微服务引擎

快速入门

文档版本 01

发布日期 2024-03-20

华为云计算技术有限公司

商标声明


和其他华为商标均为华为技术有限公司的商标。
本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为云计算技术有限公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为云计算技术有限公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

华为云计算技术有限公司

地址： 贵州省贵安新区黔中大道交兴功路华为云数据中心 邮编：550029

网址： https://www.huaweicloud.com/

目 录

1 ServiceComb 引擎专享版

1.1 快速创建 ServiceComb 引擎

本章节帮助您快速了解如何创建ServiceComb引擎。

前提条件

- ServiceComb引擎运行于虚拟私有云，创建ServiceComb引擎前，需保证有可用的虚拟私有云和子网。
  创建虚拟私有云和子网，请参考**创建虚拟私有云和子网**。
- 当前登录账号拥有创建ServiceComb引擎的权限。账号权限授权与绑定，请参考**创建用户并授权使用微服务引擎**。

操作步骤

步骤1 进入**购买ServiceComb引擎专享版**页面。

步骤2 参考下表设置参数，参数前面带*号的是必须设置的参数。

参数	说明
*计费模式	选择计费方式，目前支持： • 包年/包月 • 按需计费
*企业项目	选择微服务引擎所在的项目，可在下拉框中搜索和选择需要的企业项目。
*选择实例数	选择引擎规格。
*引擎类型	选择微服务引擎的类型。 引擎类型为集群，其为集群模式部署，主机级容灾。
*ServiceComb引擎名称	输入ServiceComb引擎的名称，例如：cse-lhy-nodelete。
*可用区	选择可用区。

参数	说明
*网络	选择已创建的虚拟私有云及子网，可在下拉框中搜索和选择合适的虚拟私有云和子网。
描述	单击，输入引擎描述信息，例如：体验快速创建微服务引擎。
标签	用于标识云资源，当您拥有相同类型的许多云资源时，可以使用标签按各种维度（例如用途、所有者或环境）对云资源进行分类。 您可以单击“添加标签”，在“添加标签”弹框输入标签键和标签值，添加标签，标签的命名规则请参见**管理标签**。在“添加标签”弹框，可单击“新增标签”同时添加多个标签，也可单击标签后的，删除该标签。
*安全认证	开启了“安全认证”的ServiceComb引擎专享版，通过微服务引擎控制台提供了基于RBAC（Role-Based Access Control，基于角色的访问控制）的系统管理功能。 ● 选择“开启安全认证”： 1. 根据业务需要确认是否需要开启“编程接口安全认证”。 开启编程接口安全认证后，需要在微服务的配置文件中添加对应用户的账号密码，否则服务无法注册到引擎。 关闭编程接口安全认证，微服务的配置文件中无需配置账号密码即可将服务注册到引擎，效率性能更高，建议用于VPC内访问时使用。 2. 输入root账号的“密码”，并在“再次输入密码”输入框输入密码进行确认。 密码请妥善保管，以免遗失。 ● 选择“关闭安全认证”： 关闭安全认证功能，可以在实例创建完成后再设置开启。
*购买时长	计费模式选择“包年/包月”时需要设置。可设置是否开通自动续费。

步骤3 单击“立即购买”，进入引擎信息确认界面。

步骤4 单击“提交”，等待引擎创建完毕。

说明

- 微服务引擎创建完成，大约需要31分钟。
- 微服务引擎创建成功后，“状态”为“可用”。查看微服务引擎状态，请参考**查看ServiceComb引擎信息**。
- 如果微服务引擎创建失败，可在操作日志页面上查看失败原因并处理后可进行以下操作：
  - 可在“微服务引擎信息”区域，单击“重试”重新创建。
  - 如果重试失败，可删除创建失败的微服务引擎，删除微服务引擎，请参考**删除ServiceComb引擎专享版**。

----结束

1.2 接入 ServiceComb 引擎

1.2.1 SpringCloud 应用通过 SpringCloudHuawei SDK 接入 ServiceComb 引擎

本章节通过一个demo进行全流程的ServiceComb引擎使用操作演示，帮助您快速了解如何使用ServiceComb引擎。

说明

本章节将使用一个provider服务和一个consumer服务接入ServiceComb引擎。

前提条件

- 已创建ServiceComb引擎，请参考**快速创建ServiceComb引擎**。
- 下载github的**demo源码**到本地并解压。

  说明

  该demo的配置文件中已经完成了集成Spring Cloud Huawei的配置操作，若您需了解详细配置信息，请参考**Spring Cloud接入ServiceComb引擎**。
- 本地编译构建打包机器环境已安装了Java JDK、Maven，并且能够访问Maven中央库。

操作步骤

步骤1 登录微服务引擎控制台。

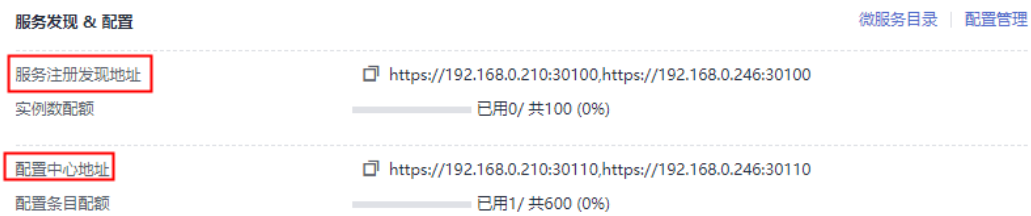
1. 登录**华为云控制台**。
2. 单击 ，选择区域。
3. 单击左上角 ，在服务列表选择“微服务引擎 CSE”，进入微服务引擎控制台。

步骤2 在左侧导航栏选择“ServiceComb引擎专享版”。

步骤3 单击**前提条件**中创建的ServiceComb引擎。

步骤4 获取ServiceComb引擎的注册中心地址和配置中心地址。

在“服务发现 & 配置”区域，查看获取引擎服务注册发现地址和配置中心地址。



步骤5 修改demo中的注册中心地址和配置中心地址。

1. 在下载到本地的demo源码目录下，分别找到“\basic\consumer\src\main\resources\bootstrap.yml”和“\basic\provider\src\main\resources\bootstrap.yml”文件。
2. 添加ServiceComb引擎的注册中心地址和配置中心地址到项目配置文件中（以“\basic\consumer\src\main\resources\bootstrap.yml”为例）。

```
spring:
  application:
    name: basic-consumer
  cloud:
    servicecomb:
      discovery:
        enabled: true
        watch: false
        # 注册中心地址
        address: https://192.168.0.210:30100,https://192.168.0.246:30100
        appName: basic-application
        serviceName: ${spring.application.name}
        version: 0.0.1
        healthCheckInterval: 30
      config:
        # 配置中心地址
        serverType: kie
        serverAddr: https://192.168.0.210:30110,https://192.168.0.246:30110
```

说明

使用ServiceStage部署的场景，服务注册中心地址和配置中心地址在部署过程中会自动注入，无需额外手工添加。

步骤6 打包demo源码成jar包。

1. 在demo源码根目录下，打开cmd命令，执行**mvn clean package**命令，对项目进行打包编译。
2. 编译成功后，生成如**表1-1**所示的两个Jar包。

表 1-1 软件包列表

软件包所在目录	软件包名称	说明
basic\consumer\target	basic-consumer-1.0-SNAPSHOT.jar	服务消费者
basic\provider\target	basic-provider-1.0-SNAPSHOT.jar	服务生产者

步骤7 部署应用。

- 方法一：直接将微服务provider和consumer部署至ServiceComb引擎所在VPC的ECS节点。
  a. 请参考**购买并登录Linux弹性云服务器**在引擎实例所属VPC下创建一台ECS节点并登录。
  b. 安装JRE，为服务提供运行环境。
  c. 将**步骤6**生成JAR包上传至ECS节点。
  d. 执行命令：**java -jar {对应jar包}**，运行生成的JAR包。
- 方法二：使用ServiceStage部署微服务provider和consumer。
  a. 将**步骤6**生成的JAR包上传至OBS。

b. 参考**快速创建Kubernetes集群**创建CCE集群，并同ServiceComb引擎实例属于同一VPC。

c. 参考**创建环境**在引擎实例所在VPC下创建ServiceStage环境，并对ServiceComb引擎和CCE资源进行纳管。

d. 参考**创建并部署组件**部署provider和consumer微服务。

步骤8 确认部署结果。

1. **可选:** 在微服务引擎控制台页面，在左侧导航栏选择“ServiceComb引擎专享版”，单击**前提条件**中创建的ServiceComb引擎。
2. 选择“微服务目录 > 微服务列表”，查看微服务basic-consumer和basic-provider的实例数量。
   - 若实例数量值不为0，则表示已经成功接入ServiceComb引擎。
   - 若实例数量为0，或者找不到basic-consumer和basic-provider服务名，则表示微服务应用接入ServiceComb引擎失败。

----结束

1.2.2 Spring Cloud 应用通过 Sermant Agent 接入 ServiceComb 引擎

1.2.2.1 概述

CSE提供Sermant Agent，支持Spring Cloud应用无需任何修改接入ServiceComb引擎，当前已支持应用注册发现、配置、优雅上下线、标签路由等功能。

说明

- 此功能目前处于公测阶段，当前仅在华东-上海一支持。
- Sermant Agent是基于**Sermant**开源社区构建的、用于CSE微服务治理场景的Agent。
- Sermant Agent基于Java Agent技术实现，应用通过Sermant Agent可实现无代理、非侵入方式接入CSE。

1.2.2.2 特性版本支持

注册发现

Sermant Agent支持Spring Cloud应用快速接入ServiceComb引擎，支持版本如下：

Spring Cloud Version	Spring Boot Version
Edgware.SR2+	1.5.x
Finchley.x	2.0.x
Greenwich.x	2.1.x
Hoxton.x	2.2.x、2.3.x
2020.0.x	2.4.x、2.5.x
2021.0.0	2.6.x

说明

注册中心版本需与spring-cloud-dependencies版本保持一致。

配置管理

Spring Boot	Nacos Config	Zookeeper Config
1.5.0.RELEASE - 2.6.2	1.5.0.RELEASE+	1.2.0.RELEASE+

说明

- Nacos Config为maven依赖的spring-cloud-starter-alibaba-nacos-config。
- Zookeeper Config为maven依赖的spring-cloud-starter-zookeeper-config。
- 配置管理能力需配合注解@RefreshScope、@Value、@ConfigurationProperties使用。

优雅上下线

优雅上下线支持版本同**注册发现**。

标签路由

标签路由版本支持：

Spring Cloud	Spring Boot	Spring Cloud Openfeign	RestTemplate	Spring Cloud Loadbalancer	Spring Cloud Netflix Ribbon	Spring Cloud Gateway	Spring Cloud Netflix Zuul
Edgware.SR2+	1.5.x	1.4.7.RELEASE	4.3.6.RELEASE	-	1.4.7.RELEASE	-	1.4.7.RELEASE
Finchley.x	2.0.x	2.0.x	5.0.x	-	2.0.x	-	2.0.x
Greenwich.x	2.1.x	2.1.x	5.1.x	-	2.1.x	-	2.1.x
Hoxton.x	2.2.x、2.3.x	2.2.x	5.2.x	2.2.5.RELEASE+	2.2.x	2.2.x	2.2.x
2020.0.x	2.4.x、2.5.x	3.0.x	5.3.x	3.0.x	-	3.0.x	-
2021.0.0	2.6.x	3.1.x	5.3.x	3.1.x	-	3.1.x	-

1.2.2.3 虚拟机部署场景接入指南

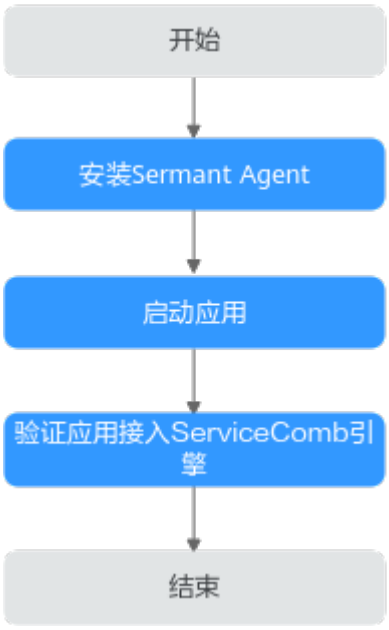
虚机部署的应用可通过Sermant Agent接入到ServiceComb引擎。

前置条件

- 已创建ECS实例， 创建ECS请参考**ECS快速入门**。
- 已安装JDK（版本为1.8及以上版本）并配置环境变量，详情请参考**Java Downloads**。
- 已创建ServiceComb引擎实例，详情请参考**快速创建ServiceComb引擎**。
- ECS与ServiceComb引擎处于相同的VPC网络下。
- Sermant Agent开源版本要求1.0.3及以上。

接入流程

基于ECS将应用接入ServiceComb引擎流程如下：



操作步骤

步骤1 安装Sermant Agent。

1. 登录Linux弹性云服务器。
   请参考**Linux弹性云服务器登录方式概述**选择相应方式登录弹性云服务器。
2. 下载并安装Sermant Agent。
   参考如下命令通过shell脚本方式下载并安装Java Agent。
   wget -O- https://cse-bucket-cn-east-3.obs.cn-east-3.myhuaweicloud.com/javaagent/install.sh | sh

   说明

   安装成功后，脚本将输出安装目录。目录为**当前用户主目录**。

步骤2 启动应用。

在应用的启动参数上添加如下参数，添加启动参数后，待应用启动完成。

```
-javaagent:${HOME}/java-agent/java-agent.jar=appName=default
-Ddynamic_config_serverAddress={CSE_CONFIG_CENTER_ENDPOINTS}
-Dregister.service.address={CSE_REGISTRY_ENDPOINTS}
```

表 1-2 启动参数说明

参数项	说明
appName	agent服务名称，默认default，无需修改。
dynamic_config_serverAddress	ServiceComb引擎配置中心地址，多个地址使用逗号隔开。
register.service.address	ServiceComb引擎服务注册发现地址。

说明

- 若需配置APP名称（默认default）、版本（默认1.0.0）请分别使用环境变量-Dservice_meta_application=yourAppName、-Dservice_meta_version=yourVersion进行设置。
- ServiceComb引擎服务注册发现地址与ServiceComb引擎配置中心地址需替换为实际地址，可参考如下方式获取：
  - ServiceComb引擎服务注册发现地址：**获取ServiceComb引擎服务注册发现地址**。
  - ServiceComb引擎配置中心地址：**获取ServiceComb引擎配置中心地址**。

步骤3 验证应用接入ServiceComb引擎。

参考**查看微服务列表**查看您的应用是否已接入ServiceComb引擎。

----结束

1.2.2.4 CCE 部署场景接入指南

1.2.2.4.1 概述

CCE部署的应用可通过Sermant Injector自动挂载Sermant Agent，从而通过Sermant Agent接入到CSE的ServiceComb引擎。当前支持通过模板管理页面和Helm客户端两种方式部署Sermant Injector。

前置条件

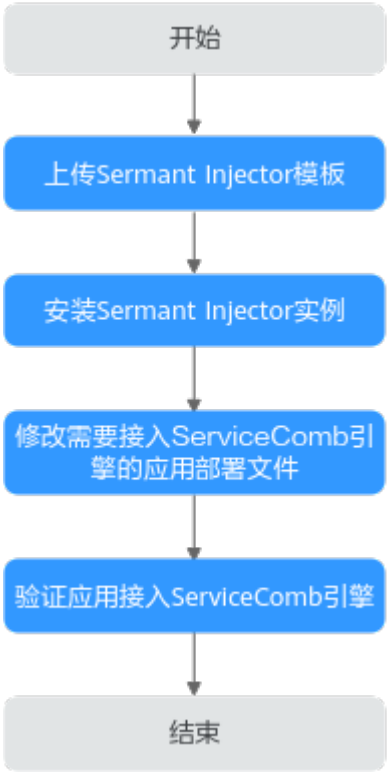
- 已创建云容器引擎（CCE），创建CCE请参考**创建CCE集群**。
- CCE集群版本需要大于等于1.15。
- 应用的基础镜像中，需要安装JDK（版本为1.8及以上版本）。
- 已安装kubectl命令，安装kubectl命令请参考**通过kubectl连接集群**中相关操作。
- 已创建ServiceComb引擎实例，详情请参考**快速创建ServiceComb引擎**。
- CCE集群与ServiceComb引擎处于相同的VPC网络下。

- 给Sermant Injector预留200m左右的cpu资源和300Mi左右的memory资源。
- Sermant Injector版本要求1.0.11及以上，Sermant Agent镜像版本要求1.0.9及以上。

1.2.2.4.2 通过模板管理页面部署 Sermant Injector

接入流程

通过模板管理页面部署Sermant Injector将应用接入ServiceComb引擎流程如下：



操作步骤

步骤1 上传Sermant Injector模板。

1. 下载模板。

   Sermant Injector模板版本及下载地址如下表所示：

Sermant Injector版本	Sermant Agent镜像版本	Sermant Agent开源版本	发行时间	获取路径
1.0.11	1.0.9	1.0.6	2023.10.31	sermant-injector-1.0.11.tgz

2. 上传模板。

   登录CCE的控制台，进入集群，在左侧导航栏中选择“模板管理”，单击右上角的“上传模板”，单击“添加文件”，选择已下载的模板包。具体操作请参考**上传模板**。

说明

如果您的Sermant Injector版本低于1.0.11，您还需要在上传Sermant Injector模板之前进行以下准备工作：

第一次启动Sermant Injector应用之前，需申请Sermant Injector https证书。

1. 登录已安装kubectl命令的CCE节点。

   请参考**Linux弹性云服务器登录方式概述**选择相应方式登录CCE节点。

2. 登录后，请在**已安装kubectl命令的CCE节点**中执行以下命令申请Sermant Injector https证书：

   wget -O- https://cse-bucket-cn-east-3.obs.cn-east-3.myhuaweicloud.com/javaagent/certificate.sh | sh

   - 该步骤会把证书挂载到cse命名空间中，如果不存在cse命名空间，则会自动创建。
   - 该步骤会向k8s集群申请名为sermant-injector.cse.svc的CertificateSigningRequest，如果之前存在，则会被覆盖。
   - 该步骤会在cse命名空间中创建名为sermant-injector-secret的Secret，如果之前存在，则会被覆盖。
   - 使用Sermant Injector时，如果提示证书失效等证书相关的错误，请重新申请证书并重新安装Sermant Injector实例。

步骤2 安装Sermant Injector实例。

登录CCE控制台，进入集群，在左侧导航栏中选择“模板管理”，在已上传的Sermant Injector模板中，单击“安装”，具体操作请参考**创建模板实例**。关于模板的更多操作请参考**通过模板部署应用** 。

安装时，按需修改配置文件，配置说明如下：

```
agent:
  image:
    # 选填配置，Sermant Agent镜像版本，默认为最新版本。
    version: ${agent.version}
cse:
  config:
    # 必填配置，ServiceComb引擎配置中心地址，获取方式可参考获取ServiceComb引擎配置中心地址。
    endpoints: https://localhost:30110
  registry:
    # 必填配置，注册中心类型，当前支持SERVICE_COMB/NACOS
    type: SERVICE_COMB
    # 必填配置，ServiceComb引擎注册中心地址，获取方式可参考获取ServiceComb引擎注册发现地址。
    endpoints: https://localhost:30100
image:
  # 选填配置，镜像拉取策略：Always(总是拉取)/IfNotPresent(本地有则使用本地镜像,不拉取)/Never(只使用本地镜像，从不拉取)
  pullPolicy: IfNotPresent
  # 必填配置，CCE所在的region，默认为cn-east-3（华东-上海一），具体请参考地区和终端节点。
  region: cn-east-3
injector:
  image:
    # 选填配置，injector镜像版本，默认为最新版本。
    version: ${injector.version}
    # 选填配置，拉取镜像的密钥。
    pullSecrets: default-secret
    # 选填配置，injector实例数，若CCE集群只有一个节点，则需配置为1。
    replicas: 2
```

说明

如果您的Sermant Injector版本低于1.0.11，还需在”injector”节点下，配置“webhooks > caBundle”的值，该值需要在已安装kubectl命令的CCE节点中使用以下命令获取：

kubectl config view --raw --minify --flatten -o jsonpath='{.clusters[].cluster.certificate-authority-data}'

步骤3 修改需要接入ServiceComb引擎的应用部署文件。

- 若是新建的应用需要在“创建无状态工作负载”高级设置页面单击界面右侧的“YAML创建”，具体请参考**通过控制台创建无状态负载**。
- 若是已经部署的应用需要在左侧导航栏中选择“工作负载 > 无状态负载 Deployment”，然后单击应用所属工作负载后的“更多 > 编辑YAML”，具体请参考**编辑YAML**。

在YAML文件中的“spec > template > metadata > labels”层级下加入内容：sermant-injection: enabled。

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: spring-cloud-registry-provider
  labels:
    app: spring-cloud-registry-provider
spec:
  replicas: 1
  selector:
    matchLabels:
      app: spring-cloud-registry-provider
  template:
    metadata:
      labels:
        app: spring-cloud-registry-provider
        sermant-injection: enabled
    spec:
      containers:
        - name: spring-cloud-registry-provider
          image: spring-cloud-registry-provider:1.0.0
```

说明

- 新建的应用在启动时会自动挂载Sermant Agent。
- 已经部署的应用在修改YAML后会自动重启并挂载Sermant Agent。

步骤4 验证应用接入ServiceComb引擎。

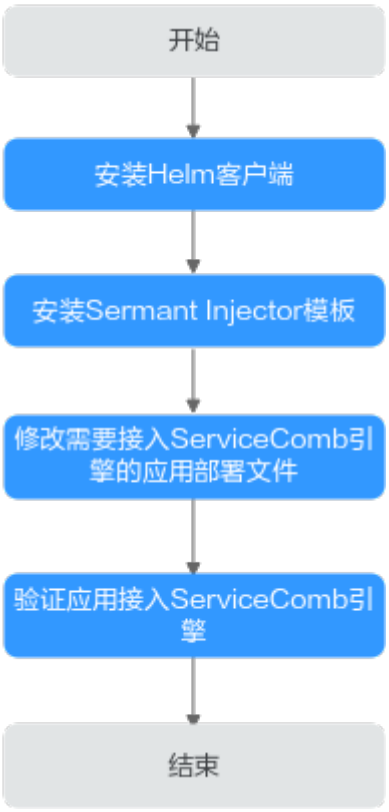
参考**查看微服务列表**查看您的应用（服务名为spring.application.name配置的值）是否已接入ServiceComb引擎。

----结束

1.2.2.4.3 通过 Helm 客户端部署 Sermant Injector

接入流程

通过Helm客户端部署Sermant Injector将应用接入ServiceComb引擎流程如下：



操作步骤

步骤1 安装**Helm**客户端。

Helm下载链接：**https://github.com/helm/helm/releases**，请选择合适的版本，本文以helm v3.3.0为例进行演示。

1. 下载**helm-v3.3.0-linux-amd64.tar.gz**。
2. 登录已安装kubectl命令的CCE节点，然后将Helm安装包上传到CCE节点上。
   请参考**Linux弹性云服务器登录方式概述**选择相应方式登录CCE节点。
3. 解压Helm安装包。
   在Helm安装包所在的路径执行命令**tar -zxf helm-v3.3.0-linux-amd64.tar.gz**解压Helm安装包。
4. 将helm移动到系统path所在路径。
   以“/usr/local/bin/helm”为例，在Helm解压包所在的路径执行命令**mv linux-amd64/helm /usr/local/bin/helm**进行移动。
5. 验证安装结果。
   执行命令**helm version**，如果输出下图中的信息，则说明安装成功：

说明

安装用户需要有安装目录和系统path路径的读写权限。

Helm客户端版本需要大于等于3.1。

如果您的Sermant Injector版本低于1.0.11，您还需要在该步骤之后进行以下准备工作：

第一次启动Sermant Injector应用之前，需申请Sermant Injector https证书。

1. 登录已安装kubectl命令的CCE节点。

   请参考**Linux弹性云服务器登录方式概述**选择相应方式登录CCE节点。

2. 登录后，请在**已安装kubectl命令的CCE节点**中执行以下命令申请Sermant Injector https证书：

   wget -O- https://cse-bucket-cn-east-3.obs.cn-east-3.myhuaweicloud.com/javaagent/certificate.sh | sh

   - 该步骤会把证书挂载到cse命名空间中，如果不存在cse命名空间，则会自动创建。
   - 该步骤会向k8s集群申请名为sermant-injector.cse.svc的CertificateSigningRequest，如果之前存在，则会被覆盖。
   - 该步骤会在cse命名空间中创建名为sermant-injector-secret的Secret，如果之前存在，则会被覆盖。
   - 使用Sermant Injector时，如果提示证书失效等证书相关的错误，请重新申请证书并重新安装Sermant Injector实例。

步骤2 安装Sermant Injector模板。

1. 登录**已安装Helm客户端的CCE节点**并执行以下命令下载Sermant Injector模板：

   wget -O- 'https://cse-bucket-cn-east-3.obs.cn-east-3.myhuaweicloud.com/javaagent/sermant-injector-1.0.11.tgz' | tar zx

   Sermant Injector模板版本信息如下：

Sermant Injector版本	Sermant Agent镜像版本	Sermant Agent开源版本	发行时间
1.0.11	1.0.9	1.0.6	2023.10.31

2. 修改配置。

   下载模板后，编辑模板包中的values.yaml文件，按需修改配置。配置说明如下：

```
agent:
  image:
    # 选填配置，Sermant Agent镜像版本，默认为最新版本。
    version: ${agent.version}
cse:
  config:
    # 必填配置，ServiceComb引擎配置中心地址，获取方式可参考获取ServiceComb引擎配置中心地址。
    endpoints: https://localhost:30110
  registry:
    # 必填配置，注册中心类型，当前支持SERVICE_COMB/NACOS
    type: SERVICE_COMB
    # 必填配置，ServiceComb引擎注册中心地址，获取方式可参考获取ServiceComb引擎注册发现地址。
    endpoints: https://localhost:30100
image:
  # 选填配置，镜像拉取策略：Always(总是拉取)/IfNotPresent(本地有则使用本地镜像,不拉取)/Never(只使用本地镜像，从不拉取)
  pullPolicy: IfNotPresent
  # 必填配置，CCE所在的region，默认为cn-east-3（华东-上海一），具体请参考地区和终端节点。
  region: cn-east-3
injector:
  image:
    # 选填配置，injector镜像版本，默认为最新版本。
    version: ${injector.version}
```

```
# 选填配置，拉取镜像的密钥。
pullSecrets: default-secret
# 选填配置，injector实例数，若CCE集群只有一个节点，则需配置为1。
replicas: 2
```

说明

如果您的Sermant Injector版本低于1.0.11，还需在”injector”节点下，配置“webhooks > caBundle”的值，该值需要在已安装kubectl命令的CCE节点中使用以下命令获取：

kubectl config view --raw --minify --flatten -o jsonpath='{.clusters[].cluster.certificate-authority-data}'

3. 安装Sermant Injector模板。

   在Sermant Injector模板包所在路径执行命令**helm install sermant-injector sermant-injector-1.0.11**进行安装。

说明

如果需要卸载Sermant Injector，请执行**helm uninstall sermant-injector**命令。

注意：卸载Sermant Injector后，不会再自动挂载Sermant Agent。已挂载Sermant Agent的应用如果未重启则不受影响，如果重启，因为Sermant Injector已被卸载，则应用不会再挂载Sermant Agent。

步骤3 修改需要接入ServiceComb引擎的应用部署文件。

- 如果是新建应用，请直接编辑YAML文件。
- 如果是已经部署的应用，请使用**kubectl edit**命令进行编辑YAML文件。

  以Deployment为例，执行命令**kubectl edit deployment {*DeploymentName*} -o yaml -n {*namespace*}**。

说明

{*DeploymentName*}需要替换为具体的Deployment名称，{*namespace*}需要替换为Deployment所在的命名空间，若您不清楚DeploymentName或者NAMESPACE，可以使用以下命令查看在CCE中部署的所有Deployment，然后找到您需要接入CSE的应用的DeploymentName与NAMESPACE：

kubectl get deployments --all-namespaces

命令执行结果示例如下：

```
[redacted]# kubectl get deployments --all-namespaces
NAMESPACE      NAME                          READY   UP-TO-DATE   AVAILABLE   AGE
[redacted]                                   2/2     2            2           41h
default        service-b                     1/1     1            1           24d
default        service-c                     1/1     1            1           17d
default        spring-demo-a                 1/1     1            1           52s
[redacted]                                   1/1     1            1           28d
[redacted]                                   1/1     1            1           28d
[redacted]                                   1/2     2            1           443d
[redacted]                                   1/2     2            1           443d
[redacted]                                   1/1     1            1           23d
```

在YAML文件中的“spec > template > metadata > labels”层级下加入内容：sermant-injection: enabled。

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: spring-cloud-registry-provider
  labels:
    app: spring-cloud-registry-provider
spec:
  replicas: 1
  selector:
    matchLabels:
      app: spring-cloud-registry-provider
  template:
    metadata:
      labels:
        app: spring-cloud-registry-provider
        sermant-injection: enabled
    spec:
      containers:
        - name: spring-cloud-registry-provider
          image: spring-cloud-registry-provider:1.0.0
```

说明

- 新建的应用在启动时会自动挂载Sermant Agent。
- 已经部署的应用在修改YAML后会自动重启并挂载Sermant Agent。

步骤4 验证应用接入ServiceComb引擎。

参考**查看微服务列表**查看您的应用（服务名为spring.application.name配置的值）是否已接入ServiceComb引擎。

----结束

1.2.2.4.4 部署 SpringCloud 应用

本章节使用**spring-cloud-registry-demo**演示接入CSE的ServiceComb引擎。

说明

本章节将使用一个provider服务和一个consumer服务接入ServiceComb引擎。

前提条件

- 已创建云容器引擎（CCE），创建CCE请参考**创建CCE集群**。
- CCE集群版本需要大于等于1.15。
- 已安装kubectl命令，安装kubectl命令请参考**通过kubectl连接集群**中相关操作。
- 已创建ServiceComb引擎实例，详情请参考**快速创建ServiceComb引擎**。
- CCE与ServiceComb引擎处于相同的VPC网络下。
- 下载**Sermant-examples**到本地并解压。
- 本地编译构建打包机器环境已安装了Java JDK、Maven，并且能够访问Maven中央库。

- 已在CCE集群上部署Sermant Injector，详情请参考**通过模板管理页面部署Sermant Injector**或者**通过Helm客户端部署Sermant Injector**。

操作步骤

步骤1 打包Sermant-examples。

1. 在“Sermant-examples”根目录下，打开**cmd**命令，执行**mvn clean package**命令，对项目进行打包编译。编译成功后，获取下表中的两个软件包。

   表 1-3 软件包列表

软件包所在目录	软件包名称	说明
Sermant-examples/registry-demo/spring-cloud-registry-demo/spring-cloud-registry-consumer/target	spring-cloud-registry-consumer.jar	服务消费者
Sermant-examples/registry-demo/spring-cloud-registry-demo/spring-cloud-registry-provider/target	spring-cloud-registry-provider.jar	服务生产者

2. 把spring-cloud-registry-consumer.jar复制到“Sermant-examples/registry-demo/spring-cloud-registry-demo/deployment/images/consumer”中。
3. 把spring-cloud-registry-provider.jar复制到“Sermant-examples/registry-demo/spring-cloud-registry-demo/deployment/images/provider”中。

步骤2 制作镜像。

1. 登录**已安装kubectl命令**且**已部署Sermant Injector**的CCE集群中的节点。
2. 把“Sermant-examples/registry-demo/spring-cloud-registry-demo”中的“deployment”文件夹上传至已登录的CCE集群中的节点上。
3. 请参考**使用容器引擎客户端上传镜像**制作docker镜像，其中，使用到的Dockerfile请参考“Sermant-examples/registry-demo/spring-cloud-registry-demo/deployment/images/consumer”与“Sermant-examples/registry-demo/spring-cloud-registry-demo/deployment/images/provider”目录下的Dockerfile文件按需修改。

步骤3 部署spring-cloud-registry-consumer.yaml与spring-cloud-registry-provider.yaml。

1. 修改镜像名。

   将已上传deployment文件夹到CCE集群中的节点中的“deployment/k8s/spring-cloud-registry-consumer.yaml“与“deployment/k8s/spring-cloud-registry-provider.yaml“中的镜像名修改为您所制作的镜像名。

2. 在已上传deployment文件夹到CCE集群中的节点中的“deployment/k8s“目录下，执行如下命令部署spring-cloud-registry-consumer.yaml与spring-cloud-registry-provider.yaml：

   kubectl create -f spring-cloud-registry-consumer.yaml

   kubectl create -f spring-cloud-registry-provider.yaml

说明

若需配置APP名称（默认default）、版本（形如a.b.c的格式，其中a、b、c均为数字，默认为1.0.0）请在yaml中增加SERVICE_META_APPLICATION与SERVICE_META_VERSION环境变量进行配置。如下所示：

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: spring-cloud-registry-provider
  labels:
    app: spring-cloud-registry-provider
spec:
  replicas: 1
  selector:
    matchLabels:
      app: spring-cloud-registry-provider
  template:
    metadata:
      labels:
        app: spring-cloud-registry-provider
        sermant-injection: enabled
    spec:
      containers:
        - name: spring-cloud-registry-provider
          image: spring-cloud-registry-provider:1.0.0
          env:
            - name: "SERVICE_META_APPLICATION"
              value: "yourAppName"
            - name: "SERVICE_META_VERSION"
              value: "yourVersion"
```

步骤4 验证应用接入ServiceComb引擎。

参考**查看微服务列表**查看应用（服务名为spring-cloud-registry-consumer与spring-cloud-registry-provider）是否已接入ServiceComb引擎。

----结束

1.2.3 Dubbo 应用通过 Sermant Agent 接入 ServiceComb 引擎

1.2.3.1 概述

CSE提供Sermant Agent，支持Dubbo应用无需任何修改接入ServiceComb引擎，当前已支持应用注册发现、标签路由功能。

说明

- 此功能目前处于公测阶段，当前仅在华东-上海一支持。
- Sermant Agent是基于**Sermant**开源社区构建的、用于CSE微服务治理场景的Agent。
- Sermant Agent基于Java Agent技术实现，应用通过Sermant Agent可实现无代理、非侵入方式接入ServiceComb引擎，并获得标签路由能力。

1.2.3.2 特性版本支持

注册发现

Sermant Agent支持Dubbo框架开发的应用快速接入ServiceComb引擎，支持的Dubbo框架版本为**2.5.8**、**2.6.x** 、**2.7.x**。

说明

- 对于**新开发**的dubbo应用，在开发时，需要配置dubbo本身注册中心地址。该配置项一般在dubbo应用的配置文件中，比如“dubbo/provider.xml”文件中：
  ```
  <dubbo:registry address="sc://127.0.0.1:30100"/>
  ```
  也可能在application.yml（或application.properties）中，以application.yml为例：
  ```
  dubbo:
   registry:
     address: sc://127.0.0.1:30100
  ```
  注意：**不会使用**这个配置项的地址信息，只使用协议名称sc（即只需要sc:// 开头即可）。
- 对于已经设置过dubbo本身注册中心地址的应用无需再进行配置。

1.2.3.3 虚拟机部署场景接入指南

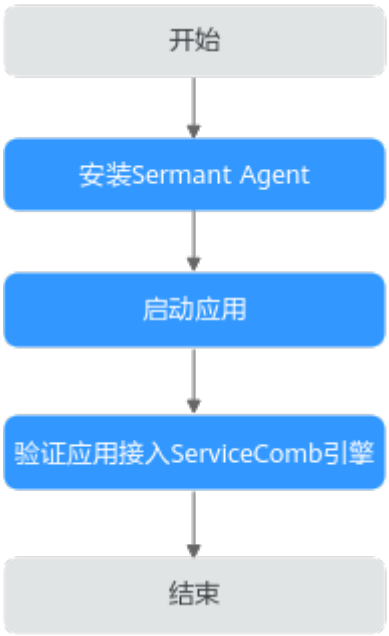
虚机部署的应用可通过Sermant Agent接入到ServiceComb引擎。

前置条件

- 已创建ECS实例， 创建ECS请参考**ECS快速入门**。
- 已安装JDK（版本为1.8及以上版本）并配置环境变量，详情请参考**Java Downloads**。
- 已创建ServiceComb引擎实例，详情请参考**快速创建ServiceComb引擎**。
- ECS与ServiceComb引擎处于相同的VPC网络下。
- Sermant Agent开源版本要求1.0.3及以上。

接入流程

基于ECS将应用接入ServiceComb引擎流程如下：



操作步骤

步骤1 安装Sermant Agent。

1. 登录Linux弹性云服务器。
   请参考**Linux弹性云服务器登录方式概述**选择相应方式登录弹性云服务器。
2. 下载并安装Sermant Agent。
   参考如下命令通过shell脚本方式下载并安装Java Agent。
   wget -O- https://cse-bucket-cn-east-3.obs.cn-east-3.myhuaweicloud.com/javaagent/install.sh | sh

   说明

   安装成功后，脚本将输出安装目录。目录为**当前用户主目录**。

步骤2 启动应用。

在应用的启动参数上添加如下参数，添加启动参数后，待应用启动完成。

```
-javaagent:${HOME}/java-agent/java-agent.jar=appName=default
-Ddynamic_config_serverAddress={CSE_CONFIG_CENTER_ENDPOINTS}
-Dregister.service.address={CSE_REGISTRY_ENDPOINTS}
-Dgrace_rule_enableSpring=false
```

表 1-4 启动参数说明

参数项	说明
appName	agent服务名称，默认default，无需修改。
dynamic_config_serverAddress	ServiceComb引擎配置中心地址，多个地址使用逗号隔开。
register.service.address	ServiceComb引擎注册发现地址。

参数项	说明
grace_rule_enableSpring	目前agent默认开启SpringCloud框架优雅上下线功能，所以Dubbo框架需要手动关闭（设置为fasle），否则可能会存在端口冲突的问题。

说明

- 若需配置APP名称（默认default）、版本（默认1.0.0）请分别使用环境变量-Dservice_meta_application=yourAppName、-Dservice_meta_version=yourVersion进行设置。
- ServiceComb引擎服务注册发现地址与ServiceComb引擎配置中心地址需替换为实际地址，可参考如下方式获取：
  - ServiceComb引擎服务注册发现地址：**获取ServiceComb引擎服务注册发现地址**。
  - ServiceComb引擎配置中心地址：**获取ServiceComb引擎配置中心地址**。

步骤3 验证应用接入ServiceComb引擎。

参考**查看微服务列表**查看您的应用是否已接入ServiceComb引擎。

----结束

1.2.3.4 CCE 部署场景接入指南

1.2.3.4.1 概述

CCE部署的应用可通过Sermant Injector自动挂载Sermant Agent，从而通过Sermant Agent接入到CSE的ServiceComb引擎。当前支持通过模板管理页面和Helm客户端两种方式部署Sermant Injector。

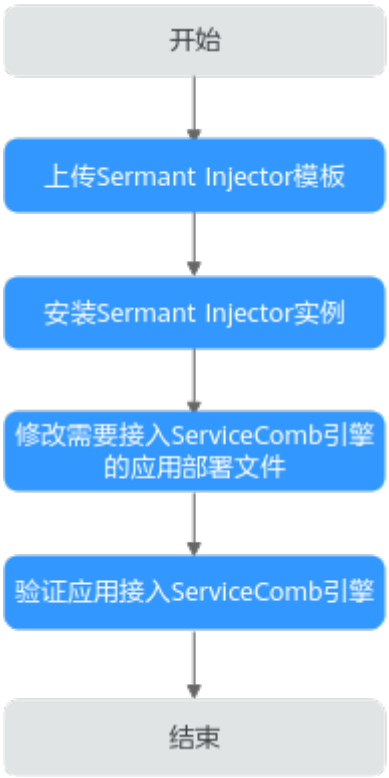
前置条件

- 已创建云容器引擎（CCE），创建CCE请参考**创建CCE集群**。
- CCE集群版本需要大于等于1.15。
- 应用的基础镜像中，需要安装JDK（版本为1.8及以上版本）。
- 已安装kubectl命令，安装kubectl命令请参考**通过kubectl连接集群**中相关操作。
- 已创建ServiceComb引擎实例，详情请参考**快速创建ServiceComb引擎**。
- CCE集群与ServiceComb引擎处于相同的VPC网络下。
- 给Sermant Injector预留200m左右的cpu资源和300Mi左右的memory资源。
- Sermant Injector版本要求1.0.11及以上，Sermant Agent镜像版本要求1.0.9及以上。

1.2.3.4.2 通过模板管理页面部署 Sermant Injector

接入流程

通过模板管理页面部署Sermant Injector将应用接入ServiceComb引擎流程如下：



操作步骤

步骤1 上传Sermant Injector模板。

1. 下载模板。

   Sermant Injector模板版本及下载地址如下表所示：

Sermant Injector版本	Sermant Agent镜像版本	Sermant Agent开源版本	发行时间	获取路径
1.0.11	1.0.9	1.0.6	2023.10.31	sermant-injector-1.0.11.tgz

2. 上传模板。

   登录CCE的控制台，进入集群，在左侧导航栏中选择“模板管理”，单击右上角的“上传模板”，单击“添加文件”，选择已下载的模板包。具体操作请参考**上传模板**。

说明

如果您的Sermant Injector版本低于1.0.11，您需要在上传Sermant Injector模板之前进行以下准备工作：

第一次启动Sermant Injector应用之前，需申请Sermant Injector https证书。

1. 登录已安装kubectl命令的CCE节点。

   请参考**Linux弹性云服务器登录方式概述**选择相应方式登录CCE节点。

2. 登录后，请在**已安装kubectl命令的CCE节点**中执行以下命令申请Sermant Injector https证书：

   wget -O- https://cse-bucket-cn-east-3.obs.cn-east-3.myhuaweicloud.com/javaagent/certificate.sh | sh

   - 该步骤会把证书挂载到cse命名空间中，如果不存在cse命名空间，则会自动创建。
   - 该步骤会向k8s集群申请名为sermant-injector.cse.svc的CertificateSigningRequest，如果之前存在，则会被覆盖。
   - 该步骤会在cse命名空间中创建名为sermant-injector-secret的Secret，如果之前存在，则会被覆盖。
   - 使用Sermant Injector时，如果提示证书失效等证书相关的错误，请重新申请证书并重新安装Sermant Injector实例。

步骤2 安装Sermant Injector实例。

登录CCE控制台，进入集群，在左侧导航栏中选择“模板管理”，在已上传的Sermant Injector模板中，单击“安装”，具体操作请参考**创建模板实例**。关于模板的更多操作请参考**通过模板部署应用** 。

安装时，按需修改配置文件，配置说明如下：

```
agent:
  image:
    # 选填配置，Sermant Agent镜像版本，默认为最新版本。
    version: ${agent.version}
cse:
  config:
    # 必填配置，ServiceComb引擎配置中心地址，获取方式可参考获取ServiceComb引擎配置中心地址。
    endpoints: https://localhost:30110
  registry:
    # 必填配置，注册中心类型，当前支持SERVICE_COMB/NACOS
    type: SERVICE_COMB
    # 必填配置，ServiceComb引擎注册中心地址，获取方式可参考获取ServiceComb引擎注册发现地址。
    endpoints: https://localhost:30100
image:
  # 选填配置，镜像拉取策略：Always(总是拉取)/IfNotPresent(本地有则使用本地镜像,不拉取)/Never(只使用本地镜像，从不拉取)
  pullPolicy: IfNotPresent
  # 必填配置，CCE所在的region，默认为cn-east-3（华东-上海一），具体请参考地区和终端节点。
  region: cn-east-3
injector:
  image:
    # 选填配置，injector镜像版本，默认为最新版本。
    version: ${injector.version}
    # 选填配置，拉取镜像的密钥。
    pullSecrets: default-secret
    # 选填配置，injector实例数，若CCE集群只有一个节点，则需配置为1。
    replicas: 2
```

说明

如果您的Sermant Injector版本低于1.0.11，还需在”injector”节点下，配置“webhooks > caBundle”的值，该值需要在已安装kubectl命令的CCE节点中使用以下命令获取：

kubectl config view --raw --minify --flatten -o jsonpath='{.clusters[].cluster.certificate-authority-data}'

步骤3 修改需要接入ServiceComb引擎的应用部署文件。

- 若是新建的应用需要在“创建无状态工作负载”高级设置页面单击界面右侧的“YAML创建”，具体请参考**通过控制台创建无状态负载**。
- 若是已经部署的应用需要在左侧导航栏中选择“工作负载 > 无状态负载 Deployment”，然后单击应用所属工作负载后的“更多 > 编辑YAML”，具体请参考**编辑YAML**。

在YAML文件中的“spec > template > metadata > labels”层级下加入内容：sermant-injection: enabled。

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: dubbo-registry-provider
  labels:
    app: dubbo-registry-provider
spec:
  replicas: 1
  selector:
    matchLabels:
      app: dubbo-registry-provider
  template:
    metadata:
      labels:
        app: dubbo-registry-provider
        sermant-injection: enabled
    spec:
      containers:
        - name: dubbo-registry-provider
          image: dubbo-registry-provider:1.0.0
```

说明

- 新建的应用在启动时会自动挂载Sermant Agent。
- 已经部署的应用在修改YAML后会自动重启并挂载Sermant Agent。

步骤4 验证应用接入ServiceComb引擎。

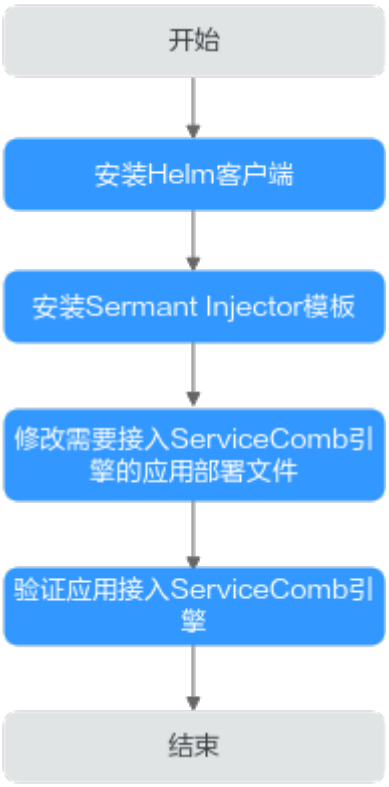
参考**查看微服务列表**查看您的应用（服务名为dubbo.application.name配置的值）是否已接入ServiceComb引擎。

----结束

1.2.3.4.3 通过 Helm 客户端部署 Sermant Injector

接入流程

通过Helm客户端部署Sermant Injector将应用接入ServiceComb引擎流程如下：



操作步骤

步骤1 安装**Helm**客户端。

Helm下载链接：**https://github.com/helm/helm/releases**，请选择合适的版本，本文以helm v3.3.0为例进行演示。

1. 下载**helm-v3.3.0-linux-amd64.tar.gz**。
2. 登录已安装kubectl命令的CCE节点，然后将Helm安装包上传到CCE节点上。
   请参考**Linux弹性云服务器登录方式概述**选择相应方式登录CCE节点。
3. 解压Helm安装包。
   在Helm安装包所在的路径执行命令**tar -zxf helm-v3.3.0-linux-amd64.tar.gz**解压Helm安装包。
4. 将helm移动到系统path所在路径。
   以“/usr/local/bin/helm”为例，在Helm解压包所在的路径执行命令**mv linux-amd64/helm /usr/local/bin/helm**进行移动。
5. 验证安装结果。
   执行命令**helm version**，如果输出下图中的信息，则说明安装成功：

说明

安装用户需要有安装目录和系统path路径的读写权限。

Helm客户端版本需要大于等于3.1。

如果您的Sermant Injector版本低于1.0.11，您还需要在该步骤之后进行以下准备工作：

第一次启动Sermant Injector应用之前，需申请Sermant Injector https证书。

1. 登录已安装kubectl命令的CCE节点。

   请参考**Linux弹性云服务器登录方式概述**选择相应方式登录CCE节点。

2. 登录后，请在**已安装kubectl命令的CCE节点**中执行以下命令申请Sermant Injector https证书：

   wget -O- https://cse-bucket-cn-east-3.obs.cn-east-3.myhuaweicloud.com/javaagent/certificate.sh | sh

   - 该步骤会把证书挂载到cse命名空间中，如果不存在cse命名空间，则会自动创建。
   - 该步骤会向k8s集群申请名为sermant-injector.cse.svc的CertificateSigningRequest，如果之前存在，则会被覆盖。
   - 该步骤会在cse命名空间中创建名为sermant-injector-secret的Secret，如果之前存在，则会被覆盖。
   - 使用Sermant Injector时，如果提示证书失效等证书相关的错误，请重新申请证书并重新安装Sermant Injector实例。

步骤2 安装Sermant Injector模板。

1. 登录**已安装Helm客户端的CCE节点**并执行以下命令下载Sermant Injector模板：

   wget -O- 'https://cse-bucket-cn-east-3.obs.cn-east-3.myhuaweicloud.com/javaagent/sermant-injector-1.0.11.tgz' | tar zx

   Sermant Injector模板版本信息如下：

Sermant Injector版本	Sermant Agent镜像版本	Sermant Agent开源版本	发行时间
1.0.11	1.0.9	1.0.6	2023.10.31

2. 修改配置。

   下载模板后，编辑模板包中的values.yaml文件，按需修改配置。配置说明如下：

```
agent:
  image:
    # 选填配置，Sermant Agent镜像版本，默认为最新版本。
    version: ${agent.version}
cse:
  config:
    # 必填配置，ServiceComb引擎配置中心地址，获取方式可参考获取ServiceComb引擎配置中心地址。
    endpoints: https://localhost:30110
  registry:
    # 必填配置，注册中心类型，当前支持SERVICE_COMB/NACOS
    type: SERVICE_COMB
    # 必填配置，ServiceComb引擎注册中心地址，获取方式可参考获取ServiceComb引擎注册发现地址。
    endpoints: https://localhost:30100
image:
  # 选填配置，镜像拉取策略：Always(总是拉取)/IfNotPresent(本地有则使用本地镜像,不拉取)/Never(只使用本地镜像，从不拉取)
  pullPolicy: IfNotPresent
  # 必填配置，CCE所在的region，默认为cn-east-3（华东-上海一），具体请参考地区和终端节点。
  region: cn-east-3
injector:
  image:
    # 选填配置，injector镜像版本，默认为最新版本。
    version: ${injector.version}
```

```
# 选填配置，拉取镜像的密钥。
pullSecrets: default-secret
# 选填配置，injector实例数，若CCE集群只有一个节点，则需配置为1。
replicas: 2
```

说明

如果您的Sermant Injector版本低于1.0.11，还需在”injector”节点下，配置“webhooks > caBundle”的值，该值需要在已安装kubectl命令的CCE节点中使用以下命令获取：

kubectl config view --raw --minify --flatten -o jsonpath='{.clusters[].cluster.certificate-authority-data}'

3. 安装Sermant Injector模板。

   在Sermant Injector模板包所在路径执行命令**helm install sermant-injector sermant-injector-1.0.11**进行安装。

   说明

   如果需要卸载Sermant Injector，请执行**helm uninstall sermant-injector**命令。

   注意：卸载Sermant Injector后，不会自动挂载Sermant Agent，已挂载的应用如果未重启则不受影响，如果重启，则重启时也不会自动挂载Sermant Agent。

步骤3 修改需要接入ServiceComb引擎的应用部署文件。

- 如果是新建应用，请直接编辑YAML文件。
- 如果是已经部署的应用，请使用**kubectl edit**命令进行编辑YAML文件。

  以Deployment为例，执行命令**kubectl edit deployment {*DeploymentName*} -o yaml -n {*namespace*}**。

  说明

  {*DeploymentName*}需要替换为具体的Deployment名称，{*namespace*}需要替换为Deployment所在的命名空间，若您不清楚DeploymentName或者NAMESPACE，可以使用以下命令查看在CCE中部署的所有Deployment，然后找到您需要接入CSE的应用的DeploymentName与NAMESPACE：

  kubectl get deployments --all-namespaces

  命令执行结果示例如下：

```
[redacted]# kubectl get deployments --all-namespaces
NAMESPACE     NAME                          READY   UP-TO-DATE   AVAILABLE   AGE
[redacted]                                  2/2     2            2           41h
default       service-b                     1/1     1            1           24d
default       service-c                     1/1     1            1           17d
default       spring-demo-a                 1/1     1            1           52s
[redacted]                                  1/1     1            1           28d
[redacted]                                  1/1     1            1           28d
[redacted]                                  1/2     2            1           443d
[redacted]                                  1/2     2            1           443d
[redacted]                                  1/1     1            1           23d
```

在YAML文件中的“spec > template > metadata > labels”层级下加入内容：sermant-injection: enabled。

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: dubbo-registry-provider
  labels:
    app: dubbo-registry-provider
spec:
  replicas: 1
  selector:
    matchLabels:
      app: dubbo-registry-provider
  template:
    metadata:
      labels:
        app: dubbo-registry-provider
        sermant-injection: enabled
    spec:
      containers:
        - name: dubbo-registry-provider
          image: dubbo-registry-provider:1.0.0
```

说明

- 新建的应用在启动时会自动挂载Sermant Agent。
- 已经部署的应用在修改YAML后会自动重启并挂载Sermant Agent。

步骤4 验证应用接入ServiceComb引擎。

参考**查看微服务列表**查看您的应用（服务名为dubbo.application.name配置的值）是否已接入ServiceComb引擎。

----结束

1.2.3.4.4 部署 Dubbo 应用

本章节使用**dubbo-registry-demo**演示接入CSE的ServiceComb引擎。

说明

本章节将使用一个provider服务和一个consumer服务接入ServiceComb引擎。

前提条件

- 已创建CCE集群，创建CCE集群请参考**创建CCE集群**。
- CCE集群版本需要大于等于1.15。
- 已安装kubectl命令，安装kubectl命令请参考**通过kubectl连接集群**中相关操作。
- 已创建ServiceComb引擎实例，详情请参考**快速创建ServiceComb引擎**。
- CCE集群与ServiceComb引擎处于相同的VPC网络下。
- 下载**Sermant-examples**到本地并解压。

- 本地编译构建打包机器环境已安装了Java JDK、Maven，并且能够访问Maven中央库。
- 已在CCE集群上部署Sermant Injector，详情请参考**通过模板管理页面部署Sermant Injector**或者**通过Helm客户端部署Sermant Injector**。

操作步骤

步骤1 打包Sermant-examples。

1. 在“Sermant-examples”根目录下，打开**cmd**命令，执行**mvn clean package**命令，对项目进行打包编译。编译成功后，获取下表中的两个软件包。

   表 1-5 软件包列表

软件包所在目录	软件包名称	说明
Sermant-examples/registry-demo/dubbo-registry-demo/dubbo-registry-consumer/target	dubbo-registry-consumer.jar	服务消费者
Sermant-examples/registry-demo/dubbo-registry-demo/dubbo-registry-provider/target	dubbo-registry-provider.jar	服务生产者

2. 把dubbo-registry-consumer.jar复制到“Sermant-examples/registry-demo/dubbo-registry-demo/deployment/images/consumer”中。
3. 把dubbo-registry-provider.jar复制到“Sermant-examples/registry-demo/dubbo-registry-demo/deployment/images/provider”中。

步骤2 制作镜像。

1. 登录**已安装kubectl命令**且**已部署Sermant Injector**的CCE集群中的节点。
2. 把“Sermant-examples/registry-demo/dubbo-registry-demo”中的deployment文件夹上传至已登录的CCE集群中的节点上。
3. 请参考**使用容器引擎客户端上传镜像**制作docker镜像，其中，使用到的Dockerfile请参考“Sermant-examples/registry-demo/dubbo-registry-demo/deployment/images/consumer”与“Sermant-examples/registry-demo/dubbo-registry-demo/deployment/images/provider”中的Dockerfile文件按需修改。

步骤3 部署dubbo-registry-consumer.yaml与dubbo-registry-provider.yaml。

1. 修改镜像名。

   将已上传deployment文件夹到CCE集群中的节点中的“deployment/k8s/dubbo-registry-consumer.yaml”与“deployment/k8s/dubbo-registry-provider.yaml”中的镜像名修改为您所制作的镜像名。

2. 在已上传deployment文件夹到CCE集群中的节点中的“deployment/k8s”目录下，执行如下命令部署dubbo-registry-consumer.yaml与dubbo-registry-provider.yaml：

   kubectl create -f dubbo-registry-consumer.yaml

   kubectl create -f dubbo-registry-provider.yaml

说明

若需配置APP名称（默认default）、版本（形如a.b.c的格式，其中a、b、c均为数字，默认为1.0.0）请在yaml中增加SERVICE_META_APPLICATION与SERVICE_META_VERSION环境变量进行配置。如下所示：

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: dubbo-registry-provider
  labels:
    app: dubbo-registry-provider
spec:
  replicas: 1
  selector:
    matchLabels:
      app: dubbo-registry-provider
  template:
    metadata:
      labels:
        app: dubbo-registry-provider
        sermant-injection: enabled
    spec:
      containers:
        - name: dubbo-registry-provider
          image: dubbo-registry-provider:1.0.0
          env:
            - name: "SERVICE_META_APPLICATION"
              value: "yourAppName"
            - name: "SERVICE_META_VERSION"
              value: "yourVersion"
```

步骤4 验证应用接入ServiceComb引擎。

参考**查看微服务列表**查看应用（服务名为dubbo-registry-consumer与dubbo-registry-provider）是否已接入ServiceComb引擎。

----结束

2 注册配置中心

2.1 快速创建 Nacos 引擎

本章节指导您根据实际业务需求创建Nacos引擎。

前提条件

Nacos运行于虚拟私有云，创建前，需保证有可用的虚拟私有云和子网。

创建虚拟私有云和子网，请参考**创建虚拟私有云和子网**。

操作步骤

步骤1 进入**购买注册配置中心**页面。

步骤2 左侧导航栏选择“注册配置中心”。

步骤3 在注册配置中心页面，单击“购买注册配置中心”。

步骤4 参考下表设置参数，参数前面带*号的是必须设置的参数。

参数	说明
*计费模式	选择计费方式，目前支持： • 包年/包月 • 按需计费
*企业项目	选择Nacos所在的项目，可在下拉框中搜索和选择需要的企业项目。
*引擎名称	输入Nacos引擎的名称。
*注册配置中心类型	选择“Nacos”。 **说明** Nacos引擎默认多可用区部署，部署在三节点上的，可提供可用区级别容灾能力。
*选择实例数	选择需要购买的容量规格。
版本	只能创建最新版本。

参数	说明
*网络	选择已创建的虚拟私有云及子网，可在下拉框中搜索和选择合适的虚拟私有云和子网。 虚拟私有云可以为您的引擎构建隔离的、用户自主配置和管理的虚拟网络环境。
标签	用于标识云资源，当您拥有相同类型的许多云资源时，可以使用标签按各种维度（例如用途、所有者或环境）对云资源进行分类。 您可以单击“添加标签”，在“添加标签”弹框输入标签键和标签值，添加标签，标签的命名规则请参见**管理标签**。在“添加标签”弹框，可单击“新增标签”同时添加多个标签，也可单击标签后的，删除该标签。
*购买时长	计费模式选择“包年/包月”时需要设置。可设置是否开通自动续费。

步骤5 单击“立即购买”，进入引擎信息确认界面。

步骤6 单击“提交”引擎开始创建，当“运行状态”为“可用”时，引擎创建完成。

----结束

2.2 接入 Nacos 引擎

2.2.1 Spring Cloud 应用快速接入 Nacos 引擎

本章节通过一个demo进行全流程的微服务应用接入Nacos引擎操作演示，帮助您快速了解如何接入Nacos引擎。

本章节将使用一个provider服务和一个consumer服务接入Nacos引擎。

前提条件

- 已创建Nacos引擎，具体操作请参考**快速创建Nacos引擎**。
- 下载github的**demo源码**到本地并解压。
- 本地编译构建打包机器环境已安装了Java JDK、Maven，并且能够访问Maven中央库。

操作步骤

步骤1 登录微服务引擎控制台。

1. 登录**华为云控制台**。
2. 单击，选择区域。
3. 单击左上角，在服务列表选择“微服务引擎 CSE”，进入微服务引擎控制台。

步骤2 获取CSE的Nacos专享版引擎注册发现地址。

1. 在左侧导航栏选择“注册配置中心”，单击创建的Nacos引擎实例。
2. 在“基础信息”页面的“连接信息”区域，获取注册发现地址。

步骤3 修改demo中的配置中心地址和服务注册中心地址和微服务名。

1. 在“bootstrap.properties”中配置Nacos配置中心。

   在下载到本地的demo源码目录中找到“nacos-examples-master\nacos-spring-cloud-example\nacos-spring-cloud-discovery-example\nacos-spring-cloud-consumer-example\src\main\resources”添加“bootstrap.properties”文件，配置Naocs配置中心：

   ```
   spring.cloud.nacos.config.server-addr=XXX.nacos.cse.com:8848 //Nacos的配置中心地址
   spring.cloud.nacos.config.prefix=example  //配置文件名前缀
   spring.cloud.nacos.config.file-extension=properties  //配置文件名后缀
   spring.cloud.nacos.config.group=XXX  //配置文件所属分组，不填默认DEFAULT_GROUP
   spring.cloud.nacos.config.namespace=XXX  //配置文件所属命名空间ID，不填默认public
   ```

   更多配置详情，请参考**Nacos配置参考**。

2. 在“application.properties”中配置Nacos的服务注册发现地址和微服务名。
   - 在下载到本地的demo源码目录中找到“nacos-examples-master\nacos-spring-cloud-example\nacos-spring-cloud-discovery-example\nacos-spring-cloud-consumer-example\src\main\resources\application.properties”文件，配置consumer服务：

     ```
     server.port=8080
     spring.application.name=service-consumer //微服务名
     spring.cloud.nacos.discovery.server-addr= XXX.nacos.cse.com:8848 //Nacos的服务注册发现地址
     spring.cloud.nacos.discovery.group=XXX //微服务所属分组，不填则默认DEFAULT_GROUP
     spring.cloud.nacos.discovery.namespace=XXX //微服务所属命名空间ID，不填则默认public
     spring.cloud.nacos.discovery.cluster-name=XXX //微服务所属集群名称，不填则默认DEFAULT
     ```

   - 在下载到本地的demo源码目录中找到“nacos-examples-master\nacos-spring-cloud-example\nacos-spring-cloud-discovery-example\nacos-spring-cloud-provider-example\src\main\resources\application.properties”文件，配置provider服务：

     ```
     server.port=8070
     spring.application.name=service-provider //微服务名
     spring.cloud.nacos.discovery.server-addr= XXX.nacos.cse.com:8848 //Nacos的服务注册发现地址
     spring.cloud.nacos.discovery.group=XXX //微服务所属分组，不填则默认DEFAULT_GROUP
     spring.cloud.nacos.discovery.namespace=XXX //微服务所属命名空间ID，不填则默认public
     spring.cloud.nacos.discovery.cluster-name=XXX //微服务所属集群名称，不填则默认DEFAULT
     ```

   更多配置详情，请参考**Nacos注册发现**。

步骤4 打包demo源码成jar包。

1. 在demo源码根目录下，打开cmd命令，执行**mvn clean package**命令，对项目进行打包编译。
2. 编译成功后，生成如**表2-1**所示的两个Jar包。

表 2-1 软件包列表

软件包所在目录	软件包名称	说明
\nacos-spring-cloud-consumer-example\target	nacos-spring-cloud-consumer-example-0.2.0-SNAPSHOT.jar	服务消费者

软件包所在目录	软件包名称	说明
\nacos-spring-cloud-provider-example\target	nacos-spring-cloud-provider-example-0.2.0-SNAPSHOT.jar	服务生产者

步骤5 部署Spring Cloud应用。

将微服务Provider和Consumer部署至Nacos引擎所在VPC的ECS节点。

1. 请参考**购买并登录Linux弹性云服务器**在引擎实例所属VPC下创建一台ECS节点并登录。
2. 安装JRE，为服务提供运行环境。
3. 将**步骤4**生成JAR包上传至ECS节点。
4. 执行命令：**java -jar {对应jar包}**，运行生成的JAR包。

步骤6 确认部署结果。

1. **可选:** 在微服务引擎控制台页面，在左侧导航栏选择“注册配置中心”，单击**前提条件**中创建的Nacos引擎。
2. 选择“服务管理”，查看微服务service-consumer和service-provider的实例数量。
   - 若实例数量值不为0，则表示已经成功接入Nacos引擎。
   - 若实例数量为0，或者找不到service-consumer和service-provider服务名，则表示微服务应用接入Nacos引擎失败。

----结束

2.2.2 Spring Cloud Eureka 应用快速接入 Nacos 引擎

本章节通过一个demo进行全流程的Spring Cloud Eureka应用接入Nacos引擎操作演示，帮助您快速了解如何接入Nacos引擎。

本章节将使用一个provider服务和一个consumer服务接入Nacos引擎。

前提条件

- 已创建Nacos引擎，具体操作请参考**快速创建Nacos引擎**。
- 下载github的**demo源码**到本地并解压。
- 本地编译构建打包机器环境已安装了Java JDK、Maven，并且能够访问Maven中央库。

约束限制

- Nacos对Eureka的兼容，主要是兼容Eureka服务端侧的API，将服务侧注册的客户端实例信息进行保存与刷新，因此如果您仅使用Eureka作为注册中心，那么Nacos的诸多特性如命名空间和配置管理是不能使用的。
- 使用Eureka作为客户端，仅能在Nacos的“服务管理”中进行查阅，Eureka服务均使用Nacos的默认属性进行展示，即：
  - 默认命名空间：**public**。
  - 默认分组名称：**DEFAULT_GROUP**。

- 在Nacos中在“服务管理”页面创建服务的“保护阈值”设置的值属于Nacos的特性，无法作用于Eureka服务。

操作步骤

步骤1 登录微服务引擎控制台。

1. 登录**华为云控制台**。
2. 单击，选择区域。
3. 单击左上角，在服务列表选择“微服务引擎 CSE”，进入微服务引擎控制台。

步骤2 获取CSE的Nacos专享版引擎注册发现地址。

1. 在左侧导航栏选择“注册配置中心”，单击创建的Nacos引擎实例。
2. 在“基础信息”页面的“连接信息”区域，获取注册发现地址。

步骤3 修改demo中的注册中心地址和微服务名。

1. 在“application.properties”中配置Nacos的服务注册发现地址和微服务名。
   - 在下载到本地的demo源码目录中找到“eureka-demo-master\eureka-consumer\src\main\resources\application.properties”文件，配置consumer服务：

     ```
     server.port=9001
     spring.application.name=eureka-client-consumer //微服务名
     eureka.client.serviceUrl.defaultZone= XXX.nacos.cse.com:8848/nacos/eureka //Eureka的服务注册发现地址
     eureka.instance.lease-renewal-interval-in-seconds=15 //服务心跳刷新间隔
     eureka.client.registry-fetch-interval-seconds=15 //拉取注册中心间隔（建议与心跳间隔一致）
     ```

   - 在下载到本地的demo源码目录中找到“eureka-demo-master\eureka-provider\src\main\resources\application.properties”文件，配置provider服务：

     ```
     server.port=9000
     spring.application.name=eureka-client-provider //微服务名
     eureka.client.serviceUrl.defaultZone= XXX.nacos.cse.com:8848/nacos/eureka //Eureka的服务注册发现地址
     eureka.instance.lease-renewal-interval-in-seconds=15 //服务心跳刷新间隔
     eureka.client.registry-fetch-interval-seconds=15 //拉取注册中心间隔（建议与心跳间隔一致）
     ```

步骤4 打包demo源码成jar包。

1. 在demo源码根目录下，打开cmd命令，执行**mvn clean package**命令，对项目进行打包编译。
2. 编译成功后，生成如**表2-2**所示的两个Jar包。

表 2-2 软件包列表

软件包所在目录	软件包名称	说明
\eureka-consumer\target	eureka-client-consumer-1.0.0-SNAPSHOT.jar	服务消费者

软件包所在目录	软件包名称	说明
\eureka-provider\target	eureka-client-provider-1.0.0-SNAPSHOT.jar	服务生产者

步骤5 部署SpringCloud应用。

将微服务Provider和Consumer部署至Nacos引擎所在VPC的ECS节点。

1. 请参考**购买并登录Linux弹性云服务器**在引擎实例所属VPC下创建一台ECS节点并登录。
2. 安装JRE，为服务提供运行环境。
3. 将**步骤4**生成JAR包上传至ECS节点。
4. 执行命令：**java -jar {对应jar包}**，运行生成的JAR包。

步骤6 确认部署结果。

1. **可选:** 在微服务引擎控制台页面，在左侧导航栏选择“注册配置中心”，单击**前提条件**中创建的Nacos引擎。
2. 选择“服务管理”，查看微服务eureka-client-consumer和eureka-client-provider的实例数量。
   - 若实例数量值不为0，则表示已经成功接入Nacos引擎。
   - 若实例数量为0，或者找不到eureka-client-consumer和eureka-client-provider服务名，则表示微服务应用接入Nacos引擎失败。

----结束

3 应用网关

3.1 快速创建应用网关

本章节指导您根据实际业务需求创建应用网关。

前提条件

- 应用网关运行于虚拟私有云，创建前，需保证有可用的虚拟私有云和子网。创建虚拟私有云和子网请参考**创建虚拟私有云和子网**。
- 应用网关的运行依赖于独享型负载均衡器，创建应用网关前，需保证已创建了独享型负载均衡器，创建独享型负载均衡器请参考**创建独享型负载均衡器**。

操作步骤

步骤1 进入**购买应用网关**页面。

步骤2 参考下表设置参数，参数前面带*号的是必须设置的参数。

参数	说明
*企业项目	选择应用网关所在的项目。
*引擎名称	输入应用网关名称。
产品版本	目前只支持专业版。
*规格	选择实例规格，当前支持规格有：小型、中型、大型和超大型。
可用区	您可根据实际情况选择可用区。
版本	只能创建最新版本。
*节点数	选择网关的节点数量，节点数量不少于2个。
*网络	选择已创建的虚拟私有云及子网。 虚拟私有云可以为您的引擎构建隔离的、用户自主配置和管理的虚拟网络环境。

参数	说明
*弹性负载均衡	选择已创建的弹性负载均衡。 **说明** 当前仅支持独享型网络型负载均衡器。会使用所选ELB的80、443端口，如果选择的负载均衡器已经占用了部分端口，会创建失败。
标签	用于标识云资源，当您拥有相同类型的许多云资源时，可以使用标签按各种维度（例如用途、所有者或环境）对云资源进行分类。 您可以单击“添加标签”，在“添加标签”弹框输入标签键和标签值，添加标签，标签的命名规则请参见**管理标签**。在“添加标签”弹框，可单击“新增标签”同时添加多个标签，也可单击标签后的，删除该标签。

步骤3 单击“立即购买”，网关开始创建，当“运行状态”为“可用”时，应用网关创建完成。

说明

应用网关创建成功后，“运行状态”为“可用”。查看应用网关状态，请参考**查看应用网关基本信息**。

----结束