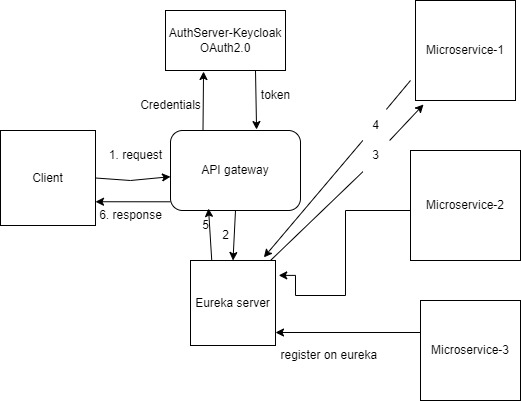
Implementing API Gateway

Spring Cloud API gateway can be positioned as an authentication and authorization control point before routing the requests to downstream services. So we can make sure that the gateway will only allow authenticated requests to be routed to downstream services (or microservices).

**Create an APIGateway Application(Spring Boot application)**



1. Include all the dependencies:

* Use this dependency for spring cloud gateway

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

* Register this on Eureka server so include the eureka dependency

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

* This is used for using the gateway as an Oauth2 client

<dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-oauth2-client</artifactId>  
 <version>2.6.6</version>  
</dependency>

1. Enable Eureka Client using @EnableEurekaClient to register this on eureka server
2. Configure Eureka Server URL

eureka:  
 instance:  
 hostname: localhost  
 client:  
 serviceUrl:  
 defaultZone: http://${eureka.instance.hostname}:8761/eureka/

1. Configuring API Gateway Routes With Spring Cloud Gateway

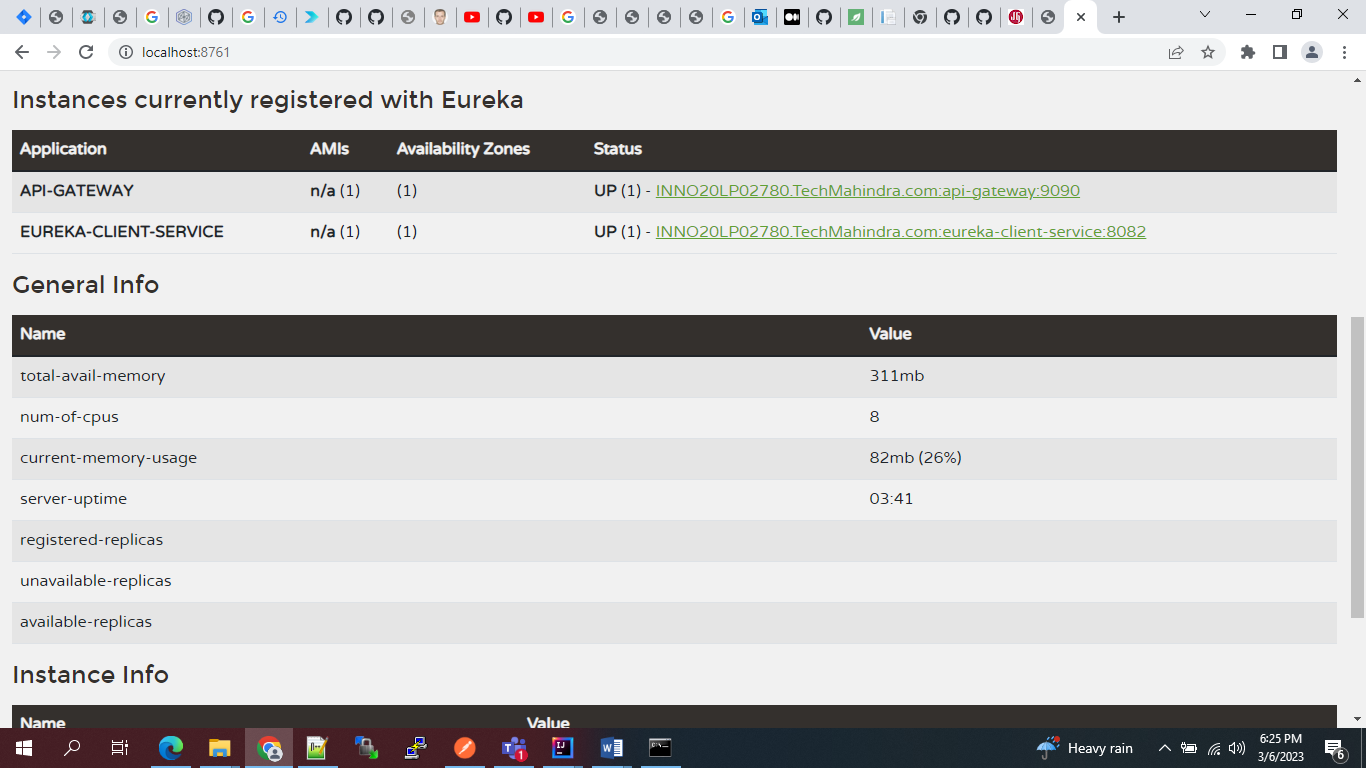
As a demo I have created the eureka-client-service application-

When a client sends a request to the API gateway, It will discover the correct service IP and PORT using the service registry to communicate and route the request to the service.

spring:  
 application:  
 name: api-gateway  
 cloud:  
 gateway:  
 routes:  
 - id: eureka-client-service  
 uri: http://localhost:8082/  
 predicates:  
 - Path=/firstService/\*\*

* id – This is just an identification of the routes , spring boot application name
* URI – <http://localhost:8082/> this is the url of the application
* predicates – In here we can set multiple paths to identify a correct routing destination. Eg:- If the API gateway gets and request like http://localhost:9090/firstService/hello  then it will be routed into <http://localhost:9090/firstService/hello>.

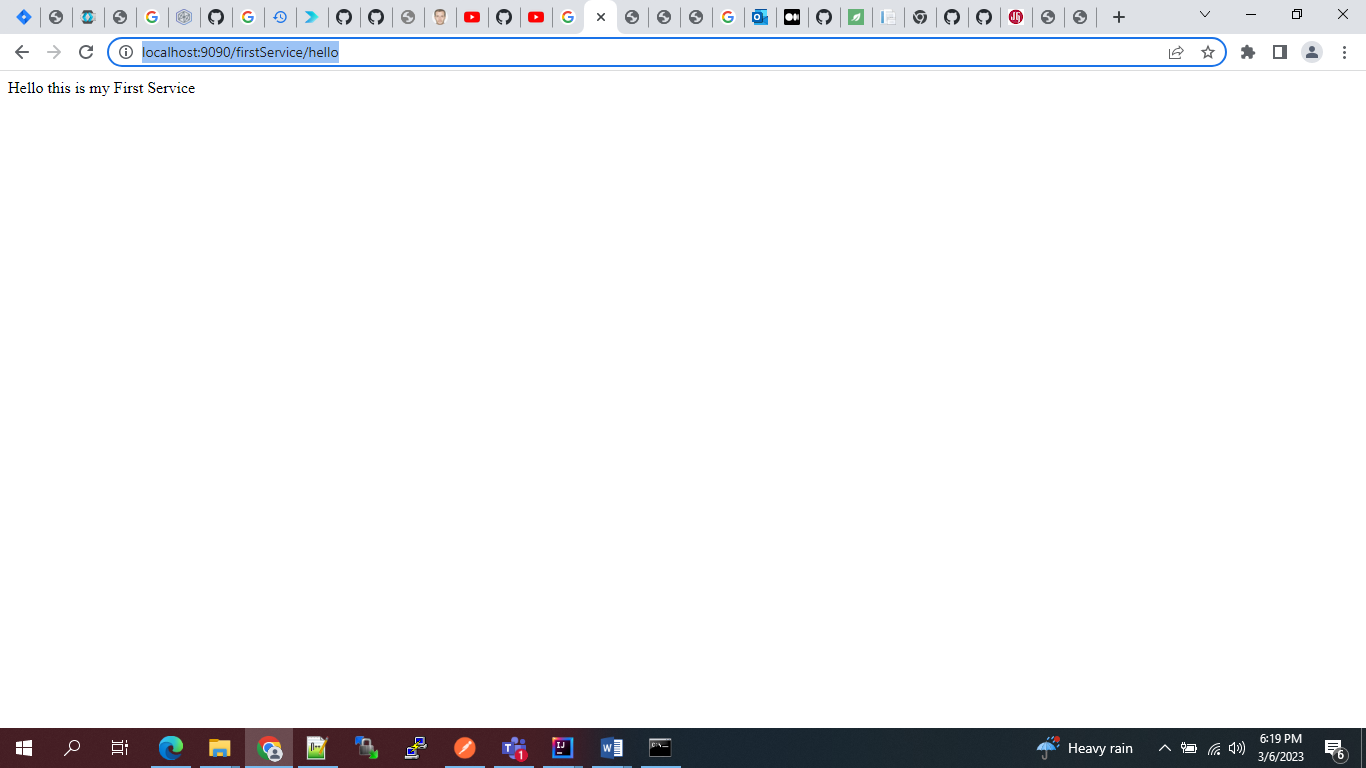
1. Verify Registered Instances in Service Registry



1. Testing API Gateway

http://localhost:9090/firstService/hello

we will be accessing the demo service using api gateway wich is running on port 9090:



**Configuring API Gateway as an oauth2 client**

When API Gateway is set as an **oauth2 client**it will redirect the user to the identity server’s login page for secured endpoints.

* 1. Include OAuth2 Client dependency
  2. we will create a simple RestController as follows:-

@RestController

public class Controller {

@GetMapping("/")

public String index(Principal principal) {

return principal.getName();

}

}

we are returning the name (Id of the Keycloak user) from the principal Object which is created by spring security once the user logs in.

* 1. create a security configuration class so that any request that comes in must be authenticated, and in case of a not logged-in user, it should use the OAuth2 login page.

@Configuration

public class SecurityConfig {

@Bean

public SecurityWebFilterChain springSecurityFilterChain ( ServerHttpSecurity http) {

http

.authorizeExchange()

.anyExchange()

.authenticated()

.and()

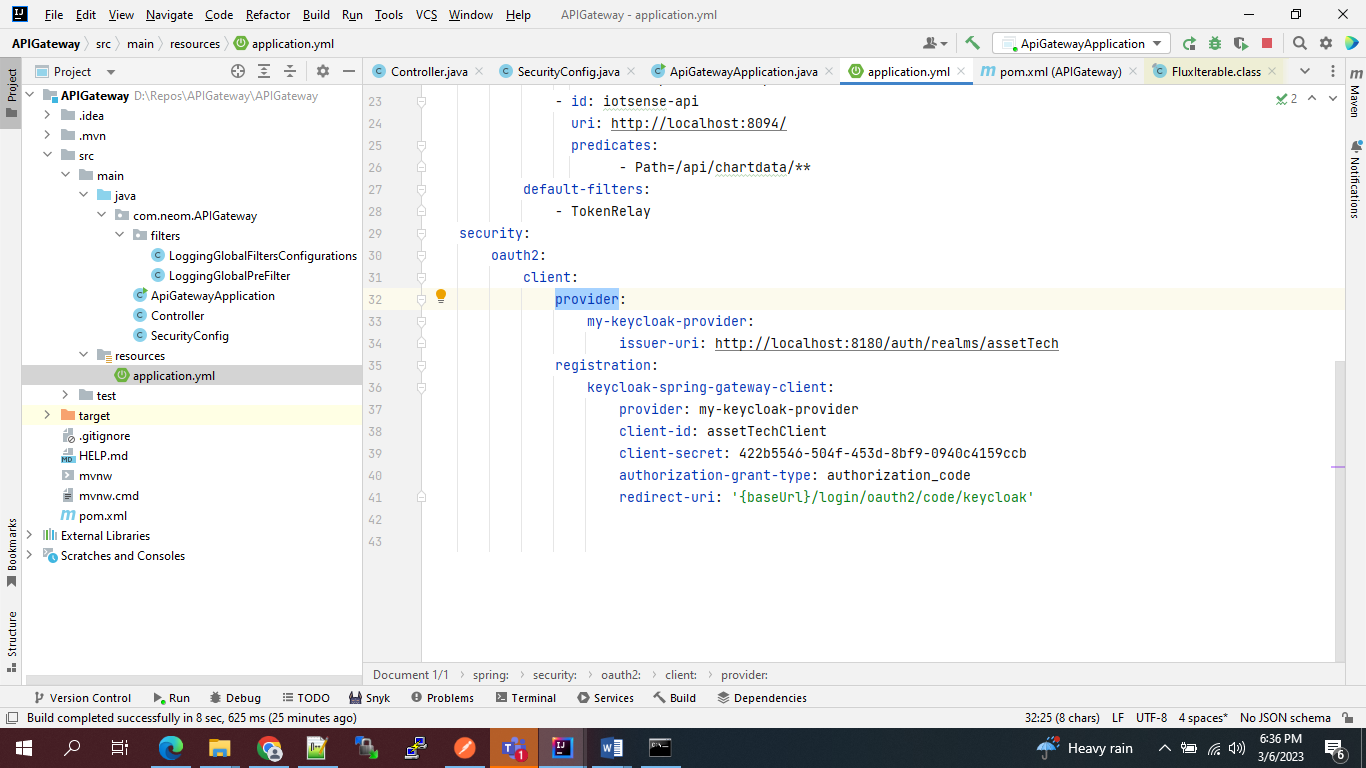
.oauth2Login(); // to redirect to oauth2 login page.

return http.build();

}

}

* 1. We set the properties to register the Oauth2 Keycloak client in our application.
     + Provider’s properties — The provider of the OAuth2 mechanism i.e the realm.
     + Client properties — These are the properties of the Keycloak client to communicate with the realm.



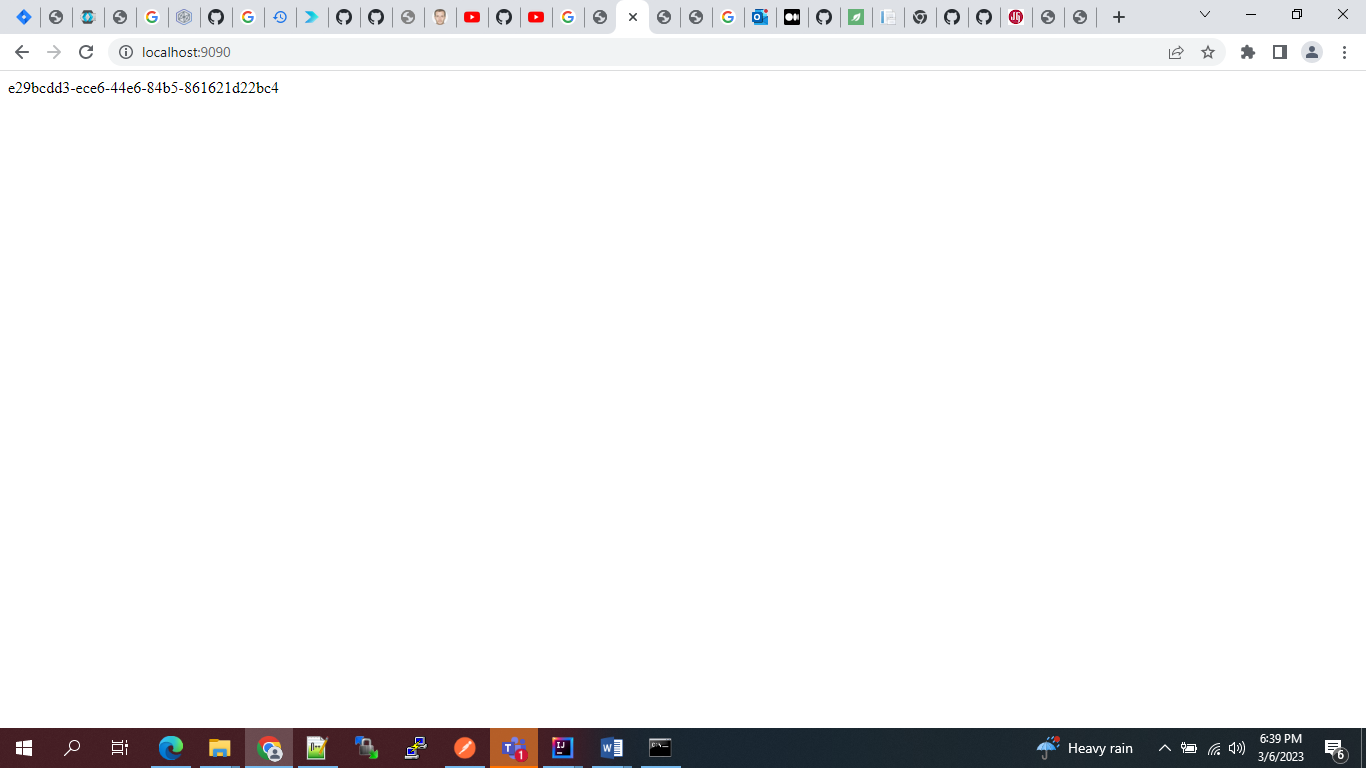
we set the client Id we created in Keycloak and the client secret from the client’s “credentials” tab in Keycloak.

We also set the provider name, from the properties before, and the redirect URI which we had registered while creating the client in the Keycloak.

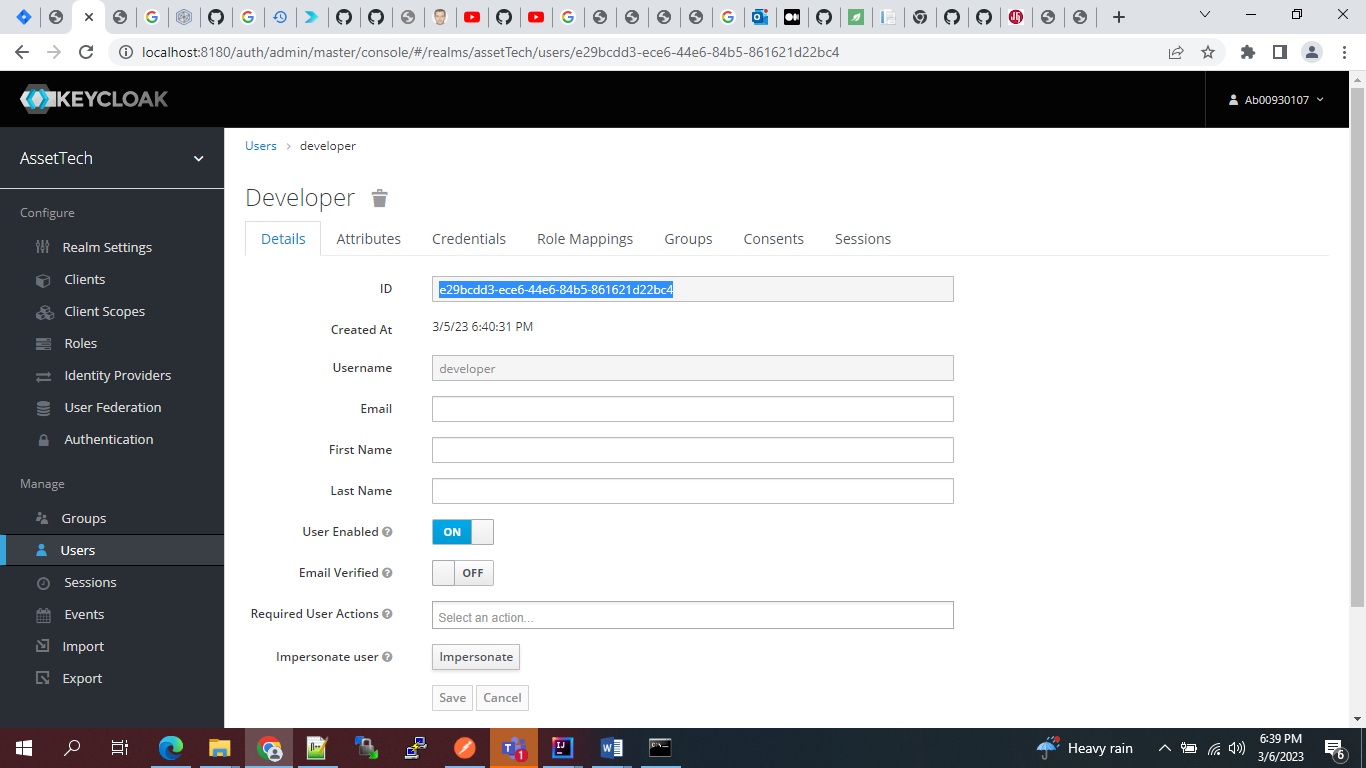
Also, Since we will be using the authorization code grant type for the OAuth2 flow, we set the authorization grant type to “authorization\_code

* 1. Starting the application

Since we set **server.port=9090**, the application starts at 9090. When we open [http://localhost:9090](http://localhost:9090/) on the web browser, It immediately redirects to the login page from Keycloak as we are querying the root resource **/**.



This is the same id of the user from the key cloak



* 1. Access the API **/api/chartdata/** from API gateway

http://localhost:9090/api/chartdata/ we get the response 200 from the api :

