Horizontal Pod Autoscaler (HPA)

Automatically scales the number of pods

Why Use HPA?

- Efficient resource usage
- Adapt to traffic/load changes
- Improve availability and responsiveness

HPA Key Fields

- **minReplicas / maxReplicas**: the range of pods allowed
- **metrics**: resource-based or custom metrics
- **scaleTargetRef**: points to the target workload (Deployment, StatefulSet, etc.)

Example: CPU-based HPA

```yaml
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
  name: myapp-hpa
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: myapp-deployment
  minReplicas: 1
  maxReplicas: 5
  metrics:
  - type: Resource
    resource:
      name: cpu
      target:
        type: Utilization
        averageUtilization: 50
```

Target Deployment

- Uses a CPU-bound loop
- Includes CPU resource requests and limits

```yaml
resources:
  requests:
    cpu: "100m"
  limits:
    cpu: "200m"
```

Metrics Server Required

Make sure the Kubernetes Metrics Server is installed:

```
kubectl top pods
kubectl top nodes
```

Monitor Autoscaling

```
kubectl get hpa
kubectl describe hpa myapp-hpa
```