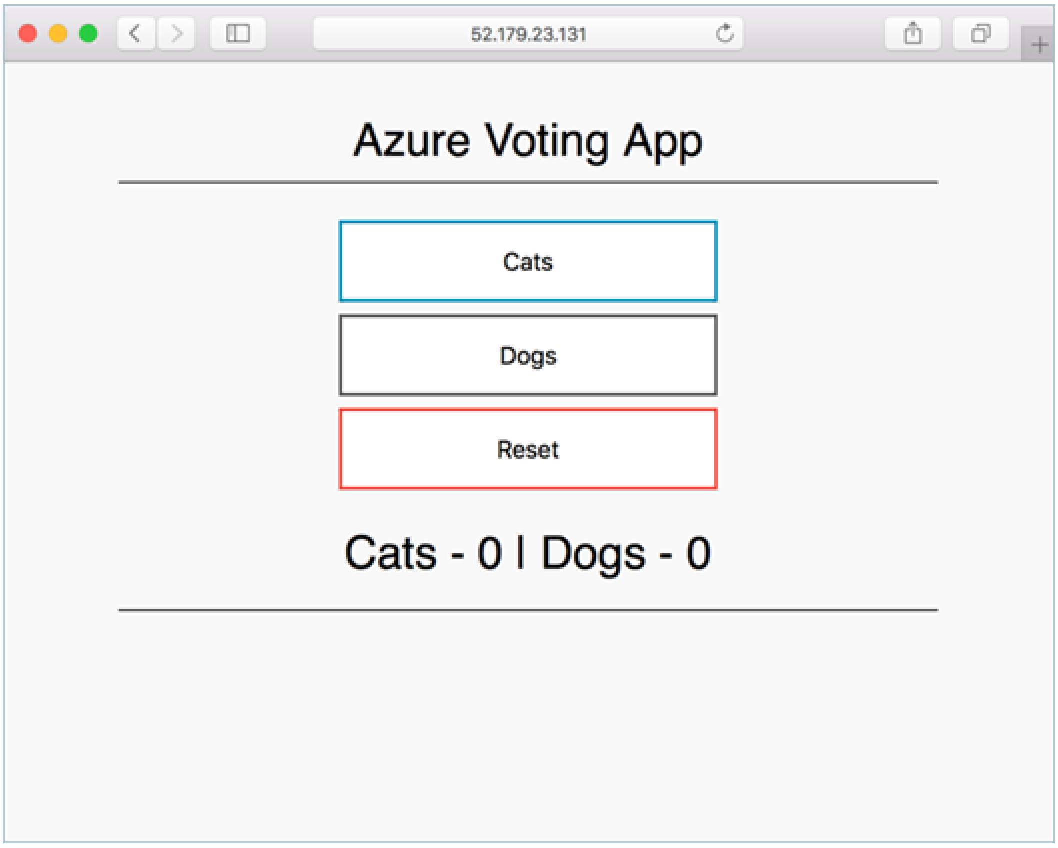
Quickstart: Deploy an Azure Kubernetes Service (AKS) cluster using the Azure portal

03/15/2021 • 6 minutes to read • +25

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 multi-container application with a web front-end and a Redis instance in the cluster.
- Monitor the health of the cluster and pods that run your application.



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

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

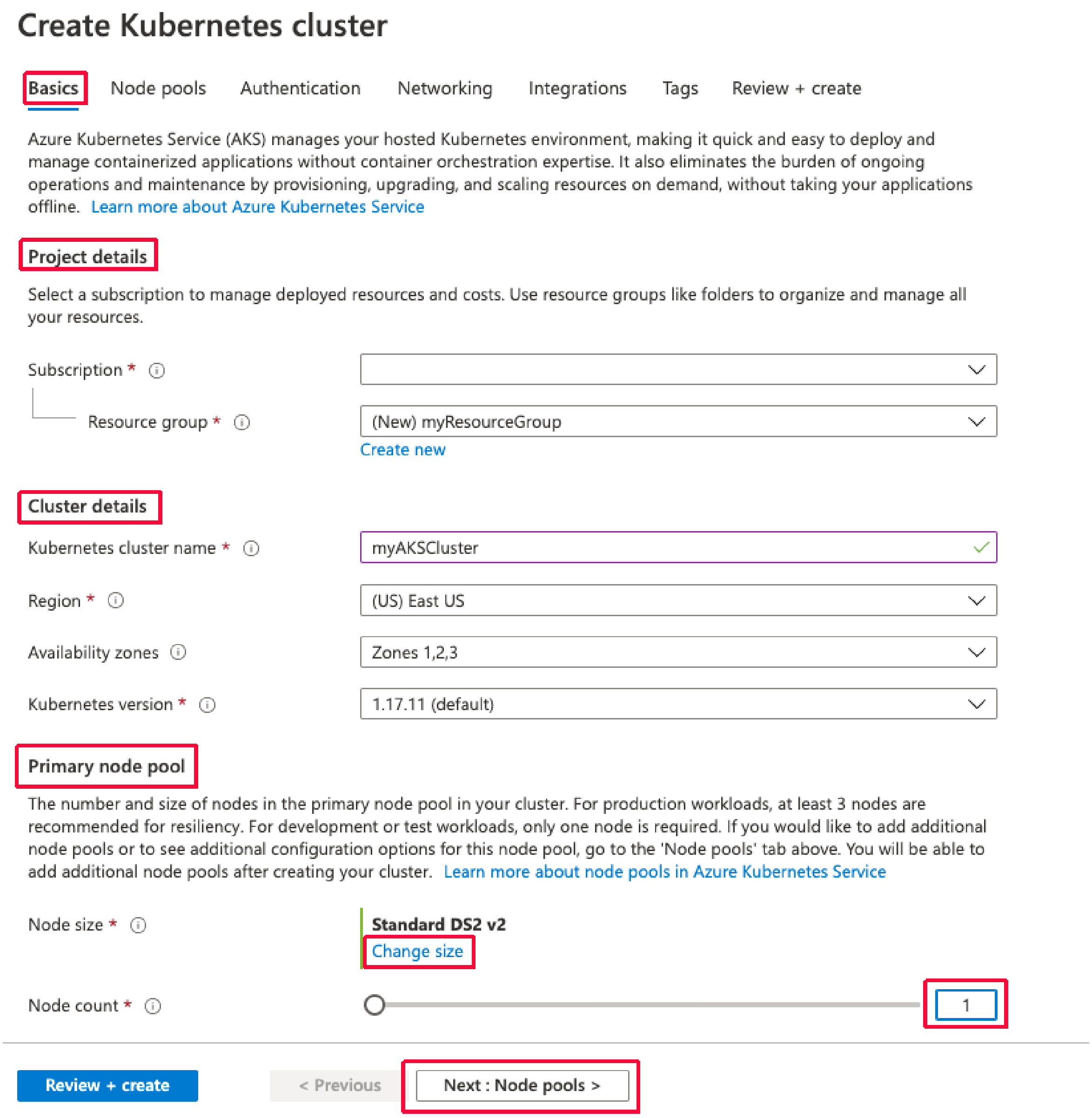
Prerequisites

Sign in to the Azure portal at https://portal.azure.com .

Create an AKS cluster

1. On the Azure portal menu or from the **Home** page, select **Create a resource**.
2. Select **Containers** > **Kubernetes Service**.
3. 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**:
     - Enter a **Kubernetes cluster name**, such as *myAKSCluster*.
     - Select a **Region** and **Kubernetes version** for the AKS cluster.
   - **Primary node pool**:

- Select a VM **Node size** for the AKS nodes. The VM size *cannot* be changed once an AKS cluster has been deployed.
- Select the number of nodes to deploy into the cluster. For this quickstart, set **Node count** to *1*. Node count *can* be adjusted after the cluster has been deployed.



Create Kubernetes cluster

Azure Kubernetes Service (AKS) manages your hosted Kubernetes environment, making it quick and easy to deploy and manage containerized applications without container orchestration expertise. It also eliminates the burden of ongoing operations and maintenance by provisioning, upgrading, and scaling resources on demand, without taking your applications offline. Learn more about Azure Kubernetes Service

Project details

Select a subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription *

Resource group * (New) myResourceGroup

Create new

Cluster details

Kubernetes cluster name * myAKSCluster

Region * (US) East US

Availability zones Zones 1,2,3

Kubernetes version * 1.17.11 (default)

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**

Change size

Node count * 1

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

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

> ⊗ Caution
>
> Newly created Azure AD service principals may take several minutes to propagate and become available, causing "service principal not found" errors and validation failures in Azure portal. If you hit this bump, please visit **our troubleshooting article** for mitigation.

6. On the **Authentication** page, configure the following options:

   - Create a new cluster identity by either:
     - Leaving the **Authentication** field with **System-assinged managed identity**, or
     - Choosing **Service Principal** to use a service principal.
       - Select *(new) default service principal* to create a default service principal, or
       - Select *Configure service principal* to use an existing one. You will need to provide the existing principal's SPN client ID and secret.
   - Enable the Kubernetes role-based access control (Kubernetes RBAC) option to provide more fine-grained control over access to the Kubernetes resources deployed in your AKS cluster.

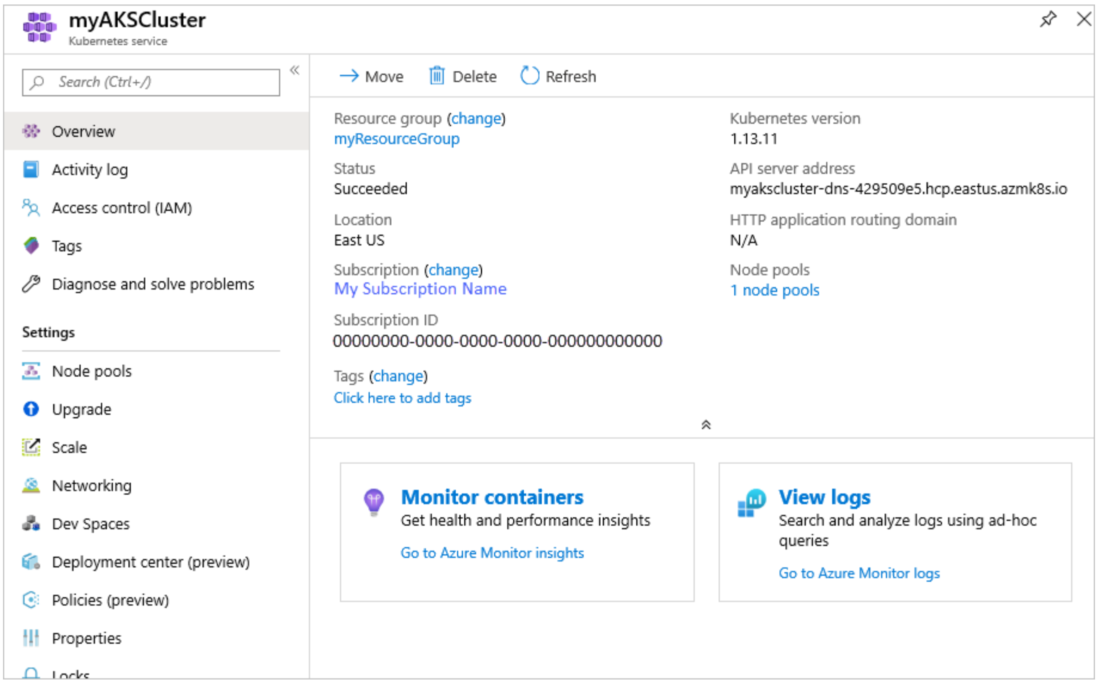
   By default, *Basic* networking is used, and Azure Monitor for containers is enabled.

7. Click **Review + create** and then **Create** when validation completes.

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

   - Clicking **Go to resource**, or

   - Browsing to the AKS cluster resource group and selecting the AKS resource.
     - Per example cluster dashboard below: browsing for *myResourceGroup* and selecting *myAKSCluster* resource.



Resource group (change)
myResourceGroup

Status
Succeeded

Location
East US

Subscription (change)
My Subscription Name

Subscription ID
00000000-0000-0000-0000-000000000000

Tags (change)
Click here to add tags

Kubernetes version
1.13.11

API server address
myakscluster-dns-429509e5.hcp.eastus.azmk8s.io

HTTP application routing domain
N/A

Node pools
1 node pools

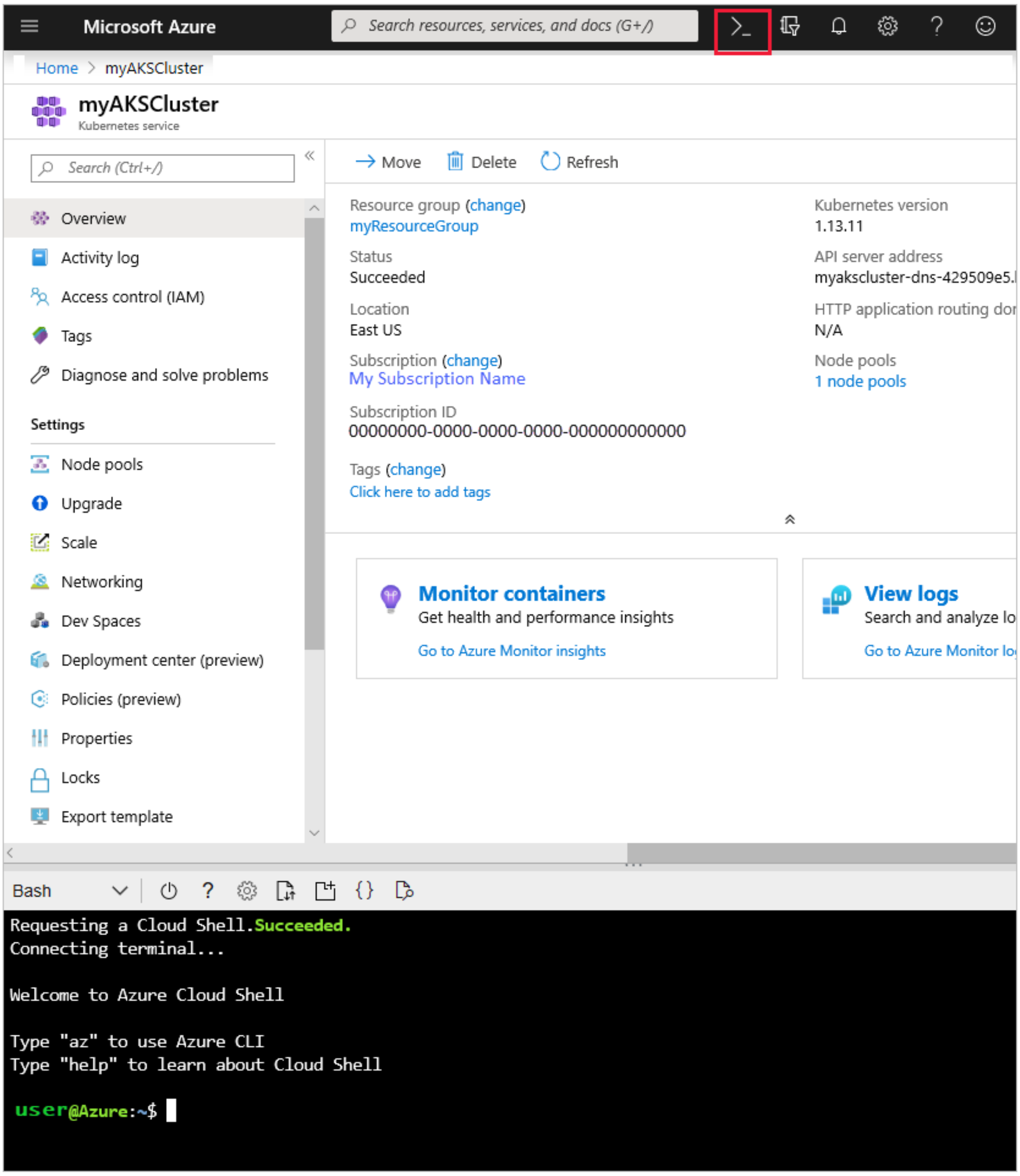
Monitor containers
Get health and performance insights
Go to Azure Monitor insights

View logs
Search and analyze logs using ad-hoc queries
Go to Azure Monitor logs

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.

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



Resource group (change)
myResourceGroup

Status
Succeeded

Location
East US

Subscription (change)
My Subscription Name

Subscription ID
00000000-0000-0000-0000-000000000000

Tags (change)
Click here to add tags

Kubernetes version
1.13.11

API server address
myakscluster-dns-429509e5.l

HTTP application routing dor
N/A

Node pools
1 node pools

```
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

user@Azure:~$ 
```

> **Note**
>
> To perform these operations in a local shell installation:
>
> a. Verify Azure CLI is installed.
>
> b. Connect to Azure via the `az login` command.

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

```
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-14693408-0   Ready    agent   15m     v1.11.5
```

Run 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:
        "beta.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:
        "beta.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
```

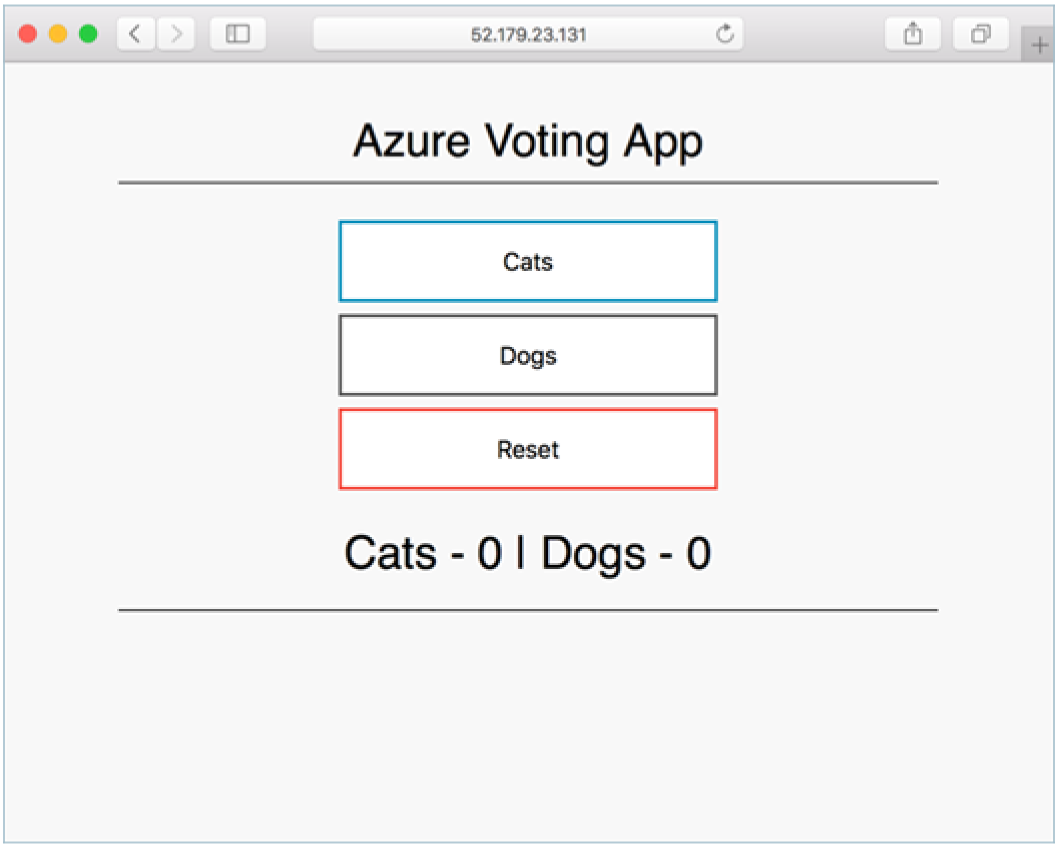
Once the **EXTERNAL-IP** address changes from *pending* to an actual public IP address, use `CTRL-C` to stop the `kubectl` watch process. The following example output shows a valid public IP address assigned to the service:

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.



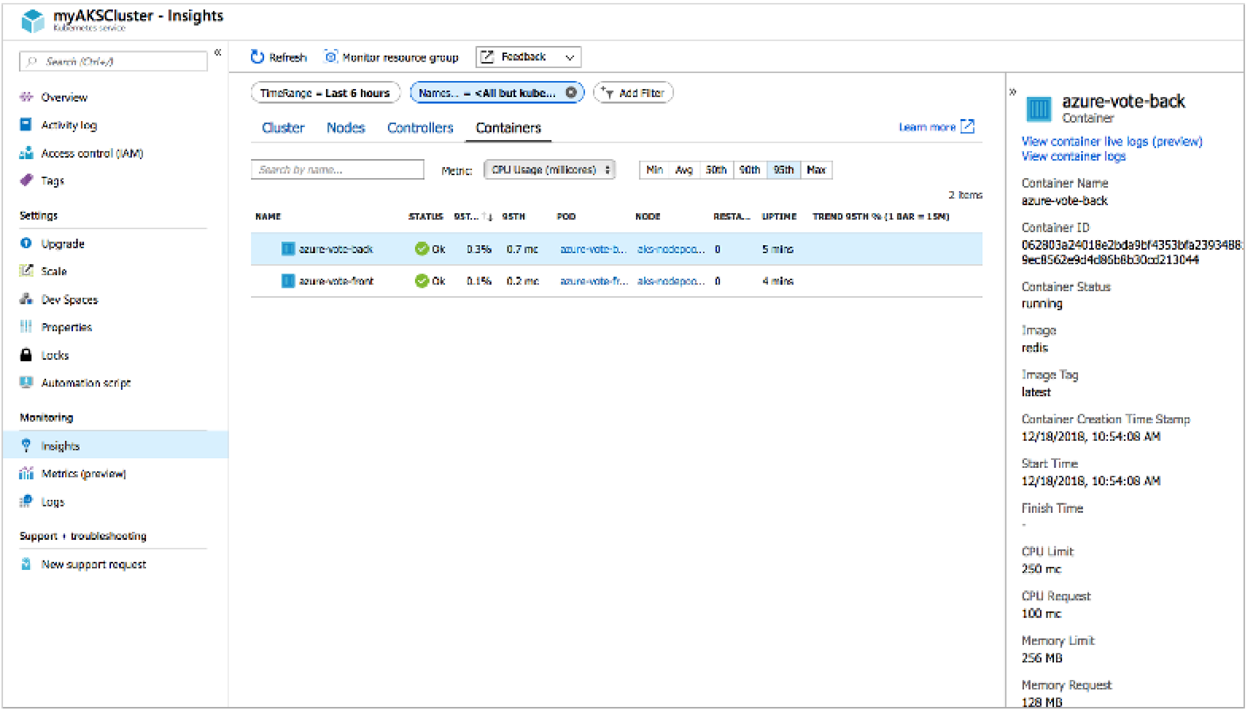
Monitor health and logs

When you created the cluster, Azure Monitor for containers was enabled. Azure Monitor for containers provides health metrics for both the AKS cluster and pods running on the cluster.

Metric data takes a few minutes to populate in the Azure portal. To see current health status, uptime, and resource usage for the Azure Vote pods:

1. Browse back to the AKS resource in the Azure portal.
2. Under **Monitoring** on the left-hand side, choose **Insights**.
3. Across the top, choose to **+ Add Filter**.
4. Select **Namespace** as the property, then choose *<All but kube-system>*.
5. Select **Containers** to view them.

The `azure-vote-back` and `azure-vote-front` containers will display, as shown in the following example:



NAME	STATUS	95T... ↑↓	95TH	POD	NODE	RESTA...	UPTIME	TREND 95TH % (1 BAR = 15M)
azure-vote-back	✓ Ok	0.3%	0.7 mc	azure-vote-b...	aks-nodepoo...	0	5 mins	
azure-vote-front	✓ Ok	0.1%	0.2 mc	azure-vote-fr...	aks-nodepoo...	0	4 mins	

azure-vote-back
Container

View container live logs (preview)
View container logs

Container Name
azure-vote-back

Container ID
062803a24018e2bda9bf4353bfa2393488:
9ec8562e9d4d86b8b30cd213044

Container Status
running

Image
redis

Image Tag
latest

Container Creation Time Stamp
12/18/2018, 10:54:08 AM

Start Time
12/18/2018, 10:54:08 AM

Finish Time
-

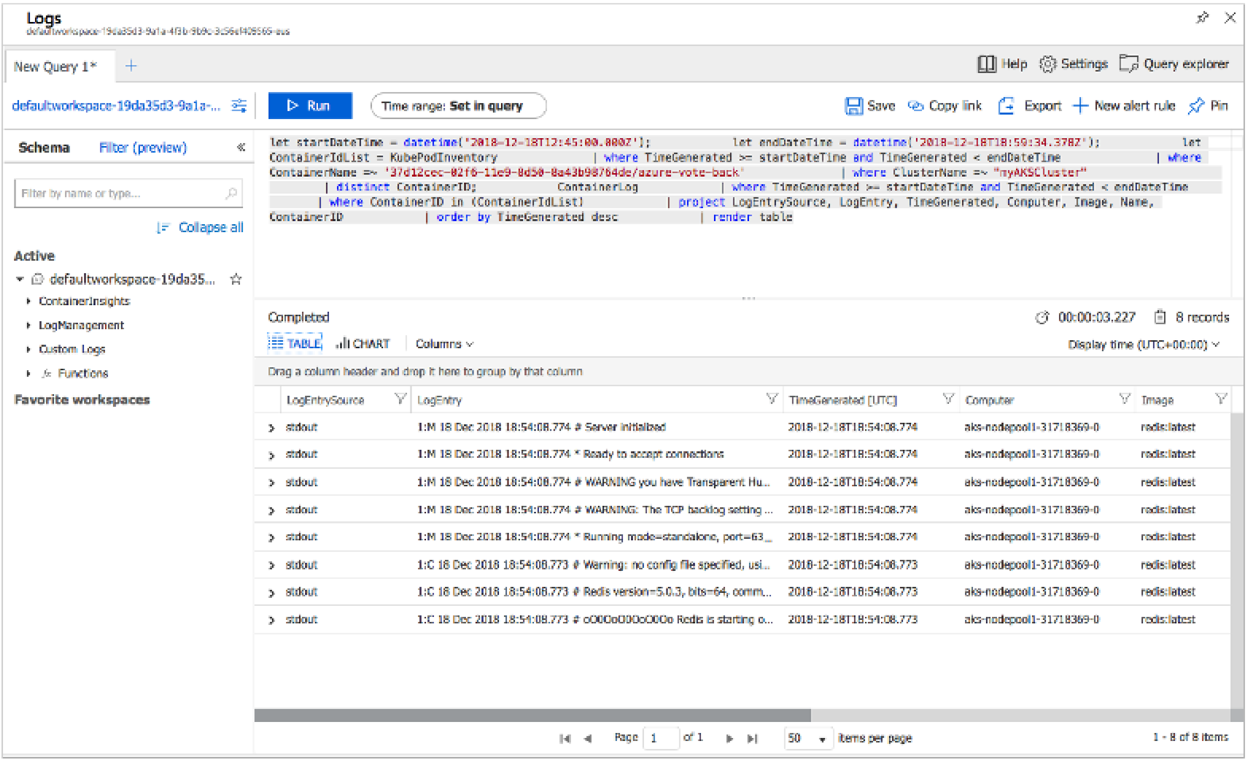
CPU Limit
250 mc

CPU Request
100 mc

Memory Limit
256 MB

Memory Request
128 MB

To view logs for the `azure-vote-front` pod, select **View container logs** from the containers list drop-down. These logs include the *stdout* and *stderr* streams from the container.



```
let startDateTime = datetime('2018-12-18T12:45:00.000Z');          let endDateTime = datetime('2018-12-18T18:59:34.378Z');          let
ContainerIdList = KubePodInventory                    | where TimeGenerated >= startDateTime and TimeGenerated < endDateTime          | where
ContainerName =~ '37d12cec-02f6-11e9-8d50-8a43b98764de/azure-vote-back'          | where ClusterName =~ "myAKSCluster"
        | distinct ContainerID;              ContainerLog              | where TimeGenerated >= startDateTime and TimeGenerated < endDateTime
        | where ContainerID in (ContainerIdList)            | project LogEntrySource, LogEntry, TimeGenerated, Computer, Image, Name,
ContainerID           | order by TimeGenerated desc           | render table
```

LogEntrySource	LogEntry	TimeGenerated [UTC]	Computer	Image
stdout	1:M 18 Dec 2018 18:54:08.774 # Server initialized	2018-12-18T18:54:08.774	aks-nodepool1-31718369-0	redis:latest
stdout	1:M 18 Dec 2018 18:54:08.774 * Ready to accept connections	2018-12-18T18:54:08.774	aks-nodepool1-31718369-0	redis:latest
stdout	1:M 18 Dec 2018 18:54:08.774 # WARNING you have Transparent Hu...	2018-12-18T18:54:08.774	aks-nodepool1-31718369-0	redis:latest
stdout	1:M 18 Dec 2018 18:54:08.774 # WARNING: The TCP backlog setting ...	2018-12-18T18:54:08.774	aks-nodepool1-31718369-0	redis:latest
stdout	1:M 18 Dec 2018 18:54:08.774 * Running mode=standalone, port=63...	2018-12-18T18:54:08.774	aks-nodepool1-31718369-0	redis:latest
stdout	1:C 18 Dec 2018 18:54:08.773 # Warning: no config file specified, usi...	2018-12-18T18:54:08.773	aks-nodepool1-31718369-0	redis:latest
stdout	1:C 18 Dec 2018 18:54:08.773 # Redis version=5.0.3, bits=64, comm...	2018-12-18T18:54:08.773	aks-nodepool1-31718369-0	redis:latest
stdout	1:C 18 Dec 2018 18:54:08.773 # oO0OoO0OoO0Oo Redis is starting o...	2018-12-18T18:54:08.773	aks-nodepool1-31718369-0	redis:latest

Delete cluster

To avoid Azure charges, clean up your unnecessary resources. Select the **Delete** button on the AKS cluster dashboard. You can also use the az aks delete command in the Cloud Shell:

Azure CLI

Copy

```
az aks delete --resource-group myResourceGroup --name myAKSCluster --no-wait
```

ⓘ Note

When you delete the cluster, the Azure Active Directory service principal used by the AKS cluster is not removed. For steps on how to remove the service principal, see **AKS service principal considerations and deletion**.

If you used a managed identity, the identity is managed by the platform and does not require removal.

Get the code

Pre-existing container images were used in this quickstart to create a Kubernetes deployment. The related application code, Dockerfile, and Kubernetes manifest file are available on GitHub.

Next steps

In this quickstart, you deployed a Kubernetes cluster and then deployed a multi-container application to it. Access the Kubernetes web dashboard for your AKS cluster.

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

Is this page helpful?

Yes No

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.

Show more