Table of contents

Docker Installation

- Launch an EC2 instance, we will execute Docker commands on EC2 instance.

```
sudo yum update -y
```

If you are using amazon linux, use below command to install docker

```
sudo amazon-linux-extras install docker
docker -v
```

- Start the docker service and verify the status of Docker Daemon

```
sudo service docker start
sudo service docker status
```

- After installation of docker service, a group `docker` is created.
- Add `ec2-user` to the docker group to execute the docker commands in linux with `ec2-user`

```
docker info
cat /etc/group | grep docker
sudo usermod -a -G docker ec2-user
cat /etc/group | grep docker
docker info
```

- If there is command error, restart the ssh session and then try below command again.

Run Docker Containers

```
docker info
docker images
docker pull ubuntu:18.04
docker images
docker run -i -t ubuntu:18.04 /bin/bash
docker run -i -t ubuntu:20.04 /bin/bash
# docker run ubuntu:18.04
docker images

docker stop <CONTAINER_ID>
```

- Stopping the docker container

```
#Stop the container - docker stop <CONTAINER_ID> OR docker stop <CONTAINER_NAME>
sudo docker ps -a
sudo docker stop ContainerID
sudo docker ps -a
```

- Search for Docker Image in Docker Hub Registry

```
sudo docker search nginx
```

- Connect to docker interactively

```
sudo docker run -i -t ubuntu:20.04 /bin/bash

docker run --name=myUbuntuContainer -i -t ubuntu:20.04 /bin/bash
```

- The above command will run container with image specified, if image is not present locally it will download it using "docker pull"
- Run below commands inside the docker container bash shell

```
hostname
id
echo $HOME
pwd
```

- Detach docker from interactive container, we can detach it from our container by using the `Ctrl + P and Ctrl + Q` escape sequence. This escape sequence will detach the TTY from the container and land us in the Docker host prompt $, however the container will continue to run.

- Run below commands in EC2 linux shell

```
sudo docker ps
sudo docker ps --no-trunc
```

- Below is the information for the details related to `docker ps` command
  - `CONTAINER ID`: This shows the container ID associated with the container.
  - `IMAGE`: This shows the image from which the Docker container has been created.
  - `COMMAND`: This shows you the command executed during the container launch.
  - `CREATED`: Time when the container was created.
  - `STATUS`: Current status of the container.
  - `PORTS`: This tells you if any port has been assigned to the container.
  - `NAMES`: The Docker engine auto-generates a random container name by concatenating an adjective and a noun. Either the container ID or its name can be used to take further action on the container. The container name can be manually configured by using the `--name` option in the docker run subcommand.
- Connect back to container prompt, replace below `CONTAINER_ID` with actual container id value also `CONTAINER_NAME` with actual container name.

```
sudo docker attach CONTAINER_ID
OR
sudo docker attach CONTAINER_NAME
```

- Track the changes inside the containers, connect into shell of the docker and create some directories and files inside the container.

```
sudo docker run -i -t ubuntu:18.04 /bin/bash
cd /home
ls -al
touch file{1..5}
touch {f1.txt,f2.txt,f3.txt}
ls -al
```

- Use docker diff command to check the changes made inside the container

```
sudo docker ps
sudo docker diff ContainerID
```

Create a Docker Image

Create a Docker Image using container

> I1 - ubuntu 20.04 > C1 > install wget or any package in C1 > I2-Custom image > C2 from I2

- Check existing images and run a container using image. #Use ubuntu image as base image , install wget inside the container , then create an image from runnning container

> This setup is not preferred in a deployment environment.

```
sudo docker images
sudo docker run --name=base-image-container -i -t ubuntu:20.04 /bin/bash
```

- Run below commands inside container

```
whereis wget
apt-get update && apt-get install -y wget
whereis wget
```

- From another linux shell, use below `docker diff` command to check changes made in the above container

```
sudo docker diff <CONTAINER_NAME>
```

- To create a new image from exising running container

```
docker images
sudo docker commit 99a7ac739cbd ubuntudocker/ubuntu_wget
sudo docker images
sudo docker run -i -t ubuntudocker/ubuntu_wget:latest /bin/bash
```

- Below is the information for the details related to `docker images` command
  - `REPOSITORY`: This is the name of the repository or image.
  - `TAG`: This is the tag associated with the image
  - `IMAGE ID`: Every image is associated with a unique ID.
  - `CREATED`: Indicates the time when the image was created.
  - `SIZE`: Highlights the virtual size of the image.

Docker Build using Dockerfile - Creating Image

- Create a Docker image using DockerFile

```
touch Dockerfile
```

- Copy the below content into DockerFile

```
# Base Image will be as below
FROM ubuntu:20.04
ENV DEBIAN_FRONTEND=noninteractive
# Install dependencies
RUN apt-get update
RUN apt-get install -y apache2
RUN apt-get install -y apache2-utils
# Replace content of Apache Home Page
RUN echo 'This is Docker Image created using Dockerfile!' >
/var/www/html/index.html
# Expose Container Port
EXPOSE 80
# Execute command at container launch
CMD ["apache2ctl", "-D", "FOREGROUND"]
```

- Use below command to build an image locally using Dockerfile.

```
docker build -t docker-apache2 .
docker build -t docker-apache2 -f Dockerfile
docker images
```

> `.` in the above command specifies the path of the `Dockerfile`

> This Dockerfile uses the `ubuntu:20.04` image. The `RUN` instructions will simply run the linux commands for that image and then write the some content to the web server's document root. The `EXPOSE` instruction exposes *port 80* on the container, and the `CMD` instruction starts the web server.

- Run a container with the newly built image and keep docker running in detached mode with `-d` parameter
- The `-p 80:80` option maps the exposed port 80 on the container to port 80 on the EC2 host system.

> `-p <CONTAINER_PORT>:<HOST_PORT>`

```
docker run -d -p 80:80 -t docker-apache2
docker ps
CONTAINER ID    IMAGE           COMMAND                  CREATED          STATUS
PORTS                NAMES
86c6b6c2212e    docker-test     "apache2ctl -D FOREG…"   2 seconds ago    Up 1
second      0.0.0.0:80->80/tcp   relaxed_curie

netstat -nltp
```

- Test the content of the html file in the browser using public ip, make sure you have port 80 open in security group.

```
curl localhost:80
```

- Checking Docker Image Size

```
docker image ls
docker image inspect <IMAGE_NAME>
```

- Docker images are stored at `/var/lib/docker/overlay2`

```
du -sh -m /var/lib/docker/overlay2
```

- We can use the `docker rmi` command to remove the images:

```
docker rmi <IMAGE_NAME>
du -sh -m /var/lib/docker/overlay2
```

Creating a ECR Repository to Push the Docker Images

- Attach a role to ec2 instance with ECR full access
- Create a ECR repository

```
aws ecr create-repository --repository-name docker-testing --region us-east-1
```

- Use the following steps to authenticate and push an image to your repository.
  - Retrieve an authentication token and authenticate your Docker client to your registry.

```
aws ecr get-login-password --region <REGION_NAME> | docker login --username AWS --
password-stdin <ACCOUNT_ID>.dkr.ecr.<REGION_NAME>.amazonaws.com
```

- After the docker build is completed, check for local docker images present and tag your image so you can push the image to this repository:

> `docker tag` is what we use to define which repository an image will be pushed to, and `docker push` is the command that does the upload itself.

```
docker tag docker-apache2:latest <ACCOUNT_ID>.dkr.ecr.
<REGION_NAME>.amazonaws.com/docker-testing:latest

docker images
```

> Note that the image id remains the same between the 2 versions of the image. This is ideally the same image, just with 2 references.

- Run the following command to push this image to your newly created AWS repository:

```
docker push 082923708139.dkr.ecr.ap-south-1.amazonaws.com/docker-testing:latest
```

- Check whether docker image is available in ECR using AWS Console OR CLI

```
aws ecr describe-images --repository-name docker-testing --region us-east-1
```

- To Delete the ECR Repository use below command

```
aws ecr delete-repository --repository-name docker-testing --region us-east-1 --
force
```

> As part of the AWS Free Tier, new Amazon ECR customers get 500 MB-month of storage for one year for your private repositories. As a new or existing customer, Amazon ECR offers you 50 GB-month of always-free storage for your public repositories.

Docker Commands

- Below Command:
  - Usage: docker top CONTAINER [ps OPTIONS]
- If you want to quickly see what processes are running inside your container, use `docker top`.

```
 docker top
```