AIM: To study container orchestration using Kubernetes.

CASE STUDY:

Introduction:

➢ **What is Kubernetes?**

Kubernetes is also known as 'k8s'. This word comes from the Greek language, which means a pilot or helmsman.

Kubernetes is an extensible, portable, and open-source platform designed by Google in 2014. It is mainly used to automate the deployment, scaling, and operations of the container-based applications across the cluster of nodes. It is also designed for managing the services of containerized apps using different methods which provide the scalability, predictability, and high availability.

➢ **Key Objects:**

- **Pod:** It is the smallest and simplest basic unit of the Kubernetes application. This object indicates the processes which are running in the cluster.
- **Node:** A node is nothing but a single host, which is used to run the virtual or physical machines. A node in the Kubernetes cluster is also known as a minion.
- **Service:** A service in a Kubernetes is a logical set of pods, which works together. With the help of services, users can easily manage load balancing configurations.
- **ReplicaSet:** A ReplicaSet in the Kubernetes is used to identify the particular number of pod replicas are running at a given time. It replaces the replication controller because it is more powerful and allows a user to use the "set-based" label selector.
- **Namespace:** Kubernetes supports various virtual clusters, which are known as namespaces. It is a way of dividing the cluster resources between two or more users.

➢ **Features:**

1. **Horizontal Scaling:** It is an important feature in the Kubernetes. This feature uses a HorizontalPodAutoscalar to automatically increase or decrease the number of pods in a deployment, replication controller, replica set, or stateful set on the basis of observed CPU utilization.

2. **Automatic Bin Packing:** Kubernetes helps the user to declare the maximum and minimum resources of computers for their containers.
3. **Service Discovery and load balancing:** Kubernetes assigns the IP addresses and a Name of DNS for a set of containers, and also balances the load across them.
4. **Automated rollouts and rollbacks:** Using the rollouts, Kubernetes distributes the changes and updates to an application or its configuration. If any problem occurs in the system, then this technique rollbacks those changes for you immediately.
5. **Persistent Storage:** Kubernetes provides an essential feature called 'persistent storage' for storing the data, which cannot be lost after the pod is killed or rescheduled. Kubernetes supports various storage systems for storing the data, such as Google Compute Engine's Persistent Disks (GCE PD) or Amazon Elastic Block Storage (EBS). It also provides the distributed file systems: NFS or GFS.
6. **Self-Healing:** This feature plays an important role in the concept of Kubernetes. Those containers which are failed during the execution process, Kubernetes restarts them automatically. And, those containers which do not reply to the user-defined health check, it stops them from working automatically.

➢ **Kubernetes Clusters:**

You can cluster together groups of hosts running Linux containers, and Kubernetes helps you easily and efficiently manage those clusters.

Kubernetes clusters can span hosts across on-premise, public, private, or hybrid clouds. For this reason, Kubernetes is an ideal platform for hosting cloud-native applications that require rapid scaling, like real-time data streaming through Apache Kafka.

➢ **Container Orchestration:**

- Container orchestration automates the deployment, management, scaling, and networking of containers. Enterprises that need to deploy and manage hundreds or thousands of Linux containers and hosts can benefit from container orchestration.
- Container orchestration can be used in any environment where you use containers. It can help you to deploy the same application across different environments without needing to redesign it. And microservices in containers make it easier to orchestrate services, including storage, networking, and security.
- Containers give your microservice-based apps an ideal application deployment unit and self-contained execution environment. They make it possible to run multiple parts of an app independently in microservices, on the same hardware, with much greater control over individual pieces and life cycles.
- Managing the lifecycle of containers with orchestration also supports DevOps teams who integrate it into CI/CD workflows. Along with application programming interfaces (APIs) and DevOps teams, containerized microservices are the foundation for cloud-native applications.

➢ **What is container orchestration used for?**

- Provisioning and deployment
- Configuration and scheduling
- Resource allocation, Container availability
- Scaling or removing containers based on balancing workloads across your infrastructure
- Load balancing and traffic routing
- Monitoring container health
- Configuring applications based on the container in which they will run
- Keeping interactions between containers secure

➢ **Benefits:**

Container orchestration is key to working with containers, and it allows organizations to unlock their full benefits. It also offers its own benefits for a containerized environment, including:

- **Simplified operations:** This is the most important benefit of container orchestration and the main reason for its adoption. Containers introduce a large amount of complexity that can quickly get out of control without container orchestration to manage it.
- **Resilience:** Container orchestration tools can automatically restart or scale a container or cluster, boosting resilience.
- **Added security:** Container orchestration's automated approach helps keep containerized applications secure by reducing or eliminating the chance of human error.

➢ **How does container orchestration work?**

- When we use a container orchestration tool, such as Kubernetes, we describe the configuration of an application using either a YAML or JSON file. The configuration file tells the configuration management tool where to find the container images, how to establish a network, and where to store logs.
- When deploying a new container, the container management tool automatically schedules the deployment to a cluster and finds the right host, taking into account any defined requirements or restrictions. The orchestration tool then manages the container's lifecycle based on the specifications that were determined in the compose file.

CONCLUSION: We haved studied about Kubernetes. We also understood its features and Key objects. We thereafter, studied about Containers and Container Orchestration, their benefits and working.