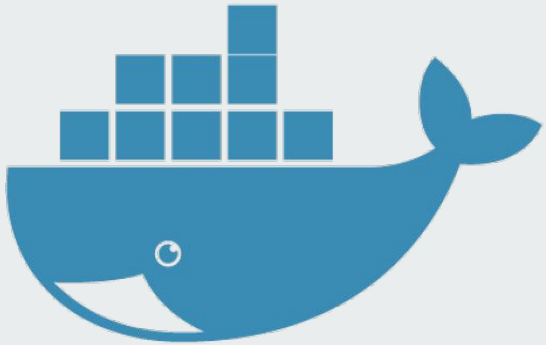
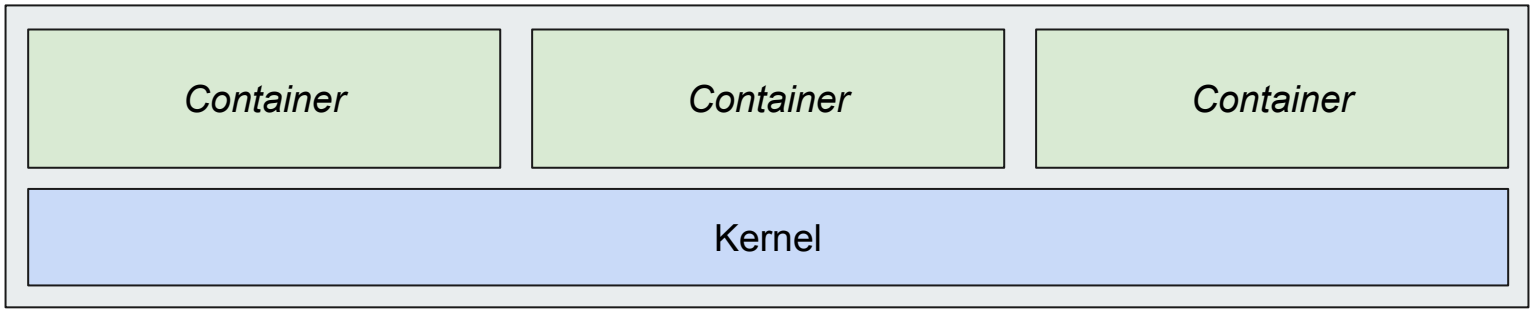
Big Picture Docker
What, Why, Who, & Features



What is Docker?

- Docker is a platform that lets you package, develop, and run applications in containers.
- A container is a virtual environment on top of the OS kernel to capture all of its software - libraries, dependencies, etc.



Why does Docker matter?

- A more lightweight approach than virtual machines to isolated coding and project environments.
- Portability to the major architectures and operating systems.
- Helps achieve continuous integration and deployment for development operations.

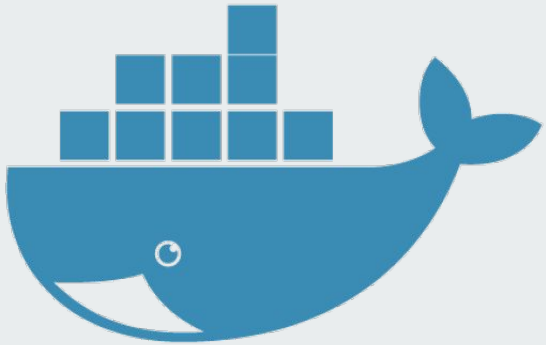
Who Does Docker affect most?

- Developers and software engineers.
- Operators and devops engineers.
- Startups and enterprises.

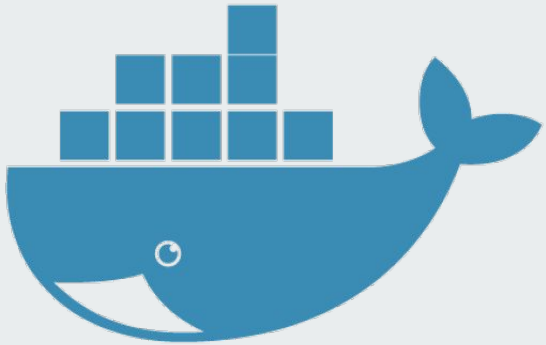
Main features

- Create containers and images.
- Docker-compose for multicontainer applications.
- Docker swarm to utilize multiple machines running Docker.

Big Picture Docker
What, Why, Who, & Features



Docker Images



Docker Images

- are “ready-only templates with instructions for creating a Docker container.”
- define the container code, libraries, environment variables, configuration files, and more.

Dockerfile

- Outlines instructions for how an image will create a container.

Image to Container Relationship

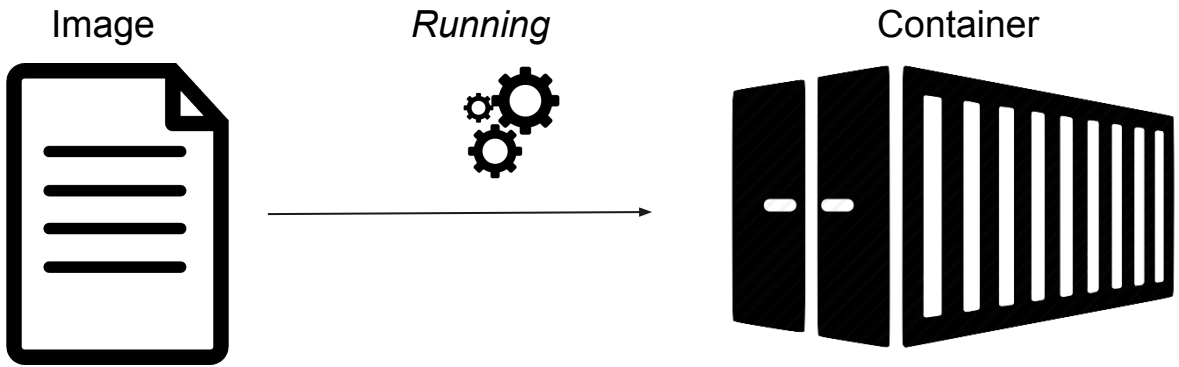
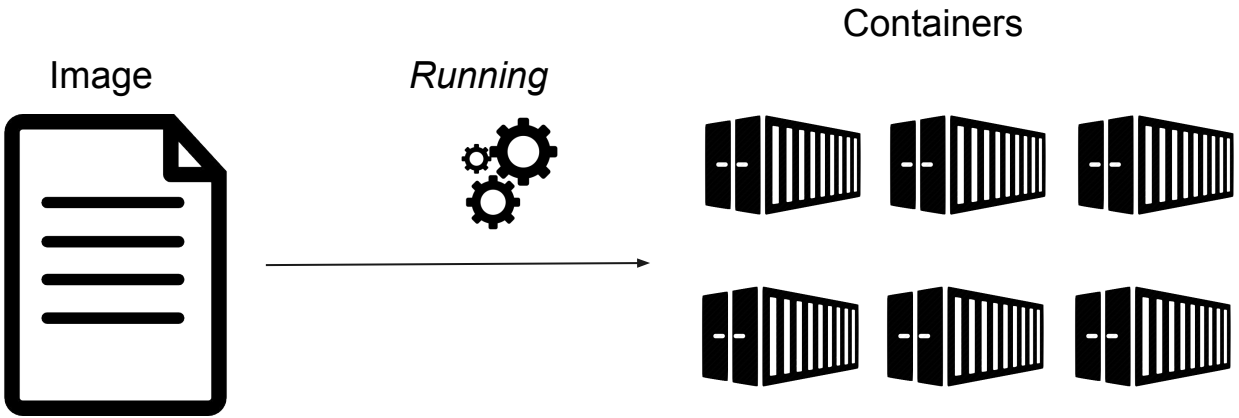


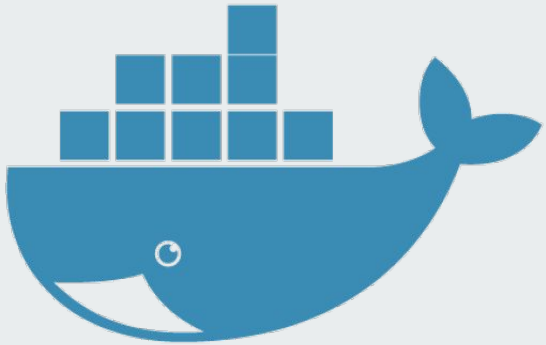
Image to Many Containers



Dockerhub

- Collects images from Docker users and hosts them as online repositories.

Docker Containers and the Docker Engine



What is a Docker Container?

A Docker container is a loosely isolated environment running within a host machine's kernel that allows us to run application-specific code.

The Kernel

- The kernel is the software at the core of an operating system, with complete control.
- The CPU is the core circuitry which executes program instructions.
- Docker runs on top of original machine's kernel - making it the *host machine*.

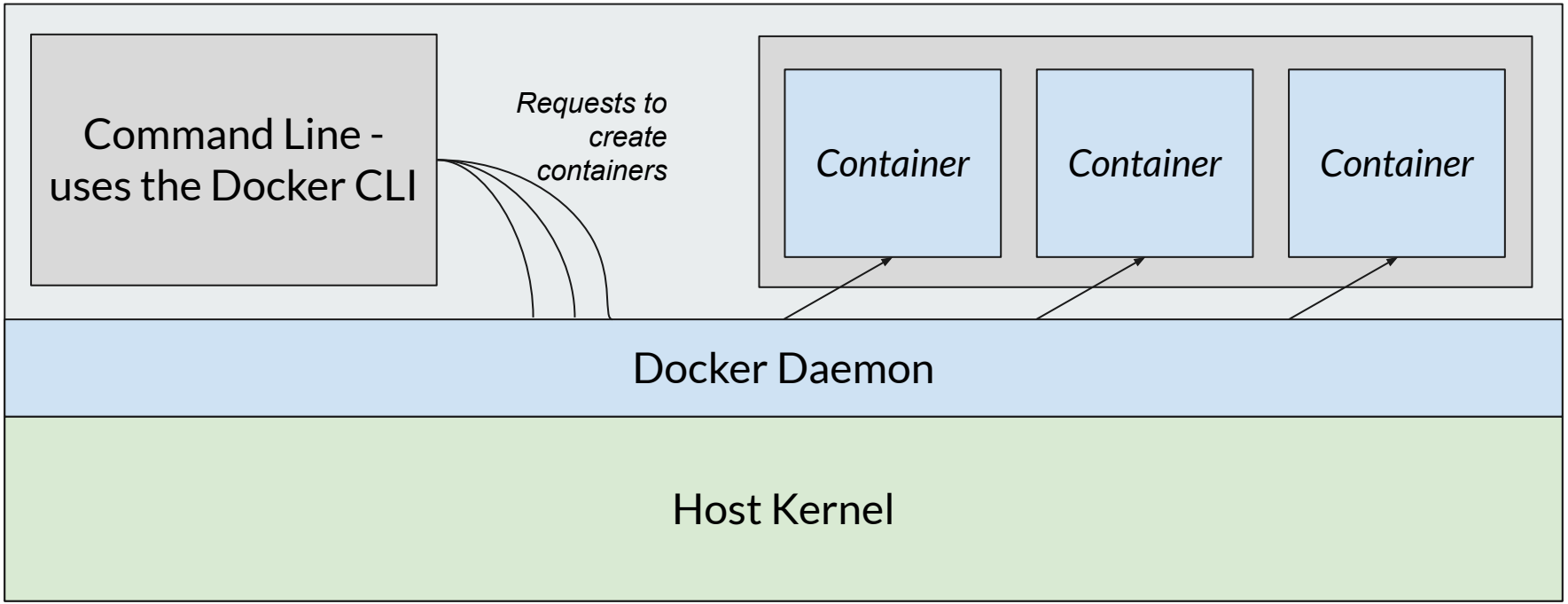
The Docker Engine

- Consists of the Docker server, an API, and command line interface.

- The server is also called the Docker daemon.
  - daemon - background processes on an operating system.

- Docker CLI is a command line interface, providing commands that allow you to interact with the Docker daemon

The Docker Engine on an Operating System



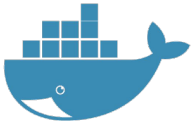
Docker Container Environments

- Independent file systems and local networks.

- Run application-specific code

What is a Docker Container?

A Docker container is a loosely isolated environment running within a host machine's kernel that allows us to run application-specific code.



Containers vs. Virtual Machines

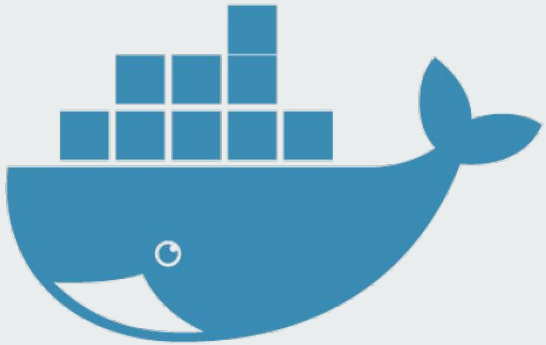
- **Both** are solutions to having isolated environments.
- Virtual machines abstract an entire operating system.
- Containers are smaller, isolated file systems and networks.
    - The ability to share common files, sets containers apart

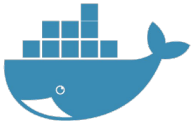
Containers vs. Virtual Machines

Run a 2GB image of software

Virtual Machines	Containers
• Require 2GB to run the image	• Use 2GB to run the image.
• 5VMs x 2GB = 10GB for 5 containers	• 2G + (0.1-0.5G per container) for 5+ containers

Docker Images and the Dockerfile





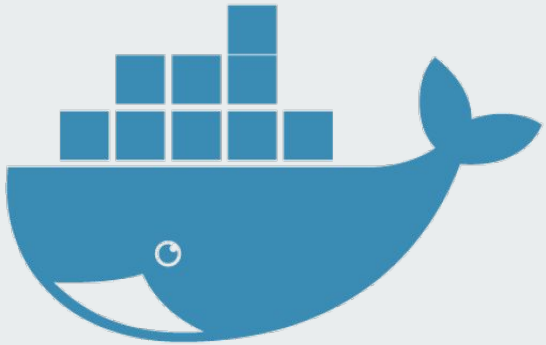
Docker Images

- A “ready-only template with instructions for creating a Docker container.”

Dockerfile

- A text document with various commands that assemble to build a container.
- $ docker create image... → FROM
- $ docker create image command → CMD
- COPY, RUN, and more...

Docker Volumes and Mounts



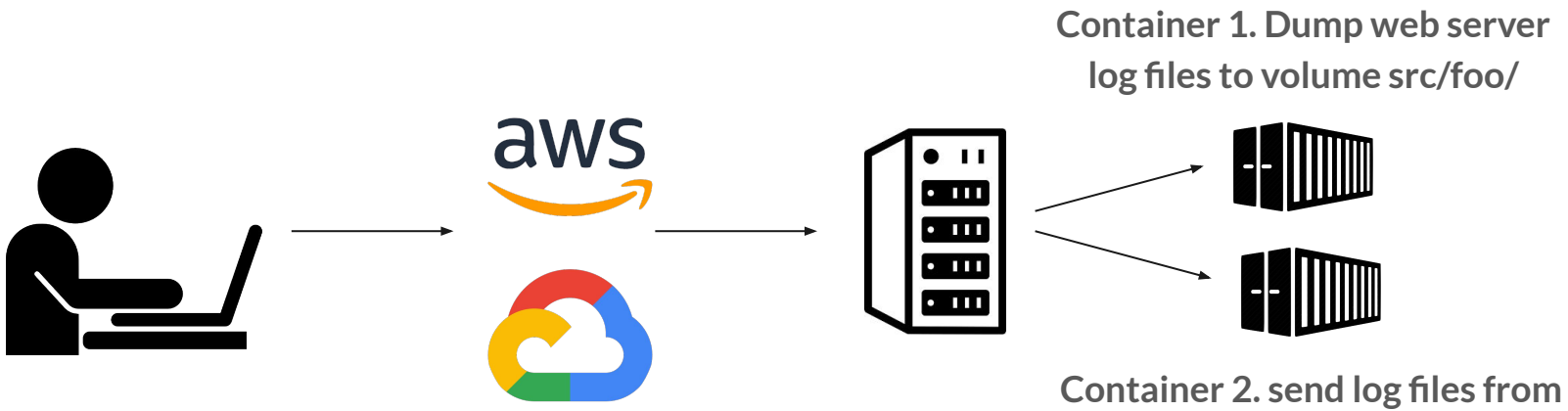
How do we share data across containers and with the host machine?

Volume mounts and Bind Mounts

Volume mounts

- Share data across containers.
- Data storage contained in the docker engine. Not the host filesystem, but within the virtualized docker engine file system.
- Container file sharing on a remote machine.

Volume on remote machine example



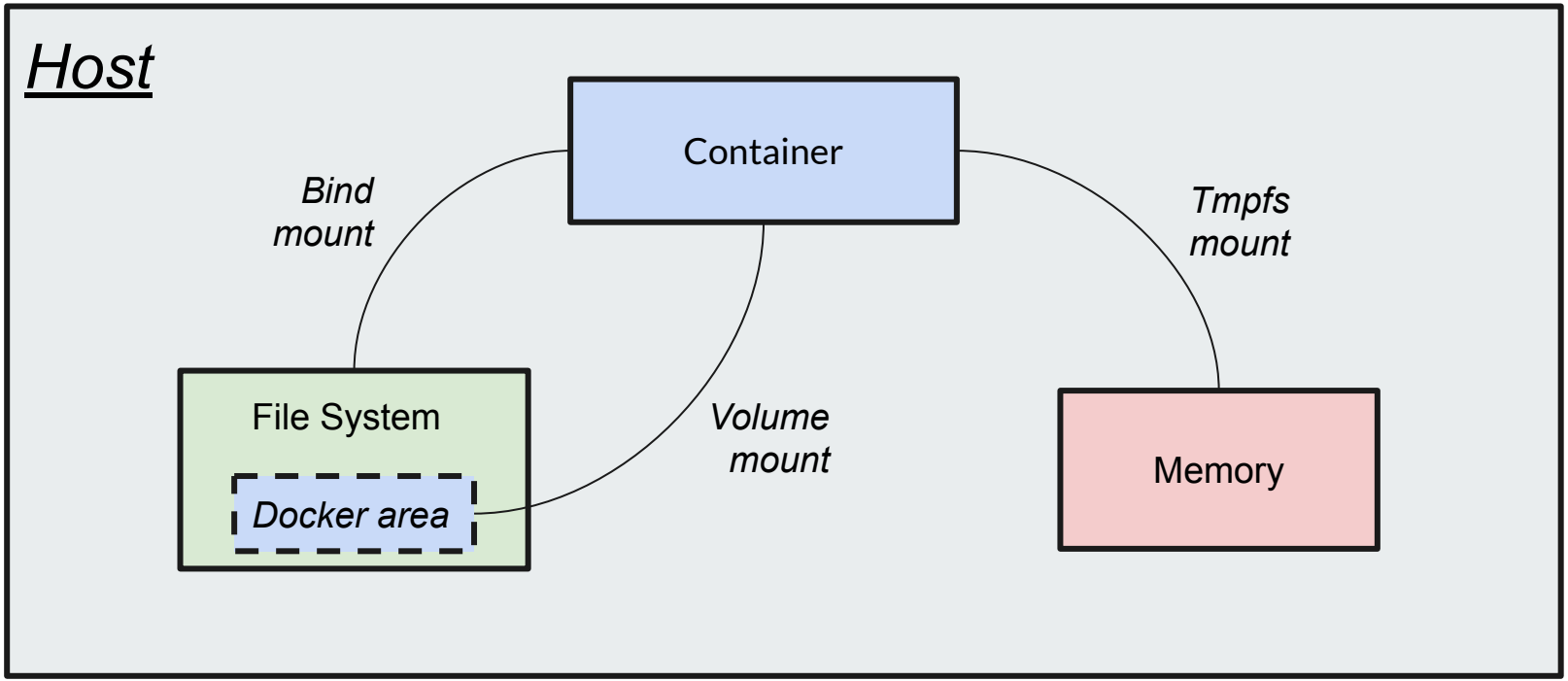
Important: the src/foo volume directory lives in the Docker engine, *not* the host filesystem

Bind mounts

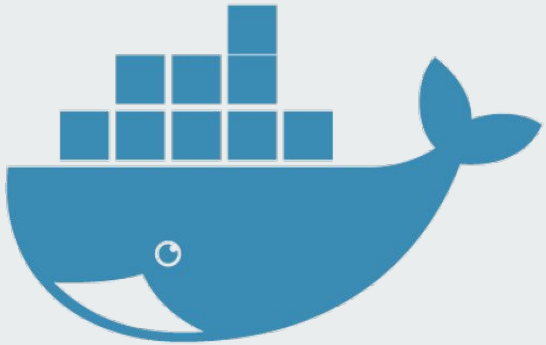
- Host machine directories, mounted into the container
- Slightly dangerous. Any non-docker application or process can modify the bind-mounted directory

Tmpfs mounts

- Don’t persist on host machine nor container. Any files here, live entirely in the docker container’s memory.


Host
Container
Bind mount
Tmpfs mount
File System
Docker area
Volume mount
Memory

Docker Networking

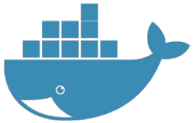


Docker Networking

- The system for how continers share resources and send messages to each other.

Default Docker Networks

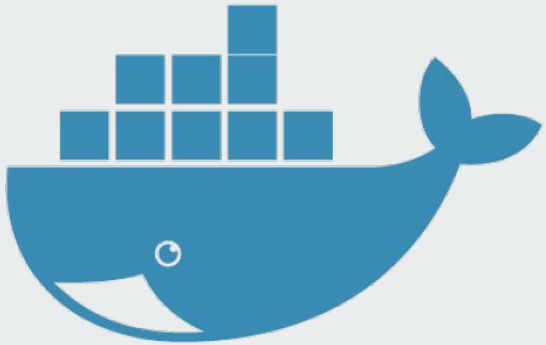
- **Bridge -** default Docker network.
- **None -** containers without an IP.
- **Host -** shares the network with the host machine.



User-defined Private Networks

- Set up connections between a handful of docker containers.
- Embedded a DNS server.
- Need network drivers, most commonly: **bridge**

Docker Compose



What is Docker Compose?

- A native tool from Docker to build applications that consist of more than one Docker container.
- Containers in compose are services.
- Define services with docker-compose.yml.

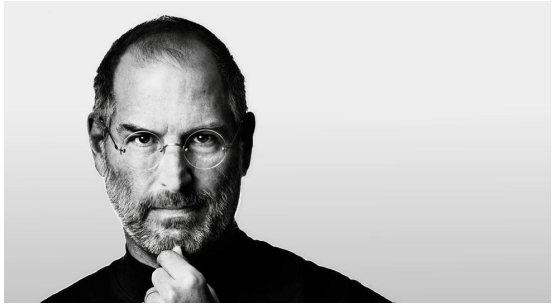
Why is Docker Compose useful?

- Start multiple containers with one command.

- Docker-compose.yml is also great documentation.

Core Features of Docker Compose

- Services (containers), private networks, and volumes.
- Caches configuration and run state for containers/services in the application.
- Multi-container isolated environments.



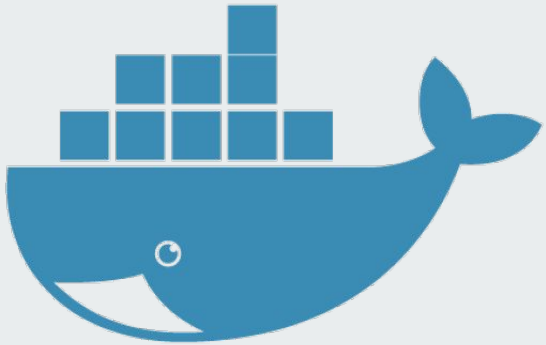
docker-compose.yml

- YAML (.yml or .yaml) file format. Nested key-value structure.

- **Docker-compose version.** Currently, on version 3.

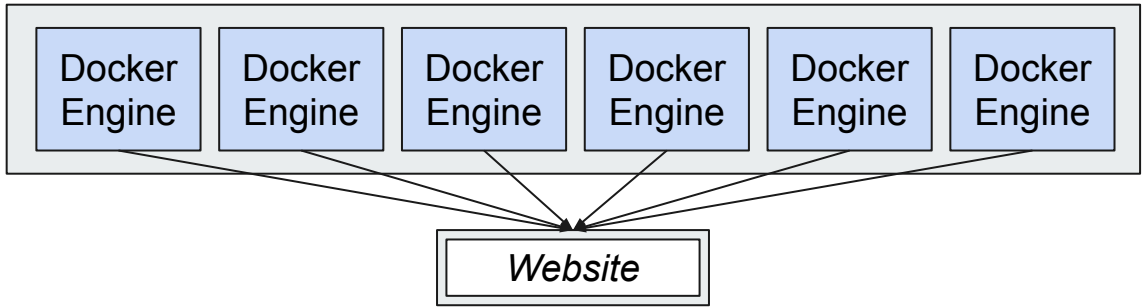
- Common entries in docker-compose.yml:
  services: ..., build: ..., image: ...,

Docker Swarm



Swarm mode

- A feature within Docker to maintain a cluster of machines running the Docker engine.
- The *swarm* is the cluster itself.



Swarm mode

- A **node** “is an instance of the Docker engine participating in the swarm.”
- Nodes will have a **manager** or **worker** role.
- A **service** runs software in the swarm based off of an image.
- A **replica** takes the image software and clones it onto nodes.

Why Docker Swarm?

- Extreme ability to scale.
- Production ready, well tested, and versatile.
- Great features: load balancing, service discovery, and more.