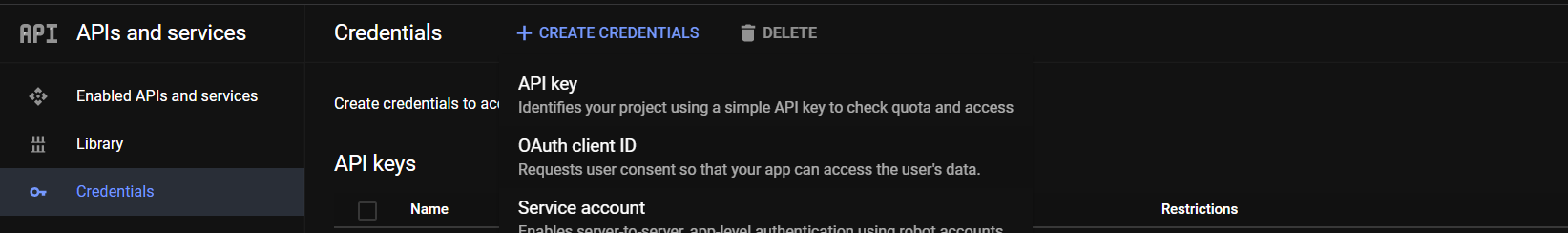
1. **Google Oauth**

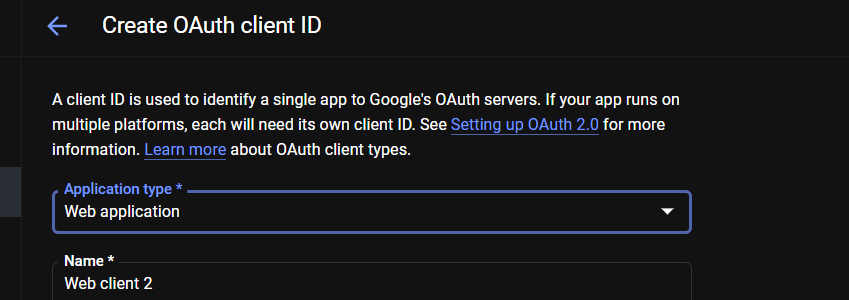
Add Oauth2 Client dependency to your spring boot starter project.

Login to Google cloud console and register yourself.

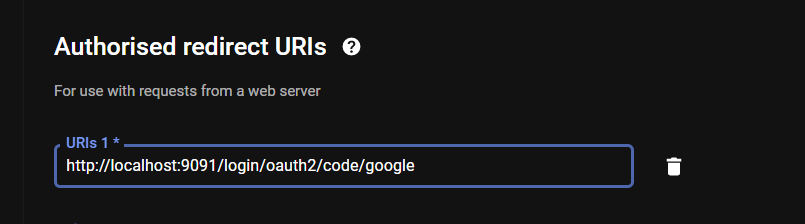
Navigate to Credentials, and click on create credentials and select oauth client id.



Select Application type as Web Application



In the redirect uri add the below uri:



You can change the port number on which the service would be hosted, you can even add <https://localhost:9091> and we can add multiple redirect uris if we are using load balancing by changing the port number if we are running on different ports locally.

If we are not running on localhost the redirect uri would be:

{baseUrl}/login/oauth2/code/google.

Once this process is completed and client id and client secret are generated, paste those with the following in the application.yml file

spring:

security:

oauth2:

client:

registration:

google:

client-id: google-client-id

client-secret: google-client-secret

Once the above process has been completed, permit oauth2login in your security config file.

http

.csrf().disable()

.authorizeRequests().antMatchers("/login").permitAll()

.anyRequest().authenticated()

.and()

.formLogin().loginPage("/login").usernameParameter("email").permitAll()

.and()

.oauth2Login().loginPage("/login").permitAll()

.and()

.logout()

.invalidateHttpSession(true)

.clearAuthentication(true)

.deleteCookies("JSESSSIONID")

.logoutSuccessUrl("/login");

And add

<a href=”/oauth2/authorization/google> Sign In with Google</a>

So in the login page when the user clicks on the hyperlink they will be redirected to google for SSO.

1. **LoadBalancing**

To LoadBalance the project we require three components

1. Registry to register our gateway and our services
2. Gateway component to redirect
3. Host our services on multiple instances(Multiple Ports if we are running locally).

**Registry Component**:

Add Eureka Server Dependency.

Annotate the class with EnableEurekaServer.

In the properties file give the application name, server port as 8761 and the fetch register and register with client eureka properties as false.

Start the Registry Service.

**Hosting Services**:

Add Eureka Discover Client Dependency

Annotate the class with EnableEurekaClient.

In the properties file add eureka.instance.hostname=localhost to ensure the redirect uri doesn’t mismatch with the google’s oauth2 redirect uri.

Start the service and change the port and start the service once again. So here we have our main service running on different ports

**Gateway Component**:

Add Spring Gateway dependency and eureka discover client dependency.

Annotate the class with EnableEurekaClient.

In the Yml file add the below properties:

Spring:

Cloud:

Gateway:

Routes:

-id:id-1

Uri:http://localhost:9091

Predicates:

-path : /\*\*

-id:id-2

Uri:http://localhost:9092

Predicates:

-path : /\*\*

Change the uris to https and ssl properties and the below additional property in the yml file if the services hosted are using https protocol.

httpclient:

ssl:

useInsecureTrustManager: true

Change the port also to 8080.

Now when we hit <http://localhost:8080> or <https://localhost:8080> our request would be redirected to either 9091 or 9092