Docker compose:

docker compose, which is a tool that helps in the definition and running of multi-container applications. With docker compose we use a single YAML file for every configuration and just single commands to start and stop all the services.

If you are using a custom image then you will need to define its configurations in a separate Dockerfile in contrast to using a prebuilt image from Docker Hub, which you can define with the docker-compose.yaml file.

The *big* advantage of using Compose is you can define your application stack in a file, keep it at the root of your project repo (It's now version controlled), and easily enable someone else to contribute to your project. Someone would only need to clone your repo and start the compose app. In fact, you might see quite a few projects on GitHub/GitLab doing exactly this now.

These are the features that docker compose support:

- All the services are isolated running on the single host.
- Containers are recreated only when there is some change.
- The volume data is not reset when creating new containers, volumes are preserved.
- Movement of variables and composition within environments.
- It creates a virtual network for easy interaction within the environments.

Docker Compose — Installation:

Download the necessary files from **github** using the following command

```
[] curl -L "https://github.com/docker/compose/releases/download/1.10.0-
rc2/dockercompose
   -$(uname -s) -$(uname -m)" -o /home/demo/docker-compose
```

Next, we need to provide **execute privileges** to the downloaded Docker Compose file, using the following command

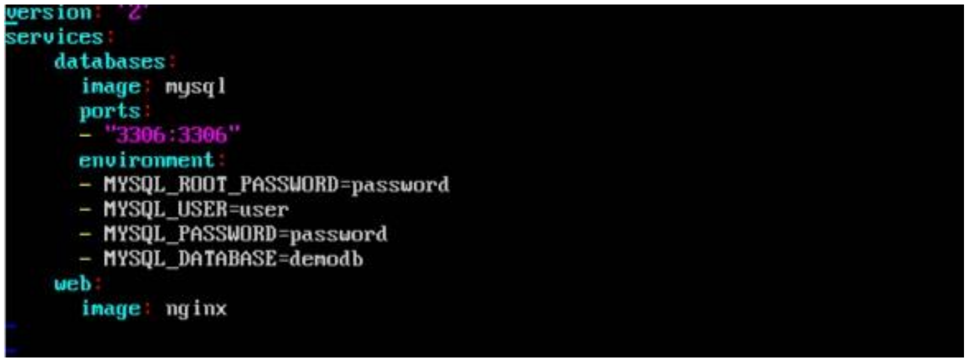
```
[] chmod +x /home/demo/docker-compose
```

We can then use the following command to see the **composed** version.

```
[] docker-compose version
```

Creating Docker-Compose File

```
[] vim docker-compose.yml
```



```
version: '2'
services:
    databases:
      image: mysql
      ports:
      - "3306:3306"
      environment:
      - MYSQL_ROOT_PASSWORD=password
      - MYSQL_USER=user
      - MYSQL_PASSWORD=password
      - MYSQL_DATABASE=demodb
    web:
      image: nginx
```

The various details of this file –

- The **database** and **web** keyword are used to define two separate services. One will be running our **mysql** database and the other will be our **nginx** web server.
- The **image** keyword is used to specify the image from **dockerhub** for our **mysql** and **nginx** containers
- For the database, we are using the ports keyword to mention the ports that need to be exposed for **mysql**.
- And then, we also specify the environment variables for **mysql** which are required to run **mysql**.

Docker compose commands:

1) Docker-compose up:

When the above command is run, docker-compose will search for a file named docker-compose.yml or docker-compose.yaml. Once it finds the file, it will read it and create the containers, volumes, images, and networks listed in the file and then run the containers.

[] docker-compose up

[] docker-compose up –d (detached mode)

2) Docker-compose down:

When this command is run, docker-compose will search for a file named docker-compose.yml or docker-compose.yaml. Once the file is located, it will stop all of the containers in the service and remove the containers from your system.

[] docker-compose down

3) docker-compose start:

This command will search for services listed in a docker-compose.yaml or docker-compose.yml file and start the containers listed in the service.

[] docker-compose start

4) docker-compose stop:

This command will stop all services located in your docker-compose.yml file.

[] docker-compose stop

5) docker-compose restart:

After this command is ran, all of your containers will restart.

[] docker-compose restart

6) docker-compose rm:

This command will remove all containers that are listed in a docker-compose file that are stopped. You can run the following command from within the directory of our docker-compose.yml or docker-compose.yaml file.

[] docker-compose rm

7) docker-compose ps:

If you need to view a list of containers that are deployed from your docker compose file, you can run the following command within the directory of your docker-compose file.

[] docker-compose ps

[] docker-compose ps –a

8) docker-compose images:

The above command will list all of the images that are on your system as part of your docker-compose file.

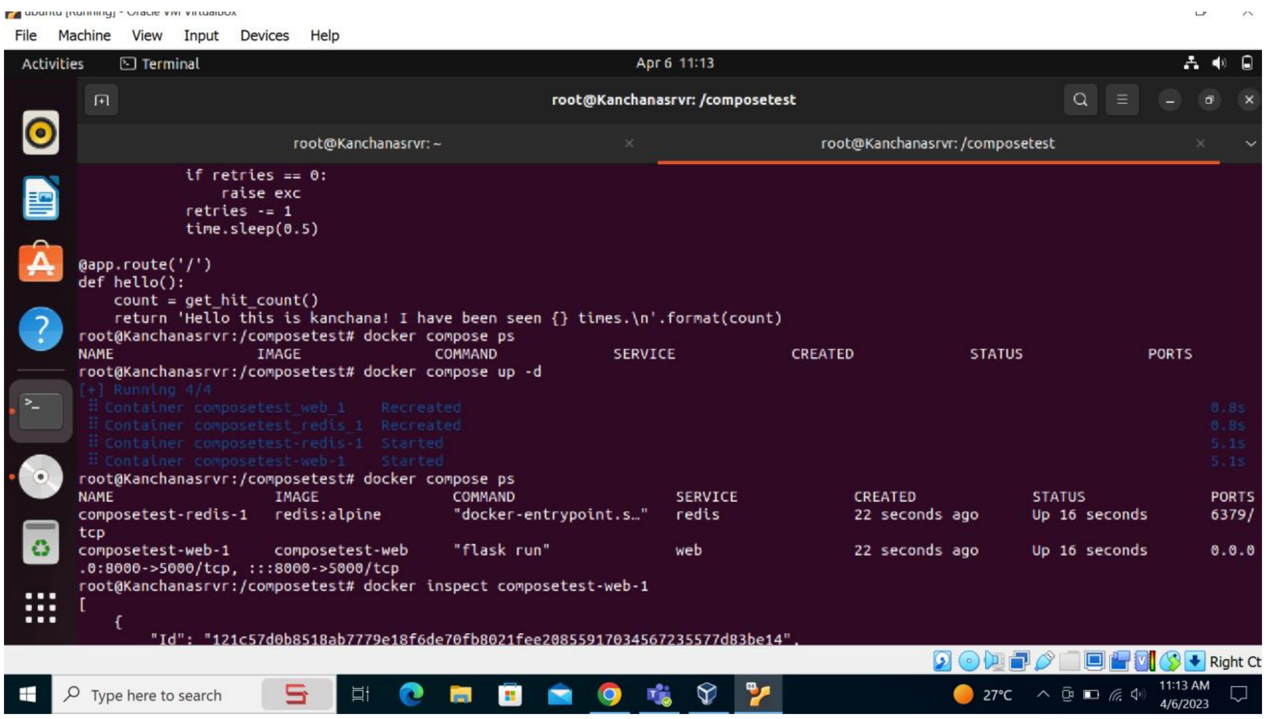
[] docker-compose images

SAMPLE COMPOSE-APP:

My Github URL For reference :

https://github.com/kanchana08/docker-compose-sample.git

[]docker compose up –d   (d is for running container in detached mode)


ubuntu [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Apr 6 11:13
root@Kanchanasrvr: /composetest
root@Kanchanasrvr: ~
root@Kanchanasrvr: /composetest
Right Ct
Type here to search
27°C
11:13 AM
4/6/2023

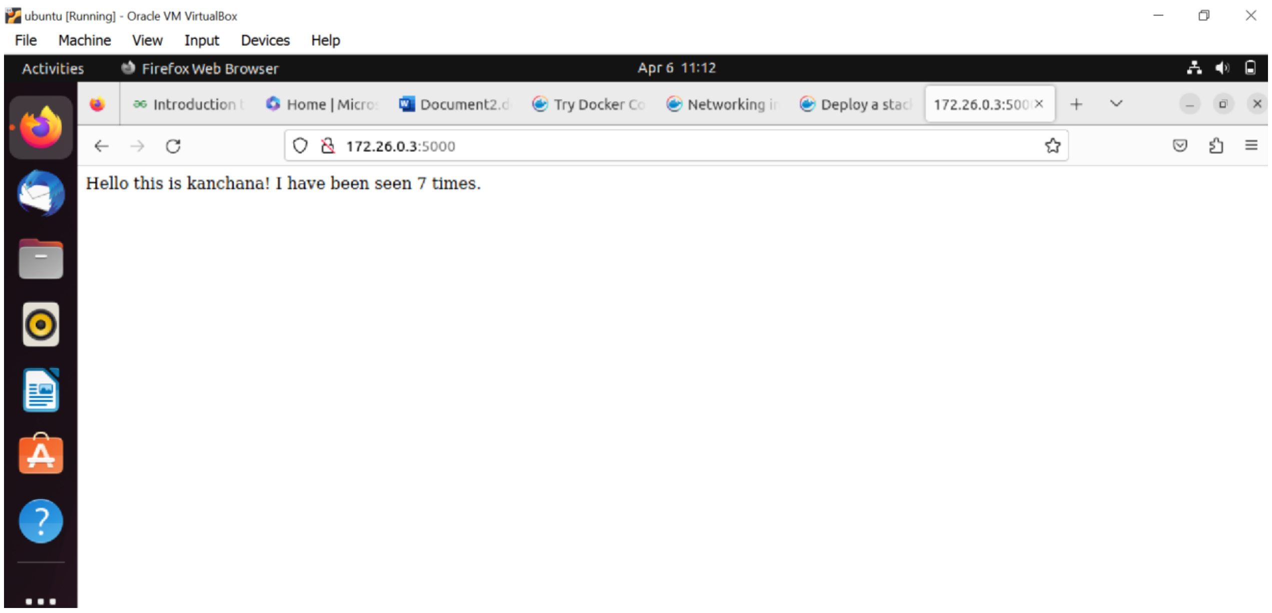
```
            if retries == 0:
                raise exc
            retries -= 1
            time.sleep(0.5)

@app.route('/')
def hello():
    count = get_hit_count()
    return 'Hello this is kanchana! I have been seen {} times.\n'.format(count)
root@Kanchanasrvr:/composetest# docker compose ps
NAME                IMAGE               COMMAND             SERVICE             CREATED             STATUS              PORTS
root@Kanchanasrvr:/composetest# docker compose up -d
[+] Running 4/4
 ⠿ Container composetest_web_1    Recreated                                                                     0.8s
 ⠿ Container composetest_redis_1  Recreated                                                                     0.8s
 ⠿ Container composetest-redis-1  Started                                                                       5.1s
 ⠿ Container composetest-web-1    Started                                                                       5.1s
root@Kanchanasrvr:/composetest# docker compose ps
NAME                  IMAGE               COMMAND                  SERVICE             CREATED             STATUS              PORTS
composetest-redis-1   redis:alpine        "docker-entrypoint.s…"   redis               22 seconds ago      Up 16 seconds       6379/
tcp
composetest-web-1     composetest-web     "flask run"              web                 22 seconds ago      Up 16 seconds       0.0.0
.0:8000->5000/tcp, :::8000->5000/tcp
root@Kanchanasrvr:/composetest# docker inspect composetest-web-1
[
    {
        "Id": "121c57d0b8518ab7779e18f6de70fb8021fee208559170345672355 77d83be14",
```

[]ls

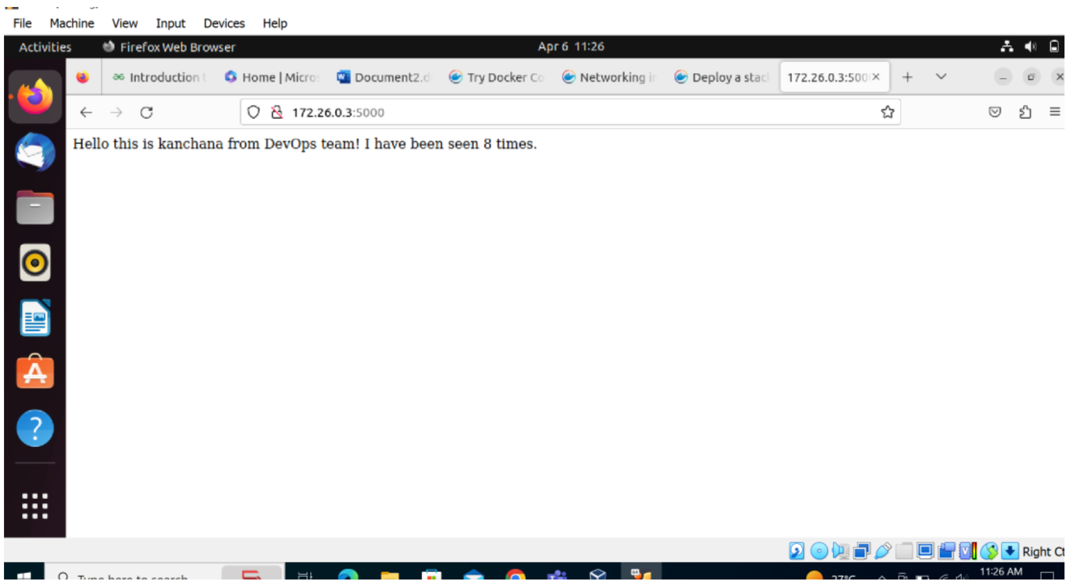
Dockerfile   docker-compose.yml   app.py   __pycache__   requirements.txt

OUTPUT:


ubuntu [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Firefox Web Browser Apr 6 11:12
172.26.0.3:5000

```
Hello this is kanchana! I have been seen 7 times.
```

Application code is now mounted into the container using a volume, you can make changes to its code and see the changes instantly, without having to rebuild the image.


File Machine View Input Devices Help
Activities Firefox Web Browser Apr 6 11:26
172.26.0.3:5000

```
Hello this is kanchana from DevOps team! I have been seen 8 times.
```

Docker swarm:

A Docker Swarm is a container orchestration tool running the Docker application. It has been configured to join together in a cluster. The activities of the cluster are controlled by a swarm manager, and machines that have joined the cluster are referred to as nodes.

Docker Swarm, there are typically several worker nodes and at least one manager node that is responsible for handling the worker nodes' resources efficiently and ensuring that the cluster operates efficiently. Docker Swarm is still included in docker-ce, but there is no longer a software-as-a-service for Docker Swarm.

Atleast we need to maintain 5 worker nodes.

Docker swarm commands:

1) Docker info:

This command used check swarm status (active/inactive)

[] docker info

Swarm:inactive

<u>2) docker swarm init :</u>

This command used to Initiate swam cluster and using this it gives docker swarm joining token

[] docker swarm init

Copy link and paste on worker nodes

Join chcdhgchj767 y986y6b

3<u>) docker node ls:</u>

This command used to Lists nodes in the swarm

[] docker node ls

4<u>) docker node promote:</u>

This commands used to Promote node to a manager role

[] docker node promote hostname (1.2.3.4)

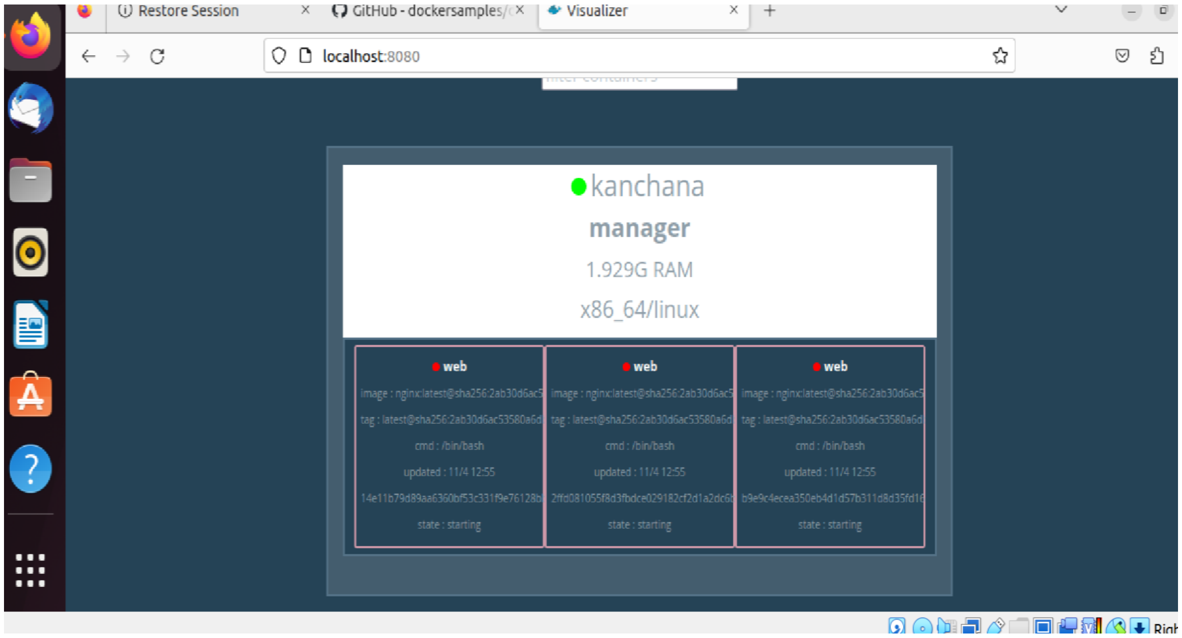
5) docker service create:

The above command used to Start new service in Docker swarm

[] docker service create -d --name webapp -p 4535:80 --hostname webappsrvr --replicas=3 nginx:latest /bin/bash

To see applications running in manager and worker nodes visualizer

Each node in the swarm will show all tasks running on it. When a service goes down it'll be removed. When a node goes down it won't, instead the circle at the top will turn red to indicate it went down. Tasks will be removed. Occasionally the Remote API will return incomplete data, for instance the node can be missing a name. The next time info for that node is pulled, the name will update.



```
● kanchana
manager
1.929G RAM
x86_64/linux

● web
image : nginx:latest@sha256:2ab30d6ac5
tag : latest@sha256:2ab30d6ac53580a6d
cmd : /bin/bash
updated : 11/4 12:55
14e11b79d89aa6360bf53c331f9e76128b
state : starting

● web
image : nginx:latest@sha256:2ab30d6ac5
tag : latest@sha256:2ab30d6ac53580a6d
cmd : /bin/bash
updated : 11/4 12:55
2ffd081055f8d3fbdce029182cf2d1a2dc6
state : starting

● web
image : nginx:latest@sha256:2ab30d6ac5
tag : latest@sha256:2ab30d6ac53580a6d
cmd : /bin/bash
updated : 11/4 12:55
b9e9c4ecea350eb4d1d57b311d8d35fd1
state : starting
```

6) <u>docker service ps</u> :

This command used to Lists tasks of service

```
Id     name      mode       image    replicas   port

123   web     replicated      nginx    0/3       9870:80
```

[] docker service ps

docker service ps web

7) docker service ls:

This command used to Lists service

[] docker service ls