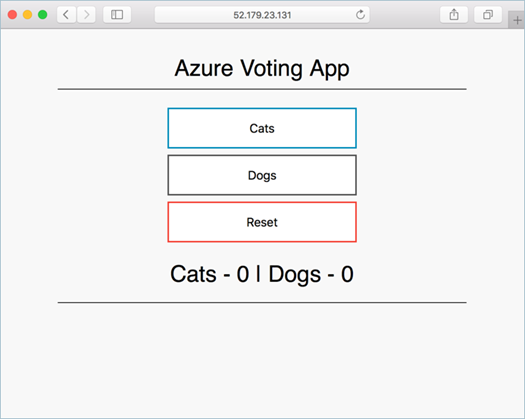
**Deploy an Azure Kubernetes Service (AKS) cluster using the Azure portal**

* Article
* 04/29/2022
* 5 minutes to read
* 2 contributors

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)](https://docs.microsoft.com/en-us/azure/aks/concepts-clusters-workloads).

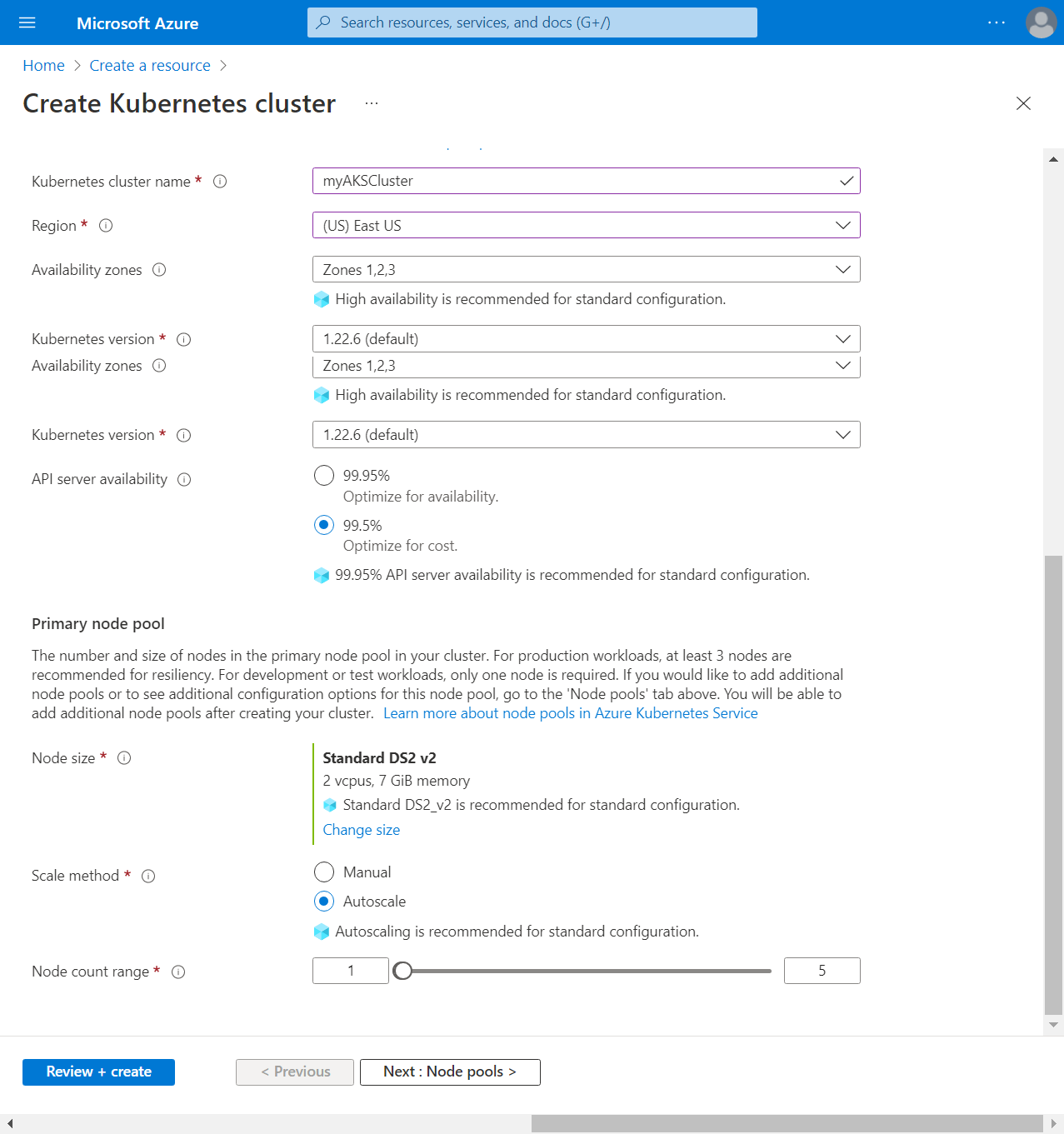
**Prerequisites**

If you don't have an [Azure subscription](https://docs.microsoft.com/en-us/azure/guides/developer/azure-developer-guide#understanding-accounts-subscriptions-and-billing), create an [Azure free account](https://azure.microsoft.com/free/?ref=microsoft.com&utm_source=microsoft.com&utm_medium=docs&utm_campaign=visualstudio) before you begin.

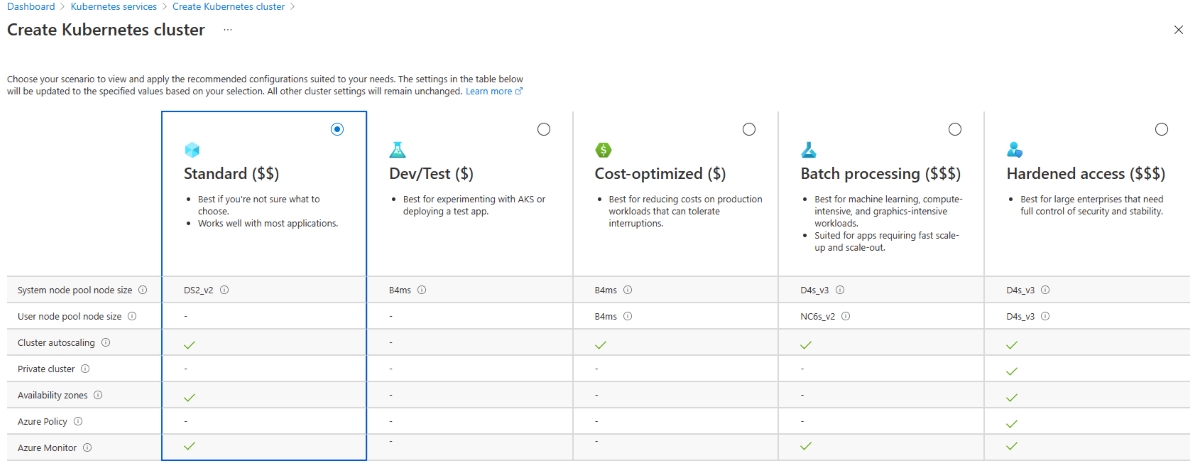
* If you are unfamiliar with using the Bash environment in Azure Cloud Shell, review [Overview of Azure Cloud Shell](https://docs.microsoft.com/en-us/azure/cloud-shell/overview).
* The identity you are using to create your cluster has the appropriate minimum permissions. For more details on access and identity for AKS, see [Access and identity options for Azure Kubernetes Service (AKS)](https://docs.microsoft.com/en-us/azure/aks/concepts-identity).

**Create an AKS cluster**

1. Sign in to the [Azure portal](https://portal.azure.com/).
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](https://docs.microsoft.com/en-us/azure/aks/quotas-skus-regions#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.



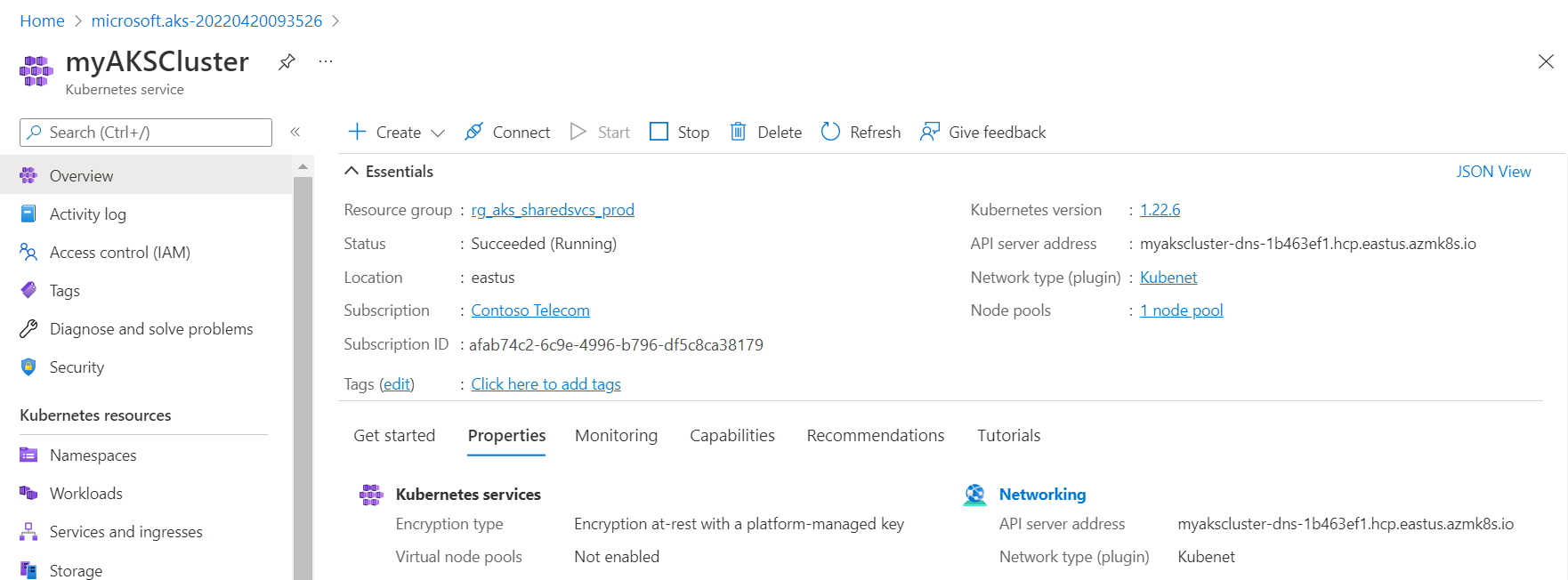
**Note**

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

1. Select **Next: Node pools** when complete.
2. Keep the default **Node pools** options. At the bottom of the screen, click **Next: Access**.
3. 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?](https://docs.microsoft.com/en-us/azure/active-directory/managed-identities-azure-resources/overview).
   * 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](https://docs.microsoft.com/en-us/azure/azure-monitor/containers/container-insights-overview) is enabled.

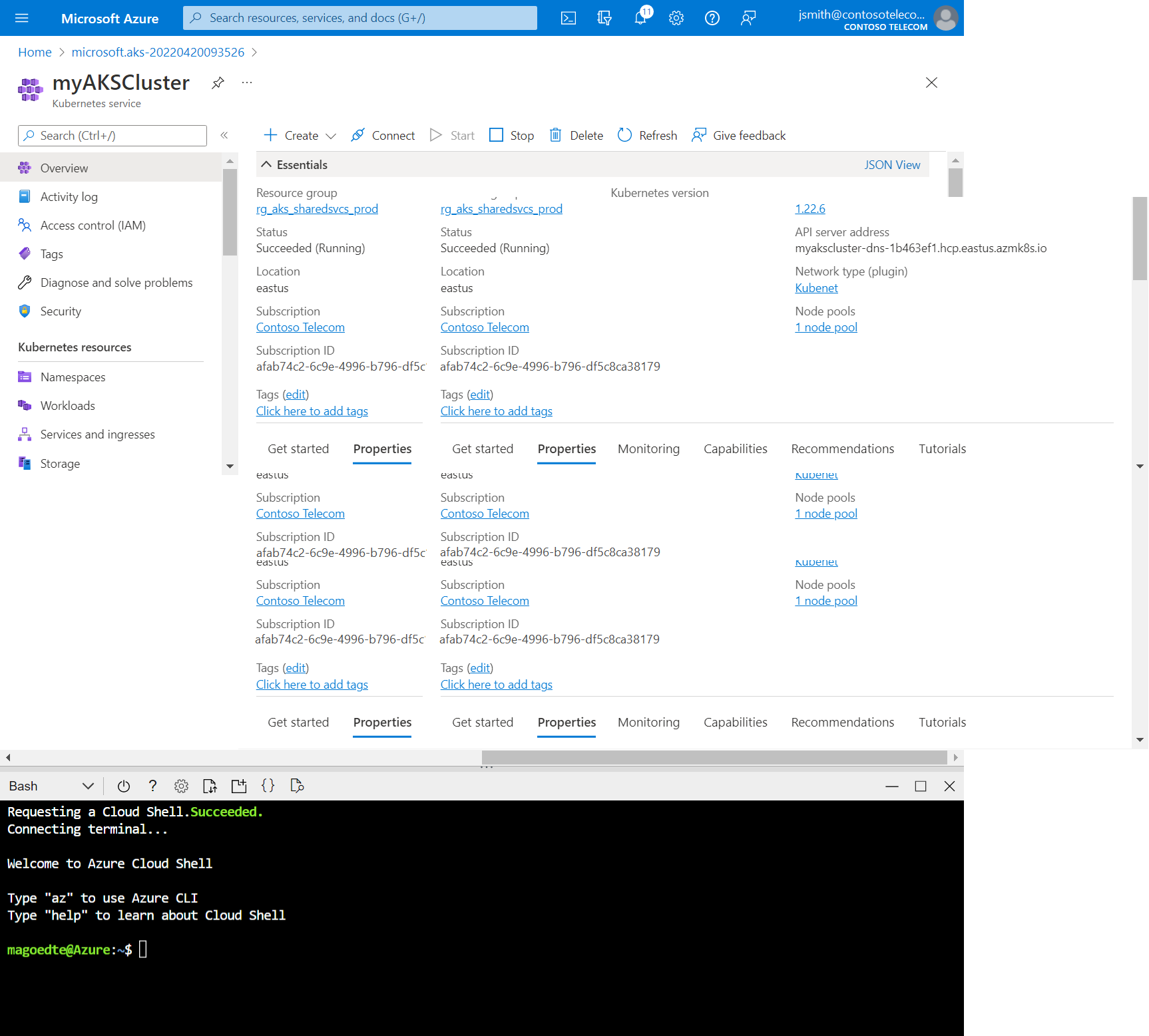
1. Click **Review + create**. When you navigate to the **Review + create** tab, Azure runs validation on the settings that you have chosen. If validation passes, you can proceed to create the AKS cluster by selecting **Create**. If validation fails, then it indicates which settings need to be modified.
2. 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*.



**Connect to the cluster**

To manage a Kubernetes cluster, use the Kubernetes command-line client, [kubectl](https://kubernetes.io/docs/user-guide/kubectl/). kubectl is already installed if you use Azure Cloud Shell. If you are unfamiliar with the Cloud Shell, review [Overview of Azure Cloud Shell](https://docs.microsoft.com/en-us/azure/cloud-shell/overview).

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



**Note**

To perform these operations in a local shell installation:

* 1. Verify Azure CLI is installed.
  2. Connect to Azure via the az login command.

1. Configure kubectl to connect to your Kubernetes cluster using the [az aks get-credentials](https://docs.microsoft.com/en-us/cli/azure/aks" \l "az_aks_get_credentials) command. The following command downloads credentials and configures the Kubernetes CLI to use them.

Azure CLICopy

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

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

ConsoleCopy

kubectl get nodes

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

OutputCopy

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:

YAMLCopy

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

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

ConsoleCopy

kubectl apply -f azure-vote.yaml

Output shows the successfully created deployments and services:

OutputCopy

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.

ConsoleCopy

kubectl get service azure-vote-front --watch

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

OutputCopy

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE

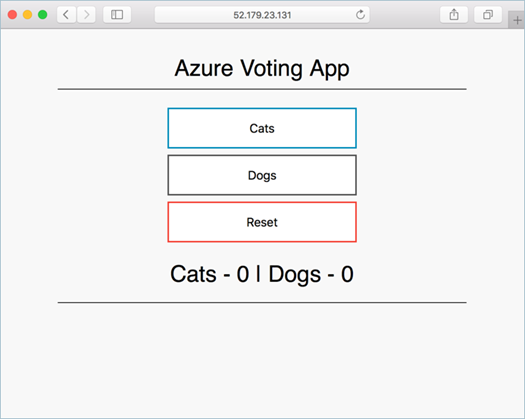
azure-vote-front LoadBalancer 10.0.37.27 <pending> 80:30572/TCP 6s

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:

OutputCopy

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 aks delete](https://docs.microsoft.com/en-us/cli/azure/aks" \l "az_aks_delete) command in the Cloud Shell:

Azure CLICopy

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