day- 10 docker_file directive. (1)

↳ Directive

↳ going Image change → going to Run. contair
→ make change → save it

↳ Dockerfile change →

capital → FROM - from where this pre-build image ~~to fetch~~

ENV - version, db-password, Configuration

RUN - Apt-get install (while building image) →

CMD - Process to start, serve end-user. (contair alive & serve user alive)

CMD → java ~~Run~~ ~~fat~~ .jar (working directory)

EXPOSE → ingx, tomcat, httpd → their port number. so publishing can be possible. [Docker port ↔ publishing port]

→ WORKDIR → Set working directory → /opt/share (all Remaing operation will take place here.)

COPY → COPY from host to image

ADD - Add directly zip file & extract automatically. URL github & all. (Add more powerful than COPY) tar file,

STOPSIGNAL - When Container Stops & give any message

HEALTHCHECH - container running or not

ENTRYPOINT -

CMD ~~or el~~ - should be ~~Run~~ add once. Second get ~~Error~~ Precident → CMD
→ CMD (gives high preeeedence)

*

```
FROM Ubuntu
RUN apt-get update
RUN apt-get install java.

ADD (to get Red off many steps)
    COPY .gz /tmp
    RUN cd tmp && tar xvs.
COPY myapp.war /opt/tomcat/webapps
EXPOSE 8080
ENTRYPOINT ['/opt/tomcat/-']
```

context whatever you pro

../code/

& docker build -t raviapp -f dockerfile name

ENTRY_POINT

→ docker PS -aq (container-ids) back-tick Not single Quote.

→ docker rm -f `docker PS -aq` (multiple container id)

→ docker ~~PS~~ → docker container Prune
↑ Remove Stoped Container.

→ CMD & ENTRY

→ Vi Dockerfile

From debian → Base_OS
CMD ["/bin/ping", "localhost"] (execute.)
ping commad to localhost

→ docker build -t test1 .

> docker run test1. → No detach option.
for Run in background

> docker run test1 google.com. (over ridden)

> docker run test1 each "ravi" (given something else)
> ravi ↑ CMD is Replaced (it is overriden)

↑ disadvantage of [CMD]

* what's gurentees, if CMD is replaced directly or indirectly.

> docker run test1 ping google.com.

* Entry_Point - → make, {Executable}

→ From debian: latest
ENTRY Point ["/bin/ping"] ← [whatever situation this will run always.]
CMD ["localhost"] ← can be overriden

docker run test2 ~~echo~~ echo 'rava'
ping: unknown

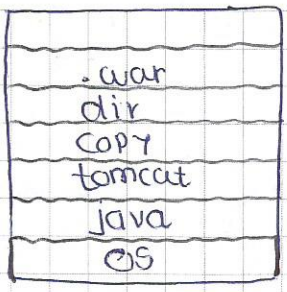
→ [will execute in any-cost.] (No two entry-Point times)

→ docker run test2 -e (replaced entry-point but not needed)

→ Entry Point [Run shell script]
add script

> docker ~~system~~ system df (comple Snap shot of image)

> docker rmi httpd (Remove your image)
docker rmi -f _
docker image prune. (all unused images)



→ docker saves Layer wise. (images as Layered in System)

– docker pull httpd.

{images are set of Layers}

> docker history httpd.
→ shows Layer.

> docker pull debian.

> docker pull nginx ← Layer of debian already exit

C4984: Already Exit Layer (OS is already there & taken from there.) Not downloading again

* Push docker-images

Repository → create → newImage (under our-account)
(forder)
→ push my image here.

1. tag image
2. push to Repo.

> docker images →

> Rename your images according forder (Repostore)

> docker tag test2 Leaddevops/newImage.
↓ ↑ ↑ new-name
copy of same image or → and renede with

> docker login
!

root@ip... ~# push docker push Leaddevops/newimage.
↑ account first ↑ image-name

Already available → mounted from library/debian → Latest

New Layer

> docker tag test2 Leaddev/newImage:123,

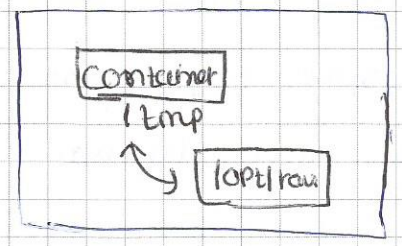
> /var/lib/docker/image/overlayer2/ ← ↑ db

↑ docker is build running container
trace layers from this {layer-db}

*

/var/lib/docker/image/overlay2/layerdb

* Volume



→ how to give data to running container

→ save the data to host or persist it

→ nginx image (Welcome to dev)

→ give data to container, get data from container.

→ docker Create ravi

→ ls -l /var/lib/docker/volumes/ ... docker/image/layer
/volume
ravi (volume) ↳ Ravi

→ ls -l /var/lib/... /ravi/_data/ #
> touch ravi Sampam giri

> { docker run -d -P -v ravi:/tmp nginx.
{Volume}

ravi: volume name
/tmp: Replaccating data of a ravi to temp location

> docker exec docker_id ls /tmp
- giri
- rav

→ you can share data to container
docker exec id touch ~~raj ram~~ /tmp/raj
toouch /tmp/ram

→ docker rm -f 'docker ps -aq'

> Inside container data is good but volume data remains

→ Create volumes & → Add to (mount to host folder)

> mkdir Dev
> mkdir Prod

> Dev → touch — dev.env → index.html

> Prod → vim prod.env

Bind mode

> docker run -d -P -v '/root/Dev' | usr share nginx

(index.html)

nginx / html

> docker run -d -P -v | root | prod: | user | share |

nginx / html

Runtime Changed value.

- db properties
- password
- Config

} Run time password.

Create the volume → keep the data give

random create folder → complete path.

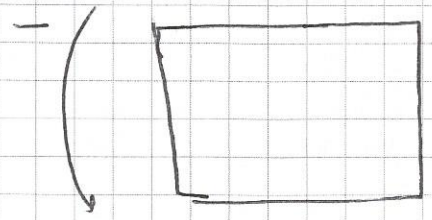
* Multiple -v (volume is also fine)

→ Splunk container → Agents Collects docker logs and

Forwarded

forward to Log Server.

* docker-compose.



→ run start only one container at a time

→ (run it 5 times)

executable file to run multiple container

→ declarative language. (put syntax in file)

→ docker rm -f 'docker ps -aq'

> vi docker-compose.yml

```
Version: '3'
Services:          → Set of container
  web:
    image: nginx
    Ports:
      - "80"
    Volumes:
      - | root | ...
```

→ Install docker compose

~~How~~ you ~~improve~~ managed

> docker-compose -v

> docker-compose up (Bring up container)

--scale web=5 -d

5 copys. (5 nginx)

run by default one container.

> docker ps

> docker-compose up --scale web=2 -d

> Multiple types of contain at one time

> {docker-compose down}

>vi test.yml

>
```
~~db:~~ db:
    - db_data:/var1-

volumes:
    ~~data~~ db_data: |__ ,
```

```
db:
  volume: __
wordpress:
  depends_on
    -db
```

store volume

> docker-compose up -f test.yml -d

docker-compose -f test.yml --scale db=3 t

--scale wordpress=3

up (here)

{

Disadvantage:

↳ Machine Crashes (Looses data) → github

↳ 10 Containers → Ran all Containers on same machine.
if this machine is gone.

↳ ~~Run all Contain~~