### **Service Registration and Discovery**

In this lesson you are going set up a Netflix Eureka service registry and then build a client that both registers itself with the registry and uses it to resolve its own host. A service registry is useful because it enables client-side load-balancing and decouples service providers from consumers without the need for DNS.

#### **Setup Eureka Server**

1. Create new eureka-server project. That command should be executed from cloud-native-workshop folder.

|  |
| --- |
| $ mvn archetype:generate -DgroupId=com.github.YOUR\_USERNAME.eureka -DartifactId=eureka\_server -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false |

1. mvn archetype:generate -DgroupId=com.github.YOUR\_USERNAME.eureka -DartifactId=eureka\_server -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false
2. Add the following to the pom file

|  |
| --- |
| <parent>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-parent</artifactId>  <version>1.5.8.RELEASE</version>  </parent>  <dependencyManagement>  <dependencies>  <dependency>  <groupId>org.springframework.cloud</groupId>  <artifactId>spring-cloud-dependencies</artifactId>  <version>Camden.SR5</version>  <type>pom</type>  <scope>import</scope>  </dependency>  </dependencies>  </dependencyManagement>  <dependencies>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-web</artifactId>  </dependency>  <dependency>  <groupId>org.springframework.cloud</groupId>  <artifactId>spring-cloud-starter-eureka-server</artifactId>  </dependency>  </dependencies>  <properties>  <java.version>1.8</java.version>  </properties>  <build>  <plugins>  <plugin>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-maven-plugin</artifactId>  </plugin>  </plugins>  </build> |

1. Create EurekaServiceApplication class with the following content.

|  |
| --- |
| @EnableEurekaServer  @SpringBootApplication  public class EurekaServiceApplication {  public static void main(String[] args) {  SpringApplication.run(EurekaServiceApplication.class, args);  }  } |

1. Create manifest for the eureka server. Don't forget to replace EUREKA\_APP\_NAME with some unique name.

|  |
| --- |
| applications:  - name: EUREKA\_APP\_NAME  path: target/eureka\_server-1.0-SNAPSHOT.jar  memory: 1G  env:  eureka.client.register-with-eureka: false  eureka.client.fetch-registry: false  logging.level.com.netflix.eureka: OFF  logging.level.com.netflix.discovery: OFF |

1. Build and push. If at that point you get memory limit message you now can delete Cloud Config application and delete @RefreshScope attribute from StockResource in 12f app. Also delete add environment variables from the application manifest.

#### **Update 12f\_app to user service registration and discovery**

1. Add eureka dependency

|  |
| --- |
| <dependency>  <groupId>org.springframework.cloud</groupId>  <artifactId>spring-cloud-starter-eureka-server</artifactId>  </dependency> |

1. Add eureka.client.serviceUrl.defaultZone: https://EUREKA\_APP\_NAME.cfapps.io/eureka environment variable to the application manifest.
2. Add EnableDiscoveryClient annotation to StockSpringBootStarter class
3. Add ServiceInstanceController class

|  |
| --- |
| @RestController  class ServiceInstanceController {  @Autowired  private DiscoveryClient discoveryClient;  @RequestMapping("/service-instances/{applicationName}")  public List<ServiceInstance> serviceInstancesByApplicationName(  @PathVariable String applicationName) {  return this.discoveryClient.getInstances(applicationName);  }  } |

1. Build and push
2. Check /service-instances/12f-app endpoint