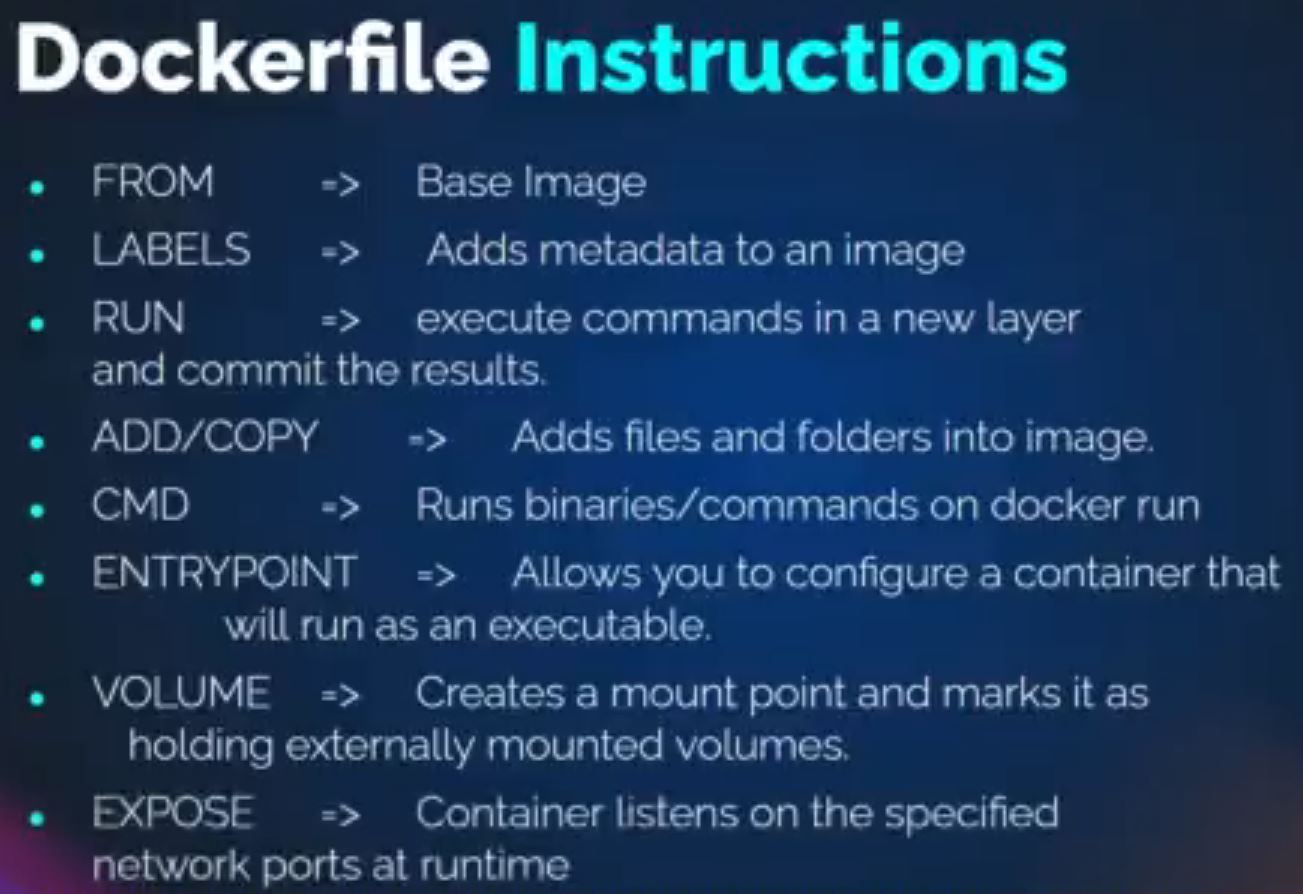
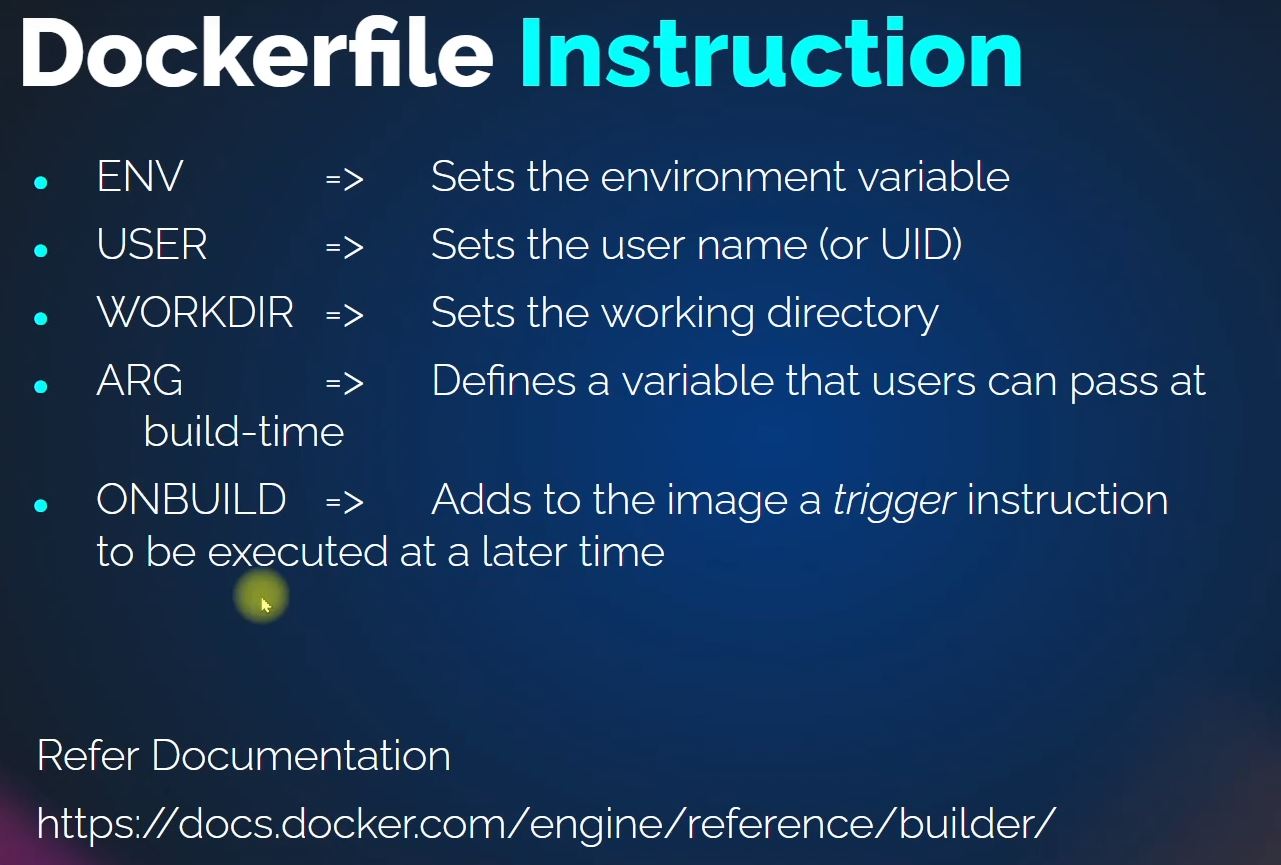
BUILD IMAGES:





STEPS:

1. Go to your Docker engine EC2 (Check the Doc: Setup of Docker)
2. No need to switch to Root user, Add the Ubuntu user to the user group
3. mkdir images
4. cd images/
5. ls
6. let’s run a website from toothplate.com on an Ubuntu container (the name of the website template is Nano)
7. mkdir nano
8. wget <https://www.tooplate.com/zip-templates/2122_nano_folio.zip>
9. ls
10. sudo apt install unzip –y
11. unzip 2122\_nano\_folio.zip
12. cd 2122\_nano\_folio/
13. tar czvf nano.tar.gz \* (the command **tar czvf nano.tar.gz \*** creates a compressed tar archive of all files and directories in the current directory and stores it in a file named **nano.tar.gz)**
14. mv nano.tar.gz ../ (moves the archived file a level up)
15. cd ..
16. rm -rf 2122\_nano\_folio 2122\_nano\_folio.zip (Deletes all other files apart from the archived file)
17. mv nano.tar.gz nano/ (moves the artifact into Nano directory)
18. cd nano/
19. ls
20. vim Dockerfile (Building the container with the Artifact)

FROM ubuntu:latest

LABEL “Author”=”Ola-Gabriel”

LABEL “Project”=”nano”

ENV DEBIAN\_FRONTEND=noninteractive

RUN apt update && apt install git –y

RUN apt install apache2 –y

CMD [“/usr/sbin/apache2ctl”, “-D”, “FOREGROUND”]

EXPOSE 80

WORKDIR /var/www/html

VOLUME /var/log/apache2

ADD nano.tar.gz /var/www/html (ADD extracts while COPY dumps)

#COPY nano.tar.gz /var/www/html

:wq

1. docker build –t nanoimg:V2 .
2. docker images
3. docker run –d - -name nanowebsite –p 9080:80 nanoimg:V2
4. In your browser: public IP of the EC2 Docker engine ; 9080
5. cd
6. Let’s Host the Image on Docker community
7. hub.docker.com, click on signup
8. fill in your information and login
9. click on Create Repository
10. Go to GitBash
11. cd images/nano/
12. docker build –t (dockerhub account name)/nanoimg:V2 .
13. docker login
14. type your username and password
15. docker push (dockerhub account name)/nanoimg:V2
16. docker stop (container ID)
17. docker rm (container ID)
18. docker rmi (dockerhub account name)/nanoimg:V2
19. To pull your image from Docker repo
20. docker run –d - -name nanowebsite –p 9080:80 (dockerhub account name)/nanoimg:V2 (Anybody in the world can run these image)
21. docker ps

……………………………………………………………………………………………………………

**DOCKER COMPOSE:**

Docker Compose is a tool that allows you to define and run multi-container Docker applications. It provides a way to define the services, networks, and volumes that make up your application in a single, easy-to-read YAML file. With Docker Compose, you can start and stop your entire application with a single command, and you can scale your application up or down by simply changing the number of instances of each service.

The main components of a Docker Compose file are:

* **services**: A service represents a single container in your application. Each service is defined by its own configuration, which includes the container image to use, the command to run, environment variables, and networking options.
* **networks**: A network defines how containers in your application can communicate with each other. By default, Docker Compose creates a network for your application, but you can define additional networks if needed.
* **volumes**: A volume provides a way to share data between containers or between a container and the host system. Volumes can be defined in your Docker Compose file and mounted in your services.

**STEPS: (Follow the Documentation link: docs.docker.com/compose/install/)**

1. Go to your Docker engine EC2 (Check the Doc: Setup of Docker)
2. No need to switch to Root user, Add the Ubuntu user to the user group
3. Go to Gitbash
4. Install Docker compose using these link: <https://docs.docker.com/compose/install/linux/>
5. Following the guide from: <https://docs.docker.com/compose/gettingstarted/>
6. mkdir composetest
7. cd composetest/
8. vim app.py

import time

import redis

from flask import Flask

app = Flask(\_\_name\_\_)

cache = redis.Redis(host='redis', port=6379)

def get\_hit\_count():

retries = 5

while True:

try:

return cache.incr('hits')

except redis.exceptions.ConnectionError as exc:

if retries == 0:

raise exc

retries -= 1

time.sleep(0.5)

@app.route('/')

def hello():

count = get\_hit\_count()

return 'Hello World! I have been seen {} times.\n'.format(count)

:wq

1. vim requirements.txt

flask

redis

:wq

1. vim Dockerfile

# syntax=docker/dockerfile:1

FROM python:3.7-alpine

WORKDIR /code

ENV FLASK\_APP=app.py

ENV FLASK\_RUN\_HOST=0.0.0.0

RUN apk add --no-cache gcc musl-dev linux-headers

COPY requirements.txt requirements.txt

RUN pip install -r requirements.txt

EXPOSE 5000

COPY . .

CMD ["flask", "run"]

:wq

1. vim Docker-compose.yml

version: "3.9"

services:

web:

build: .

ports:

- "8000:5000"

volumes:

- .:/code

environment:

FLASK\_DEBUG: "true"

redis:

image: "redis:alpine"

:wq

1. docker-compose up -d
2. Go to the security group of the Docker EC2 (all traffic from MyIP)
3. In your browser: public IP of the EC2 Docker engine ; 8000
4. To remove the container
5. docker-compose down