Name: Mahammad Anis Vahora Student ID: 101492068

Kubernetes Deployment Architecture for docker-ethereum Application

Introduction:

This document presents the Kubernetes deployment architecture for the docker-ethereum application, a decentralized application (dApp) featuring Ganache, React, Node.js, and Ethereum technologies. Designed for optimal performance, our architecture prioritizes high availability, scalability, and resource efficiency.

Components:

Storage

Stateless

Our Kubernetes deployment strategy uses stateless storage to guarantee that replacing or rescheduling pods can be done seamlessly and without compromising data integrity. Disassociating storage from specific pods minimises the effect of pod migrations or failures. This is in line with the transitory character of containerised applications running in a Kubernetes environment, allowing for resource efficiency and dynamic scaling.

2. Scaling

Horizontal Scaling

Kubernetes deployment design incorporates Horizontal Pod Autoscaler (HPA) to provide automatic pod replica adjustment depending on workload variables such as CPU utilisation. By allocating resources optimally through dynamic scaling, consistent performance is maintained even during periods of increased traffic. By minimising manual intervention, cutting down on operating expenses, and improving system responsiveness, HPA helps the docker-ethereum application's infrastructure become more flexible and effective.

3. Load Balancing

HPA for React and dApp

The docker-ethereum application and React both benefit from the implementation of Horizontal Pod Autoscaler (HPA), which guarantees responsiveness and effective resource allocation at different traffic volumes. Kubernetes enables this dynamic scaling, which dynamically modifies the number of pods to maintain consistent performance and maximise resource usage. Load balancing improves availability and responsiveness by dividing incoming traffic among several pods, guaranteeing continuous application access even during periods of high demand.

4. User Management and Role Assignment

Managing users and role assignments are essential to upholding security norms and protecting sensitive resources in a Kubernetes environment. An essential approach for managing access to different parts and information in the docker-ethereum application is Role-Based Access Control (RBAC). Through the use of RBAC, the application makes sure that each user is given rights that are appropriate for the job at hand, reducing the possibility of unwanted access or data breaches. By strictly controlling access to vital resources, this strategy improves the system's overall security posture and lowers the risks related to unauthorised use or data leakage.

Conclusion:

The architecture designed for deploying the docker-ethereum application on Kubernetes places a strong emphasis on scalability, reliability, and security. The architecture ensures excellent performance even in times of high demand by utilising techniques like as load balancing, horizontal scaling, and stateless storage to assure effective resource usage. by implementing strict access control guidelines within the Kubernetes cluster, user management and role assignment procedures significantly contribute to strengthening security measures. By using a proactive approach, possible security flaws are reduced, guaranteeing a safe environment for the efficient deployment and management of the docker-ethereum application within Kubernetes.