

## Tutorial 05 to do in class – Remember to complete the task in Teams.

### Laravel

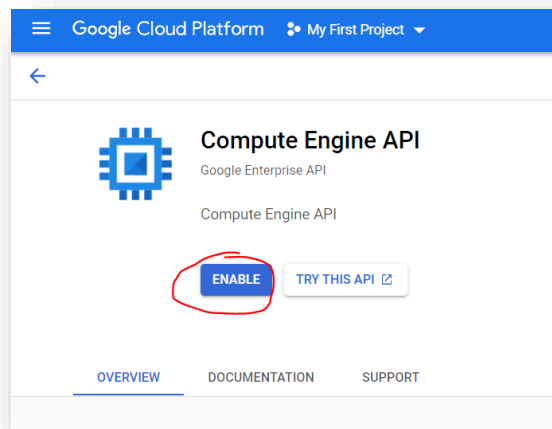
#### Antes de iniciar:

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

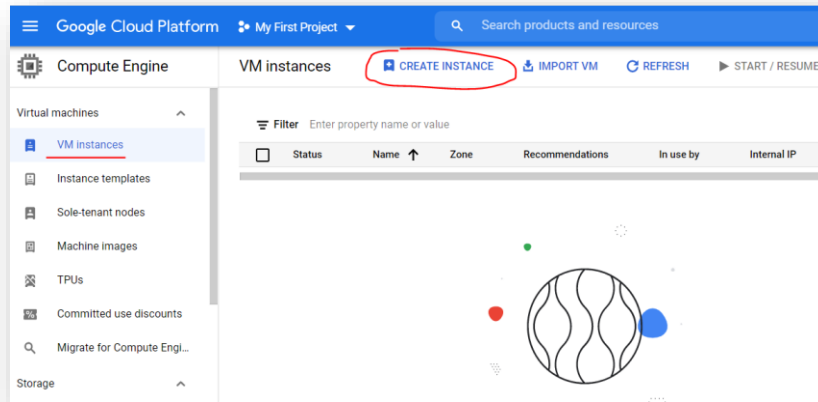
#### A. Creando la máquina virtual.

**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 la primera vez que se conecta, le tocará antes crear un proyecto, y ponga el nombre que desee) Si ya hizo esto antes, no debe repetirlo.



**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.

**Identify your VM**

Name \*  
instance-laravel-docker

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

Zone \*  
Any  
Google will choose a zone on your behalf, maximizing VM obtainability. Zone is permanent.

MANAGE TAGS AND LABELS

Availability policies

VM provisioning model  
Standard  
Choose "Spot" to get a discounted, preemptible VM. Otherwise, stick to "Standard". [Learn more](#)

VM PROVISIONING MODEL ADVANCED SETTINGS

VM basics  
instance-laravel-docker, us-central1

**Machine configuration**  
e2-micro

OS and storage  
Debian GNU/Linux 12 (bookworm)

Networking  
1 network interface

Observability

Security

Advanced

### Machine configuration

General purpose

Compute optimizedMemory optimizedStorage optimizedGPUs

Machine types for common workloads, optimized for cost and flexibility


Series	Description	vCPUs	Memory	Platform
<input type="radio"/> C4	Consistently high performance	2 - 192	4 - 1,488 GB	Intel Emerald Rapids
<input type="radio"/> N4	Flexible & cost-optimized	2 - 80	4 - 640 GB	Intel Emerald Rapids
<input type="radio"/> C3	Consistently high performance	4 - 192	8 - 1,536 GB	Intel Sapphire Rapids
<input checked="" type="radio"/> E2	Low cost, day-to-day computing	0.25 - 32	1 - 128 GB	Based on availability
<input type="radio"/> M2	Balanced price & performance	2 - 128	2 - 864 GB	Intel Cascade and Ice Lake
<input type="radio"/> N2D	Balanced price & performance	2 - 224	2 - 896 GB	AMD EPYC
<input type="radio"/> T2A	Scale-out workloads	1 - 48	4 - 192 GB	Ampere Altra Arm
<input type="radio"/> T2D	Scale-out workloads	1 - 60	4 - 240 GB	AMD EPYC Milan
<input type="radio"/> N1	Balanced price & performance	0.25 - 96	0.6 - 624 GB	Intel Skylake

**Machine type**  
Choose a machine type with preset amounts of vCPUs and memory that suit most workloads. Or, you can create a custom machine for your workload's particular needs. [Learn more](#)


PRESET

CUSTOM

e2-micro (2 vCPU, 1 core, 1 GB memory)

vCPU

0.25-2 vCPU (1 shared core)

Memory

1 GB

CPU platform

Automatic

**Monthly estimate**  
**\$7.11**  
That's about \$0.01 hourly

Pay for what you use, no upfront costs and per second billing

Item	Monthly estimate
2 vCPU + 1 GB memory	\$6.11
10 GB balanced persistent disk	\$1.00
<b>Total</b>	<b>\$7.11</b>

[Compute Engine pricing](#)  
[LESS](#)

Verificamos la versión del SO, (Debian 12).

Create an instance

CREATE VM FROM...

VM basics  
instance-laravel-docker, us-central1

**Machine configuration**  
e2-micro

**OS and storage**  
Debian GNU/Linux 12 (bookworm)

Networking  
1 network interface

Observability

Security

Advanced

### Operating system and storage

Name	instance-laravel-docker
Type	New balanced persistent disk
Size	10 GB
Snapshot schedule	No schedule selected
License type	Free
Image	Debian GNU/Linux 12 (bookworm)

CHANGE

### Additional storage and VM backups

+ ADD NEW DISK

+ ATTACH EXISTING DISK

+ ADD LOCAL SSD

**Backup plan** **PREVIEW**

Secure your backups against deletion through backup vault storage and enable centralized backup management across projects. Managed by Backup and DR Service, a separate service from Compute Engine with independent certifications and accreditation. [Learn more](#)

Backup planSELECT A PLAN

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

**Networking**

- VM basics  
instance-laravel-docker, us-central1
- Machine configuration  
e2-micro
- OS and storage  
Debian GNU/Linux 12 (bookworm)
- **Networking**  
2 firewall rules, 1 network interface
- Observability
- Security
- Advanced

### Firewall

Add tags and firewall rules to allow specific network traffic from the Internet

- ☒ Allow HTTP traffic
- ☒ Allow HTTPS traffic
- ☐ Allow Load Balancer Health checks

Network tags

Hostname

Set a custom hostname for this instance or leave it default. Choice is permanent

IP forwarding

☐ Enable

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

Google Cloud

2024-1

Search (/) for

## Compute Engine

Virtual machines

- **VM instances**
- Instance templates
- Sole-tenant nodes
- Machine images
- TPUs
- Committed use discounts
- Reservations

### VM instances

**CREATE INSTANCE** **IMPORT VM**

**INSTANCES** **OBSERVABILITY** **INSTANCE SCHEDULES**

#### VM instances

Filter Enter property name or value

Status	Name	Zone	Recommendation
<input checked="" type="checkbox"/>	<a href="#">instance-laravel-docker</a>	us-central1-a	

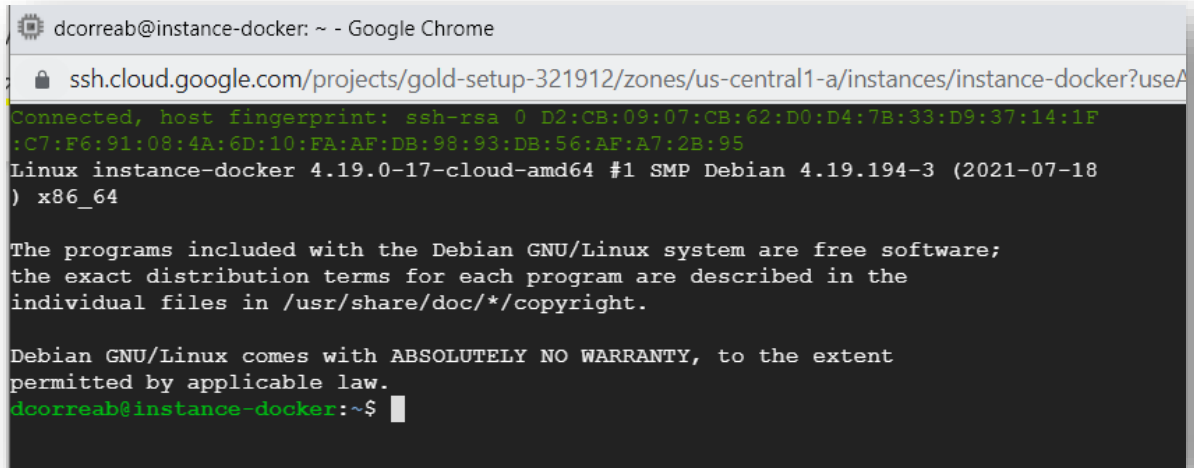
#### Related actions

- **Explore Backup and DR** **NEW**  
Back up your VMs and set up disaster recovery
- **View billing report**  
View and manage your Com billing

## 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 -
```

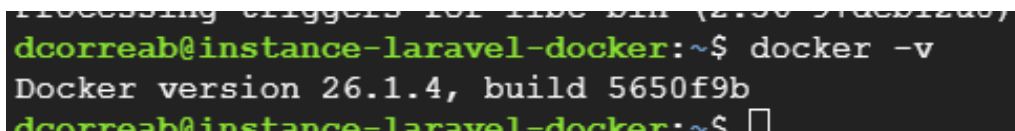
```
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/debian buster stable"
```

```
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:

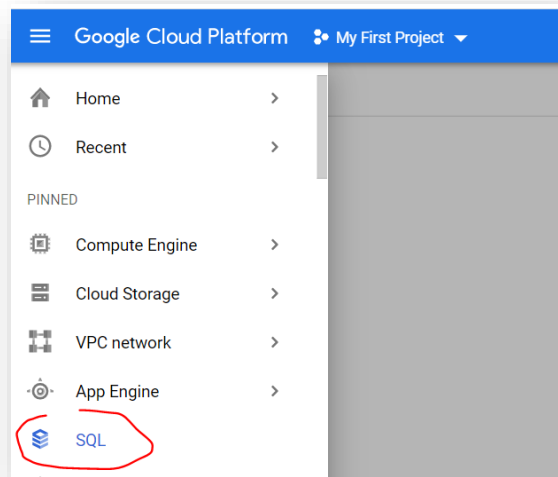
```
docker -v
```



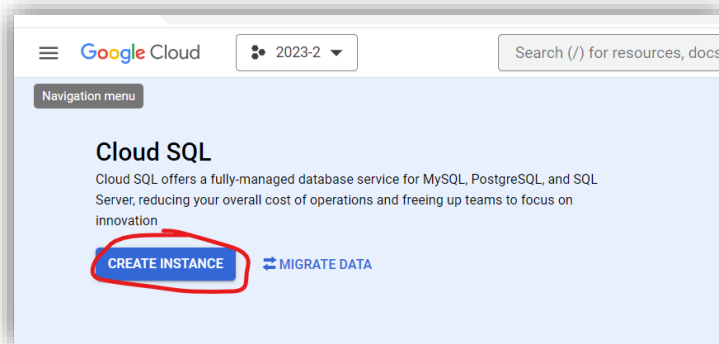
```
dcorreab@instance-laravel-docker:~$ docker -v
Docker version 26.1.4, build 5650f9b
dcorreab@instance-laravel-docker:~$
```

## 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.

Luego seleccione “Enterprise” -> y seleccione “Sandbox”.

### Choose a Cloud SQL edition

A Cloud SQL edition determines foundational characteristics of your instance and cannot be changed later. Choose based on your price and performance needs. [Learn more](#)

☐ Enterprise Plus

- 99.99% availability SLA for eligible instances
- High-performance machines, up to 128 vCPUs
- Up to 35 days point-in-time recovery
- Data cache (optional)

☒ Enterprise

- 99.95% availability SLA for eligible instances
- General purpose machines, up to 96 vCPUs
- Up to 7 days point-in-time recovery

Choose a preset for this edition. Presets can be customized later as needed.

Sandbox

**Nota: si no le aparece Sandbox use Development**

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

### Instance info

Database version \*  
MySQL 8.0

[SHOW MINOR VERSIONS](#)

Instance ID \*  
laravelbdgcp

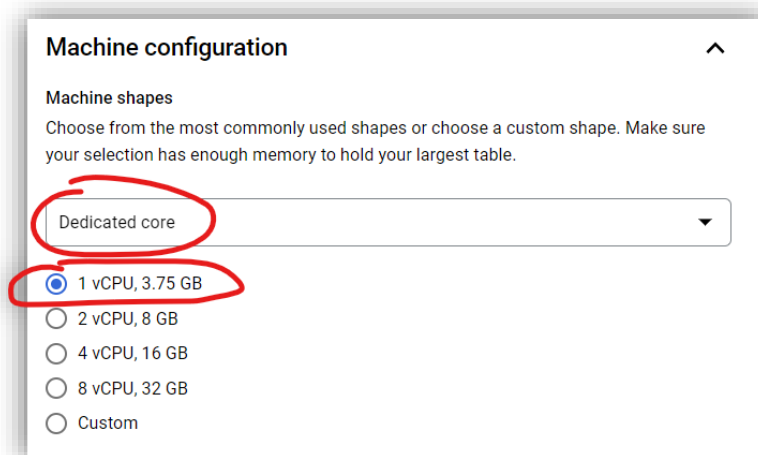
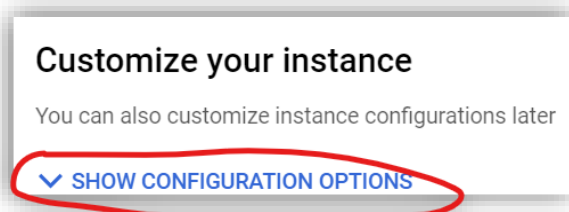
Use lowercase letters, numbers, and hyphens. Start with a letter.

Password \*  
.....

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

☐ No password

Luego de click en “SHOW CONFIGURATION OPTIONS” y configure su base de datos basado en los siguientes pantallazos.





**Storage**

**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

☐ 250 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. Make sure your storage can support the automated backups and days of logs you're retaining. [Learn more](#)

☐ **Automated daily 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".

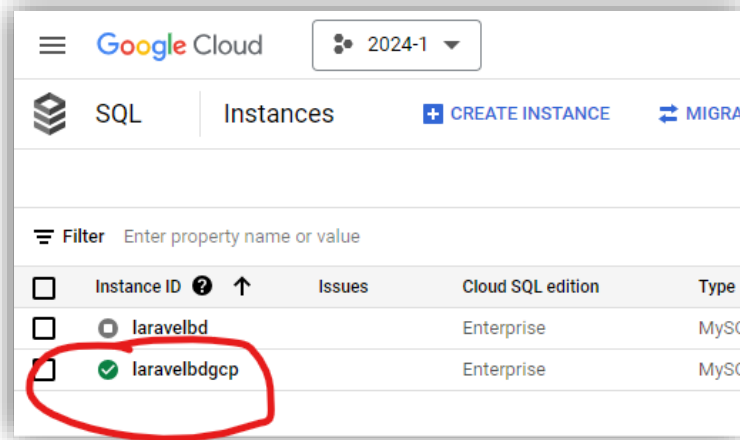
Region	us-central1 (Iowa)
DB Version	MySQL 8.0
vCPUs	1 vCPU
RAM	3.75 GB
Data Cache	Disabled
Storage	10 GB
Connections	Public IP
Backup	Manual
Availability	Single zone
Point-in-time recovery	Disabled
Network throughput (MB/s) ?	250 of 250
Disk throughput (MB/s) ?	Read: 4.8 of 200.0 Write: 4.8 of 200.0
IOPS ?	Read: 300 of 12,000 Write: 300 of 10,000

### Pricing estimate (without discounts)

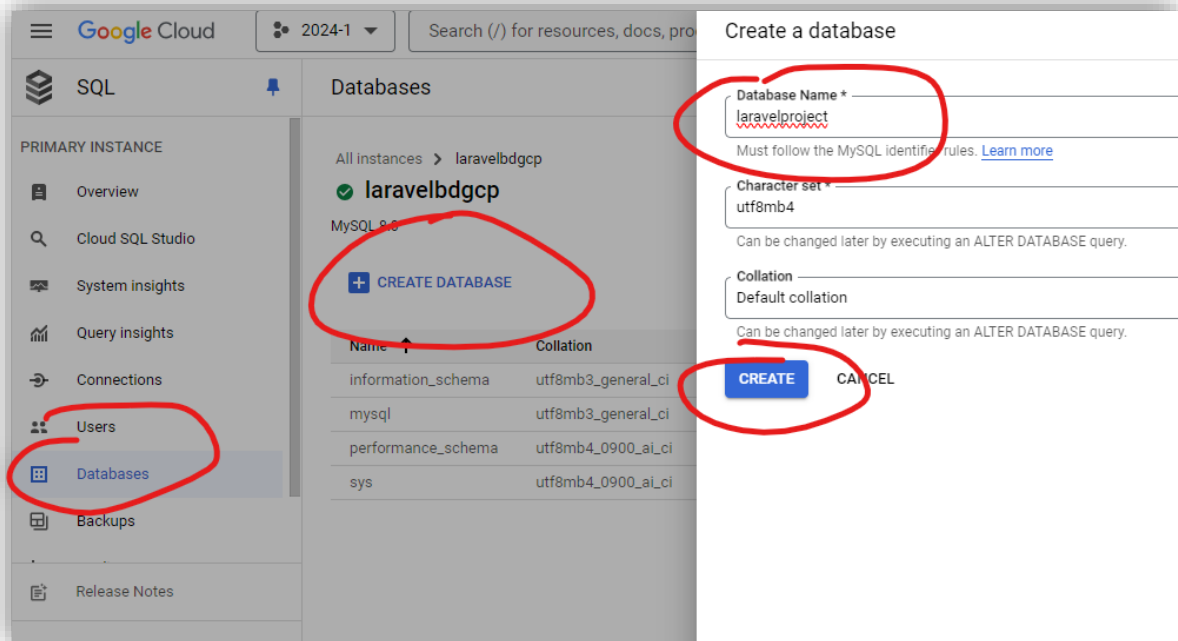
These items represent Cloud SQL compute, memory and storage resources only, and reflect how you configured your instance so far. Discounts not included in estimate. [Learn more](#)

Item	Hourly cost (estimate)
1 vCPU (\$0.041 per vCPU/hour)	\$0.04
3.75 GiB RAM (\$0.007 per GiB/hour)	\$0.03
10 GiB SSD (\$0.17 per GiB/month)	\$0.002
Total without usage discounts	\$0.07

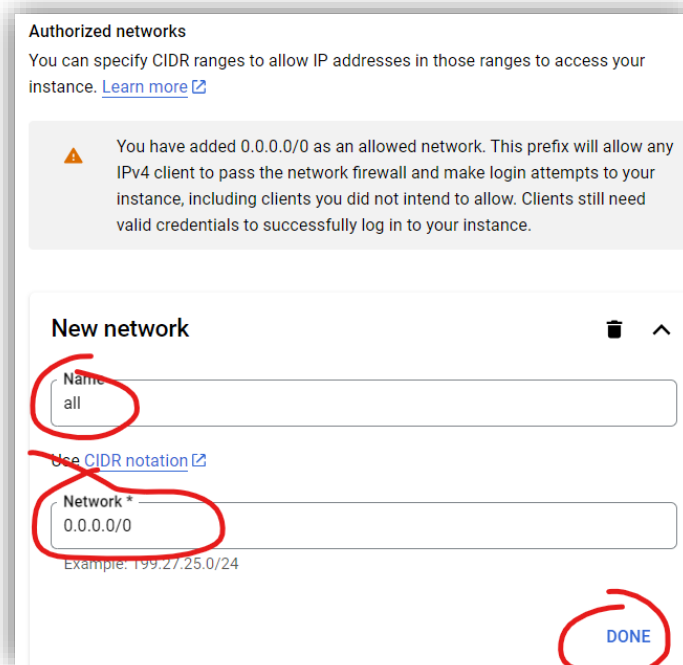
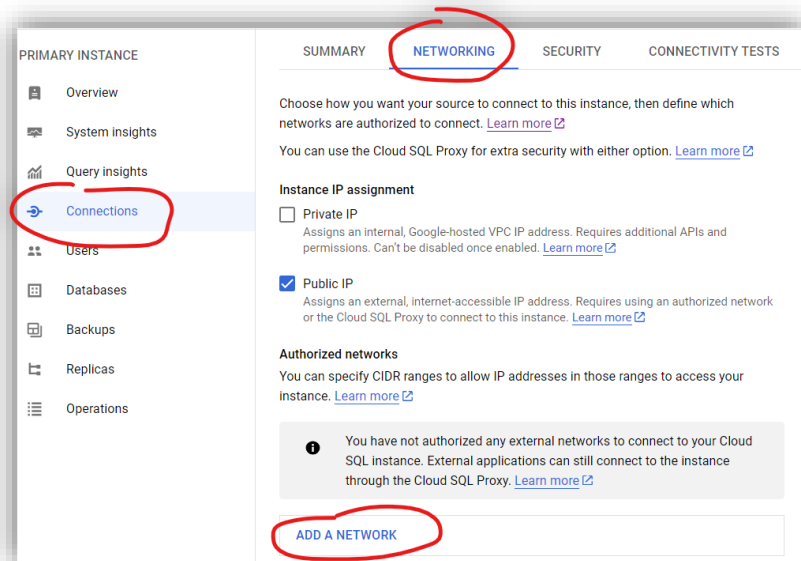
**D6.** Una vez creada la base de datos (toma hasta 5 minutos), vaya a su instancia de base datos.



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



**D8.** 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”.



all (0.0.0.0/0)

(Not saved) ▼

[ADD A NETWORK](#)

#### Google Cloud services authorization

☐ Enable private path

Allows other Google Cloud services like BigQuery to access data and make queries over Private IP. [Learn more](#)

#### 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

## E. Configurar proyecto Laravel 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 Laravel en nuestra instancia. A continuación, utilizaré un proyecto de ejemplo. Luego lo puede reemplazar por su propio proyecto.

```
git clone https://github.com/danielgara/laravel11-docker.git
```

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

```
cd laravel11-docker
```

```

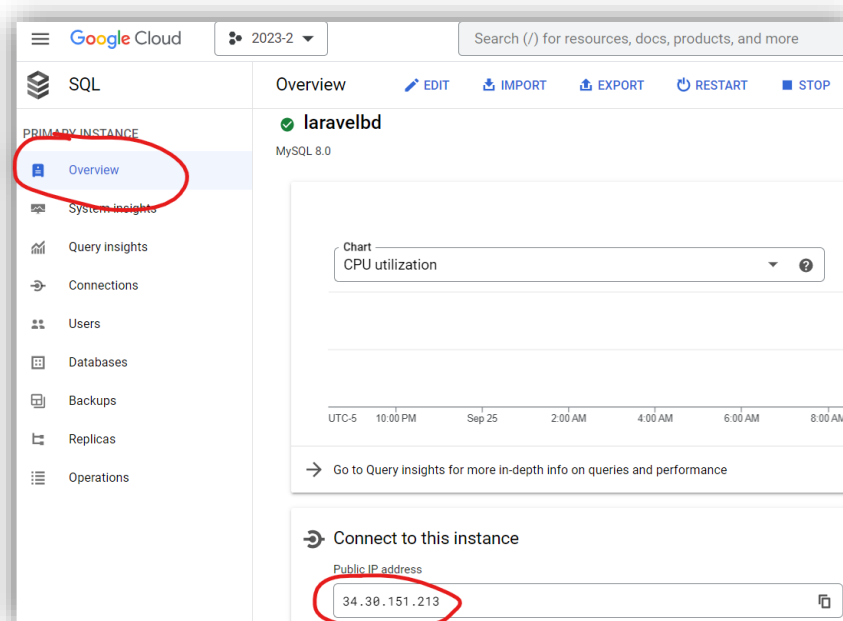
Last login: Mon Sep 30 20:44:21 2024 from 35.235.241.17
dcorreab@instance-laravel-docker:~$ git clone https://github.com/danielgara/laravel11-docker.git
Cloning into 'laravel11-docker'...
remote: Enumerating objects: 159, done.
remote: Counting objects: 100% (159/159), done.
remote: Compressing objects: 100% (116/116), done.
remote: Total 159 (delta 23), reused 159 (delta 23), pack-reused 0 (from 0)
Receiving objects: 100% (159/159), 89.37 KiB | 2.55 MiB/s, done.
Resolving deltas: 100% (23/23), done.
dcorreab@instance-laravel-docker:~$ cd laravel11-docker/
dcorreab@instance-laravel-docker:~/laravel11-docker$ 

```

**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`

```

APP_NAME=Laravel
APP_ENV=local
APP_KEY=
APP_DEBUG=true
APP_URL=http://localhost

LOG_CHANNEL=stack
LOG_DEPRECATIONS_CHANNEL=null
LOG_LEVEL=debug

DB_CONNECTION=mysql
DB_HOST=34.30.151.213
DB_PORT=3306
DB_DATABASE=laravelproject
DB_USERNAME=root
DB_PASSWORD=root

```

**Nota:** recuerde colocar su password de la base de datos cuando la creo.

## F. Correr proyecto Laravel 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/danielgara/laravel11-docker/blob/main/Dockerfile>).

```
dcorreab@instance-laravel-docker:~/laravel11-docker$ ls
Dockerfile  README.md  app  artisan  bootstrap  composer.json  composer.lock  config
database  lang  package.json  phpunit.xml  public  resources  routes  storage  tes
ts  vite.config.js
```

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

*sudo docker image build -t laravel-app .*

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

```
=> [11/13] RUN chmod -R 777 storage 3.3s
=> [12/13] RUN a2enmod rewrite 4.0s
=> [13/13] RUN service apache2 restart 5.5s
=> exporting to image 16.5s
=> => exporting layers 16.3s
=> => writing image sha256:775eccc8059e8ec7e61e91fd7285d9ce9939ee42f57261198 0.0s
=> => naming to docker.io/library/laravel-app 0.2s
dcorreab@instance-laravel-docker:~/laravel11-docker$
```

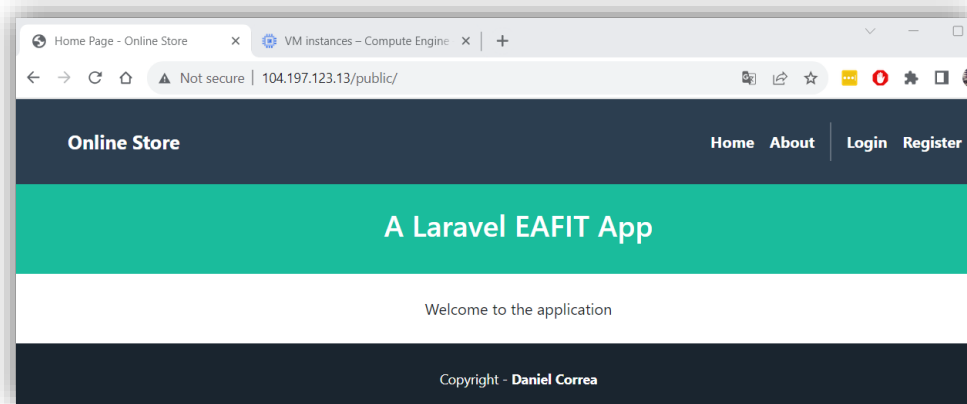
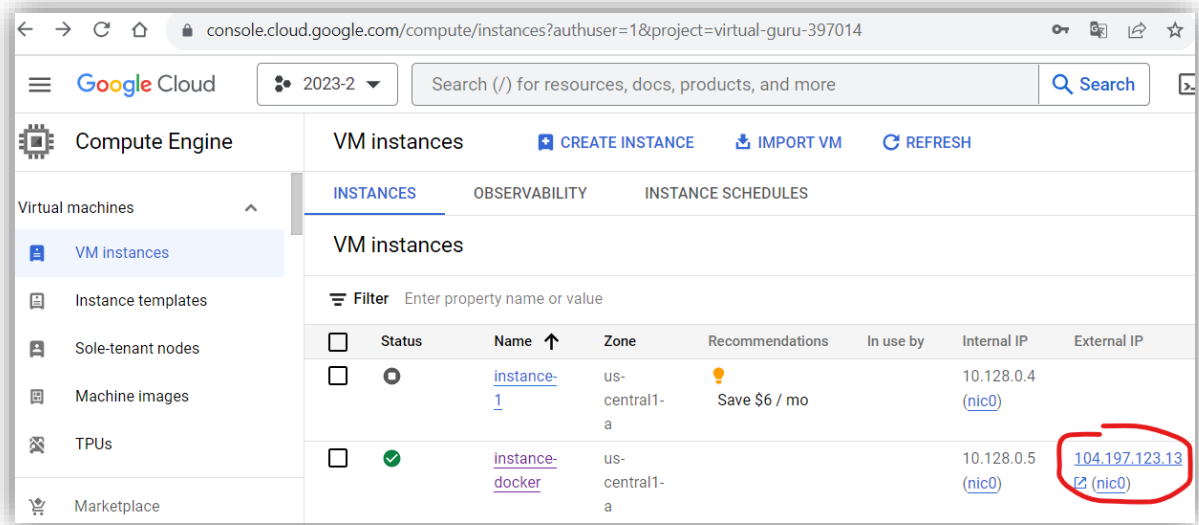
**F3.** Ahora estamos listos para desplegar un nuevo contenedor con la imagen creada anteriormente. A este nuevo contenedor lo llamaremos *laraveldocker* y lo desplegaremos en el puerto 80. Para eso ejecutamos el siguiente comando:

*sudo docker container run -d --name laravel-docker -p 80:80 laravel-app*

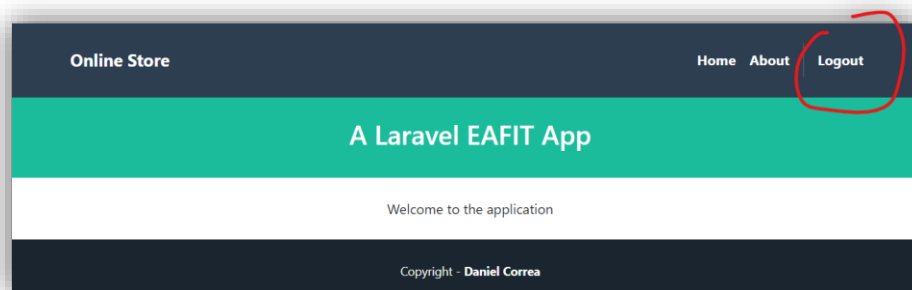
```
dcorreab@instance-laravel-docker:~/laravel11-docker$ sudo docker container run -d --
name laravel-docker -p 80:80 laravel-app
6d21cf3c1e313681d321289fc383e844feb1a8db35083f145278ac671b26b243
```

**F4.** Si todo salió bien, ahora podrá ingresar al navegador, a través de IP de su instancia GCP. Y podrá ver el proyecto Laravel corriendo (**recuerde usar solo http, con https no funciona**). Ejemplo: <http://104.197.123.13/public/>





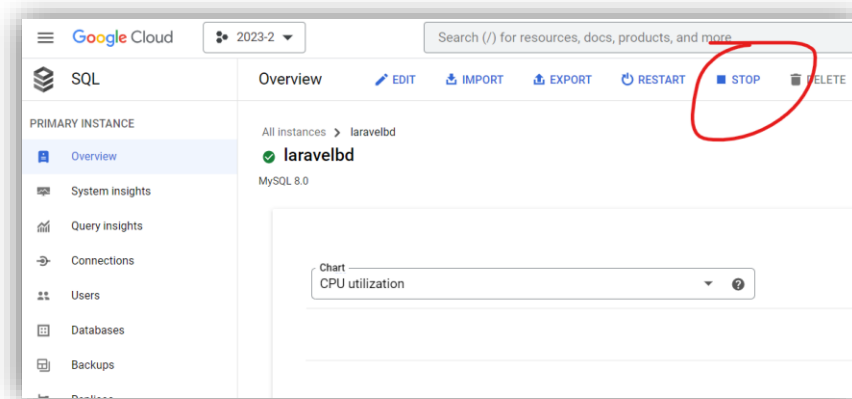
F5. Como paso final opcional, regístrese y verifique que le funcionó.



## 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 laravel-docker -f
```

//borra el contenedor laravel-docker

```
sudo docker exec -it laravel-docker bash
```

//se conecta al contenedor laravel-docker