**Docker Topices**

**Engine**

**Images**

**Containers**

**Registry**

**Volumes**

**Docker file**

**Docker Commit**

**Docker network**

**DockerLink**

**Docker-Compose**

**Docker-Swarm**

1. **Now Go to Aws Take One Ubuntu machine And connect This Via putty. Set sudo permission.**
2. **Docker -Engine**
3. **For Docker Instalation go to below Url and Run Below two scripts (c,d)**
4. **https://get.docker.com/**
5. **$ curl -fsSL get.docker.com -o get-docker.sh**
6. **$ sh get-docker.sh**
7. **Docker –Images**
8. **We can Pull Images using GUI Or CLI**
9. **Docker imges pull from GUI Go to hub.docker.com**
10. **GUI:Now open Docker Hub Page.at Right side top up Explore option is there just click that button Now see list of images .**
11. **CLI: Now go To putty RUN Below Command**

**docker serach images (In Putty)**

**here also see list of Images.**

**To serach particular Images Below is command**

**docker serach images Centos (In Putty)**

**To serach local images below is command(after pull any images from GUI Or CLI)**

**docker images (In Putty)**

To Pull Imges from GUI OR CLI Use Below Command (Here Centos is Image Name)

**docker pull centos (In Putty)**

**docker imges (in putty)**

Now get Centos Image in local

**3)Docker-Containers:**

To Run containers on images below is command

**docker run -dti --name mycontainer centos /bin/bash**

now get 64 bit code One container is created on centos image

To check Running containers .Just Run Below command

**docker ps**

To check Both Running and stopded Container Run below Command

**docker ps –a**

container Id and name get from below command

**docker ps**

To Login to particular Container Below is Command

**docker attach Container Id or Name**

now we are in docker container(noting but one instance or server)

here we are install everything because it is a lightweight

to logout form container just press **Clt+PQ**

Now we are in docker engine

To get conatner Information Execute Below command

**docker inspect conatinerId or name**

directly create a container and login into container below is command

**docker run -ti --name cent2 centos /bin/bash**

just check how may continers are ruuning

**docker ps**

to stop container below command

**docker stop container id or name**

To start stop container below is command

**docker start container id or name**

To remove container below command

**docker rm –f container Id or name(this for running container stop)**

normally first we have to stop the container then remove container

**docker rm container id or name**

to remove existing image

**docker rmi image id or image**

to remove existing image forcefully

**docker rmi –f image id or image(if any containers are running also)**

**4)Docker- Registry**

1)First go to <https://hub.docker.com/> and create you’re A/C with Email and password.

2) create one Repository

3) pull some images using(in putty)

docker pull java

docker pull centos

docker pull tomcat

Create your own custom image and push this image into **DockerHUb(**like git)**docker tag java Repository path:sample (give any name)**

the above command java is your recently pull image

Repositoty path=abc/repository name

Sample =your image name

Now push image into Repository using below command

**docker login**

**username:**username for cteate an account time

**password:** give password as per username

**docker push central Repo path:image name**

**(Ex: docker push abc/myrepo:sample)**

Now pull image from central Repo to local Repo

**docker pull path of cenral Repo:image name**

**Ex:(docker pull abc/myrepo:sample)**

**Just check :docker images(now get new image from central Repo)**

**5)Docker-Volumes**

Now Go to aws console And go to EBS(Elastic Block Storage) create volume 5GB

Note:create volume as per instance availability Zone

Now attach this volume to instance(Goto action attach select instance and then attach).

Note:Below all work done in putty.

Below command for partision

**fdisk/dev/xvdf**

**command (m for help) : n(create new voulume)**

**select (default P): p (primary partision)**

**partition number: 1**

**first sector :**

**second sector: +5000MB**

**command (m for help) :w**

**lsblk**

now format this part ion using Below command

**mkfs.ext4 /dev/xvdf1**

Now attach this volume to kernel by using below command

**Partx /dev/xvdf1**

**Or**

**Kpartx /dev/xvdf1**

**Or**

**Partprobe /dev/xvdf1**

**Now create one directory**

**mkdir volume**

Now Mount this Voulme using below command

**This Temporary mount: mount /dev/xvdf1 volume(directory name)**

**For perment mount : vi/etc/fstab**

**/dev/xvdf1 /root/volume ext4 defaults 0 0(save and quit)**

**Just check: df –h**

**Now 5GB also mount to the instance.**

now create a container along with volume by using below command.

**docker run -dti - -name c1 –v /root/volume: /tmp –p 81:80 ubuntu /bin/bash**

**observation:**

**c1=creating conatainer name.**

**/root/volume=create this voulume in this path**

**/tmp =attach this volume in container tmp path**

**Ubuntu=our pull image**

**81:80 we forward port from 80 to 81 for container**

**Just check in engine**

**Cd volume(directory name)**

**Type:ls**

**No data.**

**Log into container. By using below command**

**docker attach c1**

**cd tmp/**

**ls**

**no data found**

**Now crate some files in container using touch command**

**touch log**

container is lightweight some every thing we are install some

apt-get update

apt-get install vim

Now add some content for log file using vi editor

vi log

this docker demo(:wq!)

logout from container **CTRL PQ**

Now we Are in docker engine just check

Cd volume

ls

Now get as it is same file and some from container.

Volume main concept is some containers are running in production are any Environment .If container is down or fail data is No lose purpose we are save data in engine also.

**Now Deployed One Web Application in Container**

**Now install apache web server in container**

**apt-get install apache2**

**/var/www/html/ vi index.html**

**<html>**

**<body> <h1>DOCKER DEMO SESSION </h1></body>**

**</html>**

**:wq!**

No Go To aws console open inbound rules .

Custom tpc 81 any where. And save

Now Just copy the public ipaddress and paste in any browser(google,firebox)with :81

Ex:123.156.135.12:81

Now web application open in browser.

**Docker –File**

Docker file is script ,composed of various commands and arguments listed successively to automatically perform acton on a base image in order to create a new one.

For more reference refer ::digitalocean.com

Docker file commands

1.ADD

2CMD

3.ENTRYPOINT

4.ENV

5.EXPOSE

6.FROM

7.MAINTAINER

8.RUN

9.USER

10.VOLUME

11.WORKDIR

First we have install docker engine and pull any image.

Ex: **docker pull ubuntu**

Just check: **docker images**

Now Create One Directory : **mkdir myfile**

**Cd / myfile**

Now create docker file

vi Dockerfile

FROM ubutntu

RUN apt-get update && apt-get install git –y && apt-get install apache2 –y && service2 start

RUN apt-get install tree-y

CMD mkdir /root/test

MANTAINER khasim

ADD /etc/passwd /tmp

EXPOSE 81

WORKDIR /bin/bash

USER shaik

VOULUME / tmp: / tmp

:wq!

ls

Dockerfile (avalible)

Now build this image using below command

**docker bulid –t khasim .**

Note here **(.**) is Represent docker file path

Suppose docker file in opt directoty

docker build –t khasim /opt/.

Suppose docker file in tmp directoty

docker build –t khasim /tmp/.

Now just check ::: docker images

1)ubuntu

2)khasim

Now create one container on this khasim image.

docker run –dti - -name=shaik khasim /bin/bash

the above command shaik is container

khasim is image.

Check running container

**docker ps**

Now log in into container

**docker attach shaik**

Now check packages are installed or not

dpkg -l git

dpkg -l apache2

service apache2 status

dpkg –l tree

netstat –l

apt-get install netstat –y

ip a

**6)Docker –Commit**

By using docker commit we can build customized image by using Running containers

By using below command we can create our own custom images.

**docker ps** (check for running container)

**docker commit container Id Or Name yourimageName**

**ex ::docker commit 30658596 cutsom**

**docker images**

**7) Docker –Network**

**By using docker network we can assign your custom ip address to running container**

**docker run -dti -- name abc - -net mynetwork --ip 192.168.0.1 –p 82:80 ubuntu /bin/bash**

**docker inspect abc**

**8)Docker –link**

Docker link nothing but linking between two containers

**docker pull wordpress**

**docker run –dti – -name press1 –p 95:80 wordpress**

**docker run - -name =press –p 96:80 -d wordpress**

**docker ps**

Now go to Aws Instance open inbound rules open customtcp 95,96 port open set any where

**docker pull mysql**

to run container on mysql image below is command

**docker run - -name msql –e MYSQL\_ROOT\_PASSWORD=abc1234 -d mysql**

Two link the two container by using below command

**docker run –name press1 - - link msql:mysql -p 100:80 –d wordpress**

now copy and paste public ipaddress with port no 100

ex:156.352.2.3:100

site Title:www.abc.com

Username:khasim

Password:khasim@1227

Email:khasim@gmail.com

Press ok

Now set login

Username:

Password: press login

**9)Docker –Compose**

**Docker compose is tool for defining and running multi-container Docker appliacation**

**With compose you use a compose file to configure your appliaction’s service.then**

**Using a single command you create and start all the service from configuration.**

**10) Docker-swarm**

Docker Swarm is a clustering and scheduling toos for Docker containers

Docker Swar m work on RAFT Algoritham

1. First Go to Aws take 5 ubuntu Instance
2. Connecting This 5 Instances using Putty.
3. Set sudohostname mg1,sudo -i(1 instance):: sudohostname mg2,sudo –i(2 instance):: sudohostname mg3,sudo –i(3 instance),:: sudohostname node1,sudo –i(4 instane)::sudohostname node2,sudo –i(5 instane)::
4. Now Install docker Engine every instance
5. **$ curl -fsSL get.docker.com -o get-docker.sh**
6. **$ sh get-docker.sh**
7. The above two scripts for installing docker Engine.
8. Now go to Aws ,Inbound Rules open port 2377 custom tcp anywhere .
9. Now mg1 set has manager .below is command
10. **docker swarm init**

**Now Get Two Token**

**1 for node(workers)**

**2 for managers(managers)**

1. Copy the token 1 and paste in **node1** and **node2.**
2. **Now get message This two nodes are joined swarm workers.**
3. Now come to mg1 and execute below command
4. **docker node ls(run under mg1)**
5. **The above command not work in nodes.Only work in managers (mg1)**
6. Now add mg2,mg3 to cluster execute this below command Then we will get one token copy that token paste in managers**.**
7. **docker swarm join-token manager (run under mg1)**
8. **Now get one token .this token copy and paste in mg2,mg3**
9. **docker node ls (run under mg1)**
10. Depend on Requirement we will run containers with out downtime.
11. docker pull nginx(run under mg1).
12. To Run multiple containers using docker swarm service below is command
13. **docker service create - - name myswarm - - replicas 10 nginx**
14. **Checking purpose in mg1 run this command**
15. **docker ps (now get two containers running) as it is check all nodes and managers.**
16. To check running service below is commnd
17. **docker service ls**
18. **docker service ps servicename**(this command no.of processor are running in particular service).
19. **ex : docker service ps myswarm**
20. now shutdown for mg1 using below command
21. **shutdown –h now**
22. **now mg2 has leader**
23. **just check below command**
24. **docker node ls**
25. **docker ps(mg2,mg3,node1,node2) 10 containers should run.**
26. As per Raft algorithm two manages are still running .In case mg2 also fail then we set

Node2 has manger .Below is command

**37)docker node promte node2(running under mg2)**

**38)docker node ls (running under mg2) now node2 has added has a manger.**

**39) docker node demote node2 (to demote a manager). (run under mg2)**

**40)now shutdown node2**

**41) shutdown –h now**

**42) Note:keep basic manager are running.**

**43) the main agenda for swarm is to keep all containers are running mode.**