📈 Kubernetes Autoscaling – HPA & VPA (Real-World Guide)

This repository demonstrates **Horizontal Pod Autoscaler (HPA)** and **Vertical Pod Autoscaler (VPA)** with real YAML manifests and production architecture.

📚 Table of Contents

🧠 Autoscaling Overview

Kubernetes supports **two main autoscaling mechanisms**:

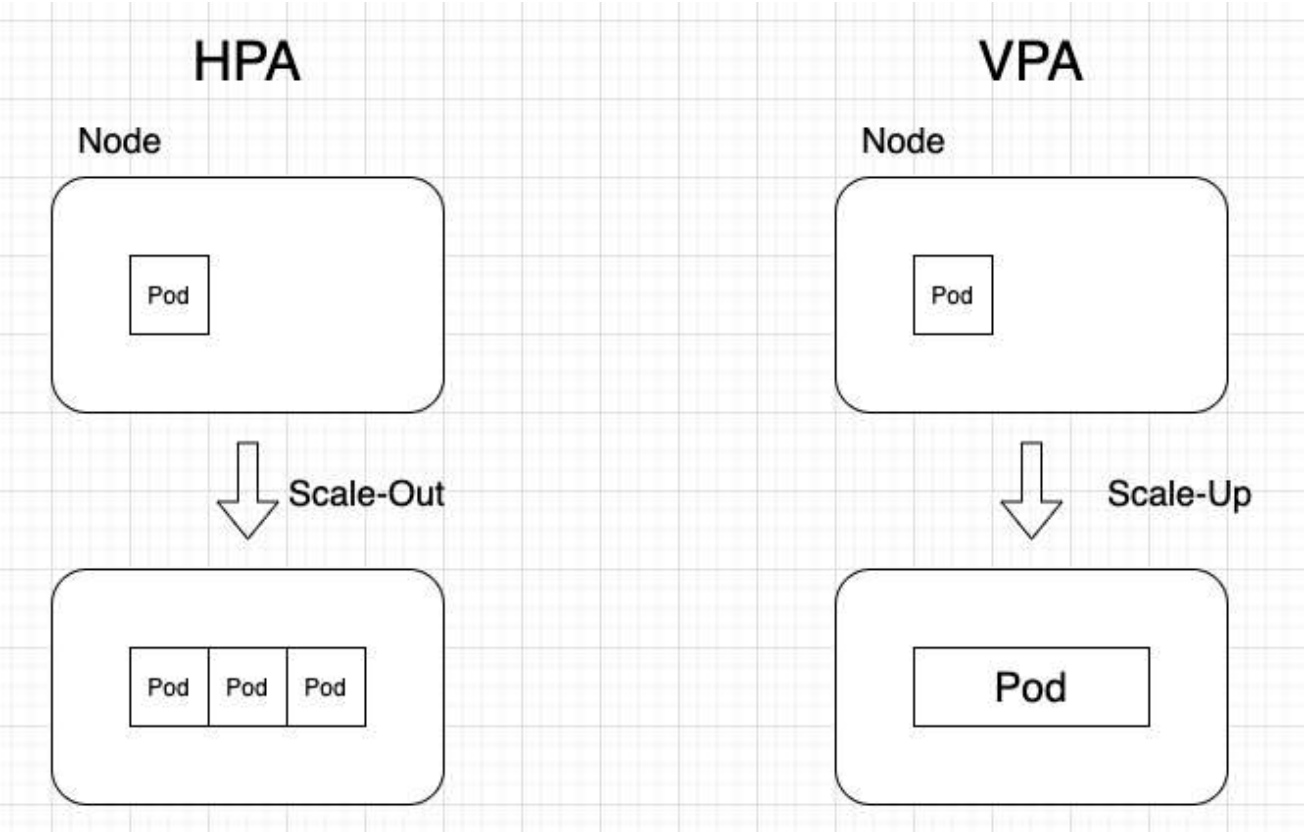
Autoscaler	Scales	Direction
HPA	Pods	Horizontal
VPA	CPU / Memory	Vertical

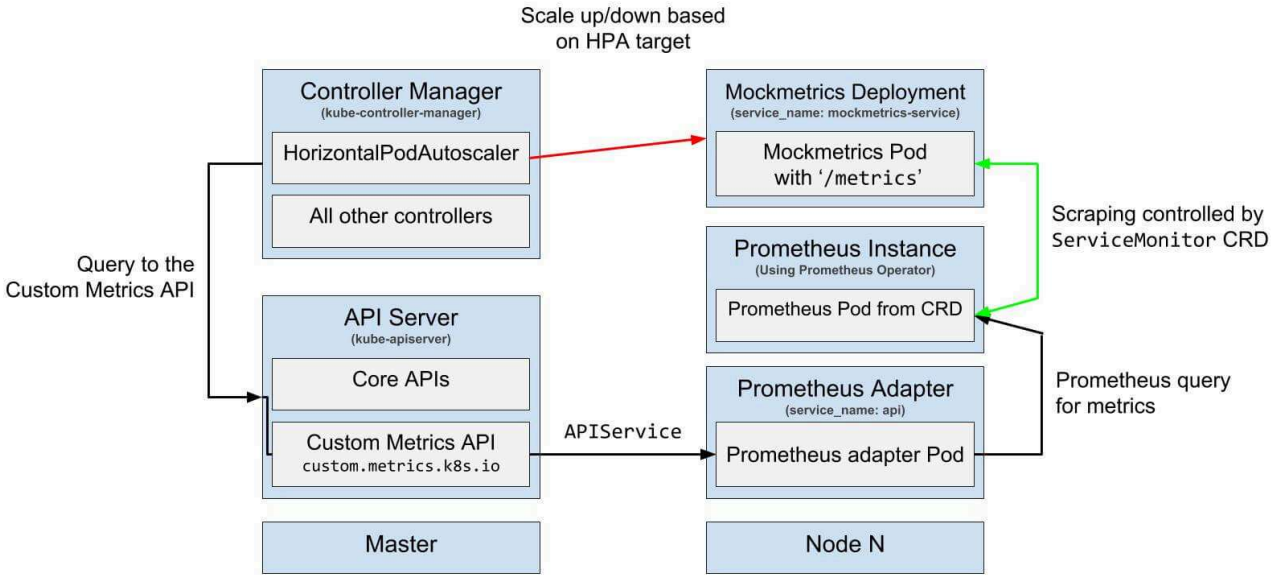
> ⚠️ HPA and VPA should NOT manage the same resource simultaneously

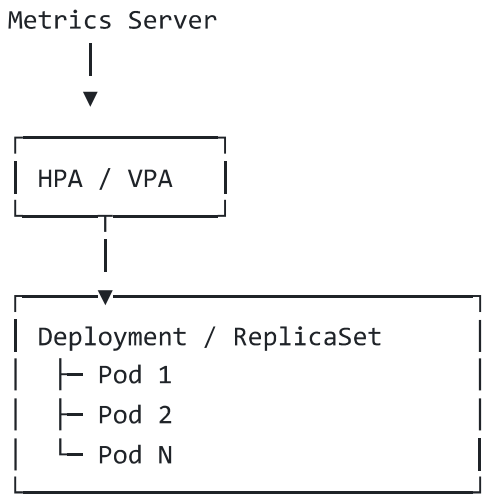
HPA vs VPA (Critical Difference)

Feature	HPA	VPA
Scales pods	✅	❌
Scales CPU/Memory	❌	✅
Restart pods	❌	✅
Uses Metrics Server	✅	❌
Used in production	⭐⭐⭐⭐⭐	⭐⭐⭐
Stateful workloads	❌	⚠️

Autoscaling Architecture







📊 Horizontal Pod Autoscaler (HPA)

🔍 What HPA Does

- Automatically increases/decreases **replica count**
- Based on:
  - CPU
  - Memory

- Custom metrics (Prometheus)

📦 HPA Prerequisites

1️⃣ Metrics Server (Mandatory)

```
kubectl get deployment metrics-server -n kube-system
```

If missing:

```
kubectl apply -f https://github.com/kubernetes-sigs/metrics-
server/releases/latest/download/components.yaml
```

2️⃣ Resource Requests MUST Exist

```yaml
resources:
  requests:
    cpu: "100m"
    memory: "128Mi"
```

Without this → **HPA will not work**

📜 HPA YAML (`hpa.yml`)

```yaml
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
  name: app-hpa
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: demo-app
  minReplicas: 2
  maxReplicas: 10
  metrics:
```

```
  - type: Resource
    resource:
      name: cpu
      target:
        type: Utilization
        averageUtilization: 60
```

🚀 How HPA Works (Flow)

```
CPU > 60% → scale UP
CPU < 60% → scale DOWN
```

🔎 Verify HPA

```
kubectl get hpa
kubectl describe hpa app-hpa
```

Live watch:

```
kubectl get pods -w
```

📐 Vertical Pod Autoscaler (VPA)

🔍 What VPA Does

- Automatically adjusts **CPU & memory**
- Based on historical usage
- Requires **pod restart**

⚠️ VPA Limitations (Important)

Limitation	Impact
Pod restart	Downtime risk
Not for stateless	Avoid
Not with HPA	Conflict

📦 VPA Installation (One-Time)

```
kubectl apply -f
https://github.com/kubernetes/autoscaler/releases/latest/download/vertical-pod-
autoscaler.yaml
```

Verify:

```
kubectl get pods -n kube-system | grep vpa
```

📜 VPA YAML (`vpa.yml`)

```yaml
apiVersion: autoscaling.k8s.io/v1
kind: VerticalPodAutoscaler
metadata:
  name: app-vpa
spec:
  targetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: demo-app
  updatePolicy:
    updateMode: Auto
```

🔁 VPA Update Modes

Mode	Behavior
Off	Only recommendations
Initial	On pod creation
Auto	Live updates (restart)

🔎 Verify VPA

```
kubectl describe vpa app-vpa
```

Check recommendations:

```
kubectl get vpa
```

🧪 Real-World Use Cases

Scenario	Use
Web apps	HPA
APIs	HPA
Batch jobs	VPA
ML workloads	VPA
Stateful apps	⚠️

🚨 Common Autoscaling Issues (On-Call)

Issue	Cause
HPA not scaling	No resource requests
VPA not updating	Mode set to Off
Pods restarting	VPA Auto mode
Metrics unavailable	Metrics server down

Fix:

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

✅ Production Best Practices

✔ Always set **requests & limits** ✔ Prefer **HPA for stateless apps** ✔ Use **VPA in recommendation mode first** ✔ Never combine **HPA + VPA on CPU** ✔ Use **Cluster Autoscaler** with HPA ✔ Monitor scaling events

🧭 Autoscaling Decision Guide

Requirement	Choose
Handle traffic spikes	HPA
Optimize resource usage	VPA
Zero downtime	HPA
Cost optimization	VPA
High availability	HPA

🔥 Interview-Ready One-Liners

- HPA scales **pods**
- VPA scales **resources**
- HPA needs **metrics-server**
- VPA restarts pods
- Never mix HPA & VPA on CPU