CODEKERDOS SRE LAB INSTRUCTIONS

Infrastructure Troubleshooting & State Reconciliation

Step 1: Deployment & Initial State

Deploy the application infrastructure using the provided GitLab URL. This will initialize the frontend and database components.

```
kubectl apply -f https://gitlab.com/debjyotimt/e-commerce-app-3-tier/-/raw/main/lab.yaml
```

> **Expected Result:** All resources (Deployments, Services, ConfigMaps, and PVCs) should be created successfully in the cluster.

Step 2: Traffic Routing & Service Health

Establish a local connection to the `sky-frontend-svc` on port `8080`. Open your browser and navigate to `localhost:8080`.

> **Expected Result:** The browser should initially fail to load the page. After you identify and resolve the mismatch between the Service and the Pod, the web page should load successfully, showing the "CodeKerdos SKY Lab" header.

Step 3: Database Connectivity

Once the web page loads, observe the "Database Status" indicator. Examine the pods in your namespace using `kubectl get pods`.

> **Expected Result:** The web UI should report **"Database: OFFLINE"**. In the terminal, the `sky-db` pod should be in a `Pending` state due to an unbound volume.

Step 4: Persistent Volume Reconciliation

Investigate the `PersistentVolumeClaim` (PVC) and identify the storage configuration error. Resolve the issue while accounting for Kubernetes resource immutability.

> **Expected Result:** The PVC should transition to **"Bound"** status. After ensuring the pod reconciles with this change, the `sky-db` pod should move to **"Running"** and **"1/1 Ready"** status.

Step 5: Full System Integration

Perform a final verification of the end-to-end communication between the frontend application and the database backend.

> **Expected Result:** Refreshing the browser at `localhost:8080` should now display a **Green** status bar and **"Database: ONLINE"**. All system components are now healthy and communicating.