企业级分布式应用服务 EDAS

快速入门

快速入门

部署 Java 应用到 ECS 集群

场景描述

为方便您快速开始使用 EDAS，EDAS 为您准备了一个仅包含欢迎页面的 Java Web 应用 Demo，您可以将它快速发布到多台 ECS 实例上。这些 ECS 实例需在阿里云上创建，并部署在阿里云 VPC 网络中。

前提条件

已完成发布应用的准备工作。包括：

1. 开通 EDAS 服务
2. 创建 VPC
3. 创建 ECS 实例
4. 创建命名空间
5. 创建 ECS 集群
6. 同步 SLB 到 EDAS：仅当你需要配置负载均衡时需完成该配置。

下载应用 Demo

创建并部署应用

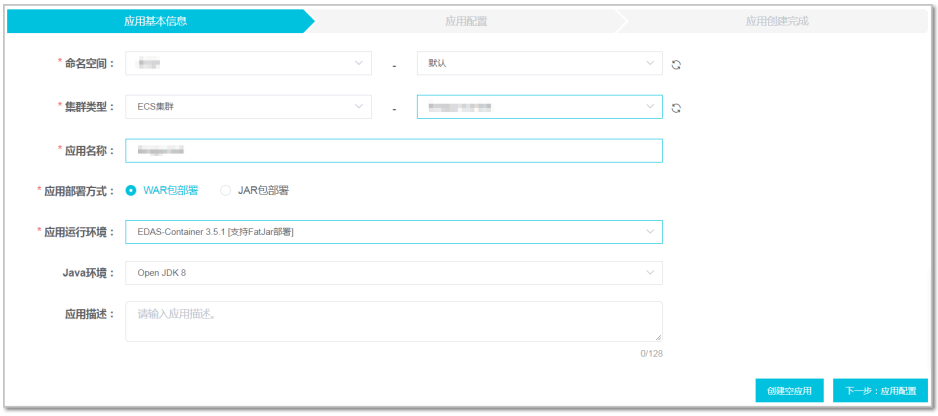
说明：你下载的是一个 WAR 包 Demo，所以本文档将介绍如果使用 WAR 包部署应用。JAR 包的创建和部署步骤和 WAR 包几乎一致。

登录 EDAS 控制台。

在左侧导航栏，单击**应用管理** > **应用列表**。

在应用列表页面右上角，单击**创建应用**。

在创建应用对话框中，输入应用相关信息，然后单击**下一步**。

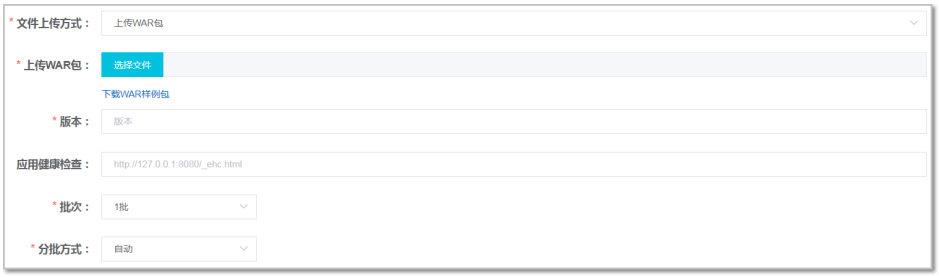


- **命名空间**：在下拉菜单中选择**地域**和**命名空间**。如果不选择会自动选择命名空间为**默认**。
- **集群类型**：在下拉菜单中选择 **ECS 集群** 并选择一个具体集群。
- **应用名称**：输入应用名称。
- **应用部署方式**：选择 **WAR 包部署**。
- **应用运行环境**：在下拉菜单中选择 EDAS-Container 版本，建议选择最新版本，如 *EDAS-Container 3.5.1 [支持 FatJar 部署]*。

在**应用配置**页面，添加**实例**，按照页面指示进行配置。完成设置后单击**确认创建**。

选定实例列表：单击**新增**，在弹出的**实例列表**页面中选择实例，单击 > 添加到右侧区域，然后单击**确定**。

是否立即部署：选择实例后才可单击打开。单击打开后按照界面进行配置。



文件上传方式：选择**上传 WAR 包**。

上传 WAR 包：单击**选择文件**，选择之前下载的 WAR 包 Demo 上传。

版本：设置版本（如：1.1.0），不建议用时间戳作为版本号。

应用健康检查（可选）：设置应用健康检查的 URL。应用的健康会在容器启动后/运行时检查应用的状态是否正常，会根据应用的健康检查结果来执行服务路由。设置参考示例为 http://127.0.0.1:8080/healthCheck.html

批次：设置批次，如果选择2次以上的批次，需要设置分批时间。

分批方式：选择自动。

应用创建可能需要几分钟，请您耐心等待。创建完成后可以前往应用详情页查看应用。在应用详情页中**实例部署信息**页签查看实例的运行状态。如果运行状态/时间为**正常运行**，说明应用发布成功。

结果验证

应用发布后，你可以通过查看实例运行状态或登录负载均衡配置网址来验证应用已成功发布。

查看应用实例运行状态

在应用详情页中**实例部署信息**页签查看 ECS 实例的运行状态。如果运行状态/时间为**运行正常**，说明应用发布成功。

配置公网负载均衡并访问应用

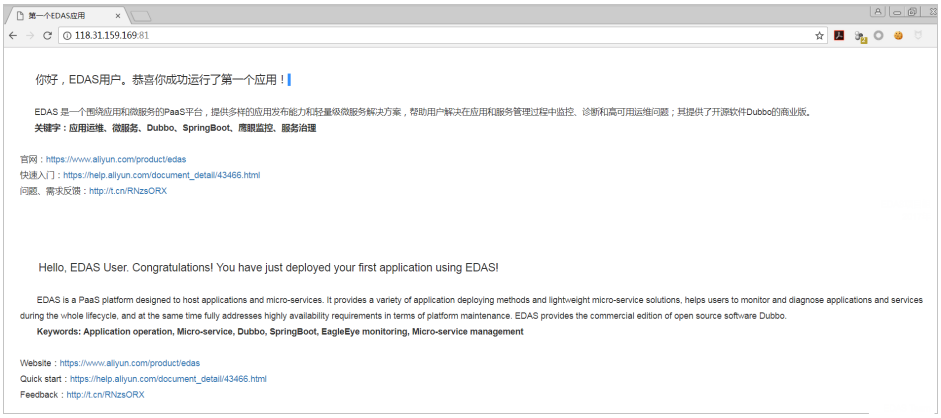
由于是在专有网络内创建发布的应用，如果没有特别配置，该应用没有公网 IP 地址。如果您的应用部署在多个 ECS 实例上，并且希望将您的应用对外开放，建议您配置公网负载均衡，以便将应用的访问流量根据转发策略分发到 ECS 实例中，增强应用的服务能力，提升应用的可用性。

在基本信息页面的应用设置区域，点击**负载均衡（公网）**右侧的**添加**。

在**添加 SLB 与应用的绑定**对话框中，设置负载均衡参数，然后点击**配置负载均衡**完成配置。

- **负载均衡（公网）**：在右侧的下拉菜单中，根据实际需求，选择内网或公网的 SLB 地址。
- **使用虚拟服务器组**：虚拟服务器组是一组处理负载均衡分发的前端请求的 ECS 实例。不同的监听可以关联不同的虚拟服务器组，实现监听维度的请求转发。如果您勾选了**使用虚拟服务器组**，则需要配置虚拟服务器组参数。
- **虚拟服务器组名称**：如果您选择了**新建虚拟服务器组**，则需要在此处输入虚拟服务器组名称。系统会按照您输入的名称为您创建虚拟服务器组。
- **监听（外网）**：负载均衡服务监听规定了如何将请求转发给后端服务器。一个负载均衡实例至少添加一个监听。您可以在监听右侧的下拉菜单中选择已创建的监听端口。如果您没有创建监听，单击**创建新监听**。请勿在服务均衡管理控制台上删除该监听，否则将影响应用访问。
- **SLB 前端协议**：默认为 TCP，不可配置。
- **SLB 前端端口**：输入 SLB 的前端端口，可自行设置端口数值。
- **应用端口**：默认为 8080，不可配置。

复制配置的 SLB IP 及端口，如 *118.31.159.169:81*，在浏览器的地址中粘贴并回车，即可进入应用的欢迎页面。



Spring Cloud 服务接入 EDAS

本文介绍如何使用 ANS 将您的 Spring Cloud 应用接入 EDAS，并使用 EDAS 服务注册中心实现服务发现。

如果您熟悉 Spring Cloud，阅读本文档后，可以发现，使用 ANS 与之前使用 Eureka 或者 Consul 实现服务注册与发现的使用方式没有任何差别。从 Eureka 和 Consul 迁移到 ANS 无需修改任何代码。

EDAS 目前只支持 Spring Cloud Finchley 和 Spring Cloud Edgware 两个版本中的所有小版本。对应 Spring Boot 版本请参考 Spring 官网。其中与 Finchley 对应的版本为 0.2.0.RELEASE， 与 Spring Cloud Edgware 对应的版本为 0.1.0.RELEASE。

为什么使用 ANS

应用名字服务 ANS（Application Naming Service）是 EDAS 提供的服务发现组件，是开源 Nacos 的商业化版本。

Spring Cloud Alibaba Ans 实现了 Spring Cloud Registry 的标准接口与规范，可以完全地替代 Spring Cloud Eureka 和 Spring Cloud Consul 提供的服务发现功能。

同时，与 Eureka 和 Consul 相比，还具有以下优势：

- ANS 为共享组件，节省了你部署、运维 Eureka 或 Consul 的成本。
- ANS 在注册和发现的调用中都进行了链路加密，保护您的服务，无需再担心服务被其他人发现。
- ANS 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。

本地开发

本文档将以一个提供者和一个消费者为例，向您介绍如何如何在本地开发服务提供者、消费者，如何部署到 EDAS 中，在 EDAS 服务注册中心完成服务注册，以及实现消费者对提供者的调用。

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 程序，可下载 service-provider 和 service-consumer。

准备工作

下载、启动及配置轻量级配置中心。

为了便于本地开发，EDAS 提供了一个包含了 EDAS 服务注册中心基本功能的轻量级配置中心。基于轻量级配置中心开发的应用无需修改任何代码和配置就可以部署到云端的 EDAS 中。

请您参考**配置轻量级配置中心**进行下载、启动及配置。推荐使用最新版本。

下载 Maven 并设置环境变量。

创建服务提供者

创建一个 Spring Cloud 工程，命名为 service-provider。

这里我们以 Spring Boot 2.0.6.RELEASE 和 Spring Cloud Finchley.SR1 为例，在 pom.xml 文件中加入如下内容。

```
 <parent>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-parent</artifactId>
<version>2.0.6.RELEASE</version>
<relativePath/>
</parent>

<dependencies>
<dependency>
<groupId>org.springframework.cloud</groupId>
<artifactId>spring-cloud-starter-alicloud-ans</artifactId>
<version>0.2.0.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>Finchley.SR1</version>
<type>pom</type>
```

```
<scope>import</scope>
</dependency>
</dependencies>
</dependencyManagement>
```

如果您需要选择使用 Spring Boot 1.x 的版本，请使用 Spring Boot 1.5.x 和 Spring Cloud Edgware 版本，对应的 Spring Cloud Alibaba 版本为 0.1.0.RELEASE。

说明：Spring Boot 1.x 版本的生命周期即将在 2019 年 8 月 结束，推荐使用新版本开发您的应用。

开发服务提供者的启动类，其中 @EnableDiscoveryClient 注解表明此应用需开启服务注册与发现功能。

```
@SpringBootApplication
@EnableDiscoveryClient
public class ServerApplication {
public static void main(String[] args) {
SpringApplication.run(ServerApplication.class, args);
}
}
```

创建一个简单的 controller，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。

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

在 application.properties 中添加如下配置，将注册中心指定为 EDAS 轻量级配置中心。

其中 *127.0.0.1* 为轻量级配置中心的地址，如果您的轻量级配置中心部署在另外一台机器，则需要修改成对应的 IP 地址。由于轻量级配置中心不支持修改端口，所以端口必须使用 8080。

```
spring.application.name=service-provider
server.port=18081
spring.cloud.alicloud.ans.server-list=127.0.0.1
spring.cloud.alicloud.ans.server-port=8080
```

注：spring.cloud.alicloud.ans.server-list=127.0.0.1 和 spring.cloud.alicloud.ans.server-port=8080这两个参数仅在本地开发环境使用轻量级配置中心作为服务注册的场景下使用，当应用部署到 EDAS 中时，这两个参数可以去掉，也可以保留，不影响服务注册和使用。

执行 service-provider 中的 main 函数，启动服务。

登录轻量级配置中心控制台界面 http://127.0.0.1:8080，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 service-provider，且可以查询该服务的分组和地址。

创建服务消费者

该部分文档我们不仅演示了服务发现的功能，还说明了 ANS 服务发现与 RestTemplate、AsyncRestTemplate 和 FeignClient 这三个客户端是如何结合的。

创建一个 Spring Cloud 工程，命名为 service-consumer。在 pom.xml 中引入需要的依赖内容：

```
 <parent>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-parent</artifactId>
<version>2.0.6.RELEASE</version>
<relativePath/>
</parent>

<dependencies>
<dependency>
<groupId>org.springframework.cloud</groupId>
<artifactId>spring-cloud-starter-alicloud-ans</artifactId>
<version>0.2.0.RELEASE</version>
</dependency>
<dependency>
<groupId>org.springframework.cloud</groupId>
<artifactId>spring-cloud-starter-openfeign</artifactId>
</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>Finchley.SR1</version>
<type>pom</type>
<scope>import</scope>
</dependency>
</dependencies>
</dependencyManagement>
```

配置 RestTemplate、AsyncRestTemplate 和 FeignClient。

FeignClient 是一个将HTTP 转为 RPC 格式调用的客户端。在使用他之前，我们需完成两项配置：

@EnableFeignClient注解。

配置对应的 HTTP URL 地址及 HTTP 方法。

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

在启动类中：

使用 @EnableDiscoveryClient 注解启用服务注册与发现；

使用 @EnableFeignClients 注解激活 FeignClient；

添加 @LoadBalanced 注解将 RestTemplate 与 AsyncRestTemplate 与服务发现结合。

```
@SpringBootApplication
@EnableDiscoveryClient
@EnableFeignClients
public class ConsumerApplication {
@LoadBalanced
@Bean
public RestTemplate restTemplate() {
return new RestTemplate();
}

@LoadBalanced
@Bean
public AsyncRestTemplate asyncRestTemplate(){
return new AsyncRestTemplate();
}

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

创建 Controller 以演示和验证服务发现功能。

```
@RestController
public class TestController {
@Autowired
private RestTemplate restTemplate;
@Autowired
private AsyncRestTemplate asyncRestTemplate;
@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-async-rest/{str}", method = RequestMethod.GET)
public String asyncRest(@PathVariable String str) throws Exception{
ListenableFuture<ResponseEntity<String>> future = asyncRestTemplate.
getForEntity("http://service-provider/echo/"+str, String.class);
return future.get().getBody();
}
@RequestMapping(value = "/echo-feign/{str}", method = RequestMethod.GET)
public String feign(@PathVariable String str) {
return echoService.echo(str);
}
}
```

在 application.properties 中添加如下配置，将注册中心指定为 EDAS 轻量级配置中心。

其中 *127.0.0.1* 为轻量级配置中心的地址，如果您的轻量级配置中心部署在另外一台机器，则需要修改成对应的 IP 地址。由于轻量级配置中心不支持修改端口，所以端口必须使用 8080。

```
 spring.application.name=service-consumer
server.port=18081
spring.cloud.alicloud.ans.server-list=127.0.0.1
spring.cloud.alicloud.ans.server-port=8080
```

执行 service-consumer 中的 main 函数，启动服务。

登录轻量级配置中心控制台界面 http://127.0.0.1:8080，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 service-consumer，且可以查询该服务的分组和地址。

结果验证

分别调用我们的演示 API，可以看到调用成功的结果。

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

部署到 EDAS

ANS 在设计之初就考虑到了从开发环境迁移到 EDAS 的场景，您可以直接将应用部署到 EDAS 中。

注意：各类型集群都支持 Spring Cloud 应用（使用 ANS 进行服务注册和发现），在 Swarm 集群和容器服务 Kubernetes 集群部署时需要添加额外配置。详情见下表：

集群类型	额外配置
ECS 集群	无
Swarm 集群	需要在应用详情页 > 应用设置 > 配置 JVM 参数时添加 -Dalicloud.deployment.mode=EDAS_MANAGED。
容器服务 Kubernetes 集群	需要应用制作镜像时在命令行参数中添加 JVM 参数 -Dalicloud.deployment.mode=EDAS_MANAGED。

发布单流程会通过优先级更高的方式去设置所有中间件相关的服务端地址、服务端口，以及中间件通信时的鉴权信息，您都无需关心。比如以下的配置可以继续保留或者不填写。

```
spring.cloud.alicloud.ans.server-list=127.0.0.1
spring.cloud.alicloud.ans.server-port=8080
```

分别在 service-rovider 和 service-onsumer 的 pom.xml 文件中添加如下配置，然后执行 mvn clean package 将本地的程序打成可执行 JAR 包。

Provider

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

Consumer

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

根据您想部署的集群类型，参考对应的部署应用文档部署应用。

配置项参考

配置项	key	默认值	说明
服务名	spring.cloud.alicloud.ans.client-domains	spring.application.name	当此项未配置时，默认从spring.application.name中获取。需要发布多个服务时，中间用英文的“,”号隔开
是否注册	spring.cloud.alicloud.ans.register-enabled	true	当只需要发现，不需要注册时，可以通过将值设置成 false来关闭注册
想要注册的IP	spring.cloud.alicloud.ans.client-ip	无	当需要指定本机注册的IP时，通过此值来配置，优先级最高
想要注册的IP所属的网卡	spring.cloud.alicloud.ans.client-interface-name	无	当确定需要发布哪块网卡对应的IP地址时，通过此参数配置，值为网卡名
想要注册的端口	spring.cloud.alicloud.ans.client-port	无	自定义想要注册的端口

FAQ

我看到我的服务注册成功了，如何调用呢？

答：spring-cloud-starter-alicloud-ans 默认支持集成了 Ribbon 支持，您可以使用 RestTemplate 和 FeignClient 调用。

为什么我的服务注册总是失败？

答：如果您在确认账号信息都准确无误的情况下，但是运行此文档中的 Demo 却注册失败了。有可能是由于您本机的时间不准确，从而导致验签鉴权失败。此时您需要校正本机的时间，建议打开时间自动同步功能。

后续操作

在将 Spring Cloud 应用接入到 EDAS，实现了服务注册与发现功能后，您还可以在应用中实现以下功能，再部署到 EDAS 中。

- 负载均衡
- 配置管理
- 搭建服务网关
- 迁移服务网关

Dubbo 服务接入 EDAS

如果您只有简单的 Java 基础和 Maven 经验，而不熟悉 Dubbo，本文档将帮助您从零开始开发 Dubbo 服务，并使用 EDAS 服务注册中心实现服务注册与发现。

注意， EDAS 目前支持的 Dubbo 版本为 2.5.3 到 2.6.5

如果您熟悉 Dubbo，可以选择性地阅读相关章节。

本文档将以一个提供者和一个消费者为例，向您介绍如何在本地使用 xml 开发服务提供者、消费者，如何部署到 EDAS 中，在 EDAS 服务注册中心完成服务注册，以及实现消费者对提供者的调用。您还可以使用 Spring Boot 开发 Dubbo 应用。

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 Dubbo 服务程序，可下载 edas-dubbo-demo。

为什么使用 EDAS 服务注册中心

EDAS 服务注册中心实现了 Dubbo 所提供的 SPI 标准的注册中心扩展，能够完整地支持 Dubbo 服务注册、路由规则、配置规则功能。

EDAS 服务注册中心能够完全代替 ZooKeeper 和 Redis，作为您 Dubbo 服务的注册中心。同时，与 ZooKeeper 和 Redis 相比，还具有以下优势：

- EDAS 服务注册中心为共享组件，节省了您运维、部署 ZooKeeper 等组件的机器成本。
- EDAS 服务注册中心在通信过程中增加了鉴权加密功能，为您的服务注册链路进行了安全加固。

- 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。

本地开发

准备工作

下载、启动并配置轻量级配置中心。

为了便于本地开发，EDAS 提供了一个包含了 EDAS 服务注册中心基本功能的轻量级配置中心。基于轻量级配置中心开发的应用无需修改任何代码和配置就可以部署到 EDAS 中。

请您参考**配置轻量级配置中心**进行下载、启动及配置。推荐使用最新版本。

下载 Maven 并设置环境变量（如已安装可跳过）。

创建服务提供者

创建 Maven 项目并引入依赖。

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

在 Maven 项目中的pom.xml文件中添加 **dubbo** 和 **edas-dubbo-extension** 依赖，版本分别为 *2.6.2* 和 *1.0.2*。

```
<dependencies>
<dependency>
<groupId>com.alibaba</groupId>
<artifactId>dubbo</artifactId>
<version>2.6.2</version>
</dependency>

<dependency>
<groupId>com.alibaba.edas</groupId>
<artifactId>edas-dubbo-extension</artifactId>
<version>1.0.2</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="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:dubbo="http://code.alibabatech.com/schema/dubbo"
xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-4.0.xsd
http://code.alibabatech.com/schema/dubbo
http://code.alibabatech.com/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 EDAS 轻量级配置中心。

其中 *127.0.0.1* 为轻量级配置中心的地址，如果您的轻量级配置中心部署在另外一台机器，则需要修改成对应的 IP 地址。由于轻量级配置中心不支持修改端口，所以端口必须使用 8080。

```
<dubbo:registry id="edas" address="edas://127.0.0.1:8080"/>
```

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。

登录轻量级配置中心控制台 http://127.0.0.1:8080，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

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="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:dubbo="http://code.alibabatech.com/schema/dubbo"
xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-4.0.xsd
http://code.alibabatech.com/schema/dubbo
http://code.alibabatech.com/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 id="edas" address="edas://127.0.0.1:8080"/>

<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。
ii. 登录轻量级配置中心控制台 http://127.0.0.1:8080，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

edas-dubbo-extension 在设计之初就考虑到了从本地迁移到 EDAS 的场景，您可以直接将应用部署到 EDAS 中。

注意：各类型集群都支持 Dubbo 应用，在 Swarm 集群和容器服务 Kubernetes 集群部署时需要添加额外配置。详情见下表：

集群类型	**额外配置**
ECS 集群	无
Swarm 集群	需要在应用详情页 > 应用设置 > 配置 JVM 参数时添加 -Dalicloud.deployment.mode=EDAS_MANAGED。
容器服务 Kubernetes 集群	需要应用制作镜像时在命令行参数中添加 JVM 参数 -

	Dalicloud.deployment.mode=EDAS_MANAGED。

分别在 Provider 和 Consumer 的 pom.xml 文件中添加如下配置，然后执行 mvn clean package 将本地的程序打成可执行的 JAR 包。

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>
```

根据您要部署的集群类型，参考对应的文档部署应用。