Lab 1 - Exploring the Power of Spring Cloud Config

**Step 1: Set up a Spring Boot project**

1. Open your preferred Integrated Development Environment (IDE).

2. Create a new Spring Boot project.

3. Configure the necessary dependencies for Spring Boot and Spring Cloud Config in your project's `pom.xml` file.

4. Create a simple RESTful service with a controller and a sample endpoint to test the configuration.

**Step 2: Set up the Spring Cloud Config Server**

1. Create a new module in your project for the Spring Cloud Config Server.

2. Add the necessary dependencies for Spring Cloud Config Server in your `pom.xml` file.

3. Configure the Spring Cloud Config Server by creating a `@EnableConfigServer` annotated class.

4. Specify the Git repository URL containing your configuration files in the `application.properties` or `application.yml` file.

**Step 3: Create configuration files in the Git repository**

1. Set up a Git repository (e.g., GitHub, GitLab) to store your configuration files.

2. Create a new file named `application.properties` or `application.yml` in the repository.

3. Add some key-value pairs in the configuration file, for example:

```

greeting.message=Hello from Spring Cloud Config!

server.port=8080

```

4. Commit and push the configuration file to the Git repository.

**Step 4: Connect the Spring Boot application to the Spring Cloud Config Server**

1. In your Spring Boot application, add the `spring-cloud-starter-config` dependency in your `pom.xml` or `build.gradle` file.

2. Configure the Spring Cloud Config Client in your `application.properties` or `application.yml` file, specifying the Config Server's URL.

3. Annotate your main Spring Boot class with `@RefreshScope` to enable dynamic configuration updates.

4. Inject the configuration properties into your application's components using `@Value` annotations.

**Step 5: Test the Spring Cloud Config setup**

1. Start the Spring Cloud Config Server module.

2. Run your Spring Boot application.

3. Access the sample endpoint defined in your application's controller to verify that the default configuration is working.

4. Make changes to the configuration file in the Git repository and commit the changes.

5. Trigger a configuration refresh in the Spring Boot application by sending a POST request to the `/actuator/refresh` endpoint.

6. Verify that the updated configuration is reflected in your application without restarting it.

Congratulations! You have successfully completed the lab exercise on setting up and configuring Spring Cloud Config. This exercise covered the basic steps, but there are many additional features and capabilities available with Spring Cloud Config that you can explore further.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*