# # Docker-for-Developers-Core-Concepts

https://www.linkedin.com/learning/docker-for-data-scientists/why-docker

yum install docker -y(Install Docker over AWS)

**To start docker engine**

* service docker status
* service docker start

------------------------------------------------------------------------------------

**To see all containers**

* docker ps -a
* on each docker run always create new container, image remain same

**only running**

* docker ps

---------------------------------------------------------------------------------

**To see all images locally**

docker images

**To find images from docker hub**

docker search jenkins

**To download images from docker hub**

* **centos – image name**

docker pull centos

**To give name to container**

* run - create and run docker,
* it – interactive terminal
* riki\_container container name
* ubuntu\_images – image name
* /bin/bash – to run on the terminal

docker run -it –name riki\_container ubuntu\_images /bin/bash/

**Docker start from container**

* **To start container(on exit from container it is automatically stop)**

docker start riki\_container

* **To attach volume with container(only if container is running)**

docker attach bhupi

* **Run in background**

**To stop container(come out of container)**

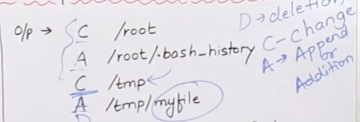
docker stop bhupi

**Docker remove**

docker rm bhupi

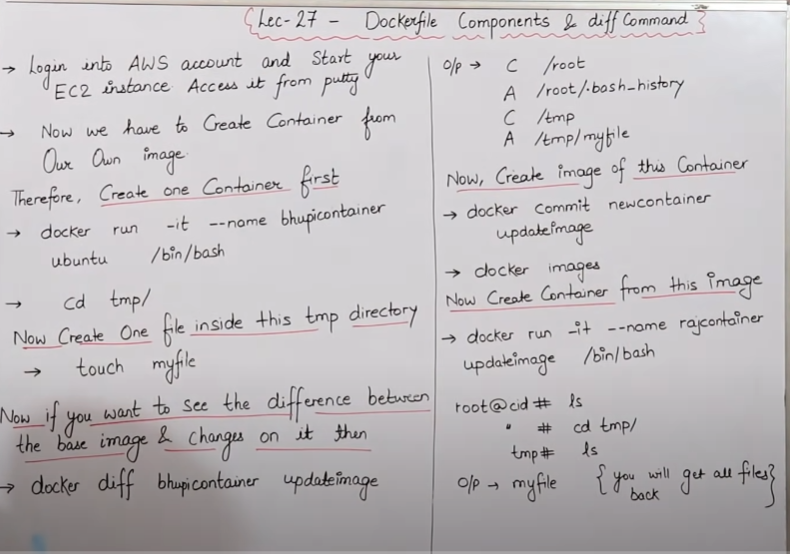
**Difference between two containers**

docker diff container1



**Create image of container**

* Docker commit newcontainer updateimage



cat /etc/os-release

**Commit over Docker hub**

docker commit imagename

**Delete all conatiner**

docker container prune

docker image prune –all

Docker file components and commands:-

## Docker image function:-

* FROM - base image
* RUN - to excute command, it will create layer in image
* MAINTAINER - Author/Owner/Description
* COPY - copy file from local( docker vm) not fom internet
* ADD - Similar to copy but do from internet, also unzip the file
* EXPOSE - expose port
* WORKDIR - set working directory for a container
* CMD - excute command but during container creation time
* ENTRYPOINT - Smilar to CMD
* ENV - environment variables
* ENV - future running containers.
* ARG - building your Docker image

## Create docker file

FROM ubuntu

RUN echo “hello rakesh” > /tmp/testfile

## To create docker image from the file:-

docker build -t myimage .

docker ps -a

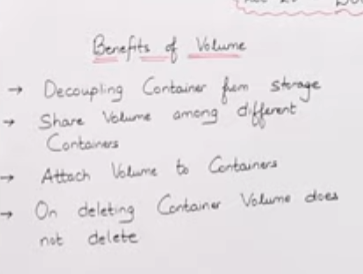
docker images’

## To create docker container form the file:-

docker run -it --name myconatiner myimage bin/bash/

**Volume is simply directory**

* Firstly declare directory as volume thn share volume
* Created and shared only while creating the container
* Even if container stop still we can access volume
* Volume can't create on existing container
* Volume can be sahred with any number of container
* Volume will not be included when we update the image(running container)



## Creating volume from dockerfile

From ubuntu

Volume [“/myvolume1”]

Create folder in root directory with name “myvolume1”

## Share container1 volume with other

**Provided all right by prevoleged**

Docker run -it –name conatiner2 new **–previleged=true** --volume-from container1 ubuntu /bin/bash

## Docker volume

Docker run -it –name hostcont -v /home.ec2-user**:**/riki\_volume **–previleged=true** --volume-from container1 ubuntu /bin/bash

#### Two volume can shared data

* /home.ec2-user -🡪 ec2 volume
* Riki\_volume 🡪 container volume

#### Some other command

* Docker volume ls
* Docker volume prune : remove unused volume
* Dockr volume create v1
* Docker volume rm v1
* Docker volume inspect vname
* Docker container inspect cnme

## Docker port expose

1.1 yum update

1.3 yum install docker -y

1.3 service docker start

1.4 Docker run -td –name techserver -p 80:80 ubuntu

* -td : container will run in background
* -p 80:80 any traffic on **host port 80** will be sent to **container port 80**(website in container).
* -p override expose, world wide access
* expose can be used for internal communication btn the container

2. docker **port** techserver

Give **all port** used by container

3. docker **exec** -it techserver /bin/bash

* It will create a new process

3.1 Docker attach : to go inside the container

* In existing process

4. apt -get update

5. apt-get install apache2

6. cd /var/ww/html/

7. echo “hello” > index.html

8. service apache2 start

------------------------------

docker run -td –name myjenkins -p 8080:8080 jenkins

* jenkins : server to run the application

## Docker commit on docker hub

* docker commit container1 image1
* docker tag image1 dockerid/newimage
* docker push dockerid/newimage

Create image1 from the container1

## Docker pull on docker hub

* dokcer pull dockerid/newimage
* docker run -it –name mycon dockerif/newimage /bin/bash