Experiment-2.3

Student Name: Aditi Pandey **UID:22BDO10031**

Branch: CSE(DevOps) **Section/Group:22BCD1/A**

Semester:5 Sem **Date of Performance:14/10/2024**

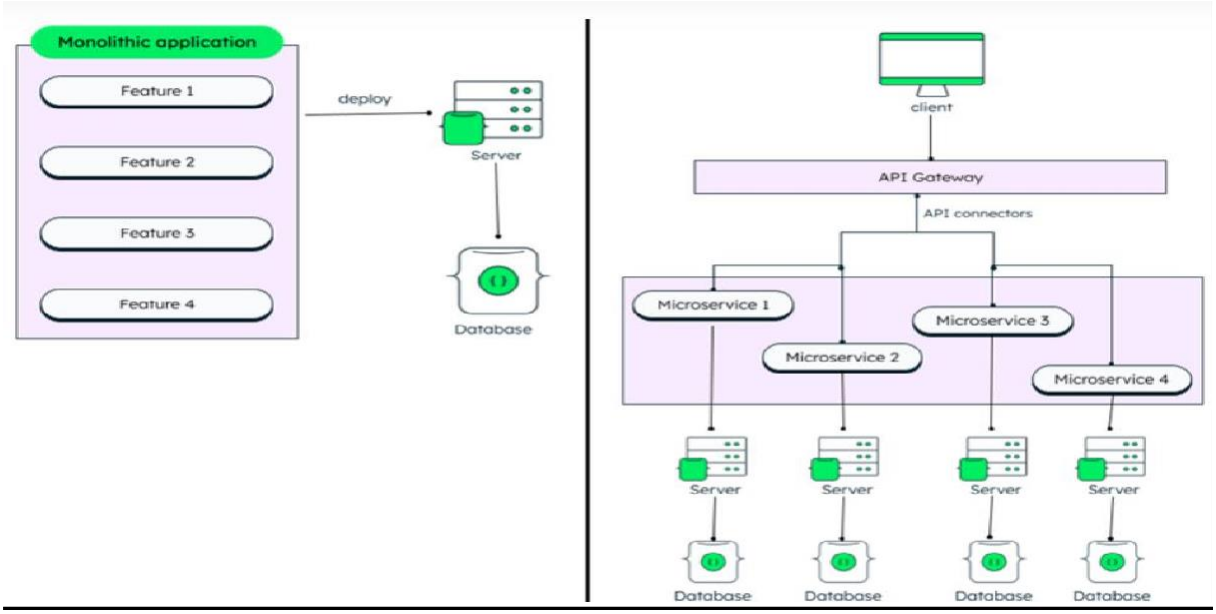
Subject: Docker and Kubernetes **Subject Code: 22CSH-343**

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

Container Orchestration: It automates the deployment, scaling, and management of containers. Crucial for handling the lifecycle of containers at scale.

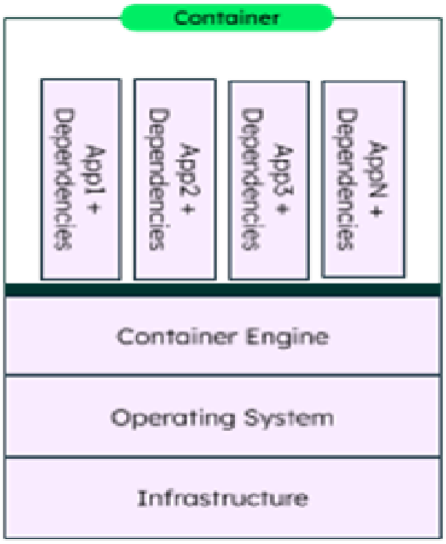
Monolithic Application: All features are bundled together, scaling or updating individual components is difficult.

Microservices Application: Decouples features into independent services, enabling scalability and easier updates.



What are Containers:

- Lightweight packages of applications with all necessary dependencies.
- Tools like Docker convert images into containers.



Container Images:

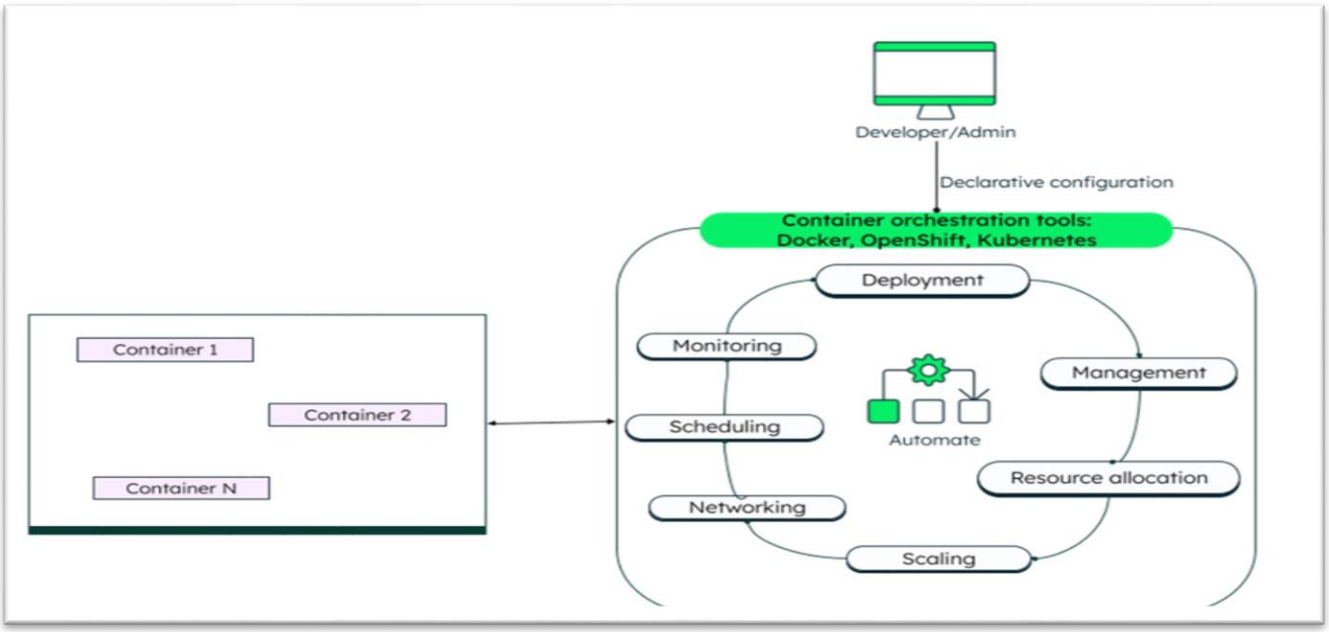
Container images consist of the code, system libraries, tools, runtime, and other settings required to run an application. Container images are light-weight, standalone executables.

Role of Container Orchestration:

- Automates tasks like scaling, load balancing, resource allocation, and container health monitoring.
- Examples: Netflix scales containers dynamically to handle peak hours.

Key Aspects of Container Orchestration:

- **Deployment**: Define how many containers should run.
- **Management**: Simplify configuration for containers.
- **Resource Allocation**: Assign limited resources to balance workloads.
- **Scaling**: Scale up/down based on traffic (vertically or horizontally).
- **Load Balancing**: Distribute traffic across container instances.
- **Networking**: Enable internal and external communication.
- **Scheduling**: Run containers on a defined schedule.
- **Monitoring**: Track container health.
- **Resilience**: Distribute containers across hardware for fault tolerance.



Benefits of Container Orchestration

- **DevOps & CI/CD**: Automates deployment, scaling, and management; integrates with CI tools for seamless code promotion.
- **Scalability**: Easily add or remove containers based on demand.
- **Isolation**: Containers run independently, preventing conflicts across environments.
- **High Availability**: Ensures redundancy and auto-recreates failed containers for uninterrupted service.

Popular Container Orchestration Tools:

- **Kubernetes**: Most popular, support scaling, load balancing, and resilience.
- **Docker Swarm**: Easier setup, integrates with Docker API.
- **Apache Mesos**: Scales to large clusters, used by enterprises like Twitter.

Kubernetes — The Most Popular Container Orchestration Tool

Kubernetes (k8s) is an open-source multi-cloud container orchestration platform developed by Google.

The purpose of Kubernetes is to host your applications in the form of containers in an automated fashion.

- **Scaling & Auto-scaling**: Manages scaling up or out of containers based on demand.
- **Load Balancing**: Distributes traffic across containers efficiently.
- **Clustering**: Supports multi-network or hybrid clusters with virtual and physical machines.
- **Inter-container Communication**: Facilitates seamless interaction between containers.
- **Dynamic Resource Management**: Can spin up new servers and containers as needed.
- **Health Monitoring**: Continuously monitors and maintains container health.
- **Rollback Capability**: Supports reverting to previous application versions.

Kubernetes Components

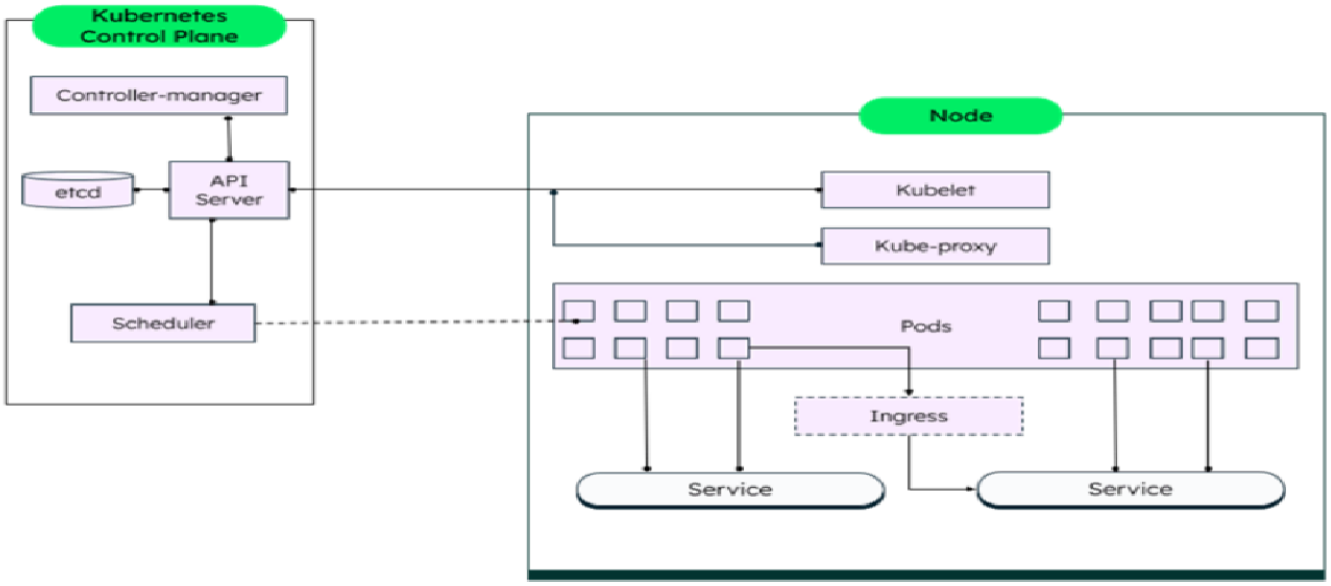
- **Master Node**: Manages the cluster and its nodes.
  - API Server, etc, Controller Manager, Scheduler.
- **Worker Node**: Runs pods, each containing one or more containers.

Key Concepts in Kubernetes

- **Pod**: Smallest deployable unit holding containers.
- **ReplicaSets**: Ensures desired pod count is maintained.
- **Deployment**: Automates lifecycle of pods and ReplicaSets.
- **Ingress**: Manages external traffic to services.

Kubernetes Architecture:

Node: A physical or virtual machine with Kubernetes installed, where containers run.



Includes services like Docker and Kubelet.

- Cluster: A group of nodes working together to ensure availability, with redundancy to handle failures.
- Master: Manages the control plane, orchestrating container deployment across worker nodes.
- Minikube: A tool to run Kubernetes locally using a virtual machine.
- Pod: The smallest deployable unit, containing one or more containers, responsible for container execution.
- ReplicaSets: Ensures the desired number of identical pods are always running by creating replacements when needed.
- Secrets: Securely stores sensitive data (e.g., tokens, passwords) for access within pods.
- Deployment: Automates deployment, scaling, and updates of pods, leveraging ReplicaSets for lifecycle management.
- Ingress: Manages external access to internal services within the cluster via a single entry point.
- Kubectl: A CLI tool to interact with the Kubernetes API for managing and deploying applications.

Learning outcomes (What I have learnt):

1. I have learned the concept of Container Orchestration.
2. I have learned about Orchestration Tools.
3. I have learnt about Kubernetes and its architecture.

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.			