- **FROM** (used for base image)

- **RUN** (to run command)

- We can RUN in two format

1.SHELL // RUN apt update// -->> RUN CMD <options> <options>

2.EXEC // EXEC ["apt","update"] -->

**EXEC ["CMD","<options","<options>"]**

examples :

**RUN apt update**

**RUN apt install -y git curl sudo**

**EXEC ["ls", "-l", "-r", "-t"]**

'''

***Docker prefers EXEC format, but industry we write in SHELL format***

'''

- **COPY** (used to copy files from local to image)

Note: we need to calculate from build path, not from system path

\* Only folders and file but not compressed or links

COPY <source\_path\_from\_build\_context> <destination\_inside\_image>

examples :

1. COPY test.txt /home/test.txt

2. COPY ./demo/file1.txt /home/demo/file1.txt or /home/demo/file\_to\_container.txt

to check this alway build image first and run container

- docker build -t myub:latest .

- docker run -it -d --name mycon myub:latest

- docker exec -it mycon /bin/bash

boot etc lib media opt root sbin sys usr

root@0a5f106e3033:/# cd home

root@0a5f106e3033:/home# ls

hack.txt hack\_test\_renamed.txt ubuntu

root@0a5f106e3033:/home# read escape sequence

- **ADD** (used to copy files from local to image but)

ADD supports extra source formats

\*If the source is a compressed file then ADD will automatically uncompressed it in the destination.

\*If the source is a downloadable link then ADD will automatically download the file in the destination.

example: -

**ADD https://dlcdn.apache.org/tomcat/tomcat-9/v9.0.93/bin/apache-tomcat-9.0.93-deployer.tar.gz /home/**

- **CMD vs ENTRYPOINT**

\* Both CMD and ENTRYPOINT are used to define the default execution command of the container (the command which will be executed in the container

as main process).

\* If we use multiple CMD or ENTRYPOINT in the same Dockerfile only the last one will be considered.

\* If we use both CMD and ENTRYPOINT in the same Dockerfile, then ENTRYPOINT gets the highest priority and the command defined using CMD will be

as parameters to ENTRYPOINT.

**Difference**

- CMD can be completely overridden at the runtime (with docker run at the end

we can provide the command to override the CMD).

- ENTRYPOINT can't be overridden at the runtime but the command passed at the runtime will become parameters to ENTRYPOINT command defined in Dockerfile.

Syntax: we can define command in 2 ways

1. shell format

**CMD "ls -lrt"**

2. EXEC format

Always first element is command.

Except first element all the other elements are parameters

to command.

**CMD ["ls","-lrt"]**

**Overriding ENTRYPOINT :**

**docker run -it --name my-con --entrypoint echo my-image:latest daya**

**ENV**

* This instruction is used to set the environment variable inside the container.
* Using this instruction we can create ENV variables at build time which means in the docker images
* ENV variables should be in capital letters
  + - * EVN TEST test\_value or
      * ENV TEST=test\_value1

**ENV F\_NAME="daya"**

**ENV L\_NAME="shankar"**

**1. For individual variable**

**ENV <variable\_name> <value>**

**(OR)**

**ENV <variable\_name>=<value>**

**2. For multiple variable**

**ENV <variable\_name1>=<value1> <variable\_name2>=<value2> .....**

**ENV F\_NAME="daya" L\_NAME="shankar" FF\_NAME="mm"**

To create environment variables at run time (means in containers)

1. With the docker run command

**docker run -e <variable\_name>=<value> -e <variable\_name>=<value>**

**docker run -it --name my-con -e demo=value1 -e demo2=value2 my-image:latest**

2. With a list of variables in a file (.env file)

**docker run --env-file <file\_path> ...**

**What to do if we have more than 10 ENV variables?**

We can use .env filve / sometime .config

With a list of variables in a file (.env file)

**docker run --env-file <file\_path> …**

**docker run -it --rm --name my-con --env-file .env my-image:latest**

­

**WORKDIR**

This is used to set the working directory for all the instructions that follows it. Such as RUN, CMD, ENTRYPOINT, COPY, ADD .…

**ex: WORKDIR <path\_in\_container>**

**WORKDIR /home/daya/**

**ARG**

Using this instruction we can pass parameters to Dockerfile as user inputs.

**ex: ARG <arg\_variable\_name>=<value>**

Note: <value> acts as default value to the arg\_variable means if user does not set the arg value at build time this value will be used.

To pass the value at build time

**docker build --build-arg <arg\_variable\_name>=<user\_value>**

**docker build -t myub:latest --build-arg IN=/home .**

**docker build -t myub:latest --build-arg WORK\_DIR=/home –build-arg DIR\_NAME=test\_arg .**