

企业级分布式应用服务 EDAS

快速入门

快速入门

发布 ECS 集群应用

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

登录 EDAS 控制台。

在左侧导航栏中单击**应用管理**，进入应用列表页面。

在应用列表页面选择地域，然后在右上角单击**创建应用**。

在**应用基本信息**页面中设置应用的基本信息和参数，然后单击**下一步:应用配置**。

应用基本信息

所在区域: 默认

部署集群: [选择集群]

应用类型: Java应用

应用名称: [输入名称]

应用运行环境: EDAS-Container 3.4.2 [支持fatjar部署]

应用描述: [输入描述]

应用描述主要介绍应用的基本情况。输入的描述信息不超过100个字符。

创建应用 下一步: 应用配置

配置	说明
所在区域	在下拉菜单中选择地域和命名空间（如无命名空间，选择默认即可）。
部署集群	在下拉菜单中选择一个 ECS 集群。
应用类型	应用类型由部署集群决定，选择了 ECS 集群，应用类型则为 Java 应用，不可配置。
应用名称	输入应用名称。
应用运行环境	在下拉菜单中选择 EDAS container 的版本。
应用描述	填写应用的基本情况。

在应用配置页面设置部署参数，单击**确认创建**。

应用配置

警告提示: 自0月1日起, Spring Cloud and Service Mesh 用户, 无需创建应用即可快速使用EDAS注册中心。 查看如何发布一个微服务

应用部署方式: War包部署 Jar包部署

上传war包: [选择文件] [上传成功]

请填写版本: [输入版本]

应用健康检查: http://127.0.0.1:8080/healthCheck.html

批次: [选择批次]

分批方式: [选择分批方式]

安全组信息 您已被加入下列安全组, 您可以在单机的安全组信息中查看 [容器服务安全组快速指南](#)

授权策略	协议类型	端口范围	授权类型	授权对象
允许	TCP	8182/8182	授权地址访问	[选择实例]
允许	TCP	8182/8182	授权地址访问	[选择实例]
允许	TCP	8182/8182	授权地址访问	[选择实例]
允许	TCP	8182/8182	授权地址访问	[选择实例]
允许	TCP	8182/8182	授权地址访问	[选择实例]
允许	TCP	8182/8182	授权地址访问	[选择实例]
允许	TCP	65000/65535	授权地址访问	[选择实例]
允许	TCP	12200/12300	授权地址访问	[选择实例]

上一步: 应用基本信息 确认创建

配置	说明
应用部署方式	勾选 WAR 包部署。
选择实例列表	单击 新增 ，在 选择单机 对话框左侧区域中选择单机，然

	后点击>将实例添加到右侧区域中，点击 确认 。
是否立即部署	添加实例后勾选。勾选后在下方设置部署参数。
文件上传方式	选择 上传 WAR 包 。
上传 WAR 包	点击 选择文件 ，在本地选择并上传您的应用WAR 包。
请填写版本	输入标识此次应用发布的部署包的版本。
请填写版本	输入标识此次应用发布的部署包的版本。部署应用的时候，可以添加一个版本号或者文字描述，不建议用时间戳作为版本号。
应用健康检查	输入应用健康检查的 URL，默认为： ：http://127.0.0.1:8080/healthCheck.html。
批次	即分几个批次进行部署。下拉菜单中的选项根据该应用的实例数自动生成。当选择 2 批或 2 批以上 时，需要 设置分批等待时间 。
分批方式	选择 自动 。
分批等待时间	不同批次部署之间的等待时间。您可以在等待时间内，检查已部署的应用是否正常，以便决定是否要继续剩余批次的部署。

应用创建后，请跳转到变更详情页面。在左侧导航栏中点击**基本信息**，在**基本信息**和**实例部署信息**页面查看应用状态。

配置公网负载均衡

由于是在 VPC 创建的 ECS 实例，如果没有特别配置，该实例没有公网 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。

添加SLB与应用的绑定
✕

开启SLB端口监听后，会自动在SLB上新增端口监听。
请勿在SLB控制台上删除该监听，否则将影响应用访问。

负载均衡(公网) : 192.168.1.1 使用虚拟服务器组

虚拟服务器组 (外网) : 新建虚拟服务器组

虚拟服务器组名称 : 虚拟服务器组名称

监听 (外网) : 监听名称 创建新监听 :

SLB 前端协议 : TCP

SLB 前端端口 : 80

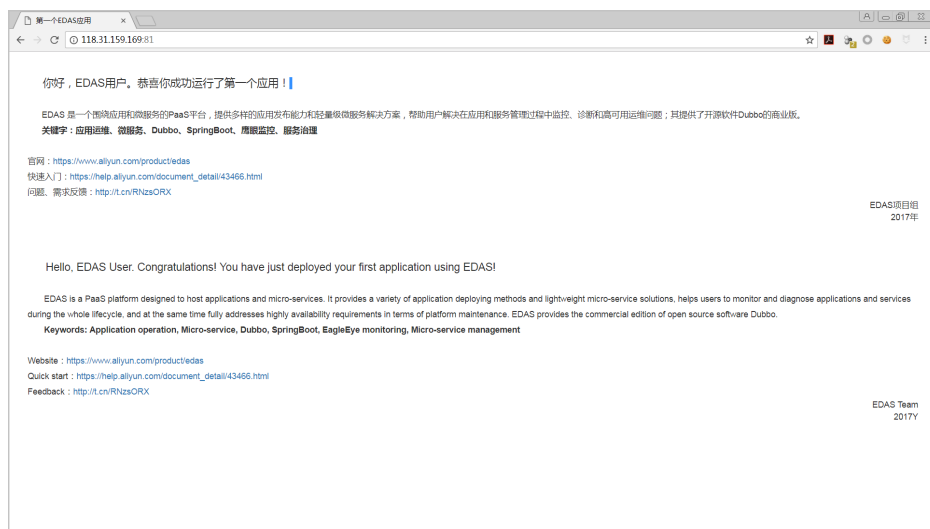
应用端口 : 8080

配置负载均衡
取消

配置	说明
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。

发布验证

应用发布完成后，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 接入服务注册与发现

本文介绍如何使用 ANS (Application Naming Service ， 服务发现组件) 将基于 Spring Cloud 开发的应用接入到 EDAS ， 并实现服务发现。

前提条件

注册阿里云账号

开通 EDAS 服务

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 1.5.8 和 Spring Cloud Dalston.SR4 为例，在 `pom.xml` 中引入需要的依赖内容。

其他版本如 Spring Boot 2 和 Spring Cloud Finchley 也同样支持，请您自行修改版本号和替换相应的组件依赖。

```
<parent>
```

```
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-parent</artifactId>
<version>1.5.8.RELEASE</version>
<relativePath/>
</parent>

<dependencies>
<dependency>
<groupId>com.alibaba.cloud</groupId>
<artifactId>spring-cloud-starter-ans</artifactId>
<version>1.1.3</version>
</dependency>
<dependency>
<groupId>com.alibaba.cloud</groupId>
<artifactId>spring-cloud-alibaba-edas-starter</artifactId>
<version>1.1.3</version>
</dependency>
</dependencies>

<dependencyManagement>
<dependencies>
<dependency>
<groupId>org.springframework.cloud</groupId>
<artifactId>spring-cloud-dependencies</artifactId>
<version>Dalston.SR4</version>
<type>pom</type>
<scope>import</scope>
</dependency>
</dependencies>
</dependencyManagement>
```

编码服务提供端的启动类，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，逻辑是将收到的参数回显。

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

权限配置，配置阿里云账号的 AccessKey、SecretKey，以及 EDAS 的命名空间信息。

配置阿里云 AccessKey 和 SecretKey。

登录阿里云 AK 管理控制台。

在左侧导航栏单击**安全信息管理**。

在**安全信息管理**页面复制 AccessKey ID 和 Access Key Secret，分别对应配置项中的 alibaba.cloud.access-key 和 alibaba.cloud.secret-key。

安全凭证信息格式如下：

```
alibaba.cloud.access-key=xxxxxxxxxx
alibaba.cloud.secret-key=xxxxxxxxxx
```

d. 将安全凭证信息粘贴到您的配置文件中，如 application.properties。

配置 EDAS 的命名空间。

登录 EDAS 控制台。

在左侧导航栏中单击**命名空间**。

在命名空间列表页面选择**地域**，并找到您想发布到的**命名空间**，复制其**命名空间 ID**，对应配置项中的 **alibaba.edas.namespace**。

```
alibaba.edas.namespace=xxxxxxxxxx
```

将命名空间信息粘贴到您的配置文件中，如 application.properties。

经过如上配置之后，我们的配置文件 application.properties 内容如下。

```
spring.application.name=service-provider
server.port=18081
alibaba.cloud.access-key=xxxxxxxxxx
alibaba.cloud.secret-key=xxxxxxxxxx
alibaba.edas.namespace=cn-hangzhou
```

登录 EDAS 控制台，进入**服务管理 > 服务查询**页面。选择**地域**和**命名空间**，在**服务查询**页面查看服务注册的信息。可以在**服务名列表**查看到创建的服务实例 service-provider，单击**详情**进入服务详情

页，可以查看本机的IP地址和端口等信息。

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>1.5.8.RELEASE</version>
<relativePath/>
</parent>

<dependencies>
<dependency>
<groupId>com.alibaba.cloud</groupId>
<artifactId>spring-cloud-starter-ans</artifactId>
<version>1.1.3</version>
</dependency>
<dependency>
<groupId>com.alibaba.cloud</groupId>
<artifactId>spring-cloud-alibaba-edas-starter</artifactId>
<version>1.1.3</version>
</dependency>
</dependencies>

<dependencyManagement>
<dependencies>
<dependency>
<groupId>org.springframework.cloud</groupId>
<artifactId>spring-cloud-dependencies</artifactId>
<version>Dalston.SR4</version>
<type>pom</type>
<scope>import</scope>
</dependency>
</dependencies>
</dependencyManagement>
```

注意：与 service-provider 相比，service-consumer 要演示 FeignClient 的使用，故 pom.xml 文件中增加了一个 spring-cloud-starter-feign 的依赖。

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();
}

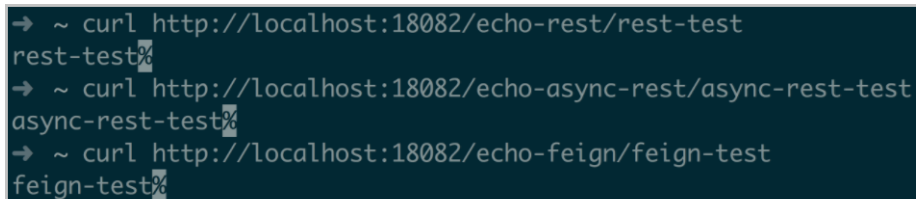
@RequestMapping(value = "/echo-feign/{str}", method = RequestMethod.GET)
public String feign(@PathVariable String str) {
    return echoService.echo(str);
}
}
```

添加应用基本配置和阿里云 AK、SK 以及 EDAS 的命名空间。

```
spring.application.name=service-consumer
server.port=18082
alibaba.cloud.access-key=xxxxxxxxxx
alibaba.cloud.secret-key=xxxxxxxxxx
alibaba.edas.namespace=cn-hangzhou
```

登录 EDAS 控制台，进入**服务管理** > **服务查询**页面。选择**地域**和**命名空间**，在**服务查询**页面查看服务注册的信息。可以在**服务名**列表查看到创建的服务实例 service-consumer。

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%
```

Demo 下载

service-provider

service-consumer

更多配置项

配置项	key	默认值	说明	补充说明
服务名	spring.cloud.ans.doms	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	production
是否注册	spring.cloud.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	production
想要注册的IP	spring.cloud.ans.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	production
想要注册的IP所属的网卡	spring.cloud.ans.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	production
想要注册的端口	spring.cloud.ans.port	无	自定义想要注册的端口	test
注册的权重	spring.cloud.ans.weight	1	数值越大权重越高	test
集群	spring.cloud.ans.cluster	DEFAULT	可以通过集群来分别标记服务	test
租户环境	spring.cloud.ans.env	1	相同租户的相同环境下的服务才能互相发现	test

FAQ

我看到我的服务注册成功了，如何调用呢？

答：spring-cloud-starter-ans 默认支持集成了 Ribbon，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。

参考

更多详情请参考 ANS 使用指南。