

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

Antes de iniciar:

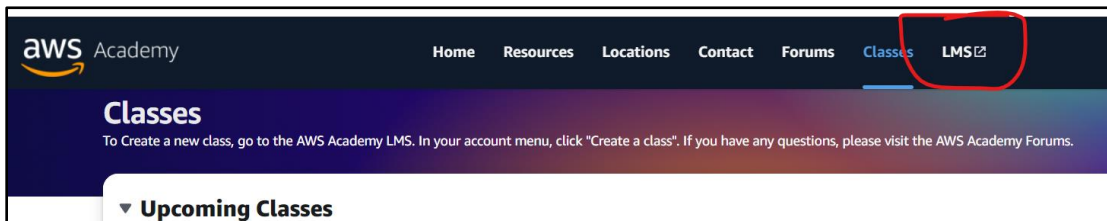
- Terminar los tutoriales anteriores
- Este taller muestra un ejemplo de despliegue de una aplicación Laravel con Docker en AWS.

## A) Acceso y Creación de la Instancia EC2

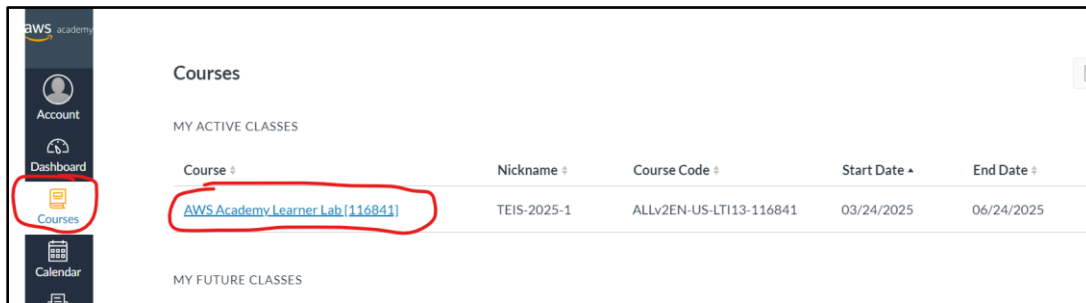
Como paso previo deberá solicitarle al docente el acceso a AWS Academy.

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

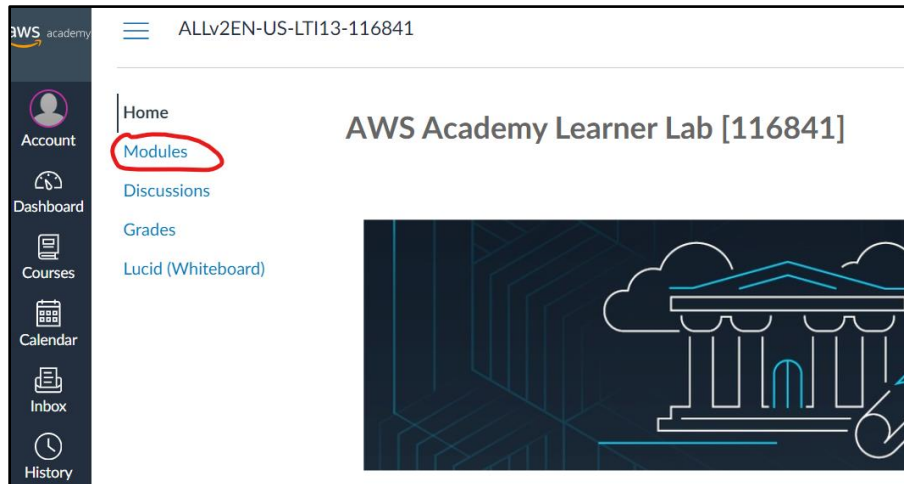
**Paso 2:** Ingrese a la sección de LMS.



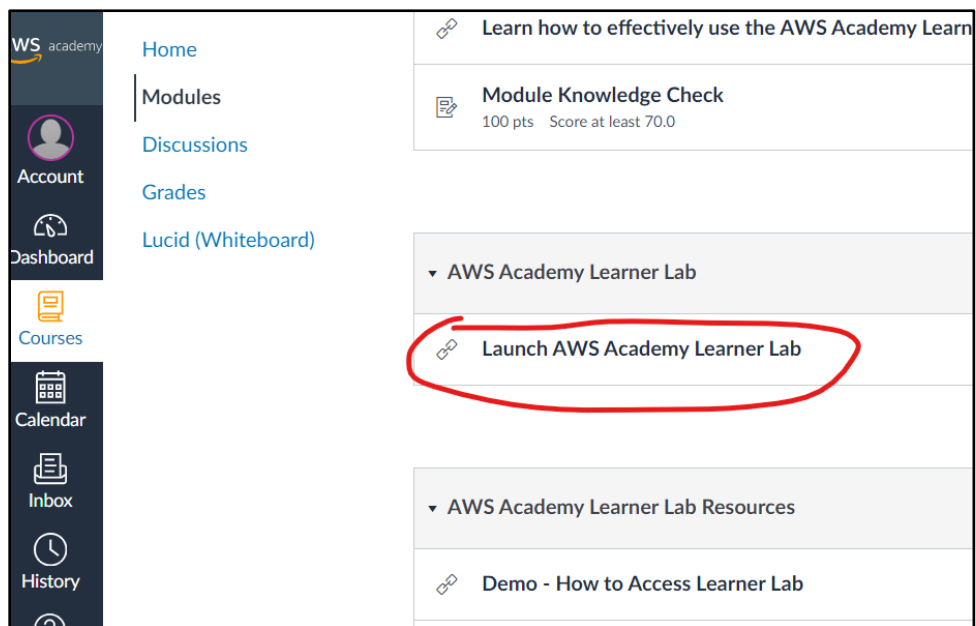
**Paso 3:** Vaya a la sección de cursos, busque el curso 116841 con el Nickname “TEIS-2025-1”, y de click en ese curso.



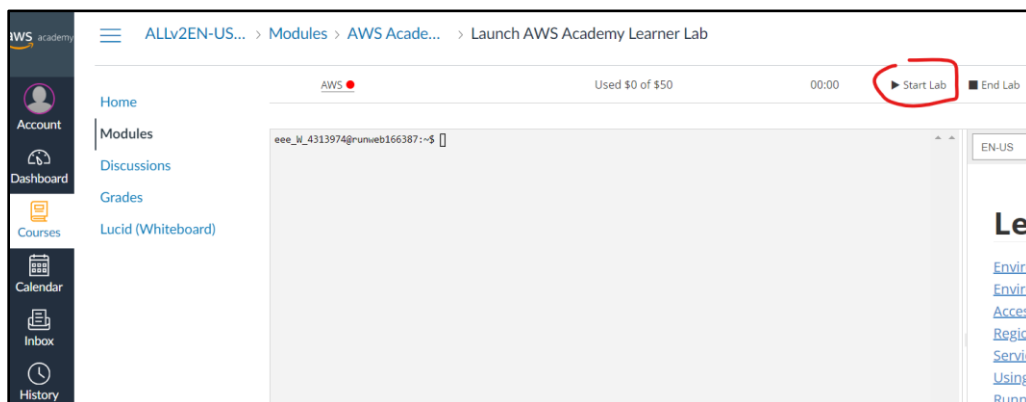
**Paso 4:** Vaya a la sección de “Modules” (algunas veces deberá aceptar unos términos y condiciones).



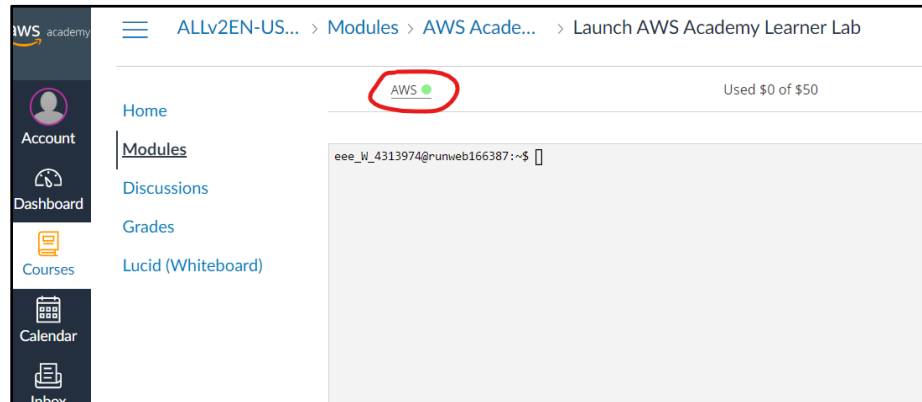
**Paso 5:** Busque y de click en “Launch AWS Academy Learner Lab”



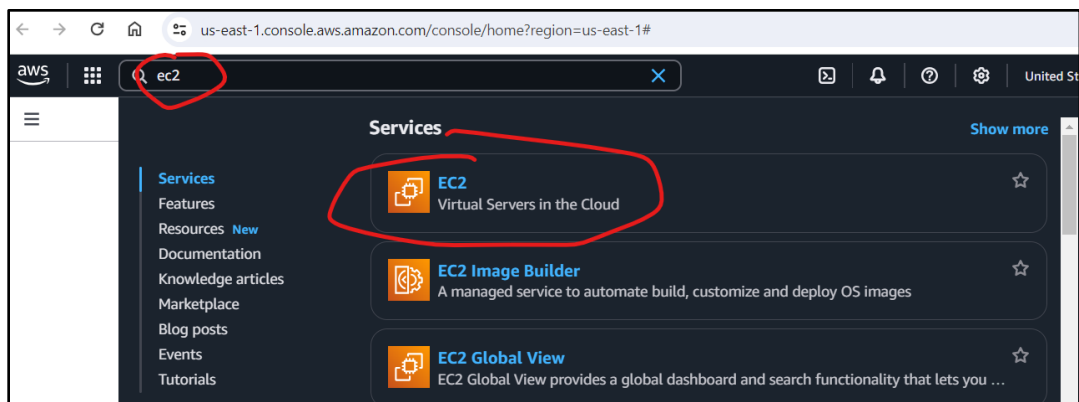
**Paso 6:** Inicie el laboratorio, dando click en “Start Lab”



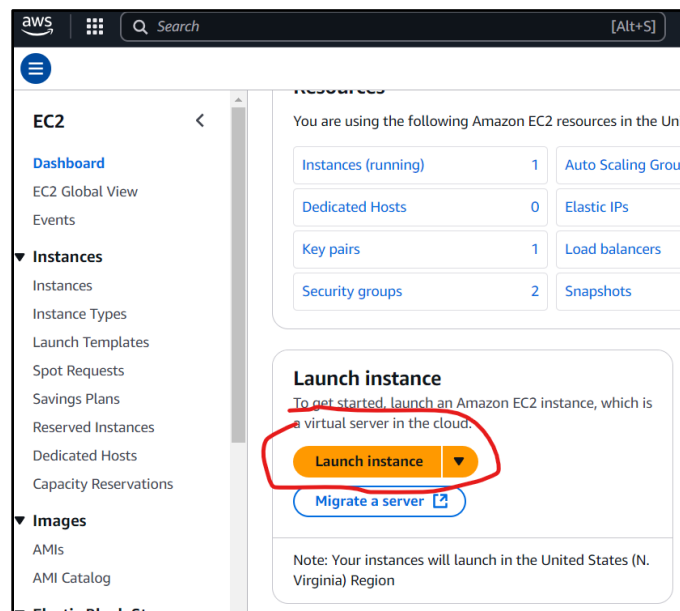
**Paso 7:** Espere unos minutos a que el bombillo de AWS se ponga verde. Y luego de click sobre AWS.



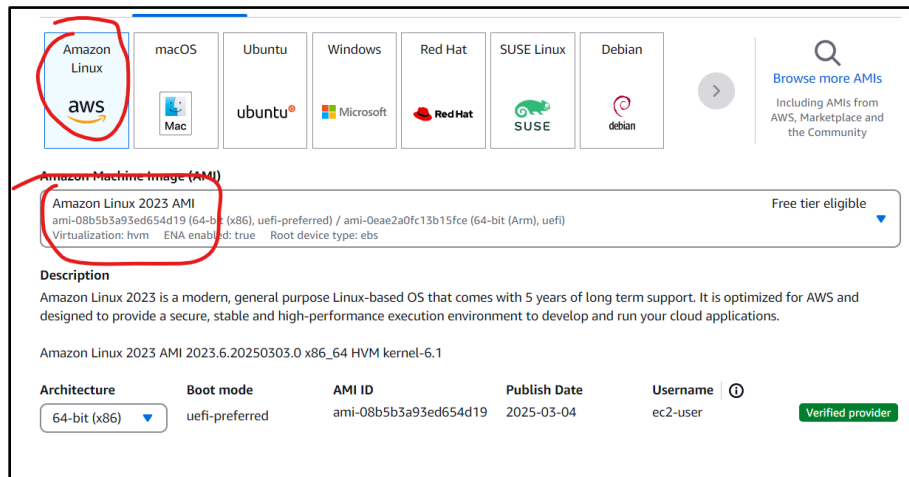
**Paso 8:** En la nueva ventana que se abre, busque “ec2”. Y luego de click sobre EC2.



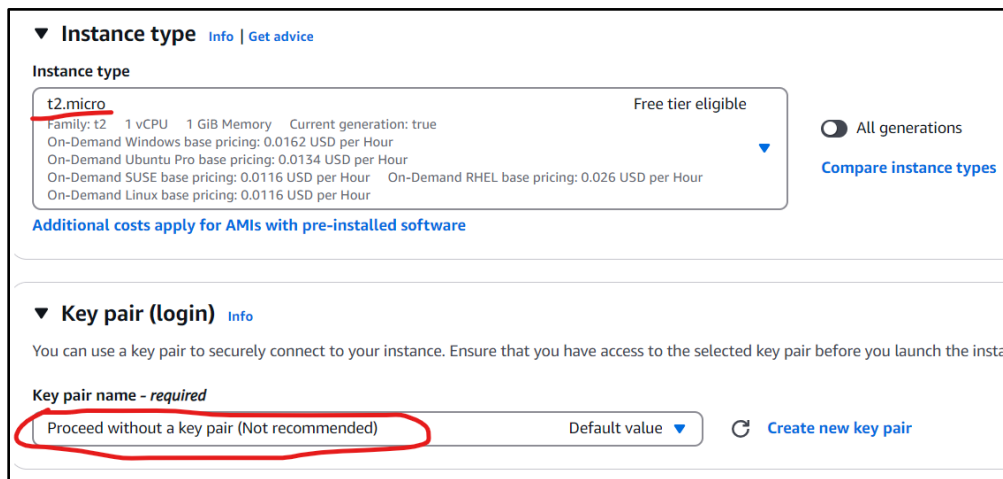
**Paso 9:** Luego de click en “Launch Instance”.



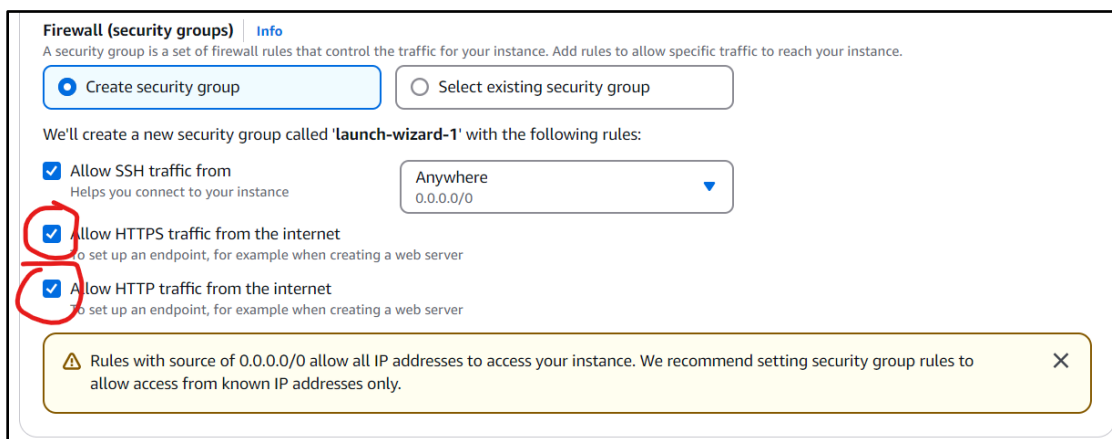
**Paso 10:** Verifique que está seleccionado “Amazon Linux” y “Amazon Linux 2023 AMI”



**Paso 11:** Verifique que está seleccionado “t2.micro” y en key pair name seleccione “Proceed without a key pair (Not recommended)”.



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



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

▼ Summary

Number of instances [Info](#)

1

Software Image (AMI)  
Amazon Linux 2023 AMI 2023.6.2...[read more](#)  
ami-08b5b3a93ed654d19

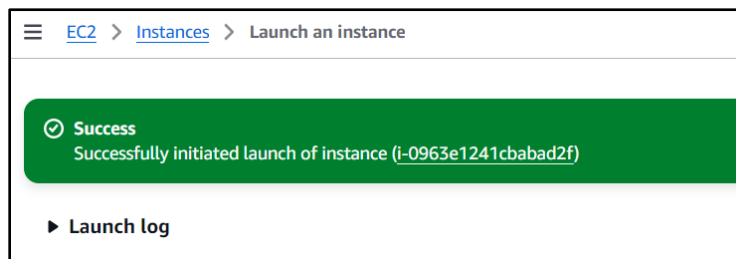
Virtual server type (instance type)  
t2.micro

Firewall (security group)  
New security group

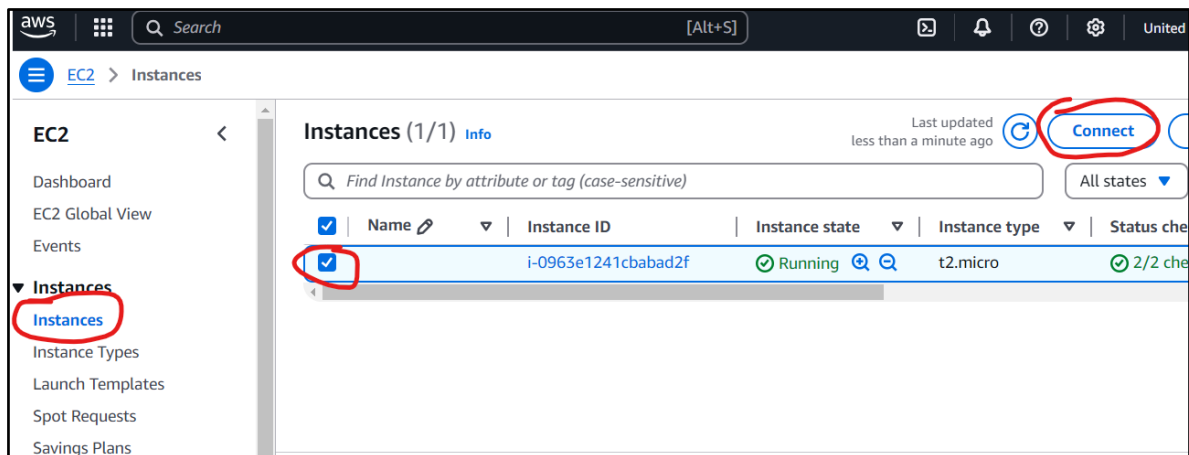
Storage (volumes)  
1 volume(s) - 8 GiB

[Cancel](#) [Launch instance](#) [Preview code](#)

**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”.

**Connection Type**

☒ **Connect using EC2 Instance Connect**  
Connect using the EC2 Instance Connect browser-based client, with a public IPv4 or IPv6 address.

☐ **Connect using EC2 Instance Connect Endpoint**  
Connect using the EC2 Instance Connect browser-based client, with a private IPv4 address and a VPC endpoint.

☒ **Public IPv4 address**  
18.212.179.221

☐ **IPv6 address**  
-

**Username**  
Enter the username defined in the AMI used to launch the instance. If you didn't define a custom username, use the default username, ec2-user.

ec2-user

**Note:** In most cases, the default username, ec2-user, is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.

Cancel **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
```

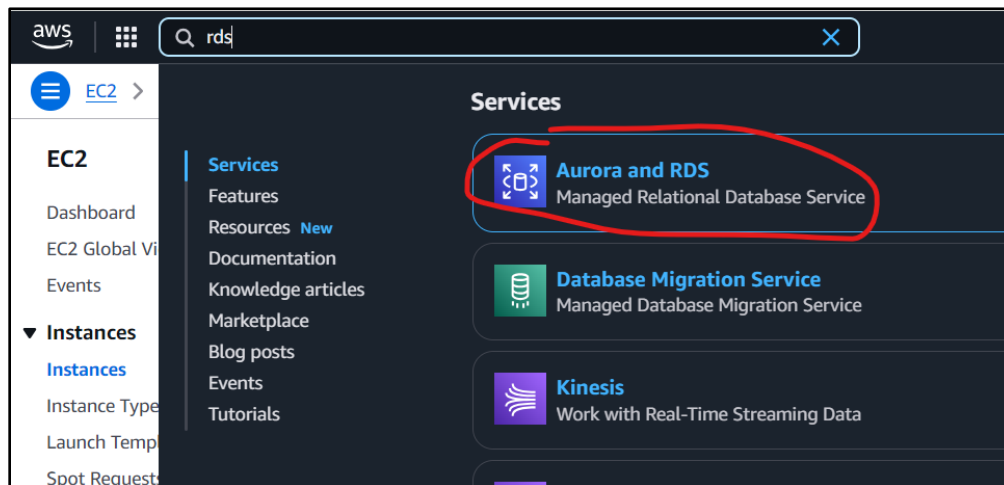
```
sudo dnf install -y docker
```

```
sudo systemctl enable --now docker
```

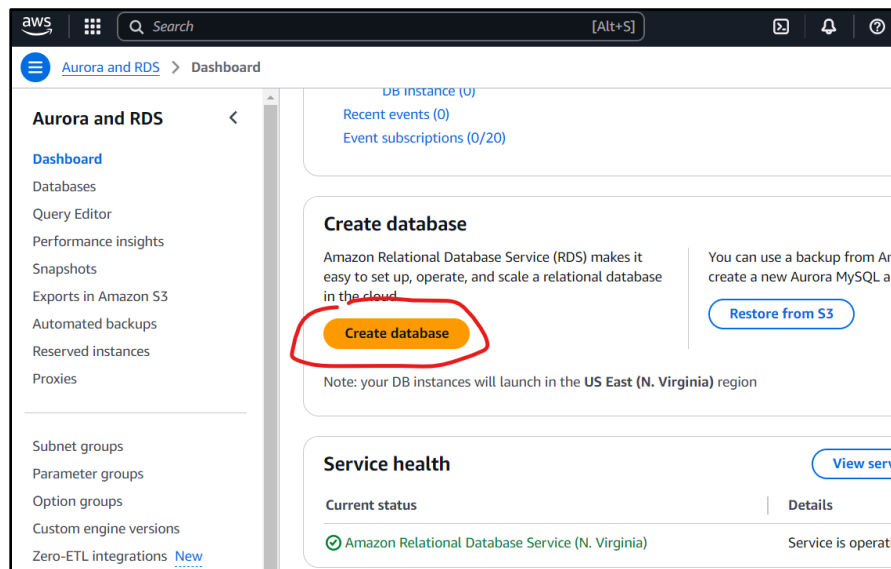
```
docker --version
```

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

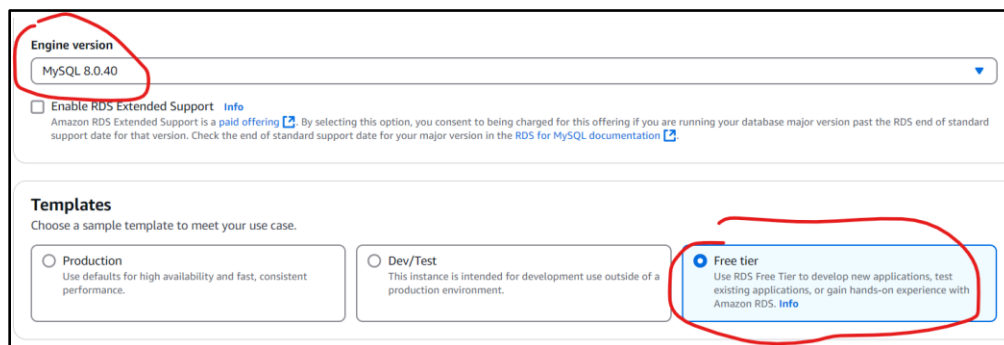
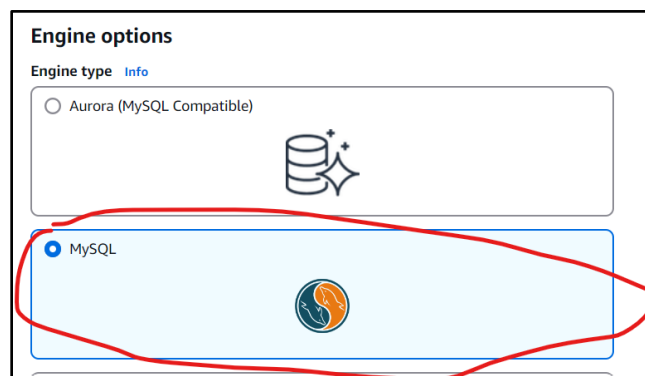
**Paso 2:** Vuelva al navegador, al sitio de AWS, y busque “Aurora and RDS”.



**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.

**Settings**

**DB instance identifier** [Info](#)  
Type a name for your DB instance. The name must be unique across all DB instances owned by your AWS account in the current AWS Region.

teis20251

The DB instance identifier is case-insensitive, but is stored as all lowercase (as in "mydbinstance"). Constraints: 1 to 63 alphanumeric characters or hyphens. First character must be a letter. Can't contain

▼ **Credentials Settings**

**Master username** [Info](#)  
Type a login ID for the master user of your DB instance.

admin

1 to 16 alphanumeric characters. The first character must be a letter.

**Credentials management**  
You can use AWS Secrets Manager or manage your master user credentials.

☐ Managed in AWS Secrets Manager - *most secure*  
RDS generates a password for you and manages it throughout its lifecycle using AWS Secrets Manager.

☒ Self managed  
Create your own password or have RDS create a password for you.

☐ Auto generate password  
Amazon RDS can generate a password for you, or you can specify your own password.

**Master password** [Info](#)

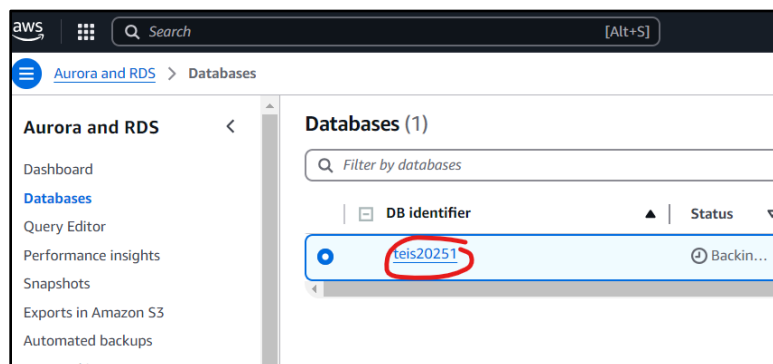
Password strength **Strong**

Minimum constraints: At least 8 printable ASCII characters. Can't contain any of the following symbols: / ' \* @

**Confirm master password** [Info](#)

**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.

The screenshot shows the AWS Aurora and RDS console for instance 'teis20251'. The left sidebar contains navigation options like Dashboard, Databases, Query Editor, Performance insights, Snapshots, Exports in Amazon S3, Automated backups, Reserved instances, Proxies, Subnet groups, Parameter groups, Option groups, Custom engine versions, Zero-ETL integrations, Events, and Event subscriptions. The main content area is titled 'teis20251' and has tabs for Connectivity & security, Monitoring, Logs & events, Configuration, Zero-ETL integrations, and Maintenance. The 'Connectivity & security' tab is active, showing details for Endpoint & port, Networking, and Security. The 'Security' section is circled in red, displaying 'VPC security groups' with the default group 'sg-0aa1dc1b59ad002c5' highlighted.

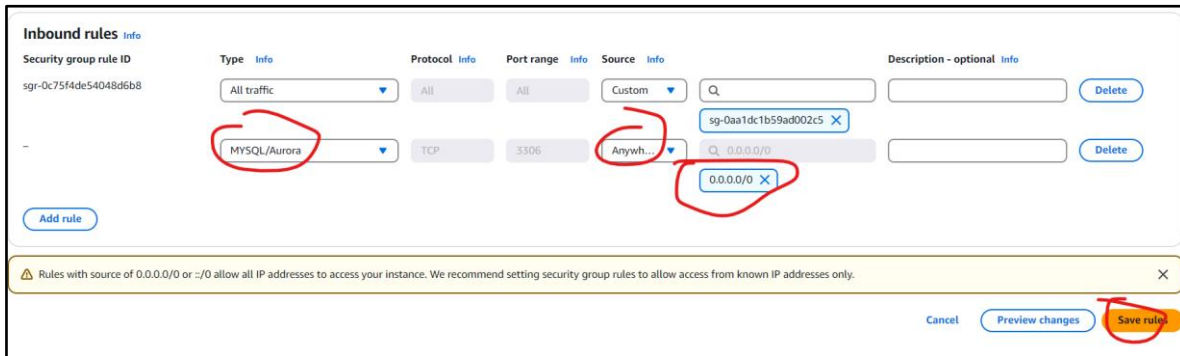
**Paso 9:** De click sobre el ID del security group.

The screenshot shows the AWS EC2 console 'Security Groups (1)' page. The left sidebar has options like Dashboard, EC2 Global View, Events, Instances, Instance Types, Launch Templates, and Spot Requests. The main content area has a search bar with 'Find resources by attribute or tag' and a filter button. Below the search bar, a table lists security groups. The first entry has the ID 'sg-0aa1dc1b59ad002c5' circled in red.

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

The screenshot shows the AWS EC2 console 'Inbound rules (1)' page. The top navigation bar includes 'Inbound rules', 'Outbound rules', 'Sharing - new', 'VPC associations - new', and 'Tags'. The main content area has a search bar and a table of inbound rules. The first rule has the ID 'sgr-0c75f4de54048d6b8' and is for 'All traffic' on 'All' ports. The 'Edit inbound rules' button is circled in red.

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



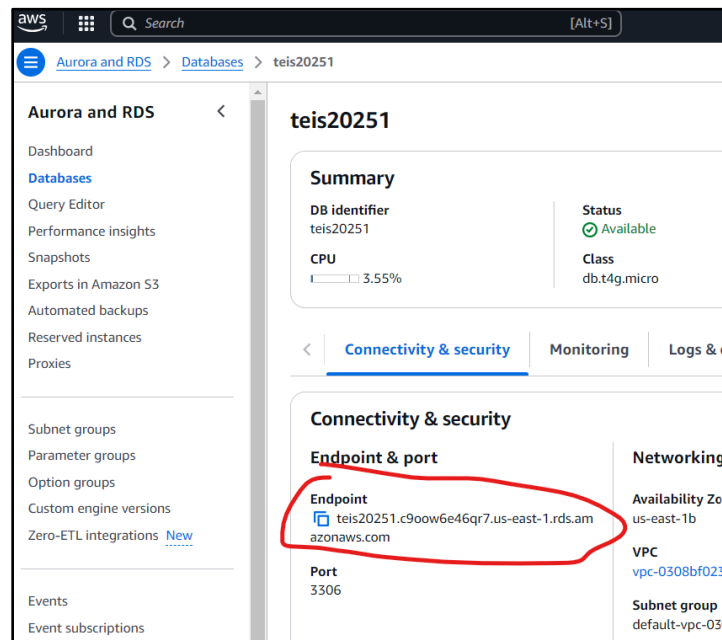
**Paso 12:** Vuelva a la consola de la instancia, y ahora instale MySQL (realmente es MariaDB) con el siguiente comando.

```
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*

```
[ec2-user@ip-172-31-25-69 ~]$ sudo docker container run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
e6590344b1a5: Pull complete
Digest: sha256:7e1a4e2d11e2ac7a8c3f768d4166c2defeb09d2a750b010412b6ea13de1efb19
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/
```

**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, deberá 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]$

```

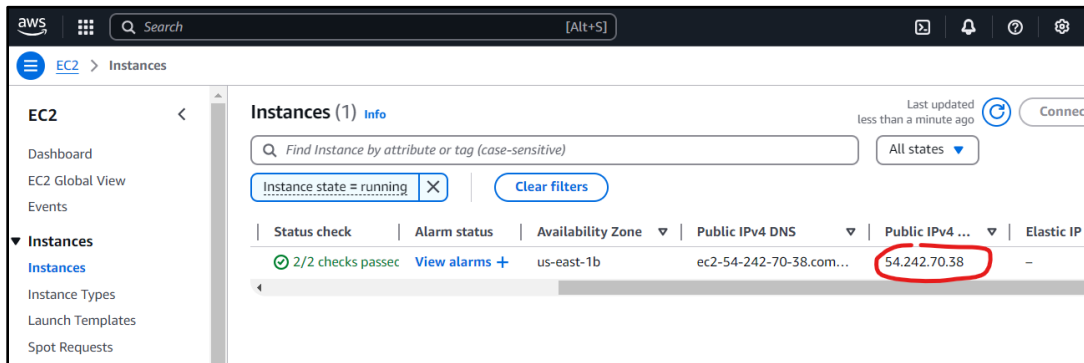
**Paso 7:** Cree el contenedor Docker.

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

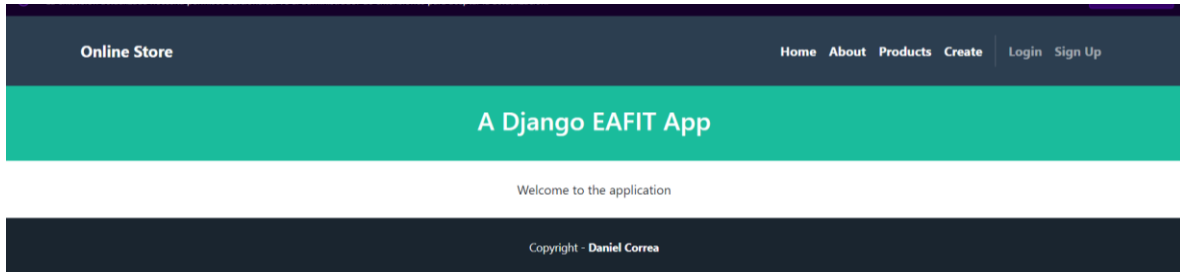
```
[ec2-user@ip-172-31-86-159 djangoDocker]$ sudo docker container run -d --name django-docker -p 80:80 django-app
```

```
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!**