Containers for virtualization

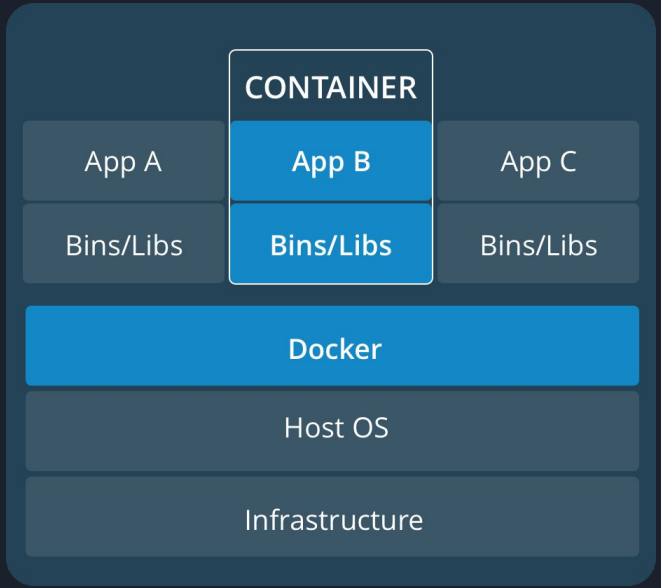
Hubert Mugabo Pitie - Salim Boutlendj

2018-2019

TABLE OF CONTENTS

Intro to containers - **Description**

- **Virtualization** means running a virtual instance of a computer system or a component of the computer system in a layer abstracted from the physical hardware.
- **Container** is a type of Virtualization at the OS level based on the **LXC** Linux Containers or using a container engine like **Docker** or **Mesos**.
- Containers can use **virtualized networks and storage**.



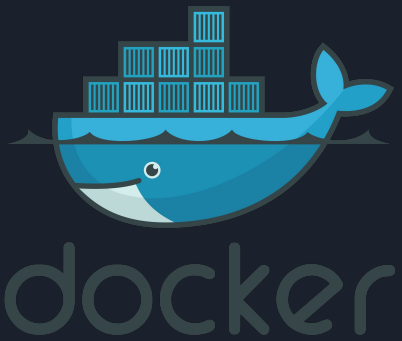
Intro to containers - **Why?**

- Containers are used to **isolate** an application **at the OS level**, by creating an **unique environment** for each container on the host.
- **Portability**: By giving the the necessary libs/dependencies within the container. Making the application **environment independent from the host**.
- Increase the developers **productivity** because of the simplicity of configuration and deployment.
- Efficient usage of the **resources**.
- Better performances than **VMs** in application deployment : **Takes less disk & memory spaces**.
- **Horizontal scalability** made easy thanks to **portability** and deployment simplicity.

Intro to containers - **Drawbacks**

- **Security** not as good as Virtual Machine since containers use the OS API. Containers share the kernel, other components of the OS and they can have **root access**.
- **Not all applications** benefit from containers. Microservice and standalone applications are suited for containers.
- Do not use it like a **golden hammer**, sometimes **resources are limited** and they should be managed properly.

Intro to containers - **Tools**



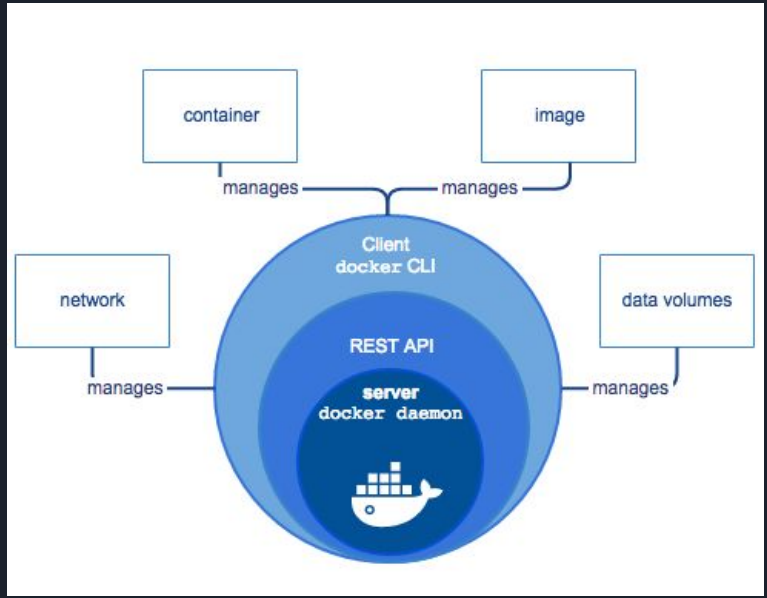
Intro to containers - **Tools :** *Docker vs Mesos*

- Docker is mainly a **containerization engine**.
- Could be joined to a **container orchestrator**.
- **Simple to setup** with one command but limited. It's great for its **simplicity** and ability to scale existing Docker Compose services.

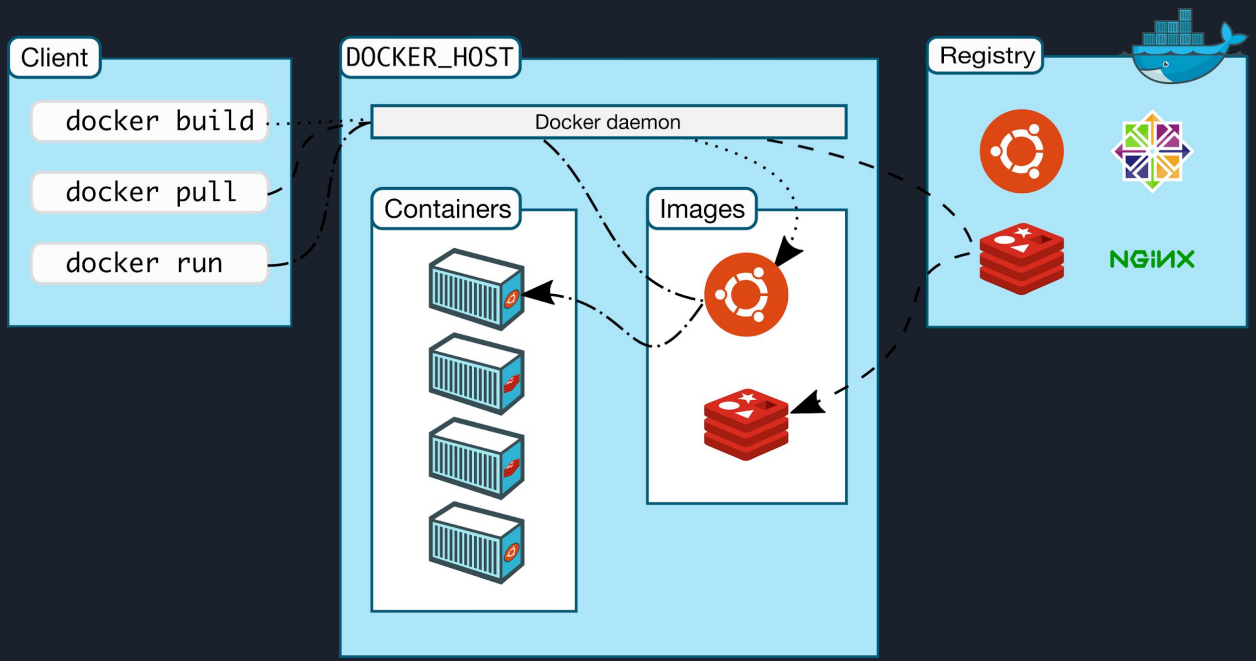
- Mesos is a **cluster manager** that uses containerization to execute tasks on slaves.
- Mesos might **use Docker** as a containerizer.
- Best suited for data centers (**large systems**) where multiple applications need to be setup and configured.

Docker - **Overview**

- Containers **images** are built using a **Dockerfile** that lists the OS image to use and the commands to execute in order to **build** the required **environment** for the application.
- Images can be built from scratch or pulled from the **docker registry** (Docker Hub), e.g : if you want a redis database running on a container, an existing image provided by redis maintainers is available on the Docker Hub.
- The containers run the built image and use **virtual network** and **data volume** that are managed by the engine to **communicate** with each other and **store data**.



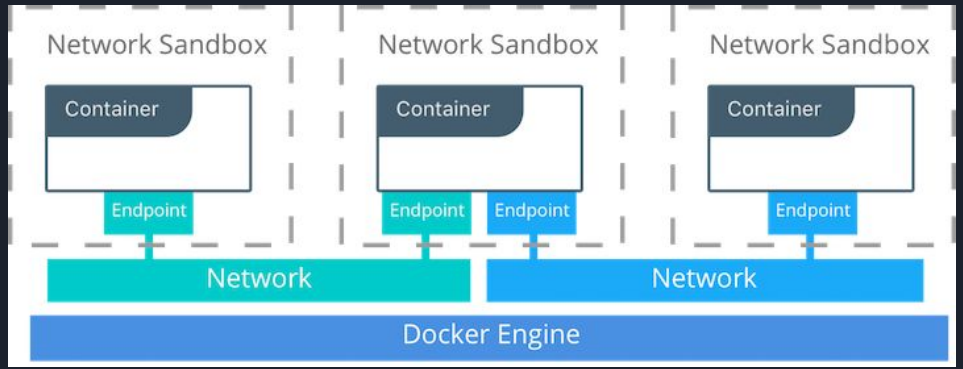
Docker - **Overview**



Docker - **Network**

The docker engine has the ability to create virtuals networks to **connect** working **containers**.
e.g: the application and its database should be on their own virtual network to share information.

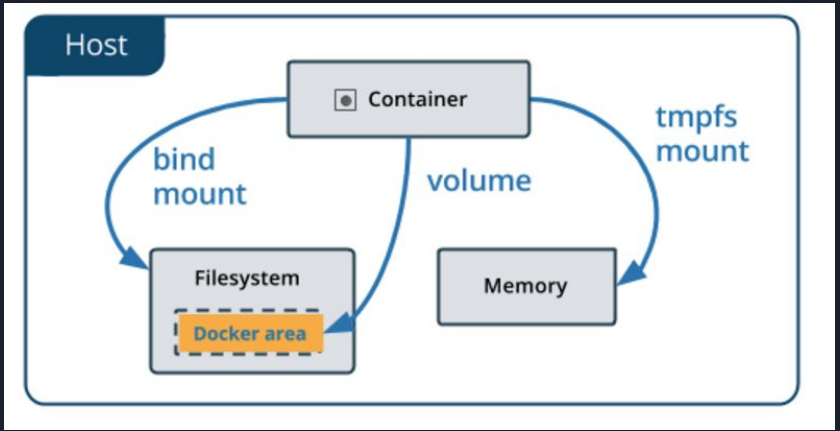
- It provides an **isolation** between the host's network and the containers.
- however it is still possible to **connect** the containers to the **host network** depending on the type of the network's driver used. e.g: To use the Bluetooth or Wifi host's interface.



Docker - **Volume**

Prefered mechanism used to **persist data** generated and used by Docker **container.**
e.g: a container running a database image should be binded to volume in order to persist data if not data will be lost at reboot.

- **Bind mounts** reference to **existing files or directories** on the host machine.
- **Volume** are used to **store containers data.**
- Volumes don't increase the container size, their content exist **outside** the lifecycle of a given **container** .



Docker - **Compose**

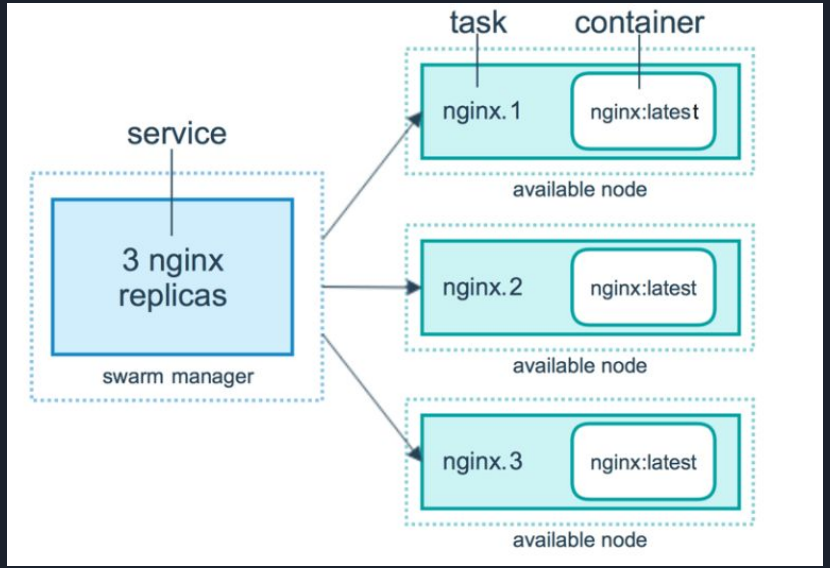
Docker compose is a tool provided by the docker engine that is used to **build** and run an entire **application** composed by many containers, networks and volumes.

- Compose defines and **runs** Docker **containers**, **networks** and **volumes**.
- Configuration is done through a **YAML file**
- Compose is a 3-steps process:
  1. Creating the app environment by building the services that we defined
  2. Creating networks and storages.
  3. Starts and runs the containers.

Docker - **Swarm**

Swarm is an **orchestrator** for docker containers.

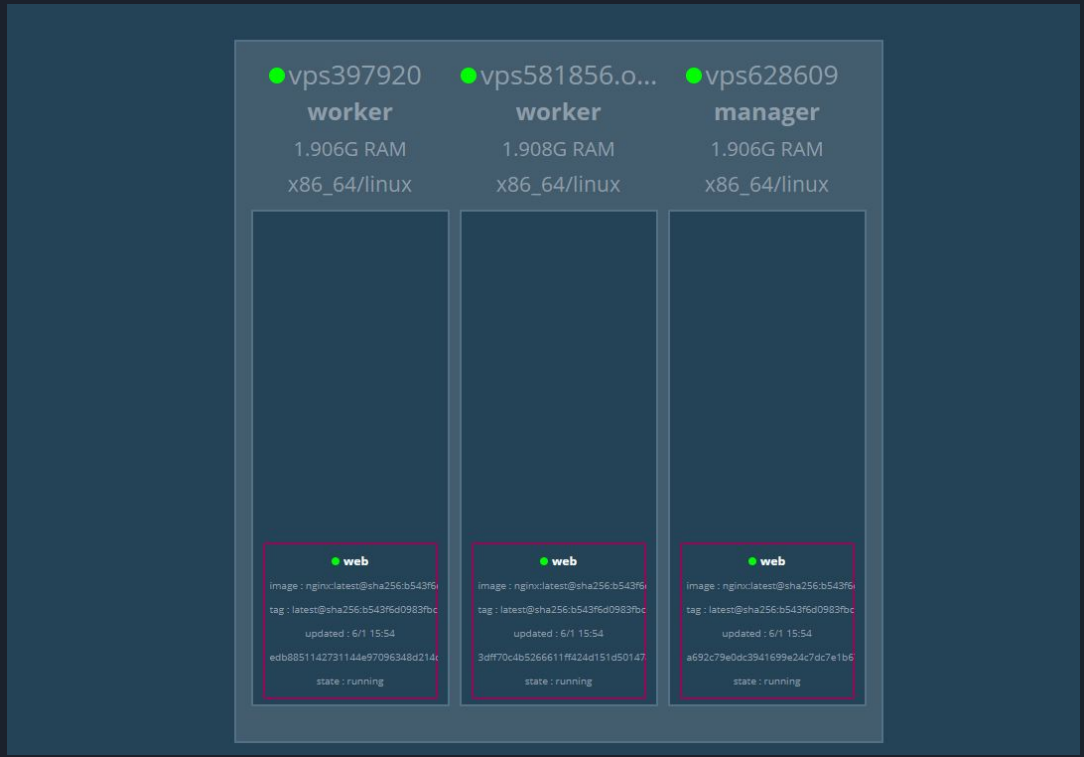
- The **architecture** is completely and physically **distributed** over multiple nodes .
- The swarm manager builds **containers images** as **services** and the **replicats** are attributed to **nodes** as **tasks**.
- The **swarm manager** acts as a **load balancer**, a **service discovery** and a **circuit breaker.**



Containers for Virtualization - **Overview**

- Users do not need to be aware of network decentralization thus the docker swarm operates a container orchestrator, load balancer and service discovery to make this possible.
  - **Container orchestrator**: used for **integrating** and **managing** containers spread on **multiple servers**.
  - **Load balancer**: act as a **scheduler** and dispatcher of **network requests** for the cluster with the use of known **algorithms as RR** or other **strategies** like **CPU** or **network** load.
  - **Service discovery**: **register** all the remote **node** using a **name resolution** to prevent issues due to IP shift.

Containers for Virtualization - **How**



Conclusion

- The development, build, test and production environments are **standardized**.
- The docker tool suite makes it possible to **abstract** a fully **distributed architecture** as if it were running on the same server.
- Ideal tool to cope with the **traffic load** thanks to its ability to **scale the application**.

Credits

- https://docs.docker.com/get-started/#containers-and-virtual-machines
- https://docs.docker.com/engine/docker-overview/#docker-engine
- https://success.docker.com/article/networking
- https://docs.docker.com/storage/volumes/
- https://hackernoon.com/kubernetes-vs-docker-swarm-a-comprehensive-comparison-73058543771e
- https://mesosphere.com/blog/docker-vs-kubernetes-vs-apache-mesos/
- https://codefresh.io/kubernetes-tutorial/kubernetes-vs-docker-swarm-vs-apache-mesos/
- https://docs.docker.com/engine/swarm/swarm-tutorial/