ServiceStage

Getting Started

Issue	01
Date	2023-11-02

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 Quick Experience

Overview

ServiceStage is an application management and O&M platform that lets you deploy, roll out, monitor, and maintain applications all in one place. It supports technology stacks such as Java, 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 implements a service feature of an application. It is in the form of code or software packages and can be deployed independently.

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 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?**

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 obtained AK/SK. For details, see **Access Keys**.

   NOTE

   ServiceStage provides professional microservice engines. If you use such an engine, configure an AK/SK for your application. In this example, the professional microservice engine is used.

   If the exclusive microservice engine is used, you do not need to configure the AK/SK. You need to create an exclusive microservice engine with security authentication disabled. For details, see **Creating a Microservice Engine**.

   - AK: access key ID. Unique identifier associated with the SK. The AK and SK are used together to obtain an encrypted signature for a request.
   - SK: secret access key, used together with AK. A secret access key works as a cryptographic signature to identify the sender of a request and prevent the request from being tampered with.
4. Create a VPC. For details, see **Creating a VPC**.
5. You have created a CCE cluster that contains at least one ECS node. (To facilitate subsequent operations, the node should have 4 vCPUs and 8 GB memory.) You have also bound an EIP to the cluster. For details, see **Buying a CCE Cluster**.
   - The VPC to which the CCE cluster belongs is the VPC created in **4**.
   - The CCE cluster cannot be bound to other environments.
   - If a CCE cluster 1.23 or later is created, **Container Engine** of the ECS node in the cluster supports only Docker.
   - The CCE cluster and its VPC must be in the same enterprise project if you have **enabled the enterprise project function**.
6. In this example, the GitHub source code repository is bound to ServiceStage to implement source code building, archiving, and application creation. Ensure that you have registered an account on **GitHub**.
7. You have created a repository authorization on ServiceStage to authorize access to your GitHub repository. For details, see **Authorizing a Repository**.

Forking the Source Code

Log in to your GitHub account and fork the demo source code.

Demo source code address: **https://github.com/servicestage-template/ServiceComb-SpringMVC**.

Creating an Organization

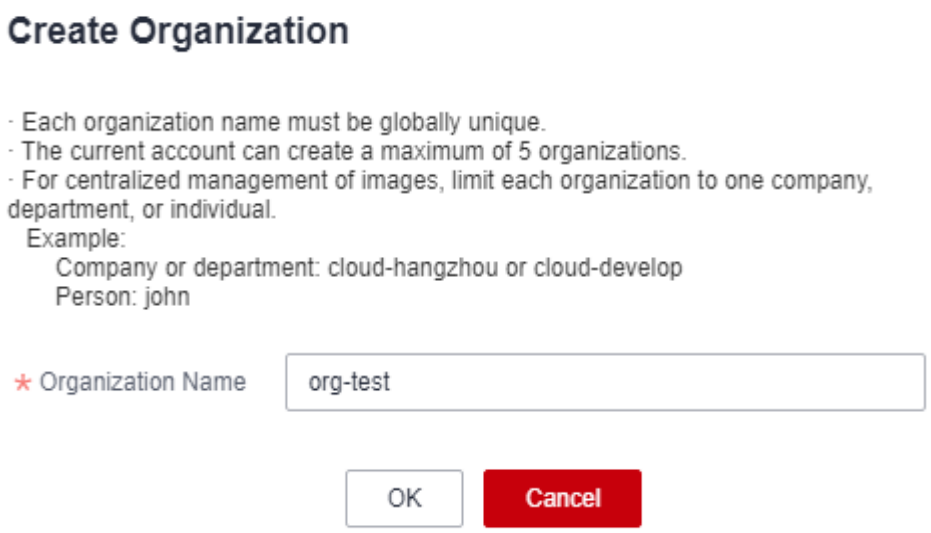
Step 1 Log in to ServiceStage.

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

Step 3 Click **Create Organization**. On the displayed page, specify **Organization Name**. For example, **org-test**.

Step 4 Click **OK**.

Figure 1-1 Creating an Organization



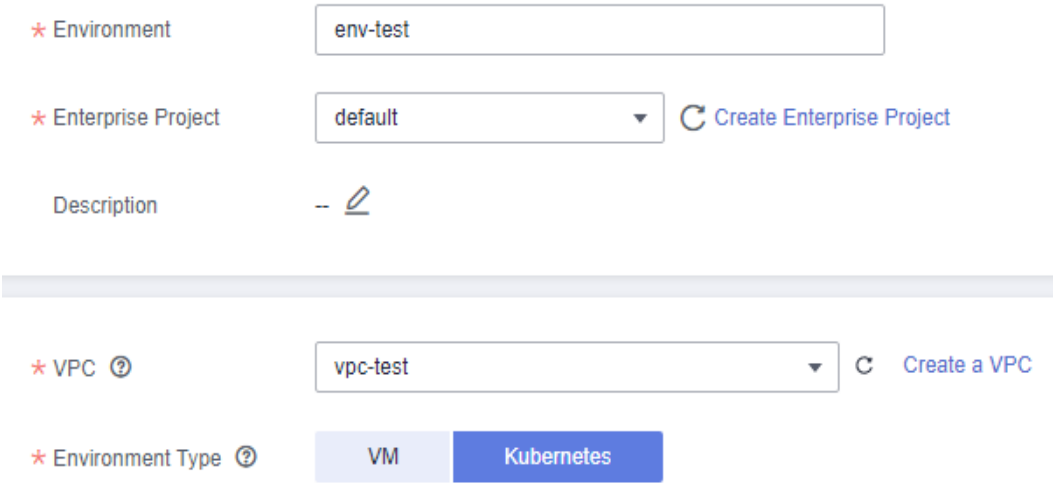
----End

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	Specify **Enterprise Project**. Enterprise projects let you manage cloud resources and users by project. It is available after you **enable the enterprise project function**. The environment and its VPC must be in the same enterprise project.
VPC	Select the VPC prepared in **Prerequisites**. **NOTE** The VPC cannot be modified after the environment is created.
Environment Type	Select **Kubernetes**.

Figure 1-2 Configuring an environment



Step 2 Click **Create Now**.

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

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

Step 5 In the **Resource Settings** area, choose **Cloud Service Engine** from **Middleware** and click **Manage Resource**.

Step 6 In the displayed dialog box, select a professional microservice engine and click **OK**.

----End

Creating an Application

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

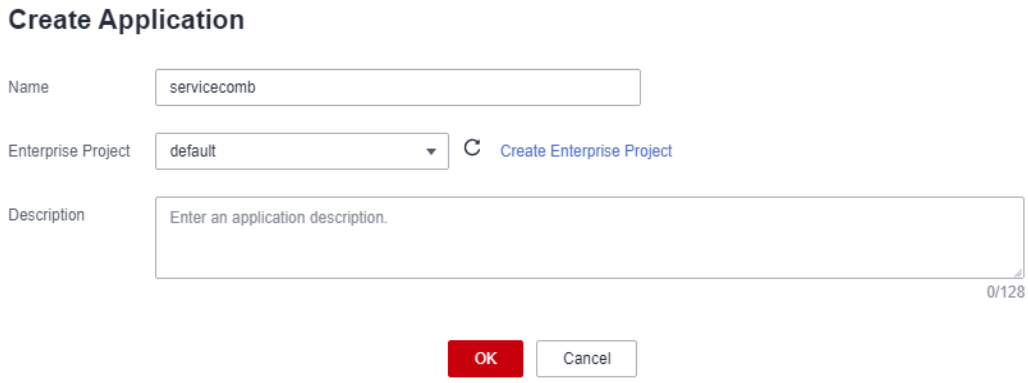
Step 2 Choose **Application Management** > **Create Application** and set basic application information.

1. **Name**: Enter an application name, for example, **servicecomb**.
2. **Enterprise Project**: Enterprise projects let you manage cloud resources and users by project.

   It is available after you **enable the enterprise project function**. The application and the **created environment** must be in the same enterprise project.

Step 3 Click **OK**.

Figure 1-3 Creating an application



----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.

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 1-4 Setting the basic component information

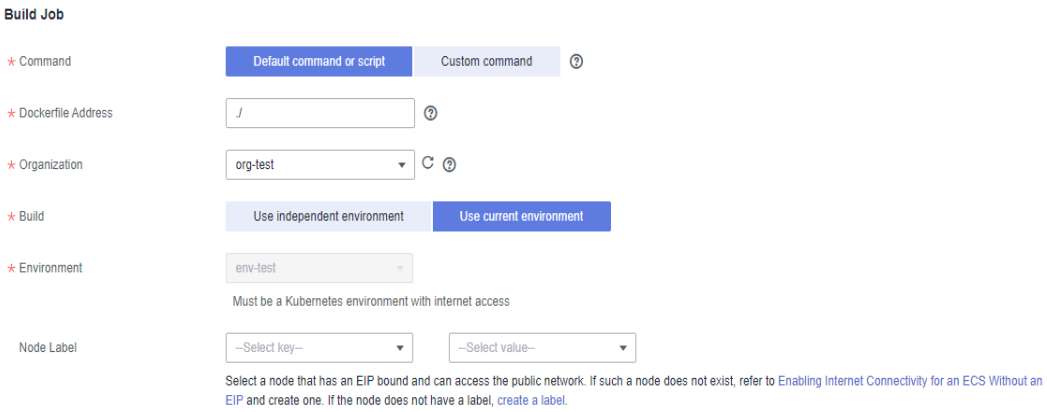
Step 3 In the **Component Package** area, set the following mandatory parameters. Retain the default values for other parameters.

Parameter	Description
Stack	Select **Java**.
Source Code/ Software Package	1. Select **Source code repository**. 2. Select **GitHub**, 3. set **Authorization** and **Username/Organization**, and select **ServiceComb-SpringMVC** for **Repository** and **master** for **Branch**.

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**. 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 1-5 Configuring build parameters



Step 5 Click **Next**.

Step 6 In the **Resources** area, set **Instances** to **2** and retain the default values for other parameters.

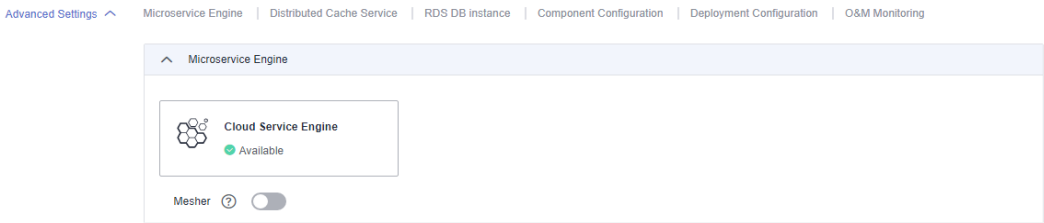
Step 7 In the **Access Mode** area, retain the default settings.

Step 8 In the **Local Time** area, retain the default settings.

Step 9 Bind a microservice engine.

1. In the **Advanced Settings** area, expand **Microservice Engine**.
2. Click **Bind Microservice Engine**.
3. Select the managed microservice engine in the current environment.
4. Click **OK**.

Figure 1-6 Binding a microservice engine



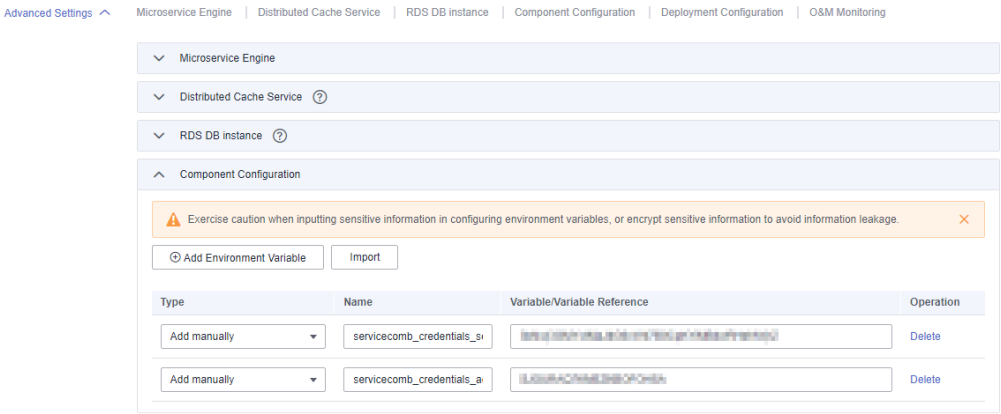
Step 10 (Optional) Configure the AK/SK.

If you use the professional microservice engine, you need to configure the AK/SK. If you use the exclusive microservice engine, skip this step.

1. In the **Advanced Settings** area, expand **Component Configuration**.
2. Click **Add Environment Variable** and configure AK/SK by referring to the following table.

Type	Name	Variable/Variable Reference
Add manually	servicecomb_credentials_accessKey	AK obtained in **Prerequisites**
	servicecomb_credentials_secretKey	SK obtained in **Prerequisites**

Figure 1-7 Configuring the AK/SK



Step 11 Click **Create and Deploy** and wait until the component is deployed.

----End

Confirming the Deployment Result

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

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

Step 3 Select **Cloud Service Engine** from the **Microservice Engine** drop-down list.

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

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

Figure 1-8 Confirming the deployment result

----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.

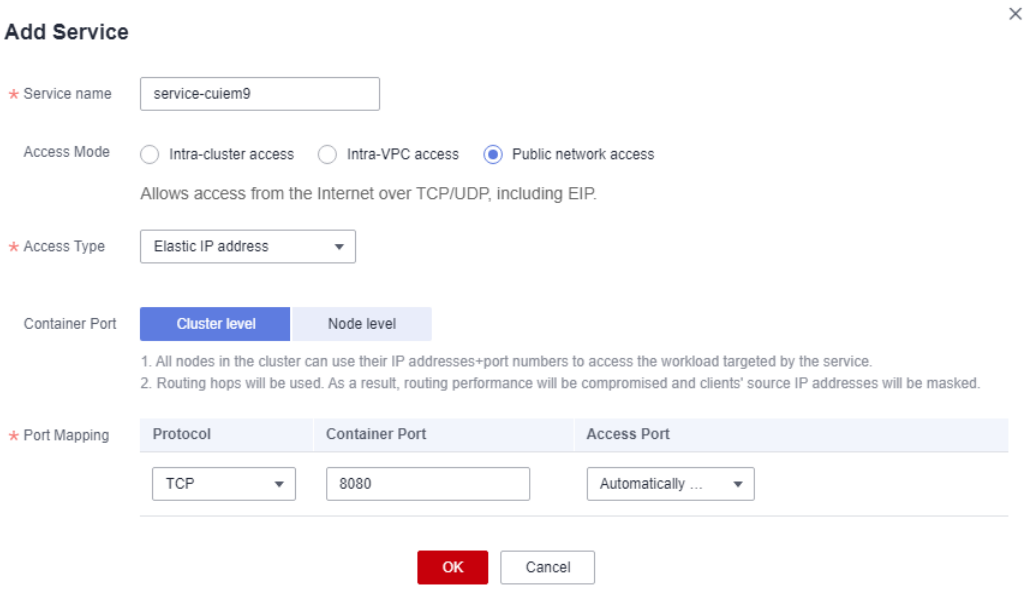
Step 3 On the **Component List** tab, click the component created in **Creating and Deploying a Component** (for example, **java-test**). The **Overview** page is displayed.

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.

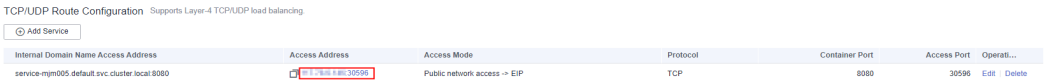
Parameter	Description
Service Name	Retain the default setting.
Access Mode	Select **Public network access**.
Access Type	Select **Elastic IP address**.
Service Affinity	Retain the default value.
Port Mapping	1. **Protocol**: Select **TCP**. 2. **Container Port**: Enter **8080**. 3. **Access Port**: Select **Automatically generated**.

Figure 1-9 Setting the access mode



Step 6 Click **OK**.

Figure 1-10 Access address



Step 7 Click the access address in the **Access Address** column to access the application, as shown in **Figure 1-10**.

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

Application O&M

Step 1 Go to ServiceStage console.

Step 2 Click **Application Management**.

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

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

For details, see **Component O&M**.

----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
Quick Experience	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.

