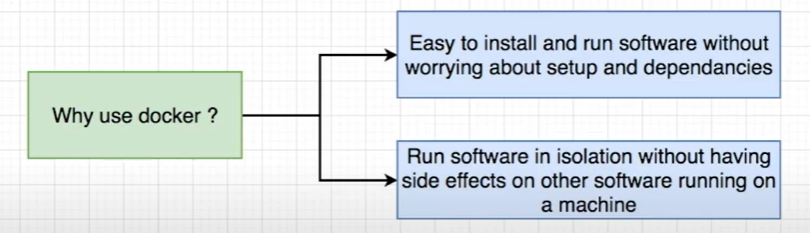
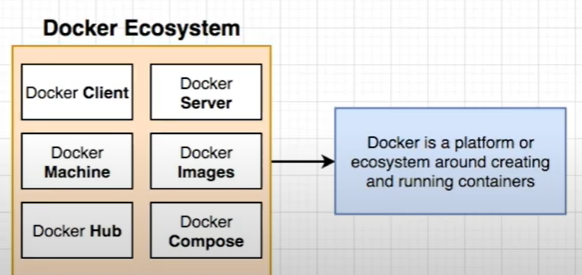
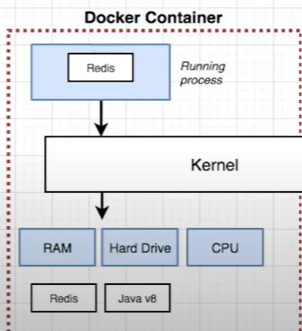
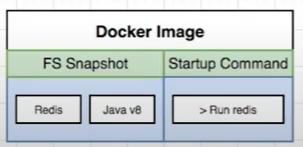
**Docker**

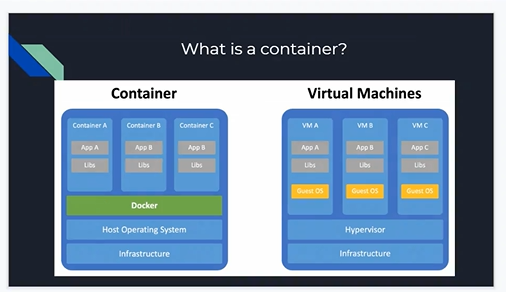
Its a tool designed to minimize the **troubleshooting** errors and **isolation** while implementing







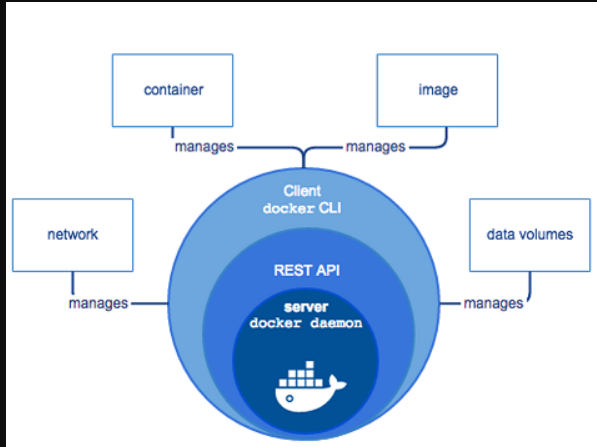
* It’s a tool designed to create, deploy, and to run apps in a container
* docker -d -p 1010:80 nginix:alpine (to create a quick container and to create web server on port 1010)
* Docker -rm -fv id (to remove any container)
* Docker rmi -f id (to remove images)
* Du -sch\* (to see the current directory size)
* sudo usermod -aG docker imagename (to execute docker commands as admin)
* Stop all running containers: docker stop $(docker ps -a -q)
* Delete all stopped containers: docker rm $(docker ps -a -q)
* .dockerignore (similar to git ignore which ignores all the files and folders in that file)
* Sudo systemctl enable docker (which enables docker upon restart of the machine)



Container

* A process inside a sandbox is called container
* Docker file : its a plane text file where we can see instructions inorder to create an image.

Architecture



* Permissions in the image layer is – Read only
* Docker Client and Docker Daemon communicate with each other by –Rest API
* Permissions in the container layer is –Read and Write
* A Machine where docker runs – Docker host

<https://docs.docker.com/get-started/overview/>  
  
 **Installation in ubuntu**

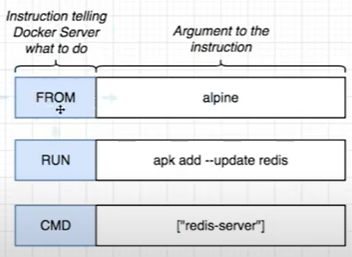
* Sudo apt-get update
* sudo apt-get install \
* ca-certificates \
* curl \
* gnupg \
* Lsb-release
* sudo mkdir -p /etc/apt/keyrings
* curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
* echo \
* "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \
* $(lsb\_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
* Sudo apt-get update
* sudo chmod a+r /etc/apt/keyrings/docker.gpg
* sudo apt-get install docker-ce docker-ce-cli containerd.io docker-compose-plugin
* sudo docker run hello-world

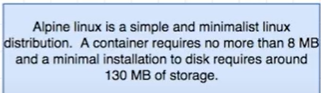
**Creating containers**

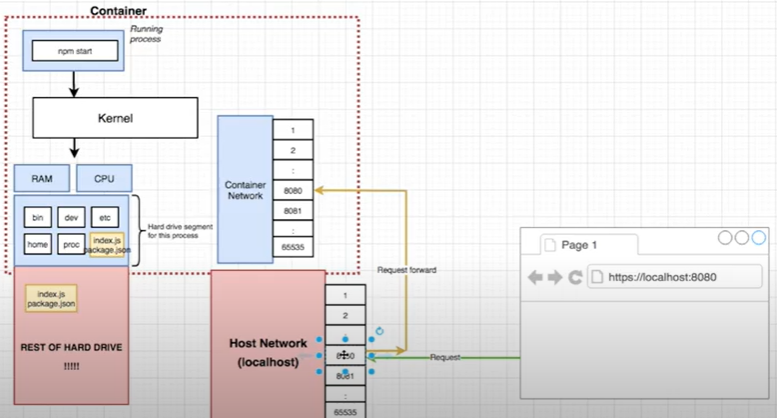
* **docker run -d nginx** (to create an run the container in the image nginix)
* Docker images (to check the list of images in docker)
* Docker ps (to check the list of containers in a docker)
* Docker ps –all (to check the list of all containers in a docker)
* Docker ps -a (to check the list of all the containers in a docker and to display)
* Docker rename pervious name newname (previous name can be obtained by docker ps)
* Docker log container id (if we want to know what is going on the bg of the container)
* Docker stop container id (we can stop the container in sequel process 10 sec delay)
* Docker kill container id (we can kill the container instantly)
* Cp filename newfilename (copy command)
* Docker stats containername or id ( to see the task manager of the running container)
* Docker inspect containername (allows the user to take a look at the configurations of the container(json file))
* Docker rm id or name (to remove stopped containers)
* Docker run -d –name somename -e “test=1234” imagename (Writing env variable outside the container )
* Docker exec -it container\_id command (we can execute commands in a running container (to go inside a container exp : centOs or ubuntu))
* Docker rm -f id or name (to forcely stop and remove the container)
* Port mapping : docker run -d -p 9090:80 nginx:latest ( to run and create a container of port 9090 which maps to 80 in the image Nginix)
* Docker cp conname:filepath . (copying files from the container to the hostmachine current directory)
* Docker cp filename conname:destinationpath (copying file from host to container at the dest path)

**Memory limitation -m**

* Docker run -d –name conname -m “sizemb” imagename:tag (to limit the memory of a container using)
* Docker stats containername (to check the memory status taskmanager in windows)
  + - Dockerfile





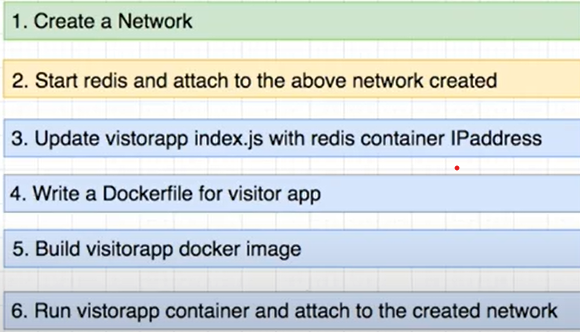


* + - Create a folder and a sample dockerfile (**from** which image **run** to run commands to start service **cmd** give the command or the service **workdir & copy, ENV, LABEL, USER** )
    - CMD command in docker file allows the container to be live and running
    - Docker build (command allows you to create images from dockerfile)
    - Docker build –t ubuntu\_apache:v1 . (it creates image from the dockerfile with in directory of **Dockerfile** name present and along with the tag name)
    - Docker build –tag ubuntu\_apache:v1 -f dockerfilename . ()
    - Tagging : docker build -t userid/reponame:versiontag (in this way we can give the name as user required)
    - Docker run -d -p 9090:80 -n con\_name ubuntu\_apache (it allows us to run a container with a con\_name and it maps the port as well)
    - If its a VM in any cloud need to allow the port number in the inbound

**Docker Networking**

* Docker network create <netname> (to create a network)
* Docker network inspect<netname> (to get more network info)
* Docker network ls (it lists all the networks in the docker)
* Docker network rm <netname> (to remove a network from docker)
* Docker run -it –name <conname> –network <netname> image (Creating container with the network)
* Docker network connect <netname> <conname> (adding container to the network)
* Docker network disconnect <netname> <conname>(disconnecting container from the network)

**How to run an application in docker**

****

**Docker Compose**

****

* It supports **yaml** format and primarily uses Yaml
* Expose : it makes the port to be available with in the docker network
* Docker-compose up –build (to build and to run the compose yml file)
* docker -compose down (to stop the running containers)
* Docker-compose ps (it shows all current running containers)

**Docker Volume**

* Linking a file or folder from the local machine to that of the docker container is called docker volume
* Primary goal of the docker volume is to have the synchronous between folder and files in local & docker
* Up on adding volume along with the paths then make some changes in the files and we can simple restart the running container itself, changes would get replicated.



**SSL using Docker**

* Open the online ssl generator and paste the command so that it will have the .cert and .key file
* **openssl req -new -newkey rsa:2048 -sha256 -days 365 -nodes -x509 -keyout docker.key -out docker.crt**
* Copy the keys to the dockerfile and then Install the required dependencies in the docker file
* Create an other conf file in it as it contains all the configurations in it (virtual host) and copy the SSL certificate and key in that conf file
* Copy the newly created conf file to the docker file
* Now build the custom image and run the container on port 443 and then test it.

**Dangling images**

* When we re-build a custom image without changing the **name** and **tag** of the old image becomes a dangling image with name **<none>** and tag as **<none>**
* To remove dangling images (docker rmi $(docker images -f dangling=true -q)

**Multi-stage -images**

* The main purpose of the multi stage is to reduce the image size (by using the previous image)