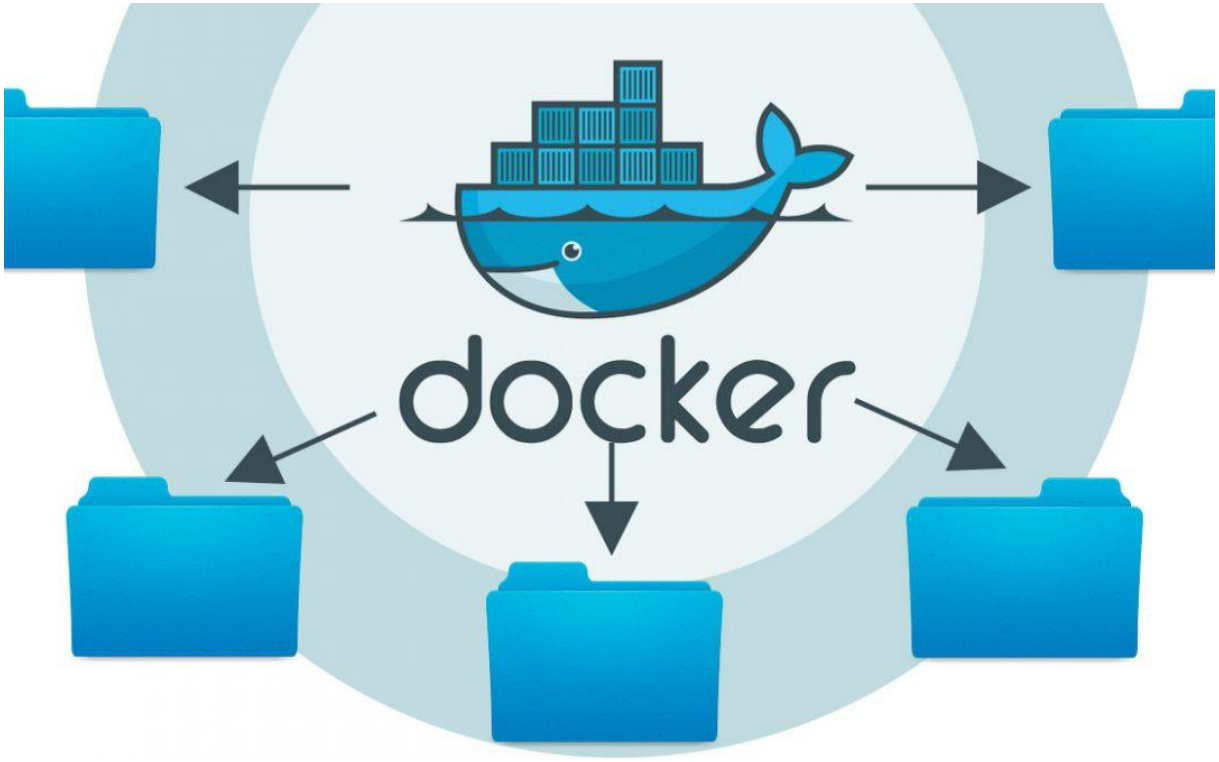
Day 19 Task: Docker for DevOps Engineers



Docker-Volume

A Docker volume is a way to store data outside of a container's filesystem. This allows the data to persist even if the container is deleted, and also allows the data to be shared between multiple containers. Volumes can be created, listed, and managed using the docker volume command, and they can be mounted to a container using the --mount or -v option when running the docker run command. Volumes can be backed by a variety of storage drivers, including local storage, network-attached storage, or cloud-based storage. This enables to keep the data persistent even after container is deleted.

There are different storage drivers available to use with Docker volumes, including:

- Local: This is the default storage driver and it uses the host's filesystem to store the data.
- NFS (Network File System): It allows to connect to a remote NFS share.
- glusterfs: It allows to connect to a GlusterFS filesystem.
- ceph: It allows to connect to a Ceph storage cluster.
- and many more.

One of the key benefits of using volumes is that they allow for data to be easily migrated between different hosts or environments. This is because the data is stored

outside of the container's filesystem and can be attached to different containers as needed. Docker volumes also provide a way to backup and restore data, which can be useful for disaster recovery or for moving data between environments.

Docker Network

Docker Networking is a way to manage the network connections between containers in a Docker environment. It allows containers to communicate with each other and with the host system, and also enables the use of various networking features such as load balancing and service discovery.

Docker provides several built-in network drivers that can be used to create networks and connect containers to them. The most commonly used network drivers are:

- bridge: This is the default network driver and it creates a virtual network internal to the host. Containers connected to this network can communicate with each other and with the host system.
- host: This network driver connects a container directly to the host system's network stack.
- overlay: This network driver allows to create a multi-host network, which allows containers to communicate with each other across multiple Docker hosts.

Each container can be connected to one or more networks. This allows to isolate different groups of containers and control the traffic flow between them.

Task-1

- Create a multi-container docker-compose file which will bring *UP* and bring *DOWN* containers in a single shot (Example - Create application and database container)

```
ubuntu@ip-172-31-52-167:~/projects$ sudo nano docker-compose.yml
ubuntu@ip-172-31-52-167:~/projects$ cat docker-compose.yml
version : "3.3"
services:
  web:
    image: nginx:latest
    ports:
      - "80-80"
  db:
    image: mysql
    ports:
      - "3306:3306"
    environment:
      - "MYSQL_ROOT_PASSWORD=User@123"
```

- Use the `docker-compose up` command with the `-d` flag to start a multi-container application in detached mode.

```
ubuntu@ip-172-31-52-167:~/projects$ docker-compose up -d
Pulling web (nginx:latest)...
latest: Pulling from library/nginx
8740c948ffd4: Pull complete
d2c0556a17c5: Pull complete
c8b9881f2c6a: Pull complete
693c3ffa8f43: Pull complete
8316c5e80e6d: Pull complete
b2fe3577faa4: Pull complete
Digest: sha256:b8f2383a95879e1ae064940d9a200f67a6c79e710ed82ac42263397367e7cc4e
Status: Downloaded newer image for nginx:latest
```

- Use the `docker-compose scale` command to increase or decrease the number of replicas for a specific service. You can also add `replicas` in deployment file for *auto-scaling*.

```
ubuntu@ip-172-31-52-167:~/projects$ docker-compose up -d --scale web=2
projects_db_1 is up-to-date
Creating projects_web_2 ... done
ubuntu@ip-172-31-52-167:~/projects$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS                                                  NAMES
fb24f04b04c9   nginx:latest   "/docker-entrypoint.…"   24 seconds ago   Up 23 seconds   0.0.0.0:49154->80/tcp, :::49154->80/tcp                projects_web_2
413b7a766a17   mysql          "docker-entrypoint.s…"   3 minutes ago    Up 3 minutes    0.0.0.0:3306->3306/tcp, :::3306->3306/tcp, 33060/tcp   projects_db_1
1993a8ecd8b0   nginx:latest   "/docker-entrypoint.…"   3 minutes ago    Up 3 minutes    0.0.0.0:49153->80/tcp, :::49153->80/tcp                projects_web_1
ubuntu@ip-172-31-52-167:~/projects$ docker-compose up -d --scale web=3
projects_db_1 is up-to-date
Creating projects_web_3 ... done
ubuntu@ip-172-31-52-167:~/projects$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS                                                  NAMES
1272c31f62a4   nginx:latest   "/docker-entrypoint.…"   5 seconds ago    Up 4 seconds    0.0.0.0:49155->80/tcp, :::49155->80/tcp                projects_web_3
fb24f04b04c9   nginx:latest   "/docker-entrypoint.…"   40 seconds ago   Up 39 seconds   0.0.0.0:49154->80/tcp, :::49154->80/tcp                projects_web_2
413b7a766a17   mysql          "docker-entrypoint.s…"   4 minutes ago    Up 4 minutes    0.0.0.0:3306->3306/tcp, :::3306->3306/tcp, 33060/tcp   projects_db_1
1993a8ecd8b0   nginx:latest   "/docker-entrypoint.…"   4 minutes ago    Up 4 minutes    0.0.0.0:49153->80/tcp, :::49153->80/tcp                projects_web_1
ubuntu@ip-172-31-52-167:~/projects$
```

- Use the `docker-compose ps` command to view the status of all containers

```
ubuntu@ip-172-31-52-167:~/projects$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS                                                  NAMES
1272c31f62a4   nginx:latest   "/docker-entrypoint.…"   5 seconds ago    Up 4 seconds    0.0.0.0:49155->80/tcp, :::49155->80/tcp                projects_web_3
fb24f04b04c9   nginx:latest   "/docker-entrypoint.…"   40 seconds ago   Up 39 seconds   0.0.0.0:49154->80/tcp, :::49154->80/tcp                projects_web_2
413b7a766a17   mysql          "docker-entrypoint.s…"   4 minutes ago    Up 4 minutes    0.0.0.0:3306->3306/tcp, :::3306->3306/tcp, 33060/tcp   projects_db_1
1993a8ecd8b0   nginx:latest   "/docker-entrypoint.…"   4 minutes ago    Up 4 minutes    0.0.0.0:49153->80/tcp, :::49153->80/tcp                projects_web_1
ubuntu@ip-172-31-52-167:~/projects$
```

- Use `docker-compose logs` to view the logs of a specific service.

```
ubuntu@ip-172-31-52-167:~/projects$ docker-compose logs
Attaching to projects_web_3, projects_web_2, projects_db_1, projects_web_1
db_1    | 2023-01-25 18:49:50+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.0.32-1.el8 started.
db_1    | 2023-01-25 18:49:50+00:00 [Note] [Entrypoint]: Switching to dedicated user 'mysql'
db_1    | 2023-01-25 18:49:50+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.0.32-1.el8 started.
db_1    | 2023-01-25 18:49:50+00:00 [Note] [Entrypoint]: Initializing database files
db_1    | 2023-01-25T18:49:50.966166Z 0 [Warning] [MY-011068] [Server] The syntax '--skip-host-cache' is deprecated and will be removed in a future r
size=0 instead.
db_1    | 2023-01-25T18:49:50.967172Z 0 [System] [MY-013169] [Server] /usr/sbin/mysqld (mysqld 8.0.32) initializing of server in progress as process
db_1    | 2023-01-25T18:49:50.984127Z 1 [System] [MY-013576] [InnoDB] InnoDB initialization has started.
db_1    | 2023-01-25T18:49:51.752268Z 1 [System] [MY-013577] [InnoDB] InnoDB initialization has ended.
db_1    | 2023-01-25T18:49:53.662309Z 6 [Warning] [MY-010453] [Server] root@localhost is created with an empty password ! Please consider switching o
db_1    | 2023-01-25 18:49:57+00:00 [Note] [Entrypoint]: Database files initialized
db_1    | 2023-01-25 18:49:57+00:00 [Note] [Entrypoint]: Starting temporary server
db_1    | 2023-01-25T18:49:57.817191Z 0 [Warning] [MY-011068] [Server] The syntax '--skip-host-cache' is deprecated and will be removed in a future r
size=0 instead.
```

- Use the `docker-compose down` command to stop and remove all containers, networks, and volumes associated with the application

```
ubuntu@ip-172-31-52-167:~/projects$ docker-compose down
Stopping projects_web_3 ... done
Stopping projects_web_2 ... done
Stopping projects_db_1  ... done
Stopping projects_web_1 ... done
Removing projects_web_3 ... done
Removing projects_web_2 ... done
Removing projects_db_1  ... done
Removing projects_web_1 ... done
Removing network projects_default
ubuntu@ip-172-31-52-167:~/projects$
```

Task-2

- Learn how to use Docker Volumes and Named Volumes to share files and directories between multiple containers.

```
ubuntu@ip-172-31-52-167:~/volume/react_django_vloume$ docker volume create --nam
e my_reactdjango_vloume2 --opt type=none --opt device=/home/ubuntu/volume/react_
django_vloume
my_reactdjango_vloume2
ubuntu@ip-172-31-52-167:~/volume/react_django_vloume$ docker volume ls
DRIVER    VOLUME NAME
local     c7eb70cecc4da83f7a27655d624a1871a43d7ad708ed8d7b5b191ffe2f0171ff
local     my_reactdjango_vloume2
ubuntu@ip-172-31-52-167:~/volume/react_django_vloume$ docker volume inspect my_reactdjango_vloume2
[
    {
        "CreatedAt": "2023-01-25T19:09:52Z",
        "Driver": "local",
        "Labels": {},
        "Mountpoint": "/var/lib/docker/volumes/my_reactdjango_vloume2/_data",
        "Name": "my_reactdjango_vloume2",
        "Options": {
            "device": "/home/ubuntu/volume/react_django_vloume",
            "type": "none"
        },
        "Scope": "local"
    }
]
ubuntu@ip-172-31-52-167:~/volume/react_django_vloume$
```

- Verify that the data is the same in all containers by using the docker exec command to run commands inside each container.

```
ubuntu@ip-172-31-52-167:~/volume/react_django_vloume$ docker exec -it react_django_vloume_web_1 /bin/sh
# ls
bin  boot  dev  docker-entrypoint.d  docker-entrypoint.sh  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
# touch file.txt file1.txt
# ls
bin  boot  dev  docker-entrypoint.d  docker-entrypoint.sh  etc  file.txt  file1.txt  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
# exit
```

- Use the docker volume ls command to list all volumes

```
ubuntu@ip-172-31-52-167:~/volume/react_django_vloume$ docker volume ls
DRIVER    VOLUME NAME
local     9c8e8c361cc63a77851651569d5d732f4f2fe9bc682cca9ff2e796453a58f0c7
local     21009facd758d66b4f63f22db9d9cbc86bc5daeb01c295d00d1c0e5efd9b3ab9
local     c7eb70cecc4da83f7a27655d624a1871a43d7ad708ed8d7b5b191ffe2f0171ff
local     my_reactdjango_vloume2
ubuntu@ip-172-31-52-167:~/volume/react_django_vloume$
```

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