**Docker**

Note/cheet sheet: <https://docs.google.com/document/d/1-lpYonxwVXt2MrAMcgoU9I27IHgTUWZLI_i4bHO9vKg/edit?usp=sharing>

Docker Installation - <https://docs.docker.com/engine/install>

Vagrantup: <https://www.vagrantup.com/>

Docker Hub: <https://hub.docker.com/>

1. Installtion: <https://docs.docker.com/engine/install>
2. Trainer(Network & Linux): <https://drive.google.com/drive/u/0/folders/1FwSLxEhHZeowmwfB5tNm-kFOwHDjMaGi>

**What is Docker?:** Docker is an **open-source centralized platform designed** to create, deploy, and run applications. Docker uses **container** on the host's operating system to run applications. It allows applications to use the same **Linux kernel** as a system on the host computer, rather than creating a whole virtual operating system. Containers ensure that our application works in any environment like development, test, or production. Docker includes components such as **Docker client, Docker server, Docker machine, Docker hub, Dockerfile, Docker-Composes,** etc.

Docker is designed to benefit both the Developer and System Administrator. There are the following reasons to use Docker -

* Docker allows us to easily install and run software without worrying about setup or dependencies.
* Developers use Docker to eliminate machine problems, i.e. "**but code is worked on my laptop**." when working on code together with co-workers.
* Operators use Docker to run and manage apps in isolated containers for better compute density.
* Enterprises use Docker to securely built agile software delivery pipelines to ship new application features faster and more securely.
* Since docker is not only used for the deployment, but it is also a great platform for development, that's why we can efficiently increase our customer's satisfaction.

## Advantages of Docker

There are the following advantages of Docker -

* It runs the container in seconds instead of minutes.
* It uses less memory.
* It provides lightweight virtualization.
* It does not a require full operating system to run applications.
* It uses application dependencies to reduce the risk.
* Docker allows you to use a remote repository to share your container with others.
* It provides continuous deployment and testing environment.

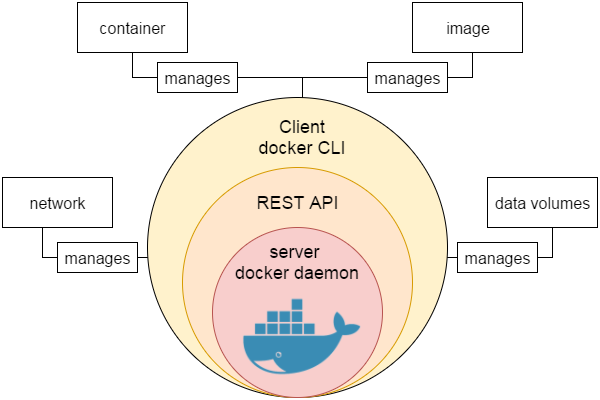
**Docker Containers:**

Docker containers are the **lightweight** alternatives of the virtual machine. It allows developers to package up the application with all its libraries and dependencies, and ship it as a single package. The advantage of using a docker container is that you don't need to allocate any RAM and disk space for the applications. It automatically generates storage and space according to the application requirement.

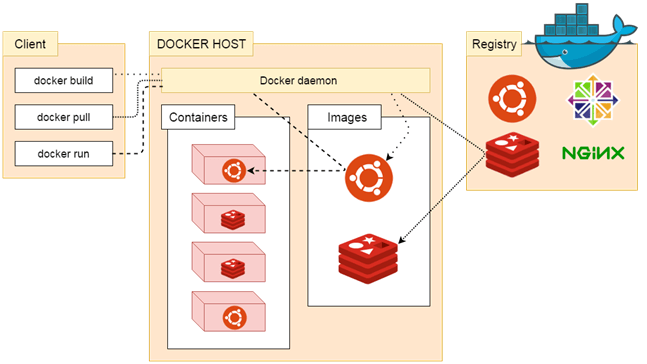
## Docker Engine

It is a client server application that contains the following major components.

* A server which is a type of long-running program called a daemon process.
* The REST API is used to specify interfaces that programs can use to talk to the daemon and instruct it what to do.
* A command line interface client.



# Docker Architecture: Docker follows Client-Server architecture, which includes the three main components that are ****Docker Client****, ****Docker Host****, and ****Docker Registry****.



### Docker Client

Docker client uses **commands** and **REST APIs** to communicate with the Docker Daemon (Server). When a client runs any docker command on the docker client terminal, the client terminal sends these docker commands to the Docker daemon. Docker daemon receives these commands from the docker client in the form of command and REST API's request. Docker Client uses Command Line Interface (CLI) to run the following commands -

**docker build, docker pull, docker run**

### Docker Host

Docker Host is used to provide an environment to execute and run applications. It contains the docker daemon, images, containers, networks, and storage.

### Docker Registry

Docker Registry manages and stores the Docker images. There are two types of registries in the Docker -

**Pubic Registry -** Public Registry is also called as **Docker hub**.

**Private Registry -** It is used to share images within the enterprise.

# Docker Dockerfile

A Dockerfile is a text document that contains commands that are used to assemble an image. We can use any command that call on the command line. Docker builds images automatically by reading the instructions from the Dockerfile.

The docker build command is used to build an image from the Dockerfile. You can use the -f flag with docker build to point to a Dockerfile anywhere in your file system.

$ docker build -f /path/to/a/Dockerfile .

**Instructions**: The instructions are not case-sensitive but you must follow conventions which recommend to use uppercase.

Docker runs instructions of Dockerfile in top to bottom order. The first instruction must be **FROM** in order to specify the Base Image.

A statement begin with # treated as a comment. You can **use**

**FROM:** From is used for bases image

**LABEL:** For set th elabel of images for batter orgnaization

**RUN:** Toexecute any command of the current image

**EXPOSE:** To exposes the docker/imgae on which port you want

**ENV**: Used to set the envireemnt variable

**CMD** - Specifies what command to run within the container

**PULL** - Adds files from your Docker repository

**COPY**: used to copy new files or directories from source to the filesystem of the container at the destination.

**WORKDIR**: working directory for any RUN, CMD and COPY instruction that follows it in the Dockerfile. If work directory does not exist, it will be created by default.

**ENTRYPOINT:**  allows specifying a command along with the parameters

**ADD:** command helps in copying data into a Docker image

FROM ubuntu

ENTRYPOINT echo "Hello, $name".

LABEL vendorl = "JavaTpoint"

RUN /bin/bash -c 'source $HOME/.bashrc; echo $HOME'

PULL. File

ADD /[source]/[destination]

COPY abc/ /xyz

WORKDIR /var/www/html

# Docker Compose

Compose is a tool for **defining and running multi-container Docker applications**. With Compose, you use a YAML file to configure your application's services. Then, with a single command, you create and start all the services from your configuration.

The big advantage of using Compose is you can define your application stack in a file, keep it at the root of your project repo (it’s now version controlled), and easily enable someone else to contribute to your project. Someone would only need to clone your repo and start the compose app. In fact, you might see quite a few projects on GitHub/GitLab doing exactly this now.

Step to create & buildimgae through the docker composes

1. Install the docker composes

docker compose version

1. At the root of the app project, create a file named docker-compose.yml

nano docker-compose.yml

1. Open the docker composes file and add like below code

nano docker-compose.yml

1. Run the docker composes

docker compose up -d

version: "3.7"

services:

app:

image: node:12-alpine

command: sh -c "yarn install && yarn run dev"

ports: - 3000:3000

working\_dir: /app

volumes:

- ./:/app

mysql:

image: mysql:5.7

volumes:

- todo-mysql-data:/var/lib/mysql

environment:

MYSQL\_ROOT\_PASSWORD: secret

MYSQL\_DATABASE: todos

volumes:

todo-mysql-data:

Docker: It’s deployment tool. Deployment application as container.

Vagrantup: It is tool by which you can create virtual machine and same virtual machine can be uses for development, testing and operation

Make Sure:

1. **You have enable port for All traffic for ipv4 and ipv6**

**sudo apt update** => for get & load the package

**sudo apt-get update** => for update the the apt package

**sudo apt install docker.io –y** => To install docker. u can uses sudo ro elimate at time of installtion but try to install by using sudo

**docker --version** => to get install varsion of docker

**sudo systemctl start docker** => to start the docker

**sudo systemctl status docker** => For get the status(is running or not) of docker

**sudo systemctl enable docker** => to enable the docker

**docker ps -a** => List of all docker / container (running/stoped)

docker ps -s => check the size of Container

**docker container ls -a** =>

**docker --help** => Get the list of command

**sudo docker pull centos** => to pull the centos image from docker to our linux server/ubunto machine

**sudo apt-get install apache2 –y** => to install apache2 image

**sudo apt-get install ubantu –y** => to install apache2 image

**sudo service apach2 status** => you check the status of apache service(apache2, centos, ubuntu) wther it’s running or not

**sudo** **service apache2 start =>** To start the service **l**ike apache2 or any

**docker pull ubuntu** => to pull the ubuntu or any image from dockerhub

**docker pull centos** => to pull the centos or any image from dockerhub

**docker search centos** => to search the image like centos any image on dockerhub

**docker image or ls** => give the list of docker image in your server

**docker run -it -p 801:80 --name containerName**=> create a docker as interactive mode which name is containerName and it’s run on port 801 and which machine running on port 80

**docker run -d -p 8080:80 nginx** => create a docker which name is nginx and it’s run on port 8080 and which machine running on port 80

**sudo docker run -itd -p 90:80** ubuntu => create a docker which name is nginx and it’s run on port 90 and which machine running on port 80

**docker exec -it conatiner\_name bash** => To login into container with the help of container name or id

**docker exec -it nginx bash** => To login into container with the help of container name or id

**docker exec -it containerID bash** => To login into container with the help of container name or id

**exit** => To exit from current container

**apt update -y** => To get & load the package in docker container

**apt install apache2 -y** => To install apache2 in docker container

**apt install apache2 git -y** => To install apache2, git, or u can pass any number or image in docker container

**docker stop container\_name** => to stop the running container by contanerName or ID

**docker start container\_name** => to start the stop container by contanerName or ID

**docker restart myContainerName =>** To restart the container by id & name

**docker inspect dockerContainerID** => to inspect and see the details about the particular container

**Docker rename old\_container\_name new\_container\_name**

**docker rename webserver-01 nginx-server** => For Rename the container

**docker logs container\_name/container\_Id**

**docker logs -f container\_name/container\_Id**

**docker logs -f webserver-01** => For check log of running container

**docker stats** => To Check resource usage (CPU/Memory/ Network/ Disk)

**docker system prune -fa** => To deleted all stopped containers and images not in use:

docker info => To get details information about docker

**Exercises -1**

**1. sudo docker run -itd -p 999:80 ubuntu**

**2. sudo docker exec -it 3a58f25 base**

**3 sudo docker exec -it 3a58f25 bash**

**3 apt update**

**4 apt install nginx**

**7 service nginx status**

**8 service nginx start**

**9 service nginx status**

**10. 54.249.179.47:999 => Open The browser then check your services: your public ip:portno**

**Exercise -2 (Not running)**

**1 sudo apt-get update -y**

1. **sudo apt-get install apache2 –y**

**3. service apache2 status**

**3 sudo apt-get install docker.io**

**4 sudo docker ps**

**5 sudo systemctl status docker**

**6 sudo docker run -itd -p 101:80 ubuntu**

**7 sudo docker ps -a**

**8 sudo docker exec -it 59f8d46b38f5 bash**

**10. apt update -y**

**5 apt install apache2 -y**

**6 apt install git -y**

**7 service apache2 status**

**8 service apache2 start**

**10 service apache2 status**

**Exercises -3**

**1 sudo docker ps**

**3 touch Dockerfile**

1. **vim Dockerfile**

**Note: Write the command which u need to perform => Docker File => Command**

**FROM ubuntu:18.04**

**RUN apt update**

**RUN apt install apache2 -y**

**ADD . /var/www/html**

**ENTRYPOINT apachectl -D FOREGROUND**

**5 sudo docker ps -a**

**8 sudo docker build . -t imagefileimage bash**

**10 sudo docker build . -t devopsintelipaat/app:v3**

**11 sudo docker run -itd -p 88:80 devopsintelipaat/app:v3**

**14 sudo docker exec -it f1d363bdb bash**

Docker hub - <https://hub.docker.com/>

Building Image

============

1. **Create docker image and clone it from git code /project from git inside the nginx/html folde**

docker run -d -p 80:80 --name web nginx

docker exec -it web bash

cd /usr/share/nginx/html

rm -rf \*

apt update -y

apt install git -y

git clone <https://github.com/mdn/beginner-html-site-styled>

mv [beginner-html-site-styled](https://github.com/mdn/beginner-html-site-styled)/\* .

Exit or Ctrl + A + D

1. **commit the docker image & to dockerhub**

**To create a image from container.**

docker commit container\_name image\_name

docker commit web myfirstimage

1. **Create & setup profile on dockerHub**

**Create a docker hub account:**

Docker hub - <https://hub.docker.com/>

1. **Login to Docker hub**

**To Login to Dockerhub account using cli**

docker login

Username - docker hub account username

Password - dockerhub account password

Login Success

1. **To tag and push the image**

docker tag image\_name dockerhub\_username/repo\_name

docker tag myfirstimage gauravdemo06/myfirstimage

docker push gauravdemo06/myfirstimage

1. **To prune  the docker**

docker system prune -fa

1. **For pull the docker image**

docker pull gauravdemo06/myfirstimage

docker run -d -p 80:80 gauravdemo06/myfirstimage

**Docker file**

1. **Create docker file**

touch Dockerfile

1. Open & Edit the docker file

vim Dockerfile

1. Docker file command

—-------------------------------------------

FROM nginx

RUN apt update -y && apt install git -y

WORKDIR /usr/share/nginx/html

RUN rm -rf \*

RUN git clone https://github.com/mdn/beginner-html-site-styled.git && mv beginner-html-site-styled/\*

EXPOSE 80

—--------------------------------------------

1. Build the image from docker file

**To build the image from Dockerfile**

docker build -t image\_name .

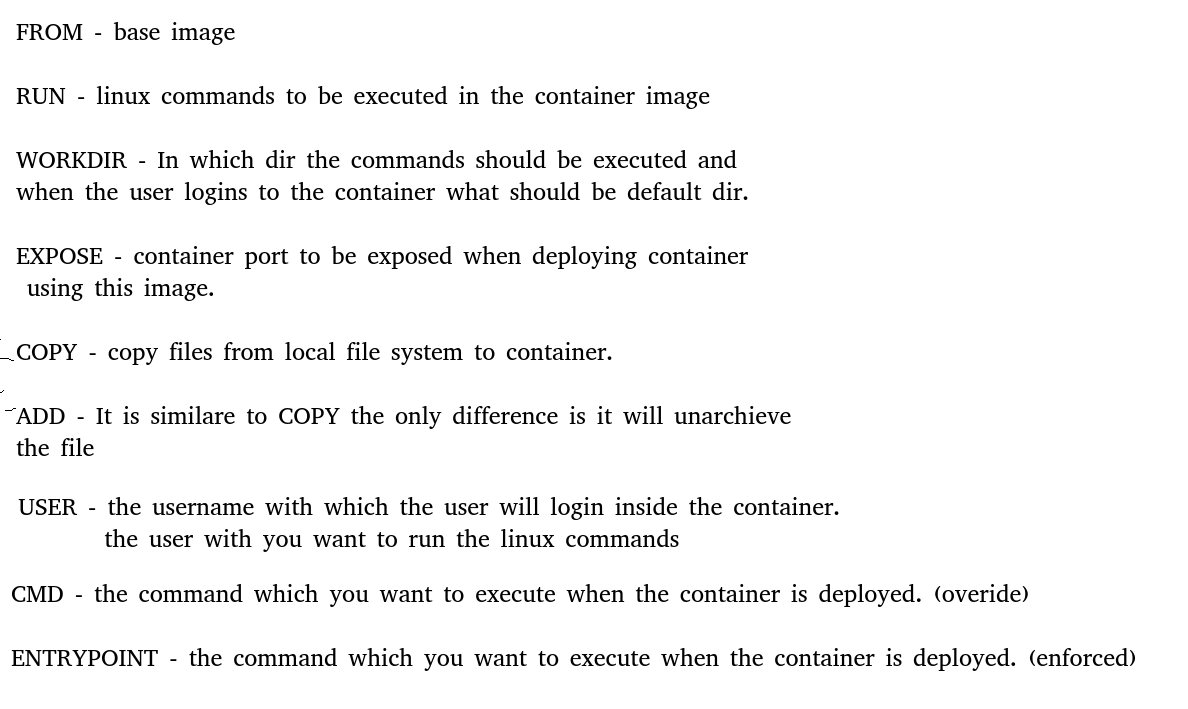
1. To buld the docker image from different dicker file

**To build the image from different file name other than Dockerfile**

docker build -f filename -t image\_name .

1. To run the docker image

docker run -d -p 80:80 --name webapp image\_name



**Home work-**

Build a docker image for a php application

<https://github.com/fajar7xx/olshopfajar>

**Day - 5**

**Docker file**

LABEL - (Metadata of Image)

ENV - (OS Env variable, which will be in the container when the container is deployed)

**Docker Networking**

1. Bridge
2. Host
3. None (also known as null)

**To List Network**

docker network ls

**To create a custom network**

docker network create --driver bridge mynetwork

**To use the network created**

docker run -d --network mynetwork --name web nginx

docker run -d --network mynetwork --name db nginx

docker exec -it web bash

apt update -y

apt install iputils-ping -y

ping db

**Homework -**

Mysql - <https://github.com/docker-library/mysql/tree/e0d43b2a29867c5b7d5c01a8fea30a086861df2b/8.0>

Wordpress - <https://github.com/docker-library/wordpress/tree/97f75b51f909fbd9894d128ea6893120cfd23979/latest/php7.4/apache>

Setup own Image Registry -

<https://hub.docker.com/_/registry>

**Day 6**

**—-----**

**Docker Storage**

**—-------------------**

1. **Bind Mount**

mkdir /home/ubuntu/data

docker run -d -p 80:80 --name webserver -v /home/ubuntu/data:/usr/share/nginx/html nginx

1. **Docker volumes**

**To create docker volume**

docker volume create mydata

**To list docker volumes**

docker volume ls

**To delete docker volume**

docker volume rm volume\_name

docker run -d -p 80:80 --name webserver -v mydata:/usr/share/nginx/html nginx

**Docker Compose**

sudo apt install docker-compose -y

**Docker-compose.yml**

version: "3"

services:

 db:

   image: mysql

   ports:

     - "3306:3306"

   environment:

     - MYSQL\_ROOT\_PASSWORD="redhat"

     - MYSQL\_DATABASE="wordpress"

     - MYSQL\_USER="wordpress"

     - MYSQL\_PASSWORD="redhat"

   volumes:

     - "mydbvol:/var/lib/mysql"

 wordpress:

   image: wordpress

   ports:

     - "80:80"

   volumes:

     - "mywordpress:/var/www/html"

volumes:

  mydbvol:

  Mywordpress:

Troubleshooting:

1. Docker exec -it wordpress bash

* Ping (connectivity)
* Connect to DB

apt update -y

apt install iputils-ping -y

ping db

apt install mysql -y

mysql -h db -u wordpress -p

password:

Access Denied

Hint:

Mysql remote connection = localhost

—----------------------------------------------------

**Day -7**

**Docker-compose integration with Dockerfile**

1. **//Create folder**

mkdir project

1. **//create docker file**

touch Dockerfile

1. **//docker file command**

FROM nginx

COPY index.html /usr/share/nginx/html

EXPOSE 80

1. **//create index.html**

echo “<h1> Hello World </h1>” >> index.html

1. **//create docker composes file**

touch docker-compose.yaml

1. **//Docker composes file command**

version: "3"

services:

 web:

  build:

    context: .

  ports:

   - "80:80"

1. **To apply the docker-compose.yml file**

docker-compose up

1. **To run the containers in background**

docker-compose up -d

1. **To stop & remove the containers deployed through docker-compose**

docker-compose down

1. **To list running containers**

docker-compose ps

1. **To check logs**

           docker-compose logs

1. **To validate the docker-compose file**

docker-compose config

1. **To apply docker-compose with a different file name other than docker-compose.yaml**

            docker-compose -f filename.yaml up

1. **To build the image and deploy the containers**

docker-compose build --no-cache

            docker-compose up

1 sudo apt-get update

2 sudo apt-get install docker.io -y

3 sudo apt-get install apache2 -y

4 service apache2 status

5 cd var/www/html

6 cd /var/www/html

7 git clone https://github.com/themes12/Real-Estate-Landing-Page-v2

8 git init

9 sudo git init

10 sudo git clone https://github.com/themes12/Real-Estate-Landing-Page-v2

11 ls

12 cd /etc/init.d

13 ./apache2 start

14 systemctl ststus apache2.service

15 systemctl ststus apache2 service

16 systemctl status apache2 service

17 ./apache2 start

18 sudo ./apache2 start

19 sudo apt install curl -y

20 curl localhost

21 sudo docker run -it -p 72:80 --name container ubuntu bash

22 history

1 apt update -y

2 apt install apache2 git -y

3 cd /var/www/html

4 cd

5 service apache2 status

6 service apache2 start

7 history

docker image ls

25 sudo docker image ls

26 docker ubuntu status

27 sudo docker ps

28 sudo docker run -itd -p 90:80 ubuntu

29 sudo docker exec -it 18d35a1be69873cc201d8cc30ef779d63add413c7412e6e54814674861ef4052 base

30 sudo docker exec -it 18d35a1be69873cc201d8cc30ef779d63add413c7412e6e54814674861ef4052 bash

31 nano index.html

32 sudo nano Dockerfile

33 sudo docker build . -t DevOpsIntelipaat/app:v3

34 sudo docker build . -t devopsintelipaat/app:v3

35 sudo docker build . -t ubantu/app:v3

36 sudo nano dockerfile

37 sudo nano Dockerfile

38 sudo docker build . -t ubuntu/app:v3

39 sudo nano Dockerfile

40 sudo docker build . -t ubuntu/app:v3

41 sudo docker images

42 sudo docker run -itd -p 88:80 ubuntu/app:v3

Docker File => Command

FROM ubuntu:18.04

RUN apt update

RUN apt install apache2 -y

ADD . /var/www/html

ENTRYPOINT apachectl -D FOREGROUND

sudo docker run -itd -p 88:80 devopsintelipaat/app:v3