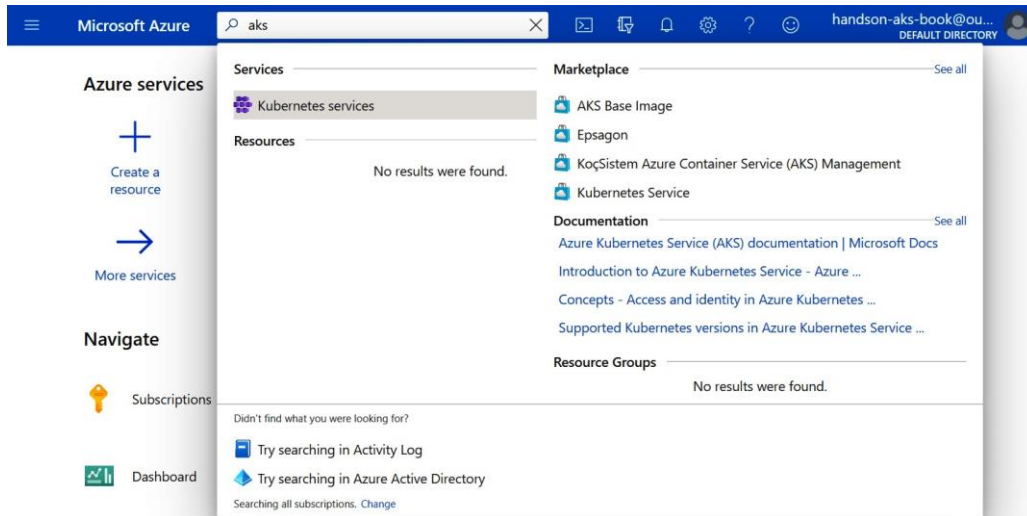


Experiment: AKS Foundation

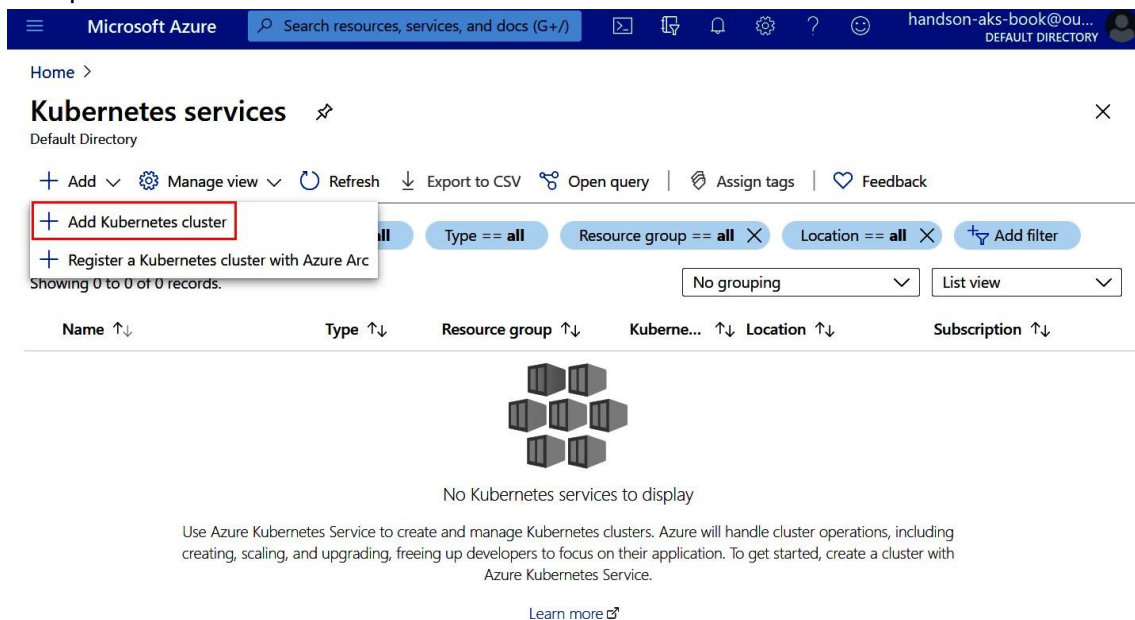
To start, browse to the Azure portal on <https://portal.azure.com>.

1. Enter the keyword **aks** in the search bar at the top of the Azure portal.
2. Click on **Kubernetes services** under the **Services** category in the search results:



This will take you to the AKS pane in the portal. As you might have expected, you don't have any clusters yet.

3. Create a new cluster by selecting the **+ Add** button, and select the **+ Add Kubernetes cluster** option:



Note: There are a lot of options to configure when you're creating an AKS cluster. For your first cluster, it is recommended sticking with the defaults from the portal and following our naming guidelines during this example. The following settings were tested to work reliably.

This will take you to the creation wizard to create your first AKS cluster.

4. Use our AzureDataEngineeringxx resource group.
5. Give your cluster a name—adeaksxx. The region we will use in the book is (US) West US 2, but you could use any other region of choice close to your location. Unselect all the zones in the Availability zones.

Note: Remember that anywhere we see xx we replace that with our student number, student1 would replace xx with 01 and student12 would replace xx with 12.

6. Select a Kubernetes version—at the time of last testing, version 1.22.6 is the latest version that is supported. Kubernetes and AKS evolve very quickly, and new versions are introduced often:

Note: For production environments, deploying a cluster in an Availability Zone is recommended. However, since we are deploying a small cluster, not using Availability Zones works best for the examples in the book.

Create Kubernetes cluster ...

Resource group *	<div>AzureDataEngineering</div> <div>Create new</div>
Cluster details	
Cluster preset configuration	<div>Standard (\$\$)</div> <div>To quickly customize your Kubernetes cluster, choose one of the preset configurations above. You can modify these configurations at any time. Learn more and compare presets</div>
Kubernetes cluster name *	<div>adeaks</div>
Region *	<div>(US) West US 2</div>
Availability zones	<div>None</div> <div>High availability is recommended for standard configuration.</div>
Kubernetes version *	<div>1.22.6 (default)</div>
API server availability	<div><input type="radio"/> 99.9% Optimize for availability. 99.95% is available when at least one availability zone is selected.</div> <div><input checked="" type="radio"/> 99.5% Optimize for cost.</div>

Next, change the node count to 2. For the purposes of this experiment, the default Standard DS2 v2 node size is sufficient.

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 *	<div>Standard DS2 v2</div> <div>Standard DS2_v2 is recommended for standard configuration. Change size</div>
Scale method *	<div><input checked="" type="radio"/> Manual</div> <div><input type="radio"/> Autoscale</div> <div>Autoscaling is recommended for standard configuration.</div>
Node count *	<div><div></div><div>2</div></div>

[Review + create](#)[< Previous](#)[Next : Node pools >](#)

Note

Free accounts have a four-core limit that will be breached if you go with the defaults.

There are a number of configuration panes, which you need not change for our experiment clusters. Review the other sub-blades, and update the Tags as shows below, hit the **Review + create** button to do a final review and create your cluster:

[Home](#) > [Kubernetes services](#) >

Create Kubernetes cluster ...

Basics Node pools Access Networking Integrations Advanced **Tags** Review + create

Tags are name/value pairs that enable you to categorize resources and view consolidated billing by applying the same tag to multiple resources and resource groups. [Learn more about tags](#)

Note that if you create tags and then change resource settings on other tabs, your tags will be automatically updated.

Name ⓘ	Value ⓘ	Resource
project	azure-data-engineering	2 selected
		2 selected

Review + create

< Previous

Next : Review + create >

In the final view, Azure will validate the configuration that was applied to your first cluster. If you get the message **Validation passed**, click **Create**:

✓ Validation passed

Basics Node pools Access Networking Integrations Advanced Tags Review + create

Basics

Subscription	Azure subscription 1
Resource group	AzureDataEngineering
Region	West US 2
Kubernetes cluster name	adeaks
Kubernetes version	1.22.6

Node pools

Node pools	1
Enable virtual nodes	Disabled

Access

Authentication method	System-assigned managed identity
Role-based access control (RBAC)	Enabled

Create

< Previous

Next >

[Download a template for automation](#)

Deploying the cluster should take roughly 10 minutes. Once the deployment is complete, you can check the deployment details.



microsoft.aks-20220517212241 | Overview

Deployment



Search (Ctrl+ /)



Delete



Cancel



Redeploy



Refresh



Overview



Inputs



Outputs



Template



We'd love your feedback! →

Deployment is in progress



Deployment name: microsoft.aks-20220517212241

Subscription: [Azure subscription 1](#)

Resource group: [AzureDataEngineering](#)



Deployment details [\(Download\)](#)

If you get a quota limitation error, as shown below, check the settings and try again. Make sure that you select the **Standard DS2_v2** node size and only two nodes and that you're using the region noted, West US 2:

The screenshot shows the Azure portal interface for an AKS cluster named 'adeaks'. The left sidebar contains a navigation menu with options like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Microsoft Defender for Cloud, and a section for Kubernetes resources including Namespaces, Workloads, Services and ingresses, Storage, Configuration, and Settings. The main content area is divided into 'Essentials' and 'Properties' tabs. The 'Essentials' tab displays key cluster information: Resource group (AzureDataEngineering), Status (Succeeded (Running)), Location (westus2), Subscription (Azure subscription 1), and Subscription ID (b7e892b5-020c-44c1-8d40-99e608f1b2d6). The 'Properties' tab is currently selected, showing details for 'Kubernetes services' (Encryption type: Encryption at-rest with a platform-managed key, Virtual node pools: Not enabled) and 'Node pools' (Node pools: 1 node pool, Kubernetes versions: 1.22.6). A 'Networking' section on the right lists API server address (adeaks-dns-bf35c493), Network type (plugin) (Kubenet), Pod CIDR (10.244.0.0/16), Service CIDR (10.0.0.0/16), DNS service IP (10.0.0.10), and Docker bridge CIDR (172.17.0.1/16).

This is a quick overview of your cluster. It displays the name, the location, and the API server address. The navigation menu on the left provides different options to control and manage your cluster. Let's walk through a couple of interesting options that the has to portal offer.

The **Kubernetes resources** experiment gives you an insight into the workloads that are running on your cluster. You could, for instance, see running deployments and running pods in your cluster. It also allows you to create new resources on your cluster. We will use this experiment later in the chapter after you have deployed your first application on AKS.

In the **Node pools** pane, you can scale your existing node pool (meaning the nodes or servers in your cluster) either up or down by adding or removing nodes. You can add a new node pool, potentially with a different virtual machine size, and you can also upgrade your node pools individually. We can see the **+ Add node pool** option at the top-left corner, and if you select your node pool, the **Upgrade and Scale** options also become available in the top bar:



[+ Add node pool](#) [Refresh](#) | [Delete](#) [Upgrade](#) [Scale](#)

You can add node pools of different types to your cluster to handle a variety of workloads, scale and upgrade your existing node pools, or delete node pools that you no longer need.
[Learn more about multiple node pools](#)

Name	Mode	Provisioning state	Kubernetes version	Availability zones	OS type
agentpool	System	Succeeded	1.19.6	None	Linux



In the **Cluster configuration** pane, you can instruct AKS to upgrade the control plane to a newer version. Typically, in a Kubernetes upgrade, you first upgrade the control plane, and then the individual node pools separately. This pane also allows you to enable **role-based access**


control (RBAC) (which is enabled by default), and optionally integrate your cluster with Azure AD:

 Save  Discard


Upgrade


You can upgrade your cluster to a newer version of Kubernetes. This will upgrade the control plane components of your cluster. To upgrade your node pools, go to the 'Node pools' menu item instead.


[Learn more about upgrading your AKS cluster](#)  [View the Kubernetes changelog](#) 

Kubernetes version 1.19.6 (current)  Cluster is using the latest available version of Kubernetes.

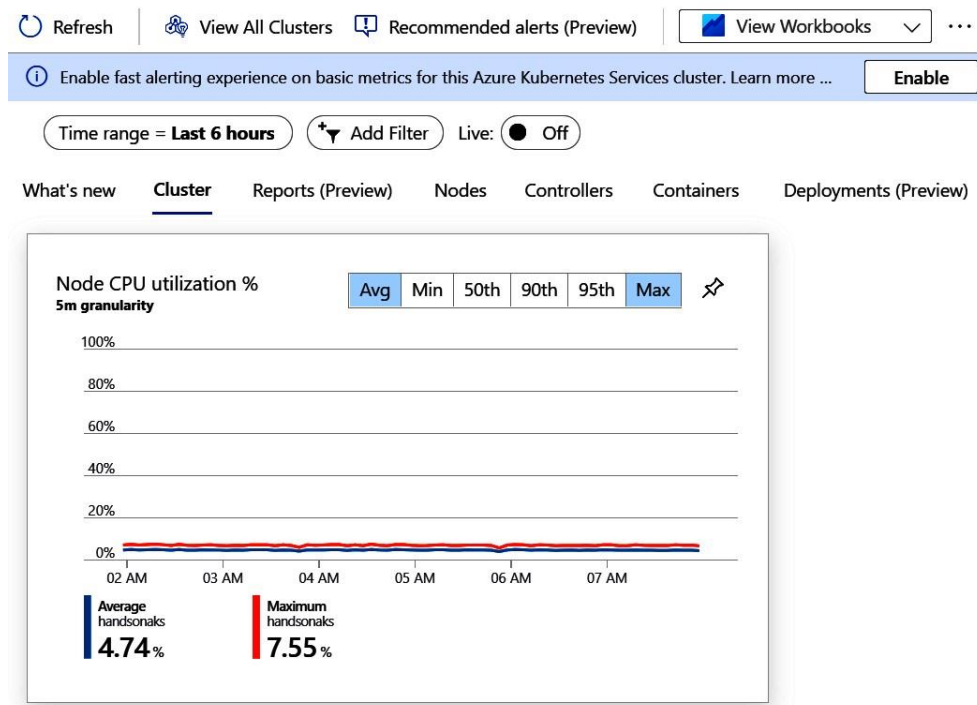
Kubernetes authentication and authorization

Authentication and authorization are used by the Kubernetes cluster to control user access to the cluster as well as what the user may do once authenticated. [Learn more about Kubernetes authentication](#) 

Role-based access control (RBAC)  ☒ Enabled

AKS-managed Azure Active Directory  ☐ Enabled ☒ Disabled

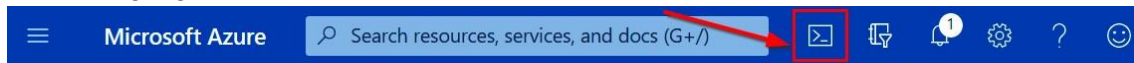
Finally, the **Insights** pane allows you to monitor your cluster infrastructure and the workloads running on your cluster. Since your cluster is brand new, there isn't a lot of data to investigate:



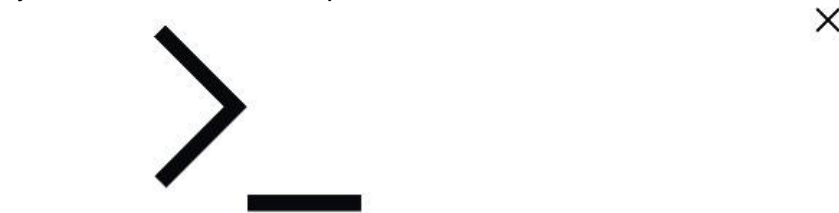
This concludes our quick overview of the cluster and some of the interesting configuration options in the Azure portal. In the next experiment, we'll connect to our AKS cluster using Cloud Shell and then launch a demo application on top of this cluster.

Accessing your cluster using Azure Cloud Shell

Once the deployment is completed successfully, find the small Cloud Shell icon near the search bar, as highlighted, and click it:



The portal will ask you to select either **PowerShell** or **Bash** as your default shell experience. As we will be working mainly with Linux workloads, please select **Bash**:



Welcome to Azure Cloud Shell

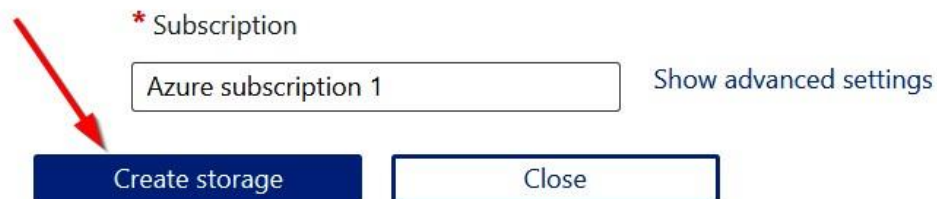
Select Bash or PowerShell. You can change shells any time via the environment selector in the Cloud Shell toolbar. The most recently used environment will be the default for your next session.



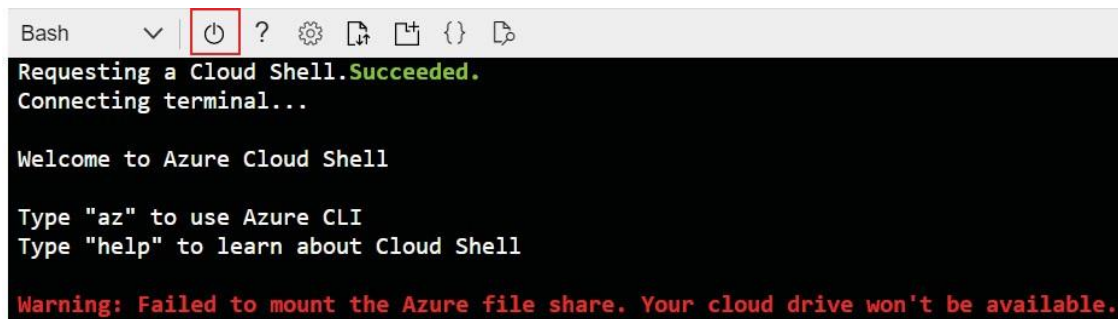
If this is the first time you have launched Cloud Shell, you will be asked to create a storage account; confirm and create it:

You have no storage mounted

Azure Cloud Shell requires an Azure file share to persist files. [Learn more](#)
This will create a new storage account for you and this will incur a small monthly cost. [View pricing](#)



After creating the storage, you might get an error message that contains a mount storage error. If that occurs, please restart your Cloud Shell:



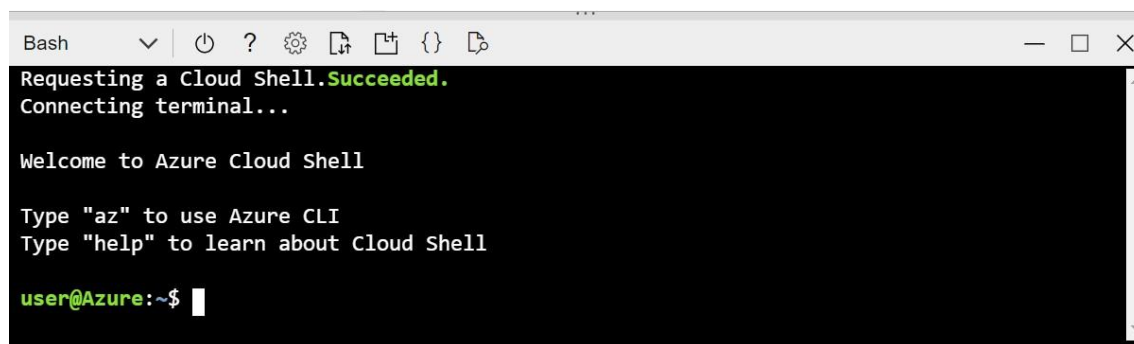
```
Bash
Requesting a Cloud Shell.Succeeded.
Connecting terminal...

Welcome to Azure Cloud Shell

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

Warning: Failed to mount the Azure file share. Your cloud drive won't be available.
```

Click on the power button. It should restart, and you should see something similar to *Figure 2.18*:



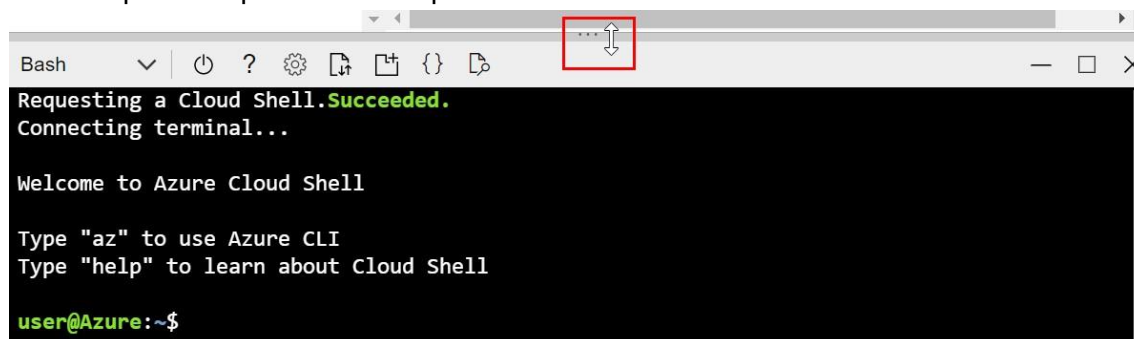
```
Bash
Requesting a Cloud Shell.Succeeded.
Connecting terminal...

Welcome to Azure Cloud Shell

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

user@Azure:~$
```

You can pull the splitter/divider up or down to see more or less of the shell:



```
Bash
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:~$
```

The command-line tool that is used to interface with Kubernetes clusters is called `kubectl`. The benefit of using Azure Cloud Shell is that this tool, along with many others, comes preinstalled and is regularly maintained. `kubectl` uses a configuration file stored in `~/.kube/config` to store credentials to access your cluster.

Note

There is some discussion in the Kubernetes community around the correct pronunciation of `kubectl`. The common way to pronounce it is either *kube-c-t-l*, *kube-control*, or *kube-cuddle*.

To get the required credentials to access your cluster, you need to type the following command:

```
az aks get-credentials \  
--resource-group AzureDataEngineering \  
--name adeaks
```

```
Requesting a Cloud Shell.Succeeded.  
Connecting terminal...  
  
student1@Azure:~$ az aks get-credentials \  
> --resource-group AzureDataEngineering \  
> --name adeaks  
Merged "adeaks" as current context in /home/student1/.kube/config
```

Note

In this book, you will commonly see longer commands spread over multiple lines using the backslash symbol. This helps improve the readability of the commands, while still allowing you to copy and paste them. If you are typing these commands, you can safely ignore the backslash and type the full command in a single line.

To verify that you have access, type the following: `kubectl get nodes`

You should see something like:

```
student1@Azure:~$ kubectl get nodes  
NAME                                STATUS    ROLES    AGE    VERSION  
aks-agentpool-22888953-vmss000000  Ready    agent    34m    v1.22.6  
aks-agentpool-22888953-vmss000001  Ready    agent    34m    v1.22.6  
student1@Azure:~$
```

This command has verified that you can connect to your AKS cluster. In the next experiment, you'll go ahead and launch your first application.

Deploying and inspecting your first demo application

As you are all connected, let's launch your very first application. In this experiment, you will deploy your first application and inspect it using `kubectl` and later using the Azure portal. Let's start by deploying the application.

Deploying the demo application

In this experiment, you will deploy your demo application. For this, you will have to write a bit of code. In Cloud Shell, there are two options to edit code. You can do this either via command-line tools such as `vi` or `nano` or you can use a GUI-based code editor by typing the code commands in Cloud Shell. Throughout this book, you will mainly be instructed to use the graphical editor in the examples, but feel free to use any other tool you feel most comfortable with.

For the purpose of this book, all the code examples are hosted in our GitHub repository. You can clone this repository to your Cloud Shell and work with the code examples directly.

You will create an application based on the definition in the `azure-vote.yaml` file. To open that file in Cloud Shell, you can type the following command: `code azure-vote.yaml`

Here is the code example for your convenience:

```
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:
      containers:
        - name: azure-vote-back
          image: redis
          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:
      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
```

You can make changes to files in the Cloud Shell code editor. If you've made changes, you can save them by clicking on the ... icon in the upper-right corner, and then click **Save** to save the file:

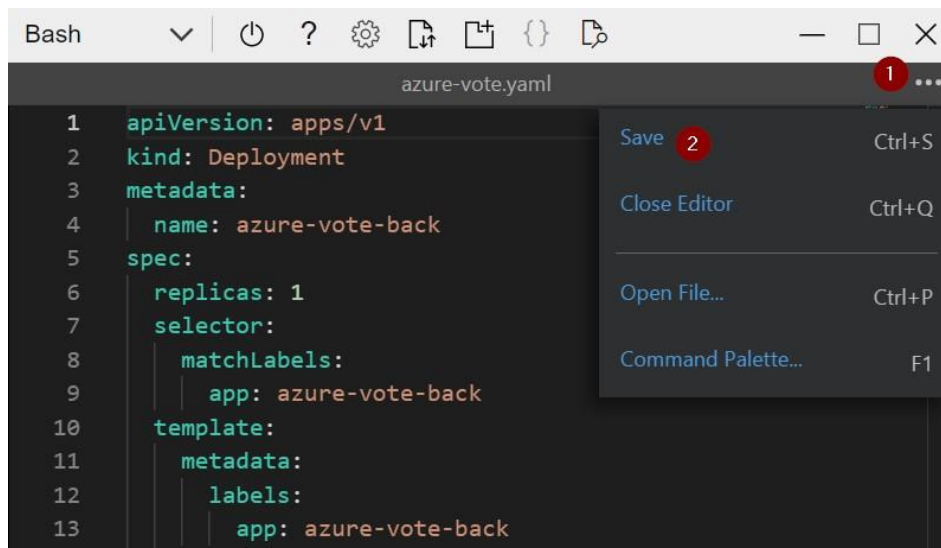


Figure 2.21: Save the azure-vote.yaml file

The file should be saved. You can check this with the following command:

```
cat azure-vote.yaml
```

Note:

Hitting the *Tab* button expands the file name in Linux. In the preceding scenario, if you hit *Tab* after typing *az*, it should expand to *azure-vote.yaml*.

Now, let's launch the application:

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

You should quickly see the output, which tells you which resources have been created:

```
student1@Azure:~$ vi azure-vote.yaml
student1@Azure:~$ kubectl create -f azure-vote.yaml
deployment.apps/azure-vote-back created
service/azure-vote-back created
deployment.apps/azure-vote-front created
service/azure-vote-front created
```

You have successfully created your demo application. In the next experiment, you will inspect all the different objects Kubernetes created for this application and connect to your application.

Exploring the demo application

In the previous experiment, you deployed a demo application. In this experiment, you will explore the different objects that Kubernetes created for this application and connect to it.

You can check the progress of the deployment by typing the following command:

kubectl get pods

If you typed this soon after creating the application, you might have seen that a certain pod was still in the ContainerCreating process:

```
student1@Azure:~$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
azure-vote-back-5f8bd8b-k5zls       1/1     Running   0           72s
azure-vote-front-798779f99d-4p6bf   1/1     Running   0           72s
```

Note

Typing kubectl can become tedious. You can use the alias command to make your life easier. You can use k instead of kubectl as the alias with the following command: alias k=kubectl. After running the preceding command, you can just use k get pods. For instructional purposes in this book, we will continue to use the full kubectl command.

Hit the *up arrow* key and press *Enter* to repeat the kubectl get pods command until the status of all pods is Running. Setting up all the pods takes some time, and you could optionally follow their status using the following command: kubectl get pods --watch

To stop following the status of the pods (when they are all in a running state), you can press *Ctrl* + *C*.

In order to access your application publicly, you need one more thing. You need to know the public IP of the load balancer so that you can access it. If you remember from *Chapter 1, Introduction to containers and Kubernetes*, a service in Kubernetes will create an Azure load balancer. This load balancer will get a public IP in your application so you can access it publicly.

Type the following command to get the public IP of the load balancer:

kubectl get service azure-vote-front --watch

At first, the external IP might show pending. Wait for the public IP to appear and then press *Ctrl* + *C* to exit:

```
user@Azure:~/Hands-On-Kubernetes-on-Azure/Chapter02$ kubectl get service azure-vote-front --watch
NAME            TYPE          CLUSTER-IP   EXTERNAL-IP   PORT(S)          AGE
azure-vote-front LoadBalancer  10.0.220.156 <pending>     80:32248/TCP     3s
azure-vote-front LoadBalancer  10.0.220.156 20.72.205.222 80:32248/TCP   24s
```

Note the external IP address and enter it in your browser.

Azure Voting App

Cats

Dogs

Reset

Cats - 0 | Dogs - 0

Click on **Cats** or **Dogs** and watch the count go up.

To see all the objects in Kubernetes that were created for your application, you can use the

kubectl get all

This should show something like the following:

```
student1@Azure:~$ kubectl get service azure-vote-front --watch
NAME                TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
azure-vote-front    LoadBalancer  10.0.127.29   20.236.42.48   80:30651/TCP     2m32s
^Cstudent1@Azure:~$ kubectl get all
NAME                                READY    STATUS    RESTARTS   AGE
pod/azure-vote-back-5f8bd8b-k5zls   1/1      Running   0           5m3s
pod/azure-vote-front-798779f99d-4p6bf 1/1      Running   0           5m3s

NAME                TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
service/azure-vote-back    ClusterIP     10.0.206.23   <none>         6379/TCP         5m3s
service/azure-vote-front  LoadBalancer 10.0.127.29   20.236.42.48   80:30651/TCP     5m3s
service/kubernetes        ClusterIP     10.0.0.1      <none>         443/TCP         47m

NAME                                READY    UP-TO-DATE    AVAILABLE    AGE
deployment.apps/azure-vote-back     1/1      1              1            5m3s
deployment.apps/azure-vote-front    1/1      1              1            5m3s

NAME                                DESIRED    CURRENT    READY    AGE
replicaset.apps/azure-vote-back-5f8bd8b 1          1          1        5m3s
replicaset.apps/azure-vote-front-798779f99d 1          1          1        5m3s
```

As you can see, a number of objects were created:

- Pods: You will see two pods, one for the back end and one for the front end.
- Services: You will also see two services, one for the back end of type ClusterIP and one for the front end of type LoadBalancer.
- Deployments: You will also see two deployments.
- ReplicaSets: And finally you'll see two ReplicaSets.

You can also view these objects from the Azure portal. To see, for example, the two deployments, you can click on **Workloads** in the left-hand navigation menu of the AKS pane, and you will see all the deployments in your cluster as shown. This figure shows you all the deployments in your cluster, including the system deployments. At the bottom of the list, you can see your own deployments. As you can also see in this figure, you can explore other objects such as pods and ReplicaSets using the top menu:

The screenshot shows the Azure portal interface for an AKS cluster. The left-hand navigation menu is visible, with 'Workloads' selected. The main content area displays a list of deployments. The table has columns for 'Name', 'Namespace', and 'Ready'. The 'Ready' column shows green checkmarks and counts (e.g., 2/2 for coredns). The table lists several system deployments (coredns, coredns-autoscaler, metrics-server, omsagent-rs, tunnelfront) and two user-defined deployments (azure-vote-back, azure-vote-front).

Name	Namespace	Ready
coredns	kube-system	2/2
coredns-autoscaler	kube-system	1/1
metrics-server	kube-system	1/1
omsagent-rs	kube-system	1/1
tunnelfront	kube-system	1/1
azure-vote-back	default	1/1
azure-vote-front	default	1/1

You have now launched your own cluster and your first Kubernetes application. Note that Kubernetes took care of tasks such as connecting the front end and the back end, and exposing them to the outside world, as well as providing storage for the services.

Before moving on, let's clean up this first AKS deployment. Since you created everything from a file, you can also delete everything by pointing Kubernetes to that file.

kubectl delete -f azure-vote.yaml

As the cleanup is done you can watch all your objects get deleted:

```
user@Azure:~/Hands-On-Kubernetes-on-Azure/Chapter02$ kubectl delete -f azure-vote.yaml
deployment.apps "azure-vote-back" deleted
service "azure-vote-back" deleted
deployment.apps "azure-vote-front" deleted
service "azure-vote-front" deleted
```

In this experiment, you have created an Azure Kubernetes Service cluster, connected to your AKS cluster using Cloud Shell, successfully launched and connected to a demo containerized

microservice application, explored the objects created using Cloud Shell and the Azure portal, and finally, cleaned up the resources that were created.