Learning Consolidation

Register Microservices on a Netflix Eureka Discovery Server

Learning Objectives

- Describe Eureka Server
- Implement the service discovery server using Eureka and describe what a load balancer is

Menu

00:02 00:00:11 02/ 11

Slide Note

Eureka Server helps in maintaining a very stale ecosystem of Microservices collaborating among each other.

Eureka Server is an application that holds the information about all client-service applications.

Think of it as a lookup service where microservices (clients) can register themselves and discover other registered microservices. When a client microservice registers with Eureka it provides metadata such as host, port, and health indicator thus allowing for other microservices to discover it. The discovery server expects a regular heartbeat message from each microservice instance. If an instance begins to consistently fail to send a heartbeat, the discovery server will remove the instance from his registry. This way we will have a very stable ecosystem of Microservices collaborating among each other, and on top of it we don't have to manually maintain address of other Microservice, which is a next to impossible task if the scale up/down is very frequent, on demand and we use virtual host to host the services specially in the cloud environment.

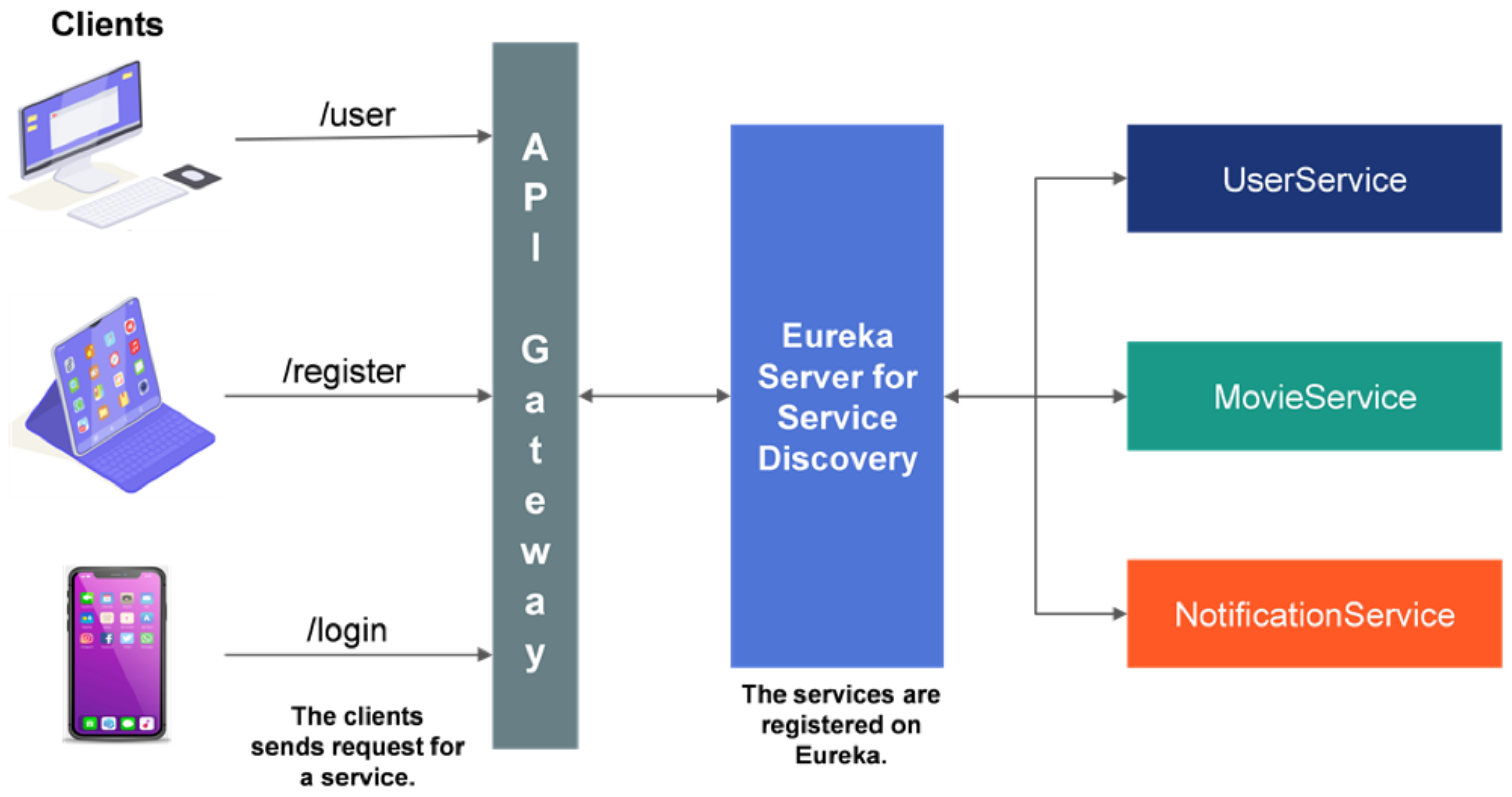
Eureka Server is an application that holds the information about all client-service applications. Every Micro service will register into the Eureka server and Eureka server knows all the client applications running on each port and IP address. Eureka Server is also known as Discovery Server.

Eureka Server

- Eureka Server acts as a registry where microservices are registered.
- The registered services can be discovered by other registered microservices for effective communication between them.
- When a microservice registers with Eureka, it provides metadata such as host, port, and health indicator thus allowing other microservices to discover it.
- Eureka Server has information of all the microservices running, like the port number and IP address of every microservice registered with it.
- Eureka Server is also known as Discovery Server.

Slide Note

Service Discovery and API Gateway

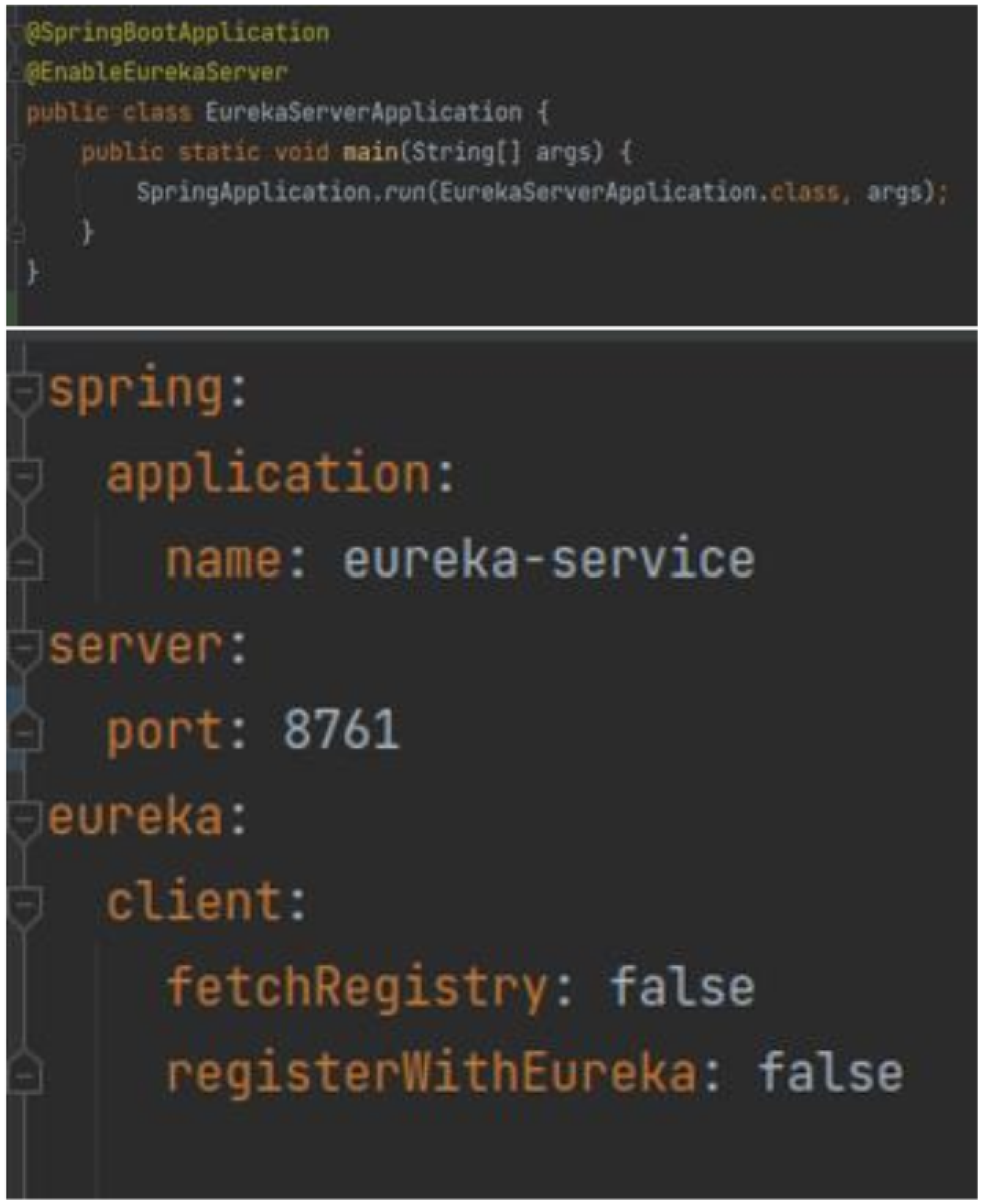

Clients
/user
/register
/login
API Gateway
Eureka Server for Service Discovery
UserService
MovieService
NotificationService
The clients sends request for a service.
The services are registered on Eureka.

Menu

00:04:00:00:11 04/ 11

Slide Note

Enable the Server in the Application



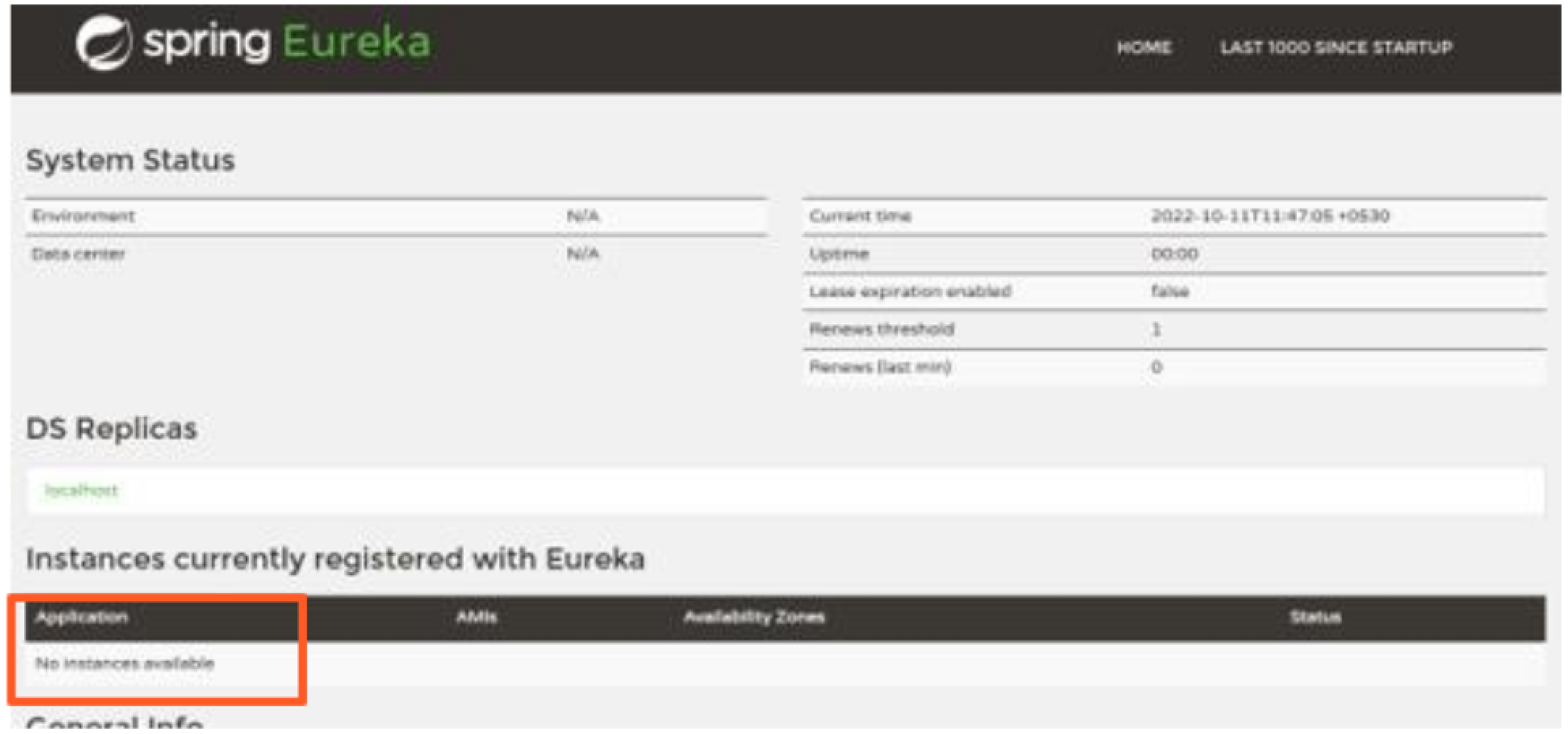
```
@SpringBootApplication
@EnableEurekaServer
public class EurekaServerApplication {
    public static void main(String[] args) {
        SpringApplication.run(EurekaServerApplication.class, args);
    }
}
```

```yaml
spring:
  application:
    name: eureka-service
server:
  port: 8761
eureka:
  client:
    fetchRegistry: false
    registerWithEureka: false
```

- Annotate the main class with `@EnableEurekaServer;` this will act as a Eureka Server.
- Mention the service name and the server port where the server will run in the `application.properties` or `application.yml` file.
- `registerWithEureka: false` tells the server not to register itself in the service registry.

Menu

Start Eureka Server



System Status

Environment	N/A
Data center	N/A

Current time	2022-10-11T11:47:05 +0530
Uptime	00:00
Lease expiration enabled	false
Renews threshold	1
Renews (last min)	0

DS Replicas

localhost

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
No instances available			

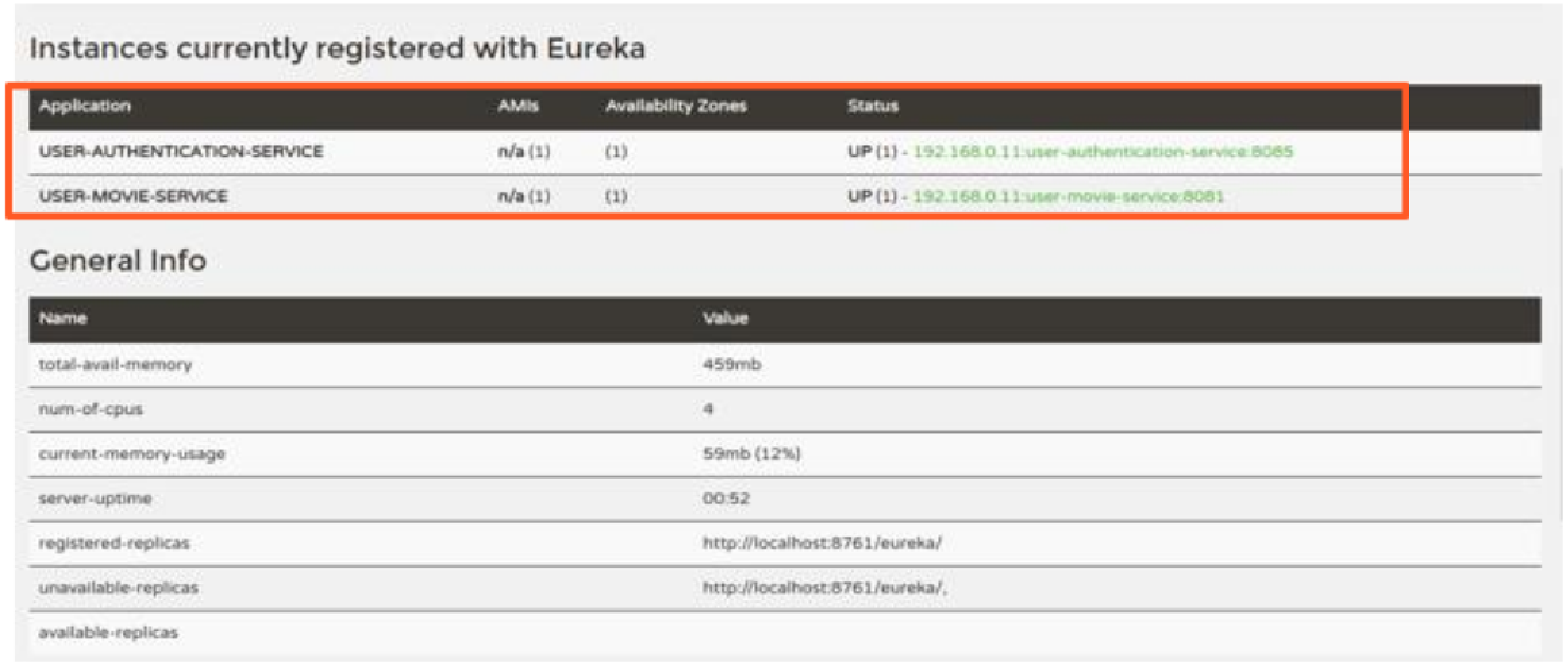
General Info

- Start the server and access the service running at http://localhost:8761/.
- At this point, no service is registered here as expected.
- Once we start the client services this server will automatically update the details of the client services

Slide Note

Eureka Server With the Registered Services

- Refresh the browser at http://localhost:8761/.



Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
USER-AUTHENTICATION-SERVICE	n/a (1)	(1)	UP (1) - 192.168.0.11:user-authentication-service:8085
USER-MOVIE-SERVICE	n/a (1)	(1)	UP (1) - 192.168.0.11:user-movie-service:8081

General Info

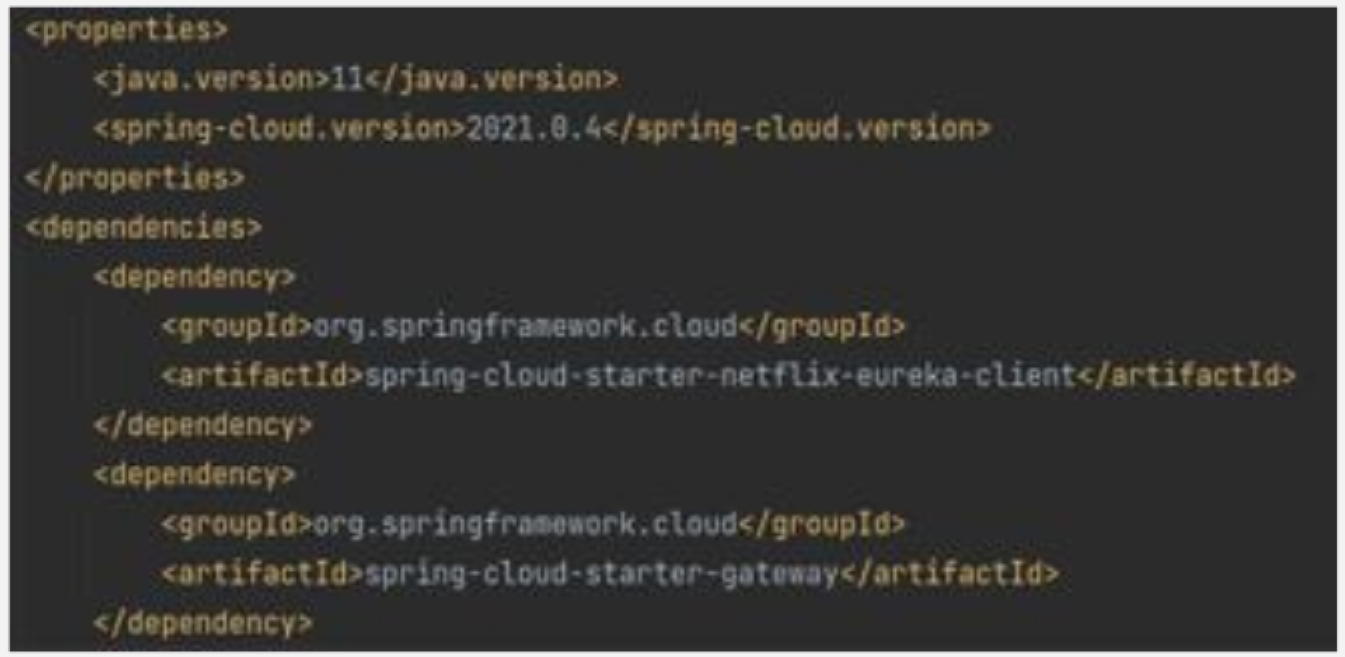
Name	Value
total-avail-memory	459mb
num-of-cpus	4
current-memory-usage	59mb (12%)
server-uptime	00:52
registered-replicas	http://localhost:8761/eureka/
unavailable-replicas	http://localhost:8761/eureka/,
available-replicas	

- Here, `UserAuthenticationService` and `UserMovieService` are Eureka clients and are getting registered with the Eureka Server.

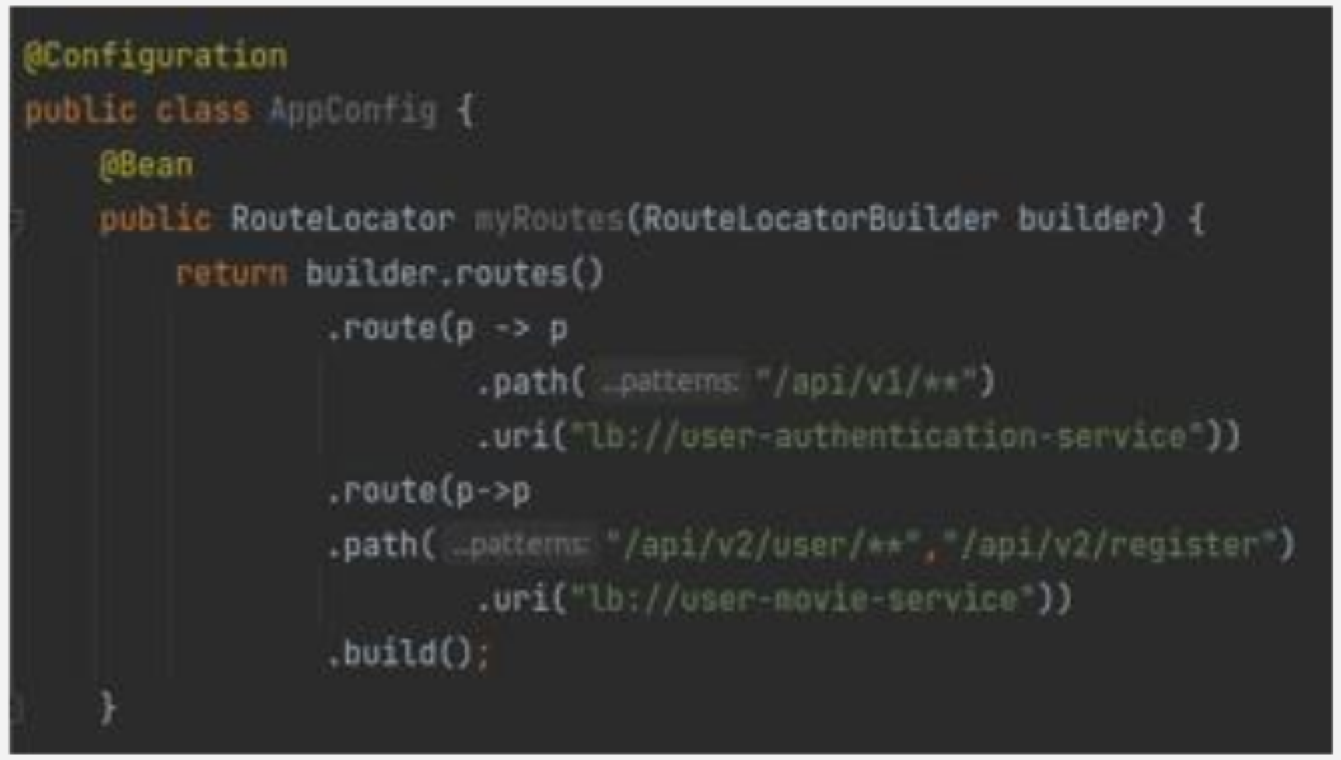
Menu 00:07:00:00:11 07/ 11

Slide Note

Register the API Gateway onto the Eureka Server



```xml
<properties>
    <java.version>11</java.version>
    <spring-cloud.version>2021.0.4</spring-cloud.version>
</properties>
<dependencies>
    <dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-gateway</artifactId>
    </dependency>
```



```java
@Configuration
public class AppConfig {
    @Bean
    public RouteLocator myRoutes(RouteLocatorBuilder builder) {
        return builder.routes()
                .route(p -> p
                        .path( ..patterns: "/api/v1/**")
                        .uri("lb://user-authentication-service"))
                .route(p->p
                .path( ..patterns: "/api/v2/user/**","/api/v2/register")
                        .uri("lb://user-movie-service"))
                .build();
    }
```

- The Spring Cloud API Gateway must be registered on the Eureka Server as a client; we need to add these dependencies.
- The route can also be written using the application name we configured in the `application.yml` file, instead of the URI of the application.
- The port number and host on which the microservice or Eureka client runs is registered on the Eureka Server, so we need not mention the URI.
- Here, `lb` stands for load balancing, which we will discuss in upcoming slides.

Menu

Slide Note

Load Balancer

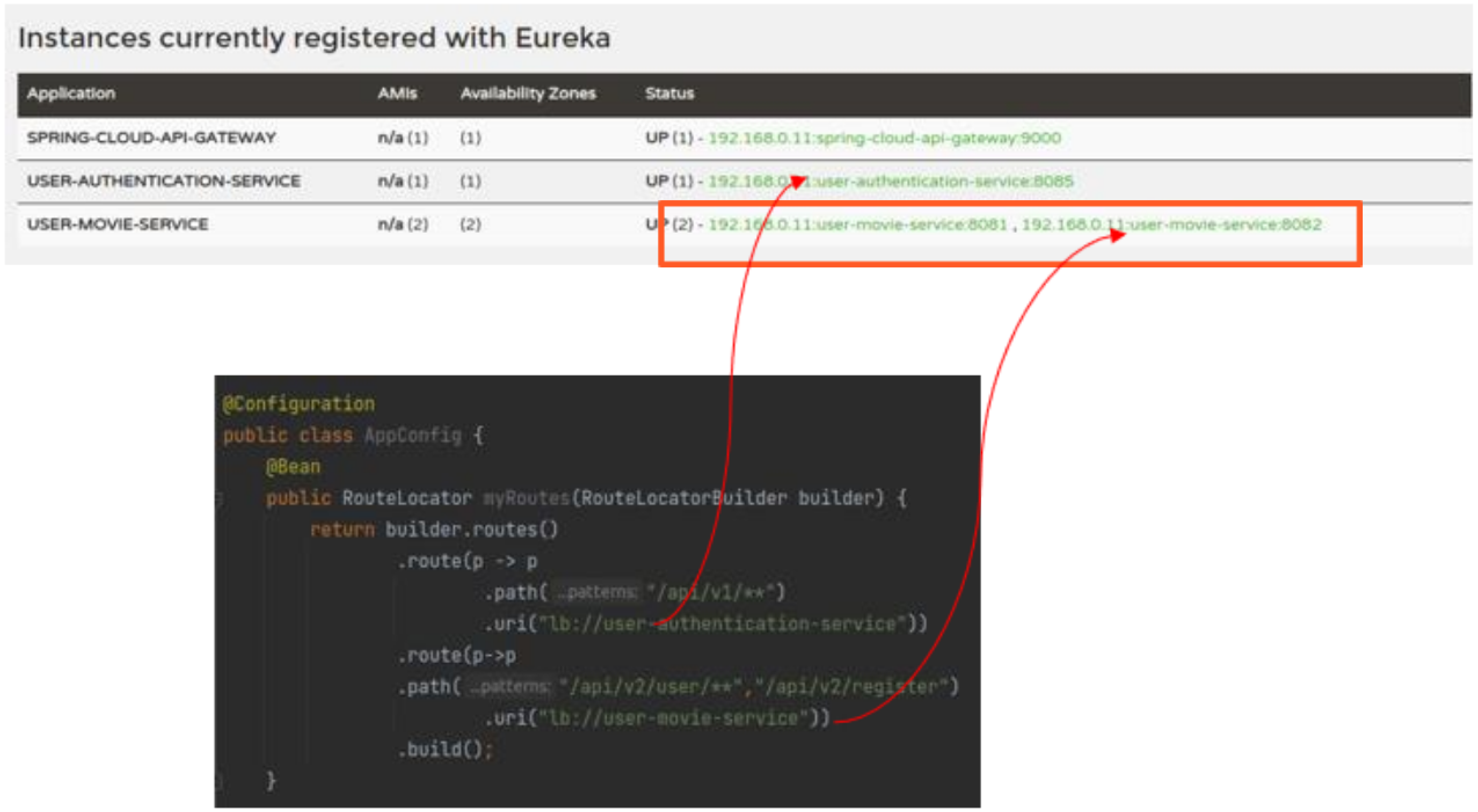
- Load balancing refers to efficiently distributing incoming request traffic across a group of backend servers.
- It acts as the traffic cop sitting in front of your servers and routing the client requests across all servers equally to ensure no one server is overworked, which could degrade the performance.
- If a server goes down, the load balancer redirects traffic to the remaining servers.
- It ensures high availability and reliability by sending requests only to servers that are online.
- Spring Cloud Gateway and Eureka make an amazing combination to scale Spring applications easily in production environments and load balance them effectively.

Slide Note

Add one point that lb is load balancer here

Eureka Server With the Registered Services

- Refresh the browser at http://localhost:8761/.



Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
SPRING-CLOUD-API-GATEWAY	n/a (1)	(1)	UP (1) - 192.168.0.11:spring-cloud-api-gateway:9000
USER-AUTHENTICATION-SERVICE	n/a (1)	(1)	UP (1) - 192.168.0.11:user-authentication-service:8085
USER-MOVIE-SERVICE	n/a (2)	(2)	UP (2) - 192.168.0.11:user-movie-service:8081 , 192.168.0.11:user-movie-service:8082

```java
@Configuration
public class AppConfig {
    @Bean
    public RouteLocator myRoutes(RouteLocatorBuilder builder) {
        return builder.routes()
                .route(p -> p
                        .path( ...patterns: "/api/v1/**")
                        .uri("lb://user-authentication-service"))
                .route(p->p
                .path( ...patterns: "/api/v2/user/**","/api/v2/register")
                        .uri("lb://user-movie-service"))
                .build();
    }
```

Menu