[Docker SecDevOps] Imágenes Capítulo 3

intro

- casi cada comando del fichero **Dockerfile** genera una capa en la imagen que se está generando
- cada capa es de *solo lectura*, inmutable
- cada capa es cacheada, de manera que se genera la primera vez y después, se utiliza (ahorrando proceso y espacio)
- se puede consultar las capas de una imagen con `docker image history <imagen>`
- cuando se lanza un contenedor, se crea una nueva capa, esta de **lectura/escritura**

manejo de imágenes

- **docker image**
  - build
  - history
  - import
  - inspect
  - load
  - ls
  - prune
  - pull
  - push
  - rm
  - save
  - tag

pull

descarga del repositorio (por defecto, hub.docker.com) la imagen solicitada

- `docker image pull <imagen>` (latest)
- `docker image pull <imagen>:<version>`

push

publicar imagen en repositorio

- por defecto, se hará sobre el registro oficial (hub.docker.com) y para ello hace falta tener una cuenta y estar autenticado:
  - `docker login -u <username>`
  - `docker info` : para saber con que cuenta estamos logeados (en caso de tener más de una)
  - para publicar una imagen, el formato del nombre de la imagen ha de ser: <usuario>/<imagen>:<version>

registro

- código abierto bajo licencia apache

- aplicación servidor sin estado y altamente escalable
- al hacer un pull sobre un registro (que no se sea por defecto), la imagen debe tener el formato <registro>/<usuario>/<imagen>:<version>
- para lanzar un registro local: `docker container run -d -p 5000:5000 -rm –name registry registry:2`
- el registro además ofrece notificaciones webhook, API-REST, etc...

Limpieza de imágenes

- `docker image rm <imagen>` : borrado individual
- `docker image prune` : borrado de las imágenes colgadas (dangling) → cuando creamos imágenes sin especificar versión (*latest*) y se hacen modificaciones, las anteriores quedan colgadas.
    - `-a` : todas las imágenes que no son usadas
- las imágenes referenciadas por contenedores, corriendo o no, no pueden ser borradas

buenas prácticas creando imágenes

- usar imágenes pequeñas: elegir una imagen base lo más pequeña posible
    - alpine : distribución linux minimalista
    - busybox : https://hub.docker.com/_/busybox
    - Scratch : imagen de 0 bytes
- usar imágenes especializadas : en lugar de «construirlas» nosotros mismos partiendo una imagen generalista
- especificar la versión de la imagen base
- reducir el número de instrucciones en los ficheros Dockerfile
- borrar los ficheros que no se necesitan
- aprovechar la multifase
- reducir el número de ficheros enviado al demonio docker : *.dockerignore*
- el orden (de las instrucciones en Dockerfile) importa : dejar las instrucciones que se supone no variaran al principio del fichero. Docker invalida una capa (y todas las siguientes) cuando hay un cambio, así que ese cambio que afecta a esa capa, afecta a todas las herederas de la misma
- reusar imágenes