#This file is written by Sayali Jadhav

**Day18: Docker for DevOps Engineers (Part-2)**

**Docker Compose**

Docker Compose is a tool that was developed to help define and share multi-container applications.

With Compose, we can create a YAML file to define the services and with a single command, can spin everything up or tear it all down.

**What is YAML?**

YAML is a data serialization language that is often used for writing configuration files. Depending on whom you ask, YAML stands for yet another markup language or YAML ain’t markup language (a recursive acronym), which emphasizes that YAML is for data, not documents.

YAML is a popular programming language because it is human-readable and easy to understand.

YAML files use a .yml or .yaml extension.

**Task-1**

**Learn how to use the docker-compose.yml file, to set up the environment, configure the services and links between different containers, and also to use environment variables in the docker-compose.yml file.**

Docker Compose is a tool that assists in defining and sharing multi-container applications. By using Compose, we can define the services in a YAML file, as well as spin them up and tear them down with one single command.

Docker Compose is used for running multiple containers as a single service. Each of the containers here run in isolation but can interact with each other when required. Docker Compose files are very easy to write in a scripting language called YAML, which is an [**XML**](https://www.simplilearn.com/tutorials/programming-tutorial/what-is-xml)-based language that stands for Yet Another Markup Language.

**1)Install docker-compose**

Text

Description automatically generated

**2)Create a docker-compose.yml file inside project folder**

Text

Description automatically generated

* **version: "3.3"** denotes that we are using version 3.3 of Docker Compose.
* The **services** section defines all the different containers we will create.
* The **build** keyword specifies the location of our Dockerfile, and . represents the directory where the docker-compose.yml file is located.
* The **image** keyword is used to specify the image from docker hub for mysql containers
* For the database and web, we are using the **ports** keyword to mention the ports that need to be exposed.
* And then, we also specify the **environment variables** for mysql which are required to run mysql.

**3)Run docker-compose.yml file**

**docker-compose up:** This command does the work of the docker-compose build and docker-compose run commands.

docker ps or **docker-compose ps** command list all the containers in the current docker-compose file.

A screen shot of a computer

Description automatically generated with low confidence

**4)Verify that application is working by accessing it in a web browser**

Graphical user interface, application, Teams

Description automatically generated

**docker-compose down** : This command stops all the services and cleans up the containers, networks, and images.

Text

Description automatically generated

**Task-2**

**1)Pull a pre-existing Docker image from a public repository (e.g. Docker Hub) and run it on your local machine. Run the container as a non-root user. Run the container as a non-root user (Hint- Use** usermod **command to give user permission to docker). Make sure you reboot instance after giving permission to user.**

For running container as a non-root user, use command **usermod** to give user permission to docker

**sudo usermod -a -G docker $USER**

Then reboot instance after giving permission to user.

**sudo reboot**



Copy docker image command from dockerhub-

Text

Description automatically generated

**2)Inspect the container's running processes and exposed ports using the docker inspect command.**

**docker inspect <container\_name or ID>**

Text

Description automatically generated

Text

Description automatically generated

**3)Use the docker logs command to view the container's log output.**

**docker logs <container\_name or ID>**

Graphical user interface, text

Description automatically generated

**4)Use the docker stop and docker start commands to stop and start the container.**

**docker stop:** To stop one or more running Docker containers.

**docker stop <container-name or ID>**

**docker start:** Start one or more stopped containers

**docker start <container-name or ID>**

Graphical user interface, text

Description automatically generated

**5)Use the docker rm command to remove the container when you're done.**

**docker rm:** Remove one or more containers.

--force, -f: Force the removal of a running container.

**docker rm <container\_name or ID>**

A screen shot of a computer

Description automatically generated with low confidence