Docker: Fundamentos y su uso en la nube

¿Qué es Docker?

Docker es una plataforma de código abierto para gestionar **contenedores**

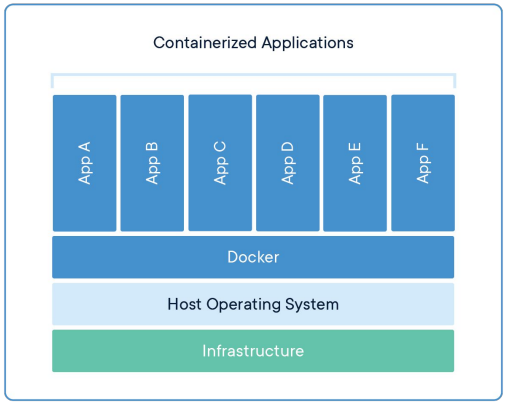
Contenedores: componentes estandarizados que son preparados para ser ejecutables independientemente de su entorno

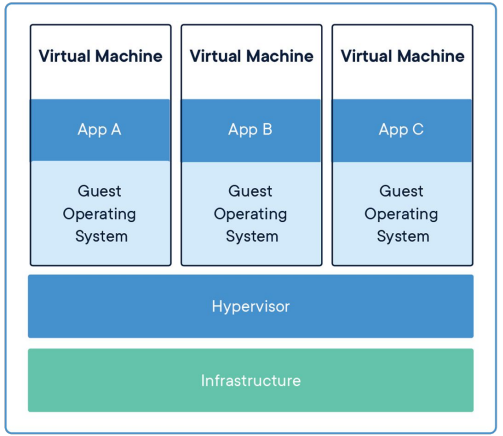
Contenedores Docker

- Unidad estándar de SW
- Reúne código y sus dependencias
- File system basado en capas
- Los contenedores Docker virtualizan el SO

Docker vs VMs

- Docker Pros
  - Fácil de usar
  - Muy ligero
  - Fácil de desplegar y compartir
- Docker Cons
  - No tan buena performance
- VMs Pros
  - Muy extendido (cada vez menos)
  - Útil para servicios de infraestructura
  - Sobre hipervisores (muy buenos gestionando HW)
- VMs Cons
  - Complejo, difícil de mantener y compartir





Componentes básicos de Docker

- Docker Engine
- Imágenes Docker
- Docker Hub

Docker Engine

Docker Engine actúa como una aplicación cliente-servidor con:

- Un servidor con un proceso daemon `dockerd`.
- API que especifican interfaces que los programas pueden usar para comunicarse con el daemon de Docker.
- Un cliente de interfaz de línea de comandos (CLI).

La CLI usa las API de Docker para controlar o interactuar con el daemon de Docker a través de secuencias de comandos o comandos directos de la CLI. Muchas otras aplicaciones de Docker utilizan la API y la CLI subyacentes. El daemon crea y administra objetos Docker, como imágenes, contenedores, redes y volúmenes.

Docker Images

- Son un tipo de contenedor
- Más bien como una instantánea, un molde
- Tienen capas intermedias reutilizables
- Para listar tus imágenes: `` `$ sudo docker images` ``

Docker Hub

- Repositorio público de imágenes docker
- https://hub.docker.com/search?q=

Comandos básicos Docker

- docker pull
  - Descargar imágenes Docker de un repositorio (por ejemplo Docker Hub)
- docker images
  - Listar las imágenes Docker que tenemos descargadas en nuestro ordenador
- docker run
  - Ejecutar un contenedor Docker a partir de una imágen

Comandos básicos Docker

- docker ps
    - Listar los contenedores Docker corriendo en tu ordenador

https://docs.docker.com/engine/reference/commandline/ps/

```
Usage: docker ps [OPTIONS]

List containers
Options:
  -a, --all             Show all containers (default shows just running)
  -f, --filter value    Filter output based on conditions provided (default [])
      --format string   Pretty-print containers using a Go template
      --help            Print usage
  -n, --last int        Show n last created containers (includes all states) (default -1)
  -l, --latest          Show the latest created container (includes all states)
      --no-trunc        Don't truncate output
  -q, --quiet           Only display numeric IDs
  -s, --size            Display total file sizes
```

Comandos básicos Docker

- docker exec
  - Ejecutar un comando dentro de un contenedor que ya está corriendo

```
Usage:  docker exec [OPTIONS] CONTAINER COMMAND [ARG...]

Run a command in a running container

  -d, --detach          Detached mode: run command in the background
  --detach-keys         Override the key sequence for detaching a container
  --help                Print usage
  -i, --interactive     Keep STDIN open even if not attached
  --privileged          Give extended privileges to the command
  -t, --tty             Allocate a pseudo-TTY
  -u, --user            Username or UID (format: <name|uid>[:<group|gid>])


Example:

docker run --name ubuntu_bash --rm -i -t ubuntu bash
docker exec -d ubuntu_bash touch /tmp/test
```

Comandos básicos Docker

- Parar contenedores: docker stop / docker kill

```
stop: SIGSTOP (Pide parar) | kill: SIGKILL (para sin pedir)

Usage:  docker stop [OPTIONS] CONTAINER [CONTAINER...]

Stop one or more running containers

Options:
      --help       Print usage
  -t, --time int   Seconds to wait for stop before killing it (default 10)
```

```
Usage:  docker kill [OPTIONS] CONTAINER [CONTAINER...]

Kill one or more running containers

Options:
      --help            Print usage
  -s, --signal string   Signal to send to the container (default "KILL")
```

Docker en la nube

Todo esto está muy bien, pero ¿cómo aplico esto a la nube?

Tenemos distintas opciones

- Utilizar docker directamente sobre una instancia
- Utilizar un orquestador:
  - ECS
  - Fargate
  - Kubernetes

Docker en la nube

En cualquier caso, hay muchas variables a tener en cuenta

- Dimensiones de las máquinas
- Requisitos mínimos/óptimos de los contenedores
- Capacidad de dar alta disponibilidad
- Necesidad de dependencias
- Necesidad de Internet…

Docker en la nube

¿De dónde saco las imágenes Docker?

- Si tenemos acceso a internet, podemos recurrir al Docker Hub
- Si queremos utilizar imágenes propias o no tenemos acceso a internet, recurriremos a ECR (Elastic Container Registry)

Docker en la nube

¿Cómo satisfacer las necesidades de un contenedor?

- Esto se aprende a base de prueba y error
- Nos valemos de pruebas de carga y performance
- Dependiendo de las características del contenedor, utilizaremos un tipo de instancia u otro

Docker en la nube

Limitar recursos de los contenedores

- Por defecto, los contenedores se asignan recursos de forma dinámica
- Memoria: flag -m
  - `docker run -m 512m nginx`
- Cpu: flag --cpus
  - `docker run --cpus=2 nginx`

Docker en la nube

Empezamos por lo básico:

- Levantamos una instancia con acceso a Internet
- Descargamos Docker
- Descargamos una imagen de Docker Hub
- Levantamos un contenedor

ALBAÑILESDIGITALES

Developing your new career

servicio navarro de empleo

das·nano

veri·das