DEPLOYMENT PROCESS

Step 1: Deploying Your Kubernetes Components

Now that everything's ready, we'll use the `apply` command to create your resources in Kubernetes. The order matters slightly for dependencies (like PVC before Deployment, or services that depend on other services' stable names).

Navigate to the directory where you saved your YAML files in your WSL2 terminal (or wherever your `microk8s` commands run).

1. Deploy MongoDB (PVC, Deployment, Service):

Start with the PersistentVolumeClaim (PVC) first, as your MongoDB Deployment relies on it.

```
microk8s.kubectl apply -f mongodb-pvc.yaml
```

○

Then, deploy the MongoDB Deployment.

```
microk8s.kubectl apply -f mongodb-deployment.yaml
```

○

And finally, the MongoDB Service.

```
microk8s.kubectl apply -f mongodb-service.yaml
```

○

2. Deploy Backend (Deployment, Service):

Deploy the Backend Deployment.

```
microk8s.kubectl apply -f backend-deployment.yaml
```

○

Deploy the Backend Service.

```
microk8s.kubectl apply -f backend-service.yaml
```

- ○

3. Deploy Frontend (Deployment, Service):

Deploy the Frontend Deployment.

```
microk8s.kubectl apply -f frontend-deployment.yaml
```

- ○

Deploy the Frontend Service.

```
microk8s.kubectl apply -f frontend-service.yaml
```

- ○

4. You should see output like `persistentvolumeclaim/mongodb-pvc created`, `deployment.apps/mongodb-deployment created`, etc.

Step 2: Checking the Status of Your Deployments

After applying, it takes a moment for Kubernetes to pull images and start Pods. Here are the commands to check their status:

Check Pods: This shows you the individual running containers.

```
microk8s.kubectl get pods
```

1. You should see pods for `mongodb`, `backend`, and `frontend` with a `STATUS` of `Running` after a short while. It might show `ContainerCreating` first. Give it a minute or two.

Check Deployments: This shows you if your Deployments are healthy and if the desired number of replicas are running.

```
microk8s.kubectl get deployments
```

2. Look for `READY` columns like `1/1` for each of your deployments.

Check Services: This shows you the Services you created and their internal/external IPs.

microk8s.kubectl get services

3. You should see `mongodb-service`, `backend-service` (both `ClusterIP`), and `frontend-service` (`NodePort`). For `frontend-service`, note the `PORT(S)` column; it will show something like `80:3XXXX/TCP`. The `3XXXX` is the dynamically assigned `NodePort` you'll use to access your frontend.

Check Persistent Volume Claims (PVCs): Confirm your MongoDB storage is claimed.

microk8s.kubectl get pvc

4. You should see `mongodb-pvc` with a `STATUS` of `Bound`.

5. Troubleshooting (if things don't look right):

Get more details about a Pod: If a Pod isn't `Running`, this is your go-to.

microk8s.kubectl describe pod <pod-name> # Replace <pod-name> with the full name from 'get pods'

- Look at the `Events` section for clues.

View container logs: If a Pod is crashing or not starting properly.

microk8s.kubectl logs <pod-name>

-

Step 3: Accessing the Frontend on Your Local Browser

Since we configured the `frontend-service` as `NodePort`, here's how you access it:

1. Get MicroK8s Node IP:

MicroK8s runs inside a virtual machine (often within WSL2 on Windows). You need the IP address of that virtual machine.

microk8s.kubectl get nodes -o wide

    - Look for the `INTERNAL-IP` column. This is your Node IP. It's often `10.152.183.1` or similar for MicroK8s.
2. Get Frontend NodePort:

    - From `microk8s.kubectl get services`, find your `frontend-service`. The `PORT(S)` column will show `80:<NodePort>/TCP`. Note down that `<NodePort>` number (e.g., `30XXX`).
3. Access in Browser:

Open your web browser (on your Windows machine) and navigate to:
http://<Node-IP-from-step-1>:<NodePort-from-step-2>

-
- For example, if your Node IP is `10.152.183.1` and your NodePort is `30080`, you would go to `http://10.152.183.1:30080`.

You should now see your `linkshare` frontend application! If you make API calls, they should internally route to your backend and then to MongoDB, all within your MicroK8s cluster.

This is a significant milestone! Do you have any questions about these deployment steps or troubleshooting, or would you like to move on to a summary of what we've learned and potentially a quiz?