AKS cluster setup.

In basic tab we provide the name for the cluster. And which region we are creating the resource (hardware zone). For control plane highest availability we can choose all zones for redundancy.

A screenshot of a computer

AI-generated content may be incorrect.

Choose AKS version, pricing tier and other details. Also choose patch if yes then recommended date.

A screenshot of a computer

AI-generated content may be incorrect.

Next nodepool creation.

In Azure we must have the system nodepool those nodes will contains default namespaces, kube system and app. This also used by metrices server (control plane, API server use those nodes).

Here, system pool is having most control for azure, while users have less access. It’s best practice to keep systempool with lower size vm and use it only for control plane.

Create another worker nodepool for worker nodes and run the application here.

System pool configuration.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Creating user nodepool.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

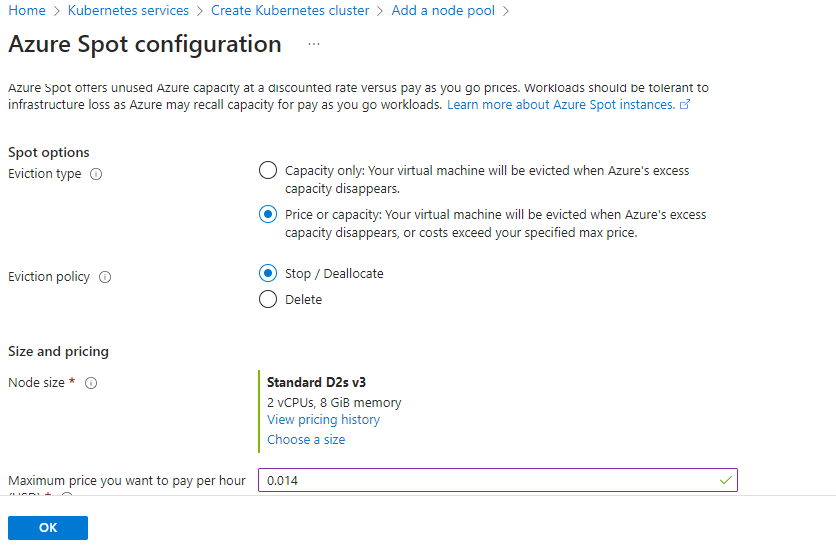
AI-generated content may be incorrect.

I am enabling azure spot instances to reduce cost.

Azure Spot offers unused Azure capacity at a discounted rate versus pay as you go prices. Workloads should be tolerant to infrastructure loss as Azure may recall capacity for pay as you go workloads.

Price or capacity: Your virtual machine will be evicted when Azure's excess capacity disappears, or costs exceed your specified max price.

I have set max price for hour.



Here we have 2 more options. Labels and taints.

**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.

Adding labels to nodes (inherited from nodepool) allows you to target Pods for scheduling on specific nodes in that nodepool. You can use this functionality to ensure that specific Pods only run on nodes (userpool) with certain isolation, security, or regulatory properties.

nodeSelector

nodeSelector is the simplest recommended form of node selection constraint. You can add the nodeSelector field to your Pod specification and specify the [node labels](https://kubernetes.io/docs/concepts/scheduling-eviction/assign-pod-node/#built-in-node-labels) you want the target node to have. Kubernetes only schedules the Pod onto nodes that have each of the labels you specify.

Ex-

spec:

nodeSelector:

environment: production

**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

A screenshot of a computer

AI-generated content may be incorrect.

Use of Taints-

**Scenario:**

You have a Kubernetes cluster with multiple nodes, and you want to **reserve a node exclusively for database workloads** so that no other application pods get scheduled on it.

We don’t required Taints, we will go for labels.

Let's assume you have a node named db-node-1. You can taint this node so that only database-related pods can be scheduled.

sh

CopyEdit

kubectl taint nodes db-node-1 role=database:NoSchedule

This applies a **taint** with:

* **Key:** role
* **Value:** database
* **Effect:** NoSchedule (prevents pods from running unless they tolerate this taint)

**🚫 Step 2: What Happens Without Toleration?**

If a pod does not have a **toleration** for this taint, Kubernetes **will not schedule it** on db-node-1.

For example, if you try to deploy a general application pod without tolerations, it will remain in a **"Pending"** state because it cannot find a suitable node.

**✅ Step 3: Allow Database Pods to Run on the Tainted Node**

To allow database workloads (e.g., PostgreSQL) to run on db-node-1, you need to add a **toleration** in the pod definition.

**Example: PostgreSQL Pod with Toleration**

yaml

CopyEdit

apiVersion: v1

kind: Pod

metadata:

name: postgres-db

spec:

tolerations:

- key: "role"

operator: "Equal"

value: "database"

effect: "NoSchedule"

containers:

- name: postgres

image: postgres:latest

* This pod **tolerates** the role=database:NoSchedule taint.
* Kubernetes will schedule it on db-node-1 because it meets the taint requirements.

**🚀 Step 4: Verify the Taint**

To check if the taint is applied to db-node-1, run:

sh

CopyEdit

kubectl describe node db-node-1 | grep Taint

If you need to **remove** the taint:

sh

CopyEdit

kubectl taint nodes db-node-1 role=database:NoSchedule-

(The - at the end removes the taint.)

**🔍 Summary Table:**

| **Action** | **Command** |
| --- | --- |
| Taint a node | kubectl taint nodes <node-name> key=value:NoSchedule |
| Check node taints | `kubectl describe node <node-name> |
| Remove a taint | kubectl taint nodes <node-name> key=value:NoSchedule- |
| Add toleration in pod | Add tolerations section in pod spec |

A screenshot of a computer

AI-generated content may be incorrect.

Set networking for the cluster.

**Private access**

Enable a private cluster to restrict worker node to API access, enhancing your Kubernetes workload's security and isolation.

**Enable private cluster**

This feature enables secure connection between API server and worker nodes.

In a private cluster, the control plane or API server has internal IP addresses that are defined in the [RFC1918 - Address Allocation for Private Internet](https://tools.ietf.org/html/rfc1918) document. By using a private cluster, you can ensure network traffic between your API server and your node pools remains only on the private network.

The control plane or API server is in an AKS-managed Azure resource group, and your cluster or node pool is in your resource group. The server and the cluster or node pool can communicate with each other through the [Azure Private Link service](https://learn.microsoft.com/en-us/azure/private-link/private-link-service-overview#limitations) in the API server virtual network and a private endpoint that's exposed on the subnet of your AKS cluster.

When you provision a private AKS cluster, AKS by default creates a private FQDN with a private DNS zone and an additional public FQDN with a corresponding A record in Azure public DNS. The agent nodes continue to use the A record in the private DNS zone to resolve the private IP address of the private endpoint for communication to the API server.

Note- Private AKS clusters do not have their API server accessible from the public internet. To access the private cluster, deploy it into a virtual network that is accessible from your computer or follow the AKS private cluster documentation.

The workstation we use to access cluster that should be in the same VNET where the cluster is.

<https://learn.microsoft.com/en-us/azure/aks/private-clusters?tabs=default-basic-networking%2Cazure-portal>

**Public access**

Set authorized IP ranges

We can access the API server only from a set of ip range.

The Kubernetes API server is the core of the Kubernetes control plane and is the central way to interact with and manage your clusters. To improve the security of your clusters and minimize the risk of attacks, we recommend limiting the IP address ranges that can access the API server. To do this, you can use the *API server authorized IP ranges* feature.

The Kubernetes API server exposes underlying Kubernetes APIs and provides the interaction for management tools like kubectl and the Kubernetes dashboard. AKS provides a single-tenant cluster control plane with a dedicated API server. The API server is assigned a public IP address by default. You can control access using Kubernetes role-based access control (Kubernetes RBAC) or Azure RBAC.

<https://learn.microsoft.com/en-us/azure/aks/api-server-authorized-ip-ranges?tabs=azure-cli>

For now I am going with IP range. will test private cluster as well.

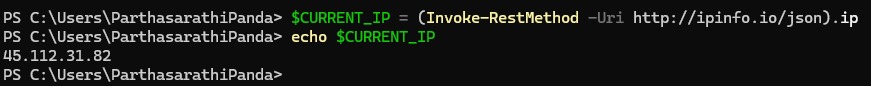
To provide my ip either I can go to browser browse what is my ip and copy or I can check it from powershell as well.

>> $CURRENT\_IP = (Invoke-RestMethod -Uri <http://ipinfo.io/json).ip>

>> echo $CURRENT\_IP

Note- if network changes and the IP got changed in that case we need to keep updating IP again and again. Instate of that if we use a azure vm and add it’s ip that will work for all time. Vm IP should be static so that will never change.

After ip add “/32” to accept only that ip. Else it will consider next ips from that till 255 range.



A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Integrations. If we want to use ACR repo for images.

A screenshot of a computer

AI-generated content may be incorrect.

**Service mesh - Istio**

Enable Istio to configure traffic management, maximize observability capabilities and reinforce service-to-service security measures without changing the application code.

Configure monitoring.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Note- did not enabled for now will do it later for test.

A screenshot of a computer

AI-generated content may be incorrect.

For underlaying infrastructure it will create a new resource group in that all the infra will be present, such as nodes cluster network etc.

Go ahead and create.

If the creation failed due to vm size or non availability at that region change and try.

A screenshot of a computer

AI-generated content may be incorrect.

In my case it’s node selection issue. so, cluster is running nodepool was also ready but nodes are not created.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Either I can go to nodepool and create new. Else simply I can go for redeploy option and change the node size here.

A screenshot of a computer

AI-generated content may be incorrect.

This size is also not available so went with D2ds\_v5

As the systempool (agent pool) and userpool both was impacted deleted both and created nodepool for both.

A screenshot of a computer

AI-generated content may be incorrect.

We can connect to the cluster im many ways.

1. Using own system powershell.
2. Cloudshell
3. By any vm

I have connected via cloudshell.

Make sure to add cloudshell IP to cluster.



Refresh the cluster

>>az aks update --name cluster\_name --resource-group RG-name

A blue screen with white text

AI-generated content may be incorrect.

Check node status.

>> kubectl get nodes

**A blue screen with white text

AI-generated content may be incorrect.**

If we are not continuing to use the cluster we can stop the cluster.

A screenshot of a computer

AI-generated content may be incorrect.

When an AKS cluster is stopped, the **control plane (managed Kubernetes services) stops running**, so the following charges **do not apply**:

1. **AKS Control Plane Cost** (Free in most cases)
   * Since the control plane is managed by Azure, you are not billed separately for it.
2. **Virtual Machines (VMs) Cost**
   * The **worker nodes (agent nodes)** are Azure Virtual Machines. When you stop the AKS cluster, these **VMs are deallocated**, meaning you do not pay for compute resources (CPU, RAM).

**Costs That CONTINUE Even in Stopped State**

Even though the cluster is stopped, some associated resources **still incur charges**:

1. **Persistent Storage (Disks) Costs**
   * If your cluster is using **Azure Disks (Managed Disks)** for persistent storage, those disks **continue to be billed** even when the cluster is stopped.
2. **Load Balancer Costs**
   * If your AKS cluster was using an **Azure Load Balancer (Standard SKU)**, you will continue to be charged for it.
3. **Public IP Addresses (Static) and VNET for aks**
   * Any reserved **static IPs** assigned to your cluster will continue incurring charges.
4. **Azure Storage Accounts**
   * Storage used for logs, container images, or backups will still be billed.
5. **Log Analytics (If enabled)**
   * If Azure Monitor and Log Analytics are enabled, it will continue storing and analyzing logs, which can incur charges.

**How to Minimize Costs While AKS is Stopped?**

If you want to reduce costs further:

* **Delete Disks** that are not needed (az disk delete).
* **Release Static IPs** (az network public-ip delete).
* **Disable Log Analytics** if not required (az monitor log-analytics workspace delete).
* **Check the Load Balancer and remove it** if necessary (az network lb delete).

**How to Check Your AKS Cost?**

Run the following Azure CLI command to check the resources still incurring charges:

>> az consumption usage list --query "[?contains(instanceName, 'aks')]" --output table

**Restarting the AKS Cluster**

When you need to start the cluster again, simply run:

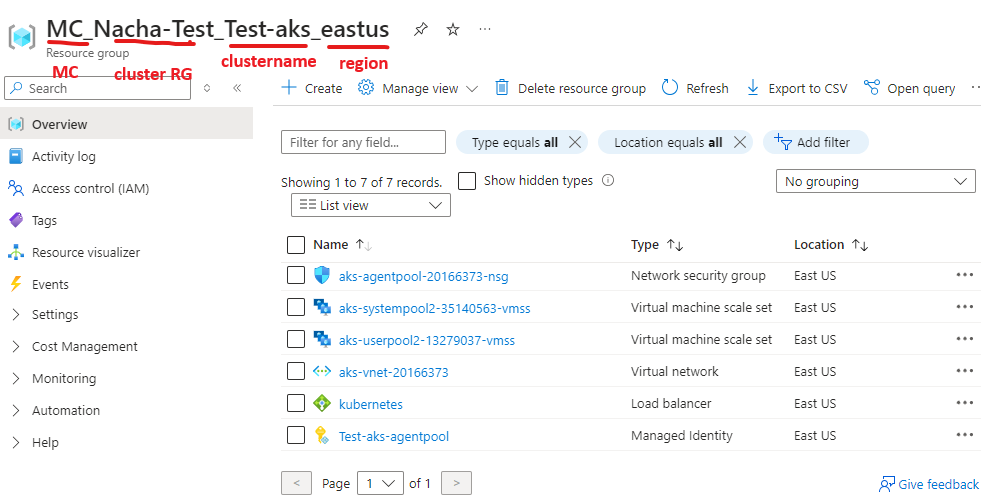
>> az aks start --resource-group <ResourceGroup> --name <ClusterName>

>> az aks stop --resource-group <ResourceGroup> --name <ClusterName>



K8S underlaying infra RG.

* It contains managed identity for the cluster, cluster network (VNET), cluster IP, node pools, load balancer. Cluster NSG.



A red line with black text

AI-generated content may be incorrect.

Note- Once application is ready go for configuring ingress controller to use as application gateway.

If using Linux workstation install az cli.

>> curl -sL https://aka.ms/InstallAzureCLIDeb | sudo bash

url- [Install the Azure CLI on Linux | Microsoft Learn](https://learn.microsoft.com/en-us/cli/azure/install-azure-cli-linux?pivots=apt)

to connect to the cluster follow connection cmd given by azure as per CLI.

A screenshot of a computer

AI-generated content may be incorrect.

Let’s configure an application to run on it.

apiVersion: apps/v1

kind: Deployment

metadata:

name: ecomm-deployment

labels:

app: ecomm

spec:

replicas: 2

selector:

matchLabels:

app: ecomm

template:

metadata:

labels:

app: ecomm

spec:

nodeSelector:

nodeselector: userpool

containers:

- name: ecomm

image: hiparthapanda297/ecomm-img

ports:

- containerPort: 80

---

apiVersion: v1

kind: Service

metadata:

name: ecomm-service

spec:

type: LoadBalancer

selector:

app: ecomm

ports:

- port: 80

targetPort: 80

Apply to create resource.

>> kubectl apply -f filename.yaml

>> kubectl get all

A screenshot of a computer screen

AI-generated content may be incorrect.

Browse the site by external IP and port. For default port 80 no need to mention port.

Note- for minikube we use nodeport Ip which is because in that technology all the pods are in a single nodes. But here for more availability we are using multiple nodes. So it’s not possible to have node IP. Even the nodes are managed by AKS so anytime they can be replaced with new node if any issue triggered.

Here aks apply the same technology which we did for high availability. Running nodes in 2 or more nodes and access via Load balancer.

Here all those managed by AKS. If new node added or old node removed those gets updated by AKS automatically.

So, we mention the service type = load balancer.

The service use loadbalancer to publish the application over internet. So this service external IP will be Load balancer’s frontend IP.

A screenshot of a computer

AI-generated content may be incorrect.

The request comes to loadbalancer then it goes to different nodes as per loadbalancing rule. From there to node to service and service to selected pods.

A diagram of a load balancer

AI-generated content may be incorrect.

We have applied Nodeselector. So all the pods should be on the same node pool.

A black screen with white text

AI-generated content may be incorrect.