# 56 Completed 😊

# Docker Run

1. docker run <image-name>
2. **Options:**
   1. **-d:** To run in detached mode. **Default** is attached mode when a container is run with command run.
   2. **-i, --interactive**:  
      
   3. **-t, --tty:**
      1. **This info is memorized when we run container with -t.   
         So later on when starting a**
   4. **--name <name>**: default is some random string.
   5. **-p:** To map host port to the container’s port to expose app to outside world.
      1. **Syntax: -p/--port host\_port:container\_port**
   6. **-rm:** When the container is stopped, the container will be removed automatically.
   7. **volume:**
      1. **Anonymous 🡺 docker run -v <container\_folder\_Path>**
      2. **Named 🡺 docker run -v <vol\_name>:<container\_folder\_Path>**
      3. **bind mount 🡺 docker run -v <host-path>:<container\_folder\_Path>**
         1. **for read-only: docker run -v <host-path>:<container\_folder\_Path>:ro**
         2. **shortcuts. Linux/MacOS 🡺 -v $(pwd):/app and windows -v %cd%:/app**
   8. **--evn = -e**
      1. **Syntax: --env <EVN\_Name>=<Value>**
      2. **For environment file 🡺 --env-file <file\_Path>**
   9. **--network <network\_name>: To start a container in this network**

# Docker Build

1. Naming an image is called tagging.
2. syntax:
   1. docker build -t name:tag .
3. Example:
   1. docker build node:4.5 .  
      where node is group or repo for all the node images and   
      4.5 is tag which is basically version.
4. **Options**:
   1. -t = --tag
   2. --build-arg: (in Dockerfile, ARG DEFAULT\_PORT=80

# Docker Attach

1. docker attach <container\_name\_id>: Attach means we are listening to the output from the container.
2. **Options**:

# Docker PS

1. **command**: Alternative is 🡺 docker container ls
   1. **docker** **ps** 🡺 **P**rocess **S**tatus as each container is a process in itself on OS Level.
2. **options**:
   1. **-a**: To show all containers (default shows just running)

# Docker container ls

1. **Command**: Alternative is 🡺 docker ps
   1. **docker container ls** 🡺 List all running containers.
2. **Options**:
   1. **-a**: To show all containers (default shows just running)

# Docker Container Prune

1. **Command**:
   1. docker container prune: To remove all stopped containers.

# Docker Start

1. **Command**: To restart a stopped container. Default mode is detached (Means command prompt will return)
   1. docker start <container\_id\_name>
2. **Options**:
   1. **-a**: To start in attached mode as default is detached.

# Docker Log

1. To print log of containerized our app till now.
2. **Options**:
   1. -f: To follow the logs.

# Docker RM

1. **Command**:
   1. docker rm <container\_name\_id>[]

# Docker RMI

1. **Command**:
   1. docker rmi <image\_name\_id>

# Docker image

1. **docker image prune**
   1. **Command**: To remove all unused images.
      1. docker image prune
   2. **Option**:
      1. **-a**: By default, only dangling images are removed (image without any name. Note: Tag is different) name:tag = repo:tag

# Docker Stop

1. **Command**:
   1. docker stop <container\_name\_id>

# Docker CP

1. To copy from/to docker container to/from host
2. **Syntax**:
   1. From Host to Container🡺 docker cp <relative\_host\_address> <container\_name>:/<folder\_or\_File>
   2. From Container to Host 🡺 docker cp <container\_name>:/<folder\_or\_File> <relative\_host\_address>

# Docker Pull

1. **Docker Pull**: To pull image from public or private registry.
2. **Command**:
   1. docker pull <image\_name>  
      To pull from private image registry, use <URL:NAME> also.

# Docker Push

1. **Docker Push**: To push image on to public or private registry.
2. **Command**:
   1. docker push <image\_name>  
      To push on to private image registry, use <URL:NAME> also

# Docker network

1. To create a **Docker Network** in which some containers can be run so that communication among them can be made easily.
2. **Command:**
   1. docker network <command>
3. **Options:**
   1. create <network\_name >
   2. ls

# Docker Hub

1. **Creating Repo**: Repo is a group of images. Like a group of all node images with different installed software and versions.
   1. Repo is created under account\_name.  
      account\_name/repo:tag 🡺 jatinbansalprogrammer/node:1.0.

# Miscellaneous Commands

1. **The following command is to run a container in interactive mode with sh command prompt**.
   1. winpty docker run --name image002 -it image001 sh
2. Docker Login, Docker Logout.

# Unsorted

1. To stop and remove **a container** in a single command:
   1. docker rm -f CONTAINER\_ID
      1. -f : for including running container
   2. **Alternative**: docker stop CONTAINER\_ID | xargs docker rm
2. To stop and remove all containers in a single command:
   1. docker rm -f $(docker ps -a -q)  
      docker rm -f $(docker container ls -q)
      1. -f : for including running container
3. To remove all the images in a single command:
   1. docker rmi $(docker images -q)
   2. If we include -f, it will stop all the containers and remove all the images.
4. To run an image with custom entry command.
   1. docker run -it --entrypoint sh image\_name.

# Docker Volume

1. **Docker volume**
   1. rm <vol\_name>
   2. prune
   3. inspect <vol\_name>

# Dockerfile

1. **FROM**: To specify the base image.
   1. **Syntax: FROM <image-name>**
2. **COPY**:
   1. **Syntax**: **COPY . .**
      1. 1st Dot:
         1. It represents the path on the host machine.
         2. From that path, all the files/folders including Dockerfile will be copied into the internal file system of the image/container.
      2. 2nd Dot:
         1. It represents the path on internal file system of the image/container.
         2. The path is relative to WORKDIR. By default, WORKDIR is the root of the internal file system of the image/container.
         3. We can change it by replacing the 2nd Dot with /app which now represents path relative to WORKDIR
3. **RUN:** This command will be executed by the Docker on the image during image creation.
   1. **Syntax:** RUN npm install
   2. **Application:** You want to update terminal, some Linux Dependencies, or you want to install some app dependencies like **npm install**.
4. **CMD**: The command after this “RUN” will be executed inside the container on the CMD relative to the WORKDIR.
   1. **Syntax**: **CMD [“node” “server”]**  
      If working directory is /app, the command **node server** will be executed on /app as soon as a container will be created.
   2. If we don’t specify CMD, then base image CMD will be executed (or the next base image one).   
      If no base image CMD, then error.
5. **EXPOSE:** To document which port to be exposed from this containerized app.
   1. This is just an info but the real implementation depends when we create container.
   2. **Syntax**:
      1. **EXPOSE 80**
6. **VOLUME**: Anonymous Volume
   1. **VOLUME [“Path/Inside/Container”]**
7. **.dockerfile**: To ignore files/folders during dockerfile copy command execution.   
   This is dot dockerignore like dot gitignore. Suppose the dot dockerignore file (.dockerignore) contains the following.   
   Dockerfile .git
8. **EVN Key=Value 🡸 They can be referred from further in the same Dockerfile using $Key**

# Communications

1. **Container to WWW**: Nothing to do. By default, active.
2. **Container to Localhost:** Instead of localhost, use **host.docker.internal**
3. **Container to container: Condition Running on the same host.**
4. **Step 01:** Under “**NetworkSettings**”:{  
    “**IPAddress**” : “172.17.0.2 🡸 This is the IP Address of the container and use this in other containerized app to connect with this container.

}