BCDV 4032 Lab 4

Nitu K Choudhary, 101529339

Design Document

KUBERNETES DEPLOYMENT ARCHITECTURE FOR DOCKER-ETHEREUM APPLICATION

1. Stateful vs Stateless for Backend Pod:
   As per my understanding the backend pod should be stateful. Ethereum nodes stable network identities and persistent storage, they need maintain its state. StatefulSets ensure that each pod has a unique identifier, persistent storage and maintains its state across restarts.
2. Deployment Options:
   Both StatefulSets and ReplicaSets will be used. StatefulSets is used for the Ethereum nodes because they require stable network identities and persistent storage. StatefulSets ensure that each pod has a unique identifier and maintains its state across restarts. Whereas ReplicaSets is used for managing stateless components of the application, which includes frontend. ReplicaSets ensure that the desired number of pod replicas are always running.
3. Storage - PersistentVolume (PV) and PersistentVolumeClaim (PVC):
   We use PVC to claim PV for stateful pods for storing the blockchain data which needs to persist across pod restarts. This is critical for the Ethereum nodes to maintain their state.
4. Scaling:
   I feel Horizontal Pod Autoscaler (HPA) would be the right choice to manage the scaling of both StatefulSets and ReplicaSets based on CPU and memory usage metrics. This way the application can scale accordingly to handle changing loads.
5. Load Balancing:
   Service of type LoadBalancer is used to distribute the incoming traffic across the pods. This service will be exposed externally and will be responsible for load balancing requests amongst the replicas.
6. Secrets:
   Kubernetes Secrets is used in this case to manage sensitive information such as private keys, passwords, and other credentials securely. There are two ways to provide access to secrets, they can be mounted as volumes or exposed as environment variables.
7. User Roles and RBAC:
   Role-Based Access Control (RBAC) can be used to create users and assign roles with specific permissions accordingly. This ensures secure and controlled access to the Kubernetes resources.