

Estudiante: Luis Fernando Arana Arias
Maestría: Desarrollo y operaciones de software
Curso: Contenedores

Creación de aplicativos en contenedores con diferentes lenguajes de programación

Este documento expone el desarrollo y despliegue de cuatro aplicaciones web utilizando diferentes tecnologías: Python con Flask, Node.js con Express, Go con Net, y Rust con Actix. El proyecto ilustra la contenerización mediante Docker para facilitar el despliegue en diversos entornos, destacando la eficiencia de combinar múltiples lenguajes y frameworks en un flujo de trabajo cohesivo y escalable.

Índice

Creación de aplicativos en contenedores con diferentes lenguajes de programación.....	1
Índice.....	1
Desarrollo.....	1
Selección del proyecto.....	1
Ejercicio 1. Dockerfiles para cada proyecto.....	2
Construcción de imágenes.....	2
Python.....	2
Nodejs.....	2
Go.....	3
Rust.....	3
Ejercicio 2. Cargar imágenes a Docker Hub.....	4
Ejercicio 3. Docker Compose.....	5

Desarrollo

Selección del proyecto

La elección de este proyecto se fundamenta en la necesidad de explorar y comparar el rendimiento y la escalabilidad de distintas tecnologías en el desarrollo de aplicaciones web. Se ha optado por Python, Node.js, Go y Rust debido a sus características únicas en cuanto a manejo de concurrencia, eficiencia de recursos y comunidad de soporte. La decisión también se ve motivada por la tendencia actual en la industria hacia la contenerización y la automatización de despliegues, aspectos cruciales para mantener la competitividad en el desarrollo de software moderno.

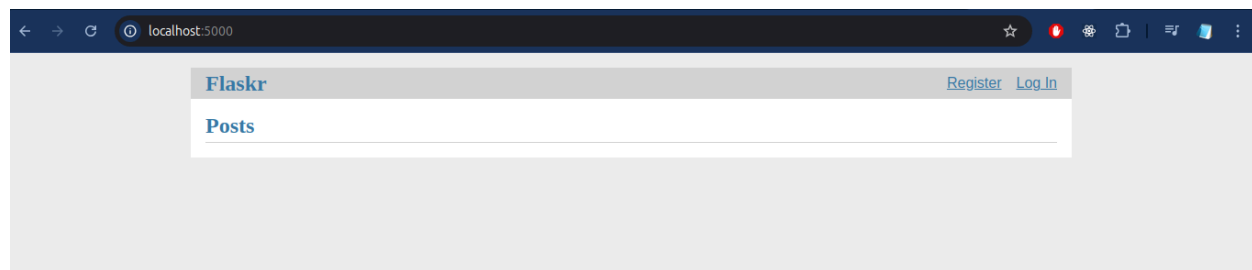
Ejercicio 1. Dockerfiles para cada proyecto

Construcción de imágenes

Python

```
docker build -t flask-app:latest ./python
```

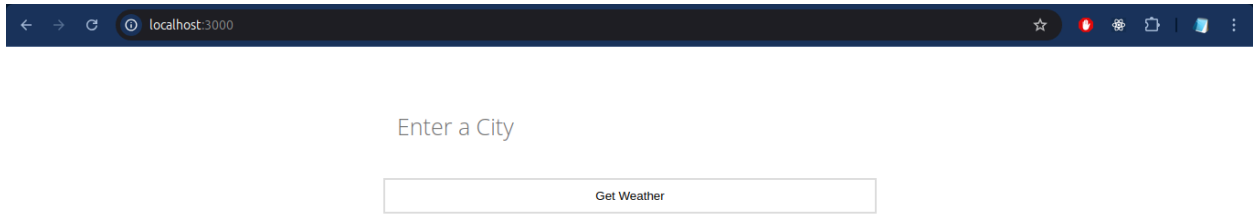
```
luis@luis-VivoBook-S15-X510UF:~/Desktop/Personal/Maestria Devops/Contenedores/Tarea 1$ docker run -d -p 5000:5000 --name mi-fla
sk-app flask-app
fdf9b4349c2ddbb220c377bddc662bd54c0ad99b38815753438eb4780fd9b494
luis@luis-VivoBook-S15-X510UF:~/Desktop/Personal/Maestria Devops/Contenedores/Tarea 1$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS
NAMES
fdf9b4349c2d   flask-app "flask run --host=0.0.0.0" 3 seconds ago  Up 2 seconds  0.0.0.0:5000->5000/tcp, :::5000->5000/tcp
mi-flask-app
luis@luis-VivoBook-S15-X510UF:~/Desktop/Personal/Maestria Devops/Contenedores/Tarea 1$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS
NAMES
fdf9b4349c2d   flask-app "flask run --host=0.0.0.0" 6 seconds ago  Up 6 seconds  0.0.0.0:5000->5000/tcp, :::5000->5000/tcp
mi-flask-app
luis@luis-VivoBook-S15-X510UF:~/Desktop/Personal/Maestria Devops/Contenedores/Tarea 1$
```



Nodejs

```
docker build -t express-app:latest ./nodejs
```

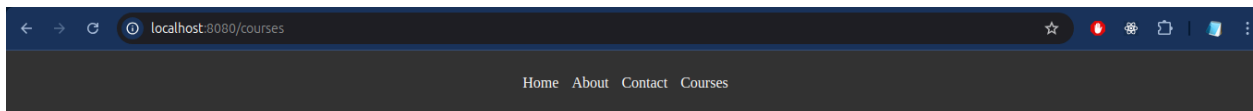
```
luis@luis-VivoBook-S15-X510UF:~/Desktop/Personal/Maestria Devops/Contenedores/Tarea 1$ docker run -d -p 3000:3000 --name mi-exp
ress-app express-app
ea4df40d2872d62be3aed3ba9c0ae1d6f9b9782f78579c1078c10e2b1f5df0
luis@luis-VivoBook-S15-X510UF:~/Desktop/Personal/Maestria Devops/Contenedores/Tarea 1$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS
NAMES
ea4df40d2872   express-app "docker-entrypoint.s..." 2 seconds ago  Up 1 second   0.0.0.0:3000->3000/tcp, :::3000->3000/tcp
mi-express-app
luis@luis-VivoBook-S15-X510UF:~/Desktop/Personal/Maestria Devops/Contenedores/Tarea 1$ docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
express-app    latest    2cd85c5e4ec0   40 seconds ago  921MB
flask-app      latest    48e2c538b0df   10 minutes ago  232MB
rhinosecuritylabs/pacu latest    712899711a4e   8 months ago   375MB
luis@luis-VivoBook-S15-X510UF:~/Desktop/Personal/Maestria Devops/Contenedores/Tarea 1$
```



Go

```
docker build -t net-app:latest ./go
```

```
• Luis@luis-VivoBook-S15-X510UF:~/Desktop/Personal/Maestria Devops/Contenedores/Tarea 1$ docker run -d -p 8080:8080 --name mi-net-app net-app
a4a06e1237b463de47b1e4b205726b58bdacf1ef5b65f7ee128fb04087d7b077
• Luis@luis-VivoBook-S15-X510UF:~/Desktop/Personal/Maestria Devops/Contenedores/Tarea 1$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS
a4a06e1237b4   net-app        "go run main.go"        2 seconds ago Up 2 seconds  0.0.0.0:8080->8080/tcp, :::8080->8080/tcp
p_mi-net-app    ea4df40d2872   express-app             "docker-entrypoint.s..." 16 minutes ago Up 13 minutes  0.0.0.0:3000->3000/tcp, :::3000->3000/tcp
p_mi-express-app fd9b4349c2d    flask-app               "flask run --host=0.0.0.0" 25 minutes ago Up 13 minutes  0.0.0.0:5000->5000/tcp, :::5000->5000/tcp
p_mi-flask-app
• Luis@luis-VivoBook-S15-X510UF:~/Desktop/Personal/Maestria Devops/Contenedores/Tarea 1$ docker images
REPOSITORY          TAG          IMAGE ID          CREATED          SIZE
net-app              latest       9a9f02467174     About a minute ago 328MB
express-app          latest       2cd85c5e4ec0     17 minutes ago   921MB
flask-app             latest       48e2c538b0df     27 minutes ago   232MB
rhinosecuritylabs/pacu latest       712899711a4e     8 months ago     375MB
```



Learn DevOps from Basics

DevOps is a set of practices that combines software development (Dev) and IT operations (Ops)

It aims to shorten the systems development life cycle and provide continuous delivery with high software quality. DevOps is complementary with Agile software development; several DevOps aspects came from Agile methodology

Main Tutorials

[DevOps Zero to Hero](#)

[AWS Zero to Hero](#)

[Azure Zero to Hero](#)

[Terraform Zero to Hero](#)

Rust

```
docker build -t rocket-app:latest ./rust
```

```

• luis@luis-VivoBook-S15-X510UF:~/Desktop/Personal/Maestria Devops/Contenedores/Tarea 1$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS
73ac2f580649   rocket-app     "cargo run --release"    2 minutes ago Up 2 minutes  0.0.0.0:8000->8000/tcp, :::8000->8000/tcp
mi-rocket-app

• luis@luis-VivoBook-S15-X510UF:~/Desktop/Personal/Maestria Devops/Contenedores/Tarea 1$ docker logs 73ac2f580649
Finished `release` profile [optimized] target(s) in 0.17s
Running `target/release/rocketapp`
Rocket has launched from http://127.0.0.1:8000

• luis@luis-VivoBook-S15-X510UF:~/Desktop/Personal/Maestria Devops/Contenedores/Tarea 1$ docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
rocket-app     latest    3a8fe9fb2494   8 minutes ago  2.09GB
net-app        latest    9a9f02467174   2 hours ago    328MB
express-app    latest    2cd85c5e4ec0   2 hours ago    921MB
flask-app      latest    48e2c538b0df   2 hours ago    232MB
rhinosecuritylabs/pacu latest    712899711a4e   8 months ago   375MB
o luis@luis-VivoBook-S15-X510UF:~/Desktop/Personal/Maestria Devops/Contenedores/Tarea 1$

```

Ejercicio 2. Cargar imágenes a Docker Hub

1. Realizar login para poder subir imágenes

docker login

```

• luis@luis-VivoBook-S15-X510UF:~/Desktop/Personal/Maestria Devops/Contenedores/Tarea 1$ docker login
Log in with your Docker ID or email address to push and pull images from Docker Hub. If you don't have a Docker ID, head over to
https://hub.docker.com/ to create one.
You can log in with your password or a Personal Access Token (PAT). Using a limited-scope PAT grants better security and is required
for organizations using SSO. Learn more at https://docs.docker.com/go/access-tokens/

Username: blocnotas
Password:
WARNING! Your password will be stored unencrypted in /home/luis/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credential-stores

Login Succeeded

```

2. Etiquetar la imagen a subir

```

docker tag rocket-app blocnotas/contenedores-unir-actividad-1:rocket-lts
docker tag net-app blocnotas/contenedores-unir-actividad-1:net-lts
docker tag express-app blocnotas/contenedores-unir-actividad-1:express-lts
docker tag flask-app blocnotas/contenedores-unir-actividad-1:flask-lts

```

3. Subir las imágenes



```

docker push blocnotas/contenedores-unir-actividad-1:rocket-lts
docker push blocnotas/contenedores-unir-actividad-1:net-lts
docker push blocnotas/contenedores-unir-actividad-1:express-lts
docker push blocnotas/contenedores-unir-actividad-1:flask-lts

```

blocnotas/contenedores-unir-actividad-1









Last pushed 1 minute ago

Add a description   INCOMPLETE

Add a category   INCOMPLETE

Tags

This repository contains 4 tag(s).

Tag	OS	Type	Pulled	Pushed
 flask-lts		Image	---	a minute ago
 express-lts		Image	2 minutes ago	2 minutes ago
 net-lts		Image	2 minutes ago	3 minutes ago
 rocket-lts		Image	3 minutes ago	4 minutes ago

Ejercicio 3. Docker Compose

Servicio app (Frontend JavaScript)

- build:
 - context:
Directorio que contiene los Dockerfiles, especificado como ./dockerfiles.
 - dockerfile:
Nombre del Dockerfile para construir este servicio, especificado como javascript.dockerfile.
- ports:
 - "8081:8081":
Mapea el puerto 8081 del host al puerto 8081 del contenedor, usado por la aplicación frontend.
- volumes:
 - ./frontend:/var/www/html:
Monta el directorio local ./frontend en /var/www/html dentro del contenedor para permitir la edición en vivo del código sin necesidad de reconstruir el contenedor.
- labels:
 - "traefik.http.routers.site.rule=Host(app.localhost)":

Configura Traefik para dirigir el tráfico a este contenedor cuando se accede a app.localhost.

- depends_on:
 - Dependencias hacia los servicios traefik y api, asegurando que estos servicios estén corriendo antes de iniciar app.

Servicio mysql (Base de Datos MariaDB)

- image:
 - Utiliza la imagen mariadb:10.6.
- restart:
 - Configurado para reiniciar automáticamente a menos que sea detenido manualmente (unless-stopped).
- tty:
 - Habilita TTY.
- ports:
 - "3306:3306":
Mapea el puerto 3306 del host al puerto 3306 del contenedor, utilizado por MySQL.
- environment:
 - Variables de entorno para configurar la base de datos.
 - MYSQL_DATABASE: Nombre de la base de datos (laravel).
 - MYSQL_USER: Usuario de la base de datos (laravel).
 - MYSQL_PASSWORD: Contraseña del usuario (secret).
 - MYSQL_ROOT_PASSWORD: Contraseña del usuario root (secret).
 - SERVICE_TAGS: Etiqueta del servicio (dev).
 - SERVICE_NAME: Nombre del servicio (mysql).

Servicio api (Backend PHP)

- build:
 - context:
Directorio que contiene los Dockerfiles, especificado como ./dockerfiles.
 - dockerfile:
Nombre del Dockerfile para construir este servicio, especificado como php.dockerfile.
- ports:
 - "8082:8082":
Mapea el puerto 8082 del host al puerto 8082 del contenedor, usado por la API backend.
- volumes:
 - ./backend:/var/www/html:
Monta el directorio local ./backend en /var/www/html dentro del contenedor.
- labels:
 - "traefik.http.routers.api.rule=(Host(app.localhost) && PathPrefix(/api))":
Configura Traefik para dirigir tráfico a este contenedor cuando se acceda a app.localhost/api.

- depends_on:
 - Dependencias hacia los servicios traefik y mysql, asegurando que estos servicios están corriendo antes de iniciar api.

Servicio traefik (Proxy Inverso)

- image:
 - Utiliza la imagen traefik:v2.9.
- command:
 - Comandos para configurar Traefik.
 - --api.insecure=true: Habilita la API de Traefik sin seguridad.
 - --providers.docker: Usa Docker como proveedor.
- ports:
 - "80:80": Mapea el puerto 80 del host al puerto 80 del contenedor, usado para tráfico HTTP.
 - "8080:8080": Mapea el puerto 8080 del host al puerto 8080 del contenedor, usado para el dashboard de Traefik.
- volumes:
 - /var/run/docker.sock:/var/run/docker.sock: Monta el socket de Docker para permitir a Traefik interactuar con la API de Docker.