

# Container vs VM

Let's set some criteria to compare them



- Operating Systems
- Architecture
- Isolation
- Efficiency
- Portability
- Scalability
- Deployment



# Container vs VM



## Operating System

### Container

Contains only the bare minimum parts of the Operating system required to run the software. Updates are easy and simple to do

### VM

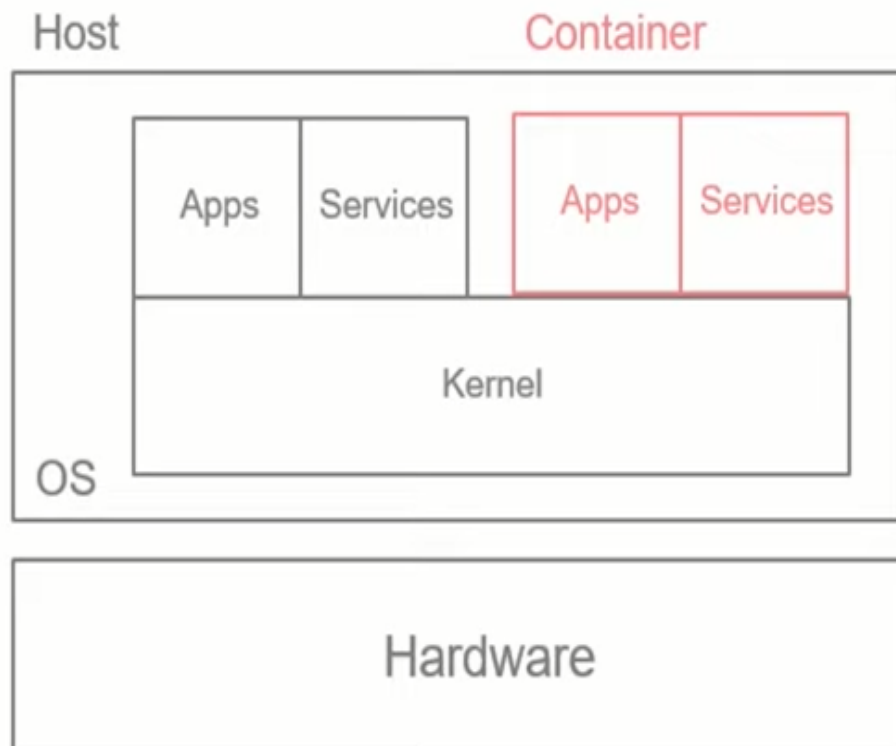
Contains the complete Operating system that is normally used on systems for general purpose. Updates are time consuming and tough.

# Container vs VM

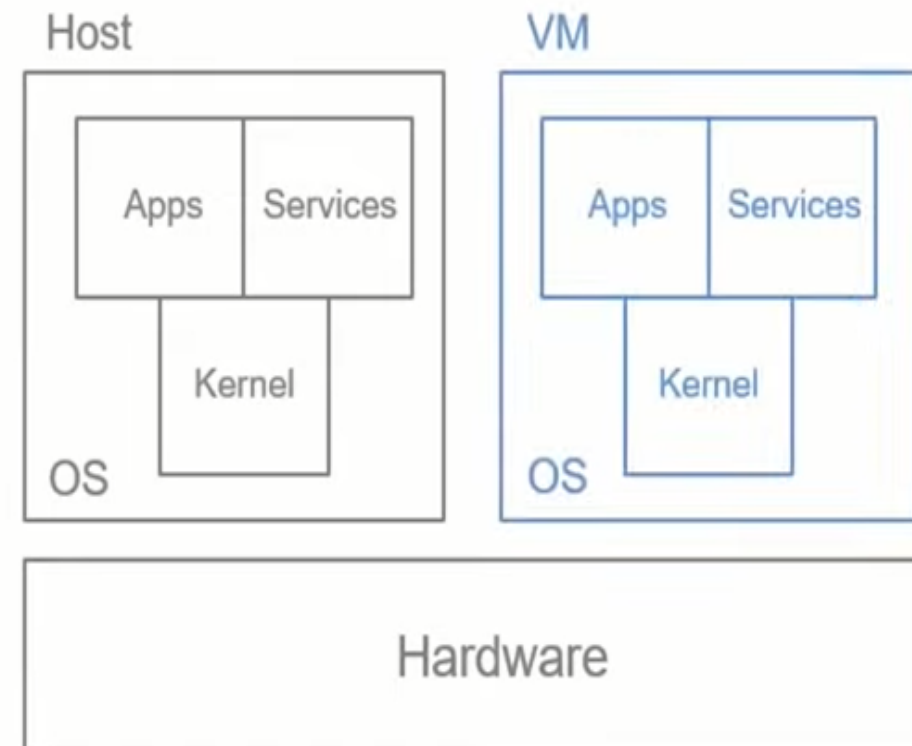


## Architecture

### Container



### VM



# Container vs VM



## Isolation

### Container

The isolation provided by a container isn't as complete as of a VM but is adequate.

### VM

VM provides complete isolation from the concerning host system and is also more secure.

# Container vs VM



## Efficiency

### Container

Container are way more efficient as they only utilise the most necessary parts of the Operating system. They act like any other software on the host system

### VM

VM are less efficient as they have to manage full blown guest operating system. VM's have to access host resource through a hypervisor.

# Container vs VM



## Portability

### Container

Containers are self-contained environments that can easily be used on different Operating systems,

### VM

VMs aren't that easily ported with the same settings from one operating system to another.

# Container vs VM



## Scalability

### Container

Containers are very easy to scale, they can be easily added and removed based on requirements due to their light weight.

### VM

VMs aren't very easily scalable as they are heavy in nature.

# Container vs VM



## Deployment

### Container

Containers can be deployed easily using the Docker CLI or making use of Cloud services provided by aws or azure.

### VM

VMs can be deployed by using the powershell or by using the VMM or using cloud services such as aws or azure.





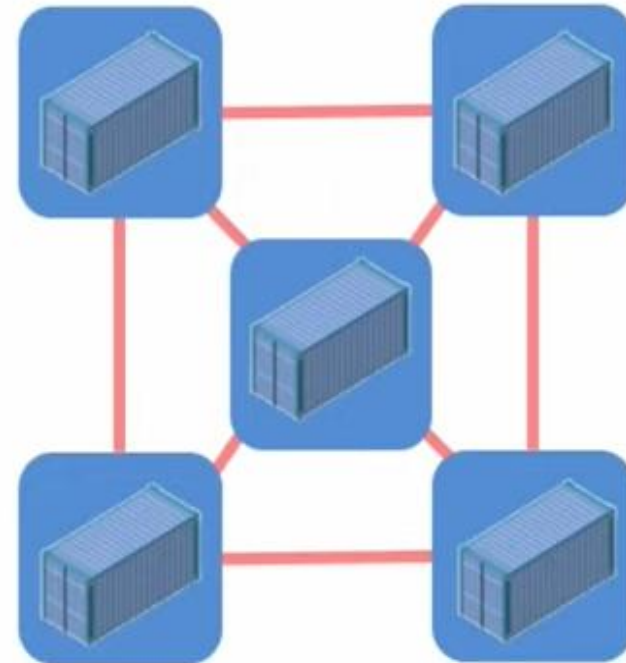
Why do we need **Containers**?

# Why do we need Containers?

Consistent Development  
Environments

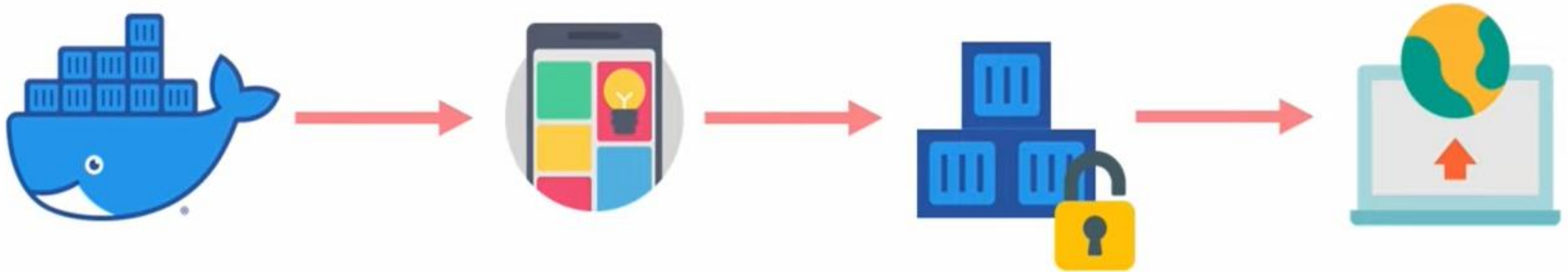


Microservices



# What is Docker?

Docker is a tool that helps in developing, building, deploying and executing software in isolation. It does so by creating containers that completely wrap a software.



The Isolation provided by container gives a layer of security to the containers.

# Why Docker?



**Simple**



**Fast**



**Easy Collaboration**



**Built for Developers, by  
Developers**



**Docker Community**





**Docker Environment**

# Docker Environment



**Docker Engine**



**Docker Objects**



**Docker Registry**



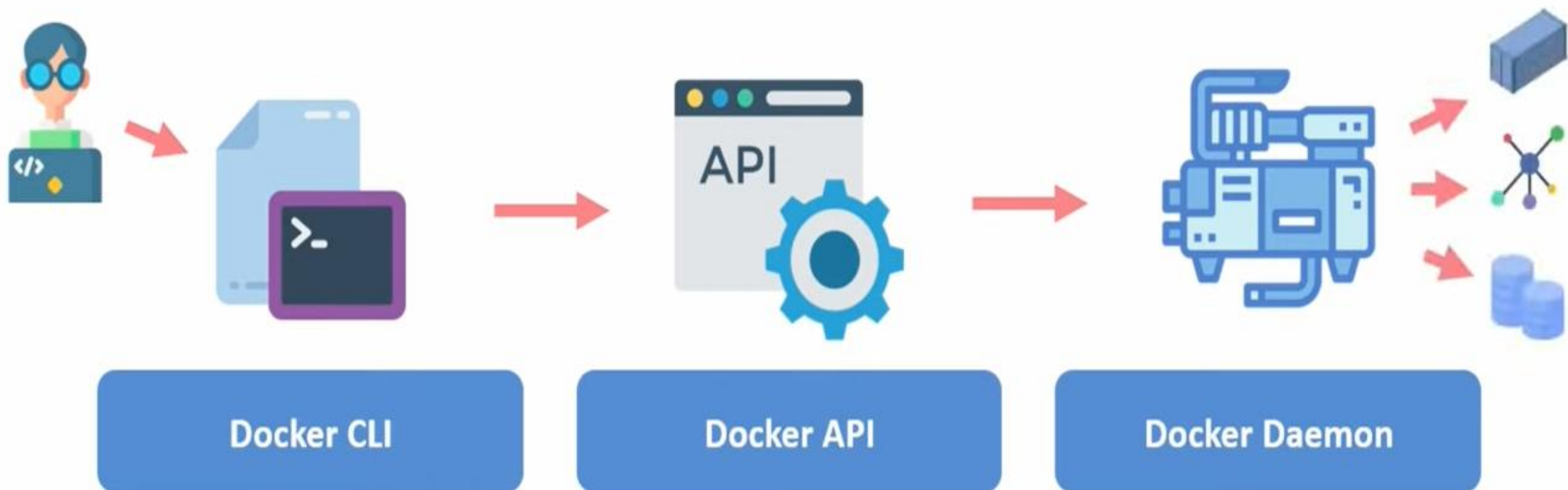
**Docker Compose**



**Docker Swarm**

# Docker Engine

Docker engine is as the name suggests, its technology that allows for the creation and management of all the Docker Processes. It has three major parts to it.



# Docker Objects



**Docker Images**



**Docker Containers**



**Docker Volumes**



**Docker Networks**



**Docker Swarm Nodes &  
Services**



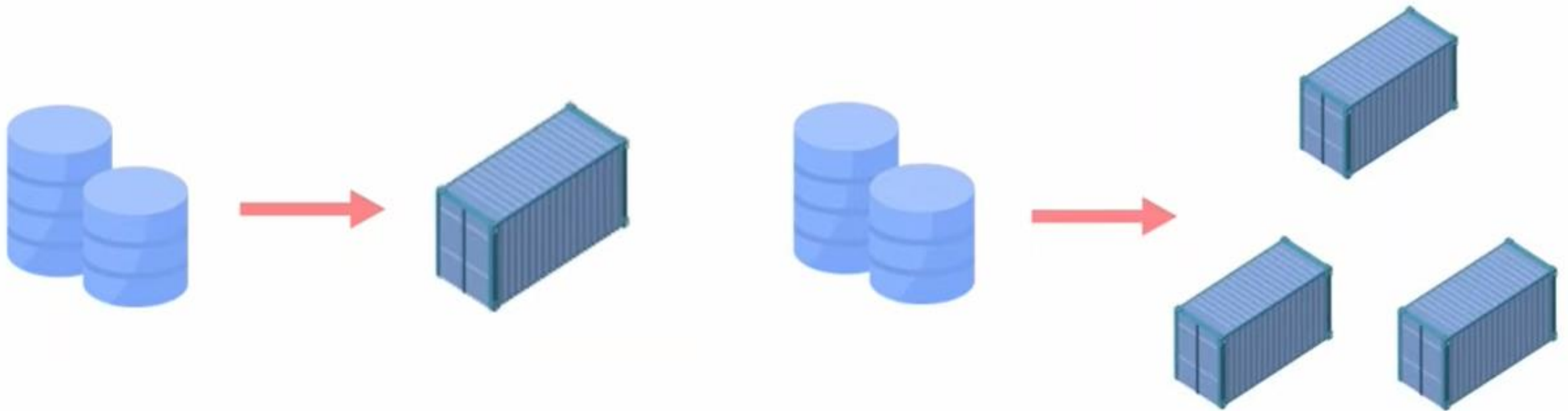
# Docker Objects - Images

Docker images are sets of instructions that are used to create containers and execute code inside it.



# Docker Objects - Volumes

Docker Volumes are basically persistent storage locations for the containers. They can be easily & safely attached and removed from different container. And they are also portable from system to another.



# Docker Objects – Volumes drivers

Docker Volumes drivers allow you to perform unique abilities such as creating persistent storage on other hosts, cloud, encrypt Volumes. They basically enhance the abilities of a Volume.



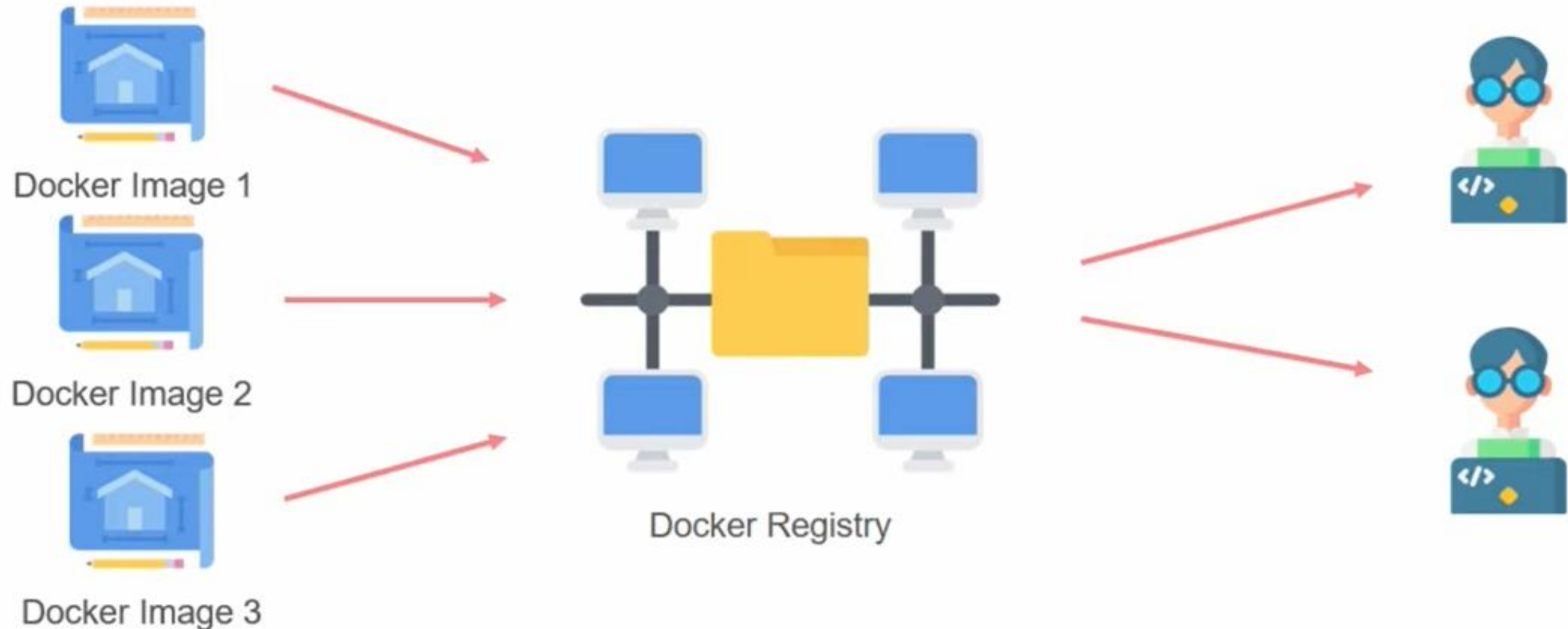
## Docker Objects - Network

A Docker network is basically a connection between one or more containers. One of the more powerful things about the Docker containers is that they can be easily connected to one other and even other software, this makes it very easy to isolate and manage the containers



# Docker Registry

You can think of registries as storage locations for Docker Images. These images can be versioned in the registry as well.





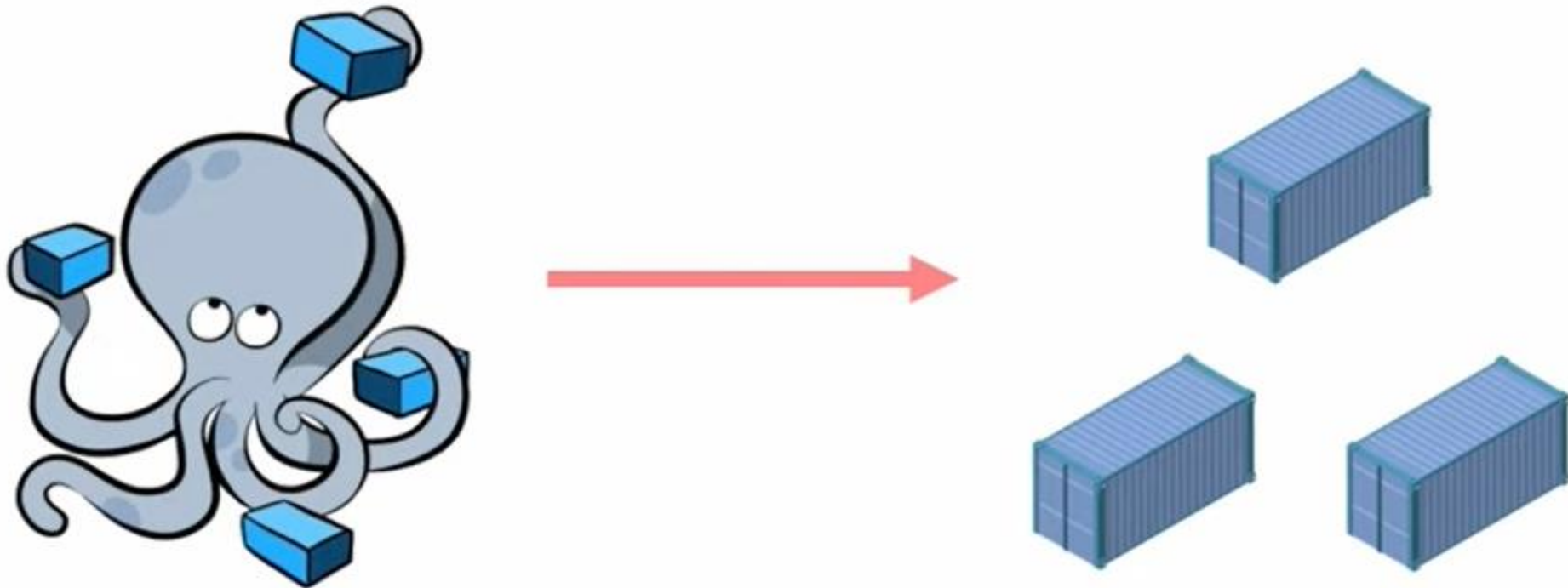
# Docker Registry – DockerHub

You have many options for a Docker Registry, you can go with DockerHub as your main Docker registry as there is already a Docker command to pull and push images to it. If you don't want to use DockerHub there are many alternatives to it.



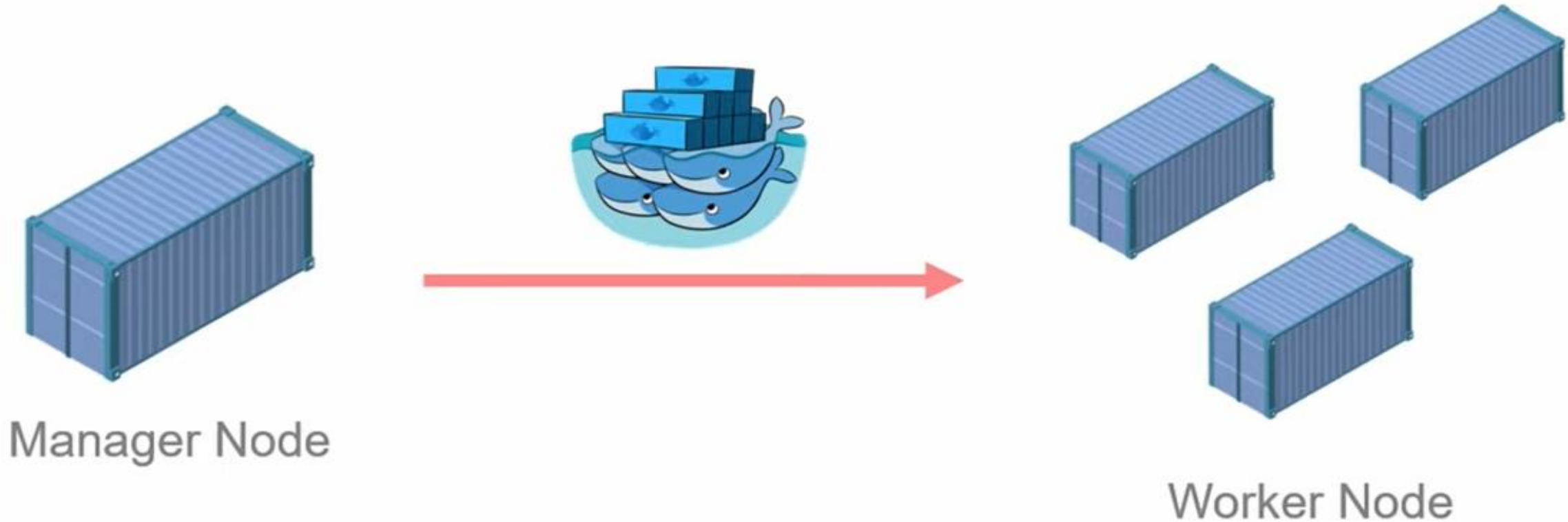
# Docker Compose

For Now let's just understand that Docker Compose is just a Service within Docker that let's us launch multiple containers at the same time.



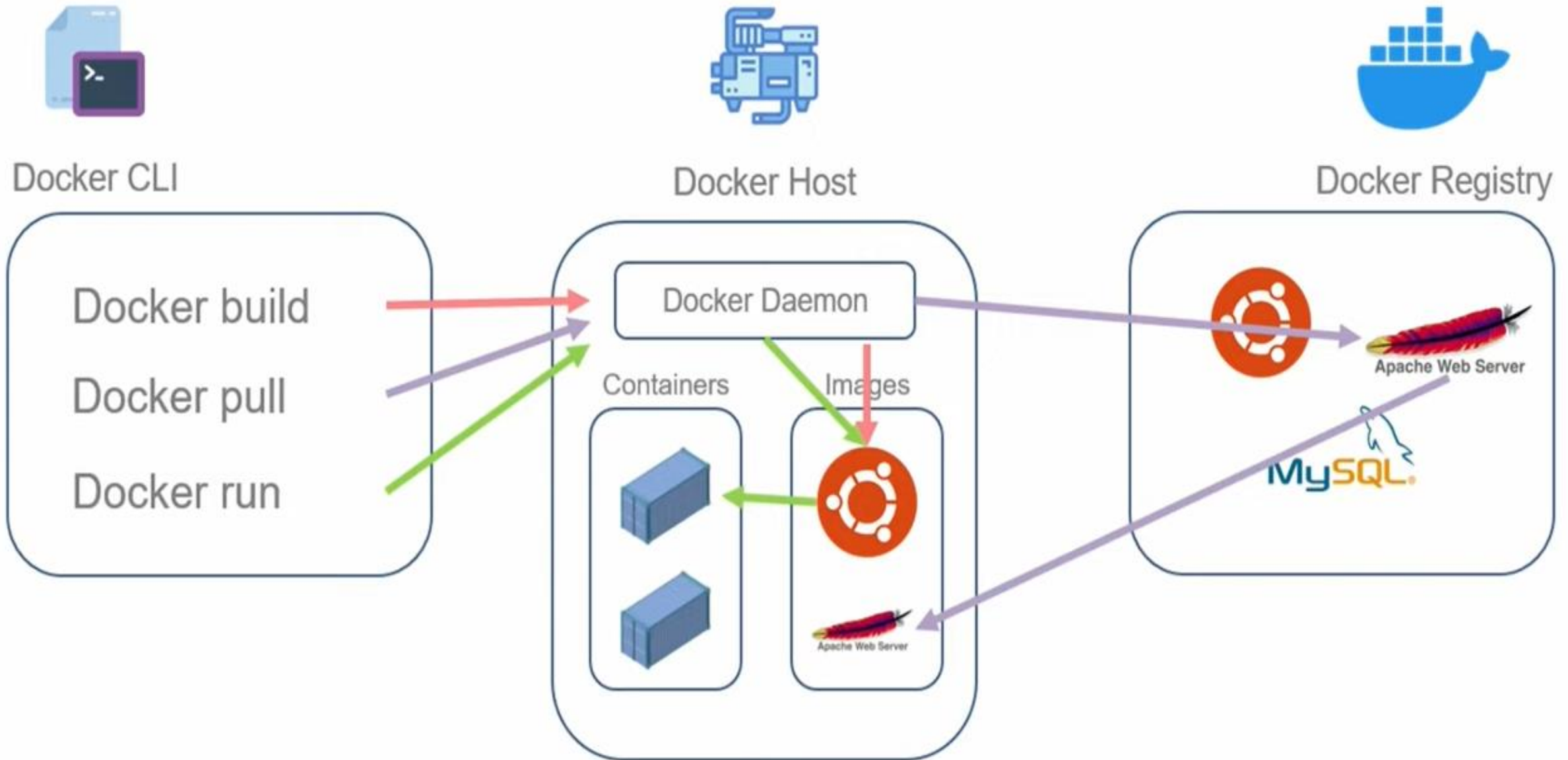
# Docker Swarm

For Now just understand that Docker Swarm is a service within Docker that allows us to manage multiple containers.





# Docker Architecture



Thanks