Lecture 5 - Exercises

Torben Worm, Tobias Kristensen, & Nicolai Krogager

Agenda

- Docker
- Docker Compose
- Example
- Exercise: Containerization

What is Docker?

Docker is an open-source platform used to develop, ship, and run applications in containers.

Containers allow developers to package up an application with all parts it needs, such as libraries and other dependencies, and ship it all out as one package.

Containers isolate the application, reducing conflicts between systems.

Benefits of Docker

Consistency: Runs the same everywhere, reduces "it works on my machine" problems.

Isolation: Provides isolated environments, avoiding conflicts between system components.

Modularity and Scalability: Breaks application into modular pieces (microservices) that can scale independently.

Resource Efficient: Consumes fewer resources compared to VMs as it uses the host OS.

Docker Architecture

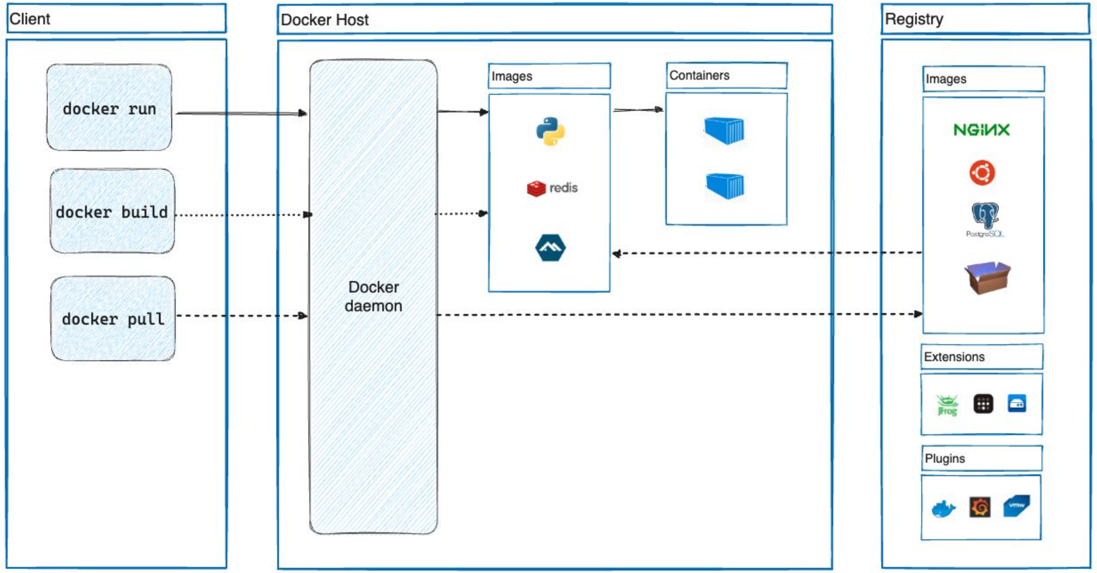
Docker Client: The command line tool used to interact with Docker.

Docker Daemon: Background service running on the host that builds, runs, and manages containers.

Docker Image: Read-only template used to create containers.

Docker Container: A runnable instance of a Docker image.

Docker Registry: A repository of Docker Images. For example Dockerhub or GitHub Container Registry (GHCR)



https://docs.docker.com/get-started/overview/

Basic Docker Commands

docker pull <image> Downloads a Docker image from Dockerhub or another container registry.

docker build -t <image-name> . Builds a Docker image from a Dockerfile in the current directory.

docker run <options> <image> Creates and starts a container from an image.

docker ps Lists running containers.

docker stop <container-id> Stops a running container.

Introduction to Docker Compose

Docker Compose is a tool for defining and running multi-container Docker applications.

Purpose

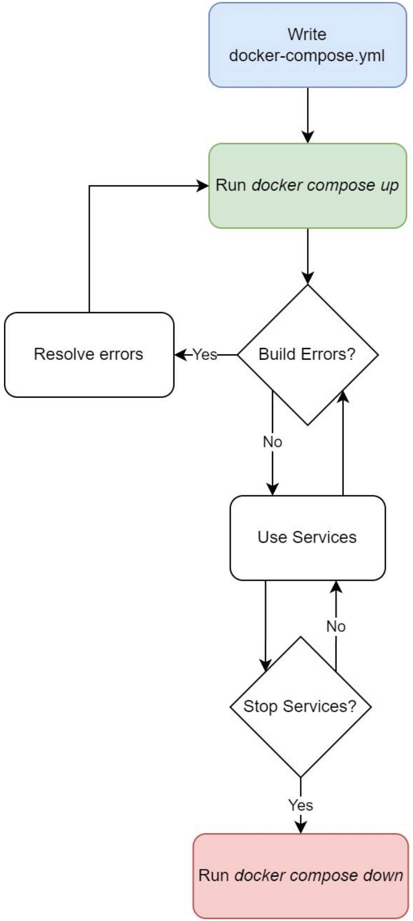
- Defines services, networks, and volumes in a single file.
- Efficient for managing multi-container setups and configurations.
- Simplifies the workflow, enabling developers to bring up and tear down environments with a few commands.

Docker Compose Workflow

docker-compose.yml: Configuration file defining services, networks, and volumes.

docker compose up: Command to start the services defined in the docker-compose file.

docker compose down: Command to stop the services defined in the docker-compose file.

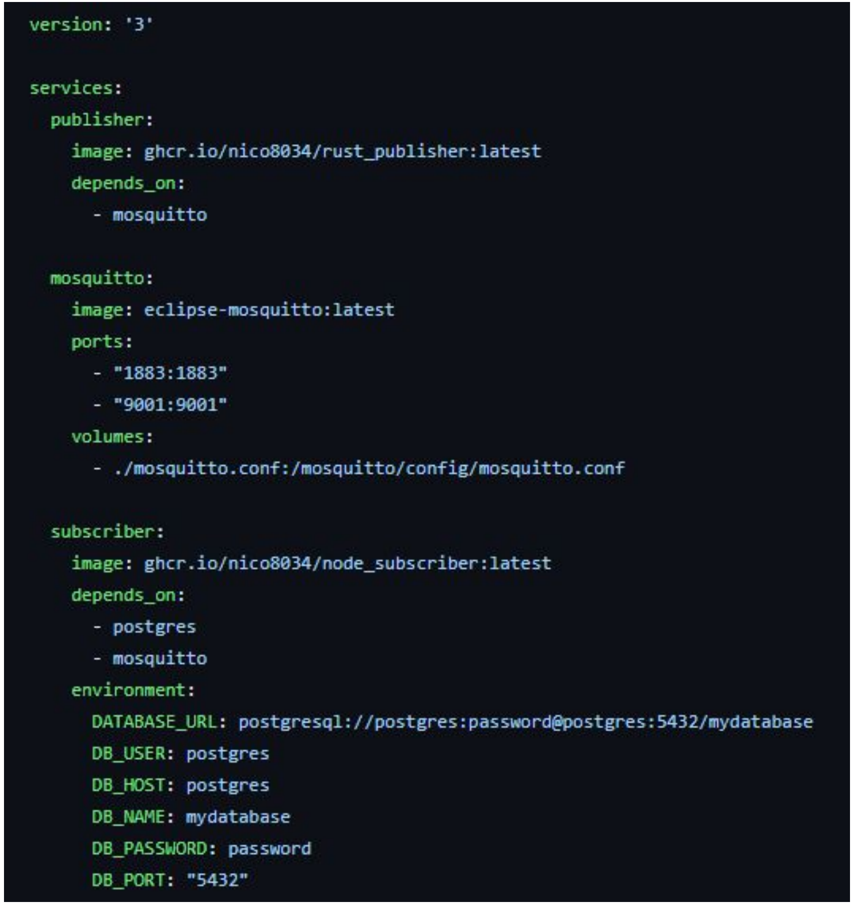


Docker Compose Example

A Docker Compose file can contain multiple services

Services can be built by referencing a Dockerfile or by referencing a pre-built image.

Other configurations such as volumes, environment variables, etc. can be defined for each service.

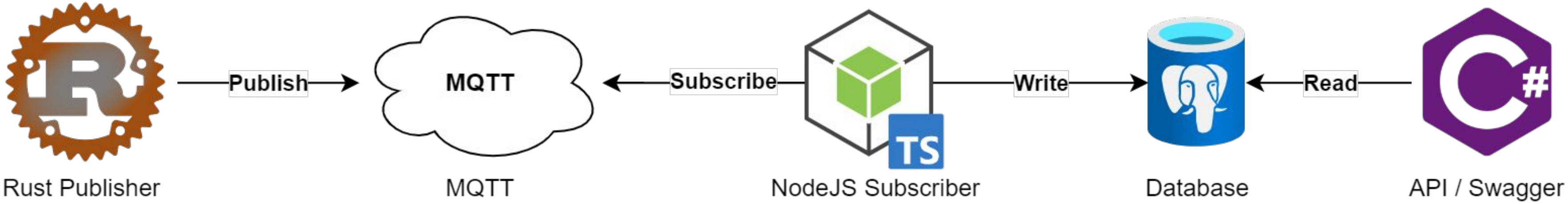


```
version: '3'

services:
  publisher:
    image: ghcr.io/nico8034/rust_publisher:latest
    depends_on:
      - mosquitto

  mosquitto:
    image: eclipse-mosquitto:latest
    ports:
      - "1883:1883"
      - "9001:9001"
    volumes:
      - ./mosquitto.conf:/mosquitto/config/mosquitto.conf

  subscriber:
    image: ghcr.io/nico8034/node_subscriber:latest
    depends_on:
      - postgres
      - mosquitto
    environment:
      DATABASE_URL: postgresql://postgres:password@postgres:5432/mydatabase
      DB_USER: postgres
      DB_HOST: postgres
      DB_NAME: mydatabase
      DB_PASSWORD: password
      DB_PORT: "5432"
```

What have we done so far?

- Programming Languages
  - purpose, quality attributes
- Databases
  - types, SQL, noSQL
- Message Busses
  - publish / subscribe
- Containerization
  - docker

Example


Rust Publisher
Publish
MQTT
MQTT
Subscribe
NodeJS Subscriber
Write
Database
Read
API / Swagger

Exercise 1 - Lecture 5

- Available on Itsleraning
- Use nico8034/docker_demo repo for inspiration