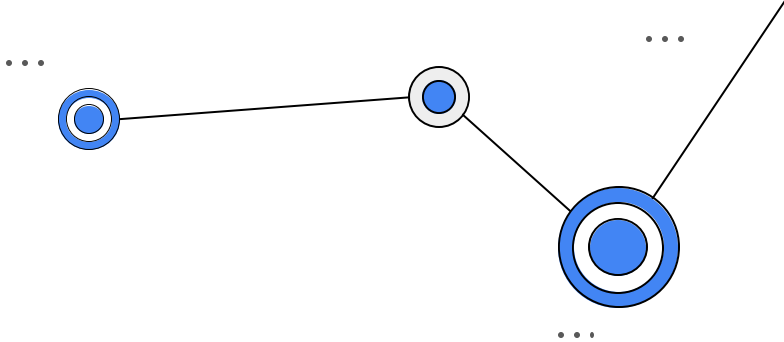
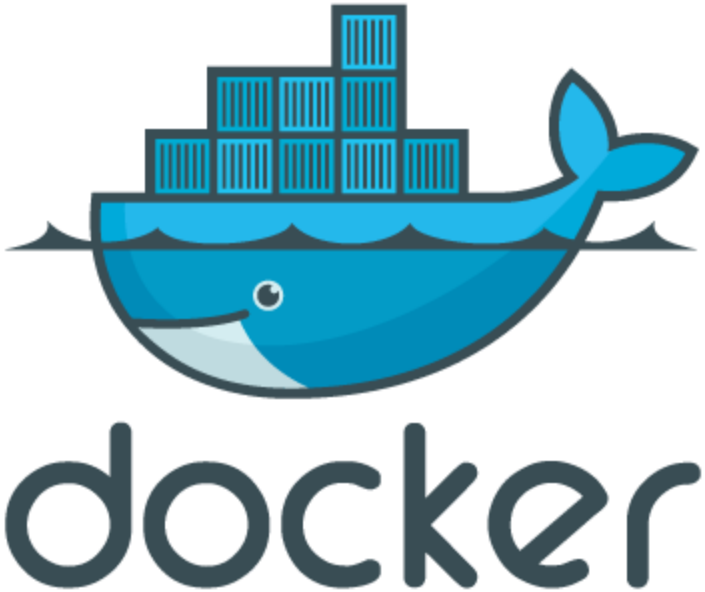




Containerización y Orquestación

Por: Rolf Sommer, Matías Orbeta



Varios orígenes

Problemas



Portabilidad

Cómo hacer que mi código funcione en todos los entornos

...



Versionamiento

Cómo aseguro un control de versiones

...



Aplicaciones

Debo instalar muchas dependencias que usaré en mi app

...

01

Containers

¿Qué es y cómo funcionan?



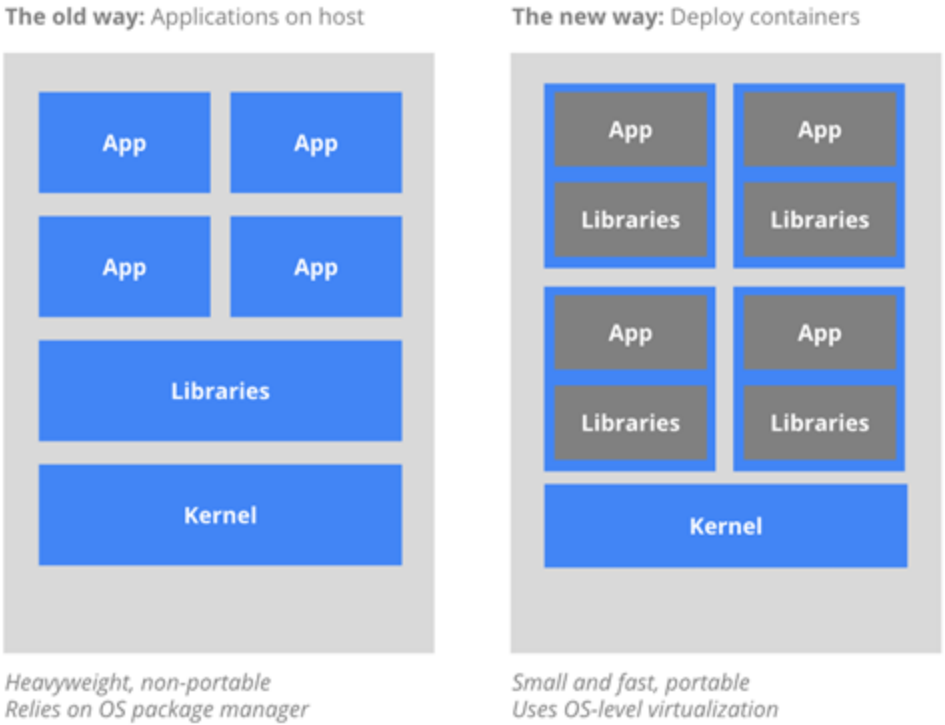
Una pequeña introducción



https://www.youtube.com/watch?v=Gjnup-PuquQ

Empaquetar software para ejecutarlo en su propio entorno consistente con sus propias dependencias, sobre diversos entornos





Portabilidad a bajo costo

https://kubernetes.io/es/docs/concepts/overview/what-is-kubernetes/

Ganancias



Consistencia de entornos

Build once, run everywhere



Portabilidad entre nubes

Computación como commodity



Aislamiento de recursos

Puedes medir y asignar recursos fácilmente y mejora la seguridad (bien configurado)



Desarrollo simplificado en CI/CD

Integración sencilla a sistemas de CI/CD



Agilidad en deploy

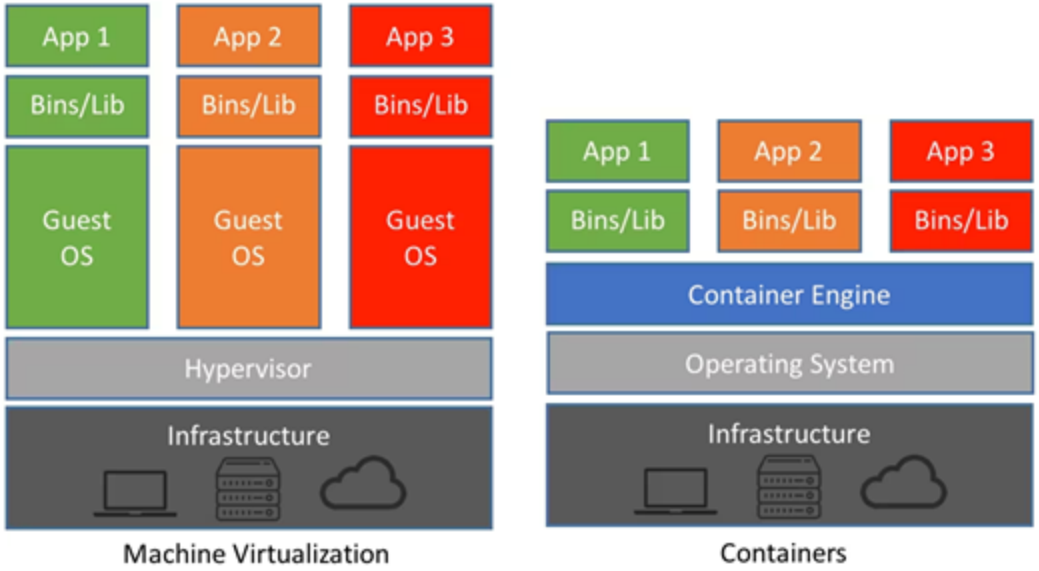
Rápido despliegue a la hora de desarrollar y lanzar a producción



Observabilidad

Poder medir performance y salud, tomar medidas eficientes y rápidas

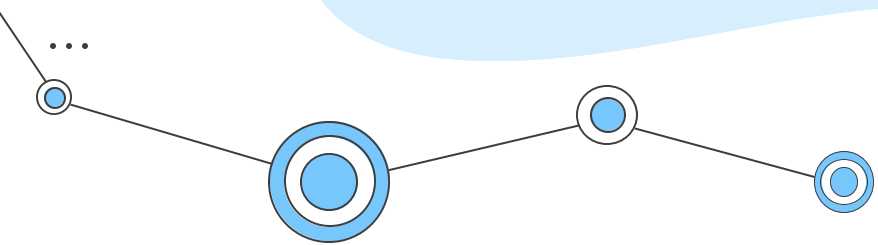
https://kubernetes.io/es/docs/concepts/overview/what-is-kubernetes/
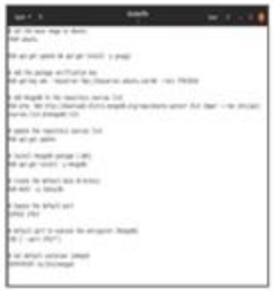


Ahorra espacio y gana velocidad usando solo lo necesario: bajo overhead

Virtualización a nivel de SSOO

Docker es una suite que nos permite manejar/crear estos containers



¿Cómo funciona Docker?



Dockerfile

build



Docker Image

run

GEEK FLARE

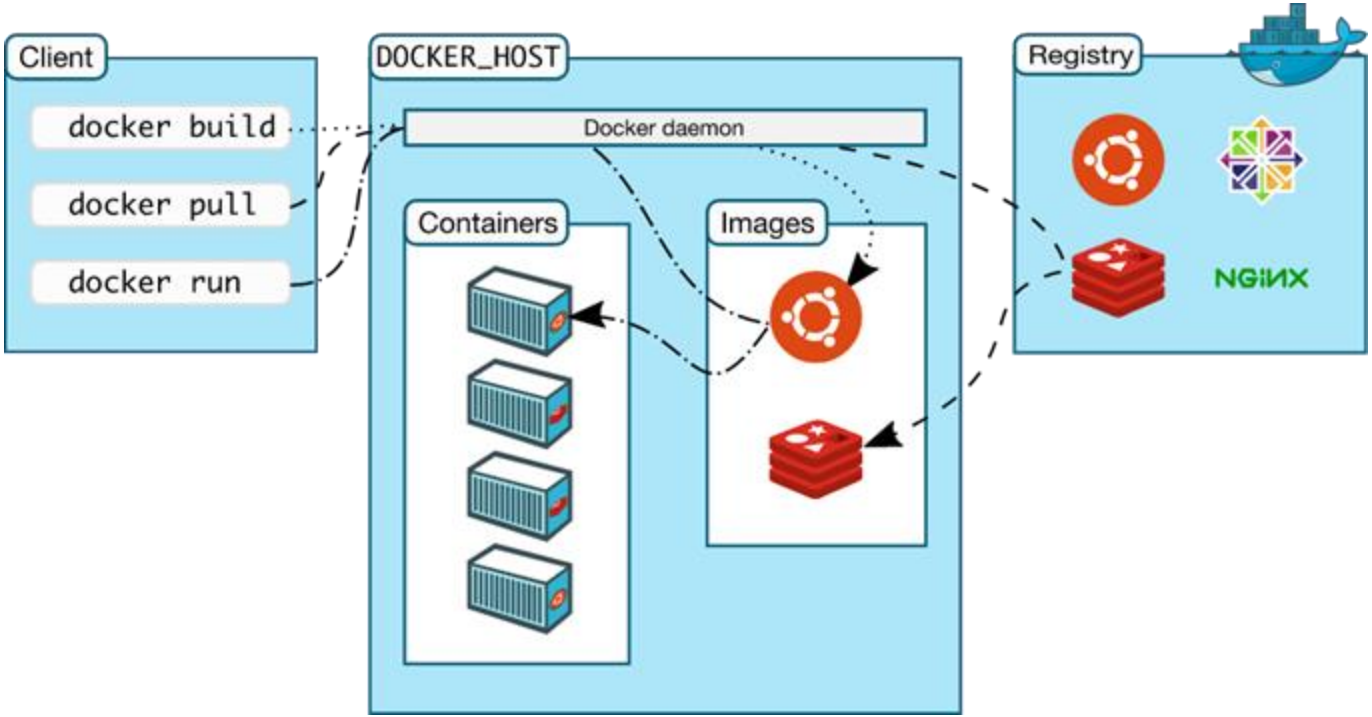


Docker Container

Cómo construyo

Cómo ejecuto

Instancia de ejecución

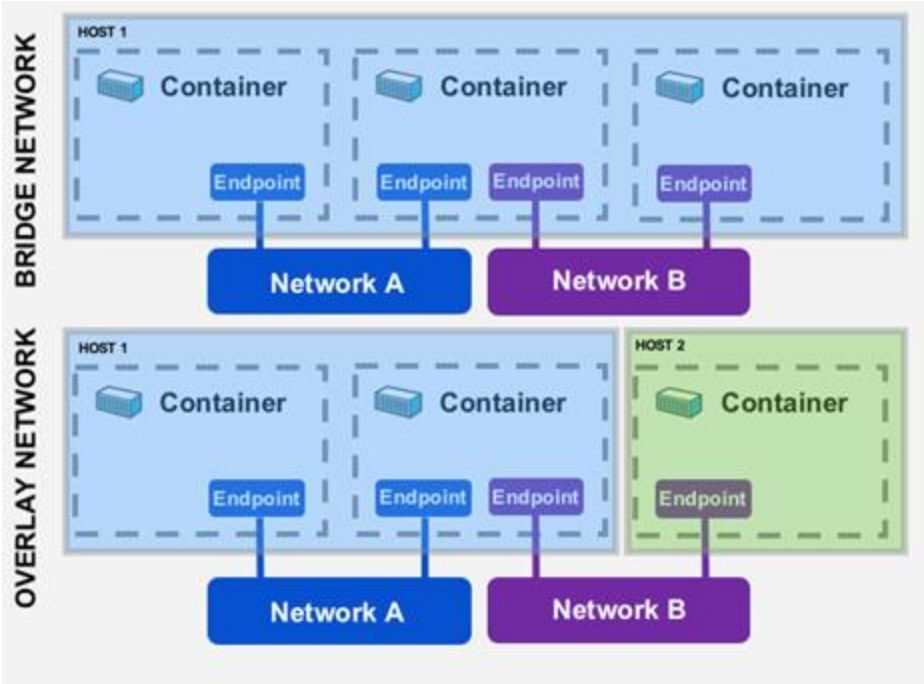

Client
docker build
docker pull
docker run
DOCKER_HOST
Docker daemon
Containers
Images
Registry
NGINX

¿Cómo sé que imagen utilizar?

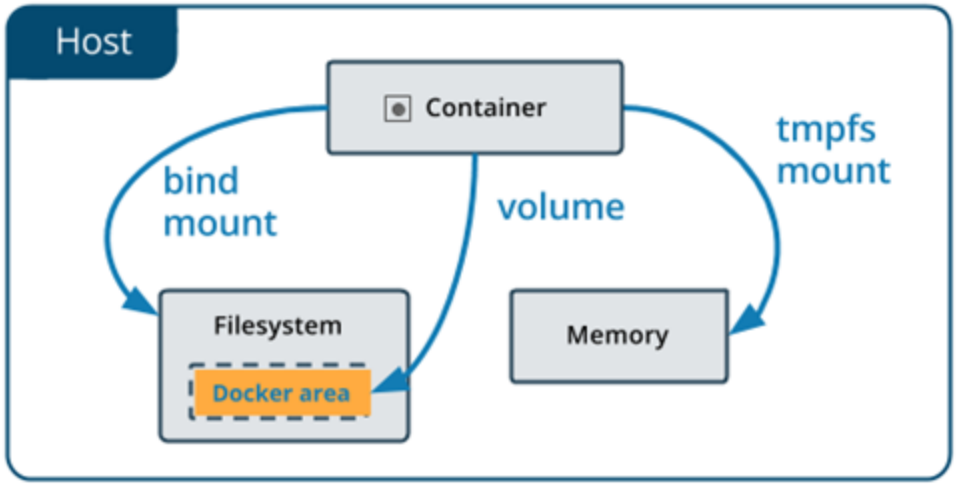
Puedes construirla
desde 0
...
O utilizar una imagen
desde algún registro:
una librería de imágenes
versionadas



¿Cómo conectamos nuestros contenedores?
Networks



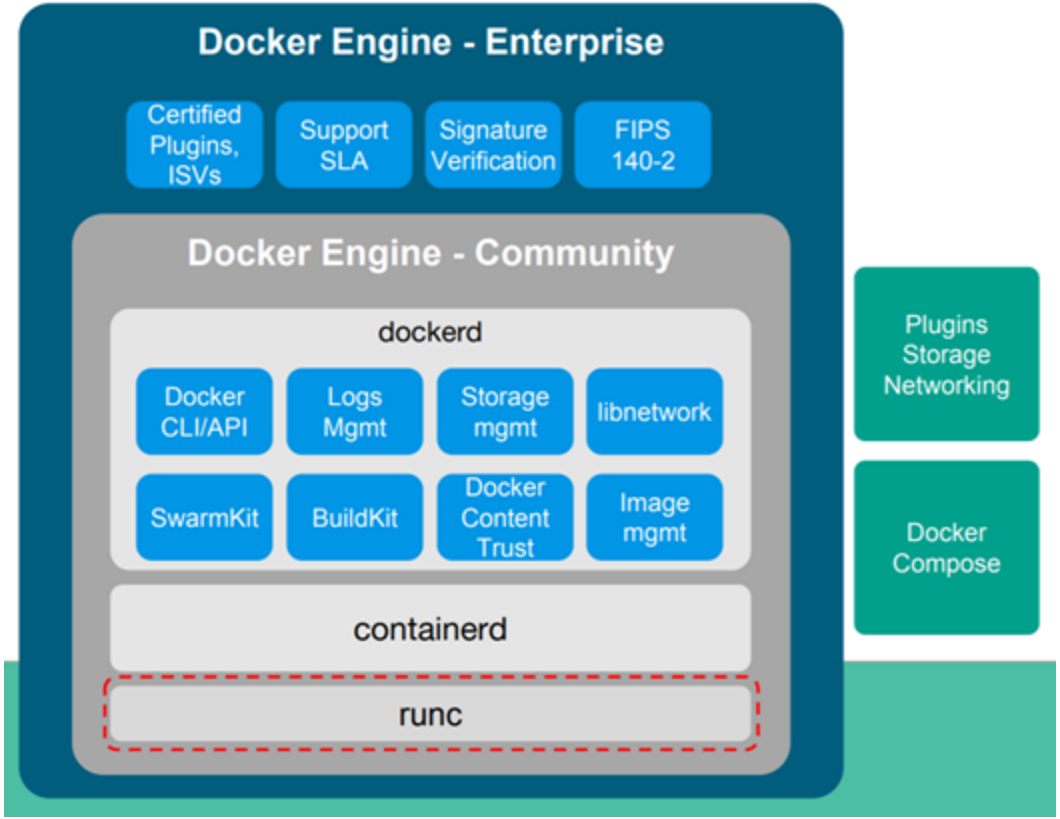
¿Cómo manejamos la persistencia dentro de los contenedores? Volumes



- Bind: directo al SSOO
- Tmpfs: alojado para diversos propósitos
- Mount: volumen estándar (como un pendrive o hdd)

https://docs.docker.com/engine/storage/volumes/

E.g. Docker Engine



02

Definiendo imágenes

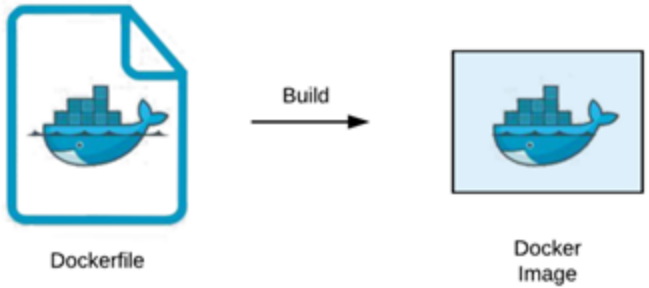
Un archivo que nos permite generar nuestras imágenes



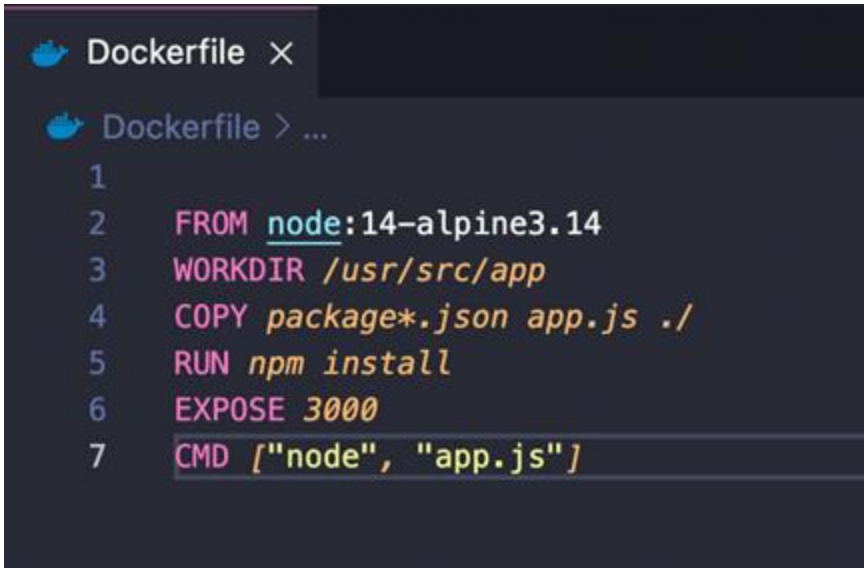


Un **Dockerfile** es un **archivo** de texto plano que contiene una serie de **instrucciones** para crear una imagen

- Es un “recetario” para tomar una “foto” de su app
- Conforma al estándar OCI



¿Cómo se ve un Dockerfile?



Sintaxis típica



FROM <image>:<tag>

Indica la imágen base desde la cual construiremos



RUN <command>

Permite ejecutar comandos en el contenedor



WORKDIR <path>

Nos permite ubicarnos en el path especificado dentro del contenedor



COPY <path_interno> <path_container>

Copiar archivos o carpetas dentro del contenedor



EXPOSE <port>

Exponer servicio en el puerto de la red interna de Docker

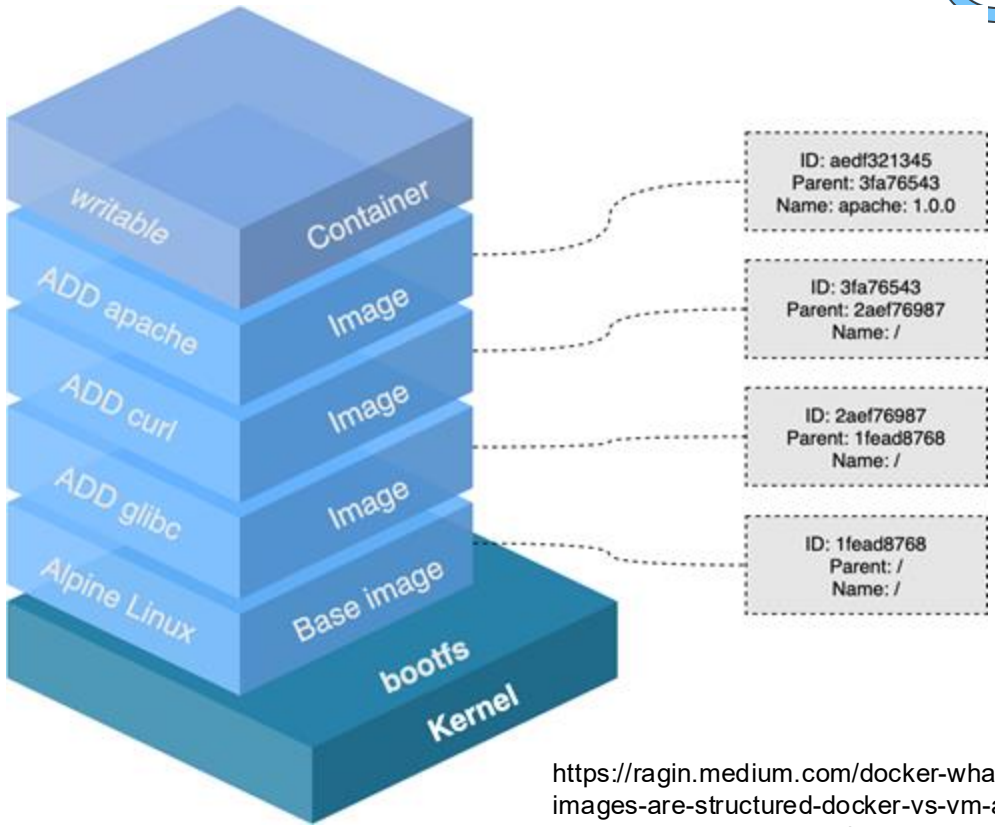


CMD <command>

Serie de comandos que se ejecutan cuando se termina de iniciar el contenedor

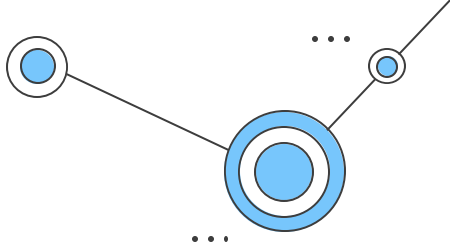
Capas

- Cada capa representa un “cambio”
- Estos cambios son cacheables: usualmente no necesito cambiar todo (e.g. npm install, gemas)
- CI/CD, dev, etc

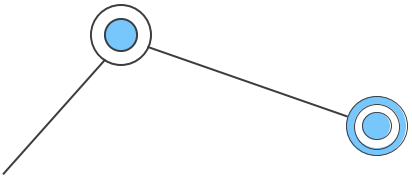


https://ragin.medium.com/docker-what-it-is-how-images-are-structured-docker-vs-vm-and-some-tips-part-1-d9686303590f

Ejemplo



```
/src
    /app
        app.js
        package.json
```



Veamos cómo funciona

app.js > ...

```
1  const express = require('express')
2  const app = express()
3
4  app.get('/', (req, res) => res.send('Hello World!'))
5  app.listen(3000, () => console.log('Server ready'))
6
```

Dockerfile

Dockerfile > ...

```
1
2  FROM node:14-alpine3.14
3  WORKDIR /usr/src/app
4  COPY package*.json app.js ./
5  RUN npm install
6  EXPOSE 3000
7  CMD ["node", "app.js"]
```

Veamos cómo funciona

```
docker build -t ay2 .
[+] Building 3.3s (9/9) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 167B
 => [internal] load .dockerignore
 => => transferring context: 34B
 => [internal] load metadata for docker.io/library/node:14-alpine3.14
 => [internal] load build context
 => => transferring context: 35.14kB
 => [1/4] FROM docker.io/library/node:14-alpine3.14@sha256:d0004e73d11c77b47d179aea71fd859f34aae49da22a16cfd613d88fdaccaecb
 => CACHED [2/4] WORKDIR /usr/src/app
 => [3/4] COPY package*.json app.js ./
 => [4/4] RUN npm install
 => exporting to image
 => => exporting layers
 => => writing image sha256:579ea41a6fb87ec7f922def59997b72ece03f34774c576996785742f0b3cb476
 => => naming to docker.io/library/ay2
```

```
docker run -p 3000:3000 ay2
Server ready
```

Veamos cómo funciona



Comandos útiles en Docker-cli

- Manejo
  - build: Clave en docker
  - rm
  - prune
- Operación:
  - run: *Lanzamiento*
  - stop: *Friendly*
  - kill: *Solo en caso de emergencia*
- Intervención (Siempre en ejecución):
  - exec: *Introducirse en un container*
  - cp: *Copiar archivos a un container*
- Observabilidad
  - logs: *Mostrar o seguir*
  - stats
  - ps



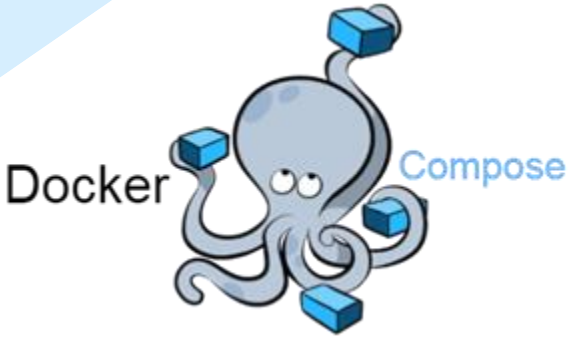
03

Orquestación

Como hago que los contenedores corran libres







Orquestación



- Movilizar containers mediante sistemas de descripción
- Soluciona problemas de escalabilidad
- Permite monitorizar y levantar sistemas a pequeña - mediana -gran escala

Ventajas (algunas)

01

Uso de entornos específicos

Puedo definir variables para distintos ambientes (prod, dev, etc) y secretos

02

Más simple de ver

El código y cómo junto los servicios es más simple

03

Ambientes aislados

Múltiples ambientes aislados en un sólo host

04

Mejora la persistencia

Sólo recrea los contenedores cuando han cambiado

Mercado



Compose

Fácil de usar y definir
Integración fácil con Docker



Kubernetes

Battle-tested
Complejo pero poderoso



Swarm

Auspiciado por docker: más poderoso que compose



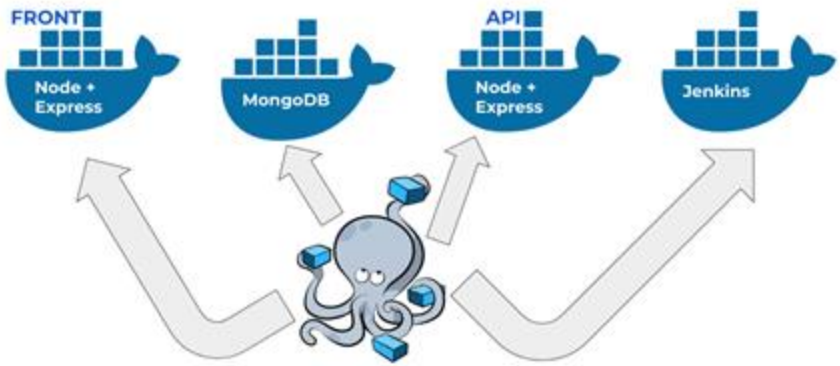


Propietarios cloud

ECS para AWS
ACS para Azure
Vendor lock-in
Facilidades en precio y entorno

Con Compose usamos un archivo YAML (imperativo) para configurar los servicios de nuestra aplicación.

Permite crear, orquestar y correr todos los servicios de nuestra aplicación.



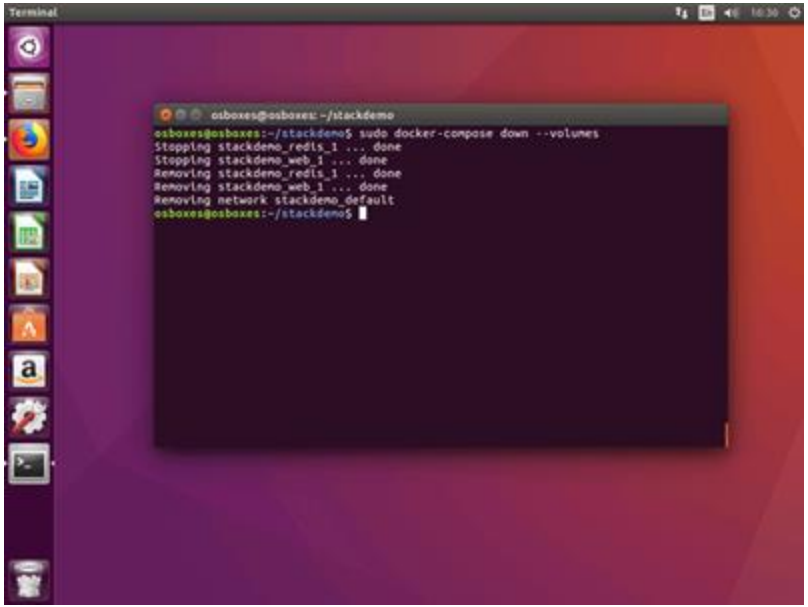
Ejemplo de sintaxis (YAML)

```yaml
version: "3.9"                          # version de docker-compose
services:                               # por cada servicio hay una aplicación en un container
  api:
    build: ./api                        # se especifica la ubicación del Dockerfile
    command: npm start                  # se puede incluir un comando cuando se inicializa el container
    volumes:                            # se mapean archivos del host al contenedor / similar a COPY en Dockerfile
      - ./api:/api
      - /api/node_modules
    networks:                           # se puede especificar la conexión de red segura entre los contenedores
      - app-network

  nginx:
    image: nginx:latest                 # Se hace pull de la imagen de nginx desde dockerhub / same as From <algo> in dockerfile
    volumes:
      - ./nginx:/etc/nginx/conf.d
    ports:
      - 80:80                           # puerto del host al container
    depends_on:                         # primero debe levantarse el contenedor api para levantar el contenedor nginx
      - api
    networks:
      - app-network

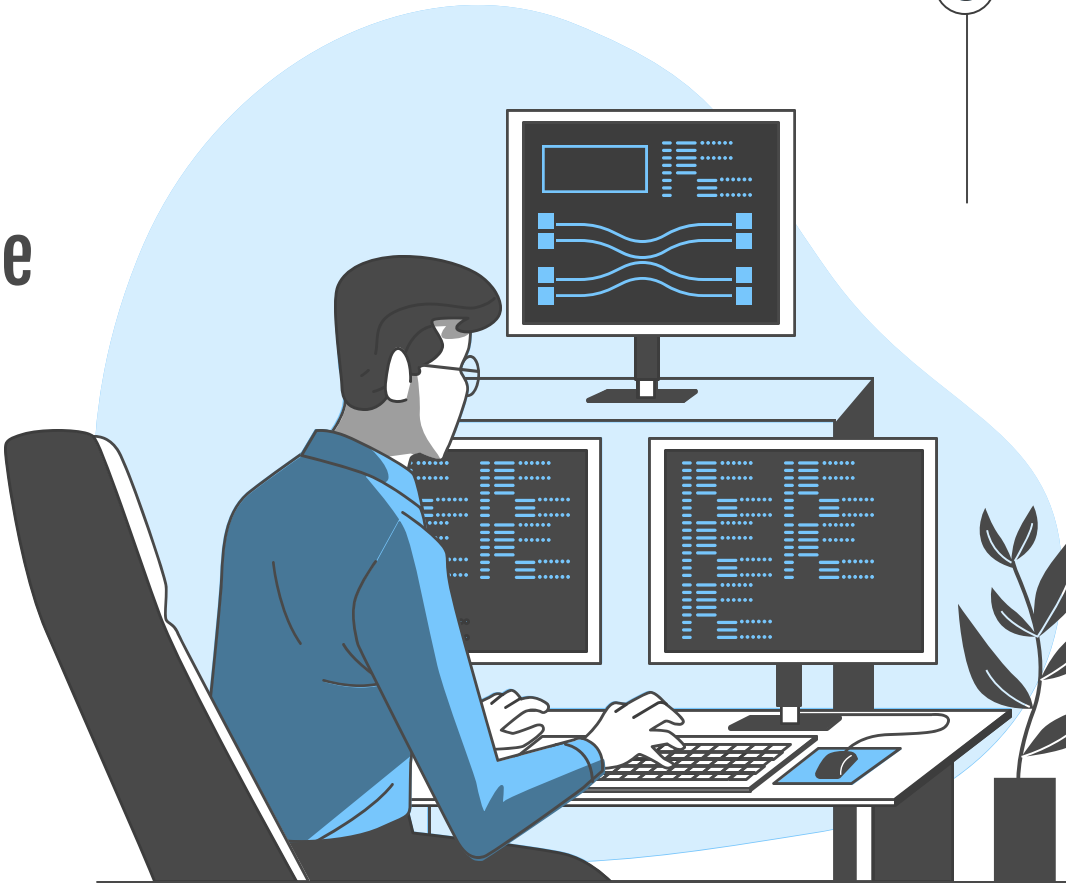
networks:                               # Se generan las dependencia para establecer la red
  app-network:
    driver: bridge
```

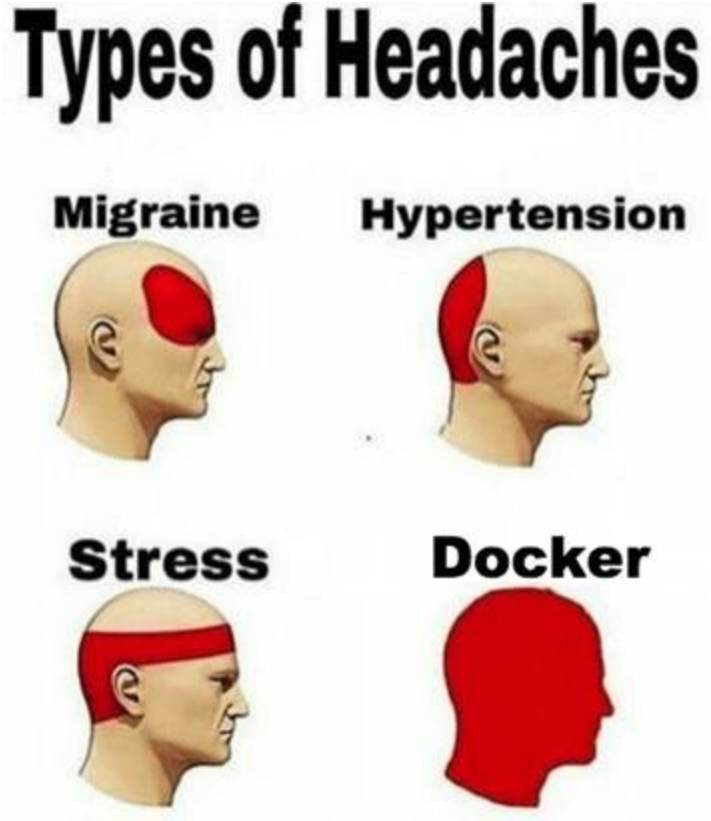
Comandos típicos para Docker-Compose



- docker compose down
- docker compose build
- docker compose up
- docker compose ps
- docker system prune
- docker ps
- docker exec -it <CONTAINER ID> bash
- sudo docker exec -it <CONTAINER ID> sh

Probemos una demo de Docker





La containerización es una ganancia a todo plazo

Dudas?

01

Proyecto

Conceptos
Avances
Ideas

02

Administrativo

Plazos
Notas
Programa

Y más allá

- Ejecutar Docker en modo NoRoot (+ Seguridad):
  - https://docs.docker.com/engine/security/rootless/
  - https://www.thegeekdiary.com/run-docker-as-a-non-root-user/ (Método 2)
- Kubernetes: https://kubernetes.io/es/docs/concepts/overview/what-is-kubernetes/
- Amazon Web Services ECS: https://aws.amazon.com/es/ecs/

Recursos

- https://www.citrix.com/en-in/solutions/app-delivery-and-security/what-is-containerization.html
- https://www.redhat.com/en/topics/cloud-native-apps/what-is-containerization
- https://www.veritas.com/information-center/containerization
- https://www.channelfutures.com/open-source/docker-downsides-container-cons-to-consider-before-adopting-docker
- **https://docs.docker.com/get-started/overview/**
- https://koukia.ca/why-docker-pros-and-cons-949d104478c5
- **https://docs.docker.com/develop/develop-images/dockerfile_best-practices/**
- **https://github.com/daleal/docker-walkthrouhg**
- https://dockerlabs.collabnix.com/docker/cheatsheet/