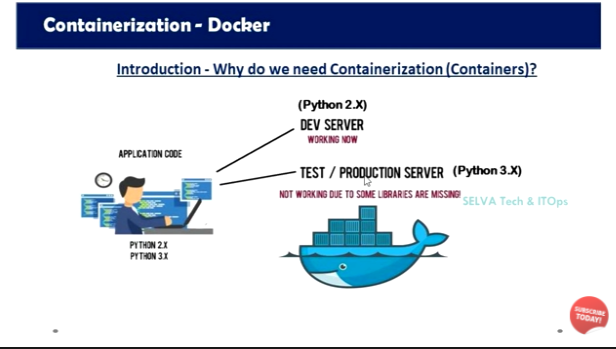
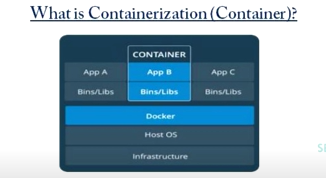
**Containerization:**

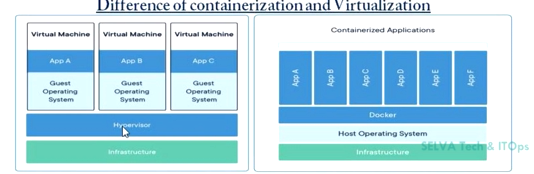


Example : The developer build the code in python 2.X and then the code was given to Dev Server python 2.X there also the code was working fine. Then the code is move to Test/ Production Server here the python version 3.X the same code was not working because of python version changed.Thats why the container is take place to resolve this issue. The code is make as a template with all libraries, binaries, compatibility. is called containerization.(BUILD, SHIP, RUN)



Containerization is a lightweight virtualization technology alternative to hypervisor virtualization, Any application can be bundled in a container can run without any worries about dependencies, libraries, and binaries. Because container create the isolated environment with all the required dependencies, libraries and binaries to run your application with out any issue. Hence you can build the package ship the application to any environment and run it.

So containers are designed to run on physical servers, Virtual machines and any cloud instances.

**Difference of containers and Virtualization:** 

Virtualization technology that allows us to have multiple operating system to share a single hardware processor.

Containerization is an application-specific virtualization, because it provides application with environments in the form containers to run on, which can be deployed and run anywhere without a dedicated virtual machine with Operating system for each application.

Also Container was designed to solve modern problems of application management issues. So it’s not a replacement of virtualization, but it’s complementary to it.

**Advantages of Containers:**

* Containers are isolated, doesn’t require operating system and it shares a host kernel. So containers run on the same server and use the same resources, they do not interact with each other because its isolated. If one application crashes, other containers with the same application will keep running without any issues.
* Its a Portable and light weight operating system and it contains only the required binaries, dependencies and libraries to run the application, so it can be move anywhere easily and can run without worrying about compatibility, dependencies kind of issue.
* Faster and Resources efficiency - its very fast to boot, because containers are lightweight and start in less than a second since they do not require an operating system to boot.
* Resource efficiency since containers do not require a separate operating system, they use less resources.
* Improve Scalability and lowers costs - By allowing more containers in the environment without the need for more servers, containerization increases scalability anywhere from 10 to 100 times that of traditional VM environment.

**DOCKER:**

Docker is a open-source platform tool designed to manage the containers, which allow us to build the application in a container with required libraries, binaries, dependencies to run the application, ship the container and run anywhere.

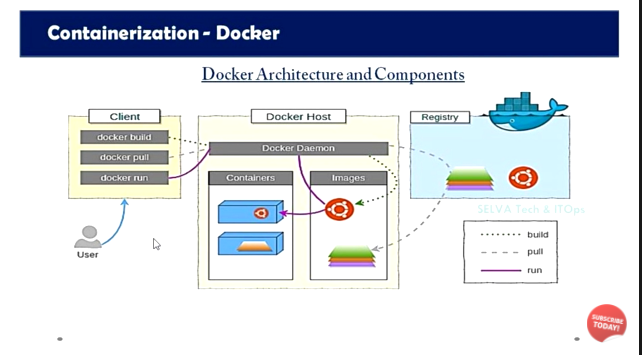
**Introduction to Docker:**

* Portability - An applications can be bundled in to a single unit and same unit can be deployed to various environments such as dev server, testing server and production server without making any changes to the containers.
* Light Weight - Docker containers are pretty lightweight so it provides a smaller footprint of the operating system via containers.
* Fast Delivery and Scalable - Since Docker containers are very pretty lightweight, so they can be deployed faster and they are very easily scalable.
* Docker used for Continuous Deployment and Testing, So with the help of containers, it becomes easier for teams across different units, such as development, QA and Operation to work seamlessly across applications.
* Docker also provides you, ability to run multiple isolated OS on single host.
* Resource Optimization - Docker enable you to utilize the maximum resources and reduce the resource wastages of your hardware.

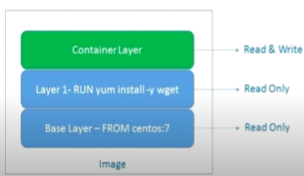
Docker File:

Docker file is nothing but its like a normal file.This file includes the set of instruction to perform an particular operat

**Docker Architecture and Components:**

****

**Docker Images:**



* Docker images are just a templates of a docker containers and it is very similar to snapshot image with smaller in size. Single Docker image can be used to create multiple containers for different environment like development, UAT and Production.
* Docker images are very lightweight, small, ans fast to deploy the containers.
* Docker images are consists of many layers with unique image id from base images. Each layer s may have some changes committed top of a existing layers.
* Docker images are read only layer of docker containers and docker containers are read write layer of docker images.

**Basic Docker Commands:**

* docker -v - This command is used to check the version for docker
* docker version - This command is used to check the version with all details.
* docker (press enter) - This command to check all the command options (docker run)
* docker info - This command is used to display the information of your docker.
* docker system prune - This command is used to remove all unused containers.
* docker system --help - This command is used to check all command options.
* docker stats - This command is used to check the current status of Docker (like PIDS, Memory, CPU, I/O, Net I/O, Container ID, Name )
* docker stats (container name) - This command is used to check system stats on particular container.
* docker top (container name) - This command is used to check the process.
* docker system events - This command is used to check the events.
* docker images - This command is used to list out the images in local docker repository.
* docker search (search image content) - This command is used to search the image in public registry.
* docker create (image name) - This command is used to create the container.
* docker ps - This command is used to check the process of running ( Container id, Image, Command, Created, Status, Ports, Names)
* docker ps -a -This command is used to check all the containers.
* docker run (image name) - This command is used run the image.
* docker pull (image name) - This command is used to pull the image from the registry.
* docker pull (image name) :tag name - This command is used to pull the image in repository with the particular tag name
* docker save (image id) > (user specified name).tar- This command is used to backup the image on local machine.
* docker load -I ( backed up file name) - This command is used to restore the backup docker image
* docker commit (container id) - This command is used to commit the changes in the image.
* docker commit (container id) (user specified name) - This command is used to create the tag which we commit the new container.
* docker system df - This command is used to check the space used in local repository.
* docker login - This command is used to login the docker hub
* docker push (Image name with Tag) - This command is used to push the image to docker register,
* docker run --name (user specified name) (image name) - This command is used to run an image with user specified container name, This is used to run the application in foreground only.
* docker run -d -it --name (user specified name) (image name) - This command is used to run an image with user specified container name, This is used to run the application in background.
* docker inspect (image id) - This command is used to check the full details of containers (like - hostname, Domain name, Network port, etc)
* docker history (image name) - This command is used to check complete history and commits for the image file.
* docker run -d -it --name (user specified name) -p 12:12 (network port) (container name) - this command is used to run the container with port forwarding to access over the internet
* docker run -d -it --name (user specified name) -P (network port) (container name) - This command is used assign the port randomly assign by docker to access the over the internet.
* docker exec -it (image name) /bin/sh - This command is used to login the container which is running in the docker.Log in to the terminal
* Ctrl D - This is used to exit from the running container.
* docker exec (image name) (command which we can use in running container image) - This command is used to perform any command without login the running container.
* docker cp (source) (container id):/designation folder - This command is used to copy an file from one location to another location,
* docker stop (image name) - This command is used to stop the container.
* docker start (image name) - This command is used to start the stopped container.
* docker rm (container id) - This command is used to remove the container.
* docker rm -f (container id) - This command is used to force remove the running container.
* docker rmi (image name) - This command is used to remove the image.
* docker logs (container name) - This command is used to check the logs in container.
* docker build . - This command is used to build the docker file present in current directory
* docker build -f /…directory - This command is used to build the docker file
* docker build -t myweb:1.1 . - This command is used to build an docker image with tag name.
* ip a - This command is used to check the docker host ip address.
* docker network create (network name) - This command is used to create an network
* docker network ls - This command is used to list the network in docker
* docker network inspect (name of the network) - This command shows the network details.

Docker file creation

FROM - From instruction used to specify the valid docker image name. So specified docker image will be downloaded from docker hub registry if it is not exists locally.

Syntax: FROM ubuntu:latest

LABEL - Label instruction used to specify the metadata information to an image. A LABLE is akey -value pair.

Syntax: LABLE “APPLICATION\_ENVIRONMENYT”=”Development”

RUN - This instruction is used to executes any commands on the top of the current image and this will create a new layer, Run instruction has two forms shell form and executable form.

Syntax: RUN yum update -y

CMD - This command is used to set a command to be executed when running container. There must be only one CMD in a docker file if more than one CMD is listed, Only last CMD takes effect. It has two forms shell and executable form.

Syntax: CMD ping google.com

ADD - This instruction is used to copy files and remote URLS files to the destination (docker container) with In the file system of the docker images. Add instruction also has two forms - shell form and executable form.

Syntax: ADD/root/testfile/data/

COPY - This instruction is used to copy the files directories and remote files to the destination within the file system of the docker images. COPY instruction also has has two form shell forms and executable form

Syntax: COPY/root/testfile/data/

EXPOSE - This instruction is used to inform about the network ports that the containers listen on runtime. Docker uses the information to interconnect conatiners using links and to set up port redirection on the host system.

Syntax: EXPOSE 80 443

EXPOSE 80/tcp 8080/udp

ENTRYPOINT - This instruction is used to configure and run a containers as an executable ENTRYPOINT instruction also has two forms shell and Executable form

Syntax: ping google.com

VOLUME - This instruction is used to create or mount a volume to the docker containers from the docker host file system.

Syntax: VOLUME/data

USER - This instruction is used to set the username, group name

Syntax: USER webname.

WORKDIR - This command is used to set the working directory.(land directory)

Syntax: WORKDIR/app/

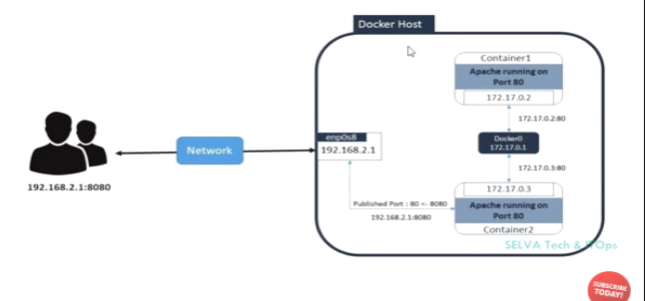
ENV - This instruction is used to set the environment variables with key and value. But the variables will set only during the image build not on the container.

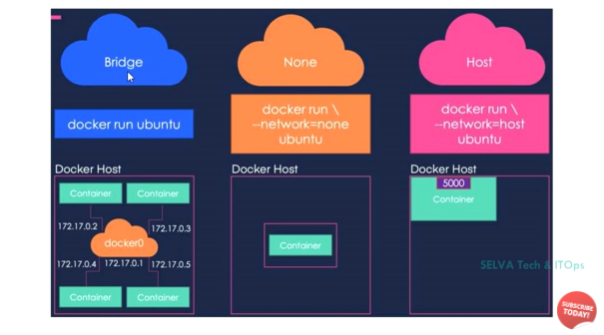
Syntax: ENV APP\_DIR/data/

ARG - This instruction is used to set the environment variables with key and value. But this variable will set only during the image not on the container.

Syntax: ARG TMP\_NAME mycustom\_image

Docker Network:





**Bridge** - This network is default while creating the containers.By assign the port forwarding to the conatiners we can access the network if not assign the port forward not able to access over network we can access only inside the container

docker run -d -it --name ABC -p 80:80 webserver

**None** - This network not able to access over network and inside the and inside the host containers.While creating this containers only login in the container and perform operations.

docker run --network=none -d -it --name ABD webserver

**Host** - This network is used to access only I outside the docker host

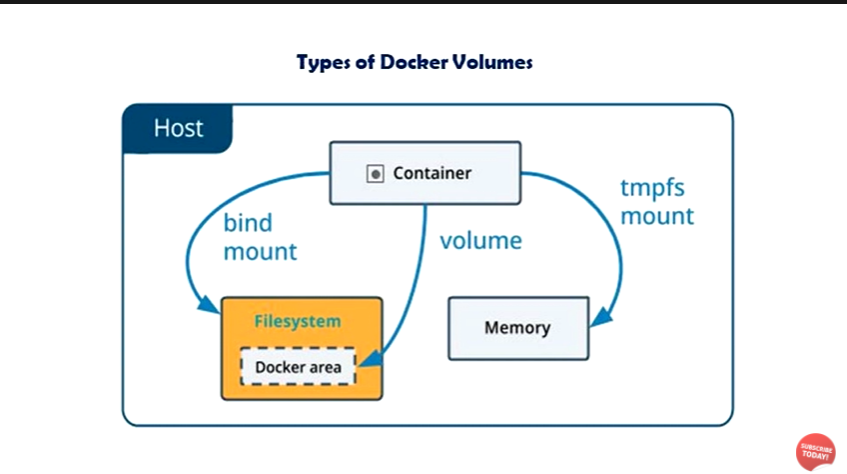
docker run --network=host -d -it --name ABD webserver

docker run -d -it --network=network name -- name ABC -p 80:80 webserver

Network related command:

* docker network create (network name) - This command is used to create an network
* docker network ls - This command is used to list the network in docker
* docker network inspect (name of the network) - This command shows the network details.
* docker network rm network name - This command is used to remove the created network
* docker network info - This command is used to show the details about the network
* docker network prune - This command is used to remove the network atleast one used.

DOCKER VOLUME:



In docker three types of volume is there

**Bind mound** - this volume can be used in any directory in the file system to store the data.

docker run -dit --network=bridge --name webserver -v /data/mnt:/azhar -P name of the container .

**General Volum**e - This volume is created by user specified and this volume is mounts in var/lib/docker after terminate the instance the user stored data is available.

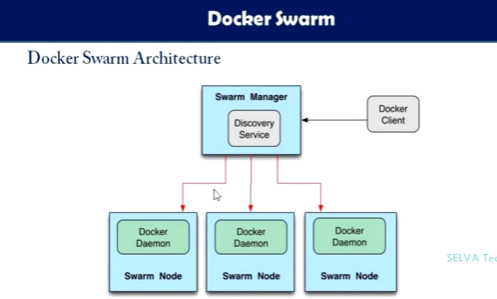
docker run -dit --network=Bridge --name webserver --mount source=Volumename,destination=container location/xxx -P name of the container .

**tmpfs mount** - This volume is default volume to run an application in container after remove the container data will delete automatically.

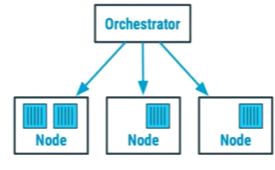
**Docker Swarm:**

A docker Swarm is a group of either physical or virtual machines that are running the docker application and that have been configured to join together in a cluster. Once a group of machine have been clustered together, you can still run the Docker commands that you are used to, but they will now be carried out by the machine in your cluster, The activities of the cluster are controlled by a swarm manager, and the machines that have joined the cluster are referred to as nodes.

**Docker Swarm Architecture:**



**Orchestrator:**



Docker swarm is one of Container Orchestration tool that allows to manage multiple containers deployed across multiple host machines.

**Container Orchestration Tools:**

* OPENSHIFT
* RANCHER
* DOCKER SWARM
* AWS ECS
* MESOS
* KUBERNETES etc…..

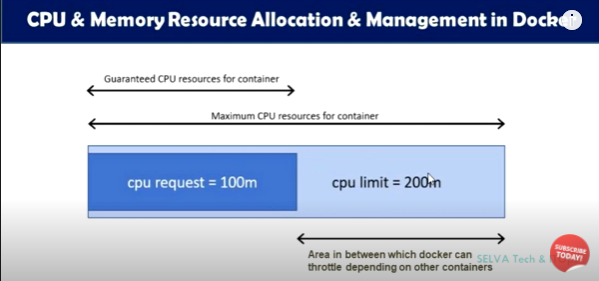
**Advantages of Docker Swarm:**

* Orchestration
* High Availability
* Scaling
* Multi - host networking
* Service discovery & Load Balancing
* Rolling Updates

Swarm commands:

* swarm init --advertise-addr (master ip) - This command is used to initialize the master in the swarm.
* docker node ls - This command is used to check the available list of nodes in Swarm.Its only works on Manager node.
* docker node promote - This command is used to promote the worker as a manager.
* docker node ps - This command is used to list the running nodes in the cluster.
* docker node demote - This command is used to demote the manager as a worker
* docker node rm - This command is used to remove the node from the cluster.
* docker node inspect - This command is used to inspect the node.
* docker update - This command is used to update a node.(avilability, add list, rm list, role, lables)
* docker service ls - This command is used to perform specific operation in the cluster
* docker service create --replicas 2 -p 80:80 web1 httpd - This command function is it create two replicas named with web1 with the image httpd with port forwarding on 80.Master creates replicas in master as well worker node but depending up on the availability.
* docker service ps abc - This command is used to check the services run in with the name of abc
* docker service create --mode global -p 81:80 --name web1 httpd - This command function is create a replicas in all the nodes in the cluster named with web1 with the image httpd with port forwarding.
* docker service scale web1=5 - This command is used scale up the service of web to 5 replicas and down also decrease the no of service.
* docker service update --image ABC/1.1 service name ABC - This command is used to update the service with the latest image ABC/1.1
* docker node update --availability active docker - node name - this command is active the node to update if the node is paused.
* docker node update --availability pause docker - nod name .

**Docker Memory and cpu Resource allocation:**



Commands related for Memory and CPU Allocation:

**Memory Reservation:**

* docker run --memory-reservation=126m -m 512m -d --name Web httpd - this command is used to run an image httpd with the container name web with the allocated memory minimum 128m limit 256m.

**CPU**

* docker run --cpus=1.5 -d --name web httpd - this command is used to allocate the cpu core here we alloted 1.5 cores
* docker run --cpu-shares=1000 --cpu=1 -d --name web httpd - this command is used to allocate the share in the mentioned 1 core limit.

Syntax to create allocation of cpu and memory while creating docker compose:

deploy :

resources :

limits :

cpus : “0.50”

memory : 1024M

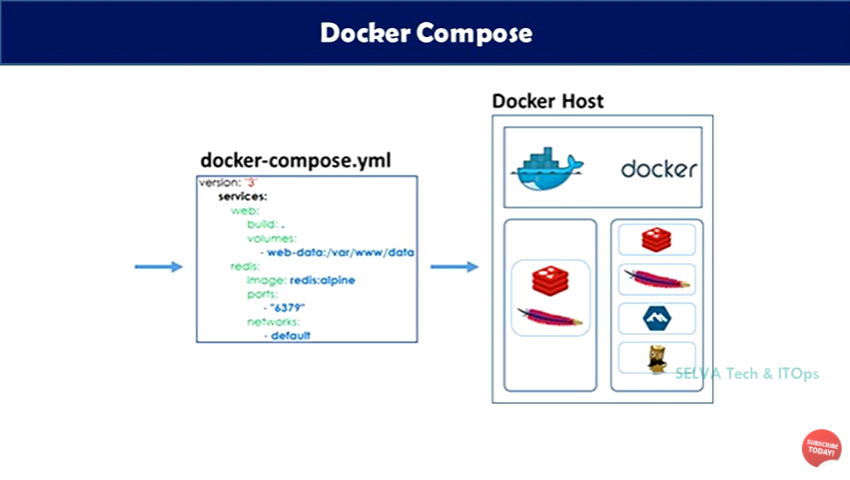
reservation :

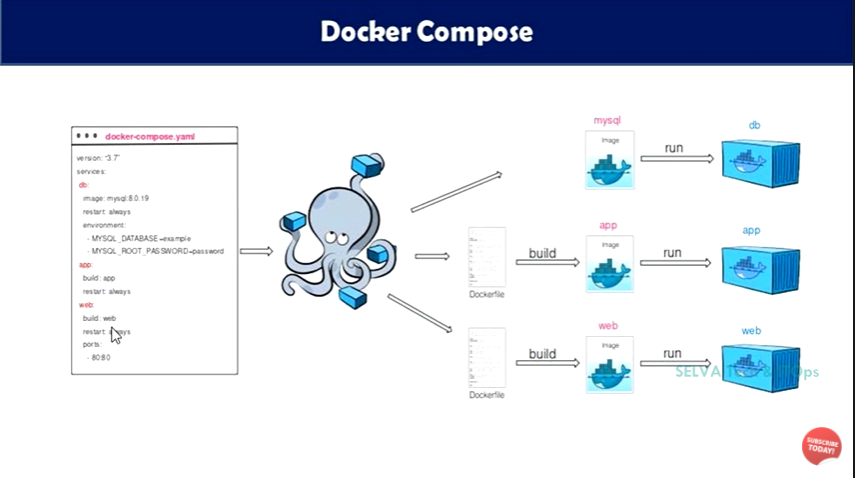
cpus : “0.25”

memory : 256M

**Docker Compose:**

Docker compose it nothing but here we can write the resources as a code and build the container docker compose file is always in .yaml format.





[(63) Docker Full Course in Tamil | Learn Docker in 4 Hours | Docker Full Tutorial for Beginners - YouTube](https://www.youtube.com/watch?v=99wj94_uyG4&list=PL20nrB6KzY4DBPSp7n8tsjUFFZaz8xXTS&index=10).