IMPLEMENTATION OF DOCKER CONTAINERS

---

1. Introduction

Containerization is a modern virtualization technique that allows applications to run in isolated environments known as containers. Unlike traditional virtual machines, containers share the host operating system kernel while maintaining application-level isolation. This approach significantly reduces resource usage and startup time, making containerization ideal for cloud computing, DevOps pipelines, and microservices architectures.

Docker is a widely used open-source container platform that simplifies application deployment by packaging software along with its dependencies into standardized units called images. These images can be executed as containers on any system with Docker installed, ensuring consistency across development, testing, and production environments.

This experiment demonstrates the installation of Docker on Kali Linux and the creation of multiple containers, including Hello-World, Ubuntu Operating System, Nginx Web Server, and Python Programming Environment. Through this implementation, fundamental concepts such as image pulling, container execution, port mapping, and isolated runtime environments are explored.

---

2. Objectives

The objectives of this experiment are:

- To install Docker Engine on Kali Linux
- To understand Docker architecture and containerization concepts
- To execute a basic Hello-World container
- To create an Ubuntu OS container and interact with it
- To deploy an Nginx web server container
- To run Python programs inside a containerized environment
- To observe isolated execution of multiple containers on a single host

---

3. System and Software Requirements

Hardware Requirements

- Laptop or Desktop Computer
- Minimum 4 GB RAM

Software Requirements

- Operating System: Kali Linux
- Container Platform: Docker Engine
- Internet Connection for downloading Docker images

---

4. Overview of Docker Technology

Docker is a container-based virtualization platform that enables applications to run inside lightweight, portable containers. Each container includes application binaries and libraries while sharing the host kernel.

Key components of Docker include:

Docker Engine

The core runtime that builds and runs containers.

Docker Images

Read-only templates used to create containers.

Docker Containers

Running instances of Docker images.

Docker Hub

A public repository that hosts official and community Docker images.

Major advantages of Docker include:

- Reduced resource consumption
- Fast container startup
- Platform independence
- Easy scalability
- Consistent application behavior

---

5. Docker Architecture

Docker follows a client–server architecture:

- Docker Client: Accepts user commands
- Docker Daemon: Builds, runs, and manages containers
- Docker Registry: Stores Docker images

Workflow:

User Command → Docker Client → Docker Daemon → Image Pull → Container Creation → Execution

---

6. Experimental Procedure and Implementation

6.1 Docker Installation

Docker Engine is installed on Kali Linux using the following commands:

```
sudo apt update
sudo apt install docker.io -y
sudo systemctl start docker
sudo systemctl enable docker
```

These commands update system packages, install Docker, start the Docker service, and enable automatic startup during boot.

Observation:
Docker service started successfully without errors.

6.2 Verification of Docker Installation

The installation is verified using:

```
docker --version
```

This command displays the installed Docker version, confirming successful setup.

Observation:
Docker version information was displayed correctly.

6.3 Execution of Hello-World Container

docker run hello-world

This command pulls the Hello-World image from Docker Hub and runs it inside a container. The container prints a confirmation message and exits.

Observation:
The welcome message confirms Docker is functioning properly.

6.4 Creation of Ubuntu Operating System Container

docker run -it --name ubuntu_container ubuntu bash

An interactive Ubuntu container is launched with a Bash shell.

Commands executed inside the container:

ls
cat /etc/os-release
exit

These commands verify filesystem access and display OS details.

Observation:
Ubuntu shell opened successfully, and operating system information was displayed.

6.5 Deployment of Nginx Web Server Container

docker run -d --name nginx_container -p 8080:80 nginx

This command runs Nginx in detached mode and maps container port 80 to host port 8080.

The web server is accessed using:

http://localhost:8080

Observation:
The default Nginx welcome page appeared in the browser, confirming successful deployment.

6.6 Creation of Python Programming Container

docker run -it --name python_container python:3.10 python

This launches a Python interpreter inside a container.

Sample program executed:

```
print("Hello from Python Container")
exit()
```

Observation:
Python code executed successfully within the containerized environment.

---

7. Experimental Output

- Docker installed successfully
- Hello-World container executed correctly
- Ubuntu container provided Linux shell access
- Nginx container hosted a functioning web server
- Python container executed sample program
- Multiple containers ran simultaneously on the same host

---

8. Advantages of Docker Containers

- Lightweight virtualization
- Faster deployment
- Reduced system overhead
- Platform-independent execution
- Easy scalability
- Improved application isolation

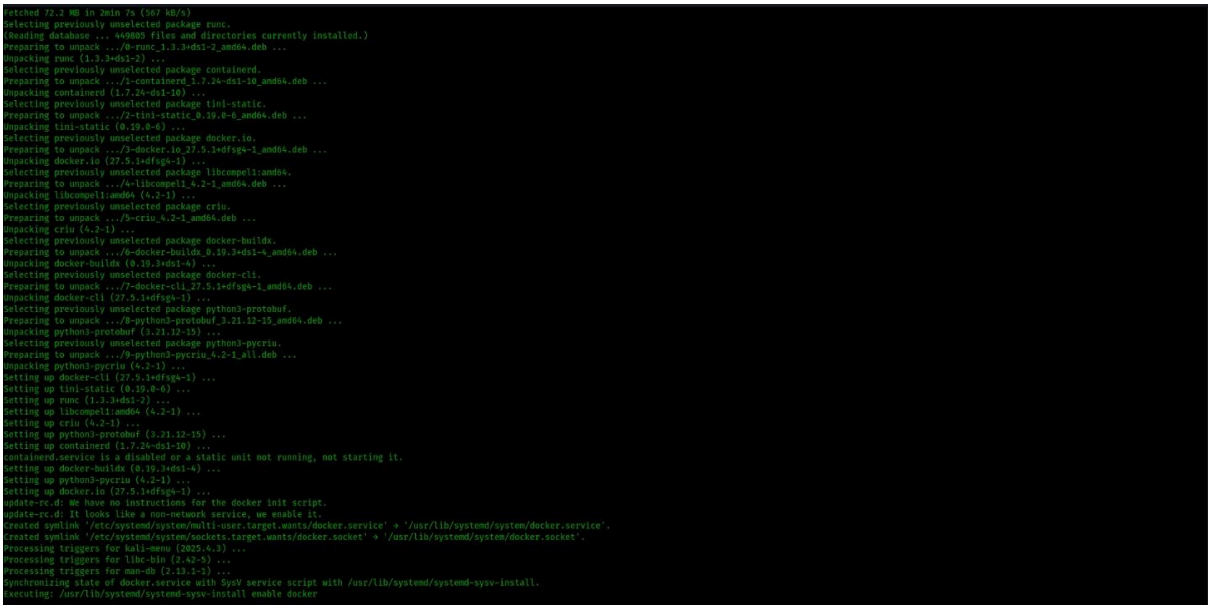
---

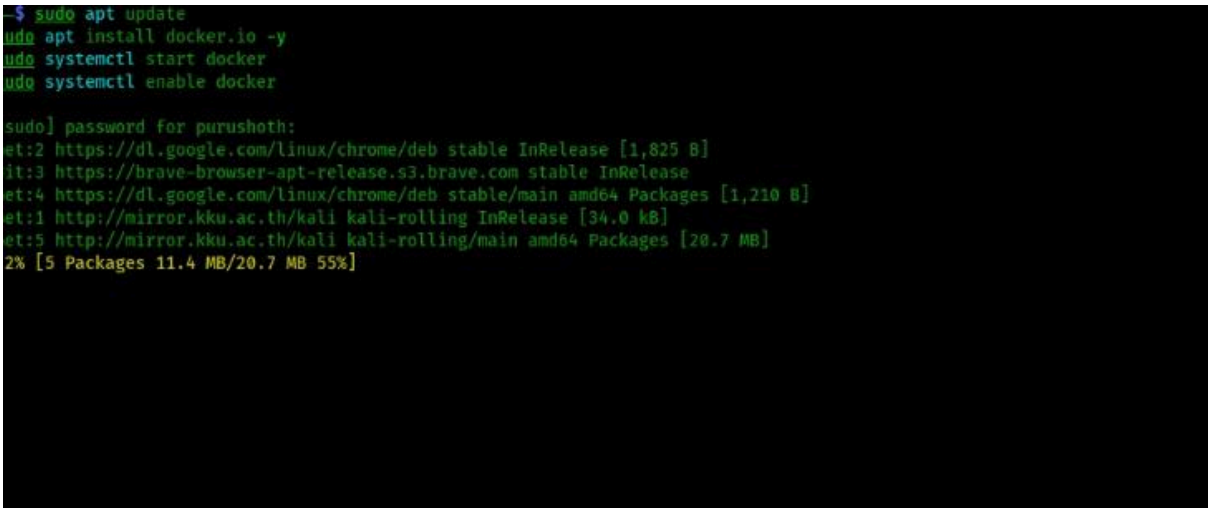
9. Applications of Docker

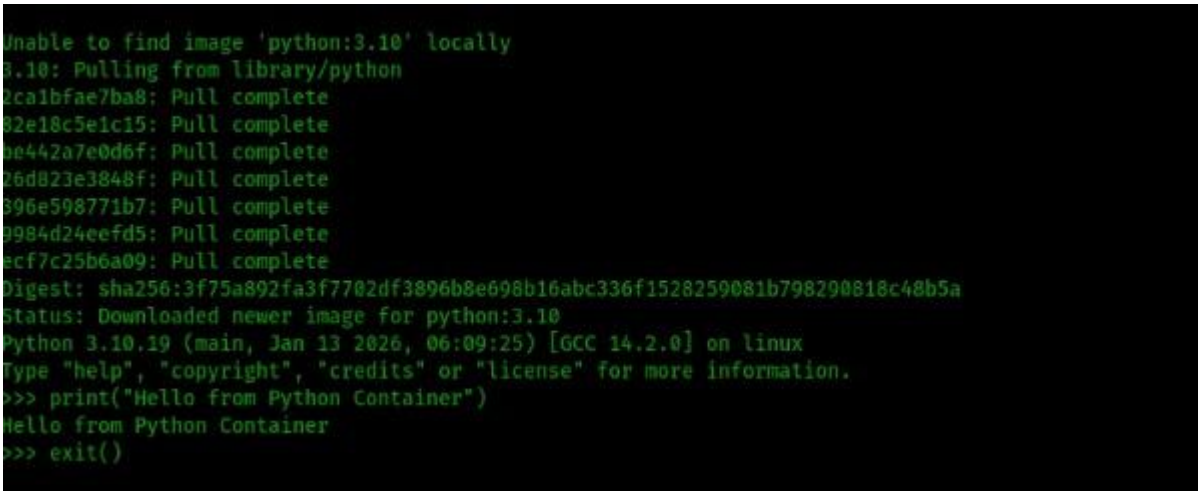
- Web application hosting
- Microservices architecture
- DevOps CI/CD pipelines
- Cloud deployments
- Testing and development environments
- Machine learning workflows

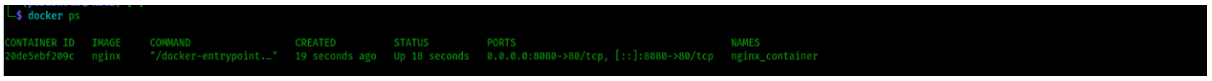
10. Result

The experiment successfully demonstrated Docker installation and execution of multiple containers including Hello-World, Ubuntu OS, Nginx Web Server, and Python Programming Environment. Each container ran independently, confirming Docker’s ability to provide isolated and efficient execution environments.









11. Conclusion

This experiment proves that Docker offers a lightweight and efficient approach to application deployment through containerization. By sharing the host kernel while maintaining isolation, Docker minimizes resource usage and improves performance. The successful execution of multiple containers highlights Docker’s capability to support modern software development, testing, and cloud-based infrastructures.
