BATCH: 149

17/10/2023

DT/NT : DT

LESSON : DevOps

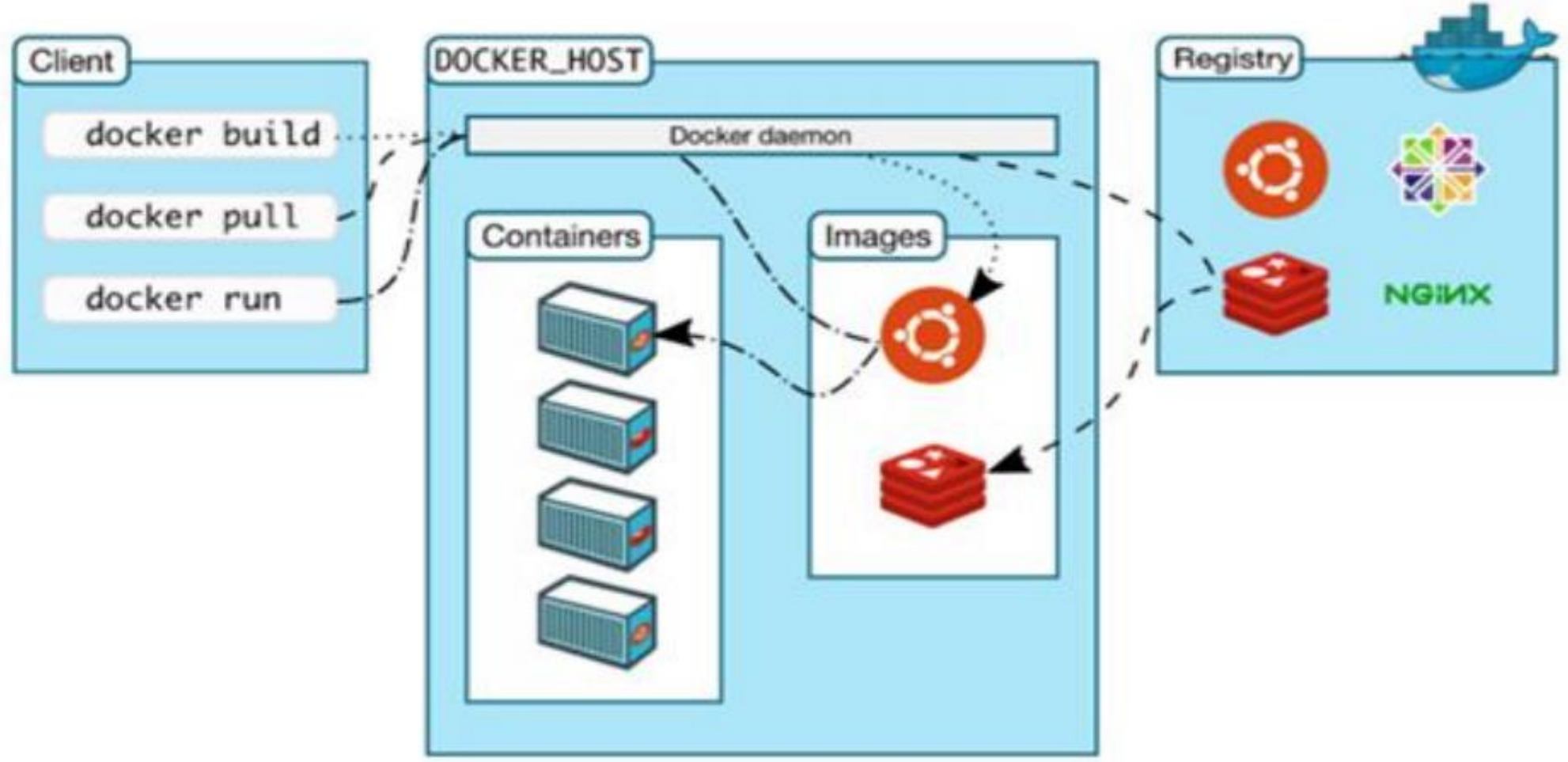
SUBJECT: Docker 2

Volume

TECHPRO EDUCATION

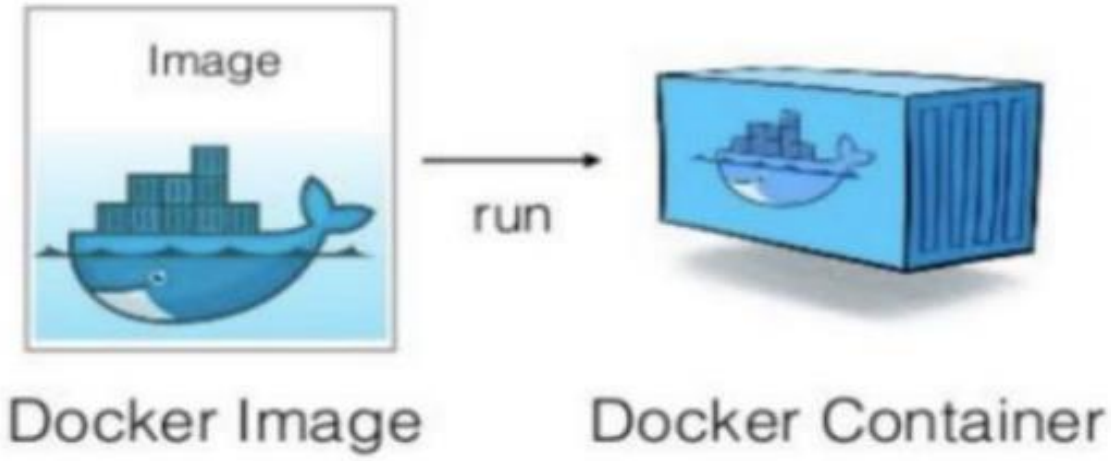
techproeducation.com +1 (585) 304 29 59

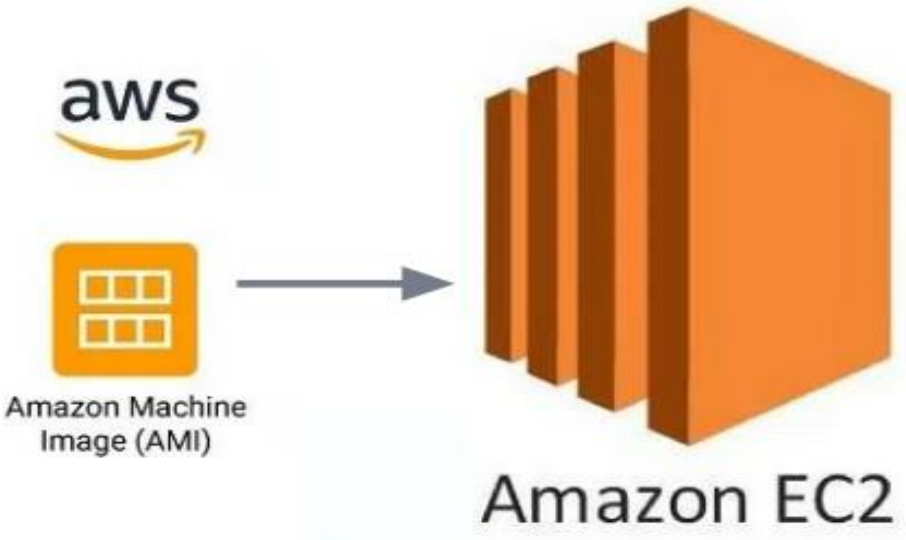
# Docker Architecture



# Images and Containers

- An image is a read-only template with instructions for creating a Docker container.
- A container is a runnable instance of an image.



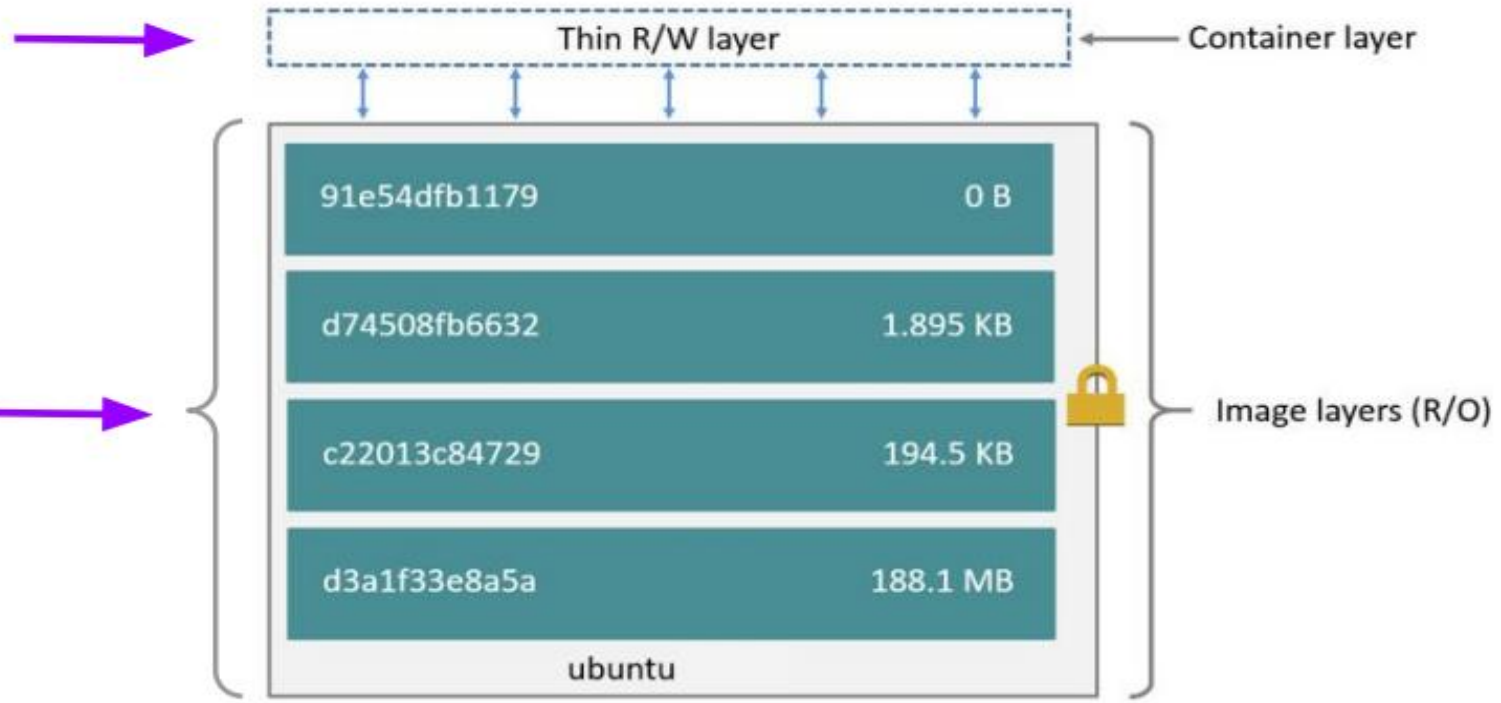
# Container Layers

- A Docker image is built up from a series of layers. Each layer represents an instruction in the image's Dockerfile. Each layer except the very last one is read-only.

```
docker run imageName
```

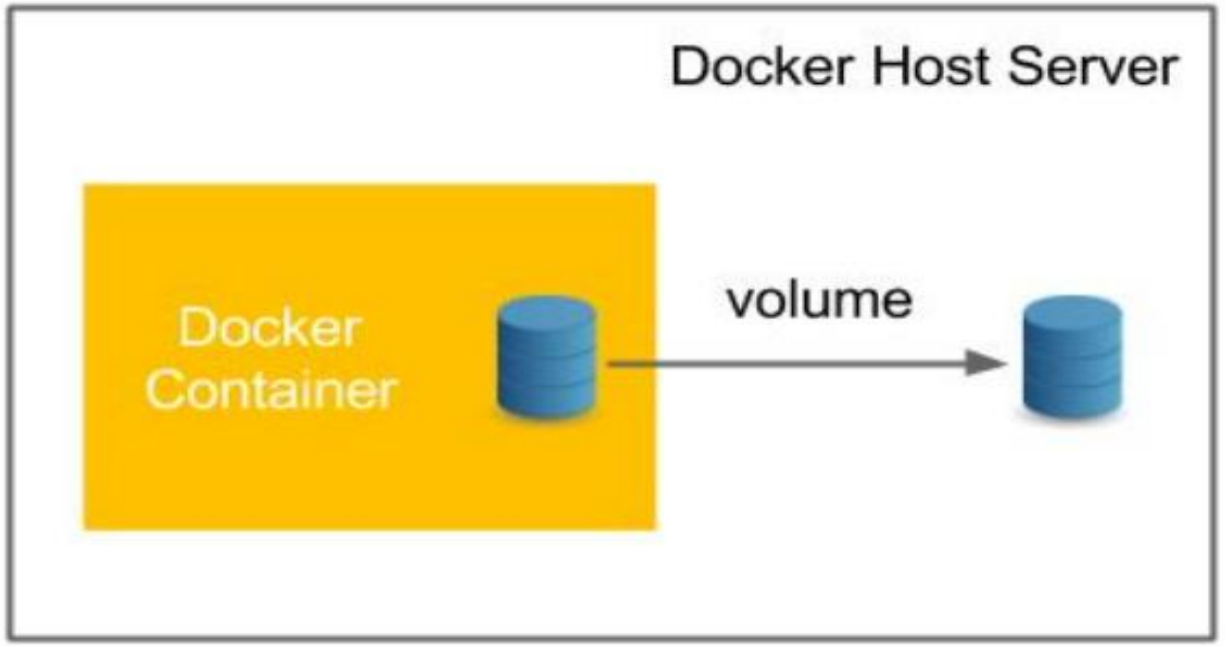
```
FROM ubuntu
COPY . /app
RUN apt-get install python3.6
CMD python3 /app/index.py
```

*Dockerfile*



# Manage data in Docker

- By default, all files created inside a container are stored on a **writable container layer**. This means that the data doesn't persist when that container no longer exists.
- Docker volumes, which are special directories in a container, store files in the host machine so that the files are **persisted** even after the container stops.



# Manage data in Docker

Volumes are created and managed by Docker. We can create a volume explicitly using the docker volume create command.

```
$ docker volume create firstvolume
```

# Manage data in Docker

- When we create a volume, it is stored within a directory on the Docker host. When we mount the volume into a container, this directory is what is mounted into the container.

```
$ docker volume inspect firstvolume
[
    {
        "CreatedAt": "2020-07-12T13:19:27Z",
        "Driver": "local",
        "Labels": {},
        "Mountpoint": "/var/lib/docker/volumes/firstvolume/_data",
        "Name": "firstvolume",
        "Options": {},
        "Scope": "local"
    }
]
```

# Declaration of volumes

- Volumes can be declared on the command-line, with the --volume or -v flag for docker run.
- v or --volume: Consists of three fields, separated by colon characters (:). The fields must be in the correct order.

```
--volume <volume_name>:<path>:<list of options>
```

# Docker Volume

CONTAINERLAR DOGAR, BUYUR VE OLUR.  BEYINLERINDEKI BILGILERI BIR YERE YEDEKLEMEMIZ GEREKIYOR; BURADA DA DEVREYE VOLUMELER GIRIYOR.

- harici bir disk alanına sahip değilseniz veya harici tutulan bir veritabanına kayıt yapılmıyorsa veriler kaybolur.
- Container yaşam süresinden daha uzun saklanması gereken verileriniz varsa bunları container içinde tutmayız. Bunun için Volume ‘ler kullanılır.

* Volume ‘leri container dışında tutmamız ve her yeni container ile ilintilendirmemiz(mount), paylaşılabilir ve erişilebilir yapmamız gerekir.

# Boş-Dolu Volume Mount Edildiğinde Ne Olur?

1. Eğer bir volume mount edildiği klasör mevcut değilse bu klasörü yaratır. Ve o anda volume içerisinde hangi dosyalar varsa bu klasörde de o dosyaları görürsünüz. Boşsa boş görürsünüz.

2. Eğer bir volume imaj içerisinde bulunan mevcut bir klasöre mount edilirse:

A: Klasör boşsa o anda volume içerisinde hangi dosyalar varsa bu klasörde de o dosyaları görürsünüz.

B: Klasörde dosya varsa ve volume boşsa bu sefer o klasördeki dosyalar volume'e kopyalanır.

C: Klasörde dosya var ya da yok fakat volume'de dosyalar varsa yani volume boş değilse, bu sefer siz o klasörün içerisinde volume'de ne dosya varsa onu görürsünüz.

# Bind Mounts


Host üstündeki bir klasör ya da dosyayı
Container içerisine map edebiliriz.
Buna Bind Mount denilir

# Docker Volume Commands

| Command | Description |
| --- | --- |
| docker volume create | Create a volume |
| docker volume inspect | Display detailed information on one or more volumes |
| docker volume ls | List volumes |
| docker volume prune | Remove all unused local volumes |
| docker volume rm | Remove one or more volumes |