All the commands we learnt till date are adhoc commands.

In the previous usecase we have installed two containers( chrome and firefox )

Letsw say you need 10 containers?

Do we need to run 10 commands?

Instead of 80 commands, we can use docker compose

+++++++++++++++++++++++++++++++++++++++++++++++

Docker compose

This is a feature of docker using which we can create multicontainer architecture using yaml files. This yaml file contains information about the containers that we want to launch and how they have to be linked with each other. Yaml is a file format. It is not a scripting language.

Yaml will store the data in key value pairs

Lefthand side – Key

Righthand side – Value

Yaml file is space indented.

**Sample Yaml file**

---

logiclabs:

trainers:

sunil: Devops

raj: Python

Coordinators:

lakshmi: Devops

rani: AWS

...

++++++++++++++++++++++

logiclabs -- root element

+++++++++++++++++++++++++

To validate the abvove Yaml file

Open http://www.yamllint.com/

Paste the above code -- Go button

++++++++++++++++++++++++++

Installing Docker compose

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

1) Open https://docs.docker.com/compose/install/

2) Go to linux section

Copy and pase the below two commands

# sudo curl -L "https://github.com/docker/compose/releases/download/1.24.0/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

# sudo chmod +x /usr/local/bin/docker-compose

How to check docker compose is installed or not?

# docker-compose --version

+++++++++++++++++++++++++

Lets remove all the running container

# docker rm -f $(docker ps -aq)

To stop the containers

# docker-compose stop

We got lot of logs coming on the screen. to avoid it we use -d option

**Create a docker compose file for setting up LAMP architecture**

# vim docker-compose.yml

---

version: '3'

services:

mydb:

image: mysql:5

environment:

MYSQL\_ROOT\_PASSWORD: sunilsunil

apache:

image: tomee

ports:

- 6060:8080

links:

- mydb:mysql

php:

image: php

links:

- mydb:mysql

- apache:tomcat

...

:wq

# docker-compose up -d

To see the list of the containers

# docker container ls

( Observation - we are unable to see the php container)

# docker ps -a

++++++++++++++++++++++++++++++++++++

Ex: Docker-compose file for setting up CI-CD Environment.

jenkins container is linked with two tomcat containers

# vim docker-compose.yml

---

version: '3'

services:

devserver:

image: jenkins/jenkins

ports:

- 7070:8080

qaserver:

image: tomee

ports:

- 8899:8080

links:

- devserver:jenkins

prodserver:

image: tomee

ports:

- 9090:8080

links:

- devserver:jenkins

...

:wq

# docker rm -f $(docker ps -aq)

# docker-compose up -d

# docker container ls

To check

public\_ip:7070 ( To check jenkins )

public\_ip:8899 ( Tomcat qa server )

public\_ip:9090 ( Tomcat prod server )

15.206.67.26:7070

15.206.67.26:8899

15.206.67.26:9090

+++++++++++++++++++++

Docker-compose file to set up testing environment.

selenium hub container is linked with two node containers.

# vim docker-compose.yml

---

version: '3'

services:

hub:

image: selenium/hub

ports:

- 4444:4444

chrome:

image: selenium/node-chrome-debug

ports:

- 5901:5900

links:

- hub:selenium

firefox:

image: selenium/node-firefox-debug

ports:

- 5902:5900

links:

- hub:selenium

...

:wq

Lets delete all the running containers

# docker rm -f $(docker ps -aq)

# docker-compose up -d

# docker container ls

As it is GUI container,

we can access using VNC viewer

Open VNC viewer

52.77.219.115:5901

password: secret

++++++++++++++++++++++++++++++++++++++++++++++

Docker volumes

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

Docker containers are ephemeral ( temporary )

Where as the data processed by the container should be permanent.

Generally, when a container is deleted all its data will be lost.

To preserve the data, even after deleting the container, we use volumes.

Volumes are of two types

1) Simple docker volumes

2) Docker volume containers ( Sharable volume )

Simple docker volumes

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

These volumes are used only when we want to access the data,

even after the container is deleted.

But this data cannot be shared with other containers.

usecase

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

1) Create a directory called /data ,

start centos as container and mount /data as volume.

Create files in mounted volume in centos container,

exit from the container and delete the container. Check if the files are still available.

Lets create a folder with the name

# mkdir /data

# docker run --name c1 -it -v /data centos ( v option is used to attach volume)

# ls ( Now, we can see the data folder also in the container)

# cd data

# touch file1 file2

# ls

# exit ( To come out of the container )

# docker inspect c1

We can see under mounts "data" folder it located in the host machine.

Copy the path

/var/lib/docker/volumes/3eaee3028ed0c305c028e61a729551d8091e5b7734ad3be0486f5491ff2b3838/\_data

Now, lets delete the container

# docker rm -f c1

After deleting the container, lets go to the location of the data folder

# cd /var/lib/docker/volumes/d867766f70722eaf8cba651bc1d64c60e9f49c5b1f1ebb9e781260f777f3c7e8/\_data

# ls ( we can see file1 file2 )

( Observe , the container is deleted but still the data is persistant )

+++++++++++++

docker volume containers

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

These are also known as reusable volume.

The volume used by one container can be shared with other containers.

Even if all the containers are deleted, data will still be available on the docker host.

Ex:

# sudo su -

Lets create a directory /data

# mkdir /data

Lets Start centos as container

# docker run --name c1 -it -v /data centos

# ls ( we can see the list of files and dir in centos )

# cd data

# ls ( currently we have no files )

Lets create some files

# touch file1 file2 ( These two files are available in c1 container)

Comeout of the container without exit

# Ctrl +p Ctrl +q ( container will still runs in background )

Lets Start another centos as container ( c2 container should use the same volume as c1)

# docker run --name c2 -it --volumes-from c1 centos

# cd data

# ls ( we can see the files created by c1 )

Lets create some more files

# touch file3 file4

# ls ( we see 4 files )

Comeout of the container without exit

# Ctrl +p Ctrl +q ( container will still runs in background )

Lets Start another centos as container

# docker run --name c3 -it --volumes-from c2 centos

# cd data

# ls ( we can see 4 files )

# touch file5 file6

# ls

Come out of the container without exit

# Ctrl +p Ctrl +q ( container will still runs in background )

Now, lets connect to any container which is running in the background

# docker attach c1

# ls ( you can see all the files )

# exit

Identify the mount location

$ docker inspect c1

( search for the mount section )

Take a note of the source path

/var/lib/docker/volumes/e22a9b39372615727b964151b6c8108d6c02b13114a3fcce255df0cee7609e15/\_data

Lets remove all the container

# docker rm -f c1 c2 c3

Lets go to the source path

# cd /var/lib/docker/volumes/e22a9b39372615727b964151b6c8108d6c02b13114a3fcce255df0cee7609e15/\_data

# ls ( we can see all the files )

+++++++++++++++++++++++++++++++++++++++++++++++++++