

# DESARROLLO DE APLICACIONES UTILIZANDO PYTHON Y ORQUESTACIÓN DE CONTENEDORES DOCKER

Universidad de Costa Rica

Instructor: Jose Pablo Ramírez Méndez

Semana I

# AGENDA

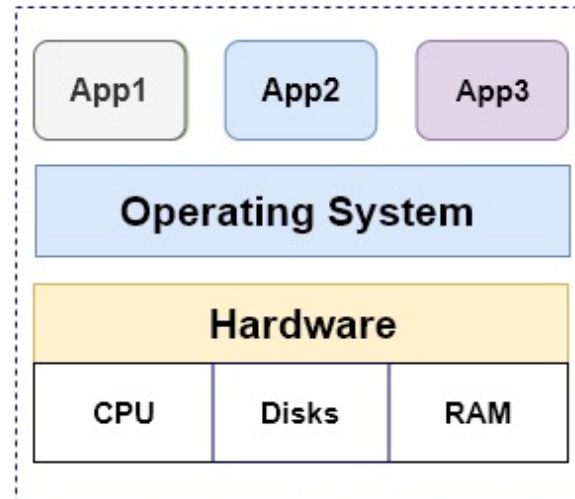
- Objetivos
- Introducción a los contenedores
- Docker containers
- Caso de estudio

# OBJETIVOS

- Entender el concepto de contenedores y su origen
- Conocer la utilidad los contenedores en el levantamiento y despliegue de aplicaciones.
- Aplicar el uso de contenedores en un caso de estudio de una aplicación en Python.

# INTRODUCCIÓN A LOS CONTENEDORES

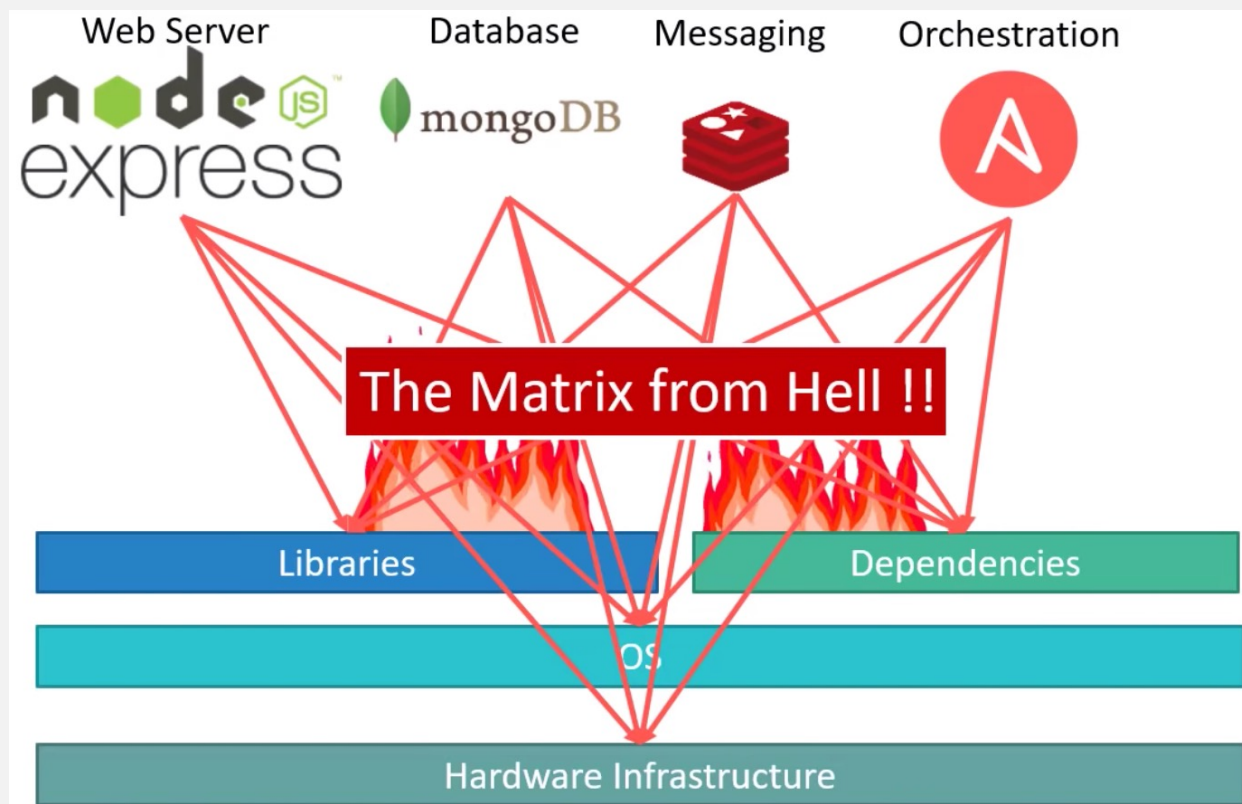
# ARQUITECTURA TRADICIONAL



**Traditional Deployment**

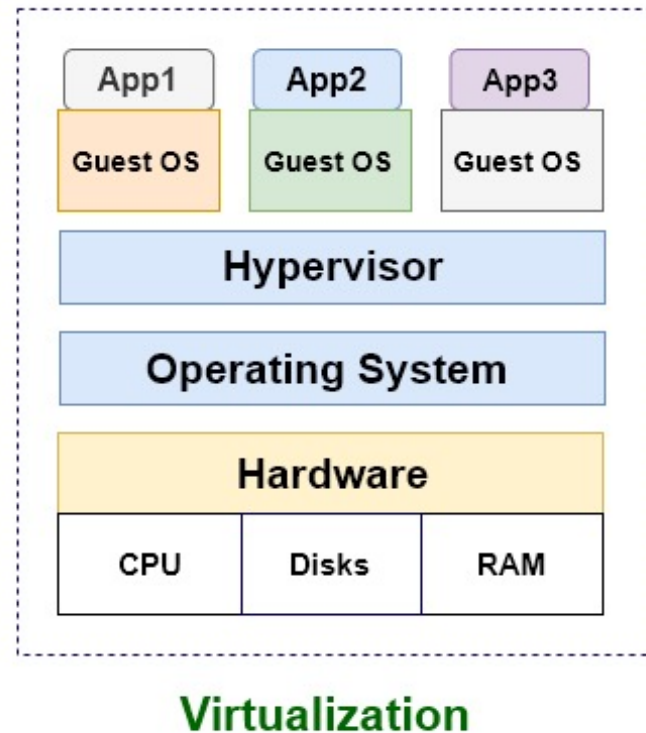
# ¿CUÁL ES EL PROBLEMA?

- Aislamiento de recursos
- Escalar se vuelve complejo
- Lucha por recursos
- Mantenimiento es costoso



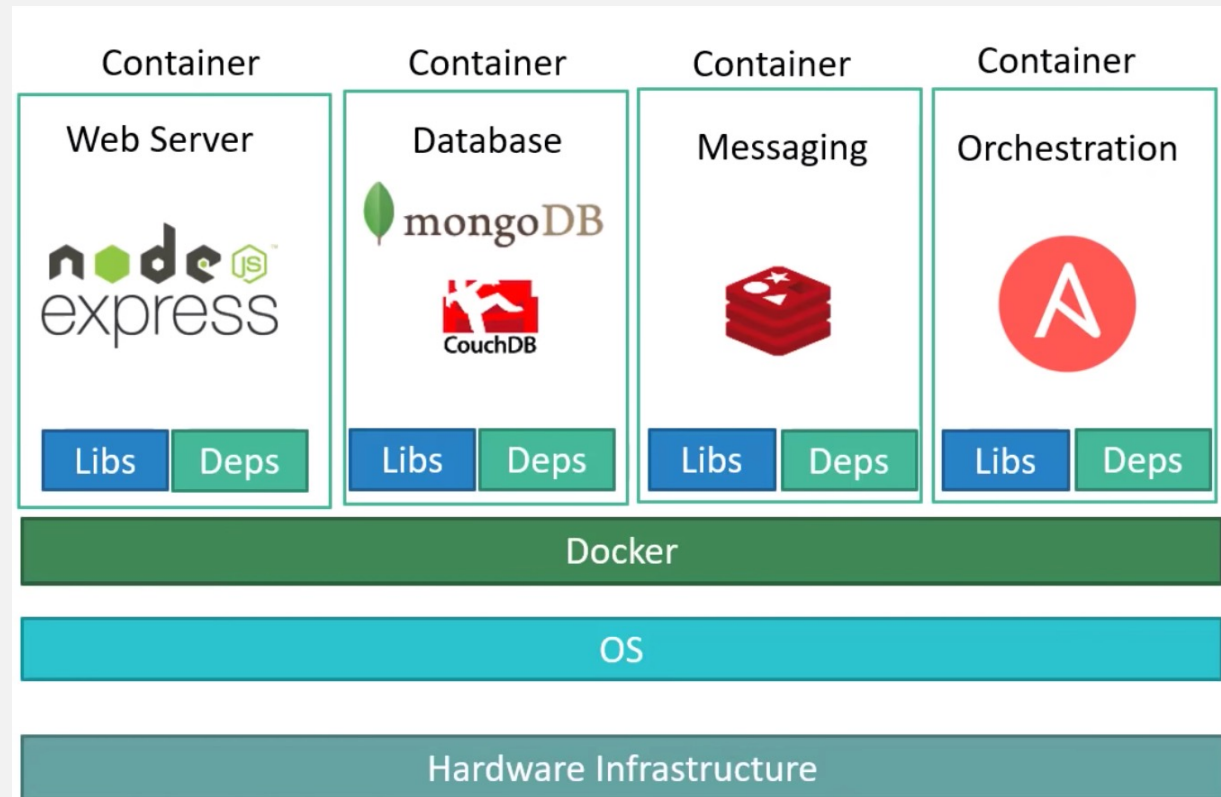
## ¿Y SI VIRTUALIZAMOS?

- Imágenes de SO pesadas
- Manejo de recursos complejo
- Downtime o tiempo fuera de línea
- Mismo problema al escalar



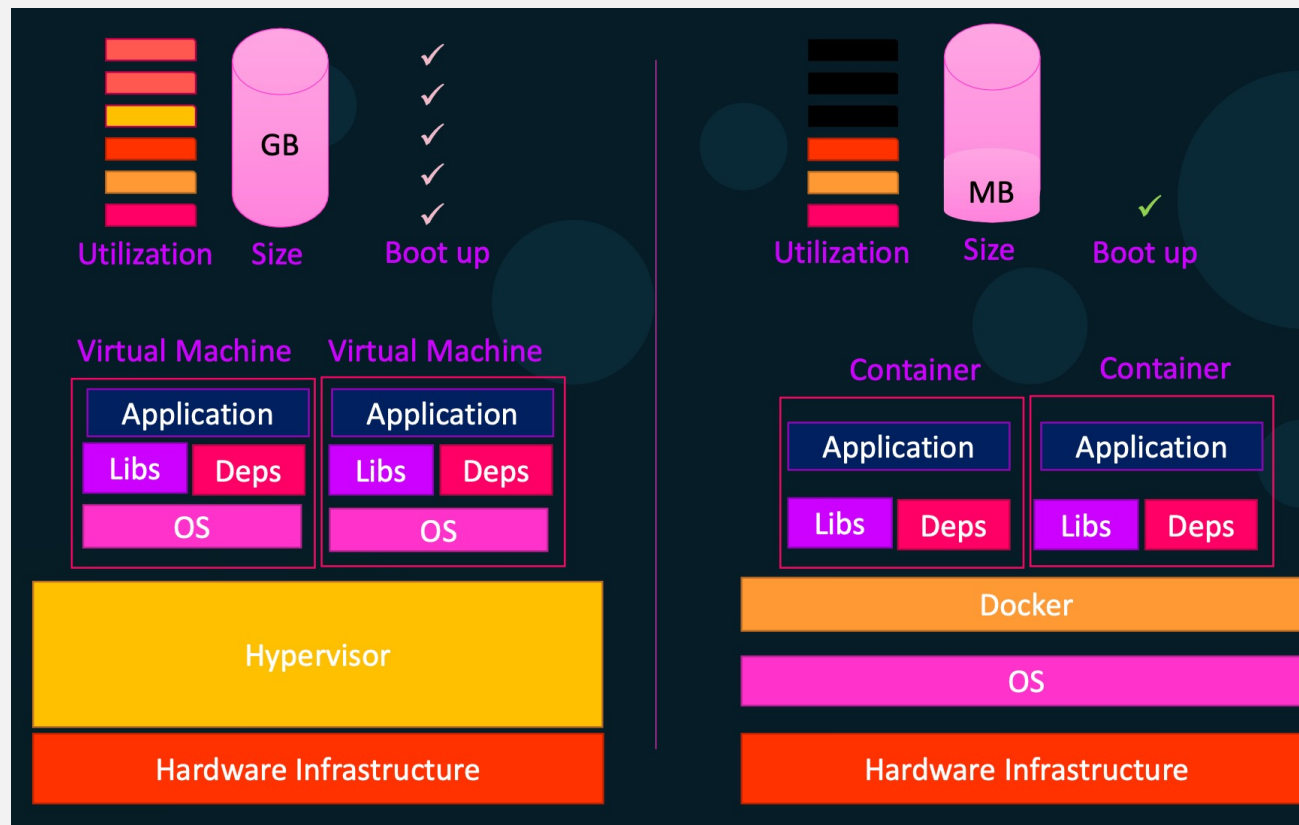
# ¿QUE ES UN CONTENEDOR?

- Servicios aislados
- Interfaz de red, procesos y servicios propios.
- SO que utilizan el mismo Kernel (Linux) **compartido**
  - **Kernel:** interfaz entre hardware y apps
  - **OS:** Capa superior. Usa el kernel e interactua con el usuario





# ¿CONTENEDORES VS MÁQUINAS VIRTUALES?



# ¿CÓMO DESPLEGAMOS UN CONTENEDOR?

## Imágen

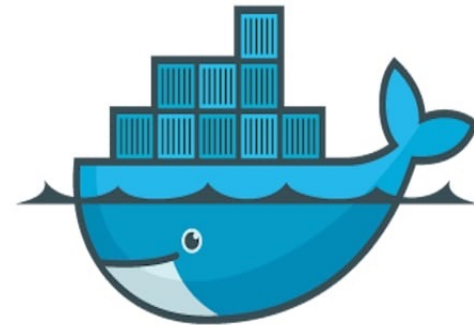
- Inmutable
- Código fuente, bibliotecas, dependencias, archivos para levantar una aplicación.
- Nuevas capas para reutilizar/nuevos archivos

## Contenedor

- Arquitectura ligera de Kernel compartido
- Depligue y levanta las imágenes

# ¿QUE SON LOS DOCKER CONTAINERS?

- Plataforma para desarrollo, entrega y levantamiento de aplicaciones utilizando contenedores
- Una de las más conocidas en el mercado.
- Versiones Community y Enterprise



docker

# DEMO: DOCKER DESKTOP

<https://www.docker.com/products/docker-desktop/>

## COMANDOS: RUN

```
docker run <imagen>
```

```
docker run nginx  
docker run nginx &
```

## COMANDOS: PS

```
docker ps
```

```
docker ps  
docker ps -a
```

## COMANDOS: STOP

```
docker stop <container>
```

```
docker ps -a  
docker stop silly_feynman
```

## COMANDOS: RM

```
docker rm <container>
```

```
docker ps -a  
docker rm silly_feynman
```



## COMANDOS: RMI

```
docker rmi <image>
```

```
docker rmi nginx
```

## COMANDOS: PULL

```
docker pull <image>
```

```
docker pull nginx
```

<https://hub.docker.com/>

## COMANDO: EXEC

```
docker exec <container> <command>
```

```
docker run ubuntu  
docker run ubuntu cat /etc/hosts  
docker run ubuntu sleep 5000  
docker exec exciting_hugle cat /etc/hosts
```

## COMANDO: RUN CON ETIQUETAS

```
docker run <image>:<tag>
```

```
docker run redis:4.0
```

## COMANDO: RUN CON MAPEO DE PUESTOS

```
docker run -p external:internal <image>:<tag>
```

```
docker run -p 12345:5000 kodekloud/webapp
```

## COMANDO: RUN CON MAPEO DE VOLUMENS

```
docker run -v dir_externo/dir_interno <image>:<tag>
```

```
sudo docker run -v /Users/jpmacproml/mysql_volumen:/var/lib/mysql -e MYSQL_ROOT_PASSWORD=my-secret-pw mysql
```

## COMANDO: RUN CON NOMBRE

```
docker run <image>:<tag> --name <name>
```

```
docker run mysql --name mysql_server
```

## COMANDO: LOGS

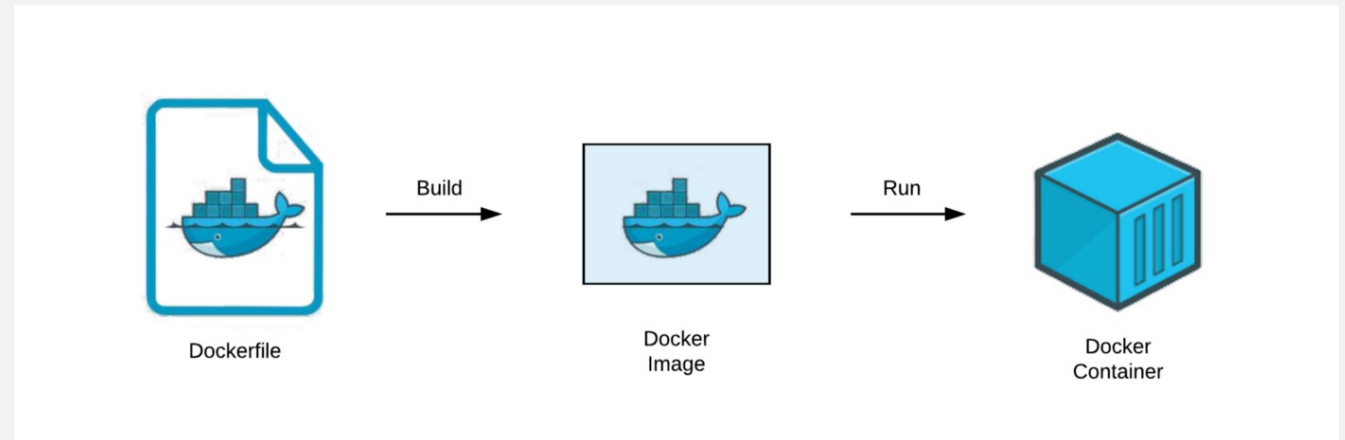
```
docker logs <container_name>
```

```
docker logs mysql_server
```



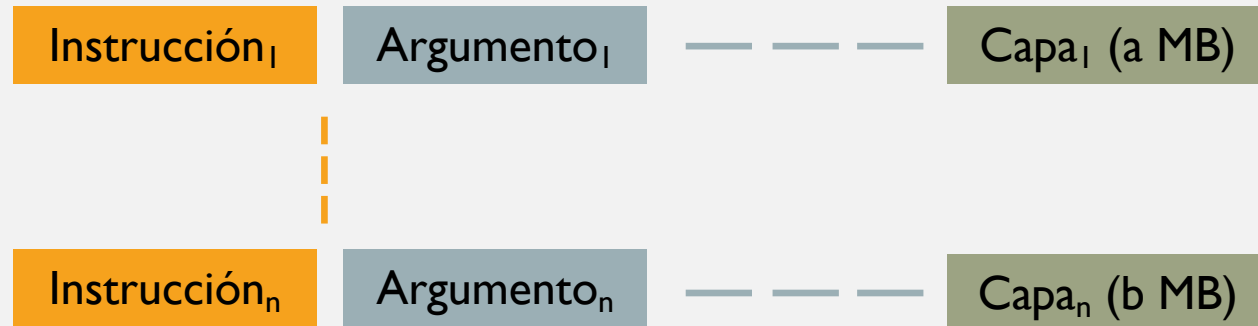
# DOCKER BUILD KIT

- Contruir nuevas imágenes/ reutilizar imágenes
- Push al Docker Container Registry (Docker Hub)
- Archivos de configuración (Dockerfile)
- ¿Cómo quiero que se vea mi imagen?
  - Pipeline de la imagen



# DOCKERFILE

- Dockerfile: pipeline para creación de la imagen
- Arquitectura por capas y cache



# DOCKERFILE: EJEMPLO

**FROM** ubuntu

**MAINTAINER** ARSTECH arstech@e-mail

**RUN** apt-get update && apt-get upgrade -y

**RUN** apt-get install -y apt-utils htop

**CMD** ["echo","It's my Docker Image "]

# DOCKERFILE: ENTRYPOINT VS CMD

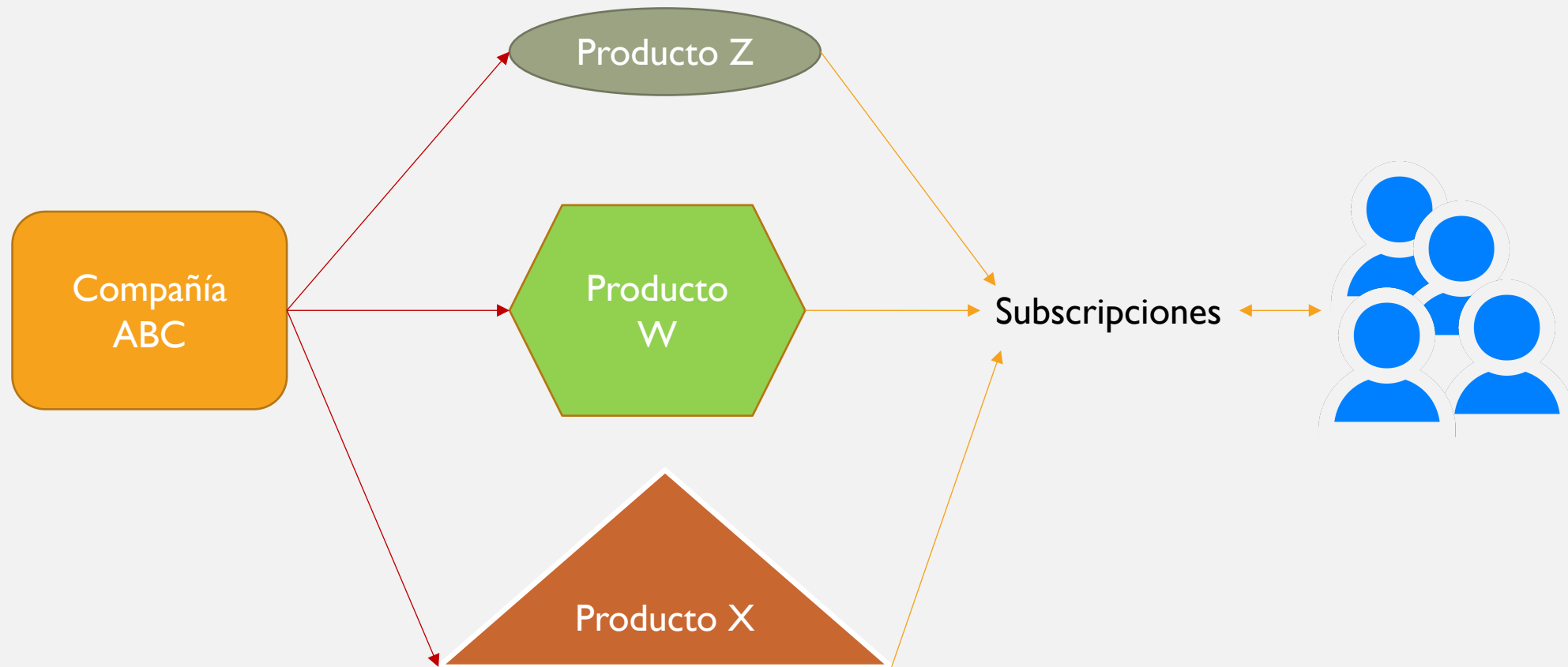
**ENTRYPOINT:** Ejecutable por defecto a correr cuando la imagen inicia. *bin/bash -c por defecto.*

**CMD:** Comando por defecto a ejecutar

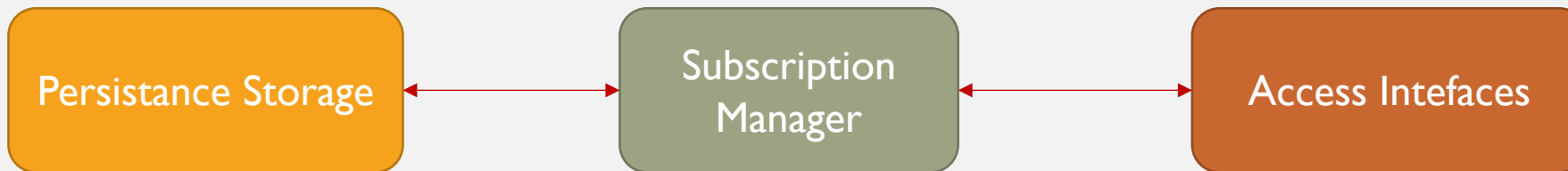
```
FROM debian:wheezy
ENTRYPOINT ["/bin/ping"]
CMD ["localhost"]
```

```
#docker run -it <image>
```

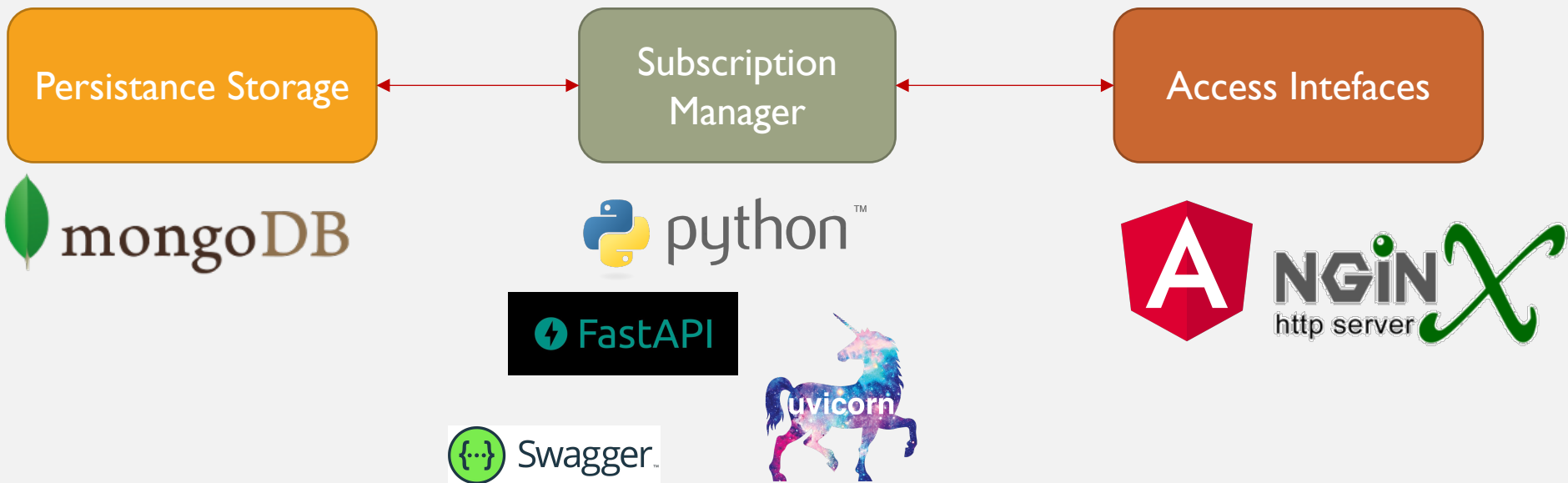
# CASO DE ESTUDIO



## CASO DE ESTUDIO



# CASO DE ESTUDIO

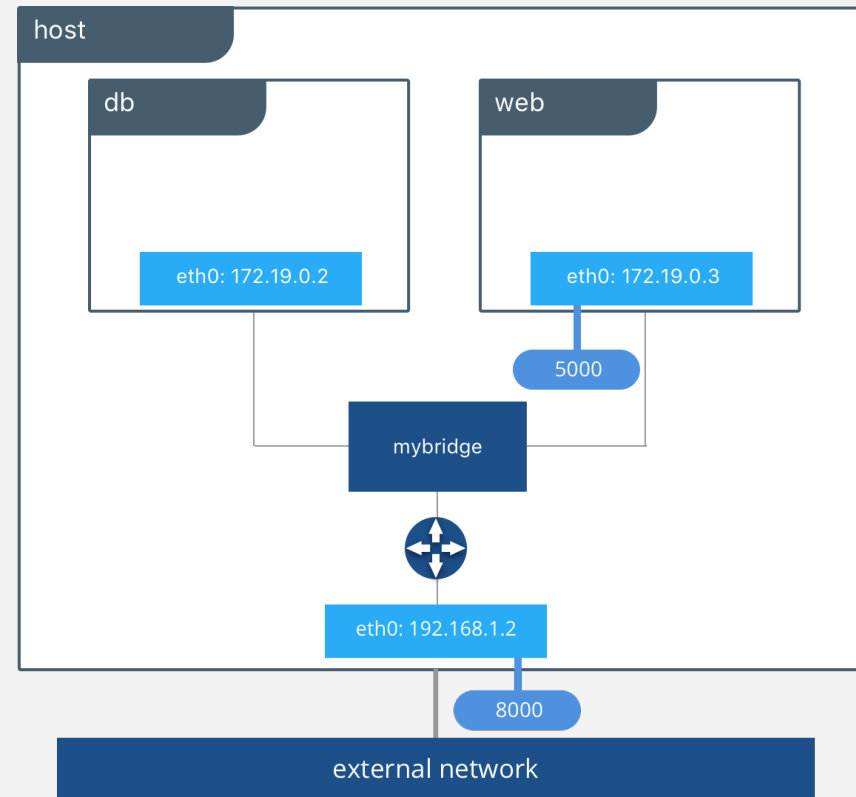


## REVISIÓN DEL CÓDIGO



# DOCKER COMPOSE

- Contruir y desplegar aplicaciones con múltiples contenedores
- Definición de la arquitectura de servicios en un solo archivo YAML
- Promueve la seguridad y el aislamiento



VOLVAMOS AL CÓDIGO

¡MUCHAS GRACIAS!