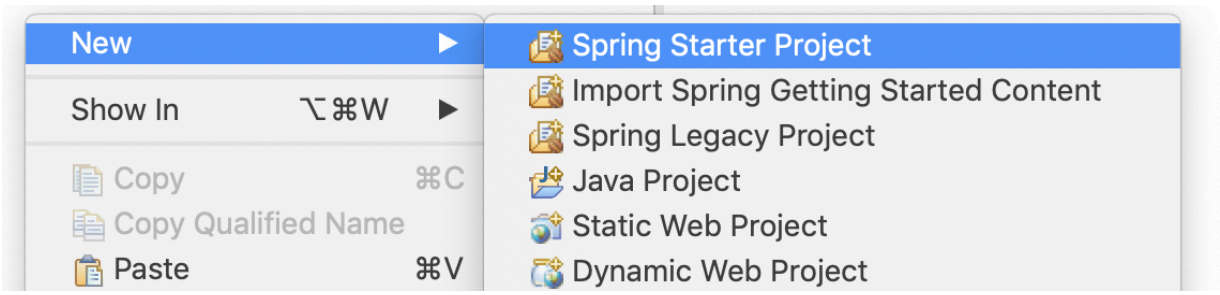
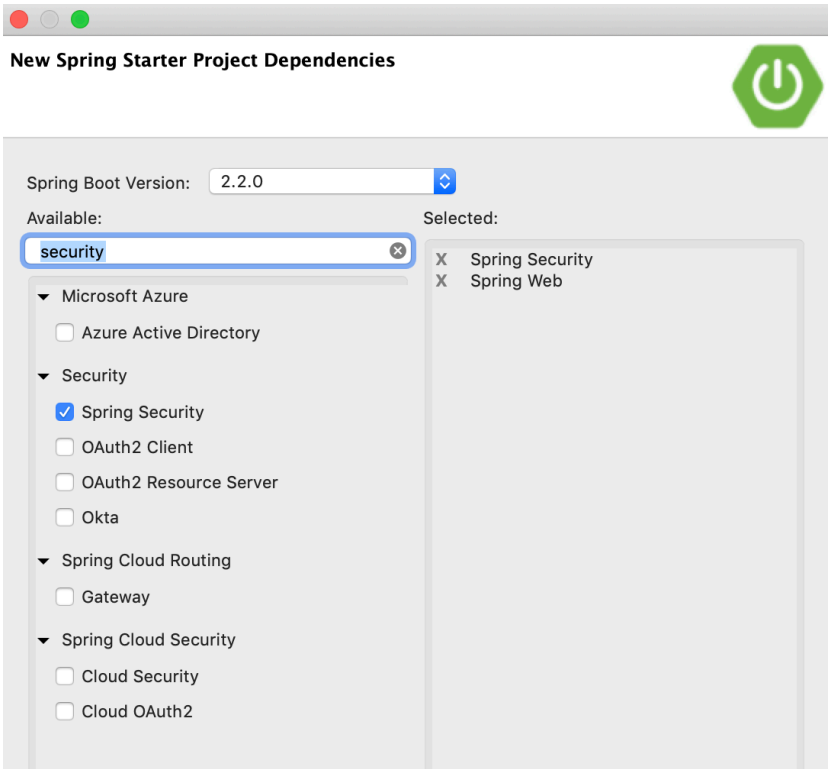
Create a Spring Boot Application to Authenticate using OAuth via Google Provider

1. Create a Spring Boot application using Spring Boot Starter



2. Add **Web & Security** modules



3. Add the following dependencies in **pom.xml** for **OAuth2** integration

```
<dependency>
     <groupId>org.springframework.security</groupId>
     <artifactId>spring-security-oauth2-client</artifactId>
</dependency>
<dependency>
     <groupId>org.springframework.security</groupId>
     <artifactId>spring-security-oauth2-jose</artifactId>
</dependency>

<!-- Other DEPENDENCIES -->
<dependency>
     <groupId>org.apache.tomcat.embed</groupId>
     <artifactId>tomcat-embed-jasper</artifactId>
</dependency>
```

4. Add the following properties in **application.properties** and observe the ***MAGIC*** when you RUN the APP

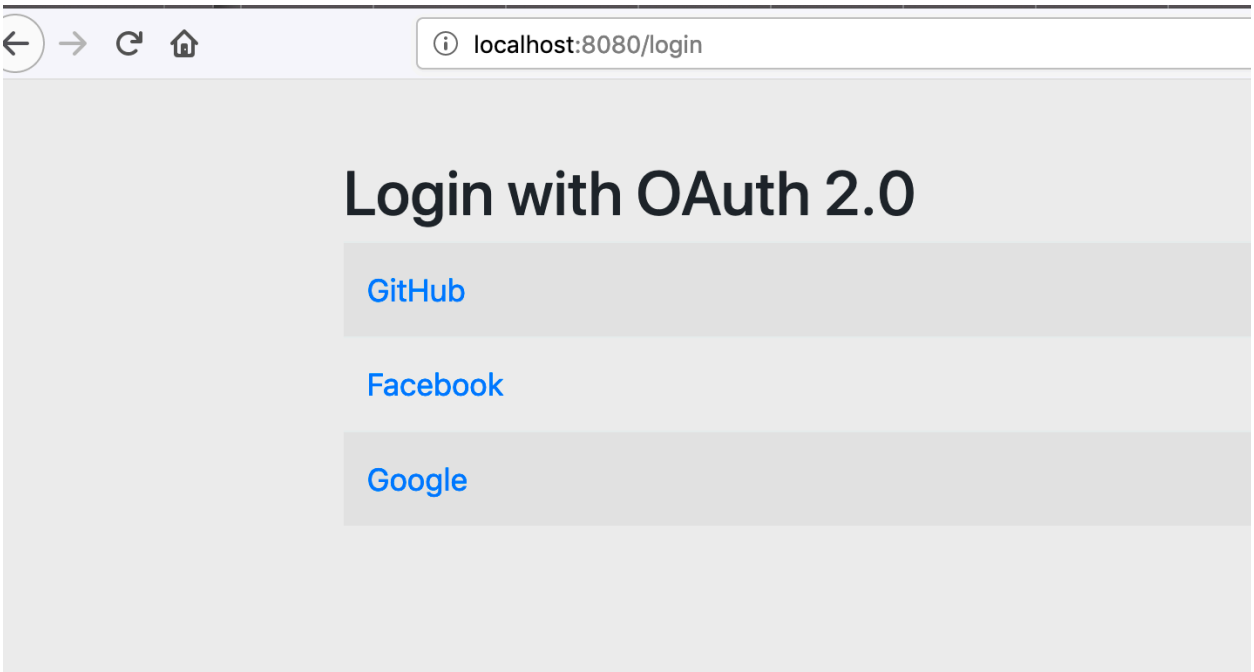
```
#Google app details
spring.security.oauth2.client.registration.google.client-
id=<your client id>
spring.security.oauth2.client.registration.google.client-
secret=<your client secret>

#Facebook app details
spring.security.oauth2.client.registration.facebook.client-
id=<your client id>
spring.security.oauth2.client.registration.facebook.client-
secret=<your client secret>

#Github app details
spring.security.oauth2.client.registration.github.client-
id=<your client id>
spring.security.oauth2.client.registration.github.client-
secret=<your client secret>

#Spring MVC details
spring.mvc.view.prefix: /WEB-INF/views/
spring.mvc.view.suffix: .jsp
spring.mvc.static-path-pattern=/resources/**
```

5. RUN the APP

SpringBootOauthGoogleDemoApplication.java
—> Run As
—> Spring Boot App

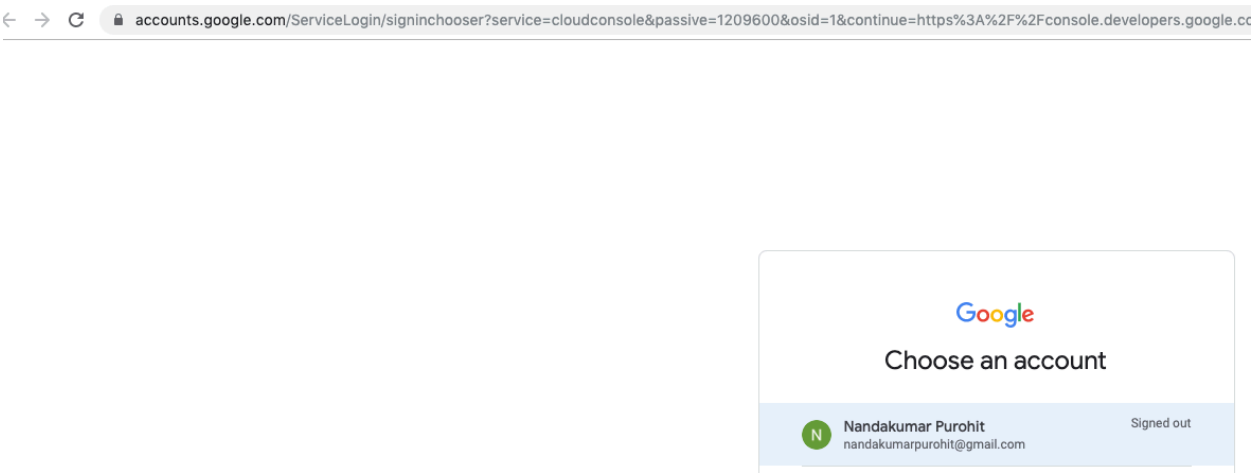
Here, we declare **two properties** for each provider. We will be implementing the **Google provider**, but you can add any number of providers.

Just adding these properties will create more magic, and your login page will suddenly change to the following:

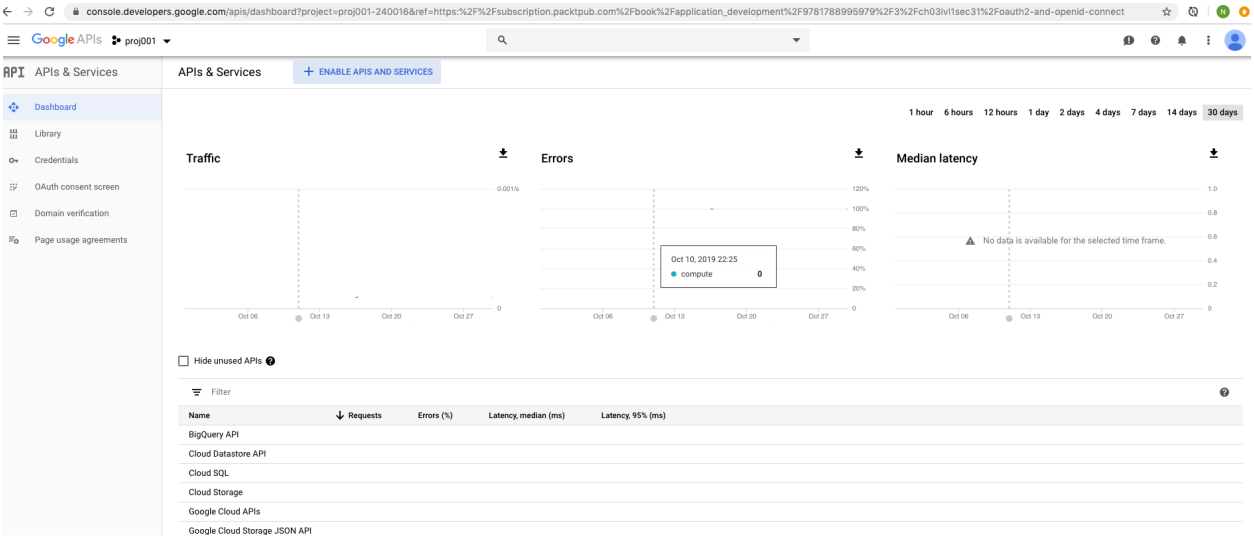


Google Provider setup

6. We will be using Google as our provider in this example. Navigate to https://console.developers.google.com/ and perform the following steps:

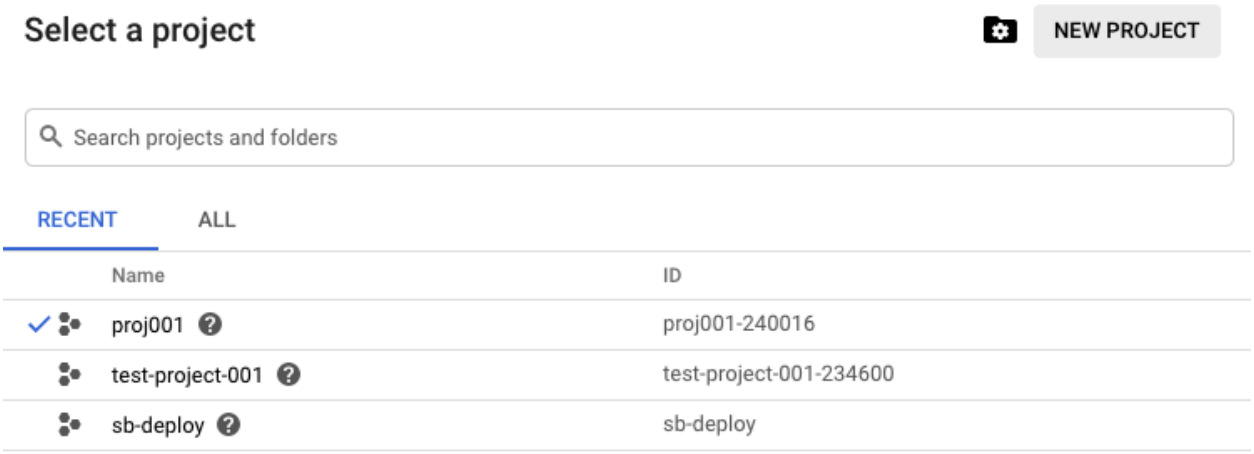


7. Login with your google credentials and you will see the following google console page

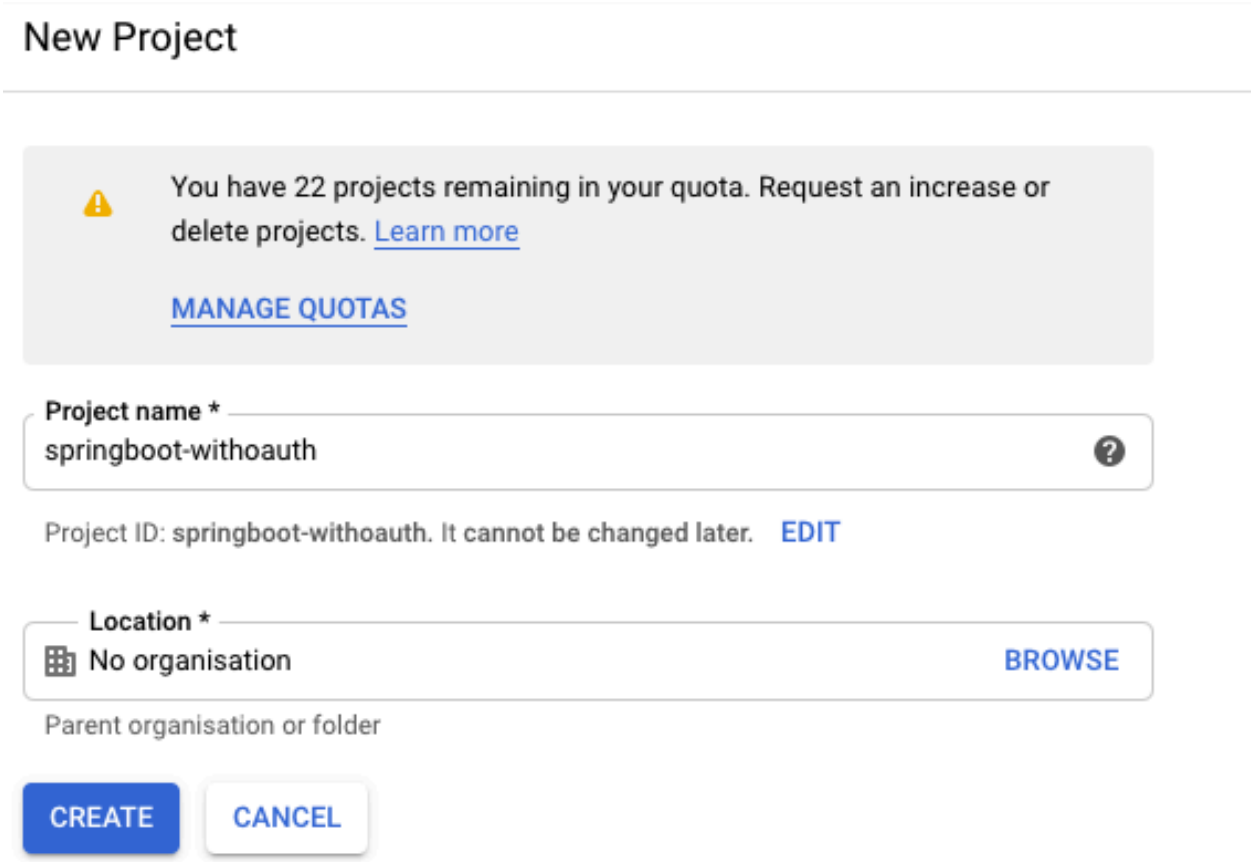


8. Click on Project Name next to Google APIs icon

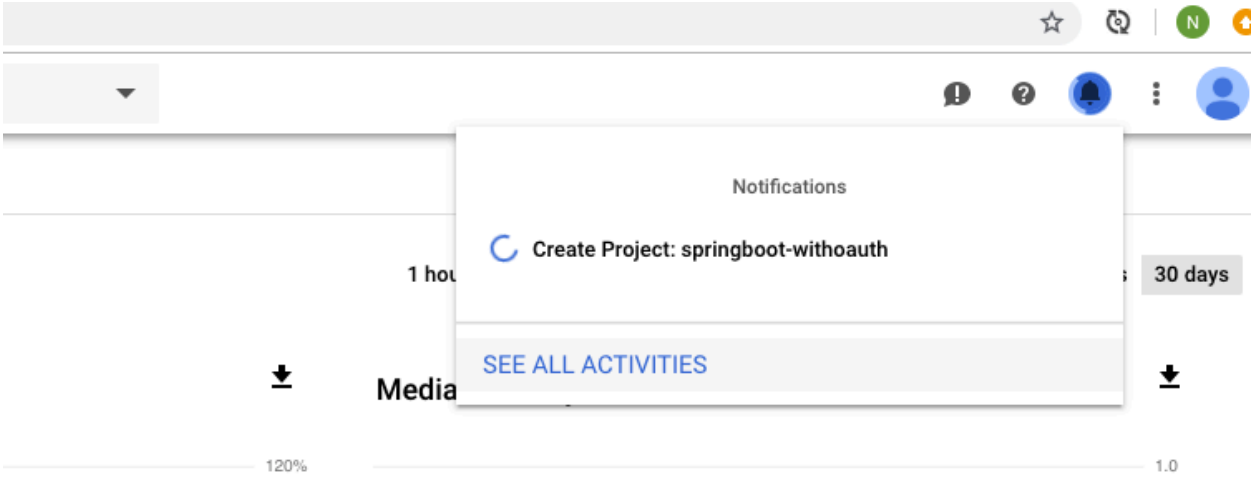
9. Click on **NEW PROJECT** button



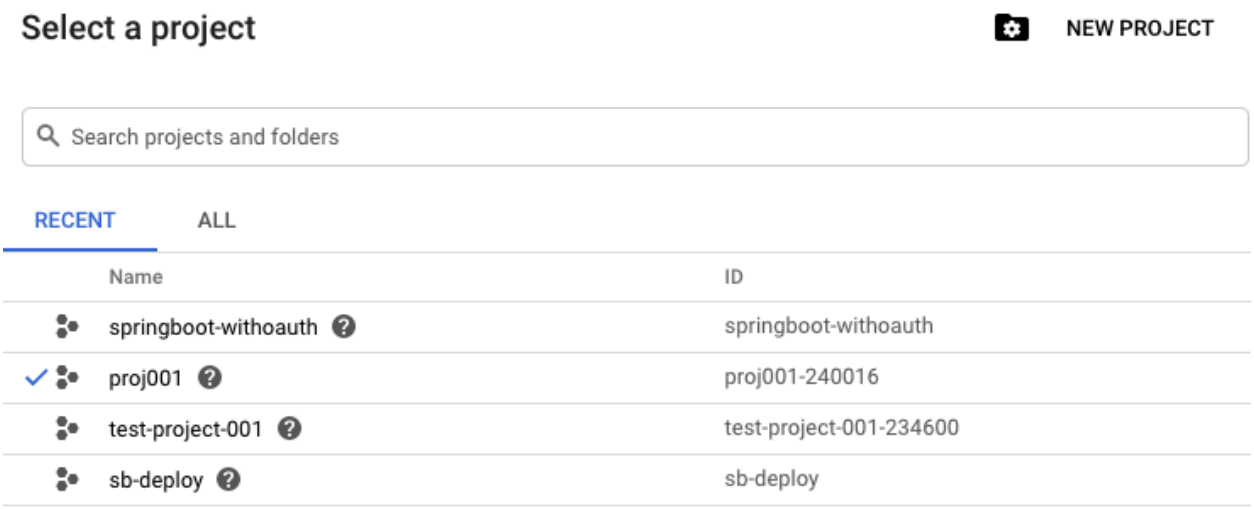
10. Enter the Project Name and click on **CREATE** button



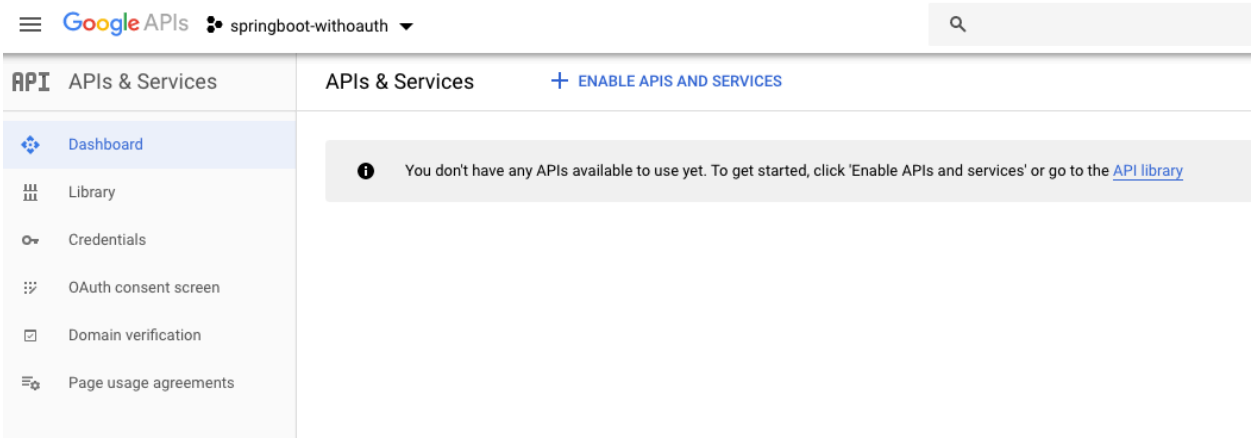
Project creation in progress



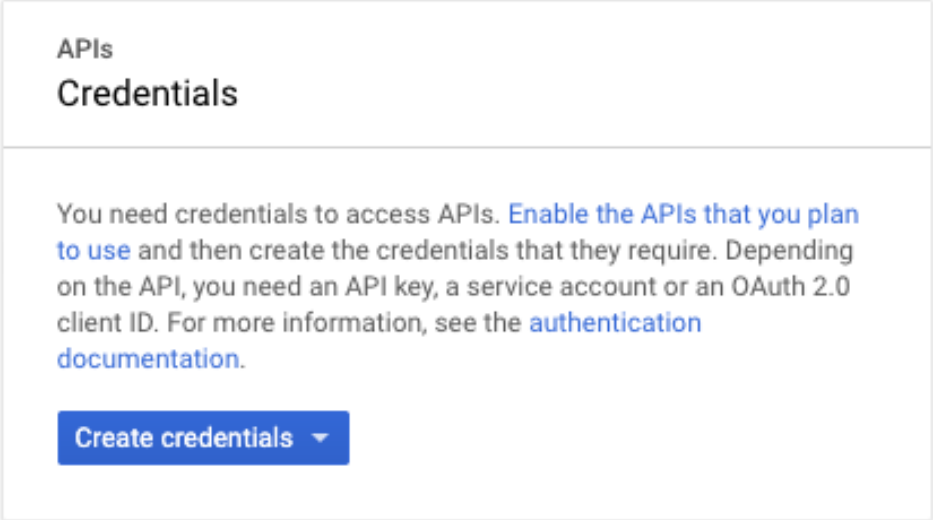
11. Click on the Project Name next to Google API link and select the newly created Project **springboot-withoauth**:

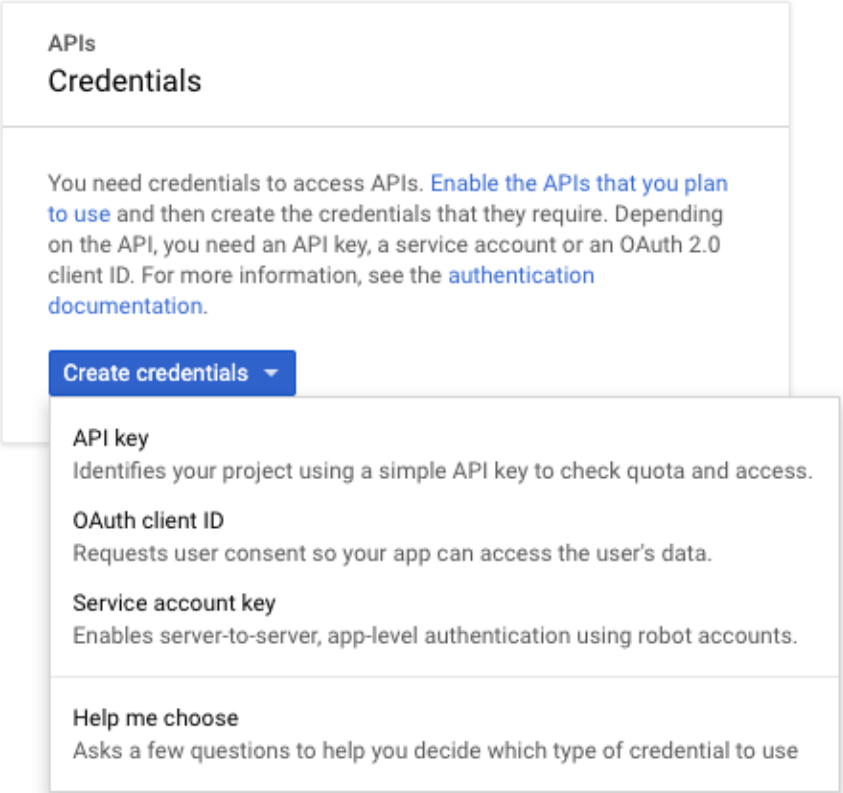


12. You will see this default screen for the new project



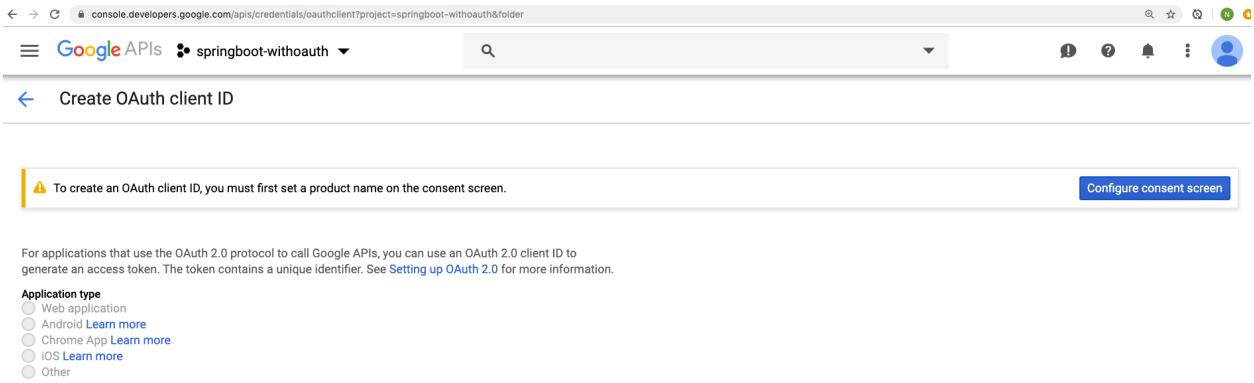
13. Click on the **Credentials** link on the side menu



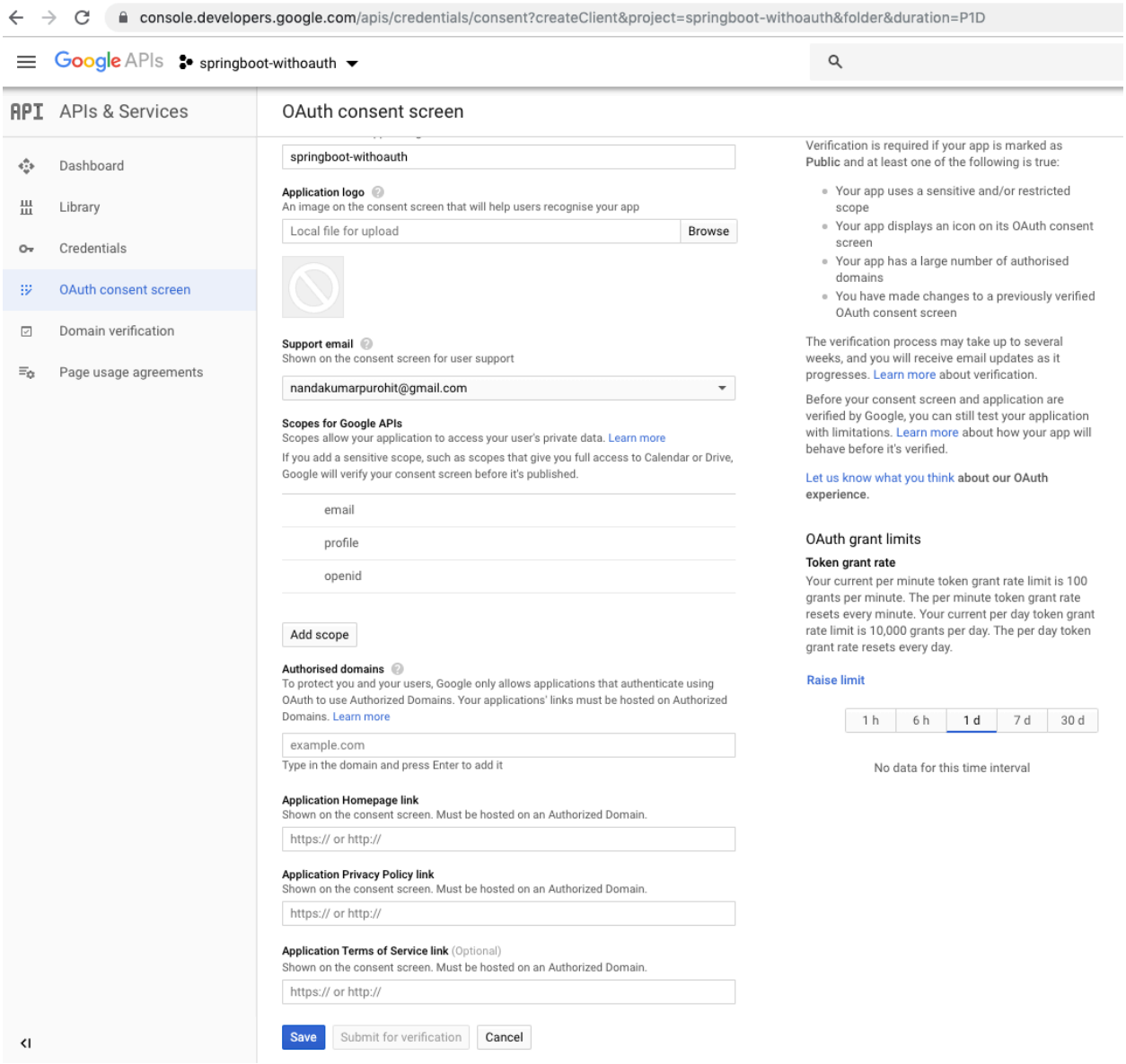


14. From the drop-down menu, click on **OAuth client ID**. This will navigate you to the page shown in the following screenshot. Please note that the **Application type** radio group will be disabled at this stage:

Click on the **Configure Consent Screen** button on the following screen



15. Enter the relevant details (***leave the optional fields out while filling in the form***), as shown in the preceding figure, and click on the **Save** button.



16. Enter the relevant details (leave the optional fields out while filling in the form), as shown in the preceding figure, and click on the **Save** button.

You will be navigated back to the page shown in the following figure.

This time the **Application Type** is enabled.

Name = ***Spring Boot Google OAuth Client***

Authorised JavaScript origins = ***http://localhost:8080***

Authorised redirect URIs = ***http://localhost:8080/login/oauth/code/google***

17. Then, click on **Create** button

console.developers.google.com/apis/credentials/oauthclient?project=springboot-withoauth&folder

Google APIs springboot-withoauth

Create OAuth client ID

For applications that use the OAuth 2.0 protocol to call Google APIs, you can use an OAuth 2.0 client ID to generate an access token. The token contains a unique identifier. See Setting up OAuth 2.0 for more information.

Application type

- (●) Web application
- ( ) Android Learn more
- ( ) Chrome App Learn more
- ( ) iOS Learn more
- ( ) Other

Name

Spring Boot Google OAuth Client

Restrictions

Enter JavaScript origins, redirect URIs or both Learn more

Origins and redirect domains must be added to the list of authorised domains in the OAuth consent settings.

Authorised JavaScript origins

For use with requests from a browser. This is the origin URI of the client application. It cannot contain a wildcard (https://*.example.com) or a path (https://example.com/subdir). If you're using a non-standard port, you must include it in the origin URI.

http://localhost:8080

https://www.example.com

Type in the domain and press Enter to add it

Authorised redirect URIs

For use with requests from a web server. This is the path in your application that users are redirected to after they have authenticated with Google. The path will be appended with the authorisation code for access. Must have a protocol. Cannot contain URL fragments or relative paths. Cannot be a public IP address.

http://localhost:8080/login/oauth2/code/google

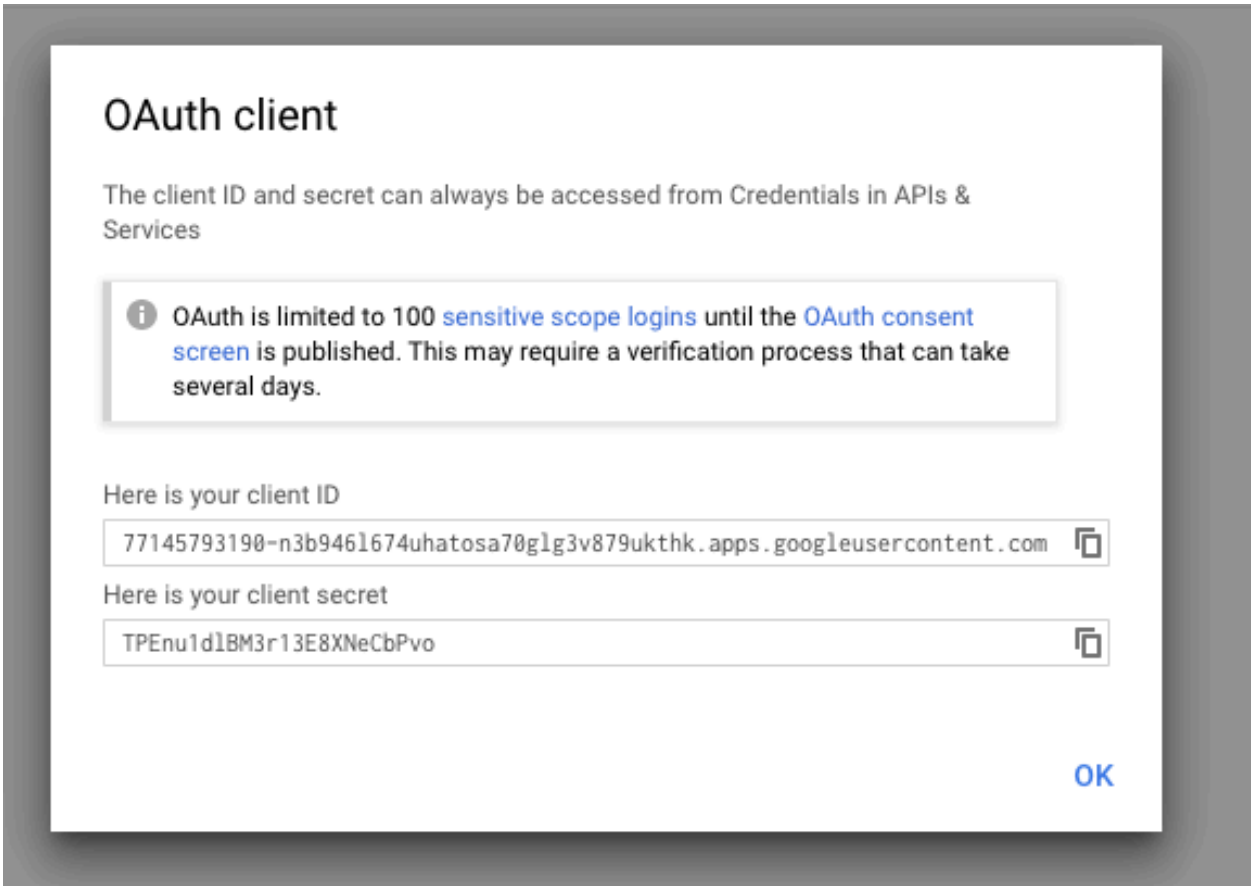
https://www.example.com

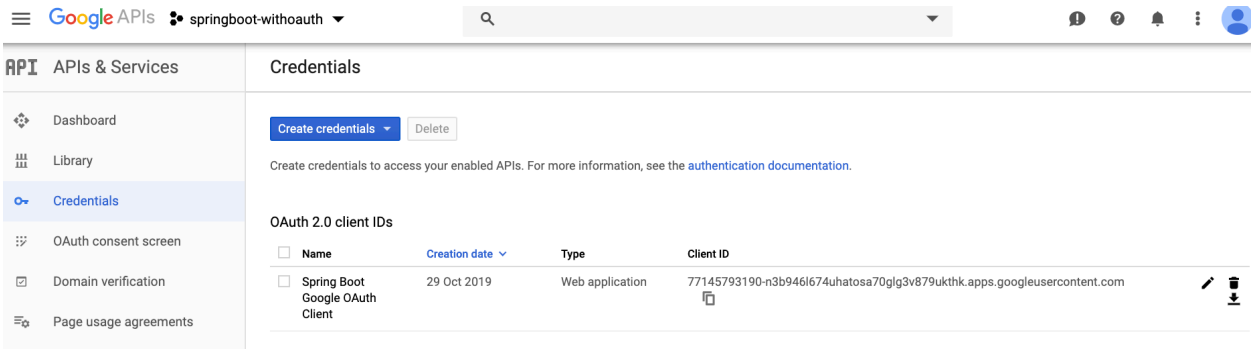
Type in the domain and press Enter to add it

Create Cancel

18. You will see a following screen with a confirmation on OAuth **Client ID** and **Client Secret** code generation.

Click on **OK**



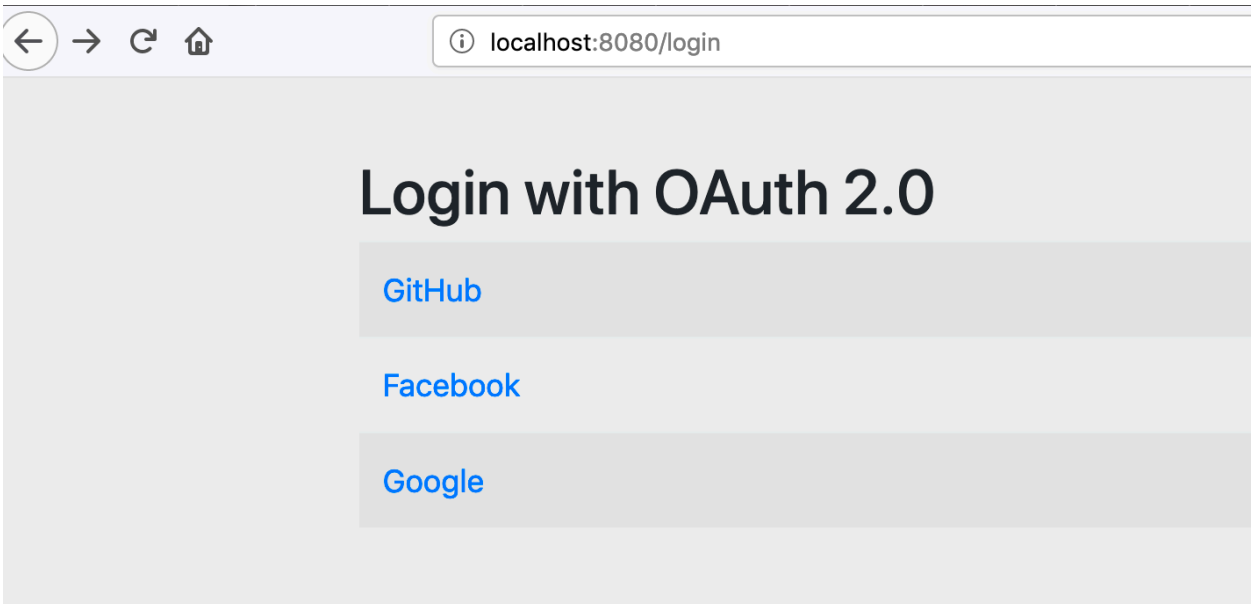


19 Copy and Paste these credentials in your **application.properties** for **Google** account

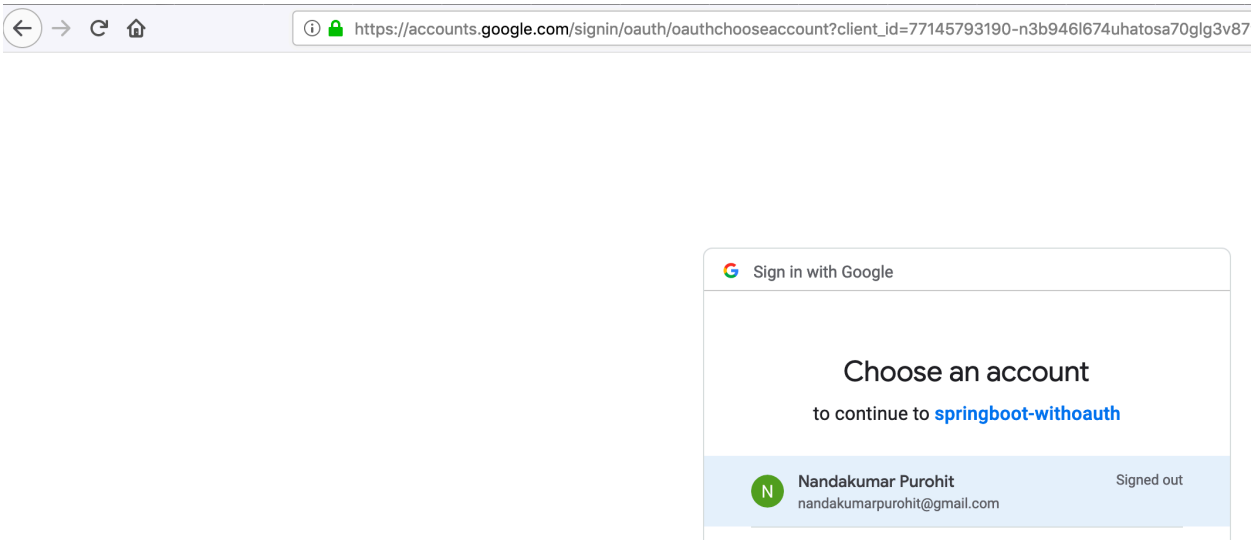
#Google app details

```
spring.security.oauth2.client.registration.google.client-id=<your-generated-client-id>
spring.security.oauth2.client.registration.google.client-secret=<your-generated-client-secret>
```

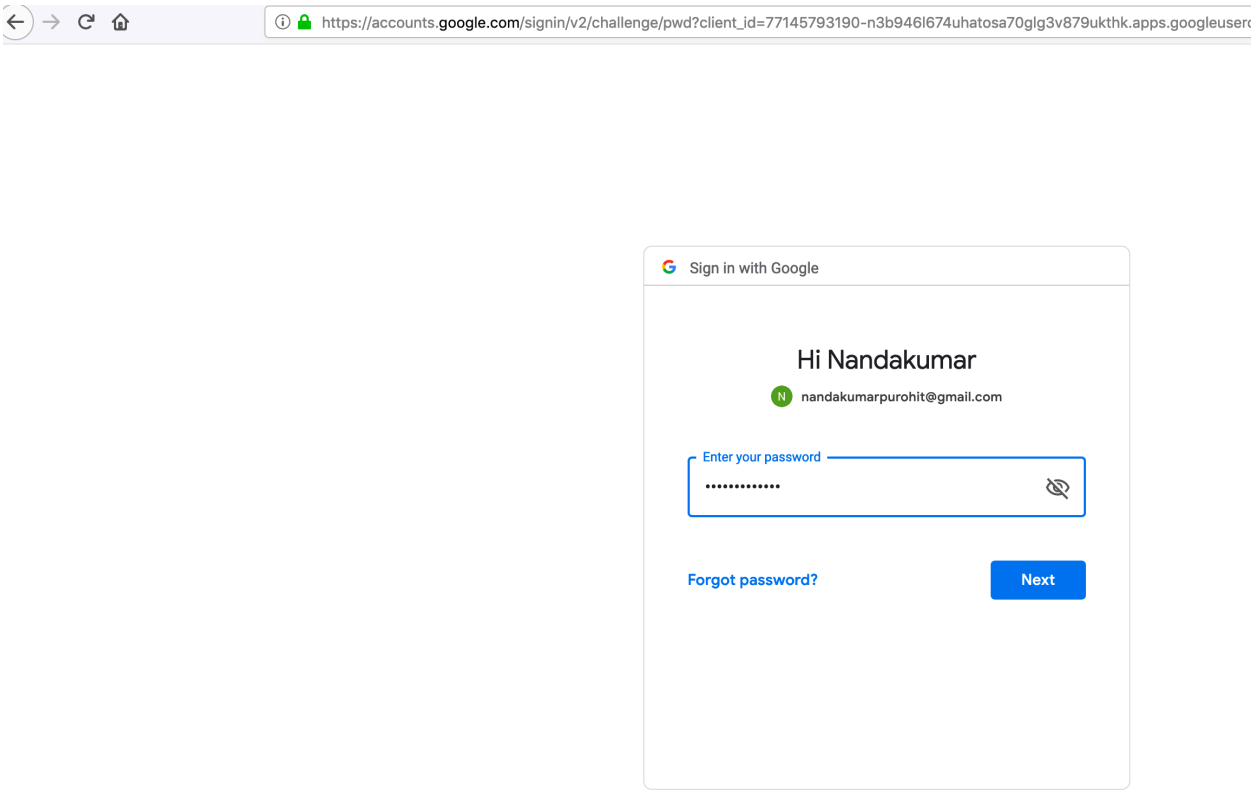
20. RUN the APP
Click on **Google** link



21. You will be redirected to Google login page



22. Login with your Google credentials



23. Observe that you get the following response