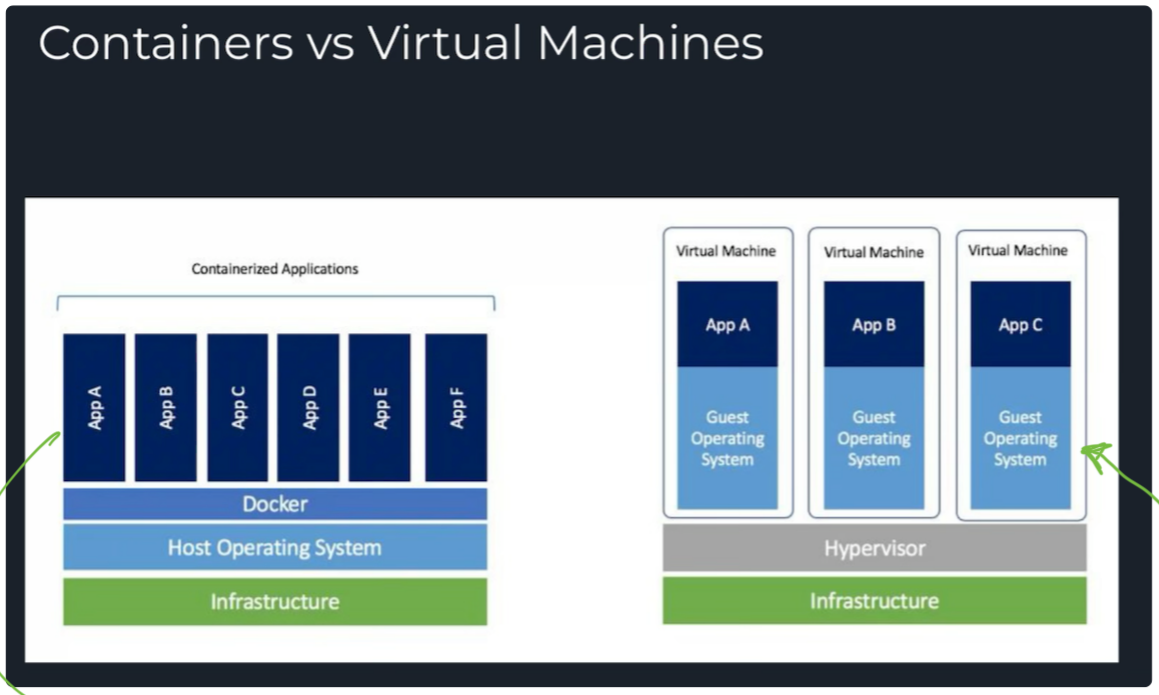
Lecture 5 - 11th Oct 2025

Containers

→ solves the issue of "Works on my machine!".

Containers vs VMs



Complete Isolation

- no guest OS required
- "Shifting left" enabled

This is Redundant

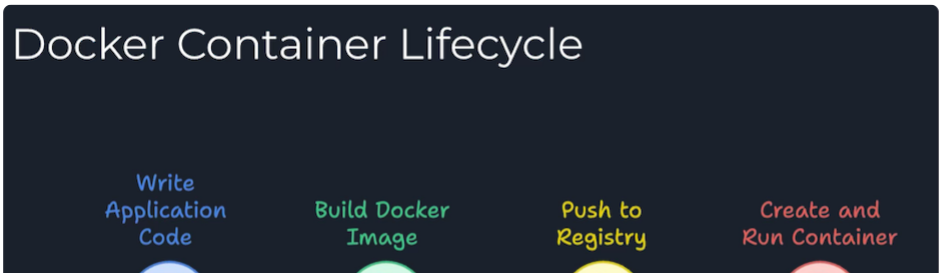
for developers.

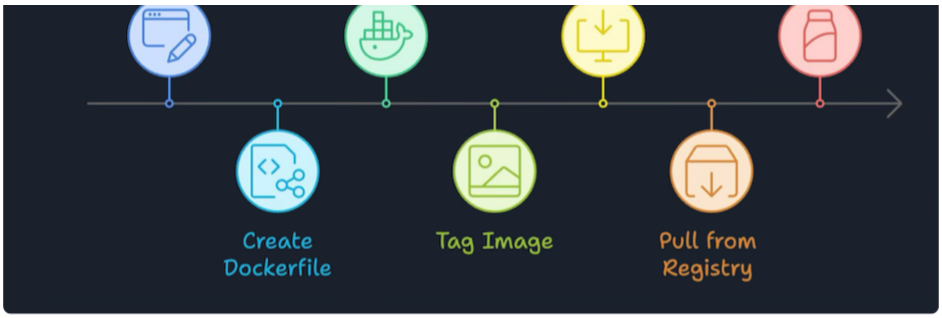
Takes up memory, storage, setup time, booting time.

Feature	Containers (Food Trucks)	Virtual Machines (Restaurants)
Infrastructure	Share the street (host OS)	Own building and utilities
Resource Usage	Efficient use of limited space	More spacious, potentially underutilize
Startup Time	Quick setup and ready to serve (seconds)	Takes time to open and prepare (minutes)
Isolation	Separate trucks, shared environment	Complete isolation from other establishments
Customization	Limited to internal equipment	Can modify entire structure
Management	Focuses on food prep and service	Responsible for entire property management

Docker.

Docker Component	Food Truck Analogy
Docker Engine	Food Truck Kitchen
Docker Images	Recipe book
Docker Containers	Operational truck
Dockerfile	Recipe instructions
Docker Hub	Recipe library





- Docker registry || AWS ECR

Dockerfile

FROM

WORKDIR

COPY

RUN

EXPOSE ← should be same with source code

ENV

CMD

Lets build the app now-

1. `docker build -t <name>:<version> .`

Docker uses "Layering" which basically means caching "requirements" for future builds.

2. `docker run -p 5000:5001 <image_name>.<version>`

machine port

<port> given in dockerfile (& source code) → WHERE container is hosted.

This is called Port forwarding.

- 2 modes for running
  - Foreground mode
  - Detached mode → add "-d" in run command

3. docker push

Docker Volumes

- attached to docker containers.
- container uses it.
- multiple containers can use it
- EXTERAL to the container

Docker Networks.

→ networking feature to enable communication between containers.

Docker Compose

→ Can run multiple containers together

→ including: networks, volumes and configurations.

compose.yaml

docker compose up

Docker networks are low-level connectivity

Docker compose is high-level Orchestration.

Docker namespaces.

Linux kernel feature which makes each container think that its "ISOLATED".

Docker Control Groups.

Linux kernel feature which decides what & how much resources can a container use.

You already learned that **namespaces isolate what a container can see —**
now we'll explore **control groups (cgroups)**, which isolate **what a container can use**.