DOCKER

By : Shubham Meshram

1 . What is Docker?

Ans : Docker is an Containerization tool which is used to create , run and deploy an application by using containers. Early, When a developer creates an application and run in its own system then it works fine but when the same application is send to an production server or testing servers it was not runs properly. So to overcome this DOCKER comes in picture. Docker helps us to create , run an application and store them in container and sends them in the Container. So it will runs properly.

DOCKERFILE & DOCKER IMAGES

2. What is DockerFile & how can we create and build it ?

Ans : Dockerfile is a textfile which is created by developers which is used to build an docker image. Dockerfile having a set of instructions to build an image like FROM, CMD , ENV , RUN etc.

Steps: 1. Create a dockerfile with name “dockerfile”.

Steps: 2. Write an instructions like:

```
FROM Ubuntu

MAINTAINER shubham shubmesh5@gmail.com

RUN apt-get update

CMD [“Hello”]
```

Steps: 3. Write a following command to build an image :

```
# docker build –t myimage:1.0
```

3. What is Images & how they can be created ?

Ans : Images are nothing but a packages that contains everything that your application needs to run.

OR

Images are the templetes which is used to create a container.

Images can be created in local or remote location.

Single image can be used to create multiple containes.

Images can be created by using dockerfile or by pulling from Dockerhub.

4. What is Container and how they can be created ?

Ans : Container is an running instances of docker images.

They can be created by running docker images.

Docker container can be created by following ways :

```
# docker run --name MyShub Ubuntu
```

or

```
# docker container create --name MyShub hello-world
```

Where , Myshub is Container name and Ubuntu, hello-world are image names.

5. What is DockerHub ?

Ans : It is a Centralized repository for a Docker, where anyone can store and retrieve the docker images from anywhere, anytime efficiently.

6 . How run image in specific port like 8080?

Ans : Syntax : # docker run –p [port number] [image name]

Example : # docker run –p 8080:80 tutum/hello-world

# docker run –p 8080:80 jenkins

7. What are Docker Layers. Explain them.

Ans : A Docker image consists of several layers. Each layer corresponds to certain instructions in your Dockerfile.

Docker layers are read only layers in which you can't modify layers.

The following instructions create a layer: RUN, COPY, ADD. The other instructions will create intermediate layers and do not influence the size of your image.

FROM : FROM must be the first non-comment instruction in the Dockerfile. It creates base image.

FROM <image_name>

MAINTAINER : The MAINTAINER instruction allows you to set the Author field of the generated images.

MAINTAINER <person_name>

RUN : This command is used for installing the packages, Run any linux commands ,Create users , Overwrite content.

RUN <command>

CMD : which can be overwritten from command line when docker container runs.

CMD <command>

EXPOSE : Informs Docker that the container listens on the specified network port(s) at runtime.

EXPOSE <port>

LABEL : The LABEL instruction adds metadata to an image.

LABEL <key>=<value>

ENV : The ENV instruction sets the environment variable <key> to the value <value>.

ENV <key> <value>

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

ADD <src> [<src> ...] <dest>

COPY : Copies new files or directories from <src> and adds them to the filesystem of the image at the path <dest>.

COPY <src> [<src> ...] <dest>

ENTRYPOINT : Allows you to configure a container that will run as an executable.

ENTRYPOINT ["<executable>", "<param1>", "<param2>"]

WORKDIR : Sets the working directory for any RUN, CMD, ENTRYPOINT, COPY, and ADD instructions that follow it.

WORKDIR </path/to/workdir>

8. How can we download Official images from Dockerhub ?

Ans : By using following docker command :

Syntax : # docker pull <Image_ name>:<Version>

ex : # docker pull nginx :1.0

Where, version is optional. By default, it takes "latest" version.

9. How to change tag of local docker images ?

Ans : Syntax : # docker tag <source_image> <dest_image>

Ex : # docker tag alpine:1.0 alpine:2.0

In above ex, alpine:2.0 image is created from alpine:1.0

10. How to remove an docker image which is present local repository ?

Ans : Syntax : # docker rmi <image_name>

Or

# docker rmi <image_id>

Ex : 1. # docker rmi alpine:1.0

2. # docker rmi f70734b6a266

11. How to create Docker container in foreground and in background ?

Ans : For Foreground :

Syntax : 1. # docker run --name <Container_name> <Image_name>

2. # docker run <Image_name>

Ex: 1. # docker run --name Mycont nginx:alpine

2. # docker run nginx:alpine

In above ex1, the container name is optional. If we don't give any name then the docker will automatically give random name to a container.

If you run a container in foreground mode, your CMD will stuck, so you will not able to do another process until you kill it.

For Background :

Syntax : 1. # docker run –d --name <Container_name> <Image_name>

2. # docker run –d <Image_name>

Ex: 1. # docker run –d --name Mycont nginx:alpine

2. # docker run –d nginx:alpine

In above ex, -d is used for detached mode. i.e for background

12. How to remove the running or stopped container?

Ans : We can delete/remove the container forcefully by using –f.

Syntax: # docker rm -f <Container_name>

Ex : # docker rm –f Mycont

And Another way is to stop the container first and the we can delete the container normally.

Syntax : # docker rm <Container_name>

Ex : # docker rm Mycont

13. Is the image will be delete when it is attached with running container?

Ans : Yes , but it is deleted when you forcefully delete the image by using –f.

Syntax : # docker rmi –f <image_name>

Ex : # docker rmi –f nginx:alpine

Where, -f is used to delete forcefully and rmi is used to remove image. But we can delete the image without forcefully by stopping the container first later deleting the image as follows :

Syntax : # docker rmi <Image_name>

Ex : # docker rmi nginx:alpine

14. Can we delete Paused Container ?

Ans : No, We cannot delete the paused container normally. We have to first stop the container and the we have to delete the container. But, we can forcefully delete the Paused container by using –f.

Syntax : # docker rm -f <Container_name>

Ex : # docker rm –f nginx:alpine

15. How to delete multiple containers at a single command ?

Ans : Syntax : # docker rm <Container1_name> , <Container2_Name>

Ex : # docker rm Mycont1 , Mycont2

16. Explain Docker start , stop, pause , unpause , remove , kill in Docker Container.

Ans : Lets take Ex : # docker run –d --name Mycont –p 9090:80 nginx:alpine

In above example, it shows the container "Mycont" is created and running on exposing 9090 port as external and 80 port as internal.

This can be seen by command : # docker ps and Running and Stopped by command : #docker ps –a

STOP : This is used to stop the running container only. When the container is stopped it will not expose 9090 port. It is free to use by other container.

To stop the running container we use following command :

# docker stop <Container_name/ContainerID>

# docker stop 43fc569cb2f9 or # docker stop Mycont

You cannot see stopped container by using # docker ps command instead you can see stopped container by using # docker ps –a command.

START : This is used when the container is stopped and you want to run the stopped Container. If You start the container again it will expose the same port that were using before stopped.

While if the Container is in stopped state and Other container used that port and then you restart the Container again the it will not get the same port instead it will show the error of "Port number already allocated to other Container".

To start the stopped container we use following command :

# docker start <Container_name/ContainerID>

# docker start 43fc569cb2f9 or # docker start Mycont

You can see the Restarted Container by using # docker ps command.

PAUSE : This is used to temperory pause the container. When the container is in paused state, the Port number is still allocated/exposed to Paused container. If other Container want to run on the same port number, they are not able to use same port number.

To pause the running container we use following command :

```
# docker pause <Container_name/ContainerID>
```

```
# docker pause 43fc569cb2f9      or      # docker pause Mycont
```

You can see the paused container by using # docker ps command. You will see the status as paused on the container list.

UNPAUSE : This is used to unpause/start the paused container. After Unpausing the container, it will exposed/allocated the same port number that it has already allocated.

To unpause the paused container we use following command :

```
# docker unpause <Container_name/ContainerID>
```

```
# docker unpause 43fc569cb2f9      or      # docker unpause Mycont
```

You can see Unpaused container by using # docker ps command.

REMOVE : This is used to remove/delete the container permanently. After removing/deleting the container, it will release the port and made available to anyone.

To delete/remove the running/stopped container we use following command :

```
# docker rm <Container_name/ContainerID>
```

```
# docker rm 43fc569cb2f9      or      # docker rm Mycont
```

You cannot see the deleted container anywhere.

KILL : This is used to kill the container but it will not release the port until we removed it.

To kill the running container we use following command :

```
# docker kill <Container_name/ContainerID>
# docker kill 43fc569cb2f9      or      # docker kill Mycont
```

17. How to build image from a dockerfile ?

Ans : Syntax : # docker build --tag <image_name>:<tag_name> .

Or

```
# docker build --tag <image_name>:<tag_name> <dockerfile_name>
```

```
Ex1 : # docker build --tag centos_apache:v1 .
Ex2 : # docker build --tag centos_apache:v1 –f dockerfile1  .
Ex3 : # docker build –t cent_apache:v1 .
Ex3 : # docker build –t cent_apache:v1 –f dockerfile1    .
```

In above ex, dot (.) is used at last because to use the dockerfile present in the current directory. Or we can give directly dockerfile name.

18. If we have a lot of Docker images are there and we want to show only specific images with specific name. How we can do this?

Ans :

EX :

```
 REPOSITORY          TAG              IMAGE ID          CREATED           SIZE
  centos_apache        v1               5aa0febd92a5       12 min ago       279MB
     nginx            alpine          89ec9da68213       5 days ago         19.9MB
       alpine           latest           f70734b6a266       5 days ago         5.61MB
     centos            latest          470671670cac       3 months ago        237MB
   hello-world         latest          bf756fb1ae65       3 months ago        13.3kB
tutum/hello-world   latest          31e17b0746e4       4 years ago         17.8MB
```

We have the above images are present. And we want to show only the list of images with named ”centos” we use following command :

Syntax : # docker images | grep <Image_name>

Ex : # docker images | grep centos

This will show only two images with named centos

19. What commands are need to start the docker daemon ?

Ans : Step1 : # sudo su

# yum install docker –y

# service docker start

20. How Dockerfile searched by Docker ?

Ans : After creating a dockerfile in a specific folder , the docker search for a file which having name by default “dockerfile/Dockerfile”.

If you have given a dockerfile name as “dockerfile/Dockerfile” then it will search for the folder in which the dockerfile is present or you can specifydot (.) at last while building the image.

For Ex : # docker build –t centos_ apache **.**

The above example shows that the docker file is named as “dockerfile/Dockerfile” and it is present in current folder.

If you have given a dockerfile name as “Docfile” or anyname then it will search for the folder in which the dockerfile is present. but the docker will not find a dockerfile. So we have to externally provide a name of dockerfile like ”Docfile” and the path of the dockerfile location.

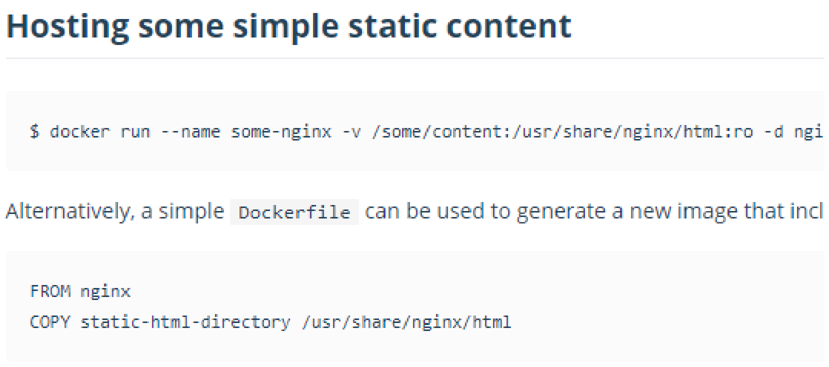
For Ex1 : # docker build –t centos_ apache –f Docfile **.**

Ex2 : # docker build –t centos_apache –f Docfile ./home/user/shubham

In Above Ex, -f is used to give the name of file.

21.How to change the Content of a images by using dockerfile?

Ans : Step1 : See the file location of image where the content is present .i.e index.html file in images.



Hosting some simple static content

```
$ docker run --name some-nginx -v /some/content:/usr/share/nginx/html:ro -d ngi
```

Alternatively, a simple `Dockerfile` can be used to generate a new image that incl

```
FROM nginx
COPY static-html-directory /usr/share/nginx/html
```

In above image, The Static content of Nginx is present in /usr/share/nginx/html folder.

So, Now create a dockerfile and write following :

FROM nginx:alpine

RUN echo “Hello Shubham…!!!” > /usr/share/nginx/html/index.html

Save this and after building the image, run the container and expose the port. This will change the contents of file.

22. How to use FROM in dockerfile with example?

Ans : FROM nginx:alpine

FROM Ubuntu

In above example, it shows that which type of base image you want to use. If you use Centos then you have to use centos commands, if you use Ubuntu the you have to use Ubuntu commands etc.

23. How to use RUN in dockerfile with example?

Ans :

FROM nginx:alpine

RUN echo “shubhu!! Changing the Content” > /usr/share/nginx/html/index.html

CMD apachectl –DFOREGROUND

The above dockerfile example shows that, by using RUN we are changing the Content of an image.

By using RUN command we can create users, run any linux commands , Change the contents, Unzip the files etc.

24. How to use CMD in dockerfile with example?

Ans :

FROM nginx:alpine

RUN echo “shubhu!! Changing the Content” > /usr/share/nginx/html/index.html

CMD apachectl –DFOREGROUND

In above ex, CMD is very useful layer because if we didn’t use the CMD in dockerfile the the container gets exited because it will not have any process after running the container.

Above examp,e shows we use apacahectl _-DFOREGROUND to start the apache process in foreground after running the container.

25. How to use COPY in dockerfile with example?

Ans :

FROM centos

RUN yum –y install httpd

COPY index.html /var/www/html

CMD apachectl –DFOREGROUND

The above exaple shows that the copy command is used to copy the code from local computer/DockerHost and move to the /var/www/html   folder to host.

Always remember, the folder, files, images etc you are copying must be in the folder you kept your dockerfile.

26. How to use ADD in dockerfile with example?

Ans :

FROM centos

RUN yum –y install httpd

ADD https://github.com/mdn/beginner-html-site-scripted/archive/gh-pages.zip /var/www/html/code.zip

CMD apachectl –DFOREGROUND

The above example shows that the folder we are copied/downloaded in /var/www/html folder with renaming as “code.zip” from “gh-pages.zip” by using ADD layer. This workd similarly like COPY but it download files from Online.

27. How to use ENV in dockerfile with example?

Ans :

FROM centos

RUN yum –y install httpd

ENV HTML beginner-html-site-scripted

ADD https://github.com/mdn/$HTML/archive/gh-pages.zip /var/www/html/code.zip

RUN echo $HTML > /var/www/html/env/html

CMD apachectl –DFOREGROUND

In above ex, it shows that “beginner-html-site-scripted” is replaced with “$HTML”. ENV helps us to store an value in key.

28. How to use WORKDIR in dockerfile with example?

Ans :

```
FROM centos

RUN yum –y install httpd

ENV HTML beginner-html-site-scripted

WORKDIR   /var/www/html/

ADD https://github.com/mdn/$HTML/archive/gh-pages.zip   ./code.zip

RUN echo $HTML > ./env.html

CMD apachectl –DFOREGROUND
```

The above example shows that, The WORKDIR is used to show the current path we are using. Instead of give path again and again we use WORKDIR and instead of that path we give “ ./ ”.

29. How to use LABEL in dockerfile with example?

Ans :

```
FROM centos

RUN yum –y install httpd

LABEL maintainer=shubham

LABEL vendor=Linsyssoft technologies

CMD apachectl –DFOREGROUND
```

The above ex shows that the LABELS are just like tags which are used to just show who is maintainer , vendor etc.

Most commonly used LABELS are maintainer and vendor.

30. How to enter/execute into the Running Container ?

Ans : Syntax : # docker exec –ti <Container_name/ContainerID> bash

Syntax : # docker exec –it <Container_name/ContainerID> bash

When you do :

```
[root@ip-172-31-87-46 ec2-user]#  docker exec -ti Cont1 bash
                              OR
[root@ip-172-31-87-46 ec2-user]#  docker exec -it Cont1 bash
```

Output : [root@7f3aabc10e68 /]#

The above example shows, “exec” is used to execute the container and “-it” or ”-ti” is used for interactive terminal/ terminal interactive which is basically to give interactive input and “bash” is used to Terminal type.

When we go to the container we can see all the things present in that container bu using commands like PWD ,LS , etc.

31. If we have some files in running container and we delete that container and again create and run the same container. So the Previous data is present in that container or not?

Ans : Yes, When we create a container, all the information are present in the image. So the data remains same and You want to permanently change the data in the container then change the you need to modify the image. So we cannot change the image directly, so we have to first change the dockerfile and build the image and run the container again.

32. How to use USER in dockerfile with example?

Ans :

```
FROM centos

RUN yum –y install httpd

ENV HTML beginner-html-site-scripted

WORKDIR   /var/www/html/

USER application

CMD apachectl –DFOREGROUND
```

In above ex, it shows that the defaulkt user is “root” so it is having the permission to delete, add, modify the content. But when we create a user by using USER then only the user is created but it will not have the permission the delete, modify or add anything. It can be changed by changing the owernership by using Chown.

33. How to use ARG in dockerfile with example?

Ans :

```
FROM centos

RUN yum –y install httpd

RUN useradd application && chown application:application  /var/www/html –R

CMD apachectl –DFOREGROUND
```

So We use ARG to provide arguments/parameters to a variable like :

```
FROM centos

ARG user=application

RUN yum –y install httpd

RUN useradd $user && chown $user:$user  /var/www/html –R

CMD apachectl –DFOREGROUND
```

And we have to pass “--build-arg   user=shubham” at last while building the dockerfile like:

```
# docker build –t myimg:v1 –f DocFile --build-arg  user=shubham .
```

34. How to use CMD in dockerfile with example?

Ans :

```
FROM centos

RUN yum –y install httpd

CMD apachectl –DFOREGROUND
```

The above example shows that, the CMD is used to start the apache process in the foreground.

The process is very important while running the docker container. If we don’t write CMD then the docker will get exited automatically.

35. How to Run the container without CMD layer in Dockerimage?

Ans : If we don’t add CMD in the Dockerfile and build the image from that dockerfile. Then when we run the container it gets exited.

So to avoid this we use following command while running the container :

```
# docker run –d –it --name MyCont –p 9091:80  myimg:v1
```

36. What is Build Context ?

Ans : The docker build command builds Docker images from a Dockerfile and a "context".

A build's context is the set of files located in the specified PATH or URL.

The build process can refer to any of the files in the context. For example, your build can use a COPY instruction to reference a file in the context.

We specify the context by using dot (.) or by giving the path of Dockerfiles etc.

Ex : # docker build --tag myimg:v1 .

Ex : # docker build --tag myimg:v2 /docker-images

In above ex, dot (.) and /docker-images both are contexts.

37. What is Docker ignore or .dockerignore file?

Ans : The docker ignore file is used to ignore the files from the context that we will mention in " .dockerignore " file.

If we have dockerfile and other files in the same folder and we don't require to execute the files except the dockerfile then write the files, folders names in the .dockerignore file by creating it.

Suppose, We have 3 files or folders : dockerfile , Pictures_my , Videos_my

And we want to execute the dockerfile only by building the dockerfile with the help of context dot (.) then copy and paste name of folder/files named Pictur es_my , Videos_my in .dockerignore file.

While building the dockerfile, docker will firstly search for .dockerignore file.

38. What is Dangling image?

Ans : Suppose we are creating one image from a dockerfile and build it as named "myimg:v1" and then we modify something in dockerfile and again we build image as same name as earlier “myimg:v1” so the old image will show as “<none>” that image is known as dangling image.

```
Ex :   FROM alpine
       RUN echo shubham
       RUN echo mayur
```

```
# docker build --tag myimg:v1 .
```

O/P:

```
REPOSITORY        TAG          IMAGE ID            CREATED           SIZE
 myimg            v1           hf424ff224c42       2 seconds ago     240 KB
```

And then we again modify the dockerfile and build again with same name :

```
       FROM alpine
       RUN echo shubham1
       RUN echo mayur1
```

```
# docker build --tag myimg:v1 .
```

O/P:

```
REPOSITORY        TAG          IMAGE ID            CREATED           SIZE
 myimg            v1           hf424ff224c42       2 seconds ago     240 KB
<none>          <none>        5js6jsb7aknd99       1 minutes ago     340 KB
```

The lower image is the older image that we have created and that image is known dangling image.

39. How to remove dangling images ?

Ans : We can remove dangling images one by one by using their id. Like this :

Syntax : # docker rmi <image_id>

Ex : # docker rmi 9ds89897sd97

But if we have a lot have dangling images and we want to delete all the dangling images at one click the do this following :

Ex : # docker rmi $(docker images –f dangling=true -q)

-f is used to filter the images by giving the attributes.

To see all the dangling images with their id then we use –q to show the id.

# echo $(docker images –f dangling=true -q)

We always delete the dangling images because it takes a lot of memory/space in the docker without any use.

40. What is fallocate command ? how can we see the size of a file? What is grep command?

Ans : Fallocate is a command used to create a file with a specific size.

Like, if we want to create a file with name "shubham" with size 10MB we do the following:

# fallocate –l 10MB shubham

And To check the size of file "shubham" we use following command :

# du –sh shubham

And To check the size of files present in current directory we use following command :

# du –shc *

The GREP command is used to filter the result by giving parameter.

Like if we want to search the images with name alpine we do following :

```
# docker images | grep alpine
```

41. What is Multi-stage build in docker ?

Ans : Multi-stage build in docker allows you to retrieve a specific file from a previous stage.

Stage 1 is stared from 1st FROM to 2nd FROM.

Stage 2 is stared from 2nd FROM to 3rd FROM and later on…

Ex :

```
FROM centos as build

RUN fallocate –l 10MB /opt/file1

RUN fallocate –l 10MB /opt/file2

RUN fallocate –l 10MB /opt/jar
```

The above example shows that the centos basic image size is around 200MB and we created 3 files with 10MB per. So the total size will be around 230MB while building the image from this dockerfile. so the image will be so big in size.

So to avoid this we use Multistage in which we create 2nd FROM. And we use the light weight base image as alpine which is around 5MB in size. And then we copy the specific file from above stage 1 i.e from 3rd RUN and copied to other folder. So when we build the image from this dockerfile, only the size of 10MB image will be build. Bcoz, Docker will only run from the 2nd FROM i.e last stage.

```
FROM centos as build

RUN fallocate –l 10MB /opt/file1

RUN fallocate –l 10MB /opt/file2

RUN fallocate –l 10MB /opt/jar

FROM alpine
```

COPY --name=build /opt/jar /dest/my.jar

DOCKER CONTAINER

42. What are two commands to show the running containers ?

Ans : 1. # docker ps

2. # docker container ls

43.What are two commands to run the containers?

Ans : 1. # docker run <image_name>

2. # docker container create <image_names>

44.How to rename a container?

Ans : Syntax : # docker rename <old container name> <new container name>

Ex : # docker rename Shubham Mayur

45.Explain –p 9090:8080?

Ans : The "–p" or "--publish" is use to expose the port. The First port no i.e 9090 is the docker host port and the Second Port i.e 8080 is the container port.

The docker host port is given by a developer and the container port is already given by the pubilisher while creating the image.

The docker host port forward internally to container port. We have to access the docker host port only. By giving the IP or Localhost like :

https://localhost:9090   or https://192.168.767.78:9090

46.How create environment variable at runtime without using dockerfile?

Ans :# docker run –d --name Env –e “name=shubhu” –e “sname=meshram”env:v1

The above ex shows that we can use “-e ” to pass an environment variable at runtime. Suppose if we create an environment variable “sname” already in dockerfile then the dockerfile env variable is override with the run time.

47.How to Check the logs of Container?

Ans : # docker logs <container_name>

This is use to check the logs. i.e what the things are done in background like errors , downloads etc.

48.How to check the total RAM and free RAM available in our host machine?

Ans : # free –h

49.How to check the resources are using by the container?

Ans : # docker stats <container_name/ContainerID>

50.How to limit the memory usage by the container?

Ans : # docker run –d --name Mycont –memory “200MB” Jenkins:v1

So above ex shows that the memory usage limit can be set by using “--memory”.

51.How to copy a file from a Docker host/Local Computer to a running container?

Ans :

Syntax : # docker cp <file_name> <Container_name>:<path_of_destination>

Ex : # docker cp test.html Mycont:/usr/local/apache2/htdocs/index.html

In above ex, The file “ test.html ” is taken from the current directory bcoz we didn’t specify the full path and copied to the path “ /usr/local/apache2/htdocs/” and replaced with index.html to test.html.

52.How to copy a file from a running container to a Docker host/Local Computer?

Ans :

Syntax : # docker cp <Container_name>:<path_of_sourcefile> <Dest_path>

Ex : # docker cp Mycont:/usr/local/apache2/htdocs/index.html .

In above ex, the destination path is given as dot (.) means current directory. So the file index.html will be copy to the current directory of the docker host.

53.How to turn a running container into a Image ?

Ans :

Syntax: # docker commit <Container_name/ContainerID> <Resulting_Imgname>

Ex: # docker commit MyCont1 resimage:v1

The above ex shows that the commit is used to convert running container to an image. The “resimage:v1” is an name of resulting image created by container.

When the changes is done in dockerfile the same changes will reflect to an image an then to container. But when the changes is done directly in running container and if the container removed, then the changes also gets deleted. To avoid this, the best practice is to make a changes in the dockerfile or the modified container converts to an image.

54.How override the CMD without using a Dockerfile when the container is running?

Ans : Lets run a centos image container bcoz it doesn't have any valid CMD so it will going to die/exited automatically. So we will use following command to make him alive :

# docker run –dti --name Mycont1 centos:v1

This will create a container but it will only make him alive I,e running but it will not help us to expose a port bcoz it does not have any process running.

So to override the existing CMD of running centos container we do the following :

Syntax : # docker run –d --name <Cont_name/ID> <img_name> python –m SimpleHTTPServer 8080

# docker run –d --name Mycont2 centos:v1 python –m SimpleHTTPServer 8080

In above ex, the "Mycont2" is a Container name , "centos:v1" is an image name and "python –m SimpleHTTPServer 8080" is a CMD command we are providing externally to override.

The lines after the image name are consider as the CMD.

55.How to delete a container automatically after we exit out of container?

Ans : Syntax : # docker run --rm –ti <image_name> bash

Ex : # docker run --rm –ti nginx:alpine bash

In the above ex, "--rm" is used to remove the container when we exited.

Actually, the container we create is basically a temporary container. It is used basically to when we are downloading some files from internet , or to check the path, or when you want to run something temporary.

56.How to check the location where all the images , containers , all the docker stuff are present & change the location/path?

Ans : All the docker stuff , images, containers are present in a “ /var/lib/docker”

To check the location do the following :

# docker info | grep –i root

It will display the Docker root directory path i.e “ /var/lib/docker”.

DOCKER VOLUME

57.What is Docker Volume?

Ans : Docker volume is a space where the container additional data is saved. Lets take an example, Suppose you created MYSQL image container and after that you want to create a new database and after creating a database you create tables and store a data in database. But when by some reason, the container dies i.e removed or stopped then the temporary data will gets deleted.

So to avoid this the Docker volumes comes in picture. You just have to create a Docker volume and you have to run the container in that volume so that if your container dies the data will remain same as it is.

58.How to create a Docker Bind Volume?

Ans : We can create a simple docker volume by following command :

Syntax : # docker run –d –v <docker_host_path>:<container_volume_path> --name <container_name> -e <environment_variables> <image_name>

Ex : # docker run –d –v /mnt/mysql:/var/lib/mysql --name Mydoc –e “MYSQL_ROOT_PASSWORD=12345678” mysql:5.7

In above example, “-v “ is used to create a docker volume.

The Docker host path is “ /mnt/mysql “ in which we want to save the data.

 The Docker volume path is "/var/lib/mysql " in which the data is actually going to save when we create a container and the " -e " is use to create an environment variable at runtime.

This shows that, when we create a data in container then the data is stored in docker volume path and the same data gets reflected in the docker host path.

This created volume are called as BIND volumes.

59.How to restore data from a volume to container?

Ans :

Syntax : # docker run –d –v <docker_host_path>:<container_volume_path> --name <container_name> -e <environment_variables> <image_name>

Ex : # docker run –d –v /mnt/mysql:/var/lib/mysql --name Mydoc –e "MYSQL_ROOT_PASSWORD=12345678" mysql:5.7

The above ex shows that the data which is present in the docker host path gets automatically restored in the docker volume path which we have mentioned.

The docker volume and the docker container is sync to each other. Means if we store a data in docker host path then the data get reflected in the docker volume path.

This created volume are called as BIND volumes.

60.How to a normal volumes / volumes?

Ans : We can create a normal volume by following command :

Syntax : # docker create volume <volume_name>

Ex : # docker create volume my_volume

We can the the volume we have created by using following command :

# docker volume ls

In above example , we created a normal volume by giving name as “my_volume”. This volume is created in “ /mnt/docker/volumes ”. You can check where the volume is stored by following command :

# docker info | grep –I Docker root Dir

Now to use this volume to store that data from container or to the container from volumes we use following command as we used earlier but having some minor change that we have to give only volume name instead of docker host path. We have already created a volume with name “ my_volume”.

Syntax : # docker run –d –v <volume_name>:<container_volume_path> --name <container_name> -e <environment_variables> <image_name>

Ex : # docker run –d –v /my_volume:/var/lib/mysql --name Mydoc –e “MYSQL_ROOT_PASSWORD=12345678” mysql:5.7

61.How to get into MySql container?

Ans :

Syntax : # docker run –d --name <container_name> -e <environment_variables> <image_name>

Ex : # docker run –d --name Mysql –e “MYSQL_ROOT_PASSWORD=12345678” mysql:5.7

There is a compulsory to pass an environment variable while creating mysql container.

Now we have to go into mysql container my using following command :

Syntax : # docker exec –ti <Container_name> bash

Ex : # docker exec –ti Mysql bash

By using above command we can go into the mysql container but we cannot use querries for creating tables. So to do this use following command :

- mysql –u root –p12345678

Now we can use Mysql to create databases , tables etc.

62.How to create Anonymous volume?

Ans : Anonymous volume is volume which does not have its name instead its name is given by docker randomly. We can create Anonymous volume by following command :

Syntax : # docker run –d --name <container_name> -e <environment_variables> -v <container_volume_path> <image_name>

Ex : # docker run –d --name Mysql –e "MYSQL_ROOT_PASSWORD=12345678" –v /var/lib/mysql mysql:5.7

In above ex, we provide –v to create Anonymous volumes to a docker container volume path i.e /var/lib/mysql

Now to see this we can see by using # docker volume ls command.

The Anonymous volumes gets deleted when you write this command to delete a container :

Syntax : # docker rm –fv <container_name>

Ex : # docker rm –fv Mysql

But, the Anonymous volumes will not get deleted when you write this command to delete a container :

Syntax : # docker rm –f <container_name>

Ex : # docker rm –f Mysql

The difference is that, if you use “-fv ” (i.e forcefully remove container and volume) to remove the container then the The Anonymous volumes gets deleted. & if you use “-f ” (i.e forcefully remove container) to remove the container then the The Anonymous volumes will not gets deleted.

63.Difference between Anonymous volume and Normal volume?

Ans : 1. When we create the container with associated with volumes and we try to delete the container and volume forcefully by using “ –fv ” the Anonymous volume gets deleted but the Normal volume well remain as it is.

2. The Anonymous volume did not have any name so it is hard to understand what type of data is stored in it but in case of normal volume, we have the name of volume so that we can easily understand the type of data stored in that volume.

64.How to create an Anonymous volume using Dockerfile?

Ans :

```
FROM centos

VOLUME  /opt
```

Now if you build the image with this dockerfile and then run the container with that image the anonymous volumes gets created.

65.What is Dangling Volumes and How they can be removed?

Ans : Dangling volumes are volumes that are not associated with containers.

Lets take an example, Suppose we create 3 container along with anonymous volumes like :

```
# docker run –dti –v /mnt --name Mycont1 mysql:3.7

# docker run –dti –v /mnt --name Mycont2 mysql:3.7

# docker run –dti –v /mnt --name Mycont3 mysql:3.7
```

Output :

```
DRIVER          VOLUME NAME
local          4be393eb0a496ad05aede93c91f1ad1f3f0c1a0629a7901b97bc691153a5a58c
local          6a710863fd894cbefceccff9dd2e417d81ff861dfe6883fb11eac08f3c478c27
local          855fcad3b43df2da5bab2f64767367b55ecca9d716a1228c64bd19f48b4fe899
```

Now remove all 3 containers only,

```
#docker rm –f Mycont1 , Mycont1 , Mycont1
```

Now the volume which were associated with the containers are now dangling because we just now deleted containers.

Now we can check dangling images with following command :

```
# docker volume ls –f dangling=true –q
```

Now, we have to delete these volumes by following command :

```
#docker volume rm $( docker volume ls –f dangling=true –q )
```

This will remove all the dangling volumes present in the docker.

DOCKER NETWORK

66.What is Docker Network and their Types?

Ans : Networks in docker are way useful to provide network connection between containers. It having 3 types :

1. Bridge Network
2. Host Network
3. None Network

67.How to ping the containers using IP addresses?

Ans : Firstly we have to create two containers :

# docker run –dti –name Cont1 centos

# docker run –dti –name Cont2 centos

Now lets ping the containers:

Syntax : #docker exec Cont1 bash –c " ping <IP aaddress of Container 2> "

Synatax : #docker exec Cont2 bash –c " ping <IP aaddress of Container 1> "

But we don't have IP address of both the containers . So to get the IP address we do the following command :

# docker network ls

You will see 3 networks i.e Bridge , Host , None. Normally we use Bridhe network to communicate. So now do following command :

#docker inspect bridge

Now it will show the running container with their IP address. Container1 having Ip address as 172.17.0.2 and Container2 172.17.0.3 resp. Now we get IP address of Containers. Now we can communicate/ ping to each other :

Ex : #docker exec Cont1 bash –c " ping <172.17.0.3 > "

Ex : #docker exec Cont2 bash –c " ping <172.17.0.2> "

68. How to Create our own Networks and How to delete them?Can we delete Default networks?

Ans :

Syntax : # docker network create –d <type_of_network/Driver> --subnet <subnet_range> --gateway <gateway> <network_name>

Ex : # docker network create –d bridge --subnet 172.18.0.0/16 --gateway 172.18.0.1 my_network

In the above ex, -d is used to specify the Driver type i.e Bridge/Host/None , --subnet is used to specify IP ranges, and the --gateway is used to provide IP address.

Now to delete the network we use following command :

Synatx : # docker network rm <network_name>

Ex : # docker network rm my_network

Note That we can only delete the networks we have created but we cannot delete the default network which is created by docker.

69.How to attach containers to our own network?

Ans :

Syntax : # docker run –dti --network <custom_network_name> --name Cont1 centos

Ex : # docker run –dti --network my_network -- name Cont1 centos

Ex : # docker run –dti --network my_network --name Cont2 centos

In above, We created 2 container which is attached to same custom network i.e my_network .

70.What is difference between Default network (Created by Docker) and The Custom Network (Created by Us)?

Ans : 1. In default network, the IP address , subnet , gateway etc are default created by docker but In custom network, the IP address , subnet , gateway etc are always created by us.

2. The main difference is when we try to ping a containers by using default network, we must require IP address of each container but when we try to ping a containers by using custom network, we did not require IP address instead we can call directly by using Container Name.

For Default Network :

Syntax : #docker exec Cont1 bash –c " ping <IP aaddress of Container 2> "

Synatax : #docker exec Cont2 bash –c " ping <IP aaddress of Container 1> "

Ex : #docker exec Cont1 bash –c " ping <172.17.0.3 > "

Ex : #docker exec Cont2 bash –c " ping <172.17.0.2> "

For Custom Network :

Syntax : #docker exec Cont1 bash –c " ping <Name_of_container2 > "

Synatax : #docker exec Cont2 bash –c " ping < Name_of_container2 > "

Ex : #docker exec Cont1 bash –c " ping Cont2 "

Ex : #docker exec Cont2 bash –c " ping Cont1 "

71.How to communicate containers in different networks?

Ans : Lets create a network.

Syntax : # docker network create –d <type_of_network/Driver> --subnet <subnet_range> --gateway <gateway> <network_name>

Ex : # docker network create –d bridge --subnet 172.30.0.0/16 --gateway 172.30.0.1 net1

Ex : # docker network create –d bridge --subnet 172.31.0.0/16 --gateway 172.31.0.1 net2

Now lets create a container and attach them to different network.

# docker run –dti –name net1_cont --network net1 centos

# docker run –dti –name net2_cont --network net2 centos

Now to connect each containers which is in different network we do the following command :

Syntax : # docker network connect <network_name> <container_name>

Ex : # docker network connect net1 net2_cont

Now, we can ping different container

Ex : #docker exec net1_cont bash –c " ping net2_cont "

Now if you want to disconnect the containers which is in different network we can do following command :

Syntax : # docker network disconnect <network_name> <container_name>

Ex : # docker network disconnect net1 net2_cont

72.How to assign IP address to a Container?

Ans : We can assign IP address to a container which is present in our ouwn network i.e Custom network.

So just lets create our custom network :

# docker network create –d bridge --subnet 172.40.0.0/16 --gateway 172.40.0.1 test_network

Now create a container in a custom network with assigning our own IP address :

# docker run –dti --network test_network --ip 172.40.0.100 --name Mycont centos

Now to check the IP address do

# docker inspect Mycont

73.How to create Host Network?

Ans : # docker run --rm –ti --network host centos bash

This will create a Host network. The host network will have the same details as the Docker Host have like DNS , HOSTNAME , IP etc. only the different is the " user ". In docker Host the user is "centos" but in Host network the user is "root".

74.How to create None Network?

Ans : # docker run –dti --name Mycont --network none centos bash

This will create a None network. The none network does not have any gateway, Ip address , etc. To check this you can inspect the container.

The None network did not have any driver. It having Null.

DOCKER-COMPOSE

75.What is docker-compose in Docker?

Ans : Docker-compose is a utility or a tool which allows you to create multi container applications.

Lets take an example , Suppose you want to create a web application. So you will need a web server which can have apache or nginx and a database . So you can done this by a single container but you will need to use a lot of flags , lot of environment variables, volumes etc to execute it. So it will be every complicated. So the best thing is you can create two different containers, One for Webserver and other for database server. So this can be done by docker-compose.

You just need to create a simple text file and you have to define all the configurations of your container.

76.How to create a nginx container using docker-compose?

Ans : Firstly create a docker-compose file with name “ docker-compose.yml ”

After that write the following code /command :

```
version:  '3'
services:
  web:
    container_name: nginx_compose
    ports:
        -  "9090:80"
    image: nginx:alpine
```

Now to run the docker-compose do the following command :

```
#docker-compose up –d
```

So it will run the nginx container by using docker-compose.

Now to delete the container created by docker compose we use following command :

```
# docker-compose down
```

77.How to use Environment variable in Docker-compose for mysql Container?

Ans : Firstly create a docker-compose file with name “ docker-compose.yml ”

After that write the following code /command :

```
version:  '3'
services:
  database:
    container_name: mysql_compose
    ports:
        -  "3306:3306"
    image: mysql:5.7
    environment:
        -MYSQL_ROOT_PASSWORD:123456
```

Now to run the docker-compose do the following command :

```
#docker-compose up –d
```

So it will run the mysql container by using docker-compose.

OR

You have to create a file with any name and any extension in the current directory like “ environmentdata.txt “ and put the environment variable in “ environmentdata.txt “ file which is require for Mysql container :

```
MYSQL_ROOT_PASSWORD=123456
```

Now, create a docker-compose file with name “ docker-compose.yml ”

After that write the following code /command :

```
version:  '3'
services:
  database:
    container_name: mysql_compose
    ports:
        -  "3306:3306"
    image: mysql:5.7
    env_file: environmentdata.txt
```

The env_file is use to write an environment variables in a different file and just add the path of file to invoke the environment variables.

Now to delete the container created by docker compose we use following command :

```
# docker-compose down
```

78.How to run docker-compose if we have name of yaml file as docker-compose-mysql.yaml?

Ans : Its very simple, do the following command :

```
#docker-compose –f docker-compose-mysql.yaml up –d
```

And, to delete these file use following command :

```
 #docker-compose –f docker-compose-mysql.yaml down –d
```

79.How to create a bind volume using docker-compose?

Ans :

```
version:  '3'
services:
  database:
    container_name: mysql_compose
    ports:
        -  "3306:3306"
    image: mysql:5.7
    volumes:
        -    “/mnt/mysql:/var/lib/mysql”
```

80.How to create a normal volume using docker-compose?

Ans :

```
version:  '3'
services:
  database:
    container_name: mysql_compose
    ports:
        -  "3306:3306"
    image: mysql:5.7
    volumes:
        -   "mysql_volume:/var/lib/mysql"
volumes:
    mysql_volume:
```

This way you can create a normal volume. You can check by using following command :

```
# docker volume ls
```

You will see the name of volume will be : docker-compose_mysql_volume

81.How to use networking in docker-compose to ping containers?

Ans : For that we will need to create 2 containers. So it can be created by docker-compose easily using following code/command :

```
version:  '3'
services:
  web:
    image: centos
    container_name: Cont1
    networks:
        -   test_network
    tty: true
  db:
    image: centos
    container_name: Cont2
    networks:
        -   test_network
     environment:
        -MYSQL_ROOT_PASSWORD:123456

    tty: true
networks:
    test_network:
```

Now to ping use the command that we used earlier :

```
#docker exec –ti Cont1 bash –c "ping Cont2"
#docker exec –ti Cont1 bash –c "ping db"
#docker exec –ti Cont2 bash –c "ping Cont1"
#docker exec –ti Cont2 bash –c "ping web"
```

We used " tty:true " in replace of " –ti " we used in command to make a container running state.

82.How to build images using docker-compose using dockerfile named Dockerfile?

Ans : To build an image using docker-compose we need a dockerfile with name " dockerfile "and must be save in the same directory and write some code :

```
FROM centos

RUN mkdir /opt/test
```

Now create a docker-compose file and write the following :

```
version: ‘3’

services:

  web:

      image: my_image

      container_name: Mycont

      build: .
```

Now, write the following command to build the image :

```
#docker-compose build
```

83. How to build images using docker-compose using dockerfile named Dockerfile2?

Ans : To build an image using docker-compose we need a dockerfile with name “ Dockerfile2 ”and must be save in the same directory and write some code :

```
FROM centos

RUN mkdir /opt/test
```

Now create a docker-compose file and write the following :

```
version: ‘3’
services:
  web:
     image: my_image
     container_name: Mycont
     build:
        context: .
        dockerfile: Dockerfile2
```

Now, write the following command to build the image :

```
#docker-compose build
```

84. How to build images using docker-compose using dockerfile named Dockerfile2 which is in different directory?

Ans : : To build an image using docker-compose we need a dockerfile with name “ Dockerfile2 ”and if it is present in “ build “directory and write some code :

```
FROM centos
RUN mkdir /opt/test
```

Now create a docker-compose file and write the following :

```
version: '3'
services:
  web:
     image: my_image
     container_name: Mycont
     build:
         context: build
         dockerfile: Dockerfile2
```

Now, write the following command to build the image :

```
#docker-compose build
```

85.How to override the CMD in image using docker-compose?

Ans :

```
version: '3'
services:
  web:
    image: centos
    container_name: Mycont
    command: python –m SimpleHTTPServer
```

And now build the image using : # docker-compose build

86.How to Push our image to DockerHub?

Ans : step 1 : Run a container using that image.

```
# docker run –dti --name Mycontainer myimg:v1
```

Step 2 : Commit the Image.

```
# docker commit Mycontainer shubmesh5/myimg:v1
```

Step 3 : Push The image.

```
# docker push shubmesh5/myimg:v1
```

87.How to Start the Docker in Linux?

Ans : Step 1 : # yum install docker –y

Step 2 : # service docker start