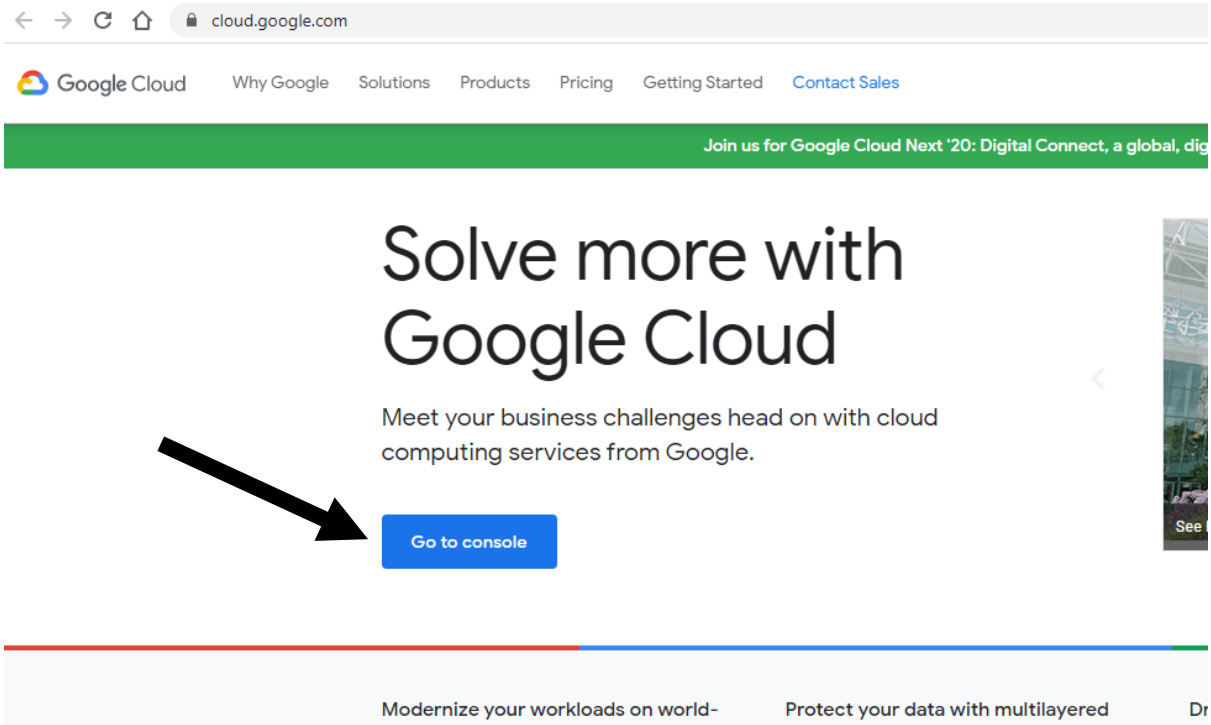
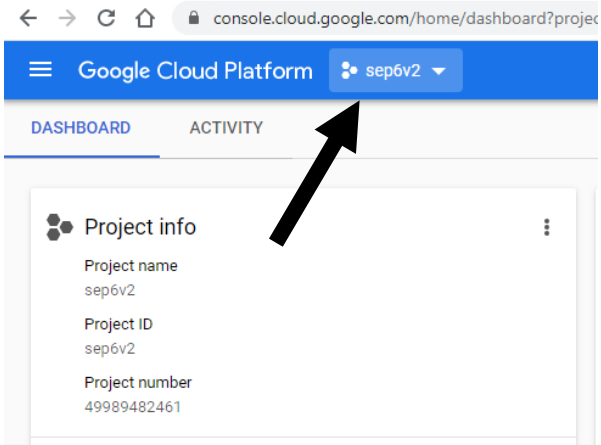
1 Creating a project

This guide describes how to setup a SQL server on google cloud.

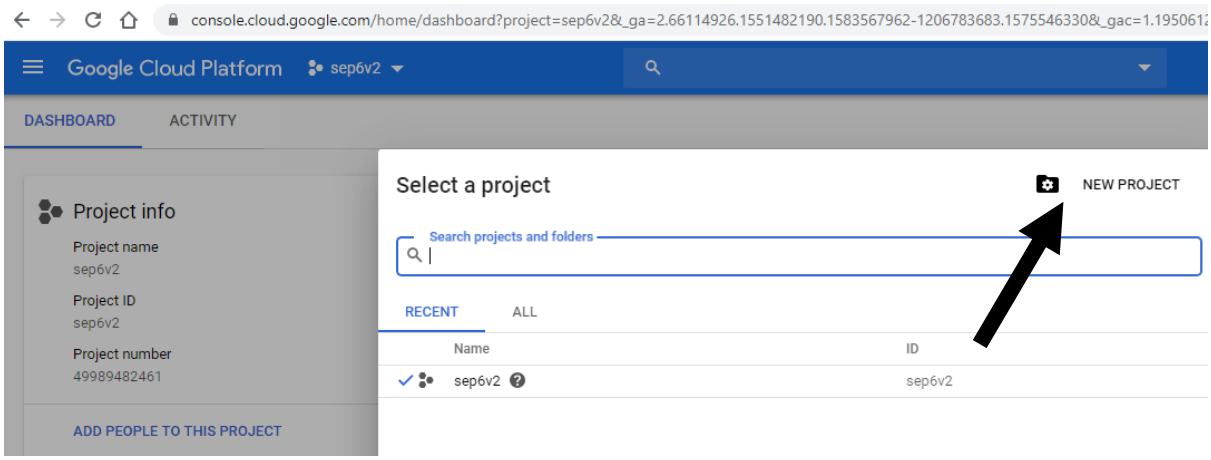
On Cloud.google.com click on Console (console.cloud.google.com)



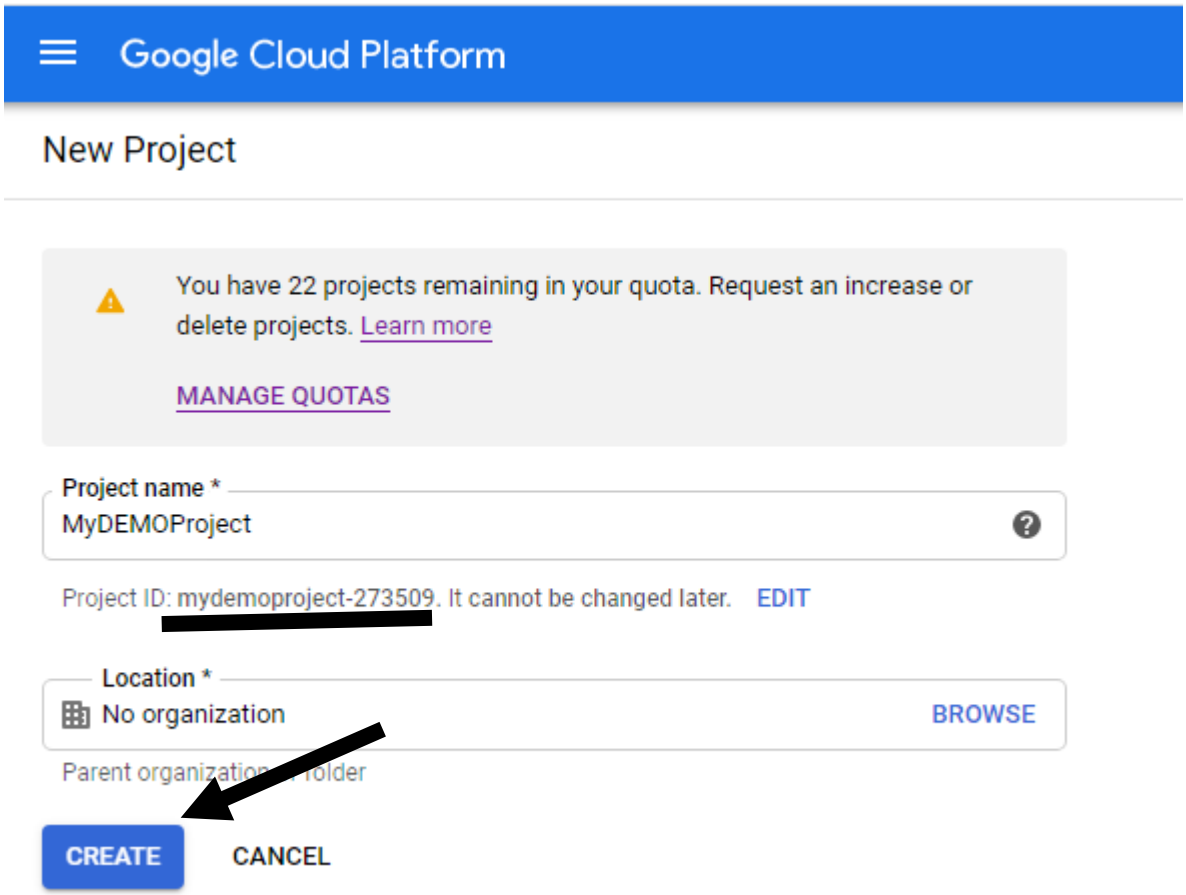
In the top you can choose between your projects. Click where the arrow points



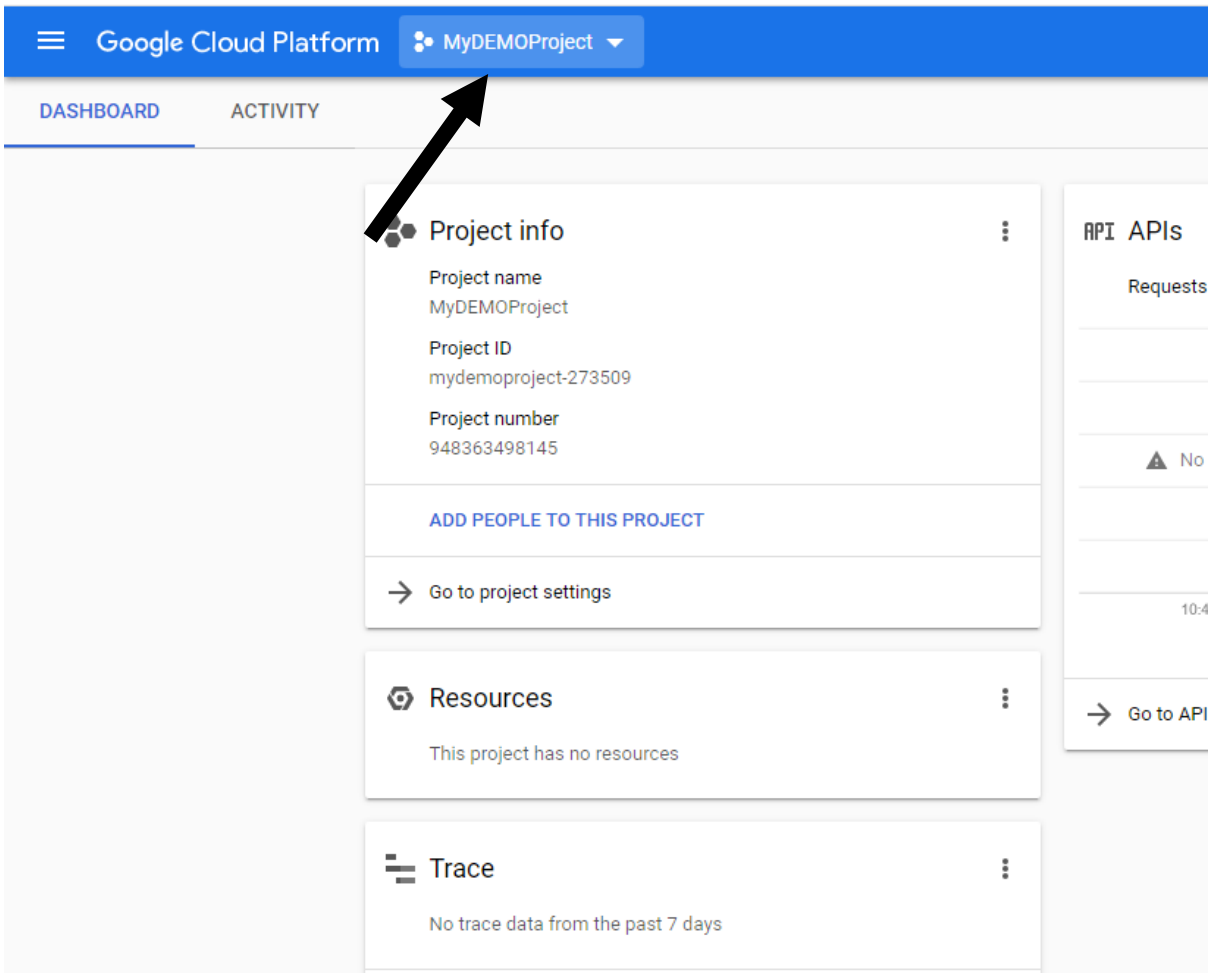
Press “Create new Project”



Choose a name, save the project ID and press “create”:

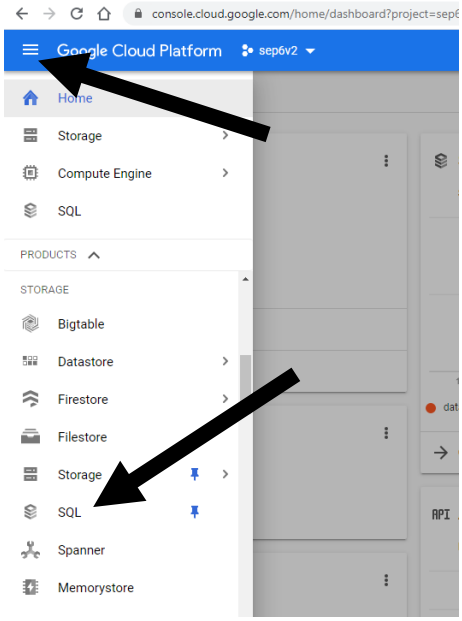


Make sure that your new project is now selected:

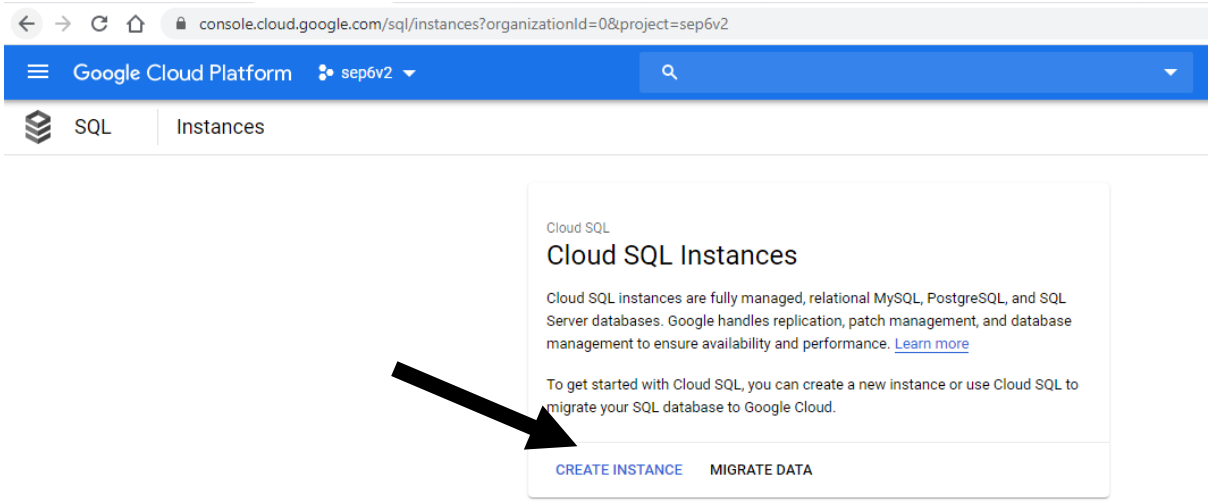


2 Create SQL database

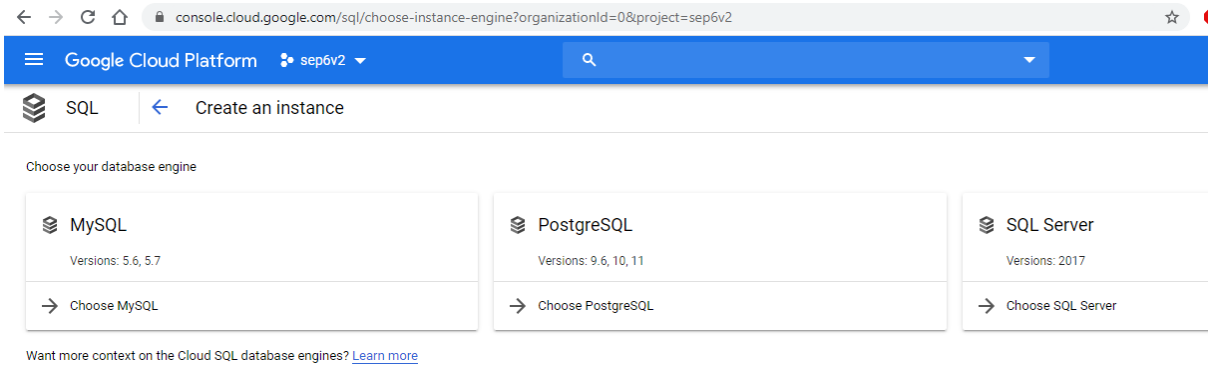
Click on the navigation window in the top left and scroll down until you find SQL and click here



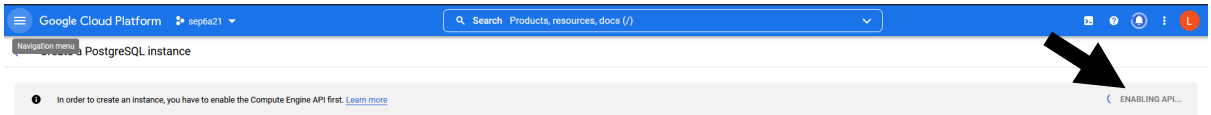
Select “CREATE INSTANCE”



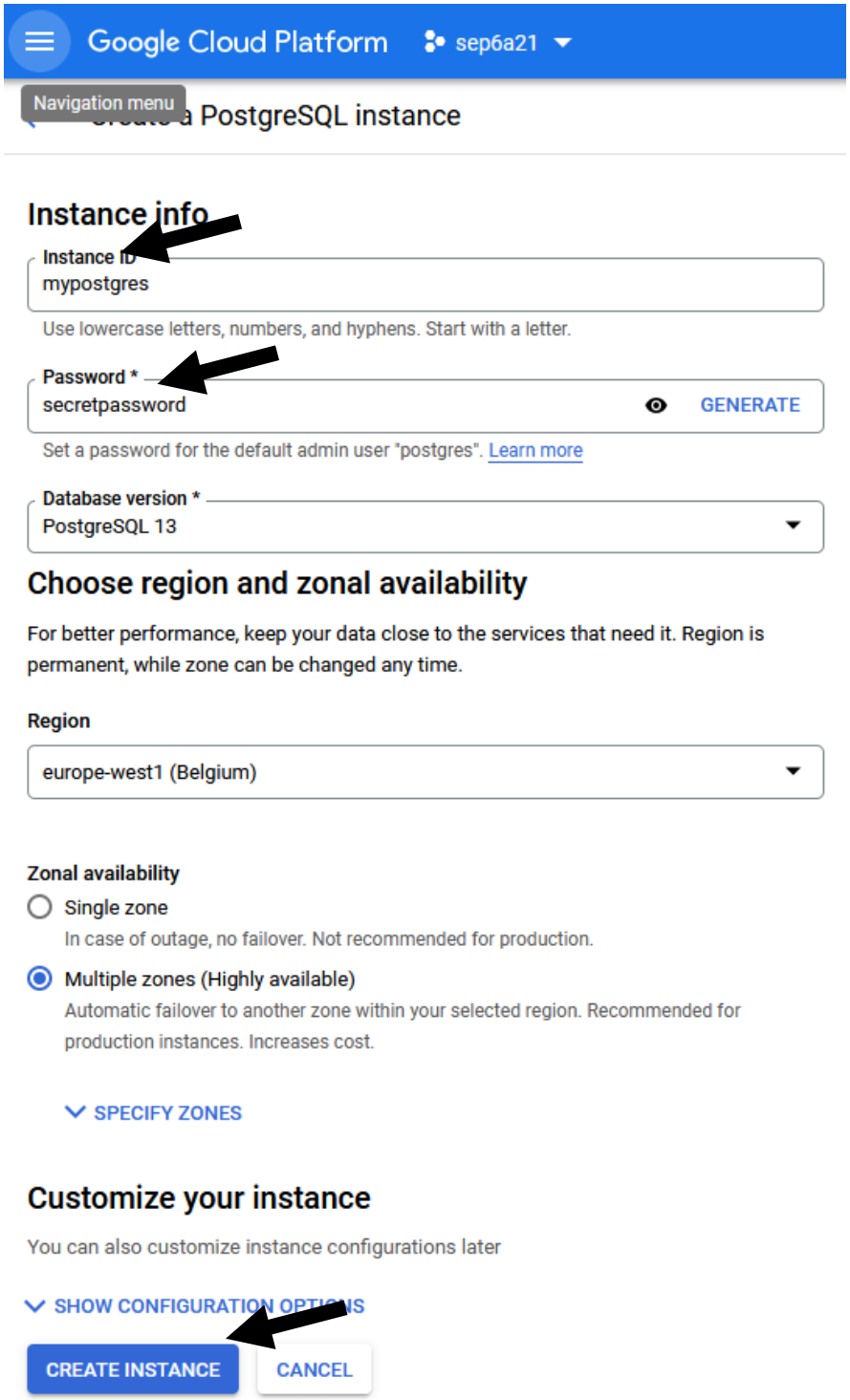
Select the database engine you prefer. In this guide PostgresSQL is used:



Then, enable API:

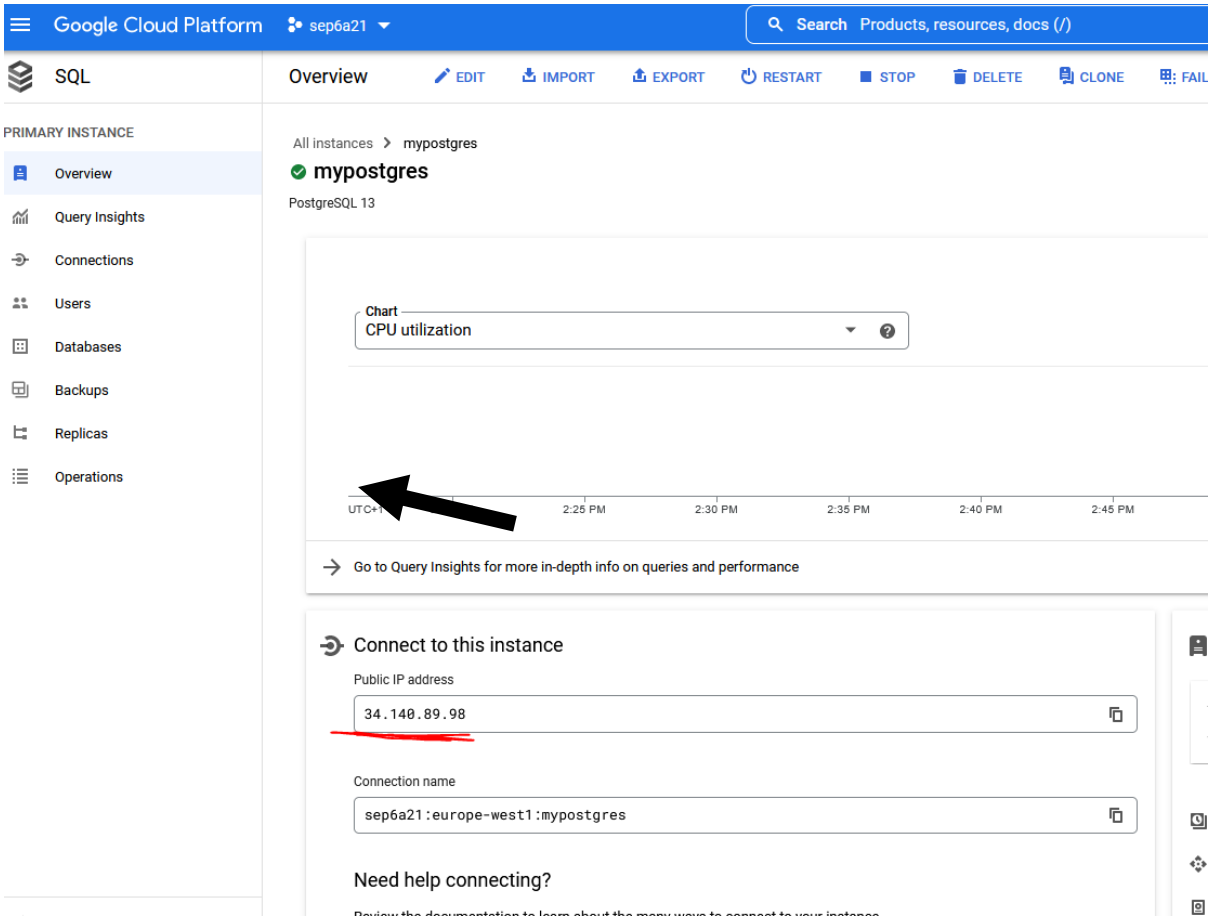


Choose an instance ID and a good password. Save the password somewhere. Then press Create.



Afterwards you will need to wait for the storage engine to be created. This will take a few minutes.

When this is done save the “public IP Address” somewhere:



Then select Connections, and “Add network”:

Google Cloud Platform sep6a21 Search Produ

SQL

Connections

PRIMARY INSTANCE

- Overview
- Query Insights
- Connections
- Users
- Databases
- Backups
- Replicas
- Operations

Release Notes

All instances > mypostgres

mypostgres

PostgreSQL 13

NETWORKING SECURITY CONNECTIVITY TESTS

Choose how you want your source to connect to this instance, then define which networks are authorized to connect. Learn more

You can use the Cloud SQL Proxy for extra security with either option. Learn more

Instance IP assignment

☐ Private IP
Assigns an internal, Google-hosted VPC IP address. Requires additional APIs and permissions. Can't be disabled once enabled. Learn more

☑ Public IP
Assigns an external, internet-accessible IP address. Requires using an authorized network or the Cloud SQL Proxy to connect to this instance. Learn more

Authorized networks

You can specify CIDR ranges to allow IP addresses in those ranges to access your instance. Learn more

You have not authorized any external networks to connect to your Cloud SQL instance. External applications can still connect to the instance through the Cloud SQL Proxy. Learn more

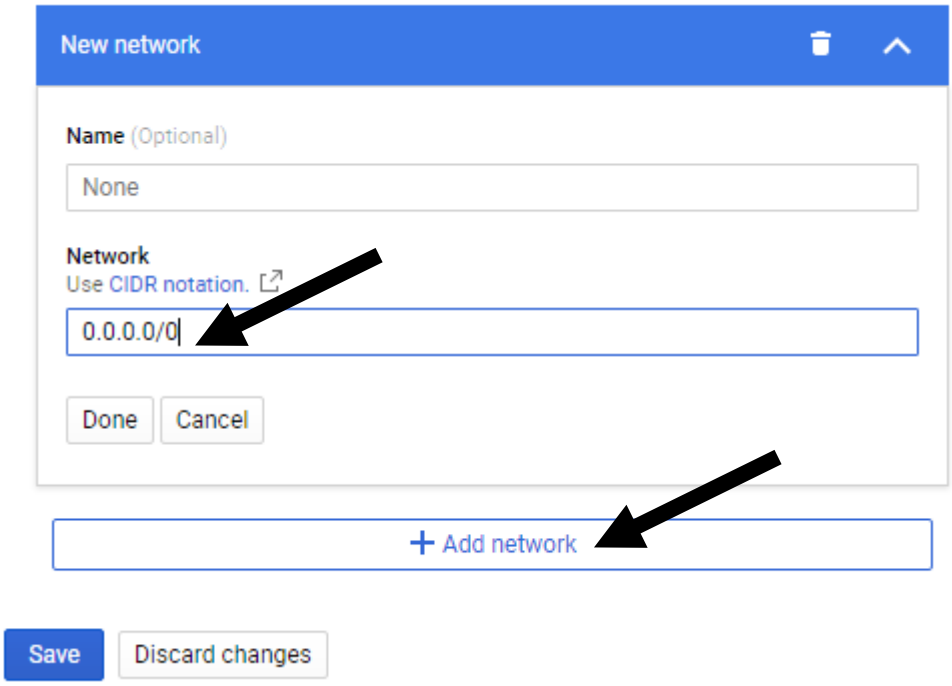
ADD NETWORK

App Engine authorization

All apps in this project are authorized by default. You can use Cloud IAM to authorize apps in other projects. Learn more

SAVE DISCARD CHANGES

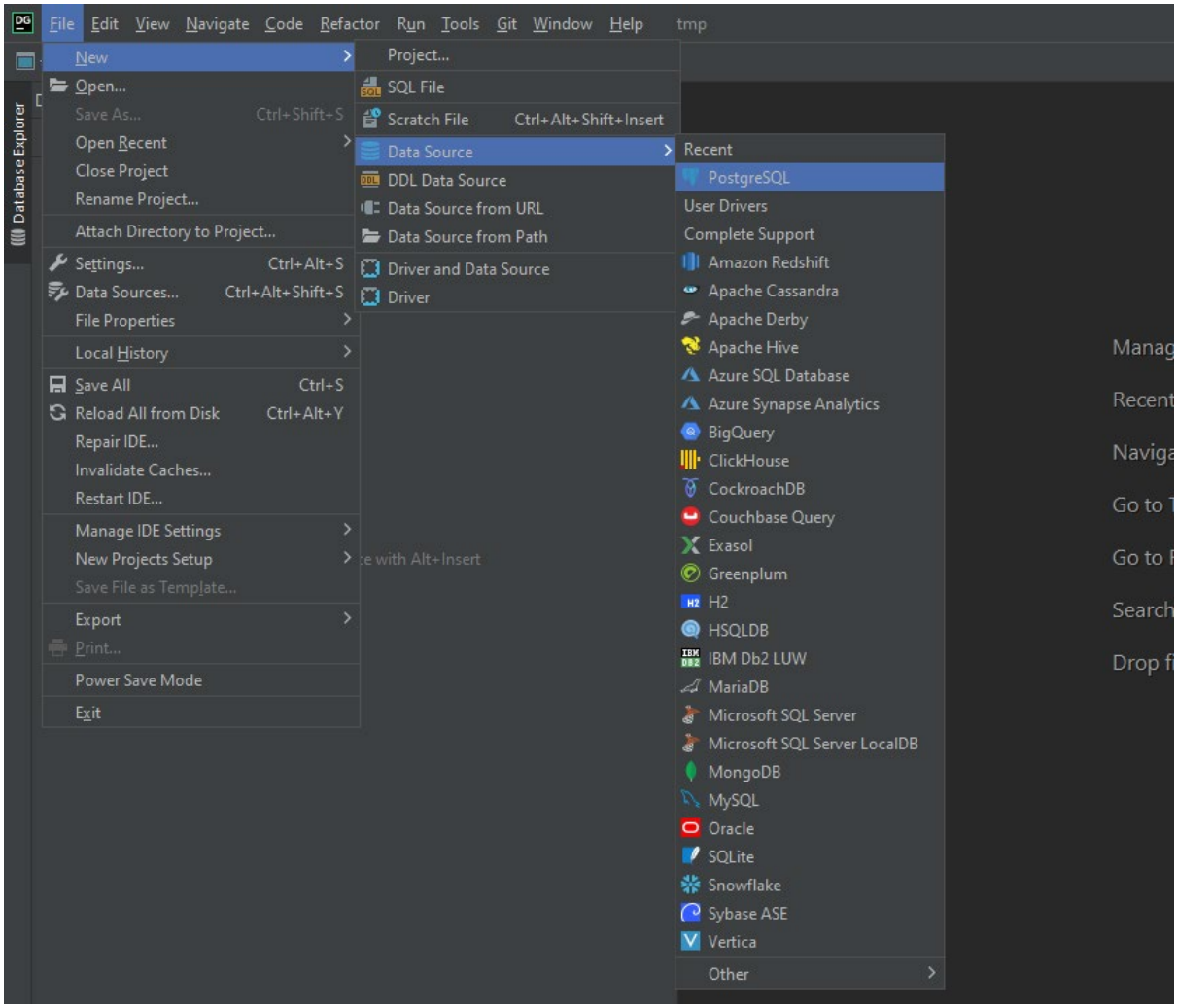
Add the IP range 0.0.0.0/0 (meaning all possible ip's) and click "done" and then "Save":



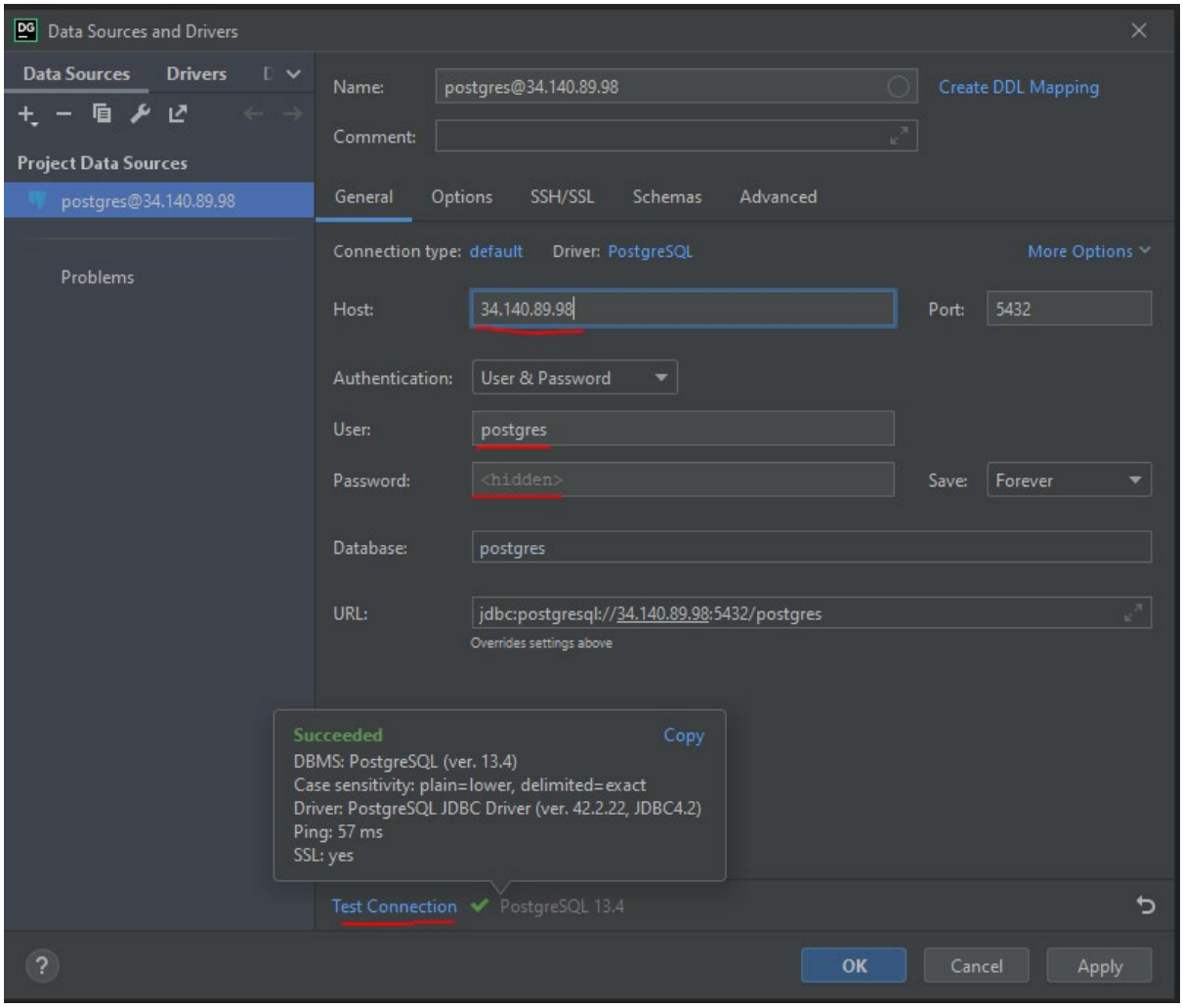
2.1 Create Database and table from datagrip

You can use whatever SQL administration tool you prefer. In this guide jetbrains datagrip is used. It can be downloaded from here.

Open datagrip and select File -> New ->Data Source ->PostgreSQL (or the engine you have chosen)



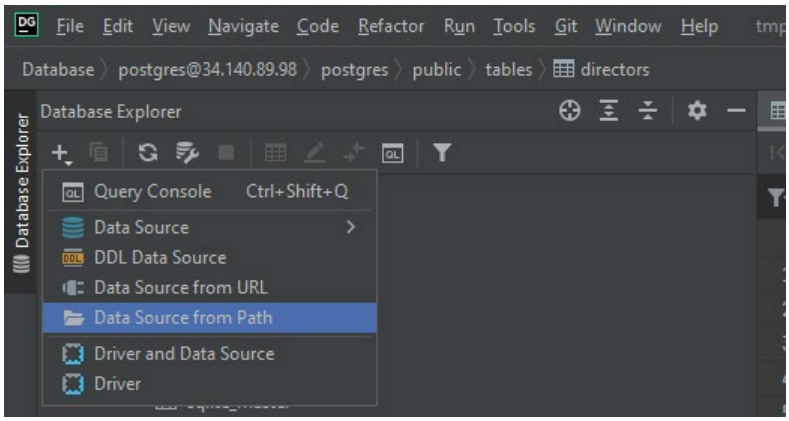
Fill out the host (the IP saved earlier), the User: which is postgres for PostgreSQL, and the password (saved earlier), and then press “Test Connection”:



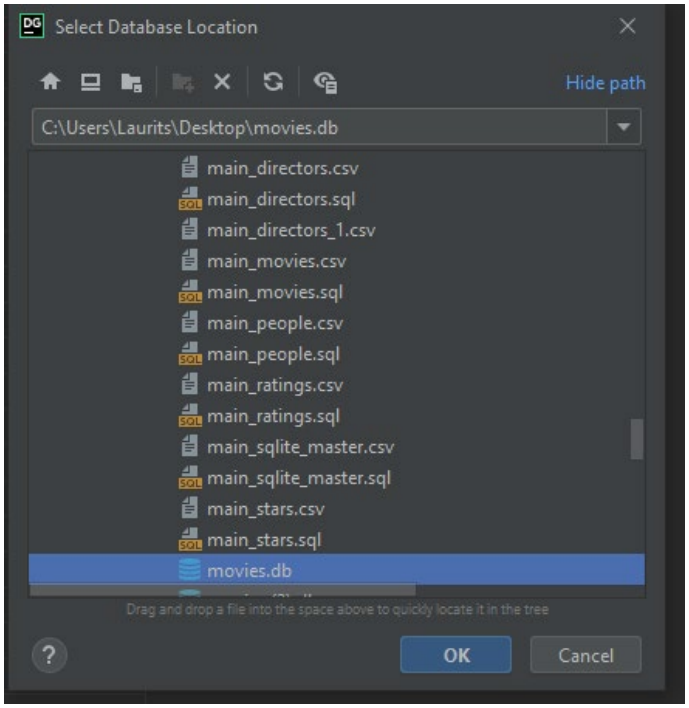
Then OK, and you have access to your database in the cloud:

2.2 Open movie.db in datagrip

The movie.db can be opened in datagrip:

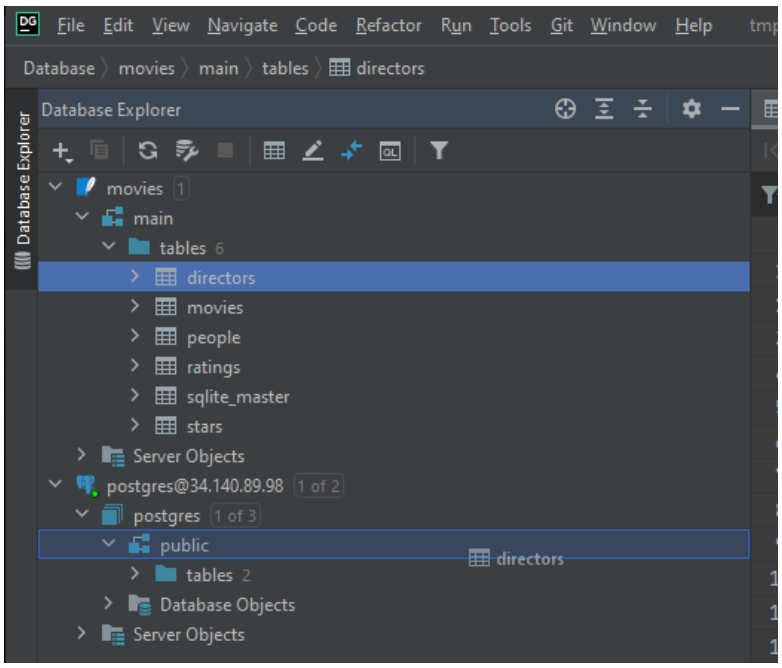


Then browse to find movie.db:



If asked which database movie.db is, then its SQLite3

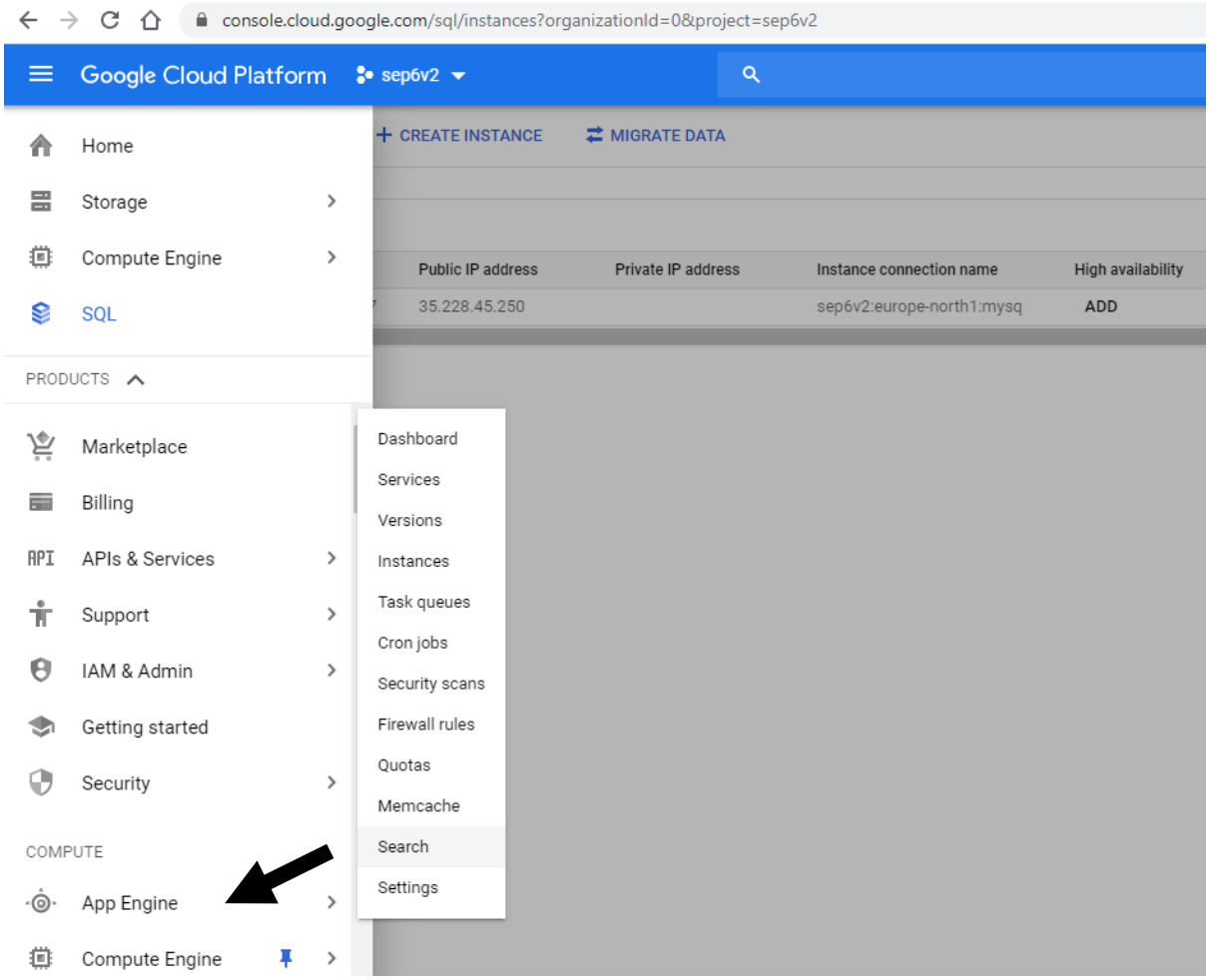
When both the local movie database (movie.db) and the google database is open in datagrip you can 'drag and drop' the tables from movie.db to the google database:



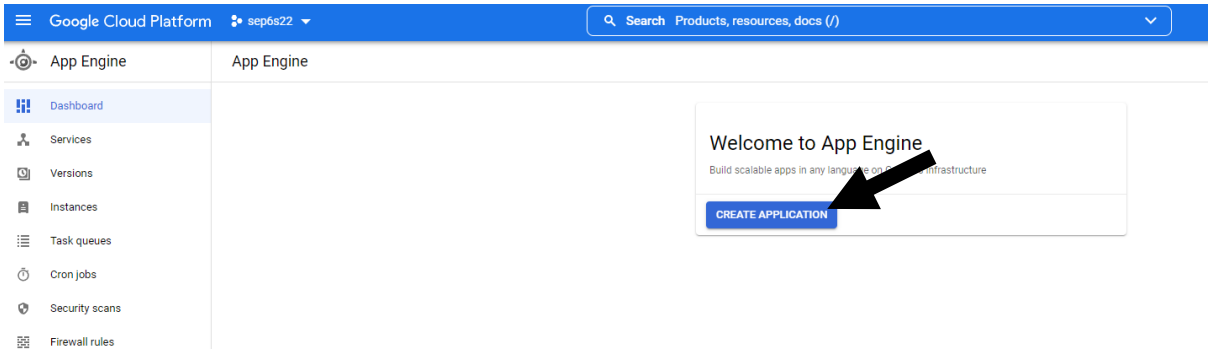
3 Create a google App engine Service

Now we need a google cloud service that can access the database.

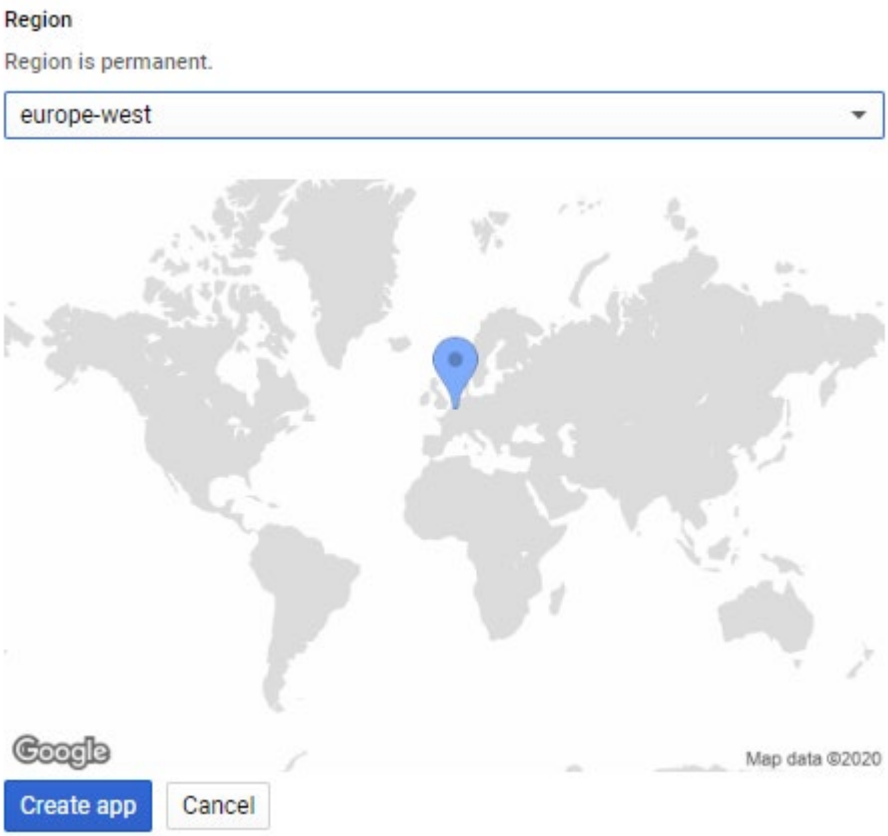
Go to the google console, click on the "navigation menu" and select "App Engine"



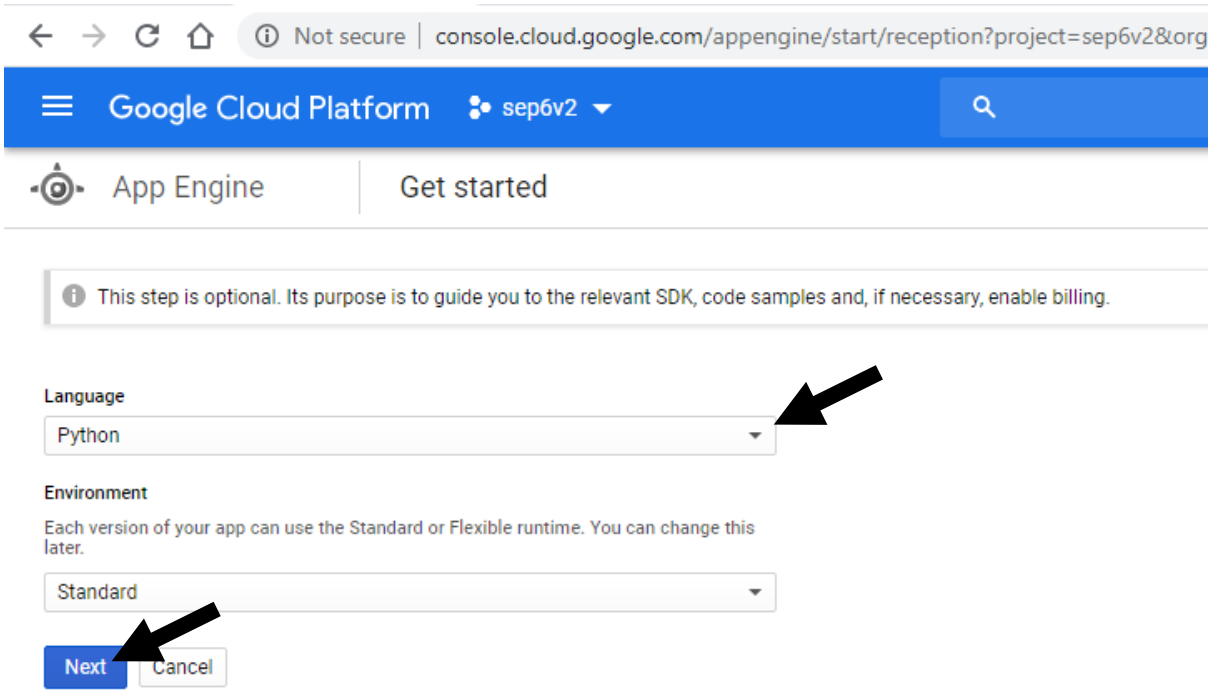
Press “Create Application”



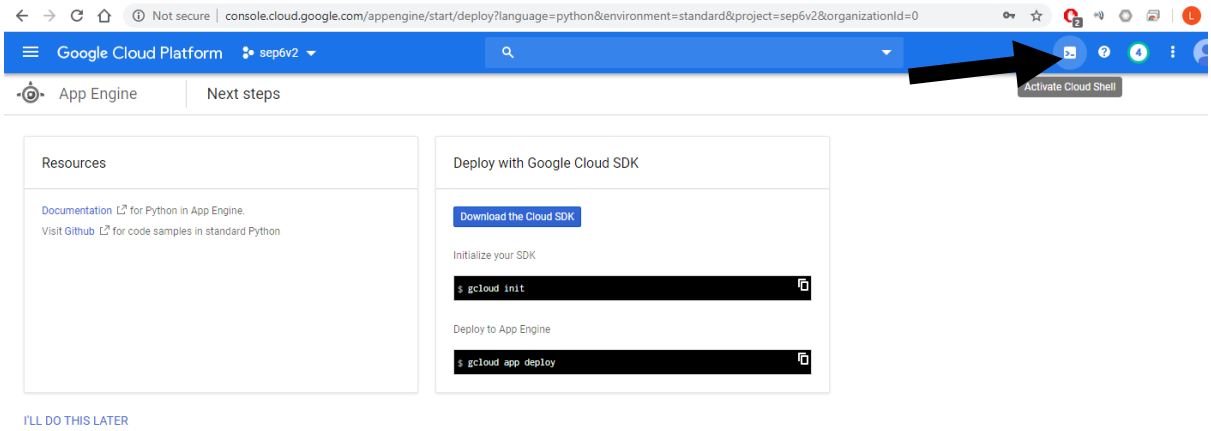
Select a region close to you (e.g. europe-west) and press create app.



Choose your programming language. In this guide Python is selected. Then press “Next”

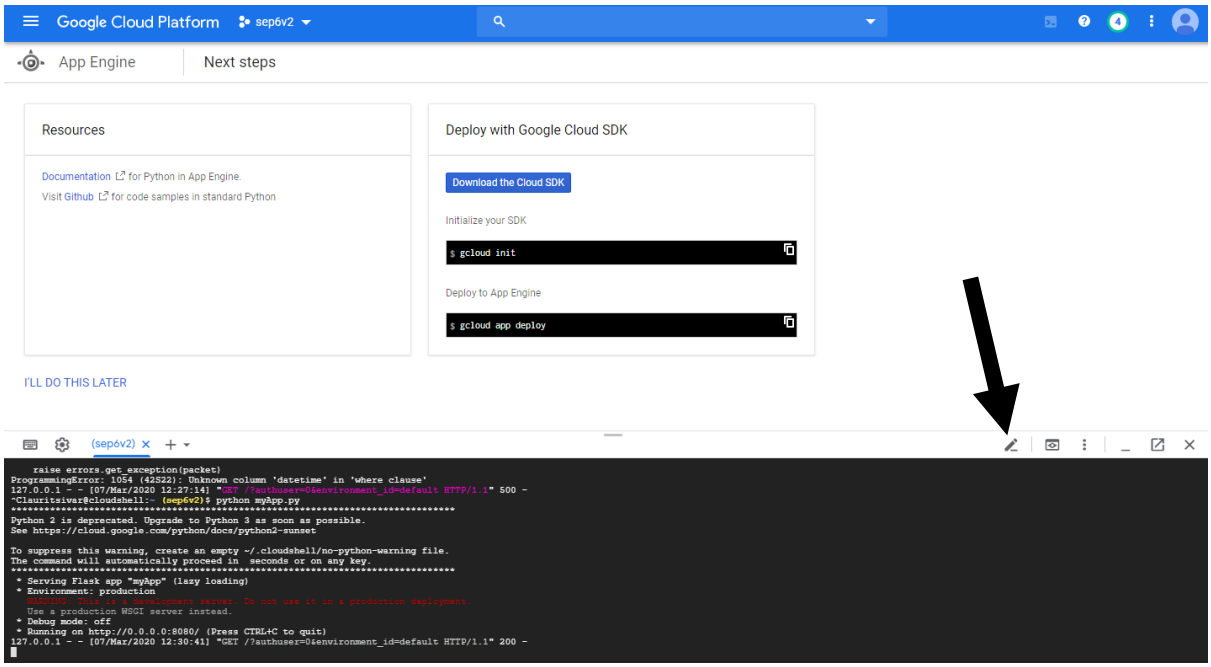


In the top right corner Activate the Cloud Shell



Write the command in the Terminal: "gcloud init" to initialize the project. Afterward follow the guidance in the terminal.

From the console you can Launch Editor:



You can now create an isolated virtual environment. This ensures that your app does not interfere with other applications that may be available on the system. You do this by entering the following commands:

```
virtualenv --python python3 ~/envs/sep6
```

Activate your newly created virtual environment:

```
source ~/envs/sep6/bin/activate
```

Now you can install any dependencies your application might have. In the console install the Flask library and the postgresql library with these two commands:

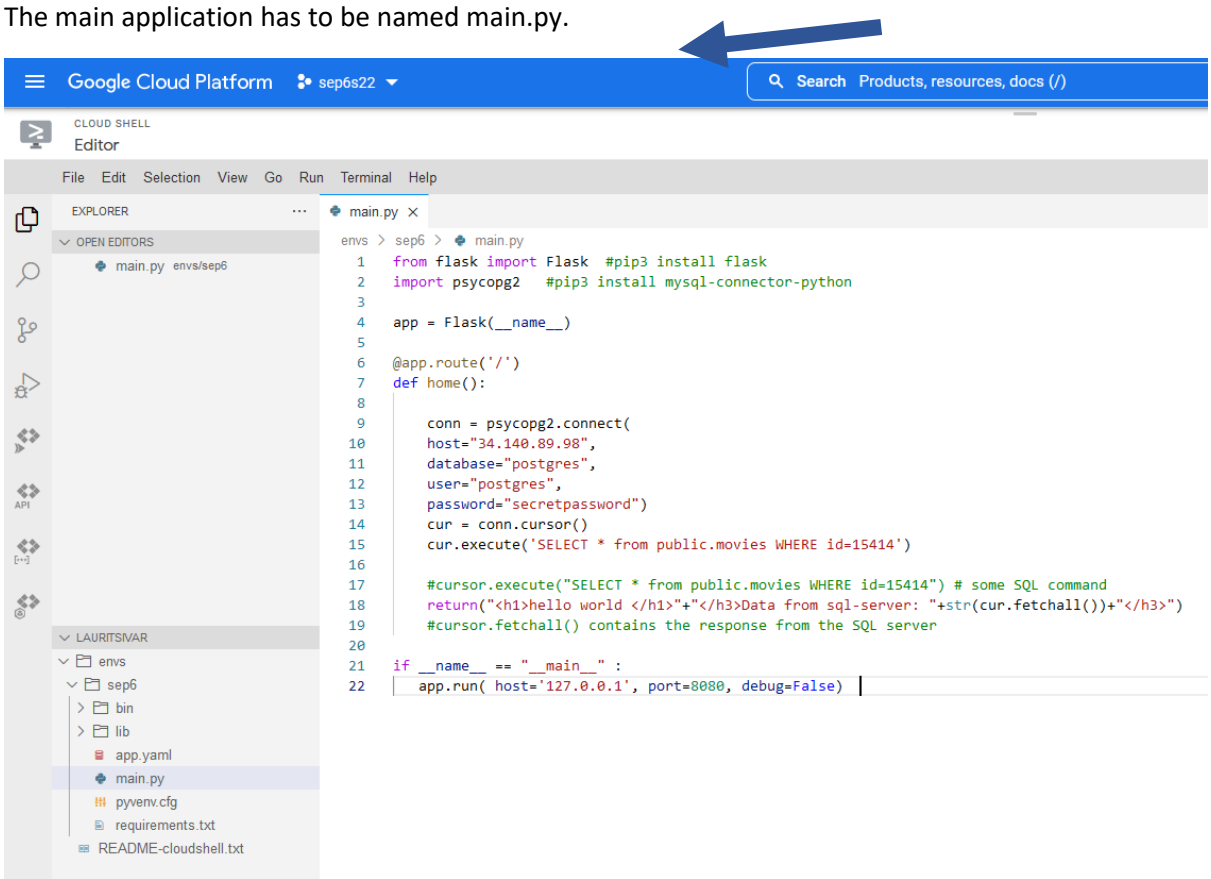
```
pip install psycopg2
```

```
pip3 install flask
```

```
(sep6) lauritsivar@cloudshell:~/envs/sep6 (sep6s22)$ pip install psycopg2
Collecting psycopg2
  Downloading psycopg2-2.9.3.tar.gz (380 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 380.6/380.6 KB 16.3 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Building wheels for collected packages: psycopg2
  Building wheel for psycopg2 (setup.py) ... done
  Created wheel for psycopg2: filename=psycopg2-2.9.3-cp39-cp39-linux_x86_64.whl size=455704 sha256=3f35263ef97b1215613769b63bb25db2b9dd60bfe654f1ed304dfb52b5029dc0
  Stored in directory: /home/lauritsivar/.cache/pip/wheels/b3/a1/6e/5a0e26314b15eb96a36263b80529ce0d64382540ac7b9544a9
Successfully built psycopg2
Installing collected packages: psycopg2
Successfully installed psycopg2-2.9.3
(sep6) lauritsivar@cloudshell:~/envs/sep6 (sep6s22)$ pip3 install flask
Requirement already satisfied: flask in ./lib/python3.9/site-packages (2.0.3)
Requirement already satisfied: click>=7.1.2 in ./lib/python3.9/site-packages (from flask) (8.0.4)
Requirement already satisfied: Jinja2>=3.0 in ./lib/python3.9/site-packages (from flask) (3.0.3)
Requirement already satisfied: Werkzeug>=2.0 in ./lib/python3.9/site-packages (from flask) (2.0.3)
Requirement already satisfied: itsdangerous>=2.0 in ./lib/python3.9/site-packages (from flask) (2.1.0)
Requirement already satisfied: MarkupSafe>=2.0 in ./lib/python3.9/site-packages (from Jinja2>=3.0->flask) (2.1.0)
(sep6) lauritsivar@cloudshell:~/envs/sep6 (sep6s22)$
```

In the editor the following code creates a site that reads data from the SQL-database:

The main application has to be named main.py.



The code can be seen below (replace IP and the "`passwd`" `line 8 with your own password and IP`):

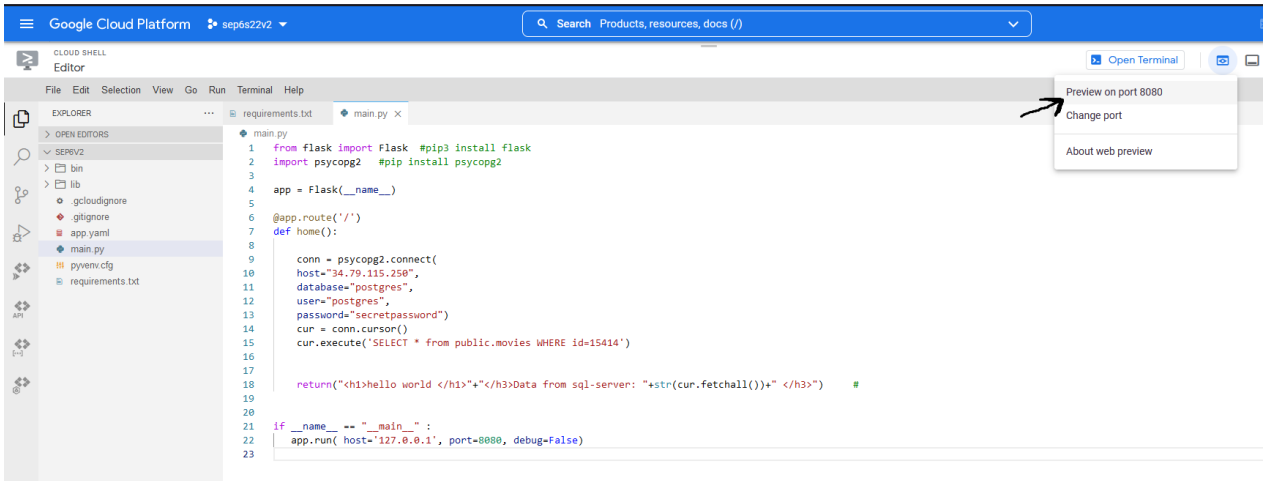
```
from flask import Flask  #pip3 install flask
import psycopg2   #pip install psycopg2

app = Flask(__name__)

@app.route('/')
def home():

    conn = psycopg2.connect(
    host="34.140.89.98",
    database="postgres",
    user="postgres",
```

```
    password="secretpassword")
    cur = conn.cursor()
    cur.execute('SELECT * from public.movies WHERE id=15414')

    return("<h1>hello world </h1>"+"</h3>Data from sql-server:
"+str(cur.fetchall())+"</h3>")


if __name__ == "__main__" :
   app.run( host='127.0.0.1', port=8080, debug=False)
```

To execute the program write the following in the console:

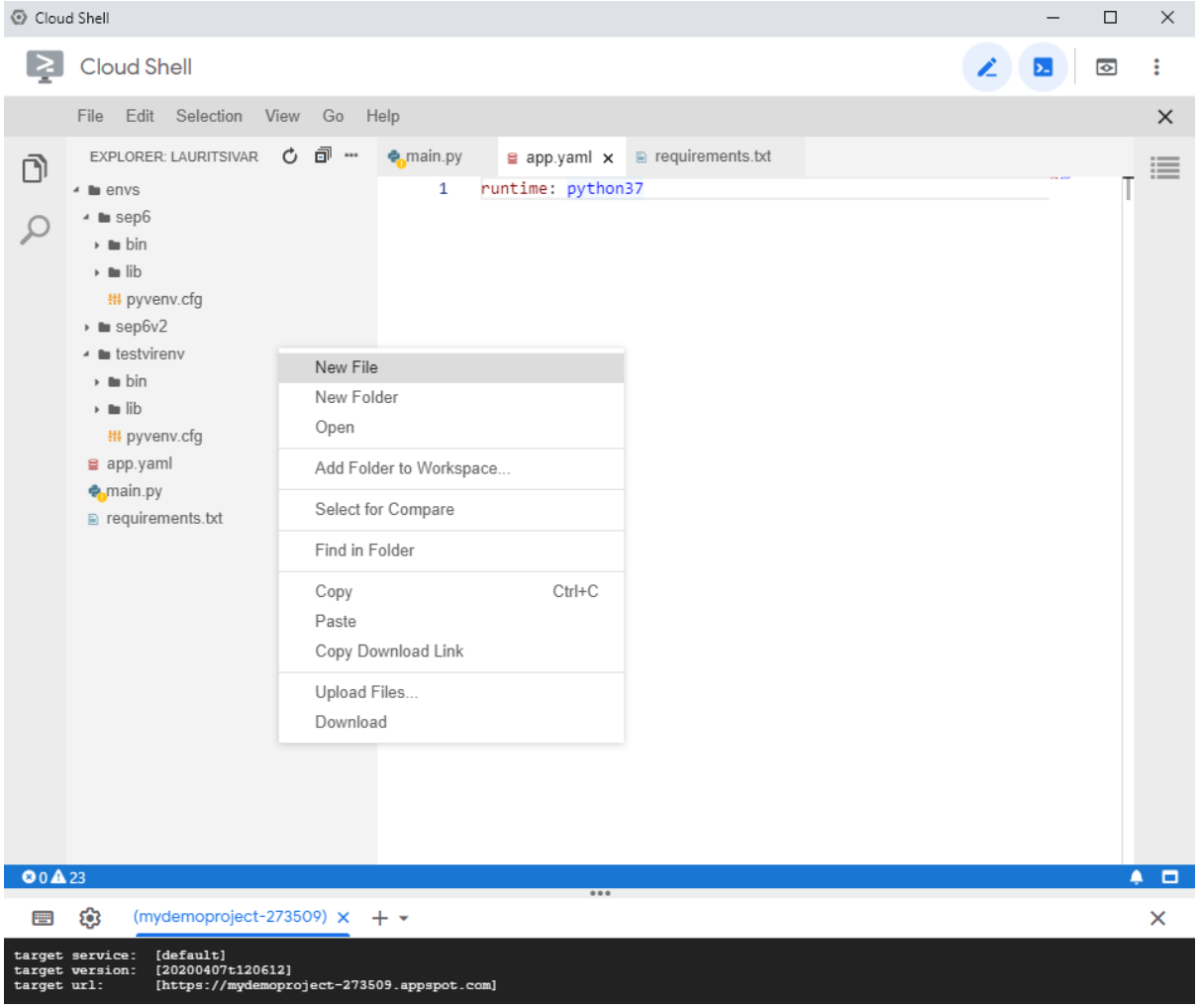
```
$ python3 main.py
```

Now the website can be previewed by clicking in the top right corner. Choose the correct port according to the code:



In this example the preview looks as seen below:

Next you need to create a .yaml-file and a requirements file for deploying the application. This is done by right clicking in Explorer window and pressing “New file”:

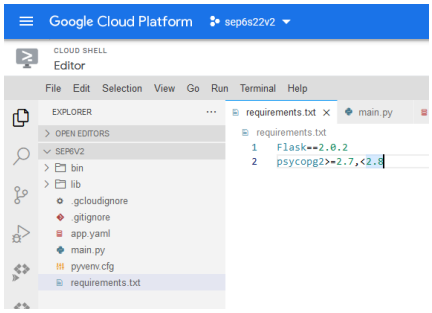


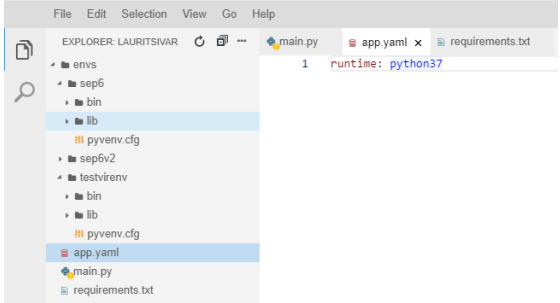
Next, create a app.yaml with information about the runtime your application is using. In this case we only need the following line:

runtime: python37

Include any dependencies for deployment by creating a requirements.txt file. In our case the file will currently contain the following two lines:

```
Flask==2.0.2
psycopg2>=2.7,<2.8
```



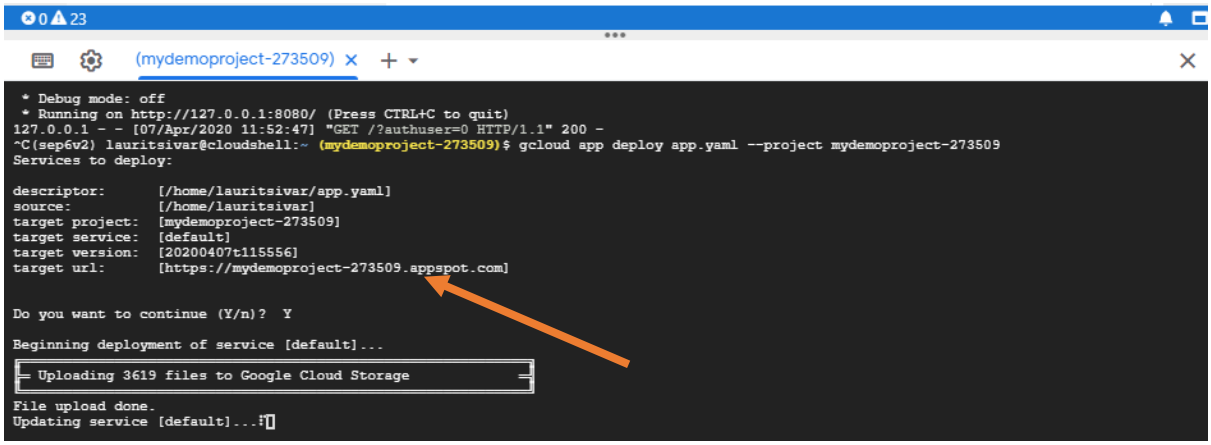


You can now deploy your app with the following command (replace the project ID with your project ID instead of mydemoproject-273509)

```
gcloud app deploy app.yaml --project mydemoproject-273509
```

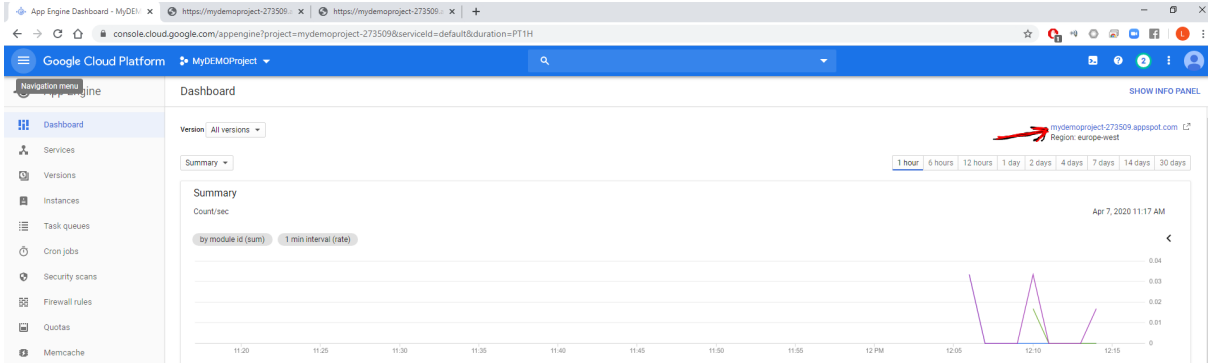
It will take some time for your project to be deployed.

While deploying your application the URL can be seen:

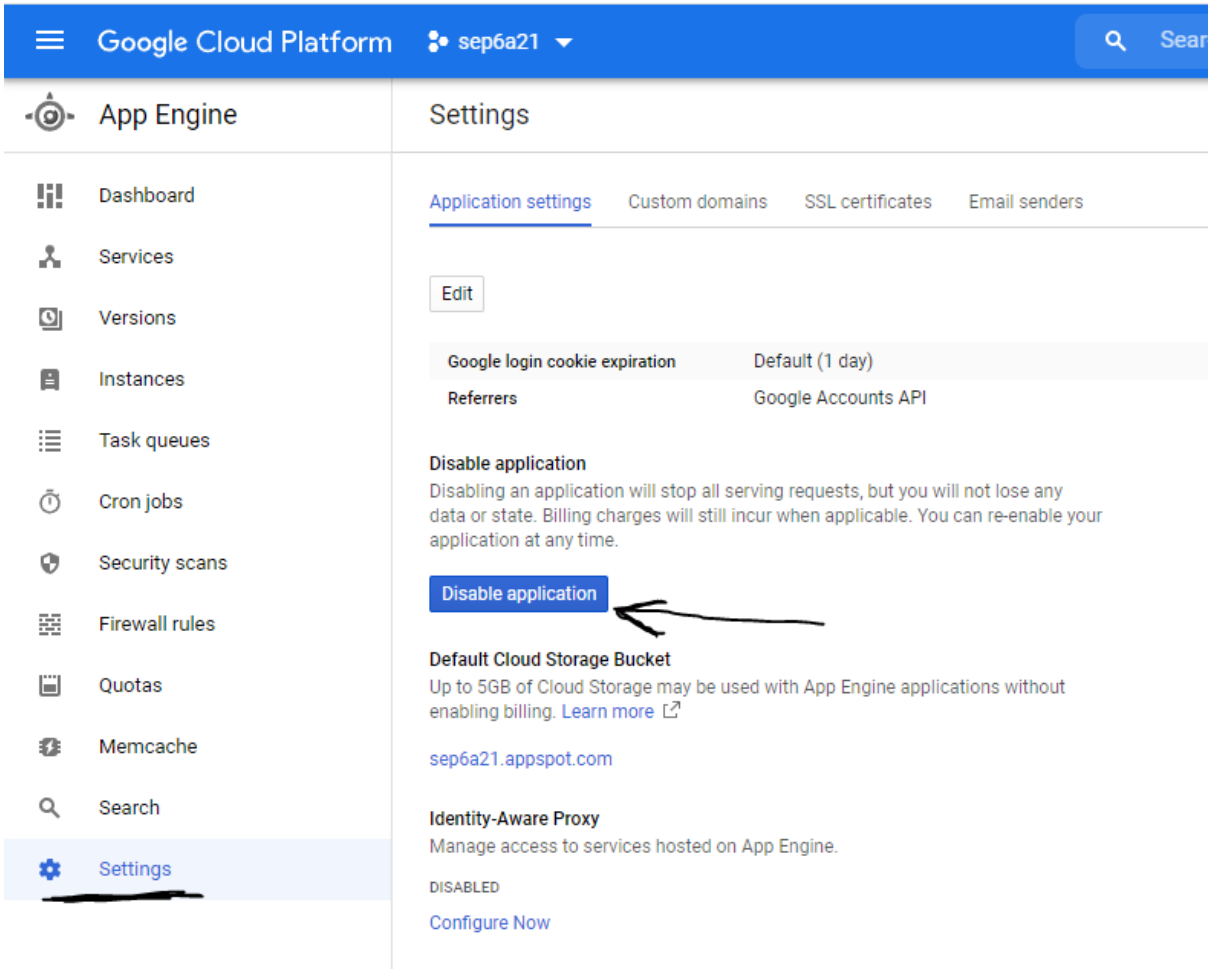


When its done, your website can be seen on that URL.

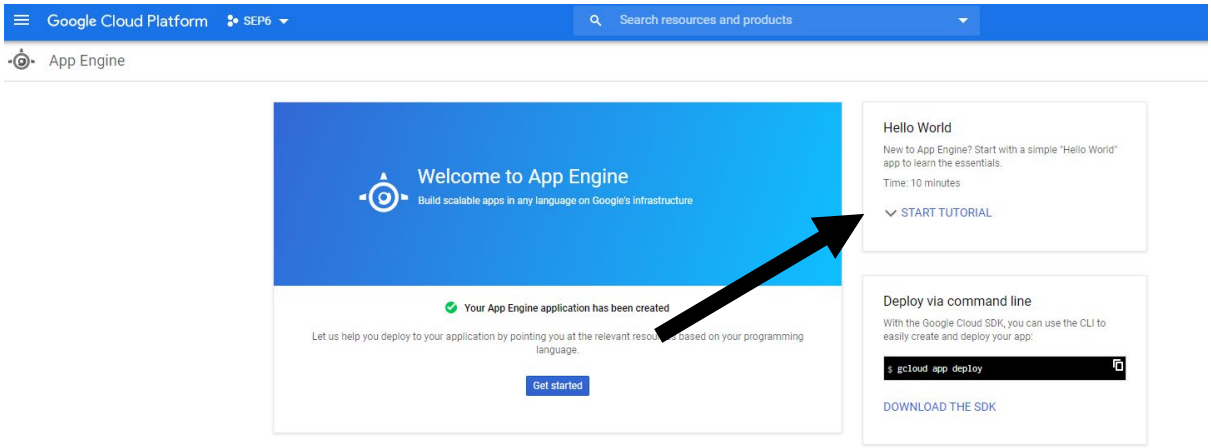
The URL can also be found in the App Engine section of console.cloud.google.com.



That is it, now you are done. This URL can be accessed from everywhere. Make sure to disable the application afterwards so you don't use all the credit: This is done in 'Settings':



If you are stuck, refer to the Getting Started tutorial in the App Engine section of console.cloud.google.com:


Google Cloud Platform
SEP6
Search resources and products
App Engine
Welcome to App Engine
Build scalable apps in any language on Google's infrastructure
Your App Engine application has been created
Let us help you deploy to your application by pointing you at the relevant resources based on your programming language.
Get started
Hello World
New to App Engine? Start with a simple "Hello World" app to learn the essentials.
Time: 10 minutes
START TUTORIAL
Deploy via command line
With the Google Cloud SDK, you can use the CLI to easily create and deploy your app:
$ gcloud app deploy
DOWNLOAD THE SDK