

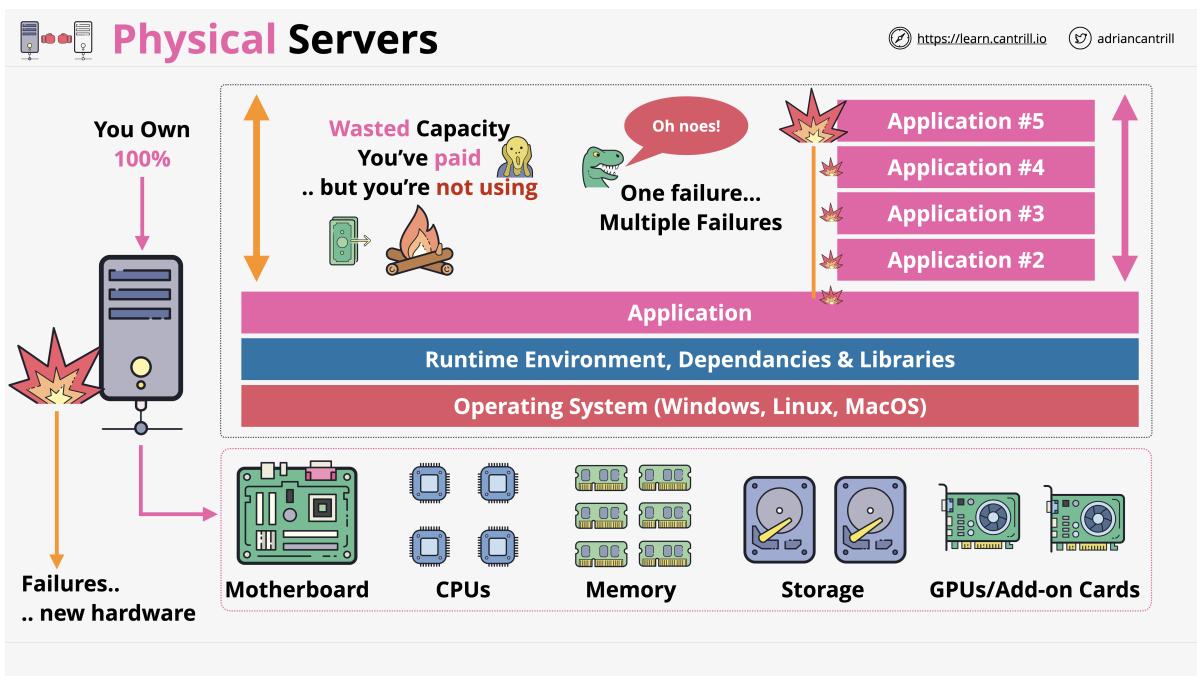


Docker Overview

▼ Repo

<https://github.com/Fulim13/docker>

▼ Physical Servers vs Virtual Machines

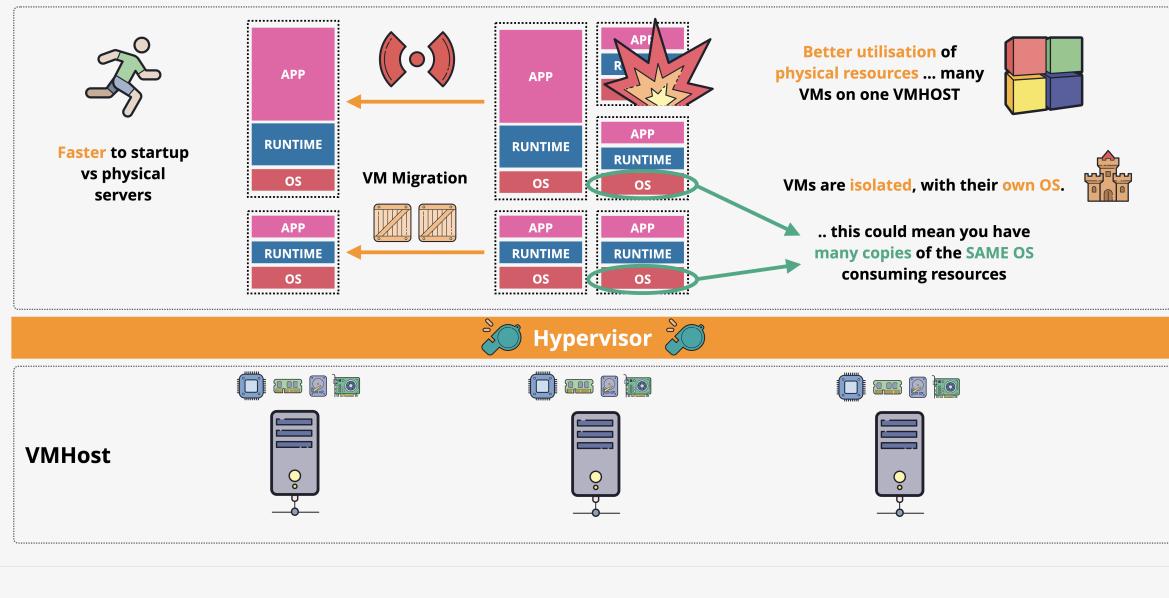




Virtual Servers/Machines (VMs)

<https://learn.cantrill.io>

@adriancantrill



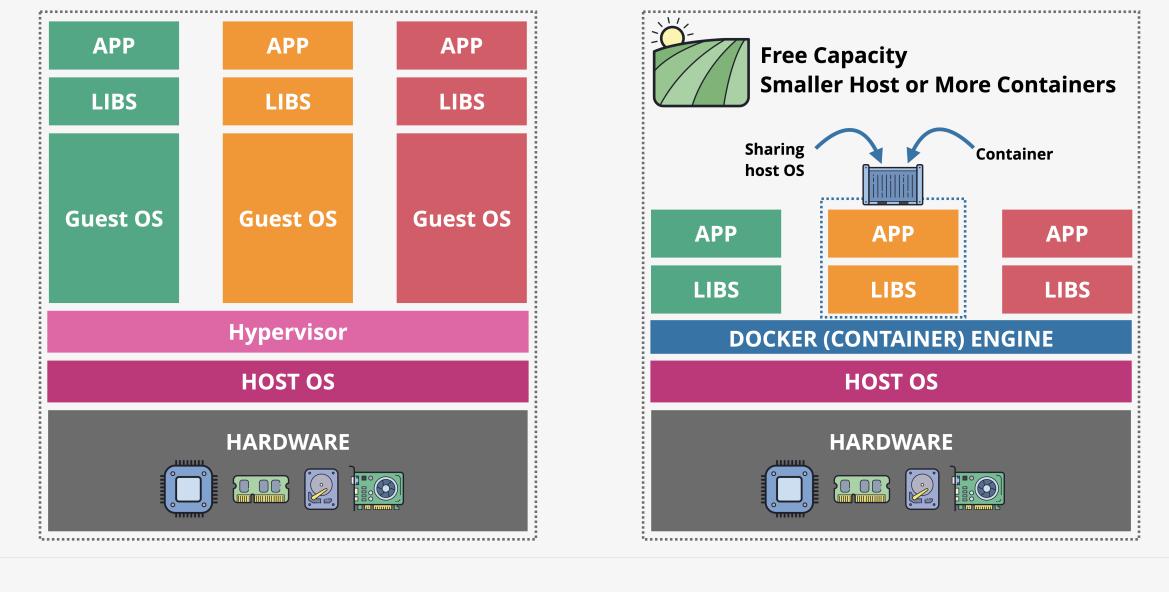
▼ Virtual Machines vs Container



Container Architecture

<https://learn.cantrill.io>

@adriancantrill





Container Architecture

<https://learn.cantrill.io>

@adriancantrill

- Containers run on a container host
- .. via Container (Docker) Engine
- Containers only run an APP & libraries / Runtime Env
- Share the container HOST OS (run as a process on it)
- Lightweight - Can be densely packed & started/restarted quickly
- Can be impacted by other containers (noisy neighbours)
- "It works on my computer" => "ok, let's ship your computer"

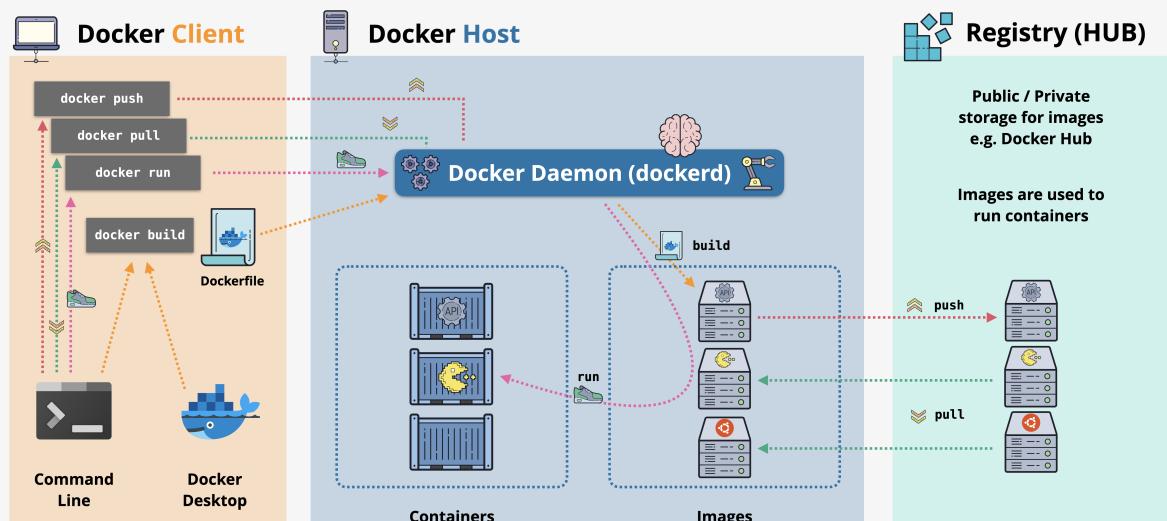
▼ Docker Architecture



Docker Architecture

<https://learn.cantrill.io>

@adriancantrill



▼ [Demo] Interacting with Docker Engine

```
# run local docker images (if not found, pull from docker hub registry)
```

```

docker run hello-world

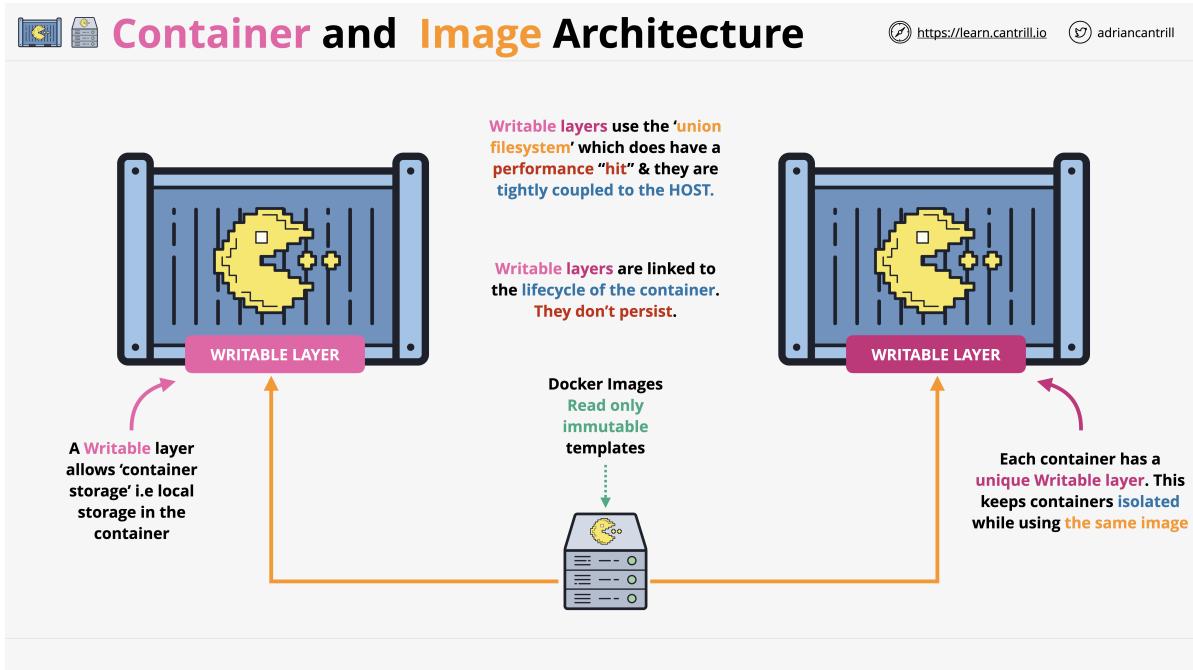
# Check running containers
docker ps

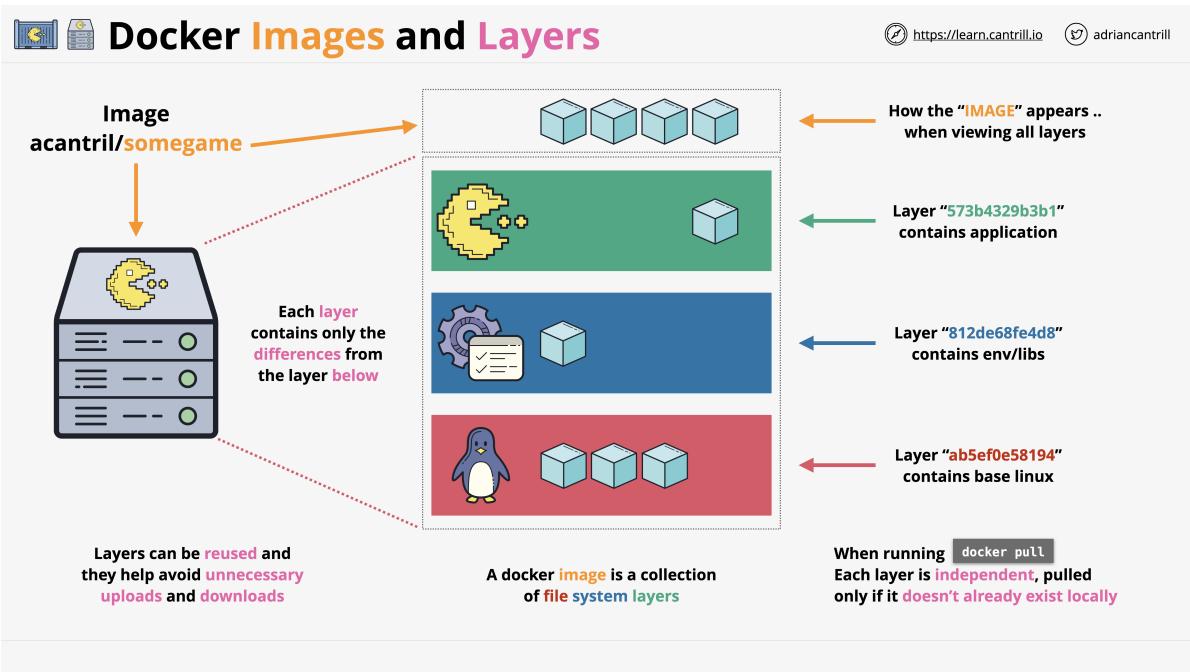
# Check all containers
docker ps -a

# Check all images
docker images

```

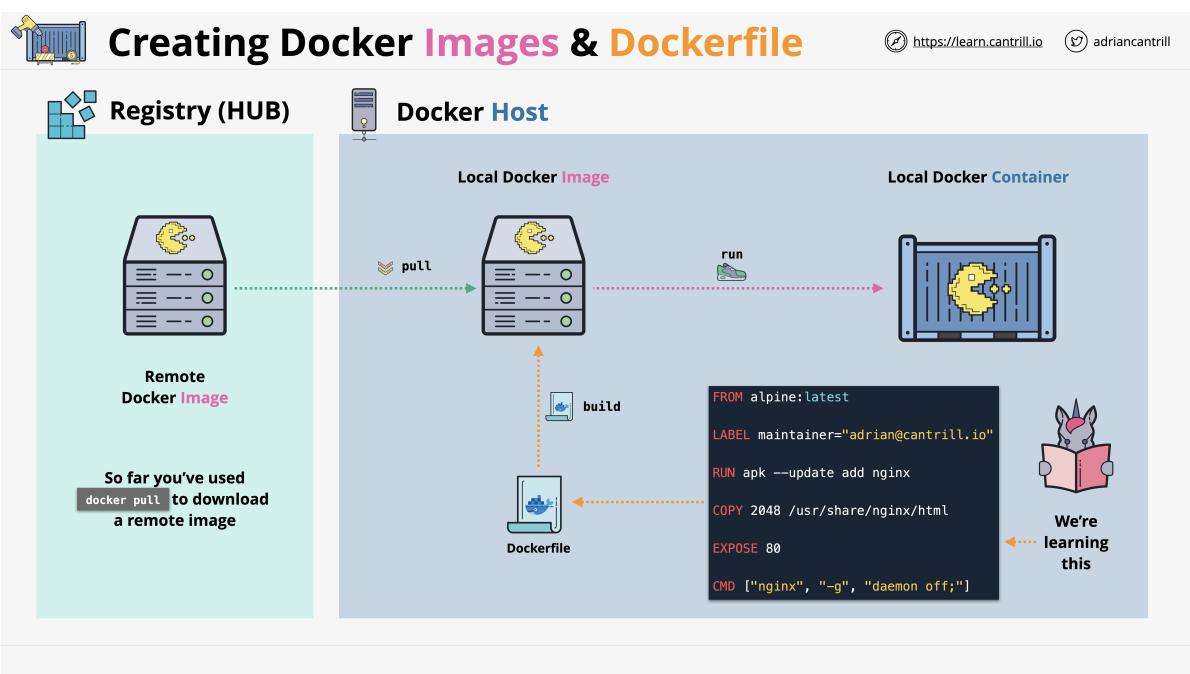
▼ Container and Image Architecture





▼ [Demo] Working with existing docker images

▼ Dockerfile syntax



Creating Docker Images & Dockerfile

<https://learn.cantrill.io>  adriancantrill

- **FROM** - Sets the base image for a build (e.g. *Alpine*)
- **LABEL** - Adds metadata to an Image (e.g. *description/maintainer*)
- **RUN** - Runs commands in a new layer (e.g *installs* or *configuration*)
- **COPY** - Copies NEW files/folders from src (client machine) to destination (new image layer)
- **ADD** - as above, but can add from a remote URL & do extraction etc (e.g. adding *application/web files*)

Creating Docker Images & Dockerfile

<https://learn.cantrill.io>  adriancantrill

- **CMD** - sets the default executable of a container & arguments (e.g. web server)
- ...can be overridden via docker run parameters.
- **ENTRYPOINT** - As above, can't be overridden
- ...Creates single purpose image.
- **EXPOSE** - Informs docker what port the container app is running on (metadata only! - no network configuration)

▼ [Demo] Build and run a simple containerized application

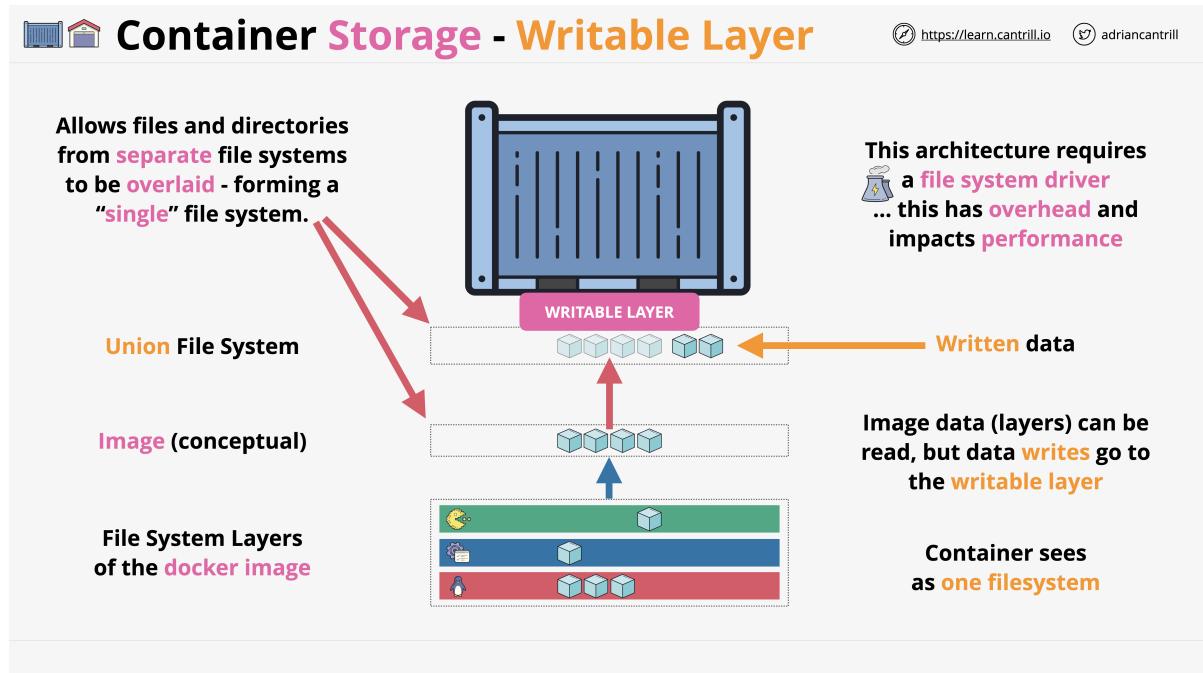
Instruction

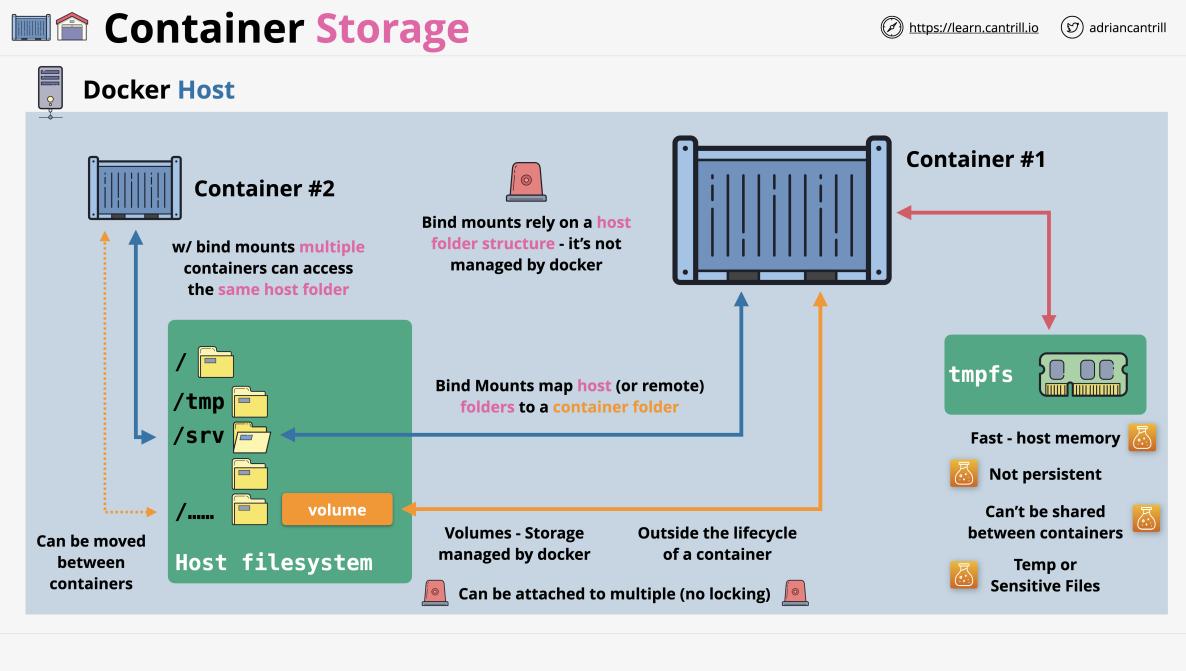
<https://github.com/Fulim13/docker/blob/main/build-a-simple-containerized-application/build-a-simple-containerized-application.md>

Files

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

▼ Docker Storage - Writable Layer, Bind Mounts & Volumes

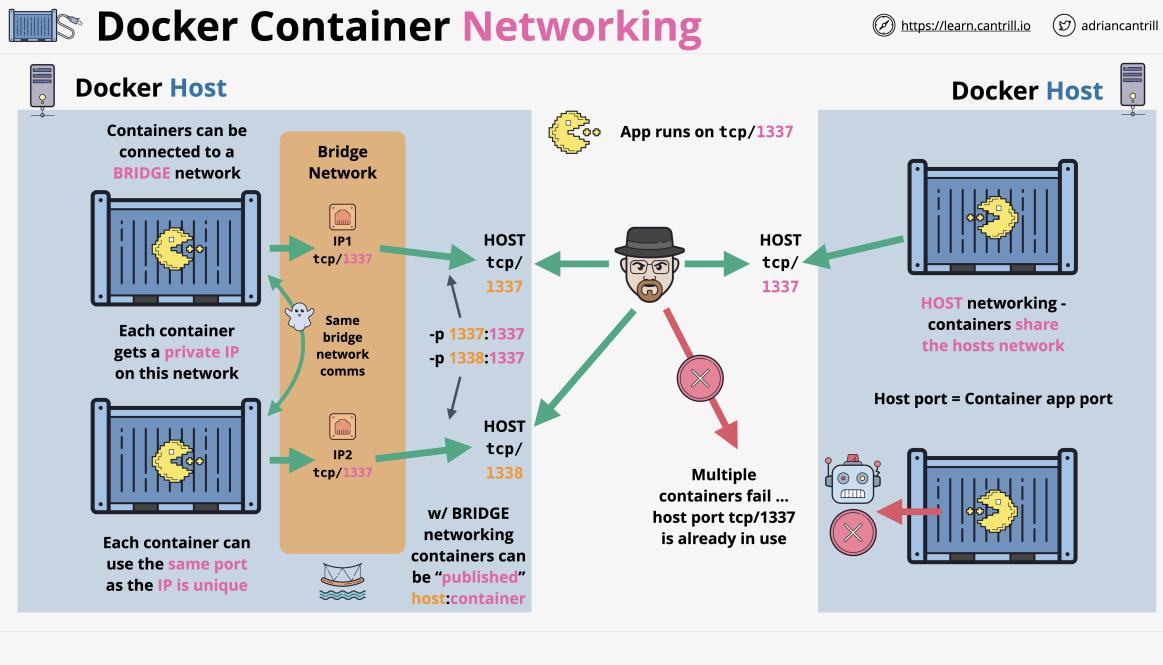




if there is not /srv folder, then we cannot bind the mount point

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

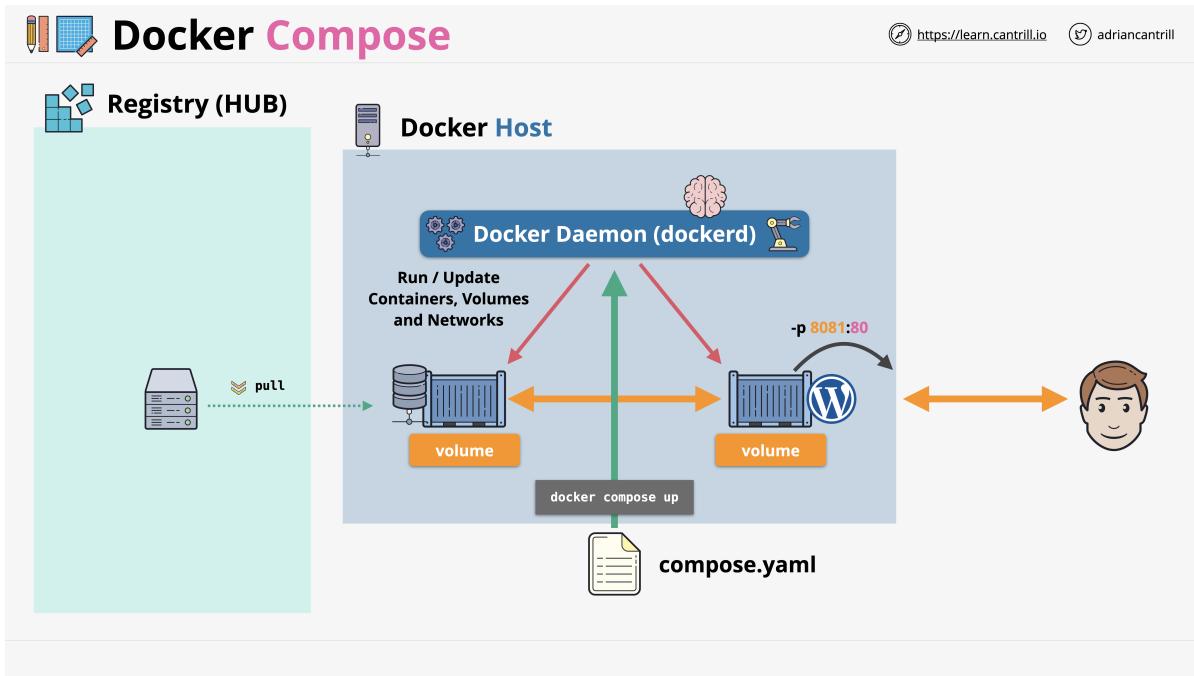
Docker Compose is software used for defining and running multi-container Docker applications. It can handle multiple containers simultaneously in the production, staging, development, testing, and CI environment. Therefore, use Docker Compose to manage the whole software development lifecycle (SDLC).

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.

The screenshot shows a slide with a light gray header bar. On the left is a small icon of a notepad and pencil. To its right is the title "Docker Compose". In the top right corner, there is a link "https://learn.cantrill.io" and a small profile picture with the name "adriancantrill". The main content area contains a bulleted list:

- **Creating, Managing & Cleanup...**
- ... **Multi**-container applications
- Reads a “**docker compose file**” ...
- **compose.yaml** (yml) .. docker-compose.yaml (yml)
- **Creates, Updates** or **deletes** based on that file
- Containers, Networking, Volumes



▼ [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.



Docker Container Registry

<https://learn.cantrill.io>

adriancantrill

- Container registries are like **GitHub** but for **docker images**
- **Docker Hub** is an example of a **registry**
- ... you can run private registries (cloud or on-premises)
- **Registries** are split into **repositories** (just like GitHub)
- You can `docker pull` from or `docker push` to **repositories**
- For docker hub you will have **username repository name & tag**
- .. e.g. **a-cantril/container-of-cats:latest**

▼ [DEMO] Upload our application to docker hub

Instruction

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