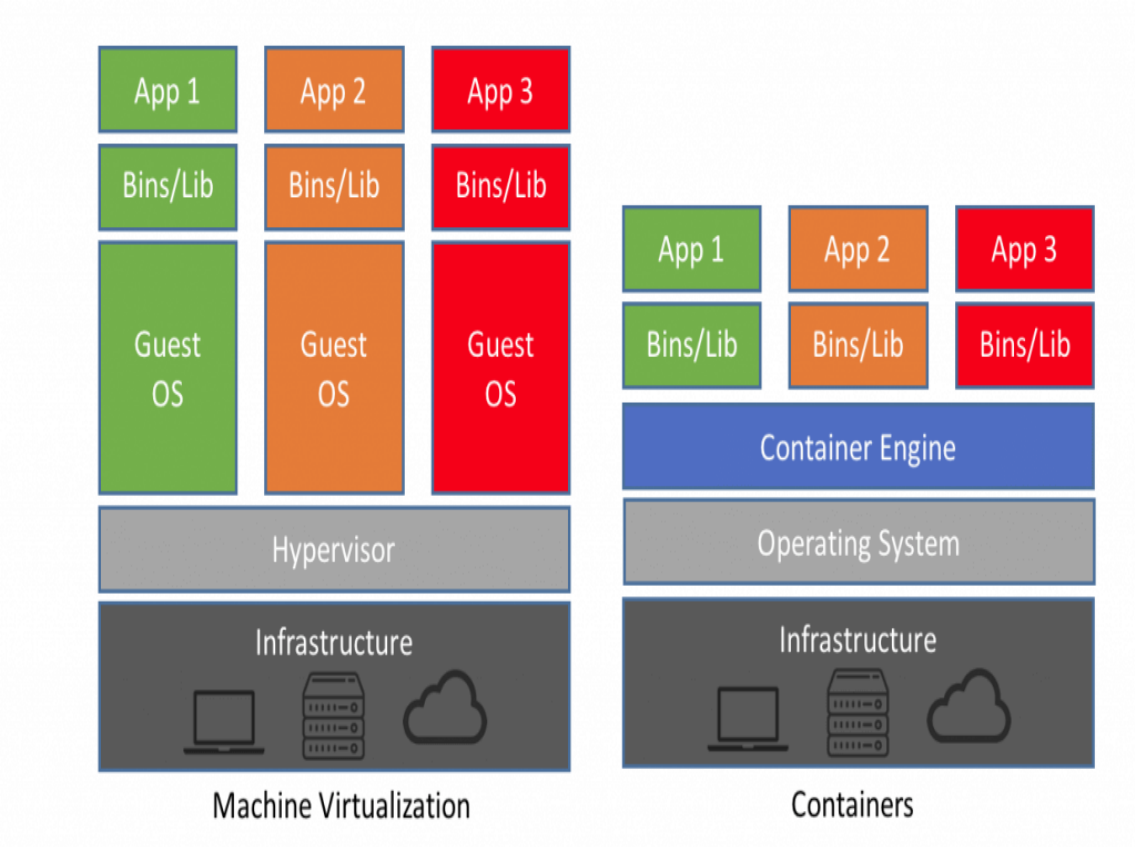
Docker, Usage and Cheat Sheet

Madhusudan Sangam

Docker is a software development platform for virtualization with multiple Operating systems running on the same host. It helps to separate infrastructure and applications to deliver software quickly. Unlike Hypervisors, which are used for creating VM (Virtual machines), virtualization in Docker is performed on system-level, also called Docker containers.
As you can see the difference in the image below, Docker containers run on top of the host's Operation system. This helps you to improves efficiency and security. Moreover, we can run more containers on the same infrastructure than we can run Virtual machines because containers use fewer resources.



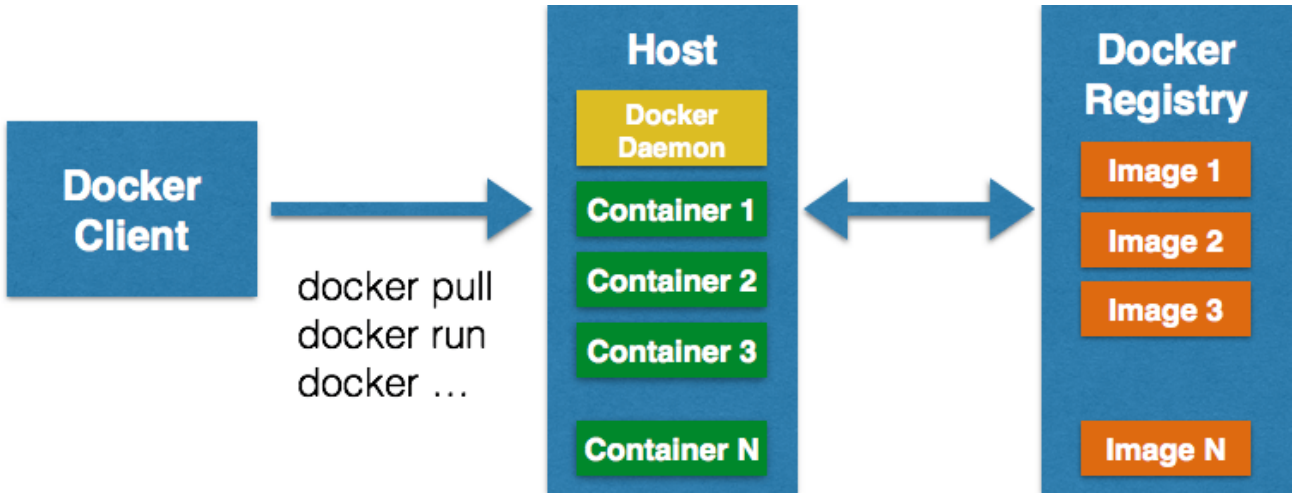
Unlike the VMs which can communicate with the hardware of the host (ex: Ethernet adapter to create more virtual adapters) Docker containers run in an isolated environment on top of the host's OS. Even if your host runs Windows OS, you can have Linux images running in containers with the help of Hyper-V, which automatically creates small VM to virtualize the system's base image, like Linux.

Docker Use

- Docker is computer software used for Virtualization to have multiple Operating systems running on the same host
- Docker is the client-server type of application which means we have clients who relay to the server
- Docker images are the "source code" for our containers; we use them to build

- Docker file has two types of registries
- 1.) public and 2) private registries
- Containers are the organizational units of Docker volume. In simple terms, an image is a template, and a container is a copy of that template. You can have multiple containers (copies) of the same image.

Docker Architecture



Docker Engine

Docker is the client-server type of application which means we have clients who relay to the server. So, the Docker daemon called: dockerd is the Docker engine which represents the server. The docker daemon and the clients can be run on the same or remote host, and they communicate through command line client binary, as well as a full RESTful API to interact with the daemon: dockerd.

Docker Images

Docker images are the "source code" for our containers; we use them to build containers. They can have software pre-installed which speeds up deployment. They are portable, and we can use existing images or build our own.

Docker Registries

Docker stores the images we build in registries. There are public and private registries. Docker company has public registry called Docker hub, where you can also store images privately. Docker hub has millions of images, which you can start using now.

Docker Containers

Containers are the organizational units and one of the Docker basics concepts. When we build an image and start running it; we are running in a container. The container analogy is used because of the portability of the software we have running in our container. We can move it, in other words, "ship" the software, modify, manage, create, or get rid of it, destroy it, just as cargo ships can do with real containers.

In simple terms, an image is a template, and a container is a copy of that template. You can have multiple containers (copies) of the same image.

Install Docker on Linux/Ubuntu

Step 1) To install Docker, we need to use the Docker team's DEB packages.

$ sudo apt-get install apt-transport-https ca-certificates curl software-properties-common

Step 2) Add the official Docker GPG key with the fingerprint.

$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

Step 3) Add the Docker APT repository.

$ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"

Step 4) After adding the GPG key, Update

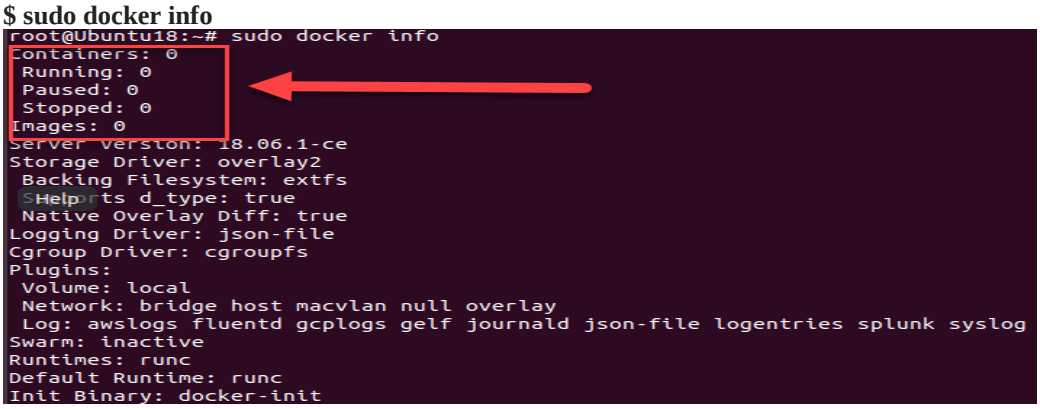
$ sudo apt-get update

Step 5) Install other packages,

$ sudo apt-get install docker-ce

Docker using basic Docker Commands

$ sudo docker info



$ **sudo docker pull <name>**
Telling docker to download the image, to pull it from the public registry, the latest version which is set by default.

$ sudo docker run -i -t alpine /bin/bash
To run the image as a container, we will use the following command.

$ sudo docker run -it ubuntu /bin/bash
Notice docker checks for the image locally, and if it's not there, the image is pulled from the image library automatically, and once again we have an interactive shell running. We can also name the containers as we run them.

$ **sudo docker run –-name our_container -it ubuntu /bin/bash**

We can also run container we previously created, without an interactive shell.

$ **sudo docker start container_name**
And stop the container writing docker stop container_name

$ sudo docker stop container_name
If we want to see all running containers, we just run

$ docker ps
And for all containers we add "- a"at the end of this same command, like this docker ps -a.

$ docker stats
You can also see which images we have downloaded locally and info about them.

$ sudo docker images
The command in the above Docker example displays the docker image with a tag which shows our image version, a distinctive image ID, when was created and image size.

Cheat Sheet:

Some Docker Commands

Below are the important Docker commands:

Command	Description
docker info	Information Command
docker pull	Download an image
docker run -i -t image_name /bin/bash	Run image as a container
docker start our_container	Start container
docker stop container_name	Stop container
docker ps	List of al running containers
docker stats	Container information
docker images	List of images downloaded
Docker Cleanup	Kill all running containers.

Install: Download and install

wget -qO- https://get.docker.com/ | sh

If it installed right, you should be able to run:

docker run hello-world

hello-world is name of docker container you want to run. If that container didn't exist locally, it would be downloaded from the hub.

Search: Search for images on hub website, or run

docker search <name>

from the command line or

docker search ubuntu

To see what images you have installed,

docker images

Remove images Remove an image by:

rmi -f <Image name>

rmi -f <image id>

Build image

1. Create docker file

2. Build image via docker build -t docker-Name.

    Builds new image that will be called docker-whale

    Looks for Docker file in current dir, hence the. at the end of the command.

Push images to docker hub

1. First find image id via docker images.

2. Tag your image via docker tag /: docker tag <tag number> <name>/dockertest:latest

3. Push to your repo via docker push / , i.e. docker push <name>/johnd

Pull images from docker hub docker

pull <name>/johnd