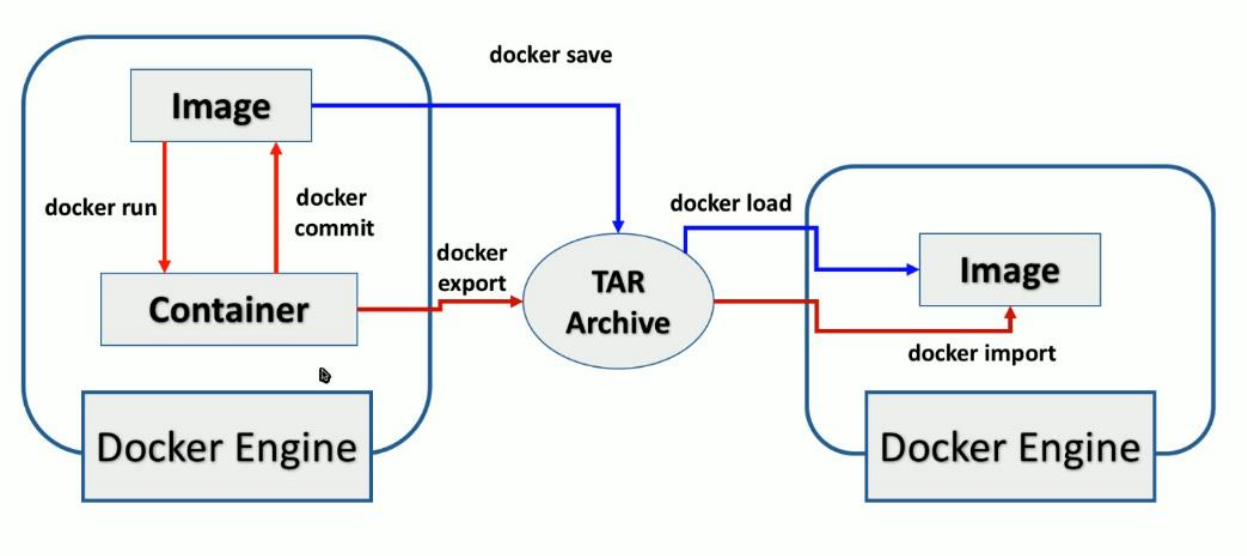


This above diagram will help us to get idea about docker save and load, docker export and import commands.

WE have docker engine1 and docker engine2 so what ever container am using in system1 so that I want to share with another system for this what we can do is

How to save docker image as tar file :

docker save <image_name> > <Imange_name.tar> ----------on system 1

Now we have generated tar file and this tar file needs to move to another docker engine in 2 ways.

1. One is using scp command but for this we have to set up private and public keys.

Rather than scp command we can go with simple way with webserver.

2. Second one and easy way is we can install one webserver and we can copy the file with that.

Ubuntu :

sudo apt-get install apache2

Redhat/centos/fedora :

sudo yum install httpd

So n our example tar file has been create root home folder so we have to that tar file into webserver html folder so that we can download from webserver via internet and ip address.

cd /var/www/html

cp /root/mycentos.tar

systemctl start apache2 (or) systemctl start httpd ----if yum

Then go to the browser and type the ip address of the docker system and slash tar file

Then we have to go to docker engine2 system and load the downloaded file.

wget http://<ip address>/mycentos.tar

docker load - -input <tarfile.tar>

Now we can see our new image on system2 and that we can run it

docker run –itd <new image>

docker ps

And it is running status we can see.

Now we are going to discuss about docker import and export commands

Here we need to have a central repository because if we have 100 systems means we cant copy to all systems so that we can keep in central repository and from there we can download.

So here we have to create docker hub account by clicking create account in www.dockerhub.com link.

Then give username and email Id and password and then go to you email inbox and verify your email then we are able to login dockerhub now. Go to dockerhub.com and gv user id and pwd and login.

And now go to terminal and try to login to docker hub with your credentials using below commands

docker login

user id : goldentech

password : xxxxxxxxxxxxx

And then push the image from your local list to dockerhub (central registry).

docker push <goldentech/mycentos>

And go to docker hub and verify for uploaded image you can able to see there.

docker logout --- getting out from docker hub

We done all basic commands in docker and we are going have new concept called volumes now.

Volumes :

Going to remove all containers

docker rm $(docker ps –a –q)

So what I want to do is we want to launch a container with below command

docker run –it - -rm centos

we are inside of container now

Ok now am going to create some data like files inside of container.

touch /tmp/configdata

vi configdata

Enter some data

:wq

And simply give exit to the container

Exit

That's all now you can give docker ps –a

Nothing will be available since we have used - -rm while running the container and as well as we lost our created data also. But even though we have removed container but we want to have our data.so what we can do iis we wil go with containers concept.

SO here we are going to store the container data into our host machene using volumes concept.

So to hold container data in my pwd in our host system going to create one folder and need to map with container data,.

mkdir /data

docker run –it - -rm –v /data:/data centos

we are now inside of container

vi index.html

exit

we are outside of container now and check for container it has been already removed

docker ps –a

cd /var/lib/docker/volumes/

ls –l

cd <volumeName>/_data

we can verify the data here.

This volume can be used to map with some another container that is the use case

If we want to create a new volume we just need to go to same location

And do the same command

docker volume ls

docker volume create <volumeName>

docker run –itd –v volname:/app Ubuntu:14.04

cd /data

we are able to see created file on our host system.

So that's how volumes are used for us.

Variables :

When ever we launch a container we can set some variables

docker run - -rm centos env

```
root@ubuntu:~# docker run --rm centos env
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=be9338078f44
HOME=/root
root@ubuntu:~#
```

There are only 3 variables one is path , hostname and home directory inside my container. If we want to set some variables to my container we have to use below command

docker run –e MYNAME=SIVA - -rm centos env

```
root@ubuntu:~# docker run --rm -e MYNAME=SIVA centos env
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=febdb7939d95
MYNAME=SIVA
HOME=/root
root@ubuntu:~#
```

Now we will take some database image

docker run –d mariadb

docker ps

docker ps –a

It has been exited why and what happened actually here just to check

docker logs <container id>

```
error: database is uninitialized and password option is not specified
  You need to specify one of MYSQL_ROOT_PASSWORD, MYSQL_ALLOW_EMPTY_PASSWORD and MYSQL_RANDOM_ROOT_PASSWORD
```

As per the output we have to pass the root password. So we can pass our root password here

docker run – d –e MYSQL_ROOT_PASSWORD=12345 mariadb

docker ps

it is still running

go and check logs now

docker logs <container_id>

So how to use this database

docker ps

fo this container ports numbers are assigned and mapped

3306/tcp

SO need to check what are ports numbers enabled on the server using below command

netstat -lntp

```
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN      1080/httpd
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      710/sshd
tcp        0      0 127.0.0.1:25            0.0.0.0:*               LISTEN      728/master
tcp6       0      0 :::22                   :::*                    LISTEN      710/sshd
tcp6       0      0 ::1:25                  :::*                    LISTEN      728/master
```

3306 is not enabled Ok.

Remove just all all the containers

docker rm <container_id>

SO what Is the right way to launch mairadb container use below command

docker run – d –e MYSQL_ROOT_PASSWORD=12345 –p 3306:3306 mariadb

Now again type same command and very enabled port numbers on this server.

netstat -lntp

```
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN      1080/httpd
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      710/sshd
tcp        0      0 127.0.0.1:25            0.0.0.0:*               LISTEN      728/master
tcp6       0      0 :::22                   :::*                    LISTEN      710/sshd
tcp6       0      0 ::1:25                  :::*                    LISTEN      728/master
tcp6       0      0 :::3306                 :::*                    LISTEN      1874/docker-proxy
```

Now 3306 port number is open on server.Now we can connecet to ths mysql database but we need client to connect.

apt-get install mariadb –y

so we are just installing a client to just connect a database.

For Ubuntu :

apt-get install mariadb-client

For redhat/sentos/fedora

yum install mariadb

After this we have to connect database via client

mysql - u root –p

```
root@ubuntu:~# mysql -u root -p
Enter password:
ERROR 2002 (HY000): Can't connect to local MySQL server through socket '/var/run
/mysqld/mysqld.sock' (2 "No such file or directory")
```

mysql –u root –p –h 192.168.224.156

and enter password

```
root@ubuntu:~# mysql -u root -p -h 192.168.224.156
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 8
Server version: 10.2.9-MariaDB-10.2.9+maria~jessie mariadb.org binary distributi
on

Copyright (c) 2000, 2017, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

Now we are able connect to the database.

Just exit after your work done.

Till now have done in my local system so if we want to connect same database from other container

docker run –it centos

so we will be inside of container

Then

yum install mariadb –y

```
[root@abedb97634f6 /]# mysql -u root -proot -h 10.128.0.6
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 3
Server version: 10.1.24-MariaDB-1~jessie mariadb.org binary distribution

Copyright (c) 2000, 2016, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> exit
Bye
```

Now we are able to connect mysql database from other container as well.

Here we know the password so if we don't know the the password we are unable to connect right but here docker provdes you one option to identify the root password using container id.

```
root@ubuntu:~# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED
     STATUS              PORTS                    NAMES
5b5f05b74b30        mariadb             "docker-entrypoint.sh"   7 minutes ago
     Up 7 minutes        0.0.0.0:3306->3306/tcp   evil_volhard
ab4f042b4cbd        mariadb             "docker-entrypoint.sh"   20 minutes ago
     Up 20 minutes       3306/tcp                 drunk_mcnulty
root@ubuntu:~# docker exec 5b5f05b74b30 env
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=5b5f05b74b30
MYSQL_ROOT_PASSWORD=12345
GOSU_VERSION=1.7
GPG_KEYS=199369E5404BD5FC7D2FE43BCBCB082A1BB943DB       430BDF5C56E7C94E848EE60C
1C4CBDCDCD2EFD2A        4D1BB29D63D98E422B2113B19334A25F8507EFA5
MARIADB_MAJOR=10.2
MARIADB_VERSION=10.2.9+maria~jessie
HOME=/root
root@ubuntu:~#
```

There is a conatainer environment variable called MYSQL_ROOT_PASSWORD variable.

docker rm $(docker ps –a -q)

docker run – d –e MYSQL_ROOT_PASSWORD=12345 –p 3306:3306 mariadb env

docker exec <container_id> env

Now we want to link the containers using link command to know the all details about other container

docker run – d –e MYSQL_ROOT_PASSWORD=12345 –p 3306:3306 - -name mariadb mariadb

Now the link command is

docker run –it - - link mariadb(container name):demo(new name) centos(new image name)

Now we are inside of container . So if we want to know the details about mariadb container just type the env command inside centos container.

env

then we are able to see demo_env_mysql_root_password=12345

Networks in Docker :

docker network ls

```
root@ip-172-31-7-15:~# docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
9fe5be21bc23        bridge              bridge              local
6048351e7acd        host                host                local
d288d2eb1d3e        none                null                local
root@ip-172-31-7-15:~#
```

docker network <network id>

example : docker network 9fe5be21bc23

To know about your network details we have to use above command

docker network --- it is going to be list all your commands of network command

IF we want to create new and our own network we have to use below command syantax

docker network create –subnet 10.0.0.0/24 openstack

```
root@ip-172-31-7-15:~# docker network create --subnet 10.0.0.0/24 openstack
c77af092aa4fb93dde7b1fcf63fd9e21e16cfc14dd60a77cdca5914316a58580
root@ip-172-31-7-15:~# docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
9fe5be21bc23        bridge              bridge              local
6048351e7acd        host                host                local
d288d2eb1d3e        none                null                local
c77af092aa4f        openstack           bridge              local
root@ip-172-31-7-15:~#
```

docker network inspect openstack

ip a

```
root@ip-172-31-7-15:~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9001 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 06:dd:19:c3:2c:7a brd ff:ff:ff:ff:ff:ff
    inet 172.31.7.15/20 brd 172.31.15.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::4dd:19ff:fec3:2c7a/64 scope link
       valid_lft forever preferred_lft forever
3: docker0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:c0:70:bc:28 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
       valid_lft forever preferred_lft forever
    inet6 fe80::42:c0ff:fe70:bc28/64 scope link
       valid_lft forever preferred_lft forever
7: vethd70a21a@if6: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master docker0 state UP group defau
lt
    link/ether 5a:86:6e:27:ce:0d brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet6 fe80::5886:6eff:fe27:ce0d/64 scope link
       valid_lft forever preferred_lft forever
8: br-c77af092aa4f: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether 02:42:c8:5e:57:c8 brd ff:ff:ff:ff:ff:ff
    inet 10.0.0.1/24 brd 10.0.0.255 scope global br-c77af092aa4f
       valid_lft forever preferred_lft forever
root@ip-172-31-7-15:~#
```

Now I want to create a my container on this newly created network.

Default is bridge network can take if you don't specify anything

docker run –it –name test –network openstack Ubuntu:14:04

```
root@ip-172-31-7-15:~# docker run -it --name test --network openstack ubuntu:14.04
root@879011cd1bf3:/# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
9: eth0@if10: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:0a:00:00:02 brd ff:ff:ff:ff:ff:ff
    inet 10.0.0.2/24 brd 10.0.0.255 scope global eth0
       valid_lft forever preferred_lft forever
root@879011cd1bf3:/#
root@879011cd1bf3:/# ping google.com
PING google.com (172.217.0.46) 56(84) bytes of data.
64 bytes from lga15s43-in-f46.1e100.net (172.217.0.46): icmp_seq=1 ttl=47 time=2.07 ms
64 bytes from lga15s43-in-f46.1e100.net (172.217.0.46): icmp_seq=2 ttl=47 time=2.14 ms
64 bytes from lga15s43-in-f46.1e100.net (172.217.0.46): icmp_seq=3 ttl=47 time=2.16 ms
64 bytes from lga15s43-in-f46.1e100.net (172.217.0.46): icmp_seq=4 ttl=47 time=2.37 ms
^C
--- google.com ping statistics ---
```

If we want to remove our network we have to stop all our running containers on this network and we have to stop or else give an error, it is not possible with out stopping containers.

```
root@ip-172-31-7-15:~# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS
         NAMES
879011cd1bf3        ubuntu:14.04        "/bin/bash"         About a minute ago  Up About a minute
         test
473377ea33ab        ubuntu:14.04        "/bin/bash"         34 minutes ago      Up 34 minutes
         server
root@ip-172-31-7-15:~# docker network rm openstack
Error response from daemon: network openstack id c77af092aa4fb93dde7b1fcf63fd9e21e16cfc14dd60a77cdca5914316a5858
0 has active endpoints
root@ip-172-31-7-15:~# docker network rm -f  openstack
unknown shorthand flag: 'f' in -f
See 'docker network rm --help'.
root@ip-172-31-7-15:~# docker stop 879011cd1bf3
879011cd1bf3
root@ip-172-31-7-15:~# docker network rm openstack
openstack
root@ip-172-31-7-15:~# docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
9fe5be21bc23        bridge              bridge              local
6048351e7acd        host                host                local
d288d2eb1d3e        none                null                local
root@ip-172-31-7-15:~#
```

Now network got deleted so we have one stopped container on this network so if we try to start again same container what will happen,

```
root@ip-172-31-7-15:~# docker start 879011cd1bf3
Error response from daemon: network c77af092aa4fb93dde7b1fcf63fd9e21e16cfc14dd60a77cdca5914316a58580 not found
Error: failed to start containers: 879011cd1bf3
root@ip-172-31-7-15:~#
```

Now how to resolve this particular issue.

Trying to connect the different network

```
root@ip-172-31-7-15:~# docker network connect bridge test
```

test -- my container name

bridge -- it is anetwork

then try to start container

```
root@ip-172-31-7-15:~# docker start 879011cd1bf3
Error response from daemon: network c77af092aa4fb93dde7b1fcf63fd9e21e16cfc14dd60a77cdca5914316a58580 not found
Error: failed to start containers: 879011cd1bf3
root@ip-172-31-7-15:~#
```

But we can identify same errorso we cant start a container with another network only thing we can delete this container and we have to create new container with bridge network.

How to assign multiple networks to a single container

```
ubuntu@ip-172-31-7-15:~$ docker network create --subnet 10.0.0.0/24 openstack
4063bdc21888ddc0fe17c96a97299ca557f5acc35fd7cd137ebbc49f1855a88f
```

Docker network ls

```
ubuntu@ip-172-31-7-15:~$ docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
9fe5be21bc23        bridge              bridge              local
5048351e7acd        host                host                local
d288d2eb1d3e        none                null                local
4063bdc21888        openstack           bridge              local
```

Docker network connect openstack <container id>

Docker attach <containerid>

Ip a

```
ubuntu@ip-172-31-7-15:~$ docker attach 473377ea33ab
root@docker:/#
root@docker:/# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
6: eth0@if7: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:ac:11:00:02 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.2/16 brd 172.17.255.255 scope global eth0
       valid_lft forever preferred_lft forever
14: eth1@if15: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:0a:00:00:02 brd ff:ff:ff:ff:ff:ff
    inet 10.0.0.2/24 brd 10.0.0.255 scope global eth1
       valid_lft forever preferred_lft forever
root@docker:/#
```

If you connect to multiple networks for single container no downtime and traffic can be splitted.

How to disconnect a network from a container

```
root@ip-172-31-7-15:~# docker network disconnect bridge server
root@ip-172-31-7-15:~# docker attach server
root@docker:/#
root@docker:/# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
14: eth1@if15: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:0a:00:00:02 brd ff:ff:ff:ff:ff:ff
    inet 10.0.0.2/24 brd 10.0.0.255 scope global eth1
       valid_lft forever preferred_lft forever
root@docker:/#
```

Docker network options

Ls

Inspect

Connect

Disconnect

Rm

Create

Prune --- IF you have many networks and in that we have not used few networks for any container so this case if you want to delete a un used networks we can use this command

docker network create –subnet 172.16.0.0/24 devops

docker network create –subnet 10.1.0.0/24 awsdDocker network ls

docker network prune

Or forcefully without asking confirmation

docker network prune –f

Removed all our un used networks from our docker engine.

How to start a container automatically when docker service restarted :

docker service status

docker run –itd ubuntu

docker ps

container is running now

lets say stopping our docker service

docker service stop

docker service status

docker service start

docker ps

docker ps –a

Here we can our container since after docker restart it has not been started automatically.

Now we have to start manually

docker start <container id> ----- this I don't want to do that

Why means if we have 1000 containers means we cant start start for 1000 times for 1000 containers right here.

docker run –itd –name container1 –hostname server2 –restart unless-stopped or always.

Only this policy can apply while starting container we cant do for existing containers.

Use a restart policy

To configure the restart policy for a container, use the `--restart` flag when using the `docker run` command. The value of the `--restart` flag can be any of the following:

Flag	Description
`no`	Do not automatically restart the container. (the default)
`on-failure`	Restart the container if it exits due to an error, which manifests as a non-zero exit code.
`unless-stopped`	Restart the container unless it is explicitly stopped or Docker itself is stopped or restarted.
`always`	Always restart the container if it stops.

The following example starts a Redis container and configures it to always restart unless it is explicitly stopped or Docker is restarted.

```
$ docker run -dit --restart unless-stopped redis
```

How to start multiple containers in a single line :

docker ps –aq

for I in 'docker ps –aq' ; do docker start $i ; done

How to remove all the containers which are available on my server :

docker ps –aq

for I in 'docker ps –aq'; do docker stop $i; do docker rm $i; done

docker ps –a

How to access running application inside a container :

docker run –it –name apacheserver –hostname webserver –p 3333:80 ubuntu

sudo apt-get update

sudo apt-get install apache2

service apache2 status

service apache2 start

go to browser type ip and with port number 3333

Hostmacheneip address with port number 3333

Above line we can use host ip address right but if want to bind any ip address that also can be possible like below this is real use case

```
docker run -ti --name testcase --hostname testserver -p127.0.0.1:8801:80 ubuntu:14.04
```

While running container we are giving ip address and port binding like above

If it is working everything we can check from terminal as well using curl command

curl –I 127.0.0.1:8081

```
HTTP/1.1 200 OK
Date: Thu, 28 Dec 2017 13:20:14 GMT
Server: Apache/2.4.7 (Ubuntu)
Last-Modified: Thu, 28 Dec 2017 13:19:13 GMT
ETag: "2cf6-561665cb17ea9"
Accept-Ranges: bytes
Content-Length: 11510
Vary: Accept-Encoding
Content-Type: text/html
```

200 means working

Other than 200 is not working.

All our data reg images and containers and volumes and everything about docker can be available at this location ----- /var/lib/docker

ls –l /var/lib/docker/containers

docker ps –a

both commands will give same out put reg our containers

Reg Images

```
root@ip-172-31-7-15:~# ls -l /var/lib/docker/image/overlay2/imagedb/content/sha256/
total 36
-rw------- 1 root root 12986 May 16 05:49 00b7c903b9e444750e6aeec9b8f6a2b3c0765e0fa663399f5212214877a62d2e
-rw------- 1 root root  3615 May 16 06:02 452a96d81c30a1e426bc250428263ac9ca3f47c9bf086f876d11cb39cf57aeec
-rw------- 1 root root  3615 May 19 23:51 8cef1fa16c778d84d2261587cbe812071df064f5a33a6e67f328ca6ada1dcd71
-rw------- 1 root root  2203 May 18 04:51 e934aafc22064b7322c0250f1e32e5ce93b2d19b356f4537f5864bd102e8531f
-rw------- 1 root root  8134 May 18 05:21 fb2f3851a97186bb0eaf551a40b94782712580c2feac0d15ba925bef2da5fc18
root@ip-172-31-7-15:~# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
jenkins             latest              00b7c903b9e4        5 days ago          696MB
httpd               latest              fb2f3851a971        2 weeks ago         178MB
ubuntu              14.04               8cef1fa16c77        3 weeks ago         223MB
ubuntu              latest              452a96d81c30        3 weeks ago         79.6MB
centos              latest              e934aafc2206        6 weeks ago         199MB
root@ip-172-31-7-15:~# ls -lh /var/lib/docker/image/overlay2/imagedb/content/sha256/
total 36K
-rw------- 1 root root  13K May 16 05:49 00b7c903b9e444750e6aeec9b8f6a2b3c0765e0fa663399f5212214877a62d2e
-rw------- 1 root root 3.6K May 16 06:02 452a96d81c30a1e426bc250428263ac9ca3f47c9bf086f876d11cb39cf57aeec
-rw------- 1 root root 3.6K May 19 23:51 8cef1fa16c778d84d2261587cbe812071df064f5a33a6e67f328ca6ada1dcd71
-rw------- 1 root root 2.2K May 18 04:51 e934aafc22064b7322c0250f1e32e5ce93b2d19b356f4537f5864bd102e8531f
-rw------- 1 root root 8.0K May 18 05:21 fb2f3851a97186bb0eaf551a40b94782712580c2feac0d15ba925bef2da5fc18
root@ip-172-31-7-15:~#
```

If we remove all our images containers can be crashed here.

-==========

Now we want to create our own image with the help of Docker file

Dockerfile

- This is a text file written docker container composition.
- We can create docker image using this file instead of commit command.
- We have some problems by commit command.
  - Manual operation to container setup
  - Write a document for composition separately
- But with Dockerfile, We can reuse a cache created at image creation.
  - When we build docker image using Dockerfile, Docker commits and caches the image of each step.

Dockerfile

- **FORMAT:**
  - INSTRUCTION Arguments
  - Not CASE-SENSITIVE, however in the convention
  - INSTRUCTION -> UPPERCASE
  - Arguments -> Lowercase

  - The first INSTRUCTION must be "FROM"
  - Docker will treat the lines that begin with # is a comment.
- **INSTRUCTIONS:**

  https://docs.docker.com/engine/reference/builder/

mkdir demo

cd demo

vi Dockerfile

FROM centos

:wq

docker build –t goldentech/demo .

docker images

I just created using base image without any additions

Now I want to install httpd on centos

Vi Dockerfile

FROM centos

RUN yum install httpd –y

:wq

Vi Dockerfile

FROM Ubuntu

MAINTAINER Siva

RUN apt update

RUN apt install git –y

RUN apt install wget tree zip unzip –y

RUN apt install apache2 –y

CMD /usr/bin/ apache2ctl -- the binaryfile have mention to start service in container like service

ENTRYPOINT - onl diff with CMD ENTRYPOINT has high priority while running in container

EXPOSE 80 - port opened inside container

:wq

docker build –t goldentech/demo .

docker build –t goldentech/demo:v1 .

docker login

user id : xxxxxxxxxx

password : xxxxxxxx

docker push goldentech/demo:v1

docker logout

Here base image is a Ubuntu if you delete Ubuntu here Ubuntu again has to pull from registry like docker hub so docker file having different layers with different images.

docker images

It has been created another image with installed httpd server.

So RUN is used to run some commands

docker ps -a

CMD is used to run default commands we use to run when launching a container

```
CONTAINER ID        IMAGE               COMMAND
d1d0009a2a7a        centos              "/bin/bash"
9d8634f6499e        mariadb             "docker-entrypoint..."
```

Like above /bin/bash all default commands will use CMD instruction to execute

Vi Dockerfile

FROM centos

RUN yum install httpd –y

CMD /bin/bash

:wq

docker build –t goldentech/demo .

docker run –itd goldentech/demo

docker ps –a

```
d1d0009a2a7a        centos              "/bin/bash"
9d8634f6499e        mariadb             "docker-entrypoint..."
```

Vi Dockerfile

FROM centos

RUN yum install httpd –y

CMD /usr/sbin/httpd –D FOREGROUND

:wq

docker build –t goldentech/demo .

docker run –itd goldentech/demo

Now we can check command in container.

docker ps –a

```
COMMAND                  CREATED             STATUS                  PORTS
"/bin/sh -c '/usr/..."   2 seconds ago       Up 2 seconds
"/bin/sh -c /bin/bash"   59 seconds ago      Up 59 seconds
```

So to specify the default command here we are going to use CMD command n docker file.

LABEL:

The Label instruction will adds the meta data to an image.

LABEL

```
LABEL <key>=<value> <key>=<value> <key>=<value> ...
```

The `LABEL` instruction adds metadata to an image. A `LABEL` is a key-value pair. To include spaces within a `LABEL` value, use quotes and backslashes as you would in command-line parsing. A few usage examples:

```
LABEL "com.example.vendor"="ACME Incorporated"
LABEL com.example.label-with-value="foo"
LABEL version="1.0"
LABEL description="This text illustrates \
that label-values can span multiple lines."
```

Vi Dockerfile

FROM centos

RUN yum install httpd –y

CMD /usr/sbin/httpd –D FOREGROUND

LABEL version=”1.0”

:wq

docker build –t goldentech/demo .

docker run –itd goldentech/demo

When you run these containers if you want to check about the labels we can use below command

docker inspect <container_id>

```
"ArgsEscaped": true,
"Image": "sha256:98b46071a866dade67b5abad6f8cd610a5e0742da6b9d57d8cd9fcc1817190ee",
"Volumes": null,
"WorkingDir": "",
"Entrypoint": null,
"OnBuild": [],
"Labels": {
    "build-date": "20170510",
    "license": "GPLv2",
    "name": "CentOS Base Image",
    "vendor": "CentOS",
    "version": "1.0"
}
```

It is only for understanding pupose so nothing is technical here. If you want to put some data or details about some image means we can use this LABEL command.

Next one is MAINTAINER :

MAINTAINER (deprecated)

```
MAINTAINER <name>
```

The `MAINTAINER` instruction sets the *Author* field of the generated images. The `LABEL` instruction is a much more flexible version of this and you should use it instead, as it enables setting any metadata you require, and can be viewed easily, for example with `docker inspect`. To set a label corresponding to the `MAINTAINER` field you could use:

```
LABEL maintainer "SvenDowideit@home.org.au"
```

This will then be visible from `docker inspect` with the other labels.

EXPOSE

```
EXPOSE <port> [<port>...]
```

The `EXPOSE` instruction informs Docker that the container listens on the specified network ports at runtime. `EXPOSE` does not make the ports of the container accessible to the host. To do that, you must use either the `-p` flag to publish a range of ports or the `-P` flag to publish all of the exposed ports. You can expose one port number and publish it externally under another number.

To set up port redirection on the host system, see using the -P flag. The Docker network feature supports creating networks without the need to expose ports within the network, for detailed information see the overview of this feature).

Now

vi Dockerfile

```
FROM centos
RUN yum install httpd -y
CMD /usr/sbin/httpd -D FOREGROUND
LABEL version="1.0"
EXPOSE 80
~
```

:wq

docker build –t goldentech/demo .

docker run –itd goldentech/demo

docker ps –a

Now I can see port Number for running container like below

COMMAND	CREATED	STATUS	PORTS	NAMES
"/bin/sh -c '/usr/..."	5 seconds ago	Up 4 seconds	80/tcp	goofy_golick

Here for this continer port number 80 is enabled using Dockerfile.

But it is really not opening to the server why because we have not mapped it.

SO when ever you run this image it is our responsibility to open the port to the server.

So I have to use below command for this

docker run –itd –p 80:80 goldentech/demo

```
2ee19bbec75cf50cbf748f38cd49b82ef3044792395a6801e0ba1d9da3dcf20b
docker: Error response from daemon: driver failed programming external connectivity on endpoint nervous_panini (cc658f18942c43daadfd3cc61a8a70d55
75d0a3a0ff940a71c7e55821b326083): Error starting userland proxy: listen tcp 0.0.0.0:80: bind: address already in use.
[root@docker1 demo]#
```

But we are getting an error, error is address is already used that means you already started web server inside your host system.

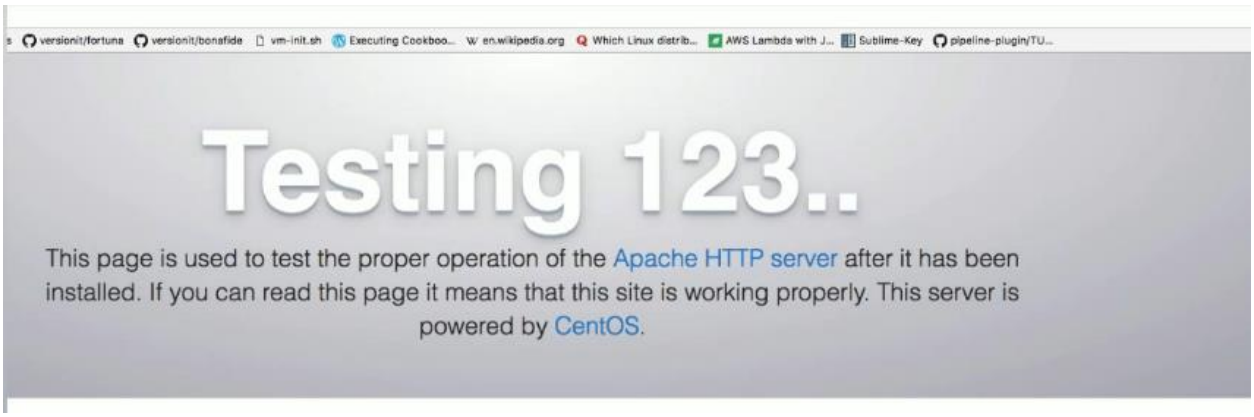
systemctl stop apache2

docker run –itd –p 80:80 goldentech/demo

Now the container can be created with above command.

And now go to the browser and take the ip address of the host system and paste browser and hit enter.

So we are going to get some message and this message we are getting from container not from the server.



Now we will work on ENV command inside Dockerfile

ENV

```
ENV <key> <value>
ENV <key>=<value> ...
```

The `ENV` instruction sets the environment variable `<key>` to the value `<value>`. This value will be in the environment of all "descendant" `Dockerfile` commands and can be replaced inline in many as well.

The `ENV` instruction has two forms. The first form, `ENV <key> <value>`, will set a single variable to a value. The entire string after the first space will be treated as the `<value>` - including characters such as spaces and quotes.

The second form, `ENV <key>=<value> ...`, allows for multiple variables to be set at one time. Notice that the second form uses the equals sign (=) in the syntax, while the first form does not. Like command line parsing, quotes and backslashes can be used

ENV why we are going to use means, when we want to share some variables or we want to share some variables in the image itself am going with ENV instruction.

Vi DOckerfile

FROM centos

RUN yum install httpd –y

CMD /usr/sbin/httpd –D FOREGROUND

LABEL version="1.0"

EXPOSE 80

ENV MYNAME=siva

:wq

Before building this image we just see the what are environement variables are available here.

docker run - -rm goldentech/demo env

```
[root@docker1 demo]# docker run --rm versionit/demo env
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=0e2bb6bcc17e
HOME=/root
```

Able to see only these environment variables.

Now am going to build new image with ENV instruction.

docker build –t goldentech/demo .

Now again we are going to run the newly created image.

docker run - -rm goldentech/demo env

Now we are able to see new environment variable all MYNAME

```
root@ubuntu:~/demo# docker build -t goldentech/demo .
Sending build context to Docker daemon 2.048 kB
Step 1 : FROM centos
 ---> 196e0ce0c9fb
Step 2 : RUN yum install httpd -y
 ---> Using cache
 ---> 6bcb28a05170
Step 3 : CMD /usr/sbin/httpd -D FOREGROUND
 ---> Running in 76dee8b46f9f
 ---> da7219a09db1
Removing intermediate container 76dee8b46f9f
Step 4 : LABEL version "1.0"
 ---> Running in 53e606558628
 ---> a538bf088dc4
Removing intermediate container 53e606558628
Step 5 : EXPOSE 80
 ---> Running in d2580e1eca3a
 ---> 6d601f18827e
Removing intermediate container d2580e1eca3a
Step 6 : ENV MYNAME siva
 ---> Running in 7da32d3746c9
 ---> ecaa6deaf670
Removing intermediate container 7da32d3746c9
Successfully built ecaa6deaf670
root@ubuntu:~/demo# docker run --rm goldentech/demo env
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=ae3a489ec6c4
MYNAME=siva
HOME=/root
root@ubuntu:~/demo#
```

This variable is coming from image so we can put variables inside of image by using ENV option.

ADD

ADD has two forms:

- `ADD <src>... <dest>`
- `ADD ["<src>",... "<dest>"]` (this form is required for paths containing whitespace)

The `ADD` instruction copies new files, directories or remote file URLs from `<src>` and adds them to the filesystem of the image at the path `<dest>`.

Multiple `<src>` resource may be specified but if they are files or directories then they must be relative to the source directory that is being built (the context of the build).

Each `<src>` may contain wildcards and matching will be done using Go's filepath.Match rules. For example:

```
ADD hom* /mydir/        # adds all files starting with "hom"
ADD hom?.txt /mydir/    # ? is replaced with any single character, e.g., "
```

COPY

This instruction is used to copy files and directories from a specified source to a destination (in the file system of the container).

Example:

```
COPY preconditions.txt /usr/temp
```

ADD

This instruction is similar to the COPY instruction with few added features like remote URL support in the source field and local-only tar extraction. But if you don't need an extra feature, it is suggested to use COPY as it is more readable.

Example:

```
ADD http://www.site.com/downloads/sample.tar.xz /usr/src
```

ADD and COPY both are same only the difference is ADD can have url but in COPY it can't be added.

Now here we are going to create one file and paasing to image.

vi ndex.html

<h1>Hello from Docker</h1>

:wq

vi Dockerfile

FROM centos

RUN yum install httpd –y

CMD /usr/sbin/httpd –D FOREGROUND

LABEL version="1.0"

EXPOSE 80

ENV MYNAME=siva

ADD index.html /var/www/html

:wq

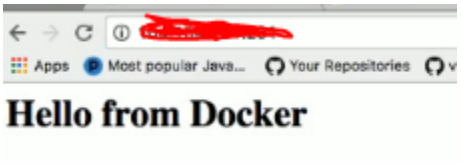
So we have created one in same folder and that same file we are going copy into image and putting into webserver folder to access.

docker build –t goldentech/demo .

docker run - -rm –itd –p 80:80 goldentech/demo

Now if we go to browser and type our ip and just enter

We are able to see output of index file since we kept inside of web server on our host.



So if you want to copy some files along with your image qe can go with ADD option.

Source should be URL also some http url

COPY also does the same thing here.

ENTRYPOINT :

ENTRYPOINT

ENTRYPOINT has two forms:

- `ENTRYPOINT ["executable", "param1", "param2"]` (*exec* form, preferred)
- `ENTRYPOINT command param1 param2` (*shell* form)

An `ENTRYPOINT` allows you to configure a container that will run as an executable.

For example, the following will start nginx with its default content, listening on port 80:

```
docker run -i -t --rm -p 80:80 nginx
```

Command line arguments to `docker run <image>` will be appended after all elements in an *exec* form `ENTRYPOINT`, and will override all elements specified using `CMD`. This allows arguments to be passed to the entry point, i.e., `docker run <image> -d` will pass the `-d` argument to the entry point. You can override the `ENTRYPOINT` instruction using the `docker run --entrypoint` flag.

Just remove all containers now.

docker rm $(docker ps –a –q)

docker run –itd goldentech/demo sleep 5000

And it is running now.

docker ps -a

Now we can see the command calles “sleep 5000”

```
root@ubuntu:~/demo# docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED
     STATUS              PORTS               NAMES
64f5ecf379ad        goldentech/demo     "sleep 5000"             5 seconds ago
     Created                                 prickly_lovelace
c5e4c70b2701        goldentech/demo     "/bin/sh -c '/usr/sbi"   8 minutes ago
     Created                                 silly_jang
22015c799fa9        goldentech/demo     "/bin/sh -c '/usr/sbi"   8 minutes ago
     Created                                 thirsty_kirch
root@ubuntu:~/demo#
```

vi Dockerfile

FROM centos

RUN yum install httpd –y

ENTRYPOINT /usr/sbin/httpd –D FOREGROUND

LABEL version=”1.0”

EXPOSE 80

ENV MYNAME=siva

ADD index.html /var/www/html

:wq

I just removed CMD and replaced with ENTRYPOINT. If we give any default command in Dockerfile it will not get replace with parameter commands while running the container.

```
FROM centos
RUN yum install httpd -y
ENTRYPOINT /usr/sbin/httpd -D FOREGROUND
LABEL version=”1.0”
EXPOSE 80
ENV MYNAME=siva
ADD index.html /var/www/html
```

```
root@ubuntu:~/demo# docker run -itd goldentech/demo sleep 5000
b7938f9f7cab1b6700d9585a4059414689b07716f8a8663ee166b7ad693bd416
root@ubuntu:~/demo# docker run -itd goldentech/demo
4ef3cbed38135d4efa95132852526039a7774bb8e7dc2b5620666b4d28add559
root@ubuntu:~/demo# docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED
     STATUS                      PORTS               NAMES
4ef3cbed3813        goldentech/demo     "/bin/sh -c '/usr/sbi"   6 seconds ago
     Exited (1) 3 seconds ago                        elegant_leavitt
b7938f9f7cab        goldentech/demo     "sleep 5000"             56 seconds ag
     Up 54 seconds               80/tcp              small_mclean
ebc80ede1c9d        goldentech/demo     "sleep 5000"             2 minutes ago
     Created                                         grave_payne
64f5ecf379ad        goldentech/demo     "sleep 5000"             8 minutes ago
     Created                                         prickly_lovelace
c5e4c70b2701        goldentech/demo     "/bin/sh -c '/usr/sbi"   15 minutes ag
     Created                                         silly_jang
22015c799fa9        goldentech/demo     "/bin/sh -c '/usr/sbi"   16 minutes ag
     Created                                         thirsty_kirch
root@ubuntu:~/demo#
```

SO here CMD will allow us to give any other command work on default command where as entry point wont allow us to do that.If you really need more details on this use inspect command.

docker inspect <container id>


```
[
    {
        "Id": "45eea619362ea01161a4a6a9a2093e63e5cd1d8bd17554417a8f57f87d534547",
        "Created": "2017-06-03T02:57:56.297816999Z",
        "Path": "/bin/sh",
        "Args": [
            "-c",
            "/usr/sbin/httpd -D FOREGROUND",
            "sleep",
            "5000"
        ],
        "State": {
            "Status": "running",
            "Running": true,
            "Paused": false,
            "Restarting": false,
            "OOMKilled": false,
            "Dead": false,
            "Pid": 4030,
            "ExitCode": 0,
            "Error": "",
            "StartedAt": "2017-06-03T02:57:56.512308527Z",
            "FinishedAt": "0001-01-01T00:00:00Z"
```


VOLUME :

VOLUME

```
VOLUME ["/data"]
```

The `VOLUME` instruction creates a mount point with the specified name and marks it as holding externally mounted volumes from native host or other containers. The value can be a JSON array, `VOLUME ["/var/log/"]` , or a plain string with multiple arguments, such as `VOLUME /var/log` or `VOLUME /var/log /var/db` . For more information/examples and mounting instructions via the Docker client, refer to *Share Directories via Volumes* documentation.

The `docker run` command initializes the newly created volume with any data that exists at the specified location within the base image. For example, consider the

vi Dockerfile

FROM centos

RUN yum install httpd –y

CMD /usr/sbin/httpd –D FOREGROUND

LABEL version=”1.0”

EXPOSE 80

ENV MYNAME=siva

ADD index.html /var/www/html

VOLUME /data

:wq

docker build –t goldentech/demo .

docker run –it --rm goldentech/demo ls

docker run –it --rm goldentech/demo ls /data

User :

USER

```
USER daemon
```

The `USER` instruction sets the user name or UID to use when running the image and for any `RUN`, `CMD` and `ENTRYPOINT` instructions that follow it in the `Dockerfile`.

Docker run - - rm <image_name> id

If we use id command what is the default id it is taking just look at below picture

```
root@ubuntu:~/demo# docker run --rm goldentechhyd234 id
uid=0(root) gid=0(root) groups=0(root)
root@ubuntu:~/demo#
```

Vi Dokcerfile

Update with USER value

```
FROM centos
RUN yum install httpd -y
ENTRYPOINT /bin/bash
EXPOSE 80
ENV MYNAME=siva
ADD index.html /var/www/html
USER daemon
~
```

:wq

```
root@ubuntu:~/demo# docker build -t goldentechhyd234 .
Sending build context to Docker daemon 3.072 kB
Step 1 : FROM centos
 ---> 196e0ce0c9fb
Step 2 : RUN yum install httpd -y
 ---> Using cache
 ---> 65f536a2e28b
Step 3 : ENTRYPOINT /bin/bash
 ---> Running in cecfdca30bb5
 ---> 0ac4524375f0
Removing intermediate container cecfdca30bb5
Step 4 : EXPOSE 80
 ---> Running in 3a3c278a3162
 ---> b69b3e48e93b
Removing intermediate container 3a3c278a3162
Step 5 : ENV MYNAME siva
 ---> Running in ecd91a93f17c
 ---> d5a5b0815891
Removing intermediate container ecd91a93f17c
Step 6 : ADD index.html /var/www/html
 ---> 3a7267238d00
Removing intermediate container 73cec00a7928
Step 7 : USER daemon
 ---> Running in a8ebc92cb55f
 ---> a22e7a9453ac
Removing intermediate container a8ebc92cb55f
Successfully built a22e7a9453ac
```

Docker build –t <imagename> .

Docker run -it <imagename> id

So here we will get Docker file user name now.

Type : id

WORKDIR :

Vi Dockerfile

WORKDIR /tmp

Docker build –t <image_name> .

Docker run –it <image_name>

Then type

Pwd

It will display /tmp

ARG :

Suppose the dockerfile if we dont have access to edit and if we want to edit with our specific user then we will go with ARG.

Whenever if we want to pass the dynamic users to container or dynamic values to containers we have to use ARG instruction.

vi Dockerfile

ARG user

USER $user

:wq

Docker build - -build-arg user=daemon –t <image_name> .

Docker run –it <image_name>

Type : id

You are able to see the given user

```
[root@145497e0f0a8 tmp]# exit
exit
root@ubuntu:~/demo# docker build --build-arg user=daemon -t goldentechhyd23456 .

Sending build context to Docker daemon 3.072 kB
Step 1 : FROM centos
 ---> 196e0ce0c9fb
Step 2 : RUN yum install httpd -y
 ---> Using cache
 ---> 65f536a2e28b
Step 3 : ENTRYPOINT /bin/bash
 ---> Using cache
 ---> 0ac4524375f0
Step 4 : EXPOSE 80
 ---> Using cache
 ---> b69b3e48e93b
Step 5 : ENV MYNAME siva
 ---> Using cache
 ---> d5a5b0815891
Step 6 : WORKDIR /tmp
 ---> Using cache
 ---> 1c241198159f
Step 7 : ARG user
 ---> Using cache
 ---> 1329610ccf51
Step 8 : USER $user
 ---> Using cache
 ---> 1b991204976e
Successfully built 1b991204976e
root@ubuntu:~/demo# docker run -it goldentechhyd23456
bash-4.2$ id
uid=2(daemon) gid=2(daemon) groups=2(daemon)
bash-4.2$
```

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. To learn more about all the features of Compose, see [the list of features](#).

Installation on Ubuntu :

sudo apt-get install docker-compose

File format :

Vi stack.yml

```
version: '2'
services:
  web:
    image: tomcat
  db:
    image: mariadb
```

:wq

docker-compose up

docker-copmose down

DB will not start here why means database is looking for root password.

```
Status: Downloaded newer image for tomcat:latest
Creating stack_web_1
Creating stack_db_1
Attaching to stack_web_1, stack_db_1
db_1    | error: database is uninitialized and password option is not specified
db_1    |   You need to specify one of MYSQL_ROOT_PASSWORD, MYSQL_ALLOW_EMPTY_PASSWORD and MYSQL_RANDOM_ROOT_PASS
WORD
```

Now we have to pass paasword to db image like below

```
version: '2'
services:
  web:
    image: tomcat
  db:
    image: mariadb
    environment:
      - MYSQL_ROOT_PASSWORD=root
```

docker-compose up

Now Database can also start. SO both are giving parallel but both are running parallel. This is not the right way to start with multiple containers.

If we need any help reg docker-compose

docker-compose –help

docker-compose ps --- for checking running containers

docker-compose logs

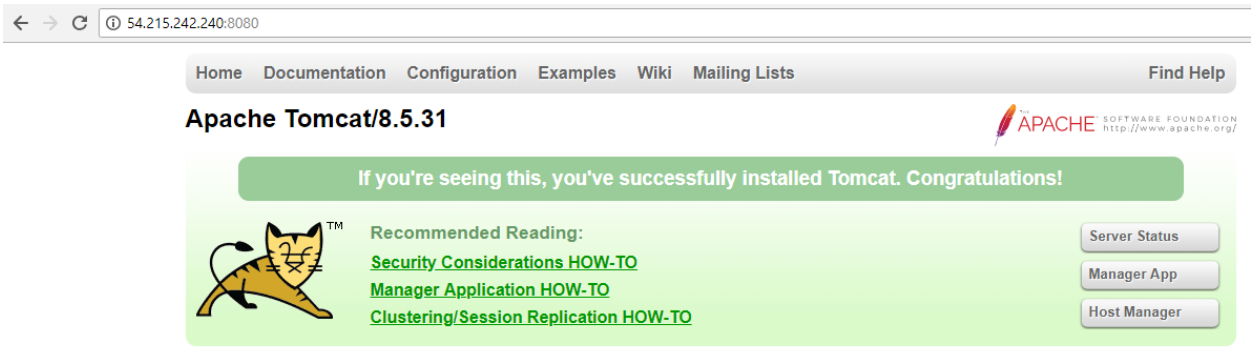
or If you want to follow logs same like docker engine

docker-compose logs -f

If you want to enable ports on images like below syntax can do

```
version: '2'
services:
  web:
    image: tomcat
    ports:
      - 8080:8080
  db:
    image: mariadb
    environment:
      - MYSQL_ROOT_PASSWORD=root
```

Now am able to see my Tomcat homepage at this url



So like this we can create and run multiple containers and we can create our own images and and we can insert into our docker compose yml file.

Create a new directory in same directory where we have docker-compose file we have.

```
root@ip-172-31-7-15:~/stack# mkdir web
root@ip-172-31-7-15:~/stack# ls
docker-compose.yaml  web
root@ip-172-31-7-15:~/stack#
```

```
root@ip-172-31-7-15:~/stack# cd web
root@ip-172-31-7-15:~/stack/web# vi Dockerfile
root@ip-172-31-7-15:~/stack/web# ls
Dockerfile
root@ip-172-31-7-15:~/stack/web# cat Dockerfile
FROM tomcat
ADD https://github.com/sivakethineni/studentapp /usr/local/tomcat/webapps/

root@ip-172-31-7-15:~/stack/web#
```

So this is my own image and this is I want to use in my docker-compose file where I need tomcat server like below

```
version: '2'
services:
  web:
    build: web/.
    ports:
      - 8080:8080
  db:
    image: mariadb
    environment:
      - MYSQL_ROOT_PASSWORD=root
```

Instead of using direct image here am building my own image.

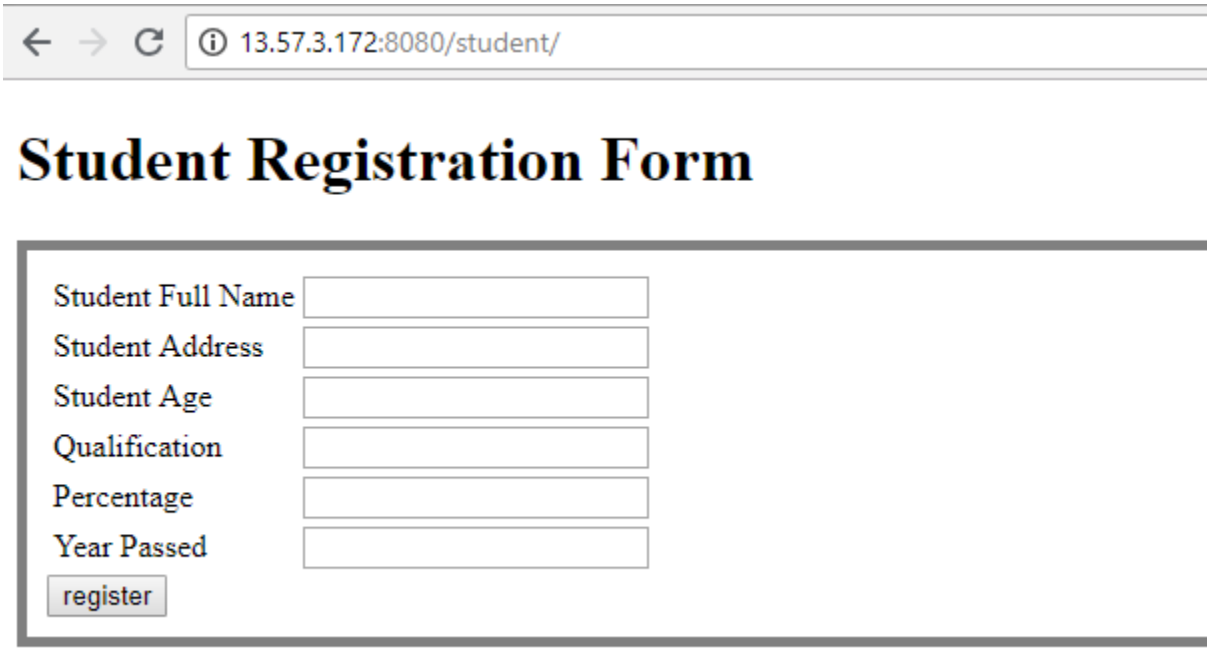
Here one more thing the new image can download into our docker engine

docker images

It can be available

Now We are going to the connection between Tomcat and database using our github example

git clone https://github.com/sivakethineni/stack-docker.git
there is one more dependency to run this example from git hub
git clone https://github.com/sivakethineni/studentapp.git Student app war file and db config files to get
sudo spt-get install docker-compose
cd stack-docker
docker-compose up
docker-compose ps
then go to browser and type ip with 8080 and enter details like below



Enter details and say register with below button

Register Student

Students List

Student ID	StudentName	Student Addrs	Student Age	Student Qualification	Student Percentage	Student Year Passed	Edit	Delete
2	admin	hyderabad	28	mca	100	2010	edit	delete

Here we can delete existing data and we can modify existing data as well into our app.