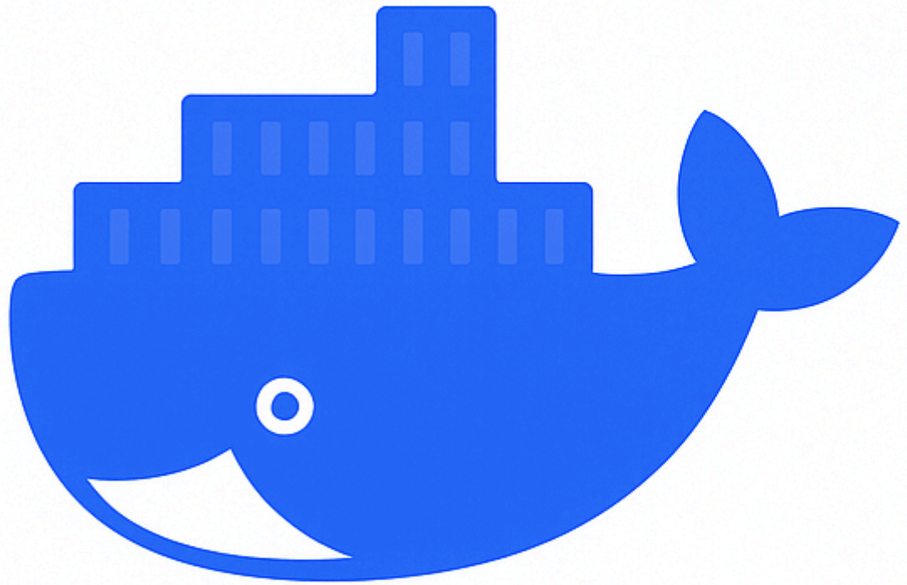
# Docker Basics

## — Part 2: Images, Volumes & Compose

A beginner-friendly guide
from my own project experience



Zainab Aitbelaid

# What is an Image?

- **Definition** A Docker image is a lightweight, standalone package that contains everything needed to run an application — code, runtime, libraries, and dependencies.
- **Immutable** Once created, images are read-only. Any changes made while running are stored in a container layer, leaving the image unchanged.
- **Portability** The same image can run on any system with Docker installed, ensuring consistency between development, testing, and production
- **Tagged Versions** Images can have multiple versions (tags), e.g. nginx:latest or nginx:1.25

Zainab Aitbelaid

# What is a Dockerfile?

A Dockerfile is a text file that contains a set of instructions for building a Docker image.

Key instructions are:

- FROM – specify base image
- RUN – execute commands
- COPY – copy files into image
- EXPOSE – document ports
- CMD – set the default command

# Docker Hub

Docker Hub is a public registry for finding and sharing container images.

- **Official vs. Community Images**
  Official Images are verified and maintained, while Community Images are shared by users
- **How It Works**
  You can push your own images to Docker Hub or pull existing images for your use
- **Flexibility and Collaboration**
  Get base images for your projects, and share your images with the team

Zainab Aitbelaid

# Basic Commands

**docker run** Start a new container from an image

**docker ps** List running containers

**docker stop** Stop a running container

**docker rm** Remove a container

**docker images** List downloaded images

**docker build** Build a new image from a Dockerfile

Zainab Aitbelaid

# Docker Networking

## Bridge (default)

Default mode for containers. Each container gets a private IP address
Containers can communicate with each other through the bridge network

## Host

Removes network isolation between the container and host
Shares the host’s IP and ports
Faster but less secure

## None

Disables all networking for a container
Used for isolated workloads with no external communication

**Key Idea:** Networking lets containers talk to each other or to the outside world

Zainab Aitbelaid

# Docker Volumes
## (Data Persistence)

### Why Volumes?

Containers are ephemeral — when deleted, all data inside is lost

### Types of Volumes

**Named Volumes**

- Managed by Docker
- Stored in Docker's default location

**Bind Mounts**

- Map host directory to container
- Offer more control

### Benefits

- Data persists even if a container is removed
- Share data between multiple containers
- Useful for databases, logs, configs, uploads

### Key Idea:

Volumes make containers stateful

Zainab Aitbelaid

# Docker Compose

- **What is it?**

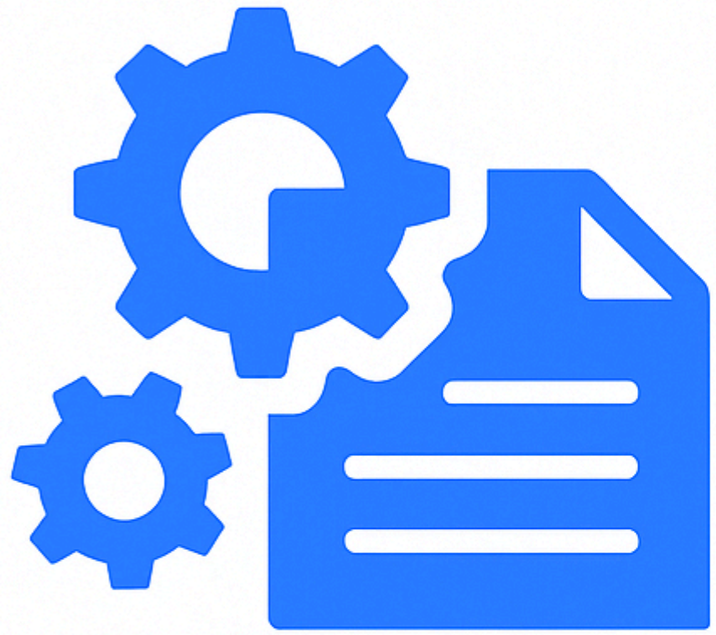
  A tool that allows you to define, configure, and run multi-container applications with a YAML file (docker-compose.yml)

- **Why use it?**

  Run multiple services (web, database, cache) together

  Replace long *docker run* commands with a single config file

  Ensure consistent setup acrross teams and environments



Key idea: Docker Compose lets you manage multi-container apps as one project.

# Docker Compose (Features)

- **Define everything in YAML**
  - Services (e.g., web, db, cache)
  - Networks for communication
  - Volumes for persistence
- **Simple commands**
  - Start with `docker-compose up`
  - Stop & clean with `docker-compose down`
- **Consistency**
  - One config file for the whole team → same setup everywhere
- **Efficiency**
  - Run, stop, and rebuild multiple containers with just one command

**Key Idea:** Docker Compose makes running multi-container apps repeatable, consistent, and efficient

Zainab Aitbelaid

# Wrap-up

✅ **You now know**: Images, Containers, Dockerfiles, Docker Hub, Commands, Networking, Volumes, and Compose.

🚀 With this, you’re ready to use Docker in real projects.

🙌🏻 Thanks for following this series!