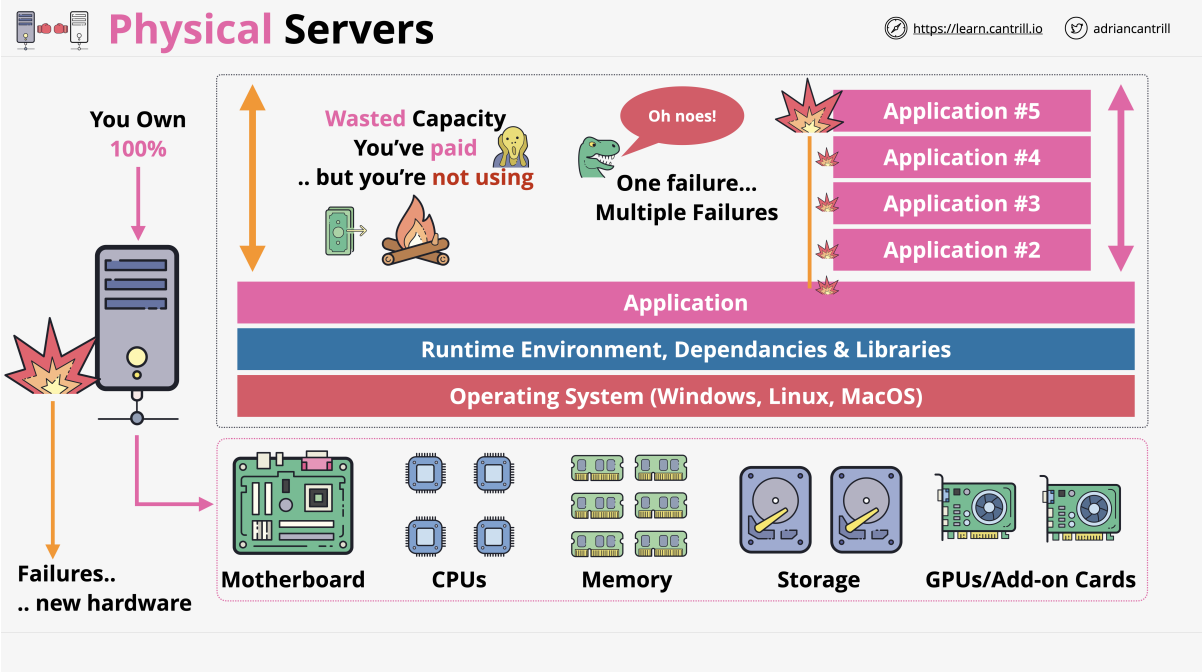
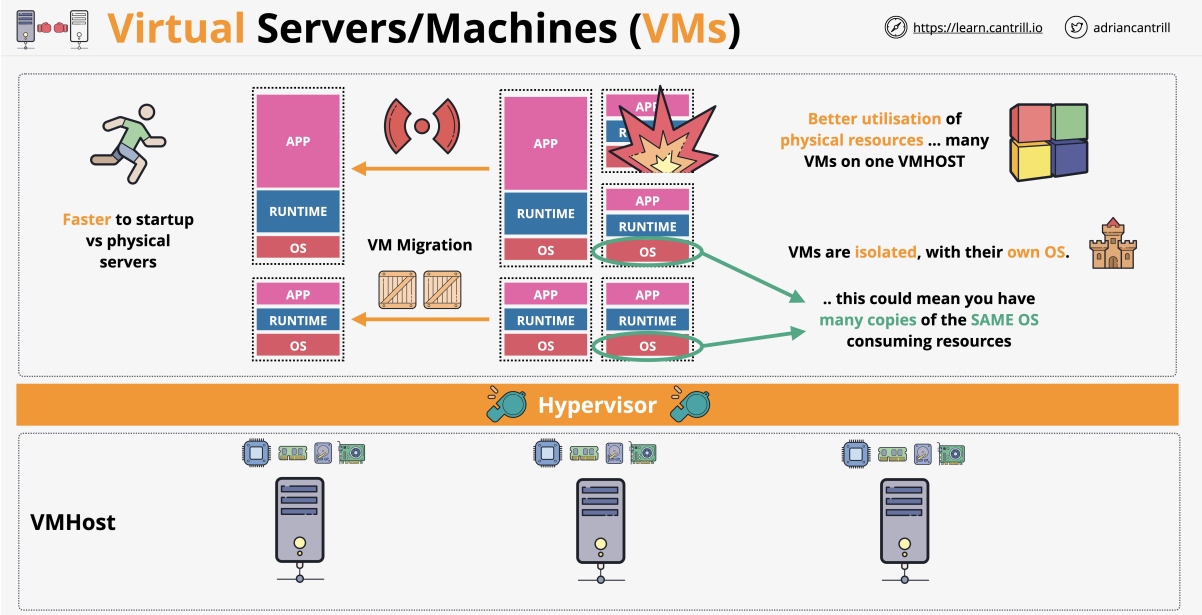
Docker Overview

▼ Repo

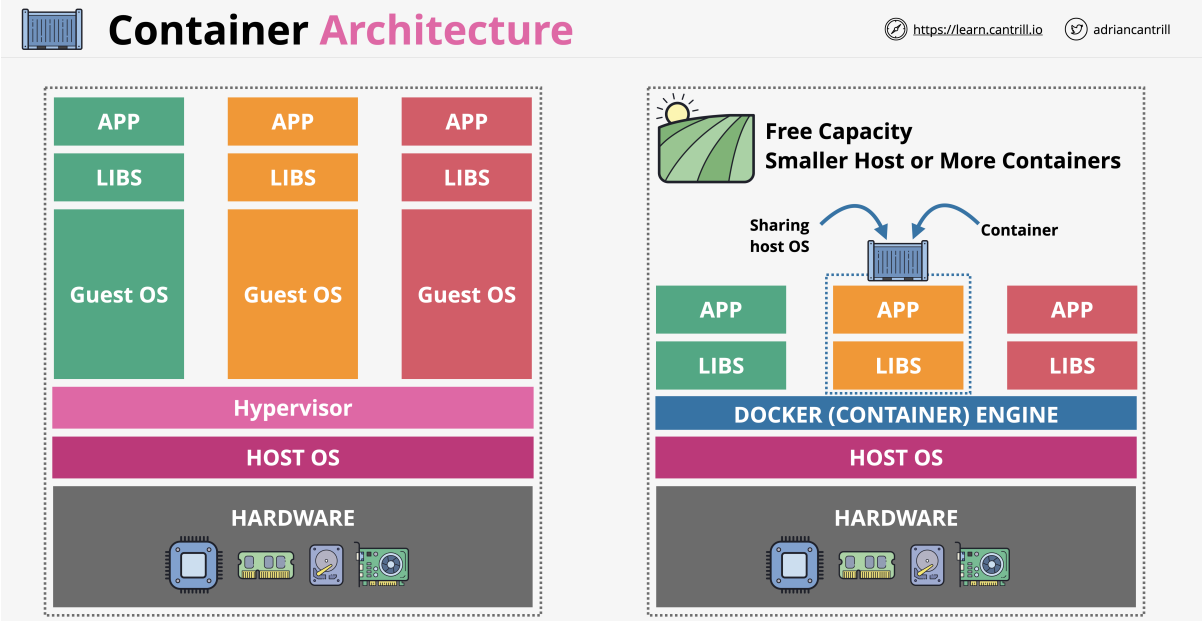
https://github.com/Fulim13/docker

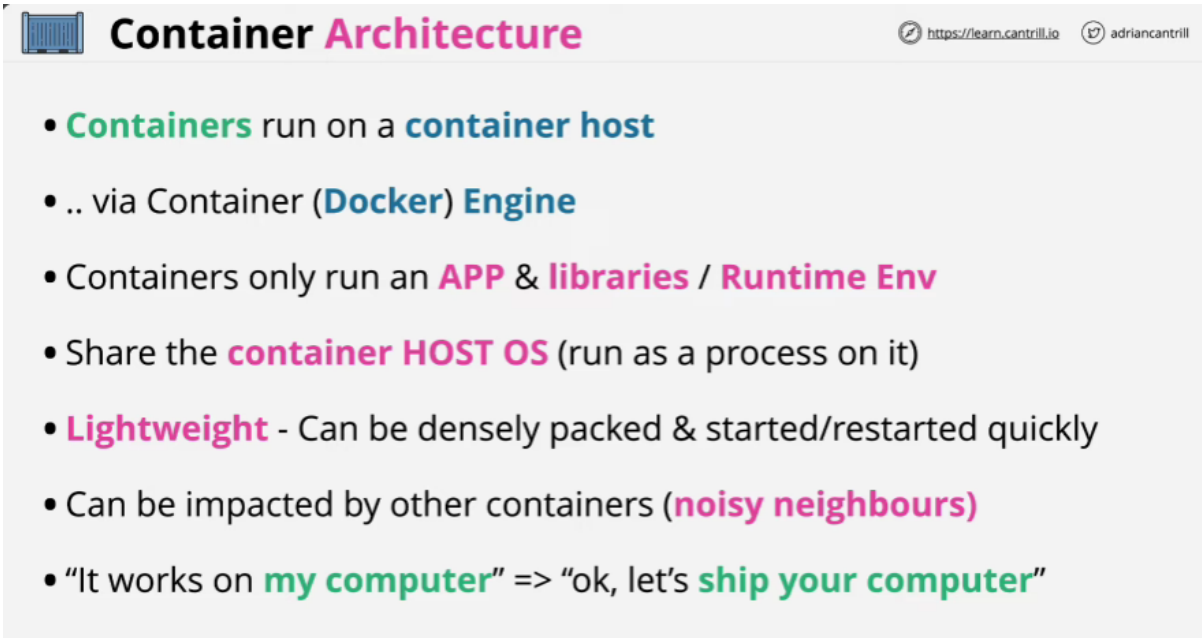
▼ Physical Servers vs Virtual Machines



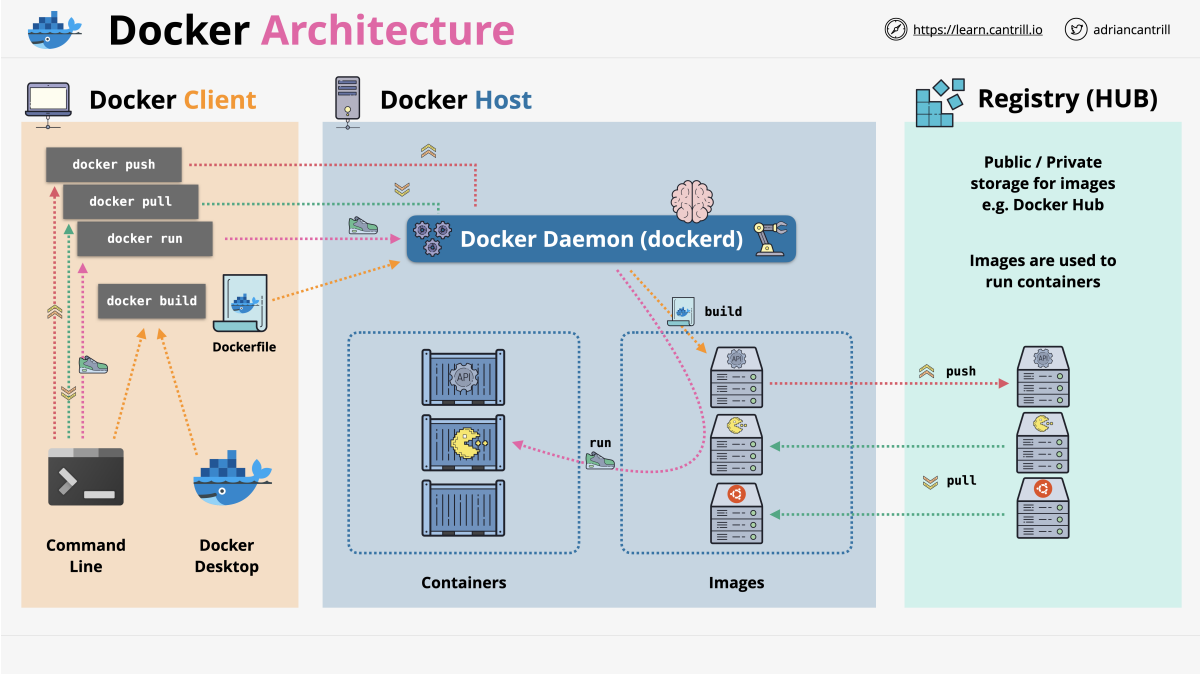


▼ Virtual Machines vs Container





▼ Docker Architecture

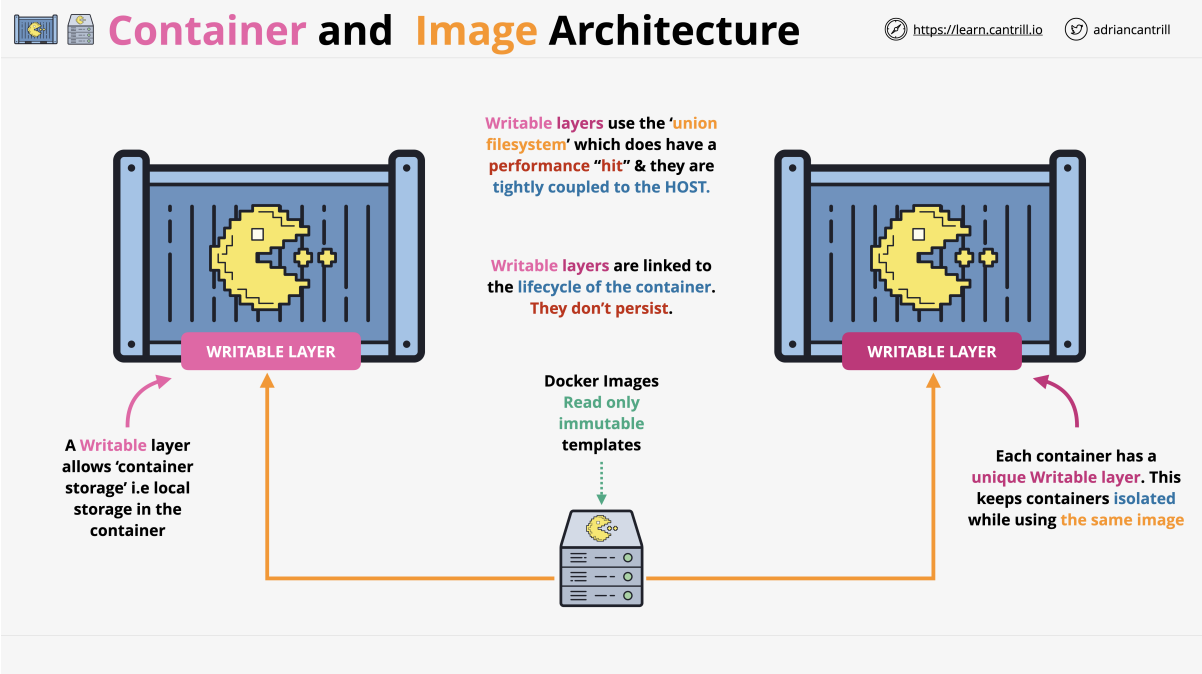


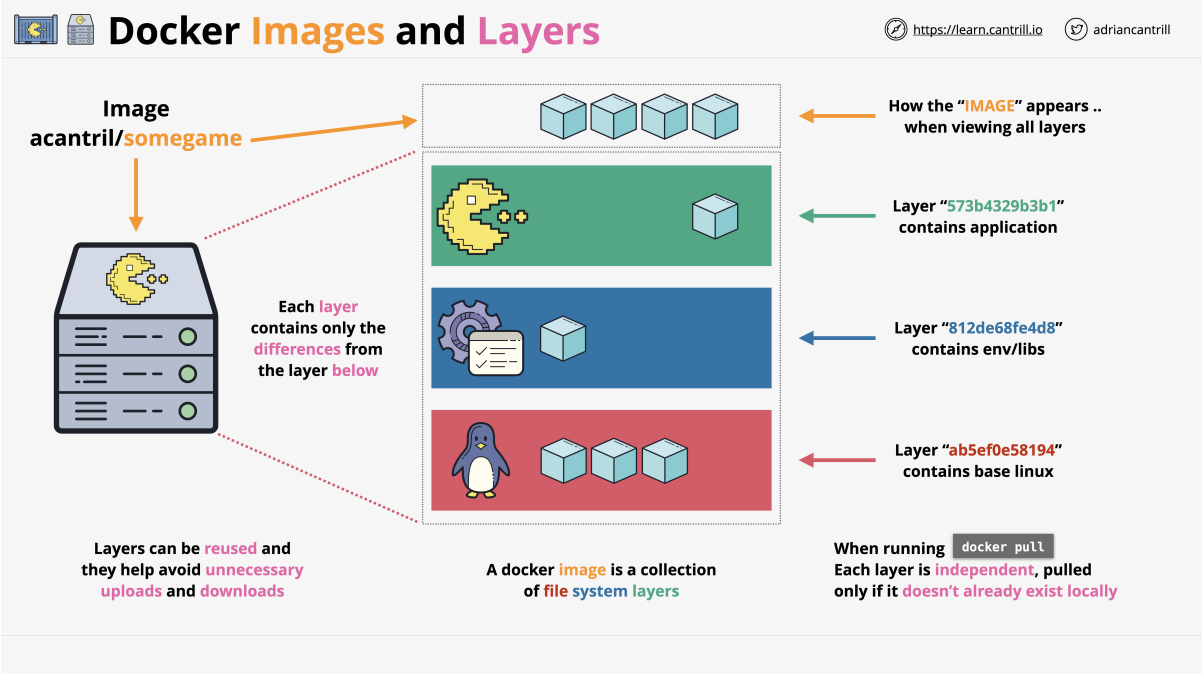
▼ [Demo] Interacting with Docker Engine

```
# run local docker images (if not found, pull from docker h
ub registry
```

```
docker run hello-world

# Check running containers
docker ps

# Check all containers
docker ps -a

# Check all images
docker iamges
```

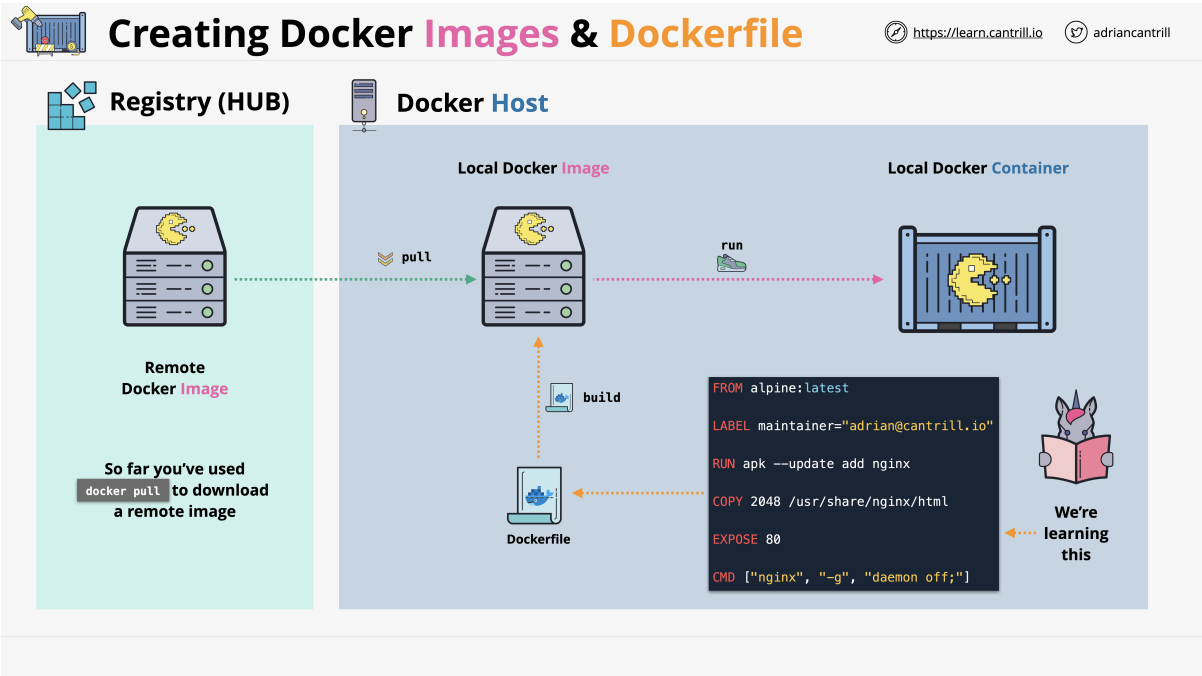
▼ Container and Image Architecture

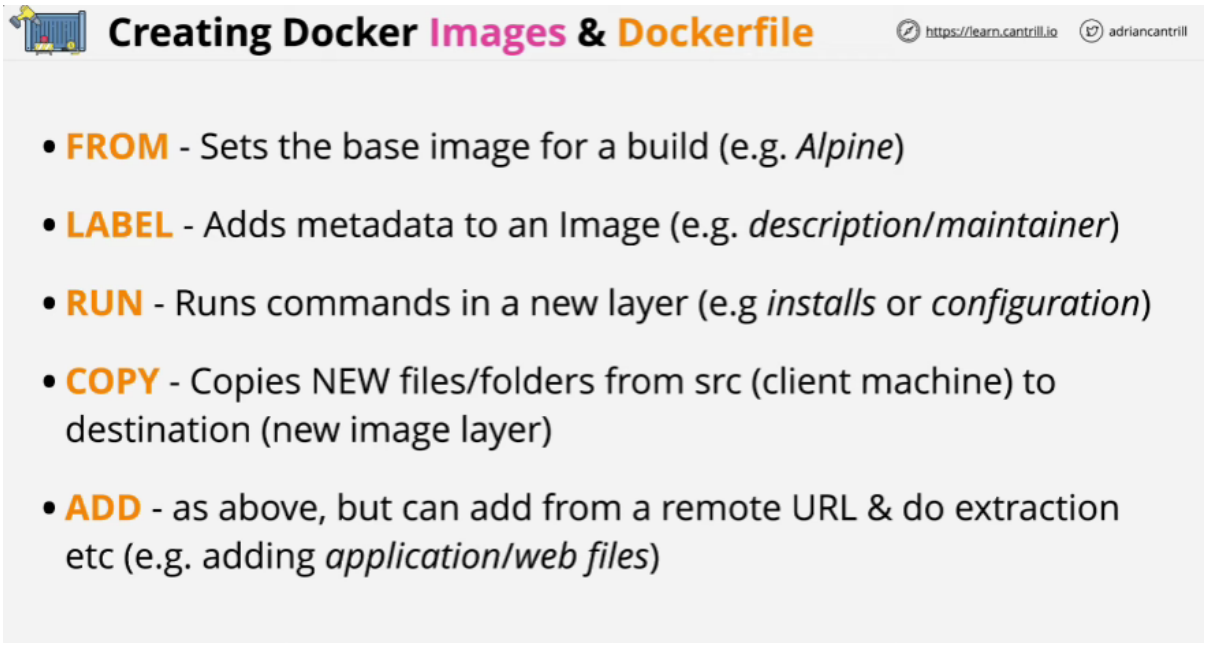


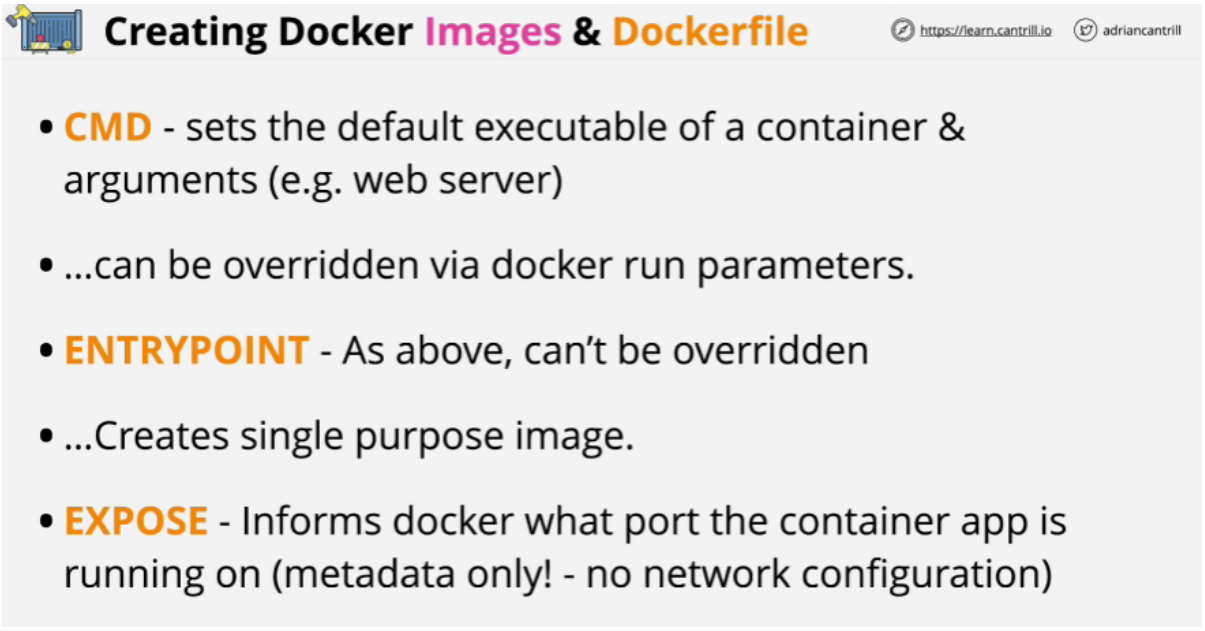


▼ [Demo] Working with existing docker images

▼ Dockerfile syntax







▼ [Demo] Build and run a simple containerized application

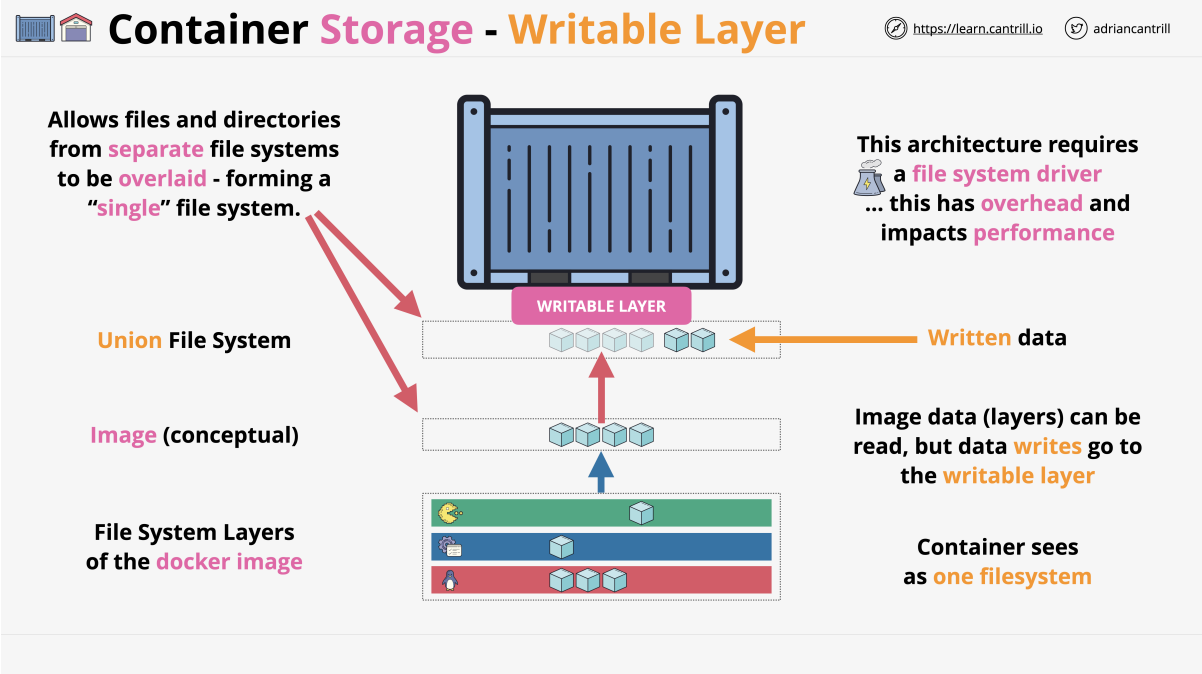
Instruction

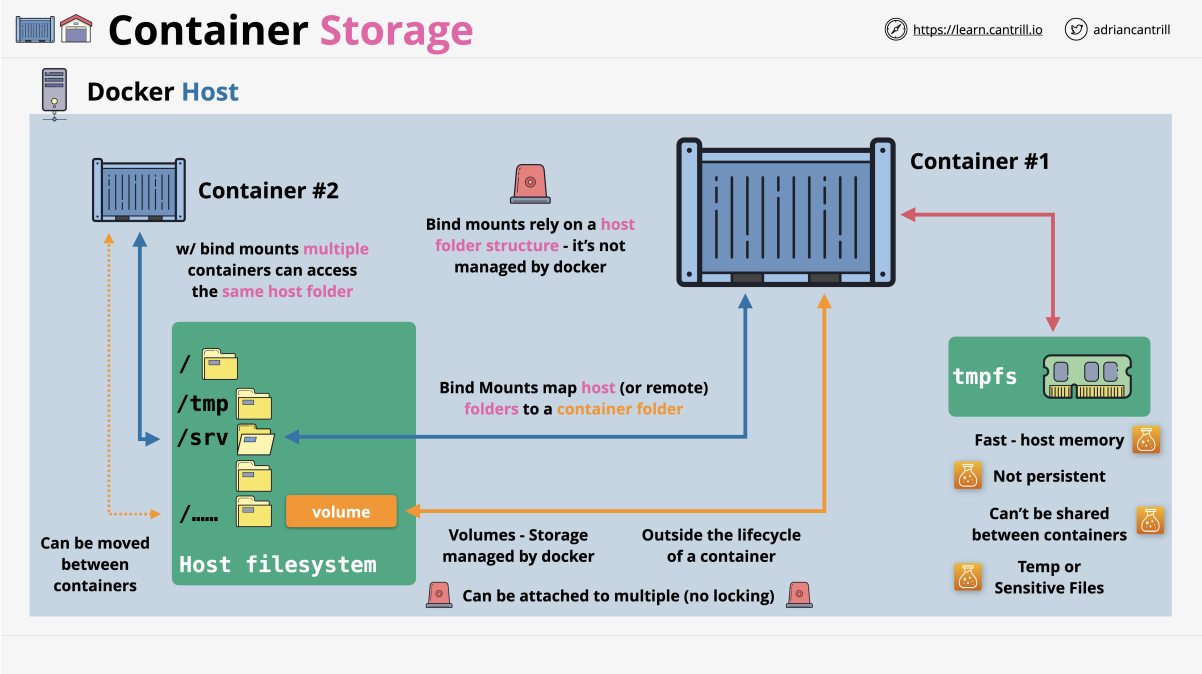
https://github.com/Fulim13/docker/blob/main/build-a-simple-containerized-application/build-a-simple-containerized-application.md

Files

https://github.com/Fulim13/docker/tree/main/build-a-simple-containerized-application

▼ Docker Storage - Writable Layer, Bind Mounts & Volumes

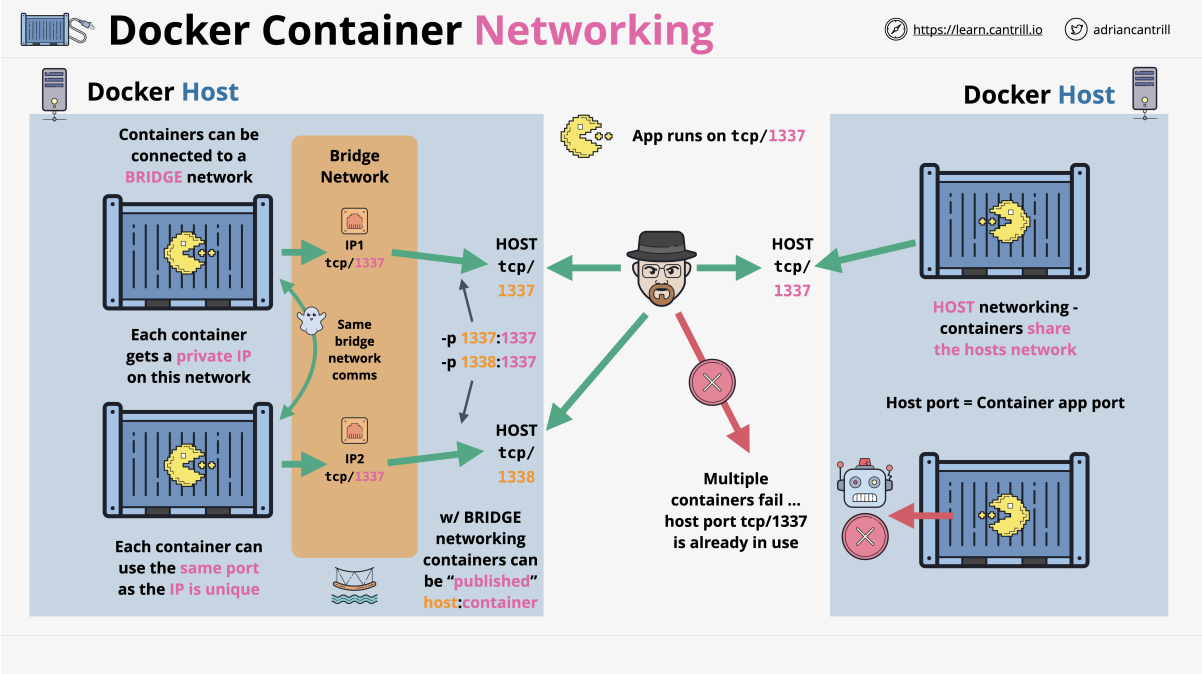




if there is not /srv folder, then we cannot bind the mount point

no file locking means - you have to careful about multiple processes/container access the storage at the same time

▼ Docker networking - modes and port exposure



▼ [Demo] Using Environment Variables

Environment variables are a way of passing configuration information into containers.

Instruction

https://github.com/Fulim13/docker/blob/main/docker-container-environment-variables/instructions.md

▼ [Demo] Docker Bind Mounts & Volumes

Bind mounts allow us to mount a file or directory on the host machine into a container. This has pros and cons but it can be useful if you need to see this files on the container host, or access a shared collection of files between container, or between containers and other compute services.

Volumes are the preferred way to adding storage to docker containers outside of the lifecycle of a container. They are managed entirely by docker and work flawlessly on windows container hosts.

Instruction

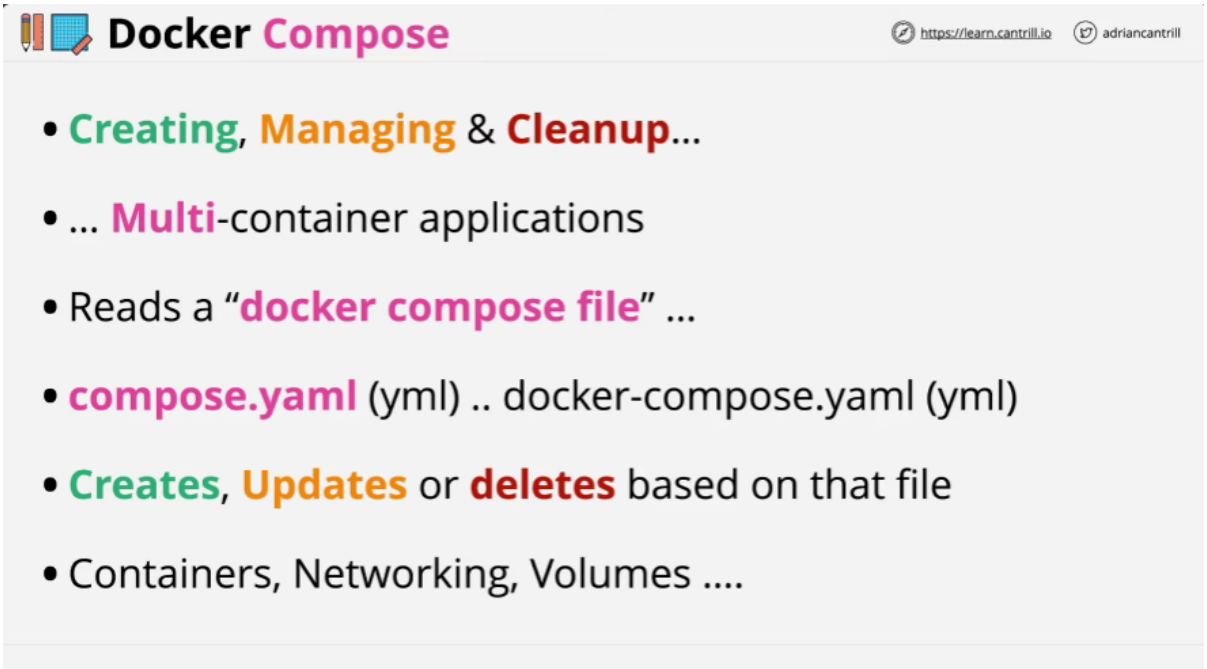
https://github.com/Fulim13/docker/blob/main/docker-container-volumes/instructions.md

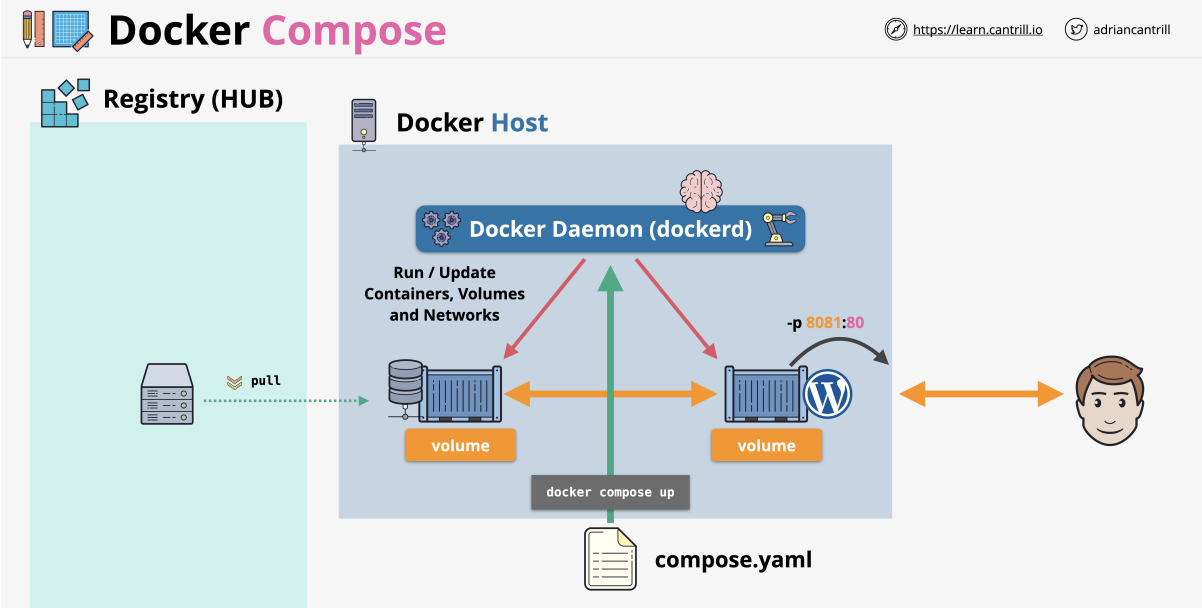
▼ Docker Compose

Docker Compose is software used for defining and running multi-container Docker applications. It can handle multiple containers simultaneously in the production, staging, development, testing, and CI environment. Therefore, use Docker Compose to manage the whole software development lifecycle (SDLC).

Docker Compose works by applying rules defined in a **docker compose** file.

The YAML file configures the application's services and includes rules specifying how you want them to run. With the file in place, you can start, stop, or rebuild all the services using a single command.





▼ [DEMO] Using Docker Compose with our application

Docker compose is a way to create and manage multi-container applications. It works using a YAML configuration file called `compose.yaml` or `docker-compose.yml`.

Instruction

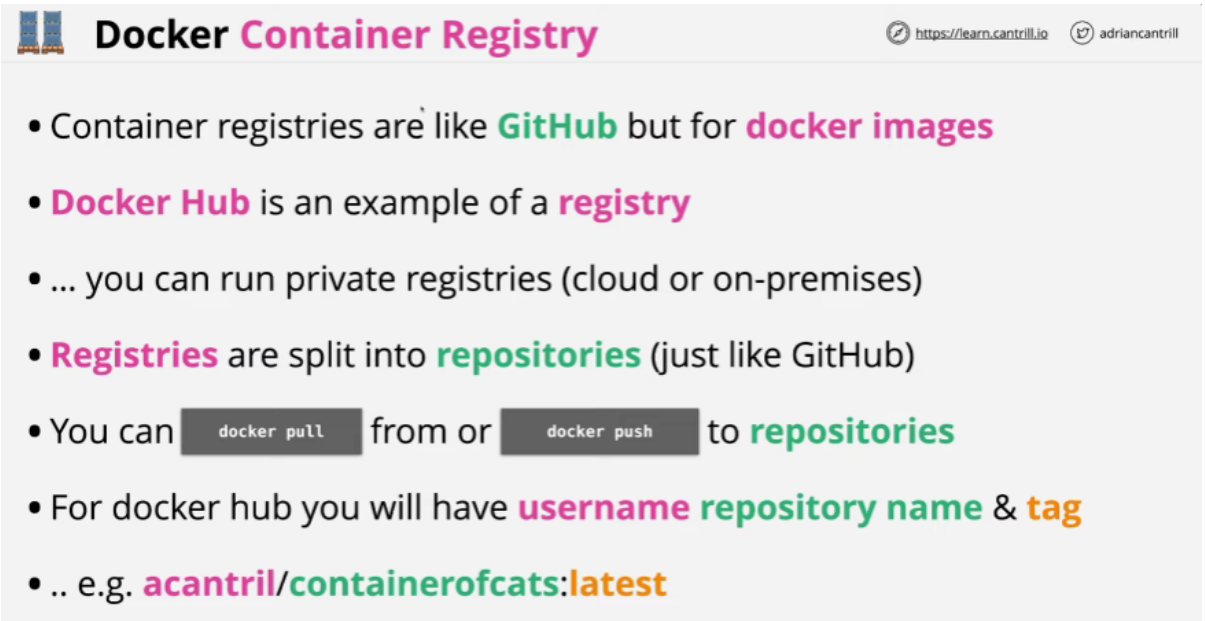
https://github.com/Fulim13/docker/blob/main/docker-compose/instructions.md

▼ Container Registry

A Docker registry is a storage and distribution system for named Docker images.

An image might have multiple different versions, identified by their tags.

A registry is organized into Docker repositories. Each repository holds all the versions of a specific image.



▼ [DEMO] Upload our application to docker hub

Instruction

https://github.com/Fulim13/docker/blob/main/docker-registry/instructions.md