Learn Docker in 1 Week

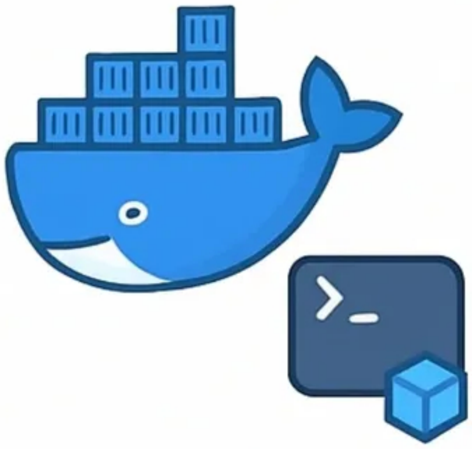
- **Day 1:** Docker Fundamentals
- **Day 2:** Installing Docker + CLI Tour
- **Day 3:** Containers, Images & Lifecycle
- **Day 4:** Dockerfile, Build, Tag, Push
- **Day 5:** Volumes, Bind Mounts & Persistence
- **Day 6:** Docker Networking & Port Mapping
- **Day 7:** Docker Compose & Final Project

@devops__community

Learn Docker in 1 Week

Day 1: Docker Fundamentals

- **What is Docker?**
  Docker is a platform that allows you to packagé applications with all their dependencies irto containers. These containers can run consistently on any environment—from your laptop to a production server.



- **Why Containers?**
  Containers are livahtwelgt, portable. and fast compare virtual machines.
- **Registry:** A place to store and distribute Dockér images (e.g. Docker Hub)

- **Key Concepts:**
  Image: A snnapshof of your application + environment.
  Registry: A runnning instance of an image.
  Regtsrg: (Docker Hub)

✅ **Hands-On: docker run hello-world**

@devops..community

Day 2: Installing Docker + CLI Tour

Install Docker

- **Linux:** Install via package manager (apt, dhf, etc.)
- **Windows/Mac:** Install Docker Desktop.

Key Commands to Know:

`docker version`	Shows Docker version
`docker info`	Shows system-ivde info

Docker Daemon

Docker works as a client-server architecture.
The **docker** command talks to the Docker daemon, which does the heavy lifting.

> After installing, test with:

```
docker run busybox echo "Docker is ready!"
```

Day 3: Containers, Images & Lifecycle

Pulling & Running Images:

- `docker pull nginx`
- `docker run -d -p 8080:80 nginx`

Managing Containers:

- `docker ps     # Running containers`
- `docker stop <id> Stop a container`
- `docker rm <id> # Remove a container`
- `docker logs <id> # View logs`

Lifecycle:

`create → run → pause → stop → remove`

Every container has a lifecycle. Managing this well ensures optimized system resources.

Inspect:

```
docker inspect <container>
```

[illegible]

Learn Docker in 1 Week

Day 4: Dockerfile, Build, Tag, Push

Dockerfile: Custom Image Blueprint

```
FROM python:3.11
WORKDIR /app
COPY . .
RUN pip install -r requirements.txt
CMD ["python","app.py"]
```

Build Image: docker build –t myapplatest .

Tag and Push to Docker Hub:

```
docker tag myapp yourusername/myapp
docker push yourusername/myapp
```

Multi-stage builds (Intro)

Use it to reduce final image size and isolate build vs run layers.

@devops__community

Day 5: Volumes, Bind Mounts & Persistence

What is a Volume?

A Docker-managed directory on host
for persistent storage.

```
docker volume create myvoldocker run -v myvol:/data
nginx
```

Bind Mounts

Mount a specific host path inside concatainer.

```
docker run -v $(pwd):/app myapp
```

Use Cases

- Keep databases (eg., PostgreSQL) persistent
- Share logs or configuration between host and

Inspect volumes

```
docker volume inspect myvol
```

@devops_.community

Learn Docker in 1 Week

Day 6: Docker Networking & Port Mapping

Port Mapping: `docker run -p 8080:80 nginx`

Maps host port 8080 to container port 80.

Networks:

Bridge: Default network (for standalone containers.)

Host: Uses host's networktack (Linux-only)

Communicate Between Containers:

```
docker network create mynet
docker run --network=mynet
  -name webapp myapp
```

Now webapp can reach redis using container name.

DAY 7: DOCKER COMPOSE & FINAL PROJECT

YANL

→ **What is Docker Compose?**
A YAML-based tool to define and run muiti-container applications.

→ **Sample docker-compose.yml:**

```
version: "3
services:
  web:
    build:
    myvol
  regi
  redis
/>
```

```
version: "3
services:
  buidge ..
  ports: 5000
 >
redis
```

→ **Key Commands:**

```
docker—compose up
Stops all servīces down
```

🧪 **FINAL PROJECT:**
A Python Flask App that counts visits using Redis.

```
redis ⇒ 1
```

@devops__community