Explanation on microservice:

Assume we have user miccroservice: userms

It has server port as 8081 configured in application.properties as

server.port=8081

Resource path to get all users, assume we have dummy data here:

http://localhost:8081/users

This microservice now needs to be registered in Eureka Server so lets us configure for Eureka:

This will be our service discovery where all micorservices gets registered.

Let us assume it port:

server.port=8761  
spring.application.name=eureka-server

Now to register userms in Eureka we need to add port configuration to USERMS as

spring.application.name=userms  
eureka.client.serviceUrl.defaultZone=http://127.0.0.1:8761/eureka

Now, API Gateway can be added. We can create it as microservice with configuration:

server.port=9090  
spring.application.name=apigateway  
eureka.client.serviceUrl.defaultZone=http://127.0.0.1:8761/eureka/

This will register API Gateway ms to Eureka Server. This allows us to route url paths from gate way to ms. The paths registered in Gateway (publicly exposed paths) and behind the gateway are different.

Let us register our USERMS in apigateway as:

zuul.routes.proxy-users.serviceId=userms  
zuul.routes.proxy-users.path=/proxy/\*\*

Also, add Oauth sercurity layer to API Gateway as

#OAuth  
security.oauth2.resource.user-info-uri=http://localhost:9000/user

And API Gateway Application should be annotated with @EnableResourceServer.

@SpringBootApplication  
@EnableEurekaClient  
@EnableZuulProxy  
@EnableResourceServer

When we add this we need to create new OAuthMS with port 9000 and endpoint /user which will redirect the request to Authorization server and checks if it has valid token.

To get valid token we need to configure OAuthMS as:

server.port=9000

This is all configuration we need. This will be authorization server and resource server as it has one endpoint as /user

@SpringBootApplication  
@EnableAuthorizationServer  
@EnableResourceServer  
@RestController  
public class Oauth2Application extends AuthorizationServerConfigurerAdapter {  
  
 @Override  
 public void configure(ClientDetailsServiceConfigurer clients) throws Exception {  
 clients  
 .inMemory()  
 .withClient("candy")  
 .secret("{noop}123")  
 .authorizedGrantTypes("password", "client\_credentials", "refresh\_token")  
 .scopes("read", "trust")  
 .accessTokenValiditySeconds(2400)  
 .refreshTokenValiditySeconds(24000);  
 }  
  
 @Override  
 public void configure(AuthorizationServerSecurityConfigurer security) throws Exception {  
 super.configure(security);  
 }  
  
 @Override  
 public void configure(AuthorizationServerEndpointsConfigurer endpoints) throws Exception {  
 endpoints  
 .tokenStore(tokenStore());  
 }  
  
 @Bean  
 public TokenStore tokenStore() {  
 return new InMemoryTokenStore();  
 }  
  
 @GetMapping("/user")  
 public Principal user(Principal user) {  
 System.*out*.println("\*\*\*\*\*\* oAuth Server called \*\*\*\*\*\*");  
 return user;  
 }

As of now it will store token in memory.

If we have multiple instances deployed the token should be stored in DB.

Here client-id = candy and grant\_types= password, client\_credentials, refresh\_token. Refresh token is used when actual token is expired.

Also, we can configure scope to read, trust, write etc.

@Override  
 public void configure(ClientDetailsServiceConfigurer clients) throws Exception {  
 // here we can write java code to store or get client ids from DB

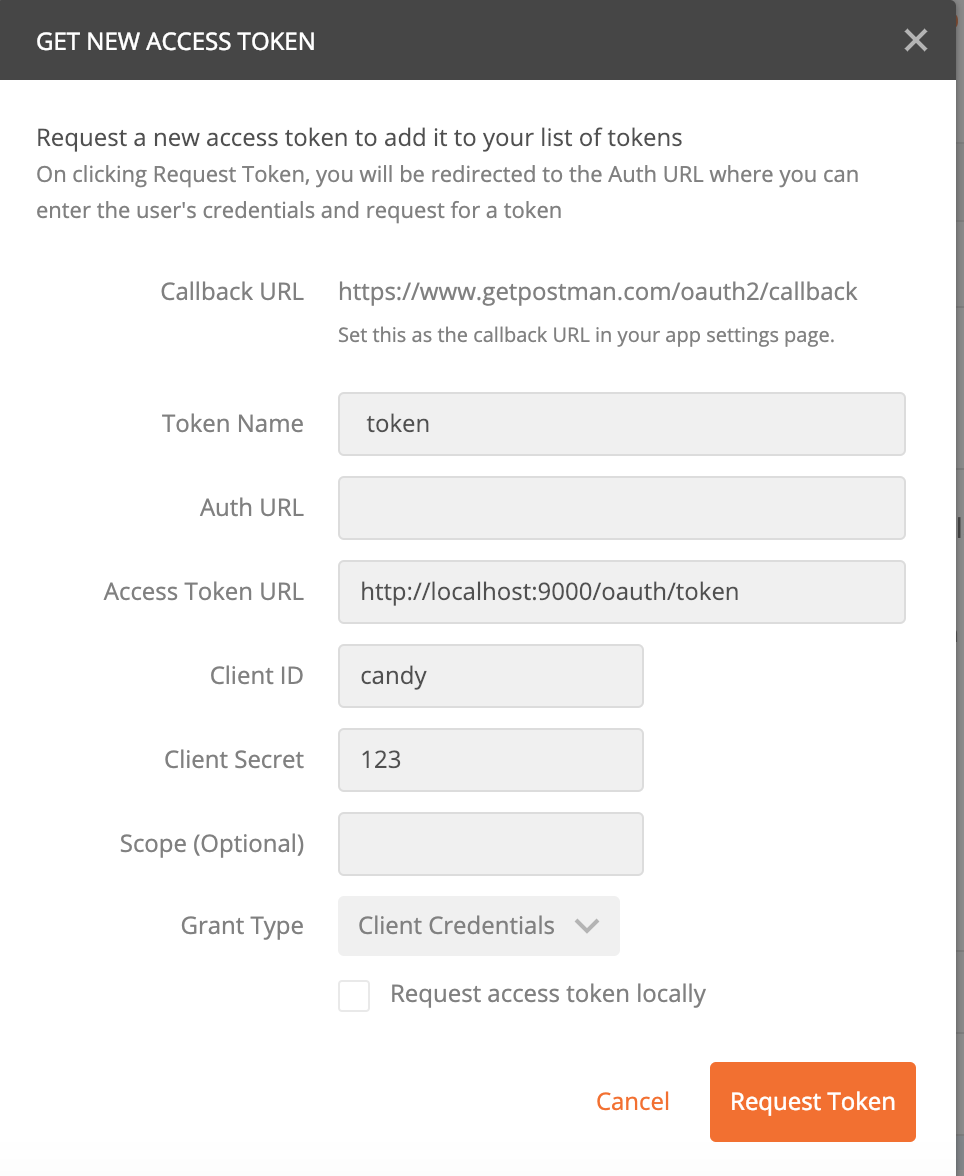
clients;

}

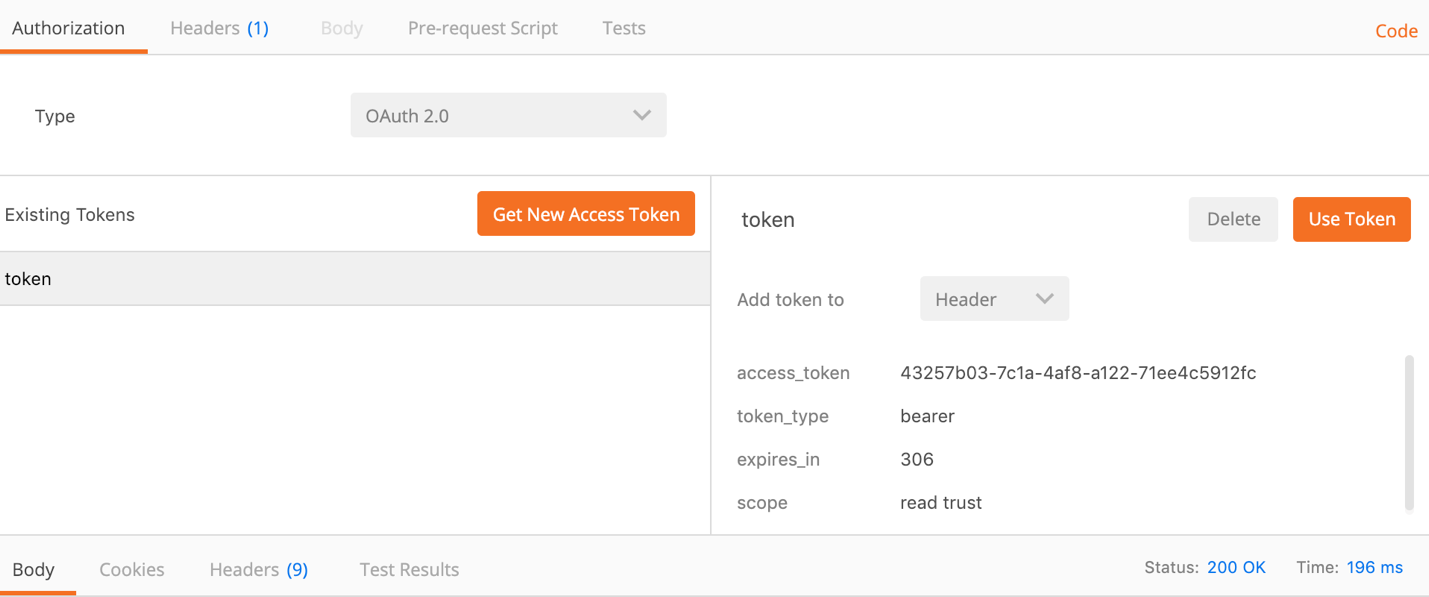
Now to get token you need the url:

http://localhost:9000/oauth/token

with configurations:



And you will get token like this:



to get users in USERMS your url will be:

http://localhost:9090/proxy/users

with Header:

Header value

Authorization Bearer 43257b03-7c1a-4af8-a122-71ee4c5912fc