Iran University of Science and Technology
School of Computer Engineering

Project #1

Clustering Using Kubernetes

Deadline: November 15th at 23:59

October 31,2024

Project Description

In this Distributed Systems course project focusing on Clustering, students will engage in a practical Kubernetes exercise. The task involves setting up a local Kubernetes cluster, preferably using virtual machines like VirtualBox. Students will develop and deploy a web service, which includes a 'GET /instance-id' API, managing at least two service instances/pods. The service should implement round-robin load balancing, with Nginx as the recommended choice. A key challenge is ensuring each service instance uniquely identifies itself upon each API call. This project aims to deepen practical skills in Kubernetes, containerization, and cluster management, bridging theoretical concepts with hands-on experience. Figure 1 illustrates a high level architecture of the system.

Requirements

1. **Kubernetes Cluster Creation:** Establish a local Kubernetes cluster. Utilize virtual machines (like VirtualBox) to add nodes. Cloud managed services are not to be used. Use Google/Youtube to find out how to create a cluster.

2. **Web Service Development and Deployment:** Create, containerize, and deploy a web service with a 'GET /instance-id' API.

3. **Service Instances:** Ensure a minimum of two web service instances/pods are operational at all times.

4. **Load Balancing:** Implement round-robin load balancing, ideally with Nginx. However, any other load balancer is acceptable.

5. **Unique Instance Identification:** Each API call to '/instance-id' endpoint must return a unique identifier (e.g., unique Pod ID) from the handling instance/pod. The ID should not be hard coded.

6. **Programming Language:** Any programming language is acceptable.

7. **Documentation and Code:** Provide comprehensive documentation along with source code.

8. **Screen Record:** A screen recording demonstrating the project's functionality is mandatory for submission.

Additional Notes

- **Screen Recording Details:** Showcase service scaling from 2 to at least 3 instances in your screen recording. Call the API at least 4 times to ensure the fourth call returns the first ID again, demonstrating the round-robin functionality.

- **Resource Limitations:** If resource limitations hinder virtual machine installation, you can ues Minikube, Kind , etc . Deploying on a virtual machine-based Kubernetes cluster is considered a bonus.

- **Submission Format:** Submit a single zip file named <name>-<student_id>.zip, containing source code, configuration files, project documentation (PDF), and the execution screen recording.

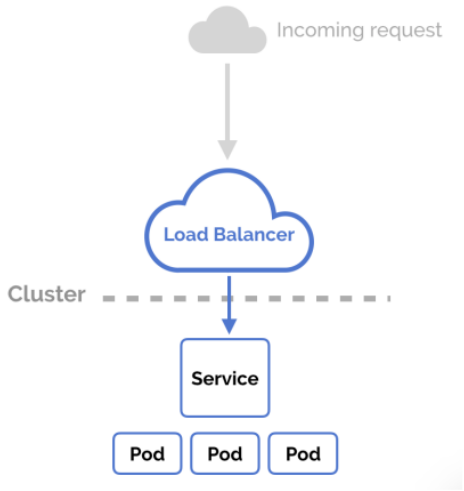

Figure 1: Project Architecture Overview. Each API call is served by a unique Pod, which then returns its own ID as response.