**Docker installation Ubuntu**:

1. Update the apt package index and install packages to allow apt to use a repository over HTTPS:

sudo apt-get update

sudo apt-get install ca-certificates curl gnupg

2.Add Docker’s official GPG key:

sudo install -m 0755 -d /etc/apt/keyrings

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg

sudo chmod a+r /etc/apt/keyrings/docker.gpg

3. Use the following command to set up the repository:

echo \

"deb [arch="$(dpkg --print-architecture)" signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \

"$(. /etc/os-release && echo "$VERSION\_CODENAME")" stable" | \

sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

4. Update the apt package index:

sudo apt-get update

5. Install Docker Engine, containerd, and Docker Compose.

sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin

6. Verify the Docker installation

sudo docker run hello-world

7. Adding User to docker group - (so that we don't have to use sudo each time with docker command)

sudo usermod -aG docker $USER

newgrp docker

8. Verify if above commands got replicated

docker run hello-world

9. Commands to start docker service automatically after server restart

sudo systemctl enable docker.service

sudo systemctl enable containerd.service

**Docker Commands:**

1. Check Docker version info

docker --version

2. Pull images from central repository

docker pull <image-name>

docker pull ubuntu

docker pull centos

docker pull nginx

3. list all the docker images downloaded on your local system

docker images

4. Create an run a container in foreground

docker run ubuntu

5. Create an run a container in background

docker run -itd centos

docker run -itd ubuntu

6. Create an run a container in background with port

docker run -d -p 80:80 nginx

7. Naming Containers

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

8. Rename Container

docker rename 21ba8b3fa32d newname

9. List running containers

docker ps

10. List all containers (Even if exited)

docker ps -a

10. accessing the container, go inside the container

docker exec -it <container-id> bash

11. Stop container

docker stop <container-id>

12. To remove a stopped container from the system

docker rm <container-id>

13. To remove an image from the system we use the command “rmi”

docker rmi <image-id>

14. To stop all containers, to remove all stopped containers, to remove all images from system

docker stop $(docker ps -aq)

docker rm $(docker ps -aq)

docker rmi $(docker images -aq)

15. **SAVING CHANGES TO A EXISTING DOCKER CONTAINER**

docker pull ubuntu

docker run –it –d ubuntu

docker exec –it <container-id> bash

Install apache2 on this container, using the following commands

apt-get update

apt-get install apache2

Exit the container, and save the container using this command.

The saved container will be converted into an image with the name specified is (use dockerhub username)

docker commit <container-id> <username>/<container-name>

**16. docker login & push**

docker login

docker logout

docker login <private url - registry.example.com>

docker logout <private url - registry.example.com>

docker push <username>/<container-id>

You can verify the push on DockerHub

17. **Write sample Docker file**

FROM ubuntu

RUN apt-get update

RUN apt-get -y install apache2

ADD . /var/www/html

ENTRYPOINT apachectl -D FOREGROUND

ENV name Devops Intellipaat

**write index.html**

<html>

<title> Sample Website </title>

<body>

Hello World

</body>

</html>

18. run the build command to make custom image

docker build -t suhailasad/custombuild .

launch that container

docker run -it -d -p 80:80 suhailasad/custom

docker exec -it containerid bash

echo $name

19. **Create Docker volume**

docker volume create my-vol

20. **Attach or Mount created docker volume**

docker run -d -it --mount source=my-vol,destination=/app ubuntu

21. **bind volume**

docker run -itd -v /root/test\_volume/:/app ubuntu

22. **Running sample Docker Compose file**

create a folder called docker and place the below file

(docker-compose.yml)

version: '3'

services:

sample1:

image: httpd

ports:

- "80:80"

sample2:

image: nginx

ports:

- "8085:80"

filename: docker-compose.yml

docker-compose up -d

docker ps

docker-compose down

**23. Docker Swam initialization**

**on Master node**

docker swarm init --advertise-addr 172.31.8.135(private ip addr of master)

**on Worker node**

make sure you have installed docker on worker node first

docker swarm join --token SWMTKN-1-5pxa8c7dvy117r49tqmtfxqtrizjozhhz1mtgvuq9hzultfsdu-cln9fexahtmfpzvwp4fhdgg0y 172.31.8.135:2377

**on master node**

docker node ls

24. **to leave swarm in worker**

docker swarm leave --force

25. **to close docker swarm in master**

docker swarm leave --force

26. **deploy multi tier application**

docker stack deploy -c wordpress.yml wordpress-stack

docker stack ls

docker stack ps wordpress-stack

docker stack rm wordpress-stack

**wordpress.yml**

version: '3.3'

services:

db:

image: mysql:5.7

volumes:

- db\_data:/var/lib/mysql

environment:

MYSQL\_ROOT\_PASSWORD: somewordpress

MYSQL\_DATABASE: wordpress

MYSQL\_USER: wordpress

MYSQL\_PASSWORD: wordpress

wordpress:

depends\_on:

- db

image: wordpress:latest

ports:

- "8000:80"

environment:

WORDPRESS\_DB\_HOST: db:3306

WORDPRESS\_DB\_USER: wordpress

WORDPRESS\_DB\_PASSWORD: wordpress

volumes:

db\_data: