**Getting Started with Docker — Notes for Beginners**

**1. What is Docker?**

- Docker is a **container platform** that lets you build, test, and deploy applications quickly in isolated environments.
- You define your application and all of its dependencies in a **Dockerfile**, which is used to build a **Docker image**.
- Docker images ensure that your application runs the **same way on any environment**.

---

**2. Why Use Docker?**

Docker helps you:

- Ship code faster.
- Gain more control over application environments.
- Deploy and scale containers easily.
- Perform rollbacks and troubleshoot efficiently.
- Save resources by efficient utilization.
- Move applications seamlessly from **development to production**.

---

**3. How Does Docker Work?**

- Docker uses **containerization**, which is a lightweight alternative to traditional virtualization.
- **Containers** encapsulate an application with its environment, libraries, and dependencies so it can run anywhere.
- Unlike **virtual machines (VMs)**, containers don't need a full guest OS — they share the host OS kernel, making them faster and smaller.

---

**4. Docker Architecture**

Docker follows a **client-server architecture** consisting of:

**Docker Daemon**

- Listens for API requests from the Docker client.

- Manages Docker objects like images, containers, and networks.

**Docker Client**

- The interface you use to run Docker commands (docker run, docker pull, etc.).
- Sends commands to the daemon.

**Docker Registries**

- Docker images are stored in registries.
- Default public registry: **Docker Hub**.
- You can pull images from Docker Hub or push your own images.

---

**5. Dockerfile**

- A **Dockerfile** is like a recipe that describes how to build a Docker image.

**Basic Steps**

1. Create a file named Dockerfile.
2. Docker automatically uses it when building an image:
3. docker build -t myimage:1.0 .
4. Commands in the RUN section execute during the build.
5. Commands in the CMD section execute when running a container.

**Example Dockerfile**

FROM ubuntu

MAINTAINER <paras@gmail.com>

RUN apt-get update

CMD [“echo”, “Hello World”]

---

**6. Docker Image**

- A Docker image is a **read-only template** used to create containers.
- It’s similar to a snapshot that includes all application dependencies.
- Images are **immutable** once built.

- They can be stored locally or in remote registries like Docker Hub.

**Basic Image Commands**

```
docker pull ubuntu:18.04      # Download image
docker images                 # List images
docker run image              # Run a container from an image
docker rmi image              # Delete an image
docker rmi $(docker images -q) # Delete all images
```

---

**7. Image Layers**

- Docker images are built in **layers** — each command in a Dockerfile creates a new layer.
- Layers are identified using SHA-256 hashes.
- The first 12 characters of the hash often appear as the **IMAGE ID** in Docker commands.

---

**8. Containers**

- A **container** is a **runnable instance** of an image.
- It isolates your application and runs independently.
- Containers can be connected to networks and have attached storage.
- Deleting a container typically removes its data unless you use **volumes** to persist it.

**Basic Container Commands**

```
docker ps                 # List running containers
docker run -it ubuntu     # Run a container interactively
docker start name         # Start container
docker kill name          # Stop container
docker rm name            # Remove container
docker rm $(docker ps -a -q) # Remove all containers
```

---

**9. What Next?**

After learning the basics:

- Practice building and running images and containers.
- Explore data persistence using Volumes.
- Learn Docker networking and orchestration tools like Docker Compose and Kubernetes