Experiment: Implement Container Management with Kubernetes

AIM:

To implement container management using Kubernetes (K8s) for deployment, scaling, and management of containerized applications.

Prerequisites:

• Basic knowledge of containers and Docker.
• Kubernetes installed via Minikube, Docker Desktop, or cloud providers (GKE, EKS, AKS).
• kubectl command-line tool installed and configured.
• A working container image (e.g., Nginx or custom Docker image).
• Internet connectivity.

Theory:

Kubernetes is an open-source container orchestration platform that automates deployment, scaling, and management of containerized applications.

Key Concepts:
• Pod: The smallest deployable unit in Kubernetes, typically encapsulating one or more containers.
• Node: A physical or virtual machine that runs container workloads.
• Cluster: A set of nodes managed by the Kubernetes control plane.
• Deployment: A controller managing replicas of pods to ensure high availability.
• Service: An abstraction exposing a set of pods and a policy to access them (e.g., LoadBalancer, ClusterIP).

Kubernetes simplifies container management by handling scheduling, updates, networking, and scaling automatically.

Steps to Deploy and Manage a Container Using Kubernetes:

Step 1: Start Minikube or K8s Cluster
```
minikube start
```

Step 2: Create a Deployment
```
kubectl create deployment my-nginx --image=nginx
```

Step 3: Verify Deployment
```
kubectl get pods
kubectl get deployments
```

Step 4: Expose the Deployment as a Service
```
kubectl expose deployment my-nginx --type=NodePort --port=80
```

Step 5: Get the Access URL
```
minikube service my-nginx --url
```

Step 6: Scale the Deployment
```
kubectl scale deployment my-nginx --replicas=3
kubectl get pods
```

Step 7: Update the Deployment (Optional)

```
kubectl set image deployment/my-nginx nginx=nginx:1.19
kubectl rollout status deployment/my-nginx
```

Step 8: Clean Up

```
kubectl delete service my-nginx
kubectl delete deployment my-nginx
minikube stop
```

Conclusion:

In this experiment, we successfully implemented container management using Kubernetes.
- Created a deployment and exposed it via a service.
- Scaled the deployment for high availability.
- Updated the container image and monitored the rollout.

Kubernetes provides robust orchestration for containers, offering high availability, scalability, and maintainability for cloud-native applications.