Project 3 Group Haramrit :- G01229319,Ajit G01210173

Project link:- <http://35.186.164.189/>

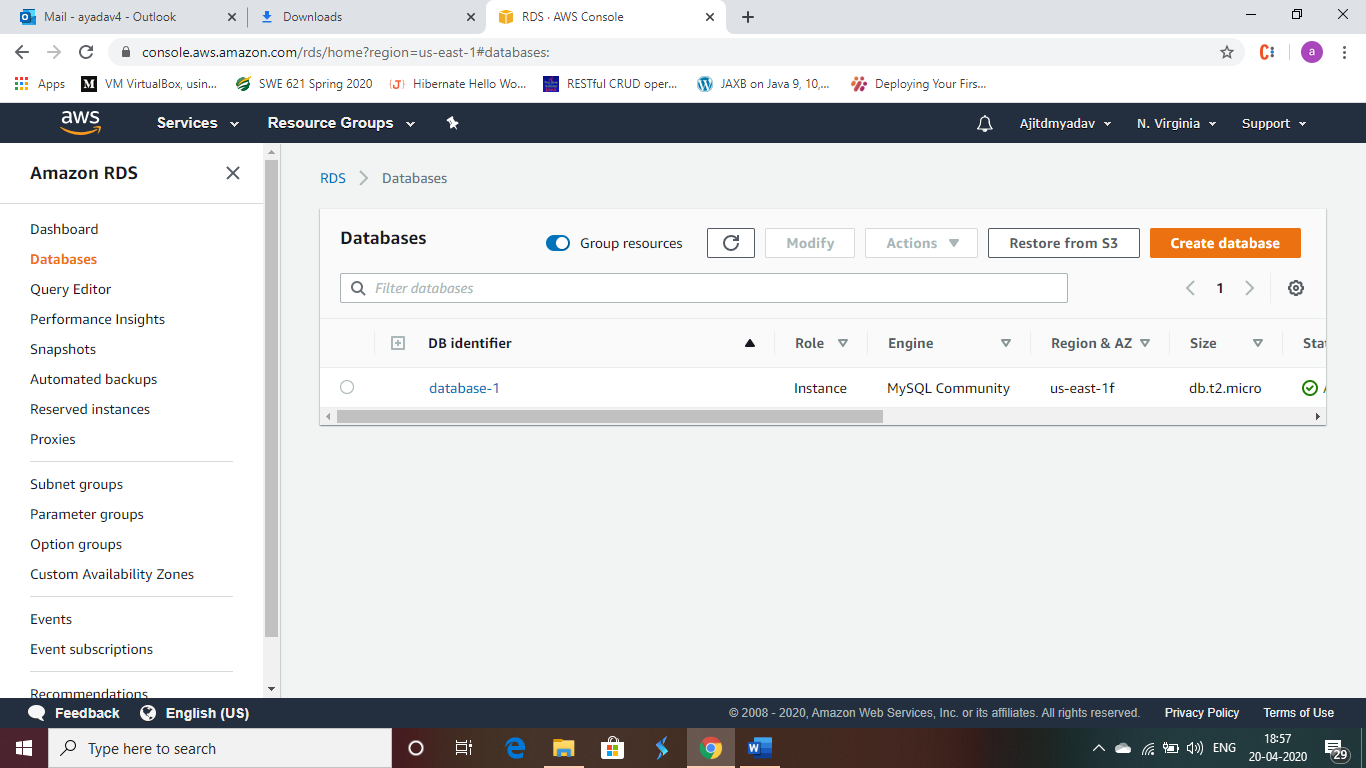
The Project Documentation can be divided in to the following parts:-

1. RDS connection with MySql database.
2. Rest implementation with jersey and JPa/Hibernate.
3. Angular Application.
4. Deploying everything on Kubernetes.

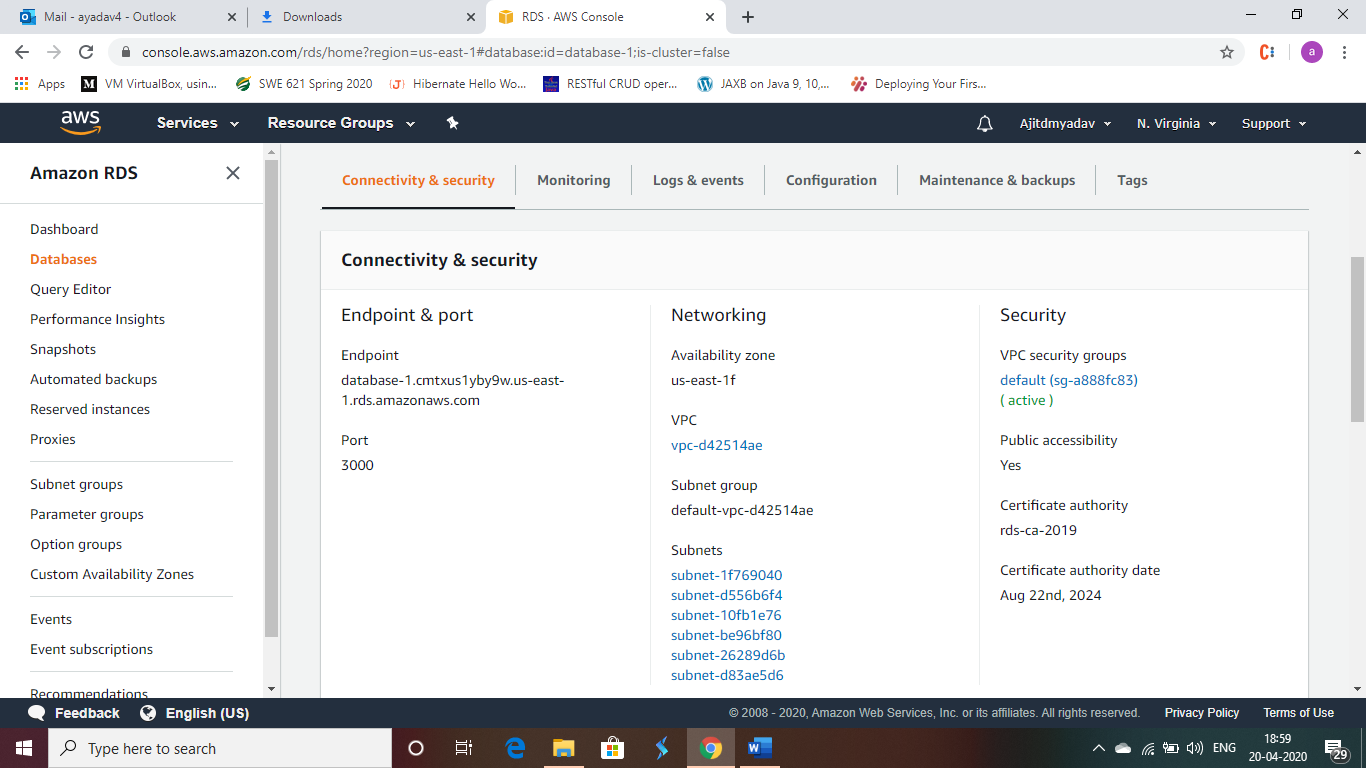
Part1:-

RDS connection with Msql Database:-

Step 1:- Click on create instance this will prompt you to create a Database instance In the picture below you can that “database-1” has been created.



Step2:-Once prompted select the Quick build database option which will ask for the instance type of the database to be created select the free tier one. After creating the instance note that allow access to the database from external sources otherwise it will throw an error.



Once that is done you can see the above image and copy the link of endpoint to access the database.

Step 3:- Accessing the Database:-

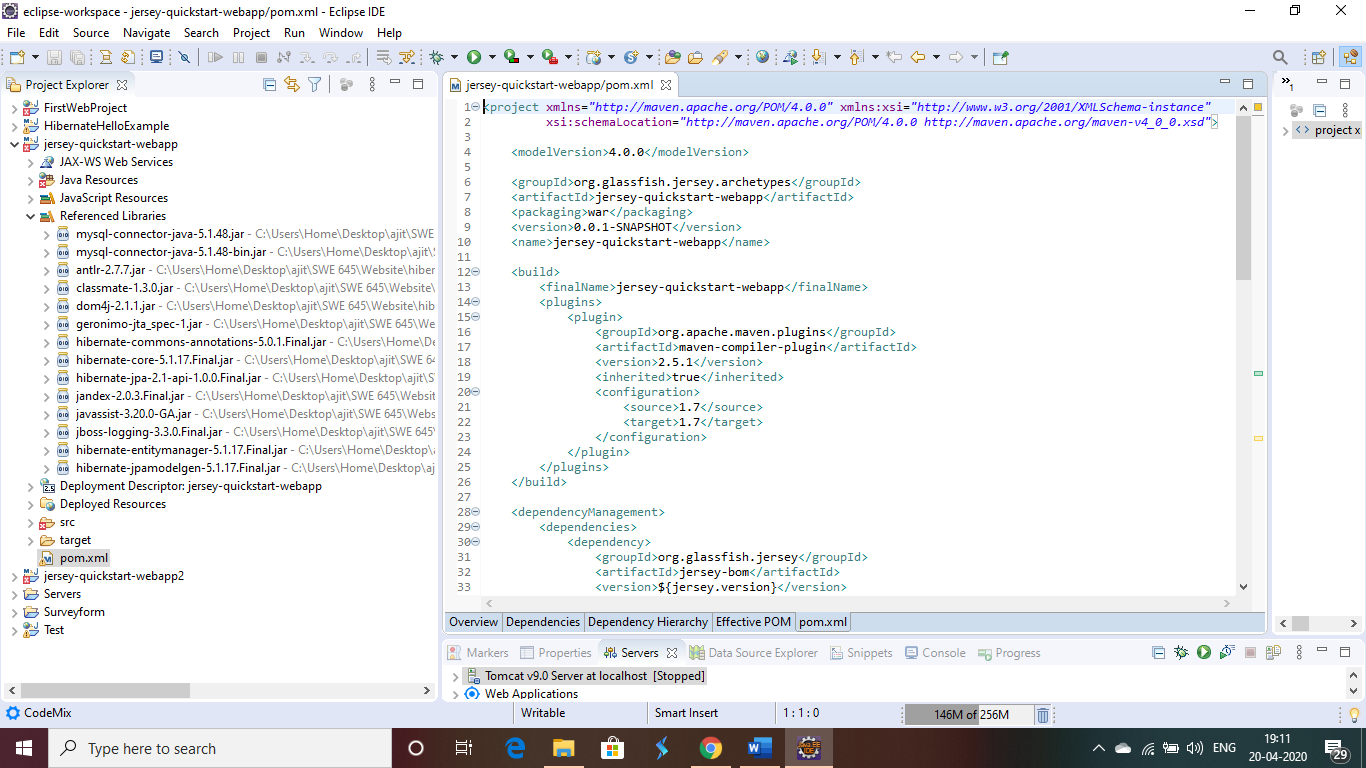
To access the database you will have to download a client workbench of the Mysql from oracle which will form the link between the my sql database and the cli where you can input the commands for the creation of the database

Then write create table <tablename>(); to create the table and done you now have the required database created and ready to be accessed from the rest server.

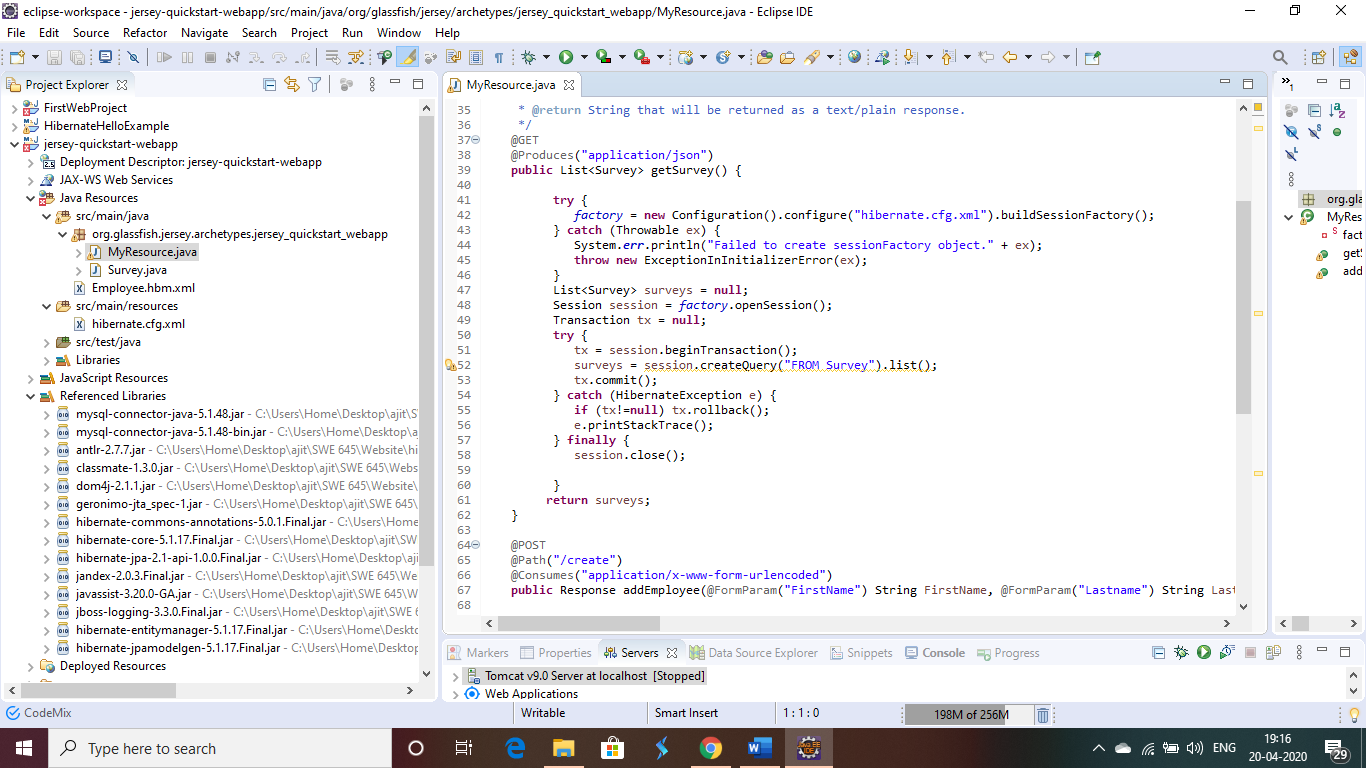
Part 2:- Rest implementation with jersey and JPA:-

To implement Rest with jersey go to the eclipse IDE:- click build a dynamic web project with glassfish jersey build this will build the default maven app with the pom file and jersey dependency,

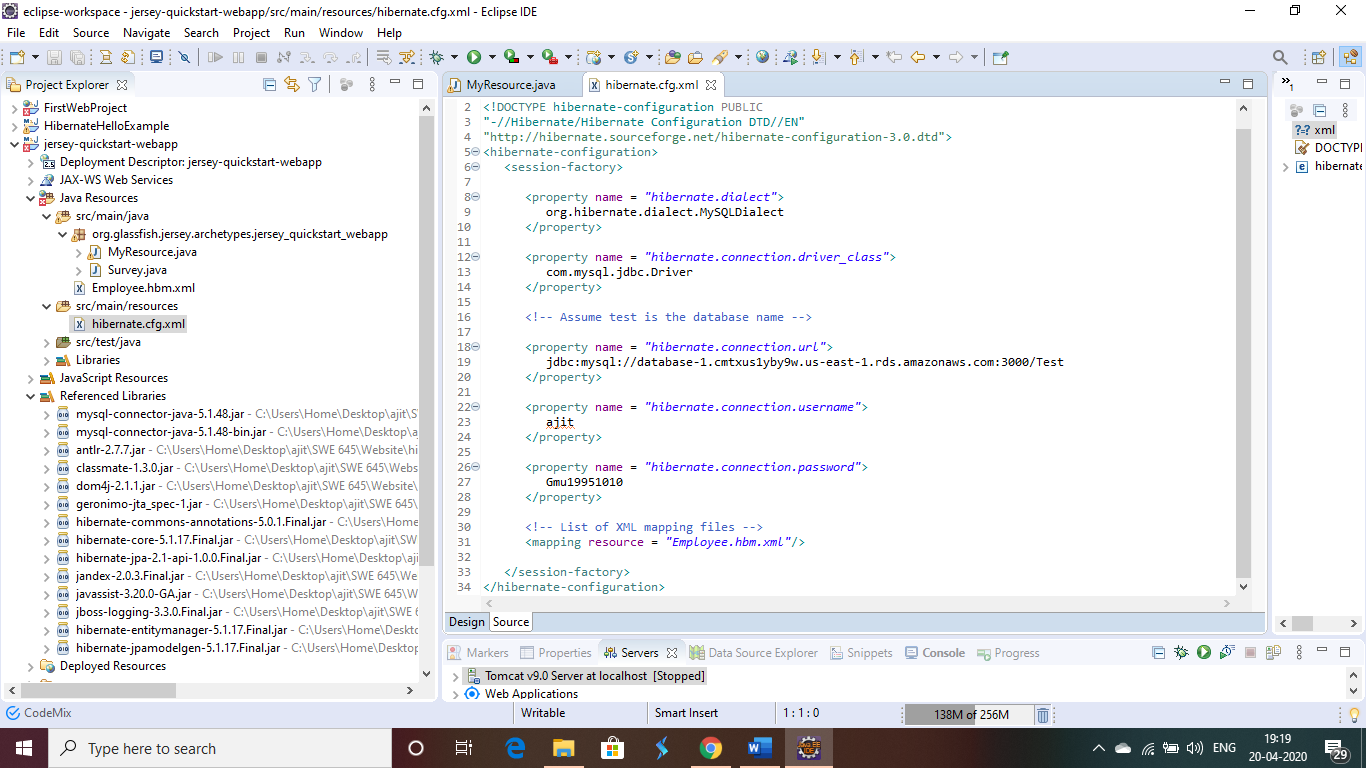
Step1:- download the hibernate and mysql jar files for the project as it is required for the connection to the database via JPA. As seen in the below picture we can see all the required dependency for the projects in the referenced library section. You would also be required to update the pom files with the below displayed dependencies.



Step2:- once done with the importing of the jar files we need to write the JAVA class for the connection with the databases and also the hibernate file for the connection as well as mapping file to map with the database .



In the above picture we can see that the REST get request will be mapped to the /my resource apth of the serve and the post request is mapped to the myresource/create path of the project once the request is called the function goes to the appropriate java method and creates a connection with the database using hibernate connection and mapping file.



In the above picture we can see that the hibernate file conists of the database connection url that was created during the RDS phase the user name is ait and the password can also be seen which is used to connect with the database instance.

Step 3:- Mapping class:-

We have the survey.java class which will map the java class to the database its just the declaration class with getter and the setter class that forms the connection with the database.

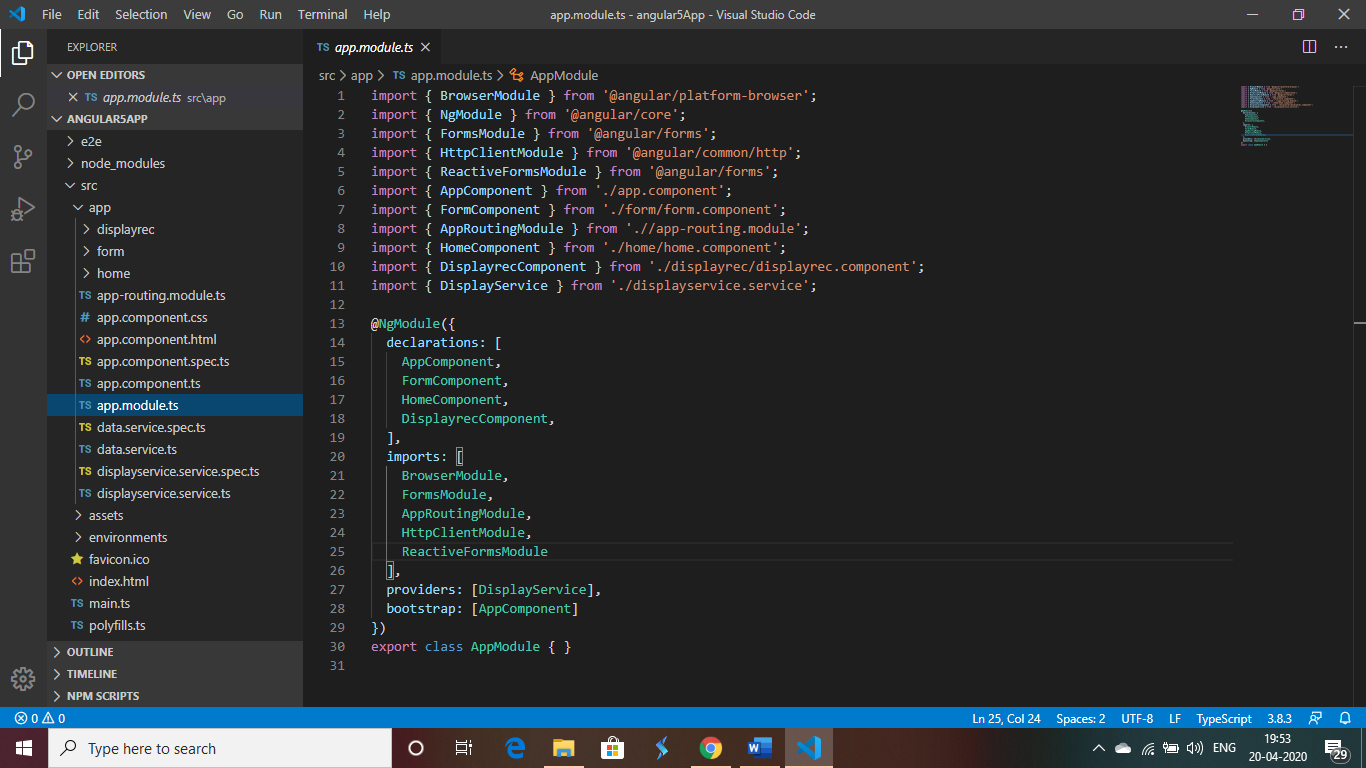
-----------------------------------------------WITH THIS OUR REST AND JPA IS COMPLETED-----------------------------------------------

Part 3:-Developing on the angular application:-

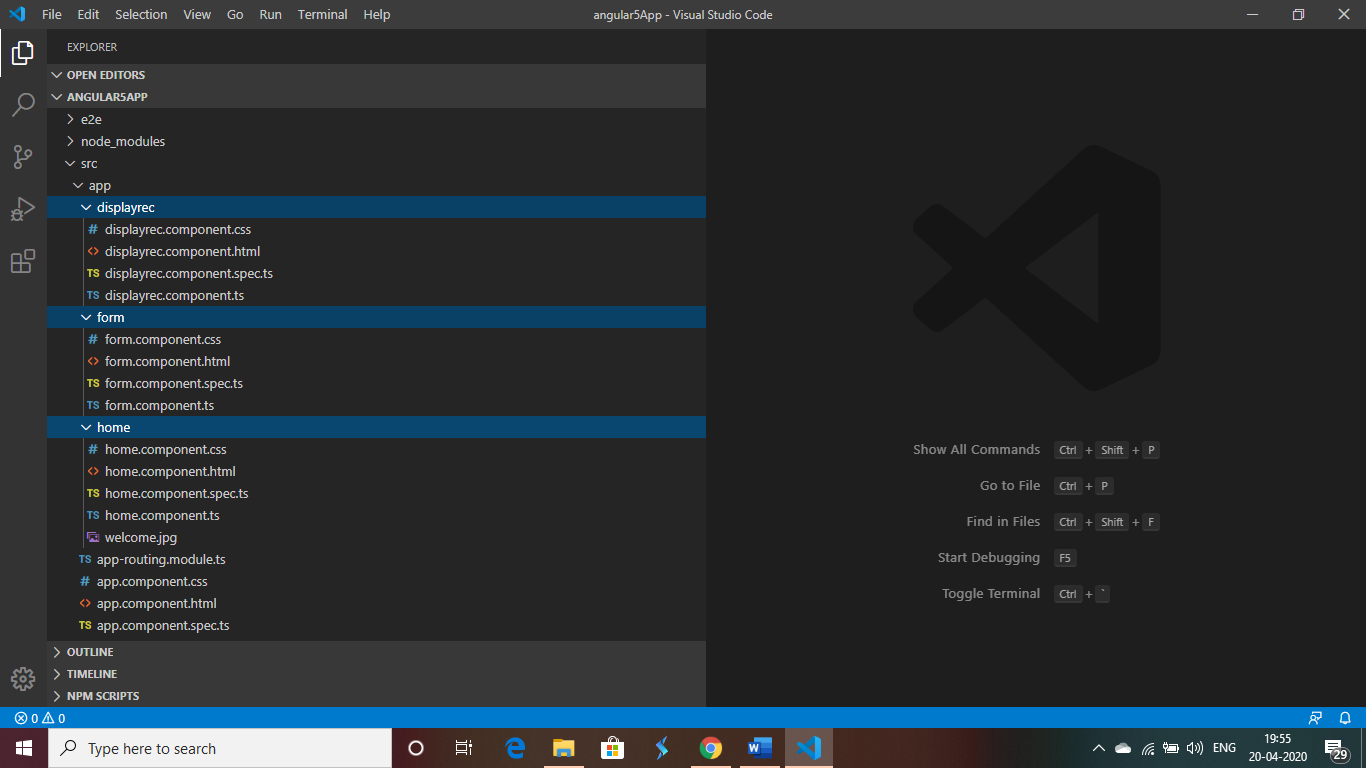
Pre requisite:-

Node.js already installed and Angular 5 and above used:-

Angular APP :- app.module is the first page to be loaded;

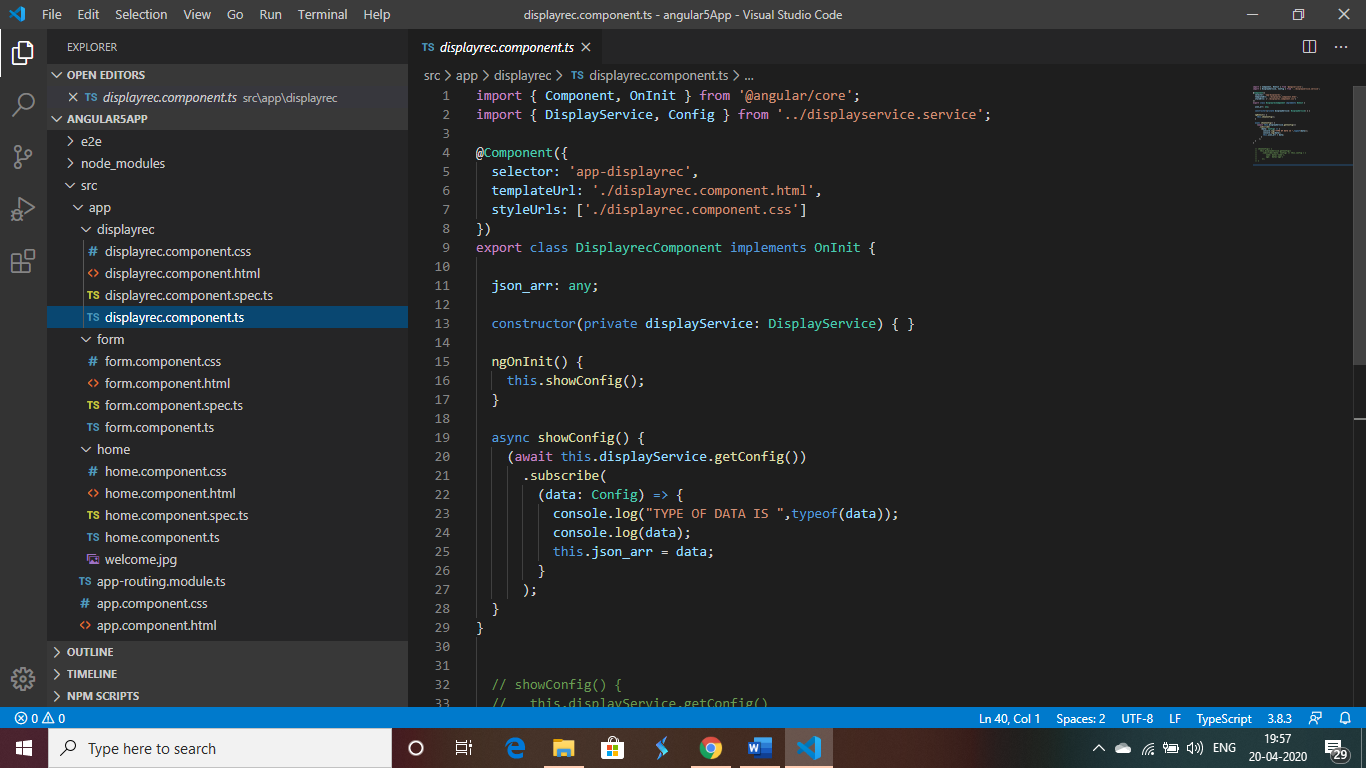


As we can see in the above image we need to import some of the library for the functionality of the applications like form module and Http module tomake a restful request.

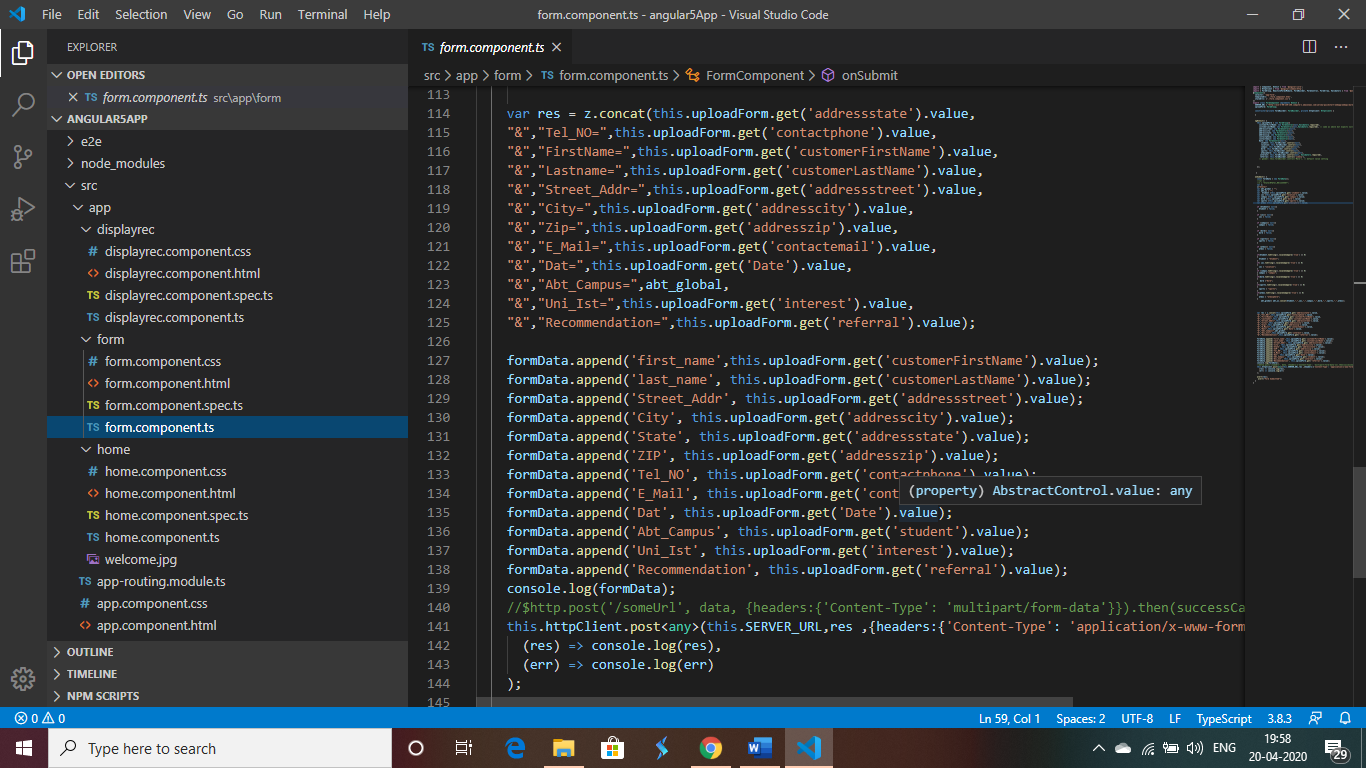


There are 3 module created for this application one for the Home page aka Welcome page and also one for the display records which will do a post request and display the records from the database th elast module is for the form page to input the survey from the user.

Display componenet.ts:- This file contains the logic for the get rest request it takes the dtain the json format and displays on the screen.

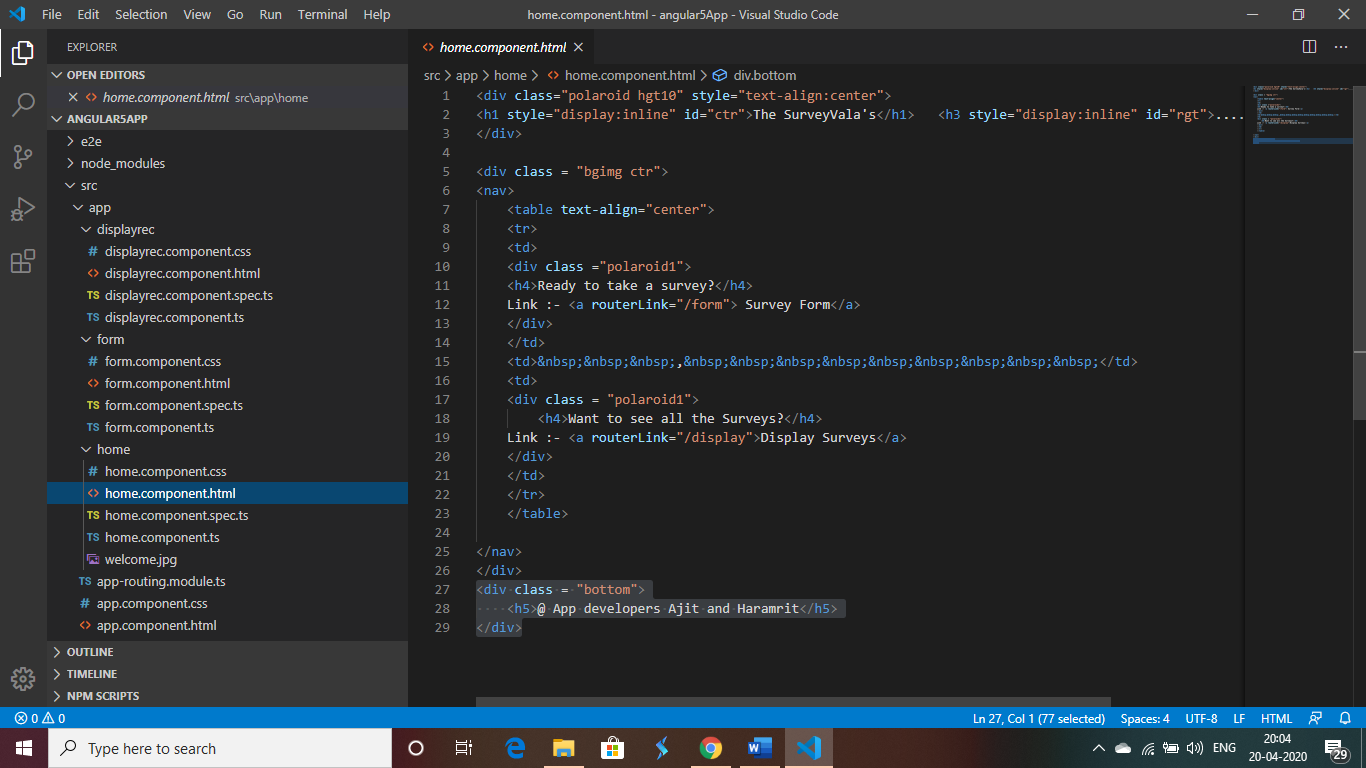


Form Component.ts:-



This file takes makes the post request to the server and it uses the post method of “XX-form data“ which is then captured by the rest server that we have developed on the previous part.

Home component.html:-



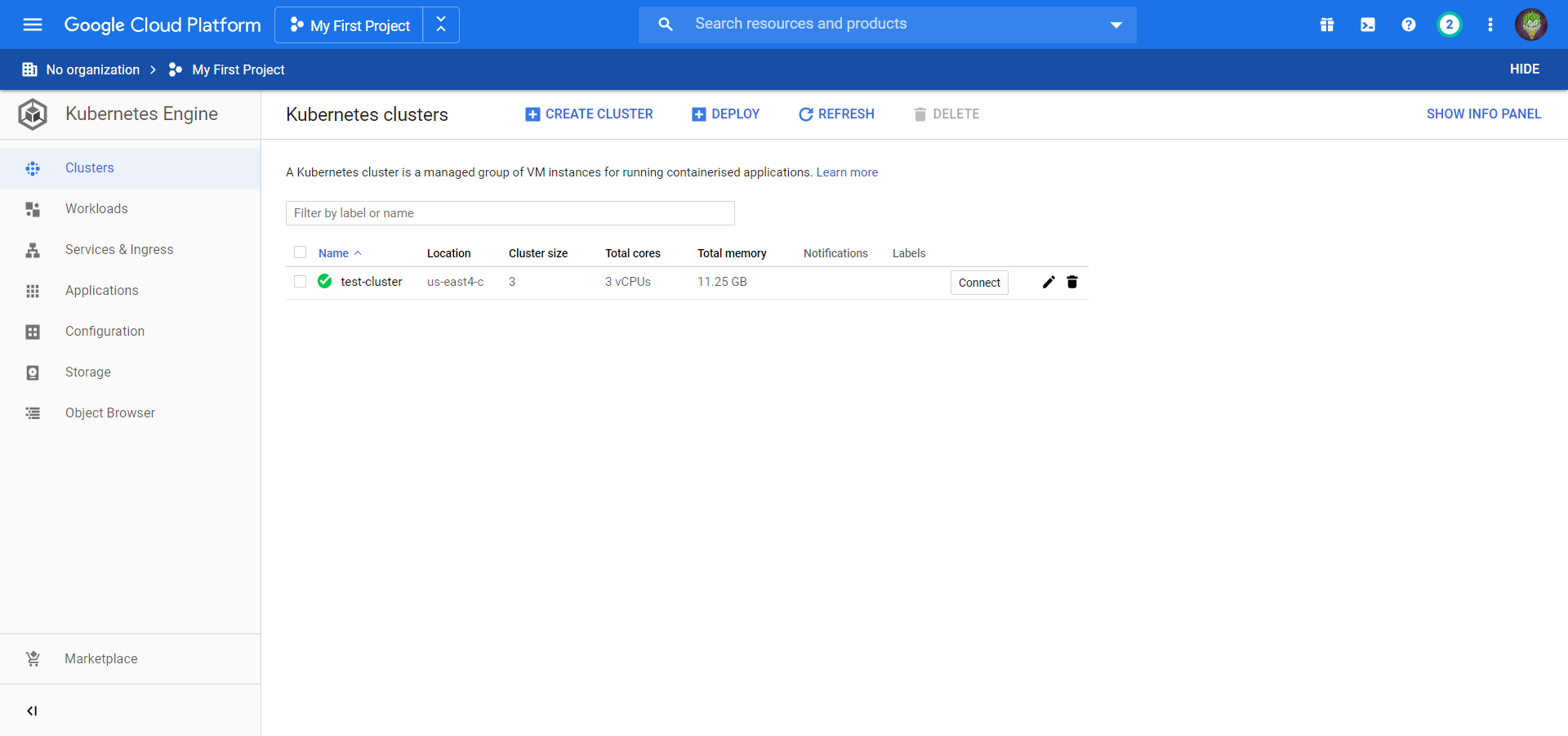
This file is the welcome page of the application.

-----------------------------------------------------------Angular application completed-------------------------------------------------------

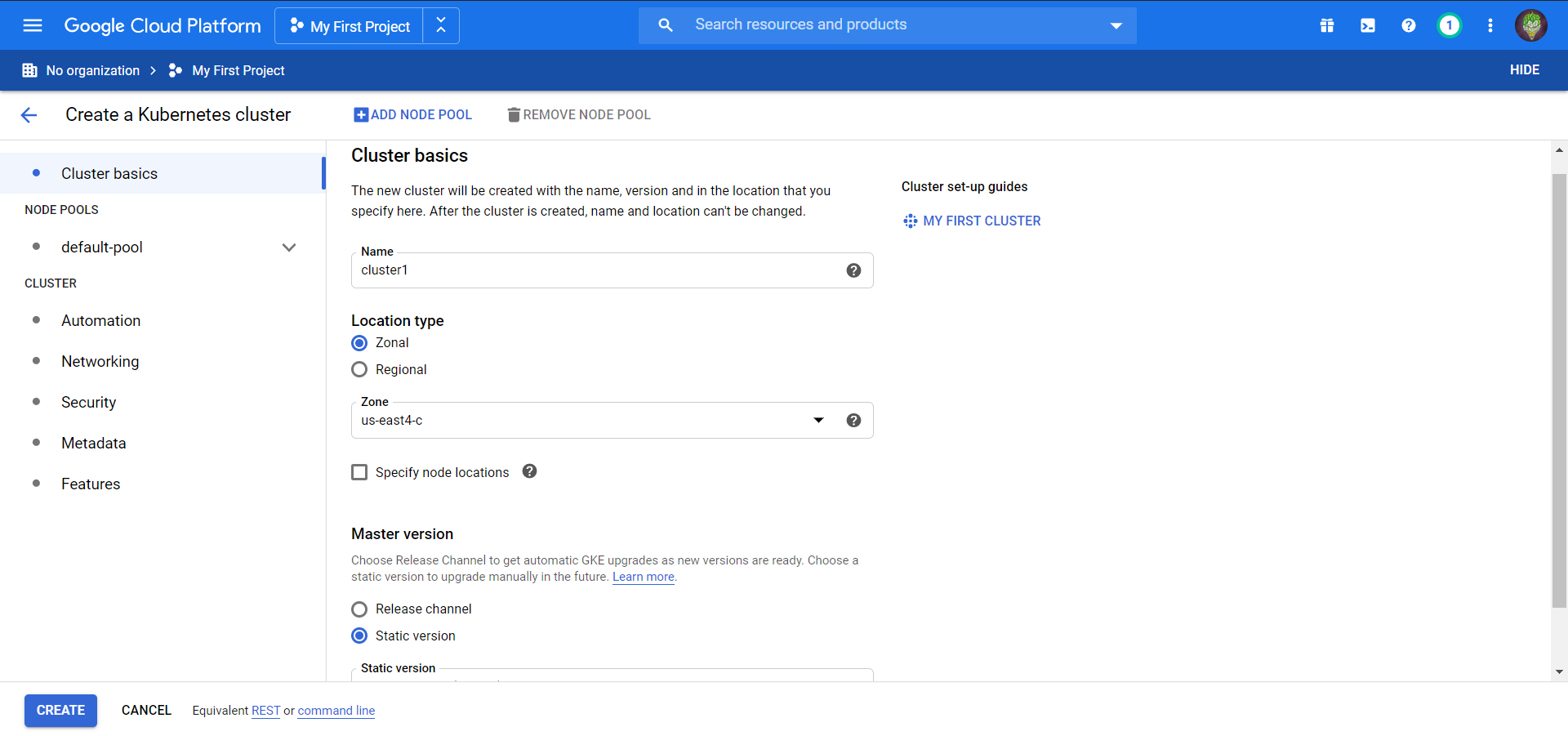
Part 4 :-

Putting up all together with kubernetes:-

1. Creating the Cluster

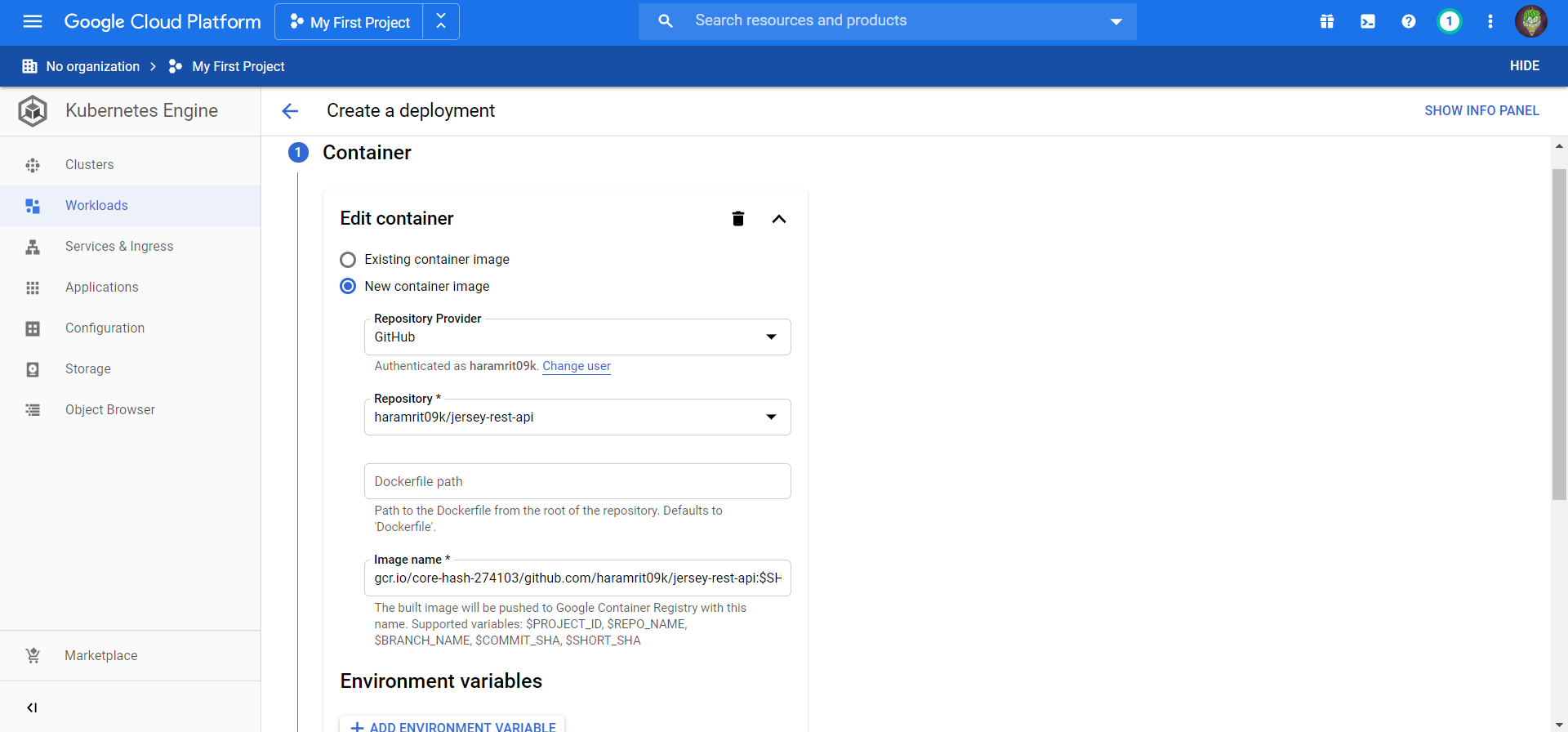


Clicking on “Create Cluster” takes you to a page where you just need to add cluster name, zone (us-east4c for Virginia/DC) and master version of GKE (whether you want to regularly update GKE to the latest version, or manually do it according to your needs)



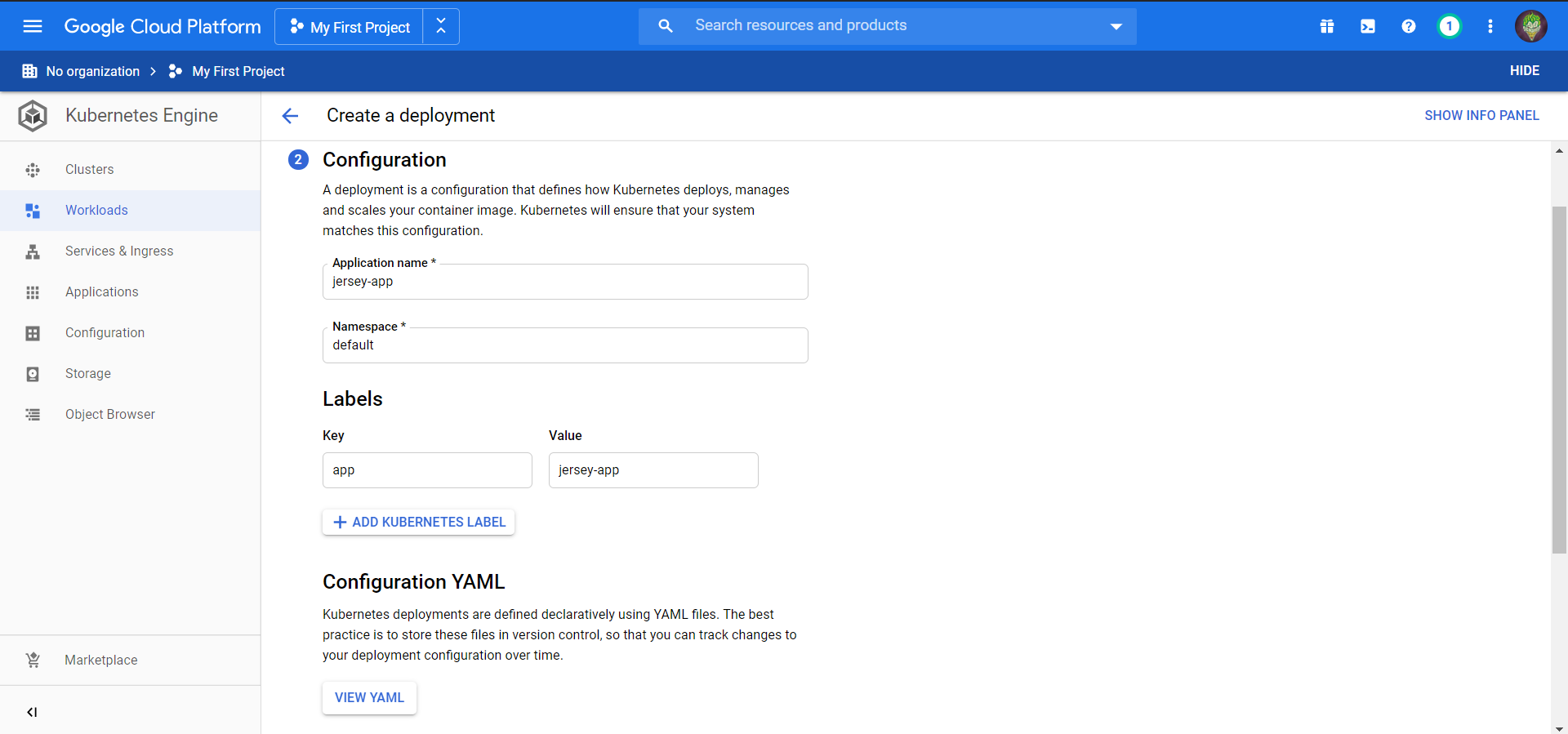
1. Deploying the app

After cluster is created, go to the cluster dashboard and click on “Deploy”.



Select “new container image” and link your github repo to GCP

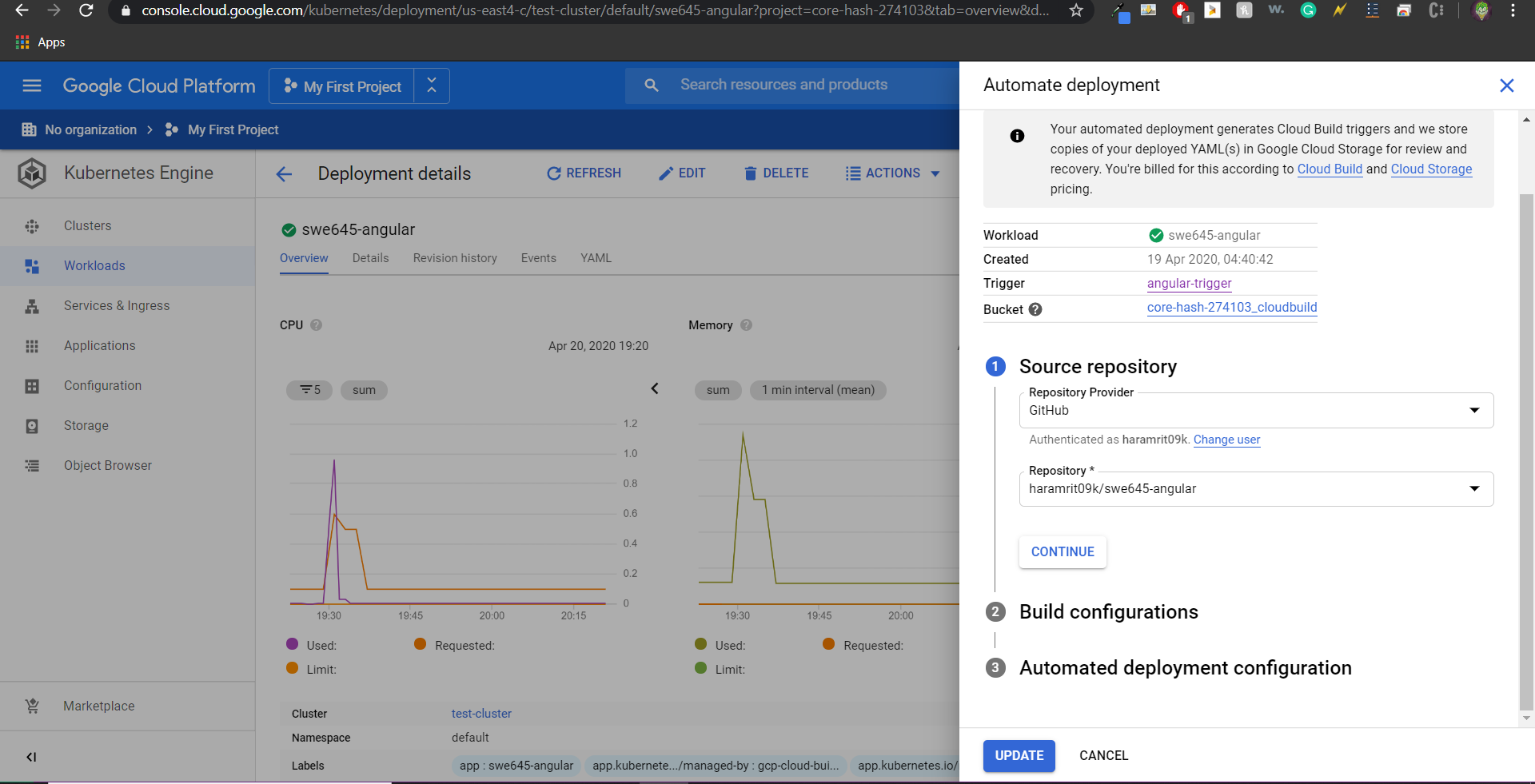
Click on Continue



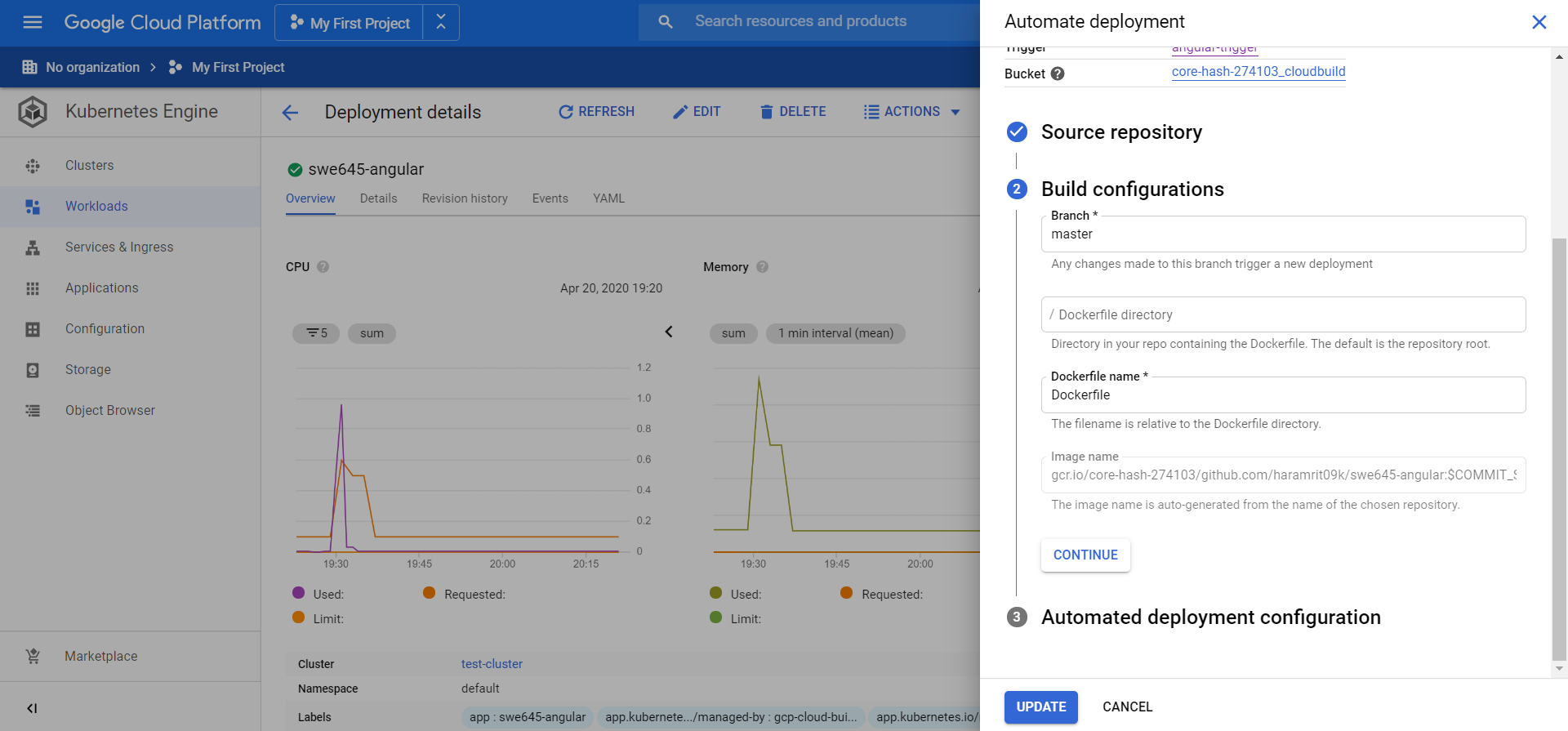
Give additional info such as application name, cluster on which to deploy, etc and click on “Deploy”

1. Automatic Deployment

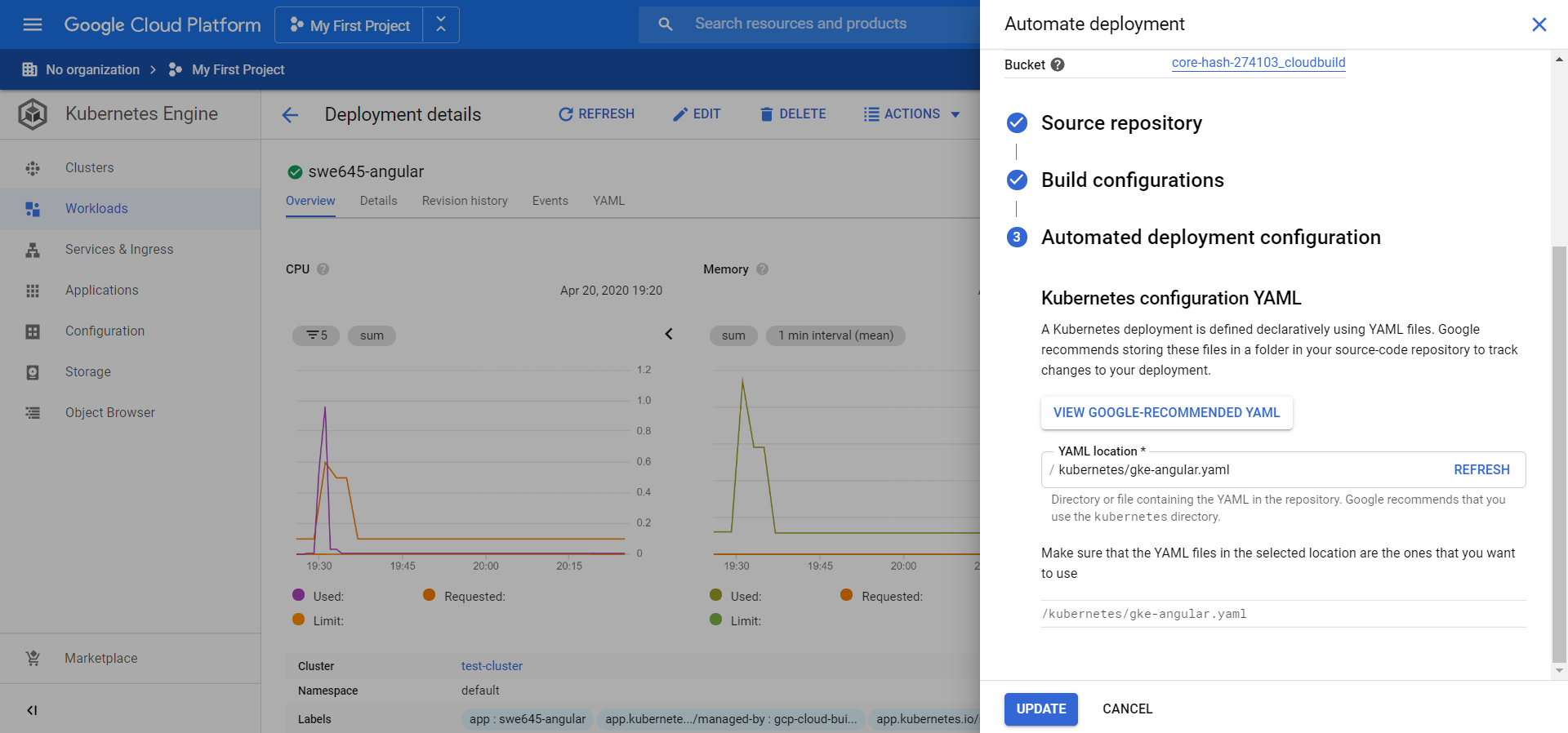
Go to Workloads, click on recently deployed app and from Actions -> select Automated deployment



Select repo where to automate deployment from



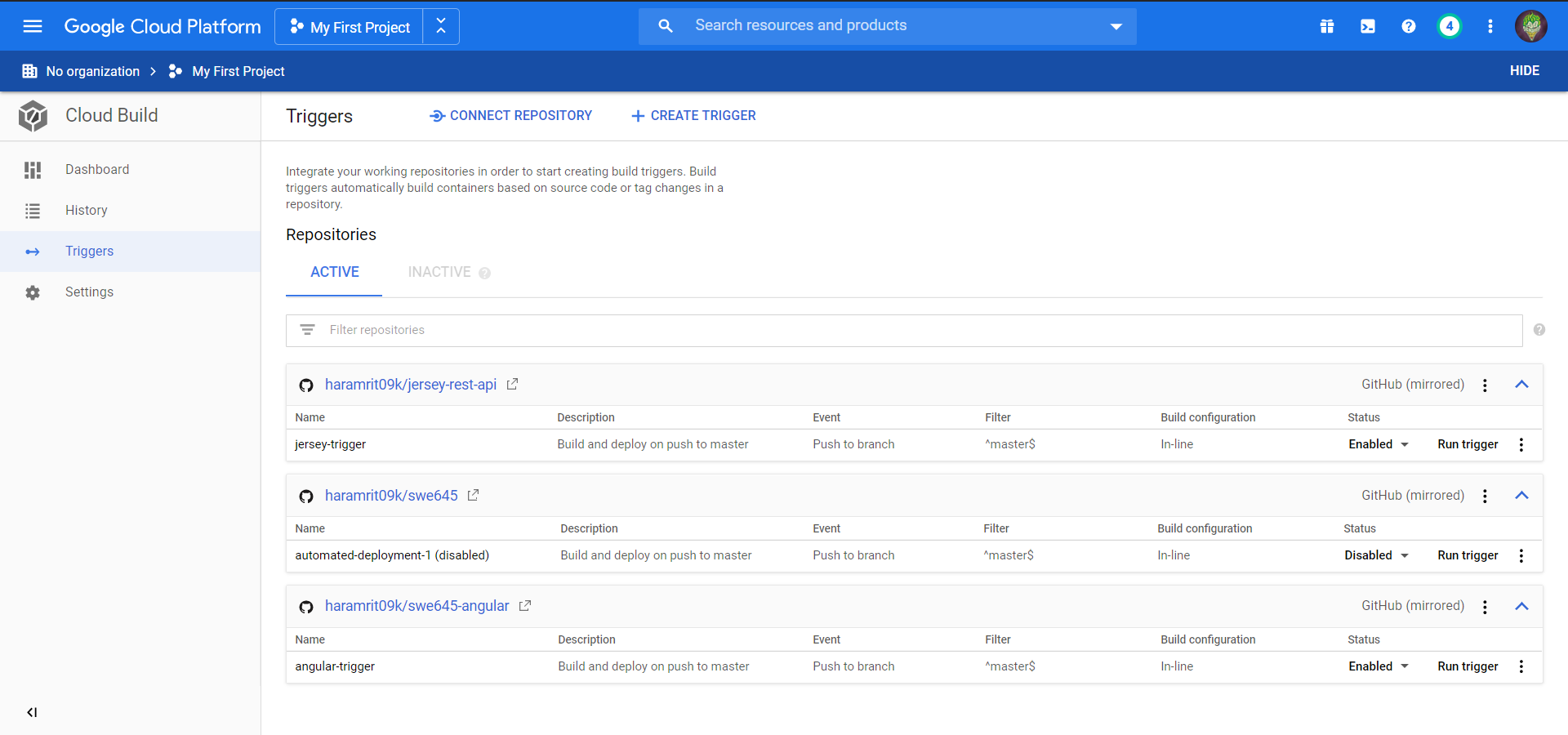
Add build configurations



Add deployment configuration -> copy “Google recommended YAML” to github repo within kubernetes directory

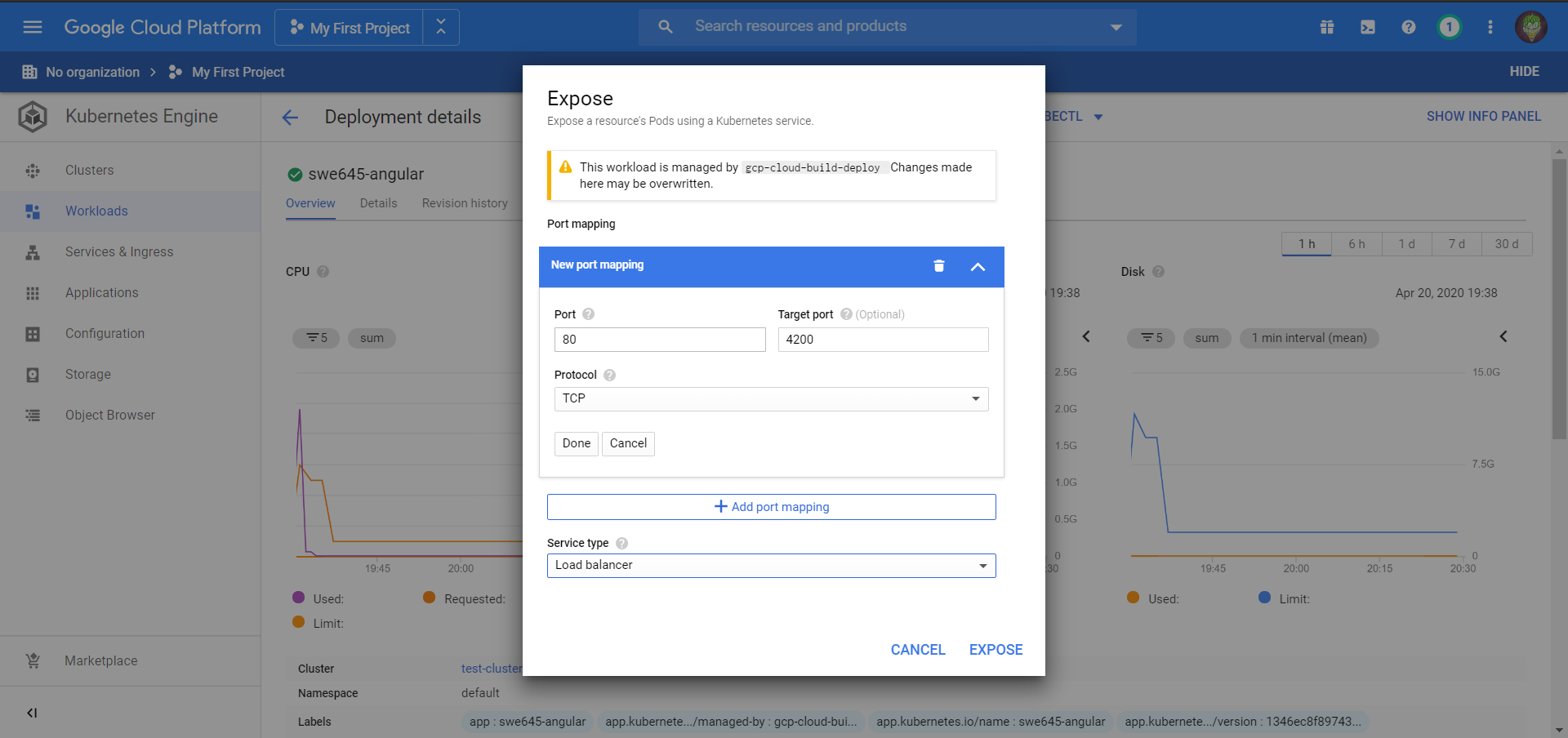
Click on “Update”

A build trigger should be created inside Cloud Build as shown below:



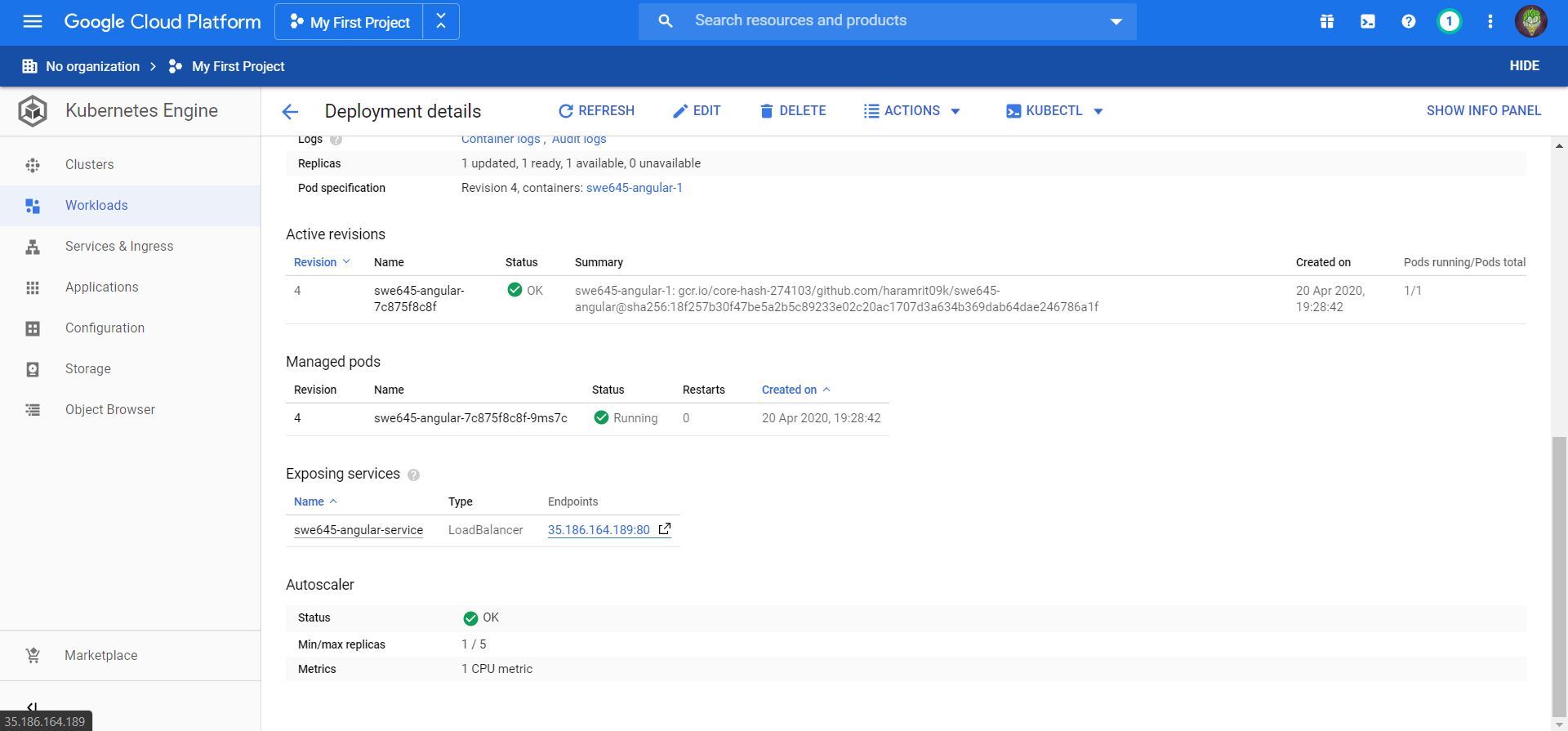
1. Expose app as a service to the public

On workloads dashboard, click on Actions -> Expose



Map port 80 (HTTP) to angular port (4200) and set service type as Load Balancer. Then EXPOSE!

After the service is exposed, an IP gets generated under “Exposing Services”



Going to that IP loads our app!

