Sarthak Kaushik | 101471600

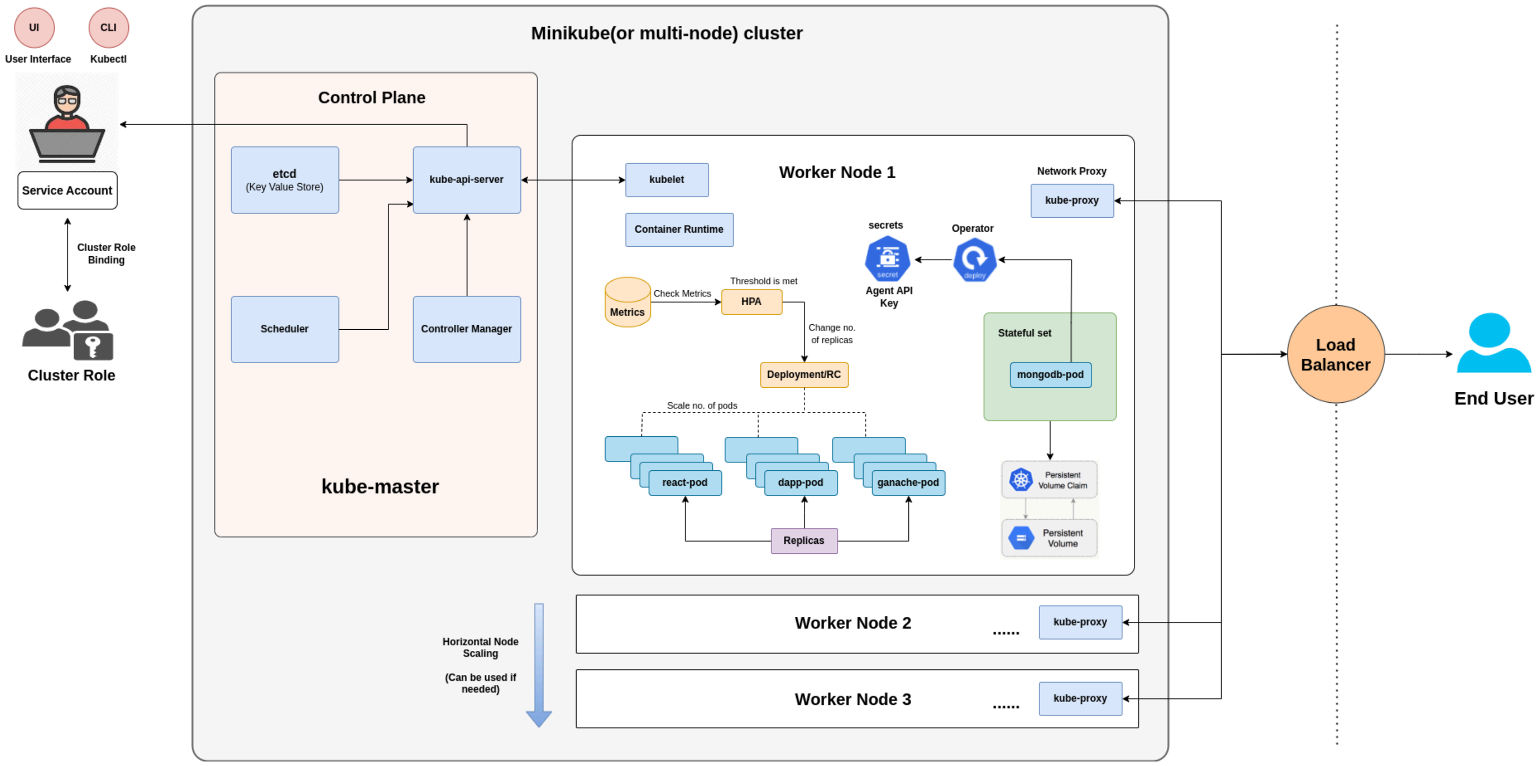
Kubernetes Design Document for Minikube Cluster

TABLE OF CONTENT

1. Introduction

This document outlines the design for a Kubernetes cluster using Minikube to deploy a system consisting of a React frontend, Ganache blockchain, and an Express backend. The deployment is designed to handle replicas of these components, ensuring scalability, reliability, and efficient resource utilisation.

2. Architecture Diagram



3. Application Components

3.1 React Frontend

- Docker Image: Used the Docker image `sudojarvis/eth-react` for the React frontend.
- Replicas: Adjusting the number of replicas based on the expected load.
- Environment Variables: Defined environment variables for dynamic configurations.

3.2 Ganache Blockchain

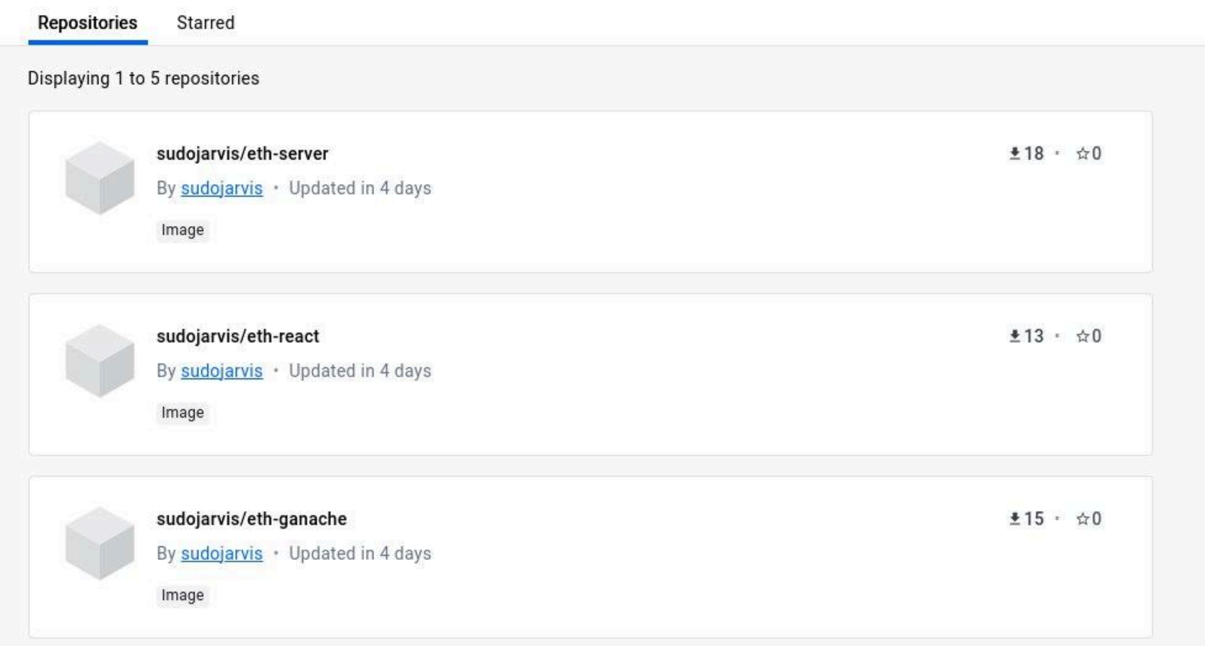
- Docker Image: Utilised the Docker image `sudojarvis/eth-ganache` for Ganache.
- Replicas: Adjusting replicas based on the requirements for parallel blockchain instances.
- Persistent Storage: Used persistent storage for blockchain data.

3.3 Express Backend

- Docker Image: Used the Docker image `sudojarvis/eth-server` for Backend
- Replicas: Scaling replicas based on the incoming traffic.
- Communication with Ganache: Configured backend to connect to the Ganache blockchain.

3.4 MongoDB Database

- Used MongoDB as a document-oriented database for the application. It is used with stateful sets and secrets in Kubernetes. StatefulSets make sure that the deployed replicas maintain a sticky, stable identity and the secrets help in managing sensitive information like database credentials.



Displaying 1 to 5 repositories

sudojarvis/eth-server
By sudojarvis · Updated in 4 days
Image
⤓18 · ☆0

sudojarvis/eth-react
By sudojarvis · Updated in 4 days
Image
⤓13 · ☆0

sudojarvis/eth-ganache
By sudojarvis · Updated in 4 days
Image
⤓15 · ☆0

4. Deployment Strategy

4.1 Replica Sets

- Implemented ReplicaSets for each deployment to ensure high availability for all the application components.
- It automatically **replaces failed pods**, ensuring all components are running
- ReplicaSets can **scale up** when the running instances are not up to the specified number, and **scale down** or delete Pods if another instance with the same label is created

5. Storage

- Used **Persistent Volumes (PV) and Persistent Volume Claims (PVC)** for **MongoDB** component requiring data persistence.

- The MongoDB pod in a **Stateful Set** provides a unique, persistent identity and its own volume(its own persistent disk), ensuring stable pod identification for MongoDB.

6. Scaling

- Utilised **Horizontal Pod Autoscaling (HPA)** to automatically adjust the number of pods based on CPU or memory usage.
- Adapts to changes in request loads.
- Minimises resource wastage by optimising resource utilisation
- Enhances performance (depends on scaling capacity given/available)

7. Load Balancing

- Load Balancing in Kubernetes is used to distribute network traffic across multiple pods, ensuring **no single pod is overworked**

- Load balancing and service discovery enhances application responsiveness and **availability**, providing a seamless user experience **even during peak traffic times**.

- It also ensures high availability during system upgrades and maintenance.

8. Secret Management

Used to **store sensitive information**, such as API keys, database credentials, etc., using Kubernetes Secrets.
Mounting secrets as environment variables or files within pods.

In Application Deployment MongoDB secrets is used for :

- Securely provides sensitive data (like passwords) to containers

- Enhancing security by storing data in an encoded format and restricting access to the database.
- Preventing exposure and potential exploitation of confidential data.

9. Roles and Permissions

Role-Based Access Control (RBAC) in Kubernetes is used to regulate access to resources based on the user's roles and the permissions associated with it.

Example: Created a Cluster Role with read access only, on creation of a new user ‘sarthak’, did Cluster Role Binding of the user with the role having predefined permission, resulting the user to have read access only.

It helps implement the **principle of “least privilege”**, ensuring users have the minimum levels of access necessary to perform their tasks. This **enhances security by controlling who can access each API resource**.

10. Conclusion

This Kubernetes design document provides a comprehensive overview of the architecture, deployment strategies, storage, scaling, load balancing, secrets management, and user roles for the Minikube cluster hosting the React frontend, Ganache blockchain, and Express backend.