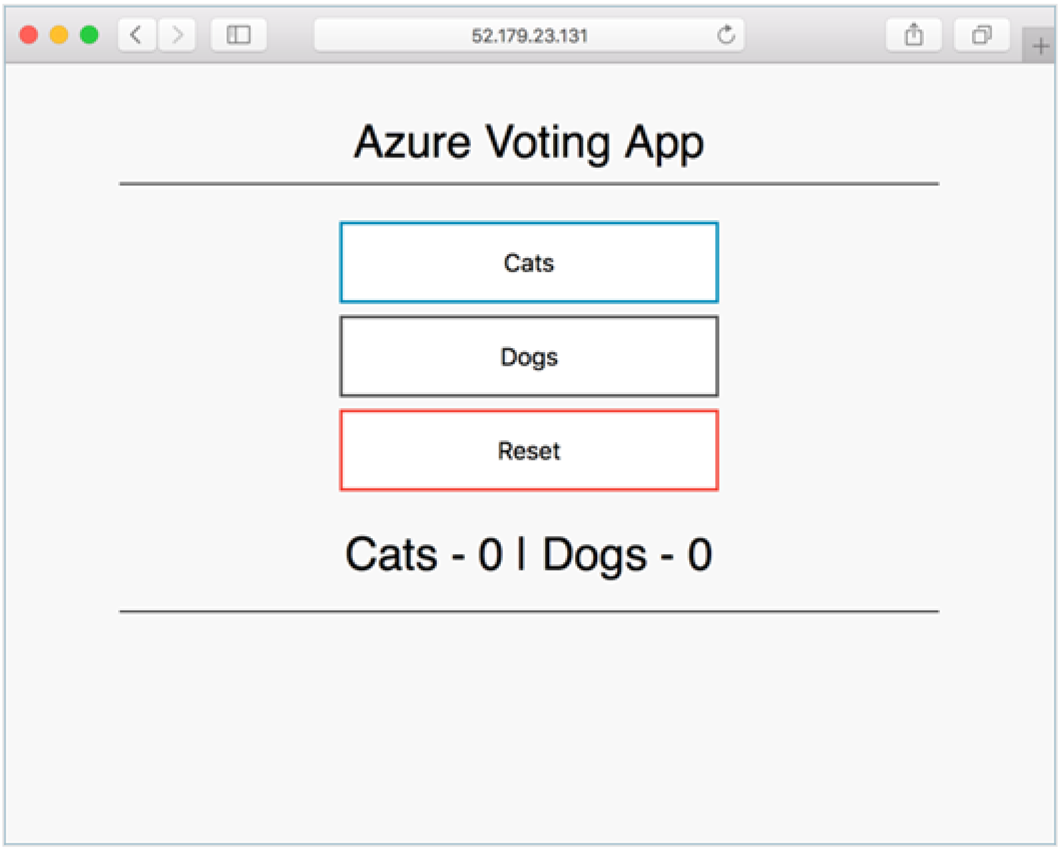
Quickstart: Deploy an Azure Kubernetes Service (AKS) cluster using the Azure portal

Article • 06/17/2022 • 6 minutes to read • 3 contributors

In this article

Azure Kubernetes Service (AKS) is a managed Kubernetes service that lets you quickly deploy and manage clusters. In this quickstart, you will:

- Deploy an AKS cluster using the Azure portal.
- Run a sample multi-container application with a web front-end and a Redis instance in the cluster.



This quickstart assumes a basic understanding of Kubernetes concepts. For more information, see Kubernetes core concepts for Azure Kubernetes Service (AKS).

Prerequisites

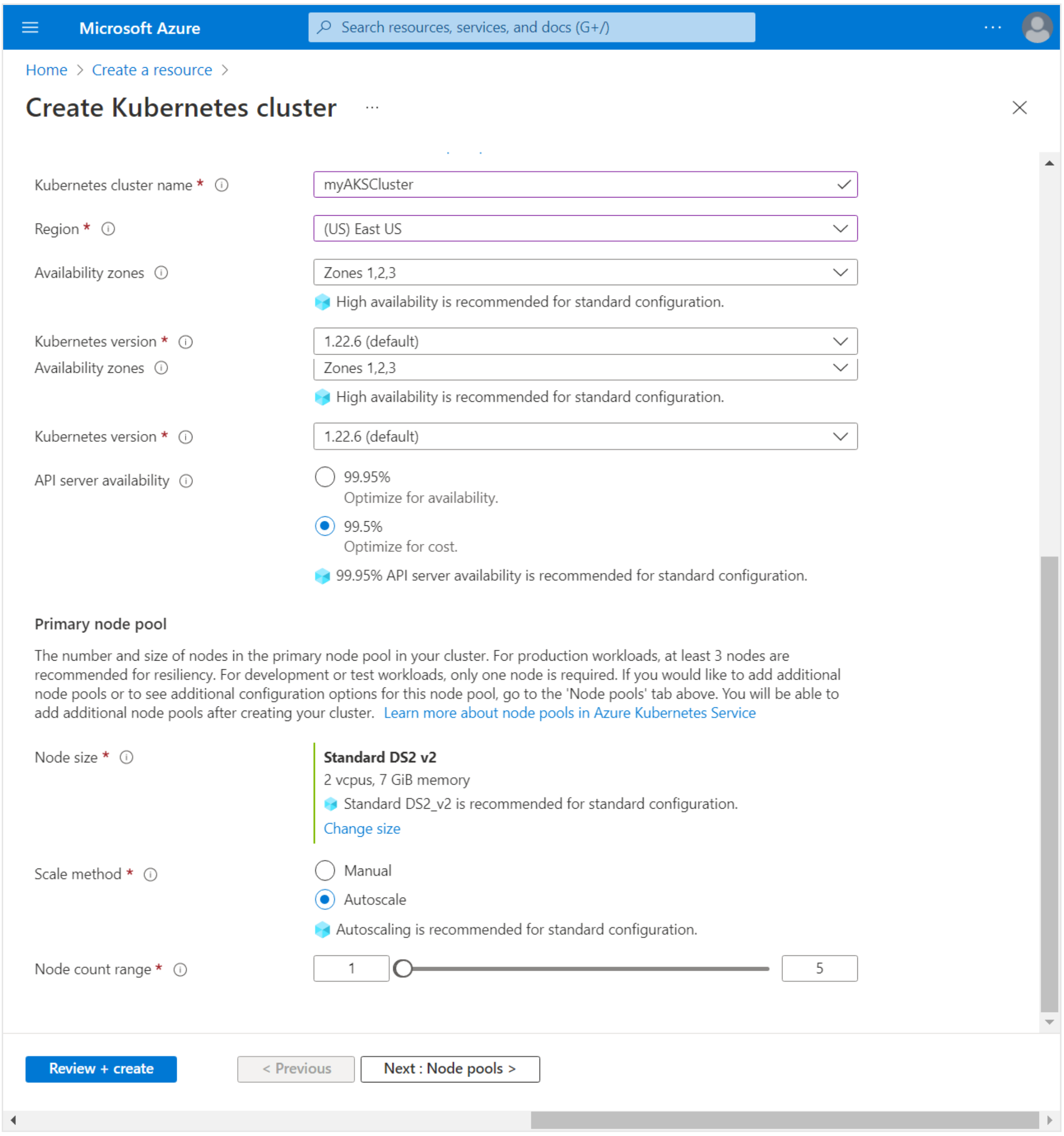
If you don't have an Azure subscription, create an Azure free account before you begin.

- If you're unfamiliar with the Azure Cloud Shell, review Overview of Azure Cloud Shell.
- The identity you're using to create your cluster has the appropriate minimum permissions. For more details on access and identity for AKS, see Access and identity options for Azure Kubernetes Service (AKS).

Create an AKS cluster

1. Sign in to the Azure portal .
2. On the Azure portal menu or from the **Home** page, select **Create a resource**.
3. Select **Containers** > **Kubernetes Service**.
4. On the **Basics** page, configure the following options:

- **Project details**:
  - Select an Azure **Subscription**.
  - Select or create an Azure **Resource group**, such as *myResourceGroup*.
- **Cluster details**:
  - Ensure the the **Preset configuration** is *Standard ($$)*. For more details on preset configurations, see Cluster configuration presets in the Azure portal.
  - Enter a **Kubernetes cluster name**, such as *myAKSCluster*.
  - Select a **Region** for the AKS cluster, and leave the default value selected for **Kubernetes version**.
  - Select **99.5%** for **API server availability**.
- **Primary node pool**:
  - Leave the default values selected.


Microsoft Azure
Search resources, services, and docs (G+/)
Home > Create a resource >
Create Kubernetes cluster
Review + create
< Previous
Next : Node pools >

Kubernetes cluster name *	myAKSCluster
Region *	(US) East US
Availability zones	Zones 1,2,3 High availability is recommended for standard configuration.
Kubernetes version *	1.22.6 (default)
Availability zones	Zones 1,2,3 High availability is recommended for standard configuration.
Kubernetes version *	1.22.6 (default)
API server availability	99.95% Optimize for availability. [✓ 99.5% Optimize for cost.] 99.95% API server availability is recommended for standard configuration.

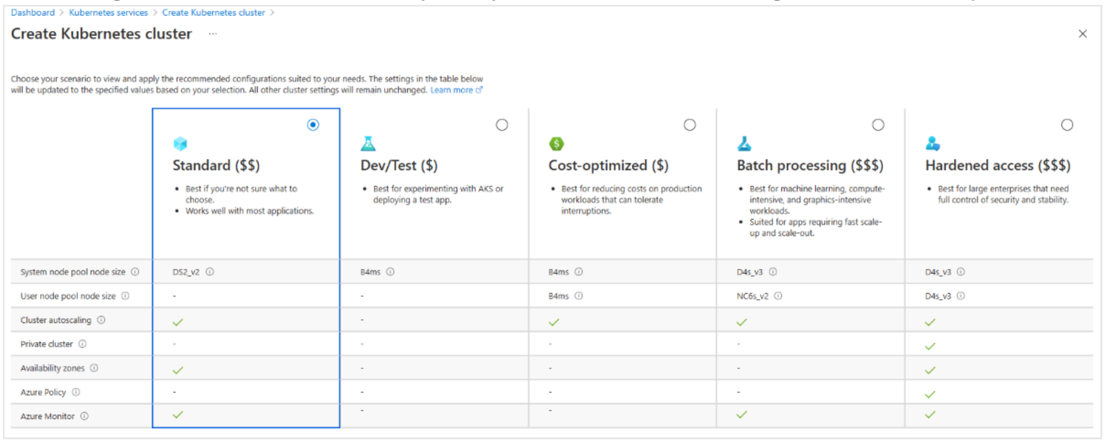
Primary node pool

The number and size of nodes in the primary node pool in your cluster. For production workloads, at least 3 nodes are recommended for resiliency. For development or test workloads, only one node is required. If you would like to add additional node pools or to see additional configuration options for this node pool, go to the 'Node pools' tab above. You will be able to add additional node pools after creating your cluster. Learn more about node pools in Azure Kubernetes Service

Node size *	**Standard DS2 v2** 2 vcpus, 7 GiB memory Standard DS2_v2 is recommended for standard configuration. Change size
Scale method *	Manual [✓ Autoscale] Autoscaling is recommended for standard configuration.
Node count range *	1 – 5

ⓘ Note

You can change the preset configuration when creating your cluster by selecting *Learn more and compare presets* and choosing a different option.



Dashboard > Kubernetes services > Create Kubernetes cluster >

Create Kubernetes cluster

Choose your scenario to view and apply the recommended configurations suited to your needs. The settings in the table below will be updated to the specified values based on your selection. All other cluster settings will remain unchanged. Learn more

	[✓ Standard ($$)]	Dev/Test ($)	Cost-optimized ($)	Batch processing ($$$)	Hardened access ($$$)
	• Best if you're not sure what to choose. • Works well with most applications.	• Best for experimenting with AKS or deploying a test app.	• Best for reducing costs on production workloads that can tolerate interruptions.	• Best for machine learning, compute-intensive, and graphics-intensive workloads. • Suited for apps requiring fast scale-up and scale-out.	• Best for large enterprises that need full control of security and stability.
System node pool node size	DS2_v2	B4ms	B4ms	D4s_v3	D4s_v3
User node pool node size	-	-	B4ms	NC6s_v2	D4s_v3
Cluster autoscaling	✓	-	✓	✓	✓
Private cluster	-	-	-	-	✓
Availability zones	✓	-	-	-	✓
Azure Policy	-	-	-	-	✓
Azure Monitor	✓	-	-	✓	✓

5. Select **Next: Node pools** when complete.

6. Keep the default **Node pools** options. At the bottom of the screen, click **Next: Access**.

7. On the **Access** page, configure the following options:

   - The default value for **Resource identity** is **System-assigned managed identity**. Managed identities provide an identity for applications to use when connecting to resources that support Azure Active Directory (Azure AD) authentication. For more details about managed identities, see What are managed identities for Azure resources?.
   - The Kubernetes role-based access control (RBAC) option is the default value to provide more fine-grained control over access to the Kubernetes resources deployed in your AKS cluster.

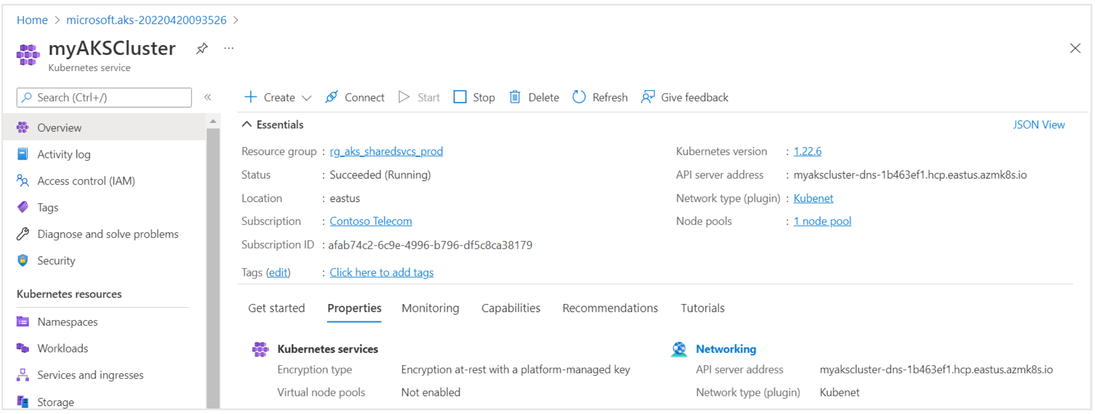
   By default, *Basic* networking is used, and Container insights is enabled.

8. Click **Review + create**. When you navigate to the **Review + create** tab, Azure runs validation on the settings that you have chosen. If validation passes, you can proceed to create the AKS cluster by selecting **Create**. If validation fails, then it indicates which settings need to be modified.

9. It takes a few minutes to create the AKS cluster. When your deployment is complete, navigate to your resource by either:

   - Selecting **Go to resource**, or

   - Browsing to the AKS cluster resource group and selecting the AKS resource. In this example you browse for *myResourceGroup* and select the resource *myAKSCluster*.



Home > microsoft.aks-20220420093526 >

myAKSCluster

Kubernetes service

Resource group	: rg_aks_sharedsvcs_prod
Status	: Succeeded (Running)
Location	: eastus
Subscription	: Contoso Telecom
Subscription ID	: afab74c2-6c9e-4996-b796-df5c8ca38179
Tags (edit)	: Click here to add tags
Kubernetes version	: 1.22.6
API server address	: myakscluster-dns-1b463ef1.hcp.eastus.azmk8s.io
Network type (plugin)	: Kubenet
Node pools	: 1 node pool

Kubernetes services

Encryption type	Encryption at-rest with a platform-managed key
Virtual node pools	Not enabled

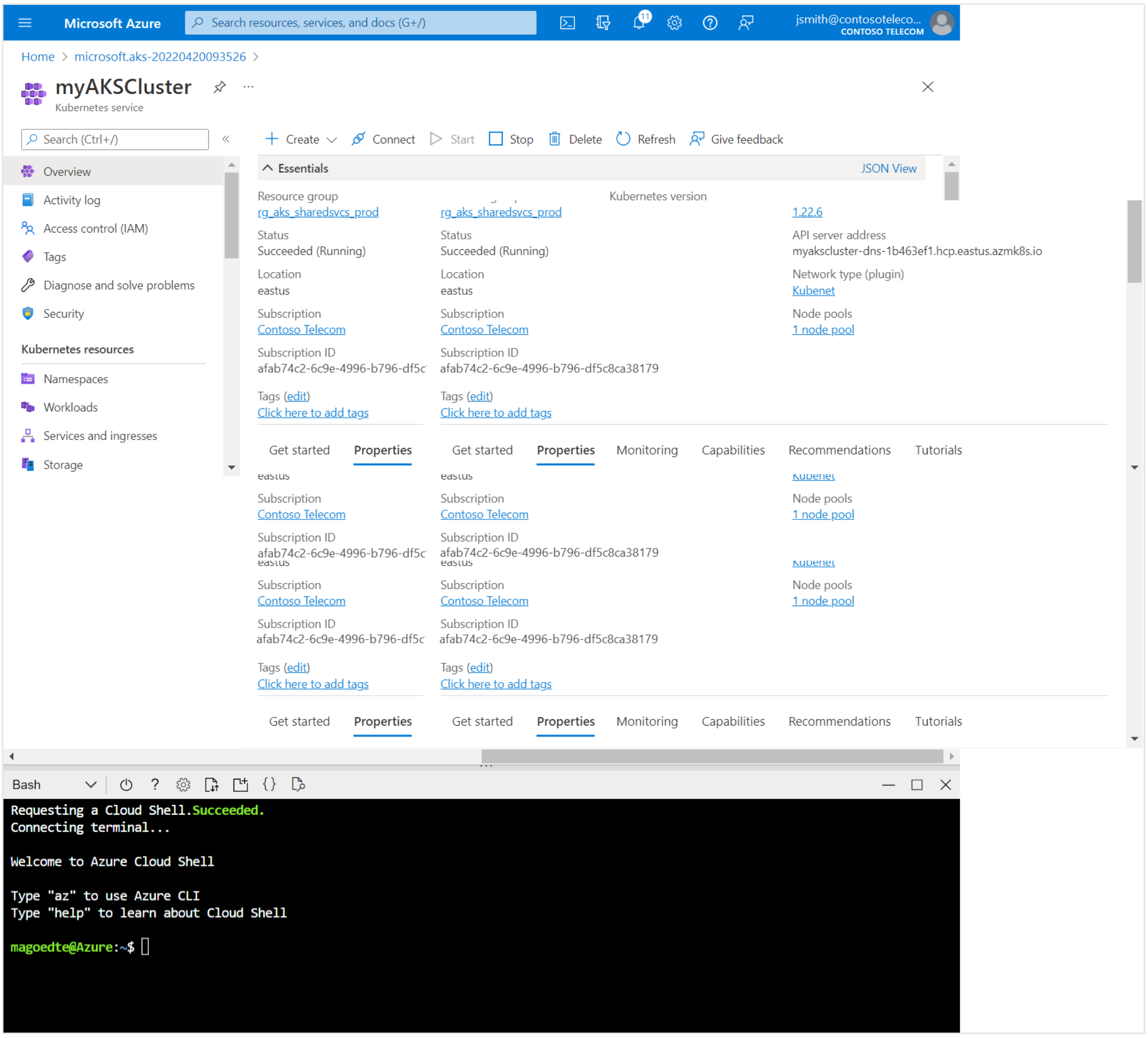
Networking

API server address	myakscluster-dns-1b463ef1.hcp.eastus.azmk8s.io
Network type (plugin)	Kubenet

Connect to the cluster

To manage a Kubernetes cluster, use the Kubernetes command-line client, kubectl . `kubectl` is already installed if you use Azure Cloud Shell. If you're unfamiliar with the Cloud Shell, review Overview of Azure Cloud Shell.

1. Open Cloud Shell using the `>_` button on the top of the Azure portal.



```
Requesting a Cloud Shell.Succeeded.
Connecting terminal...

Welcome to Azure Cloud Shell

Type "az" to use Azure CLI
Type "help" to learn about Cloud Shell

magoedte@Azure:~$ 
```

ⓘ **Note**

To perform these operations in a local shell installation:

a. Verify Azure CLI or Azure PowerShell is installed.

b. Connect to Azure via the `az login` or `Connect-AzAccount` command.

Azure CLI | Azure PowerShell

2. Configure `kubectl` to connect to your Kubernetes cluster using the az aks get-credentials command. The following command downloads credentials and configures the Kubernetes CLI to use them.

Azure CLI — Copy — Try It

```
az aks get-credentials --resource-group myResourceGroup --name
myAKSCluster
```

3. Verify the connection to your cluster using `kubectl get` to return a list of the cluster nodes.

Console — Copy

```
kubectl get nodes
```

Output shows the single node created in the previous steps. Make sure the node status is *Ready*:

Output — Copy

```
NAME                                 STATUS   ROLES   AGE   VERSION
aks-agentpool-12345678-vmss000000   Ready    agent   23m   v1.19.11
aks-agentpool-12345678-vmss000001   Ready    agent   24m   v1.19.11
```

Deploy the application

A Kubernetes manifest file defines a cluster's desired state, like which container images to run.

In this quickstart, you will use a manifest to create all objects needed to run the Azure Vote application. This manifest includes two Kubernetes deployments:

- The sample Azure Vote Python applications.

- A Redis instance.

Two Kubernetes Services are also created:

- An internal service for the Redis instance.
- An external service to access the Azure Vote application from the internet.

1. In the Cloud Shell, use an editor to create a file named `azure-vote.yaml`, such as:

   - `code azure-vote.yaml`
   - `nano azure-vote.yaml`, or
   - `vi azure-vote.yaml`.

2. Copy in the following YAML definition:

YAML Copy

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: azure-vote-back
spec:
  replicas: 1
  selector:
    matchLabels:
      app: azure-vote-back
  template:
    metadata:
      labels:
        app: azure-vote-back
    spec:
      nodeSelector:
        "kubernetes.io/os": linux
      containers:
      - name: azure-vote-back
        image: mcr.microsoft.com/oss/bitnami/redis:6.0.8
        env:
        - name: ALLOW_EMPTY_PASSWORD
          value: "yes"
        resources:
          requests:
            cpu: 100m
            memory: 128Mi
          limits:
            cpu: 250m
            memory: 256Mi
        ports:
        - containerPort: 6379
          name: redis
---
apiVersion: v1
```

```
kind: Service
metadata:
  name: azure-vote-back
spec:
  ports:
  - port: 6379
  selector:
    app: azure-vote-back
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: azure-vote-front
spec:
  replicas: 1
  selector:
    matchLabels:
      app: azure-vote-front
  template:
    metadata:
      labels:
        app: azure-vote-front
    spec:
      nodeSelector:
        "kubernetes.io/os": linux
      containers:
      - name: azure-vote-front
        image: mcr.microsoft.com/azuredocs/azure-vote-front:v1
        resources:
          requests:
            cpu: 100m
            memory: 128Mi
          limits:
            cpu: 250m
            memory: 256Mi
        ports:
        - containerPort: 80
        env:
        - name: REDIS
          value: "azure-vote-back"
---
apiVersion: v1
kind: Service
metadata:
  name: azure-vote-front
spec:
  type: LoadBalancer
  ports:
  - port: 80
  selector:
    app: azure-vote-front
```

3. Deploy the application using the `kubectl apply` command and specify the name of your YAML manifest:

   Console Copy

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

   Output shows the successfully created deployments and services:

   Output Copy

   ```
   deployment "azure-vote-back" created
   service "azure-vote-back" created
   deployment "azure-vote-front" created
   service "azure-vote-front" created
   ```

Test the application

When the application runs, a Kubernetes service exposes the application front end to the internet. This process can take a few minutes to complete.

To monitor progress, use the `kubectl get service` command with the `--watch` argument.

Console Copy

```
kubectl get service azure-vote-front --watch
```

The **EXTERNAL-IP** output for the `azure-vote-front` service will initially show as *pending*.

Output Copy

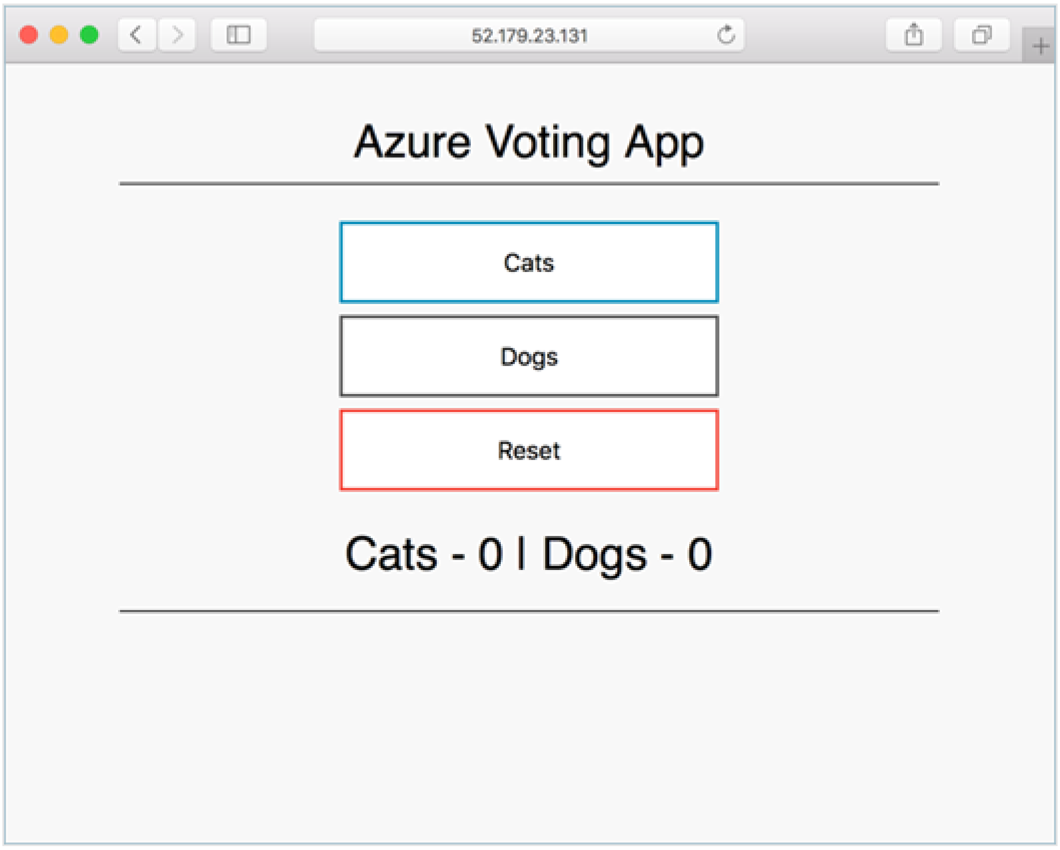
```
NAME               TYPE           CLUSTER-IP    EXTERNAL-IP   PORT(S)
AGE
azure-vote-front   LoadBalancer   10.0.37.27    <pending>     80:30572/TCP
6s
```

Once the **EXTERNAL-IP** address changes from *pending* to an actual public IP address, use `CTRL-C` to stop the `kubectl` watch process. The following example output shows a valid public IP address assigned to the service:

Output Copy

```
azure-vote-front   LoadBalancer   10.0.37.27   52.179.23.131   80:30572/TCP
2m
```

To see the Azure Vote app in action, open a web browser to the external IP address of your service.



Delete cluster

To avoid Azure charges, if you don't plan on going through the tutorials that follow, clean up your unnecessary resources. Select the **Delete** button on the AKS cluster dashboard. You can also use the az group delete command or the Remove-AzResourceGroup cmdlet to remove the resource group, container service, and all related resources.

Azure CLI | Azure PowerShell

Azure CLI

```
az group delete --name myResourceGroup --yes --no-wait
```

Note

The AKS cluster was created with a system-assigned managed identity. This identity is managed by the platform and doesn't require removal.

Next steps

In this quickstart, you deployed a Kubernetes cluster and then deployed a sample multi-container application to it.

To learn more about AKS by walking through a complete example, including building an application, deploying from Azure Container Registry, updating a running application, and scaling and upgrading your cluster, continue to the Kubernetes cluster tutorial.

AKS tutorial

Recommended content

Quickstart: Deploy an AKS cluster by using Azure CLI - Azure Kubernetes Service

Learn how to quickly create a Kubernetes cluster, deploy an application, and monitor performance in Azure Kubernetes Service (AKS) using the Azure CLI.

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.

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.

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.

Concepts - Kubernetes basics for Azure Kubernetes Services (AKS) - Azure Kubernetes Service

Learn the basic cluster and workload components of Kubernetes and how they relate to features in Azure Kubernetes Service (AKS)

Kubernetes on Azure tutorial - Scale Application - Azure Kubernetes Service

In this Azure Kubernetes Service (AKS) tutorial, you learn how to scale nodes and pods in Kubernetes, and implement horizontal pod autoscaling.

Show more