

Actividad: Paso a Paso Despliegue de una API en la nube

Objetivo

El objetivo de este trabajo práctico es que los alumnos desarrollen las pruebas unitarias y de integración utilizando jest y supertest. Las pruebas deben aplicarse sobre la API de Biblioteca.

Consigna

1. Hacer un Fork del repositorio base.
2. Seguir las instrucciones del video.
 - a. Crear la cuenta de AWS.
 - b. Ingresar a la consola.
3. Crear imagen docker y publicarla en AWS.
4. Configurar Cluster de Elastic Container Service (ECS) para desplegar el contenedor con la imagen del punto anterior.
5. Desplegar Contenedor.

Requisitos para el práctico

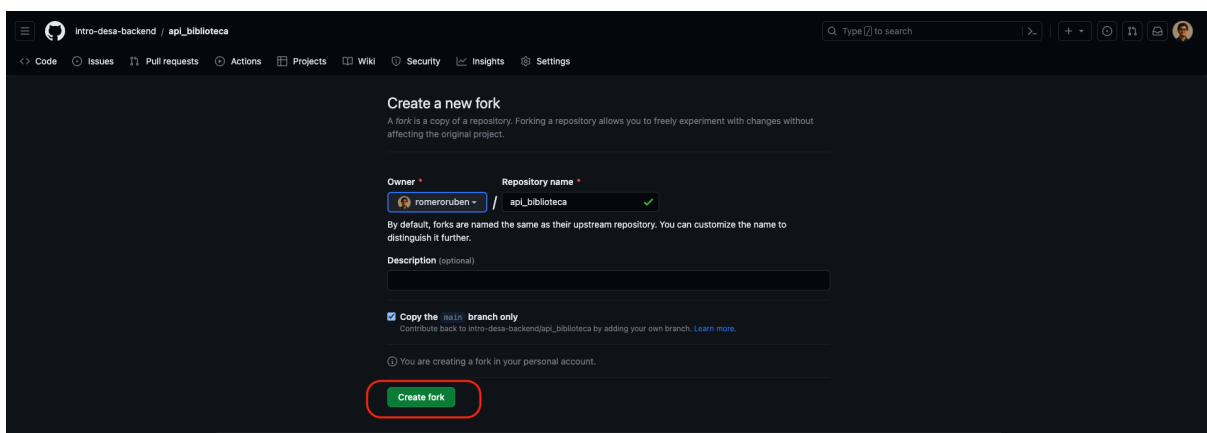
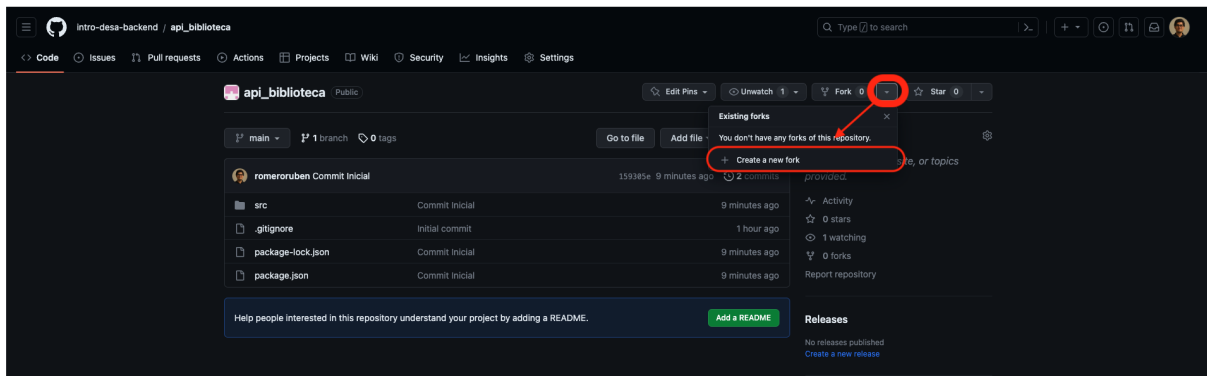
- Instalar las siguientes aplicaciones
 - **Docker.** Para crear imágenes y contenedores Docker.
 - <https://www.docker.com/products/docker-desktop/>
 - **AWS CLI.** Para poder desplegar aplicaciones en AWS.
 - https://docs.aws.amazon.com/es_es/cli/latest/userguide/getting-started-install.html

Desarrollo Paso a Paso

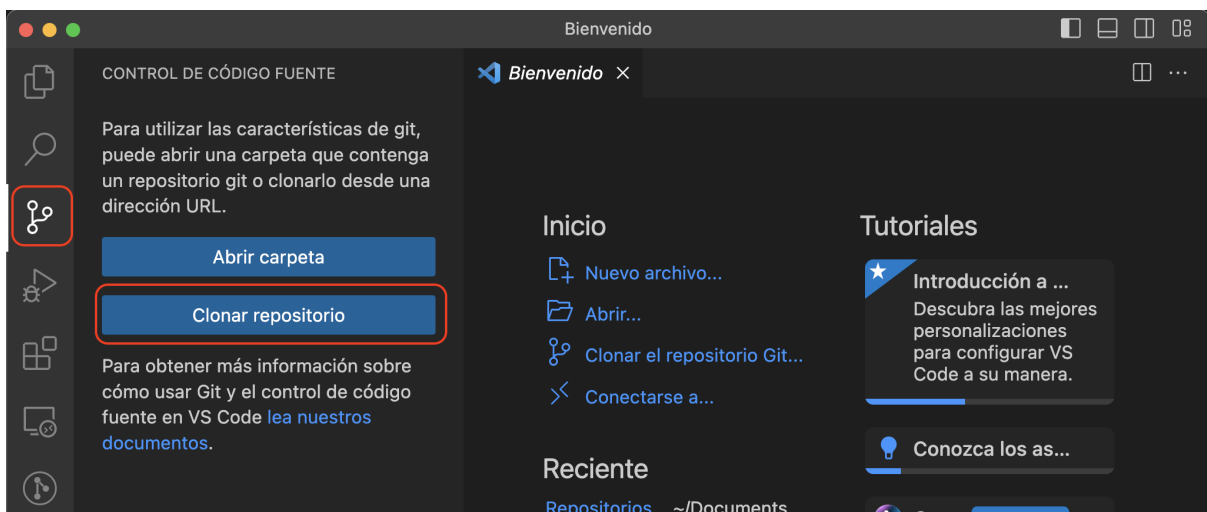
1. Primero vamos a ingresar al siguiente link que contiene el repositorio `api_productos_despliegue` de GitHub:

https://github.com/intro-des-a-backend/api_productos_despliegue

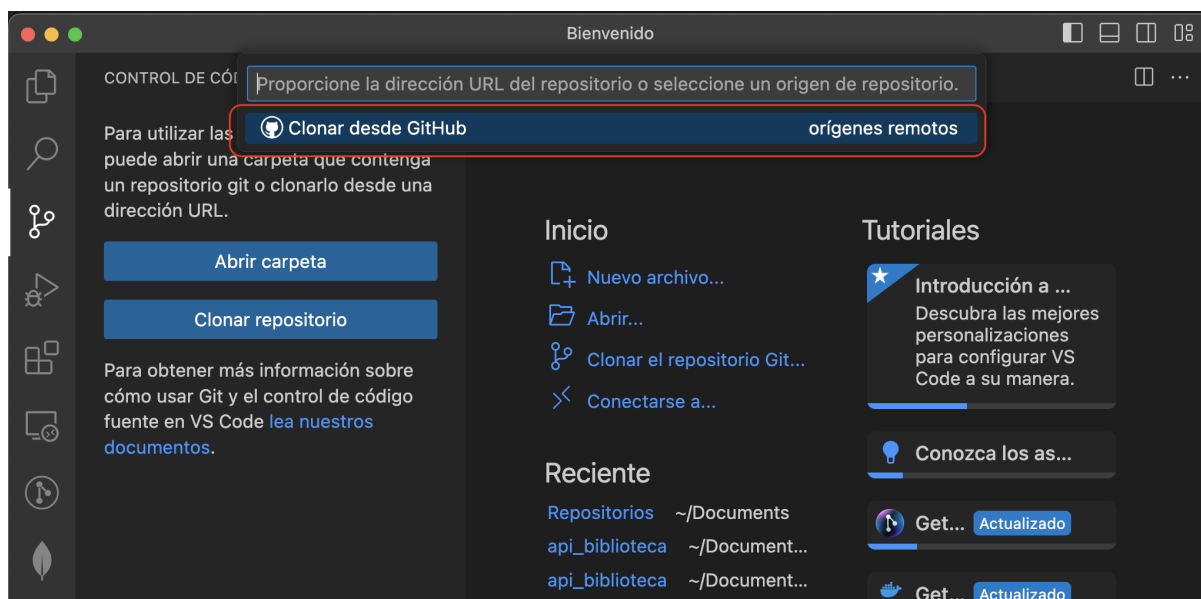
2. Vamos a crear un FORK de este repositorio para nuestra cuenta de GitHub. FORK en GitHub es una funcionalidad que permite a los usuarios hacer una copia de un repositorio ajeno en su cuenta de GitHub.



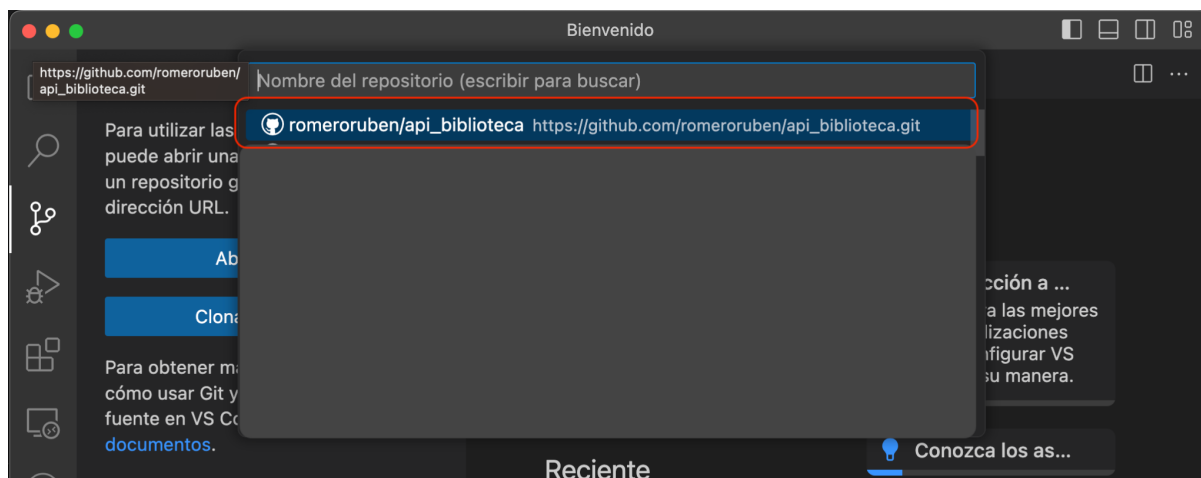
- Una vez que hicimos el FORK ya podemos clonar el repositorio localmente. Iniciamos la aplicación VSCode.
- Vamos a la opción de “Control de código fuente”, y seleccionamos la opción “Clonar repositorio” y seguimos los pasos:



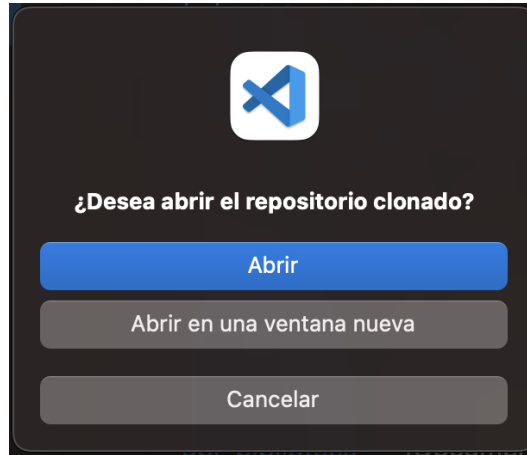
5. Hacemos click en “Clonar desde GitHub”



6. Seleccionamos el repositorio api_productos_despliegue que está en nuestra cuenta:



7. En el siguiente paso nos muestra el mensaje “Desea abrir el repositorio clonado?”, y hacemos click en **Abrir**



8. Ahora vamos a instalar los paquetes de npm localmente con el comando desde un terminal:

```
npm install
```

9. Debemos crear el archivo **.env** (de variables de entorno) con el siguiente contenido:

```
MONGO_URL=mongodb://mongodb:27017/empresa
OAUTH_AUDIENCE=http://localhost:3000/api/productos
OAUTH_URL=https://dev-utn-frc-iaew.auth0.com/
```

10. En el archivo package.json se definieron los scripts **start** (para iniciar la API) y **test** (para ejecutar los tests):

```
"scripts": {
  "start": "node src/app.js",
  "test": "jest"
},
```

11. Vamos a iniciar la API para probar que funcione todo bien:

```
npm start
```

12. Para probar la API necesitan crear un TOKEN, con el siguiente CURL, lo pueden importar desde Postman y crear un access_token:

```
curl --location 'https://dev-utn-frc-iaew.auth0.com/oauth/token' \
--header 'content-type: application/json' \
--data '{
  "client_id": "Qiw8AlH9oykBg7ofBrHs6ToYvrdmhOeE",
  "client_secret":
  "7kZPQqNnhRAuCXipSVSdHUsV9MzgfUzBB3AYbfemGPqxtpxI6j1GNxDBfYBSUume",
  "audience": "http://localhost:3000/api/productos",
  "grant_type": "client_credentials"
}'
```

Docker Compose

13. Docker Compose nos permite ejecutar nuestra aplicación en un contenedor docker localmente en nuestras máquinas. Es necesario antes de iniciar el proceso de despliegue confirmar que este funcionando correctamente ejecutando el siguiente comando:

```
docker-compose up
```

14. Con el siguiente comando podemos verificar que los contenedores se encuentran funcionando:

```
docker ps
```

15. Importando este CURL en Postman pueden probar que la API funcione ok, reemplazando el TOKEN.

```
curl --location 'http://127.0.0.1:3000/api/productos' \
--header 'Authorization: Bearer TOKEN'
```

Desplegar API en AWS

Configuración Local de AWS CLI

16. Es necesario para realizar las tareas de despliegue desde nuestras máquinas configurar el AWS CLI con las credenciales de AWS. Para esto vamos a crear si no existe el siguiente archivo:

- Linux / GitBash (Windows)
 - ~/.aws/credentials

17. Podemos abrir el archivo desde VSCode con el siguiente comando:

- Linux / GitBash (Windows)
 - code ~/.aws/credentials

18. Vamos a ingresar a la consola de AWS, vamos a copiar las credenciales y vamos a pegar el siguiente contenido dentro del archivo credentials

The screenshot shows the AWS Cloud Labs interface. At the top, there's a navigation bar with 'AWS' (highlighted with a red box), 'Used \$0 of \$100', '03:55', 'Start Lab', 'End Lab', 'AWS Details' (highlighted with a red box), 'Readme', 'Reset', and a close icon. The main area is split into two panes. The left pane shows a terminal window with the prompt 'eee_w_2269612@runweb88374:~\$'. The right pane is titled 'Cloud Access' and contains the following text:

AWS CLI:
Copy and paste the following into ~/.aws/credentials

```
[default]
aws_access_key_id=
aws_secret_access_key=
aws_session_token=
```

Below this, there's a 'Cloud Labs' section with session information: 'Remaining session time: 03:57:06(238 minutes)', 'Session started at: 2023-07-31T08:37:07-0700', 'Session to end at: 2023-07-31T12:37:07-0700', and 'Accumulated lab time: 04:02:00 (242 minutes)'. It also states 'No running instance'. There are buttons for 'Show', 'Download PEM', 'Download PPK', and 'Download URL'. At the bottom, there's a field for 'AWSAccountid' with the value '906339610862'.

19. Para verificar que todo quedo funcionando podemos ejecutar el siguiente comando, y verificar el resultado esperado:

```
aws sts get-caller-identity
```

Resultado Esperado (pero con los datos de su usuario):

```
{
  "UserId": "AROASGBQM6DXHBQTS7J3N:user2665987=Estudiante_de_prueba",
  "Account": "906339610862",
  "Arn":
  "arn:aws:sts:906339610862:assumed-role/voclabs/user2665987=Estudiante_d
  e_prueba"
}
```

Crear y publicar una Imagen Docker de nuestra API

20. Primero vamos a crear el repositorio en Amazon Elastic Container Registry para poder publicar nuestras imágenes Docker de manera privada:

```
aws ecr create-repository --repository-name api-productos --region
us-east-1
```

21. Vamos a ingresar a la consola de AWS haciendo click como se ve a continuación:

AWS

Used \$0 of \$100

03:55

Start Lab

End Lab

AWS Details

Readme

Reset

eee_w_2269612@runweb88374:~\$

Cloud Access

AWS CLI:

Copy and paste the following into ~/.aws/credentials

```
[default]
aws_access_key_id=
aws_secret_access_key=
aws_session_token=
```

Cloud Labs

Remaining session time: 03:57:06(238 minutes)

Session started at: 2023-07-31T08:37:07-0700

Session to end at: 2023-07-31T12:37:07-0700

Accumulated lab time: 04:02:00 (242 minutes)

No running instance

SSH key

Show

Download PEM

Download PPK

AWS SSO

Download URL

AWSAccountid

906339610862

22. Vamos a buscar el repositorio **api-productos** que creamos en AWS.

Amazon ECR > Repositorios

Private

Public

Repositorios privados (1)

Ver comandos de envío

Eliminar

Acciones

Crear repositorio

Buscar repositorios

	Nombre del repositorio	URI	Creado en	Inmutabilidad de etiqueta	Frecuencia de análisis	Tipo de cifrado	Caché de extracción
<input type="checkbox"/>	api-productos	906339610862.dkr.ecr.us-east-1.amazonaws.com/api-productos	24 de julio de 2023, 21:13:47 (UTC-03)	Desactivado	Manual	AES-256	Inactivo

23. Ingresamos al repositorio **api-productos**, y seleccionamos la opción “Ver comandos de envío”:

Amazon ECR > Repositorios > api-productos

api-productos

Ver comandos de envío

Editar

Imágenes (1)

Eliminar

Detalles

Analizar

Buscar imágenes

	Etiqueta de imagen	Tipo de artefacto	Enviado a	Tamaño (MB)	URI de imagen	Resumir	Estado del análisis	Vulnerabilidades
<input type="checkbox"/>	latest	Image	24 de julio de 2023, 21:24:52 (UTC-03)	401.05	<div>Copiar URI</div> <div>sha256:c4e2fe73015090f...</div>		-	-

24. Seleccionamos el sistema operativo que estamos usando y copiamos cada comando en el Terminal de VSCode. :

Comandos de envío para api-productos

macOS / Linux

Windows

Asegúrese de tener instalada la versión más reciente de AWS Tools para PowerShell y Docker. Para obtener más información, consulte [Empezar a utilizar Amazon ECR](#).

Siga los siguientes pasos a fin de autenticar y enviar una imagen a su repositorio. Para obtener métodos de autenticación de registro adicionales, incluido el ayudante de credenciales de Amazon ECR, consulte [Autenticación del registro](#).

1. Recupere un token de autenticación y autentique su cliente de Docker en el registro.

Utilice AWS Tools para PowerShell:

```
(Get-ECRLoginCommand).Password | docker login --username AWS --password-stdin 906339610862.dkr.ecr.us-east-1.amazonaws.com
```

2. Cree una imagen de Docker con el siguiente comando. Para obtener información sobre cómo crear un archivo de Docker desde cero, consulte las instrucciones [aquí](#). Puede omitir este paso si ya se creó la imagen:

```
docker build -t api-productos .
```

3. Cuando se complete la creación, etiquete la imagen para poder enviarla a este repositorio:

```
docker tag api-productos:latest 906339610862.dkr.ecr.us-east-1.amazonaws.com/api-productos:latest
```

4. Ejecute el siguiente comando para enviar esta imagen al repositorio de AWS recién creado:

```
docker push 906339610862.dkr.ecr.us-east-1.amazonaws.com/api-productos:latest
```

Cerrar

25. Luego de esto deberíamos ver nuestra versión de la imagen Docker de api-productos disponible en el repositorio de AWS.

Amazon ECR > Repositorios > api-productos

api-productos

Ver comandos de envío

Editar

Imágenes (1)

Buscar imágenes

<input type="checkbox"/>	Etiqueta de imagen	Tipo de artefacto	Enviado a	Tamaño (MB)	URI de imagen	Resumir	Estado del análisis	Vulnerabilidades
<input type="checkbox"/>	latest	Image	24 de julio de 2023, 21:24:52 (UTC-03)	401.05	Copiar URI	sha256:c4e2fe73015090f...	-	-

Crear el Cluster de Elastic Container Service

26. Primero vamos a crear la configuración de red con el siguiente comando, utilizando el terminal de VSCode y estando en la raíz del repositorio:

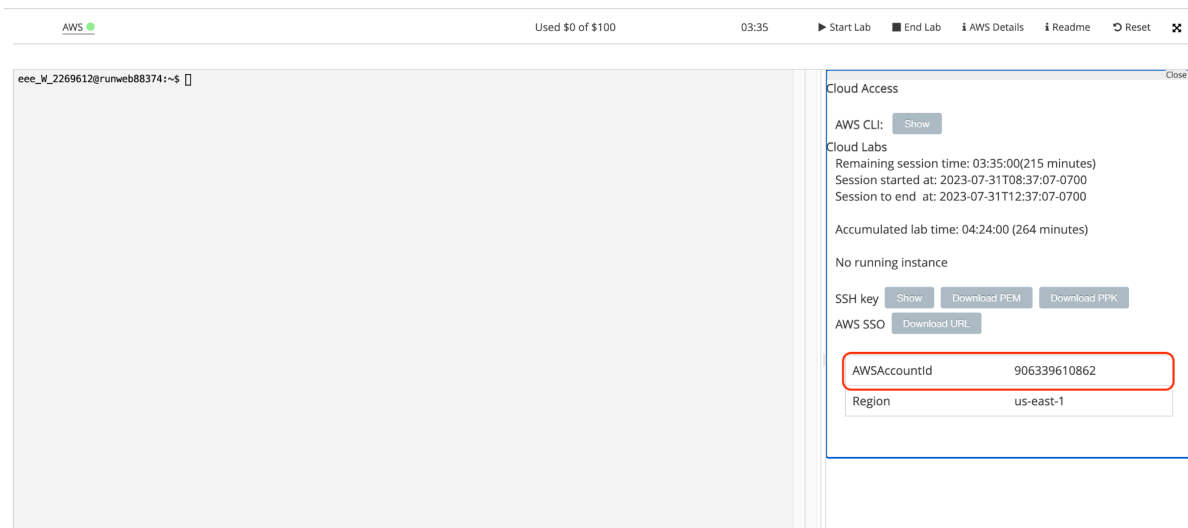
```
aws cloudformation create-stack --template-body
file://$PWD/infrastructure/vpc.yml --stack-name
api-productos-docker-ecs-vpc
```

27. Segundo vamos a crear el cluster con el siguiente comando:

```
aws cloudformation create-stack --template-body
file://$PWD/infrastructure/app-cluster.yml --stack-name
api-productos-docker-ecs-cluster
```

Finalmente! Vamos a desplegar nuestra API

28. En el archivo **infrastructure/api.yml** es necesario reemplazar el nro de cuenta de AWS, por el número de cuenta de cada alumno:



The screenshot shows a terminal window with the prompt `eee_w_2269612@runweb88374:~$`. On the right, the 'Cloud Access' panel is open, displaying the following information:

- AWS CLI:** [Show]
- Cloud Labs:**
 - Remaining session time: 03:35:00(215 minutes)
 - Session started at: 2023-07-31T08:37:07-0700
 - Session to end at: 2023-07-31T12:37:07-0700
- Accumulated lab time:** 04:24:00 (264 minutes)
- No running instance**
- SSH key:** [Show] [Download PEM] [Download PPK]
- AWS SSO:** [Download URL]
- AWSAccountid:** 906339610862
- Region:** us-east-1

```
api.yml — api_productos

! api.yml 9+ x
infraestructure > ! api.yml > {} Resources > {} Task > {} Properties > [aws] ExecutionRoleArn

6   Type: AWS::ECS::TaskDefinition
7   Properties:
8     Cpu: 1024
9     Memory: 2048
10    NetworkMode: awsvpc
11    RequiresCompatibilities:
12      - FARGATE
13    ExecutionRoleArn: 'arn:aws:iam::959309513507:role/LabRole'
14    ContainerDefinitions:
15      - Name: api-productos
16        Image: '959309513507.dkr.ecr.us-east-1.amazonaws.com/api-productos:latest'
17        Cpu: 512
```

29. Por último, vamos a indicarle al Cluster de ECS que utilice nuestra imagen Docker para desplegar los contenedores con el siguiente comando:

```
aws cloudformation create-stack --template-body
file://$PWD/infraestructure/api.yml --stack-name
api-productos-docker-ecs-api
```

La ejecución de estos comandos lleva tiempo, ¡tengan paciencia! jejeje. El resultado esperado del comando es una URL que nos indique cómo podemos consumir la API desde Postman.

30. IMPORTANTE! Para dar por finalizada la actividad compartir la url de la API desplegada en AWS en la actividad del aula virtual en <https://uve.frc.utn.edu.ar/>.