DATE:18/07/2024

DAY4

1.Docker

Docker is a tool designed to make it easier to create, deploy, and run applications by using containers. Containers are lightweight, portable, and ensure that an application runs the same way regardless of where it is deployed.

Key Components of Docker:

- **Docker Engine:** The runtime that builds and runs Docker containers.
- **Docker Images:** Read-only templates used to create containers. Images are built from Dockerfiles.
- **Docker Containers:** Instances of Docker images. Containers are isolated environments where applications run.
- **Docker Hub:** A public registry for sharing Docker images.
- **Docker Compose:** A tool for defining and running multi-container Docker applications.
- **Dockerfile:** A text file with instructions on how to build a Docker image.

2. Docker Core Concepts

Containers vs Virtual Machines

- Containers: Share the host OS kernel, use less memory, and start faster.
- Virtual Machines: Include a full OS and are more resource-intensive.

Images and Containers

- Image: A blueprint for creating containers. Immutable and can be versioned.
- Container: A running instance of an image. Mutable and can be stopped, started, and deleted.

Docker Architecture

- Client: The Docker CLI tool used to interact with the Docker daemon.
- Daemon: The background service that handles Docker containers.
- Registry: A repository for Docker images. Docker Hub is the default public registry.

➢ Repository: A collection of related Docker images, typically for a single application.

3. Basic Docker Commands

A list of essential Docker commands for managing images, containers, and other Docker resources.

Managing Docker Images

➢ **List Images:**
 docker images

➢ **Pull an Image:**

 docker pull <image_name>:<tag>

 Example:

 docker pull nginx:latest

➢ **Build an Image:**

 docker build -t <image_name>:<tag> <path_to_dockerfile>

 Example:

 docker build -t myapp:1.0 .

➢ **Remove an Image:**

 docker rmi <image_id>

➢ **Tag an Image:**

 docker tag <source_image>:<source_tag> <target_image>:<target_tag>

 Example:

 docker tag myapp:1.0 myrepo/myapp:latest

Managing Docker Containers

➢ **List Containers:**

 docker ps

 Add -a to list all containers including stopped ones:

 docker ps -a

- ➤ **Run a Container:**

  docker run [OPTIONS] <image_name>:<tag>

  Example:

  docker run -d -p 80:80 nginx:latest

  -d: Run in detached mode

  -p: Map host port to container port

- ➤ **Stop a Container:**

  docker stop <container_id>

- ➤ **Remove a Container:**

  docker rm <container_id>

- ➤ **View Container Logs:**

  docker logs <container_id>

- ➤ **Execute Commands in a Running Container:**

  docker exec -it <container_id> <command>

  Example:

  docker exec -it mycontainer /bin/bash

Managing Docker Networks

- ➤ **List Networks:**

  docker network ls

- ➤ **Create a Network:**

  docker network create <network_name>

- ➤ **Inspect a Network:**

  docker network inspect <network_name>

Managing Docker Volumes

- ➤ **List Volumes:**

  docker volume ls

- ➢ **Create a Volume:**

    docker volume create <volume_name>

- ➢ **Inspect a Volume:**

    docker volume inspect <volume_name>

- ➢ **Remove a Volume:**

    docker volume rm <volume_name>

4. Docker Use Cases

Development and Testing

- ➢ Create consistent development environments.
- ➢ Test applications in isolated containers.

Deployment

- ➢ Package applications with their dependencies.
- ➢ Deploy applications across different environments without compatibility issues.

Continuous Integration and Continuous Deployment (CI/CD)

- ➢ Automate the building, testing, and deployment of applications.
- ➢ Integrate Docker with CI/CD tools like Jenkins, GitLab CI, and GitHub Actions.

Microservices Architecture

- ➢ Develop and deploy microservices as separate containers.
- ➢ Simplify scaling and management of microservices.

Legacy Application Modernization

- ➢ Containerize legacy applications to run in modern environments.

5. Docker Best Practices

Write Efficient Dockerfiles

- ➢ Minimize Layers: Combine commands where possible.
- ➢ Use Official Images: Start with well-maintained base images.
- ➢ Specify Exact Versions: Avoid using latest tag for reproducibility.

- ➢ Order Instructions Wisely: Place frequently changing instructions at the bottom.

Example Dockerfile:

dockerfile

Use an official Node.js runtime as a parent image

FROM node:14

Set the working directory in the container

WORKDIR /usr/src/app

Copy package.json and package-lock.json to the working directory

COPY package*.json ./

Install dependencies

RUN npm install

Copy the rest of the application code

COPY . .

Expose the port the app runs on

EXPOSE 8080

Define the command to run the app

CMD ["node", "app.js"]

Secure Docker Containers

- ➢ Scan Images: Use tools to scan for vulnerabilities.
- ➢ Use Least Privilege: Avoid running containers as root.
- ➢ Regular Updates: Update images to include security patches.

Optimize Docker Images

- ➢ Use Multi-Stage Builds: Reduce image size and remove build dependencies.
- ➢ Clean Up Unused Images:

  docker image prune

Monitor Docker Containers

➢ Use Docker Stats:

docker stats

➢ Leverage Docker Logging: Integrate with logging solutions like ELK Stack, Prometheus, or Grafana.

6. Docker Tools and Ecosystem

Docker Compose

➢ Purpose: Define and manage multi-container applications.

Basic Commands:

➢ Start Services:

docker-compose up

➢ Stop Services:
docker-compose down

➢ Build Images:

docker-compose build

Docker Swarm

➢ Purpose: Native clustering and orchestration for Docker.

Basic Commands:

➢ Initialize Swarm:

docker swarm init

➢ Create a Service:

docker service create --name myservice nginx

Kubernetes

➢ Purpose: Advanced container orchestration platform.

Basic Commands:

➢ Deploy an Application:

kubectl create deployment myapp --image=myimage

➢ Scale Deployment:

kubectl scale deployment myapp --replicas=3

Summary:

Component	Description
Docker	Platform for building, running, and managing containers.
Image	A read-only template used to create containers.
Container	A running instance of an image.
Dockerfile	A file with a set of instructions to build a Docker image.
Docker Hub	A public registry to share and download Docker images.
Docker Compose	Tool for defining and running multi-container applications.
Docker Swarm	Docker's native clustering and orchestration solution.

7.Learning Resources:

Books

➢ Docker Deep Dive by Nigel Poulton
➢ Docker Up & Running by Kelsey Hightower, Brendan Burns, and Joe Beda