Docker

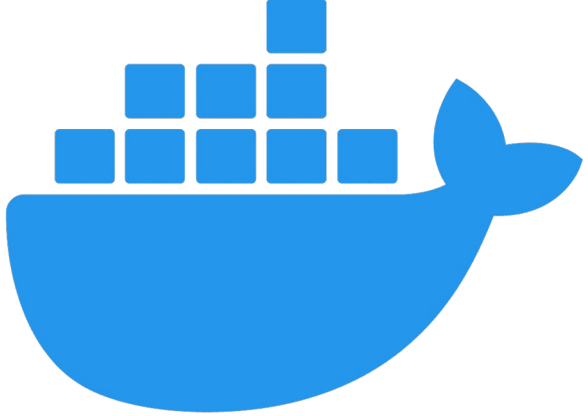
Additional Information

Distributed Systems 2022-2023

Docker

Basic concepts

- Images
- Containers
- Dockerfile
- Layers & Caching
- Docker compose
- Docker hub



docker

podman

Docker

Images

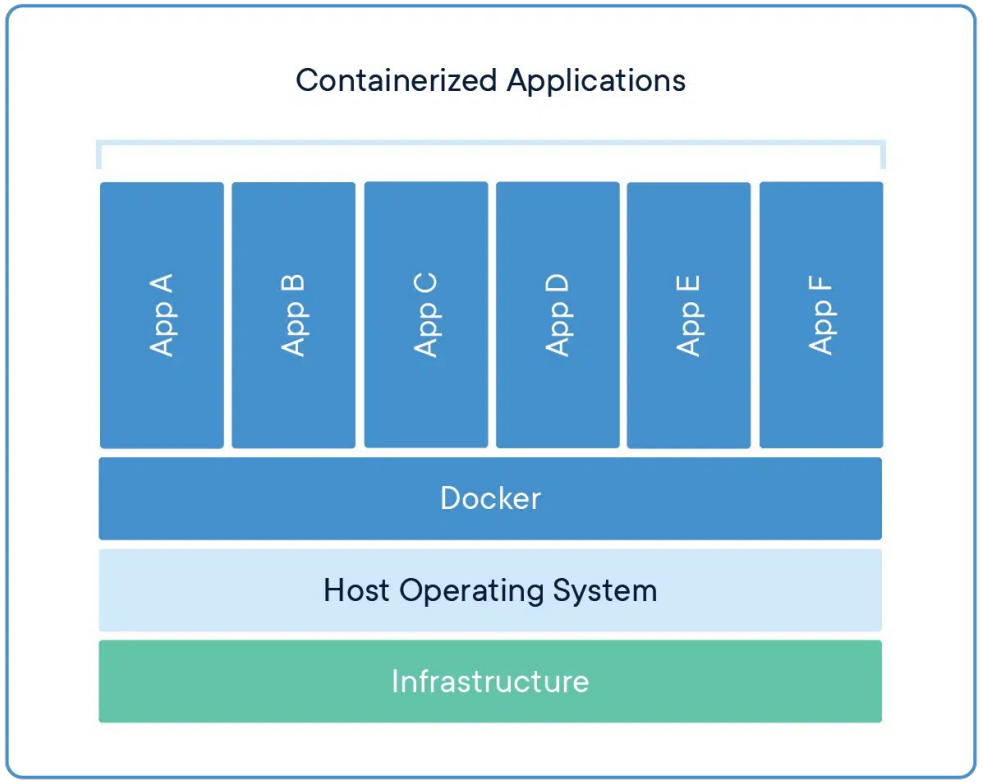
- Template for a container
- Immutable
- Created via a Dockerfile

Docker

Containers

- Lightweight
- Self-contained
- Shares OS kernel
- ≠ Virtual machines

≠



Docker

Dockerfile

- Heart of the image

- `$ docker build .` → *build the image*
- `$ docker run ...` → *create the container from the image*
- `$ docker stop ...`

- Typically built upon other images
  - `FROM …`

- Other keywords :
  - `COPY, RUN, EXPOSE, ENV, CMD, ENTRYPOINT, WORKDIR, ...`

Docker

Layers & Caching

- Dockerfile = stack of layers
  - `RUN, COPY` add layers
- Layers add overhead
  - Docker caches layers
- Pay attention to cache issues!
  - `COPY data/*` → Each time a single file is modified, all data is re-copied
  - `RUN apt upgrade`
    `RUN apt install ...` → outdated version might be downloaded
- Once a layer invalidates the cache, subsequent layers are rebuilt as well.

Docker

Docker compose

- On top of Docker → combines multiple services in one file
- docker-compose.yml
- `$ docker compose up`
- `$ docker compose stop`
- Suggested when having multiple containers

Docker

Docker compose

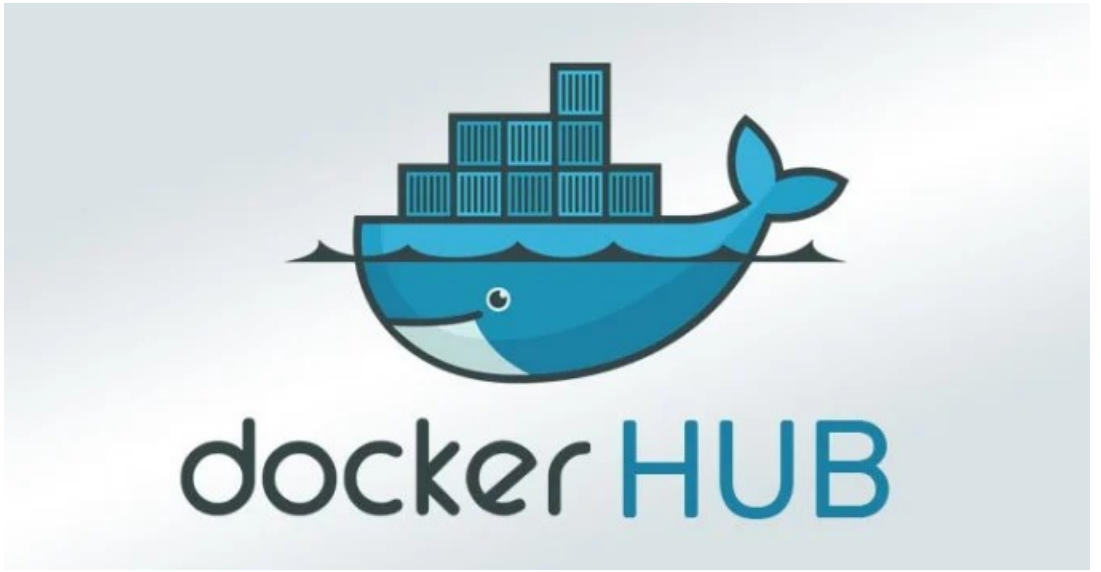
- Typical docker-compose.yml file

- ```
  services:
      microA:
          image: …
      microB:
          build: path
  ```
  ```

Docker

Docker hub

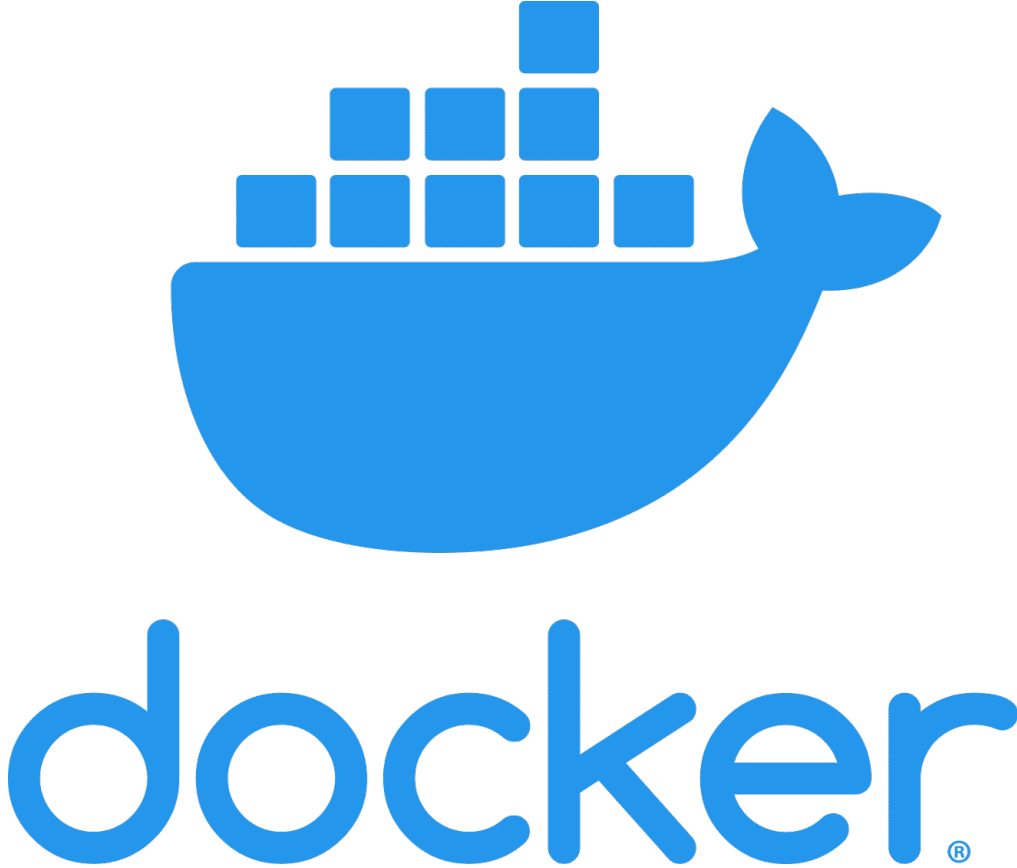
- hub.docker.com
- Many useful images
- Extend images easily
  - f.e., `FROM python:3.10-buster`
  - Try to do this as much as possible



Docker

Advanced concepts

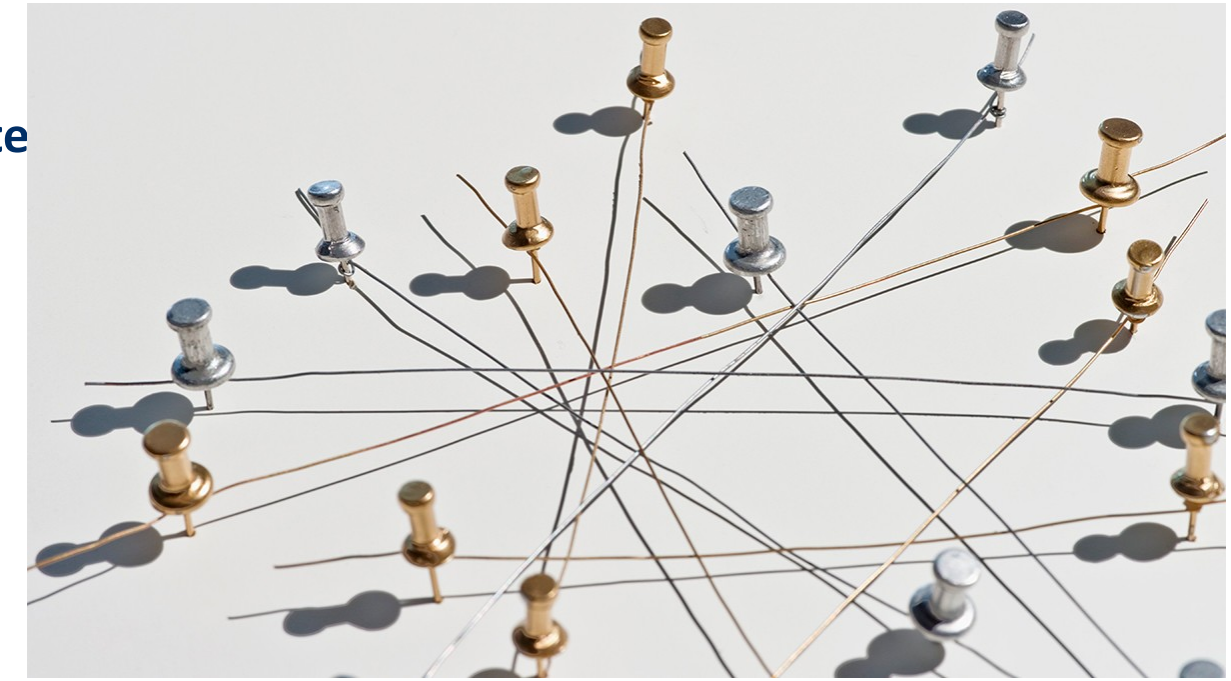
- Networks
- Ports
- Volumes
- RUN
- depends_on
- Startup commands



Docker

Networks

- Allow communications between containers
- Docker compose automatically generates one for application & connects containers to network
- By default each microservice has an assigned IP
  - Networks allow microservices to communicate by name rather than IP
  - Recommended! (IPs are not deterministic, can change between invocations)
- `$ docker network create netw`
- `$ docker network connect netw cont`



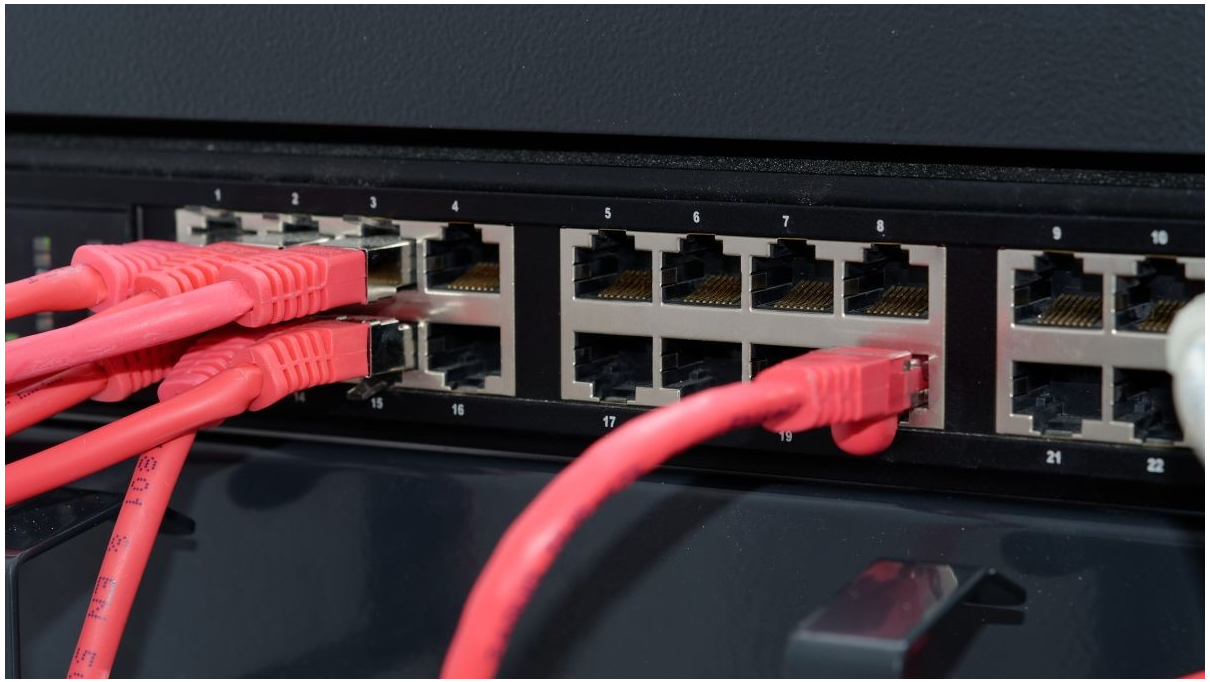
Docker

Ports

- Map container port on host port

- ```
  MicroA:
      ports:
          - 5001:5000
          - 1234:9876
  ```

  → localhost:5001 = MicroA:5000
  → localhost:1234 = MicroA:9876


  ```

Docker

Volumes

- Map container file paths on host file paths
- Two-way street!
- Can be used for:
  - Persisting data outside containers
  - Transfering files to containers
- `volumes:`
  `volumename:containerpath`
  `hostpath:containerpath`

Docker

Volumes

- `$ docker volume create ...`
- `$ docker volume inspect ...`
- `$ docker volume rm ...`

Docker

RUN

- Executes commands
- Creates extra layers → Group related `RUN` commands together (via &&)
- Two forms:
  - Shell form: `RUN <command>`
  - Exec form: `RUN [.., .., ....,..]`
- Shell form uses default shell, exec allows choice
  → exec form is recommended

Docker

depends_on

- Create dependencies between microservices

  → `MicroA` will start after `MicroB, MicroC` both started
  → `MicroB, MicroC` will stop after `MicroA` stopped

-
```
MicroA:
    depends_on:
      - MicroB
      - MicroC
```

- Does not check if microservice is ready!

Docker

Startup Commands

- `ENTRYPOINT & CMD`
  - Similar, but subtly different
  - Support both exec & shell form (see `RUN`)

- Best practice:
  - `CMD` for default command
  - Use exec form
  - Can combine `CMD & ENTRYPOINT` to provide default arguments.

Docker

Additional resources

- Dockerfile reference : https://docs.docker.com/engine/reference/builder/
- Docker-compose reference : https://docs.docker.com/compose/compose-file/
- Docker-compose getting started : https://docs.docker.com/compose/gettingstarted/
- Dockerfile best practice : https://docs.docker.com/develop/develop-images/dockerfile_best-practices/
- `CMD` in combination with `ENTRYPOINT` : https://docs.docker.com/engine/reference/builder/#understand-how-cmd-and-entrypoint-interact
- Dockerhub : https://hub.docker.com/