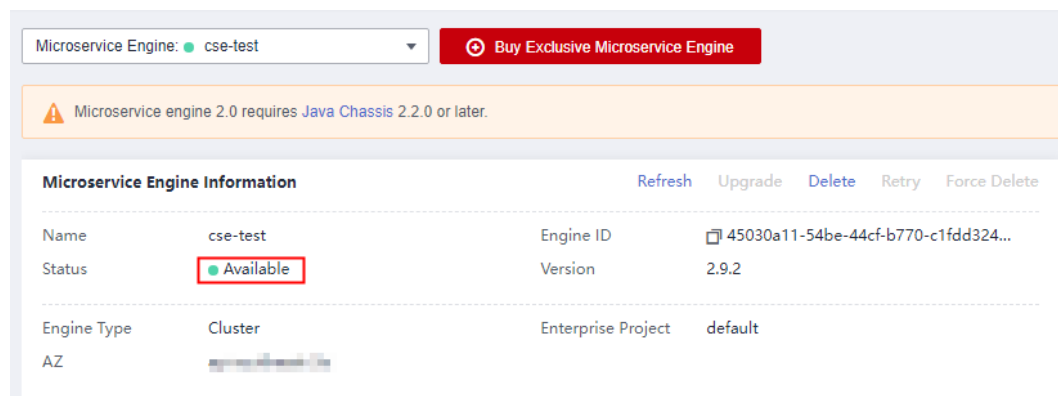
Parameter	Description
Billing Mode	Select Pay-per-use .
Enterprise Project	default is selected by default. Enterprise projects let you manage cloud resources and users by project. It is available after you enable the enterprise project function .
Specifications	Retain the default value.
Engine Type	Select 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 1-3 Creating a microservice engine



----End

V. Creating an Environment

Step 1 Choose **Environment Management > Create Environment**. Then set required environment parameters by referring to the following table, and retain the default values for other parameters.

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 .
Environment Type	Select Kubernetes .
HA Environment	Select No .
VPC	Select the VPC prepared in Prerequisites . The VPC cannot be modified after the environment is created.
Configuration Mode	Select Resource management .

Figure 1-4 Creating an environment

Basic Info

★ Environment

★ Enterprise Project [Create Enterprise Project](#)

★ Environment Type [?](#)

★ HA Environment

★ VPC [?](#) [Create a VPC](#)

Environment Tag [+](#) Add Tag
TMS's predefined tags let you add the same tag to different cloud resources. [View predefined tags](#)

Description 0/128 [↗](#)

Resource Settings

★ Configuration Mode [?](#)

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 Choose **ServiceComb Engines** under **Middleware** and click **Manage Resource**.

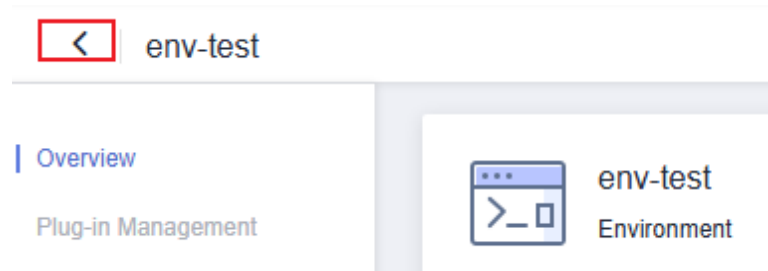
Step 6 In the dialog box that is displayed, select the ServiceComb engine created in [IV. Creating an Exclusive Microservice Engine](#) and click **OK**.

----End

VI. Creating an Application

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

Figure 1-5 Backing to Environment Management



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 1-6 Creating an application**Create Application**

★ Name

★ Enterprise Project ? C Create

Tags OK

You can add 20 more tags.

Description 0/128

OK Cancel

----End

VII. Creating and Deploying a Component

- Step 1** In the **Operation** column of the application (for example, **servicecomb**) created in [VI. Creating an Application](#), choose **More > Create Component**.

Figure 1-7 Creating a component

Name	Comp...	Created	Creator	Description	Enterprise Project	Tags	Operation
servicecomb	0	Nov 14, 2022 16:04:01 GM...	default	...	Edit Delete More Create Now Use Template

Total Records: 1

- 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 .
Application	Select the application created in VI. Creating an Application , for example, servicecomb .
Environment	Select the environment created in V. Creating an Environment , for example, env-test .
Namespace	Select default to isolate component instances.

Figure 1-8 Setting the basic component information

Basic Information

* Component Name	java-test
* Component Version	1.0.0
* Application	servicecomb
* Environment	env-test
* Namespace	default
* Workload Type	Stateless
* Workload	java-test-env-test-9usj1a
Label	+ Add Label
Description	--

[Generate](#) [Create Application](#) [Create Environment](#)

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 [II. Creating Repository Authorization](#).
5. **Username/Organization:** Select the GitHub account created when [I. Registering a GitHub Account and Forking the Demo Source Code](#).
6. **Repository:** Select demo source code repository **ServiceComb-SpringMVC** forked when [I. Registering a GitHub Account and Forking the Demo Source Code](#).
7. **Branch:** Select **master**.

Figure 1-9 Setting the component source

* Source Code/Softw...

Source code repository JAR package WAR package

GitHub GitLab Bitbucket

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

Authorization [Create Authorization](#) [Auth list](#)

Username/Organization Repository **ServiceComb-Spring...** Branch **master**

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 III. 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.
Namespace	Select default to isolate build data.

Figure 1-10 Configuring build parameters

Build Job

* Command Default command or script Custom command ⓘ

* Dockerfile Address ⓘ

* Organization org-test ▼ ⓘ ⓘ

* Build Use independent environment Use current environment

* Environment test-able2 ▼
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](#).

Step 5 Click **Next**.

Step 6 In the **Resources** area, set parameters by referring to the following table.

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

Figure 1-11 Setting component instance resources

Resources

★ Resources

CPU	<input type="checkbox"/> Request	0.25	Core Minimum number of CPU cores required by the container
	<input type="checkbox"/> Limit	0.25	Core Maximum number of CPU cores allowed for the container
Memory	<input type="checkbox"/> Request	0.5	GiB Minimum amount of memory required by the container
	<input type="checkbox"/> Limit	0.5	GiB Maximum amount of memory allowed for the container

★ Instances 1

★ Namespace

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 1-12 Binding a microservice engine


Cloud Service Settings ^


Microservice Engine Distributed Cache Service Relational Database Service

CSE

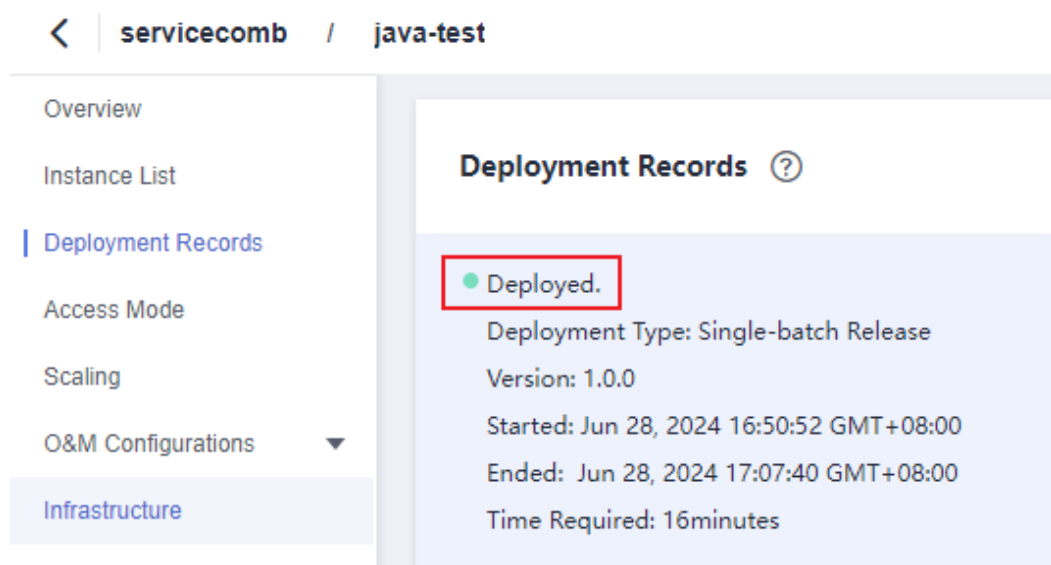
 **cse-test**
Available

Addons

 **Mesher**
Multi-language service mesh ac...

 **Sermant Injector** ⓘ [Install](#)
The Sermant Agent is automatic...

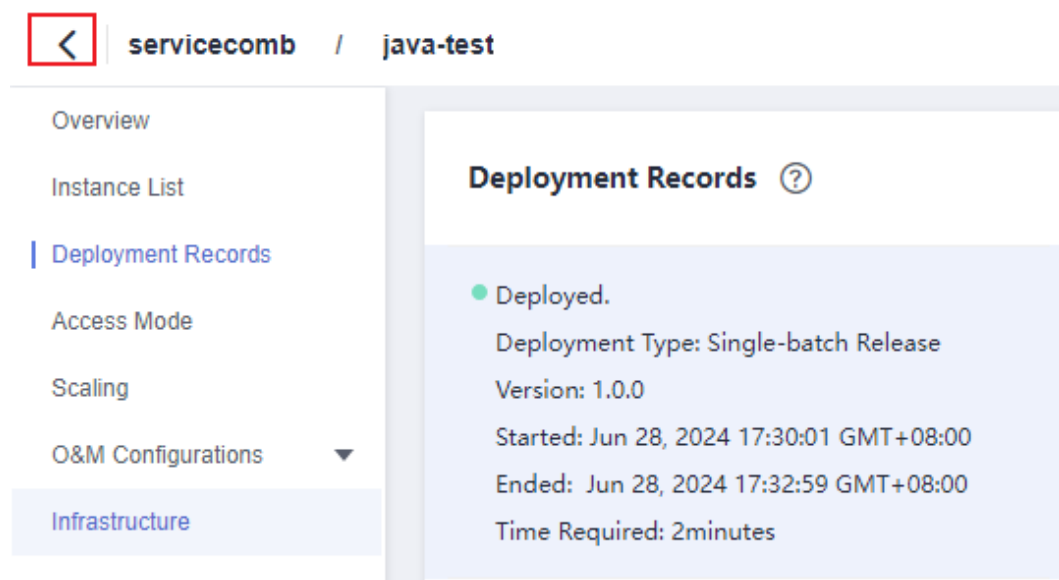
Step 8 Click **Create and Deploy** and wait until the component is deployed.

Figure 1-13 The component is deployed.

----End

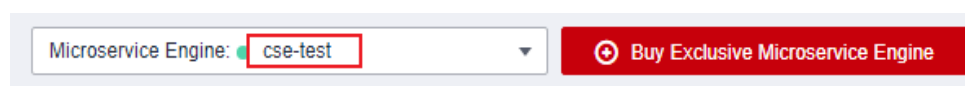
VIII. Confirming the Deployment Result

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

Figure 1-14 Backing to Component Management

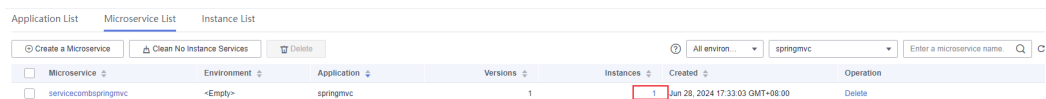
Step 2 Choose **Cloud Service Engine > Microservice Catalog**.

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



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 1-15 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

IX. Accessing the Application

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

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

Step 3 On the **Component List** tab, click the component created in **VII. 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 1-16 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 ...

OK

Cancel

Step 6 Click **OK**.

Figure 1-17 Generating an access address

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

⊕ Add Service

Internal Domain Name Access Address	Access Address	Access Mode	Protocol	Container Port	Access Port	Operati...
service-mjm005.default.svc.cluster.local:8080	EIP: 1.4.1.40:30596	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 1-17**.

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	<p>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.</p> <p>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.</p>

Practice	Description
Connecting Microservice Engine Dashboard Data to AOM through ServiceStage	<p>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.</p> <p>This practice uses the application deployed using a software package as an example to describe how to complete the connection.</p>
Migrating the Registered Microservice Engine Using ServiceStage Without Code Modification	<p>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.</p>