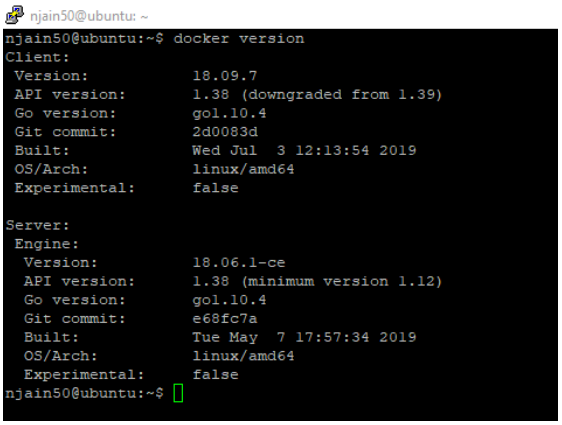
Docker concepts

Saturday, 3 August 2019 11:36 AM

https://www.youtube.com/watch?v=Gw2Jrid4SaQ

Docker version shows server by default as Linux.



Docker image would run in any environment.

When we talk about Docker, we are talking about entire ecosystem of Docker products which includes:

- Docker Client (CLI)
- Docker Server (daemon)
- Docker Machine
- Docker Hub/ Docker Registry
- Docker Compose

Docker Client:

- Tool we issue commands to reach out to docker daemon so it's a place where we run docker commands.

  For example: docker run hello-world

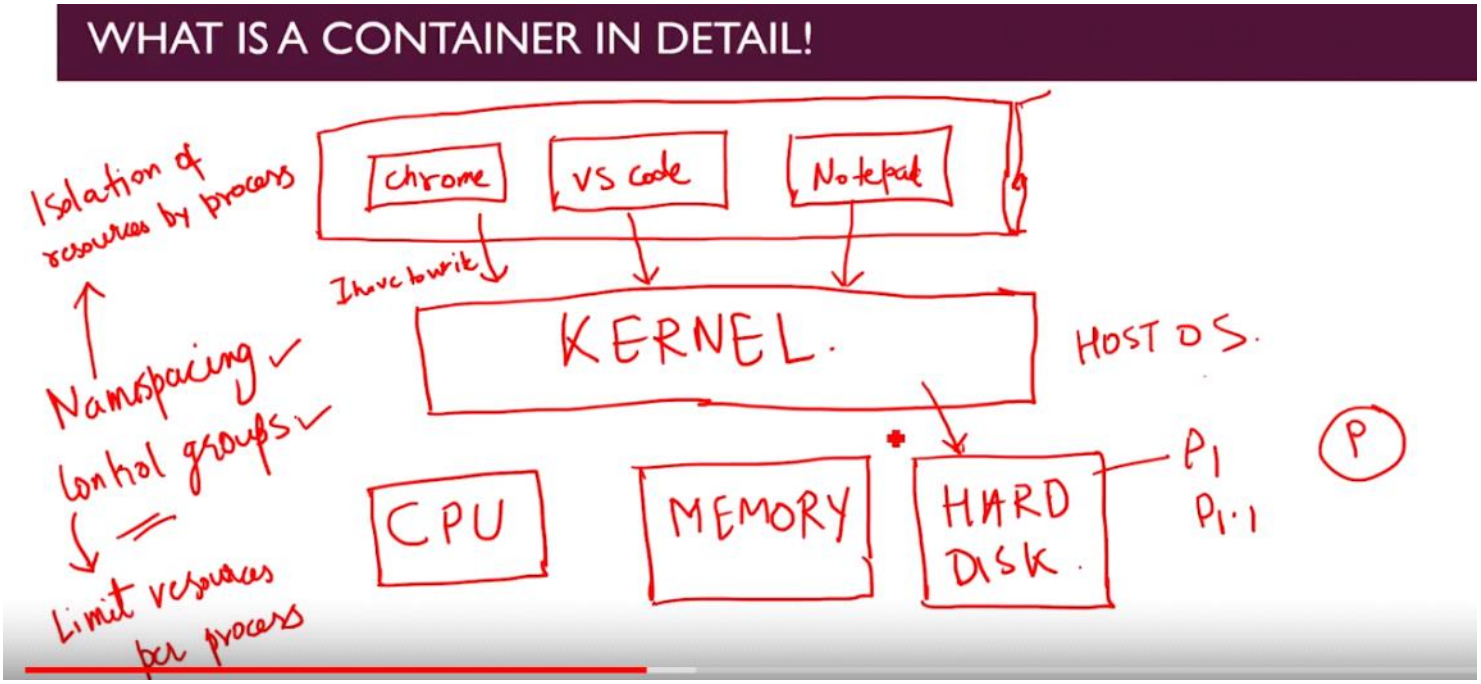
- Whenever we write command to Docker client it talks to server and get response.
- Docker server search for image locally (in local cache) and if retrieve from there.
- IF no image on local, it checks on docker hub or other configured repository.
- Download image from docker hub and store in local cache.
- Create container from that image.
- Start container.

Docker Server: It's a service that runs on host operating system .
This service is responsible for downloading, building, and running containers.

IMAGE: Analogy: it's like a seed which has all capabilities to turn into a tree.
So, it's like a software which contains all required data to create a container from it. No resources are assigned to it yet.

CONTAINER:

- Its and isolated environment .
- Part of hard disk Allocated to it.
- It have its own resources viz. CPU, memory, etc.
- When we allocate resources to image it turns to container.



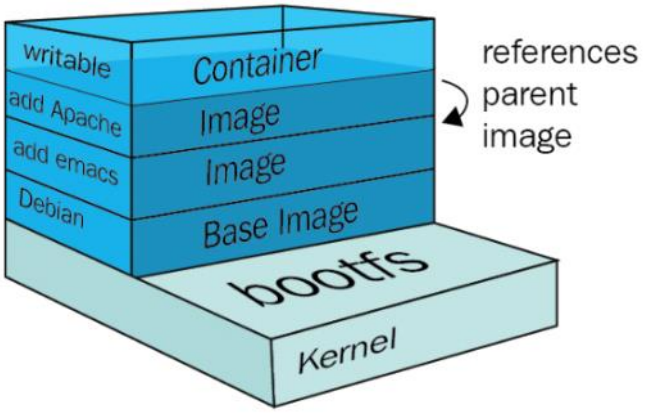
Screen clipping taken: 3/8/2019 3:16 PM

Kernel is heart of Operating System and has full rights on management of key resources - CPU ; MEMORY; HARD DISK.
So When different apps viz. chrome, VS code, notepad++ ask for resources kernel decides who will get it and in what order.
There are two key components which helps kernel to manage these resources:

1- Name spacing: This is isolation of resources by process
2- Control Groups: Limit resources per process.

If Chrome requires program P1 and VS Code requires P1.1 then only one app would run here.

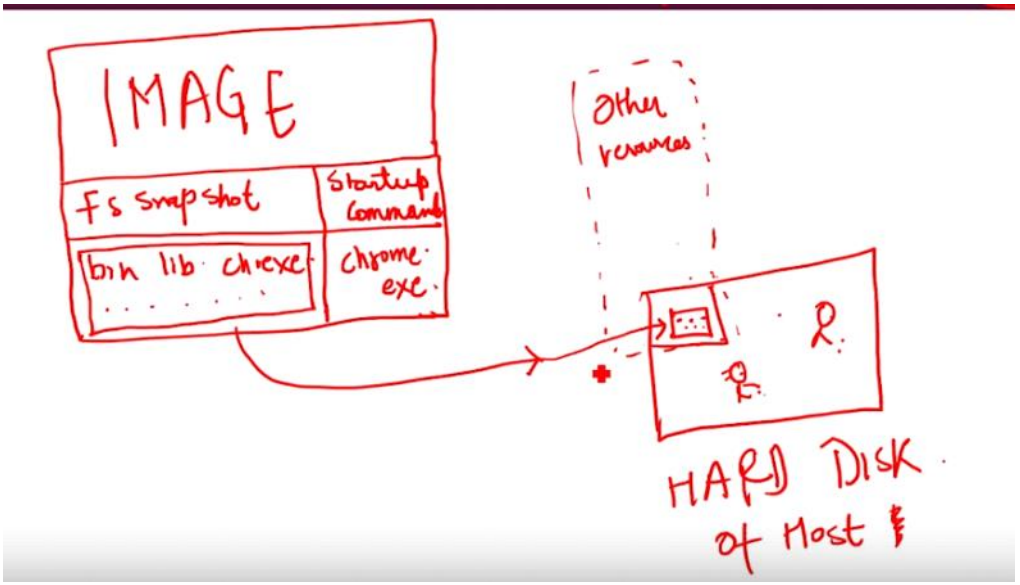
Now Container is process which will have its own hard disk, memory and CPU which is part of hosted OS. Docker helps in facil itating this process called container.



The container filesystem, used for every Docker image, is represented as a list of read-only layers stacked on top of each other. These layers eventually form a base root filesystem for a container. In order to make it happen, different storage drivers are bein g used. All the changes to the filesystem of a running container are done to the top level image layer of a container. This layer is call ed a Container layer

Image:

Chrome is set of file stored somewhere in hard disk hence its basically file system snapshot.
So image just contains file system snapshot and start-up command. When docker users this image to convert to container it just replaces file system snapshot into part of OS hard disk allocated to Container. Also Kernel allocates other resources require d by container. Finally it will run start-up command which default comes with image only. But we can overwrite this start-up command when launching container.



An image consists of a number of layers that are combined into a single virtual filesystem accessible for Docker applications

A Docker image usually consists of several layers, stacked one on top of the other. The top layer has read-write permissions, and all the remaining layers have read-only permissions

Docker Registry

A registry is a stateless, highly scalable server-side application which you can use to store and download Docker images

Docker registry is an open source project, under the permissive Apache license

Docker supports several types of docker registry:

- Public registry
- Private registry

many vendors add their own public registries to the Docker configuration at installation time. For example, Red Hat has its own *proven and blessed* public Docker registry which you can use to pull Docker images and to build containers.

Docker configuration

Docker daemon configuration is managed by the Docker configuration file (/etc/docker/daemon.json) and Docker daemon startup options are usually controlled by the systemd unit named Docker. On Red Hat-based operating systems, some configuration options are available at /etc/sysconfig/docker and/etc/sysconfig/docker-storage. Modification of the mentioned file will allow you to change Docker parameters such as the UNIX socket path, listen on TCP sockets, registry configuration, storage backends, and so on