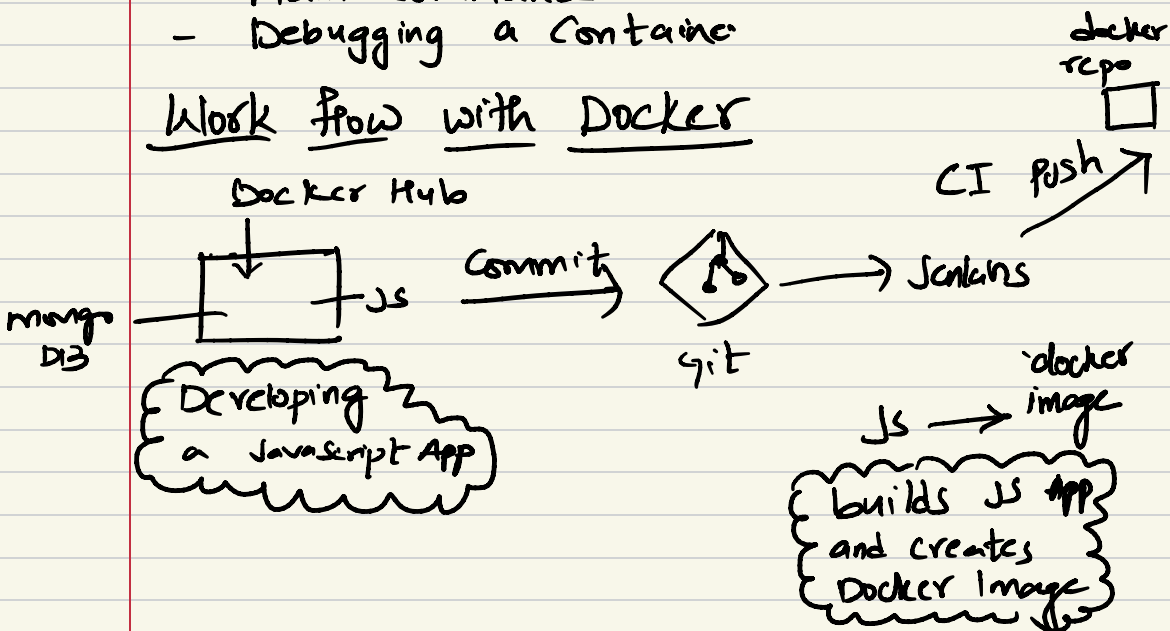
(1) Introduction to Docker

- what is docker?
- what is container?
- Docker vs VM
- Main Commands
- Debugging a Container

Work flow with Docker



- Volumes - Persisting Data
- Developing with containers
- Docker Compose - Running multiple services
- Docker File - Building own Docker image
- Private Docker Repository (AWS)
- Deploying the containerized App
- volumes Demo

What is a Container?

- A way to package application with all the necessary dependencies and configuration
- Portible artifact, easily shared and moved around.
- Makes development and deployment more efficient

Where do Containers live?

- Container Repository
- Private repositories
- Public repository for docker

Application Development

Before Containers

- Installation process different on each OS environment.
- many steps where something could go wrong.

After Containers

- own isolated environment
- packaged with all needed configuration
- one command to install the app.

Application Deployment

Before Containers

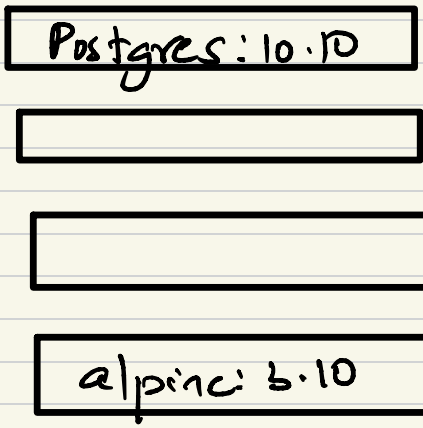
- Configuration on the server needed
- dependency version conflict
- textual guide of deployment
- misunderstandings

After Containers

- Developers and Operations work together to package the application in a Container.
- No environmental Configuration needed on server - (except docker runtime)

What is a Container (Technically)

- Layers of stacked images on top of each other
- The base of most of the images is Linux Base Image because small in size alpine 3.10
- Application image top.



Layer - application image

Layer - linux base image

The benefit with layers is if we have to download a newer version of software, the layers that are same between 2 versions would not be downloaded again. Only different layers will get downloaded.

Docker Image

- The actual package
- Artifact, that can be moved around

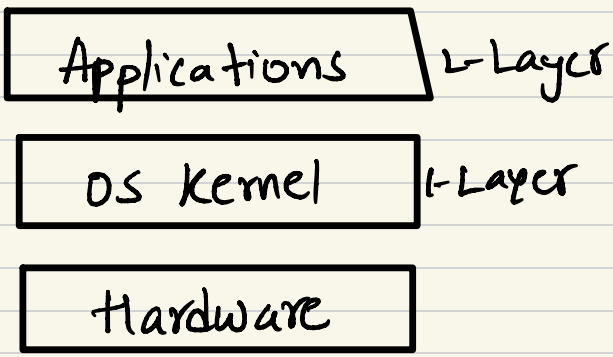
Not running

Docker Container

- actually start the application
- Container environment created.

running

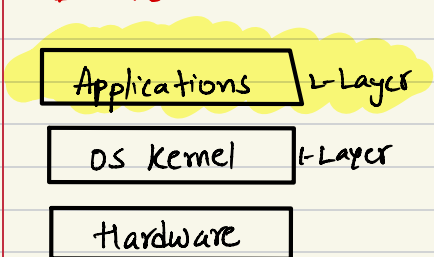
Docker vs Virtual Machine



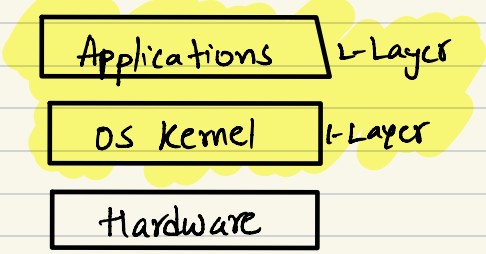
Operating systems have 2 layers

Docker and VM both are virtualization tools.

Docker



VM



Size of docker images are much smaller

Speed of docker images are much faster

* install docker on your computer.

Main Docker Commands:

docker pull
docker run
docker start
docker stop
docker ps
docker exec -it
docker logs

Difference between Image and Container

- CONTAINER is a running environment for IMAGE
- application image: postgres, redis, mongoDB, environment config
- port binded: talk to app inside container

PORT 5000

has virtual File system

docker pull redis

docker run redis

docker ps (status of all containers)

docker run -d redis

starts new container with a command

get the id and run in detach mode

To stop a Container

docker stop <id>

To get the history of lists running and stopped Containers

docker ps -a

`docker run redis:4.0`

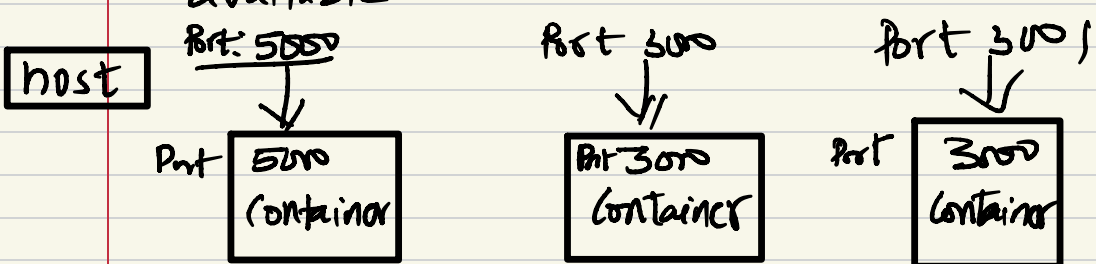
— pulls image and starts Container

How to use any Container that we started

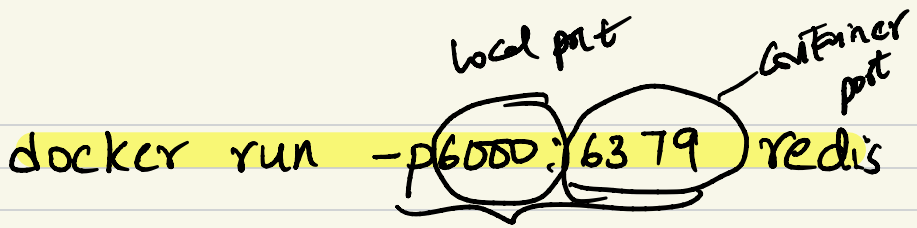
In the logs of container you can see information, running mode and standalone port

Container port vs Host port

- Multiple Containers can run on your host machine
- your laptop has only certain ports available



- Conflict when same port on host machine



binding container port to the host mode

```
docker images
```

Debugging Containers

```
docker logs <name> or <id>
```

```
docker run -d -p6001:6379 --name redis-older redis:4.0
```

→ To name a container

```
docker exec -it <container_id> /bin/bash
```

-it → interactive

if we want to get inside the container and edit the stuff as a root user.

docker run } Pull & run or run

docker stop } stop

docker start } restart

Demo Project