📘 Kubernetes Notes – (Namespaces to HPA/VPA)

📌 Table of Contents

📁 Namespaces

Namespaces in Kubernetes are like virtual clusters. They allow you to divide cluster resources between multiple users (via resource quota).

✅ Example

```
apiVersion: v1

kind: Namespace

metadata:

 name: dev



kubectl create -f namespace.yaml

kubectl get namespaces

kubectl config set-context --current --namespace=dev
```

🧱 Pods

A Pod is the smallest deployable unit in Kubernetes. It can contain one or more containers that share storage/network.

✅ Example

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-pod
  labels:
    app: nginx
spec:
  containers:
    - name: nginx
      image: nginx:latest
      ports:
        - containerPort: 80
```

```
kubectl apply -f pod.yaml
kubectl get pods
```

🔁 ReplicaSets

ReplicaSet ensures a specified number of pod replicas are running at all times.

✅ Example

```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: nginx-replicaset
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx
```

🚀 Deployments

Deployments manage ReplicaSets and provide declarative updates for Pods and ReplicaSets.

✅ Example

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx
```

```
kubectl apply -f deployment.yaml
kubectl rollout status deployment nginx-deployment
```

🌐 Services

Services expose a set of pods to other services inside/outside the cluster.

🔹 Types of Services

1. **ClusterIP** – default; accessible within the cluster.
2. **NodePort** – accessible via <NodeIP>:<NodePort>.
3. **LoadBalancer** – external IP for cloud.

✅ Example: NodePort

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

⚙️ Resource Requests and Limits

Kubernetes allows setting CPU and Memory resources for containers.

- **Request** = minimum guaranteed
- **Limit** = maximum allowed

✅ Example

```
resources:
  requests:
    memory: "64Mi"
    cpu: "250m"
  limits:
    memory: "128Mi"
    cpu: "500m"
```

📈 Horizontal Pod Autoscaler (HPA)

HPA automatically scales the number of pods in a deployment based on observed CPU/memory utilization.

✅ Example

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

```
kubectl autoscale deployment nginx-deployment --cpu-percent=50 --min=1 --max=5
```

📊 Vertical Pod Autoscaler (VPA)

VPA automatically adjusts CPU and memory **requests** and **limits** for containers.

✅ Example

```
apiVersion: autoscaling.k8s.io/v1
kind: VerticalPodAutoscaler
metadata:
 name: nginx-vpa
spec:
 targetRef:
  apiVersion: "apps/v1"
  kind: Deployment
  name: nginx-deployment
 updatePolicy:
  updateMode: "Auto"
```

Note: VPA may restart pods to apply new resource values.

🔄 HPA vs VPA

Feature	HPA	VPA
Scales	Number of pods	CPU/memory of pods
Based On	Metrics like CPU/Memory	Historical + live resource usage
Impact	More pods	Pod restarts
Use Case	Stateless apps	Stateful apps (like DB)
Update Mode	Continuous	Off/Auto/Initial

✅ Summary Commands

```
# Create all from YAMLs
kubectl apply -f <filename>.yaml


# Check HPA
kubectl get hpa


# Check VPA (if vpa components installed)
```

```
kubectl get vpa


# Set context to namespace

kubectl config set-context --current --namespace=dev
```

📝 **Pro Tip:** Use kubectl describe <resource> to debug deeper!