QUES 9 : Configure autoscaling in your cluster (Horizontal scaling)

SOLN :

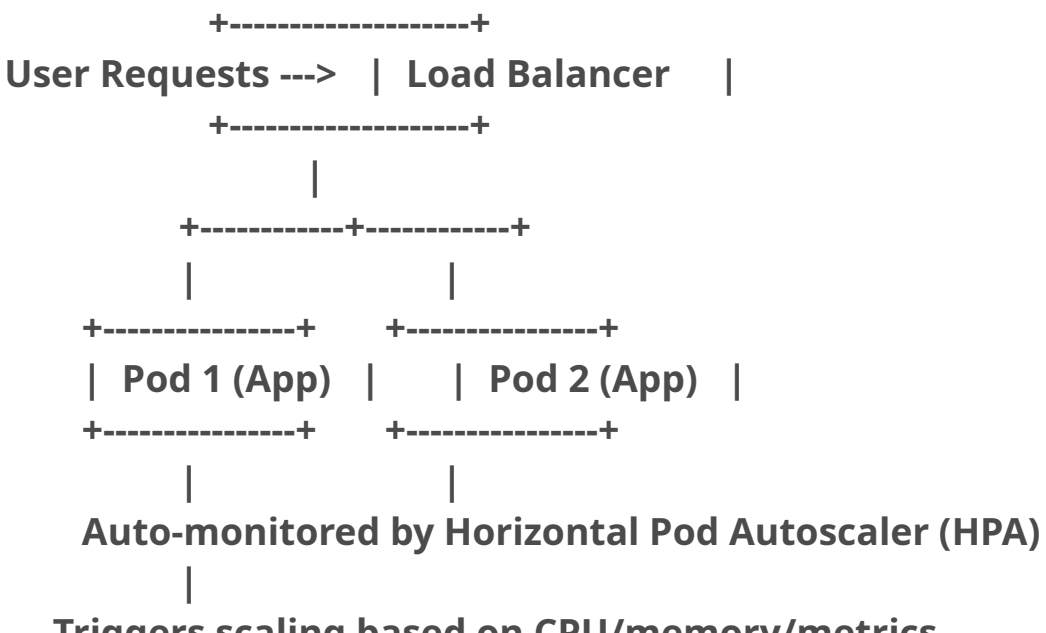
Definition

Horizontal Pod Autoscaling (HPA) automatically increases or decreases the number of pod replicas in a deployment based on observed CPU utilization or custom metrics.

Layman Explanation

Imagine a restaurant where more chefs are called in during rush hours and sent home when it’s quiet. Similarly, in Kubernetes, when your app gets more traffic (CPU usage goes high), HPA adds more pods (chefs). When traffic slows down, it removes extra pods, saving resources.

Diagrammatic Representation

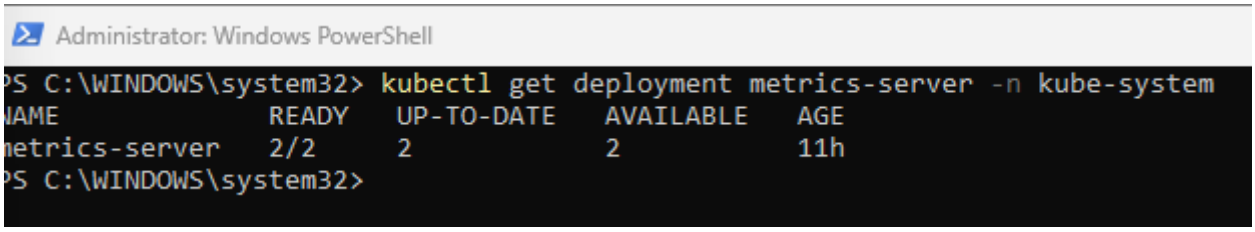


Steps to Implement in Your Deployment :

You must have Metrics Server running to use HPA.

To check:

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



Step 2: Expose Deployment with a Service (if not already)

MINE IS ALREADY DEPLOYED ON DIFFERENT DIFFERNT MEANSURES

```
NAME            TYPE           CLUSTER-IP     EXTERNAL-IP    PORT(S)        AGE
clusterip-svc   ClusterIP      10.0.51.13     <none>         80/TCP         6h20m
kubernetes      ClusterIP      10.0.0.1       <none>         443/TCP        11h
myapp-service   LoadBalancer   10.0.121.175   4.213.203.76   80:32143/TCP   11h
nodeport-svc    NodePort       10.0.100.208   <none>         80:32000/TCP   6h18m
PS C:\WINDOWS\system32>
```

Step 3: Apply Horizontal Pod Autoscaler

**kubectl autoscale deployment myapp-deploy **
**--cpu-percent=50 **
**--min=2 **
--max=5

Horizontal Pod Autoscaler (HPA), it doesn't depend on the *type of Service (LoadBalancer, NodePort, etc.)*.

HPA works directly on the Deployment and needs the metrics-server to be active.

Since deployment is already running and service (`myapp-service`) is exposed, we are ready to apply HPA.

```
PS C:\WINDOWS\system32> kubectl autoscale deployment myapp-deploy --cpu-percent=50 --min=2 --max=5
>>
horizontalpodautoscaler.autoscaling/myapp-deploy autoscaled
PS C:\WINDOWS\system32>
```

This command will:

- **Create an HPA on your `myapp-deploy` deployment**
- **Scale pods between 2 to 5 based on 50% CPU usage threshold**

To Check HPA Status

`kubectl get hpa`

```
PS C:\WINDOWS\system32> kubectl get hpa
>>
NAME           REFERENCE                 TARGETS              MINPODS   MAXPODS   REPLICAS   AGE
myapp-deploy   Deployment/myapp-deploy   cpu: <unknown>/50%   2         5         2          58s
PS C:\WINDOWS\system32>
```

What's the likely reason? For showing unknown

Your application (container from `prateek2004/my-frontend`) is not generating enough CPU load to be measured — which is normal for idle or light web apps.

Now lets verify HPA actually works (by simulating load):

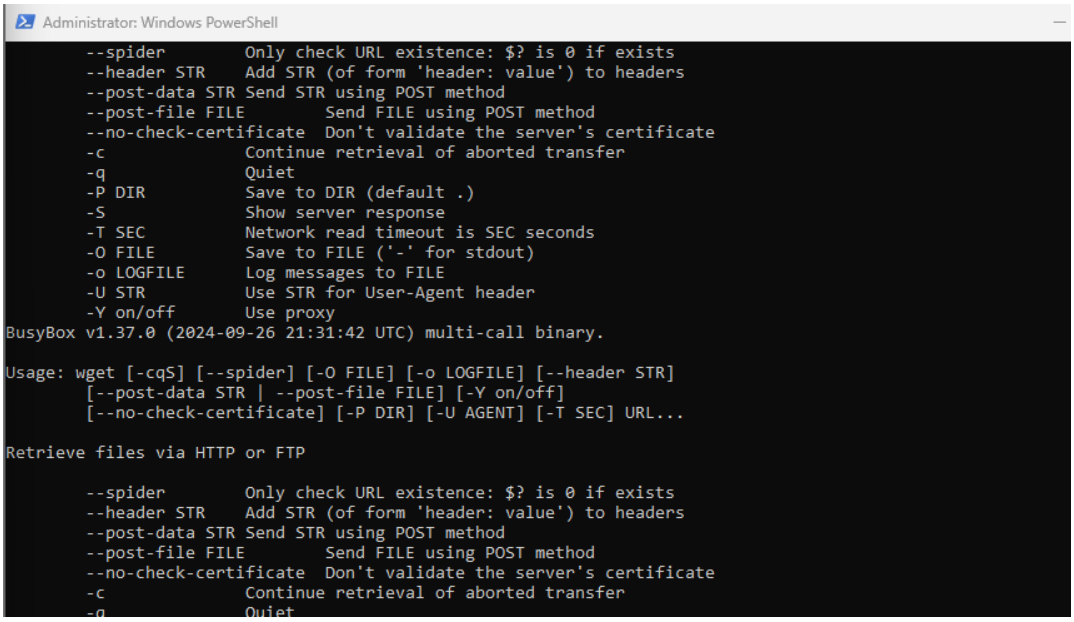
Step-by-Step Load Testing:

Step 1: Create a temporary pod to generate traffic:

Step 2: Inside busybox, simulate CPU usage by continuously hitting your app:

while true; do wget -q -O- http://myapp-service; done

it will start running the constant requests to the server :

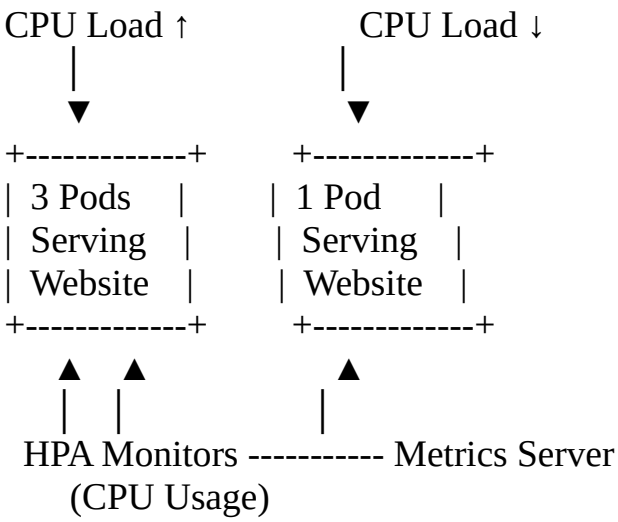


Watch HPA Again

we should now see something like:

```
myapp-deploy   Deployment/myapp-deploy   cpu: 230m/50% 2
5   3
```

Diagrammatic Representation:



These are the things we have done in this assignment:

Step	Description	Status
1	Deployed your app (`myapp-deploy`)	✅

Step	Description	Status
2	Created a LoadBalancer service for access	✅
3	Installed **metrics-server** properly in AKS	✅
4	Exposed CPU requests in your pod spec (`100m`)	✅
5	Simulated CPU load using `yes > /dev/null`	✅
6	Verified `kubectl top pods` shows CPU usage	✅
7	Created an HPA object using `kubectl autoscale`	✅
8	`kubectl get hpa` eventually showed real CPU values	✅
9	HPA scaled based on load (2 → 3 pods)	✅

Horizontal Pod Autoscaler (HPA) has been successfully configured and verified in my AKS cluster. The setup includes a running metrics-server, resource-based pod scaling, and real-time CPU monitoring, ensuring that my application automatically scales based on demand.

Real-World Use Case of Horizontal Pod Autoscaler (HPA):

Imagine an **e-commerce website** like Flipkart or Amazon. During **normal hours**, only a few people browse and place orders — the traffic is low, so **2 pods** are enough to serve users.

But during a **flash sale or festival season**, **suddenly thousands of users** start browsing at once. The **CPU usage increases rapidly**, and your existing pods can't handle the load.

With **HPA enabled**, Kubernetes detects high CPU usage and **automatically creates more pods (like 5 or 10)** to share the load. When traffic reduces, it **scales back down to 2 pods**, saving resources and cost.

🔁 **In short:**

> HPA = Auto backup plan for load.
> No need to wake up at midnight to scale manually.
> Used by real-world apps in finance, e-commerce, OTT, EdTech — anywhere with **variable user traffic**.