Minikube Multi-node Cluster Lab

Lab 2: Creating a 3-Node Minikube Cluster

🎯 Objective:

- Set up a 3-node Kubernetes cluster using Minikube on your local machine.

`NOTE:` Minikube Issue #15725: Docker driver on macOS arm64 shows no allocatable memory

🛠️ Steps

1. **Start the Primary Control Plane Node**

```
minikube delete
minikube start --nodes=3 --driver=docker --memory=8128
```

This creates the initial node (`minikube`) as the control plane node using the Docker driver.

2. **Verify Cluster Nodes**

```
kubectl get nodes
```

You should see:

```
NAME           STATUS   ROLES           AGE   VERSION
minikube       Ready    control-plane   40s   v1.32.0
minikube-m02   Ready    <none>          15s   v1.32.0
minikube-m03   Ready    <none>          4s    v1.32.0
```

3. **Check Node Labels and Roles**

```
kubectl get nodes --show-labels
```

4. **View Nodes by using Describe**

```
kubectl describe nodes
```

Example `Describe` output:

```
Name:               minikube
Roles:              control-plane
Labels:             beta.kubernetes.io/arch=arm64
                    beta.kubernetes.io/os=linux
                    kubernetes.io/arch=arm64
                    kubernetes.io/hostname=minikube
                    kubernetes.io/os=linux
                    minikube.k8s.io/commit=dd5d320e41b5451cdf3c01891bc4e13d189586ed
                    minikube.k8s.io/name=minikube
                    minikube.k8s.io/primary=true
                    minikube.k8s.io/updated_at=2025_04_21T18_58_33_0700
                    minikube.k8s.io/version=v1.35.0
                    node-role.kubernetes.io/control-plane=
                    node.kubernetes.io/exclude-from-external-load-balancers=
Annotations:        kubeadm.alpha.kubernetes.io/cri-socket: unix:///var/run/cri-dockerd.sock
                    node.alpha.kubernetes.io/ttl: 0
                    volumes.kubernetes.io/controller-managed-attach-detach: true
```

5. **Deploy a Multi-Pod App**

Use the following resource definition and create a file named `busy3.yaml`. This file will be used to in the next step.

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: busybox-demo
spec:
  replicas: 3
  selector:
    matchLabels:
      app: busybox-demo
  template:
    metadata:
      labels:
        app: busybox-demo
    spec:
      containers:
        - name: busybox
          image: busybox:musl
          command: ["sleep", "3600"]  # keep container running for 1 hour
```

6. **Deploy a the defined resources using the YAML file**

Use the following resource definition and create a file named `busy3.yaml`. This file will be used to in the next step.

```
kubectl apply -f busy3.yaml
```

View the status of your Pods using the following command, note the use of the `-o wide` flag to see detailed information about each Pod, including their node assignments:

```
kubectl get pods -o wide
```

Example output with the NODE column:

```
NAME                          READY   STATUS    RESTARTS   AGE   IP           NODE           NOMINATED NODE   READINESS GATES
busybox-demo-b4d8498f9-ggx4f  1/1     Running   0          29s   10.244.0.5   minikube       <none>           <none>
busybox-demo-b4d8498f9-jffmx  1/1     Running   0          29s   <none>       minikube-m02   <none>           <none>
busybox-demo-b4d8498f9-kww94  1/1     Running   0          29s   <none>       minikube-m03   <none>           <none>
```

Each Pod should be scheduled across available nodes.

8. **Cleanup (Ask Instructor for permission to clean up)**

```
kubectl delete -f busy3.yaml
```