Introduction

Docker Concepts

- Docker
- Docker Images
- Docker Containers
- Docker Volumes
- Docker Networking
- Docker Compose

Docker Commands

- Basic Commands
- Managing Docker Containers
- Managing Docker Images
- Docker Compose Commands
- **Project Setup**
    - Dockerfile
    - Docker Compose File
- **Building and Running Containers**
    - Building Docker Images
    - Running Containers
- **Conclusion**

1. Introduction

Docker is an open-source platform that automates the deployment of applications inside lightweight, portable containers. Containers package an application with all its dependencies, making it easy to run on any system.

2. Docker Concepts

Docker: Docker is a platform that allows you to automate the deployment of applications inside containers.

Docker Images: An image is a read-only template used to create containers. It contains the application code, libraries, dependencies, and environment variables.

Docker Containers: A container is a runnable instance of an image. Containers are lightweight and provide a consistent environment across different systems.

Docker Volumes: Volumes are used for persistent storage. They allow data to persist even after a container is stopped or removed.

Docker Networking: Docker provides networking capabilities to enable communication between containers. By default, containers are connected to a bridge network.

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

3. Docker Commands

Basic Commands:

- `docker --version`: Check Docker version.
- `docker pull <image>`: Download an image from Docker Hub.
- `docker run <image>`: Run a container from an image.

Managing Docker Containers:

- `docker ps`: List running containers.
- `docker stop <container_id>`: Stop a running container.
- `docker start <container_id>`: Start a stopped container.
- `docker rm <container_id>`: Remove a stopped container.

Managing Docker Images:

- `docker images`: List Docker images.
- `docker rmi <image_id>`: Remove an image.
- `docker build -t <tag> .`: Build an image from a Dockerfile.

Docker Compose Commands:

- `docker-compose up`: Start services defined in `docker-compose.yml`.
- `docker-compose down`: Stop and remove containers, networks, and volumes.
- `docker-compose build`: Build or rebuild services.

4. Project Setup

Dockerfile: A Dockerfile is a script with instructions on how to build a Docker image. It typically includes instructions to set up the base image, copy application files, install dependencies, and configure the environment.

Unset

```
# Use JDK 21 slim version as the base image
FROM openjdk:21-jdk-slim

# Set the working directory in the container
WORKDIR /app

# Copy the packaged JAR file into the container
COPY target/BasicsOfDockerLab-0.0.1-SNAPSHOT.jar
/app/BasicsOfDockerLab-0.0.1-SNAPSHOT.jar

# Expose the application port
EXPOSE 4000

# Run the Spring Boot application
ENTRYPOINT ["java", "-jar", "BasicsOfDockerLab-0.0.1-SNAPSHOT.jar"]
```

Docker Compose File: A `docker-compose.yml` file defines multi-container applications. It specifies the services, networks, and volumes needed for the application.

Unset

```
version: '3.8'

services:
 app:
   build: .
   ports:
     - "4000:4000"
   environment:
     - SPRING_DATASOURCE_URL=${SPRING_DATASOURCE_URL}
     - SPRING_DATASOURCE_USERNAME=${SPRING_DATASOURCE_USERNAME}
     - SPRING_DATASOURCE_PASSWORD=${SPRING_DATASOURCE_PASSWORD}
   depends_on:
     - db

 db:
   image: postgres:15
   environment:
     - POSTGRES_DB=${POSTGRES_DB}
```

```
    - POSTGRES_USER=${POSTGRES_USER}
    - POSTGRES_PASSWORD=${POSTGRES_PASSWORD}
  ports:
    - "5432:5432"
  volumes:
    - postgres_data:/var/lib/postgresql/data

volumes:
 postgres_data:
```

5. Building and Running Containers

Building Docker Images:

Unset

```
docker build -t BasicsOfDockerLab .
```

6. Conclusion

Docker simplifies the deployment process by using containers to encapsulate applications and their dependencies. By understanding Docker concepts and commands, you can effectively build, manage, and deploy containerized applications.