

Clase 7-Contenedores -docker

miércoles, 30 de abril de 2025 18:06

Plataforma para hacer contenedores, mete todo junto lo que necesita el programa (librerías, etc)

Tiene:

- Docker daemon: servidor, crea, ejecuta y monitorea los contenedores
- Api: especifica interface que los programas pueden usar para interactuar con el sv
- Cli:cliente: permite interactuar con el sv mediante comandos, es lo que en general decimos que es docker creo

Está bueno para

- desarrollo, testing(pq es facil cambiar las versiones de cosas y demás)
- Escalado y despliegue: puedo tener varios contenedores funcionando en simultáneo en una máquina
- + rápido que mv, + simple

Arquitectura [cliente servidor](#).

Cliente y sv pueden estar en el mismo sistema o que estén en distintas redes.

Sv escucha api requests, cliente manda http

Cliente y sv se distribuyen como binarios.

Docker desktop es interfaz gráfica

Namespaces: nos permite aislar procesos, espacio de trabajo para el container

Control groups: para limitar lo que puede usar un contenedor. No es obligatorio

Union file system: no es filesystem en si, permite montar directorios como uno sobre otro para que se ven todos como si fuera uno solo. Si hay dos archivos de igual nombre muestra solo el de arriba.

Las capas de abajo son solo readonly. Puedo hacer como una truchada para borrar un archivo de abajo pero es borrado lógico

Contenedores se crean a partir de [imágenes](#) que son como moldes de solo lectura para construir el contenedor. Imagen es conjunto de capas

[Registry](#): almacén de imágenes. Puede ser pública o privada. Ahí la gente sube imágenes

[Dockerfile](#): archivo de texto con los pasos para construir una imagen. Podemos modificarlo para customizarlo

Cada imagen tiene varias capas. Las agarra el union file system, las junta y me las muestra como si fuera uno solo.

Las capas se pueden reutilizar entre imágenes.

Cuando creamos el contenedor es que se crea la capa de lecto-escritura. Sino no. Ahí es donde metemos los cambios temporales. La imagen no tiene esa capa, son todas menos ella

Le damos la imagen al contenedor y se crea la capa escribible. Cada capa se extrae el contenido en un directorio del filesystem del nodo

Cuando ejecutamos el contenedor desde una imagen se genera un union donde se apilan las capas

Cuando usamos chroot se le dice al contenedor este es tu directorio raíz, a partir de acá ves.

Con [dockerfile](#) creamos un contenedor a partir de otro

```
FROM ubuntu:24.10
MAINTAINER user1 <user1@example.com>
RUN apt-get -y update && apt-get -y install nginx
COPY index.html /usr/share/nginx/html/
ENV APP_USER=nginx-user
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

Parte de ubuntu,
Maintainer es solo data para el contenedor, no crea capa nueva
El run apt si crea una capa, instala un server http (crea un nuevo directorio donde queda todo eso)
Copy: crea copia del host al contenedor
Hay que indicar que version estamos usando

Contenedor es una instancia de una imagen.

A partir de la imagen creo los contenedores que necesito, por eso tiene que ser inmutables las capas
Principal diferencia entre contenedor e imagen es a capa escribible.

Cuando borras el contenedor, la capa escribible se elimina tb. Las capas de abajo no.

Contenedores se pueden iniciar, parar, pausar borrar desde la cli de docejer.

Cada uno es autónomo y corre su propio entorno

Para persistir datos:

Docker permite pasarle al contenedor cuando lo creo un espacio de disco del host anfitrión para que pueda escribir ahí. Cuando se elimina el contenedor ese no se borra.

- Volumenes: espacios de discos manejados por docker
- Bind mounts: no manejado directamente por docker, pueden modificarse por procesos no de docker. Si muevo el contenedor, tengo que tener cuidado de no cagarla

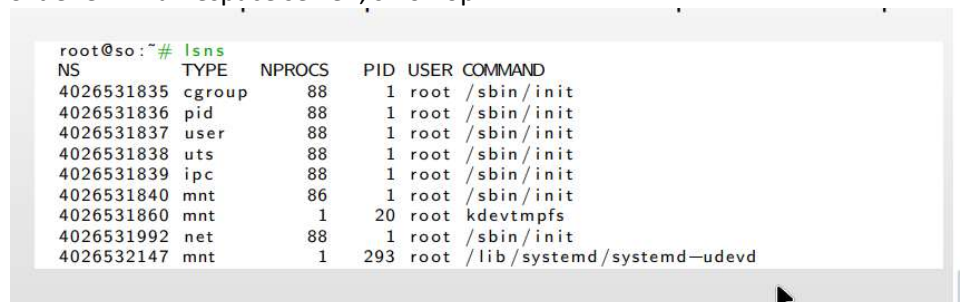
Redes:

Contenedores tienen por defecto el networking habilitado or default, puede hacer conexiones salientes.

Puede estar conectado a 0, 1 o varias redes. Tiene que publicar el puerto para hacer disponible un servicio

Dos contenedores del mismo host no pueden publicar el mismo puerto.

Si tienen = namespace se ven, sino nop



```
root@so:~# lsns
NS      TYPE      NPROCS   PID  USER  COMMAND
4026531835 cgroup    88       1    root  /sbin/init
4026531836 pid       88       1    root  /sbin/init
4026531837 user      88       1    root  /sbin/init
4026531838 uts       88       1    root  /sbin/init
4026531839 ipc       88       1    root  /sbin/init
4026531840 mnt       86       1    root  /sbin/init
4026531860 mnt       1        20    root  kdevtmpfs
4026531992 net       88       1    root  /sbin/init
4026532147 mnt       1        293   root  /lib/systemd/systemd-udevd
```