Experiment -2.3

Student Name: Chayan Gope
UID: 22BDO10036
Branch: CSE(DevOps)
Section/Group: 22BCD-1/A
Semester: 5th.
Date of Performance: 14/10/24
Subject Name: Docker & Kubernetes
Subject Code: 22CSH-343

1. **Aim/Overview of the practical**: To perform Kubernetes architecture, building blocks and container orchestration.

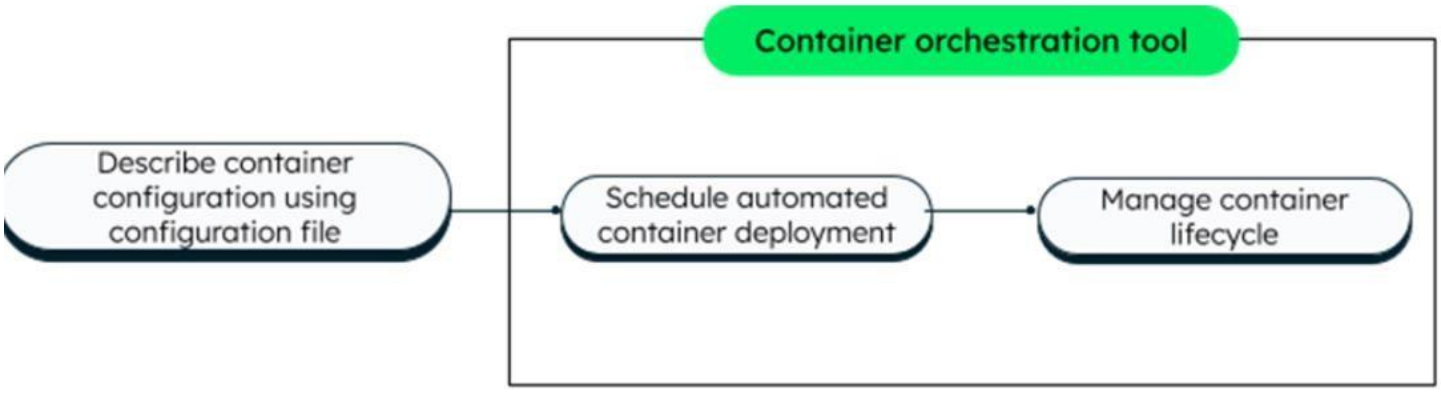
2. **Apparatus:** PC, Web Browser, Docker Engine, DockerHub, Ubuntu Linux

3. **Steps for experiment/practical:**

- **Container Orchestration:**

1. It refers to the automated management of containerized applications across multiple hosts.
2. It includes processes such as deployment, scaling, networking, and management of containers to ensure that applications run efficiently and reliably in a distributed environment.
3. Popular container orchestration tools include:
    - **Kubernetes:** The most widely used orchestration platform. It automated deployment, scaling, and management of containerized applications.
    - **Docker Swarm:** A native clustering and orchestration solution for Docker containers.
    - **Apache Mesos:** A resource management platform that can handle both containers and other workloads.

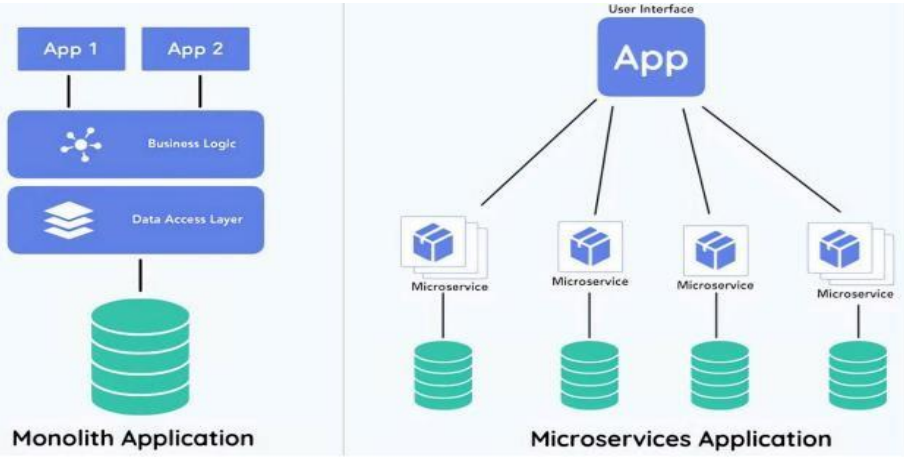
4. Container orchestration benefits are as listed below:
    - DevOps and continuous delivery
    - Scalability
    - Isolation
    - High availability
    - Resilience



- **Monolithic architecture:**
    1. All components of the application are part of a single codebase and deployed together.
    2. Components are tightly integrated and dependent on each other.
    3. Scaling requires scaling the entire application, even if only one part of the application requires more resources.
    4. Difficult to update or change individual components without affecting the entire system.
    5. Failure in one part of the system can affect the entire application.
    6. Deployed as a single unit, which can lead to long build and deployment times.

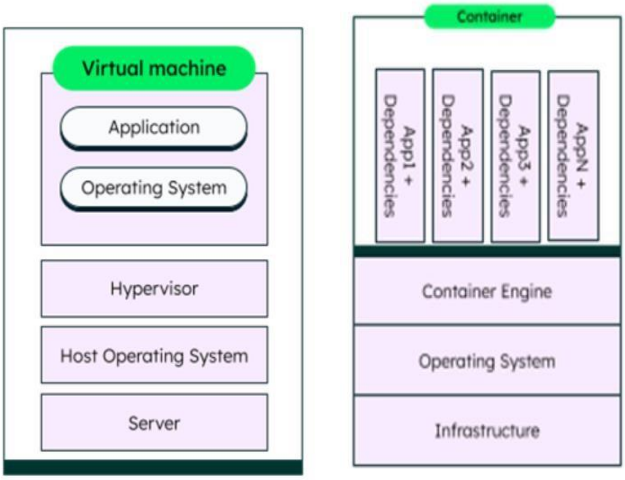
- **Microservices architecture:**
    1. The application is split into independent, loosely coupled services that communicate via APIs.
    2. Each service is independent and can be developed, deployed, and maintained separately.

3. Allows for selective scaling; individual services can be scaled independently based on their specific resource needs.
4. Easier to maintain and update individual services without affecting the entire application.
5. Failure in one microservice does not affect the others, improving system resilience. Each microservice can be deployed independently, enabling more frequent updates and deployments.



Virtualization and Containerization:

1) Virtualization is the process of creating a virtual version of computing resources such as hardware, operating systems, storage devices, or network resources. Each VM operates with its own operating system and functions as a separate entity, independent of other VMs.
2) Containerization is a lightweight form of virtualization that involves encapsulating an application and its dependencies (libraries, binaries, configurations) into a single unit called a container. Containers share the host system's operating system kernel but run isolated processes.

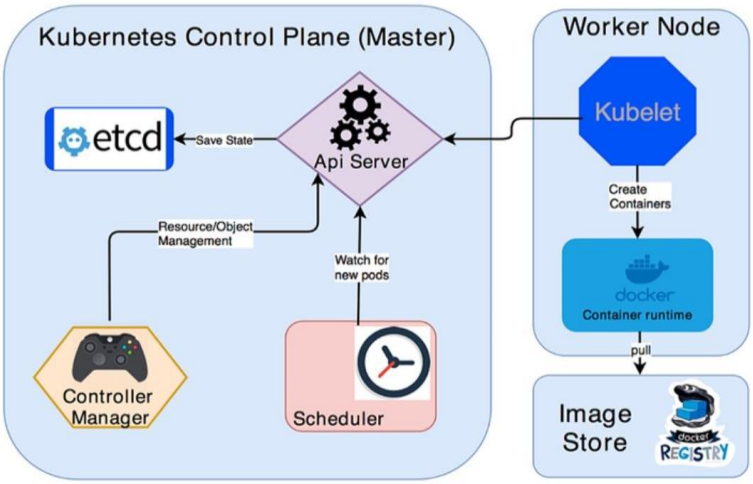


- **Kubernetes**:

  1) Kubernetes (often abbreviated as K8s) is an open-source container orchestration platform that automates the deployment, scaling, management, and networking of containerized applications

  2) Benefits of K8s include:

  - Automated Deployment
  - Scalability
  - Self-Healing
  - Load Balancing
  - Ecosystem Integration (CI/CD pipelines)



3) Kubernetes architecture:

- Kubernetes Control Plane (Master): This is the central management entity in Kubernetes that controls and manages the entire cluster.
- API server: Acts as the front-end for the Kubernetes control plane and Receives requests from administrators (via kubectl) or internal components.
- etcd: Keeps the current state of the cluster (e.g., the configuration, nodes, pods, etc.).
- Controller Manager: Monitors the cluster state to ensure the desired state is maintained.
- Scheduler: Watches for new pods that need to be scheduled (i.e., those that are not yet assigned to any node) and Assigns pods to appropriate worker nodes based on resource availability and other constraints (CPU, memory, etc.).
- Worker Node: Each worker node runs the containerized applications and communicates with the Kubernetes control plane.
- Kubelet: agent running on each worker node, takes instructions from the API server to create, start, and manage containers (pods) on the node & communicate with the container runtime (e.g., Docker) to manage containers.
- Docker (Container Runtime): The container runtime responsible for running the containers.

4) Key components making up Kubernetes include:

- Node: A worker machine in Kubernetes where containers are deployed, managed, and run.
- Cluster: A set of nodes (worker + control plane) managed by Kubernetes to run containerized applications.
- Minikube: A tool that runs a single-node Kubernetes cluster on your local machine for development and testing.
- Pod: The smallest deployable unit in Kubernetes, representing a single instance of a running process in a container.

- ReplicaSet: A Kubernetes resource that ensures a specified number of pod replicas are running at any given time.
- Ingress: A Kubernetes API object that manages external access to services, usually HTTP, within a cluster.
- kubectl: The command-line tool used to interact with and manage Kubernetes clusters and their resources.

Learning outcomes (What I have learnt):

1. I have learnt the concept of containerization.

2. I have learnt to configure Docker to work with different environments.

3. I have learnt how to build docker images using Dockerfile.

4. I have learnt the purpose of Docker volumes and their role in data persistence.

5. I have learnt how to use Docker Hub to pull and push Docker images.

Evaluation Grid (To be created as per the SOP and Assessment guidelines by the faculty):

Sr. No.	Parameters	Marks Obtained	Maximum Marks
1.			
2.			
3.			