St. Francis Institute of Technology, Mumbai-400 103

**Department Of Information Technology**

A.Y. 2024-2025

Class: TE-ITA/B, Semester: V

Subject: **Advanced DevOps Lab**



**Experiment –4: Case Study : To understand Kubernetes cluster architecture and its applications.**

1. **Aim:** To understand Kubernetes Cluster Architecture and its installation.
2. **Objectives:** Aim of this experiment is that, the students will learn:

* Kubernetes concepts
* Kubernetes architecture on EC2
* Kuberenetes applications, solutions and impact

1. **Lab objective mapped : ITL504.2:** To deploy single and multiple container applications and manage application deployments with rollouts in Kubernetes
2. **Prerequisite:**

* Basic Kubernetes and Container-based concepts

1. **Requirements:** AWS account, browser,Personal Computer, Windows operating system, Internet Connection, AWS CLI, kubectl, Required IAM permissions,
2. **Pre-Experiment Exercise:**

**Brief Theory:**

**Amazon EKS** Amazon Elastic Kubernetes Service (Amazon EKS) is a managed service that you can use to run Kubernetes on AWS without needing to install, operate, and maintain your own Kubernetes control plane or nodes. Kubernetes is an open-source system for automating the deployment, scaling, and management of containerized applications. Amazon EKS:

Runs and scales the Kubernetes control plane across multiple AWS Availability Zones to ensure high availability.

Automatically scales control plane instances based on load, detects and replaces unhealthy control plane instances, and it provides automated version updates and patching for them.

Is integrated with many AWS services to provide scalability and security for your applications, including the following capabilities:

Amazon ECR for container images

Elastic Load Balancing for load distribution

IAM for authentication

Amazon VPC for isolation

**How does Amazon EKS work?**

Create an Amazon EKS cluster in the AWS Management Console or with the AWS CLI or one of the AWS SDKs.

Launch managed or self-managed Amazon EC2 nodes, or deploy your workloads to AWS Fargate.

When your cluster is ready, you can configure your favorite Kubernetes tools, such as kubectl, to communicate with your cluster.

Deploy and manage workloads on your Amazon EKS cluster the same way that you would with any other Kubernetes environment. You can also view information about your workloads using the AWS Management Console.

**Amazon EKS pricing** You pay $0.10 per hour for each Amazon EKS cluster that you create. You can use a single EKS cluster to run multiple applications by taking advantage of Kubernetes namespaces and IAM security policies. You can run EKS on AWS using either Amazon Elastic Compute Cloud (Amazon EC2) or AWS Fargate, and on-premises using AWS Outposts.

**1 Clusters x 0.10 USD per hour x 730 hours per month = 73.00 USD**

**EKS Total Cost (monthly): 73.00 USD**

**Kubectl** is a command line tool that you use to communicate with the Kubernetes API server. The kubectl binary is available in many operating system package managers.

CLI – A command line tool for working with AWS services, including Amazon EKS.

**Features:**

1. Auto-scaling. Automatically scale containerized applications and their resources up or

down based on usage.

2. Lifecycle management. Automate deployments and updates with the ability to Rollback to previous versions and Pause and continue a deployment.

3. Declarative model. Declare the desired state, and K8s work in the background to maintain that state and recover from any failures.

4. Resilience and self-healing. Auto placement, auto restart, auto replication

and auto-scaling provide application self-healing.

5. Persistent storage. Ability to mount and add storage dynamically.

6. Load balancing. Kubernetes supports various internal and external load balancing options to address diverse needs.

7. DevSecOps support. DevSecOps is an advanced approach to security that simplifies and automates container operations across clouds, integrates security throughout the container lifecycle, and enables teams to deliver secure, high-quality software more quickly. Combining DevSecOps practices and Kubernetes improves developer productivity.

Kubernetes works with Amazon EC2, Azure Container Service, Rack space, GCE,

IBM Software, and other clouds. And it works with bare-metal (using CoreOS),

Docker, and vSphere.

Kubernetes is used by Google, Spotify, The New York Times, Pinterest, Adidas,

Tinder, Capital One, etc.

**7. Laboratory Exercise :**

Find answers to the following questions and make a Google document for the same. Upload this doc here on GC and click turn in.

Name of the document :

EXP 4\_Kubernates\_Studentname

**1. What is Kubernetes?**

Kubernetes is a portable, extensible, open source platform for managing containerized workloads and services, that facilitates both declarative configuration and automation. It has a large, rapidly growing ecosystem. Kubernetes services, support, and tools are widely available.

Developed by Google, it provides a robust framework for managing containerized workloads and services across clusters of machines, ensuring efficient resource utilization and high availability. Kubernetes abstracts away the underlying infrastructure and allows developers to deploy applications in a consistent manner regardless of the environment.

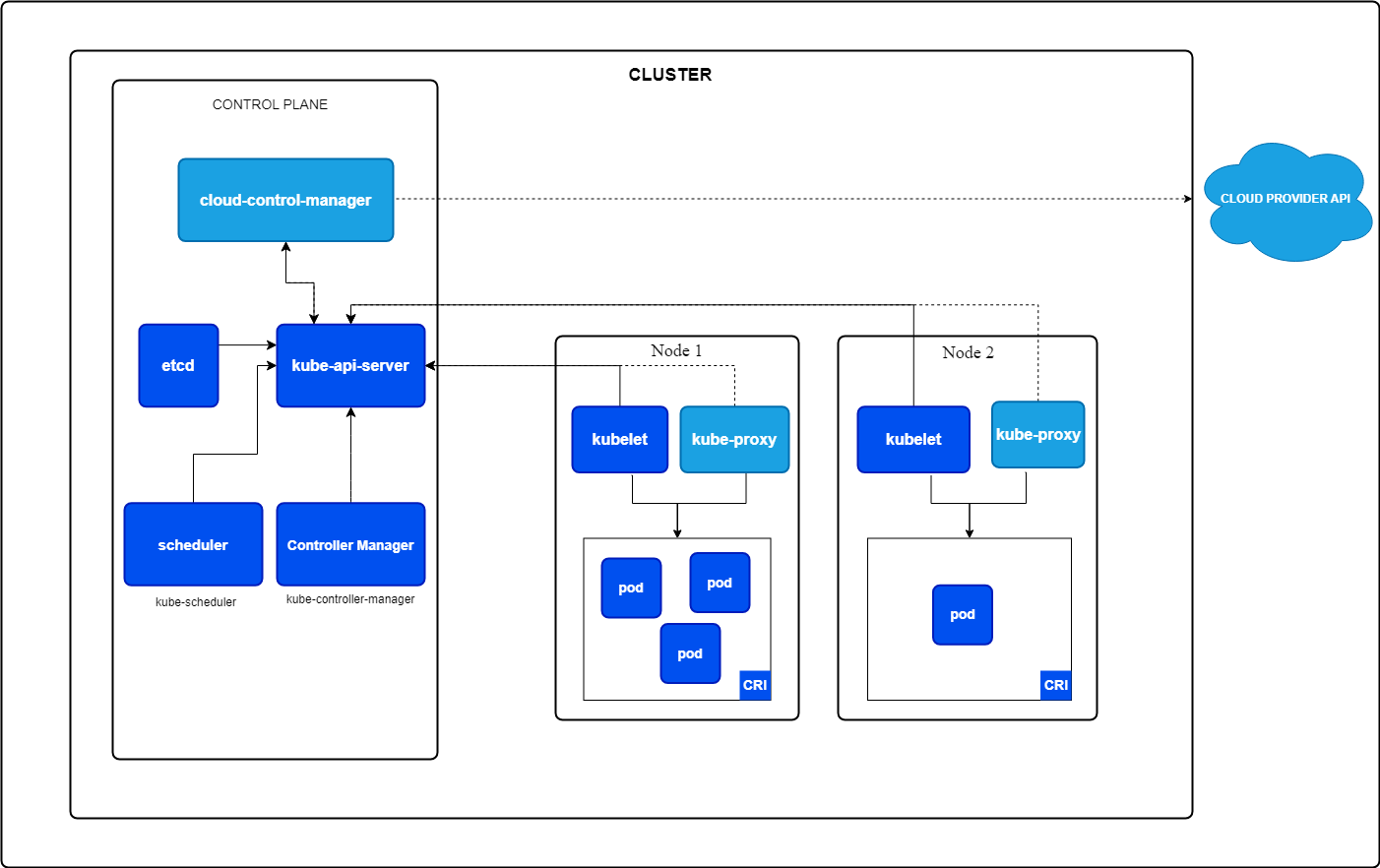
**2. Architecture of Kubernetes**

Kubernetes architecture components or K8s components include the Kubernetes control plane and the nodes in the cluster. The control plane machine components include the Kubernetes API server, Kubernetes scheduler, Kubernetes controller manager, and etcd. Kubernetes node components include a container runtime engine or docker, a Kubelet service, and a Kubernetes proxy service.

#### Key Components of Kubernetes Architecture:

1. **Master Node (Control Plane):**
   1. **API Server:** Acts as the front-end for the Kubernetes control plane. It exposes the Kubernetes API and is the primary entry point for all administrative tasks.
   2. **etcd:** A distributed key-value store that Kubernetes uses to store all cluster data. It is the central source of truth for the cluster state.
   3. **Scheduler:** Watches for newly created Pods that have no assigned node, and selects a node for them to run on.
   4. **Controller Manager:** Runs controller processes in the background. Each controller watches the state of the cluster via the API server and makes changes to move the current state towards the desired state.
2. **Worker Nodes:**
   1. **Kubelet:** An agent that runs on each node in the cluster. It ensures that containers are running in a Pod and communicates with the Master Node.
   2. **Kube-Proxy:** Maintains network rules on nodes, enabling network communication to your Pods from network sessions inside or outside of your cluster.
   3. **Container Runtime:** The software responsible for running containers. Kubernetes supports multiple container runtimes, including Docker..
3. **Pods:** A Pod is the smallest and simplest Kubernetes object. A Pod represents a single instance of a running process in your cluster and can contain one or more containers.
4. **Namespaces:** Namespaces provide a mechanism to scope resources in a cluster. Namespaces are intended for use in environments with many users spread across multiple teams or projects.
5. **Volumes:** Kubernetes Volumes provide a way to share data between containers in a Pod and persist data across container restarts.
6. **Controllers** Controllers are responsible for ensuring that the cluster’s desired state matches the actual state. Examples include the ReplicaSet, Deployment, and StatefulSet controllers.

The architecture of Kubernetes is modular and designed for flexibility, allowing it to integrate with various cloud providers, networking solutions, and storage systems.



This diagram illustrates the architecture of Kubernetes.

**3. Features of Kubernetes**

1. **Automated Rollouts and Rollbacks:** Deploy updates to applications with zero downtime, and rollback to previous versions automatically if issues are detected.
2. **Service Discovery and Load Balancing:** Kubernetes automatically assigns IPs to containers and balances network traffic to ensure consistent performance.
3. **Storage Orchestration:** Easily mount storage systems, whether from local disks, public clouds, or network storage like NFS, ensuring persistent data across container restarts.
4. **Self-Healing:** Kubernetes monitors the health of containers, automatically restarting, replacing, or rescheduling them when failures occur to keep the application running smoothly.
5. **Secret and Configuration Management:** Securely store and manage sensitive information like passwords and API keys, and update configurations without needing to rebuild container images.
6. **Horizontal Scaling:** Automatically scales applications up or down based on real-time resource usage, ensuring the application can handle varying loads efficiently.

Refer to the given link to understand and map case studies to Kubernetes solutions.

Case Studies examples link: https://kubernetes.io/case-studies/

Choose any one case study example and answer the following questions

(All 4 question answers are to be written in hand on SFIT papers. )

1. History of the company in 15-20 lines.

2. Problems/Challenges faced by the company.

3. Solution given by Kubernetes.

4. Impact on company.

1. **Post-Experiments Exercise**
2. **Extended Theory:**

Nil

1. **Questions:**

Nil

1. **Conclusion:**(write in hand)

* 1. Write what was performed in the experiment
* 2. Mention few applications of what was studied.
* 3. Write the significance of the studied topic

1. **References:**
2. https://kubernetes.io/case-studies/
3. https://cloudacademy.com/lab/eks-voteapp/

**—--------------------—---------------**