企业级分布式应用服务 EDAS

快速入门

快速入门

准备工作

开通 EDAS

在您使用 EDAS 前，请先开通 EDAS。

前提条件

在使用阿里云 EDAS 之前，请确保您已经注册了阿里云账号并完成实名认证。

操作步骤

在浏览器中打开企业级分布式应用服务 EDAS 产品主页。

在产品主页上单击 **1-5 个应用实例免费开通**。

在 EDAS 的售卖页中选择购买的配置和规格。

您可以选择**按量付费**或**包年包月**两种付费方式。购买价格请参见价格说明。

按量付费：

当前环境：默认为**公共云**，不可设置。

系列：默认为**标准版**，按量付费仅支持此唯一系列，不可设置。

应用实例数：默认为**按应用实例数收费**，，即按您实际使用的应用实例数收费

，不可设置。

包年包月：

当前环境：默认为**公共云**，不可设置。

系列：包括**标准版**、**专业版**和**铂金版**。不同版本系列的功能差异请参见**产品系列**。

付费模式：默认为**包年包月**，不可设置。

说明：包年包月支持与按量混合付费的模式。正常情况下使用包年包月的付费模式，当使用的应用实例数超过订单规格限制后，多出的实例将自动使用按量付费模式。详情请参见**价格说明**中**超出购买规格之外的实例付费方式**章节。

应用实例数：根据您的实际需求选择。

- 若您刚开始使用 EDAS 并且不清楚实际使用需求，建议您先选择 1~5 个实例，后续可以根据业务需求进行扩容。
- 若您明确实际使用需求，选择时请参见**可购区间说明**，避免所选配置与规格不匹配。

订购时长：选择包年包月的时长。

按照页面提示登录阿里云账号并完成产品购买操作。

开通 EDAS 服务后，单击**管理控制台/控制台**，选择账号进入 EDAS 管理界面。

说明：如果您初次使用 EDAS，登录控制台后，会弹出**安全授权提示**。单击**立即授权**，进入**云资源访问授权**页面，单击**同意授权**。

创建资源

在将应用部署到 EDAS 前，需要创建相关资源。

为了帮助您更方便的使用相关资源，下面简单介绍一下 EDAS 中涉及的资源的基本概念。

基本概念

资源	定义	使用场景
云服务器（ECS）	阿里云提供的一种可弹性伸缩的计算服务。	您在 EDAS 上部署的应用将部署在 ECS 实例上。
集群	在 EDAS 中，集群是用于部署应用的实例的集合。	当您有较高的安全要求时，创建集群时需要指定命名空间和 VPC 网络。
命名空间	在具体地域（Region）中，命名空间用于实现资源和服务的隔离。	当您对资源有较高的安全要求时，需要创建命名空间。
专有网络（VPC）	在阿里云中构建的隔离的网络环境。	当您对网络有较高的安全要求时，建议创建 VPC、添加 ECS 实例。
负载均衡（SLB）	阿里云提供的对多台 ECS 进行流量分发的负载均衡服务。	应用实例之间通过 SLB 实现负载均衡。应用需要通过公网负载均衡的 IP 和端口对外开放。

说明：建议您使用 Chrome 浏览器进行控制台操作。

创建 VPC

出于安全考虑，建议您创建 VPC 并在 VPC 中创建集群并部署应用。如果想要在 ECS 集群中部署应用，且对网络没有安全要求，可以跳过此步骤，使用经典网络即可。

登录**专有网络控制台**，创建 VPC。详细步骤请参考**搭建专有网络**。

说明：如果您尚未开通专有网络 VPC 服务，需要点击**开通专有网络服务**。进入 VPC 售卖页面，阅读并勾选**专有网络 VPC 开通协议**，然后单击**立即开通**。

将 VPC 同步到 EDAS 控制台中。

登录 EDAS 控制台，在左侧导航栏中选择**资源管理** > **VPC** 。

在 **VPC** 页面中选择创建的 VPC 所在地域（如：华东1），在页面右上角单击**同步 VPC** 。

确认创建的 VPC 已同步到 EDAS 控制台。

创建 ECS 实例

注意：如果需要使用且创建了 VPC，则创建 ECS 实例时需要选择已创建的 VPC。

登录 EDAS 控制台。

在左侧导航栏中选择**资源管理** > **ECS**。

选择您要创建 ECS 实例的地域及命名空间（可选），在页面右上角单击**创建实例**。

在 ECS 购买页面，根据您的需要，参考**创建 ECS 实例**，完成 ECS 的规格配置和支付。

各应用配置规格说明：

集群类型	配置要求	挂载数据盘	其他说明
ECS 集群	CPU≥1 vCPU 内存≥1 GiB	支持	- JVM 是多线程应用，当所需内存不足时容易产生问题。故请避免生产环境所创建的集群中添加的 ECS 实例使用最低配置。 - 当集群添加完 ECS 实例后，请设置 Swap 交换分区。以免日志记录等进程占用较大内存资源，造成操作系统无法运行。

容器服务的 Kubernetes 集群	CPU≥1 vCPU 内存≥2 GiB	支持	当集群添加完 ECS 实例后，请设置 Swap 交换分区。以免日志记录等进程占用较大内存资源，造成操作系统无法运行。

说明：实例创建后，会添加到默认的命名空间和集群中。

创建命名空间

命名空间用于实现资源和服务的隔离。如果您没有相关要求，可以跳过此步骤。使用默认命名空间即可。

登录 EDAS 控制台。

在左侧导航栏中选择**应用管理** > **命名空间**。

在命名空间列表页面选择**地域（Region）**，然后在页面右上角单击**创建命名空间**。

在创建命名空间对话框中，设置**命名空间名称**（必选）、**命名空间 ID**（必选），选择命名空间类型为**普通命名空间**，选择性输入描述信息。然后单击**确定**。

注意：命名空间 ID 的前缀已根据所选地域（归属地域）而确定，不可编辑，只能设置自定义部分。

创建集群

EDAS 中目前主要包含 ECS 集群和容器服务 Kubernetes 集群。

- ECS 集群在 EDAS 控制台创建，创建完成后即可使用。
- 容器服务 Kubernetes 集群在容器服务-Kubernetes 控制台创建，创建完成后需要导入到 EDAS 控制台才能使用。

本节介绍如何创建 ECS 集群。如果您需要使用容器服务 Kubernetes 集群，请参见**创建并导入容器服务 Kubernetes 集群**。

创建 ECS 集群

登录 EDAS 控制台，在左侧导航栏中选择**资源管理** > **集群**。

在**集群**页面选择**地域**和**命名空间**（可选），然后页面右上角单击**创建集群**。

在**创建集群**对话框中设置集群参数，然后单击**创建**。

配置	说明
集群名称	集群的名称，仅支持字母、数字、下划线（_）、中划线（-）和英文句号（.），不能超过 64 个字符。
集群归属	包括**阿里云**和**非阿里云**。创建 ECS 集群选择**阿里云**，**非阿里云**用于创建混合云集群。
集群类型	选择 **ECS**。
集群网络类型	包括 **VPC 网络**或**经典网络**。选择 **VPC 网络**，**经典网络**仅适用于无网络安全需求的场景。
VPC 网络	选择您所创建的 VPC。
命名空间	默认使用创建集群前选择的地域中的命名空间，不可配置。如果没有选择命名空间，则为该地域下的默认命名空间，显示为地域名，如：*cn-hangzhou*。

为集群添加 ECS 实例。

ECS 集群中添加实例目前有两种方式：一种是直接导入实例，无需镜像转化；另一种将使用 EDAS 官方镜像重装系统，重装后，实例中的所有数据都会被删除，并且需要设置新的实例登录密码。

满足以下任一情况，则不能在集群中直接导入实例：

- 2017年12月1日之前创建的实例；
- 经典网络的机器导入经典网络的集群；
- 实例没有运行，如实例已经停止或者在启动中或者停止中；
- 如 Windows 机器或者不支持简单 shell 命令的机器，如果您创建了 Windows 系统的机器，在 EDAS 中不能成功导入并实现转化；
- 非 ECS 集群间导入实例。

添加 ECS 实例的步骤如下：

在**集群**页面单击之前创建的集群名称。

在集群详情页面右上角单击**添加 ECS 实例**。

在**选择集群和已有云服务器实例**页面的实例列表中，选择添加实例的方式和 ECS 实例，然后单击**下一步**。

i. **导入 ECS**：命名空间和导入集群不可配置，从该地域的默认命名空间及集群中导入。
ii. **从已有集群选择**：在该地域下，选择命名空间以及集群，然后将页面左侧的 ECS 实例添加到右侧。

说明：如果没有符合条件的实例，在页面右上角单击**创建 ECS 实例**，跳转到阿里云官网 ECS 购买页面，购买并创建新的 ECS 实例。详情请参考**创建 ECS 实例**。

在**准备导入**页面，查看选择的实例信息。

i. 如果实例能够直接导入则单击**确认并导入**。
ii. 如果实例需要转化导入，则勾选**同意对以上需要转化的实例进行转化，并已知转化后原有系统中的数据将会丢失**。然后输入转化后系统 root 用户的新密码。完成设置后单击**确认并导入**。

在**进行导入**页签上可以看到实例的导入状态。

i. 能够直接导入的集群实例：在**进行导入**页面可以快速显示实例的导入状态，当显示**实例转移成功**时说明实例已成功导入。单击**点击返回集群详情页**，当实例状态显示为**运行中**时也说明实例导入集群成功。
ii. 实例需转化的集群实例：在**进行导入**页面会显示实例的导入状态为**正在转化中，该转化操作预计耗时五分钟**。在导入完成前单击**点击返回集群详情页**，健康检查状态会显示为**转化中**并且会显示转化进度百分比，当导入完成时，健康检查的状态会显示为**运行中**。

同步 SLB 到 EDAS

登录负载均衡管理控制台，创建 SLB。详细步骤请参考**创建负载均衡实例**。

将 SLB 同步到 EDAS 控制台中。

登录 EDAS 控制台，在左侧导航栏中选择**资源管理** > **SLB**。

在 **SLB** 页面选择创建的 SLB 所在地域（如：*华东1*），在页面右上角单击**同步 SLB**。

确认创建的 SLB 已同步到 EDAS 控制台。

将 Spring Cloud 应用托管到 EDAS

本文档将以一个 Spring Cloud 应用（包含一个服务提供者和一个服务消费者）为例，介绍如何在本地开发（添加依赖和所需配置）、测试，并部署到 EDAS 中，实现应用的服务注册与发现，以及消费者对提供者的调用。

> 如果您完全不了解 Spring Cloud，只有简单的 Spring 和 Maven 基础，详细阅读本文后，您将了解如何通过 Spring Cloud Alibaba Nacos Discovery 实现 Spring Cloud 应用的服务注册与发现，以及实现消费者对提供者的调用。
>
> 如果您熟悉 Spring Cloud 中的服务注册组件（如 Eureka、Consul 和 ZooKeeper），但尚未使用过 Spring Cloud Alibaba 的服务注册组件 Nacos Discovery，那么您只需要将这些服务注册组件的依赖和配置替换成 Spring Cloud Alibaba Nacos Discovery，无需修改任何代码。
>
> Spring Cloud Alibaba Nacos Discovery 同样实现了 Spring Cloud Registry 的标准接口与规范，和您之前使用 Spring Cloud 接入服务注册与发现的方式基本一致。
>
> 如果您熟悉如何使用开源版本的 Spring Cloud Alibaba Nacos Discovery 实现 Spring Cloud 应用的服务注册与发现，那么您可以将应用直接部署到 EDAS（详情请参见将应用部署到 EDAS），即可使用到 EDAS 提供的商业版服务注册与发现的能力。

EDAS 服务注册中心提供了开源 Nacos Server 的商业化版本，使用开源版本 Spring Cloud Alibaba Nacos Discovery开发的应用可以直接使用 EDAS 提供的商业版服务注册中心。

商业版的 EDAS 服务注册中心，与 Nacos、Eureka 和 Consul 相比，具有以下优势：

- 共享组件，节省了你部署、运维 Nacos、Eureka 或 Consul 的成本。
- 在服务注册和发现的调用中都进行了链路加密，保护您的服务，无需再担心服务被未授权的应用发现。
- EDAS 服务注册中心 与 EDAS 其他组件紧密结合，为您提供一整套的微服务解决方案，包括 环境隔离、平滑上下线、灰度发布等。

本地开发

本地开发中主要描述开发中涉及的关键信息，如果您想了解完整的 Spring Cloud 应用程序，可下载 service-provider 和 service-consumer。

准备工作

在开始开发前，请确保您已经完成以下工作：

- 下载 Maven 并设置环境变量。
- 下载最新版本的 Nacos Server。
- 按以下步骤启动 Nacos Server。
    i. 解压下载的 Nacos Server 压缩包
    ii. 进入nacos/bin目录，启动 Nacos Server。
        - Linux/Unix/Mac 系统：执行命令sh startup.sh -m standalone。
        - Windows 系统：双击执行startup.cmd文件。

创建服务提供者

在本地创建一个服务提供者应用工程，添加依赖，开启服务注册与发现功能，并将注册中心指定为 Nacos Server。

操作步骤

创建一个 Maven 工程，命名为nacos-service-provider。

在 pom.xml 文件中添加依赖。

以 Spring Boot 2.1.4.RELEASE 和 Spring Cloud Greenwich.SR1 为例，依赖如下：

```
 <parent>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-parent</artifactId>
<version>2.1.4.RELEASE</version>
<relativePath/>
</parent>

<dependencies>
<dependency>
<groupId>com.alibaba.cloud</groupId>
<artifactId>spring-cloud-starter-alibaba-nacos-discovery</artifactId>
<version>2.1.0.RELEASE</version>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-web</artifactId>
</dependency>
</dependencies>

<dependencyManagement>
```

```
<dependencies>
<dependency>
<groupId>org.springframework.cloud</groupId>
<artifactId>spring-cloud-dependencies</artifactId>
<version>Greenwich.SR1</version>
<type>pom</type>
<scope>import</scope>
</dependency>
</dependencies>
</dependencyManagement>

<build>
<plugins>
<plugin>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-maven-plugin</artifactId>
</plugin>
</plugins>
</build>
```

示例中使用的版本为 Spring Cloud Greenwich ，对应 Spring Cloud Alibaba 版本为 2.1.0.RELEASE。

- 如果使用 Spring Cloud Finchley 版本，对应 Spring Cloud Alibaba 版本为 2.0.0.RELEASE。

  如果使用 Spring Cloud Edgware 版本，对应 Spring Cloud Alibaba 版本为 1.5.0.RELEASE。

  说明：Spring Cloud Edgware 版本的生命周期即将在 2019 年 8 月结束，不推荐使用这个版本开发应用。

在src\main\java下创建 Packagecom.aliware.edas。

在 Packagecom.aliware.edas中创建服务提供者的启动类ProviderApplication，并添加如下代码：

其中@EnableDiscoveryClient注解表明此应用需开启服务注册与发现功能。

```
package com.aliware.edas;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.client.discovery.EnableDiscoveryClient;

@SpringBootApplication
@EnableDiscoveryClient
public class ProviderApplication {

public static void main(String[] args) {
SpringApplication.run(ProviderApplication.class, args);
}
```

```
}
```

在 Packagecom.aliware.edas中创建EchoController，指定 URL mapping 为 {/echo/{String}}，指定 HTTP 方法为 GET，方法参数从 URL 路径中获得，回显收到的参数。

```
package com.aliware.edas;

import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class EchoController {
@RequestMapping(value = "/echo/{string}", method = RequestMethod.GET)
public String echo(@PathVariable String string) {
return string;
}
}
```

在src\main\resources路径下创建文件application.properties，在application.properties中添加如下配置，指定 Nacos Server 的地址。

其中127.0.0.1为 Nacos Server 的地址。如果您的 Nacos Server 部署在另外一台机器，则需要修改成对应的 IP 地址。如果有其它需求，可以参考**配置项参考**在application.properties文件中增加配置。

```
spring.application.name=service-provider
server.port=18081
spring.cloud.nacos.discovery.server-addr=127.0.0.1:8848
```

结果验证

执行nacos-service-provider中ProviderApplication的main函数，启动应用。

登录本地启动的 Nacos Server 控制台http://127.0.0.1:8848/nacos（本地 Nacos 控制台的默认用户名和密码同为 nacos），在左侧导航栏中选择**服务管理** > **服务列表**，可以看到服务列表中已经包含了service-provider，且在详情中可以查询该服务的详情。

创建服务消费者

这一节中不仅会演示服务注册的功能，还将说明 Nacos 服务发现与 RestTemplate 和 FeignClient 这两个客户端是如何配合使用的。

操作步骤

创建一个 Maven 工程，命名为nacos-service-consumer。

在pom.xml中添加依赖。

```
<parent>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-parent</artifactId>
<version>2.1.4.RELEASE</version>
<relativePath/>
</parent>

<dependencies>
<dependency>
<groupId>com.alibaba.cloud</groupId>
<artifactId>spring-cloud-starter-alibaba-nacos-discovery</artifactId>
<version>2.1.0.RELEASE</version>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
<groupId>org.springframework.cloud</groupId>
<artifactId>spring-cloud-starter-openfeign</artifactId>
</dependency>
</dependencies>

<dependencyManagement>
<dependencies>
<dependency>
<groupId>org.springframework.cloud</groupId>
<artifactId>spring-cloud-dependencies</artifactId>
<version>Greenwich.SR1</version>
<type>pom</type>
<scope>import</scope>
</dependency>
</dependencies>
</dependencyManagement>

<build>
<plugins>
<plugin>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-maven-plugin</artifactId>
</plugin>
</plugins>
</build>
```

在src\main\java下创建 Packagecom.aliware.edas。

在 Packagecom.aliware.edas中配置 RestTemplate 和 FeignClient。

在 Packagecom.aliware.edas 中创建一个接口类EchoService，添加@FeignClient注解，并配置对应的 HTTP URL 地址及 HTTP 方法。

```
package com.aliware.edas;

import org.springframework.cloud.openfeign.FeignClient;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

@FeignClient(name = "service-provider")
public interface EchoService {
@RequestMapping(value = "/echo/{str}", method = RequestMethod.GET)
String echo(@PathVariable("str") String str);
}
```

在 Packagecom.aliware.edas 中创建启动类ConsumerApplication并添加相关配置。

i. 使用@EnableDiscoveryClient注解启用服务注册与发现。
ii. 使用@EnableFeignClients注解激活 FeignClient。
iii. 添加@LoadBalanced注解将 RestTemplate 与服务发现集成。

```
package com.aliware.edas;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.client.discovery.EnableDiscoveryClient;
import org.springframework.cloud.client.loadbalancer.LoadBalanced;
import org.springframework.cloud.openfeign.EnableFeignClients;
import org.springframework.context.annotation.Bean;
import org.springframework.web.client.RestTemplate;

@SpringBootApplication
@EnableDiscoveryClient
@EnableFeignClients
public class ConsumerApplication {

@LoadBalanced
@Bean
public RestTemplate restTemplate() {
return new RestTemplate();
}

public static void main(String[] args) {
SpringApplication.run(ConsumerApplication.class, args);
}
}
```

在 Packagecom.aliware.edas 中创建类TestController以演示和验证服务发现功能。

```
package com.aliware.edas;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.client.RestTemplate;

@RestController
public class TestController {

@Autowired
private RestTemplate restTemplate;
@Autowired
private EchoService echoService;

@RequestMapping(value = "/echo-rest/{str}", method = RequestMethod.GET)
public String rest(@PathVariable String str) {
return restTemplate.getForObject("http://service-provider/echo/" + str,
String.class);
}

@RequestMapping(value = "/echo-feign/{str}", method = RequestMethod.GET)
public String feign(@PathVariable String str) {
return echoService.echo(str);
}

}
```

在src\main\resources路径下创建文件application.properties，在application.properties中添加如下配置，指定 Nacos Server 的地址。

其中127.0.0.1:8848为 Nacos Server 的地址。如果您的 Nacos Server 部署在另外一台机器，则需要修改成对应的地址。如果有其它需求，可以参考配置项参考在application.properties文件中增加配置。

```
spring.application.name=service-consumer
server.port=18082
spring.cloud.nacos.discovery.server-addr=127.0.0.1:8848
```

结果验证

执行nacos-service-consumer 中 ConsumerApplication 的 main 函数，启动应用。

登录本地启动的 Nacos Server 控制台 http://127.0.0.1:8848/nacos（本地 Nacos 控制台的默认用

户名和密码同为 nacos ），在左侧导航栏中选择**服务管理** > **服务列表**，可以看到服务列表中已经包含了service-consumer，且在详情中可以查询该服务的详情。

本地测试

在本地测试消费者对提供者的服务调用结果。

Linux/Unix/Mac 系统：执行curl http://127.0.0.1:18082/echo-rest/rest-rest和curl http://127.0.0.1:18082/echo-feign/feign-rest。

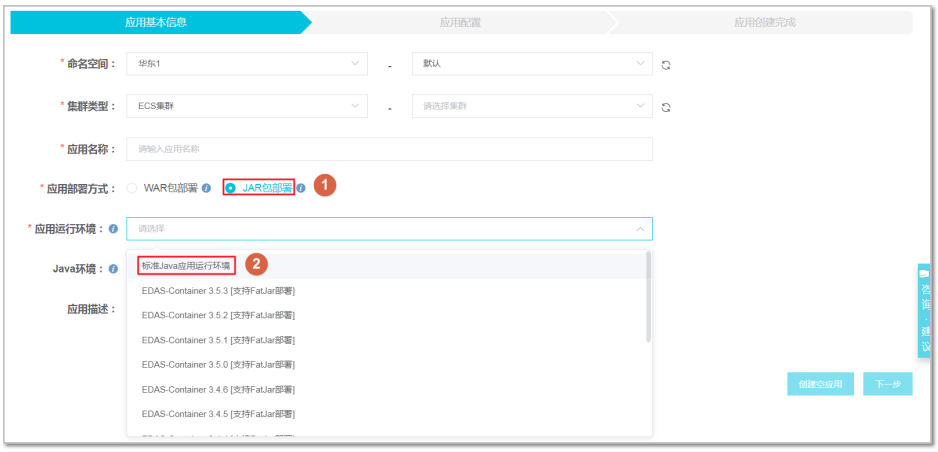
```
➜  ~ curl http://127.0.0.1:18082/echo-rest/rest-test
rest-test%
➜  ~ curl http://127.0.0.1:18082/echo-feign/feign-test
feign-test%
```

Windows系统：在浏览器中输入http://127.0.0.1:18082/echo-rest/rest-rest和http://127.0.0.1:18082/echo-feign/feign-rest。

将应用部署到 EDAS

当在本地完成应用的开发和测试后，便可将应用程序打包并部署到 EDAS。您可以根据您的实际情况选择将 Spring Cloud 应用可以部署到 ECS 集群、容器服务 Kubernetes集群或 EDAS Serverless。部署应用的详细步骤请参见部署应用概述。

说明：第一次部署建议通过控制台部署，且如果使用 JAR 包部署，在创建应用时**应用运行环境** 务必选择 **标准 Java 应用运行环境**。



EDAS 服务注册中心提供了商业化版本 Nacos Server。当您将应用部署到 EDAS 的时候，EDAS 会通过优先级更高的方式去设置 Nacos Server 服务端地址和服务端口，以及 namespace、access-key、secret-key、context-path 信息。您无需进行任何额外的配置，原有的配置内容可以选择保留或删除。

结果验证

在部署完成之后，在 EDAS 控制台左侧导航栏选择**微服务管理** > **服务查询**，在**服务查询**页面选择**地域**和**命名空间**，然后通过搜索service-provider和service-consumer查询您部署的应用。

配置项参考

配置项	Key	默认值	说明
服务端地址	spring.cloud.nacos.discovery.server-addr	无	Nacos Server 启动监听的 IP 地址和端口。
服务名	spring.cloud.nacos.discovery.service	${spring.application.name}	给当前的服务命名。
网卡名	spring.cloud.nacos.discovery.network-interface	无	当 IP 未配置时，注册的 IP 为此网卡所对应的 IP 地址。如果此项也未配置，则默认取第一块网卡的地址。
注册的 IP 地址	spring.cloud.nacos.discovery.ip	无	优先级最高
注册的端口	spring.cloud.nacos.discovery.port	-1	默认情况下不用配置，系统会自动探测。
命名空间	spring.cloud.nacos.discovery.namespace	无	常用场景之一是不同环境的注册的区分隔离，例如开发测试环境和生产环境的资源（如配置、服务）隔离等。
Metadata	spring.cloud.nacos.discovery.metadata	无	使用 Map 格式配置，用户可以根据自己的需要自定义一些和服务相关的元数据信息。
集群	spring.cloud.nacos.discovery.cluster-name	DEFAULT	配置成 Nacos 集群名称。
接入点	spring.cloud.nacos.discovery.enpoint	UTF-8	地域的某个服务的入口域名，通过此域名可以动态地拿到服务端地址，此配置在部署到 EDAS 时无需填写。
是否集成 Ribbon	ribbon.nacos.enabled	true	一般不需要修改

更多关于 Spring Cloud Alibaba Nacos Discovery 的信息请参见开源版本的 Spring Cloud Alibaba Nacos Discovery 文档。

FAQ

使用其他版本

示例中使用的 Spring Cloud 版本为 Greenwich，对应的 Spring Cloud Alibaba 版本为 2.1.0.RELEASE。Spring Cloud Finchley 对应的 Spring Cloud Alibaba 版本为 2.0.0.RELEASE，Spring Cloud Edgware 对应的 Spring Cloud Alibaba 版本为 1.5.0.RELEASE。

说明：Spring Cloud Edgware 版本的生命周期即将在 2019 年 8 月结束，不推荐使用这个版本开发应用。

从 ANS 迁移

EDAS 注册中心在服务端对 ANS 和 Nacos 的数据结构做了兼容，在同一个命名空间下，且 Nacos 未设置 group 时，Nacos 和 ANS 客户端可以互相发现 对方注册的服务。

将 Dubbo 应用托管到 EDAS

您可以将 Dubbo 微服务应用托管到 EDAS，使用 EDAS 提供的共享组件、企业级安全加固和完整的微服务解决方案，帮助您节省运维成本、提升安全性和开发效率。本文将介绍如何在本地通过 XML 配置的方式开发一个 Dubbo 微服务示例应用（包含一个服务提供者 Provider 和一个服务消费者 Consumer）并部署到 EDAS。

为什么托管到 EDAS

将 Dubbo 应用托管到 EDAS，您只需要关注 Dubbo 应用自身的逻辑，无需再关注注册中心、配置中心和元数据中心的搭建和维护，托管后还可以使用 EDAS 提供的弹性伸缩、限流降级、监控及微服务治理能力，而且整个托管过程对您来说是完全透明的，不会增加您的理解成本和开发成本。

准备工作

在开始开发前，请确保您已经完成以下工作：

下载 Maven 并设置环境变量。

下载最新版本的 Nacos Server。

按以下步骤启动 Nacos Server。

i. 解压下载的 Nacos Server 压缩包
ii. 进入nacos/bin目录，启动 Nacos Server。
- Linux/Unix/Mac 系统：执行命令sh startup.sh -m standalone。
- Windows 系统：双击执行startup.cmd文件。

创建服务提供者

在本地创建一个提供者应用工程，添加依赖，配置服务注册与发现，并将注册中心指定为 Nacos。

创建 Maven 项目并引入依赖。

使用 IDE（如 IntelliJ IDEA 或 Eclipse）创建一个 Maven 项目。

在pom.xml文件中添加 **dubbo**、**dubbo-registry-nacos** 和 **nacos-client** 依赖。

注意：Dubbo 要求使用 2.7.3 及以上版本。

```
<dependencies>

<dependency>
<groupId>org.apache.dubbo</groupId>
<artifactId>dubbo</artifactId>
<version>2.7.3</version>
</dependency>

<dependency>
<groupId>org.apache.dubbo</groupId>
<artifactId>dubbo-registry-nacos</artifactId>
<version>2.7.3</version>
</dependency>

<dependency>
<groupId>com.alibaba.nacos</groupId>
<artifactId>nacos-client</artifactId>
<version>1.1.1</version>
</dependency>

</dependencies>
```

开发 Dubbo 服务提供者。

Dubbo 中服务都是以接口的形式提供的。

在src/main/java路径下创建一个 package com.alibaba.edas。

在com.alibaba.edas下创建一个 接口（interface） IHelloService，里面包含一个 SayHello 方法。

```
package com.alibaba.edas;

public interface IHelloService {
String sayHello(String str);
}
```

在com.alibaba.edas下创建一个类IHelloServiceImpl，实现此接口。

```
package com.alibaba.edas;

public class IHelloServiceImpl implements IHelloService {
public String sayHello(String str) {
return "hello " + str;
}
}
```

配置 Dubbo 服务。

在 src/main/resources路径下创建 provider.xml文件并打开。

在provider.xml中，添加 Spring 相关的 XML Namespace（xmlns） 和 XML Schema Instance（xmlns:xsi），以及 Dubbo 相关的 Namespace（xmlns:dubbo） 和 Scheme Instance（xsi:schemaLocation）。

```
<beans xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:dubbo="http://dubbo.apache.org/schema/dubbo"
xmlns="http://www.springframework.org/schema/beans"
xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-4.3.xsd
http://dubbo.apache.org/schema/dubbo http://dubbo.apache.org/schema/dubbo/dubbo.xsd">
```

在 provider.xml 中将接口和实现类暴露成 Dubbo 服务。

```
<dubbo:application name="demo-provider"/>

<dubbo:protocol name="dubbo" port="28082"/>

<dubbo:service interface="com.alibaba.edas.IHelloService" ref="helloService"/>

<bean id="helloService" class="com.alibaba.edas.IHelloServiceImpl"/>
```

在provider.xml中将注册中心指定为本地启动的 Nacos Server。

```
<dubbo:registry address="nacos://127.0.0.1:8848" />
```

i. *127.0.0.1* 为 Nacos Server 的地址。如果您的 Nacos Server 部署在另外一台机器，则需要修改成对应的 IP 地址。当将应用部署到 EDAS 后，无需做任何修改，注册中心会替换成EDAS上的注册中心的地址。
ii. *8848* 为 Nacos Server 的端口号，不可修改。

启动服务。

在 com.alibaba.edas中创建类 Provider，并按下面的代码在 Provider 的 main 函数中加载 Spring Context，将配置好的 Dubbo 服务暴露。

```
package com.alibaba.edas;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class Provider {
public static void main(String[] args) throws Exception {
ClassPathXmlApplicationContext context = new ClassPathXmlApplicationContext(new String[]
{"provider.xml"});
context.start();
System.in.read();
}
}
```

执行 Provider 的 main 函数，启动服务。

登录 Nacos 控制台 http://127.0.0.1:8848，在左侧导航栏中单击**服务列表** ，查看提供者列表。可以看到服务提供者里已经包含了 com.alibaba.edas.IHelloService，且可以查询该服务的**服务分组**和**提供者 IP**。

创建服务消费者

在本地创建一个消费者应用工程，添加依赖，添加订阅服务的配置。

创建 Maven 项目并引入依赖。

步骤和创建服务提供者相同，不再赘述。具体步骤可参考创建服务提供者的相关步骤。

开发 Dubbo 服务。

Dubbo 中服务都是以接口的形式提供的。

在src/main/java路径下创建 package com.alibaba.edas。

在com.alibaba.edas下创建一个接口（interface） IHelloService，里面包含一个 SayHello 方法。

说明：通常是在一个单独的模块中定义接口，服务提供者和服务消费者都通过 Maven 依赖来引用此模块。本文档为了简便，服务提供者和服务消费者分别创建两个完全一模一样的接口，实际使用中不推荐这样使用。

```
package com.alibaba.edas;

public interface IHelloService {
String sayHello(String str);
}
```

配置 Dubbo 服务。

在 src/main/resources路径下创建 consumer.xml文件并打开。

在consumer.xml中，添加 Spring 相关的 XML Namespace（xmlns） 和 XML Schema Instance（xmlns:xsi），以及 Dubbo 相关的 Namespace（xmlns:dubbo） 和 Scheme Instance（xsi:schemaLocation）。

```
<beans xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:dubbo="http://dubbo.apache.org/schema/dubbo"
xmlns="http://www.springframework.org/schema/beans"
xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-4.3.xsd
http://dubbo.apache.org/schema/dubbo http://dubbo.apache.org/schema/dubbo/dubbo.xsd">
```

在 consumer.xml 中添加如下配置，订阅 Dubbo 服务。

```
<dubbo:application name="demo-consumer"/>

<dubbo:registry address="nacos://127.0.0.1:8848"/>

<dubbo:reference id="helloService" interface="com.alibaba.edas.IHelloService"/>
```

启动、验证服务。

在com.alibaba.edas下创建类 Consumer，并按下面的代码在 Consumer 的 main 函数中加载 Spring Context，订阅并消费 Dubbo 服务。

```
package com.alibaba.edas;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import java.util.concurrent.TimeUnit;

public class Consumer {
public static void main(String[] args) throws Exception {
ClassPathXmlApplicationContext context = new ClassPathXmlApplicationContext(new String[]
{"consumer.xml"});
context.start();
while (true) {
try {
TimeUnit.SECONDS.sleep(5);
IHelloService demoService = (IHelloService)context.getBean("helloService");
String result = demoService.sayHello("world");
System.out.println(result);
} catch (Exception e) {
e.printStackTrace();
}
}
}
}
```

执行 Consumer 的 main 函数，启动服务。

验证创建结果。

i. 启动后，可以看到控制台不断地输出 hello world，表明服务消费成功。

登录 Nacos 控制台 http://127.0.0.1:8848，在左侧导航栏中单击**服务列表** ，再在**服务列表**页面选择**调用者列表**。

可以看到包含了 com.alibaba.edas.IHelloService，且可以查看该服务的**服务分组**和**调用者 IP**。

部署到 EDAS

本地使用 Nacos 作为注册中心的应用可以直接部署到 EDAS 中，无需做任何修改，注册中心会被自动替换为 EDAS 上的注册中心。

您可以根据实际需求选择部署的集群类型（主要为 ECS 集群或容器服务 Kubernetes 集群）和部署途径（控制台或工具），详情请参见部署应用概述。

如果您使用控制台部署，在部署前，需要在本地应用程序中完成以下操作：

在pom.xml文件中添加以下打包插件的配置。

Provider

```
<build>
<plugins>
<plugin>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-maven-plugin</artifactId>
<executions>
<execution>
<goals>
<goal>repackage</goal>
</goals>
<configuration>
<classifier>spring-boot</classifier>
<mainClass>com.alibaba.edas.Provider</mainClass>
</configuration>
</execution>
</executions>
</plugin>
</plugins>
</build>
```

Consumer

```
<build>
<plugins>
<plugin>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-maven-plugin</artifactId>
<executions>
<execution>
<goals>
<goal>repackage</goal>
</goals>
<configuration>
<classifier>spring-boot</classifier>
<mainClass>com.alibaba.edas.Consumer</mainClass>
</configuration>
</execution>
</executions>
</plugin>
</plugins>
</build>
```

执行 mvn clean package 将本地的程序打成 JAR 包。

在将 Dubbo 微服务应用部署到 EDAS 后，您可以使用 EDAS 进行微服务治理，详情请参见Dubbo 服务治理

。

更多信息

您还可以通过 Spring Boot 的方式开发 Dubbo 应用，详情请参见使用 Spring Boot 开发 Dubbo 应用。

如果您熟悉 Dubbo，仅想体验如何托管到 EDAS，可以使用 Alibaba Cloud Toolkit 创建一个 Demo 示例，然后部署到 EDAS，详情请参见使用 Cloud Toolkit 创建并部署 Apache Dubbo 应用样例工程。

如果您使用了edas-dubbo-extension，请参见通过 edas-dubbo-extension 将 Dubbo 应用托管到 EDAS。由于edas-dubbo-extension的方式不能使用 EDAS 提供的相关功能，如 Dubbo 服务治理，所以不推荐使用，建议您迁移到 Nacos。

部署多语言微服务应用

随着 Python、Node.js 等语言的快速发展，多语言微服务应用越来越多。EDAS 能够通过服务网格支持部署多语言微服务应用，且提供应用托管和服务发现、链路追踪、负载均衡等服务治理能力。本文将通过一个示例介绍如何使用 EDAS 部署多语言微服务应用。

背景信息

应用从最初的单体架构演变到目前的微服务架构，在带来便利的同时也大大增加了服务部署、运维的复杂度。而微服务本身可以是任意语言开发的，在部署多语言服务后，如何对多语言的微服务提供通用的链路追踪、服务发现、负载均衡等能力，一种做法是提供多语言的 SDK，一种是服务网格。SDK 对应用有侵入性，而服务网格在无侵入性的同时也能提供服务发现、负载均衡、链路追踪等能力，EDAS 对多语言的支持正是采用的后者——服务网格。

服务网格是致力于解决服务间通讯的基础设施层。它负责在现代云原生应用程序的复杂服务拓扑中可靠地传递请求，通常是通过一组轻量级网络代理，与应用程序部署在一起来实现，而无需感知应用程序本身。

服务网格的价值和接入成本

价值

现在的服务基本上都是多实例部署的，天然需要服务发现、负载均衡、链路跟踪能力，部署应用时根据代码填写服务的服务名和服务端口，EDAS 的服务网格会依据这两个信息自动完成服务注册，而用户在代码中使用http://服务名:服务端口发起访问时，服务网格会从请求中解析出服务名，完成服务发现、负载均衡和链路追踪。

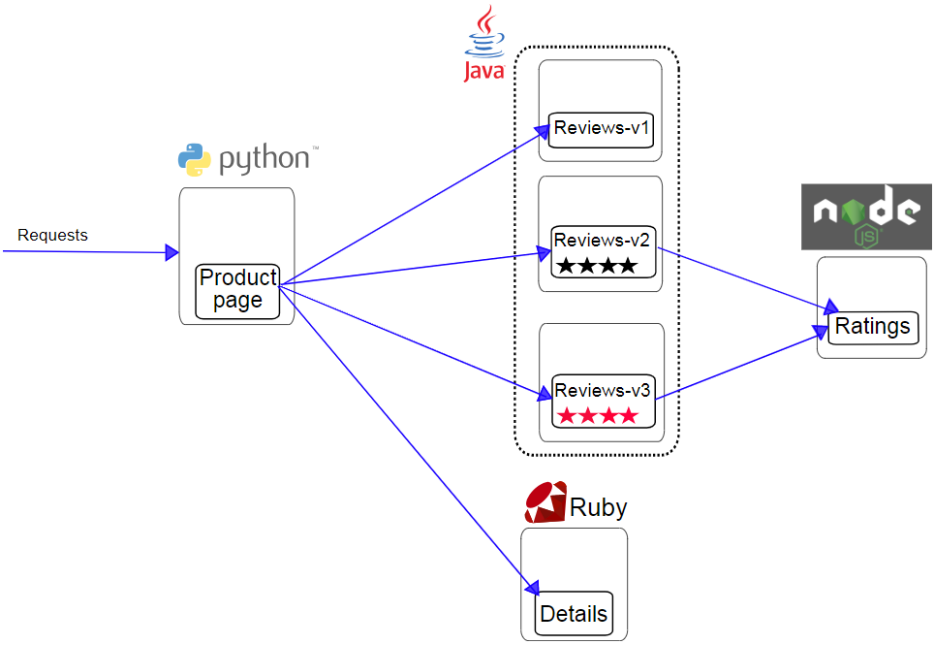
接入成本

应用 A 提供了 test 服务，通常我们会采用域名或者 IP 的方式访问，如*http://test.com:8080/*或者*http://xx.xx.xx.xx:8080*。使用服务网格后，原部署服务的实例就可以抽象成为一个服务，仍以 test 为例，假设服务名与端口号分别为*my-test-service* 和*8080*，在服务代码中将调用格式修改为http://服务名:服务端口，如*http://my-test-service:8080*，然后将服务 *my-test-service* 以镜像方式部署时启用服务网格，并设置服务名（*my-test-service*）和端口号（*9080*），就完成了接入。

示例场景说明

示例应用 BookInfo 模仿在线书店的一个分类，显示一本书的信息。页面上会显示一本书的描述，书籍的细节（ISBN、页数等），以及关于这本书的一些评论。

Bookinfo 是一个异构应用，几个微服务应用是由不同的语言编写的。这些服务构成了一个有代表性的服务网格的例子：它由多个服务、多个语言构成，并且 reviews 服务具有多个版本。微服务架构如下：



Bookinfo 应用包含四个单独的服务：

productpage ：为 Python 服务，会调用 details 和 reviews 两个服务，用来生成页面。同时，productpage 还包含登入和登出功能。

details ：为 Ruby 服务，包含了书籍的信息。

reviews ：为 Java 服务，包含了书籍相关的评论。它还会调用 ratings 服务。

ratings ：为 Node.js 服务，包含了由书籍评价组成的评级信息。包含 3 个版本：

- v1 版本不会调用 ratings 服务。
- v2 版本会调用 ratings 服务，并使用 1 到 5 个黑色星形图标来显示评分信息。
- v3 版本会调用 ratings 服务，并使用 1 到 5 个红色星形图标来显示评分信息。

前提条件

如果想在 EDAS 部署多语言微服务应用，您需要先完成以下工作：

将示例应用制作成镜像，并上传到阿里云镜像仓库。上传镜像请参见上传本地镜像。

示例应用下载地址：BookInfo Sample。

创建并导入容器服务 Kubernetes 集群。

- 创建集群时，需要开通**公网访问**功能，即勾选**使用 EIP 暴露 API Server**。
- 需要确保 Kubernetes 版本为 1.8.4，且该集群中未安装任何服务网格组件。

开通 Tracing Analysis 服务（公测期，暂时免费）。

如提前没有开通，在集群内安装服务网格时会提示未开通 Tracing Analysis 服务，需要开通。

说明：

本文以 BookInfo 的示例向您介绍如何部署，而实际使用时您要部署的是您自己的应用（而且很可能是微服务架构），所以，在部署您还需要对您的服务做以下规划和开发：

如果部署多个服务，请保证每个服务的服务名是唯一的。因为在 EDAS 中同一个命名空间中服务名不能重复，以保证能被其它服务调用。

如果部署的多个服务存在调用关系，在调用者的服务中需要修改按以下格式修改调用代码：

http://<提供者的服务名>:<提供者的服务端口>

步骤一：为容器服务 K8s 集群安装服务网格

登录 EDAS 控制台。

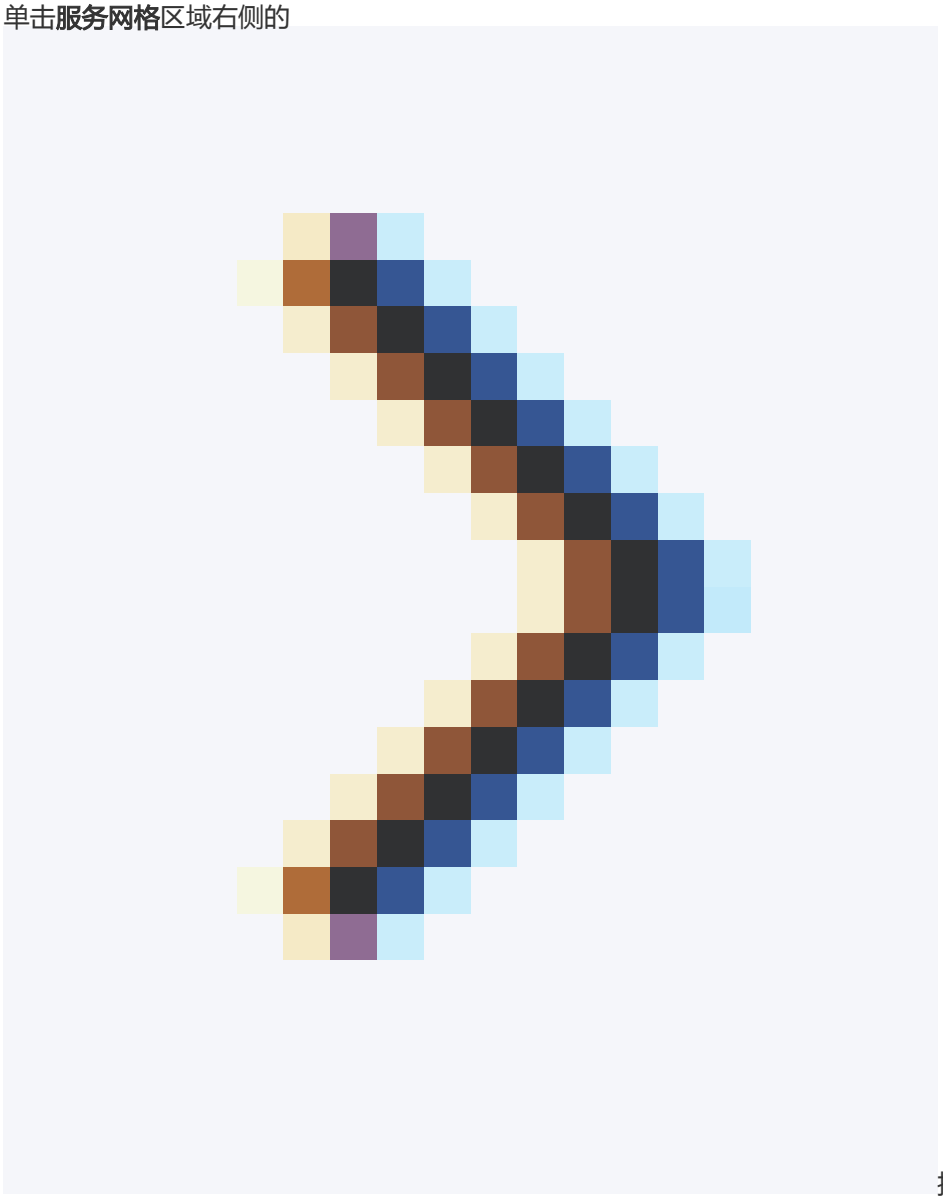
在左侧导航栏选择**资源管理** > **集群**。

在**集群**页面选择您导入容器服务 K8s 集群的**地域**和**命名空间**，单击**容器服务 K8s 集群**页签，再单击之前导入的容器服务 K8s 名称。

在集群详情页面最下方的**服务网络**区域单击**安装服务网格。**

在弹出的确认对话框中单击**确定。**

确认对话框会提示**执行中**，然后消失，集群详情页面顶部显示**服务网格安装中。**等待约 1 分钟，当**服务网格安装中**消失后，说明安装完成。

单击**服务网格**区域右侧的  按钮，展开服务网格区域，查看**组件版本**、**组建健康状况**和**跟踪采样率。**

步骤二：部署应用

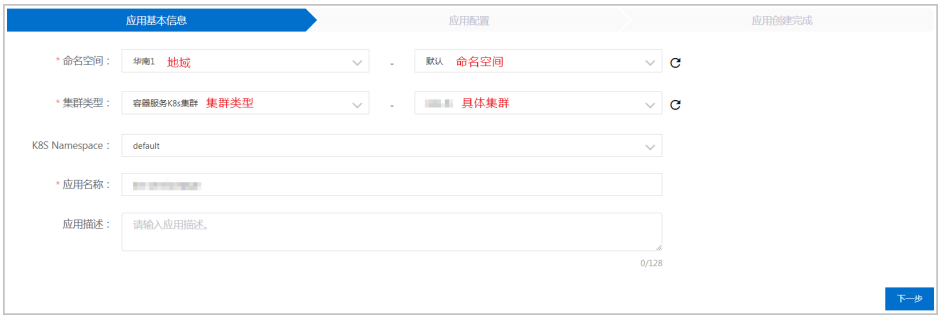
您需要将场景示例中的几个服务以应用的形式分别部署到 EDAS 中。下面介绍单个服务的部署过程。

说明：多语言应用目前仅支持使用**镜像**部署。

登录 EDAS 控制台。

在左侧导航栏选择**应用管理** > **应用列表**，在**应用列表**页面右上角单击**创建应用**。

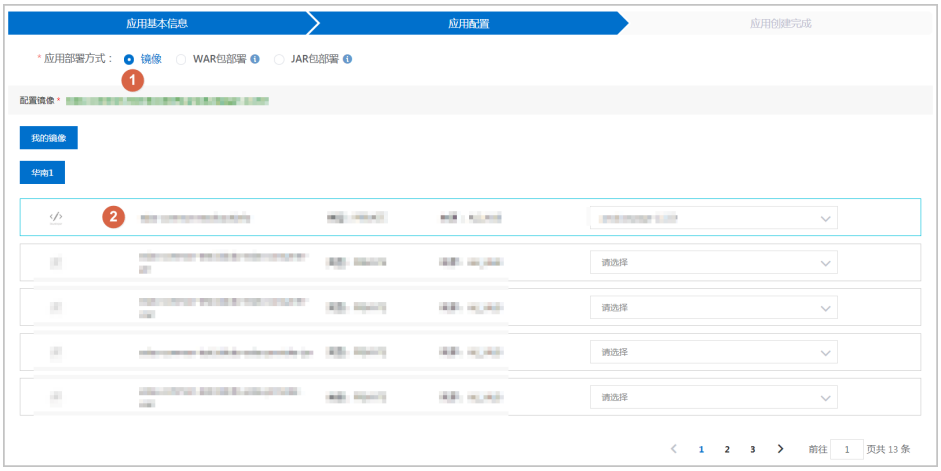
在**应用基本信息**页面设置应用参数，然后单击**下一步**。



基本参数说明：

- **命名空间**：在左侧下拉选择框选择地域；在右侧下拉选择框选择命名空间，如果不做选择命名空间则设置为**默认**。
- **集群类型**：在左侧下拉选择框中选择集群类型为 **容器服务 K8S 集群**，右侧下拉选择框内选择具体的集群。
- **K8S Namespace**：K8S Namespace 通过将系统内部的对象分配到不同的 Namespace 中，形成逻辑上分组的不同项目、小组或用户组，便于不同的分组在共享使用整个集群的资源的同时还能被分别管理。
    - **default**：没有其他命名空间的对象的默认命名空间。
    - **kube-system**：系统创建的对象的命名空间。
    - **kube-public**：此命名空间是自动创建的，并且可供所有用户（包括未经过身份验证的用户）读取。
    - **istio-system**：部署服务网格后系统自动创建的命名空间。
- **应用名称**：输入应用名称。
- **应用描述**：填写应用的基本情况。

在**应用配置**页面勾选**应用部署方式**为**镜像**，并选择您之前上传的的服务镜像。



设置 Pod。

Pod 是应用最小的部署单元。应用可以有多个 Pod，在负载均衡下，请求会被随机分配给某个 Pod 处理。

设置 **Pod 总数**。

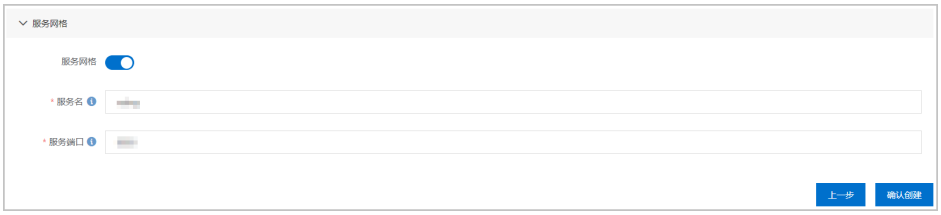
Pod 在运行失败或出现故障时，可以自动重启或者快速迁移，保证应用的高可用。有状态应用如果使用了持久化存储，能保存实例数据；无状态应用重新部署时不保存实例数据。您最多可以设置 Pod 总数为 50 。

设置**单 Pod 资源配额**。

系统默认不做配额限制，即单 Pod 的 CPU 和 Memory 显示为 0。如果需要限制配额，请填设置数字。

启动命令、**环境变量**、**持久化存储**、**本地存储**和**应用生命周期管理**都为可选设置。需要您需要设置，请参见在容器服务 K8S 集群中部署应用（镜像）中的参数说明。

设置服务网格。



服务网格：启用服务网格。

服务名：应用提供的服务名，要和应用代码中的服务名一致，以保证服务能成功注册和被调用。本示例 Demo 中的 4 个服务的**服务名**分别为 *productpage*、*details*、*ratings* 和 *reviews*。如果部署您自己的服务，请填写服务代码中的服务名。

服务端口：应用提供的服务端口，要和应用代码中的服务端口一致，以保证服务能成功注册和被调用。本示例 Demo 中的 4 个服务的**服务端口**号均为 *9080*。如果部署您自己的服务，请填写服务代码中的服务端口。

设置完成后，单击**确认创建**。

应用创建可能需要几分钟，创建过程中，可以通过查看应用变更跟踪创建的过程。

在容器服务 Kubernetes 集群中应用创建完成即部署完成。创建完成后，返回应用详情页面查看实例部署信息中 Pod 状态若为**正常运行**则说明应用部署成功。

（可选）步骤三：启用公网访问

如果服务需要被公网访问，需要开启并设置公网访问。本示例中，主服务 productpage 需要设置公网访问。

在应用详情页**应用设置区域**启用**公网访问**。

设置**公网访问路径**。

说明：

服务名是在部署时设置的，不可修改。

服务端口也是在部署时设置的，可以修改。不过在本示例中统一为**9080**，不可修改。

公网 IP 和**公网端口**是在为容器服务 K8s 集群安装服务网格时 EDAS 通过 SLB 为集群自动分配的，不可修改。

公网访问路径：由于在安装服务网格时，系统为该集群分配了公网 IP 和端口，所以该集群中部署的服务需要使用路径区分。在本示例中，仅主服务需要被公网访问，需要分别设置 productpage 的访问、登入、登出服务的访问路径：分别为：

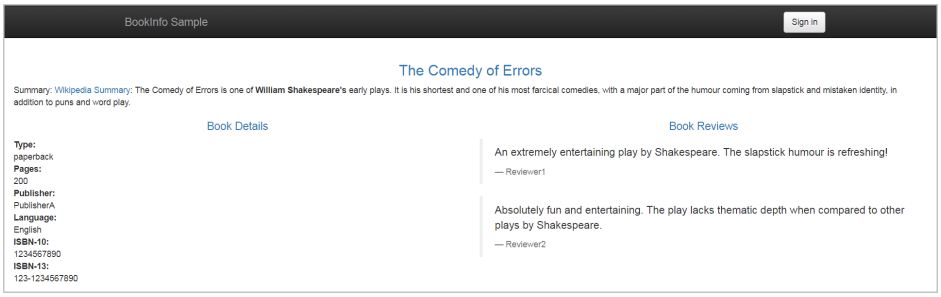
- 访问：/productpage
- 登入：/login
- 登出：/logout

说明：如果部署您自己的服务，请按服务代码填写实际的访问路径。

结果验证

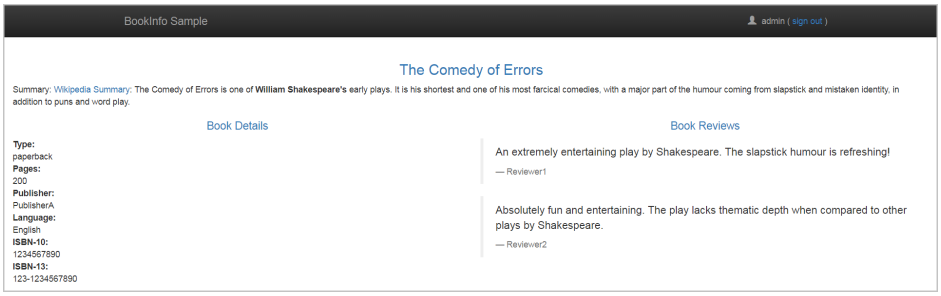
在将 4 个应用都部署完成后，访问主服务，页面上会显示一本书的描述，书籍的细节（ISBN、页数等），以及关于这本书的一些评论。同时还能进行登入、登出操作。

在浏览器的地址栏中输入http://<公网 IP>:<公网端口><主服务路径>，如 *http://xxx.xxx.xxx.xxx:80/productpage*，然后回车。



页面可以访问，且 **Book Details**（书籍的细节）和 **Book Reviews**（评论）两个区域可以正常显示。说明主服务 productpage、子服务 datails 和 子服务 reviews 都正常。

单击 **Sign in**，在弹出的对话框中输入用户名和密码（均为 *admin*），然后单击 **Sign in**，登录成功。



登录后，单击 **Sign out**，可以正常登出。

后续操作

在服务部署完成后，您还可以通过 EDAS 控制台监控服务的运行状态，在遇到问题是可以通过日志进行诊断。

- 监控：您可以通过 EDAS 集成的 Tracing Analysis （**需要开通，暂时免费**）在应用详情页监控应用，查看调用链信息。具体功能（应用总览、应用详情和接口调用）的详细使用方式请参见 Tracing Analysis 中的相关文档。
- 日志：您可以通过 EDAS 的日志管理功能查看容器的标准日志和业务日志。详情请参见实时日志和业

务日志。

快速创建 Demo 微服务应用

快速创建 Demo 微服务应用简介

在 EDAS 中创建应用时，选择 VPC、集群及实例等资源的过程比较复杂。为了提升创建应用的体验，EDAS 提供了一种简化的应用创建方式，即根据您已有资源（包括 VPC、ECS、集群）的情况，针对性的引导您完成创建应用的过程。

应用的创建流程如下：

选择要部署的集群类型。

ECS 和容器服务 Kubernetes 集群均支持创建 Spring Cloud、Dubbo 和 HSF 应用。

选择应用运行环境。

- Java：适用于部署 Spring Cloud 应用或通过 JAR 包部署 Dubbo 应用。
- Tomcat：适用于使用 WAR 包部署 Dubbo 应用。
- EDAS-Container：适用于使用 WAR 包部署 HSF 应用。

选择应用部署包。

可以选择本地部署包、官方 Demo。

- 本地部署包：您在本地开发并打包的应用。
- 官方 Demo：EDAS 为您提供了 Spring Cloud、Dubbo 和 HSF 的微服务应用 Demo，包含服务端应用（Provider）和客户端应用（Consumer）。

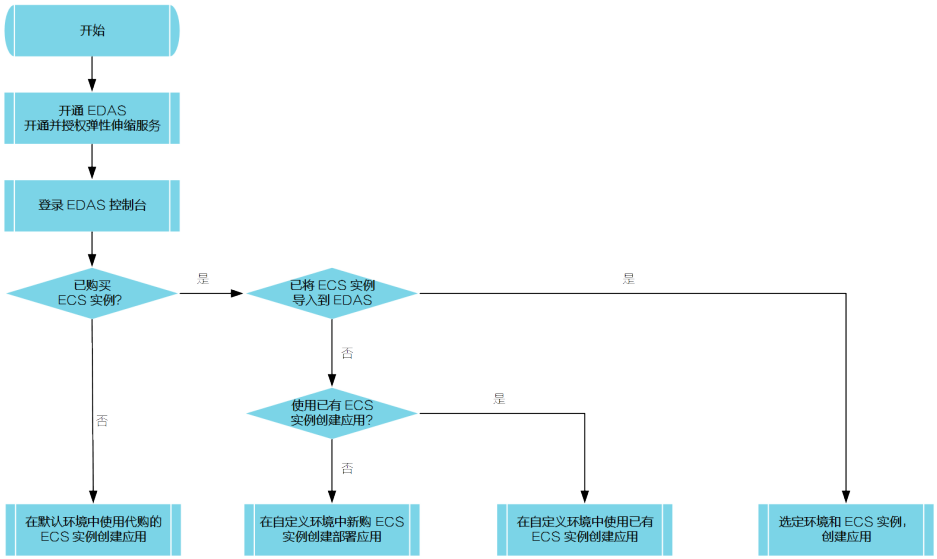
选择部署应用的资源。

- 可以在默认环境（在当前命名空间下的默认的 VPC 网络和默认集群）代购默认 ECS 实例（２Ｃ４Ｇ）。
- 可以在自定义环境（指定命名空间、VPC 和 集群）中代购的推荐规格的 ECS 实例或已有 ECS 实例。

设置应用高级参数。

创建并部署应用。

在 ECS 集群中创建应用的引导流程如下：



在默认环境中使用代购的 ECS 实例部署微服务应用

为方便您快速开始使用 EDAS，EDAS 为您准备了基于不同应用框架（Spring Cloud、Dubbo 和 HSF）的微服务应用 Demo，您可以将应用 Demo 快速部署到 EDAS 中。本文介绍如何在默认环境（默认的 VPC 和集群）中代购 ECS 实例快速部署 HSF 微服务应用 Demo。

前提条件

在默认环境中代购的 ECS 实例中部署应用前，请先完成以下工作：

- 开通 EDAS 服务。
- 开通并授权弹性伸缩服务，详情请参见开通并授权服务。

操作步骤

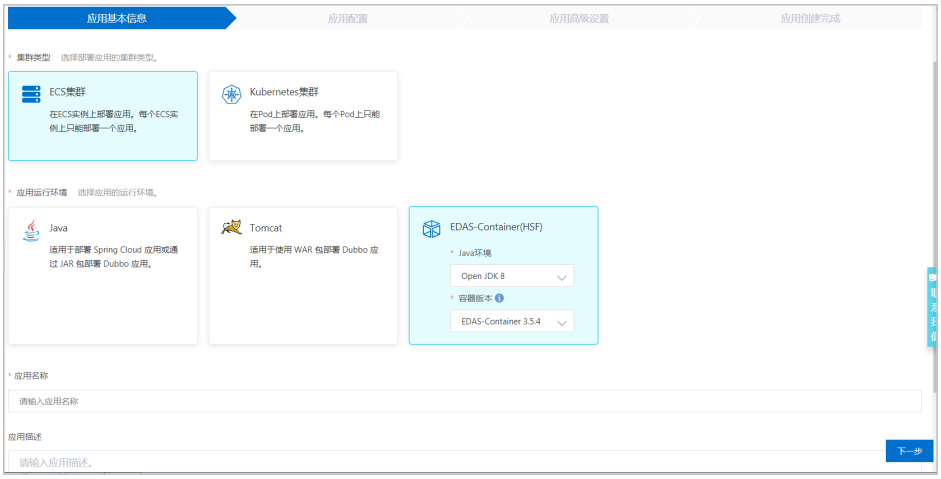
微服务应用 Demo 中都会包含一个服务提供者和服务消费者。下面以服务提供者为例介绍如何创建，在创建完服务提供者之后，请按照操作步骤再创建服务消费者。

创建服务提供者的步骤如下：

登录 EDAS 控制台。

在**概览**页面**应用数**上方单击**创建新应用**。

在**创建应用**页面的**应用基本信息**页签中选择**集群类型**、**应用运行环境**，输入**应用名称**及**应用描述**（可选），然后单击**下一步**。



- **集群类型**：选择 **ECS 集群**。
- **应用运行环境**：选择 **EDAS-Container(HSF)**。
  - **Java 环境**：选择 **Open JDK 8**
  - **容器版本**：选择 **EDAS-Container 3.5.4**
- **应用名称**：输入应用名称。

在**应用配置**页签中选择**部署包来源**、**Demo 类型**和**实例**，然后单击**下一步**。

- **部署包来源**：选择**官方 Demo**。
- **Demo 类型**：在下拉列表中选择 **HSF 客户端应用**。

实例：选择**默认**，即**在当前命名空间下的默认的 VPC 网络和默认集群中购买默认（2 核**

4G）新实例。

- 当前命名空间
  - 如果在应用列表的入口处已经选择了命名空间，那么表示使用选择的命名空间。
  - 如果在应用列表的入口处没有选择命名空间，或者从概览页作为入口，那么使用默认命名空间，
- 默认 VPC：首先选择 VPC 中的默认 VPC 实例。如果没有，则选择 EDAS 中名为 **edas-default-vpc** 的默认 VPC。如果没有，则创建名为 **edas-default-vpc** 的 VPC 并使用。如果创建 VPC 失败（可能会因为 VPC 限额等原因创建失败），则随机选择您已有的一个 VPC 实例。
- 默认集群：选择所选择的命名空间和 VPC 下的“默认集群”。如果没有名为“默认集群”的集群，则在所选的命名空间和 VPC 下创建名为**默认集群**的集群，作为默认集群。

在**应用高级配置**页签中输入**版本**和**应用健康检查**（可选），然后单击**创建应用**。

- **版本**：EDAS 配置使用当前时间戳作为版本，格式如 **yyyymmdd:hhmmss**。您也可以设置其它版本标识。
- **应用健康检查**：设置健康检查的 URL，用来检测应用是否健康运行。

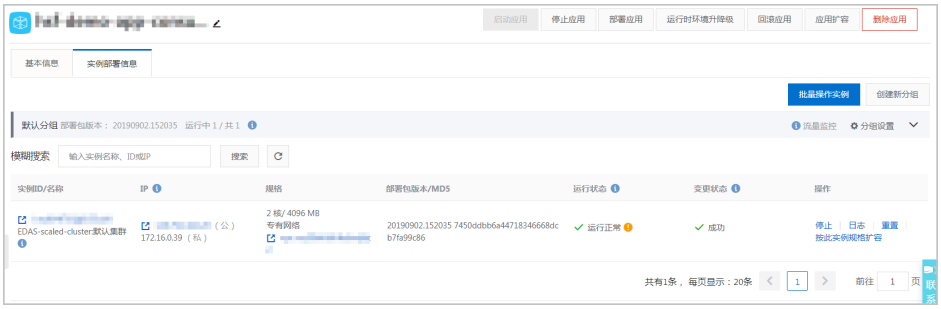
在**应用创建完成**页签确认应用基本信息、应用配置和应用高级设置，确认无误后，单击**确定创建应用**。

结果验证

创建应用后，EDAS 将为该应用自动创建一条表更记录，您可以通过变更详情查看该应用创建的进度和状态。应用创建成功后，变更记录显示**执行成功**。

应用创建成功后，返回该应用的**基本信息**页面，查看应用的相关信息，应该和创建时保持一致。

单击**实例部署信息**查看该应用的状态，运行状态应为**运行正常**，变更状态（即第一次创建、部署）应为**成功**。同时，显示代购的 ECS 实例的 ID、IP、规格及 VPC 信息。



在自定义环境中代购的 ECS 实例上部署微服务应用

为方便您快速开始使用 EDAS，EDAS 为您准备了基于不同应用框架（Spring Cloud、Dubbo 和 HSF）的微服务应用 Demo，您可以将应用 Demo 快速部署到 EDAS 中。本文介绍如何在自定义环境中（指定 VPC 和集群）中代购 ECS 实例快速部署 Spring Cloud 微服务应用 Demo。

前提条件

在默认环境中代购的 ECS 实例中部署应用前，请先完成以下工作：

- 开通 EDAS 服务。
- 开通并授权弹性伸缩服务，详情请参见开通并授权服务。

操作步骤

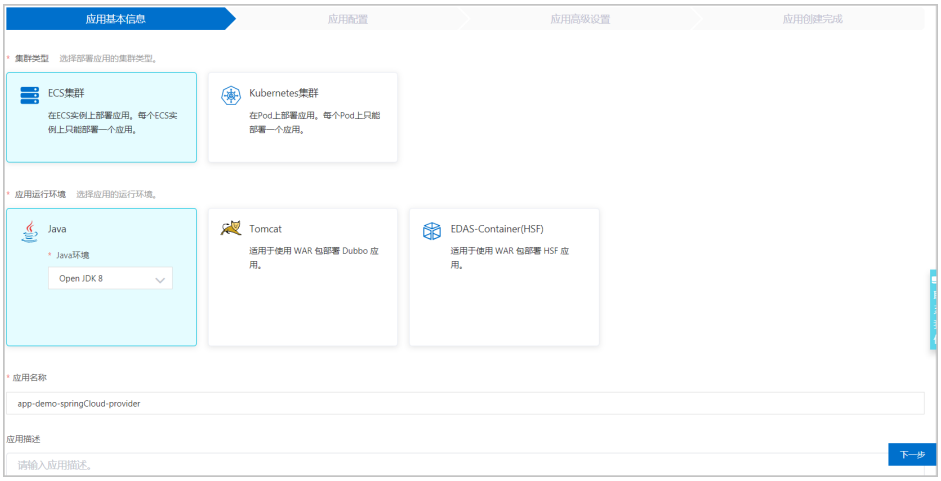
微服务应用 Demo 中都会包含一个服务提供者和服务消费者。下面以服务提供者为例介绍如何创建，在创建完服务提供者之后，请按照操作步骤再创建服务消费者。

创建服务提供者的步骤如下：

登录 EDAS 控制台。

在**概览**页面**应用数**上方单击**创建新应用**。

在**创建应用**页面的**应用基本信息**页签中选择**集群类型**、**应用运行环境**，输入**应用名称**及**应用描述**（可选），然后单击**下一步**。



- **集群类型**：选择 **ECS 集群**。

  应用运行环境：针对不同应用框架，EDAS 提供了不同的应用运行环境，您可以根据您的应用框架选择，本示例中选择 **Java**。

  Java 环境：选择 **Open JDK 8**

- **应用名称**：输入应用名称，如 **app-demo-springCloud-provider**。

在**应用配置**页签中选择**部署包来源**、**Demo 类型**和**实例**，然后单击**下一步**。

部署包来源：选择**官方 Demo**。

Demo 类型：在下拉列表中选择具体的应用 Demo，如 **Spring Cloud 服务端应用**。

实例：选择**自定义**。

选择自定义后，界面会根据您的账号当前的资源情况有所不同。

网络和环境

如果您当前没有 VPC、命名空间和集群，EDAS 会为您创建默认环境。

如果您已经创建过 VPC、命名空间和集群等资源，会显示对应资源的选择列表。您可以在下拉列表中选择对应资源。

实例

- **选择实例规格**：选择实例规格，如**超小规格实例**。
- **购买数量**：选择要购买的实例数量，如 **1**。
- **服务协议**：勾选**《云服务器 ECS 服务条款》|《镜像商品使用条款》**。

在**应用高级配置**页签中输入**版本**和**应用健康检查**（可选），然后单击**创建应用**。

- **版本**：EDAS 配置使用当前时间戳作为版本，格式如 **yyyymmdd:hhmmss**。您也可以设置其它版本标识。
- **应用健康检查**：设置健康检查的 URL，用来检测应用是否健康运行。

在**应用创建完成**页签确认应用基本信息、应用配置和应用高级设置，确认无误后，单击**确定创建应用**。

结果验证

在服务端应用和客户端应用创建后，EDAS 将为该应用自动创建一条变更记录，您可以通过变更详情查看该应用创建的进度和状态。应用创建成功后，变更记录显示**执行成功**。

应用创建成功后，返回该应用的**基本信息**页面，查看应用的相关信息，应该和创建时保持一致。

单击**实例部署信息**查看该应用的状态，运行状态应为**运行正常**，变更状态（即第一次创建、部署）应为**成功**。同时，显示代购的 ECS 实例的 ID、IP、规格及 VPC 信息。

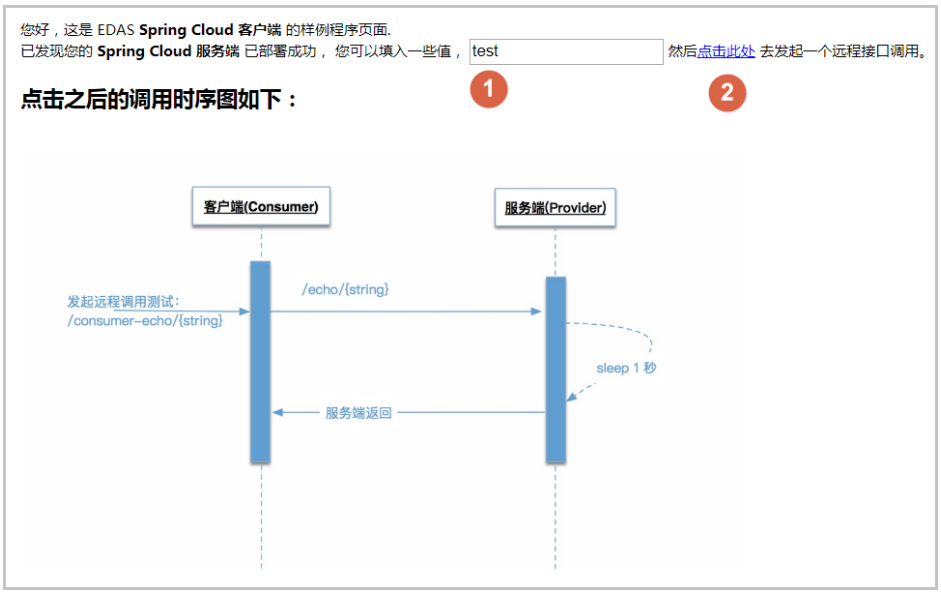
验证应用 Demo 的调用是否成功。

说明：验证调用需要按结果验证的步骤 1 和 2 确保服务端应用和客户端应用都部署成功。

进入客户端应用的详情页，在左侧导航栏单击**基本信息**，然后在**实例部署信息**页面单击

ECS 的 IP 地址。

在 **SC 客户端** 页面的 Echo this String 文本框中输入任意字符串，如 *test*，然后单击 **点击此处**。



在 **SC 客户端** 页面下方可以看到调用结果，返回了 test 字符串，说明调用成功。