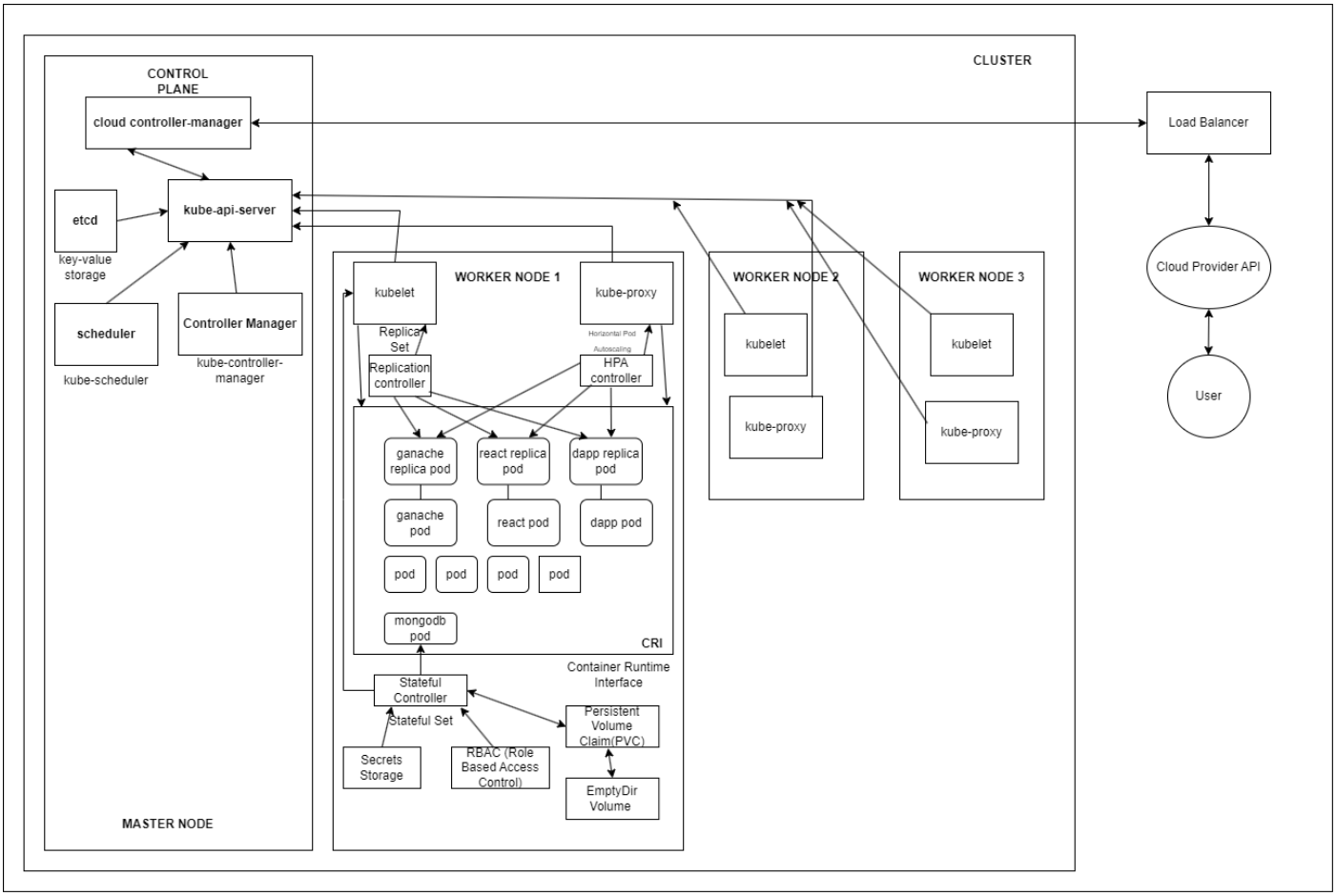
Design Document

1. Kubernetes Deployment Architecture



2. Design Rationale

1. **Deployments.** I will use ReplicaSets for the Docker-Ethereum application part (ganache, react, and dapp pods) because our dApp is stateless and doesn't have any persistent data. ReplicaSets helps to maintain a specified number of pod replicas at any given time, automatically replacing failed pods, and it provides high availability and scalability. Also, I will use StatefulSets for the MongoDB database because it needs persistent data.
2. **Storage.** For the storage, I will use EmpyDir volume and Persistent Volume Claims to store MongoDB data persistently because it's a database, and we need to store potentially sensitive data persistently, but I also want to make sure that volumes are deleted after pod deletion as I don't want to waste any resources, and EmptyDir will be automatically deleted because it's stored on the pod, not on the node as in HostPath volume. MongoDB pod will be connected to the stateful controller to manage the StatefulSet of the Mongo database.
3. **Scaling.** I will use Horizontal Pod Autoscaling (HPA) to change the number of pods based on CPU and memory utilization levels. HPA will allow me to automatically scale the number of pod replicas based on CPU and memory load, and it will make sure that the dApp can handle different workloads efficiently and doesn't waste any resources. I don't need a huge database in the Docker Ethereum app, and that's why I can scale a number of pods, not the size (Horizontal Pod Autoscaling) because I already have 1 pod for the database that doesn't store a lot of data, and I can increase the number of pods without risk of losing any important data.
4. **Load Balancing.** External LoadBalancer service will help to automatically distribute incoming traffic between all running pods. I don't need an internal LoadBalancer because HPA balances pods internally, and I need to only manage incoming traffic from outside the cluster. LoadBalancer helps to maintain high availability even when the load on the cluster is huge.
5. **Secrets.** Secrets will be used in the Docker Ethereum to store sensitive information such as API keys or passwords from Mongodb. It allows to avoid exposure of sensitive data because it's stored in an encrypted

way and initial passwords can’t be accessed by people checking code for this dApp. The Docker Ethereum app can access sensitive data securely without exposing it in the form of text inside the container.
6. **Role-Based Access Control (RBAC).** Role-Based Access Control allows to allow/prohibit access to certain Kubernetes resources and services using user roles and permissions. I will create a role in the cluster for users to have read-only access to make sure people who access the application don’t collapse it. With the use of roles and permissions, our dApp will follow the principle of least privilege to make sure that users have a minimum level of access within the cluster and only to necessary resources and nothing else.