### **Centralized Configuration**

You’ll setup a Config Server and then build a client that consumes the configuration on startup and then refreshes the configuration without restarting the client.

#### **Setup Cloud Config Server**

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

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

1. 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-config-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 ConfigServiceApplication class with the following content.

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

1. Create new github repository github.com/YOUR\_USERNAME/config-repository
2. Add 12f-app.properties file to the root of the repository with the following content

|  |
| --- |
| message = Hello world |

1. Commit and push the changes.
2. Create manifest for the config server application. Replace YOUR\_USERNAME with your github username and CONFIG\_APP\_NAME with some unique name.

|  |
| --- |
| applications:  - name: CONFIG\_APP\_NAME  path: target/config\_server-1.0-SNAPSHOT.jar  memory: 1G  env:  spring.cloud.config.server.git.uri: https://github.com/YOUR\_USERNAME/config-repository |

1. Build and push to CF
2. Check http://CONFIG\_APP\_NAME.cfapps.io/12f-app/cloud url to ensure that configuration server is actually providing data.

#### **Update 12f\_app to use Cloud Config Server**

1. Add client config dependencies

|  |
| --- |
| <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> |

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

1. Add RefreshScope annotation to StockResource class
2. Add the following section to 12f\_app manifest

|  |
| --- |
| env:  spring.application.name: 12f-app  spring.cloud.config.uri: https://YOUR\_CONFIG\_APP\_NAME.cfapps.io  management.security.enabled: false |

1. Redeploy the app
2. Access get-env endpoint to check the value of message variable.

|  |
| --- |
| $ https://APP\_NAME.cfapps.io/stock/get-env?env=message |