Introduction to Docker

Docker is a powerful platform for developing, shipping, and running applications.Docker ensures that our application runs the same way across different environments, whether it's your local machine, a testing server, or in production. This consistency reduces the chances of "it works on my machine" issues.

Docker Images

- Docker images are lightweight, standalone, executable packages that contain everything needed to run a piece of software, including the code, runtime, libraries, and dependencies.

Key Concepts:

- Base Images: Fundamental images used as a starting point for creating custom images.
- Layers: Docker images are composed of multiple read-only layers, which are stacked on top of each other.

Managing Images:

1. **Build an Image:**
   docker build -t <image_name> <path_to_dockerfile>
2. **List Images:**
   docker images
3. **Pull an Image from Docker Hub:**
   docker pull <image_name>:<tag>
4. **Remove an Image:**
   docker rmi <image_name>
5. **Tag an Image:**
   docker tag <source_image> <target_image>:<tag>
6. **Push an Image to Docker Hub:**
   docker push <image_name>:<tag>
7. **Inspect an Image:**
   docker image inspect <image_name>
8. **Prune Unused Images:**
   docker image prune

Docker Containers

What are Docker Containers?

- Docker containers are lightweight, portable, and self-sufficient execution environments that run instances of Docker images.

Key Concepts:

- Isolation: Containers provide process isolation, ensuring that applications run consistently across different environments.
- Immutability: Containers are immutable, meaning they can be easily replaced or updated without affecting the underlying system.

Managing Containers:

- **List Running Containers:**
  docker ps
- **List All Containers (Including Exited Ones):**
  docker ps -a
- **Stop a Running Container:**
  docker stop <container_id>
- **Start a Stopped Container:**
  docker start <container_id>
- **Restart a Container:**
  docker restart <container_id>
- **Remove a Container:**
  docker rm <container_id>
- **Execute a Command Inside a Running Container:**
  docker exec -it <container_id> <command>
- **Inspect a Container:**
  docker inspect <container_id>
- **View Logs of a Container:**
  docker logs <container_id
- **Rename a Container:**
  docker rename <old_name> <new_name>

Docker Compose

What is Docker Compose?

- Docker Compose is a tool for defining and running multi-container Docker applications. It allows you to define application services in a YAML file and run them with a single command.

Key Concepts:

- Service Definitions: Defining individual services and their configurations.
- Networking: Automatic creation of networks for communication between containers.
- Volume Mounts: Mounting host directories or named volumes into containers for persistent storage.

Managing Applications with Docker Compose:

- Defining Services: Creating a `docker-compose.yml` file to define services.
- Starting and Stopping Applications: Using `docker-compose up` and `docker-compose down` commands.
- Scaling Services: Running multiple instances of a service with a single command.

Docker Compose Commands:

- docker-compose up: Starting Docker Compose services.
- docker-compose down: Stopping Docker Compose services.

Conclusion

Docker revolutionizes the way we develop, ship, and deploy applications by providing a consistent and efficient environment across different platforms. By mastering Docker and its associated tools, developers can streamline their workflows and accelerate the pace of software delivery.