Docker Concepts and Commands

1. Introduction to Docker

Docker is an open-source platform designed to automate the deployment, scaling, and management of applications using containerization. Containers provide a lightweight, consistent environment across various computing environments.

2. Key Docker Concepts

- **Container**: A lightweight, standalone executable package that includes everything needed to run a piece of software, including code, runtime, libraries, and system settings.
- **Image**: A lightweight, immutable file that contains the source code, libraries, dependencies, tools, and other files needed for running applications. Containers are instances of images.
- **Dockerfile**: A text file containing a series of instructions used to build a Docker image. Each instruction in a Dockerfile creates a layer in the image.
- **Docker Hub**: A public registry that hosts Docker images. You can also create private registries to store your images.
- **Volumes**: Mechanisms for persisting data in containers. Volumes store data outside the container lifecycle and can be shared across containers.
- **Docker Compose**: A tool for defining and running multi-container Docker applications. It uses a YAML file to configure the application's services.

3. Docker Commands Overview

3.1 Docker Installation and Configuration

- **Check Docker version**:

```
docker --version
```

Displays the installed version of Docker.

- **Start Docker**:

```
systemctl start docker
```

Starts the Docker service on your system.

- **Enable Docker on startup**:

```
systemctl enable docker
```

Configures Docker to start automatically when the system starts.

3.2 Docker Image Management

- **Pull an image from Docker Hub**:

```
docker pull <image_name>
```

Downloads an image from a Docker registry.

- **List Docker images**:

```
docker images
```

Lists all Docker images available locally.

- **Build an image from a Dockerfile**:

```
docker build -t <image_name> .
```

Builds a Docker image from a Dockerfile in the current directory.

- **Tag an image**:

```
docker tag <image_id> <repository_name>:<tag>
```

Tags a local image with a specific tag, which can be used to push to a registry.

- **Remove a Docker image**:

```
docker rmi <image_name>
```

Deletes an image from the local storage.

3.3 Docker Container Management

- **Run a Docker container**:

```
docker run -d -p 8080:80 <image_name>
```

Runs a Docker container from an image, exposing port 80 in the container to port 8080 on the host machine.

- **List running containers**:

```
docker ps
```

Displays a list of currently running containers.

- **List all containers**:

```
docker ps -a
```

Displays all containers (running and stopped).

- **Stop a container**:

```
docker stop <container_id>
```

Stops a running container.

- **Start a stopped container**:

```
docker start <container_id>
```

Starts a container that was previously stopped.

- **Remove a container**:

```
docker rm <container_id>
```

Removes a stopped container.

- **Execute a command in a running container**:

```
docker exec -it <container_id> <command>
```

Runs a command (such as opening a shell) inside a running container.

- **View container logs**:

```
docker logs <container_id>
```

Retrieves logs from a running or stopped container.

3.4 Docker Volume Management

- **Create a volume**:

```
docker volume create <volume_name>
```

Creates a new Docker volume to persist data.

- **List Docker volumes**:

```
docker volume ls
```

Lists all Docker volumes.

- **Remove a volume**:

```
docker volume rm <volume_name>
```

Removes a specific volume.

- **Mount a volume to a container**:

```
docker run -d -v <volume_name>:/path/in/container <image_name>
```

Mounts a volume to a specific directory inside a container.

3.5 Docker Compose Commands

- **Run services defined in docker-compose.yml**:

```
docker-compose up
```

Starts all services defined in a docker-compose.yml file.

- **Run services in the background (detached mode)**:

```
docker-compose up -d
```

Starts services in detached mode (in the background).

- **Stop services**:

```
docker-compose down
```

Stops and removes all services, containers, and networks created by docker-compose up.

- **View logs for all services**:

```
docker-compose logs
```

Displays the logs from all services in the Compose file.

3.6 Docker Networking

- **List Docker networks**:

```
docker network ls
```

Displays all Docker networks.

- **Create a Docker network**:

```
docker network create <network_name>
```

Creates a new custom network.

- **Run a container on a custom network**:

```
docker run -d --network=<network_name> <image_name>
```

Runs a container on a specific network.

- **Inspect a Docker network**:

```
docker network inspect <network_name>
```

Shows detailed information about a Docker network.

4. Best Practices with Docker

1. **Keep Images Lightweight**: Use small base images like alpine to reduce image size.
2. **Use Volumes for Data Persistence**: Store important data in volumes to persist across container restarts.
3. **Multi-stage Builds**: Use multi-stage Docker builds to reduce image size and improve efficiency.
4. **Tagging**: Always use meaningful tags for your images to differentiate between versions.