Tutorial: Scale applications in Azure Kubernetes Service (AKS)

Article • 03/31/2022 • 5 minutes to read • 18 contributors

In this article

Before you begin
Manually scale pods
Autoscale pods
Manually scale AKS nodes
Next steps

If you've followed the tutorials, you have a working Kubernetes cluster in AKS and you deployed the sample Azure Voting app. In this tutorial, part five of seven, you scale out the pods in the app and try pod autoscaling. You also learn how to scale the number of Azure VM nodes to change the cluster's capacity for hosting workloads. You learn how to:

- ✓ Scale the Kubernetes nodes
- ✓ Manually scale Kubernetes pods that run your application
- ✓ Configure autoscaling pods that run the app front-end

In later tutorials, the Azure Vote application is updated to a new version.

Before you begin

In previous tutorials, an application was packaged into a container image. This image was uploaded to Azure Container Registry, and you created an AKS cluster. The application was then deployed to the AKS cluster. If you haven't done these steps, and would like to follow along, start with Tutorial 1 – Create container images.

Azure CLI | Azure PowerShell

This tutorial requires that you're running the Azure CLI version 2.0.53 or later. Run `az --version` to find the version. If you need to install or upgrade, see Install Azure CLI.

Manually scale pods

When the Azure Vote front-end and Redis instance were deployed in previous tutorials, a single replica was created. To see the number and state of pods in your cluster, use the kubectl get command as follows:

Console Copy

```
kubectl get pods
```

The following example output shows one front-end pod and one back-end pod:

Output Copy

```
NAME                                READY     STATUS    RESTARTS   AGE
azure-vote-back-2549686872-4d2r5    1/1       Running   0          31m
azure-vote-front-848767080-tf34m    1/1       Running   0          31m
```

To manually change the number of pods in the *azure-vote-front* deployment, use the kubectl scale command. The following example increases the number of front-end pods to *5*:

Console Copy

```
kubectl scale --replicas=5 deployment/azure-vote-front
```

Run kubectl get pods again to verify that AKS successfully creates the additional pods. After a minute or so, the pods are available in your cluster:

Console Copy

```
kubectl get pods

                                    READY     STATUS    RESTARTS   AGE
azure-vote-back-2606967446-nmpcf    1/1       Running   0          15m
azure-vote-front-3309479140-2hfh0   1/1       Running   0          3m
azure-vote-front-3309479140-bzt05   1/1       Running   0          3m
azure-vote-front-3309479140-fvcvm   1/1       Running   0          3m
azure-vote-front-3309479140-hrbf2   1/1       Running   0          15m
azure-vote-front-3309479140-qphz8   1/1       Running   0          3m
```

Autoscale pods

Azure CLI

Azure PowerShell

Kubernetes supports horizontal pod autoscaling to adjust the number of pods in a deployment depending on CPU utilization or other select metrics. The Metrics Server is used to provide resource utilization to Kubernetes, and is automatically deployed in AKS clusters versions 1.10 and higher. To see the version of your AKS cluster, use the az aks show command, as shown in the following example:

Azure CLI Copy

```
az aks show --resource-group myResourceGroup --name myAKSCluster --query
kubernetesVersion --output table
```

Note

If your AKS cluster is less than *1.10*, the Metrics Server is not automatically installed. Metrics Server installation manifests are available as a `components.yaml` asset on Metrics Server releases, which means you can install them via a url. To learn more about these YAML definitions, see the **Deployment** section of the readme.

Example installation:

Console Copy

```
kubectl apply -f https://github.com/kubernetes-sigs/metrics-
server/releases/download/v0.3.6/components.yaml
```

To use the autoscaler, all containers in your pods and your pods must have CPU requests and limits defined. In the `azure-vote-front` deployment, the front-end container already requests 0.25 CPU, with a limit of 0.5 CPU.

These resource requests and limits are defined for each container as shown in the following example snippet:

YAML Copy

```
containers:
- name: azure-vote-front
  image: mcr.microsoft.com/azuredocs/azure-vote-front:v1
  ports:
  - containerPort: 80
  resources:
    requests:
      cpu: 250m
    limits:
      cpu: 500m
```

The following example uses the kubectl autoscale command to autoscale the number of pods in the *azure-vote-front* deployment. If average CPU utilization across all pods exceeds 50% of their requested usage, the autoscaler increases the pods up to a maximum of *10* instances. A minimum of *3* instances is then defined for the deployment:

Console

```console
kubectl autoscale deployment azure-vote-front --cpu-percent=50 --min=3 --max=10
```

Alternatively, you can create a manifest file to define the autoscaler behavior and resource limits. The following is an example of a manifest file named `azure-vote-hpa.yaml`.

YAML

```yaml
apiVersion: autoscaling/v1
kind: HorizontalPodAutoscaler
metadata:
  name: azure-vote-back-hpa
spec:
  maxReplicas: 10 # define max replica count
  minReplicas: 3  # define min replica count
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: azure-vote-back
  targetCPUUtilizationPercentage: 50 # target CPU utilization

---

apiVersion: autoscaling/v1
kind: HorizontalPodAutoscaler
metadata:
  name: azure-vote-front-hpa
spec:
  maxReplicas: 10 # define max replica count
  minReplicas: 3  # define min replica count
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: azure-vote-front
  targetCPUUtilizationPercentage: 50 # target CPU utilization
```

Use `kubectl apply` to apply the autoscaler defined in the `azure-vote-hpa.yaml` manifest file.

Console Copy

```
kubectl apply -f azure-vote-hpa.yaml
```

To see the status of the autoscaler, use the `kubectl get hpa` command as follows:

Copy

```
kubectl get hpa

NAME               REFERENCE                     TARGETS    MINPODS
MAXPODS   REPLICAS   AGE
azure-vote-front   Deployment/azure-vote-front   0% / 50%   3         10
3          2m
```

After a few minutes, with minimal load on the Azure Vote app, the number of pod replicas decreases automatically to three. You can use `kubectl get pods` again to see the unneeded pods being removed.

Note

For additional examples on using the horizontal pod autoscaler, see **HorizontalPodAutoscaler Walkthrough** .

Manually scale AKS nodes

If you created your Kubernetes cluster using the commands in the previous tutorial, it has two nodes. You can adjust the number of nodes manually if you plan more or fewer container workloads on your cluster.

The following example increases the number of nodes to three in the Kubernetes cluster named *myAKSCluster*. The command takes a couple of minutes to complete.

Azure CLI Azure PowerShell

Azure CLI Copy

```
az aks scale --resource-group myResourceGroup --name myAKSCluster --
node-count 3
```

When the cluster has successfully scaled, the output is similar to following example:

Output

```
"agentPoolProfiles": [
  {
    "count": 3,
    "dnsPrefix": null,
    "fqdn": null,
    "name": "myAKSCluster",
    "osDiskSizeGb": null,
    "osType": "Linux",
    "ports": null,
    "storageProfile": "ManagedDisks",
    "vmSize": "Standard_D2_v2",
    "vnetSubnetId": null
  }
```

Next steps

In this tutorial, you used different scaling features in your Kubernetes cluster. You learned how to:

- ✓ Manually scale Kubernetes pods that run your application
- ✓ Configure autoscaling pods that run the app front-end
- ✓ Manually scale the Kubernetes nodes

Advance to the next tutorial to learn how to update application in Kubernetes.

Update an application in Kubernetes

Recommended content

Kubernetes on Azure tutorial - Upgrade a cluster - Azure Kubernetes Service

In this Azure Kubernetes Service (AKS) tutorial, you learn how to upgrade an existing AKS cluster to the latest available Kubernetes version.

Kubernetes on Azure tutorial - Update an application - Azure Kubernetes Service

In this Azure Kubernetes Service (AKS) tutorial, you learn how to update an existing application deployment to AKS with a new version of the application code.

Kubernetes on Azure tutorial - Deploy an application - Azure Kubernetes Service

In this Azure Kubernetes Service (AKS) tutorial, you deploy a multi-container application to your cluster using a custom image stored in Azure Container Registry.

Kubernetes on Azure tutorial - Create a container registry - Azure Kubernetes Service

In this Azure Kubernetes Service (AKS) tutorial, you create an Azure Container Registry instance and upload a sample application container image.

Concepts - Scale applications in Azure Kubernetes Services (AKS) - Azure Kubernetes Service

Learn about scaling in Azure Kubernetes Service (AKS), including horizontal pod autoscaler, cluster autoscaler, and the Azure Container Instances connector.

Kubernetes on Azure tutorial - Deploy a cluster - Azure Kubernetes Service

In this Azure Kubernetes Service (AKS) tutorial, you create an AKS cluster and use kubectl to connect to the Kubernetes master node.

Kubernetes on Azure tutorial - Prepare an application - Azure Kubernetes Service

In this Azure Kubernetes Service (AKS) tutorial, you learn how to prepare and build a multi-container app with Docker Compose that you can then deploy to AKS.

Concepts - Storage in Azure Kubernetes Services (AKS) - Azure Kubernetes Service

Learn about storage in Azure Kubernetes Service (AKS), including volumes, persistent volumes, storage classes, and claims

Show more