DESARROLLO DE APLICACIONES UTILIZANDO PYTHON Y ORQUESTACIÓN DE CONTENEDORES DOCKER

Universidad de Costa Rica

Instructor: Jose Pablo Ramírez Méndez

Semana 1

AGENDA

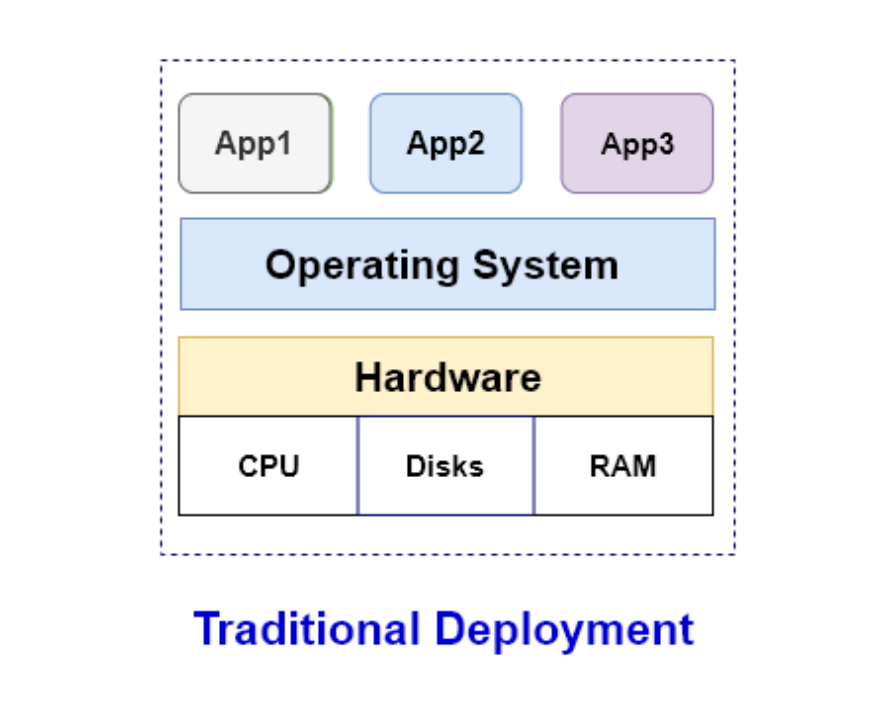
- Objetivos
- Introducción a los contenedores
- Docker containers
- Caso de estudio

OBJETIVOS

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

INTRODUCCIÓN A LOS CONTENEDORES

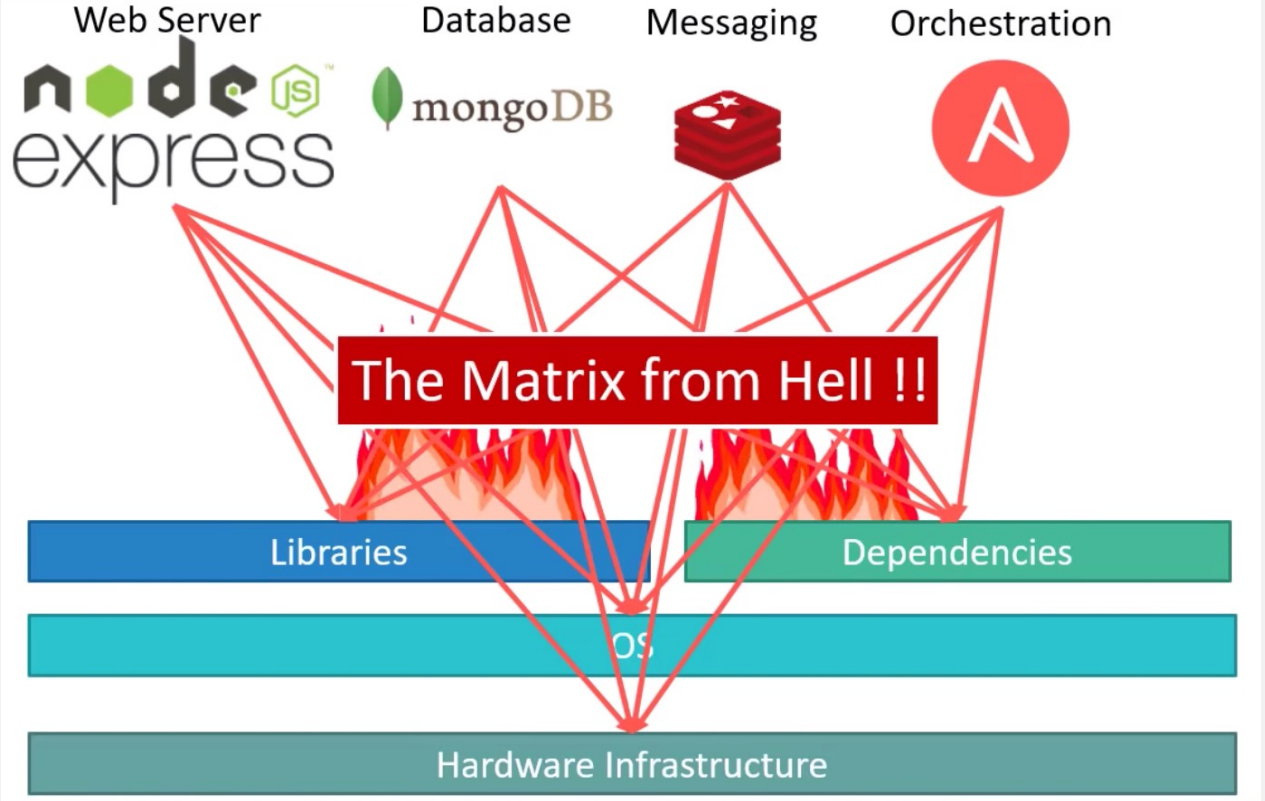
ARQUITECTURA TRADICIONAL



https://miro.medium.com/max/892/1*w5qzf2FLU6frS8-nRg9HpQ.png

¿CUÁL ES EL PROBLEMA?

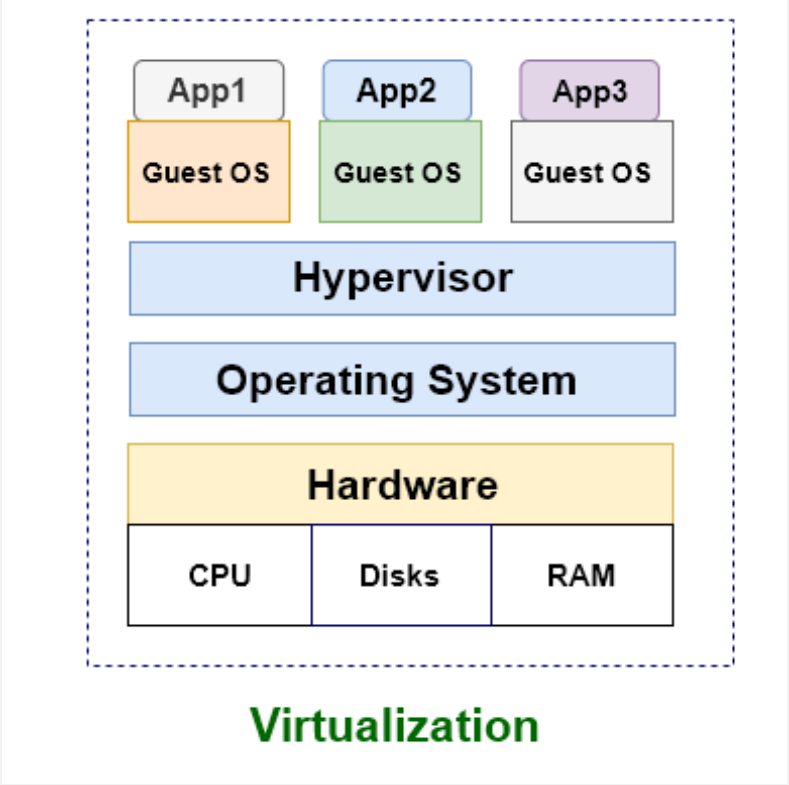
- Aislamiento de recursos
- Escalar se vuelve complejo
- Lucha por recursos
- Mantenimiento es costoso



https://www.filepicker.io/api/file/we7xGcXrTHa3XsNZsE8t

¿Y SI VIRTUALIZAMOS?

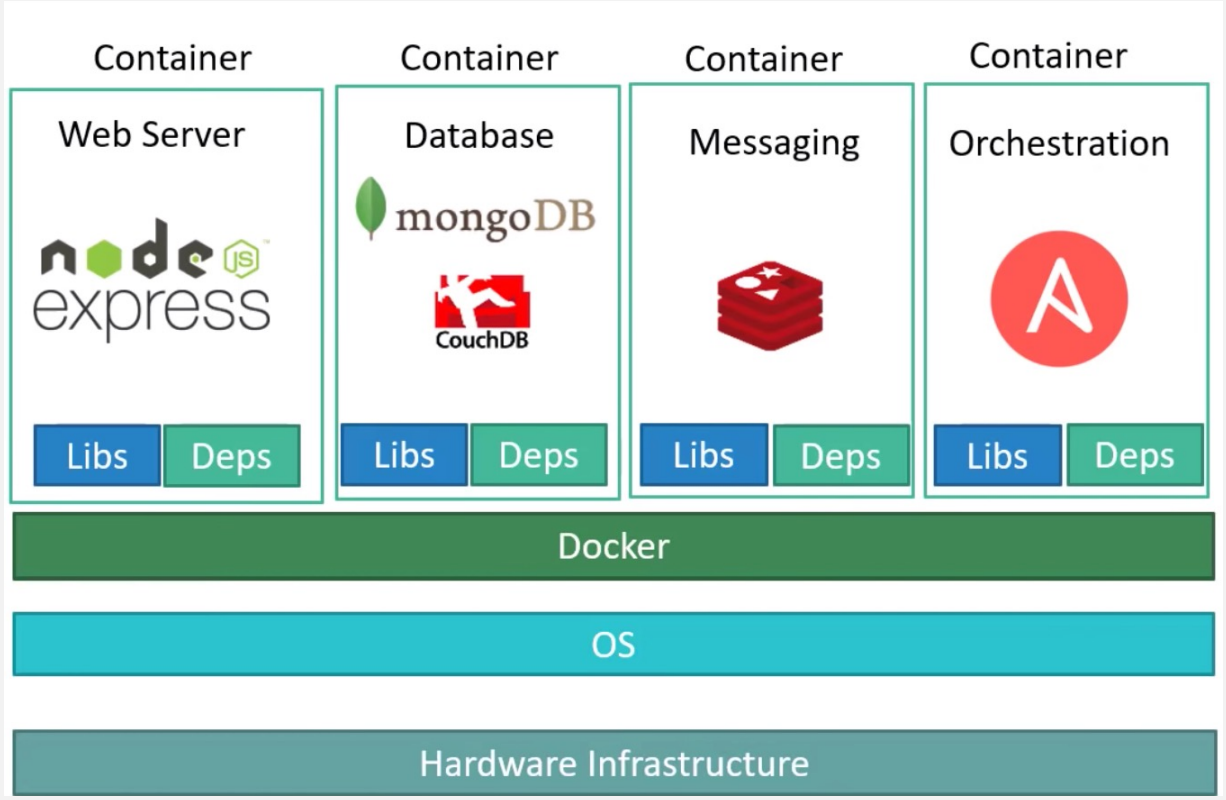
- Imágenes de SO pesadas
- Manejo de recursos complejo
- Downtime o tiempo fuera de línea
- Mismo problema al escalar



https://miro.medium.com/max/778/1*cui2bkQmuAHKIRiq8tF42Q.png

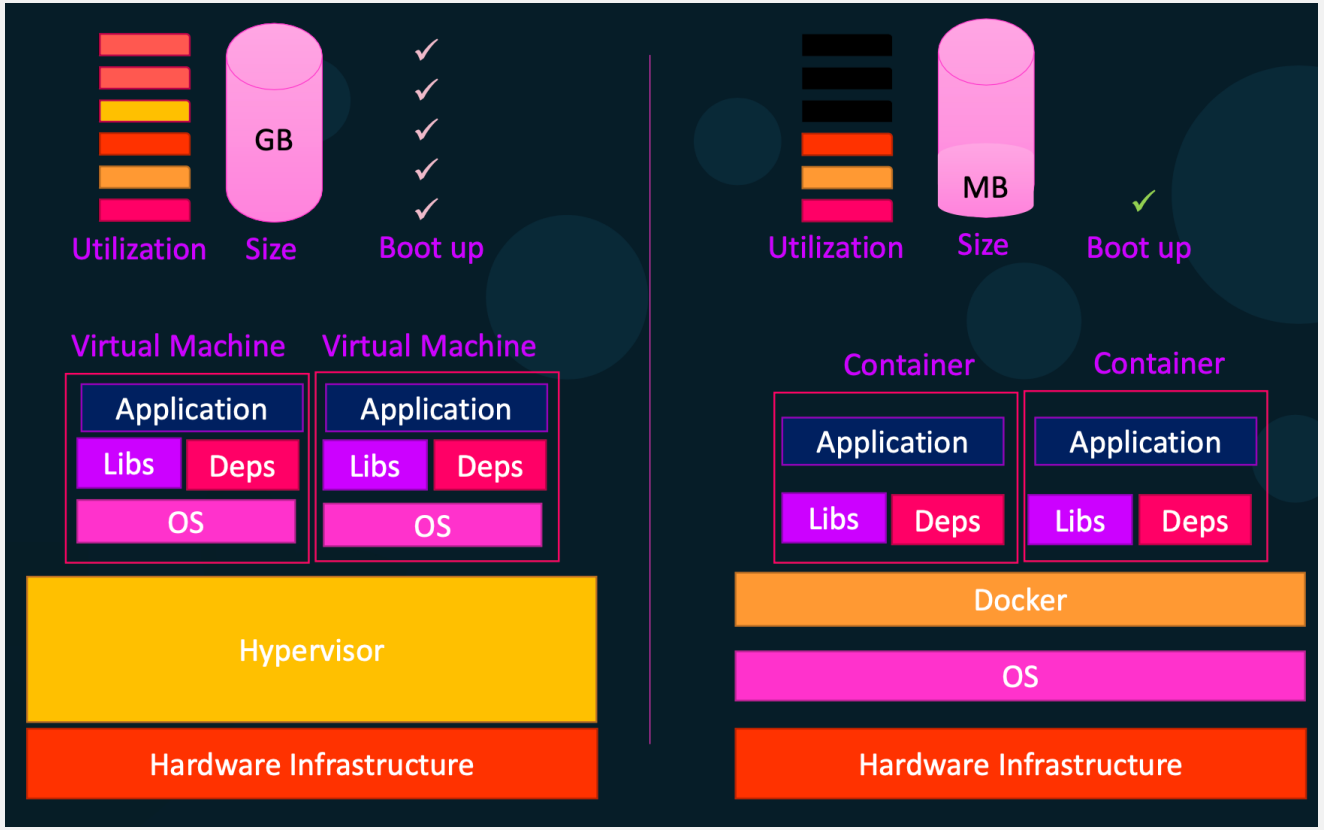
¿QUE ES UN CONTENEDOR?

- Serivicios 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



https://www.filepicker.io/api/file/Dqy8g16LQ5G5mEA9m2fK

¿CONTENEDORES VS MÁQUINAS VIRTUALES?



https://www.filepicker.io/api/file/ylQ1HSGURZauDnxatQVF

¿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

https://docs.docker.com/get-started/overview/

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 -p12345:5000 kodekloud/webapp
```

COMANDO: RUN CON MAPEO DE VOLUMENS

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

```
docker run –p3306:3306 -v /Users/jpmacprom1/mysql_volumen:/var/lib/mysql -e MYSQL_ROOT_PASSWORD=my-secret-pw mysql
```

COMANDO: RUN CON NOMBE

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

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

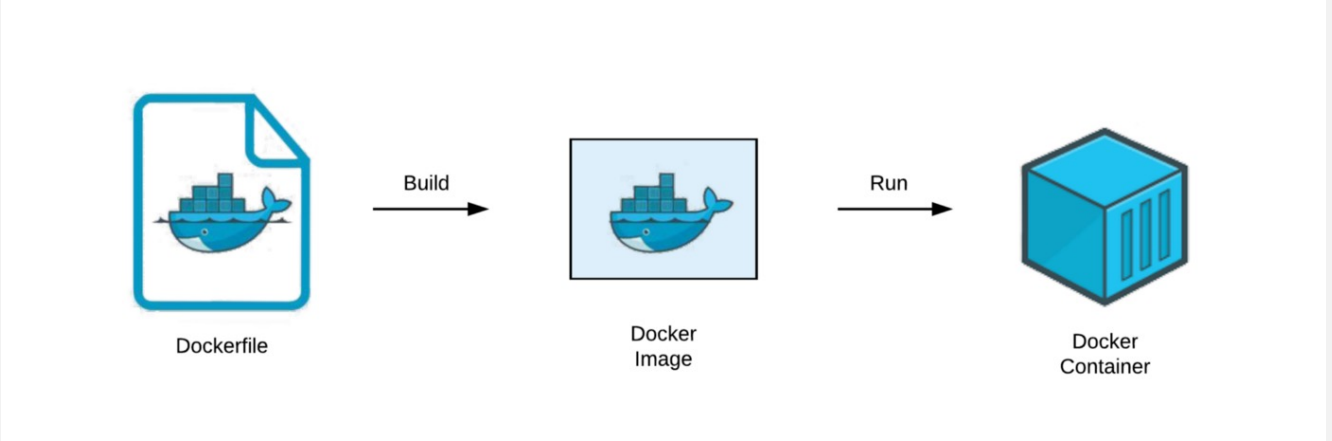
COMANDO: LOGS

```
docker logs <container_name>
```

```
docker logs mysql_server
```

DOCKER BUILD KIT

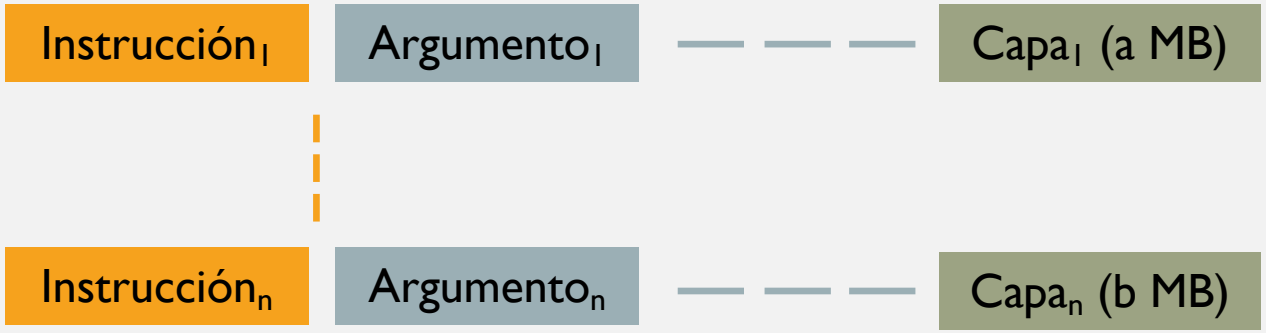
- Contruir nuevas imágenes/ reutilizar imágenes
- Push al Docker Container Registry (Docker Hub)
- Archivos de configuración (Dockerfile)
- ¿Cómo quiero que se vea mi imagen?
  - Pipeline de la imagen



https://miro.medium.com/max/1400/0*CP98BIIBgMG2K3u5.png

DOCKERFILE

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



DOCKERFILE: EJEMPLO

```
FROM ubuntu

LABEL ARSTECH pablow75@hotmail.com

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 corer cuando la imagen inicia. *bin/bash –c por defecto*.

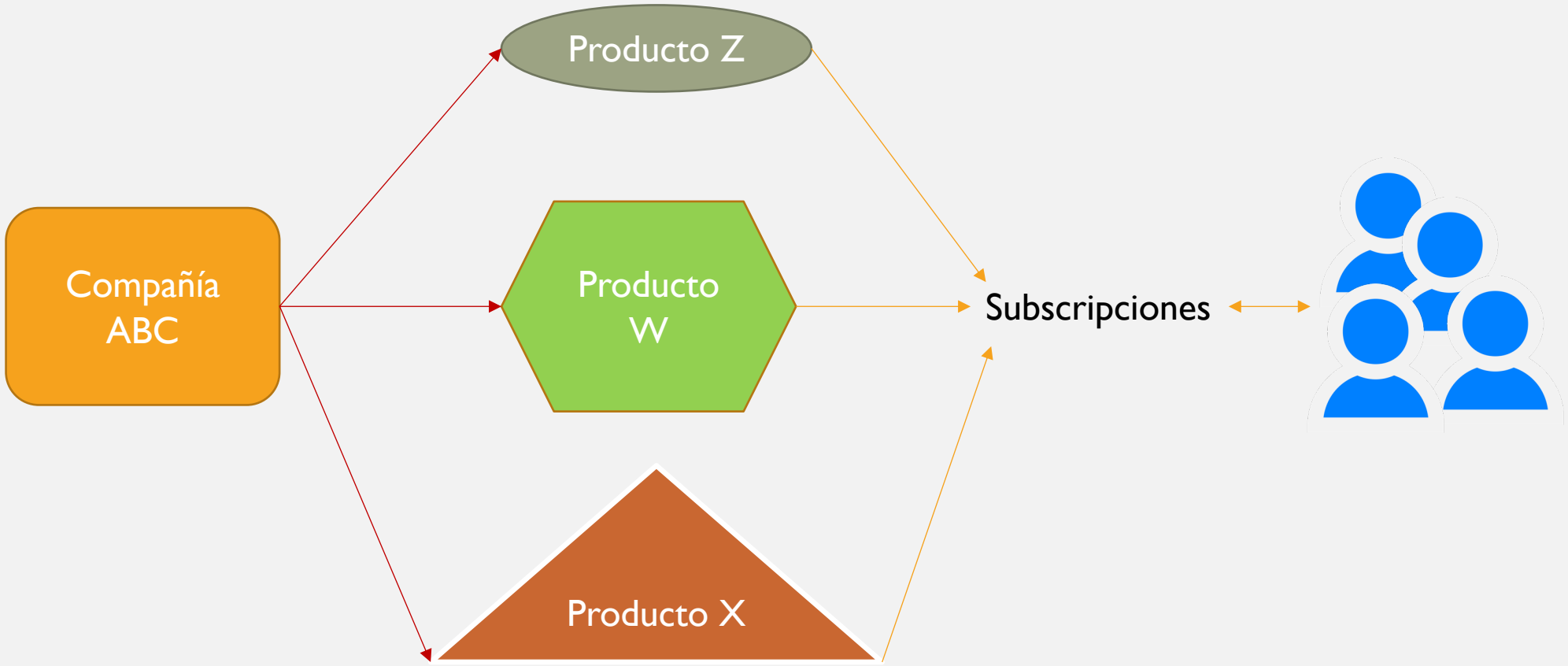
CMD: Comando por defecto a ejecutar

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

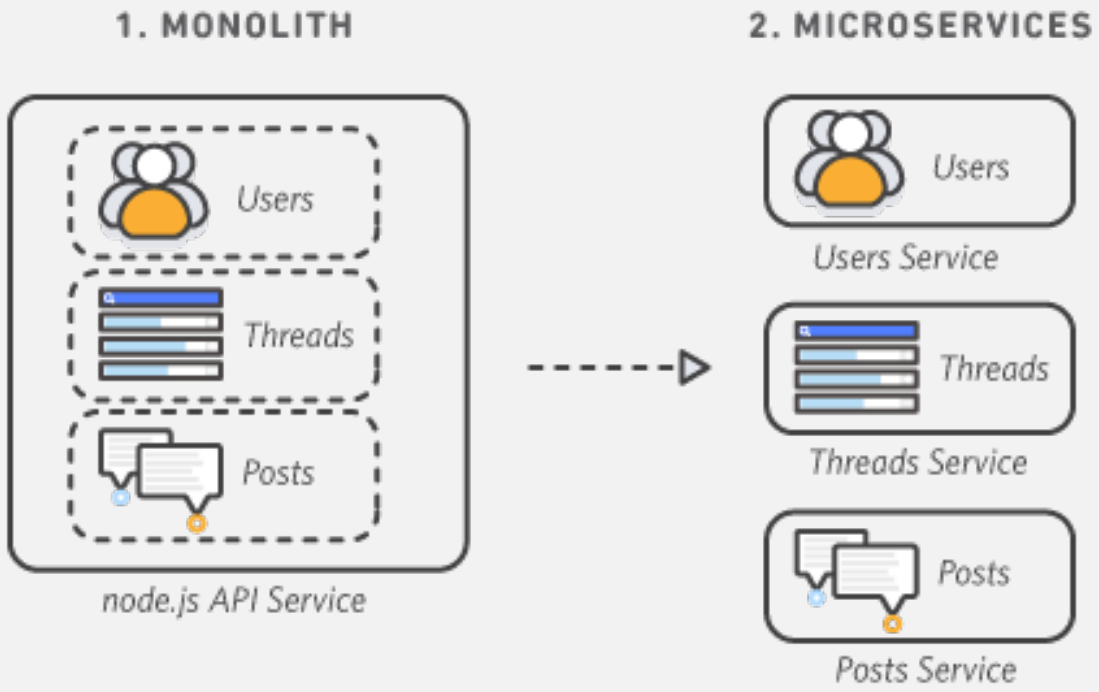
#docker run -it <image>

https://docs.docker.com/engine/reference/builder/

CASO DE ESTUDIO



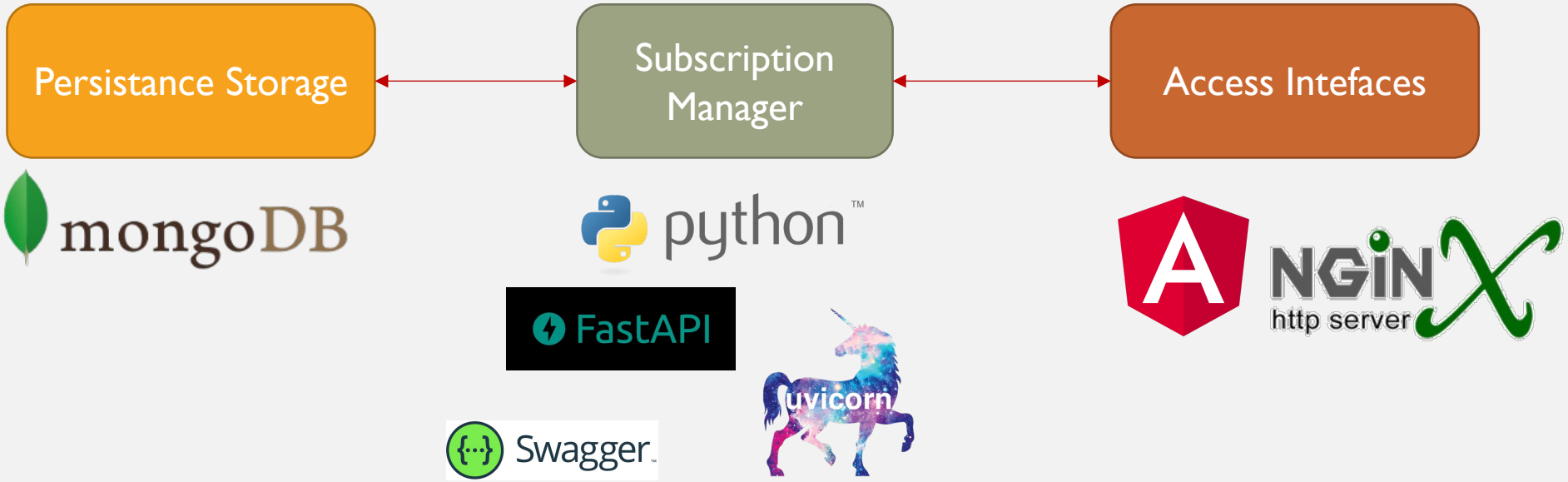
MICROSERVICIOS VS ARQ. MONOLÍTICA



https://aws.amazon.com/es/microservices/

CASO DE ESTUDIO

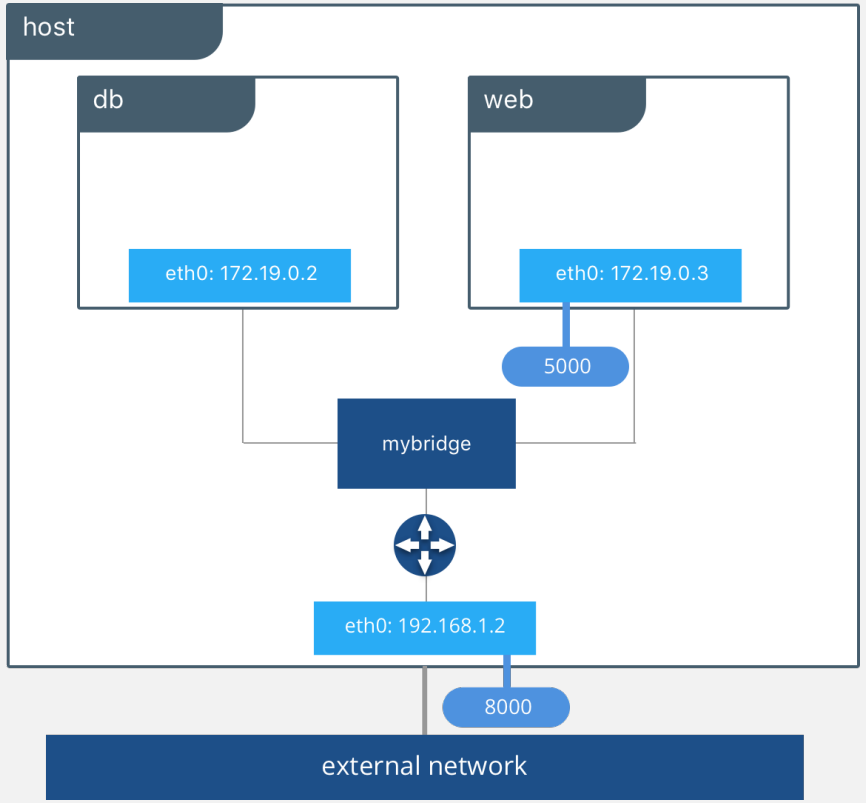
CASO DE ESTUDIO



REVISIÓN DEL CÓDIGO

DOCKER COMPOSE

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



https://docs.docker.com/compose/

http://img.scoop.it/bmExZyvGWidultcwx9hCb7nTzqrqzN7Y9aBZTaXoQ8Q=

VOLVAMOS AL CÓDIGO

¡MUCHAS GRACIAS!