🚀 Docker Tutorial: Essential Commands & Use Cases

Corporate Training by aneesh ansari

CGI Batch

follow me https://www.linkedin.com/in/aneeshansari/

This guide walks you through the core Docker commands to manage images, containers, and volumes effectively. Let's get started!

📋 1. Check Docker Installation

```
curl -fsSL https://get.docker.com -o get-docker.sh
sh ./get-docker.sh
docker --version
docker version
```

Check your Docker installation and see version details.

🖼️ 2. Work with Docker Images

List Images

```
docker images
```

Lists all Docker images stored locally.

Pull an Image

```
docker pull ubuntu:22.04
```

Downloads the Ubuntu 22.04 image.

Inspect Image Metadata

```
docker image inspect ubuntu:22.04
```

Displays detailed info about the image.

🧱 3. Run and Manage Containers

Run a Basic Container

```
docker run ubuntu:22.04 cat /etc/passwd
```

Runs a container, executes a command, and exits.

Interactive Shell

```
docker run -it --name app1 ubuntu:22.04
```

Starts an interactive terminal session inside Ubuntu.

Detached Mode

```
docker run -itd --name app3 ubuntu:22.04
```

Runs a container in the background.

Check Containers

```
docker ps         # Running containers
docker ps -a      # All containers (running + stopped)
```

🛠️ 4. Container Interaction

Execute Commands Inside a Running Container

```
docker exec -it app3 bash
```

Opens a new terminal session inside the container.

Attach to a Container's Main Process

```
docker attach app3
```

Connects your terminal to a running container.

🏷️ 5. Tag & Save Docker Images

Tag an Image

```
docker tag <image_id> ubuntu:24.04
```

Assigns a new name/tag to an existing image.

Commit a Container as an Image

```
docker commit app1 app:v1
```

Creates a new image from a container.

Save and Load Images

```
docker save app:v1 -o app.tar
docker load < app.tar
```

Export and import images as tar files.

🌐 6. Port Mapping Example

Run Web Server in Container

```
docker run -d --name app1 -p 8080:80 httpd:latest
```

Starts an Apache server accessible via localhost:8080.

Resource Limits

```
docker run -d --name app1 --memory 256m --cpus 1 httpd:latest
```

Limits the container to 256MB RAM and 1 CPU.

📦 7. Docker Volumes

Create and Inspect Volumes

```
docker volume create myvol
docker volume inspect myvol
```

Manages persistent data using named volumes.

Run Container with Volume

```
docker run -d --name web1 -p 8080:80 -v myvol:/usr/share/nginx/html nginx:latest
```

Mounts the volume to serve content from Nginx.

---

8. Bind Mounts

Bind a Local Directory

```
docker run -d --name app3 -p 8090:80 -v /myapp:/usr/share/nginx/html nginx:latest
```

Mounts a host directory to the container.

Alternative Syntax

```
docker run -d --name app4 -p 8091:80 --mount
type=bind,src=/myapp,dst=/usr/share/nginx/html nginx:latest
```

---

9. Monitor Containers

```
docker stats
docker logs web1
docker logs -f web1
docker top web1
```

Monitor performance, logs, and running processes.

---

10. System & Container Info (Optional but Useful)

```
lscpu          # CPU architecture
free -h        # Memory usage
docker container inspect app4
```

---

11. Copy Files to and from Containers

```
docker cp mango.txt web1:/tmp
docker cp web1:/tmp/apple.txt .
```

Transfer files between the host and a container.

Summary of Core Docker Workflow

- Pull image:

  ```
  docker pull ubuntu:22.04
  ```

- Run container:

  ```
  docker run -it --name mycontainer ubuntu:22.04
  ```

- Inspect:

  ```
  docker ps
  docker inspect
  ```

- Interact:

  ```
  docker exec -it <name> bash
  ```

- Persist data:
  Use volumes or bind mounts

- Save & move images:

  ```
  docker save
  docker load
  ```

Docker Networking and Container Commands

A list of common Docker commands with short descriptions, focusing on container inspection, networking, and container lifecycle.

Container and Process Inspection

```
docker container inspect web6
```

Inspect details of the container named `web6`.

```
docker ps
```

List running Docker containers.

```
docker exec -it web5 bash
```

Open an interactive terminal inside the running container `web5`.

Networking Inspection

```
docker network ls
```

List all Docker networks.

```
docker network inspect bridge
```

Display detailed information about the default `bridge` network.

```
ip a
```

Show all IP addresses assigned to network interfaces (host system).

Container Cleanup

```
docker rm -f $(docker ps -aq)
```

Forcefully remove all containers (running and stopped).

Running Containers

```
docker run -d nginx:latest
```

Run a new detached container with the latest `nginx` image.

```
docker run -d --name app1 nginx:latest
```

Run a new detached `nginx` container with the name `app1`.

```
docker run -d --name app2 nginx:latest
```

Run a new detached `nginx` container with the name `app2`.

Interactive Container Access

```
docker exec -it app2 bash
```

Open an interactive shell in the `app2` container.

```
docker exec -it web2 bash
```

Open an interactive shell in the `web2` container.

```
docker exec -it web1 bash
```

Open an interactive shell in the `web1` container.

Custom Docker Network

```
docker network create mynet
```

Create a custom Docker bridge network named `mynet`.

```
docker run -d --name web1 --network mynet nginx:latest
```

Run `nginx` in a detached container named `web1` on the `mynet` network.

```
docker run -d --name web2 --network mynet nginx:latest
```

Run `nginx` in a detached container named `web2` on the `mynet` network.

```
docker network inspect mynet
```

Display detailed information about the `mynet` network.

Display detailed information about the `wordpress and mysql` container to container communication.

```
docker run -d --name wordpress -p 8082:80 --network mynet wordpress:latest
```

Mysql database contianer using `mynet`.

```
docker run -d --name mydb --network mynet -e MYSQL_ROOT_PASSWORD=test123 -e
MYSQL_DATABASE=wpdb mysql:5.7
```

```
docker network rm mynet
```

Remove the custom Docker network `mynet`.
