# Kubernetes Scaling Strategy

Contents

[Kubernetes Scaling Strategy 1](#_Toc120451189)

[Ways of scaling AKS clusters: 1](#_Toc120451190)

[Manually scale pods or nodes 1](#_Toc120451191)

[Horizontal Pod Autoscaler (HPA) 1](#_Toc120451192)

[Cluster autoscaler 1](#_Toc120451193)

## Ways of scaling AKS clusters:

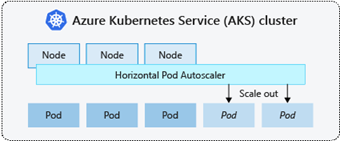
* manually scale pods or nodes
* horizontal pod autoscaler: it automatically scales the number of Pods
* cluster autoscaler: it automatically changes the number of Nodes in a cluster

Manually scale pods or nodes

Manually scaling resources also lets you define a set amount of resources to use to maintain a fixed cost, such as the number of nodes. To manually scale, you define the replica or node count. The Kubernetes API then schedules creating additional pods or draining nodes based on that replica or node count.  
To get started with manually scaling pods and nodes see [Scale applications in AKS](https://docs.microsoft.com/en-us/azure/aks/tutorial-kubernetes-scale)..

### Horizontal Pod Autoscaler (HPA)

The Horizontal Pod Autoscaler uses the Metrics Server in a Kubernetes cluster to monitor the resource need of pods and scales the number of pods automatically.  
The HPA checks the Metrics API every 30 seconds by default for any required changes in replica count. When it is needed, the number of replicas is increased or decreased accordingly.



The Horizontal Pod Autoscaler is implemented as a Kubernetes API resource and a controller. The resource determines the behavior of the controller. The controller periodically adjusts the number of replicas in a replication controller or deployment to match the observed metrics such as average CPU utilization, average memory utilization or any other custom metric to the target specified by the user.  
Horizontal pod autoscaler works with AKS clusters that have deployed the Metrics Server for Kubernetes 1.8+.

#### Examples for HPA creation

The command above creates a HPA that maintains between 1 and 10 replicas of Pods, which controlled by the test-deployment.  
HPA will increase and decrease the number of replicas to maintain an average CPU utilization across all Pods of 50%. Since each pod requests 200 milli-cores by kubectl run, this means an average CPU usage of 100 milli-cores. For more details on the algorithm click [here](https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/#algorithm-details).

**kubectl autoscale deployment test-deployment --cpu-percent=50 --min=1 --max=10**

Checking the status of the autoscaler:  
**kubectl get hpa**

It is possible to deploy the HPA as a yaml file or to get the yaml file of an existing HPA.

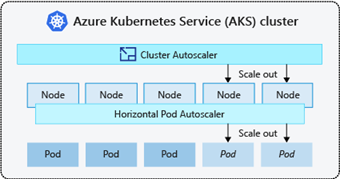
For more details please check [https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale-walkthrough/#create-horizontal-pod-autoscaler](https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale-walkthrough/#create-horizontal-pod-autoscaler+)

#### Cooldown of scaling events

Previous scale events may not have successfully completed before another check is made, as the HPA checks the Metrics API every 30 seconds. This could cause the horizontal pod autoscaler to change the number of replicas before the previous scale event could receive application workload and the resource demands to adjust accordingly.  
A delay value defines how long the HPA must wait between scale events. This allows the new replica count to take effect.  
**Note**: If the delay (cooldown) value is set too long, there could be complaints that the Horizontal Pod Autoscaler is not responsive to workload changes. However, if the delay value is set too short, the scale of the replicas set may keep thrashing as usual.

### Cluster autoscaler

Kubernetes has a cluster autoscaler, that adjusts the number of nodes based on the requested compute resources in the node pool (watches for pods that can't be scheduled on nodes because of resource constraints). The cluster autoscaler checks the Metrics API server every 10 seconds for any required changes in node count. The number of nodes in AKS cluster is increased or decreased if the cluster autoscale determines that it is required.



Cluster autoscaler is typically used with the horizontal pod autoscaler. With that way, the horizontal pod autoscaler increases or decreases the number of pods based on the application needs, and the cluster autoscaler adjusts the number of nodes as needed to run those additional pods.  
The cluster autoscaler works with Kubernetes RBAC-enabled AKS clusters that run Kubernetes 1.10.x or higher.

#### Enable the cluster autoscaler

The following command enables the cluseter autoscaler/

* --enable-cluster-autoscaler: To enable and configure the cluster autoscaler on the node pool for the cluster,