# Spring Cloud Server

manages all the application related configuration properties

## Creating Spring Cloud Configuration Server

```
<dependency>
   <groupId>org.springframework.cloud</groupId>
   <artifactId>spring-cloud-config-server</artifactId>
</dependency>
```

- Gradle users can add the below dependency in your build.gradle file.

```
compile('org.springframework.cloud:spring-cloud-config-server')
```

# Creating Spring Cloud Configuration Server

- Add the **@EnableConfigServer** annotation in your main Spring Boot application class file. The @EnableConfigServer annotation makes your Spring Boot application act as a Configuration Server

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

## application.properties

- Add the below configuration to your properties file and replace the application.properties file into bootstrap.properties file. Observe the code given below −

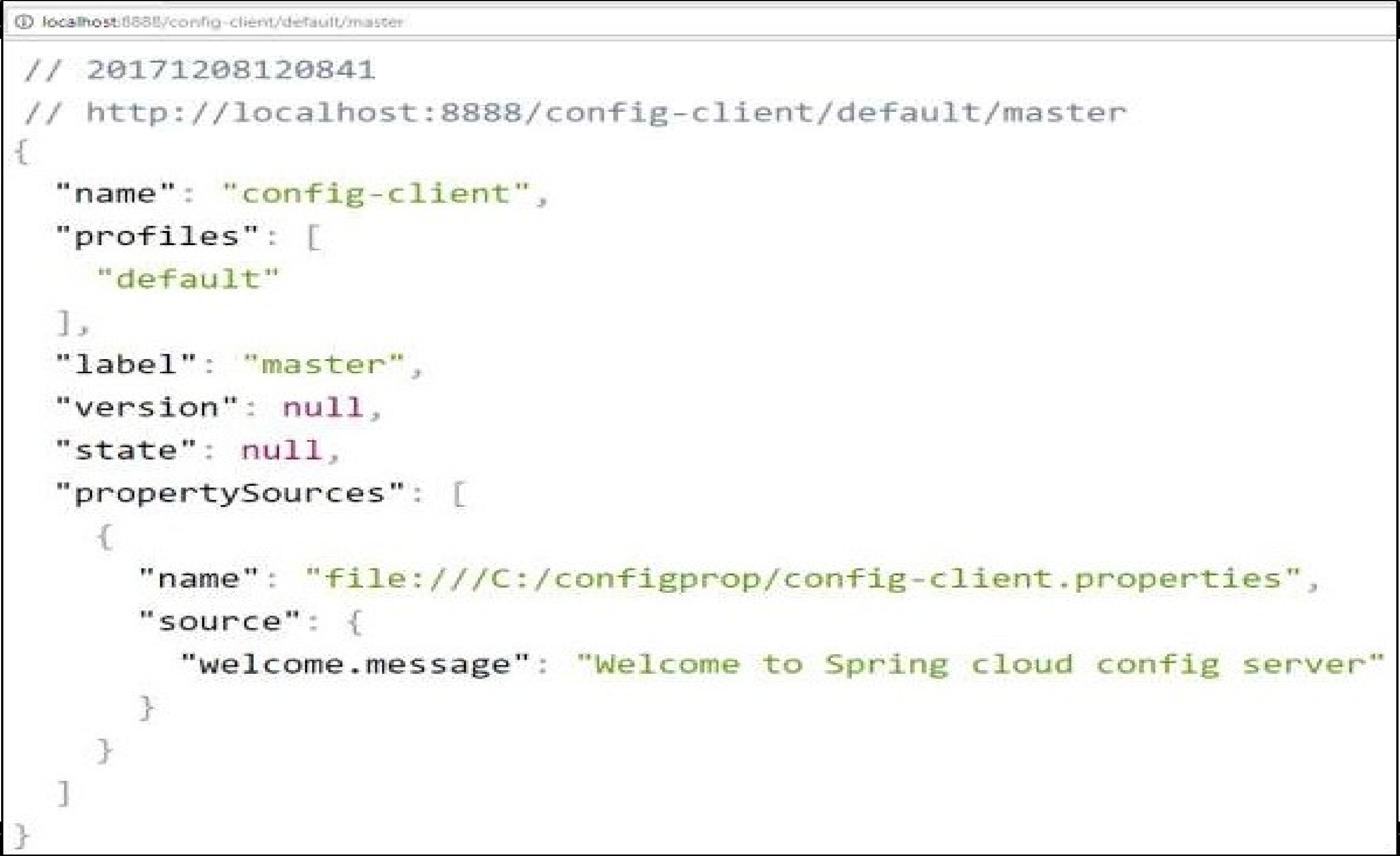
```
server.port = 8888
spring.cloud.config.server.native.searchLocations=file:///C:/configprop/
SPRING_PROFILES_ACTIVE=native
```

- Configuration Server runs on the Tomcat port 8888 and application configuration properties are loaded from native search locations.

## config-client.properties

- in **file:///C:/configprop/**, place your client application - **application.properties** file. For example, your client application name is **config-client**, then rename your **application.properties** file as **config-client.properties** and place the properties file on the path **file:///C:/configprop/**.
- The code for config-client properties file is given below –

```
welcome.message = Welcome to Spring cloud config server
```


localhost:8888/config-client/default/master

```
// 20171208120841
// http://localhost:8888/config-client/default/master
{
  "name": "config-client",
  "profiles": [
    "default"
  ],
  "label": "master",
  "version": null,
  "state": null,
  "propertySources": [
    {
      "name": "file:///C:/configprop/config-client.properties",
      "source": {
        "welcome.message": "Welcome to Spring cloud config server"
      }
    }
  ]
}
```

## Cloud Configuration Client

- Some applications may need configuration properties that may need a change and developers may need to take them down or restart the application to perform this.
- However, this might be lead to downtime in production and the need of restarting the application.
- Spring Cloud Configuration Server lets developers to load the new configuration properties without restarting the application and without any downtime.

# Cloud Configuration Client

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

- Gradle users can add the following dependency into the build.gradle file.

```
compile('org.springframework.cloud:spring-cloud-starter-config')
```

## Cloud Configuration Client

- Add the **@RefreshScope** annotation to your main Spring Boot application. The @RefreshScope annotation is used to load the configuration properties value from the Config server.

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

## Cloud Configuration Client

- Add the config server URL in your application.properties file and provide your application name.
- Note – http://localhost:8888 config server should be run before starting the config client application.

```
spring.application.name = config-client

spring.cloud.config.uri = http://localhost:8888
```

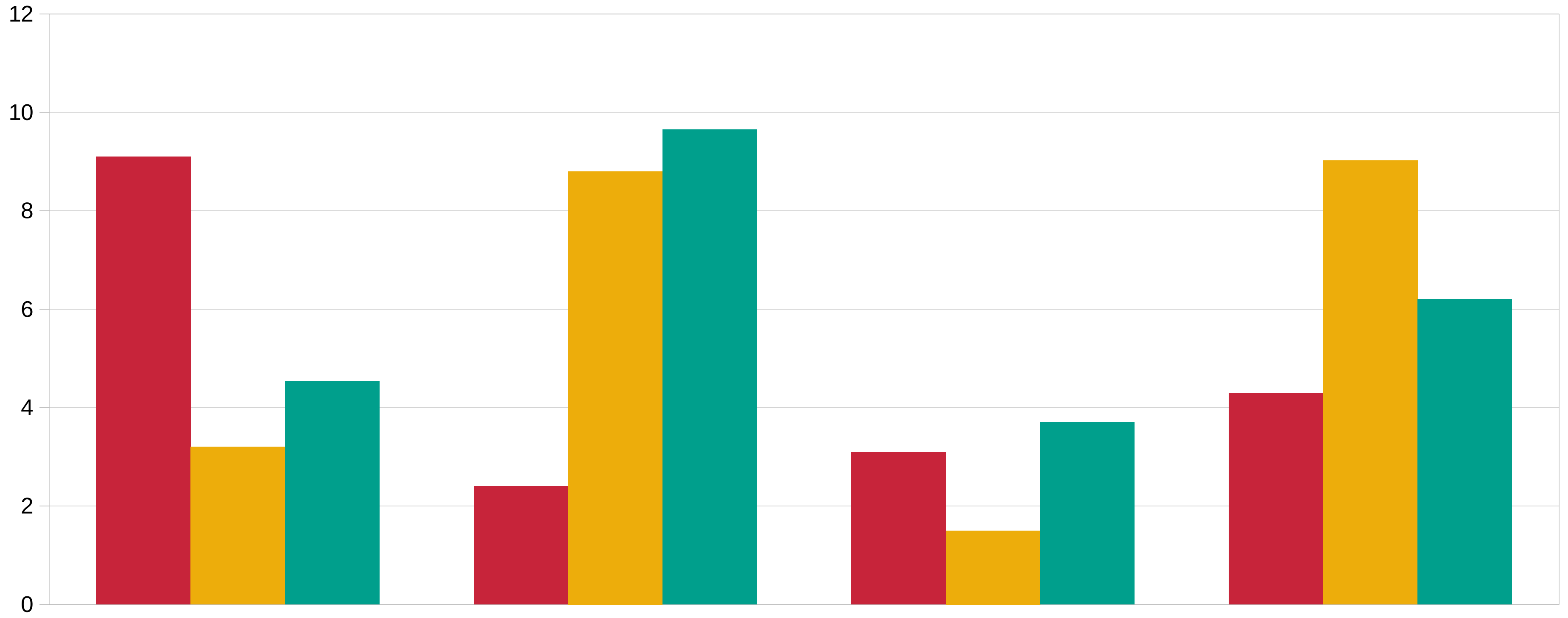
## Cloud Configuration Client

```
@RestController
public class ConfigclientApplication {
   @Value("${welcome.message}")
   String welcomeText;
   @RequestMapping(value = "/")
   public String welcomeText() {
      return welcomeText;
   }
}
```

## Cloud Configuration Client

```
2017-12-08 12:41:57.682  INFO 1104 --- [
   main] c.c.c.ConfigServicePropertySourceLocator :
   Fetching config from server at: http://localhost:8888
```

- Now hit the URL, **http://localhost:8080/** welcome message is loaded from the Configuration server.
- Now, go and change the property value on the Configuration server and hit the actuator Endpoint POST URL **http://localhost:8080/refresh** and see the new configuration property value in the URL **http://localhost:8080/**


12
10
8
6
4
2
0