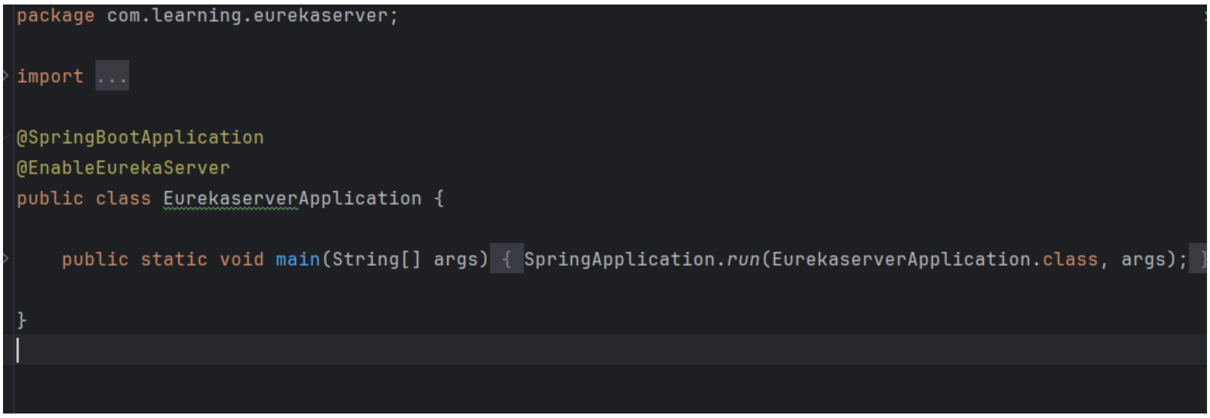
1. Create an spring boot application as EurekaServer and add the dependency **Eureka Server** or add these below lines in pom.xml file

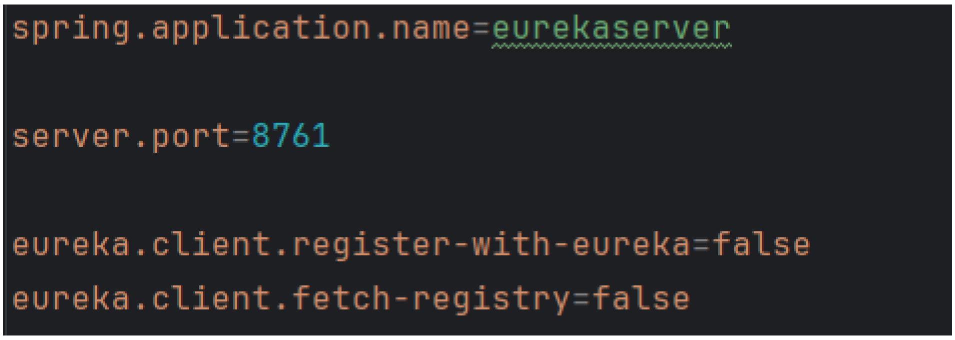
```xml
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-netflix-eureka-server</artifactId>
</dependency>
```

2. Add @EnableEurekaServer annotation in SpringBootApplication file of EurekaServer



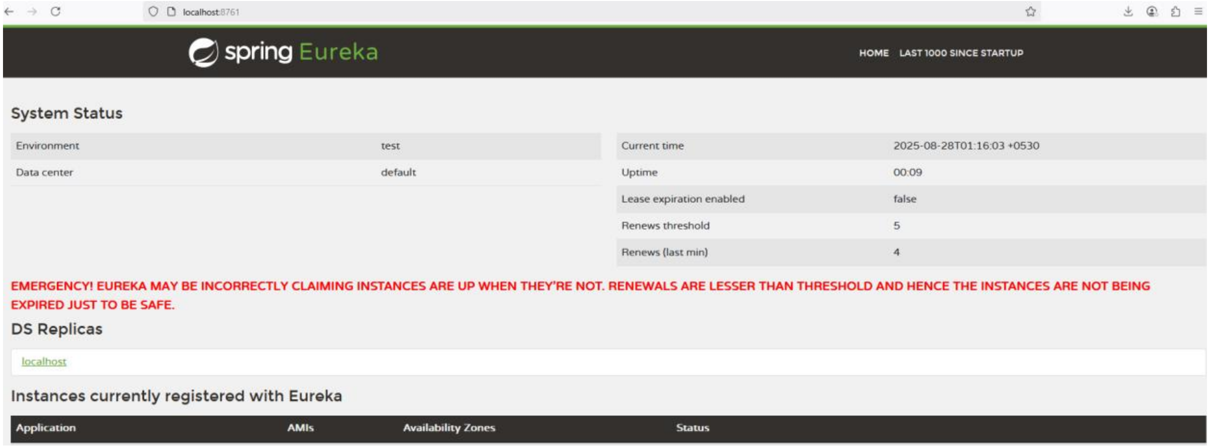
```java
package com.learning.eurekaserver;

import ...

@SpringBootApplication
@EnableEurekaServer
public class EurekaserverApplication {

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

}
```

3. Add some configuration properties related to Eureka Server in application.properties file



```properties
spring.application.name=eurekaserver

server.port=8761

eureka.client.register-with-eureka=false
eureka.client.fetch-registry=false
```

4. Run the application and verify http://localhost:8761/



System Status

Environment	test
Data center	default

Current time	2025-08-28T01:16:03 +0530
Uptime	00:09
Lease expiration enabled	false
Renews threshold	5
Renews (last min)	4

EMERGENCY! EUREKA MAY BE INCORRECTLY CLAIMING INSTANCES ARE UP WHEN THEY'RE NOT. RENEWALS ARE LESSER THAN THRESHOLD AND HENCE THE INSTANCES ARE NOT BEING EXPIRED JUST TO BE SAFE.

DS Replicas

localhost

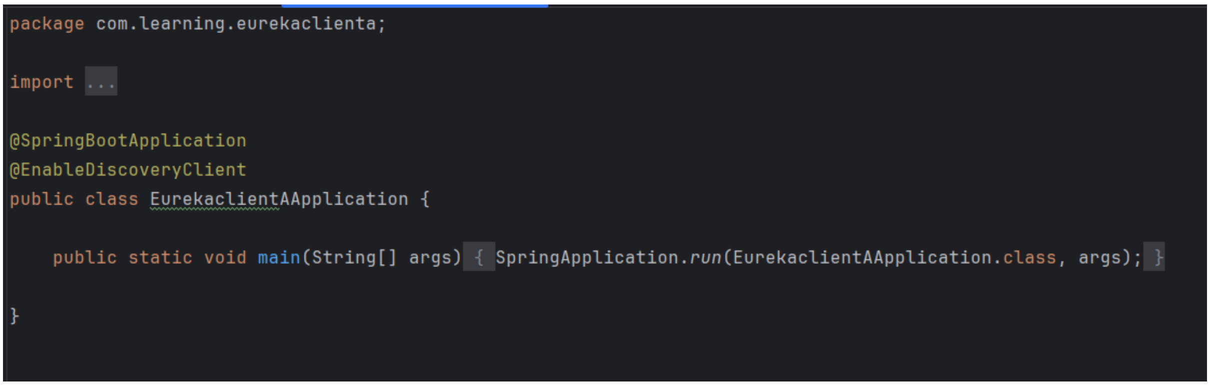
Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status

1. Create an spring boot application as EurekaClientA with the Eureka Discovery Client dependency or add these below lines in pom.xml file

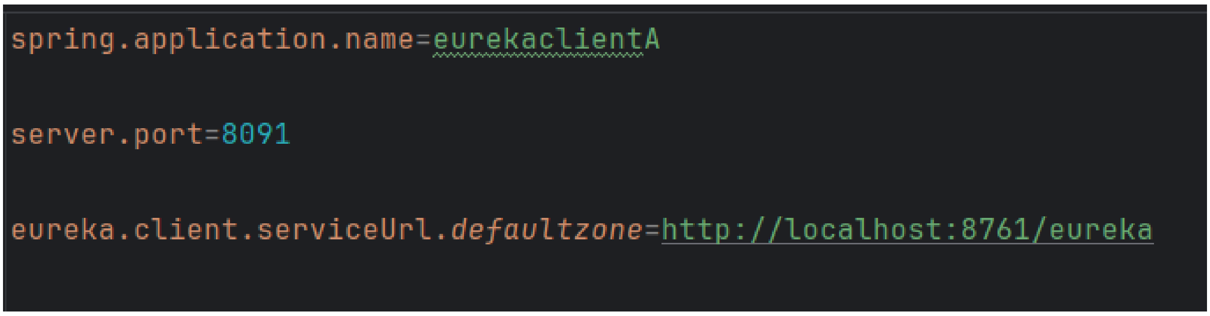
```
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-netflix-eureka-
client</artifactId>
</dependency>
```

2. Add the @EnableDiscoveryClient annotation in the Spring boot application file of EurekaClient as below:



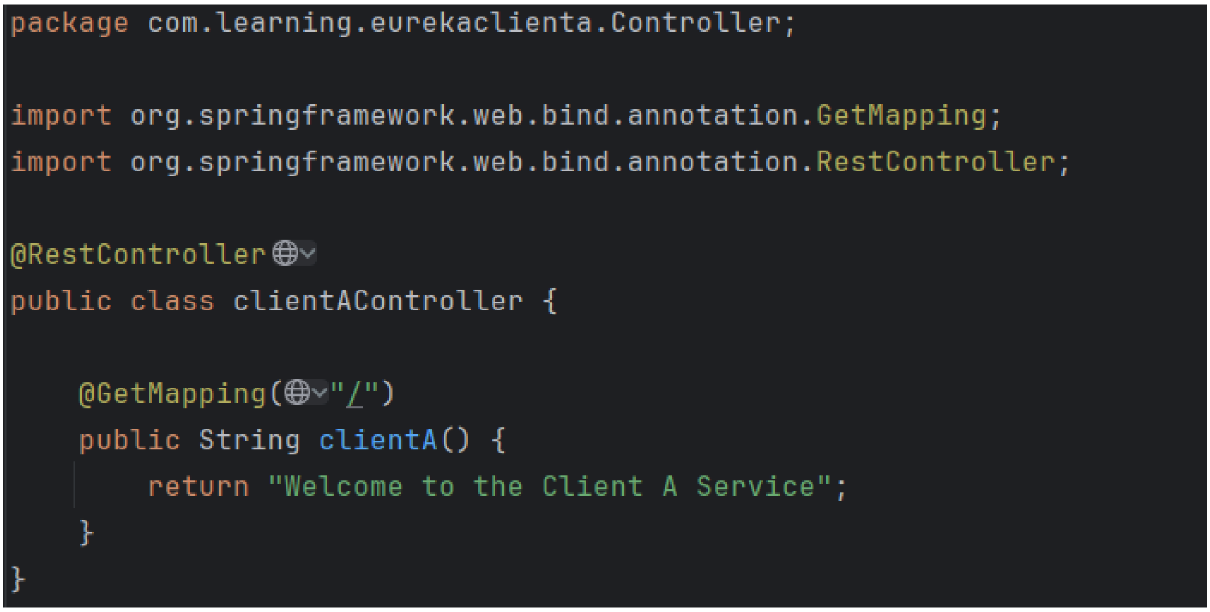
```
package com.learning.eurekaclienta;

import ...

@SpringBootApplication
@EnableDiscoveryClient
public class EurekaclientAApplication {

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

}
```

3. Add some configuration properties for eureka client microservice as below:



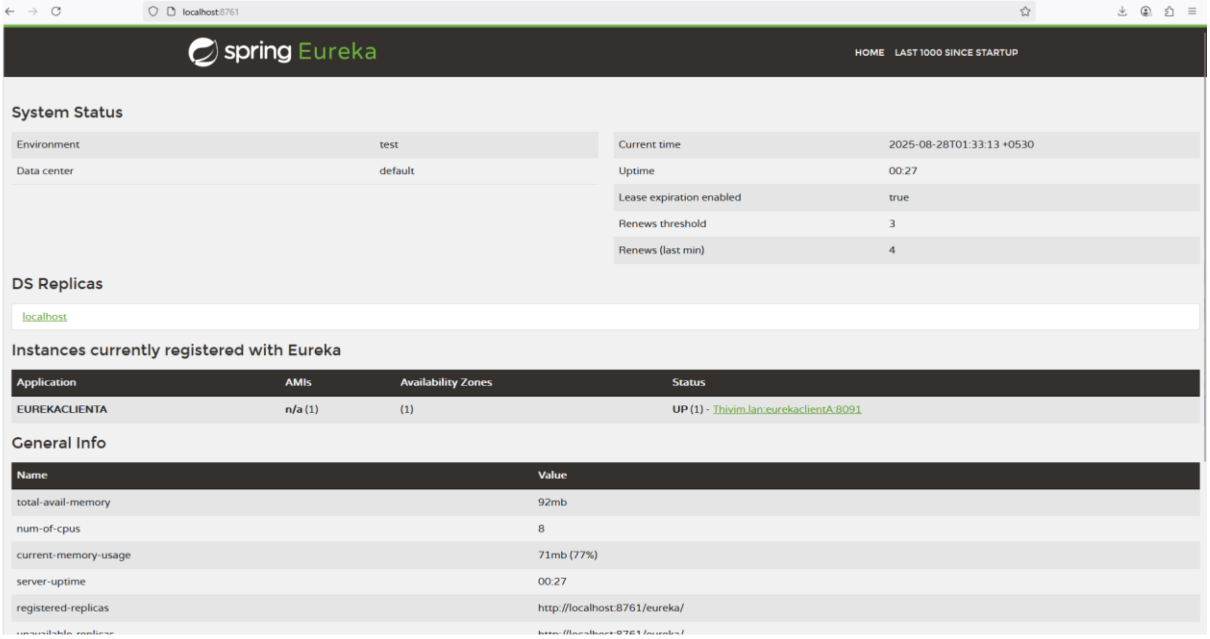
```
spring.application.name=eurekaclientA

server.port=8091

eureka.client.serviceUrl.defaultzone=http://localhost:8761/eureka
```

4. Add Controller for demo purpose



```
package com.learning.eurekaclienta.Controller;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class clientAController {

    @GetMapping("/")
    public String clientA() {
        return "Welcome to the Client A Service";
    }
}
```

5. Run the application and refresh the eureka server browser to get the location details of EurekaClientA



System Status

Environment	test
Data center	default

Current time	2025-08-28T01:33:13 +0530
Uptime	00:27
Lease expiration enabled	true
Renews threshold	3
Renews (last min)	4

DS Replicas

localhost

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
EUREKACLIENTA	n/a (1)	(1)	UP (1) - Thivim.lan:eurekaclientA:8091

General Info

Name	Value
total-avail-memory	92mb
num-of-cpus	8
current-memory-usage	71mb (77%)
server-uptime	00:27
registered-replicas	http://localhost:8761/eureka/
unavailable-replicas	http://localhost:8761/eureka/

6. As above steps, we can multiple microservices to the eureka server by creating new microservices and those configurations