

# #How to deploy Kubernetes on Azure service

## What is Azure Kubernetes Service(AKS)—

Azure Kubernetes Service (AKS) is a managed Kubernetes service that helps developers build and deploy cloud-native applications.

## Steps—

Create a resource group → select containers → search Azure Kubernetes Service (AKS) → create a cluster → add a node pool → review+create → connect the cluster → in Vs Code add aks-store-quickstart.yaml file → deploy the application → delete the cluster → delete the resource group.

### Resource groups

Default Directory (rushikeshkarpe03@gmail.onmicrosoft.com)

+ Create ⚙️ Manage view ▾ ↻ Refresh ⬇️ Export to CSV 🔗 Open query | 🏷️ Assign tags

Filter for any field...

Subscription equals **all**

Location equals **all** ✕

+🔍 Add filter

Showing 1 to 1 of 1 records.

☐ Name ↑↓

<input type="checkbox"/> 📦 deploy
-----------------------------------

- **Kubernetes cluster name:** Enter a cluster name, such as *myAKSCluster*.

- **Region:** Select a region
- **Availability zones:** Select **None**.
- **AKS pricing tier:** Select **Free**.
- Leave the default values for the remaining settings, and select **Next**.

## Create Kubernetes cluster ...

**Basics**

Node pools

Networking

Integrations

Monitoring

Advanced

Tags

**Review + create**

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](#)

### Project details

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

Subscription \* ⓘ

mySubscription

Resource group \* ⓘ

(New) myResourceGroup

[Create new](#)

### Cluster details

Cluster preset configuration \*

Dev/Test

To quickly customize your Kubernetes cluster, choose one of the preset configurations above. You can modify these configurations at any time. [Compare presets](#)

Kubernetes cluster name \* ⓘ

myAKSCluster

Region \* ⓘ

(US) East US 2

Availability zones ⓘ

None

AKS pricing tier ⓘ

Free

Kubernetes version \* ⓘ

1.29.7 (default)

Automatic upgrade ⓘ

Enabled with patch (recommended)

Automatic upgrade scheduler

Every week on Sunday (recommended)

**Start on:** Sat Aug 10 2024 00:00 +00:00 (Coordinated Universal Time)

[Edit schedule](#)

○ On the **Node pools** tab, configure the following settings:

- Select **Add node pool** and enter a **Node pool name**, such as *nplinux*.
- **Mode**: Select **User**.
- **OS SKU**: Select **Ubuntu Linux**.
- **Availability zones**: Select **None**.
- Leave the **Enable Azure Spot instances** checkbox unchecked.
- **Node size**: Select **Choose a size**. On the **Select a VM size** page, select **D2s\_v3**, and then select **Select**.
- Leave the default values for the remaining settings, and select **Add**.
- Select **Review + create** to run validation on the cluster configuration. After validation completes, select **Create**.

It takes a few minutes to create the AKS cluster. When your deployment is complete, navigate to your resource by selecting **Go to resource**, or by browsing to the AKS cluster resource group and selecting the AKS resource.

- To connect the cluster open the powershell and type the following commands

**Add a node pool** ...

myAKSCluster

Node pool name \* ⓘ

Mode \* ⓘ ☒ User ☐ System

OS SKU \* ⓘ ☐ Azure Linux ☒ Ubuntu Linux ☐ Windows 2022 ☐ Windows 2019

Availability zones ⓘ

Enable Azure Spot instances ⓘ ☐

Node size \* ⓘ 

Standard D2s v3  
2 vcpus, 8 GiB memory  
[Choose a size](#)

Scale method ⓘ ☐ Manual ☒ Autoscale - Recommended  
This option is recommended so that the cluster is automatically sized correctly for the current running workloads.

Minimum node count \* ⓘ

Maximum node count \* ⓘ   
The maximum node count allowed for an AKS cluster is 1000 per node pool and 5000 nodes across all node pools in this cluster.

**Optional settings**

Max pods per node \* ⓘ  10 - 250

Enable public IP per node ⓘ ☐

**Labels**

Labels are key/value pairs that can be used to categorize or add identifying information to Kubernetes resources such as nodes. Labels for the node pool will be applied to each node in the node pool. [Learn more](#) ⓘ

Key	Value
<input type="text"/>	<input type="text"/>

**Taints**

Taints are tuples that are used in conjunction with tolerations to determine which pods can be scheduled on which nodes. In order for a pod to be scheduled to a node, it must tolerate all of the taints applied to that node. Taints for the node pool will be applied to each node in the node pool. [Learn more](#) ⓘ

Key	Value	Effect
<input type="text"/>	<input type="text"/>	<input type="text"/>

## Connect the cluster:

1. Configure kubectl to connect to your Kubernetes cluster using the [Import-AzAksCredential](#) cmdlet. This command downloads credentials and configures the Kubernetes CLI to use them.

## **Import-AzAksCredential -ResourceGroupName myResourceGroup -Name myAKSCluster**

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

### **kubectl get nodes**

open an editor and create a file named aks-store-quickstart.yaml.

```
apiVersion: apps/v1
```

```
kind: Deployment
```

```
metadata:
```

```
  name: rabbitmq
```

```
spec:
```

```
  replicas: 1
```

```
  selector:
```

```
    matchLabels:
```

```
      app: rabbitmq
```

```
  template:
```

```
    metadata:
```

```
      labels:
```

```
        app: rabbitmq
```

```
    spec:
```

```
      nodeSelector:
```

"kubernetes.io/os": linux

containers:

- name: rabbitmq

image:

mcr.microsoft.com/mirror/docker/library/rabbitmq:3.10-management-alpine

ports:

- containerPort: 5672

name: rabbitmq-amqp

- containerPort: 15672

name: rabbitmq-http

env:

- name: RABBITMQ\_DEFAULT\_USER

value: "username"

- name: RABBITMQ\_DEFAULT\_PASS

value: "password"

resources:

requests:

cpu: 10m

memory: 128Mi

limits:

cpu: 250m

memory: 256Mi

volumeMounts:

- name: rabbitmq-enabled-plugins

mountPath: /etc/rabbitmq/enabled\_plugins

subPath: enabled\_plugins

volumes:

- name: rabbitmq-enabled-plugins

configMap:

name: rabbitmq-enabled-plugins

items:

- key: rabbitmq\_enabled\_plugins

path: enabled\_plugins

---

apiVersion: v1

data:

rabbitmq\_enabled\_plugins: |

[rabbitmq\_management,rabbitmq\_prometheus,rabbitmq\_a  
mqp1\_0].

kind: ConfigMap

metadata:

name: rabbitmq-enabled-plugins

---

apiVersion: v1

kind: Service

metadata:

name: rabbitmq

spec:

selector:

app: rabbitmq

ports:

- name: rabbitmq-amqp

port: 5672

targetPort: 5672

- name: rabbitmq-http

port: 15672

targetPort: 15672

type: ClusterIP

---

apiVersion: apps/v1

kind: Deployment

metadata:

name: order-service

spec:



```
replicas: 1
selector:
  matchLabels:
    app: order-service
template:
  metadata:
    labels:
      app: order-service
  spec:
    nodeSelector:
      "kubernetes.io/os": linux
    containers:
      - name: order-service
        image: ghcr.io/azure-samples/aks-store-demo/order-
service:latest
        ports:
          - containerPort: 3000
        env:
          - name: ORDER_QUEUE_HOSTNAME
            value: "rabbitmq"
          - name: ORDER_QUEUE_PORT
            value: "5672"
```

- name: ORDER\_QUEUE\_USERNAME

value: "username"

- name: ORDER\_QUEUE\_PASSWORD

value: "password"

- name: ORDER\_QUEUE\_NAME

value: "orders"

- name: FASTIFY\_ADDRESS

value: "0.0.0.0"

resources:

requests:

cpu: 1m

memory: 50Mi

limits:

cpu: 75m

memory: 128Mi

initContainers:

- name: wait-for-rabbitmq

image: busybox

command: ['sh', '-c', 'until nc -zv rabbitmq 5672; do echo waiting for rabbitmq; sleep 2; done;']

resources:

requests:

cpu: 1m

memory: 50Mi

limits:

cpu: 75m

memory: 128Mi

---

apiVersion: v1

kind: Service

metadata:

name: order-service

spec:

type: ClusterIP

ports:

- name: http

port: 3000

targetPort: 3000

selector:

app: order-service

---

apiVersion: apps/v1

kind: Deployment

metadata:

```
name: product-service
spec:
  replicas: 1
  selector:
    matchLabels:
      app: product-service
  template:
    metadata:
      labels:
        app: product-service
    spec:
      nodeSelector:
        "kubernetes.io/os": linux
      containers:
        - name: product-service
          image: ghcr.io/azure-samples/aks-store-demo/product-
service:latest
          ports:
            - containerPort: 3002
          resources:
            requests:
              cpu: 1m
```

memory: 1Mi

limits:

cpu: 1m

memory: 7Mi

---

apiVersion: v1

kind: Service

metadata:

name: product-service

spec:

type: ClusterIP

ports:

- name: http

port: 3002

targetPort: 3002

selector:

app: product-service

---

apiVersion: apps/v1

kind: Deployment

metadata:

name: store-front

spec:

replicas: 1

selector:

matchLabels:

app: store-front

template:

metadata:

labels:

app: store-front

spec:

nodeSelector:

"kubernetes.io/os": linux

containers:

- name: store-front

image: ghcr.io/azure-samples/aks-store-demo/store-front:latest

ports:

- containerPort: 8080

name: store-front

env:

- name: VUE\_APP\_ORDER\_SERVICE\_URL

value: "http://order-service:3000/"

- name: VUE\_APP\_PRODUCT\_SERVICE\_URL

value: "http://product-service:3002/"

resources:

requests:

cpu: 1m

memory: 200Mi

limits:

cpu: 1000m

memory: 512Mi

---

apiVersion: v1

kind: Service

metadata:

name: store-front

spec:

ports:

- port: 80

targetPort: 8080

selector:

app: store-front

type: LoadBalancer

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

**kubectl apply -f aks-store-quickstart.yaml.**

.....In this way we have successfully deploy the Azure Kubernetes Service.

Delete the Cluster and also delete the Resource group.