Docker Commands: Complete Guide

# 1. Steps to install docker on linux:

* Update Your System - sudo apt update
* Install Required Packages -

sudo apt install apt-transport-https ca-certificates curl software-properties-common

* Add Docker’s Official GPG Key-

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg

* Add the Docker Repository-

echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] \

https://download.docker.com/linux/ubuntu $(lsb\_release -cs) stable" | \

sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

* Update Package Index Again-

sudo apt update

* **Install Docker Engine -**

**sudo apt install docker-ce docker-ce-cli containerd.io**

* **Verify Docker Installation-**

**sudo docker --version**

**sudo docker run hello-world**

# 2. Common Docker Commands with Explanation

* docker pull <image-name>

Downloads a Docker image from Docker Hub.

* docker run <image-name>

Runs a container from the specified image.

* docker images

Lists all Docker images on your system.

* docker ps

Lists currently running containers.

* docker ps -a

Lists all containers (running and stopped).

* docker run -d --name <name> <image>

Runs a container in detached mode with a custom name.

* docker exec -it <container> /bin/bash

Access the shell inside a running container.

* docker logs <container>

Displays logs for a container.

* docker run -p <host-port>:<container-port> <image>

Maps a host port to a container port.

* docker compose build

Builds images defined in docker-compose.yml.

* docker compose up --build

Builds and starts containers.

* docker-compose up

Starts containers using legacy syntax (older Docker versions).

* docker-compose up --build

Builds and starts containers using legacy syntax.

* docker compose down

Stops and removes containers, networks, volumes.

* docker system prune

Removes unused data (containers, images, networks, volumes).

* docker system prune -a

Removes all unused images, not just dangling ones.

# 2. Volume Mapping for Live Code Changes

Use the following volume mapping in your docker-compose.yml to reflect code changes without rebuilding the container:

volumes:  
 - .:/app

This maps the current directory on the host machine to the /app directory inside the container.

# 3. Export and Load Docker Images as .tar

Step-by-step to export and use Docker images on another machine:

1. Step 1: Export Docker Image

docker save -o myapp.tar myapp:latest

1. Step 2: Copy to Another Machine

scp myapp.tar user@remote:/home/user/

1. Step 3: Load on New Machine

docker load -i myapp.tar

1. Step 4: Run Container

docker run -d --name mycontainer -p 8000:8000 myapp:latest

# 4. Extra Useful Docker Tips and Comments

* Tag your images before saving/exporting

Use `docker tag` to rename or version images before export:  
 docker tag myapp:latest myapp:v1.0

* Clean up dangling volumes

Use `docker volume prune` to remove unused volumes.

* Inspect container details

Use `docker inspect <container>` to view low-level container information (e.g., network IP, mounts).

* Follow logs in real-time

Use `docker logs -f <container>` to continuously stream logs.

* Run with environment variables

Pass variables at runtime:  
 docker run -e ENV\_VAR=value myapp

* Attach container to a specific network

Use `--network` to assign containers to user-defined networks:  
 docker run --network=mynet myapp

* Remove all stopped containers

docker container prune

* Rebuild container without cache

docker compose build --no-cache

* Automatically restart container on crash or reboot

docker run --restart=always -d myapp

* Use named volumes for persistent data

In docker-compose.yml:  
 volumes:  
 - db\_data:/var/lib/mysql