Kubernetes Services, Ingress & Load Balancing

Day 25 of DevOps 90 Challenge

Why Networking Matters in Kubernetes

- Pods are ephemeral and have dynamic IPs
- Need stable way to access Pods
- Kubernetes uses Services and Ingress for networking and load balancing

What is a Kubernetes Service?

- Abstracts a set of Pods
- Provides a stable IP and DNS name
- Enables load balancing across Pods

Types of Services

Type	Scope	Use Case
ClusterIP	Internal cluster only	Default, internal microservices
NodePort	External access via node IP and static port	Simple access during development
LoadBalancer	External access via cloud provider LB	Production, cloud environments
ExternalName	Maps to external DNS	Redirect to external services

Anatomy of a Service YAML

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
  type: NodePort
```

- selector: Finds matching Pods
- port: Port on the Service
- targetPort: Port on the container
- type: Exposure method

Accessing Services

- ClusterIP: Internal DNS (e.g., nginx-service.default.svc.cluster.local)
- NodePort: http://<NodeIP>:<NodePort>
- LoadBalancer: External IP from cloud provider
- Command: minikube service nginx-service

What is Ingress?

- Ingress manages external HTTP/S access to Services
- Works at Layer 7 (Application Layer)
- Supports routing, SSL, virtual hosting

Ingress vs Service

Feature	Service (NodePort/LB)	Ingress
Layer	4 (TCP)	7 (HTTP)
Use case	Basic port exposure	Smart routing, SSL, domains
SSL handling	External	Centralized in Ingress

Ingress Controller

- Not built-in by default
- Needs to be installed (e.g., nginx, Traefik, HAProxy)
- Handles incoming requests, maps to Services

Ingress YAML Example

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: example-ingress
spec:
  rules:
    - host: example.com
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: nginx-service
                port:
                  number: 80
```

- host: Domain
- path: URL path
- backend: Target service and port

Load Balancing in Kubernetes

- Internal (Service): ClusterIP balances across Pods
- External (Ingress/LoadBalancer): Handles routing from users
- Scaling: Services integrate with HPA and Deployments

Real-World Use Case

- Users visit app.mysite.com
- Ingress routes to frontend-service
- Frontend talks to backend-service via ClusterIP
- LoadBalancer ensures external accessibility

Demo Commands

- kubectl apply -f deployment.yaml
- kubectl apply -f service.yaml
- kubectl apply -f ingress.yaml
- minikube tunnel
- kubectl get ingress

Cleanup

- kubectl delete -f ingress.yaml
- kubectl delete -f service.yaml
- kubectl delete -f deployment.yaml

Recap

Concept	Purpose
Service	Stable access to ephemeral Pods
NodePort	Expose service on a node's IP and port
Ingress	Smart HTTP routing
LoadBalancer	External access via cloud infra
Ingress Controller	Manages ingress traffic

Thank You

Stay tuned for Day 26: Helm & Package Management in Kubernetes