Step 1: Copy the Full Content

Copy everything from this block:

Kubernetes Assignments 1-5 Detailed Summary

Assignment 1: Deploy a Kubernetes Cluster & NGINX Deployment

Tasks

1. Deploy a Kubernetes cluster with 3 nodes.
2. Create an NGINX deployment with 3 replicas.

Deployment YAML

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:latest
        ports:
        - containerPort: 80
```

Commands

kubectl apply -f nginx-deployment.yaml
kubectl get pods

Problems Faced

- Error: unknown field "metadata.spec"
    - Cause: spec nested incorrectly inside metadata.
    - Solution: Correct YAML indentation.
- Lesson: YAML indentation is critical in Kubernetes.

Assignment 2: Create NodePort Service for NGINX

NodePort Service YAML

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-nodeport
spec:
  type: NodePort      # Exposes service outside cluster
  selector:
    app: nginx
  ports:
    - port: 80
      targetPort: 80
      nodePort: 30080
```

Commands

kubectl apply -f nginx-nodeport.yaml
kubectl get svc

Problems & Resolutions

1. Connection refused with curl
    - Cause: NodePort not directly accessible in WSL Minikube.
    - Solution: Use minikube service nginx-nodeport or port-forward.
2. Minikube service "SVC_NOT_FOUND"
    - Cause: kubectl context was mycluster, not Minikube.
    - Solution: kubectl config use-context minikube
3. Port already in use
    - Cause: Local port busy.
    - Solution: Use a different port, e.g., 8081.

Assignment 3: Scale Deployment

Command

kubectl scale deployment nginx-deployment --replicas=5
kubectl get pods

- Deployment automatically balances traffic across 5 pods.
- Lesson: Services route traffic to all pods automatically.

Assignment 4: Change NodePort → ClusterIP

ClusterIP Service YAML

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-clusterip
spec:
  type: ClusterIP      # Internal-only access
  selector:
    app: nginx
  ports:
    - port: 80
      targetPort: 80
```

- NodePort removed; service is internal.
- Lesson: ClusterIP is default; NodePort exposes externally.

Assignment 5: Second NGINX Deployment + Ingress (Conceptual)

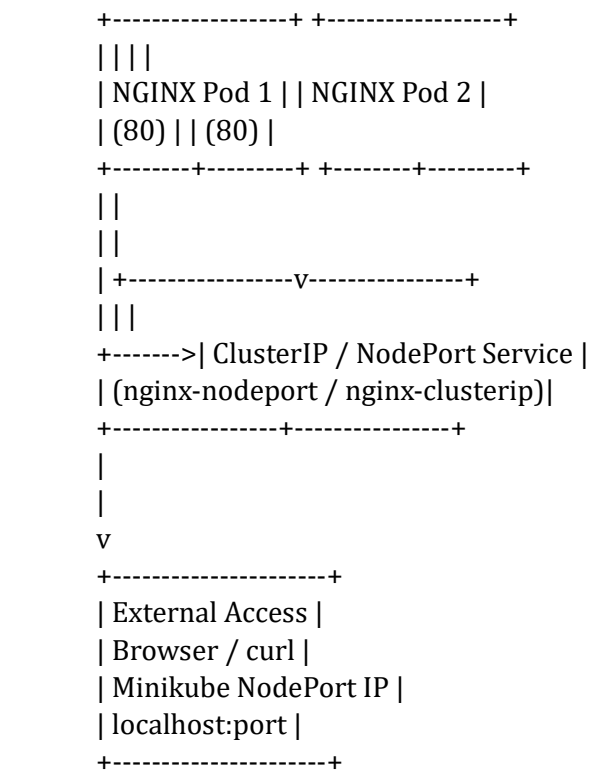
- Deploy another NGINX deployment (3 replicas).
- Create ClusterIP service for new deployment.
- Use Ingress to route paths: /app1 -> first NGINX, /app2 -> second NGINX.
- Ingress acts as a single entry point for multiple services.

Key Learnings

1. YAML indentation matters.
2. Kubernetes contexts: Minikube vs other clusters.
3. NodePort may not work directly in WSL; use port-forward or minikube service.
4. Port-forward requires free local port.
5. Service types: ClusterIP = internal, NodePort = external, LoadBalancer = cloud LB.

6. Pods & service labels must match.
7. Scaling replicas is simple; services automatically load-balance.
8. Ingress centralizes traffic for multiple services.

Traffic Flow Diagram (Text Version)



Summary Table of Service Types

Type	Accessible From	NodePort Needed?	Use Case
ClusterIP	Internal pods only	No	Internal communication between pods
NodePort	External + internal	Yes	Expose service outside cluster
LoadBalancer	External + internal	Optional	Cloud environments with LB

Instructor Notes

- Always verify cluster, context, and namespace before creating resources.
- Use kubectl get pods, svc, endpoints for debugging.
- Service type determines accessibility.
- Replicas scaling is automatic via Deployment.
- Ingress routes multiple services via paths/domains.