**AKS Private Cluster**

**Create a Private VM:**

az vm create \

--resource-group MC\_internal-training\_aks-training\_westus \

--name aks-jumpbox \

--image UbuntuLTS \

--admin-username azureuser \

--authentication-type ssh \

--generate-ssh-keys \

--vnet-name aks-vnet-17585922 \

--subnet aks-subnet \

--public-ip-address "" \

--nsg "" \

--output table

This creates a private VM (no public IP), secured. You’ll access it using Azure Bastion

**How to Connect Private Cluster**

**Simpliest way to connect:**

az aks command invoke (No Need for SSH or Jumpbox)

az aks command invoke \

--resource-group <rg-name> \

--name <aks-cluster-name> \

--command "kubectl get pods -A"

az aks command invoke \

--resource-group <rg-name> \

--name <aks-cluster-name> \

--command "kubectl run nginx --image=nginx"

az aks command invoke \

--resource-group <rg-name> \

--name <aks-cluster-name> \

--command "kubectl get pods"

**Option 2: Create Jump Server on same Vnet**

**🔐 Why You Need a Jumpbox for Private AKS**

* **Private AKS** has no public IP on its API server.
* You can **only access it from within the same VNet** (or connected networks like via VPN or ExpressRoute).
* A **Jumpbox (or Bastion host)** is a **Linux VM** deployed inside the same VNet/subnet. You SSH into it and use kubectl from there.

**step-by-step documentation** for setting up Azure CLI, SSH access, and connecting to a private AKS cluster on a VM (Linux or Windows):

**🔧 Step-by-Step: Setup Jumpbox in Same VNet**

**🧱 Step 1: Get AKS VNet and Subnet Details**

Run this command from your local machine:

az aks show --resource-group internal-training --name aks-training --query "nodeResourceGroup" -o tsv

**Now get the VNet details:**

az network vnet list --resource-group <nodeResourceGroup> -o table

az network vnet list --resource-group MC\_internal-training\_aks-training\_westus -o table

**Now get the subnet name:**

az network vnet subnet list --resource-group MC\_internal-training\_aks-training\_westus --vnet-name aks-vnet-17585922 -o table

🖥️ Step 2: Create Jumpbox VM in Same VNet/Subnet

Use this command to create a VM:

az vm create --resource-group MC\_internal-training\_aks-training\_westus --name aks-jumpbox --image Ubuntu2204 --size Standard\_D2s\_v3 --admin-username azureuser --authentication-type ssh --generate-ssh-keys --vnet-name aks-vnet-17585922 --subnet aks-subnet --output table

az vm create \

--resource-group MC\_internal-training\_aks-training\_westus \

--name aks-jumpbox \

--image Ubuntu22.04 \

--admin-username azureuser \

--authentication-type ssh \

--generate-ssh-keys \

--vnet-name aks-vnet-17585922 \

--subnet aks-subnet \

--public-ip-address "" \

--nsg "" \

--output table

az vm create \

--resource-group <rg-name> \

--name aks-jumpbox \

--image UbuntuLTS \

--vnet-name <aks-vnet-name> \

--subnet <aks-subnet-name> \

--admin-username azureuser \

--generate-ssh-keys \

--public-ip-address-dns-name <unique-dns>

This creates a private VM (no public IP), secured. You’ll access it using Azure Bastion

This creates a **private VM** inside the AKS VNet/subnet.

Enable the Network ports. Allow the port and source IP

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**🌐 Step 3: Enable Azure Bastion (Optional but Recommended)**

If your VM has no public IP, Azure Bastion allows secure browser-based SSH:

az network bastion create \

--name aks-bastion \

--public-ip-address aks-bastion-ip \

--resource-group <nodeResourceGroup> \

--vnet-name <vnet-name-from-step-1> \

--location <region> \

--sku Basic

az network bastion create \

--name aks-bastion \

--resource-group MC\_internal-training\_azurecluster\_westus \

--vnet-name <your-vnet-name> \

--public-ip-address aks-bastion-ip \

--location westus \

--sku Basic

Then go to **VM → Connect → Bastion** in portal and login with azureuser.

🧰 Step 4: Install Required Tools (Inside VM)

SSH into the VM and run:

# Update system

sudo apt update && sudo apt install -y curl apt-transport-https ca-certificates

# Install Azure CLI

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

# Install kubectl

az aks install-cli

# Optional: Install Helm

curl https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3 | bash

**🔧 Section 1: Install Azure CLI**

**📍 On Windows (PowerShell as Administrator)**

Invoke-WebRequest -Uri https://aka.ms/installazurecliwindows -OutFile .\AzureCLI.msi

Start-Process msiexec.exe -Wait -ArgumentList '/I AzureCLI.msi /quiet'

Remove-Item .\AzureCLI.msi

📍 On Linux (VM)

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

az version

**🔐 Section 2: Configure SSH Permissions (Windows Only)**

**📍 Open PowerShell in your SSH key directory:**

cd <path-to-your-key>

**📍 Restrict PEM File Permissions:**

icacls .\azurejump-server\_key.pem /inheritance:r

icacls .\first-vm\_key.pem /grant:r "${env:USERNAME}:R"

# Remove inheritance (if not already done)

icacls "C:\04.Kubernetes\Azure\first-vm\_key.pem" /inheritance:r

# Remove overly permissive groups

icacls "C:\04.Kubernetes\Azure\first-vm\_key.pem" /remove "BUILTIN\Administrators"

icacls "C:\04.Kubernetes\Azure\first-vm\_key.pem" /remove "NT AUTHORITY\SYSTEM"

icacls "C:\04.Kubernetes\Azure\first-vm\_key.pem" /remove "NT AUTHORITY\Authenticated Users"

# Grant read access to only your current user

icacls .\first-vm\_key.pem /grant:r "${env:USERNAME}:R"

**✅ This will:**

* Remove inherited permissions
* Allow **only your user account** to read the key file

🚀 Section 3: SSH into the Azure VM

📍 Command (from Windows):

ssh -i .\azurejump-server\_key.pem [azureuser@135.13.13.230](mailto:azureuser@135.13.13.230)

**☁️ Section 4: Azure CLI Authentication and Subscription Setup**

**📍 Login to Azure:**

az login

az login --use-device-code

az account show --output table

📍 Set your active subscription:

az account set --subscription "24c4fb07-0fb5-4b37-bc45-5cb7e6e95520"

**🧰 Section 5: Clone Your Repository and Run Setup Scripts**

**📍 Clone Git Repository:**

sudo su

apt update -y

apt install git -y

git clone https://github.com/Sumanth17-git/APMTrianing.git

cd APMTraining

📍 Make Scripts Executable:

chmod +x \*

📍 Run Setup Scripts:

./setup\_ubuntu.sh

./setup\_kubectl.sh

**☸️ Section 6: Connect to AKS Cluster**

**📍 Get AKS credentials:**

az aks get-credentials --resource-group internal-training --name aks-training --overwrite-existing

📍 Verify AKS access:

kubectl get deployments --all-namespaces=true

kubectl get pods --all-namespaces=true

**Using an Azure VM**

**Alternatively, we can use an Azure VM to access an AKS private cluster. This VM must reside in the same virtual network that is used by the AKS cluster. So, let’s create a VM.**

**Step-1**

**From the Azure portal, click on ‘Create resource’ and type ‘Virtual machine’ in the search bar. Then, click on ‘Virtual machine.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Step-2**

**Click on the ‘Create’ button.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Step-3**

**In the ‘Basics’ tab select the appropriate ‘Subscription’ and the ‘Resource groups’. Azure sets the username field to ‘azureuser’ by default. This would be required to access the VM by SSH. Provide a name for the VM and select the Ubuntu 20.04 LTS image. Then, click on ‘Next’.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Step-4**

**Leave the default parameters in the ‘Disks’ screen and click ‘Next’.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Step-5**

**In the ‘Networking’ screen, select the virtual network and the subnet that we have just created during the AKS cluster deployment.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Step-6**

**Leave the default parameters in the ‘Management’, ‘Advanced’, and ‘Tags’ screens and click ‘Next’ at each.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Step-7**

**After getting the ‘Validation passed’ message click on ‘Create’.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Step-8**

**Download and store the private key that would be required to set up SSH access to the VM.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Step-9**

**A screenshot of a computer

AI-generated content may be incorrect.**

**After the deployment is completed, obtain the public IP address assigned to the VM. Login to the VM via SSH using the private key that we downloaded in the earlier step.**

**Step-10**

**Install the Azure CLI in the VM.**

**$ sudo apt-get update$ sudo apt-get install ca-certificates curl apt-transport-https lsb- release gnupg$ curl -sL https://packages.microsoft.com/keys/microsoft.asc | gpg --dearmor | sudo tee /etc/apt/trusted.gpg.d/microsoft.gpg> /dev/null$ AZ\_REPO=$(lsb\_release -cs)$ echo "deb [arch=amd64] https://packages.microsoft.com/repos/azure- cli/ $AZ\_REPO main" | sudo tee /etc/apt/sources.list.d/azure- cli.list$ sudo apt-get update$ sudo apt-get install azure-cli**

**Step-11**

**In the VM, set the credentials for ‘kubectl’ to access the cluster.**

**$ az aks get-credentials --resource-group cloud-qubes --name private-aks**

**Now, we can access our AKS cluster directly from this host.**

**$ kubectl get nodes**