Kubernetes Deployment Architecture for Docker-Ethereum Application

Overview:-

The Kubernetes deployment architecture for the Docker-Ethereum application involves multiple components, including Ethereum server (eth-server), Ganache server (eth-ganache), and a React frontend (eth-react). Below is a detailed design rationale for each component:

Components

Ethereum Server (eth-server)

Deployment: Stateless

Rationale: Ethereum servers are typically stateful, but for scalability and ease of management, we have chosen a stateless deployment. This allows easy scaling horizontally and ensures that each instance of the deployment is independent.

Replicas: 2

Rationale: Deploying two replicas ensures high availability and load balancing for incoming requests.

Service Type: ClusterIP

Rationale: Since Ethereum servers don't need to be accessed from outside the cluster, a ClusterIP service type provides internal connectivity.

Ganache Server (eth-ganache)

Deployment: Stateless

Rationale: Similar to the Ethereum server, Ganache servers are deployed in a stateless manner for scalability and independence.

Replicas: 1

Rationale: One replica is sufficient for development and testing purposes, and it helps in keeping resource usage minimal.

Service Type: ClusterIP

Rationale: Ganache servers are internal components and don't require external access.

React Frontend (eth-react)

Deployment: Stateless

Rationale: The React frontend is designed to be stateless, allowing easy scaling and management.

Replicas: 1

Rationale: For simplicity, a single replica is deployed. However, it can be scaled based on the user load and requirements.

Service Type: LoadBalancer

Rationale: A LoadBalancer service type is used to expose the React frontend to external traffic. This is suitable for user access to the application.

Horizontal Pod Autoscaler (HPA): Enabled

Rationale: HPA automatically adjusts the number of replicas based on CPU utilization, ensuring optimal resource usage and application responsiveness.

Permissions and RBAC

Role and RoleBinding for Pod Reading (pod-reader):

A Role (pod-reader) is defined with permissions to get, watch, and list pods.

A RoleBinding (read-pods) assigns the pod-reader role to a user (vikram_singh) and a ServiceAccount (test-sa) within the default namespace.

A ClusterRole (pod-reader) and ClusterRoleBinding (pod-reader-global) allow the user vikram_singh to read pods globally.

Secret: ConfigMap

ConfigMap for Node App Configuration (node-app-config):

A ConfigMap is created to store configuration data for the Ethereum server, specifically the NODE_PORT.

The NODE_PORT is set to "8080" in this example.

Summary

This Kubernetes deployment architecture ensures a scalable, resilient, and well-organized setup for the Docker-Ethereum application. The choice of statelessness for the Ethereum and Ganache servers allows for flexibility and simplicity, while the React frontend is designed to handle external traffic efficiently. RBAC controls and a ConfigMap for configuration enhance security and configurability, respectively.