I want to run docker app I host

$docker run -p

If it is docker based app I no need to install python

--------------------------------------------------------------

$ docker run let assume we are running stateful apps (ex : data base

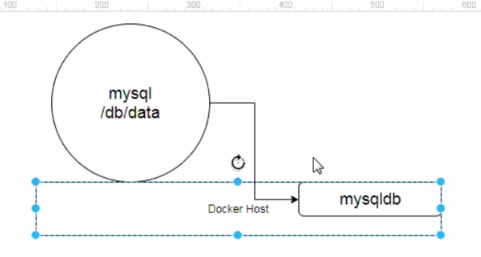
$docker volume list

$docker volume create mysqldb ( we are going to run mysql data base in container

(volume is a folder on a host

$docker volume inspect mysqldb ( I will show path )

$docker run -itd -v mysqldb:/opt/data alpine (mapping volume to this container . container generates data is stored on this host in this location , my sql stores data , if I loose container data I can spin up back to brand new container )



docker exec -it 34jdkfkr3k ash

#echo “welcome to docker”> one .sh

#ls

#cat one.sh

sh

#docker volume inspect mysqldb

Cd /var………..

Create a container by mounting that volume.docker run -itd -v mysqldb:/opt/data alpine

Docker rm -f paste id

docker run -itd -v mysqldb:/opt/data alpine ( any data generated stores in mysql db) ( where we see on container ? - ( /opt/data))

docker exec -it pasteid ash

**Assignment:**

There are diff types of volumes bind mount – stores data in file system tmpfs fount -stores data in memory(ram ) like cache – container dies data gets lost.

How two containers runining on same host share data ? -using volumes.

My app geerates some logs those logs need to be pushed to splunk we can use voumes there .

**Writing Docker files or dockerizing applications:**

- In order to dockerize any app we have to write docker file

To write docker file we must know docker file instructions

Did you write docker files ? – **YES**  and its not day to day work.

(If there are 5 micro services we want to write 5 docker files)

Cd ~

Git clone https://github.com/javahometech/python-app.git

Git add docker file

Vi docker file

(we are building our own image we typically extend some other image , I want python inside my container)

(#FROM alphine:3.10

#LABEL AUTHOR=”Hari kammna”

(LABELS allows us to add meta information to the docker file)

#LABEL RIER=api

#LABEL app=myapps

#RUN apk add python3

#RUN pip3 install Flask

#RUN pip3 install redis

#docker build -t myapp1.0

(Take python app put it on image

App.py is my app code )

**Run**: is used to execute commands inside docker image at build time.

I want to deploy py file on tomcat

**#WORKDIR /app** (instruction workdir becomes /app)

**WORKDIR**: will creates folder if it doesnot exists , uses a same folder it exists, it makes that folder as present working directory , current directory becomes /app

**#COPY app.py .** ( copy files from docker host in to docker image)

**#EXPOSE 5000**

I want to start my python app

**CMD [“python”, “app.py”]** ( CMD – is run time instruction (i.e commands are executed when we use docker run command )

What is diff b/w RUN and CMD ?

- Run is build time instruction and cmd is runtime instruction

**Docker build -t myapp:1.0 .**

**Docker run -d myapp1.0**

- (aftr creating container it calls python app.py)

Docker logs (containerid )-paste here

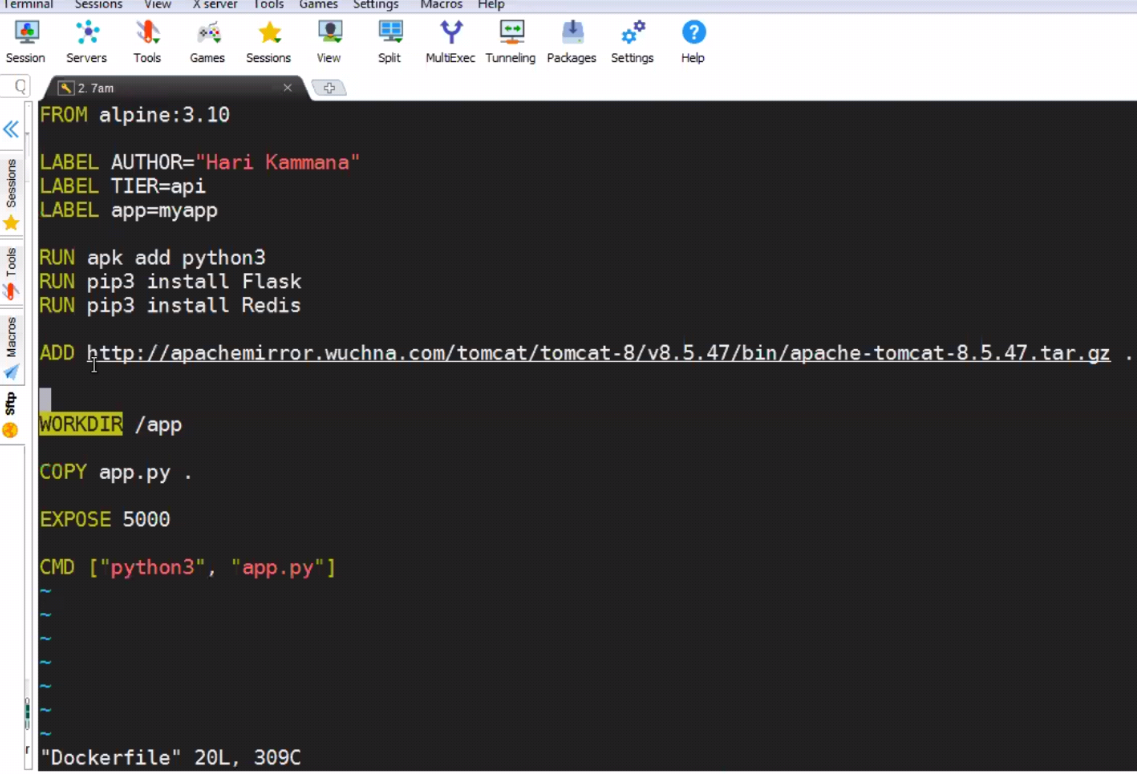
If I don’t do port mapping I cant access

Docker run -d -p 8080:5000 my app:1.0

ADD – instead of coppy we could use ADD

For copy cmd source can be local host.

For add source can be local host or a remote url.



11-5-19

CMD : it is used to start your app

docker run -d -p 8080:8080 tomcat:8

$ docker run -it javahome:1.0

**CMD instructions can be overwritten at cmd line.**

**IQ) Entry point:**

It is command similar to CMD

Entry point is a run time instruction we can’t entry point instructions at cmd line.

If we try to overwrite entry point it appends entry point command inside the docker with command at command line

**Using entry point with CMD:**

**FROM alpine:latest**

**Entrypoint[“ping”]**

**CMD[“googl.com”]**

**If we put two entries in docker file it eill skip the first and consider latest entry points.**

**IQ) diff b/w entry point and cmd line**

**CMD and**

**Entry point they have 2 forms**

* **Executable form**
* **Shell form**

**Executale form:**

**CMD [“echo”, “Hi $APP\_ENV”]**

**#shell form**

**CMD echo “Hi $APP\_ENV”**

Executable form will not consider variables ,i.e variables also printed as plain text instead of printing as value.

Environment variables are helpful to pass arguments to our application at run time.

When you need to write a docker file we need to get input from developer.

We can change or we can pass environment variales at run time also.

**$ docker run -it -e APP ENV=prod javahome:1.0 (env is at run time)**

**ARG : ( stands for arguments )**

ARG is at build time, at build time you want to change something dynamically use ARG.

**ARG PY\_VERSION=3**

**RUN apk add python${PY\_VERSION}**

**docker build –build-arg PY\_VERSION=3,5 -t javahome:1.0 .**