
GEEKSHUBS

Empezando con Docker

¿Qué es un contenedor?

- Conocido como **virtualización ligera** o de sistema operativo.
- Los contenedores comparten **el mismo kernel del host**.
- Es una técnica que permite múltiples user spaces **aislados e independientes entre ellos.**

Contenedores… me suenan

- **Chroot**
  - FS
- **BSD Jails**
  - Users
  - Process
  - Netoworking
  - FS
- **SUN Solaris Zones**
  - ZFS
- **IBM AIX Wpars**
  - P/series
  - Hardware
- **Linux OpenVz / Virtuozzo**
  - Hosting
  - Proxmox
  - Módulos no oficiales del kernel.

Los grandes llevan tiempo usándolos

- **Los grandes sistemas distribuidos**
  - Consolidación de recursos
- **Google** (2007)
  - LMCTFY
- **IBM** (2007)
- **Hostings compartidos** (2006)

¿Por qué ahora están tan de moda?

- **Módulos oficiales en kernel de Linux** que permiten crear contenedores seguros
  - Kernel 3.8 (FEB 2013)
    - **CLONE_NEWUSER** ()
      - El ID de usuario y el grupo pueden ser diferentes dentro de un namespace
  - Kernel 3.12 (Nov 2013)
    - **Unprivileged containers**
      - Los contenedores no pueden acceder directamente al hardware
- **Momento adecuado**
  - Crecimiento de internet
  - Servicios más grandes que hace unos años
- **El cloud público y la virtualización**
  - Consolidación y precio

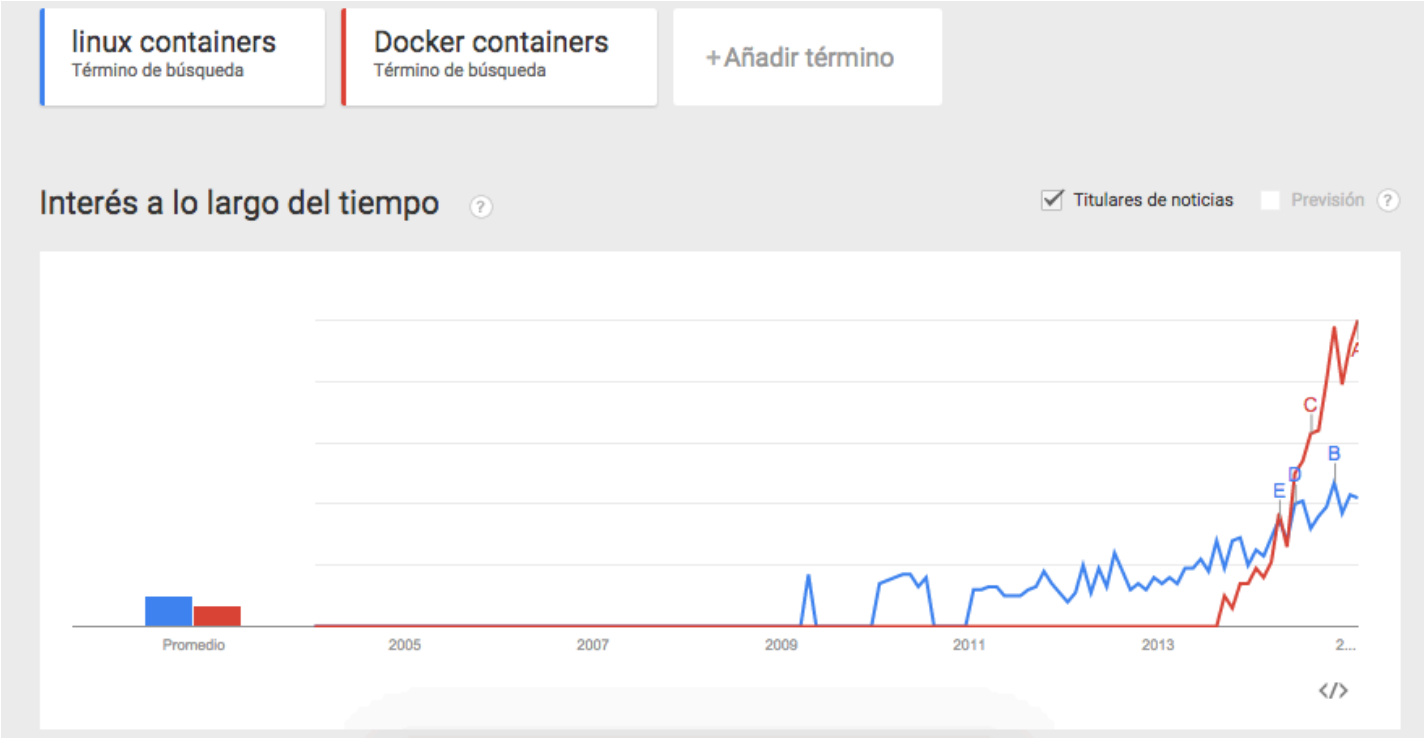
¿Acabará con la virtualización tradicional?

- La respuesta es: **no**
  - **El mismo kernel que el HOST**
- En la combinación de tecnologías suele estar la clave.

¿Cómo se está apostando por los contenedores?

- **Google**
  - Kubernetes
  - GCE containers
- **Microsoft**
  - Azure Docker
  - Mark Russinovich
  - Containers en Windows
- **Amazon**
  - Servicio de containers

Docker está de moda



Centrándonos en Docker

- **Application centric**
  - **No es una VM es una APP**
- **Ciclo de vida de la aplicación**
- **Versionado del contenedor**
- **Red social**
- **Consolidación de recursos**
- **Gestión de desastres**

Stack de Docker

- **Repo**
  - Publico (Demo)
  - Privado
  - Propio
- **Demonio**
  - CLI
  - API

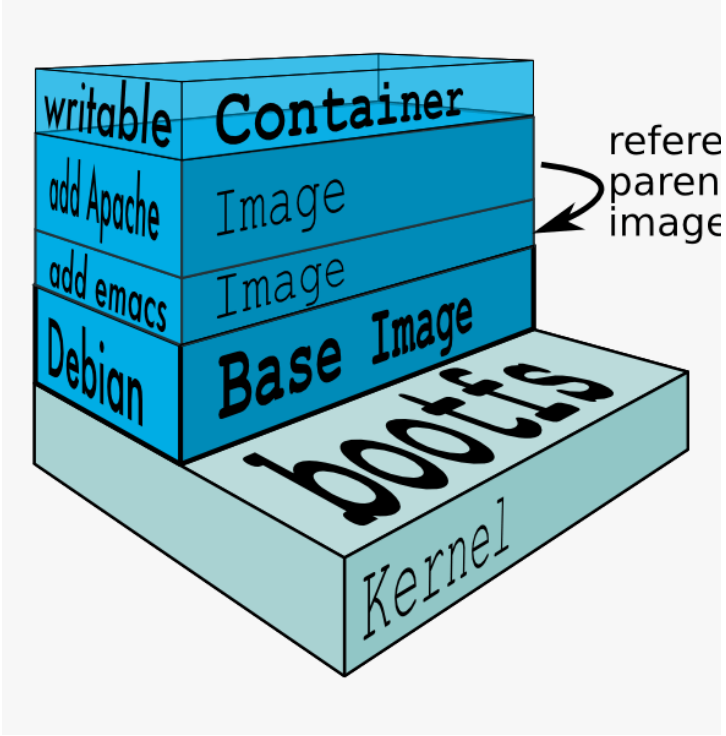
- **Libcontainer**
  - LXC y Cgroups
    - Namespaces
      - **Mount** (**CLONE_NEWNS**, Linux 2.4.19)
      - **UTS** (**CLONE_NEWUTS**, Linux 2.6.19)
      - **IPC** (**CLONE_NEWIPC**, Linux 2.6.19)
      - **PID** (**CLONE_NEWPID**, Linux 2.6.24)
      - **Network** (**CLONE_NEWNET**, started in Linux 2.4.19 2.6.24 and largely completed by about Linux 2.6.29)
      - **User** (**CLONE_NEWUSER**, started in Linux 2.6.23 and completed in Linux 3.8)

Arrancando una app Docker

- **Pull de la imagen**
  - Se chequea si la imagen existe en local
  - Si no existe se baja del hub
- Crea un **nuevo container**
- **Crea un filesystem** y monta una capa nueva de R/W
- **Asigna un interfaz** o un bridge de red
- **Configura una IP**
  - Busca y adjunta una IP disponible en un pool definido
- **Ejecuta el proceso de inicio** que hayamos configurado
- **Captura y provisiona la aplicación**
  - Conecta el estándar input, output y error para chequear el estado de la aplicación

Filesystem

- Storage layers
- Arrancamos R/O FS
- Union mount para R/W FS



Networking

- Iptables
- Brctl
  - Bridges

- docker run -t -d -**p 8081:5000** contenedor1
- **-b BRIDGE** or --bridge=BRIDGE
- **--icc=true|false** comunicación entre contenedores

Monitorización

- **Cgroups**
  - CPU
    - /sys/fs/cgroup/cpuacct/docker/***ID***/cpuacct.stat
  - Memoria
    - /sys/fs/cgroup/memory/lxc/***ID***/memory.stat
  - I/O
    - /sys/fs/cgroup/blkio/docker/***ID***/blkio.sectors
    - /sys/fs/cgroup/blkio/docker/***ID***/blkio.io_serviced
    - /sys/fs/cgroup/blkio/docker/***ID***/blkio.io_sectors
- **Iptables**
  - Networking

Control recursos CPU

- Cada contenedor tiene 1024 recursos de CPU por defecto
- Los limites son forzados

- $ docker run -it --rm stress **--cpu 4**
- docker run -it --rm **-c 512 stress** --cpu 4
- docker run -it --rm --**cpuset=0,1 stress --cpu 2**
- systemctl set-property **docker-id.scope CPUShares=512**

Control recursos de memoria

- Cuidado con la Swap

- docker run -it --rm **-m 128m** fedora bash
- docker run -it --rm -m 128m stress --vm 1 --**vm-bytes 128M** --vm-hang 0

Control de recursos de disco

- Velocidad acceso
- Capacidad de volumen

- **systemctl** set-property --runtime docker-d2115072c442b0453b3df3b16e8366ac9fd3defd4cecd182317a6f195dab3b88.scope "**BlockIOWriteBandwidth**=/dev/mapper/docker-253:0-3408580-d2115072c442b0453b3df3b16e8366ac9fd3defd4cecd182317a6f195dab3b88 **10M**"
- docker -d **--storage-opt dm.basesize**=5G

Relación con Systemd

- Ayuda a crear dependencias de un container con otro
- Ayuda a la gestión recursos
- Facilita la gestión de “microservicios”

Dockerfile: APP personalizada

- Desplegar o crear una app custom a partir de otra

- **INSTRUCTION arguments**
- **ENV** variable de entorno
- **ADD** copiar un fichero de host o URL (opciones avanzadas)
- **COPY** copiar un fichero
- **WORKDIR** como un CD
- **EXPOSE** como el parametro -p , expone un puerto al host
- **VOLUME** monta un volumen o directorio del host
- **USER** el usuario que correrá el contenedor
- **ENTRYPOINT** el programa que se ejecuta al arrancar el contenedor

Seguridad

- Ayuda aislar un proceso de otros
- ¿Es seguro?
  - Docker HUB se basa en una relación de confianza
  - CVE-2014-6407
  - CVE-2014-6408

Docker y los sistemas distribuidos

- Mesosphere
- Kubernetes

Demo Docker

- AL FIN ¡LLEGA LA ACCIÓN!

¿PREGUNTAS?

¡GRACIAS!

GEEKSHUBS

Colaboradores

iukanet