Containers and Docker: An Introduction

April 11, 2025

What are Containers?

- A container is a lightweight, standalone package of software.
- It includes everything needed to run an app: code, runtime, system tools, libraries.
- Containers isolate applications from their environments.
- Think of them as a more efficient, minimal alternative to virtual machines.

Why Use Containers?

- Portability — works on any system with a container runtime.
- Consistency — dev = test = prod.
- Speed — containers start in seconds.
- Efficiency — less overhead than full virtual machines.

Containers vs Virtual Machines

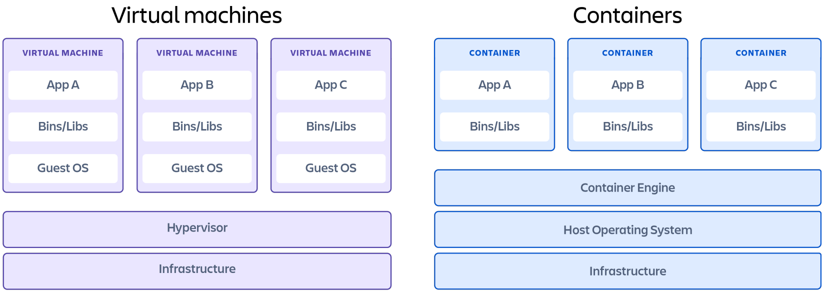

Figure: VMs vs Containers — Key Architectural Differences

What is Docker?

- Docker is a platform that makes it easy to build, run, and manage containers.
- Provides CLI tools, image building, container lifecycle management, and orchestration support.
- Uses Dockerfiles to define container environments.

Key Docker Terminology

Image	A read-only template used to create containers. It contains the app code, runtime, libraries, and dependencies.
Container	A runnable instance of an image. Containers are isolated, lightweight, and portable.
Dockerfile	A script of instructions to build a Docker image. Defines base image, files to copy, commands to run, etc.
Registry	A storage and distribution system for Docker images. Docker Hub is the default public registry.
Volume	A persistent storage mechanism for containers. Used to save data between container restarts.
Network	Docker allows isolated networks for containers to communicate securely.
Docker Engine	The runtime that builds and runs containers. It includes the daemon and CLI.

Basic Docker Commands

```bash
# Pull an image from the main registry
docker pull hello-world

# Run a container from an image
docker run hello-world

# List running containers
docker ps

# Stop and remove containers
docker stop container_id
docker rm container_id
```

Writing a Simple Dockerfile

```
# Use official Python image
FROM python:3.11-slim

# Set working directory
WORKDIR /app

# Copy local code
# the first . refers to the local files while the second is
    the destination inside the container
COPY . .

# Install dependencies
RUN pip install -r requirements.txt

# Run the app
CMD ["python", "main.py"]
```

Build and Run Your Image

```
# Build the Docker image
docker build -t my-python-app .

# Run it
docker run my-python-app
```

Advanced example

```
docker run -d --hostname my-rabbit \
           --name some-rabbit \
           -e RABBITMQ_DEFAULT_USER=user \
           -e RABBITMQ_DEFAULT_PASS=password \
           -p 5672:5672 \
           -p 15672:15672 \
           rabbitmq:3-management
```

What is Docker Compose?

- Tool for defining and running multi-container applications.
- Configured using a ‘docker-compose.yml‘ file.
- Useful for setting up dev environments and microservices.
- Especially useful when running multiple containers that depend on each other.
- For more complicated configurations like the previous slide the commands can get tedious.

docker-compose.yml Example

```yaml
services:
  rabbitmq:
    image: rabbitmq:3-management
    container_name: some-rabbit
    hostname: my-rabbit
    environment:
      RABBITMQ_DEFAULT_USER: user
      RABBITMQ_DEFAULT_PASS: password
    ports:
      - "5672:5672"   # AMQP protocol
      - "15672:15672" # Management UI
    restart: unless-stopped
```

```bash
# run the containers defined in the docker-compose.yml
docker compose up -d
```

Recap

- Containers isolate applications in lightweight environments.
- Docker simplifies working with containers.
- Dockerfiles define how to build images.
- Docker Compose manages multi-container setups.

References

- Official Docker Documentation:
  `https://docs.docker.com`
- Docker CLI Reference:
  `docker command reference`
- Docker Compose Docs:
  `https://docs.docker.com/compose/`
- Dockerfile Reference:
  `https://docs.docker.com/engine/reference/builder/`
- Atlassian: Containers vs. virtual machines:
  `https://www.atlassian.com/microservices/cloud-computing/containers-vs-vms`

Q & A

Questions?