**Part III. Spring Cloud Netflix**

**11.服务发现：Eureka Clients**

服务发现是微服务框架中的一项重要原则。而如何对每一个client或他们之间的约束进行处理却是非常困难和不稳定的。Eureka作为Netflix服务发现的server和client。它的每个节点都可以复制自身的状态，并且相互注册服务，这样就保证了配置和部署的高可用性。

**11.1如何引入Eureka Client**

引入Eureka Client需要org.springframework.cloud组下的spring-cloud-starter-netflix-eureka-client 启动包。有关使用当前的Spring Cloud发布列表设置构建系统的详细信息，请参阅Spring Cloud项目页面。

**11.2 在Eureka上注册**

当一个服务节点在Eureka上注册时，它会向Eureka发送自己的元数据，其中包括主机地址,端口号，健康检查URL，主页等等。Eureka接收每一个服务注册实例的心跳信息。如果心跳信息在可配置的一定时间内发送失败，那么注册中心通常会将该实例移除。

Eureka client 示例：

*@Configuration*

*@ComponentScan*

*@EnableAutoConfiguration*

*@RestController*

**public** **class** Application {

*@RequestMapping("/")*

**public** String home() {

**return** "Hello world";

}

**public** **static** **void** main(String[] args) {

**new** SpringApplicationBuilder(Application.**class**).web(true).run(args);

}

}

（该示例是完全普通的一个Spring Boot工程）。通过引入spring-cloud-starter-netflix-eureka-client,你的工程会自动在Eureka服务器上注册，你需要做的只是在配置文件中定位Eureka服务器的位置，示例如下：

**application.yml.**

eureka:

client:

serviceUrl:

defaultZone: http://localhost:8761/eureka/

其中“defaultZone”魔法值作为一个错误处理备用值可以向任何一个没有进行高优先级设置的服务节点提供服务URL。（即这是一个很有用的默认值）。

默认的应用工程名称（服务ID）,虚拟host和非安全端口分别由Environment中的${spring.application.name}, ${spring.application.name}和${server.port}属性决定。

引入spring-cloud-starter-netflix-eureka-client使得应用既是一个Eureka服务实例（即注册自己）也是一个客户端（即它能通过注册信息定位其他的服务）。The instance behaviour is driven by eureka.instance.\* configuration keys, but the defaults will be fine if you ensure that your application has a spring.application.name (this is the default for the Eureka service ID, or VIP).

有关可配置选项的更多详细信息，请参阅EurekaInstanceConfigBean和EurekaClientConfigBean。

禁止Eureka的服务发现可以将eureka.client.enabled设置为false;

**11.3 使用Eureka服务器进行身份验证**

如果eureka.client.serviceUrl.defaultZone中配置的一系列URL有凭证的嵌入（curl 风格，如<http://user:password@localhost:8761/eureka>），对于更为复杂的需求，你可以创建一个DiscoveryClientOptionalArgs的@Bean并且注入ClientFilter实例，所有这些都将应用于从客户端到服务器的调用。

注意：由于Eureka中的限制，不可能支持每个服务器的基本身份验证凭据，所以只能使用第一个找到的集合。

**11.4 状态页和健康指示器**

一个Eureka实例的状态页和健康指示器地址默认为“/info”和”/health”，同时也是Spring Boot Actuator 应用中一些有用端点的默认位置。如果你使用了非默认的上下文路径或者servlet路径(例如server.servletPath=/foo)或管理端点路径(例如 management.contextPath=/admin)，那么你需要更改配置信息，示例如下：

**application.yml.**

eureka:

instance:

statusPageUrlPath: ${management.context-path}/info

healthCheckUrlPath: ${management.context-path}/health

这些链接显示在客户端使用的元数据中，并在某些情况下用于决定是否将请求发送到应用程序，因此如果它们是准确配置的，将会很有帮助。

**11.5 注册一个安全的应用**

如果你的应用想通过https协议进行通信，你可以在EurekaInstanceConfig中设置两个标志位，即eureka.instance.[nonSecurePortEnabled,securePortEnabled]=[false,true]。这将使Eureka发布实例信息表明安全通信的优先级为最高。Spring Cloud 的DiscoveryClient将始终给服务返回以https开头的URI，并且Eureka（本机）实例信息将对URL进行安全健康检查。

由于Eureka内部的工作方式，它仍然会发布状态页和主页的非安全URL，除非你也明确地覆盖这些配置。你可以使用占位符来配置eureka实例URL，例如：

**application.yml.**

eureka:

instance:

statusPageUrl: https://${eureka.hostname}/info

healthCheckUrl: https://${eureka.hostname}/health

homePageUrl: https://${eureka.hostname}/

（注意，${eureka.hostname}是仅在Eureka较后版本中可用的本地占位符，你也可以使用Spring占位符实现同样的功能，例如使用${eureka.instance.hostName）。

如果您的应用程序通过代理服务器运行，并且SSL终端在代理中（例如，如果您运行在Cloud Foundry或其他平台的程序作为服务），则需要确保代理“转发”头部被应用程序拦截并处理。Spring Boot应用程序中，如果“X-Forwarded - \ \*”头被显示配置，嵌入式Tomcat容器会自动进行拦截处理。这个错误的一个标志是指向你自身应用的链接会是错误的。（错误的主机，端口或协议）。

**11.6 Eureka的健康状态检查**

默认情况下，Eureka使用客户端心跳来判断客户端是否启动。除非有特别设定，否则Discovery Client将不会根据Spring Boot Actuator发送应用当前的健康检查状态。这意味着成功注册后Eureka将永远声明该应用处于“UP”状态。通过启用Eureka运行健康检查可以改变这种状况，从而将应用程序状态发送给Eureka。因此，每个其他应用将不会在“UP”之外的状态下将流量发送到当前应用。  
**application.yml.**

eureka:

client:

healthcheck:

enabled: true

eureka.client.healthcheck.enabled=true只能在application.yml中设置。在bootstrap.yml中对该值进行设置将导致意想不到的错误，譬如在Eureka上注册了未知的状态。

如果你需要对应用的健康检查有更多的控制，你可以自己实现com.netflix.appinfo.HealthCheckHandler

**11.7 实例和客户端的Eureka元数据**

花点时间了解Eureka元数据的工作原理是很有用的，so you can use it in a way that makes sense in your platform。元数据包括有主机名，IP地址，端口号，状态页和运行状况检查等标准元数据。这些信息被放在在服务注册表中，客户端通过服务注册表直接的访问服务，除此之外，额外的元数据也可以添加到注册实例的eureka.instance.metadataMap中，并且远程客户端也可以访问这些数据，但一般不会影响客户端的运行，除非客户端使用元数据作为有意义的数据进行处理。下面描述了几个特殊情况，其中Spring Cloud已经为metadata map指定了含义。

**11.7.3 更改Eureka实例ID**

一个vanilla Netflix Eureka实例注册后，ID与其主机名一一对应（即每个host只有一个服务）。Spring Cloud Eureka 提供了一个如下默认配置：

${spring.cloud.client.hostname}:${spring.application.name}:${spring.application.instance\_id:${server.port}}}.

例如myhost:myappname:8080.

使用Spring Cloud你可以通过设置eureka.instance.instanceId为一个unique identifier覆盖这种默认配置。例如：

**application.yml.**

eureka:

instance:

instanceId: ${spring.application.name}:${vcap.application.instance\_id:${spring.application.instance\_id:${random.value}}}

通过这个元数据和在localhost上部署的多个服务实例，随机值会分配给各个实例使得每个实例都是唯一的。在Cloudfoundry中，vcap.application.instance\_id将在Spring Boot应用程序中自动填充，因此不需要随机值。

**11.8 使用EurekaClient**

你的应用一旦具有了服务发现功能，你就可以利用它通过Eureka Server找到其他的服务实例。其中一种方法是使用本地的com.netflix.discovery.EurekaClient（而不是Spring Cloud 的DiscoveryClient），例如：

@Autowired

private EurekaClient discoveryClient;

public String serviceUrl() {

InstanceInfo instance = discoveryClient.getNextServerFromEureka("STORES", false);

return instance.getHomePageUrl();

}

不要在@PostConstruct方法或者@Scheduled方法中（或者ApplicationContext还未运行的任何地方）使用EurekaClient。它会在一个SmartLifecycle (with phase=0)被初始化，所以，当其他SmartLifecycle 的phase值比0大时，你才能使用这个EurekaClient。

**11.8.1 使用没有Jersey的EurekaClient**

EurekaClient默认使用Jersey作为HTTP通信，如果你想排除对Jersey的依赖，你可以在你的dependencies中排除。Spring Cloud会自动基于Spring的RestTemplate构建一个transport client。

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-eureka</artifactId>

<exclusions>

<exclusion>

<groupId>com.sun.jersey</groupId>

<artifactId>jersey-client</artifactId>

</exclusion>

<exclusion>

<groupId>com.sun.jersey</groupId>

<artifactId>jersey-core</artifactId>

</exclusion>

<exclusion>

<groupId>com.sun.jersey.contribs</groupId>

<artifactId>jersey-apache-client4</artifactId>

</exclusion>

</exclusions>

</dependency>

**11.9 本机Netflix EurekaClient的替代方案**

您不必使用原生的Netflix EurekaClient，而且通常封装后使用它更为方便。Spring Cloud支持Feign（REST客户端构建器），还支持Spring RestTemplate使用logical Eureka service identifiers (VIPs)而不是物理URL。要使用固定的物理服务器列表配置Ribbon，您可以将<client>.ribbon.listOfServers设置为逗号分隔的物理地址（或主机名）列表，其中<client>是客户端的ID。

你还可以使用org.springframework.cloud.client.discovery.DiscoveryClient，它发现客户端提供了一个简单的接口而并不是只针对Netflix，例如：

@Autowired

private DiscoveryClient discoveryClient;

public String serviceUrl() {

List<ServiceInstance> list = discoveryClient.getInstances("STORES");

if (list != null && list.size() > 0 ) {

return list.get(0).getUri();

}

return null;

}

**11.10 为什么注册一个服务这么慢？**

作为一个实例，需要定期发送心跳信息到注册中心（通过客户端的serviceUrl），默认持续时间为30秒。在服务实例，注册服务器和客户端在其本地缓存中都具有相同的元数据（因此可能需要3个心跳）之前，客户端才能发现服务。你可以通过更改eureka.instance.leaseRenewalIntervalInSeconds更改这段时长，这会加快客户端连接到其他服务的过程。在生产环境中，最好坚持使用默认值，因为服务器内部有一些计算可以make assumptions about the lease renewal period。

**11.11 区**

如果您已将Eureka客户端部署到多个区域，您可能希望这些客户端在使用另一个区域中的服务之前，首先调用同一区域的服务。为此，您需要正确配置您的Eureka客户端。

首先，您需要确保将Eureka服务器部署到每个区域，并且它们是彼此的对等体。有关详细信息，请参阅区域和区域部分 。

接下来，您需要告知Eureka您的服务所在的区域。您可以使用metadataMap属性来执行此操作。例如，如果service 1部署到zone 1和zone 2，则需要在service 1中设置以下Eureka属性

**Service 1 in Zone 1**

eureka.instance.metadataMap.zone = zone1

eureka.client.preferSameZoneEureka = true

**Service 1 in Zone 2**

eureka.instance.metadataMap.zone = zone2

eureka.client.preferSameZoneEureka = true

**12 服务发现：Eureka服务器**

**12.1 如何include Eureka服务器**

要在项目中include Eureka服务器，请使用组org.springframework.cloud下组件id为 spring-cloud-starter-eureka-server的启动器。

**12.1 如何运行 Eureka服务器**

Eureka服务器示例：

@SpringBootApplication

@EnableEurekaServer

public class Application {

public static void main(String[] args) {

new SpringApplicationBuilder(Application.class).web(true).run(args);

}

}

服务器有一个带UI的主页，per the normal Eureka functionality的/eureka/\*下有HTTP API端点