Hes·SO GENÈVE
Haute Ecole Spécialisée
de Suisse occidentale

Docker, Docker Compose, Swarm

Dr. Assane Wade
Francisco Mendonca
2023-2024

h e p i a
Haute école du paysage, d'ingénierie
et d'architecture de Genève

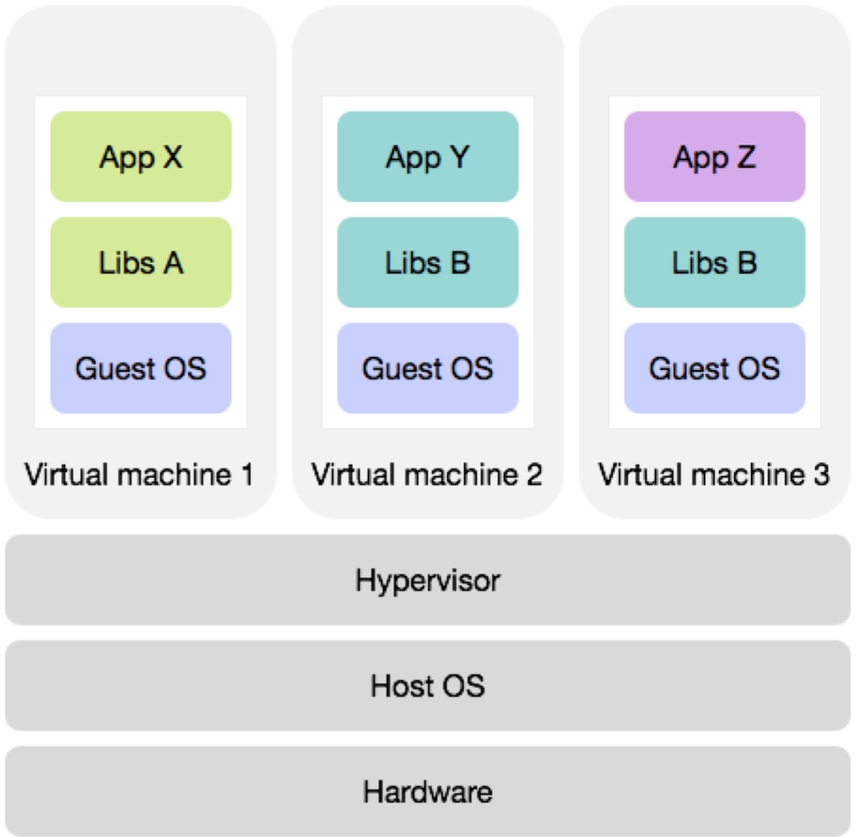
Container

A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another.

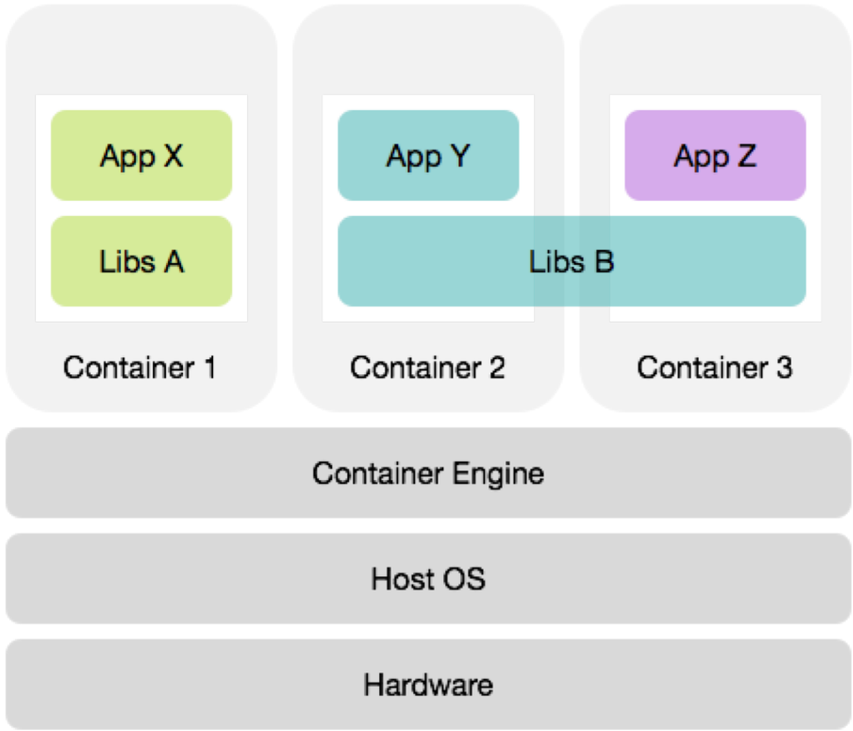
Eg. A Docker container image is a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings.

Virtualisation vs. “Containerisation”

Containers (OS level virtualisation) vs. Virtualisation (Full & Para)



Three VMs running on a single host



Three containers running on a single host

Virtualization vs. “Containerization”

- A container may be only **tens of megabytes** in size
- A virtual machine with its own entire operating system may be **several gigabytes** in size.
- A single server can host far more containers than virtual machines.
- Virtual machines may take **several minutes** to boot up their operating systems and begin running the applications they host
- Containerized applications can be started **almost instantly**.

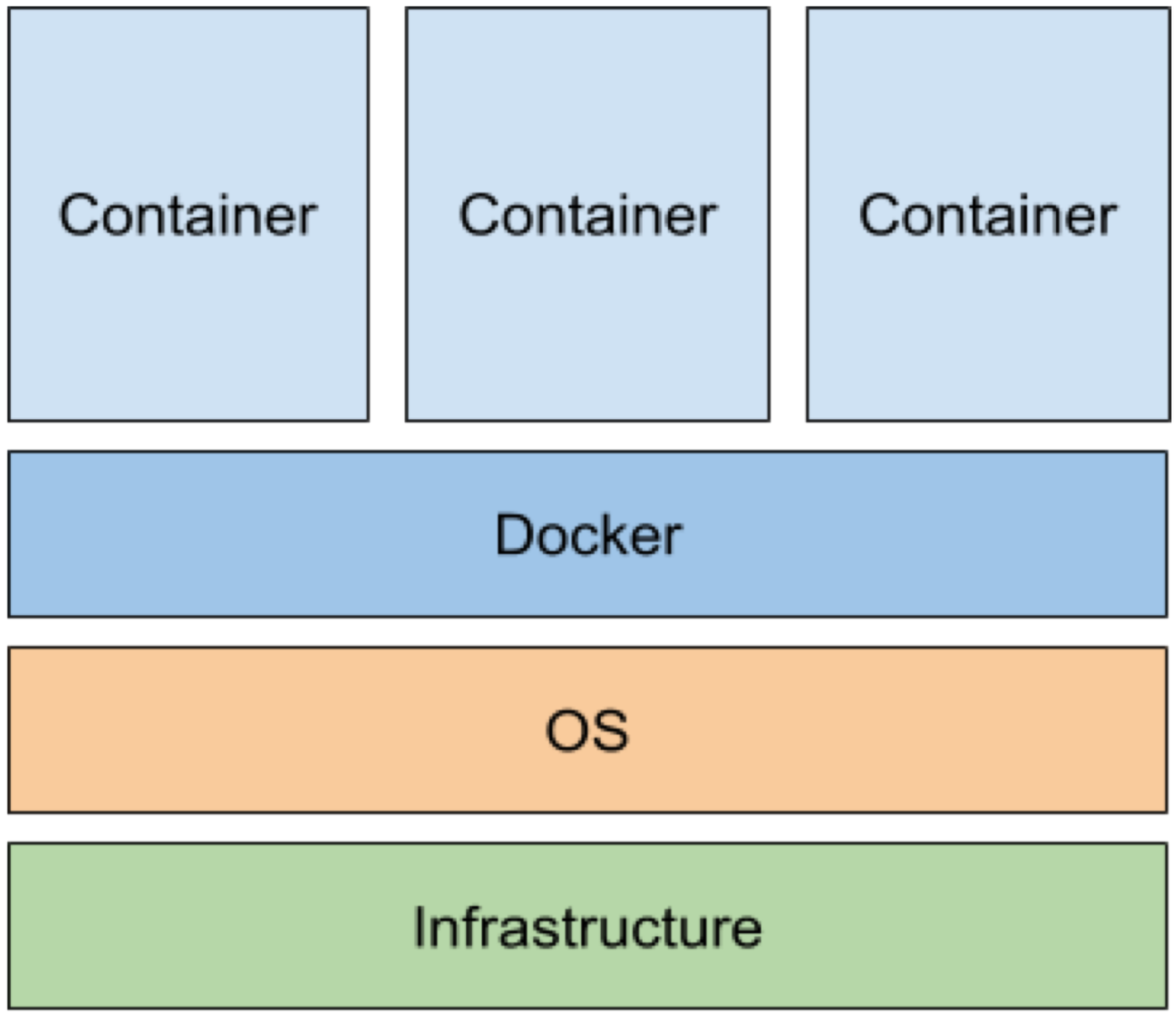
Docker - Definition

- OS-Based Virtualization to deliver software packages called **Containers**
- These containers are mostly isolated between one another
- All the containers run using a single Kernel
- Each Container has resource and environment isolation, controlled using Cgroups and Namespaces provided by the Linux Kernel
  - The level of isolation can be controlled
- A docker is represented by a Dockerfile
- Each Container interacts with the **Docker Engine**, which in turn interacts with the OS

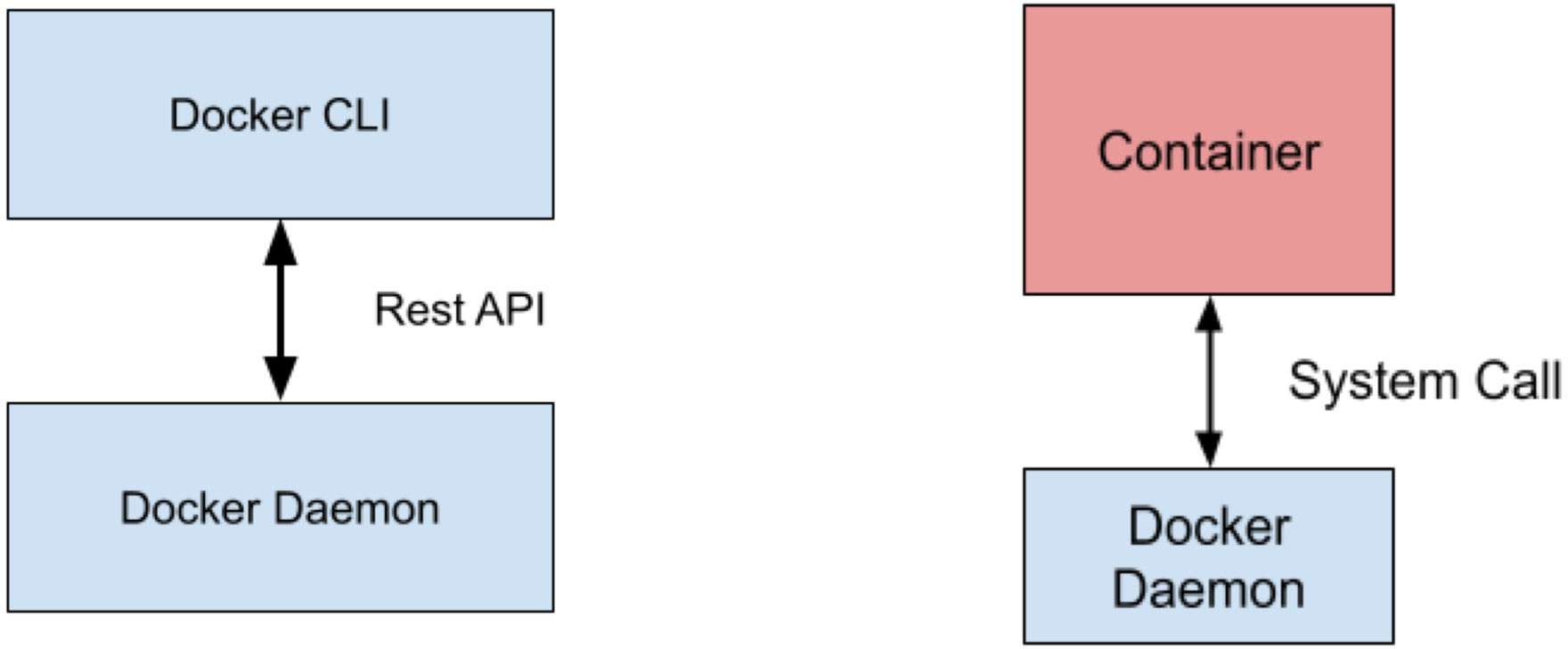
Docker Containers

- Runnable instance of an application
- Each container has all the files and folders to be able to run the application
- Each container can be Run, Stopped, Moved or Deleted using the Docker Client

Docker - Architecture



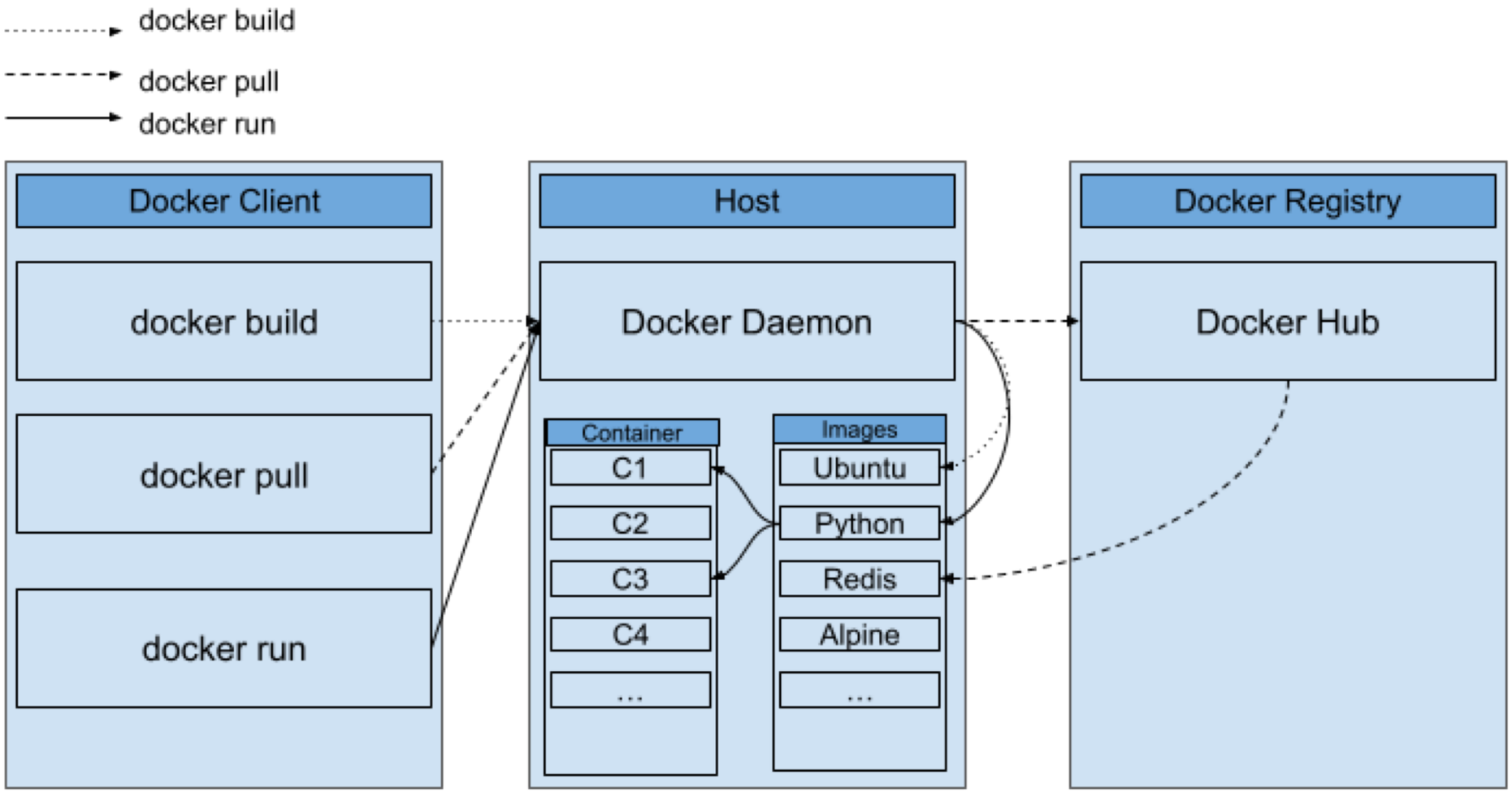
Docker - Daemon



Executes system calls made by the Containers
Manages the Images, Containers, Networks and Volumes
A Daemon can communicate with other daemons (Swarm)

Docker - Docker Client

- Used to interact with the Docker Daemon
- CLI commands
- One client can connect to several daemons

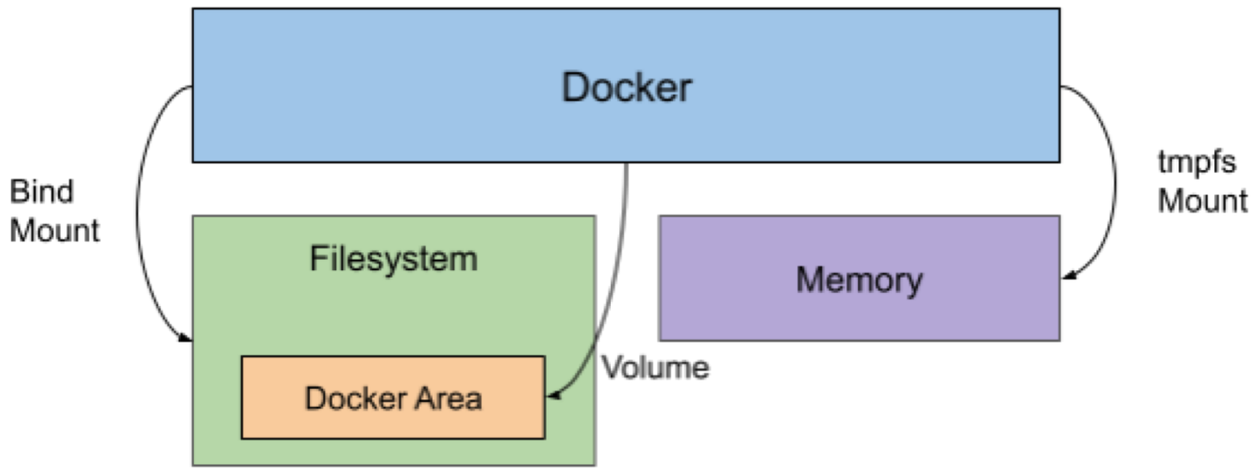


Docker Images

- A read-only template with instructions to build and run a Docker application
- An image is described in a Dockerfile
- Each instruction in the Dockerfile is a layer of the container

Docker Volumes

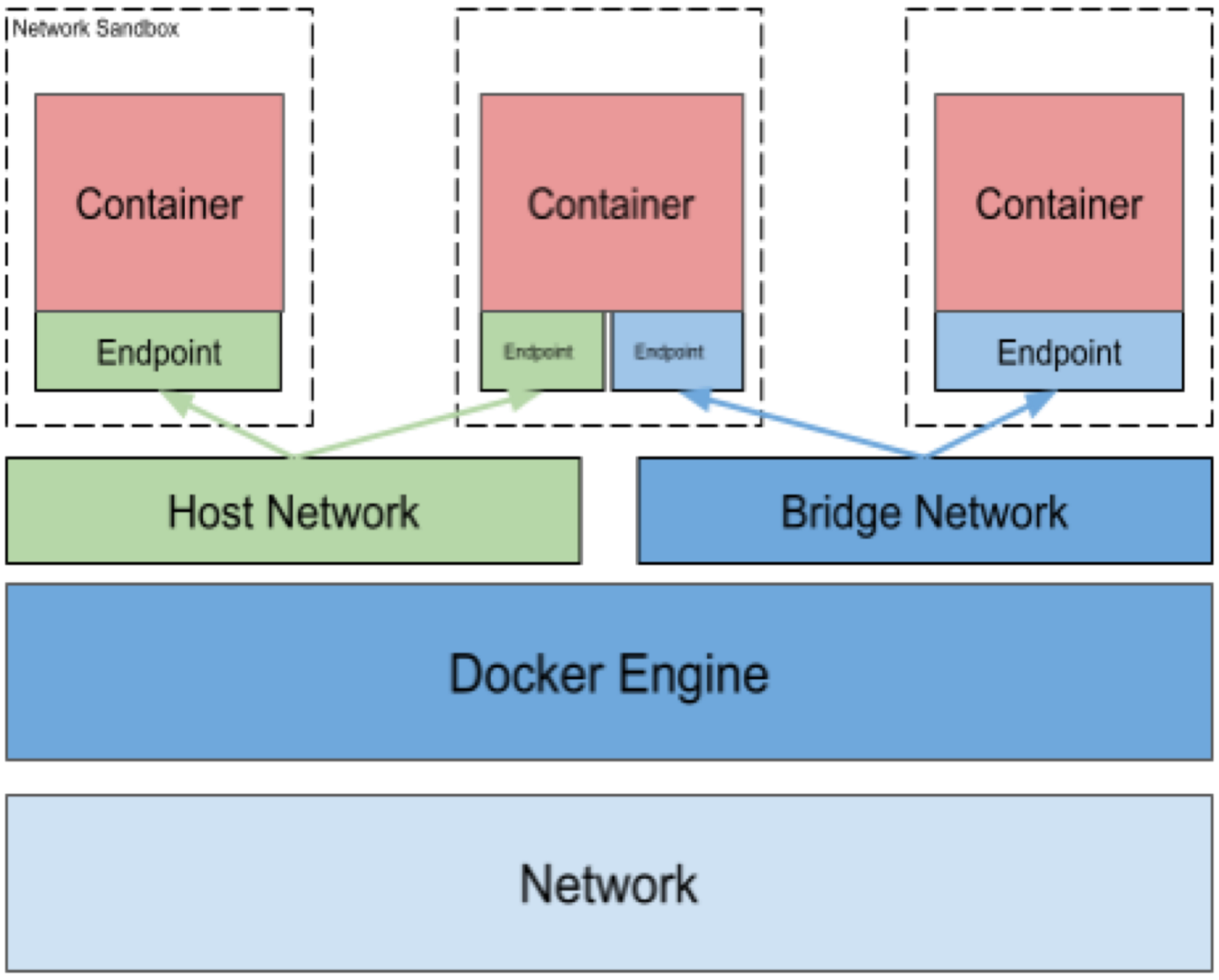
- Mechanism to use and store persistent data for a Docker Container
- Area of a filesystem that is managed by the Docker Engine



https://docs.docker.com/storage/volumes/

Docker Networks

Container networking refers to the ability for containers to connect to and communicate with each other, or to non-Docker workloads.



Docker Commonly Commands List

Container Management

`docker ps`	List the running containers.
`docker ps -a`	List all the containers, both running and stopped.
`docker create [image]`	Create a container without starting it.
`docker create -it [image]`	Create an interactive container with pseudo-TTY.
`docker rename [container] [new-name]`	Rename a container.
`docker rm [container]`	Remove a stopped container.
`docker rm -f [container]`	Force remove a container, even if it is running.
`docker logs [container]`	View logs for a running container.
`docker logs -f --until= [interval] [container]`	Retreive logs before a specific point in time.
`docker events [container]`	View real time events for a container.
`docker update [container]`	Update the configuration of a container.
`docker port [container]`	Show port mapping for a container.
`docker top [container]`	Show running processes in a container.
`docker stats [container]`	Show live resource usage statistics for a container.
`docker diff [container]`	Show changes to files or directories on the filesystem.
`docker cp [file-path] CONTAINER:[path]`	Copy a local file to a directory in a container.

Networking

`docker network ls`	View available networks.
`docker network rm [network]`	Remove a network.
`docker network inspect [network]`	Show information about a network.
`docker network connect [network] [container]`	Connect a container to a network.
`docker network disconnect [network] [container]`	Disconnect a container from a network.

General Management

`docker login`	Log in to a Docker registry.
`docker logout`	Log out of a Docker registry.
`docker inspect [object]`	Show low-level information about an object.
`docker version`	Show the version of the local Docker installation.
`docker info`	Display information about the system.
`docker system prune`	Remove unused images, containers, and networks.

Running a Container

`docker run [image] [command]`	Run a command in a container based on an image.
`docker run --name [container-name] [image]`	Create, start, and name a container.
`docker run -p [host]: [container-port] [image]`	Map a host port to a container port.
`docker run --rm [image]`	Run a container and remove it after it stops.
`docker run -d [image]`	Run a detached (background) container.
`docker run -it [image]`	Run an interactive process, e.g., a shell, in a container.
`docker start [container]`	Start a container.
`docker stop [container]`	Stop a container.
`docker restart [container]`	Stop a container and start it again.
`docker pause [container]`	Pause processes in a running container.
`docker unpause [container]`	Unpause processes in a running container.
`docker wait [container]`	Block input until the container stops.
`docker kill [container]`	Send a SIGKILL signal to stop a container.
`docker attach [container]`	Attach local standard input, output and error.
`docker exec -it [container] [shell]`	Run a shell inside a running container.

Image Management

`docker build [dockerfile-path]`	Create an image from a Dockerfile.
`docker build .`	Build an image using the files from the current path.
`docker build -t [name]: [tag] [location]`	Create an image from a Dockerfile and tag it.
`docker build -f [file]`	Specify a file to build from.
`docker pull [image]`	Pull an image from a registry.
`docker push [image]`	Push an image to a registry.
`docker import [url/file]`	Create an image from a tarball.
`docker commit [container] [new-image]`	Create an image from a container.
`docker tag [image] [image]:[tag]`	Tag an image.
`docker images`	Show all locally stored top level images.
`docker history [image]`	Show history for an image.
`docker rmi [image]`	Remove an image.
`docker load --image [tar-file]`	Load an image from a tar archive file.
`docker save [image] > [tar-file]`	Save an image to a tar archive file.
`docker search [query]`	Search Docker Hub for images.
`docker image prune`	Remove unused images.

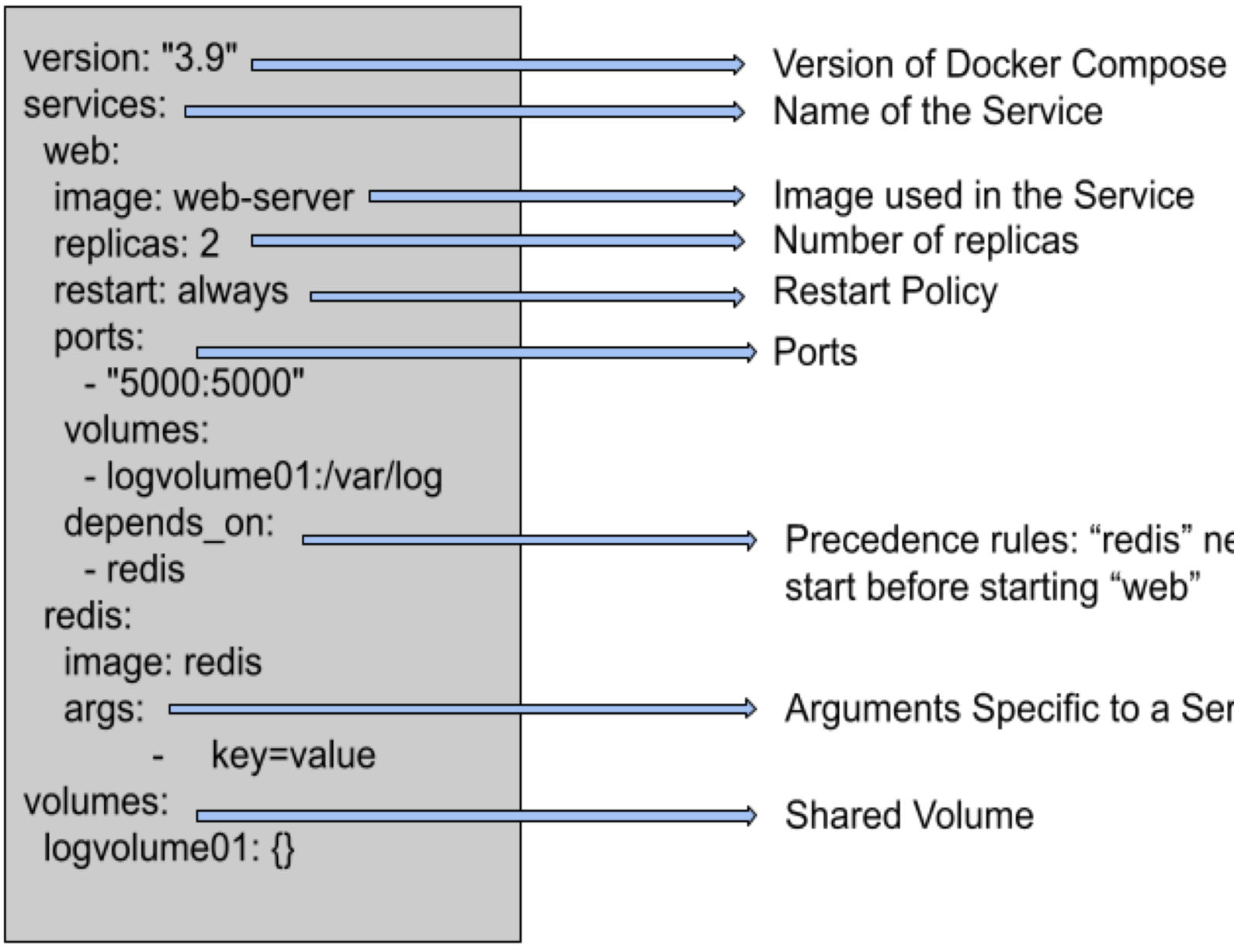
Plugin Management

`docker plugin enable [plugin]`	Enable a Docker plugin.
`docker plugin disable [plugin]`	Disable a Docker plugin.
`docker plugin create [plugin] [path-to-data]`	Create a plugin from config.json and rootfs.
`docker plugin inspect [plugin]`	View details about a plugin.
`docker plugin rm [plugin]`	Remove a plugin.

Docker Compose

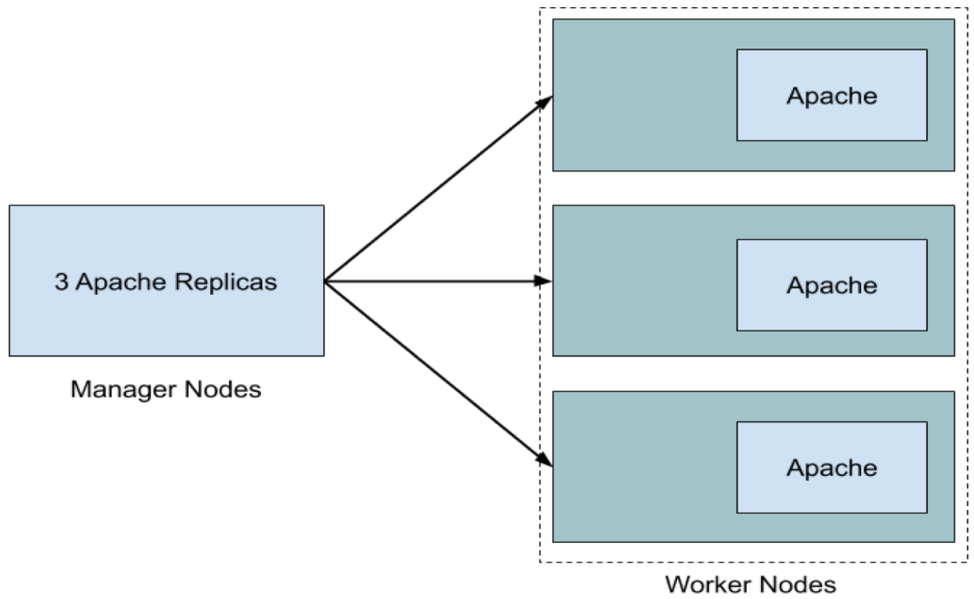
- Tool used to define and run Docker Containers.
- Built with YAML, it allows to create and run one or several containers with only one command.
- It defines *Services* - A group of containers with a specific state. The Daemon tries to maintain that state.
- Also able to define Volumes, Networks, and Global Variables
- Can be used locally or in a Docker Swarm
- To test locally:
  - Docker-compose up -d

Docker Compose



Docker Swarm - Definition

- A swarm consists of multiple Docker hosts which run in swarm mode (Managers, Workers or both)
- A Swarm allows:
  - To deploy applications without the need of additional orchestration software
  - Specialization handled at runtime
  - Automatic scaling by defining the number of tasks for each service
  - Load balancing. Can be internal or external load balancer.



Docker Swarm - Definition

- **Declarative service model:** Docker Engine uses a declarative approach to let you define the desired state of the various services in your application stack.
- **Desired state reconciliation:** The swarm manager node constantly monitors the cluster state and reconciles any differences between the actual state and your expressed desired state.
- **Service discovery:** Swarm manager nodes assign each service in the swarm a unique DNS name and load balance running containers.
- **Secure by default:** Each node in the swarm enforces TLS mutual authentication and encryption to secure communications between itself and all other nodes. You have the option to use self-signed root certificates or certificates from a custom root CA
- **Rolling updates:** At rollout time you can apply service updates to nodes incrementally. The swarm manager lets you control the delay between service deployment to different sets of nodes. If anything goes wrong, you can roll back to a previous version of the service.

Docker Swarm - Service

- To define the tasks on the nodes
- When you create a service, you specify which container image to use and which commands to execute inside running containers.
- In the replicated services model, the swarm manager distributes a specific number of replica tasks among the nodes based upon the scale you set in the desired state.
- For global services, the swarm runs one task for the service on every available node in the cluster.

Swarm vs Standalone Containers

- With swarm services you can modify a service's configuration without the need to manually restart the service.

- When Docker is running in swarm mode, you can still run standalone containers on any of the Docker hosts participating in the swarm, as well as swarm services.

- Only swarm managers can manage a swarm, while standalone containers can be started on any daemon.

Docker Swarm Command List

SWARM MANAGEMENT

SET UP MASTER	docker swarm init --advertise-addr <ip>
FORCE MANAGER ON BROKEN CLUSTER	docker swarm init --force-new-cluster -advertise-addr <ip>
ENABLE AUTO LOCK	docker swarm init --autolock
GET TOKEN TO JOIN WORKERS	docker swarm join-token worker
GET TOKEN TO JOIN NEW MANAGER	docker swarm join-token manager
JOIN HOST AS A WORKER	docker swarm join <server> worker
LEAVE THE SWARM	docker swarm leave
UNLOCK A MANAGER HOST AFTER DOCKER	docker swarm unlock
PRINT KEY NEEDED FOR 'UNLOCK'	docker swarm unlock-key

HANDLING NODES

PRINT SWARM NODE LIST	docker node ls
REMOVE ONE OR MORE NODES FROM THE SWARM	docker node rm <node id>
DISPLAY DETAILED INFORMATION ON ONE OR MORE NODES	docker node inspect --pretty <node id>
PROMOTE NODE TO MANAGER	docker node promote <node id>
DEMOTE ONE OR MORE NODES FROM MANAGER IN THE SWARM	docker node demote <node id>

LABELLING NODES

ADD LABEL	docker node update --label-add <key>=<value> <node>
REMOVE LABEL	docker node update --label-rm <key> <node>
LIST LABELS	docker node inspect <node> | grep Labels -C5