

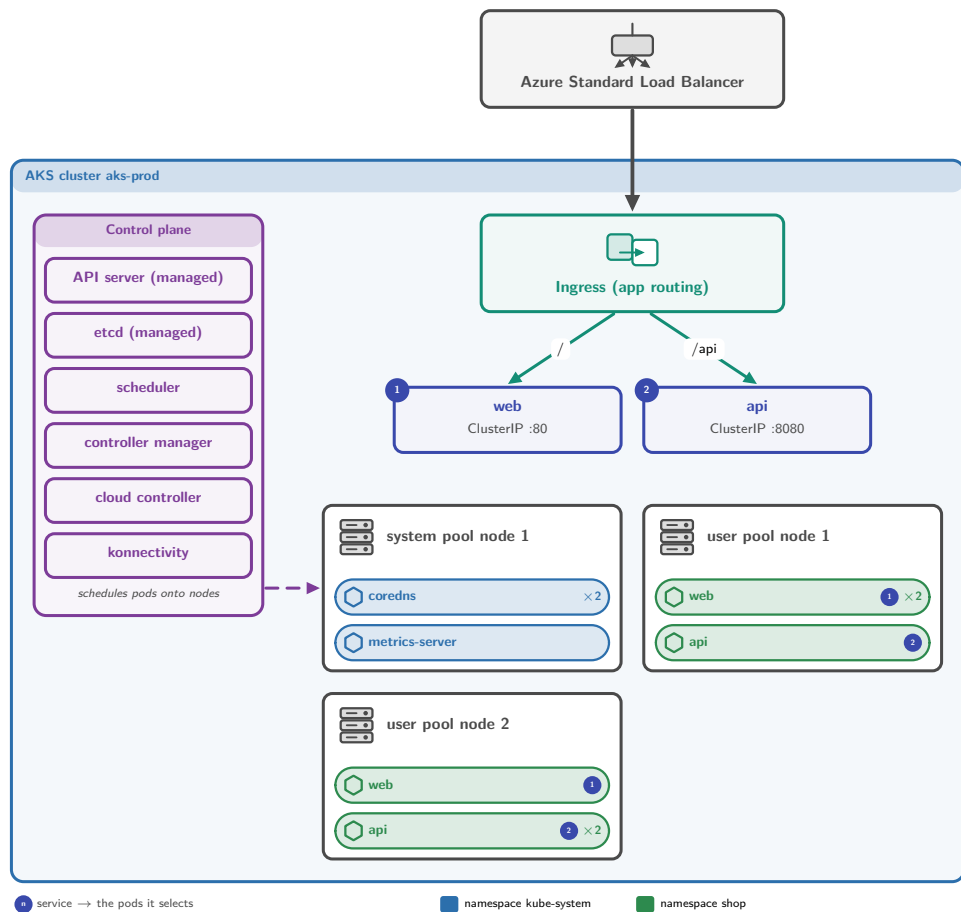


AKS Architecture and Cluster Setup

Microsoft Learn AKS documentation, as of 2026 | Control plane and pricing tiers, Kubernetes version support, system and user node pools, VM sizes and OS, availability zones, AKS Automatic vs Standard, az CLI, Bicep and Terraform

AKS Node pools AKS Automatic Availability zones Bicep

What Microsoft runs, what you run



Control plane in a Microsoft subscription; node VMs in the MC_ node resource group of yours

Control-plane pricing tiers (as of 2026)

	Free	Standard	Premium
SLA	—	99.95 % with zones, 99.9 % without	as Standard
Scale	dev/test, small clusters	up to 5,000 nodes	up to 5,000 nodes
Control-plane fee	—	per cluster-hour (approx. 0.10 USD)	higher per cluster-hour
Version support	about 12 months per minor	about 12 months per minor	LTS about 24 months
Use	labs, dev	production	slow-moving regulated estates

- **Versions** AKS supports three GA Kubernetes minor versions (N to N-2); a new minor arrives roughly every 4 months, so each is supported about 12 months
- **Upgrade path** control plane one minor at a time (1.31 → 1.32 → 1.33); node pools may lag the control plane, but not beyond the supported skew
- **Out of support** clusters on an unsupported minor still run but get no fixes; platform support only; upgrade before the end-of-life date on the AKS release calendar
- **Change the tier** `az aks update --tier standard` (or free/premium) without recreating the cluster

Node pools

Setting	System pool	User pool
Purpose	critical add-ons	your workloads
OS	Linux (Ubuntu or Azure Linux)	Linux or Windows Server
Minimum nodes	1 (3 recommended across zones for production)	0 (can scale to zero)
Taint	optional <code>CriticalAddonsOnly=true:NoSchedule</code>	your own taints (GPU, spot)
Spot VMs	no	yes (eviction Delete, taint added automatically)

VM size	general purpose, 4+ vCPUs (e.g. D4ds_v5)	per workload: D, E (memory), F (CPU), NC/ND (GPU)
Limits	max 100 pools per cluster, 1,000 nodes per pool	max pods per node set at creation (overlay default 250, Azure CNI node subnet 30, kubenet 110)

- 0

```
$ az aks nodepool add -g rg-aks --cluster-name aks-prod -n gpu
```
- 1

```
$ az aks nodepool add -g rg-aks --cluster-name aks-prod -n gpu --node-vm-size Standard_NC24ads_A100_v4
```

GPU VM family (needs quota)
- 2

```
$ az aks nodepool add -g rg-aks --cluster-name aks-prod -n gpu --node-vm-size Standard_NC24ads_A100_v4 --zones 1 2 3
```

spread nodes across zones at creation (cannot add later)
- 3

```
$ az aks nodepool add -g rg-aks --cluster-name aks-prod -n gpu --node-vm-size Standard_NC24ads_A100_v4 --zones 1 2 3 --enable-cluster-autoscaler --min-count 0 --max-count 4
```

scale to zero when no GPU pod is pending
- 4

```
$ az aks nodepool add -g rg-aks --cluster-name aks-prod -n gpu --node-vm-size Standard_NC24ads_A100_v4 --zones 1 2 3 --enable-cluster-autoscaler --min-count 0 --max-count 4 --node-taints sku=gpu:NoSchedule
```

only pods tolerating it land here
- 5

```
$ az aks nodepool add -g rg-aks --cluster-name aks-prod -n gpu --node-vm-size Standard_NC24ads_A100_v4 --zones 1 2 3 --enable-cluster-autoscaler --min-count 0 --max-count 4 --node-taints sku=gpu:NoSchedule --labels workload=training
```

target with nodeSelector or affinity
- 6

```
$ az aks nodepool add -g rg-aks --cluster-name aks-prod -n gpu --node-vm-size Standard_NC24ads_A100_v4 --zones 1 2 3 --enable-cluster-autoscaler --min-count 0 --max-count 4 --node-taints sku=gpu:NoSchedule --labels workload=training --max-pods 50
```

fixed at creation; drives IP use
- 7

```
$ az aks nodepool add -g rg-aks --cluster-name aks-prod -n gpu --node-vm-size Standard_NC24ads_A100_v4 --zones 1 2 3 --enable-cluster-autoscaler --min-count 0 --max-count 4 --node-taints sku=gpu:NoSchedule --labels workload=training --max-pods 50 --os-sku AzureLinux
```

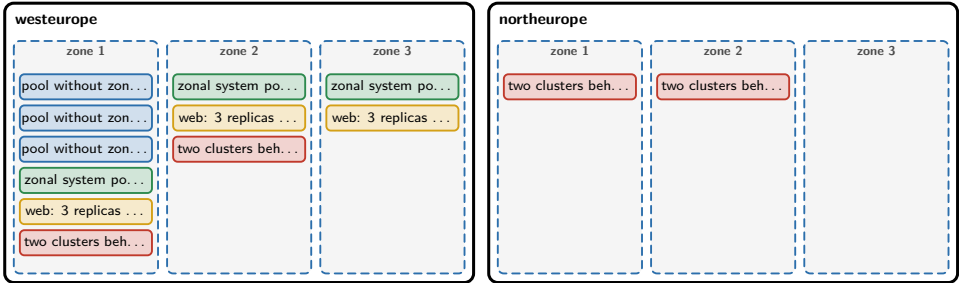
lighter host OS

A GPU POOL THAT COSTS NOTHING IDLE

```
$ az aks nodepool add -g rg-aks --cluster-name aks-prod -n gpu --node-vm-size Standard_NC24ads_A100_v4 --zones 1 2 3 --enable-cluster-autoscaler --min-count 0 --max-count 4 --node-taints sku=gpu:NoSchedule --labels workload=training --max-pods 50 --os-sku AzureLinux
```

```
Pin a pod to the GPU pool
spec:
  nodeSelector:
    workload: training
  tolerations:
  - key: "sku"
    operator: "Equal"
    value: "gpu"
    effect: "NoSchedule"
  containers:
  - name: trainer
    image: acrprod.azurecr.io/trainer:1.4
    resources:
      limits: {nvidia.com/gpu: 1}
```

Availability zones and resilience



Deployment	zone 1 down	zones 1 + 2 down	region down
pool without zones (1 zone in practice) 3 replicas, needs 2	fails 0/3 left	fails 0/3 left	fails 0/3 left
zonal system pool 3 nodes 3 replicas, needs 2	degraded 2/3 left	fails 1/3 left	fails 0/3 left
web: 3 replicas + topology spread 3 replicas, needs 2	degraded 2/3 left	fails 1/3 left	fails 0/3 left
two clusters behind Front Door 4 replicas, needs 2	degraded 3/4 left	degraded 2/4 left	degraded 2/4 left

Survives = no replica lost; degraded = replicas lost but at least the minimum still healthy; fails = fewer healthy replicas than the minimum. A regional service survives any zone loss but not the loss of its region.

- **Zones at creation only** --zones 1 2 3 on each node pool; existing pools cannot be made zonal, create a new pool and drain
- **Spread pods** topologySpreadConstraints on topology.kubernetes.io/zone with maxSkew: 1; PodDisruptionBudgets keep replicas during drains
- **Storage follows zones** Azure Disk (LRS) volumes are zonal: a pod cannot move to another zone with its disk; use ZRS disks or Azure Files for cross-zone
- **Region failure** one AKS cluster is regional; survive a region loss with a second cluster and Azure Front Door or Traffic Manager (and AKS Fleet Manager to coordinate)

🔗 Create the cluster as code

aks.bicep: production-shaped Standard cluster

```
resource aks 'Microsoft.ContainerService/managedClusters@2025-05-01' = {
  name: 'aks-prod'
  location: location
  sku: { name: 'Base', tier: 'Standard' }
  identity: { type: 'SystemAssigned' }
  properties: {
    kubernetesVersion: '1.33'
    dnsPrefix: 'aksprod'
    agentPoolProfiles: [
      {
        name: 'system', mode: 'System', count: 3
        vmSize: 'Standard_D4ds_v5', osSKU: 'AzureLinux'
        availabilityZones: [ '1', '2', '3' ]
        vnetSubnetID: nodeSubnetId
        nodeTaints: [ 'CriticalAddonsOnly=true:NoSchedule' ]
      }
    ]
    networkProfile: { networkPlugin: 'azure',
      networkPluginMode: 'overlay', networkDataplane: 'cilium' }
    aadProfile: { managed: true, enableAzureRBAC: true }
    disableLocalAccounts: true
    oidcIssuerProfile: { enabled: true }
  }
  securityProfile: { workloadIdentity: { enabled: true } }
  autoUpgradeProfile: { upgradeChannel: 'patch',
    nodeOSUpgradeChannel: 'NodeImage' }
}
```

AKS Standard

you design node pools, sizes, zones

choose CNI, ingress, policies

opt in to workload identity, KEDA, VPA
upgrades as you configure
any customisation

AKS Automatic

node autoprovisioning picks VM sizes per pending pod

Azure CNI Overlay + Cilium, managed ingress and egress defaults

enabled by default
automatic upgrades in maintenance windows
deployment safeguards enforce best practices; fewer knobs

📄 Worked example: size the system pool

System pods (CoreDNS, metrics-server, connectivity, CSI drivers, Azure Monitor agents) request about 1.5 vCPU and 3 GiB per node in a typical cluster (check `kubectl describe node`). On D4ds_v5 nodes (4 vCPU, 16 GiB) AKS reserves part of each node for kubelet and OS, leaving roughly 3.3 vCPU allocatable: 3 nodes give about 10 vCPU, enough for system pods plus headroom during a zone loss (2 nodes = 6.6 vCPU > 1.5 × 3 = 4.5). A 2-vCPU size would leave too little room.

🚩 Recap and self-test

MYTH

❌ You can add availability zones to an existing node pool

❌ The Free tier is fine for production

❌ You may edit resources in the MC_ group freely

FACT

✅ Zones are fixed at pool creation
Create a zonal pool, cordon and drain the old one, delete it.

✅ Free has no financially backed SLA
Use Standard (99.95 % with zones) for production.

✅ AKS manages that group
Manual changes can break reconciliation and upgrades; node resource group lockdown can forbid them.

Q1 Three AKS pricing tiers?

A Free, Standard (uptime SLA), Premium (Standard + long-term support).

Q2 Uptime SLA of Standard with availability zones?

A 99.95 % (99.9 % without zones).

Q3 How many Kubernetes minor versions does AKS support at once?

A Three GA minors (N, N-1, N-2), each roughly 12 months.

Q4 Can you upgrade from 1.31 straight to 1.33?

A No, one minor version at a time.

Q5 Which pool type can scale to zero?

A User node pools (system pools need at least one node).

Q6 Taint that keeps app pods off a system pool?

A CriticalAddonsOnly=true:NoSchedule.

Q7 Name of the resource group holding node VMs?

A MC_<resourceGroup>_<cluster>_<region> (customisable at creation).

Q8 Which AKS mode autoprovisions node sizes and enables Cilium by default?

A AKS Automatic.

Q9 Why prefer ZRS or Azure Files for stateful pods across zones?

A LRS Azure Disks are zonal; a pod cannot follow its disk to another zone.

