Cloud computing

Containers and images: Docker core

Aleix Llusà Serra

Master's degree in Machine Learning and Cybersecurity for Internet Connected Systems
Universitat Politècnica de Catalunya
https://epsem.upc.edu

10 de setembre de 2026

1 Containers

2 Images

Docker TLDR

Mainly oriented to single application or service.

- Build: images
- Run: containers
  When created they begin from image (except data).

Documentation: `docs.docker.com` and Poulton (2020)

Lab: `https://labs.play-with-docker.com/?`

Containers: the basic

- Start a new container running a bash process
  `docker run -it debian:latest /bin/bash`
- Start a new container on background
  `docker run -d --name merit debian sleep infinity`
- Attach to a container
  `docker exec -it merit bash`
  Execute `ps axu` on the guest (If needed `apt install procps`)
- Show running containers: `docker ps`
  Also execute `ps axu` on the host and find the `sleep` process
- Stop container
  `docker kill merit` (SIGKILL)
  `docker stop merit` (SIGTERM)
- Show all containers: `docker ps -a`
- Remove container: `docker rm merit`

Containers: publishing ports

By default, containers ports are not visible from host nor externally.

- Web CMS with published ports:
  `docker run -d --name web --publish 8080:8080 plone:latest`
- Read the logs: `docker logs web`

Containers: internal linking

- MySQL managed with a phpmyadmin (with legacy link):

```
docker run -d --name db -e MYSQL_ROOT_PASSWORD=XXX
mysql
docker run -d --name admin --link db:db -p 8081:80
phpmyadmin
```

- MySQL managed with a phpmyadmin (with network):

```
docker network create database
docker run -d --name db -e MYSQL_ROOT_PASSWORD=XXX
--network database mysql
docker run -d --name admin --network database -p
8081:80 phpmyadmin
```

Image registry

- The most common: Docker Hub `https://hub.docker.com/`
- You can also have your own registry on your cloud provider or on-premises registry.

Images

Definition

An object that contains an OS filesystem, an application, and all application dependencies. It's like a virtual machine template. (Poulton 2020)

A read-only package that contains everything (except data). A single image can be used to start one or more containers, similar to classes.

Images: the basic

- List local images:
  `docker images`
- Download an image to local repository:
  `docker pull nginx:latest`
- By default from Docker hub:
  i.e. `docker pull docker.io/nginx:latest`

Dockerfile

- Reproducibility
- Infrastructure as code
- Multiple layers: shared between different images

Example: nginx image `https://hub.docker.com/_/nginx`

Dockerfile: build your own image

Build context: let `mkdir web; touch web/a.html; touch web/Dockerfile`

web/Dockerfile

```
FROM nginx:latest

COPY a.html /usr/share/nginx/html/
```

```
docker build -t own:latest web
docker run -d --name web --publish 8080:80 own:latest
```

We could then proceed to Docker Hub `docker push own:latest`

Base image OS

Some examples:

- FROM debian:latest (standard, 117 MB)
- FROM alpine:latest (7 MB, only one shell `sh`)
- distroless
  `https://github.com/GoogleContainerTools/distroless`
  Not for building, only for running [we'll see multistage]: no package managers, no shell, almost no anything
- FROM scratch (Not an image. Only if you ask how did Earth begin!)

Dockerfile: RUN

web/Dockerfile

```
FROM nginx:latest

RUN apt update && apt install -y \
    git \
    wget \
  && rm -rf /var/lib/apt/lists/*

RUN cd /usr/share/nginx/html/ \
  && wget https://www.w3schools.com/w3css/4/w3.css
COPY a.html /usr/share/nginx/html/
RUN nginx -t  # test configuration
```

Be aware of docker caching layers. If unsure: `docker build --no-cache -t own:latest web`

Dockerfile: COMMAND

web/Dockerfile

```
FROM nginx:latest

RUN [...]

COPY a.html /usr/share/nginx/html/

CMD ["nginx-debug", "-g", "daemon off;"]
```

Recall: 1 service/app per container! A foreground process! Logs to stdout!

If needed, one can replace the command. Recall:

```
docker run -d --name web own:latest sleep infinity
docker exec -it web bash
```

Dockerfile: EXPOSE

web/Dockerfile

```
FROM nginx:latest

RUN [...]
COPY [...]
CMD [...]

EXPOSE 1111
EXPOSE 2222/udp
```

Recall: exposed ports are not pusblished by default!

```
docker run -d --name web -p 8080:80 -p 8081:1111 own
```

Tagging images

Tagging/versioning

```
docker build -t own:25 web
docker tag own:25 own:latest

docker tag own:25 your.registry.test/own:25
docker build -t your.registry.test/own:26 web
```

References


Poulton, Nigel (2020). Docker deep dive. Zero to Docker in a single book. 2a ed. Birmingham: Packt Publishing. ISBN: 9781835087916. URL: https://discovery.upc.edu/permalink/34CSUC_UPC/8e3cvp/alma991005249877806711.