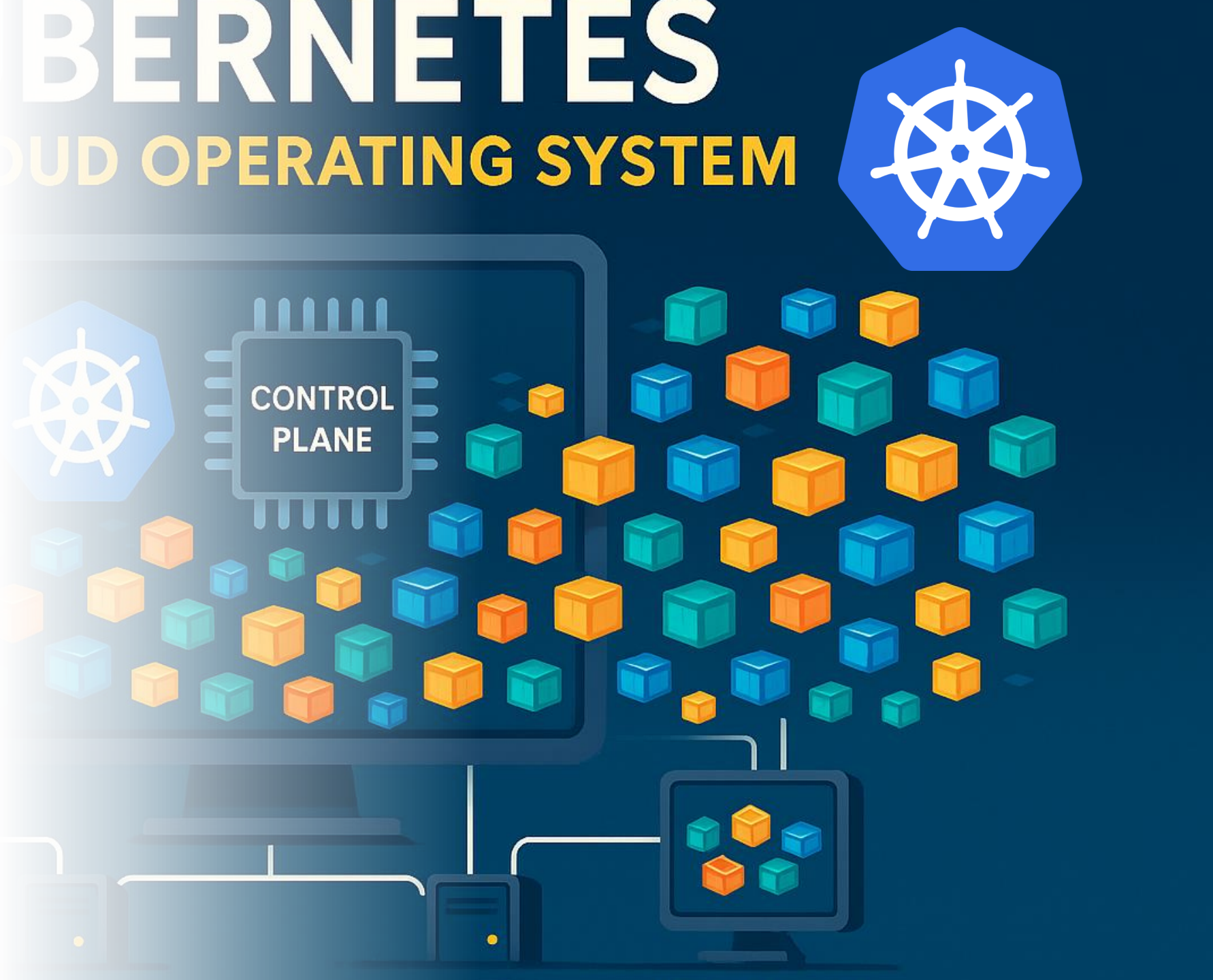
CS-UY 3913

Container Operating Systems



Instructor: Magdy Salem

Agenda

- Who am I?
- What to expect
- Tools
- Container OS / Kubernetes
- Docker Overview & Features
- Docker CLI

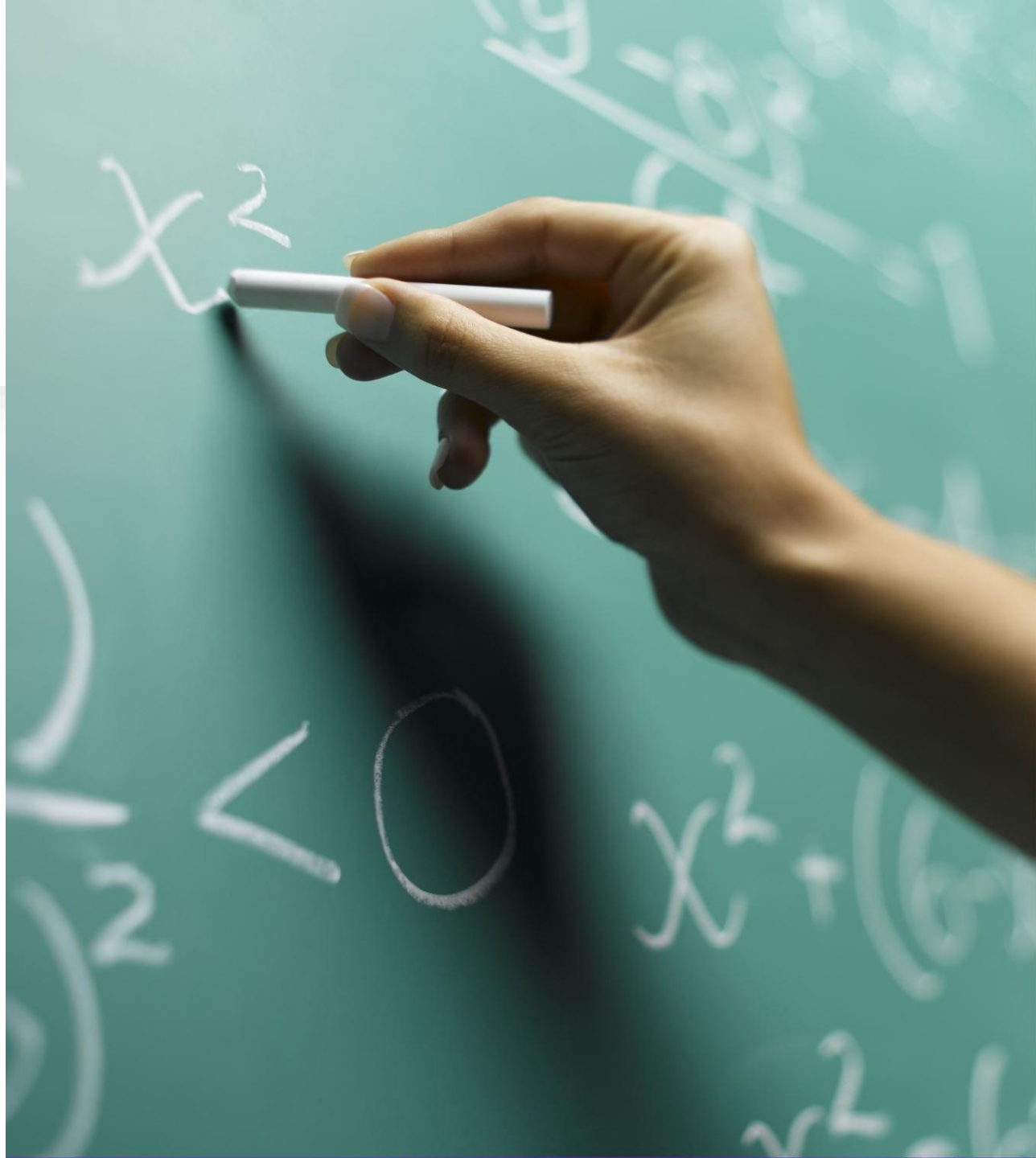
Who am I?

Magdy Salem

- Sr Cloud Solution Architect
- 25 yrs experience in enterprise and scales solutions
- Open Source Tech Enthusiastic
- Open Sources Contributor
- Email: magdy.salem@nyu.edu
- Github: msalemcode

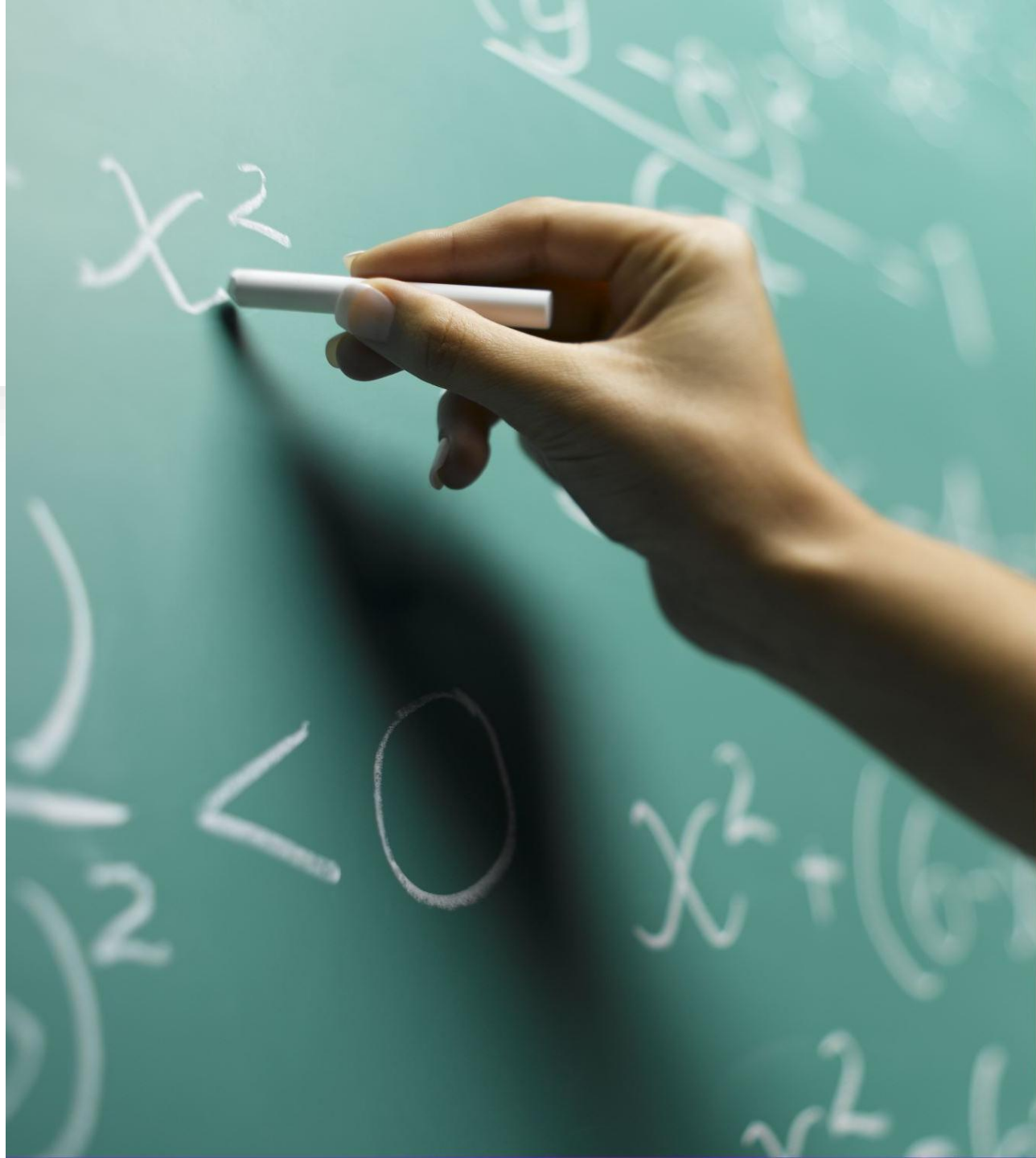
What to expect from the course?

- Hand-On Class
- Not textbooks
- Learn about OSS: Docker, Kubernetes, Helm, CICD.
- Kubernetes Operator skills



Tools

- VS Code
- Docker windows/Mac/Linux
- MiniKube /K3D
- Github account



HOMEWORK & LAB

Recommended online resources:

- https://kubernetes.io/docs/
- https://docs.docker.com/
- https://helm.sh/docs

Grading Policy

- 40% – WEEKLY LABS & ASSIGNMENTS
- 40% – FINAL EXAM
- 20% – PARTICIPATION & ENGAGEMENT

Kubernetes: The Cloud OS

- Abstracts heterogeneous infrastructure
- Declarative, API-driven control plane
- Self-healing
- Pluggable architecture

Why We Needed a Cloud OS

- Scalability & Efficiency
- Portability
- Resilience
- Extensibility

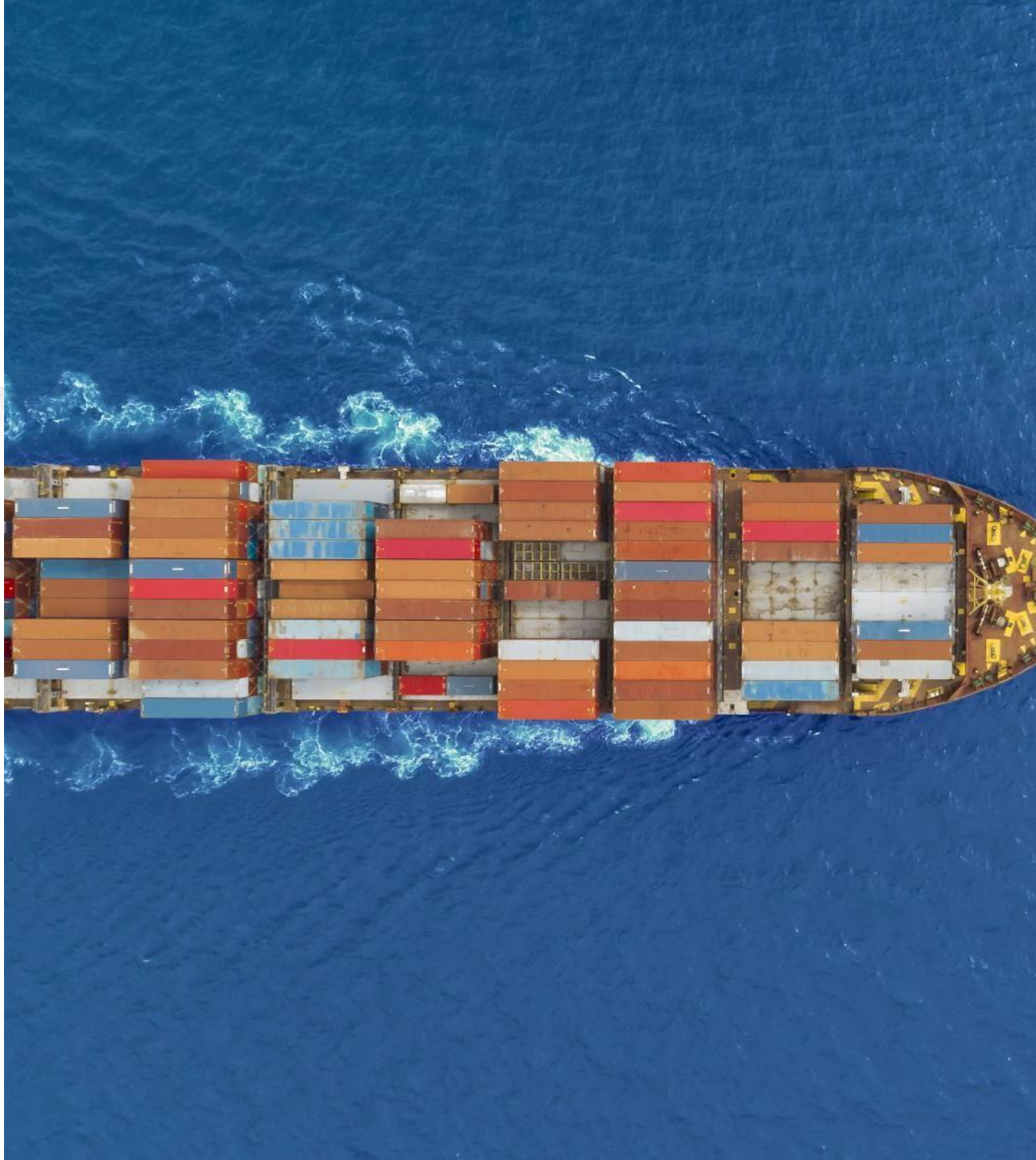
A Brief Kubernetes Timeline

- Origins at Google (2003–2014)
- Project Launch (June 2014)
- 1.0 Release (July 2015–March 2016)
- Rapid Ecosystem Growth (2016–2018)
- Maturing Platform (2019–2021)

Why Container first?

Bottom Line:
Kubernetes orchestrates containers, not raw binaries

mastering containers first means you'll immediately understand what Kubernetes is doing under the hood.

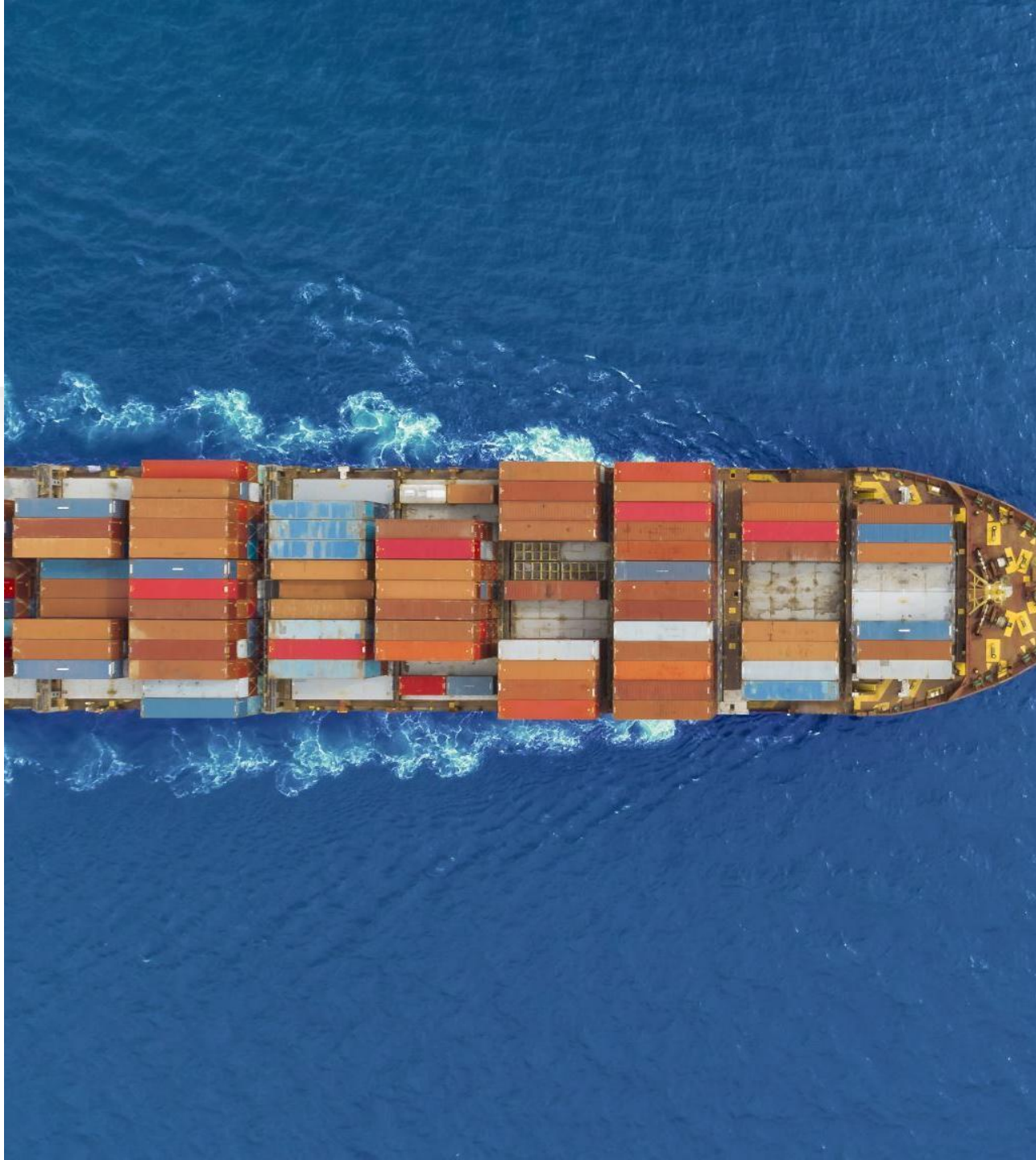


A Brief Container Timeline

- 1979 chroot
- 2000s: chroot, Solaris Zones, LXC
- 2013: Docker simplifies containers
- 2015: OCI standardizes runtimes
- 2015+: Swarm, Mesos, Kubernetes orchestrators

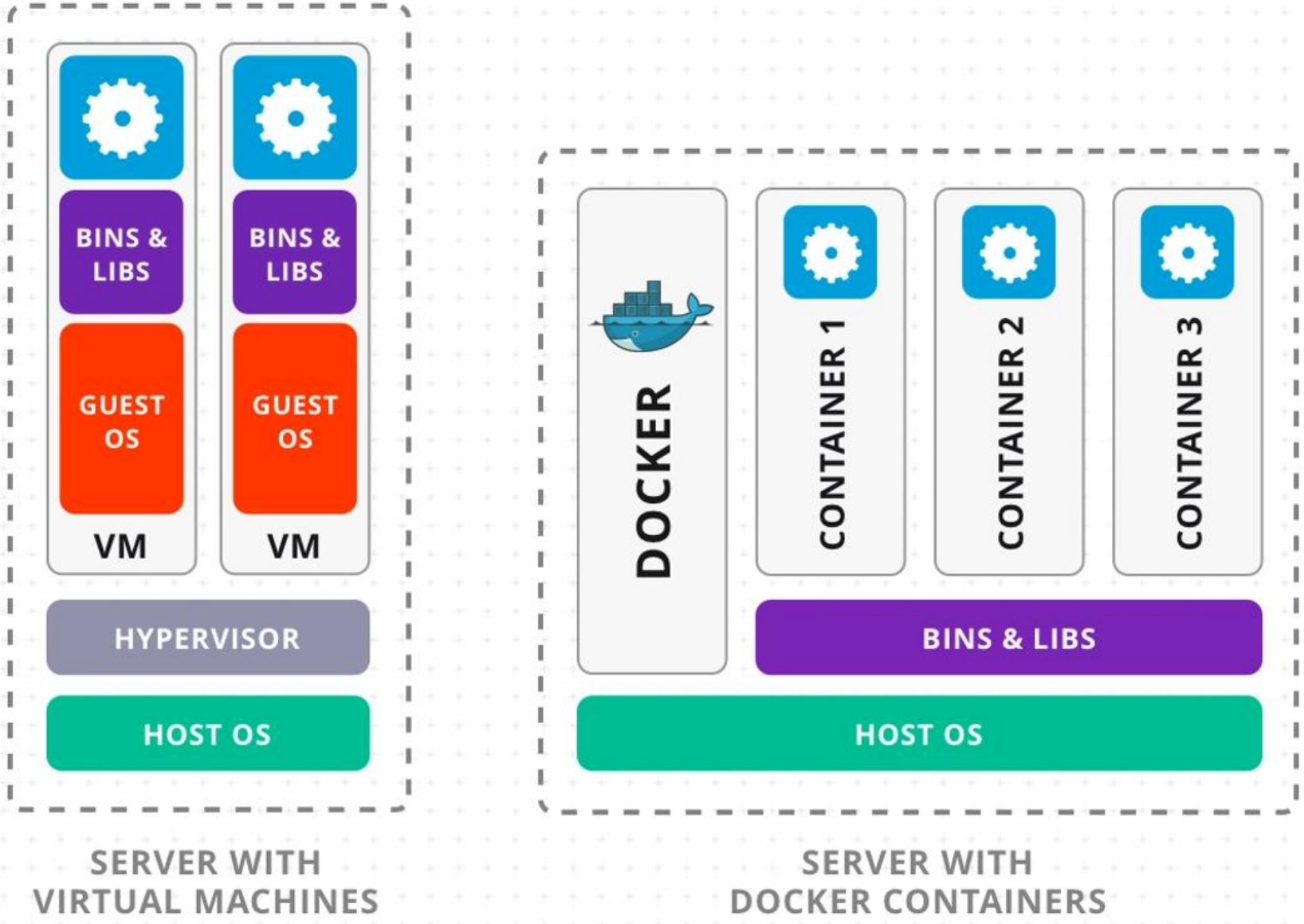
Enter Container world

- Process-level isolation (namespaces, cgroups)
- Ship once, run anywhere consistently
- Faster & lighter than VMs



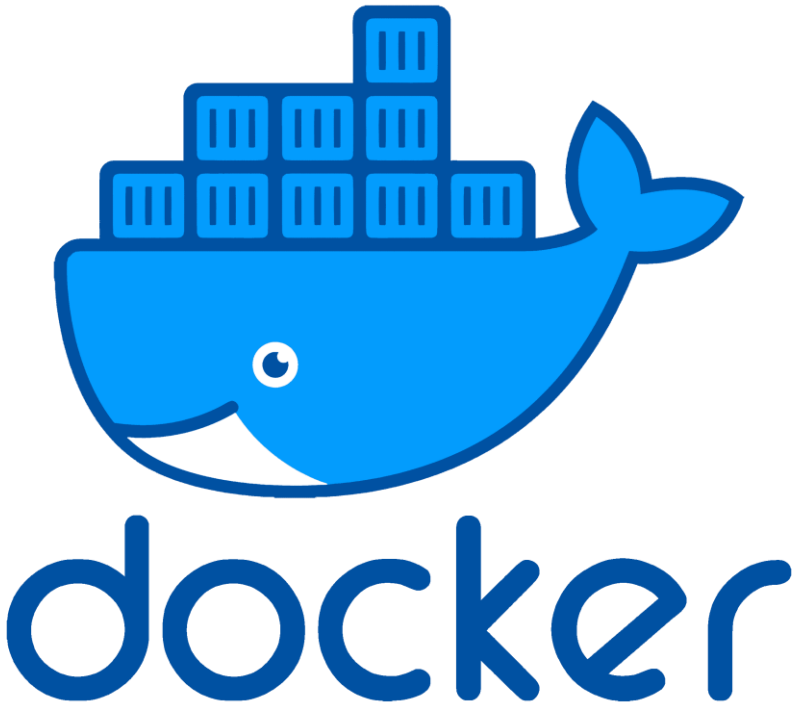
Docker vs VM

- Isolation
- Portability
- Density
- Efficiency
- Security



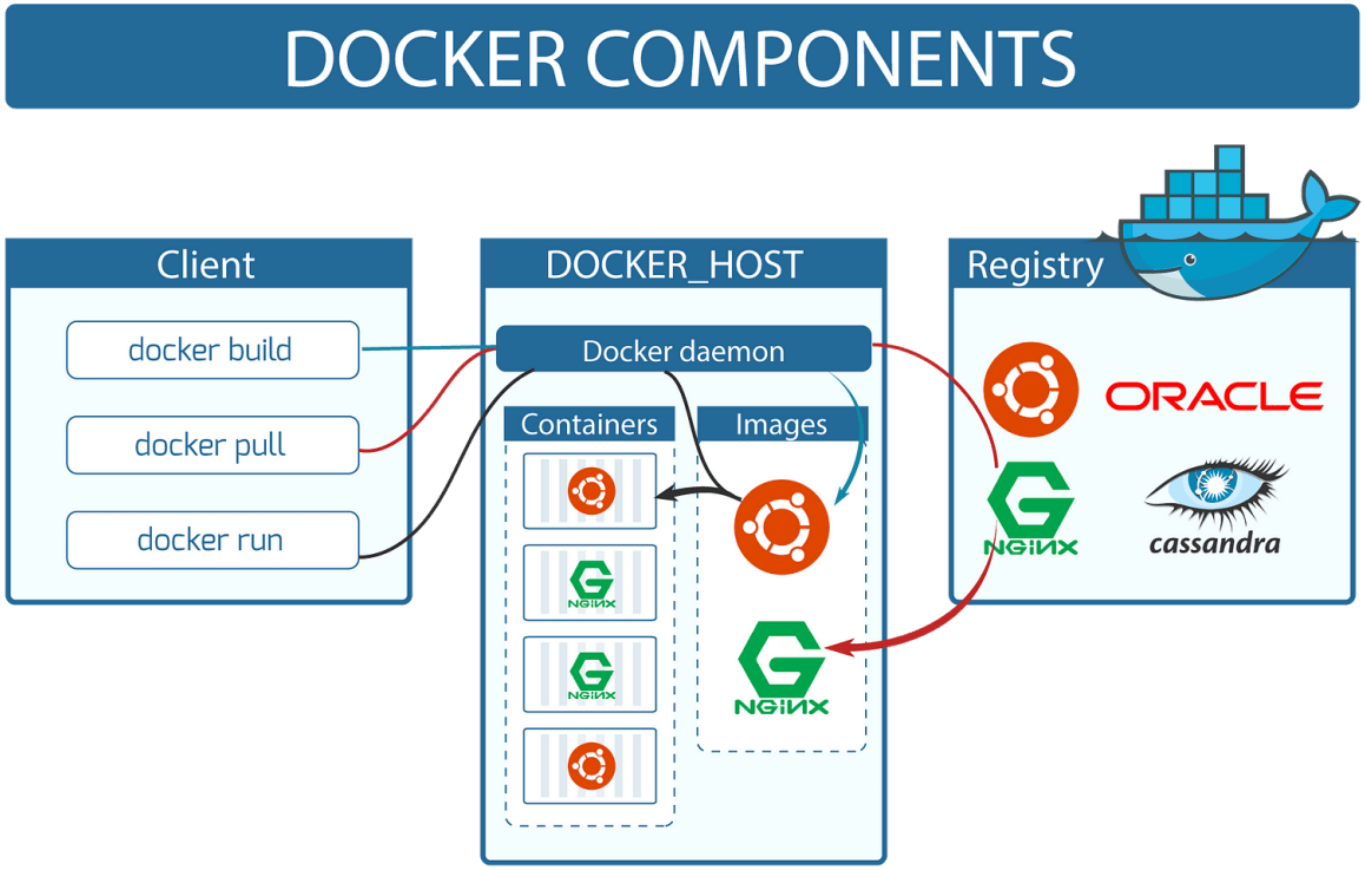
Docker Features

- Layered image architecture
- Dockerfile & declarative builds
- Image registry integration
- Portable “build once, run anywhere”
- Networking modes (bridge, host, overlay)
- Data persistence with volumes



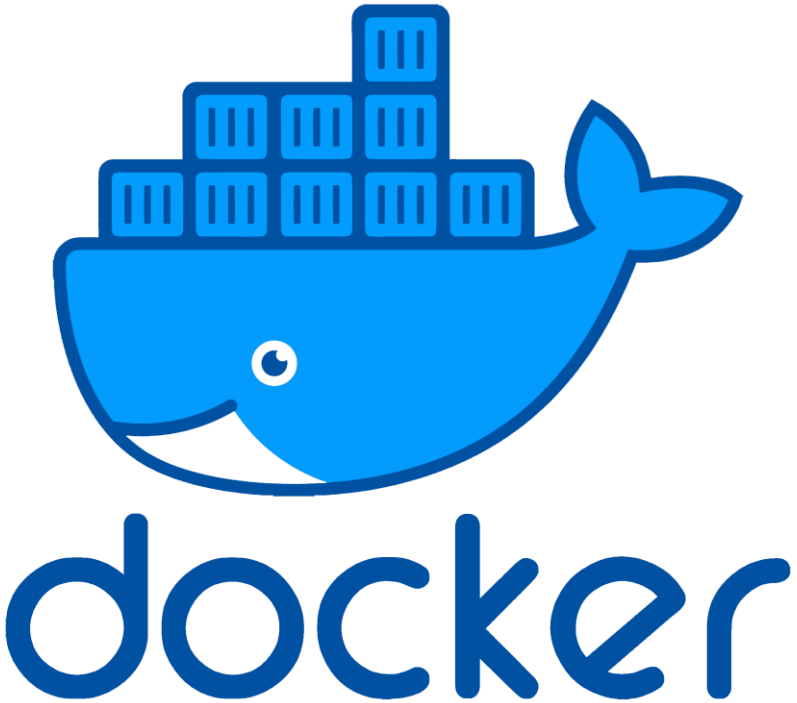
Docker Architecture

- Registry
- Docker Host
- Client



Docker Command

- docker pull
- docker build
- docker run
- docker ps
- docker logs
- docker images



Homework