Project Title: Cloud-Native Application Development with IBM Cloud and Kubernetes

PHASE 4 - FINAL DOCUMENT

College Name:Rajeev Institute of Technology

Group Members:

- **Name:** Vaibhavi M S

  CAN ID Number: CAN_33428859

- **Name:** Suhana

  CAN ID Number: CAN_33282007

- **Name:** Sindhu H P

  CAN ID Number: CAN_33316766

- **Name:** Vidyashri Basavaraj Angadi

  CAN ID Number: CAN_33382863

- **Overview of Cloud-Native Application Development**

This project focuses on leveraging IBM Cloud Kubernetes Service (IKS) and IBM Cloud Container Registry (ICR) to streamline the development and deployment of cloud-native applications. The goal is to build a scalable, efficient, and automated pipeline for deploying containerized applications on Kubernetes clusters. The project will cover containerization, image management, Kubernetes orchestration, and CI/CD automation.

Key Components:

Containerization: Package applications into containers for consistent and portable deployments.

IBM Cloud Container Registry: Store and manage container images securely.

IBM Kubernetes Service (IKS): Orchestrate container deployment, scaling, and management.

Automation & CI/CD: Automate the build, push, and deployment pipeline for rapid and consistent application updates.

- **Configuring IBM Cloud Kubernetes and Container Registry**
- Steps to Set Up IBM Cloud Kubernetes Service (IKS)

Create an IBM Cloud Account & Log In:

Provision the Kubernetes Cluster:

Step 1: From the IBM Cloud Dashboard, navigate to Kubernetes.

Step 2: Click Create Cluster and select your desired Region and Cluster Plan (Standard or Free).

Step 3: Choose the worker node type (e.g., Standard, Compute).

Step 4: Once the cluster is created, configure the kubectl CLI tool to manage the cluster from your local machine:

ibmcloud ks cluster config --cluster <cluster-name>
Integrate IBM Cloud Container Registry (ICR):

Navigate to Container Registry from the IBM Cloud Dashboard.

Create a private registry for storing your container images.

Note your Registry URL and Credentials to authenticate the Docker CLI for pushing images.

- Containerizing Applications with Docker

Create a Dockerfile:
Example for a Python Flask application:

FROM python:3.9-slim

WORKDIR /app

```
COPY requirements.txt /app/requirements.txt
RUN pip install -r requirements.txt

COPY . /app
CMD ["python", "app.py"]
```

Build the Docker Image:

```
docker build -t myapp:latest .
```

Tag the Image for IBM Cloud Container Registry:

```
docker tag myapp:latest <REGISTRY_URL>/<namespace>/myapp:latest
```

Push the Docker Image to IBM Cloud Container Registry:

Authenticate Docker to IBM Cloud:

```
ibmcloud cr login
```

Push the image:

```
docker push <REGISTRY_URL>/<namespace>/myapp:latest
```

- **Deploying Containers on IBM Kubernetes**

- Create Kubernetes Deployment and Service

Deployment YAML:

Example deployment.yaml:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: myapp-deployment
spec:
  replicas: 3
  selector:
  matchLabels:
```

```
      app: myapp
  template:
  metadata:
      labels:

        app: myapp
    spec:
      containers:
      - name: myapp
        image: <REGISTRY_URL>/<namespace>/myapp:latest
        ports:
        - containerPort: 5000
```

Service YAML:

```
apiVersion: v1
kind: Service
metadata:
  name: myapp-service
spec:
  selector:
    app: myapp
  ports:
  - protocol: TCP
    port: 80
    targetPort: 5000
type: LoadBalancer
```

Deploy to Kubernetes:

```
kubectl apply -f deployment.yaml
kubectl apply -f service.yaml
```

Verify Deployment:

```
kubectl get pods
kubectl get svc
```

- **Automating the Deployment with CI/CD Pipeline**

- Integrating with IBM Cloud Continuous Delivery

Create a Delivery Pipeline:

From the IBM Cloud Dashboard, navigate to Continuous Delivery.

Create a new pipeline to automatically build and deploy the containerized application to your Kubernetes cluster.

Pipeline Configuration:

Step 1: Add a Build stage to build the Docker image using the Dockerfile in your repository.

Step 2: Add a Deploy stage to deploy the image to your IBM Kubernetes cluster using kubectl.

Trigger the Pipeline:

Set up the pipeline to trigger on code commits to the repository (e.g., via GitHub webhook).

- **User Interface Development for Monitoring and Management**

To monitor and manage Kubernetes deployments, integrate tools like IBM Cloud Monitoring or Prometheus with Grafana.

Set Up IBM Cloud Monitoring:

Use IBM Cloud Monitoring to track the health of your Kubernetes clusters and applications.

Set up custom alerts for failures, resource exhaustion, or scaling issues.

Prometheus and Grafana:

Install Prometheus and Grafana on your Kubernetes cluster to monitor performance and create interactive dashboards.

- **IBM Cloud Platform Features and Considerations**

Scalability

Benefits: Handles fluctuating workloads and large datasets.
Best Practices: Enable auto-scaling and monitor cluster usage.

Security
Benefits: Protects sensitive data with IAM roles and encryption.
Best Practices: Use least privilege policies, enable encryption, and enforce MFA for admin roles.

Monitoring
Benefits: Tracks containerized application performance and health.
Best Practices: Set alerts for critical metrics and use dashboards for proactive issue resolution.

Cost Efficiency
Benefits: Minimize costs by optimizing resource allocation and storage classes.
Best Practices: Analyze usage patterns and adjust scaling and storage policies accordingly.

- **Conclusion**

This project integrates IBM Cloud Kubernetes Service and IBM Cloud Container Registry to streamline the development and deployment of cloud-native applications. The system ensures scalable, efficient, and automated deployment pipelines, enabling quick updates and reliable application performance.

- **Further Enhancements**

Automated Rollbacks: Implement automatic rollback mechanisms in the deployment pipeline to revert failed deployments to stable versions.

Multi-cluster Management: Extend the solution to manage multiple Kubernetes clusters across regions or cloud environments.

Cost Optimization: Integrate cost management tools to analyze usage and optimize resource allocation for better cost-efficiency.

GitHub Repository:

https://github.com/vaibhavimss/Cloud-Native-Application-Development-with-kubernetes.