**Docker File:**

* Docker can build images automatically by reading the instructions from a Dockerfile.
* It is a text document that contains all the commands a user could call on the command line to assemble an image.
* Users can create an automated build that executes several command-line instructions in succession.

**Example : Dockerfile for Streamlit Framework**

//FROM: Sets the Base Image for subsequent instructions

FROM python:3

//LABEL: adds metadata to an image

//MAINTAINER: sets the Author field of the generated images.

LABEL maintainer="Huzefa Mustafa @huzefahdk"

//EXPOSE: Informs Docker that the container listens on the specified network ports at runtime.

EXPOSE 8503

ENV STREAMLIT\_SERVER\_PORT=8503

WORKDIR /app/src

//RUN: Execute any commands in a new layer on top of the current image and commit the results.

RUN pip install pipenv

//COPY: copies new files or directories from <src> and adds them to the filesystem of the container at the path <dest>

//The <dest> is an absolute path, or a path relative to WORKDIR

COPY Pipfile\* /

//--keep-outdated flag ensures pipenv doesn’t try to update dependencies if the lockfile is out of sync.

RUN pipenv lock --keep-outdated --requirements > requirements.txt

RUN pip install -r requirements.txt

COPY . .

//ENTRYPOINT: allows you to configure a container that will run as an executable.

ENTRYPOINT [ "streamlit", "run"]

//CMD: To provide defaults for an executing container.

CMD ["main.py"]

**OR**

RUN pip install streamlit

RUN pip install -r requirements.txt

For more information about dockerfile : https://hub.docker.com/search?q=&type=image

**BUILDING AND RUNNING DOCKER CONTAINERS:**

**Building an Image:**

docker build -t [image\_name] .

**Run the container with Streamlit defined port:**

docker run -it --name [container\_name] -p 8502:8502 -e STREAMLIT\_SERVER\_PORT=8502 -d [image\_name]

// -d : To run docker container in demonize mode, in the background

**Show list of containers:**

docker ps -a

//-a to show all containers

**Stopping Container:**

docker stop [container\_name]

**Removing Container:**

docker rm [container\_name]

**Saving the image in .tar**:

docker save [image\_name] -o test-app2.tar