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

Antes de iniciar:

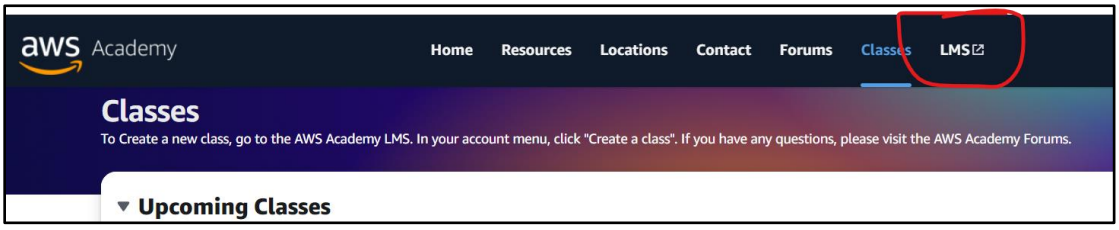
- Terminar los tutoriales anteriores
- Este taller muestra un ejemplo de despliegue de una aplicación Laravel con Docker en AWS.

A) Acceso y Creación de la Instancia EC2

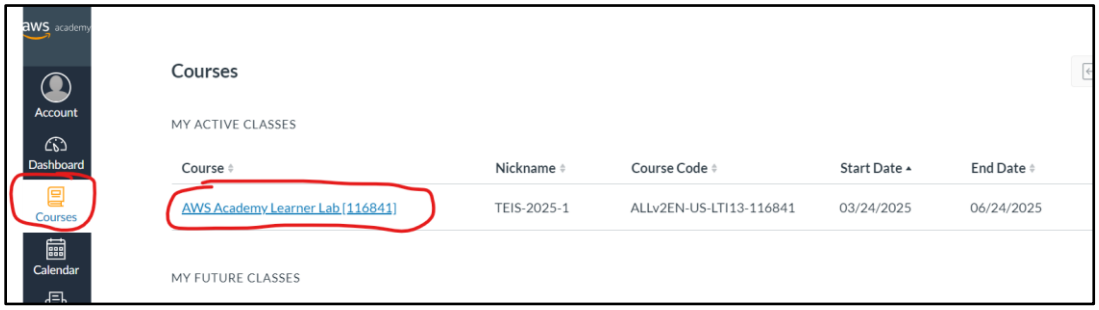
Como paso previo deberá solicitarle al docente el acceso a AWS Academy.

Paso 1: Acceda a https://www.awsacademy.com/ con su cuenta de estudiante.

Paso 2: Ingrese a la sección de LMS.

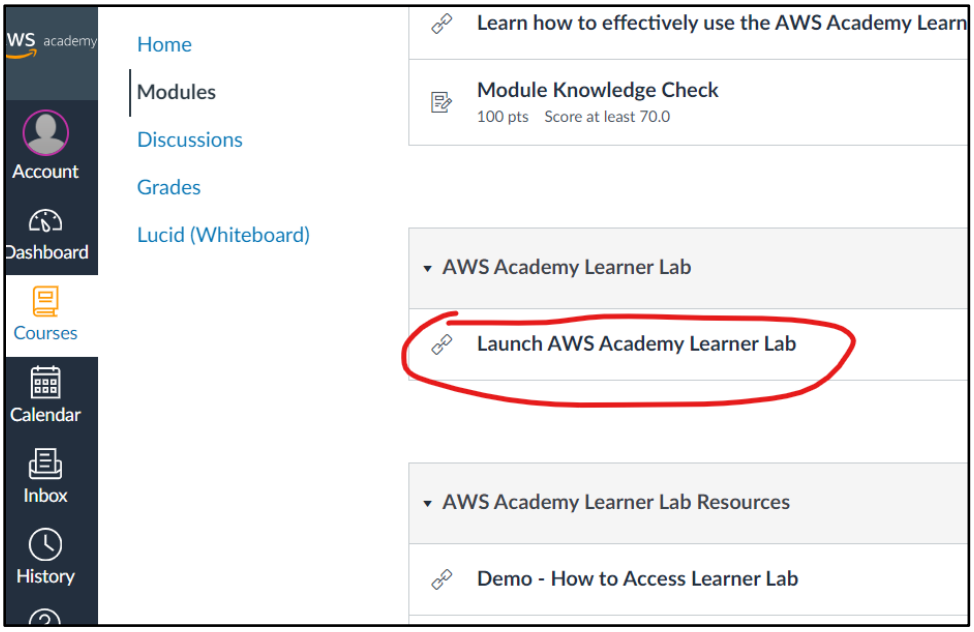


Paso 3: Vaya a la sección de cursos, busque el curso 116841 con el Nickname “TEIS-2025-1”, y de click en ese curso.

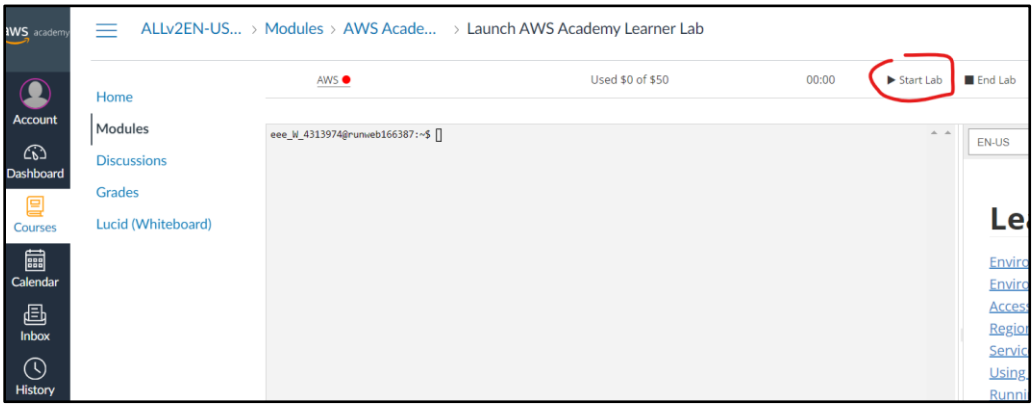


Paso 4: Vaya a la sección de “Modules” (algunas veces deberá aceptar unos términos y condiciones).

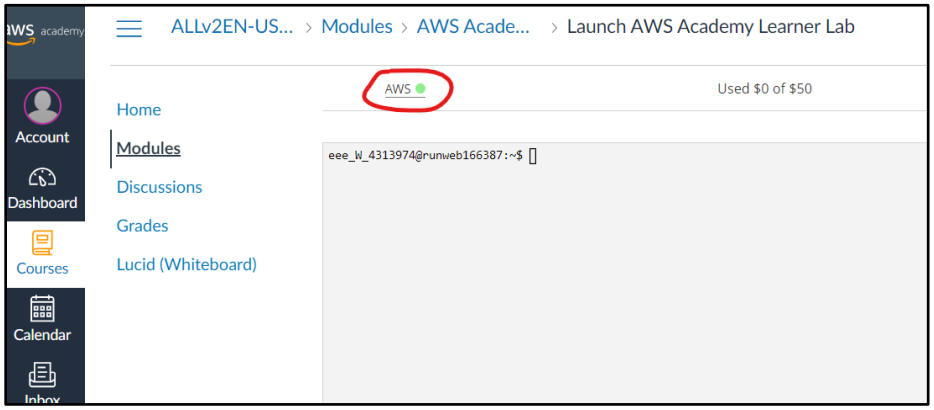
Paso 5: Busque y de click en “Launch AWS Academy Learner Lab”



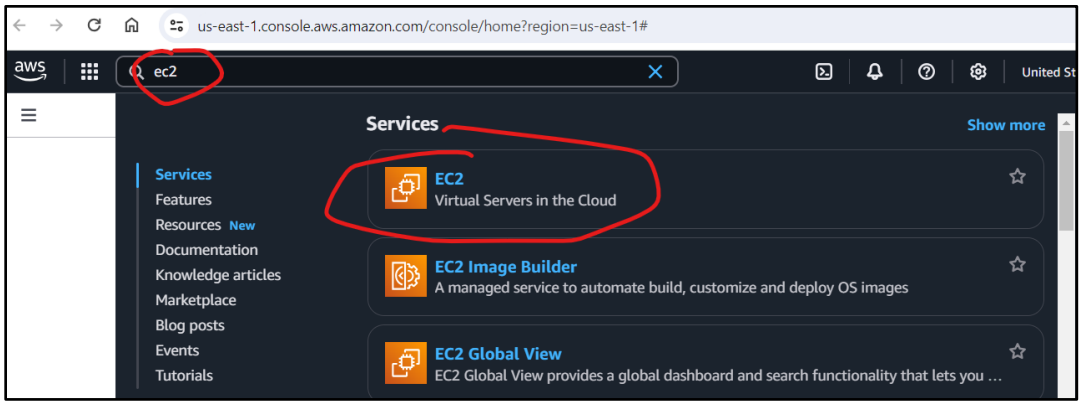
Paso 6: Inicie el laboratorio, dando click en “Start Lab”



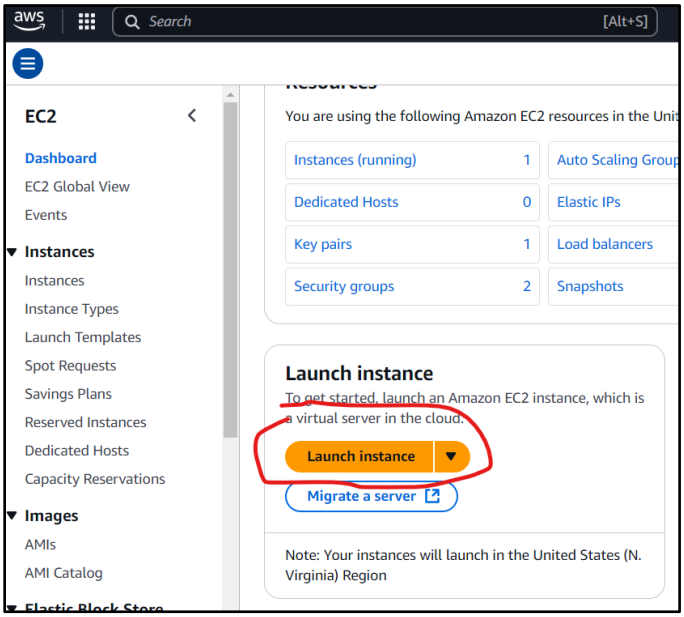
Paso 7: Espere unos minutos a que el bombillo de AWS se ponga verde. Y luego de click sobre AWS.



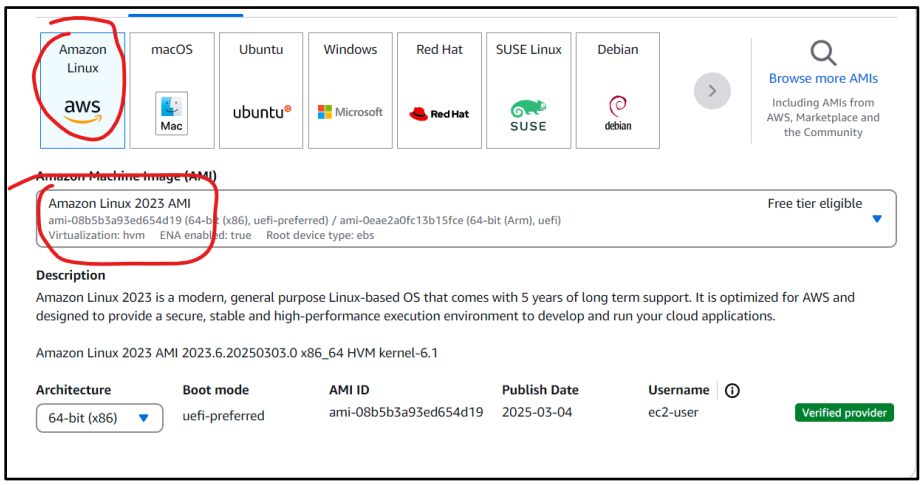
Paso 8: En la nueva ventana que se abre, busque “ec2”. Y luego de click sobre EC2.



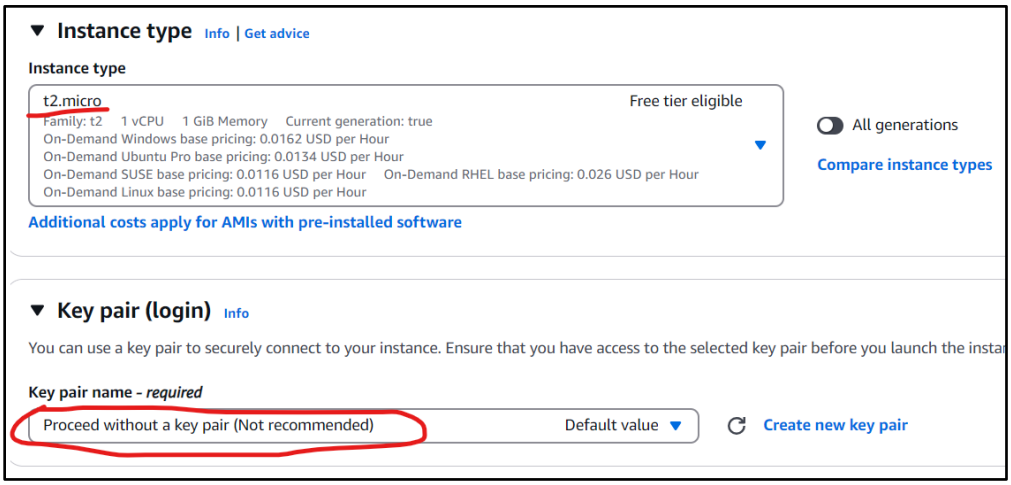
Paso 9: Luego de click en “Launch Instance”.



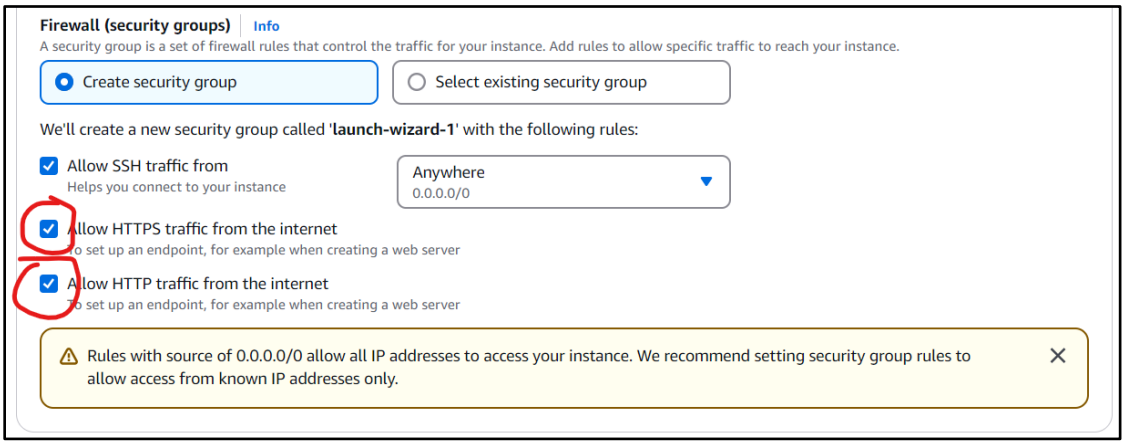
Paso 10: Verifique que está seleccionado “Amazon Linux” y “Amazon Linux 2023 AMI”



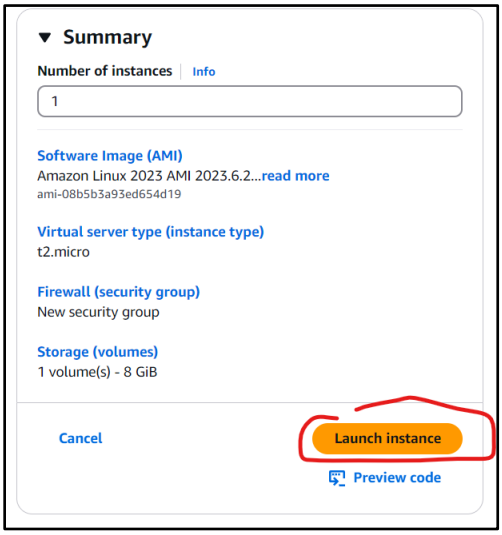
Paso 11: Verifique que está seleccionado “t2.micro” y en key pair name seleccione “Proceed wihtout a key pair (Not recommended)”.



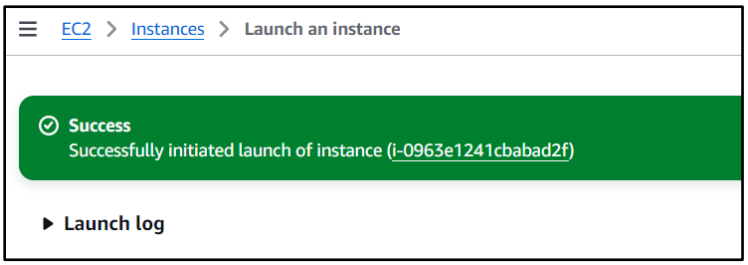
Paso 11: En “Network Settings”, active “Allow HTTPS” y “Allow HTTP”.



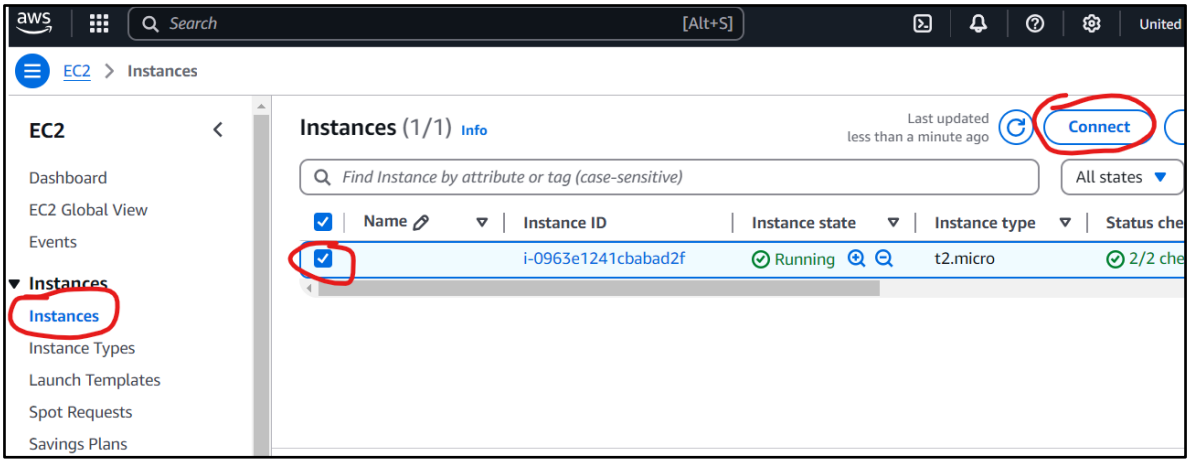
Paso 12: Finalmente de click en “Launch Instance”.



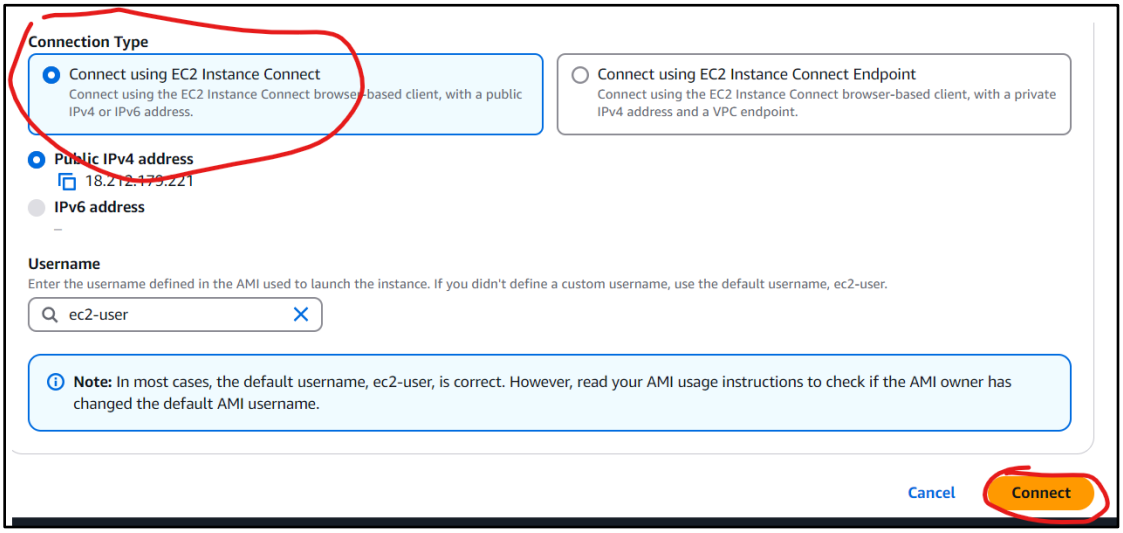
Paso 13: Espere a que la instancia se cree.



Paso 14: Vaya a la sección de “Instances”, luego seleccione la instancia que acabó de crear, y luego de click en “Connect”.



Paso 15: Verifique que “Connect using EC2 Instance Connect” está seleccionado, y de click en “Connect”.



B) Instalación de Docker y de la Base de Datos

Como paso previo deberá completar la parte A de este tutorial.

Instalar librerías para poder correr Laravel con Docker en nuestra máquina virtual sobre la consola abierta previamente y el sistema de base de datos

Paso 1: Corra el siguiente comando desde la consola de AWS (estamos en una maquina Amazon Linux) para actualizar el sistema, instalar y activar Docker, y verificar su instalación.

sudo dnf update -y

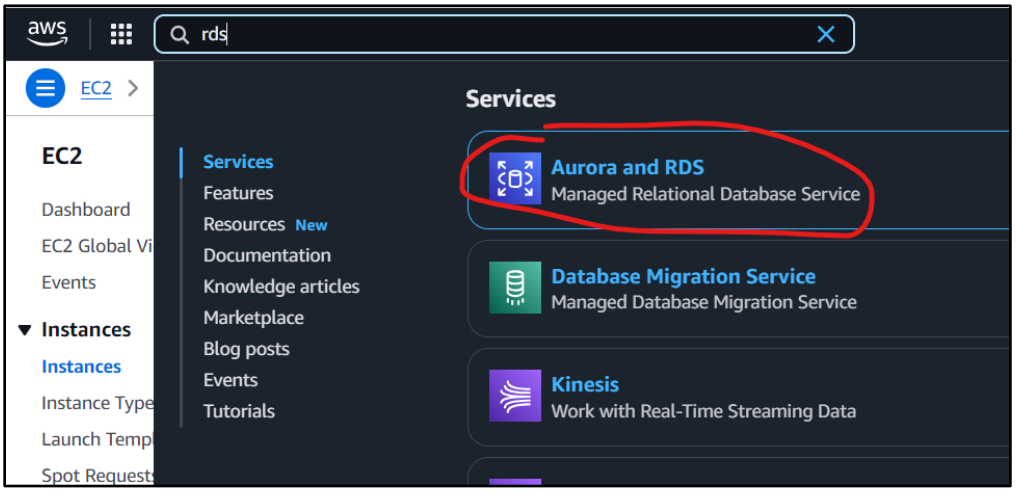
sudo dnf install -y docker

sudo systemctl enable --now docker

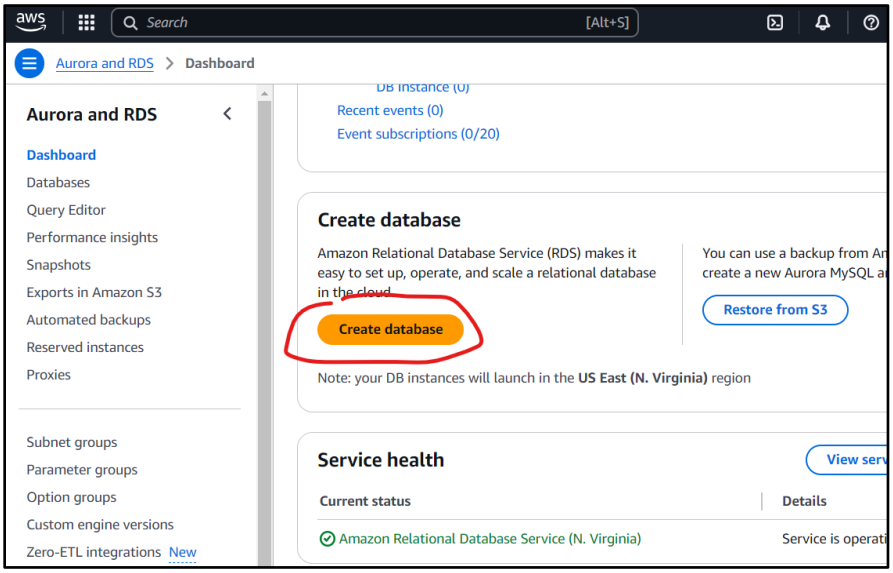
docker --version

```
[ec2-user@ip-172-31-21-75 ~]$ docker --version
Docker version 25.0.8, build 0bab007
```

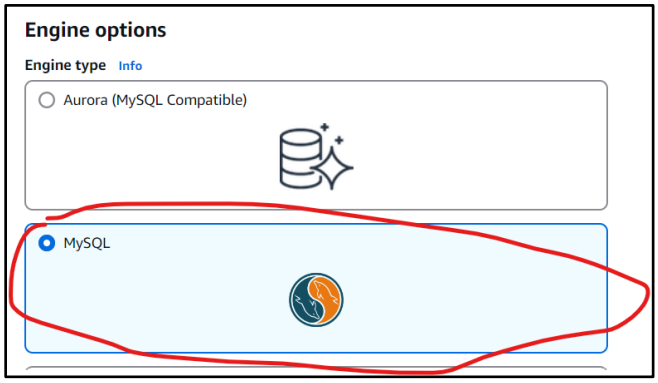
Paso 2: Vuelva al navegador, al sitio de AWS, y busque “Aurora and RDS”.

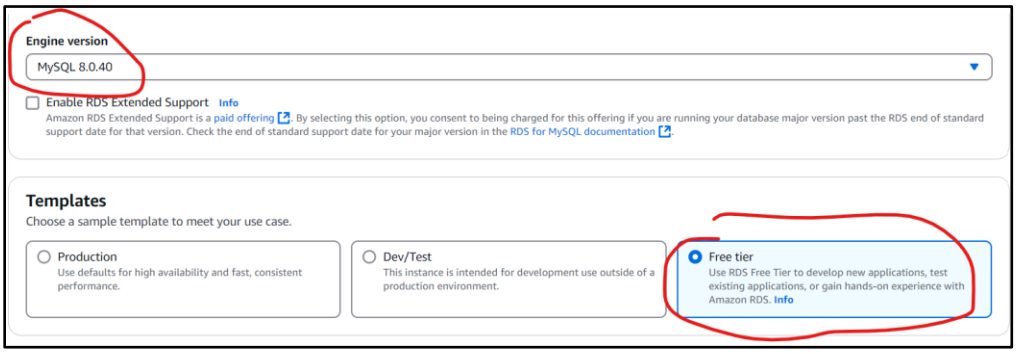


Paso 3: Busque y de click en la opción “Create database”.

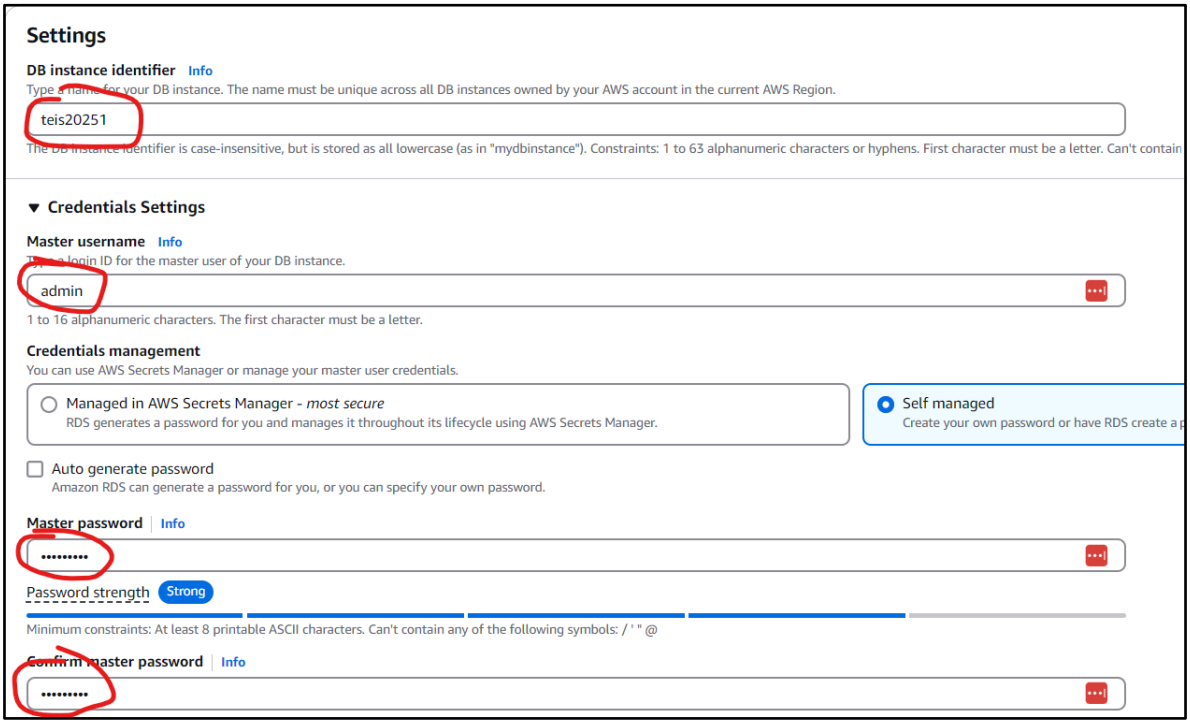


Paso 4: Seleccione “MySQL”, verifique que tiene activa la versión “MySQL 8.0”, y seleccione “Free tier”.



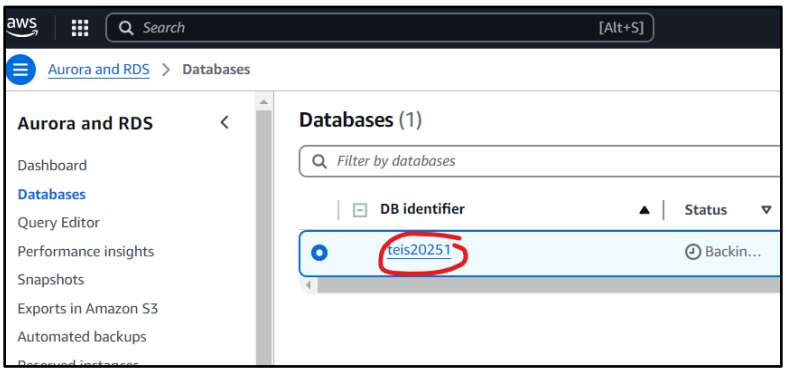


Paso 5: Como identificador coloque “teis20251”, username lo puede dejar en “admin”, y masterpassword trate de utilizar uno seguro que ya nos han hackeado varias veces.

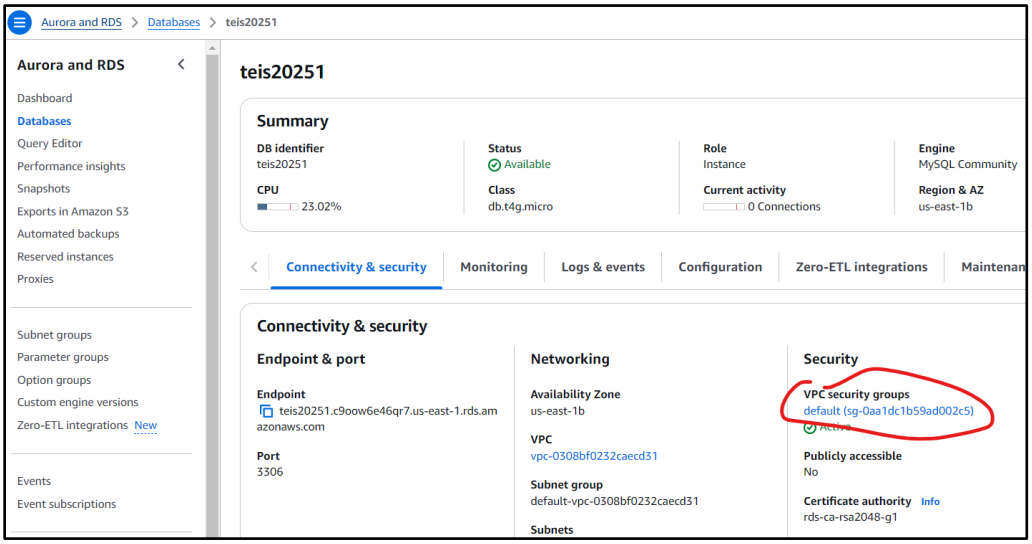


Paso 6: deje el resto de información como aparece y de click en “Create database”.

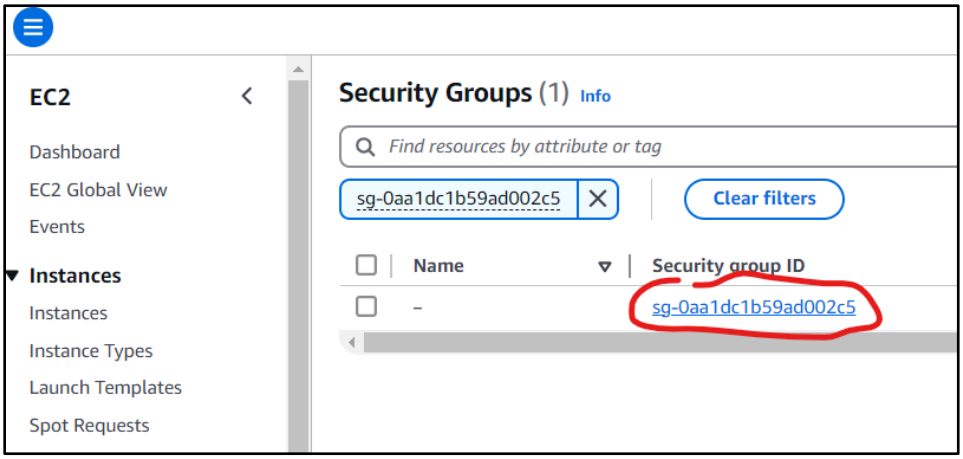
Paso 7: espere unos minutos mientras se crea la base de datos, una vez creada de click en la base de datos.



Paso 8: Busque el “VPC security groups” y de click sobre este.

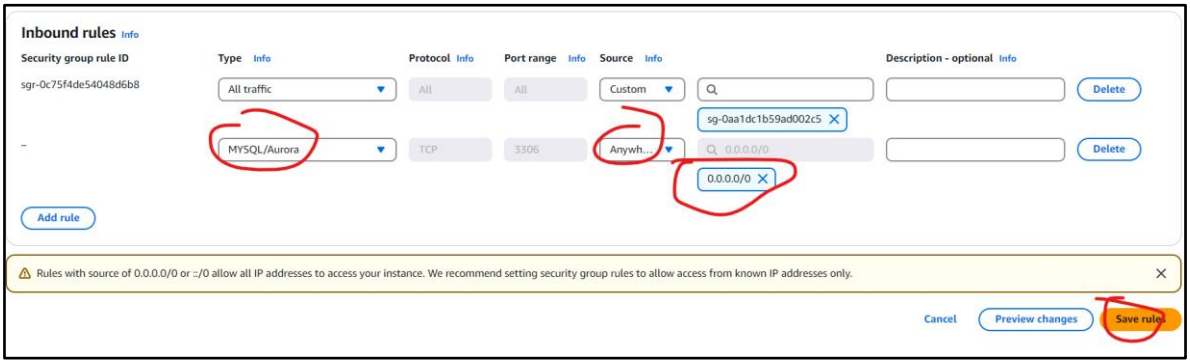


Paso 9: De click sobre el ID del security group.



Paso 10: De click en “Edit inbound rules”.

Paso 11: Añada la siguiente regla que nos permitirá conectarnos a la base de datos desde la instancia EC2.



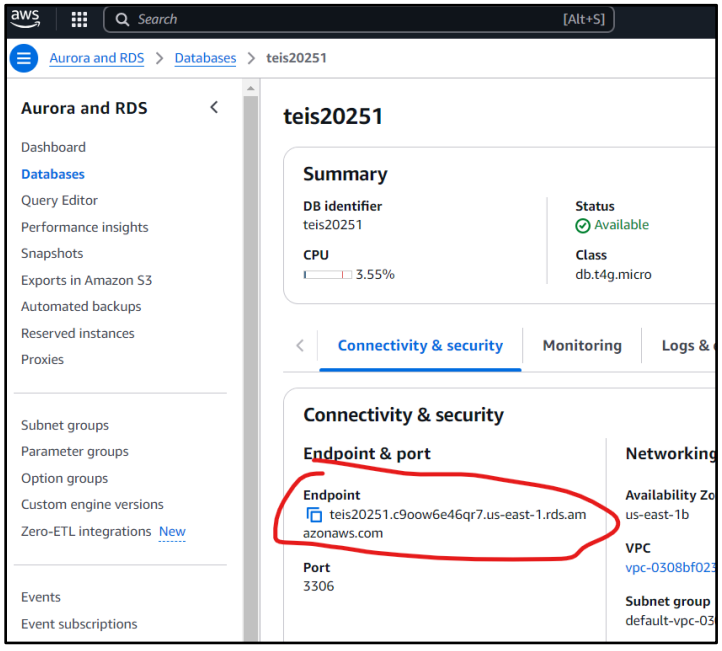
Paso 12: Vuelva a la consola de la instancia, y ahora instale MySQL (realmente es MariaDB) con el siguiente comando.

sudo dnf install -y mariadb105

Paso 13: Verifique que se puede conectar a la instancia de MySQL y cree una base de datos. Para poder conectarse con la base de datos (cambie los datos por su propio *endpoint* y su *usuario*). Más adelante puede ver una imagen de ejemplo de cómo ejecutar ese comando.

mysql -h endpoint -P 3306 -u admin -p

El endpoint lo encontrará desde la instancia de la base de datos en AWS.



Luego le pedirá ingresar el password y presionar Enter. Una vez conectado ejecute el siguiente comando para crear la base de datos.

CREATE DATABASE djangodocker;

```
[ec2-user@ip-172-31-86-159 ~]$ mysql -h teis20251.cpsc8aigcnqs.us-east-1.rds.a
mazonaws.com -P 3306 -u admin -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 29
Server version: 8.0.40 Source distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]> CREATE DATABASE djangodocker;
```

Por último, ejecute el siguiente comando para salir de la conexión de la base de datos.

exit

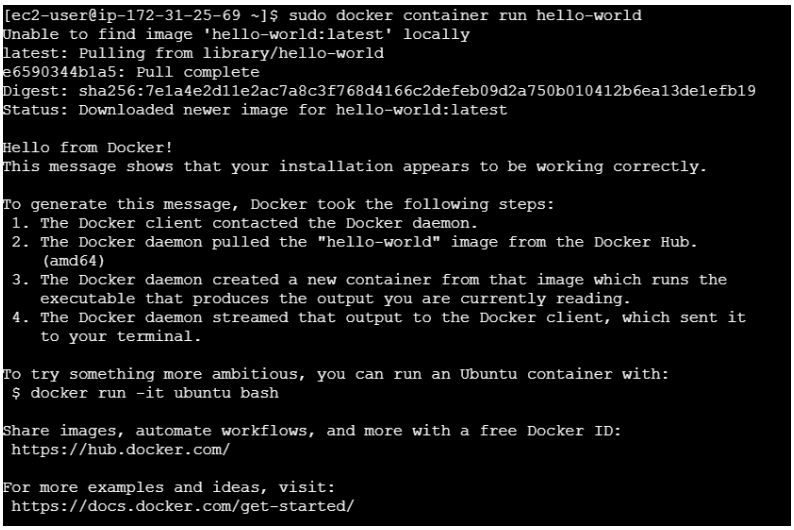
```
MySQL [(none)]> CREATE DATABASE djangodocker;
Query OK, 1 row affected (0.011 sec)

MySQL [(none)]> exit
Bye
[ec2-user@ip-172-31-86-159 ~]$
```

C) Descargando y corriendo el proyecto con Docker

Como paso previo deberá completar la parte B de este tutorial.

Paso 1: Corra el siguiente comando desde la consola de AWS para correr su primer contenedor Docker.

sudo docker container run hello-world



Paso 2: Corra el siguiente comando para instalar git.

sudo dnf install git -y

Paso 3: Descargue el código del proyecto (**por el momento utilice los siguientes comandos**, cuando termines el tutorial podrías cambiar el código para que funcione con el repositorio de tu proyecto / añadiendo el Dockerfile, en tal caso, deberás cambiar los valores que aparecen en rojo).

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

cd djangoDocker

Paso 4: Clone el .env.example.

cp .env.example .env

Paso 5: Modifique el archivo .env (utilice el comando *nano .env*). A continuación, le mostraré la configuración del proyecto de prueba (reemplace los valores DB_HOST, DB_DATABASE, DB_USERNAME y DB_PASSWORD con los que usted creó anteriormente). Una vez haga los cambios en el servidor, utilice “ctrl+x” -> luego “y” -> luego “enter” para guardar.

nano .env

```
  GNU nano 8.3                                .env
DEBUG=1
DJANGO_ALLOWED_HOSTS=localhost 127.0.0.1 0.0.0.0 [::1]
DATABASE=mysql
SQL_ENGINE=django.db.backends.mysql
SQL_PORT=3306
SQL_DATABASE=djangodocker                  # name of the database
SQL_ROOT_PASSWORD=password# CHANGE THIS
SQL_USER=admin                    # do not use 'root'
SQL_PASSWORD=password      # CHANGE THIS
SQL_HOST=teis20251.cpsc8aigcnqs.us-east-1.rds.amazonaws.com
```

Paso 6: Cree la imagen Docker.

sudo docker image build -t django-app .

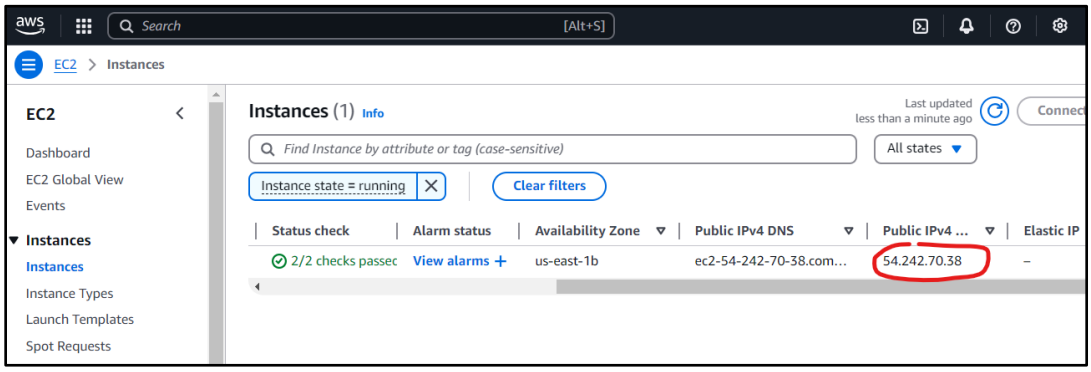
```
 => => sha256:a173f2aee8e962ea19db1e418ae84a0c9f71480 64.39MB / 64.39MB   5.2s
 => => sha256:7aa279fb41dad2962d3c915aa6f6615134baa412a 2.32kB / 2.32kB   0.0s
 => => sha256:3ea6eaad4f175bd42f39dae10098b1820ee522628 5.97kB / 5.97kB   0.0s
 => => sha256:cdd62bf39133c498a16f7a7b1b6555ba43d02b2 49.56MB / 49.56MB   1.5s
 => => sha256:d411270700143fa2683cc8264d9fa5d3279fd3b 10.35kB / 10.35kB   0.0s
 => => sha256:01272fe8adbacc44afd2b92994b31c40a151f 211.27MB / 211.27MB   6.2s
 => => sha256:cddc73e4e6c704bfa2325e53c32ddb3553c8fc3a9 6.16MB / 6.16MB   1.8s
 => => extracting sha256:cdd62bf39133c498a16f7a7b1b6555ba43d02b2511c508   4.9s
 => => sha256:cc48f13b5f0f44b2e298de83a94a99fe7abdfb3 18.06MB / 18.06MB   2.7s
 => => sha256:5a98c896c047f960c5fd29d44fa778899a68e7ebfb6a6 249B / 249B   2.7s
 => => extracting sha256:a47cff7f31e941e78bf63ca19f0811b675283e2c00ddea   0.8s
 => => extracting sha256:a173f2aee8e962ea19db1e418ae84a0c9f71480b51f768   3.2s
 => => extracting sha256:01272fe8adbacc44afd2b92994b31c40a151f4324ca392   9.1s
 => => extracting sha256:cddc73e4e6c704bfa2325e53c32ddb3553c8fc3a91dab6   0.3s
 => => extracting sha256:cc48f13b5f0f44b2e298de83a94a99fe7abdfb3335fe9b   0.9s
 => => extracting sha256:5a98c896c047f960c5fd29d44fa778899a68e7ebfb6a6a   0.0s
 => [internal] load build context                                        0.1s
 => => transferring context: 2.57MB                                      0.1s
 => [2/5] WORKDIR /code                                                  0.1s
 => [3/5] COPY requirements.txt /code                                    0.0s
 => [4/5] RUN pip install -r requirements.txt                           12.1s
 => [5/5] COPY . /code                                                   0.2s
 => exporting to image                                                   1.5s
 => => exporting layers                                                  1.5s
 => => writing image sha256:ed6da6b0c14f86049433120625be16b721d5045ccda  0.0s
 => => naming to docker.io/library/django-app                            0.0s
[ec2-user@ip-172-31-86-159 djangoDocker]$
```

Paso 7: Cree el contenedor Docker.

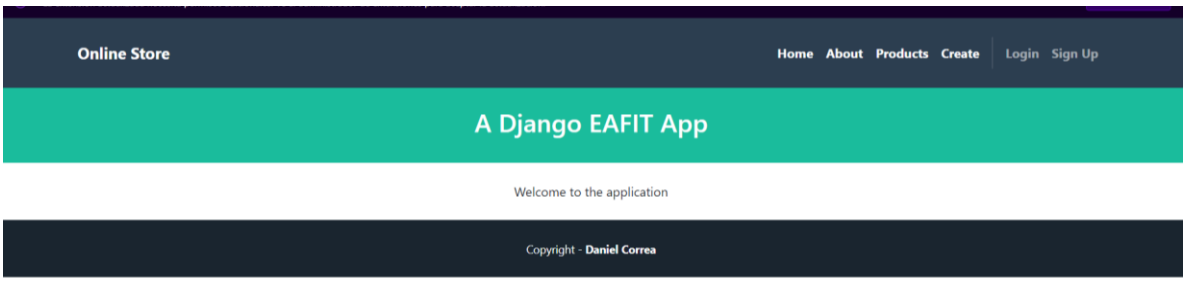
sudo docker container run -d --name django-docker -p 80:80 django-app

```
[ec2-user@ip-172-31-86-159 djangoDocker]$ sudo docker container run -d --name django-docker -p 80:80 django-app
cf92688b5a45e64523b413ef3ba9597b6c11766994afbc4fd5270c0293bb4354
```

Paso 8: Busque la ip de su instancia EC2, añádale “http://” al inicio, y “/public” al final. Acceda desde el navegador y deberá ver el proyecto corriendo.



Ejemplo: http://54.242.70.38/public/



¡Felicidades! ¡Ya tienes tu proyecto corriendo con Docker en la nube!

¡Borre la base de datos y detenga el contenedor una vez complete el tutorial para ahorrar créditos!