Experiment no.6

Title: Implementation of Containerization using Docker

Objective: To implement Docker Containerization.

Theory :

● Containers: Containers are lightweight, stand-alone executable packages that include everything needed to run a piece of software, including the code, runtime, system tools, libraries, and settings. Containers ensure that an application runs consistently and reliably across different environments, from a developer's laptop to a production server.

● Images: Container images are the templates for creating containers. They are read-only and contain all the necessary files and configurations to run an application. Images are typically built from a set of instructions defined in a Dockerfile.

● Docker: Docker is a popular containerization platform that simplifies the creation, distribution, and management of containers. It provides tools and services for building, running, and orchestrating containers at scale.

● Isolation: Containers provide process and filesystem isolation, ensuring that applications and their dependencies do not interfere with each other. This isolation enhances security and allows multiple containers to run on the same host without conflicts.

Benefits of Containerization:

● Consistency: Containers ensure that applications run consistently across different environments, reducing the "it works on my machine" problem.

● Portability: Containers are portable and can be easily moved between different host machines and cloud providers.

● Resource Efficiency: Containers share the host operating system's kernel, which makes them lightweight and efficient in terms of resource utilization.

● Scalability: Containers can be quickly scaled up or down to meet changing application demands, making them ideal for microservices architectures.

● Version Control: Container images are versioned, enabling easy rollback to previous application states if issues arise.

● DevOps and CI/CD: Containerization is a fundamental technology in DevOps and CI/CD pipelines, allowing for automated testing, integration, and deployment.

Containerization vs. Virtualization:

● Containerization differs from traditional virtualization, where a hypervisor virtualizes an entire operating system (VM) to run multiple applications. In contrast:

● Containers share the host OS kernel, making them more lightweight and efficient.

● Containers start faster and use fewer resources than VMs.

● VMs encapsulate an entire OS, while containers package only the application and its.

[illegible] Python script that prints "Hello World".

Write a Dockerfile:

Create a file named "Dockerfile" in the same directory as the application. In the Dockerfile, specify the base image, copy the application into the container, and specify the command to run the application. Here's an example Dockerfile for a Python script:

- Use the official Python image as the base image FROM python:3.9
- Copy the Python script into the container COPY hello.py /app/

[illegible]

Run the Python script when the container starts CMD ["python", "hello.py"]

Build the Docker image: Run the following command to build the Docker image:

$ docker build -t myimage .

This command builds a new Docker image using the Dockerfile and tags theimage with the name "myimage".

Run the Docker container: Run the following command to start a new container based on the image:$ docker run --name mycontainer myimage.

This command starts a new container named "mycontainer" based on the"myimage" image and runs the Python script inside the container.

Verify the output: Run the following command to verify the output of the container:

$ docker logs mycontainer This command displays the logs of the container and should show the "HelloWorld" output.