CRAFTECH

Your DevOps team

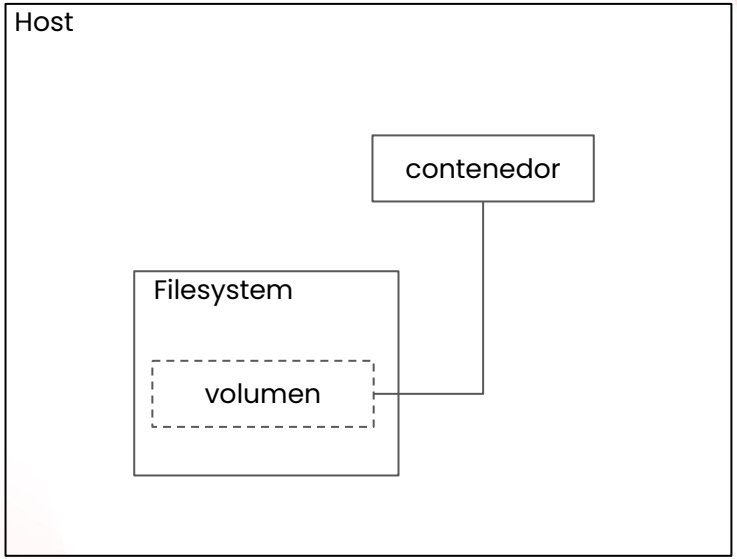
Módulo 3
Docker compose

Repaso

- Contenedores
- VMs vs Contenedores
- Docker
- Buenas prácticas de docker

Volúmenes

- Mecanismo preferido para persistir data utilizada y generada por los contenedores.



Persistir data

- Sin volumen:
  - `docker run -d --name bb busybox sleep 9999`
- Con volumen:
  - `docker run -d --name bb --volume=my-vol:/var/log busybox sleep 9999`

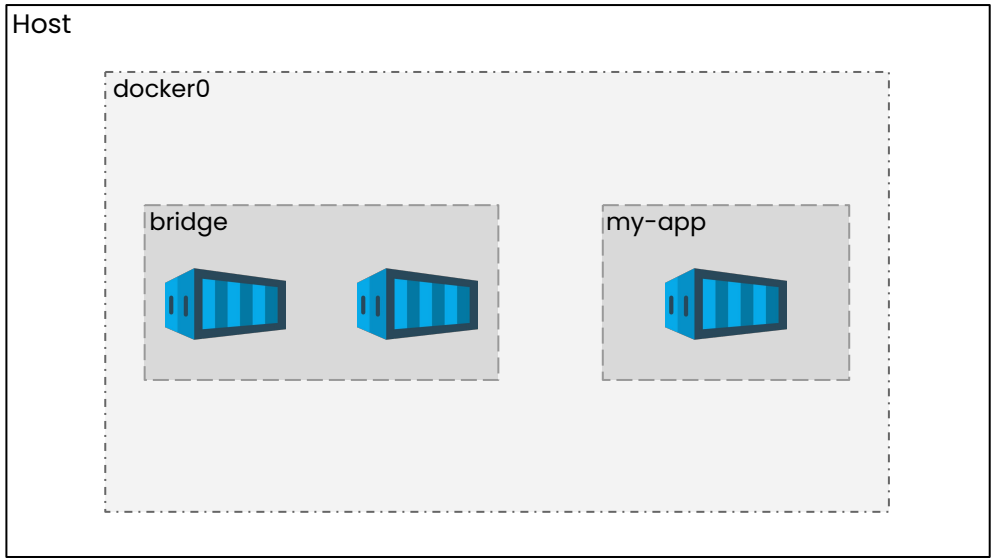
Modificar archivos sin reiniciar contenedor

- Comando:
  - `docker run -d --name bb --volume=/home/$USER/test:/home busybox sleep 9999`

Compartir data entre contenedores

- Comandos:
  - `docker run -d --name bb2 --volumes-from=bb busybox sleep 9999`

Redes en docker



Redes en docker

Comandos útiles:

- Listar:
  - `docker network ls`
- Crear:
  - `docker network create <NOMBRE>`
- Inspeccionar:
  - `docker inspect <NOMBRE_RED>`

What if...?

Tenemos que hacer el deploy de una aplicación y su base de datos en un servidor usando Docker.

- Clonar el código
- Construir la imagen (Dockerfile)
- Crear una red para los contenedores
- Create el volumen para la base de datos
- Poner en marcha la base de datos con su volumen y agregarla a la red
- Poner en marcha nuestra aplicación, agregarla a la red y conectarla a la base de datos

Un montón de tiempo...

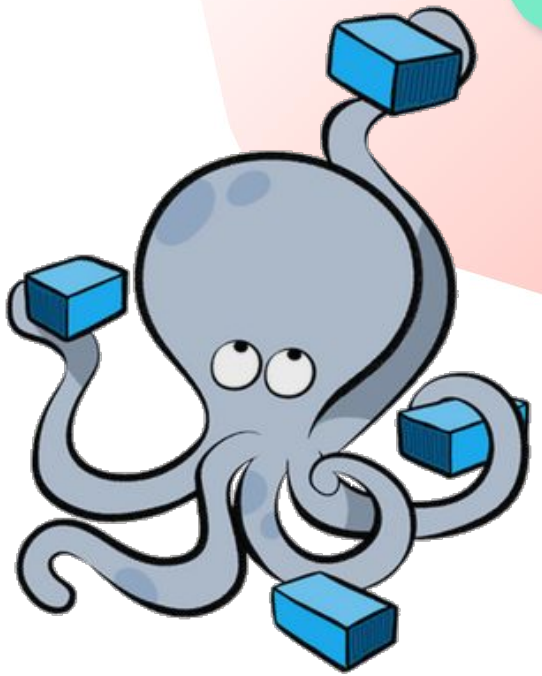
Terminás así

Docker compose

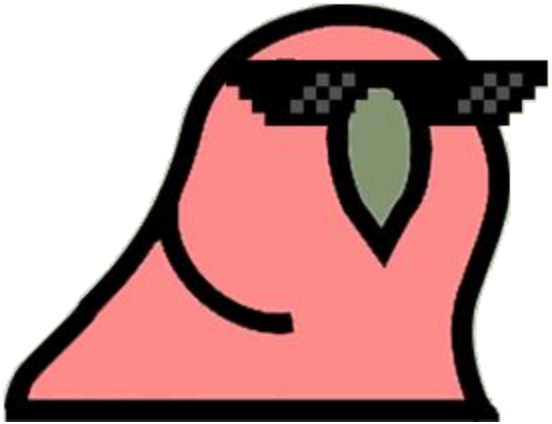
- Herramienta que **simplifica** el uso de Docker.
- No extiende funcionalidades.

¿Qué podemos hacer?

- Crear redes
- Crear volúmenes
- Compartir volúmenes entre contenedores
- Crear varios contenedores a la vez
- Asignar contenedores a redes



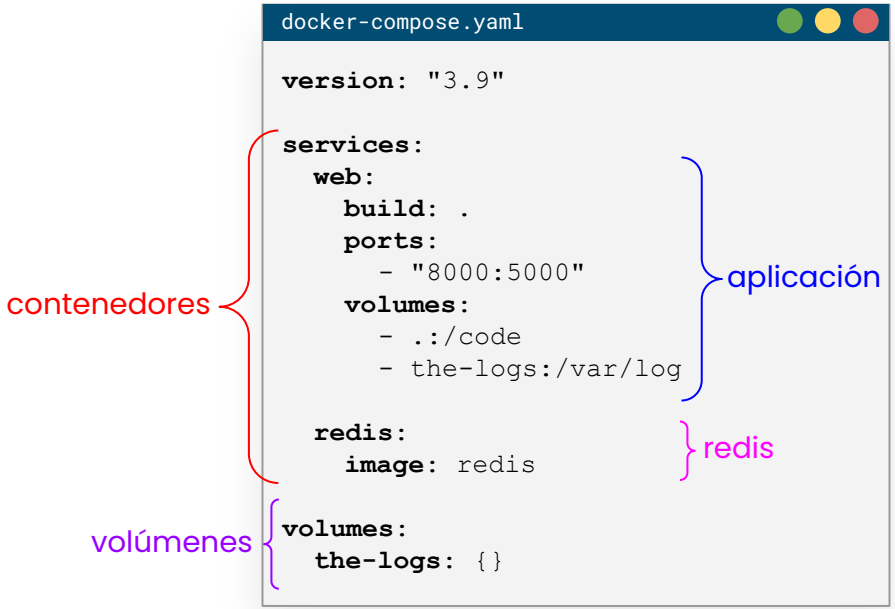
¡Con un solo comando!



Docker compose

- Archivo YAML (`docker-compose.yaml`)
- Contiene toda la información necesaria sobre cómo funciona el servicio junto con sus dependencias.

YAML


docker-compose.yaml
version: "3.9"
services:
web:
build: .
ports:
- "8000:5000"
volumes:
- .:/code
- the-logs:/var/log
redis:
image: redis
volumes:
the-logs: {}
contenedores
aplicación
redis
volúmenes

CRAFTECH

Your DevOps team

Hands-on

Comandos útiles 👀

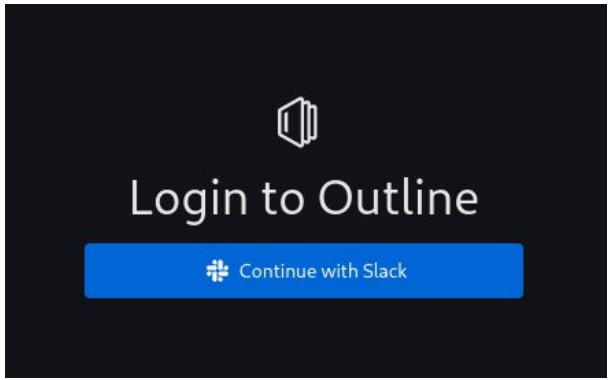
- `docker-compose up`
- `docker-compose up -d`
- `docker-compose down`
- `docker-compose down -v`
- `docker-compose ps`
- `docker-compose build`
- `docker-compose logs -f`
- `docker logs -f <NOMBRE_CONTENEDOR>`
- `docker exec -it <NOMBRE_CONTENEDOR> /bin/sh`
- `docker exec -it <NOMBRE_CONTENEDOR> /bin/bash`

Tareas

- Crear un **único** docker-compose para las aplicaciones presentadas las clase pasada:
  - Cada aplicación debe estar en su propia red:
    - flask-network
    - django-network
  - Las imágenes que se construyan deben estar con su *tag* correspondiente:
    - `flask:local`
    - `django:local`

Tareas

- Correr Outline localmente (*localhost*). Obteniendo la pantalla de Acceso:



¡Muchas gracias!