

Tutorial 05 to do in class – Remember to upload the repo link to Teams.

Django

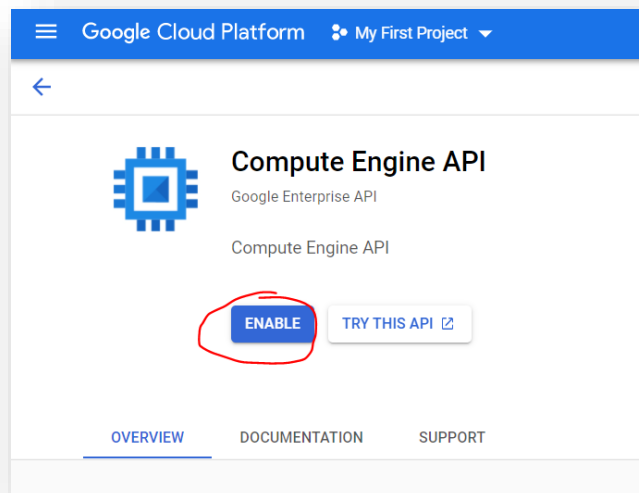
Antes de iniciar:

- Terminar los tutoriales anteriores
- Este taller muestra un ejemplo de despliegue de una aplicación Django con Docker en GCP.

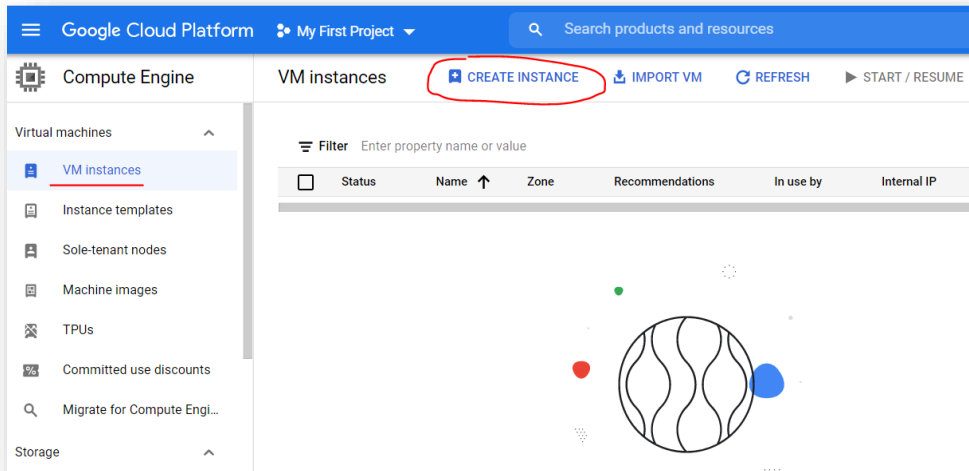
A. Creando la máquina virtual.

A1. Conéctese a su cuenta de Google Cloud (recuerde conectarse usando su correo EAFIT) <https://console.cloud.google.com/>

A2. Vaya a “Compute Engine” y active el uso de este servicio (si todavía no lo ha hecho).



A3. Luego vaya a “VM instances” -> “Create instance”.



A4. Configura la instancia con los siguientes valores. Verifica que al lado derecho aparece que la instancia gastará \$7 dólares mensuales estimados.

Google Cloud Platform VM instances configuration page.

Name (Name is permanent)
instance-docker

Labels (Optional)
+ Add label

Region (Region is permanent)
us-central1 (Iowa)

Zone (Zone is permanent)
us-central1-a

Machine configuration

Machine family
General-purpose | Compute-optimized | Memory-optimized | GPU
Machine types for common workloads, optimized for cost and flexibility

Series
E2

CPU platform selection based on availability

Machine type
e2-micro (2 vCPU, 1 GB memory)

vCPU	Memory	GPUs
1 shared core	1 GB	-

✓ CPU platform and GPU

\$7.12 monthly estimate
That's about \$0.01 hourly
Pay for what you use: No upfront costs and per second billing
Details

También permite el acceso por HTTP, y HTTPS de la siguiente manera, y luego da click en “Create”.

Boot disk ?

New 10 GB balanced persistent disk
Image
Debian GNU/Linux 10 (buster) [Change](#)

Identity and API access ?

Service account ?
Compute Engine default service account

Access scopes ?
☒ Allow default access
☐ Allow full access to all Cloud APIs
☐ Set access for each API

Firewall ?
 Add tags and firewall rules to allow specific network traffic from the Internet
☒ Allow HTTP traffic
☒ Allow HTTPS traffic
[Management, security, disks, networking, sole tenancy](#)

You will be billed for this instance. [Compute Engine pricing](#)

[Create](#) [Cancel](#)

Equivalent [REST](#) or [command line](#)

A5. Finalmente, luego de un par de minutos, te deberá aparecer la instancia disponible.

VM instances are highly configurable virtual machines for running workloads on Google infrastructure. [Learn more](#)

Filter Enter property name or value

<input type="checkbox"/>	Status	Name ↑	Zone	Recommendations	In use by	Internal IP	External IP
<input type="checkbox"/>	✓	instance-docker	us-central1-a			10.128.0.4 (nic0)	34.72.21.78 ↗
<input type="checkbox"/>	✓	instance-project1	us-central1-a	💡 Save \$6 / mo		10.128.0.3 (nic0)	34.133.88.38 ↗

Related actions

B. Accediendo a la máquina virtual.

B1. Da click sobre el nombre de la instancia creada.

B2. Selecciona “SSH” -> “Open in browser window”. Lo cual abrirá una nueva ventana, desde donde podremos administrar la instancia creada.

```
dcorreab@instance-docker: ~ - Google Chrome
ssh.cloud.google.com/projects/gold-setup-321912/zones/us-central1-a/instances/instance-docker?useA
Connected, host fingerprint: ssh-rsa 0 D2:CB:09:07:CB:62:D0:D4:7B:33:D9:37:14:1F
:C7:F6:91:08:4A:6D:10:FA:AF:DB:98:93:DB:56:AF:A7:2B:95
Linux instance-docker 4.19.0-17-cloud-amd64 #1 SMP Debian 4.19.194-3 (2021-07-18
) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
dcorreab@instance-docker:~$
```

C. Instalar Docker

C1. Ejecute los siguientes comandos para poder instalar Docker en distribuciones Debian.

```
sudo apt-get update
```

```
sudo apt-get install -y apt-transport-https ca-certificates curl gnupg2 software-properties-common
```

```
curl -fsSL https://download.docker.com/linux/debian/gpg | sudo apt-key add -
```

```
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/debian buster
stable"
```

```
sudo apt-get update
```

```
sudo apt-get install -y docker-ce docker-ce-cli containerd.io
```

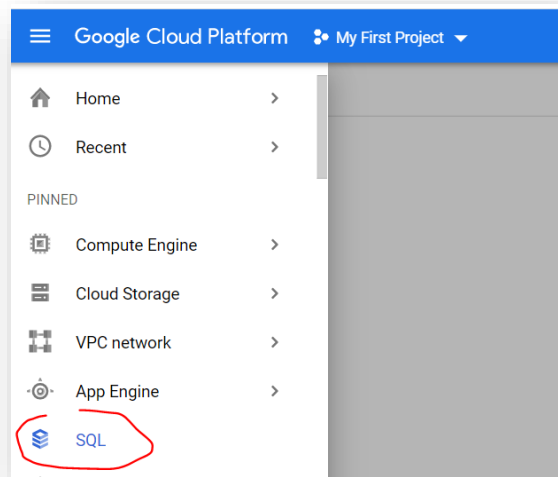
C2. Verifique si Docker quedó instalado, corriendo el siguiente comando:

```
docker -v
```

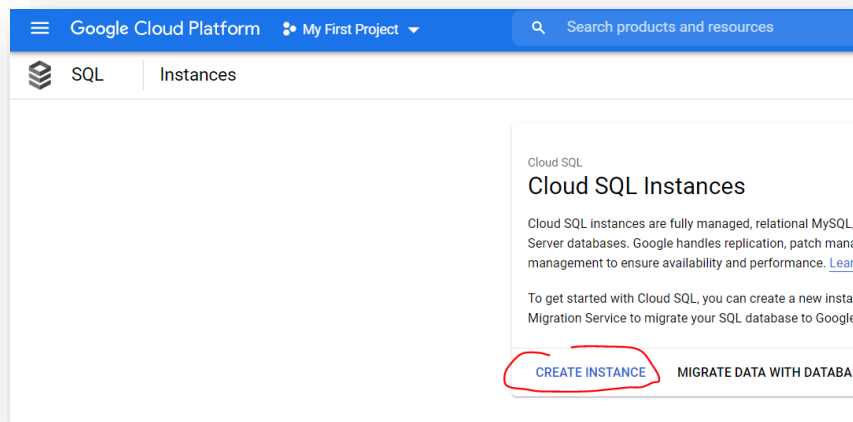
```
dcorreab@instance-docker:~$ docker -v
Docker version 23.0.2, build 569dd73
```

D. Instalar la base de datos en Cloud SQL

D1. En GCP, vaya a SQL.



D2. Luego de click en “Create Instance”



D3. Selecciona MySQL.

D4. Configure la base de datos con los siguientes valores. En *password*, coloque el *password* para el usuario root de la base datos.

← Create a MySQL instance

Instance info

Instance ID *

laravelbd

Use lowercase letters, numbers, and hyphens. Start with a letter.

Password *

••••••••



GENERATE

Set a password for the root user. [Learn more](#)

☐ No password

✓ PASSWORD POLICY

Database version *

MySQL 8.0



✓ SHOW MINOR VERSIONS

Choose a configuration to start with

These suggested configurations will pre-fill this form as a starting point for creating an instance. You can customize as needed later.

☐ Production

Optimized for the most critical workloads. Highly available, performant, and durable.

☐ Development

Performant but not highly available, while reducing cost by provisioning less compute and storage.

☒ Sandbox

Good for getting started and kicking the tires. Just the basics at minimal cost.

Nota: si no le aparece Sandbox use Development

Zonal availability

☒ Single zone

In case of outage, no failover. Not recommended for production.

☐ Multiple zones (Highly available)

Automatic failover to another zone within your selected region. Recommended for production instances. Increases cost.

▼ SPECIFY ZONES

Customize your instance

You can also customize instance configurations later

Machine type

Machine Type

Choose a preset or customize your own. For better performance, choose a machine type with enough memory to hold your largest table.

Lightweight

☒ 1 vCPU, 3.75 GB

☐ 2 vCPU, 3.75 GB

☐ 4 vCPU, 3.75 GB

☐ Custom

Storage type

Choice is permanent. Storage type affects performance.

☐ SSD (Recommended)

Most popular choice. Lower latency than HDD with higher QPS and data throughput.

☒ HDD

Lower performance than SSD with lower storage rates.

Storage capacity

10 - 65,536 GB. Higher capacity improves performance, up to the limits set by the machine type. Capacity can't be decreased later.

☒ 10 GB

☐ 20 GB

☐ 100 GB

☐ 200 GB

☐ Custom

☐ Enable automatic storage increases

If enabled, whenever you are nearing capacity, storage will be incrementally (and permanently) increased. [Learn more](#)

Data Protection

Automated backups and point-in-time recovery

Protect your data from loss at a minimal cost. [Learn more](#)

☐ Automate backups

☐ Enable point-in-time recovery

Allows you to recover data from a specific point in time, down to a fraction of a second. Enables binary logs (required for replication).

Instance deletion protection

Safeguard against accidental deletion and data loss. [Learn more](#)

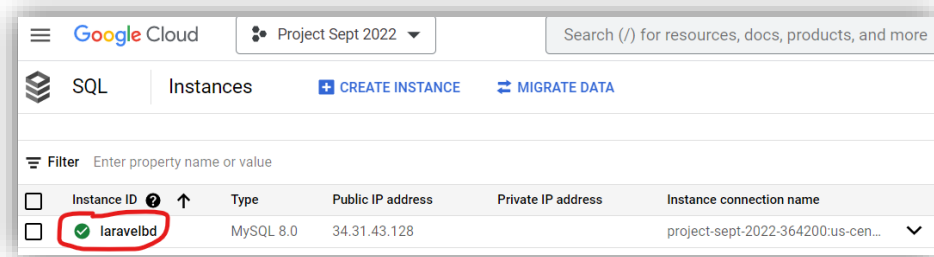
☒ Enable deletion protection

If enabled, this instance won't be able to be deleted until this feature is disabled

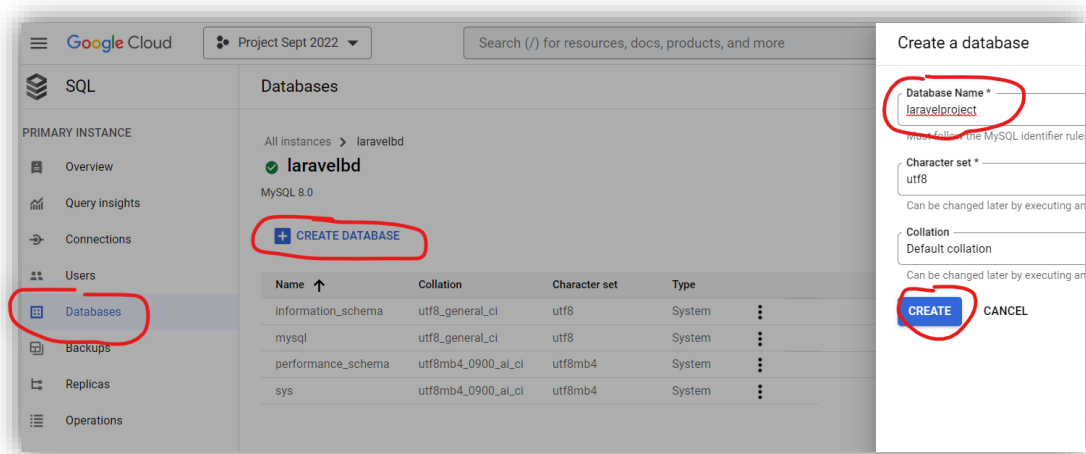
D5. A continuación, se muestra el resumen de la creación de la base de datos. Luego de click en “Create Instance”.

Summary	
Region	us-central1 (Iowa)
DB Version	MySQL 8.0
vCPUs	1 vCPU
Memory	3.75 GB
Storage	10 GB
Network throughput (MB/s) ?	250 of 2,000
Disk throughput (MB/s) ?	Read: 1.2 of 240.0
	Write: 1.2 of 75.8
IOPS ?	Read: 7.5 of 3,000
	Write: 15 of 15,000
Connections	Public IP
Backup	Manual
Availability	Single zone
Point-in-time recovery	Disabled

D6. Una vez creada la base de datos (toma hasta 5 minutos), vaya a su instancia de base datos.



D7. Cree la base de datos para su proyecto (yo le puse *django*project).



D8. Luego parece en su base de datos, y luego vaya a “Connections”, de click en “Networking”, y baje hasta dar click en “Add Network” y allí autorice el acceso a todo mundo con la siguiente configuración (0.0.0.0/0) y click en “Done” y “Save”.

PRIMARY INSTANCE

Overview

Query insights

Connections

Users

Databases

Backups

Replicas

Operations

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 added 0.0.0.0/0 as an allowed network. This prefix will allow any IPv4 client to pass the network firewall and make login attempts to your instance, including clients you did not intend to allow. Clients still need valid credentials to successfully log in to your instance.

New network

Name

all

Use [CIDR notation](#)

Network *

0.0.0.0/0

Example: 199.27.25.0/24

CANCEL

DONE

E. Configurar proyecto Django con Docker

E1. Vuelva a conectarse por SSH a su instancia Docker. Y ejecute el siguiente comando:

```
sudo docker container ls
```

Le deberá salir algo como lo siguiente:

```
dcorreab@instance-docker:~$ sudo docker container ls
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS        NAMES
```

E2. Creemos un contenedor “hola mundo” para verificar que todo está funcionando:

```
sudo docker container run hello-world
```

Le deberá salir algo como lo siguiente:

```
dcorreab@instance-docker:~$ sudo docker container run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
2db29710123e: Pull complete
Digest: sha256:fffb13da98453e0f04d33a6eee5bb8e46ee50d08ebel7735fc0779d0349e889e9
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
   (amd64)
3. The Docker daemon created a new container from that image which runs the
   executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
   to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/
```

E3. Ahora descargaremos el proyecto Django en nuestra instancia. A continuación, utilizaré un proyecto de ejemplo. Luego lo puede reemplazar por su propio proyecto.

```
git clone https://github.com/Nram94/djangoDocker.git
```

E4. Una vez descargado el proyecto, ubíquese en la carpeta del proyecto. En este caso yo utilice el siguiente comando:

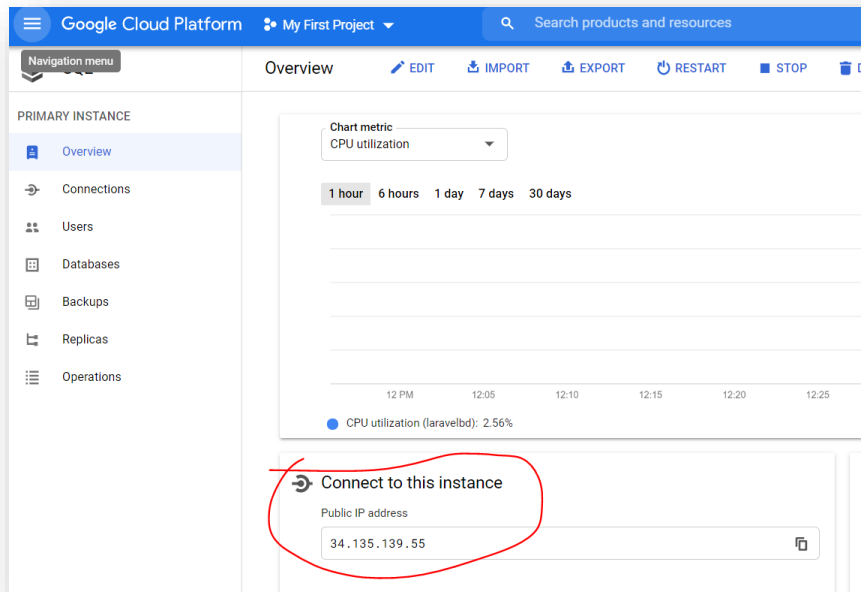
```
cd djangoDocker
```

```
nram_vel94@instance-docker:~$ git clone https://github.com/Nram94/djangoDocker.git
Cloning into 'djangoDocker'...
remote: Enumerating objects: 268, done.
remote: Counting objects: 100% (268/268), done.
remote: Compressing objects: 100% (131/131), done.
remote: Total 268 (delta 127), reused 265 (delta 124), pack-reused 0
Receiving objects: 100% (268/268), 2.17 MiB | 12.17 MiB/s, done.
Resolving deltas: 100% (127/127), done.
nram_vel94@instance-docker:~$ ls
djangoDocker
```

E5. Mueva el archivo `.env.example` a `.env` con el siguiente comando:

```
sudo mv .env.example .env
```

E6. Modifique el archivo `.env` (coloque los datos de la base de datos de la siguiente manera). Reemplace los valores `DB_HOST`, `DB_DATABASE`, `DB_USERNAME` y `DB_PASSWORD` con los que usted creó. Una vez haga los cambios en el servidor, utilice “ctrl+x” -> luego “y” -> luego “enter” para guardar. El `DB_HOST` lo puede encontrar según el siguiente pantallazo desde GCP.



```
nano .env
```

```
LOG_CHANNEL=stack
LOG_DEPRECATIONS_CHANNEL=null
LOG_LEVEL=debug
DB_CONNECTION=mysql
DB_HOST=34.31.43.128
DB_PORT=3306
DB_DATABASE=laravelproject
DB_USERNAME=root
DB_PASSWORD=12345678
BROADCAST_DRIVER=log
CACHE_DRIVER=file
FILESYSTEM_DISK=local
QUEUE_CONNECTION=sync
SESSION_DRIVER=file
SESSION_LIFETIME=120
```

Nota: recuerde colocar su password.

F. Correr proyecto Django con Docker

F1. Ahora crearemos una imagen en Docker con el proyecto anterior. Estando “parado” en la raíz del proyecto, verifique con el comando “ls” que en esa ruta se encuentra un archivo Dockerfile (si no lo encuentra, copie y pegue el archivo Dockerfile del proyecto de prueba <https://github.com/Nram94/djangoDocker/blob/main/Dockerfile>).

```
nram_vel94@instance-docker:~/djangoDocker$ ls
Dockerfile  db.sqlite3      helloworld_project  pages
accounts    docker-compose.yml  manage.py           requirements.txt
```

F2. Luego ejecute el siguiente comando (se iniciará un proceso de múltiples pasos donde se creará la imagen del proyecto Django) – **OJO el comando incluye el punto final:**

sudo docker image build -t django-app .

Si todo salió bien, deberá ver un pantallazo similar al siguiente:

```
=> [internal] load .dockerignore                                0.0s
=> => transferring context: 34B                                  0.0s
=> [internal] load build definition from Dockerfile             0.0s
=> => transferring dockerfile: 247B                             0.0s
=> [internal] load metadata for docker.io/library/python:3.8    0.6s
=> [1/5] FROM docker.io/library/python:3.8@sha256:2e652376e0db17d97355ebfe7871736fb30d551ab164172157f  0.0s
=> [internal] load build context                                0.1s
=> => transferring context: 2.55MB                              0.1s
=> CACHED [2/5] WORKDIR /code                                    0.0s
=> CACHED [3/5] COPY requirements.txt /code                     0.0s
=> CACHED [4/5] RUN pip install -r requirements.txt             0.0s
=> [5/5] COPY . /code                                           0.2s
=> exporting to image                                           0.1s
=> => exporting layers                                           0.1s
=> => writing image sha256:1883c80a1b77a2a2386ffa86ba43478fbf6a9d279ef81a8755d8f29b54db4be1  0.0s
=> => naming to docker.io/library/django-app                   0.0s
nram_vel94@instance-docker:~/djangoDocker$
```

F3. Ahora estamos listos para desplegar un nuevo contenedor con la imagen creada anteriormente. A este nuevo contenedor lo llamaremos *django-docker* y lo desplegaremos en el puerto 8000. Para eso ejecutamos el siguiente comando:

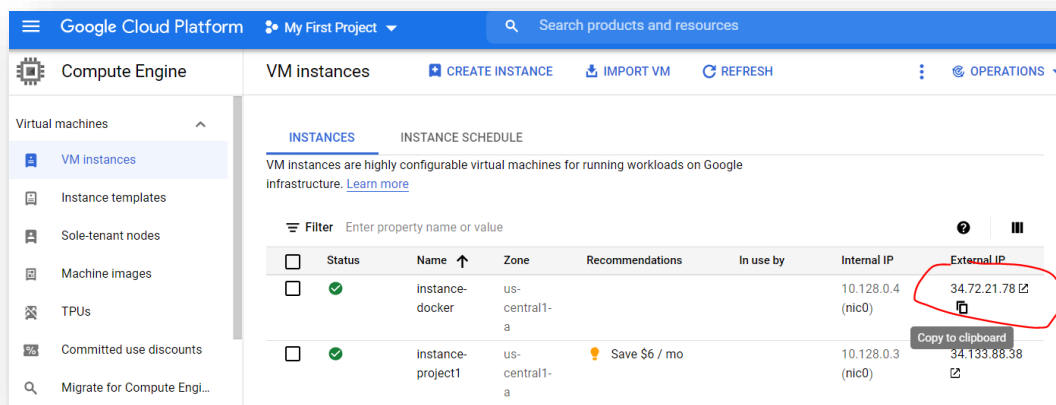
```
sudo docker container run -d --name django-docker -p 8000:8000 django-app
```



```
nram_vel94@instance-docker:~/djangoDocker$ sudo docker container run -d --name django-docker -p 80:80 django-app
9773c3a1e27cb5def4818c9979ccffbf34f3de102502f49ef118e822cdca40e2
nram_vel94@instance-docker:~/djangoDocker$
```

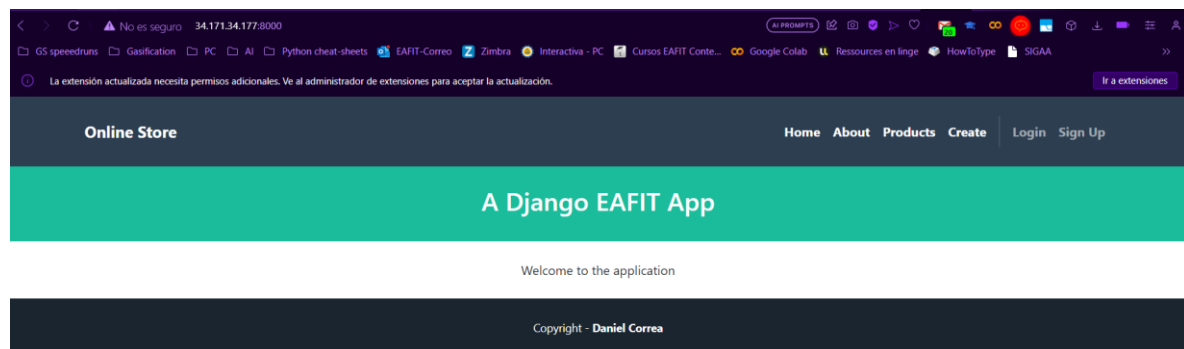
F4. Para lanzar la instancia, ejecute el siguiente comando:

```
sudo docker compose up
```

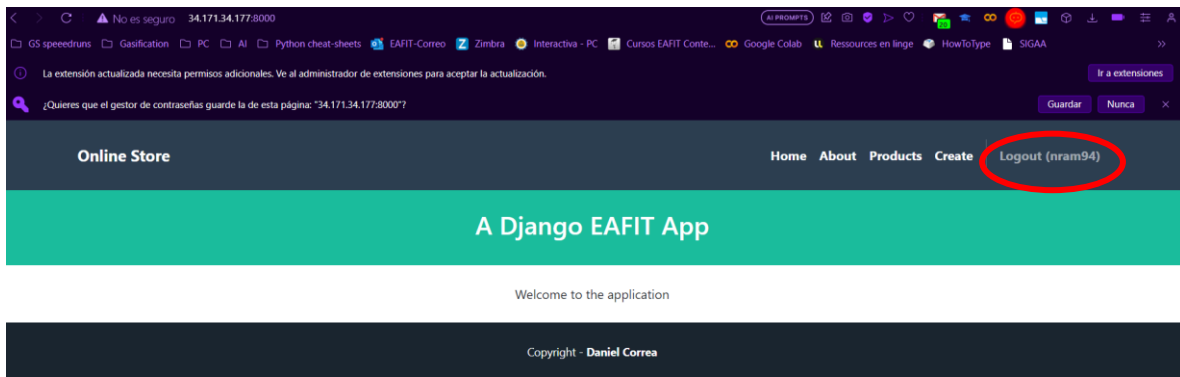
F5. Si todo salió bien, ahora podrá ingresar al navegador, a través de IP de su instancia GCP. Y podrá ver el proyecto Django corriendo (**recuerde usar solo http, con https no funciona**). Ejemplo: <http://EXTERNAL-IP:8000/>



Status	Name	Zone	Recommendations	In use by	Internal IP	External IP
✓	instance-docker	us-central1-a			10.128.0.4 (nic0)	34.72.21.78 
✓	instance-project1	us-central1-a	Save \$6 / mo		10.128.0.3 (nic0)	34.133.88.38 



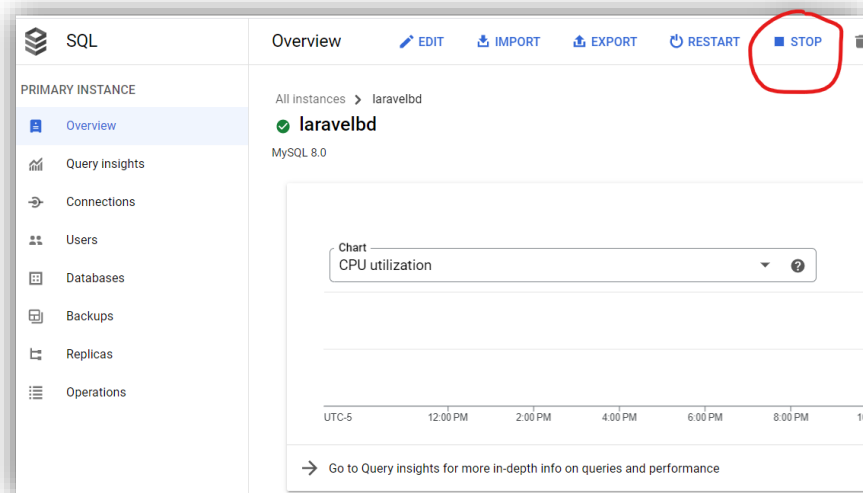
F5. Como paso final opcional, regístrese y verifique que le funcionó.



Pregunta

- ¿Qué ventaja tiene utilizar Docker para desplegar un proyecto en GCP versus la forma inicial en la que lo hicimos?

OJO “CLOUD SQL” CONSUME MUCHOS CRÉDITOS. LUEGO DE QUE TERMINE EL TALLER, PARE LA INSTANCIA DE CLOUD SQL PARA QUE NO GASTE CRÉDITOS



Comandos adicionales que le pueden ser útiles

```
sudo docker system prune -a
```

//borra todo lo que Docker haya creado, útil para liberar memoria

```
sudo docker container rm django-docker -f
```

//borra el contenedor django-docker

```
sudo docker exec -it django-docker bash
```

```
//se conecta al contenedor django-docker
```