

## **Resolución del ejercicio práctico DevOps**

### Contenido

1. Dockerización de la aplicación. ....	2
2. Desarrollo del pipeline. ....	3
2.1 Diagrama general del pipeline CI/CD. ....	3
2.2 Configuraciones .....	3
2.2.1 .gitlab-ci.yml.....	4
2.2.2 . Configuración para el despliegue de Kubernetes .....	5
2.3 Resultados.....	6
3. Conclusiones. ....	9

## Practical Assessment Devsu -DevOps

Candidato: Alejandro Avalos | email: [avalos2904@outlook.com](mailto:avalos2904@outlook.com) | +591 78790200

### 1. Dockerización de la aplicación.

Se ha utilizado la aplicación desarrollada en **Python**: <https://bitbucket.org/devsu/demo-devops-python>

Repositorio donde se encuentran todos los ficheros generados para esta actividad: <https://github.com/ale0072/devsu01.git>

Para esto se utilizó un Dockerfile que contiene lo siguiente:

1. Se utilizó la imagen oficial de Python en la versión 3.11.3.
2. Se definieron variables de entorno Python para optimizar el tiempo de ejecución dentro del contenedor Docker.
3. Se definió el directorio de trabajo dentro del contenedor en la ruta `/app`.
4. Se copió el contenido del directorio de la aplicación dentro del contenedor en el directorio `/app`.
5. Se instalaron las dependencias especificadas en el archivo `requirements.txt`.
6. Se ejecutaron los comandos para aplicar la migración de base de datos específicas en el `README.md` de aplicación.
7. Se informó a Docker que el contenedor escuchara en el puerto 8000.
8. Se definió un *healthcheck* para el contenedor, usando `curl` para verificar la disponibilidad de la URL es cuestión.
9. Se especificó el comando por defecto para ejecutar cuando el contenedor se inicialice.

En resumen, este Dockerfile encapsula los pasos necesarios para configurar una imagen de Docker para una aplicación Django, incluida la configuración del entorno, copiar el código, instalar dependencias, aplicar migraciones de bases de datos, exponer un puerto, definir una verificación de estado e iniciar el servidor de desarrollo Django.

```
# Use an official Python runtime as a parent image
FROM python:3.11.3

# Set environment variables
ENV PYTHONDONTWRITEBYTECODE 1
ENV PYTHONUNBUFFERED 1

# Set the working directory to /app
WORKDIR /app

# Copy the current directory contents into the container at /app
COPY . /app

# Install dependencies
RUN pip install --no-cache-dir -r requirements.txt

# Run database migrations
RUN python manage.py makemigrations
RUN python manage.py migrate

# Expose the port that Django runs on
EXPOSE 8000

# Define healthcheck
HEALTHCHECK --interval=5m --timeout=3s \
| CMD curl -f http://localhost:$PORT/ || exit 1

# Start the Django development server
CMD ["python", "manage.py", "runserver", "0.0.0.0:8000"]
```

## Practical Assessment Devsu -DevOps

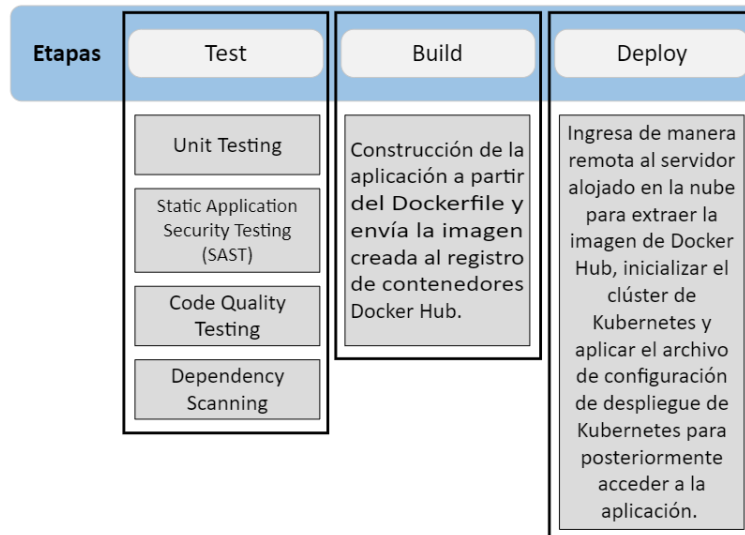
Candidato: Alejandro Avalos | email: [avalos2904@outlook.com](mailto:avalos2904@outlook.com) | +591 78790200

### 2. Desarrollo del pipeline.

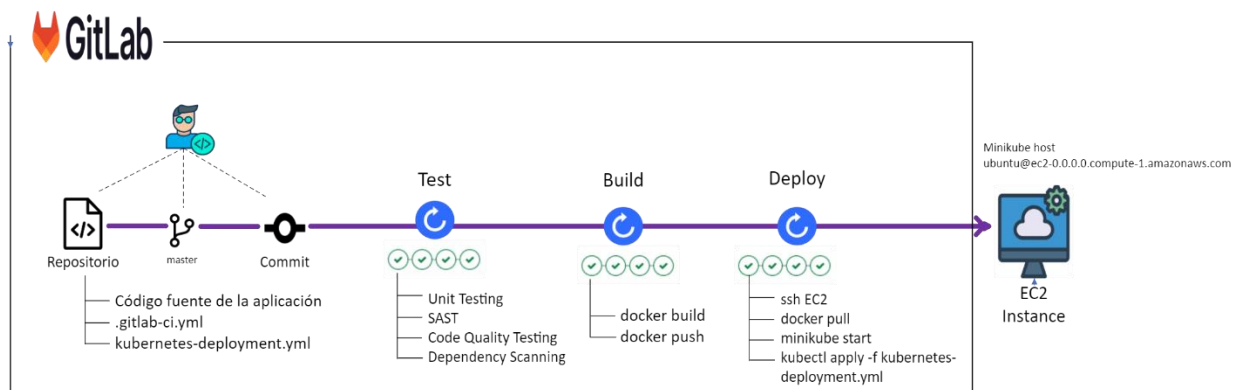
Para este caso, se desarrollo un pipeline como código utilizando la plataforma GitLab.

En el repositorio de GitLab, se alojaron tanto el código fuente de la aplicación, Dockerfile y el archivo de configuración de despliegue para Kubernetes.

Este pipeline consta de 3 etapas y ejecuta las siguientes tareas:



#### 2.1 Diagrama general del pipeline CI/CD.



#### 2.2 Configuraciones

Todos los archivos de configuración se encuentran disponibles en el siguiente repositorio de GitHub:

<https://github.com/ale0072/devsu>

## Practical Assessment Devsu -DevOps

Candidato: Alejandro Avalos | email: [avalos2904@outlook.com](mailto:avalos2904@outlook.com) | +591 78790200

### 2.2.1 .gitlab-ci.yml

A continuación, se describe cada una de las tareas implementadas en el pipeline CICD para alcanzar el despliegue de la aplicación en cuestión.

#### 1. stages:

- Define tres etapas: *test*, *build* y *deploy*. Las tareas se ejecutarán en estas etapas en secuencia.

#### 2. include:

- Incluye plantillas predefinidas de GitLab CI para realizar análisis de seguridad estática (SAST), calidad de código, y escaneo de dependencias. Además, la ejecución de las pruebas unitarias.

#### 3. variables:

- Define variables de entorno, como `IMAGE_NAME` y `IMAGE_TAG`, que se utilizan para construir y etiquetar la imagen Docker.

#### 4. run\_tests

- En la etapa *test*, utiliza la imagen base de Python 3.11.
- Instala las dependencias (`requirements.txt`) del proyecto antes de ejecutar los tests con `python manage.py test`.

#### 5. build\_image

- Para configurar esta etapa, se requiere tener Docker disponible en un contenedor Docker, también conocido como Docker en Docker. Se utiliza una imagen oficial de Docker que se adapta a este escenario, permitiendo construir un contenedor Docker con una imagen Docker. Esto garantiza que los comandos de Docker estén disponibles en este contenedor. GitLab también menciona el caso de uso Docker in Docker (dind), donde los GitLab Runners utilizan Docker executors para ejecutar tareas en imágenes Docker.

- Inicia un servicio Docker In Docker dentro del contenedor y realiza el *login* en un registro de contenedores.

- Construye una imagen Docker con el nombre y la etiqueta definidos en las variables y la sube al registro.

#### 6. deploy

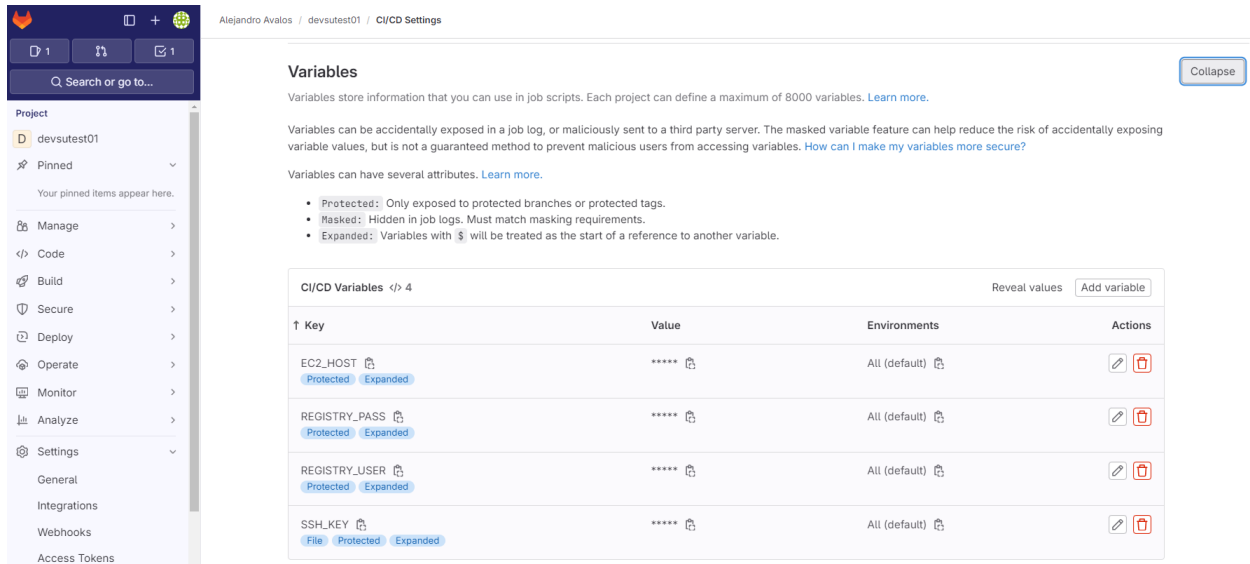
- En la etapa *deploy*, realiza algunas configuraciones previas.
- Utiliza una clave SSH para conectarse a un servidor EC2.
- En el servidor EC2, realiza un login en el registro de contenedores, descarga la imagen y la inicia con Kubernetes (utilizando Minikube en modo Docker).

En resumen, este archivo configura un pipeline que realiza pruebas, construye y sube una imagen Docker, y finalmente, despliega la aplicación en un entorno Kubernetes utilizando un servidor EC2 y Minikube.

## Practical Assessment Devsu -DevOps

Candidato: Alejandro Avalos | email: [avalos2904@outlook.com](mailto:avalos2904@outlook.com) | +591 78790200

Asimismo, como buena práctica de seguridad en GitLab se almacenaron todas las variables utilizadas en el archivo de configuración del pipeline, así como las credenciales de Docker Hub, la clave privada de la instancia EC2 y el nombre del host EC2. Esto fue configurado en CI/CD Settings de GitLab.



The screenshot shows the GitLab CI/CD Settings page for a project named 'devsute01'. The 'Variables' section is active, displaying a table of CI/CD Variables. The table has four columns: Key, Value, Environments, and Actions. There are four variables listed: EC2\_HOST, REGISTRY\_PASS, REGISTRY\_USER, and SSH\_KEY. Each variable is marked as 'Protected' and 'Expanded'. The values are masked with asterisks. The 'Environments' column shows 'All (default)' for all variables. The 'Actions' column contains edit and delete icons for each variable.

Key	Value	Environments	Actions
EC2_HOST	*****	All (default)	[Edit] [Delete]
REGISTRY_PASS	*****	All (default)	[Edit] [Delete]
REGISTRY_USER	*****	All (default)	[Edit] [Delete]
SSH_KEY	*****	All (default)	[Edit] [Delete]

### 2.2.2 . Configuración para el despliegue de Kubernetes

La configuración de despliegue de Kubernetes se lo realizo en el archivo `kubernetes-deployment.yml` en el cual se describen un conjunto de recursos que incluye lo siguiente:

- **Deployment:**  
Describe el estado deseado para desplegar pods.  
Replicas: 2 en este caso.  
Selector: Define las etiquetas utilizadas para seleccionar los pods controlados por este despliegue.  
Template: Describe los pods que se crearán.  
Containers: Define el contenedor Docker a ejecutar, incluyendo su nombre, imagen, puertos, y variable de entorno para la clave secreta de Django.
- **Service**  
Servicio: Expone el despliegue como un servicio.  
Selector: Selecciona los pods a exponer.  
Puertos: Mapea el puerto 80 en el servicio al puerto 8000 en los pods.  
Tipo: Especifica el tipo de servicio como LoadBalancer.
- **ConfigMap**  
Mapa de Configuración: Almacena datos de configuración.  
Datos: Contiene pares clave-valor para parámetros de configuración, incluyendo la clave secreta de Django y el nombre de la base de datos.
- **Secret**  
Clave Secreta: Almacena información sensible como contraseñas o tokens.  
Tipo: Especifica el tipo de secreto (Opaque para datos genéricos).  
Datos: Contiene valores codificados, como la clave secreta de Django codificada.
- **Horizontal Pod Autoscaler**  
Autoscaler Horizontal: Ajusta automáticamente el número de pods en un despliegue.

## Practical Assessment Devsu -DevOps

Candidato: Alejandro Avalos | email: [avalos2904@outlook.com](mailto:avalos2904@outlook.com) | +591 78790200

scaleTargetRef: Se refiere al despliegue a escalar.

minReplicas: Número mínimo de réplicas (2).

maxReplicas: Número máximo de réplicas (5).

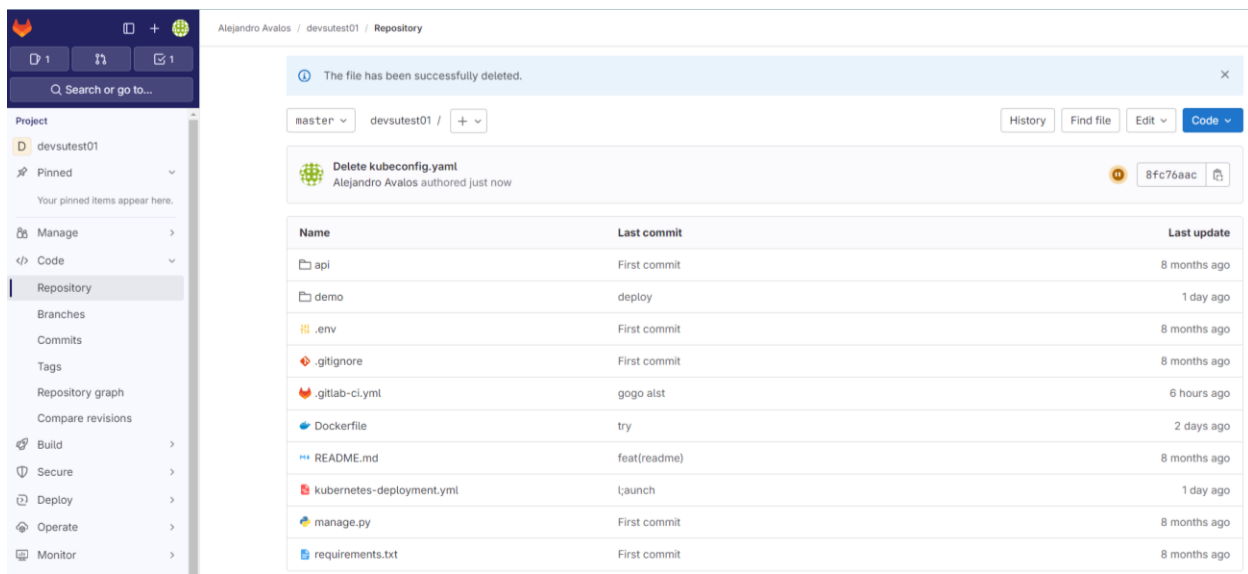
Métricas: Configura el escalado automático basado en la utilización de la CPU (50% de utilización promedio).

Este archivo YAML define un conjunto de recursos de Kubernetes para desplegar la aplicación Django con configuraciones, servicios, secretos y capacidades de autoescalado especificados.

## 2.3 Resultados

1. El proyecto de GitLab se encuentra disponible en el siguiente enlace:

<https://gitlab.com/ale0072/devsutest01.git>



Alejandro Avalos / devsutest01 / Repository

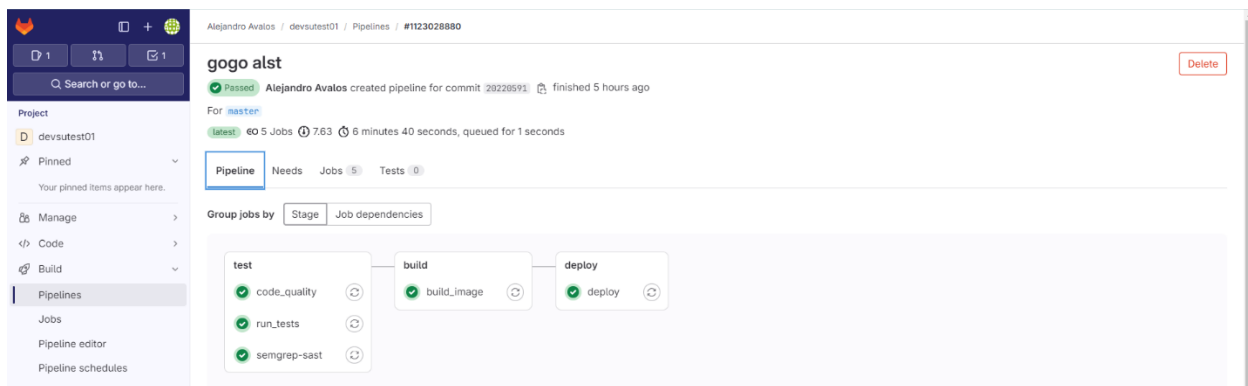
The file has been successfully deleted.

master devsutest01

Delete kubeconfig.yaml  
Alejandro Avalos authored just now

Name	Last commit	Last update
api	First commit	8 months ago
demo	deploy	1 day ago
.env	First commit	8 months ago
.gitignore	First commit	8 months ago
.gitlab-ci.yml	gogo alst	6 hours ago
Dockerfile	try	2 days ago
README.md	feat(readme)	8 months ago
kubernetes-deployment.yml	launch	1 day ago
manage.py	First commit	8 months ago
requirements.txt	First commit	8 months ago

2. Ejecución exitosa del pipeline CI/CD, como se observa en la captura, todas las tareas definidas en cada etapa han sido exitosas y la última ejecución del pipeline (ID= 20220591) tomo 6 minutos y 40 segundos en completarse.



Alejandro Avalos / devsutest01 / Pipelines / #1123028880

**gogo alst** Delete

Passed Alejandro Avalos created pipeline for commit 20220591 finished 5 hours ago

For master

latest 60 5 Jobs 7.63 6 minutes 40 seconds, queued for 1 seconds

Pipeline Needs Jobs 5 Tests 0

Group jobs by Stage Job dependencies

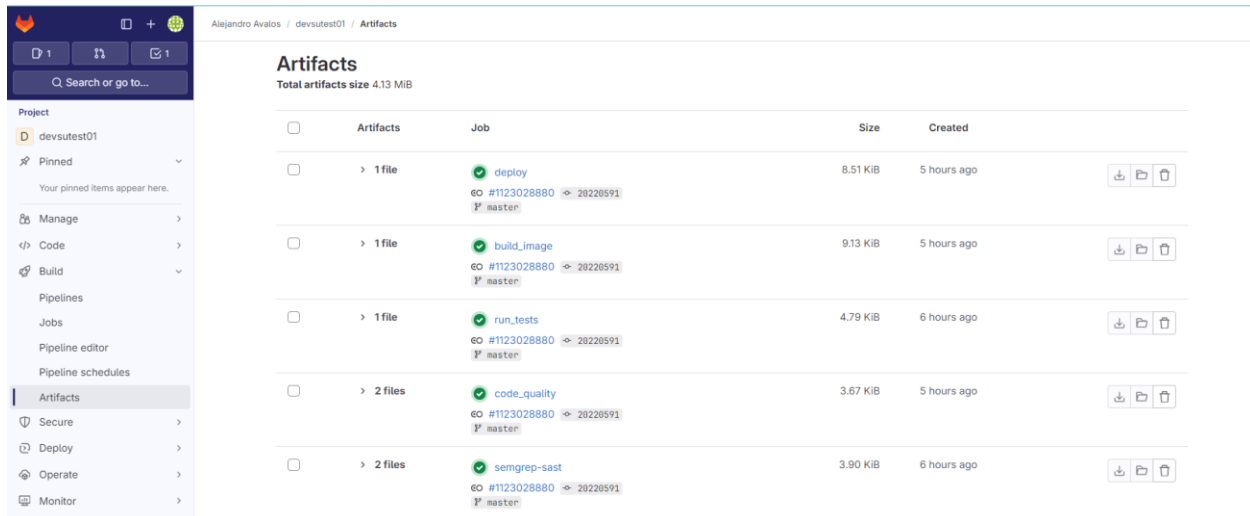
test build deploy

- code\_quality
- run\_tests
- semgrep-sast
- build\_image
- deploy

## Practical Assessment Devsu -DevOps

Candidato: Alejandro Avalos | email: [avalos2904@outlook.com](mailto:avalos2904@outlook.com) | +591 78790200

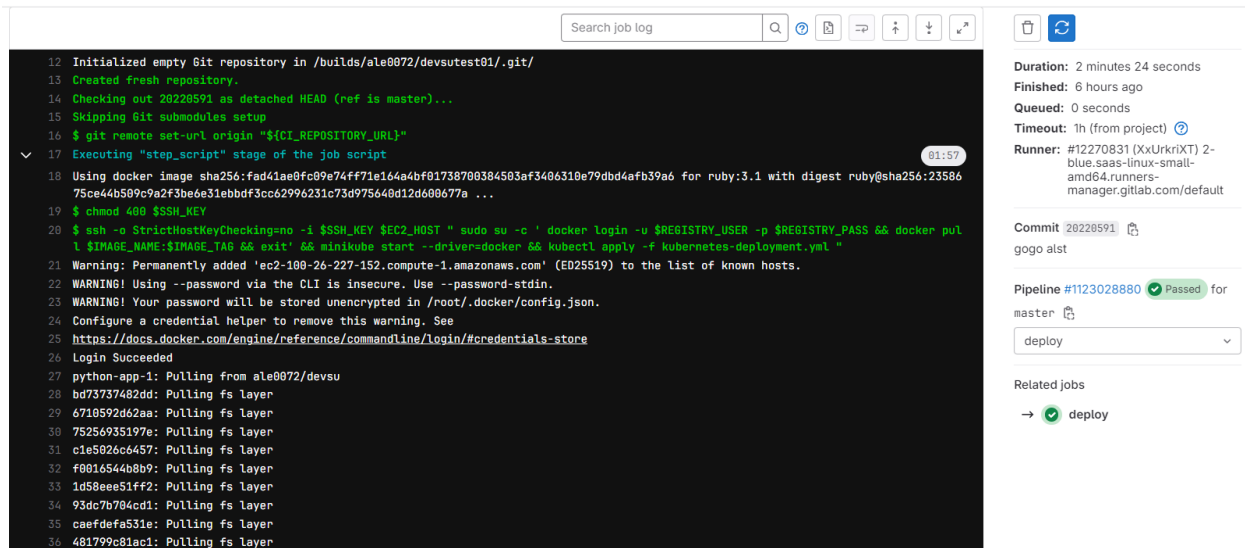
En esta esta captura se observan los artefactos de cada tarea que han sido generados tras la ejecución del pipeline. Estos artefactos incluyen los archivos de registros de cada tarea y en el caso de los mecanismos de pruebas y seguridad incluye en archivo JSON con el resumen de lo encontrado.



Project	Artifacts	Job	Size	Created
devstest01	1 file	deploy GO #1123028880 -> 20220591 P master	8.51 KiB	5 hours ago
	1 file	build_image GO #1123028880 -> 20220591 P master	9.13 KiB	5 hours ago
	1 file	run_tests GO #1123028880 -> 20220591 P master	4.79 KiB	6 hours ago
	2 files	code_quality GO #1123028880 -> 20220591 P master	3.67 KiB	5 hours ago
	2 files	semgrep-sast GO #1123028880 -> 20220591 P master	3.90 KiB	6 hours ago

Este es una captura de los registros de la etapa de despliegue del pipeline, en este caso, se observa específicamente como accede a la instancia EC2 y ejecuta todas las instrucciones definidas en la tarea. En el archivo comprimido a entregar adjuntare todos los archivos de registros generados en cada tarea (Pruebas Unitarias, SAST, Code Quality, Build y Deploy).

Alejandro Avalos / devstest01 / Jobs / #5839044943



```
12 Initialized empty Git repository in /builds/ale0072/devstest01/.git/
13 Created fresh repository.
14 Checking out 20220591 as detached HEAD (ref is master)...
15 Skipping Git submodules setup
16 $ git remote set-url origin "${CI_REPOSITORY_URL}"
17 Executing "step_script" stage of the job script
18 Using docker image sha256:fad41ae9fc09e74ff71e164a4bf01738708384503af3406310e79dbd4afb39a6 for ruby:3.1 with digest ruby@sha256:2358675ce44b509c9a2f3be6e31ebbf3cc62996231c73d975640d12d60677a ...
19 $ chmod 400 $SSH_KEY
20 $ ssh -o StrictHostKeyChecking=no -i $SSH_KEY $EC2_HOST "sudo su -c 'docker login -u $REGISTRY_USER -p $REGISTRY_PASS && docker pull $IMAGE_NAME:$IMAGE_TAG && exit' && minikube start --driver=docker && kubectl apply -f kubernetes-deployment.yml"
21 Warning: Permanently added 'ec2-100-26-227-152.compute-1.amazonaws.com' (ED25519) to the list of known hosts.
22 WARNING! Using --password via the CLI is insecure. Use --password-stdin.
23 WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
24 Configure a credential helper to remove this warning. See
25 https://docs.docker.com/engine/reference/commandline/login/#credentials-store
26 Login Succeeded
27 python-app-1: Pulling from ale0072/devsu
28 bd73737482dd: Pulling fs layer
29 6710592d42aa: Pulling fs layer
30 75256935197e: Pulling fs layer
31 c1e5026c6457: Pulling fs layer
32 f0816544b8b9: Pulling fs layer
33 1d58eee51ff2: Pulling fs layer
34 93dc7b704cd1: Pulling fs layer
35 caefdefa531e: Pulling fs layer
36 481799c81ac1: Pulling fs layer
```

Duration: 2 minutes 24 seconds  
Finished: 6 hours ago  
Queued: 0 seconds  
Timeout: 1h (from project)  
Runner: #12270831 (XxUrkiXT) 2-blue.saas-linux-small-amd64.runners-manager.gitlab.com/default  
Commit 20220591  
gogo alst  
Pipeline #1123028880 Passed for master  
deploy  
Related jobs  
→ Passed deploy

2. En la siguiente captura, se observa que desde la consola de la instancia EC2 que el clúster de minikube ha sido creado y se encuentra en ejecución (`kubectl cluster-info`). Además, se muestra que el despliegue realizado se encuentra disponible (`kubectl get deployments`), asimismo, los pods se encuentra corriendo (`kubectl get pods`), y al ejecutar el comando `minikube service demo-devops-python-service` se obtiene la URL para acceder a la aplicación.

## Practical Assessment Devsu -DevOps

Candidato: Alejandro Avalos | email: [avalos2904@outlook.com](mailto:avalos2904@outlook.com) | +591 78790200

```
aws Services Search [Alt+S] N. Virginia ale0072
ubuntu@ip-172-31-59-183:~$ kubectl cluster-info
Kubernetes control plane is running at https://192.168.49.2:8443
CoreDNS is running at https://192.168.49.2:8443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
ubuntu@ip-172-31-59-183:~$ kubectl get deployments
NAME                                READY  UP-TO-DATE  AVAILABLE  AGE
demo-devops-python-deployment      2/2    2           2          6h35m
ubuntu@ip-172-31-59-183:~$ kubectl get pods
NAME                                READY  STATUS      RESTARTS  AGE
demo-devops-python-deployment-79dbf7f5b7-fmclp  1/1    Running    0         6h35m
demo-devops-python-deployment-79dbf7f5b7-qnb9s  1/1    Running    0         6h35m
ubuntu@ip-172-31-59-183:~$ minikube service demo-devops-python-service
|-----|
| NAMESPACE | NAME                | TARGET PORT | URL                    |
|-----|
| default   | demo-devops-python-service | 80          | http://192.168.49.2:31928 |
|-----|
* Opening service default/demo-devops-python-service in default browser...
http://192.168.49.2:31928
ubuntu@ip-172-31-59-183:~$
```

i-02429879aeca42208 (devsu-host)

Enseguida, se abre el navegador ingresando a la aplicación a través de la URL obtenida de Minikube donde se creó algunos usuarios.

The screenshot shows a web browser with the URL `192.168.49.2:31928/api/`. The page displays the Django REST framework API Root. The "Api Root" section shows the default basic root view for DefaultRouter. The "GET /api/" endpoint is highlighted, showing the response: `HTTP 200 OK`, `Allow: GET, HEAD, OPTIONS`, `Content-Type: application/json`, and `Vary: Accept`. The response body is `{ "users": "http://192.168.49.2:31928/api/users/" }`.

The browser then navigates to the `192.168.49.2:31928/api/users/` endpoint. The page displays the "User List" section. The "GET /api/users/" endpoint is highlighted, showing the response: `HTTP 200 OK`, `Allow: GET, POST, HEAD, OPTIONS`, `Content-Type: application/json`, and `Vary: Accept`. The response body is `[ { "id": 1, "dni": "100500", "name": "Alejandro Avalos" }, { "id": 2, "dni": "001", "name": "devsutest" } ]`.

Below the response, there is a form to create a new user. It has two input fields: "Dni" and "Name". There is a "POST" button to submit the form. The form is currently empty.



### 3. Conclusiones.

- Se ha logrado Dockerizar la aplicación, con lo que ha sido posible desplegar la aplicación en un entorno local o de nube. En el Dockerfile se definieron variables de entorno para optimizar la ejecución del mismo, se expuso un puerto para el contenedor y se definio un healthcheck para verificar la disponibilidad del contenedor.
- Se ha logrado configurar un pipeline utilizando la plataforma GitLab, en el que se han configurado las etapas de construcción de la aplicación, pruebas (Pruebas Unitarias, Análisis estático, Calidad de código y escaneo de dependencias) y finalmente la etapa de *Deploy* donde se realiza hasta el despliegue de Kubernetes en una instancia EC2, la cual dispone de sistema operativo Ubuntu 22.04, Docker, Minikube y kubectl. No se agregaron otros escáneres de vulnerabilidades puesto a que en esta ocasión no disponía de la versión Ultimate de GitLab que habilita todas las características de seguridad de GitLab, así como *dashboards* o más escáneres. Ninguna de las pruebas ejecutadas encontró algún error durante la ejecución y tampoco vulnerabilidades. Hubiese habilitado el escaneo de contenedores puesto que se trata de una aplicación Dockerizada pero no disponía la licencia Ultimate de GitLab.
- La intención inicial era emplear Terraform para implementar Kubernetes, ya sea en Google Cloud Platform mediante (GKE) o en Amazon Web Services (AWS) mediante (EKS). Sin embargo, mis cuentas personales ya habían excedido los límites de recursos gratuitos, por lo que, en esta ocasión opté por no proceder con el despliegue en la nube y en su lugar, desplagué Kubernetes sobre una instancia EC2 de AWS utilizando Minikube.
- Además de todos los archivos generados para realizar esta actividad, adjunto en el repositorio de compartido de GitHub, los artefactos sobre logs y resultados de las pruebas y escaneo de vulnerabilidades ejecutadas.

<https://github.com/ale0072/devsu>

<https://gitlab.com/ale0072/devsutest01>