Docker Commands

# IMAGES :

## List all Local images

docker images

## Delete an image

docker rmi <image\_name>

## Remove unused images

docker image prune

## Build an image from a Dockerﬁle

docker build -t <image\_name>:<version>. //version is optional

docker build -t <image\_name>:<version>. -no-cache //build without cache

# CONTAINER :

## List all Local containers (running & stopped)

docker ps -a

## List all running containers

docker ps

## Create & run a new container

docker run <image\_name>

//if image not available locally, it’ll be downloaded from DockerHub

## Run container in background

docker run -d <image\_name>

## Run container with custom name

docker run - -name <container\_name> <image\_name>

**Stop multiple containers at once**

docker stop $ (docker ps -aq)

## Port Binding in container

docker run -p<host\_port>:<container\_port> <image\_name>

## Set environment variables in a container

docker run -e <var\_name>=<var\_value> <container\_name> (or <container\_id)

## Start or Stop an existing container

docker start|stop <container\_name> (or <container\_id)

## Inspect a running container

docker inspect <container\_name> (or <container\_id)

**To Inspect and get required information from containers**

Docker inspect <container name> | grep -i <keyword to search>

## Delete a container

docker rm <container\_name> (or <container\_id)

# TROUBLESHOOT :

## Fetch logs of a container

docker logs <container\_name> (or <container\_id)

## Open shell inside running container

docker exec -it <container\_name> /bin/bash docker exec -it <container\_name> sh

# DOCKER HUB :

## Pull an image from DockerHub

docker pull <image\_name>

## Publish an image to DockerHub

docker push <username>/<image\_name>

## Login into DockerHub

docker login -u <image\_name>

Or docker login

//also, docker logout to remove credentials

## Search for an image on DockerHub

docker search <image\_name>

# VOLUMES :

## List all Volumes

docker volume ls

## Create new Named volume

docker volume create <volume\_name>

## Delete a Named volume

docker volume rm <volume\_name>

## Mount Named volume with running container

docker run - -volume <volume\_name>:<mount\_path>

//or using - -mount

docker run - -mount type=volume,src=<volume\_name>,dest=<mount\_path>

## Mount Anonymous volume with running container

docker run - -volume <mount\_path>

## To create a Bind Mount

docker run - -volume <host\_path>:<container\_path>

//or using - -mount

docker run - -mount type=bind,src=<host\_path>,dest=<container\_path>

## Remove unused local volumes

docker volume prune //for anonymous volumes

# NETWORK :

## List all networks

docker network ls

## Create a network

docker network create <network\_name>

## Remove a network

docker network rm <network\_name>

## Remove all unused networks

docker network prune

**Use Cases**

* **Want the container to get deleted once stopped?**

docker run –rm -d –name <container name> <image name: Tag>

* **Want to take logs of container? In a standard format?**

#### **1. Install Required Tools**

sudo apt update && sudo apt install net-tools jq -y

Note: jq is used to format JSON logs for better readability.

#### **2. Locate Docker Container Logs**

cd /var/lib/docker/containers/

ls -al

Note: Each container has a directory named after its container ID.

#### **3. Identify the Container**

docker ps

Note: Get the CONTAINER ID from the output. Match it with a directory name.

#### **4. Navigate to the Container's Directory**

cd /var/lib/docker/containers/<CONTAINER\_ID>

Example:

cd /var/lib/docker/containers/2e5902d7fb763afa3d518f9eed6cb6734b79c390d26bd99bf7929e414aa4059a

#### **5. View or Export the Log File**

cat <CONTAINER\_ID>-json.log

Note: This file contains all logs generated by the container in raw JSON format.

#### **6. Format the Logs Using jq**

cat <CONTAINER\_ID>-json.log | jq

Note: This will present the logs in a structured format with log, stream, and time fields.

#### **7. Optional – Save the Logs to a File**

cat <CONTAINER\_ID>-json.log | jq '.' > exported\_container\_logs.json

Note: Saves formatted logs for backup, audit, or sharing.

* **To ping containers in bridge(default) network from a container in custom network?**

docker network connect <custom network name> < bridge network container’s name>

* **How do I move container data from root Dir to another folder (like external mount)?**

Move Docker's storage to /dockerdata to free up space from the root volume.

### **Stop Docker**

sudo systemctl stop docker

* Stops the Docker service safely.
* Ensures files are not in use while moving data.

### **Move Existing Docker Data**

sudo mv /var/lib/docker /dockerdata/docker

* Moves the entire Docker data directory to a new location /dockerdata/docker.
* This includes containers, images, volumes, and logs.

### **Create a Symlink**

sudo ln -s /dockerdata/docker /var/lib/docker

* Creates a symbolic link (shortcut) from /var/lib/docker to the new location.
* Docker will continue working as if data is still in the original path.

### **Start Docker**

sudo systemctl start docker

* Restarts the Docker service.
* Docker will now use /dockerdata/docker due to the symlink.

### **Verify Docker Root Directory**

docker info | grep "Docker Root Dir"

* Displays the path Docker is using to store data.
* Confirms the new location /dockerdata/docker is being used.
* **What is the use of docker socker?**

### Use of Docker Socket (/var/run/docker.sock) – Explained Simply

### **✅ What is /var/run/docker.sock?**

* It is a Unix domain socket used as the main communication method between the Docker CLI and Docker daemon (server).
* It allows programs (like Docker CLI or APIs) to send commands to the Docker Engine.

### **🔄 How It Works**

* Normally, when you run a command like docker ps, it doesn't directly run containers.
* Instead, the Docker CLI sends a request through /var/run/docker.sock to the Docker daemon.
* The daemon executes the action and sends the result back.

### **📌 Why It’s Important**

* It provides low-level control over Docker operations.
* Any process with access to docker.sock can control all Docker containers, create, stop, delete them – basically root-level access inside Docker.

### **🔐 Security Concern**

* Giving access to /var/run/docker.sock (e.g., to a container or user) is equivalent to giving root access to your host machine.
* Always be cautious and limit who/what can access it.

### **🛠️ Common Use Cases**

1. Mounting docker.sock into containers

docker run -v /var/run/docker.sock:/var/run/docker.sock -it docker

* + This gives the container the ability to control Docker (like a Docker-in-Docker setup).
  + Used in CI/CD tools like Jenkins, GitLab Runner, Portainer, etc.

1. Tools using Docker API
   * Any tool that interacts with Docker (like Portainer, Watchtower, Traefik) can use the socket to manage containers dynamically.