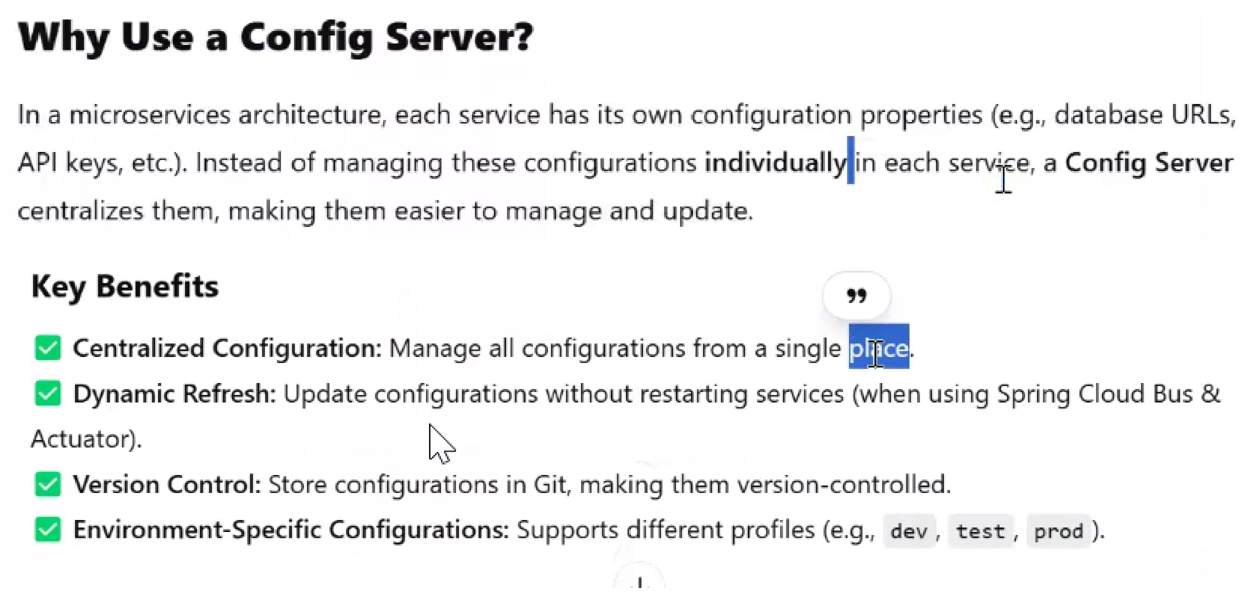
03-03-2025

Config Server

A **Config Server** is a centralized server used to manage and distribute configuration properties across multiple microservices or applications. It is commonly used in **microservices architecture** to ensure that all services get their configuration settings from a single source, making management easier and more consistent.

Key Features of a Config Server

1. **Centralized Configuration Management -** Stores configuration settings in a single place, making it easy to update and maintain.
2. **Dynamic Configuration Updates** - Microservices can fetch updated configurations without requiring redeployment.
3. **Environment-Specific Configurations -** Supports different configurations for development, testing, and production environments.
4. **Version Control Integration** - Often integrates with Git or other repositories for managing configuration files.
5. **Security & Encryption** - Supports encrypting sensitive configuration data like database credentials and API keys.



Why Use a Config Server?

In a microservices architecture, each service has its own configuration properties (e.g., database URLs, API keys, etc.). Instead of managing these configurations **individually** in each service, a **Config Server** centralizes them, making them easier to manage and update.

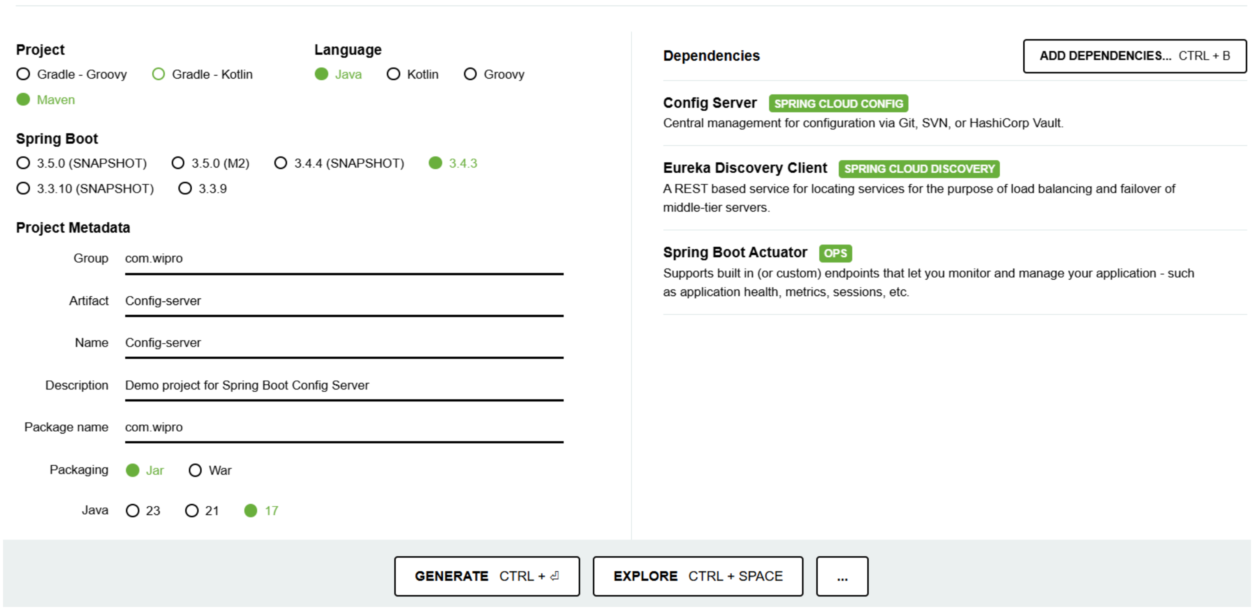
Key Benefits

✅ **Centralized Configuration:** Manage all configurations from a single place.

✅ **Dynamic Refresh:** Update configurations without restarting services (when using Spring Cloud Bus & Actuator).

✅ **Version Control:** Store configurations in Git, making them version-controlled.

✅ **Environment-Specific Configurations:** Supports different profiles (e.g., `dev`, `test`, `prod`).



Project
○ Gradle - Groovy ○ Gradle - Kotlin [✓ Maven]

Language
[✓ Java] ○ Kotlin ○ Groovy

Spring Boot
○ 3.5.0 (SNAPSHOT) ○ 3.5.0 (M2) ○ 3.4.4 (SNAPSHOT) [✓ 3.4.3] ○ 3.3.10 (SNAPSHOT) ○ 3.3.9

Project Metadata

Group	com.wipro
Artifact	Config-server
Name	Config-server
Description	Demo project for Spring Boot Config Server
Package name	com.wipro
Packaging	[✓ Jar] ○ War
Java	○ 23 ○ 21 [✓ 17]

Dependencies

Config Server SPRING CLOUD CONFIG
Central management for configuration via Git, SVN, or HashiCorp Vault.

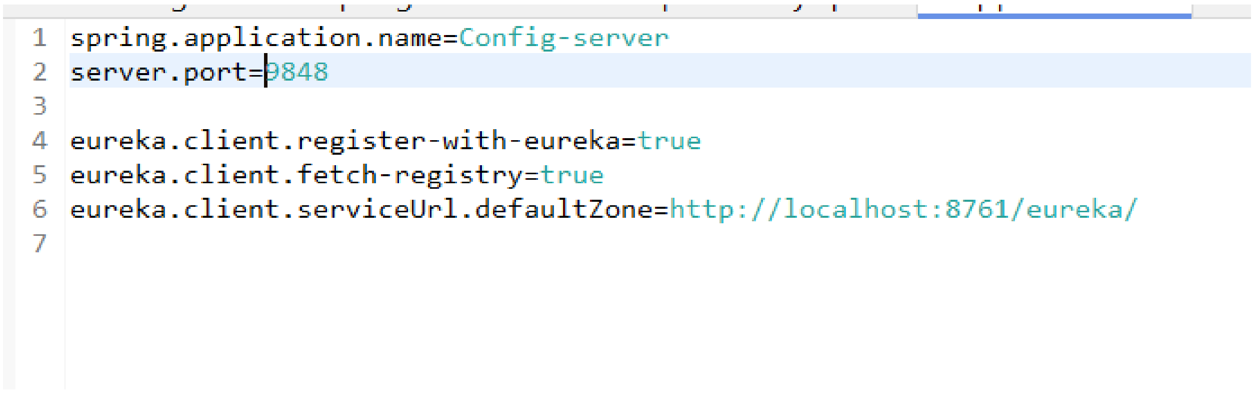
Eureka Discovery Client SPRING CLOUD DISCOVERY
A REST based service for locating services for the purpose of load balancing and failover of middle-tier servers.

Spring Boot Actuator OPS
Supports built in (or custom) endpoints that let you monitor and manage your application - such as application health, metrics, sessions, etc.

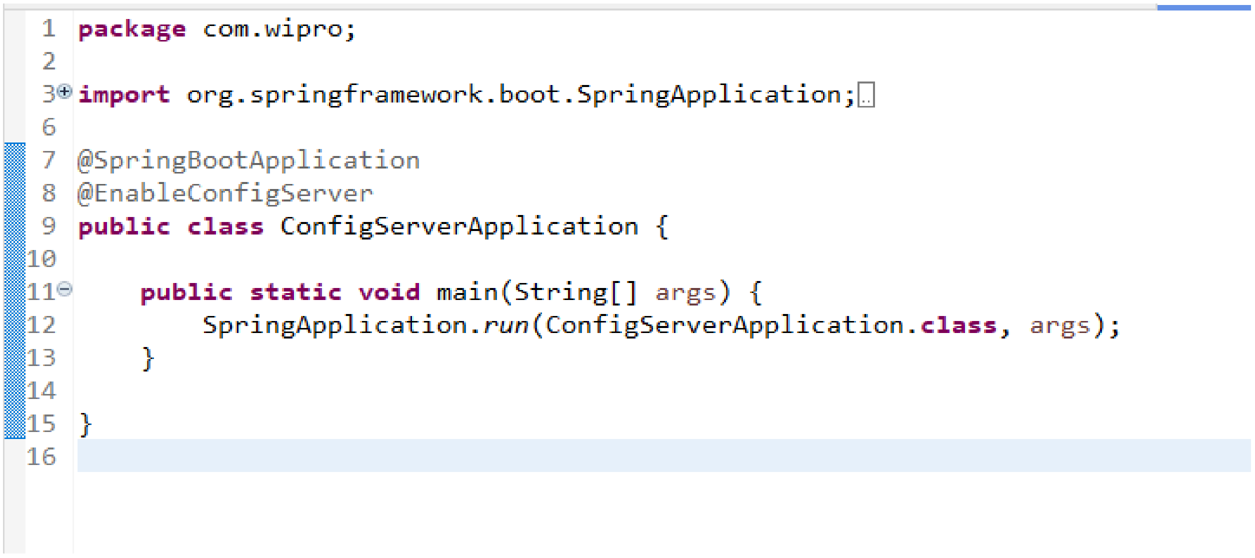
download>import >add the git location

To register with the eureka
We need to configure with this in the Application properties of configserver

eureka.client.register-with-eureka=true
eureka.client.fetch-registry=true
eureka.client.serviceUrl.defaultZone=http://localhost:8761/eureka/

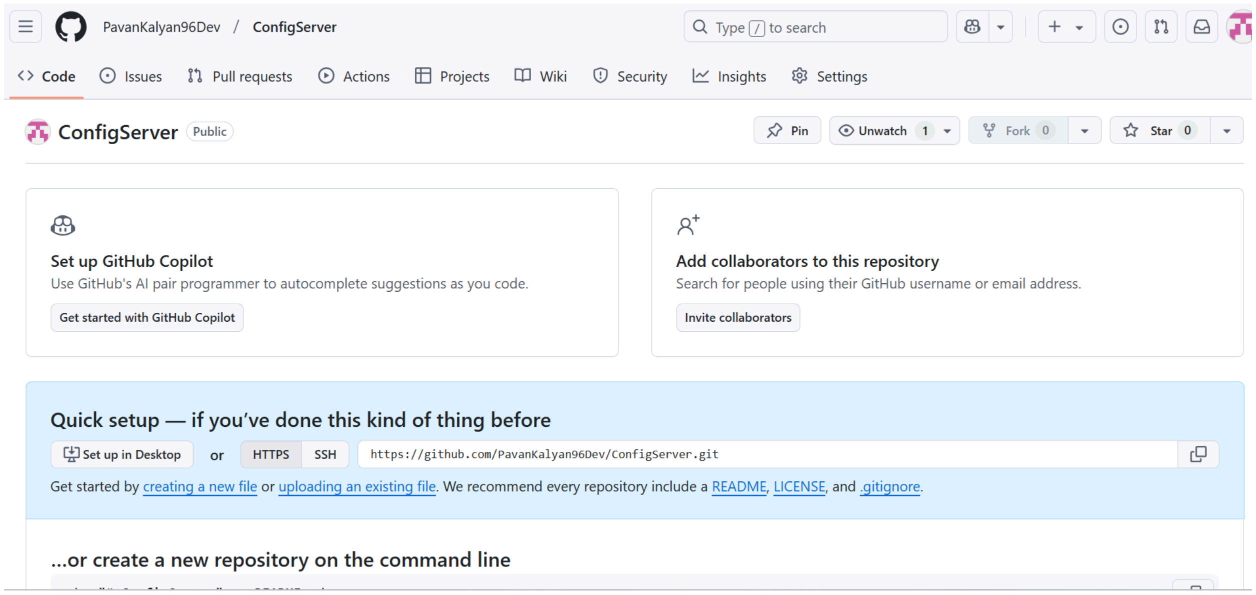


```
spring.application.name=Config-server
server.port=9848

eureka.client.register-with-eureka=true
eureka.client.fetch-registry=true
eureka.client.serviceUrl.defaultZone=http://localhost:8761/eureka/

```



```java
package com.wipro;

import org.springframework.boot.SpringApplication;

@SpringBootApplication
@EnableConfigServer
public class ConfigServerApplication {

    public static void main(String[] args) {
        SpringApplication.run(ConfigServerApplication.class, args);
    }

}
```

Now we need to set up the git location
And create the repository



ConfigServer Public

Set up GitHub Copilot
Use GitHub's AI pair programmer to autocomplete suggestions as you code.
Get started with GitHub Copilot

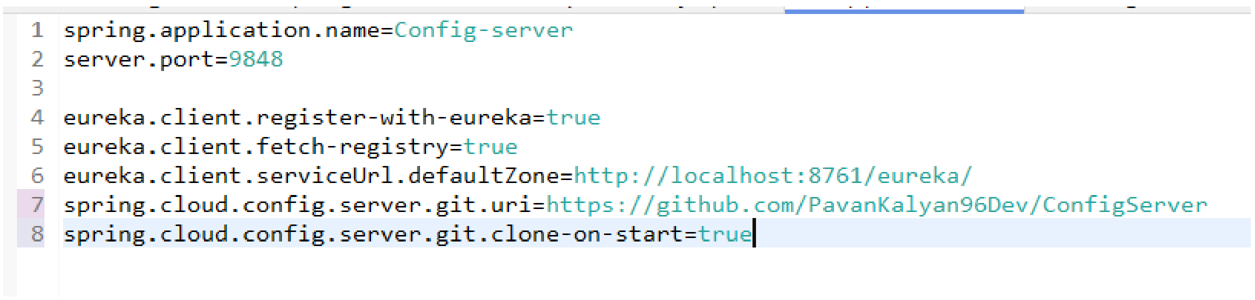
Add collaborators to this repository
Search for people using their GitHub username or email address.
Invite collaborators

Quick setup — if you've done this kind of thing before
Set up in Desktop or HTTPS SSH `https://github.com/PavanKalyan96Dev/ConfigServer.git`
Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

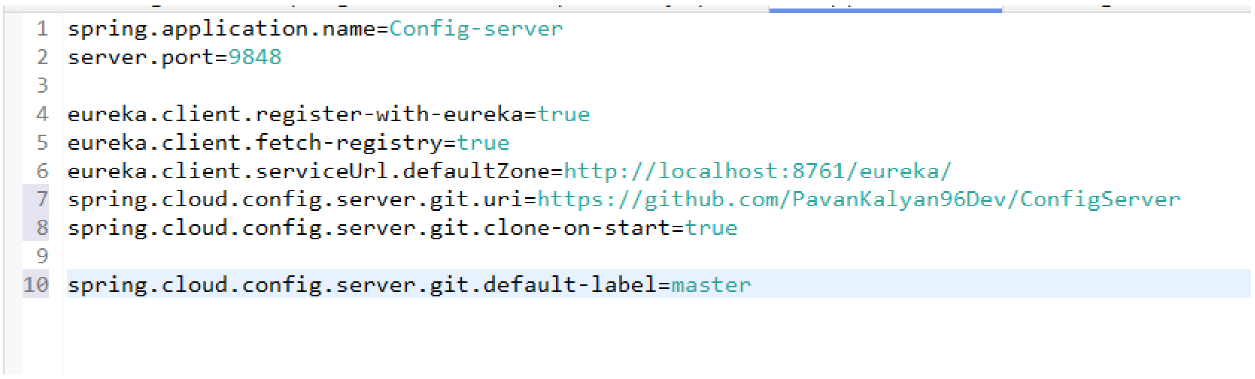
...or create a new repository on the command line

Copy that url

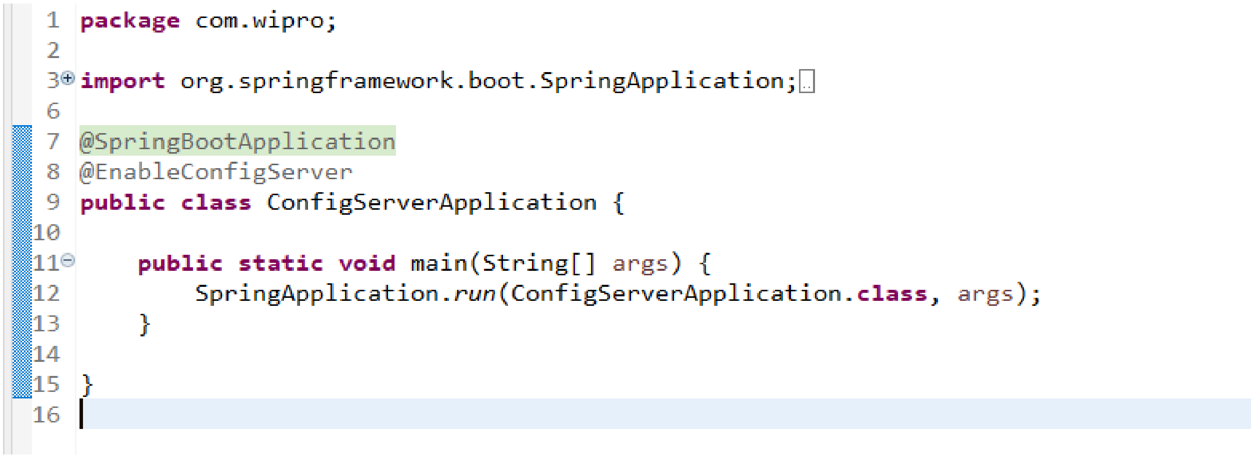
https://github.com/PavanKalyan96Dev/ConfigServer
Now configure them in the application properties of config server

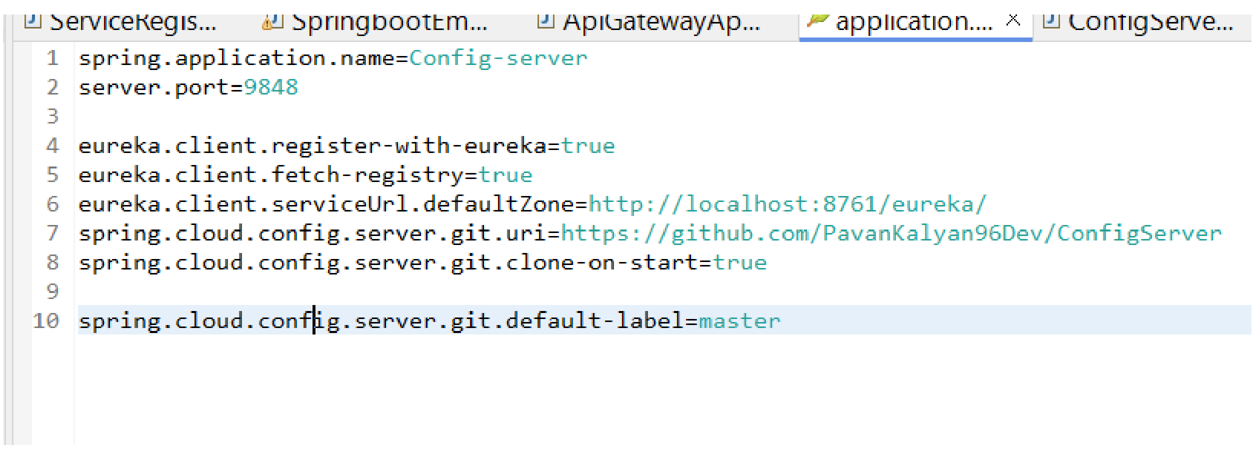

```
spring.application.name=Config-server
server.port=9848

eureka.client.register-with-eureka=true
eureka.client.fetch-registry=true
eureka.client.serviceUrl.defaultZone=http://localhost:8761/eureka/
spring.cloud.config.server.git.uri=https://github.com/PavanKalyan96Dev/ConfigServer
spring.cloud.config.server.git.clone-on-start=true
```

For which branch we need to upload now…for that one we need to specify

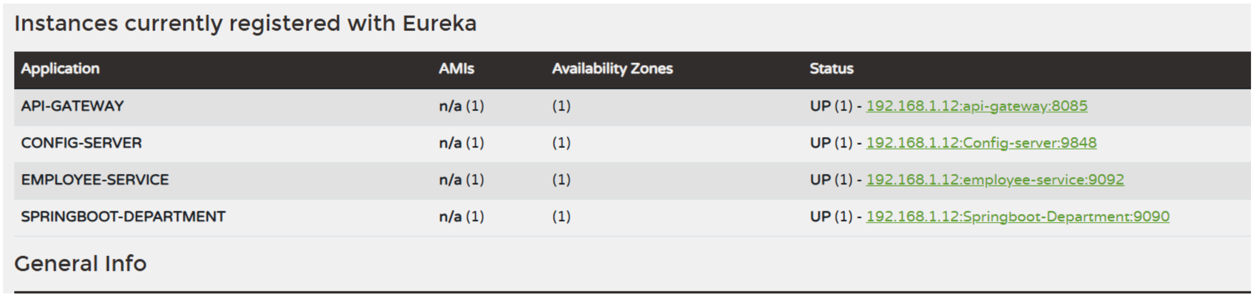

```
spring.application.name=Config-server
server.port=9848

eureka.client.register-with-eureka=true
eureka.client.fetch-registry=true
eureka.client.serviceUrl.defaultZone=http://localhost:8761/eureka/
spring.cloud.config.server.git.uri=https://github.com/PavanKalyan96Dev/ConfigServer
spring.cloud.config.server.git.clone-on-start=true

spring.cloud.config.server.git.default-label=master
```

Replace main with the master


```java
package com.wipro;

import org.springframework.boot.SpringApplication;

@SpringBootApplication
@EnableConfigServer
public class ConfigServerApplication {

    public static void main(String[] args) {
        SpringApplication.run(ConfigServerApplication.class, args);
    }

}
```



```
spring.application.name=Config-server
server.port=9848

eureka.client.register-with-eureka=true
eureka.client.fetch-registry=true
eureka.client.serviceUrl.defaultZone=http://localhost:8761/eureka/
spring.cloud.config.server.git.uri=https://github.com/PavanKalyan96Dev/ConfigServer
spring.cloud.config.server.git.clone-on-start=true

spring.cloud.config.server.git.default-label=master
```



Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
API-GATEWAY	n/a (1)	(1)	UP (1) - 192.168.1.12:api-gateway:8085
CONFIG-SERVER	n/a (1)	(1)	UP (1) - 192.168.1.12:Config-server:9848
EMPLOYEE-SERVICE	n/a (1)	(1)	UP (1) - 192.168.1.12:employee-service:9092
SPRINGBOOT-DEPARTMENT	n/a (1)	(1)	UP (1) - 192.168.1.12:Springboot-Department:9090

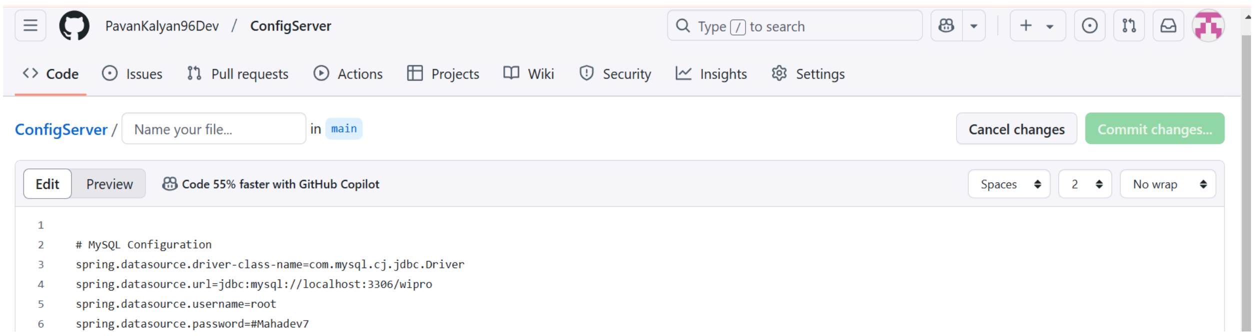
General Info

Now we should set up the department services in the server config

Now open the application properties of department service and take the properties

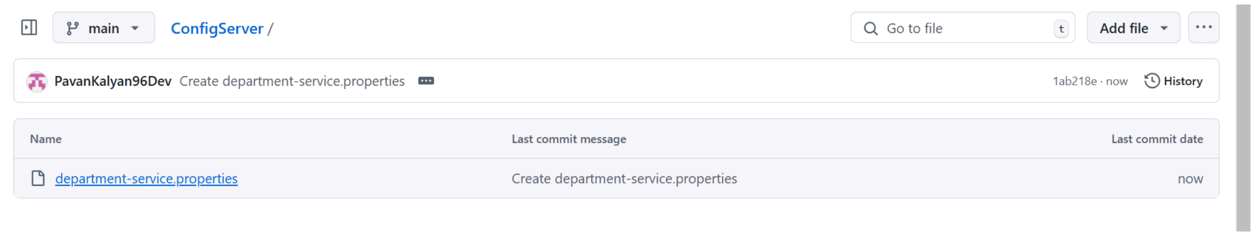
```
# MySQL Configuration
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/wipro
spring.datasource.username=root
spring.datasource.password=#Mahadev7
server.port:9090
# JPA & Hibernate
spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
eureka.client.register-with-eureka=true
eureka.client.fetch-registry=true
eureka.client.serviceUrl.defaultZone=http://localhost:8761/eureka/
```

Copy them and add it into the git



```

# MySQL Configuration
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/wipro
spring.datasource.username=root
spring.datasource.password=#Mahadev7
```

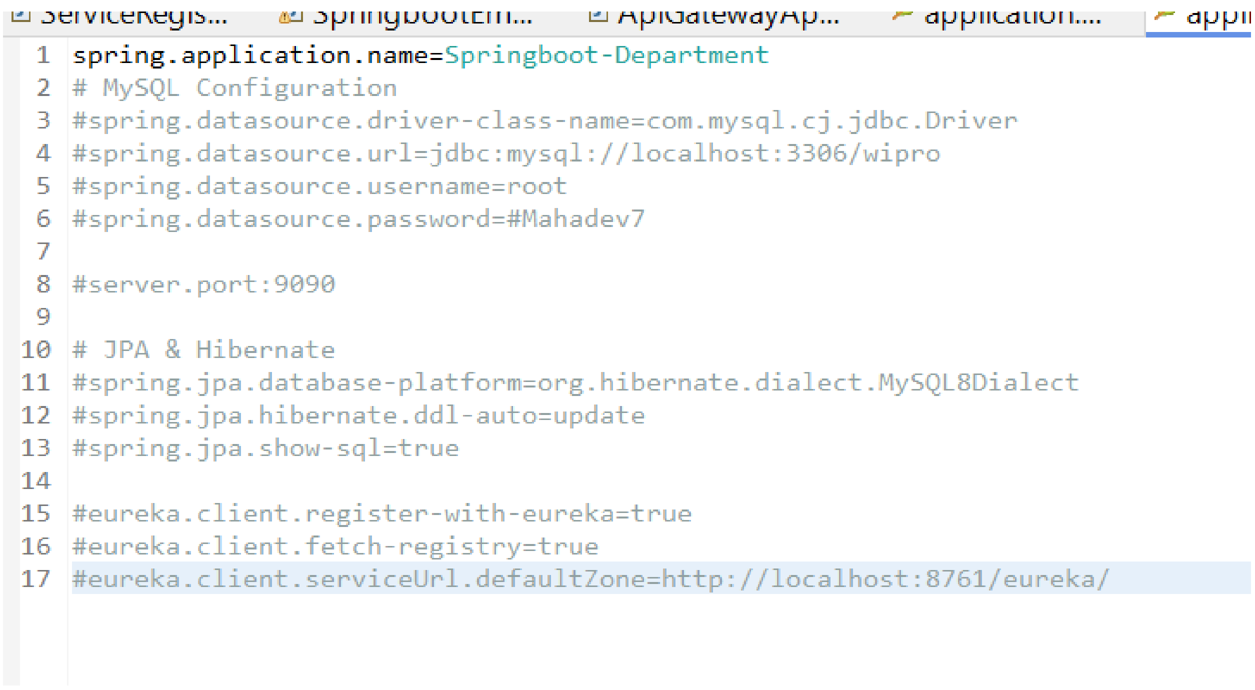
Name the file as department-service as **SPRINGBOOT-DEPARTMENT**



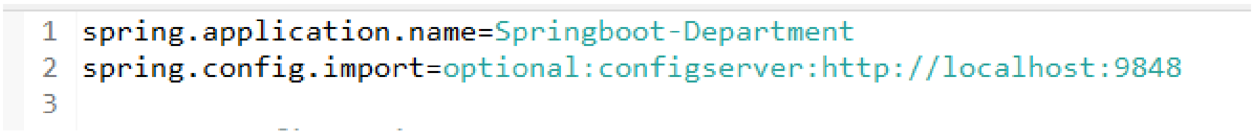
PavanKalyan96Dev Create department-service.properties 1ab218e · now

Name	Last commit message	Last commit date
department-service.properties	Create department-service.properties	now

As i configured all the information in the git..now we can comment those properties in the department properties

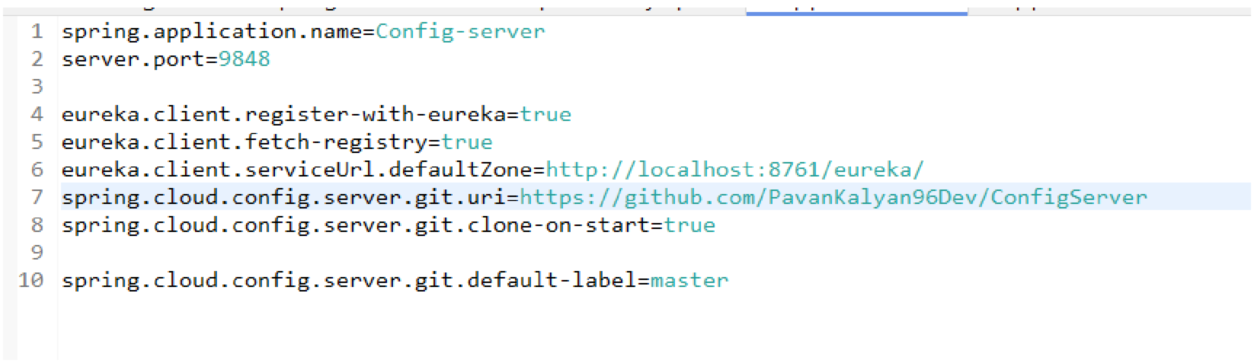


```
spring.application.name=Springboot-Department
# MySQL Configuration
#spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
#spring.datasource.url=jdbc:mysql://localhost:3306/wipro
#spring.datasource.username=root
#spring.datasource.password=#Mahadev7

#server.port:9090

# JPA & Hibernate
#spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect
#spring.jpa.hibernate.ddl-auto=update
#spring.jpa.show-sql=true

#eureka.client.register-with-eureka=true
#eureka.client.fetch-registry=true
#eureka.client.serviceUrl.defaultZone=http://localhost:8761/eureka/
```



```
spring.application.name=Springboot-Department
spring.config.import=optional:configserver:http://localhost:9848

```

Now spring boot is internally getting connected with the config server



```
spring.application.name=Config-server
server.port=9848

eureka.client.register-with-eureka=true
eureka.client.fetch-registry=true
eureka.client.serviceUrl.defaultZone=http://localhost:8761/eureka/
spring.cloud.config.server.git.uri=https://github.com/PavanKalyan96Dev/ConfigServer
spring.cloud.config.server.git.clone-on-start=true

spring.cloud.config.server.git.default-label=master
```

Add this dependency into the pom.xml file of dept service

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



   <dependency>
```

```xml
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
</dependency>
```