

EXPERIMENT 9

Aim: To understand Docker Architecture and Container Life Cycle, install Docker and execute docker commands to manage images and interact with containers.

Theory:

Docker is a popular platform that enables developers to build, package, and deploy applications as lightweight, portable, and self-sufficient containers. These containers encapsulate all the necessary dependencies and libraries required for an application to run, ensuring consistency across different environments. Here is a theoretical overview of Docker: Containerization:

Docker utilizes containerization technology to create isolated environments for applications. Containers are lightweight, standalone, and executable packages that include everything needed to run an application, such as code, runtime, system tools, libraries, and settings. This isolation ensures that applications run consistently across different environments, from development to production.

Docker Engine:

At the core of Docker is the Docker Engine, which is responsible for building, running, and managing containers. It consists of the Docker daemon, which manages containers, images, networks, and volumes, and the Docker client, which allows users to interact with the daemon through the Docker API.

Docker Images:

Docker images are read-only templates used to create containers. They contain the application code, runtime, libraries, dependencies, and other files needed to run the application. Images are built using Dockerfiles, which are text files that define the steps needed to create the image.

Docker Containers:

Containers are instances of Docker images that are running as isolated processes on a host machine. They are lightweight, portable, and can be easily started, stopped, moved, and deleted. Containers provide a consistent environment for applications to run, regardless of the underlying infrastructure.

Benefits of Docker:

Portability: Docker containers can run on any platform that supports Docker, making it easy to deploy applications across different environments.

Efficiency: Containers share the host OS kernel, reducing overhead and improving resource utilization.

Isolation: Containers provide a level of isolation that helps prevent conflicts between applications and dependencies.

Scalability: Docker enables easy scaling of applications by quickly spinning up additional containers.

Output:



3

4

[illegible]

Conclusion:

Docker revolutionizes the software development and deployment process by providing a powerful platform for containerization. By encapsulating applications and their dependencies into lightweight, portable containers,