Course: DevOps Name: CH.Lakshmi Priyanka

Module: Dockers Mail Id:chlakshmipriyanka9@gmail.com

Topic: DOCKER Batch No: 115 9 AM

Trainer Name: Mr. Madhukar Assignment: 06

Date of Submission: 8 - DEC-2024

1) What about DOCKER architecture?

^ DOCKER IS A PLATFORM AND TOOL THAT ENABLES DEVELOPERS TO AUTOMATE APPLICATIONS ' DEPLOYMENT INSIDE LIGHTWEIGHT, PORTABLE CONTAINERS.

^ CONTAINERS ARE STANDALONE, EXECUTABLE PACKAGES THAT INCLUDE EVERYTHING NEEDED TO RUN A PIECE OF SOFTWARE, INCLUDING THE CODE, RUNTIME, LIBRARIES, AND SYSTEM TOOLS.

^ DOCKER USES CONTAINERIZATION TECHNOLOGY TO PROVIDE A CONSISTENT AND REPRODUCIBLE ENVIRONMENT ACROSS DIFFERENT DEPLOYMENT ENVIRONMENTS.

Docker Daemon:

^ Background process managing Docker containers. Listens for Docker API

requests. **Docker Client:**

^ Command-line tool for users to interact with the Docker daemon. Issues command to build, ship, and run containers.

Docker Images:

^ Lightweight, standalone, and executable packages. Include code, runtime, libraries, and system

tools. **Docker Containers:**

^ Instances of Docker images running as isolated processes—portable, consistent application environments.

Docker Registry:

^ Stores Docker images. Docker Hub is a public registry; private registries also

exist. **Docker Compose:**

Tool for defining and running multi-container Docker applications. Uses YAML files to specify application stacks.

2) WHAT ARE THE DIRECTIVES?

These directives help define how to build a Docker image and how containers based on that image should behave.

- **FROM**: Specifies the base image.

Example: **FROM ubuntu:20.04**

- **RUN**: Executes commands in the image.

Example: **RUN apt-get update && apt-get install -y python3**

- **COPY / ADD:** Copies files into the image.

Example: **COPY ./app /app**

- **WORKDIR:** Sets the working directory.

Example: **WORKDIR /app**

- **EXPOSE**: Inform Docker about network ports.

Example: **EXPOSE 80**

- **CMD:** Provides default command.

Example**: CMD ["python3", "app.py"]**

- **ENTRYPOINT**: Configures container as an executable.

Example: **ENTRYPOINT ["python3", "app.py"]**

- **ENV**: Sets environment variables.

Example: **ENV APP_PORT=8080**

- **LABEL**: Adds metadata to the image.

Example: **LABEL maintainer="yourname@example.com"**

3) What are image-related commands?

^ These commands help you manage, build, and interact with Docker images during the development and deployment of containerized applications.

^ **List Images:**

Command: **docker images**

Purpose: Lists all Docker images on your machine.

- **Pull Image** :
  Command: **docker pull <image_name>**
  Purpose: Download a Docker image from a registry
- **build image:**

  Command: **docker build -t <image_name: tag>**

<path> Purpose: Builds a Docker image from a Docker file.

- **Remove Image:**

  Command: **docker rmi <image_name>**

  Purpose: Remove a Docker image from your machine.

- **Tag Image:**

  Command: **docker tag <source_image> <target_image:tag>** Purpose: Tags an image with a new name or version.

- **Push Image:**

  Command: **docker push <image_name>**

  Purpose: Uploads a Docker image to a registry.

- **Save Image**:

  command: **docker save -o <output_file.tar> <image_name:tag>** Purpose: Saves a Docker image to a tarball archive.

- **Load Image**:

  Command: **docker load -i <input_file.tar>**

  Purpose: Loads a Docker image from a tarball archive.

- **Image History:**

  Command: **docker history <image_name>**

  Purpose: Shows the history of an image, including its layers.

- **Inspect Image:**

  Command: **docker inspect <image_name>**

  Purpose: Displays detailed information about an image.

- **Prune Images:**

  Command: **docker image prune**

  Purpose: Removes unused images to free up disk space

4) What are container-related commands?

^ These commands help you manage the lifecycle, logs, and interactions with Docker containers during development and deployment.

- **Run Container**:
  Command: **docker run <image_name>**
  Purpose: Starts a new container based on an image.

- **List Containers:**
  Command: **docker ps**
  Purpose: Shows running containers.
- **List All Containers:**
  Command: **docker ps -a**
  Purpose: Lists all containers, including stopped ones.
- **Stop Container**: Command: **docker stop <container or container_name>** Purpose: Halts a running container.
- **Start Container:**
  Command: **docker start <container_id or container_name>**
  Purpose: Resumes a stopped container.
- **Remove Container:**
  Command: **docker rm <container_id or container_name>**
  Purpose: Deletes a stopped container.
- **Remove Running Container:**
  Command: **docker rm -f <container_id or container_name>**
  Purpose: Forces removal of a running container.
- **Inspect Container**:
  Command: **docker inspect <container_id or container_name>**
  Purpose: Displays detailed information about a container.
- **Logs from Container:**
  Command: **docker logs <container_id or container_name>**
  Purpose: Shows logs from a container.
- **Execute Command in Container:**
  Command: **docker exec -it <container_id or container_name> <command>** Purpose: Runs a command inside a running container.
- **Copy Files to/from Container:**
  Command: **docker cp <source_path> <container or container_name>:<destination_path>**
  Purpose: Copies files between the host and a container.
- **Pause/Unpause Container:**
  Commands: **docker pause <container or container name> and docker unpause <container or container_name>**
  Purpose: Pauses or unpauses a running container.

—THANK YOU —