Day 18: Docker-Compose Project

This is #90DaysofDevops challenge under the guidance of Shubham Londhe sir.

Day 18 TASK

check this for task:

https://github.com/LondheShubham153/90DaysOfDevOps/blob/master/2023/day18/tasks.md

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.
- Learn more about docker-compose visit here

What is YAML?

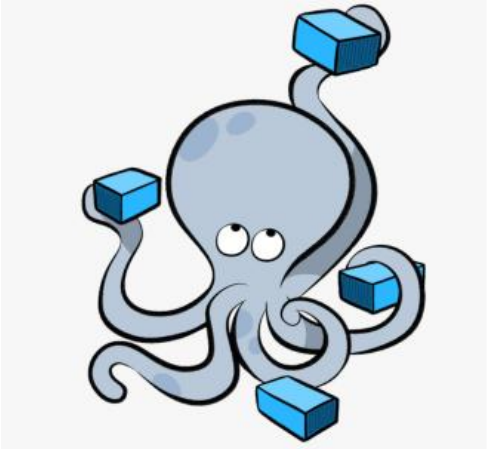
- YAML is a data serialization language that is often used for writing configuration files. Depending on whom you ask, YAML stands for yet another markup language or YAML ain't markup language (a recursive acronym), which emphasizes that YAML is for data, not documents.
- YAML is a popular programming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

What is Docker-Compose?

Docker Compose is a tool for defining and running multi-container Docker applications. It allows you to define your application as a set of services, each with its own container, and specify how these containers should interact with each other.



With Docker Compose, you can define your application's infrastructure, including databases, message queues, and web servers, in a simple YAML file.

Docker Compose provides a simple way to orchestrate and manage the containers that make up your application, allowing you to start and stop them together, scale them up and down as needed, and manage their network connections.

This makes it easier to develop, test, and deploy complex applications that consist of multiple services.

Step 1: Installing Docker-Compose in the server

https://kenfavors.com/code/how-to-install-docker-compose-on-linux-systems/

```
[ec2-user@ip-172-31-33-72 ~]$ sudo curl -L https://github.com/docker/compose/releases/dow
sr/local/bin/docker-compose
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
100 10.3M  100 10.3M    0     0   9.9M      0  0:00:01  0:00:01 --:--:-- 57.5M
[ec2-user@ip-172-31-33-72 ~]$ sudo chmod +x /usr/local/bin/docker-compose
[ec2-user@ip-172-31-33-72 ~]$ docker-compose --version
docker-compose version 1.21.0, build 5920eb0
```

Step 2 : Create a docker-compose.yml file

```
[root@ip-172-31-38-175 django-todo-cicd]# cat docker-compose.yml
version : "3.3"
services :
  web :
    container_name : "app"
    build : .
    ports :
         - "8000:8000"
db:
  container_name : "db"
  image: mysql
  ports:
      - "3306:3306"
  environment:
      - "MYSQL_ROOT_PASSWORD=test@123"
[root@ip-172-31-38-175 django-todo-cicd]#
```

So now we will see what all are the things written in the docker-compose file.

- version: 3.3 denotes that we are using version 3.3 of Docker Compose.
- The service informs us about the various containers the compose file is going to create in this case two will be created as we have two services : web and database.
- The build specifies the location of the Dockerfile, and . represents the directory where the docker-compose.yml file is present.
- The image keyword is used to specify the image from docker hub for MYSQL containers
- For both the services we are using different ports and using port keyword to mention the ports that need to expose for those services.
- And then, we also specify the environment variables for mysql which are required to run mysql.

Step 3 : Running docker-compose.yml file

The `docker-compose up` command is used to start up a set of containers defined in a Docker Compose file.

By default, `docker-compose up` starts containers in the foreground and logs their output to the console. You can use the `-d` flag to start the containers in detached mode, which runs them in the background.

The `docker-compose ps` command is used to display the status of containers defined in a Docker Compose file.

Step 4 : Verify that application is working on web browser

Task-2

- Pull a pre-existing Docker image from a public repository (e.g. Docker Hub) and run it on your local machine. Make sure you reboot instance after giving permission to user.

```
[root@ip-172-31-38-175 ~]# docker pull nginx:alpine
alpine: Pulling from library/nginx
63b65145d645: Pull complete
8c7e1fd96380: Pull complete
86c5246c96db: Pull complete
b874033c43fb: Pull complete
dbe1551bd73f: Pull complete
0d4f6b3f3de6: Pull complete
2a41f256c40f: Pull complete
Digest: sha256:6f94b7f4208b5d5391246c83a96246ca204f15eaf7e636cefda4e6348c8f6101
Status: Downloaded newer image for nginx:alpine
docker.io/library/nginx:alpine
```

- Run the container as a non-root user (Hint- Use `usermod` command to give user permission to docker).

```
[root@ip-172-31-38-175 ~]# sudo usermod -a  -G docker $USER
[root@ip-172-31-38-175 ~]#
```

- Then reboot instance after giving permission to user.

```
[root@ip-172-31-38-175 ~]# sudo reboot
```

- Inspect the container's running processes and exposed ports using the docker inspect command.

```
docker inspect <container_name or ID>
```

- Use the docker logs command to view the container's log output.

```
docker logs <container_name or ID>
```

- Use the docker stop and docker start commands to stop and start the container.

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

- Use the docker rm command to remove the container when you're done.

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

Please, feel free to drop any questions in the comments below. I would be happy to answer them.

If this post was helpful, please do follow and click the clap

_Thank you for reading

_Rajani