**Docker Compose**

This is a feature of docker using which we can create multi-container architecture using YAML files. This YAML file contains information about the containers that we want to launch and how they must 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

The Yaml file is space indented.

Sample Yaml file

---

durgasoft: root element

trainers: child element

shiva: Devops

raj: Python

Coordinators:

lakshmi: Devops

rani: AWS

...

durgasoft --> root element

* To validate the above 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 whether docker-compose is installed or not?

# docker-compose --version

1. **Create a docker-compose file for setting up the dev environment. The MySQL container is linked with the WordPress container.**

# vim docker-compose.yml ( Name of the file should be docker-compose.yml)

---

version: '3'

services:

mydb:

image: mysql:5

environment:

MYSQL\_ROOT\_PASSWORD: sunilsunil

mysite:

image: wordpress

ports:

- 5050:80

links:

- mydb:mysql

...

:wq

Let’s remove all the running container

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

How to start the above services from dockerfile

# docker-compose up

We got a lot of logs coming on the screen. to avoid it we use the -d option. Before adding -d stop the container.

# docker-compose stop

# docker-compose up -d

To check WordPress

public\_ip:5050

To stop both the containers

# docker-compose stop

Remove the container

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

1. **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: shiva

apache:

image: tomee

ports:

- 6060:8080

links:

- mydb:mysql

php:

image: php

links:

- mydb:mysql

- apache:tomcat

...

:wq

How to start the above services from dockerfile

# 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

1. **Docker-compose file for setting up CI-CD Environment. The 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) ---------> 13.126.58.183:7070

public\_ip:8899 (Tomcat qa server) ---------> 13.126.58.183:8899

public\_ip:9090 (Tomcat prod server) ---------> 13.126.58.183:9090

1. **Docker-compose file to set up a 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

Let’s delete all the running containers.

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

# docker-compose up -d

# docker container ls

As it is a GUI container, we can access using the VNC viewer.

Open VNC viewer

52.77.219.115:5901

password: secret