Managing Docker Container Images

Download, view its history, tag, delete, view, clean up, for **IMAGES**

Layering Mode of a Container

- All docker images based on the same image, use disk space only to take up storage for added files to the base image.
- The layers in an image are read-only and each new layer is DELTA of any changes compared to the layer beneath it.
- Similar as Git control system. This speed ups the rebuilding images after small change has been made.

Download Images

```
docker pull nginx
# takes an image name as an argument
# it downloads that img to the local host system
```

```
# check images
docker images
```

```
# see how img was constructed
docker histroy nginx
# the output shows each layer line by line

# Tags
docker tag --help
# Show useful information
# If no tag, show 'latest'

docker tag nginx:latest nginx:myglog_stable
# create two images with the same ID but different alias
```

Dockerfiles

Dockerfiles can range from being very simple to very complex.

Example simple dockerfile

```
FROM debian:buster-slim
LABEL maintainer='NGINX Docker Maintainers <docker-maint@nginx.com>"
```

```
ENV NGINX_VERSION 1.21.1
ENV NJS_VERSION 0.6.1
ENV PKG_RELEASE 1~buster

RUN apt-get update \
    && apt-get install -y nginx \
    && apt-get clean

RUN ln-sf /dev/stdout /var/log/nginx/access.log \
    && ln -sf /dev/stderr /var/log/nginx/error.log

COPY index.html /var/www/html

EXPOSE 80
cmd["nginx","-g","daemon off;"]
```

- FROM: base image
- dockerbuild means that a layer will be created on the top of the base image in your image
- LABEL: entry is a good way of offering contact details for the author
- ENV introduce environment variables, like bash to use
- RUN instruction tells docker to run a shell command to help build your image
- COPY command copies a file from the local system into the image at build time. This copy statement will put a copy of the index.html file in the /var/www/html directory. COPY command is most commonly used to copy small config files into an image
- EXPOSE entry is telling Docker to open up TCP port 80 when a container runs from the image.
- CMD instruction runs the nginx binary with two options from inside the container
- daemon off keep ngninx running in foreground of the container. Elsewhere the container would stop running as soon as it started

Generic Tips

- Having fewer layers is generally a good idea
- 'apt-get clean' is a good practice to add command on debian based distros

Build

- `docker build -t mynginx .`
- -t option allows specify a tag for the image
- docker rmi nginx

Remove Images

- docker rmi --help
- docker rmi -f nginx:latet -> Force
- docker rmi --no-prune -> To not dfelete untagged parents

Clearing disk Space

- Non tagged images listed as tag are called DANGLING IMAGES
- docker image prune -> Remove all dangling images
- docker system prune -a
    - stopped containers
    - networks not used by at least one container
    - All images with at least one container associated to them
    - all build cache
- docker system prune
    - all stopped containers
    - All networks not used by at least one container
    - all dangling images
    - all dangling build cache
- docker system df -> see disk space docker is using