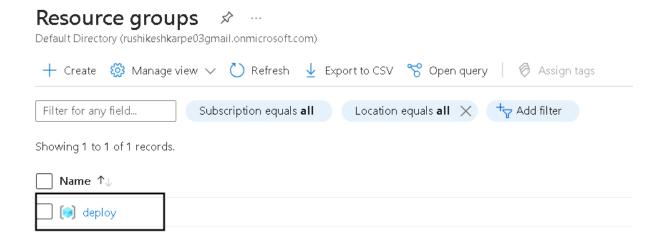
### **#How to deploy Kubernetes on Azure service**

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



 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.

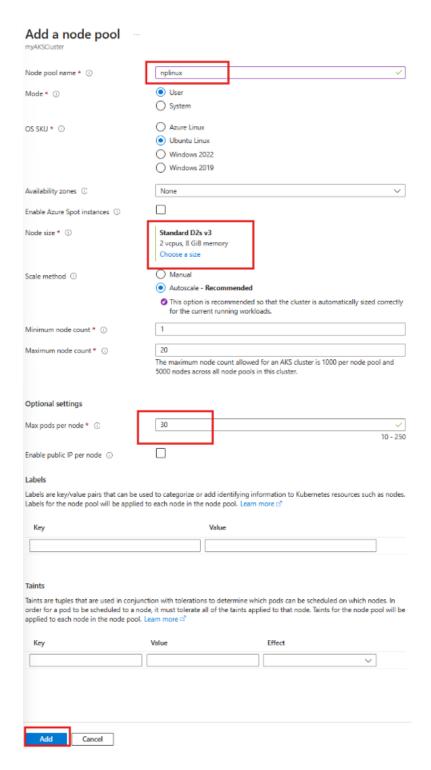
#### Create Kubernetes cluster Node pools Review + create Basics Networking Integrations Monitoring Advanced Azure Kubernetes Service (AKS) manages your hosted Kubernetes environment, making it quick and easy to deploy and manage containerized applications without container orchestration expertise. It also eliminates the burden of ongoing operations and maintenance by provisioning, upgrading, and scaling resources on demand, without taking your applications offline. Learn more of Project details Select a subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources. mySubscription Subscription \* (i) Resource group \* ① (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 myAKSCluster Kubernetes cluster name \* (i) Region \* (i) (US) East US 2 Availability zones (i) None AKS pricing tier (1) 1.29.7 (default) Kubernetes version \* (i) Automatic upgrade (i) Enabled with patch (recommended) Every week on Sunday (recommended) Automatic upgrade scheduler 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



#### **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

nodeSelector:

```
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:
```

```
"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
```

```
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
```

memory: 256Mi

```
___
```

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.