Create a Kubernetes Cluster

Azure Kubernetes Service (AKS) is a managed Kubernetes service that lets you quickly deploy and manage clusters. In this quickstart, you deploy an AKS cluster using the Azure CLI. A multi-container application that includes a web front end and a Redis instance is run in the cluster.

1- Create an AKS environment

✔ **Enable cluster monitoring**

Verify Microsoft.OperationsManagement and Microsoft.OperationalInsights are registered on your subscription. To check the registration status:

```
$ az provider show -n Microsoft.OperationsManagement -o table
$ az provider show -n Microsoft.OperationalInsights -o table
```

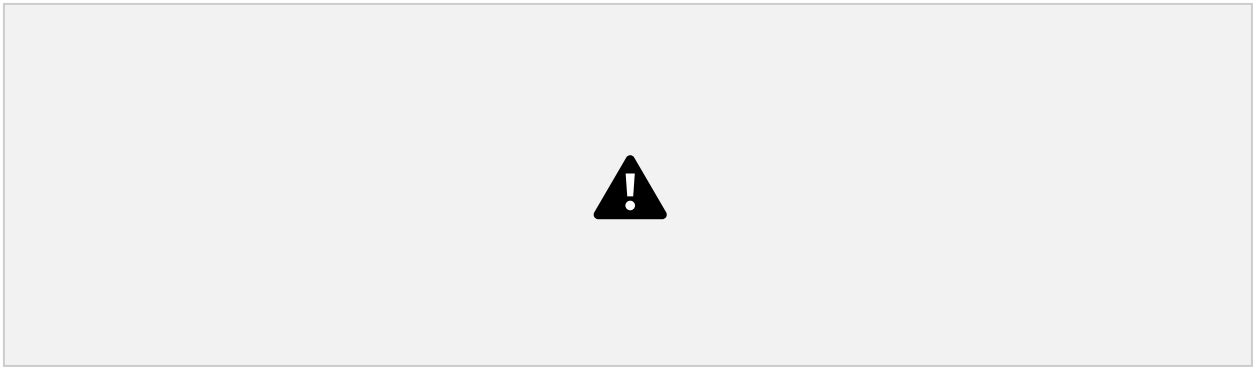
✔ **Create a resource group**

An Azure resource group is a logical group in which Azure resources are deployed and managed. When you create a resource group, you will be prompted to specify a location. This location is:

- The storage location of your resource group metadata.
- Where your resources will run in Azure if you don't specify another region during resource creation.

The following example creates a resource group named **myAKSResourceGroup** in the eastus location. Create a resource group using the az group create command.

```
$ az group create --name myAKSResourceGroup --location eastus
```



✔ **Picking a Kubernetes API Version**

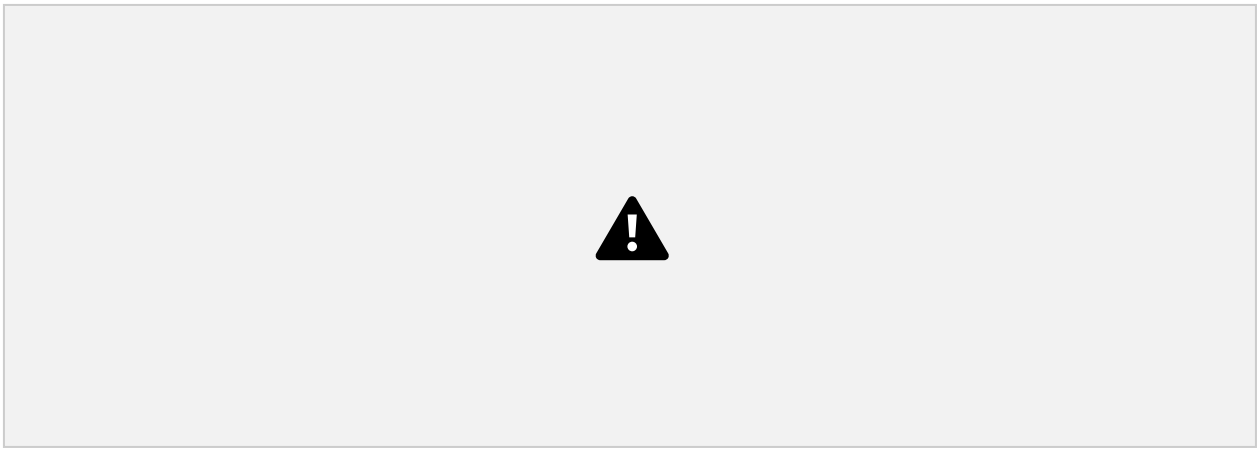
To learn what Kubernetes API versions are available in each region, run the following AZ CLI command:

```
$ az aks get-versions --location region_name
```

✔ **Create AKS cluster**

Create an AKS cluster using the az aks create. The following example creates a cluster named myAKSCluster with one node:

```
$ az aks create  --resource-group myAKSResourceGroup --name
myAKSCluster --node-count 1 --kubernetes-version 1.21.2
--generate-ssh-keys --enable-addons monitoring
```



✔ 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.
- Uses ~/.kube/config, the default location for the Kubernetes configuration file. Specify a different location for your Kubernetes configuration file using --file.

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

Verify the connection to your cluster using the kubectl get command. This command returns a list of the cluster nodes. Output shows the single node created in the previous steps. Make sure the node status is Ready:

2- Run the application

A Kubernetes manifest file defines a desired state for the cluster, such as what container images to run. In this lab, a manifest is used to create all objects needed to run the Azure Vote application. This manifest includes two kubernetes-deployment - one for the sample Azure Vote Python applications, and the other for a Redis instance. Two kubernetes-service are also created - an internal service for the Redis instance, and an external service to access the Azure Vote application from the internet.

1- Create a file named azure.yaml
2- Copy in the following YAML definition: https://gitlab.com/hosniah/az300-training/-/raw/main/azure.yaml
3- Deploy the application using the kubectl apply command and specify the name of your YAML manifest:

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

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

3- Test the application

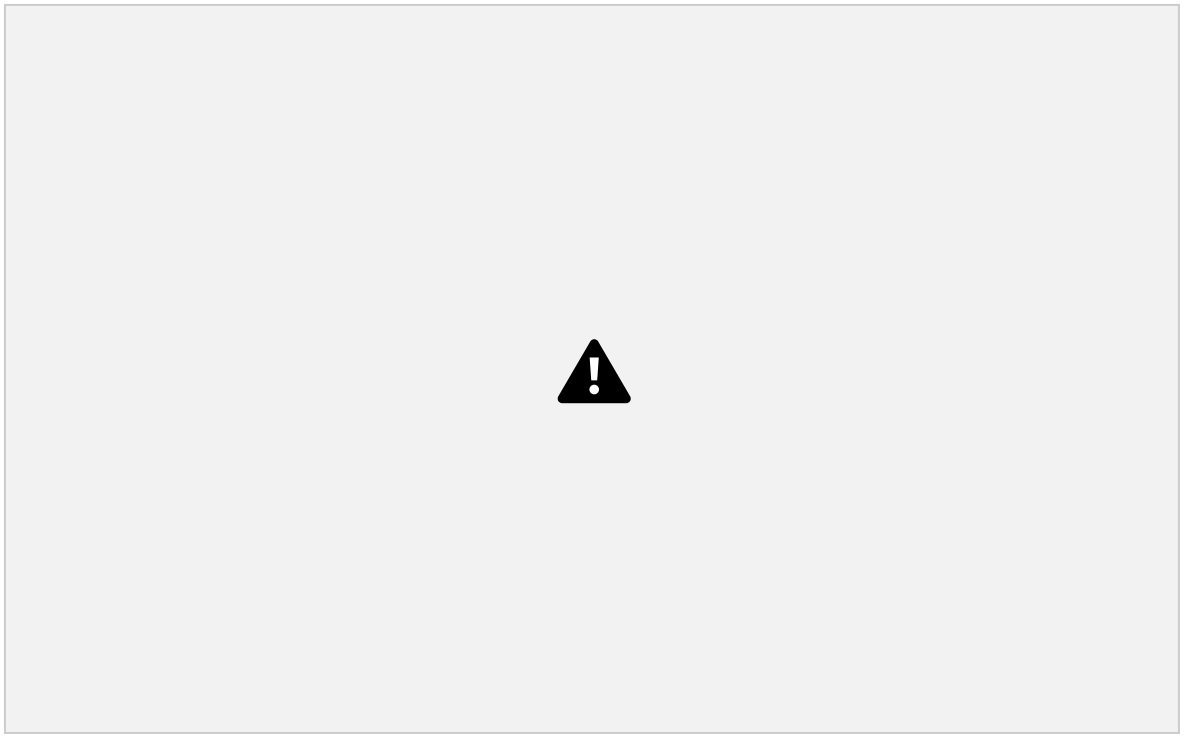
1- To monitor progress, use the kubectl-get command with the --watch argument.

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

2- Initially the EXTERNAL-IP for the azure-vote-front service is shown as pending. When 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:

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

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



4- **Manually scale pods**

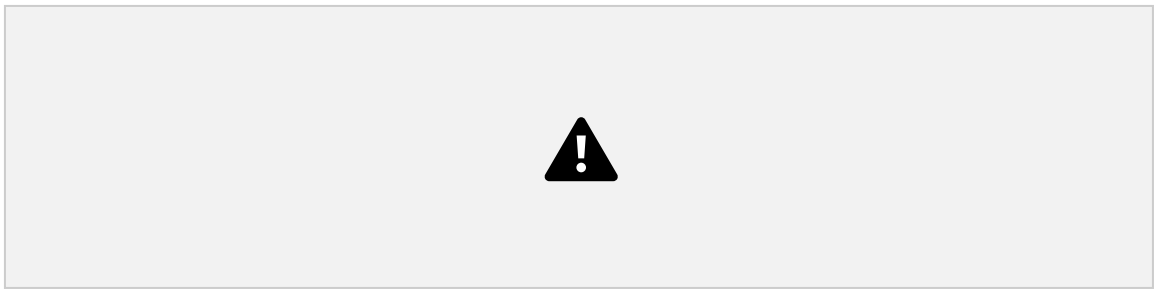
When the Azure Vote front-end and Redis instance were deployed in previous tutorials, a single replica was created. To see the number and state of pods in your cluster, use the kubectl get command as follows:

```
$ kubectl get pods
```

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:

```
$ 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:



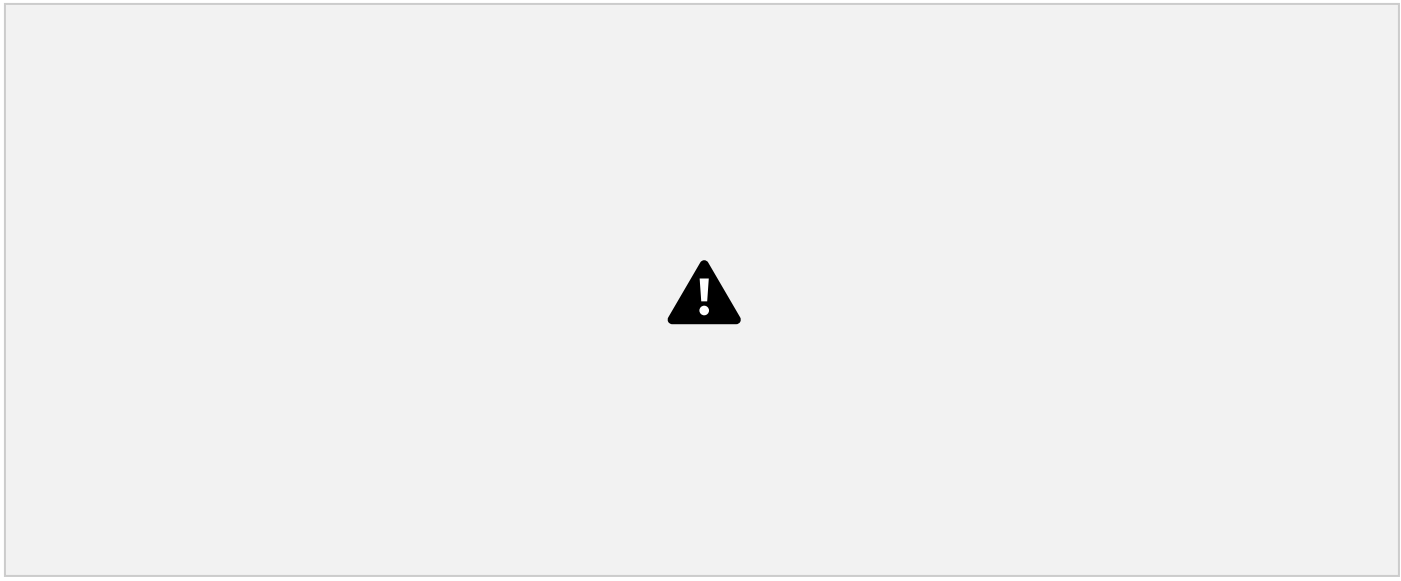
5- Manually scale AKS nodes

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.

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

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



6- Delete the cluster

When the cluster is no longer needed, use the az group delete command to remove the resource group, container service, and all related resources.

Run the following command in the Cloud Shell in Bash mode to delete the Resource Group:

```
$ az group delete --name myAKSResourceGroup --yes --no-wait
```

Note: It may take some time to delete the Resource Group. The --no-wait option runs the command in the background.