

## Tutorial 07 to do in class – Remember to upload the repo link to Teams.

### Antes de iniciar:

- Terminar los tutoriales anteriores.
- Este taller muestra como desplegar el proyecto “randomquotes” como un microservicio (utilizando Docker Swarm, Docker service y GCP).
- REPOSITORIO: <https://github.com/Nram94/randomquotes-flask>

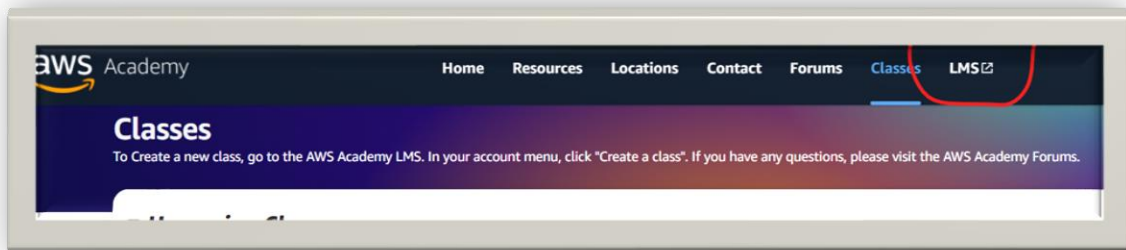
## A. Creación de template y lanzamiento de nueva instancia

### Sugerencia

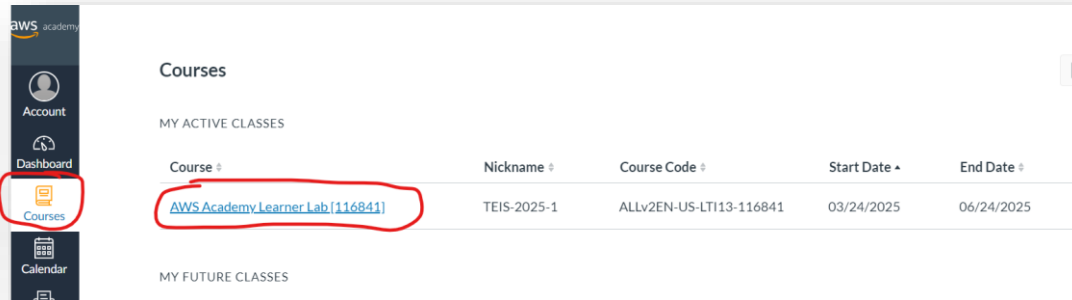
Pare o borre todas las instancias que ha creado hasta el momento (menos la de su proyecto si está trabajando en el).

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

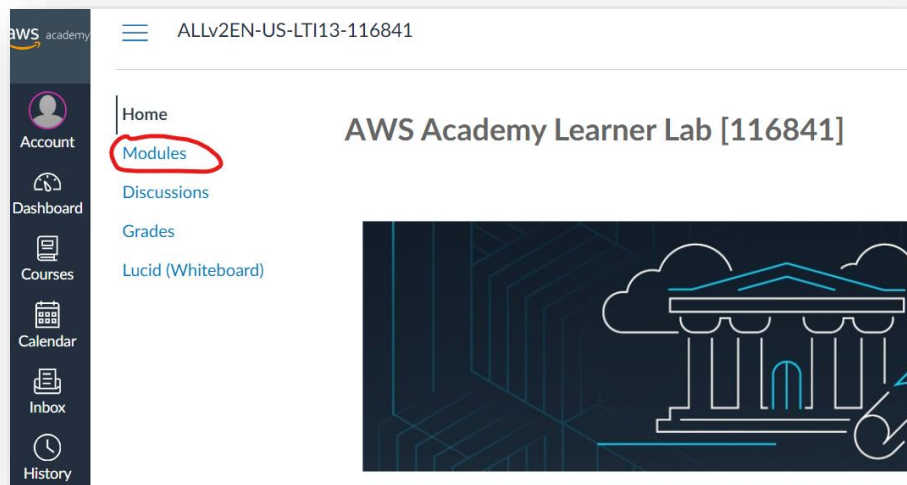
**Paso 2:** Ingrese a la sección de LMS.



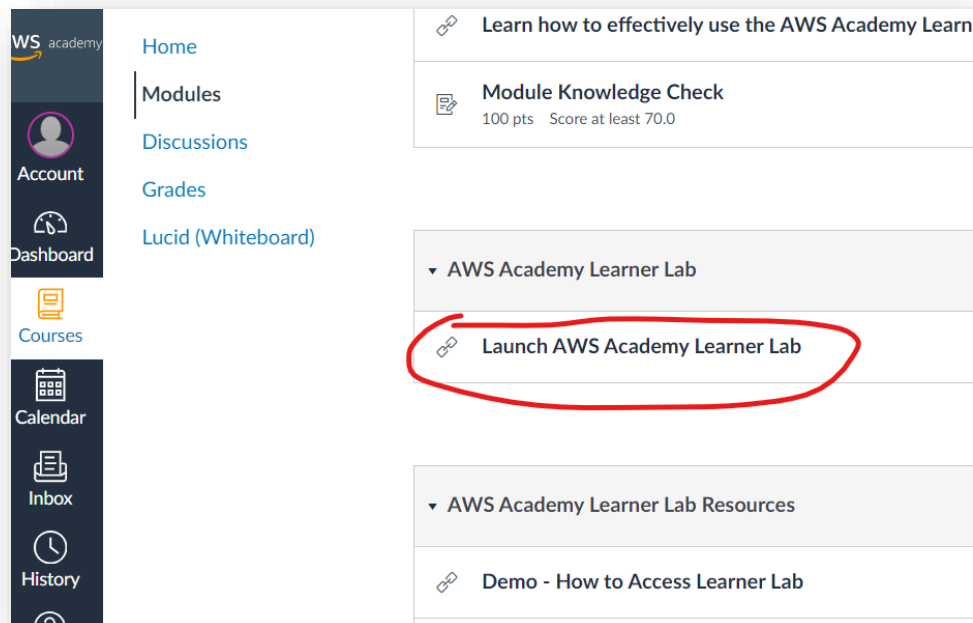
**Paso 3:** Vaya a la sección de cursos, busque el curso 116841 con el Nickname “TEIS-2025-1”, y de click en ese curso.



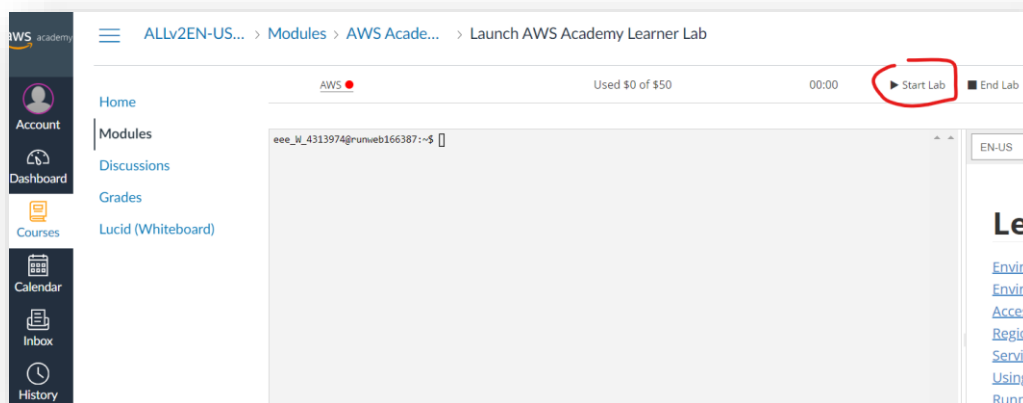
**Paso 4:** Vaya a la sección de “Modules” (algunas veces deberá aceptar unos términos y condiciones).



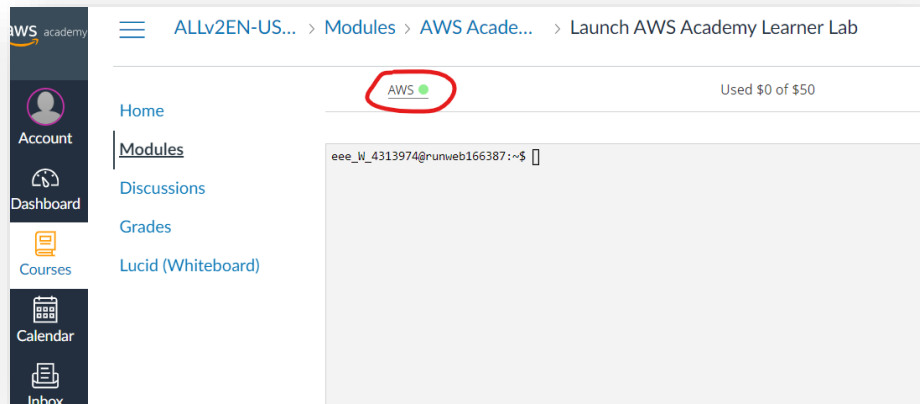
**Paso 5:** Busque y de click en “Launch AWS Academy Learner Lab”



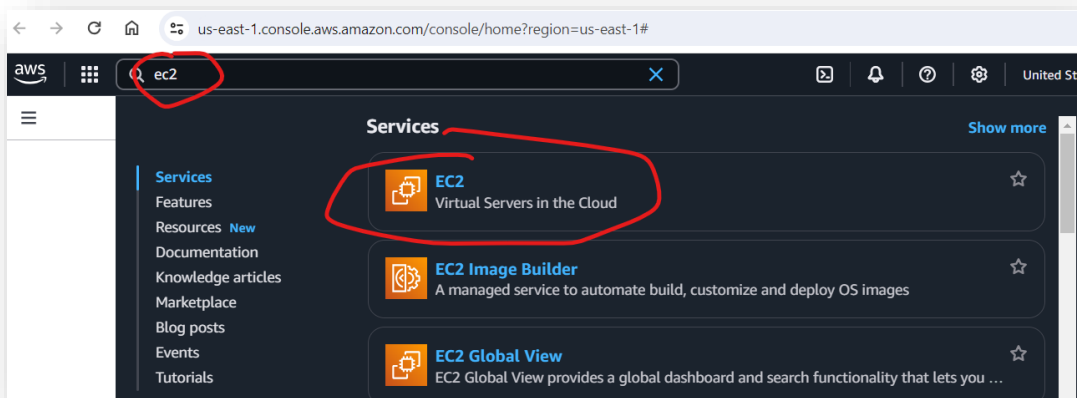
**Paso 6:** Inicie el laboratorio, dando click en “Start Lab”



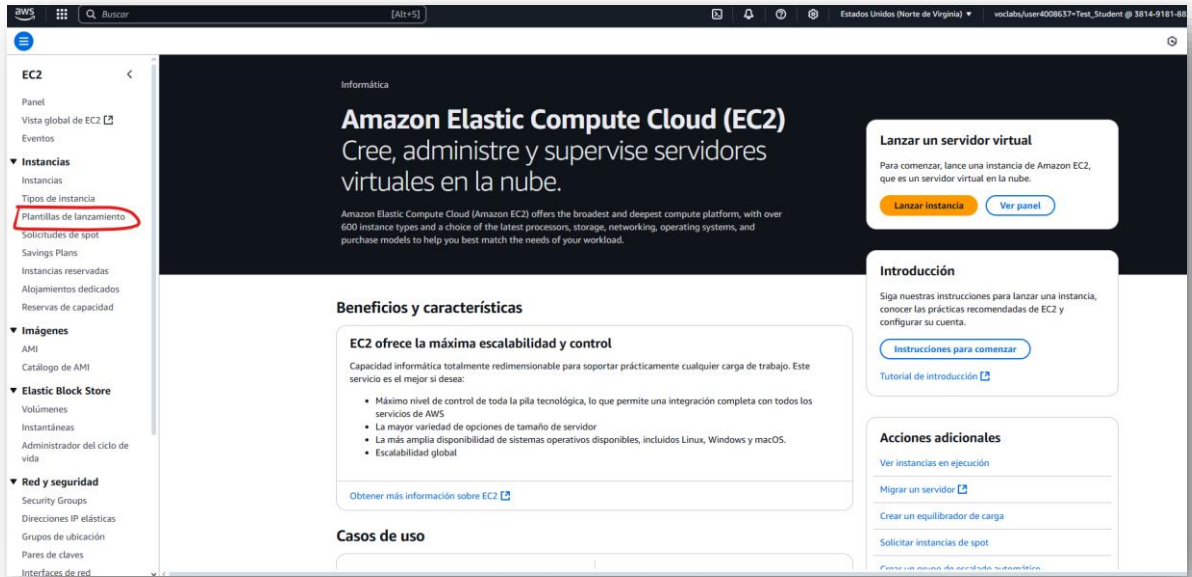
**Paso 7:** Espere unos minutos a que el bombillo de AWS se ponga verde. Y luego de click sobre AWS.



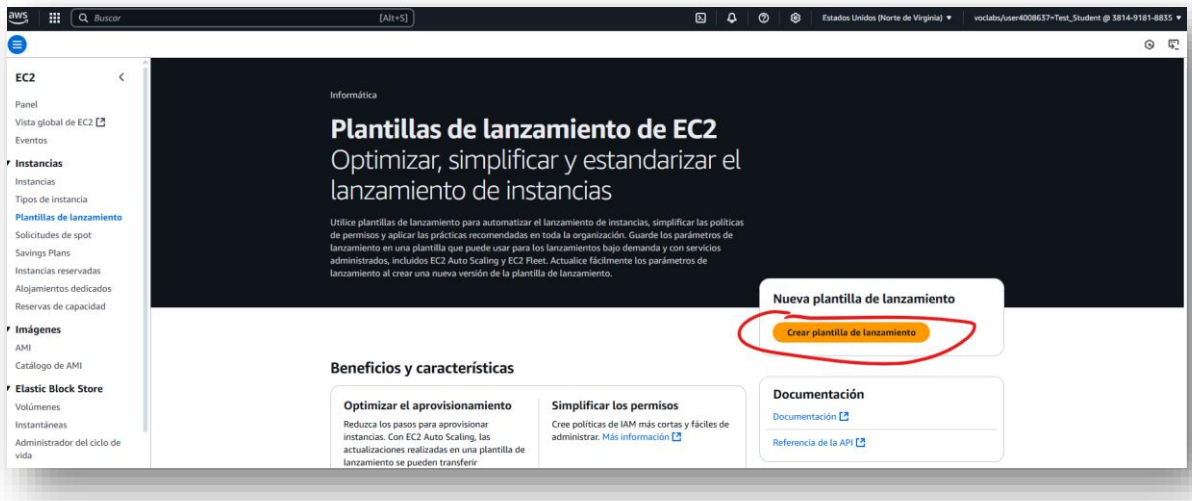
**Paso 8:** En la nueva ventana que se abre, busque “ec2”. Y luego de click sobre EC2.



**Paso 9:** Vaya a “Plantillas de Lanzamiento”



**Paso 10:** De click en “Crear plantilla de lanzamiento”



**Paso 11:** Dele un nombre a la plantilla

aws [Alt+S]

EC2 > Plantillas de lanzamiento > Crear plantilla de lanzamiento

## Crear plantilla de lanzamiento

La creación de una plantilla de lanzamiento le permite crear una configuración de instancia guardada que se puede reutilizar, compartir y lanzar más adelante. Las plantillas pueden tener varias versiones.

### Nombre y descripción de la plantilla de lanzamiento

Nombre de la plantilla de lanzamiento - *obligatorio*

teis20251-microservices

Debe ser única para esta cuenta. Máximo de 128 caracteres. Sin espacios ni caracteres especiales, como "&", "+", "\*", "@".

Descripción de la versión de la plantilla

Un servidor web de producción para MyApp

Máximo de 255 caracteres

**Orientación sobre Auto Scaling** | [Información](#)

Secciónelo si va a utilizar esta plantilla con EC2 Auto Scaling

☐ Proporcionar orientación que me ayude a configurar una plantilla que pueda utilizar con EC2 Auto Scaling

**Paso 12:** Asegúrese de seleccionar, en Imágenes de aplicaciones y sistemas operativos, Amazon Linux. Luego en Tipo de Instancia t2.micro. en Par de claves no incluir ninguna en la plantilla y crear un grupo de seguridad en Configuraciones de red, dándole un nombre y una descripción.

aws

Estados Unidos (Norte de Virgi

voclabs/user4008637=Test\_Student @ 3814-9181

EC2 > Plantillas de lanzamiento > Crear plantilla de lanzamiento

▼ Imágenes de aplicaciones y sistemas operativos (Imagen de máquina de Amazon)

Información

Una AMI es una plantilla que contiene la configuración de software (sistema operativo, servidor de aplicaciones y aplicaciones) necesaria para lanzar la instancia. Busque o examine las AMI si no ve lo que busca a continuación.

Busque en nuestro catálogo completo que incluye miles de imágenes de sistemas operativos y aplicaciones

Recientes

Inicio rápido

No incluir en la plantilla de lanzamiento

Amazon Linux

aws

macOS

Mac

Ubuntu

ubuntu

Windows

Microsoft

Red Hat

Red Hat

SUS

Buscar más AMI

Inclusión de AMI de AWS, Marketplace y la comunidad

Imágenes de máquina de Amazon (AMI)

AMI de Amazon Linux 2023

ami-00a929b66ed6e0de6 (64 bits (x86), uefi-preferred) / ami-05f417c208be02d4d (64 bits (Arm), uefi)

Virtualización: hvm Activado para ENA: true Tipo de dispositivo raíz: ebs

Apto para la capa gratuita

Descripción

Amazon Linux 2023 es un sistema operativo moderno y de uso general basado en Linux que incluye 5 años de soporte a largo plazo. Está optimizado para AWS y diseñado para proporcionar un entorno de ejecución seguro, estable y de alto desempeño para desarrollar y ejecutar sus aplicaciones en la nube.

Amazon Linux 2023 AMI 2023.7.20250331.0 x86\_64 HVM kernel-6.1

Arquitectura

64 bits (...)

Modo de arranque

uefi-preferred

ID de AMI

ami-00a929b66ed6e0de6

Fecha de publicación

2025-03-29

Nombre de usuario

ec2-user

Proveedor verificado

aws

Estados Unidos (Norte de Virgi

voclabs/user4008637=Test\_Student @ 3814-9181

EC2 > Plantillas de lanzamiento > Crear plantilla de lanzamiento

▼ Tipo de instancia

Información | Obtener asesoramiento

Avanzado

Tipo de instancia

No incluir en la plantilla de lanzamiento

Q |

Obtenga asesoramiento para elegir el tipo de instancia...

No incluir en la plantilla de lanzamiento

t2.nano

Familia: t2 1 vCPU 0.5 GiB Memoria Generación actual: true

Bajo demanda Linux base precios: 0.0058 USD por hora

Bajo demanda SUSE base precios: 0.0058 USD por hora

Bajo demanda Windows base precios: 0.0081 USD por hora

Bajo demanda Ubuntu Pro base precios: 0.0076 USD por hora

t2.micro

Familia: t2 1 vCPU 1 GiB Memoria Generación actual: true

Bajo demanda Windows base precios: 0.0162 USD por hora

Bajo demanda Ubuntu Pro base precios: 0.0134 USD por hora

Bajo demanda SUSE base precios: 0.0116 USD por hora

Bajo demanda RHEL base precios: 0.026 USD por hora

Bajo demanda Linux base precios: 0.0116 USD por hora

t2.small

Familia: t2 1 vCPU 2 GiB Memoria Generación actual: true

Bajo demanda Windows base precios: 0.032 USD por hora

Bajo demanda Linux base precios: 0.023 USD por hora

Bajo demanda RHEL base precios: 0.0376 USD por hora

Bajo demanda SUSE base precios: 0.053 USD por hora

Bajo demanda Ubuntu Pro base precios: 0.025 USD por hora

t2.medium

Familia: t2 2 vCPU 4 GiB Memoria Generación actual: true

Bajo demanda Ubuntu Pro base precios: 0.0499 USD por hora

Bajo demanda Linux base precios: 0.0464 USD por hora

Bajo demanda RHEL base precios: 0.0752 USD por hora

Bajo demanda Windows base precios: 0.0644 USD por hora

Bajo demanda SUSE base precios: 0.1464 USD por hora

Apto para la capa gratuita

t2.micro

Todas las generaciones

Comparar tipos de instancias

que tiene acceso al par de claves

Crear un nuevo par de claves

Crear nueva subred

reglas para permitir que un tráfico

Compare reglas de grupo de



**Configuraciones de red** Información

**Subred** Información

No incluir en la plantilla de lanzamiento [Crear nueva subred](#)

Al especificar una subred, se agrega automáticamente una interfaz de red a la plantilla.

**Firewall (grupos de seguridad)** Información

Un grupo de seguridad es un conjunto de reglas de firewall que controlan el tráfico de la instancia. Agregue reglas para permitir que un tráfico específico llegue a la instancia.

☐ Seleccionar un grupo de seguridad existente ☒ **Crear grupo de seguridad**

Nombre del grupo de seguridad - obligatorio

GrupoSeguridadTEIS

Este grupo de seguridad se agregará a todas las interfaces de red. El nombre no se puede editar después de crear el grupo de seguridad. La longitud máxima es de 255 caracteres. Caracteres válidos: a-z, A-Z, 0-9, espacios y .-:/()#,@!+=&:{}\$\*

**Descripción - obligatorio** Información

Grupo de seguridad para microservicios TEIS

**VPC** Información

vpc-03b1c746dda8cc8d4 (predeterminado)

**Reglas de grupos de seguridad de entrada**

No hay reglas del grupo de seguridad incluidas actualmente en esta plantilla. Agregue una nueva regla para incluirla en la plantilla de lanzamiento.

[Agregar regla del grupo de seguridad](#)

**Configuración de red avanzada**

**Paso 13:** Agregue las siguientes reglas para el grupo de seguridad. Luego en Detalles avanzados, en Datos de usuario, copié el siguiente script. Finalmente, Vaya a Resumen y dele Crear plantilla de lanzamiento

```
#!/bin/bash
# Script de ejemplo para instalar Docker en Amazon Linux 2023.
sudo dnf update -y
# Instalar Docker (Amazon Linux 2023 usa 'dnf' en vez de 'yum')
sudo dnf install -y docker
# Habilitar el servicio de Docker
sudo systemctl enable docker
sudo systemctl start docker
# Agregar el usuario ec2-user al grupo docker (opcional)
sudo usermod -aG docker ec2-user
# Verificar la instalación (requiere reconexión si agregaste ec2-user al grupo)
docker version
```

aws

Buscar

[Alt+S]

EC2 > Plantillas de lanzamiento > Crear plantilla de lanzamiento

Reglas de grupos de seguridad de entrada

▼ Regla del grupo de seguridad 1 (TCP, 22, 0.0.0.0/0)

Eliminar

Tipo | Información

ssh

Protocolo | Información

TCP

Intervalo de puertos | Información

22

Tipo de origen | Información

Cualquier lugar

Origen | Información

Q Agregue CIDR, lista de prefijos o grupo de seguridad

0.0.0.0/0 X

Descripción - opcional | Información

por ejemplo, SSH para Admin Desktop

▼ Regla del grupo de seguridad 2 (TCP, 80, 0.0.0.0/0)

Eliminar

Tipo | Información

HTTP

Protocolo | Información

TCP

Intervalo de puertos | Información

80

Tipo de origen | Información

Cualquier lugar

Origen | Información

Q Agregue CIDR, lista de prefijos o grupo de seguridad

0.0.0.0/0 X

Descripción - opcional | Información

por ejemplo, SSH para Admin Desktop

▼ Regla del grupo de seguridad 3 (TCP, 443, 0.0.0.0/0)

Eliminar

Tipo | Información

HTTPS

Protocolo | Información

TCP

Intervalo de puertos | Información

443

Tipo de origen | Información

Cualquier lugar

Origen | Información

Q Agregue CIDR, lista de prefijos o grupo de seguridad

0.0.0.0/0 X

Descripción - opcional | Información

por ejemplo, SSH para Admin Desktop

Datos de usuario - opcional | Información

Cargue un archivo con los datos de usuario o escríbalos en el campo.

⬆ Elegir archivo

```
#!/bin/bash
# Script de ejemplo para instalar Docker en Amazon Linux 2023.

sudo dnf update -y

# Instalar Docker (Amazon Linux 2023 usa 'dnf' en vez de 'yum')
sudo dnf install -y docker

# Habilitar el servicio de Docker
sudo systemctl enable docker
sudo systemctl start docker

# Agregar el usuario ec2-user al grupo docker (opcional)
sudo usermod -aG docker ec2-user
```

☐ Los datos de usuario ya han sido codificados en base64

▼ Resumen

Imagen de software (AMI)

Amazon Linux 2023 AMI 2023.7.2...[más información](#)

ami-00a929b66ed6e0de6

Tipo de servidor virtual (tipo de instancia)

t2.micro

Firewall (grupo de seguridad)

Nuevo grupo de seguridad

Almacenamiento (volúmenes)

Volúmenes: 1 (8 GiB)

ⓘ Nivel gratuito:

Durante el primer año que abre una cuenta de AWS, obtiene 750 horas al mes de uso de instancias t2.micro (o t3.micro cuando t2.micro no esté disponible) si se utiliza con AMI de nivel gratuito, 750 horas al mes de uso de direcciones IPv4 públicas, 30 GiB de almacenamiento de EBS, 2 millones de E/S, 1 GB de instantáneas y 100 GB de ancho de banda para Internet.

✕

Cancelar

→ Crear plantilla de lanzamiento

**Paso 14:** Vaya a Plantillas de Lanzamiento y seleccione su plantilla. Luego, en el botón de Acciones, seleccione “Lanzar la instancia desde una plantilla”.

EC2

Panel

Vista global de EC2

Eventos

▼ Instancias

Instancias

Tipos de instancia

Plantillas de lanzamiento

Solicitudes de spot

Savings Plans

Instancias reservadas

Alojamientos dedicados

Reservas de capacidad

Plantillas de lanzamiento (1/1) Información

Acciones

Crear plantilla de lanzamiento

ID de plantilla de lanzamie...	Nombre de plantilla de lanz...	Versión predet...	Última versión	Hora de creación	Creado por	Administradas	Ope...
lt-0949ef3547149e95a	teis20251-microservices	1	2	2025-04-13T21:33:08.000Z	arn:aws:sts:381491818835:ass...	falso	-



**Paso 15:** Verifique que la información sea la misma que en la configuración.

**Paso 16:** En el cuadro de resumen indique el número de instancias a lanzar, verifique que la información sea correcta y presione Lanzar instancia. Si le pide crear un par de claves, dele Continuar sin claves y Lanzar instancia.

## ▼ Resumen

Número de instancias | [Información](#)

2

Al lanzar más de una instancia, [considere utilizar EC2 Auto Scaling](#)

### Imagen de software (AMI)

Amazon Linux 2023 AMI 2023.7.2...[más información](#)

ami-00a929b66ed6e0de6

### Tipo de servidor virtual (tipo de instancia)

t2.micro

### Firewall (grupo de seguridad)

GrupoSeguridadTEIS

### Almacenamiento (volúmenes)

Volúmenes: 1 (8 GiB)

**i Nivel gratuito:** Durante el primer año que abre una cuenta de AWS, obtiene 750 horas al mes de uso de instancias t2.micro (o t3.micro cuando t2.micro no esté disponible) si se utiliza con AMI de nivel gratuito, 750 horas al mes de uso de direcciones IPv4 públicas, 30 GiB de almacenamiento de EBS, 2 millones de E/S, 1 GB de instantáneas y 100 GB de ancho de banda para Internet.

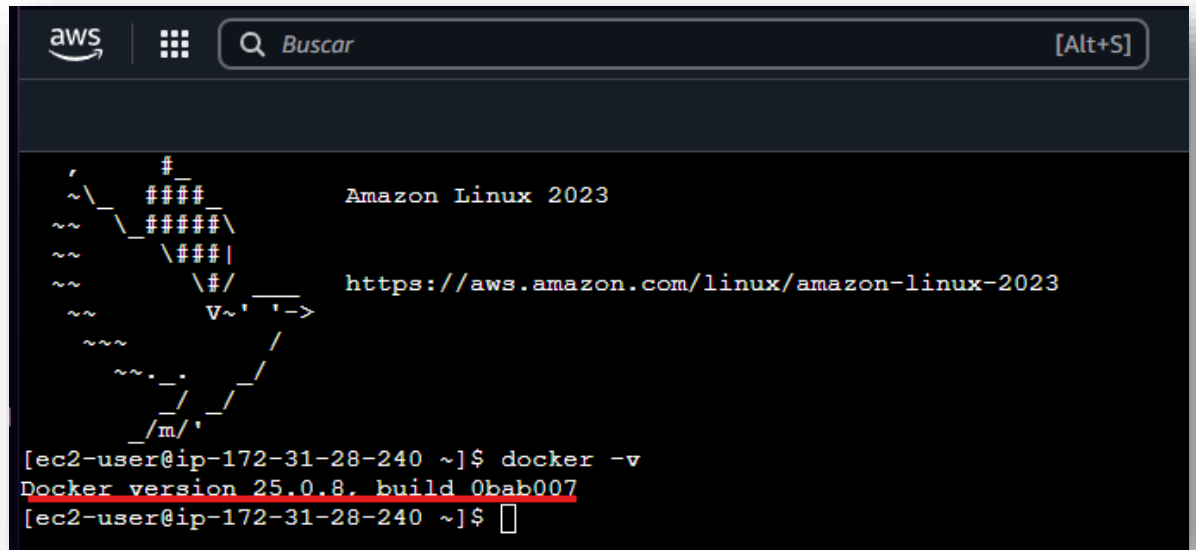


[Cancelar](#)

[Lanzar instancia](#)

[Código de versión preliminar](#)

- **Nota:** verifique que las instancias quedaron bien creadas. Conéctese a cada instancia y ejecute el comando `docker -v` (debe esperar un par de minutos mientras se instala Docker). Debería ver la versión de Docker en ambas instancias.



# Inicialización Docker Swarm

## Identificación de instancias

- De ahora en adelante hablaremos de las 2 instancias creadas de la siguiente manera.
  - La primera instancia que le aparezca listada en, la llamaremos **nodo-01**.
  - La segunda instancia que le aparezca listada en VM instances, la llamaremos **nodo-02**.

**NOTA:** En la columna Name puede asignarle un nombre a sus instancias presionando el ícono del lápiz, asignando el nombre y luego presionando el ícono de verificación.

Instancias (2) Información

Última actualización  
Hace less than a minute

Conectar

Estado de la instancia

Acciones

Lanzar instancias

Buscar Instancia por atributo o etiqueta (case-sensitive)

Todos los ...

Name

ID de la instancia

Estado de la i...

Tipo de inst...

Comprobación de

Estado de la al...

Zona de dispon...

DNS de IPv4 pública

Dirección IP...

IP elástica

nodo-01

i-0e9cbdcf49299f42

En ejecución

t2.micro

Inicializando

Ver alarmas +

us-east-1d

ec2-3-80-91-246.comp...

3.80.91.246

-

nodo-02

i-012693e24e75fac35

En ejecución

t2.micro

Inicializando

Ver alarmas +

us-east-1d

ec2-44-220-95-94.com...

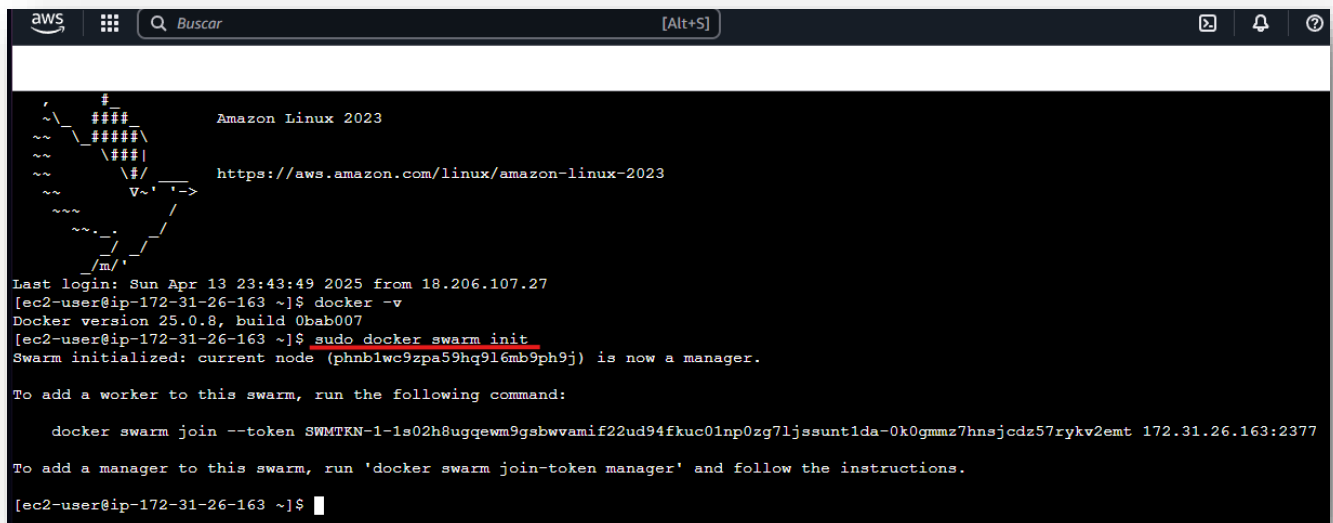
44.220.95.94

-

## Inicializando un cluster de Docker Swarm desde nodo-01

- Conéctese al nodo-01.
- Ejecute ahora este comando para iniciar Docker Swarm

*sudo docker swarm init*



```
#####
#####\
~~~~\#####
~~~~\#/\
~~~~\V-'\-->
~~~~\
~~~~\
~~~~\m/'

Amazon Linux 2023

https://aws.amazon.com/linux/amazon-linux-2023

Last login: Sun Apr 13 23:43:49 2025 from 18.206.107.27
[ec2-user@ip-172-31-26-163 ~]$ docker -v
Docker version 25.0.8, build 0bab007
[ec2-user@ip-172-31-26-163 ~]$ sudo docker swarm init
Swarm initialized: current node (phnb1wc9zpa59hq9l6mb9ph9j) is now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-1s02h8ugqewm9gsbwvamif22ud94fkuc01np0zg7ljssuntlda-0k0gmmz7hnsjcdz57rykv2emt 172.31.26.163:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.

[ec2-user@ip-172-31-26-163 ~]$
```

- Ahora ejecutaremos un comando, que permitirá que otros nodos se conecten con el cluster de Docker Swarm que acabamos de iniciar. En este caso como “manager”. Para eso ejecutaremos el siguiente comando (el cual utilizaremos LUEGO en el nodo-02):

*sudo docker swarm join-token manager*

- Copie y guarde el comando que aparece en rojo.

```
To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.

[ec2-user@ip-172-31-26-163 ~]$ sudo docker swarm join-token manager
To add a manager to this swarm, run the following command:
docker swarm join --token SWMTKN-1-1s02h8uggewm9gsbwvamif22ud94fkuc01np0zg7ljssunt1da-3kbv4p27fs4d0d5qt6dy2559z 172.31.26.163:2377
[ec2-user@ip-172-31-26-163 ~]$
```

- Ejecute el siguiente comando para ver los nodos activos en el cluster (debería aparecer solo 1).

*sudo docker node ls*

```
[ec2-user@ip-172-31-26-163 ~]$ sudo docker node ls
```

ID	HOSTNAME	STATUS	AVAILABILITY	MANAGER STATUS	ENGINE VERSION
phnb1wc9zpa59hq9l6mb9ph9j *	ip-172-31-26-163.ec2.internal	Ready	Active	Leader	25.0.8

```
[ec2-user@ip-172-31-26-163 ~]$
```

## B. Conectando nodos con un cluster Docker Swarm

### Conexión del nodo-02

- Conéctese al nodo-02.
- Ejecute el comando copiado anteriormente (agréguete *sudo* al inicio), el que incluía el token para unirse como manager. **Verifique que el comando quedo todo en una sola línea, o si no saldrá error (a veces el comando se parte en 2 partes al copiar y pegar).**

```
[ec2-user@ip-172-31-28-240 ~]$ sudo docker swarm join --token SWMTKN-1-1s02h8uggewm9gsbwvamif22ud94fkuc01np0zg7ljssunt1da-3kbv4p27fs4d0d5qt6dy2559z 172.31.26.163:2377
This node joined a swarm as a manager.
[ec2-user@ip-172-31-28-240 ~]$
```

### Verificación desde el nodo-01 (también se puede hacer desde el nodo-02)

- Conéctese al nodo-01.



- Liste los nodos para verificar si el nodo-02 quedó correctamente enlazado (le deberán aparecer 2 nodos).

*sudo docker node ls*

```
[ec2-user@ip-172-31-26-163 ~]$ sudo docker node ls
ID                                HOSTNAME                                STATUS    AVAILABILITY    MANAGER STATUS    ENGINE VERSION
phnb1wc9zpa59hq9l6mb9ph9j *      ip-172-31-26-163.ec2.internal          Ready     Active           Leader            25.0.8
kzzz43yhbgiwj4elzo9580h2z        ip-172-31-28-240.ec2.internal          Ready     Active           Reachable         25.0.8
[ec2-user@ip-172-31-26-163 ~]$
```

- **Nota:** ese mismo comando lo podría ejecutar desde el nodo-02 dado que es un manager.

## C. Modificando el proyecto randomquotes

### Controller

- Realice el siguiente cambio en el archivo *index.js*. Luego suba los cambios a GitHub (si no lo ha hecho, cree el repo y suba los cambios).

#### Modify Bold Code

```
from flask import Flask, jsonify
import os
import random

app = Flask(__name__)

phrases = [
    "Get ready to be inspired...",
    "See rejection as redirection.",
    "There is beauty in simplicity.",
    "You can't be late until you show up.",
    "Maybe life is testing you. Don't give up.",
    "Impossible is just an opinion.",
    "Alone or not you gonna walk forward.",
]

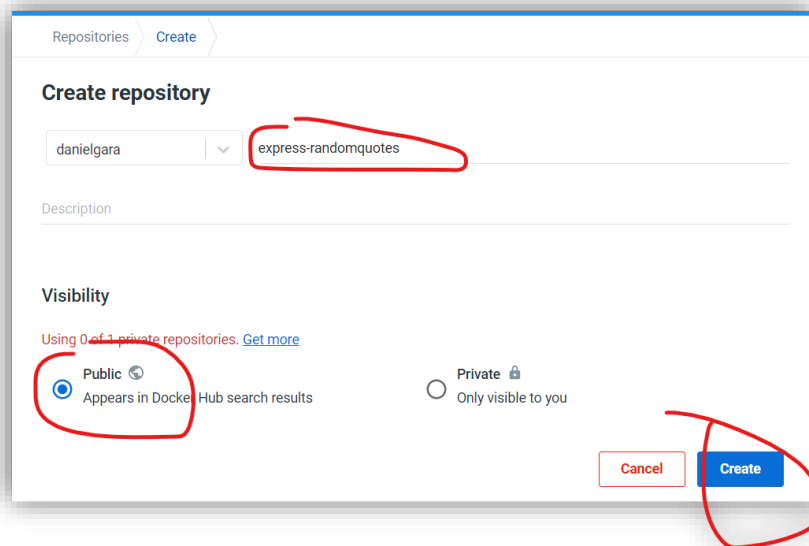
@app.route('/')
def get_random_quote():
    phrase = random.choice(phrases)
    container_id = os.uname()[1]
    return f"{phrase} - Container Id: {container_id}"
```

```
if __name__ == '__main__':  
    app.run(host='0.0.0.0', port=80)  
...
```

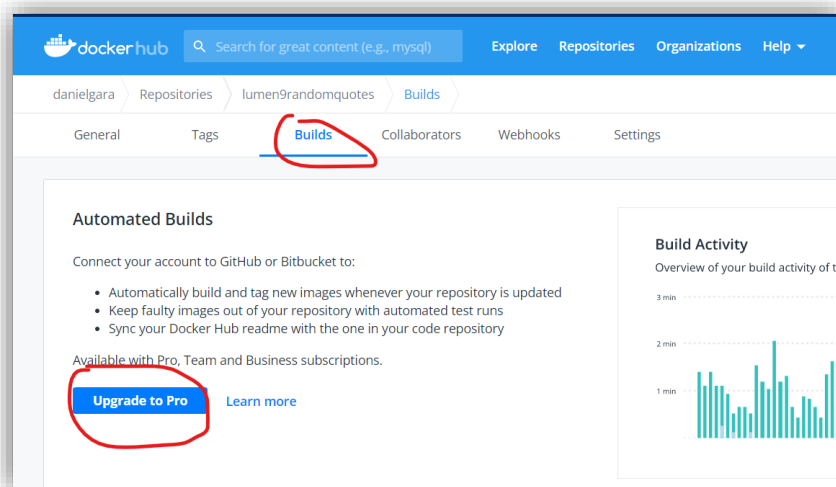
## D.Desplegando el proyecto randomquotes en DockerHub

### DockerHub

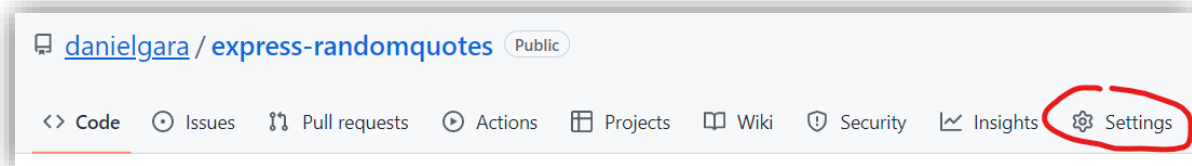
- Visite dockerhub.com, cree una cuenta, y luego cree un repositorio.



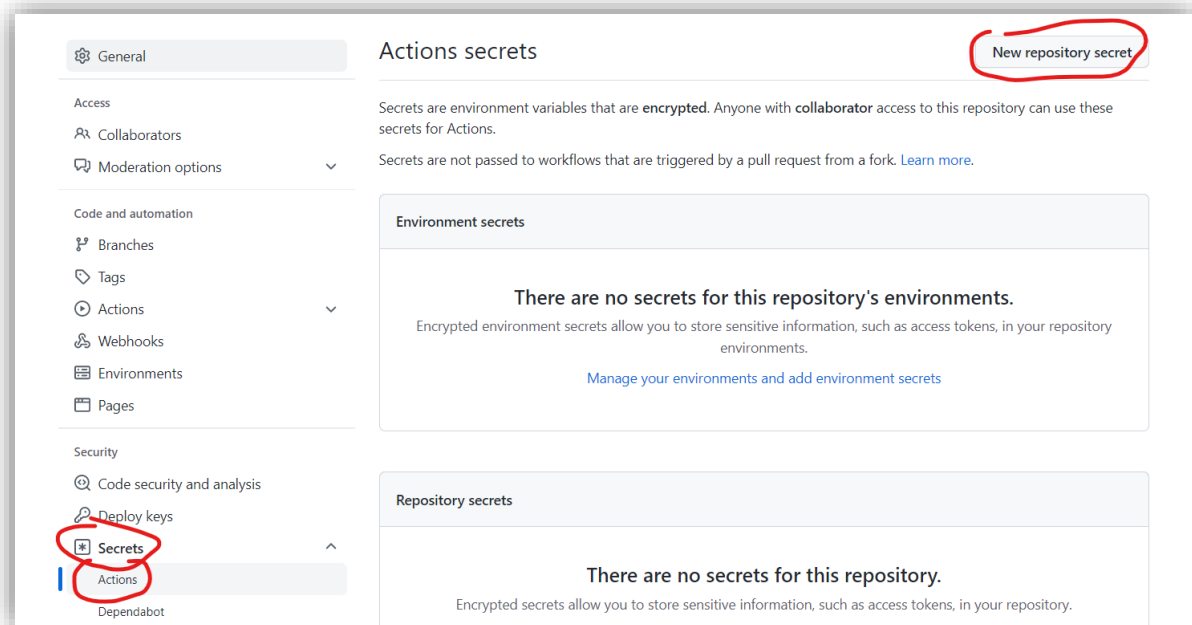
- Anteriormente una vez creado el Repositorio en DockerHub, podíamos conectarlo directamente a nuestro proyecto en GitHub; y crear automáticamente las imágenes de nuestro proyecto. Pero ya no, ya es solo para planes pro (carita triste).



- Veamos cómo solucionar el problema. Vaya a su repositorio en GitHub, y de click en la pestaña "Settings".



- Vaya luego a “Secrets” -> “Actions”, y luego a “New repository secret”.



- Agregue en Name `DOCKERHUB_USER` y en Value coloque su usuario de DockerHub.

A screenshot of the 'Actions secrets / New secret' form. The 'Name' field contains the text 'DOCKERHUB\_USER'. The 'Value' field contains the text 'danielgara'.

- Repita el proceso, pero ahora agregue en Name `DOCKERHUB_PASS` y en Value coloque su contraseña de DockerHub.

Actions secrets / New secret

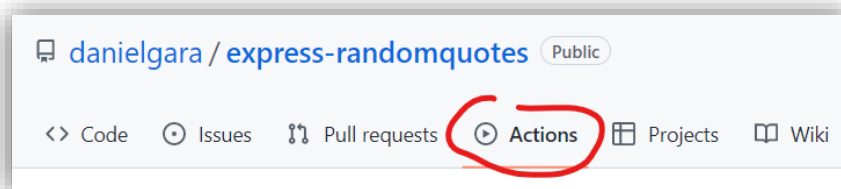
Name

DOCKERHUB\_PASS

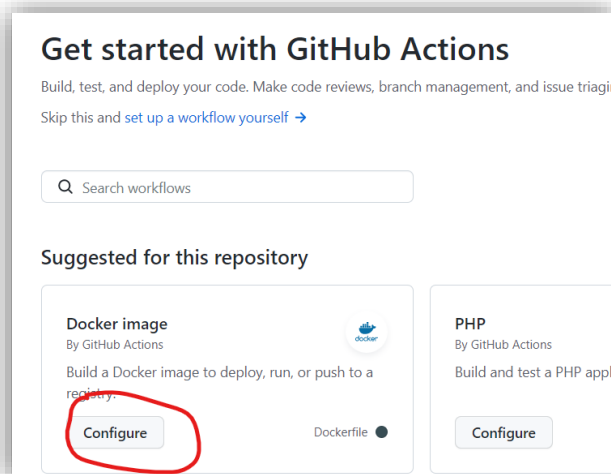
Value

CASICAIGOPERONO

- Ahora, vaya entonces a su repositorio en GitHub, y de click en la pestaña “Actions”.



- Dado que nuestro proyecto tiene un archivo *Dockerfile* en la raíz, entonces GitHub nos va a sugerir crear un Docker image. De click a “Configure”:



- Borre todo lo que le sale por defecto, y reemplázelo por lo siguiente. Ojo, lo que hay en rojo debe reemplazarlo por: el nombre de su usuario en DockerHub / nombre de su repositorio en DockerHub.

name: Docker Image CI

on:

```

push:
  branches: [ master ]
pull_request:
  branches: [ master ]

jobs:

  build:

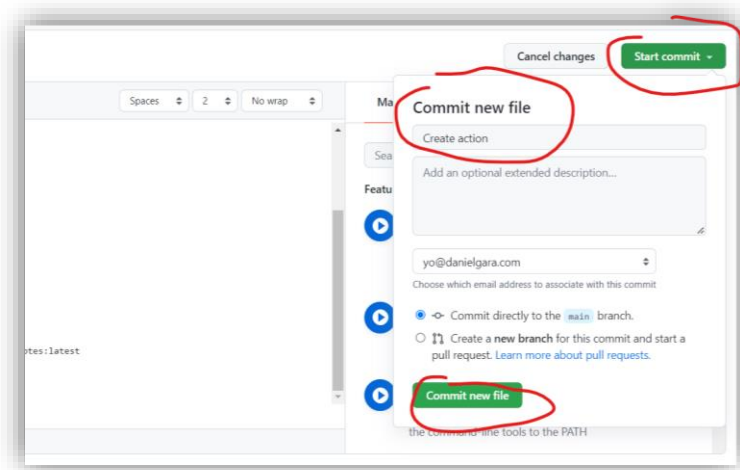
    runs-on: ubuntu-latest

    steps:
      - uses: actions/checkout@v4
      - name: docker login
        env:
          DOCKER_USER: ${secrets.DOCKERHUB_USER}}
          DOCKER_PASS: ${secrets.DOCKERHUB_PASS}}
        run: |
          docker login -u $DOCKER_USER -p $DOCKER_PASS
      - name: Build the Docker image
        run: docker build . --file Dockerfile --tag danielgara/express-randomquotes:latest

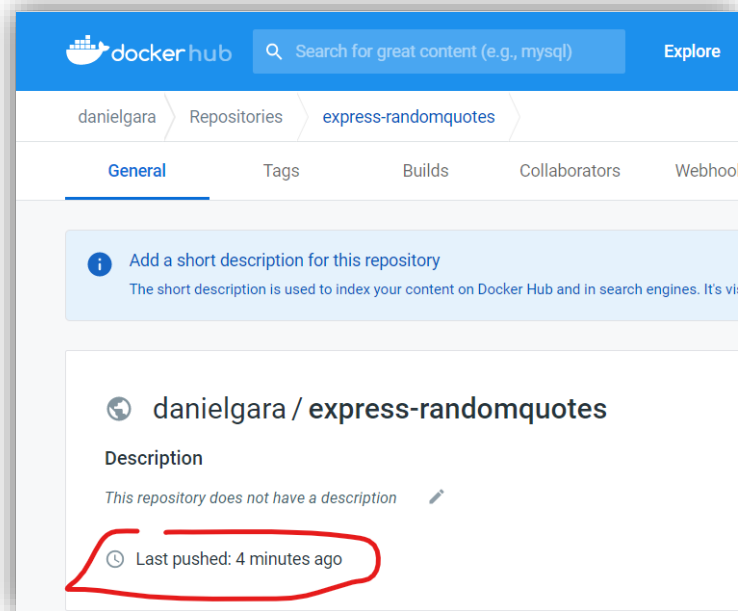
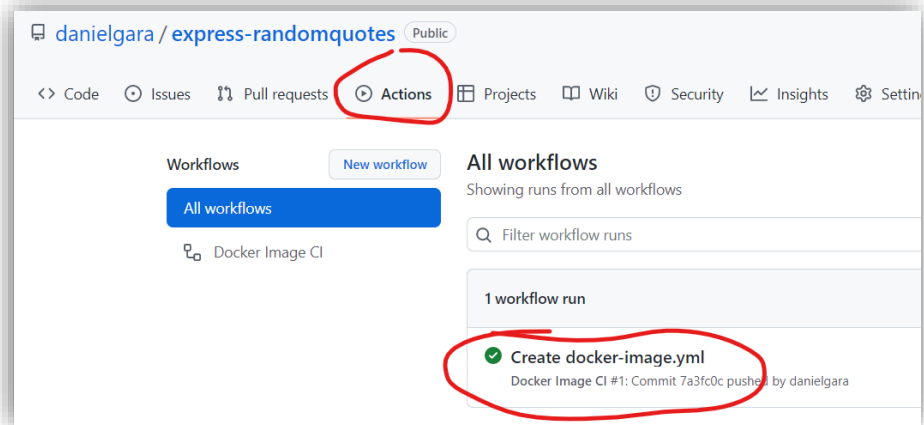
      - name: Docker Push
        run: docker push danielgara/express-randomquotes

```

- Luego de click en “Start commit”, coloque un texto al commit, y luego en “Commit new file”.



- Regrese a GitHub Actions, en un par de minutos le debe salir en verde el workflow ejecutado, lo cual quiere decir que ya debería tener la imagen de su proyecto montada en DockerHub.



- Ya no necesitamos la cuenta pro (carita feliz). Y no solo eso, cada vez que usted haga un push a Github, se va a ejecutar ese GitHub actions, y actualizar su imagen en Docker Hub automáticamente.

## E. Desplegando el proyecto con Docker service

### Conexión del nodo-01

- Conéctese al nodo-01.

### Despliegue del servicio

- Ejecute el siguiente comando para verificar cuantos servicios están corriendo en este momento:

*sudo docker service ls*

```
[ec2-user@ip-172-31-26-163 ~]$ sudo docker service ls
ID                NAME                MODE                REPLICAS        IMAGE                PORTS
[ec2-user@ip-172-31-26-163 ~]$
```

- Cree un servicio llamado *randomquotes* basado en la imagen de su proyecto en DockerHub, y defínale 4 réplicas. Para eso utilice el siguiente comando (reemplace la última palabra en rojo por su repositorio DockerHub -> por ejemplo yo use *danielgara/express-randomquotes*):

*sudo docker service create --name randomquotes --replicas 4 -p 80:80 REPO\_DOCKER\_HUB*

```
[ec2-user@ip-172-31-26-163 ~]$ sudo docker service create --name randomquotes --replicas 4 -p 80:80 nram94/flask-randomquotes
jwqa2exrsubkc7huc9xl6rsxd
overall progress: 4 out of 4 tasks
1/4: running [=====>]
2/4: running [=====>]
3/4: running [=====>]
4/4: running [=====>]
verify: Service converged
[ec2-user@ip-172-31-26-163 ~]$
```

- Ahora veamos en que nodos quedaron desplegados las 4 réplicas, con el siguiente comando:

*sudo docker service ps randomquotes*

```
[ec2-user@ip-172-31-26-163 ~]$ sudo docker service ps randomquotes
ID                NAME                IMAGE                NODE                DESIRED STATE    CURRENT STATE    ERROR    PORTS
14zq01wpjdr      randomquotes.1      nram94/flask-randomquotes:latest    ip-172-31-28-240.ec2.internal    Running          Running 4 minutes ago
zec68g4qexcu      randomquotes.2      nram94/flask-randomquotes:latest    ip-172-31-26-163.ec2.internal    Running          Running 4 minutes ago
z0wf9satjooc      randomquotes.3      nram94/flask-randomquotes:latest    ip-172-31-28-240.ec2.internal    Running          Running 4 minutes ago
bs2z4ub5lvfa      randomquotes.4      nram94/flask-randomquotes:latest    ip-172-31-26-163.ec2.internal    Running          Running 4 minutes ago
[ec2-user@ip-172-31-26-163 ~]$
```



- Como vemos 2 réplicas están corriendo en el nodo-01 y las otras 2 en el nodo-02.
- Para verificar las réplicas, podemos mostrar la lista de contenedores desde el nodo-01 y nodo-02.

*sudo docker container ls*

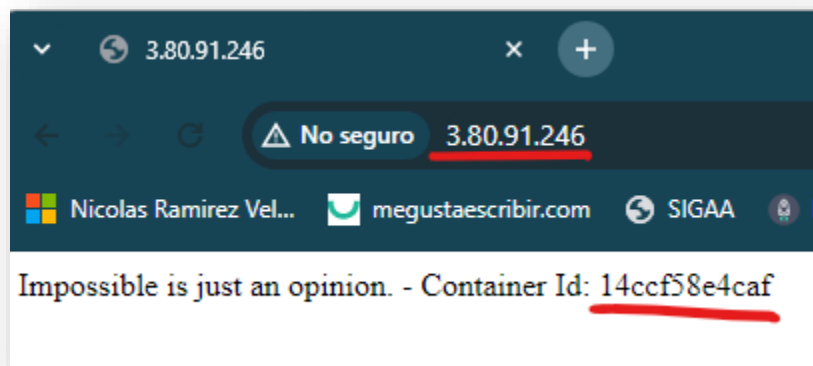
```
[ec2-user@ip-172-31-26-163 ~]$ sudo docker container ls
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS          NAMES
78a8695aa71e   nram94/flask-randomquotes:latest   "python randomquotes..." 7 minutes ago  Up 7 minutes  8080/tcp       randomquotes.4.os2z4ub51vfajhl8bzhqj39
14ccf58e4caf   nram94/flask-randomquotes:latest   "python randomquotes..." 7 minutes ago  Up 7 minutes  8080/tcp       randomquotes.2.zec68g4gexcujj22a5bjvut57
[ec2-user@ip-172-31-26-163 ~]$
```

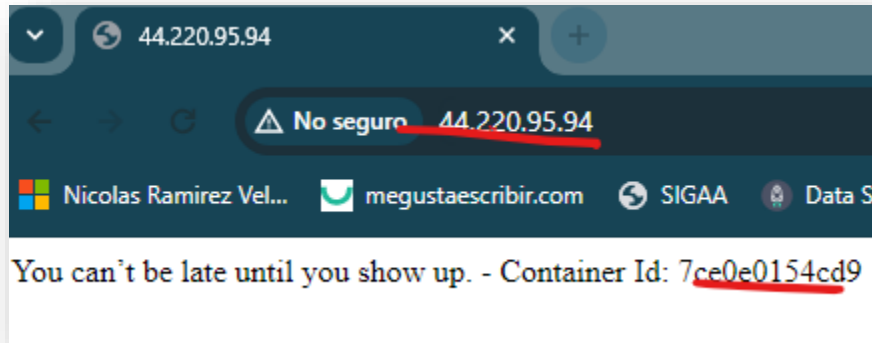
i-0e9cbdcf49299f42 (nodo-01)

```
[ec2-user@ip-172-31-28-240 ~]$ sudo docker container ls
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS          NAMES
7ce0e0154cd9   nram94/flask-randomquotes:latest   "python randomquotes..." 5 minutes ago  Up 5 minutes  8080/tcp       randomquotes.3.z0wf9satjoocvdqbzclmelpny
adacec63ba21   nram94/flask-randomquotes:latest   "python randomquotes..." 5 minutes ago  Up 5 minutes  8080/tcp       randomquotes.1.14zq01wpjdr26118e07whp4i
[ec2-user@ip-172-31-28-240 ~]$
```

i-012693e24e75fac35 (nodo-02)

- Si accedemos a desde la IP pública del nodo-01 o desde la IP pública del nodo-02, podremos ver la aplicación corriendo. Recargue la página con un espacio de 3 segundos (o abra en incognito), y podrá ver que el **container id** cambia en múltiples ocasiones. Esto quiere decir que, desde la misma IP, estamos cargando la aplicación desde diferentes contenedores (que podrían o no estar alojados en el mismo servidor). Vea el siguiente ejemplo, desde la IP del nodo-01, está cargando un contenedor alojado en el nodo-01, pero al recargar, carga otro alojado en el nodo-02.





## Clusters

- El sistema es tan “inteligente”, que independiente desde la IP o DNS que se acceda, cargará un contenedor disponible del cluster (no importa si pertenece a esa misma IP o DNS desde que se accedió).

## Actividad

- Conéctese a el nodo-01, liste los contenedores, elimine un contenedor.
- Espere unos segundos y vuelva a listar los contenedores del nodo-01.
- ¿Entendió que acabó de pasar?

Para detener el cluster hay dos opciones. Puede conectarse al nodo-01 y ejecutar cualquiera de estas:

`sudo docker service rm nombre_del_servicio` -> Remueve el servicio y detiene las replicas

`sudo docker service scale nombre_del_servicio=0` -> Retira todas las instancias del contenedor, pero conserva la configuración.

**Felicidades! ¡Ya tienes tus microservicios corriendo con Docker en la nube!**

- **¡Detenga los contenedores y detenga las instancias una vez complete el tutorial para ahorrar créditos!**

