Project Overview:

This project demonstrates the **end-to-end deployment of a three-tier application** using **Docker and Kubernetes**, following **DevOps best practices**. The application consists of:

- **Frontend:** React
- **Backend:** Node.js (Express)
- **Database:** MongoDB

The goal of this project is containerization, orchestration, persistent storage, service discovery, and application exposure through Kubernetes.

Project Objective

To design, containerize, and deploy a production-ready three-tier application using React, Node.js, and MongoDB, while implementing Kubernetes resources such as namespaces, deployments, services, persistent volumes, ingress, and load balancing, enabling browser-based access to the application.

Repository Structure

```
├── backend/
│   ├── Dockerfile
│   ├── src/
│   └── package.json
├── frontend/
│   ├── Dockerfile
│   └── src/
├── k8s/
│   ├── namespace.yaml
│   ├── mongodb-pv.yaml
│   ├── mongodb-pvc.yaml
│   ├── mongodb-deployment.yaml
│   ├── mongodb-service.yaml
│   ├── backend-deployment.yaml
│   ├── backend-service.yaml
│   ├── frontend-deployment.yaml
│   └── frontend-service.yaml
└── README.md
```

Deployment Steps

1. Build and Push Docker Images

   docker build -t <dockerhub-username>/chatapp-backend:latest ./backend
   docker push <dockerhub-username>/chatapp-backend:latest

   docker build -t <dockerhub-username>/chatapp-frontend:latest ./frontend
   docker push <dockerhub-username>/chatapp-frontend:latest

2. Create Namespace

   kubectl apply -f k8s/namespace.yaml

```
root@ip-172-31-5-158: ~/projects/full-stack_chatApp/k8s
root@ip-172-31-5-158:~/projects/full-stack_chatApp/k8s# kubectl get ns
NAME                 STATUS   AGE
chat-app-ns          Active   20h
default              Active   27h
ingress-nginx        Active   25h
kube-node-lease      Active   27h
kube-public          Active   27h
kube-system          Active   27h
local-path-storage   Active   27h
root@ip-172-31-5-158:~/projects/full-stack_chatApp/k8s#
```

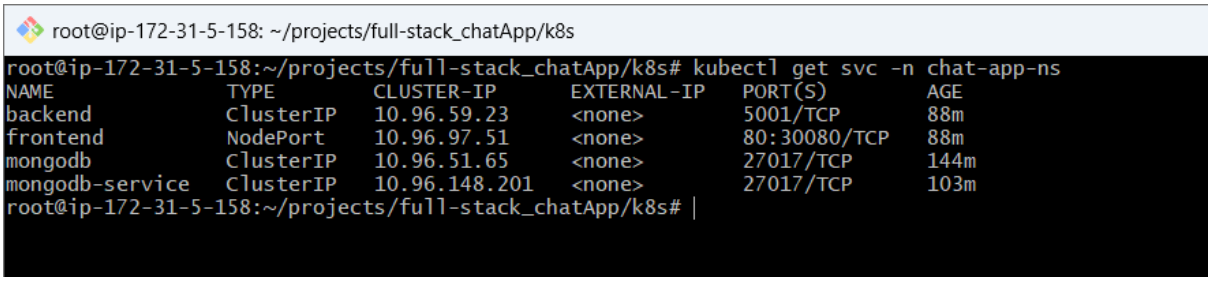
3. Configure MongoDB Persistent Storage
   kubectl apply -f k8s/mongodb-pv.yaml
   kubectl apply -f k8s/mongodb-pvc.yaml

```
root@ip-172-31-5-158: ~/projects/full-stack_chatApp/k8s
root@ip-172-31-5-158:~/projects/full-stack_chatApp/k8s# kubectl get pv -n chat-app-ns
NAME         CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM                    STORAGECLASS   VOLUMEATTRIBUTESCLASS   REASON   AGE
mongodb-pv   5Gi        RWO            Retain           Bound    chat-app-ns/mongodb-pvc   manual         <unset>                          17h
root@ip-172-31-5-158:~/projects/full-stack_chatApp/k8s# kubectl get pvc -n chat-app-ns
NAME          STATUS   VOLUME       CAPACITY   ACCESS MODES   STORAGECLASS   VOLUMEATTRIBUTESCLASS   AGE
mongodb-pvc   Bound    mongodb-pv   5Gi        RWO            manual         <unset>                 17h
root@ip-172-31-5-158:~/projects/full-stack_chatApp/k8s#
```

4. Deploy MongoDB

   kubectl apply -f k8s/mongodb-deployment.yaml
   kubectl apply -f k8s/mongodb-service.yaml

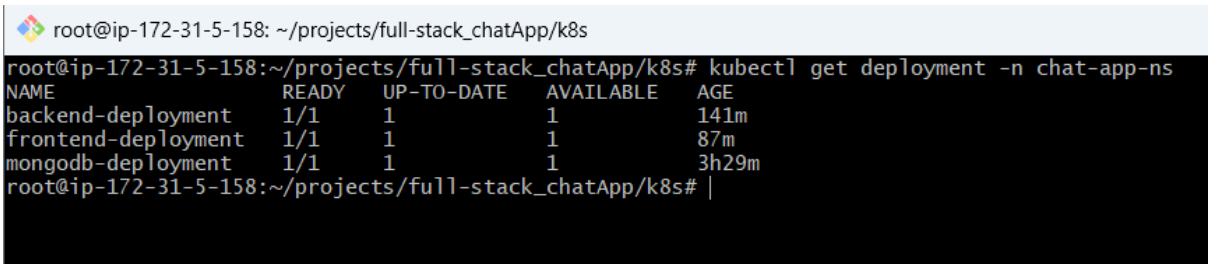
```
root@ip-172-31-5-158: ~/projects/full-stack_chatApp/k8s
root@ip-172-31-5-158:~/projects/full-stack_chatApp/k8s# kubectl get deployment -n chat-app-ns
NAME                  READY   UP-TO-DATE   AVAILABLE   AGE
backend-deployment    1/1     1            1           141m
frontend-deployment   1/1     1            1           87m
mongodb-deployment    1/1     1            1           3h29m
root@ip-172-31-5-158:~/projects/full-stack_chatApp/k8s#
```

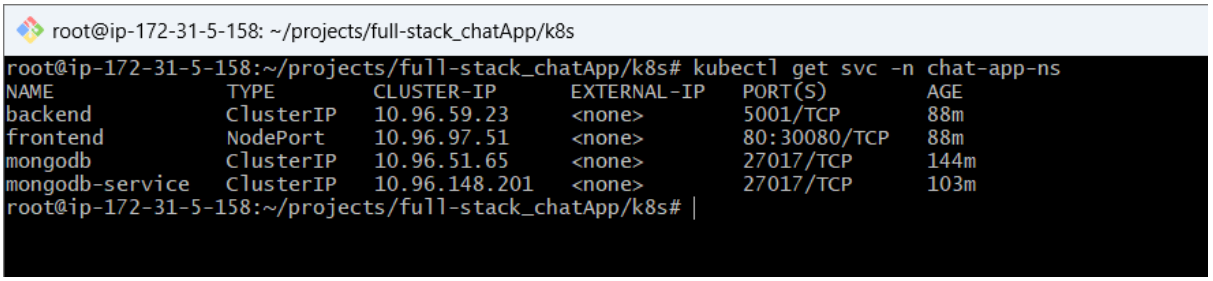


5. Deploy Backend and Frontend

   kubectl apply -f k8s/backend-deployment.yaml
   kubectl apply -f k8s/backend-service.yaml

   kubectl apply -f k8s/frontend-deployment.yaml
   kubectl apply -f k8s/frontend-service.yaml





6. Accessing the Application

   NodePort
   kubectl get svc -n chat-app-ns
   http://<EC2-PUBLIC-IP>:<NodePort>

7. Verification Commands

   kubectl get pods -n chat-app-ns
   kubectl get svc -n chat-app-ns
   kubectl logs <pod-name> -n chat-app-ns

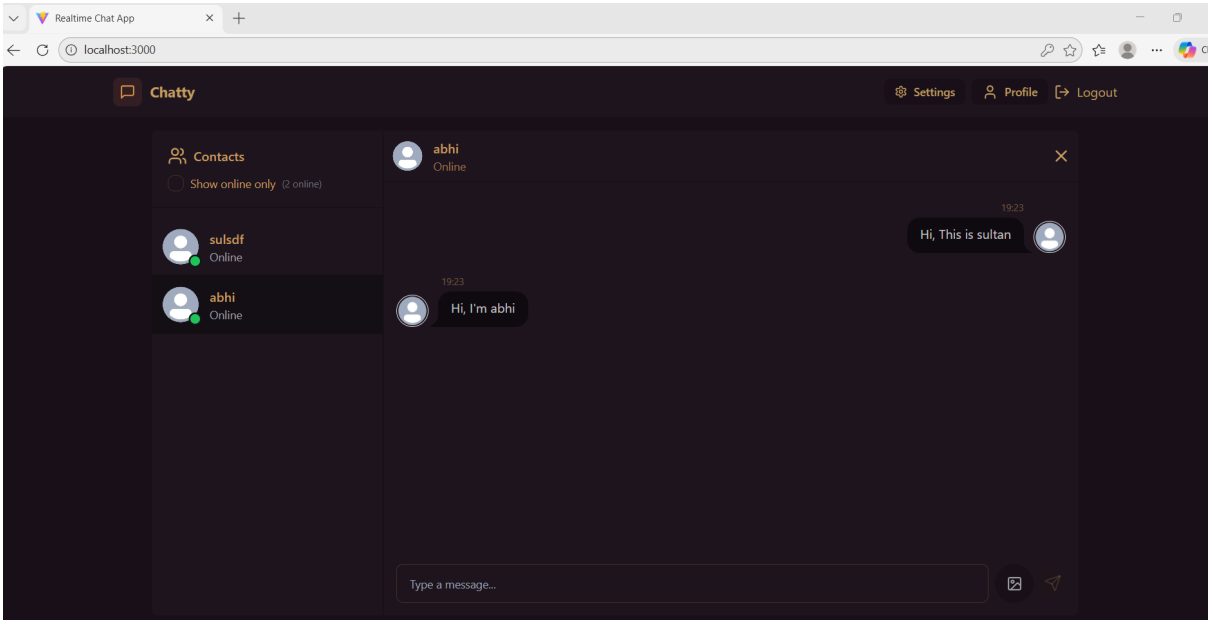
8. Key Learnings

   - Kubernetes service discovery using ClusterIP
   - Persistent storage with PV & PVC
   - Debugging CrashLoopBackOff and DNS issues

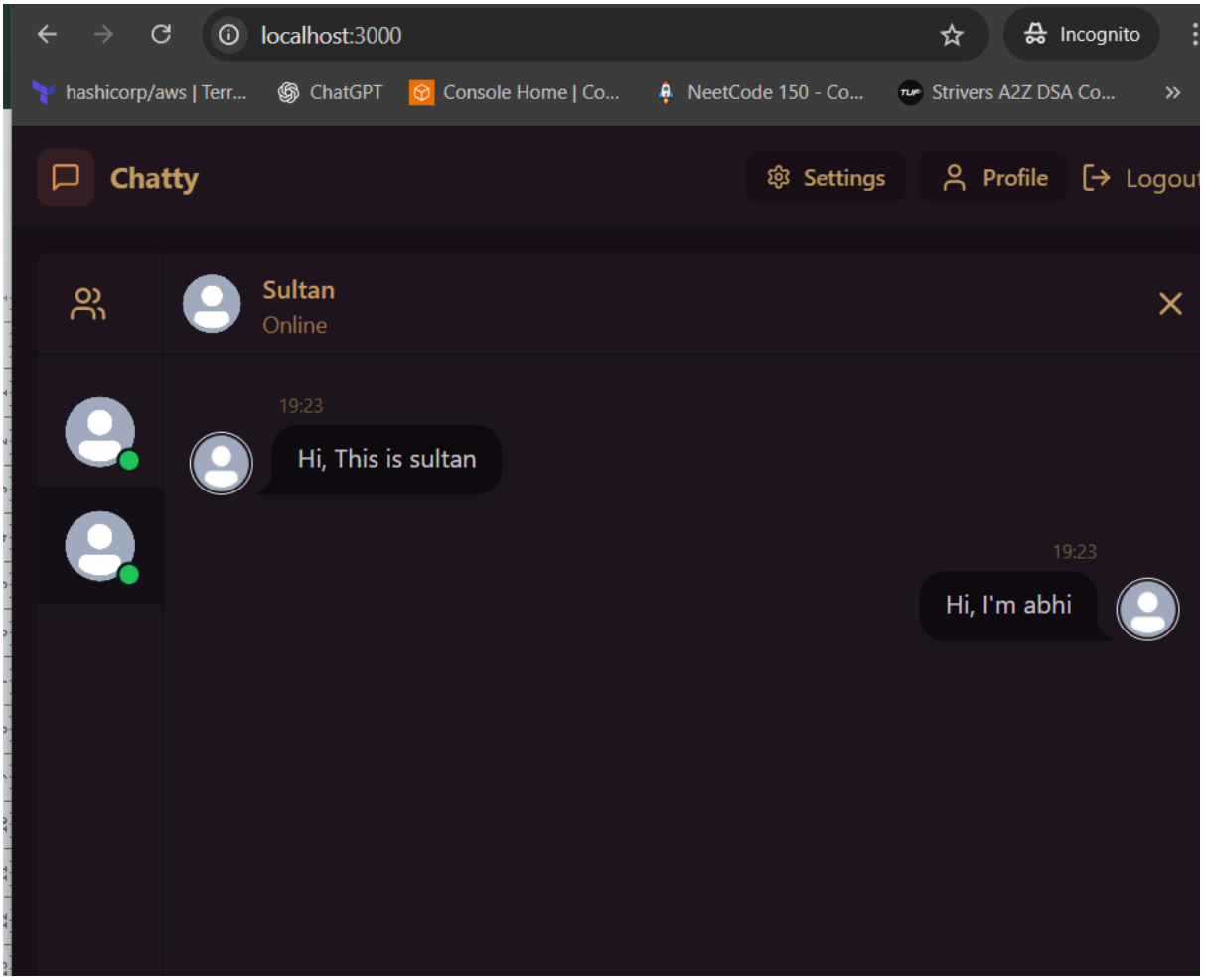
- Environment variable management
- Exposing applications from KIND clusters

9. Validate the output

USER 1



USER 2:



Troubleshooting:

1. Error: PersistentVolume: json: cannot unmarshal string into Go struct field PersistentVolumeSpec.spec.hostPath of type v1.HostPathVolumeSource

   `spec.hostPath` is written as a string, but Kubernetes expects it to be an object.
   `hostPath` expects this structure:
   hostPath:
   path: <string>
   type: <optional>

2. PVC stays in `Pending`, it means Kubernetes cannot find a matching PersistentVolume (PV).
   Reason: StorageClass mismatch

3. no matches for kind "Deployment" in version "v1" ensure CRDs are installed first
   Reason: Kubernetes simply cannot find Deployment in v1.

4. 2 pods have issue like CrashLoopBackOff

   `CrashLoopBackOff` means **the container starts → crashes → Kubernetes restarts it → repeats**.

   Resource: server is running on PORT:undefined MongoDB connection error: Error: MONGODB_URI environment variable is required at connectDB (file:///app/src/lib/db.js:6:13) at Server.<anonymous> (file:///app/src/index.js:46:3)