- Docker!

  - ↳ VMs virtualise hardware
  - ↳ Containers virtualizes Operating system

- How containers are made?

— three major building blocks for container formation:

  - ↳ Linux Namespaces
  - ↳ Control groups
  - ↳ Layers – Union filesystem & CoW

We stack these to create a container
↳ this is internally an operating system

Container

Pid 1	↔	Process id
root filesystem	↔	filesystem (mount
eth0	↔	Network
Process inside	↔	Inter Process Comm.
Our hostname	↔	Unix timesharing system
		Users

Linux

Linux Namespaces
Pid, net, mnt, ipc, uts, user

Control groups
CPU, RAM, grouping, resources

Layers
Union file system & Cow

Linux namespaces are used for isolation of containers

Since these containers run some processes, resources like RAM, CPU etc need to be managed, it is done by Cgroups.

- Unified view
- Shared stack
- Each container has a writable layer

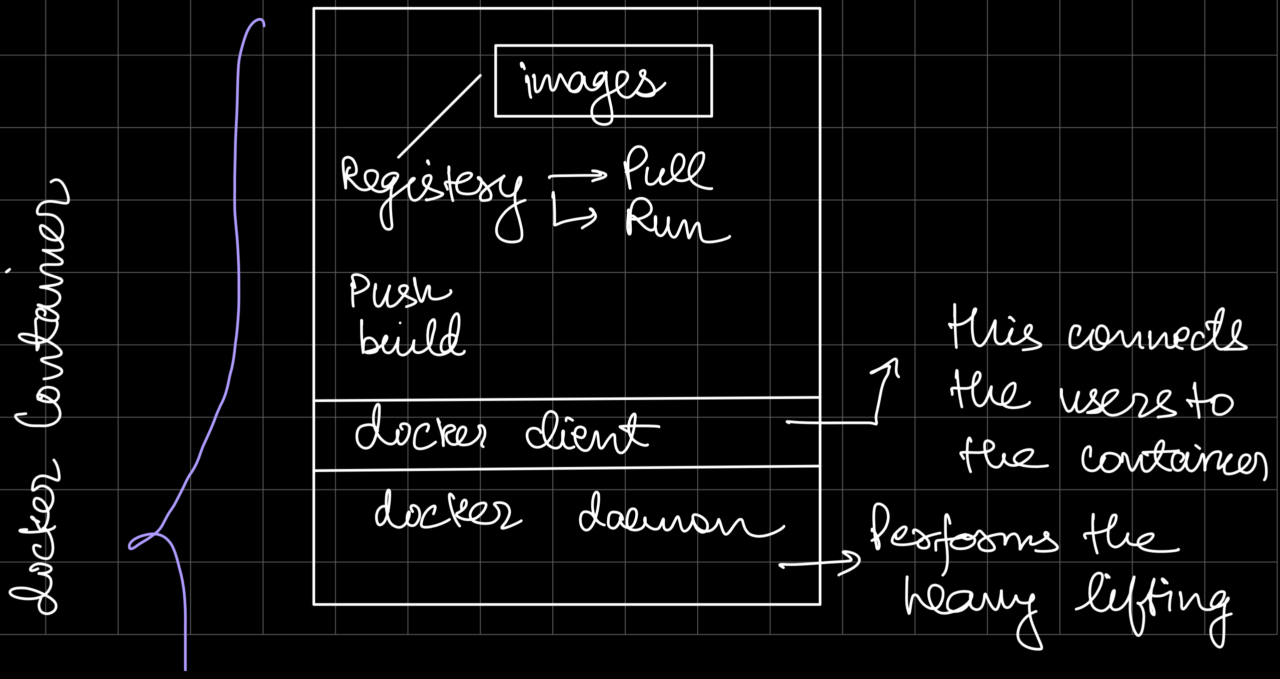
- As we all know that a "running image" is called a container.
- An image is built in multiple layers.

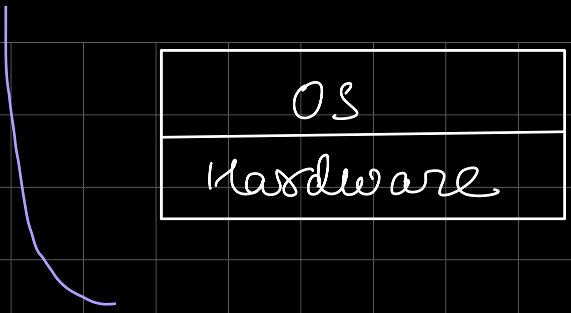
- Every image will have a writable layer, so that the image still works, in case it is running on other containers simultaneously.

"Containers are isolated but a lot is needed to be done in terms of security"

● "Docker World!":-

Docker is a containerization tool.





of managing docker "objects"

- docker objects are the images, containers etc.

- The images that are used to run containers reside on an online regestery
Eg:- "Dockerhub"

Some docker commands:

- Run : Used to run an image to create a contanier
- Pull : Used to pull an image from registery
- Push : Push an image to the regestery
- Build : Uses dockerfile to build an image.

- What happens when you write "docker run {image}"

  → docker client talks to the "daemon".

— the daemon doesn't know how to start a container

  → daemon forwards the call to "Containerd."

- Containerd is a long running process & it manages container lifecycle.

- Even containerd does not know how to spin up a container.

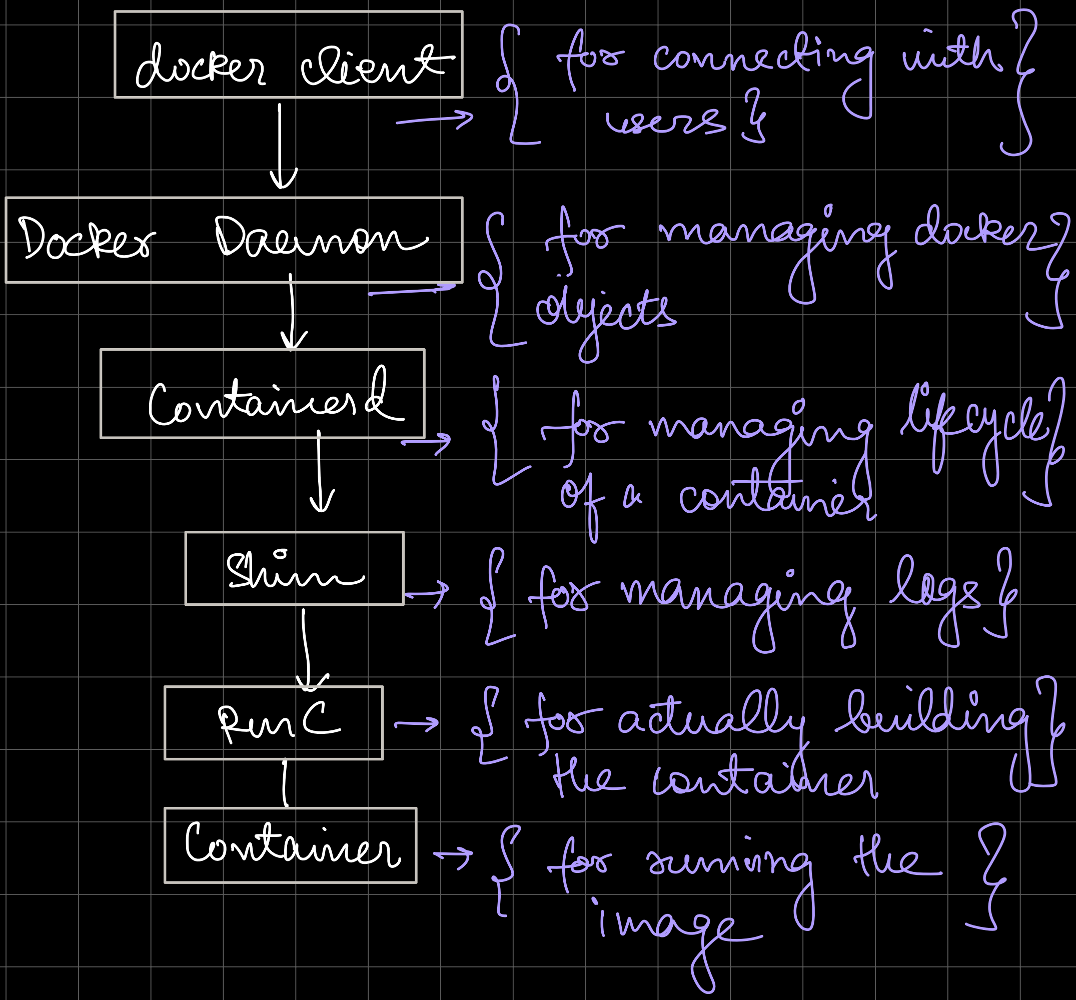
  → Containerd forwards the call to "RunC"

- Linux Namespaces, cgroups & other stuff is managed by RunC.

- Shim is present b/w containerd, that is responsible for managing logs & communi-cation b/w user & container.

- ** for each container, you will have seperate shim process.

- ** Containerd is a seperate process, that is unique & manages all the container lifecycle.


docker client
{ for connecting with users }
Docker Daemon
{ for managing docker objects }
Containerd
{ for managing lifecycle of a container }
Shim
{ for managing logs }
RunC
{ for actually building the container }
Container
{ for running the image }