Date of Submission: 7days after reception.

Lab Task: Building and Deploying a Microservices Architecture

Objective

This lab guides you through creating and deploying a basic microservices architecture. The goal is to understand microservices principles, containerization, and orchestration using Docker and Kubernetes.

Prerequisites

1. Basic knowledge of:
    - REST APIs
    - Docker
    - Kubernetes
2. Installed tools:
    - Docker
    - Kubernetes (Minikube, K3s, or a cloud provider like AWS EKS)
    - Postman or curl for testing

Task Overview

You will design a simple e-commerce application with the following services:

1. **Product Service**: Manages product catalog.
2. **Order Service**: Handles order creation.
3. **User Service**: Manages user information.

Each service will be containerized and deployed to Kubernetes.

Steps

1. Set Up Microservices

1. Create a directory for each service: product-service, order-service, and user-service.
2. Implement REST APIs for each service using a language of your choice (e.g., Python, Node.js, Java).
    - **Product Service**: Should have an endpoint to list products.
    - **Order Service**: Should have an endpoint to create orders.

    - **User Service**: Should have an endpoint to list users.
3. Ensure each service listens on a different port.

2. Dockerize the Microservices

1. Write a Dockerfile for each service.
    - The Dockerfile should specify the base image, dependencies, and the command to run the service.
2. Build Docker images for each service.
    - Use meaningful tags for the images (e.g., product-service:1.0).
3. Test the images locally by running the containers and accessing the services using localhost.

3. Push Images to a Registry

1. Log in to a Docker registry (e.g., Docker Hub, AWS ECR).
2. Push the Docker images to the registry to make them accessible for deployment.
3. Verify the images are available in the registry.

4. Prepare Kubernetes Deployment Files

1. Write a Deployment manifest for each service. Include:
    - Replica count.
    - Image details (registry and tag).
    - Ports exposed by the container.
2. Write a Service manifest for each service to expose it within the Kubernetes cluster.
3. Use ClusterIP type for internal communication between services.

5. Deploy Services to Kubernetes

1. Deploy the manifests using kubectl apply.
2. Verify that the pods are running using kubectl get pods.
3. Check the services using kubectl get services.

6. Test the Services

1. Use kubectl port-forward or a LoadBalancer service to expose the microservices externally.
2. Test the APIs using Postman or curl to confirm they are working as expected.

7. Add Inter-Service Communication

1. Modify the microservices to communicate with each other using internal Kubernetes DNS.
    - Example: http://<service-name>.<namespace>.svc.cluster.local
2. Update the code to make HTTP requests to the appropriate services.

8. Enhance Deployment

1. Implement an API Gateway for routing requests to the services.
2. Configure autoscaling for the deployments.
3. Set up monitoring using tools like Prometheus and Grafana.

Submission Requirements

- A GitHub repository containing:
    - Source code for all services.
    - Dockerfiles.
    - Kubernetes manifests.
- Screenshots of:
    - Running containers.
    - Kubernetes pods and services.
    - API responses from testing.

Outcome

By completing this lab, you will understand:

- Designing a microservices architecture.
- Containerizing applications with Docker.
- Deploying and managing services in Kubernetes.

NB: You have an Opportuinity to learn this from me. Make use of it.