AZURE KUBERNETES SERVICE / FIELD GUIDE

Production-ready. By design.

The AKS Checklist

An evidence-led guide to architecting, securing, and operating resilient Kubernetes platforms on Azure.

202 practical checks

15 review areas

5 Well-Architected pillars

November 2026
Guidance reviewed 2026-11-05

Community maintained · Microsoft guidance · Independent project

BEFORE YOU BEGIN

A stronger baseline, not a rubber stamp.

Use this guide to make architecture decisions explicit. Select applicable checks, capture evidence, and assign owners to risks and exceptions. Completion is not a security certification or an application availability guarantee.

Guidance applies to AKS Automatic and Standard where supported. Validate your region, cloud, Kubernetes version, operating system, and feature support. Preview capabilities are not a production default. Always verify current Microsoft documentation before implementation.

> **Lifecycle watch / November 2026**
>
> Ingress migration is time-sensitive — Upstream Ingress NGINX maintenance ended in March 2026. AKS application routing NGINX critical-security-patch support runs through November 2026. Plan and test a supported Gateway API migration now.
>
> Retired is not production-ready — Replace Azure Linux 2.0, Windows Server 2019, and the retired pod-managed identity add-on. Upgrade Basic Load Balancer and Availability Sets. Migrate kubenet before 31 March 2028.

How to use this guide

1. Define business requirements, platform ownership, service objectives, and trust boundaries.
2. Review every applicable check with the workload and platform teams.
3. Record implementation evidence or a time-bound exception with an owner.
4. Test failure, upgrade, and recovery paths; revisit the review after material changes.

Priority: High highlights foundational or time-sensitive controls; Medium covers important design and operational practices; Low is generally context-dependent. Priority is not a vulnerability severity rating.

Review map

01	Architecture & governance	7 checks
02	Identity & access	10 checks
03	Cluster security	17 checks
04	Multi-tenancy & isolation	3 checks
05	Networking & ingress	25 checks
06	Storage & data	10 checks
07	Resources & scaling	15 checks
08	Cluster operations	24 checks
09	Lifecycle & retirements	10 checks
10	Observability & incident response	13 checks
11	Reliability & disaster recovery	14 checks
12	Windows workloads	13 checks
13	Application delivery	26 checks
14	Images & supply chain	10 checks
15	Cost & sustainability	5 checks

REVIEW AREA 01 / 15

Architecture & governance

Start with business outcomes, shared responsibility, and a deliberate platform design.

1.01 Choose AKS only when Kubernetes fits the workload

High

Compare AKS with Azure Container Apps, App Service, and other managed platforms against portability, orchestration, networking, team skills, operational cost, and compliance needs. Record the decision and the workload owner before provisioning a cluster.

READ MORE

Choose AKS only when Kubernetes fits the workload

1.02 Choose AKS Automatic or Standard deliberately

High

Evaluate AKS Automatic for managed production defaults and Standard for infrastructure customization. Verify feature, region, networking, Windows, and operational constraints. Validate service-managed defaults rather than assuming either mode makes applications production-ready.

READ MORE

Choose AKS Automatic or Standard deliberately

1.03 Review the platform against the AKS Well-Architected baseline

High

Review reliability, security, cost optimization, operational excellence, and performance efficiency using the AKS baseline architecture and Well-Architected guidance. Record design trade-offs, owners, residual risks, and evidence for each applicable control.

READ MORE

Review the platform against the AKS Well-Architected baseline

1.04 Document shared responsibility and operational ownership

High

Define ownership for cluster configuration, workload patching, identities, data protection, upgrades, monitoring, incident response, and costs. AKS manages the control plane; customers still own application resilience, workload configuration, and service dependencies.

READ MORE

Document shared responsibility and operational ownership

1.05 Use landing-zone governance and explicit environment boundaries

Medium

Place clusters in the intended management group, subscription, and network architecture. Define Azure Policy, resource naming and tags, budget ownership, data residency, and separation of production from nonproduction access and change paths.

READ MORE

Use landing-zone governance and explicit environment boundaries

1.06 Define evidence-based review outcomes and exceptions

High

For each applicable check, record an owner, evidence, status, and a time-bound remediation or risk acceptance. Priorities are review guidance, not a universal risk score. Explicitly mark nonapplicable items and revisit the assessment after major changes.

READ MORE

Define evidence-based review outcomes and exceptions

1.07 Gate preview features and verify regional availability

High

Prefer generally available supported features for production. Review preview terms, SLA exclusions, region and cloud availability, Kubernetes minimums, and exit plans before any approved exception. Feature names alone do not establish production support.

READ MORE

Gate preview features and verify regional availability

REVIEW AREA 02 / 15

Identity & access

Use Microsoft Entra ID, least privilege, and secretless access.

2.01 Use AKS-managed Microsoft Entra authentication

High

Integrate cluster sign-in with AKS-managed Microsoft Entra ID. Use group-based access, Conditional Access where applicable, and kubelogin; do not use the retired legacy Entra integration.

READ MORE

Use AKS-managed Microsoft Entra authentication

2.02 Choose and enforce least-privilege cluster authorization

High

Choose Azure RBAC for Kubernetes authorization or Kubernetes RBAC with Microsoft Entra groups. Distinguish Azure management-plane roles from in-cluster data-plane permissions, avoid routine cluster-admin access, and test denied operations.

READ MORE

Microsoft Entra ID with Kubernetes RBAC

Microsoft Entra ID authorization for the Kubernetes API

2.03 Pull images from ACR using the kubelet managed identity

Medium

Grant the kubelet identity AcrPull for a non-ABAC registry. For an ABAC-enabled registry, assign Container Registry Repository Reader at the appropriate scope; --attach-acr is not supported in that permissions mode. Keep the ACR admin account disabled.

READ MORE

Pull images from ACR using the kubelet managed identity

2.04 Use managed identities for cluster infrastructure

High

Use managed identities rather than service principals for the AKS control plane and kubelet. Scope permissions separately; workloads must use their own federated identities, not inherit node permissions.

READ MORE

Use managed identities for cluster infrastructure

2.05 Limit access to admin kubeconfig (get-credentials --admin)

Medium

To limit who can get that Kubernetes configuration (kubeconfig) information and to limit the permissions they then have, you can use Azure role-based access control (Azure RBAC).

READ MORE

Use Azure RBAC to define access to the Kubernetes configuration file

2.06 Use kubelogin and federated identities for automation

Medium

Use the supported kubelogin authentication flow for Microsoft Entra-integrated clusters. Prefer workload identity federation or managed identity for CI/CD instead of long-lived client secrets, and validate noninteractive access from the deployment runner.

READ MORE

Use kubelogin and federated identities for automation

2.07 Disable local accounts and test emergency access

Medium

Disable local Kubernetes accounts on Microsoft Entra-integrated clusters. Document an audited emergency-access procedure, restrict who can change the setting, and rotate previously issued cluster certificates where required.

READ MORE

Disable local accounts and test emergency access

2.08 Use time-bound privileged access where required

Low

Use Microsoft Entra Privileged Identity Management for eligible group or Azure role assignments. Require approval and MFA for privileged operations, log activation, and test access expiry.

READ MORE

Use time-bound privileged access where required

2.09 Pre-create a kubelet identity when role assignment requires it

Low

Use a pre-created user-assigned kubelet identity when permissions must exist before cluster creation. Grant only the registry and infrastructure access it needs; account for role-assignment propagation delays.

READ MORE

Pre-create a kubelet identity when role assignment requires it

2.10 Apply Conditional Access to interactive cluster access

Low

Evaluate Microsoft Entra Conditional Access for administrators, including MFA and device requirements. Test the chosen kubelogin flow and document separate controls for workload and automation identities.

READ MORE

Apply Conditional Access to interactive cluster access

REVIEW AREA 03 / 15

Cluster security

Harden the platform and enforce a measurable security baseline.

3.01 Map regulated workloads to current compliance requirements

Medium

Map controls to the current regulatory standard, such as PCI DSS 4.x, and validate the AKS shared-responsibility model. Use supported FIPS-enabled images where mandated; an architecture example or Azure certification does not itself make an application compliant.

READ MORE

Map regulated workloads to current compliance requirements

3.02 Assess customer-managed encryption keys and their lifecycle

Low

AKS encrypts data at rest using platform protection. If customer-managed keys are required, assess supported KMS options and Key Vault availability, permissions, purge protection, rotation, and recovery. The newer KMS data-encryption experience is documented as preview; do not treat it as a production default.

Distinguish legacy etcd KMS from the newer KMS data-encryption configuration. Check Kubernetes version, regional availability, and support status before enabling either; loss of a required key can make encrypted secrets unrecoverable.

READ MORE

Assess customer-managed encryption keys and their lifecycle

3.03 Enforce risk-based image admission controls

Medium

Scan images in CI and continuously in the registry. Enforce approved registries, digest-based releases, and risk policy through supported admission controls. Verify the support status and actual enforcement of any vulnerability-aware integration, define time-bound exceptions with owners, and do not assume that registry scanning alone blocks deployment.

READ MORE

Enforce risk-based image admission controls

3.04 Integrate Key Vault using the Secrets Store CSI driver

High

Use the AKS Azure Key Vault provider with Microsoft Entra Workload ID. Enable rotation where required and test that applications reload mounted secrets. Environment variables do not automatically refresh; synchronizing to a Kubernetes Secret creates an additional stored copy.

READ MORE

Integrate Key Vault using the Secrets Store CSI driver

3.05 Monitor cloud security posture with Microsoft Defender for Cloud

Medium

Review Defender for Cloud recommendations and assign remediation owners. Connect security posture to Azure Policy and your incident workflow; distinguish posture assessment from the paid Defender for Containers runtime protection plan.

READ MORE

Monitor cloud security posture with Microsoft Defender for Cloud

3.06 Clean unused cached images without confusing cleanup with remediation

High

Evaluate AKS Image Cleaner for unused images on nodes. It does not patch images or remove vulnerable containers that are currently running: rebuild the image and redeploy affected workloads to remediate those risks.

READ MORE

Clean unused cached images without confusing cleanup with remediation

3.07 Enable Microsoft Defender for Containers

High

Microsoft Defender for Containers provides protections for your Kubernetes clusters wherever they're running (AKS and on-premises)

READ MORE

Introduction to Microsoft Defender for Containers

3.08 Use Azure Policy for Kubernetes to ensure cluster compliance

Medium

Azure Policy integrates with the Azure Kubernetes Service (AKS) to apply at-scale enforcements and safeguards on your clusters in a centralized, consistent manner.

READ MORE

Azure Policies for AKS

Azure Kubernetes Services Governance with Azure Policy

Gatekeeper

3.09 Separate system and application node pools

Medium

In AKS Standard, dedicate appropriately sized Linux system pools to critical cluster components and run applications on user pools. Use the CriticalAddonsOnly taint where appropriate and provide production redundancy. System pools are not the managed control plane.

READ MORE

Separate system and application node pools

3.10 Migrate legacy service principals and rotate remaining credentials

High

Migrate cluster infrastructure to managed identities. Where a legacy service principal remains, rotate credentials before expiry under an audited policy and test the change; a fixed quarterly interval alone does not manage credential risk.

READ MORE

Migrate legacy service principals and rotate remaining credentials

3.11 Use a private registry for your images, such as ACR

High

Azure Container Registry allows you to build, store, and manage container images and artifacts in a private registry

READ MORE

Azure Container Registry documentation

3.12 Continuously assess cluster security configuration

Medium

Run supported CIS-aligned configuration assessments and review drift in RBAC, network exposure, admission policy, and workload permissions. Record accepted risks and remediation dates rather than treating an assessment score as proof of compliance.

READ MORE

Continuously assess cluster security configuration

3.13 Evaluate confidential computing for specific threat models

Low

Evaluate the supported AKS confidential-computing options against your workload and compliance needs. Distinguish application enclaves from confidential virtual machines and confidential containers; verify SKU, region, attestation, OS, and feature support rather than assuming every container automatically runs inside an enclave.

READ MORE

Application enclave support on AKS

3.14 Define app separation requirements (namespace/nodepool/cluster)

High

Plan for multi-tenant clusters and separation of resources and use logical or physical isolation in your AKS clusters

READ MORE

Best practices for cluster isolation in AKS

3.15 Evaluate Dedicated Hosts only for explicit isolation requirements

Low

Use Azure Dedicated Hosts when dedicated physical capacity is required and supported for the node configuration. Plan host capacity, failure domains, maintenance, and additional cost; this does not remove application-level security responsibilities.

READ MORE

Evaluate Dedicated Hosts only for explicit isolation requirements

3.16 Enforce Pod Security Standards and least-privilege containers

High

Apply Pod Security Admission or equivalent Azure Policy controls. Progress from audit to enforcement; for Linux workloads prefer restricted settings, non-root users, seccomp RuntimeDefault, dropped capabilities, read-only filesystems, and no privilege escalation. Document narrowly scoped exceptions.

READ MORE

Enforce Pod Security Standards and least-privilege containers

3.17 Protect node metadata and minimize node identity permissions

High

Minimize node managed-identity permissions and use federated workload identities. AKS IMDS restriction is documented as preview, applies to non-host-network pods, and requires assessment of support terms and compatibility. Do not assume a generic NetworkPolicy blocks all metadata or host-network access.

READ MORE

Protect node metadata and minimize node identity permissions

REVIEW AREA 04 / 15

Multi-tenancy & isolation

Match isolation boundaries to trust, ownership, and regulatory requirements.

4.01 Use logical isolation for mutually trusted tenants

Medium

Combine namespaces, scoped RBAC, resource quotas, admission policy, and network policies for trusted tenants. A namespace alone is not a security boundary and does not isolate cluster-scoped resources or a shared kernel.

READ MORE

Use logical isolation for mutually trusted tenants

4.02 Use separate clusters for incompatible trust boundaries

High

Choose separate clusters or subscriptions where tenants are untrusted, compliance demands stronger isolation, or lifecycle and blast-radius requirements differ. Balance operational cost against the risk of a shared control plane and nodes.

READ MORE

Use separate clusters for incompatible trust boundaries

4.03 Use Azure tags in Azure Kubernetes Service (AKS)

Medium

With Azure Kubernetes Service (AKS), you can set Azure tags on an AKS cluster and its related resources by using Azure Resource Manager, through the Azure CLI.

READ MORE

Use Azure tags in Azure Kubernetes Service

Use Azure tags in Azure Kubernetes Service (AKS)

REVIEW AREA 05 / 15

Networking & ingress

Design address space, private connectivity, ingress, and outbound dependencies.

5.01 Choose Azure CNI Overlay or Pod Subnet deliberately

Medium

Prefer Azure CNI Overlay for efficient address use, or Azure CNI Pod Subnet when directly routable pod addresses are required. Validate OS and policy-engine compatibility. Kubenet is a legacy model retiring on 31 March 2028; do not select it for new deployments.

READ MORE

Choose Azure CNI Overlay or Pod Subnet deliberately

5.02 Plan node and pod address capacity including upgrade surge

High

Size address space for the actual network model, maximum nodes, maxPods, scale-out, and concurrent surge upgrades. Overlay separates pod and node address space; flat networking consumes VNet pod addresses. Include private endpoints, load balancers, peering, and future growth.

READ MORE

Plan node and pod address capacity including upgrade surge

5.03 Choose supported L7 ingress and plan Gateway API migration

Medium

Use a supported ingress or Gateway API implementation for HTTP routing, TLS, and policy. Upstream Ingress NGINX maintenance ended in March 2026; AKS application routing NGINX critical-security-patch support runs through November 2026. Evaluate application routing Gateway API, Application Gateway for Containers, or Istio ingress against feature and support requirements.

READ MORE

Choose supported L7 ingress and plan Gateway API migration

5.04 Protect internet-facing HTTP applications with an appropriate WAF

Medium

Choose a supported WAF architecture, such as Azure Front Door Premium or Application Gateway WAF, when required by the threat model. Tune managed rules, restrict origin access, validate TLS and health probes, and test that the origin cannot bypass the WAF.

READ MORE

Protect internet-facing HTTP applications with an appropriate WAF

5.05 Restrict ingress hostnames and Gateway API delegation

Low

Constrain approved hostnames, cross-namespace references, and who can attach routes to shared gateways. Use admission controls and Gateway API allowedRoutes/ReferenceGrant rules as appropriate; test that tenants cannot claim another application hostname.

READ MORE

Restrict ingress hostnames and Gateway API delegation

5.06 Expose only the load balancers that need public access

High

Use internal load balancers for private services and public endpoints only for intentional internet-facing applications. An internal load balancer still needs authentication, network controls, DNS, and a tested access path.

READ MORE

Expose only the load balancers that need public access

5.07 Enforce pod network policy using a supported engine

Medium

Apply and test ingress and egress policies with explicit DNS and dependency allowances. Azure CNI Powered by Cilium is recommended for supported Linux configurations; validate Windows support separately. A configured policy engine without effective policies does not isolate workloads.

READ MORE

Enforce pod network policy using a supported engine

5.08 Configure default network policies in each namespace

Medium

Start by creating a deny all policy in each namespace and then add specific policies.

READ MORE

Recipes of best default network policies

Calico global network policy

5.09 Filter egress traffic with AzFW/NVA if your security requirements mandate it (egress lockdown)

Medium

Use Azure Firewall to secure and control all egress traffic going outside of the cluster.

READ MORE

Egress traffic requirements

5.10 Don't expose your container registry on Internet

Low

When possible, use private link to only allow private network to reach your registry.

READ MORE

Azure Container Registry Private Link

5.11 Test workload access to metadata and host-network services

Medium

Explicitly test access to 169.254.169.254 and host-network workloads from ordinary application pods. Network policy behavior depends on the data plane and host-network semantics. AKS IMDS restriction is documented as preview and does not restrict host-network pods; assess support terms before adoption and minimize node identity permissions.

READ MORE

Test workload access to metadata and host-network services

5.12 Use private clusters if your requirements mandate it

High

In a private cluster, the control plane or API server has internal IP addresses and is not exposed to Internet

Plan private DNS and an approved network path for administrators and deployment runners, such as a connected management host, VPN, or ExpressRoute. Azure Bastion provides access to a management VM rather than acting as a direct Kubernetes API proxy. Review command invoke permissions separately because that management-plane path can reach a private cluster without direct client network connectivity.

READ MORE

Create a private cluster (using private link)

API Server VNet Integration

Use azure CLI to run command on a private cluster

Use public DNS with a private cluster

5.13 Choose global routing based on protocol and failover requirements

Medium

Use Azure Front Door for global HTTP(S) edge routing and WAF integration, or Traffic Manager for DNS-based routing of supported endpoints. Configure health checks, TLS, DNS TTLs, origin restrictions, and failover tests; neither service automatically replicates application state.

READ MORE

Choose global routing based on protocol and failover requirements

5.14 Set maxPods explicitly for the chosen networking model

High

Check the default and supported maximum pod density for the actual OS, network model, VM size, and provisioning method. Do not assume a universal default of 30. Account for DaemonSets, system reservations, IP capacity, and storage attachment limits.

READ MORE

Set maxPods explicitly for the chosen networking model

5.15 If using a public API endpoint, restrict the IP addresses that can access it

Medium

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.

READ MORE

Secure access to the API server using authorized IP address ranges

5.16 Choose and monitor an explicit outbound connectivity design

Medium

Choose a supported outboundType such as NAT gateway, load balancer, or user-defined routing based on throughput and inspection requirements. Size SNAT capacity and allow AKS dependencies. NAT Gateway is not a firewall; private clusters still need an explicit egress plan.

READ MORE

Choose and monitor an explicit outbound connectivity design

5.17 Evaluate Pod Subnet allocation when flat pod routing is needed

Medium

For VNet-routable pod IPs, evaluate Azure CNI Pod Subnet dynamic or static block allocation against supported features and scale limits. Plan separate node and pod subnet capacity; use Overlay when direct pod reachability is not required.

READ MORE

Evaluate Pod Subnet allocation when flat pod routing is needed

5.18 Use private endpoints or service endpoints where appropriate

Medium

For PaaS dependencies, select private endpoints or supported service endpoints based on service capability and trust boundaries. Configure private DNS and public-access restrictions explicitly; a private AKS API endpoint does not make workload dependencies private.

READ MORE

Use private endpoints or service endpoints where appropriate

5.19 Adopt a service mesh only for explicit application requirements

Medium

Evaluate a supported mesh, such as the AKS Istio add-on, when service identity, mutual TLS, traffic policy, or detailed telemetry justify its operational cost. Validate supported revisions, upgrades, resource overhead, and failure behavior; a mesh is not a prerequisite for every microservice application.

READ MORE

About service meshes

5.20 Use a bring-your-own CNI only with clear support ownership

Low

Use bring-your-own CNI only when its capabilities are necessary and the team can operate and support it. Review AKS support boundaries, Kubernetes and OS compatibility, IPAM, policy enforcement, upgrades, and recovery; Azure does not operate a third-party CNI on your behalf.

READ MORE

Bring your own Container Network Interface (CNI) plugin for AKS

5.21 Use per-node-pool subnets when the network design requires them

Medium

Evaluate separate node-pool subnets for address-space organization, routing, or security requirements. Validate support for the selected Azure CNI mode and subnet delegation, private DNS, and policy behavior; subnet separation alone is not tenant isolation.

READ MORE

Add a node pool with a unique subnet

5.22 Reserve nonoverlapping service, pod, node, and connected-network ranges

Medium

Validate service, pod, node-subnet, peered-VNet, VPN, and on-premises address ranges before deployment. Apply the rules of the selected network model, including Overlay routing and masquerading behavior, and test reachability to all dependencies. Address conflicts can make destinations unreachable.

READ MORE

Pod CIDR, Service CIDR and Subnet CIDR

Use the right CIDR ranges for networking parameters

5.23 Validate DNS resolution and private-zone ownership

High

Test cluster DNS, upstream DNS, private endpoints, and cross-network name resolution from pods and deployment runners. Monitor DNS latency and failures, scale CoreDNS when needed, and evaluate LocalDNS only against current feature availability and workload requirements.

READ MORE

Validate DNS resolution and private-zone ownership

5.24 Evaluate Azure CNI Powered by Cilium for Linux data planes

Medium

Use the supported managed Cilium data plane for compatible Linux workloads. Validate NetworkPolicy behavior and migration constraints; advanced L7, FQDN, and observability features can require Advanced Container Networking Services and additional cost.

READ MORE

Evaluate Azure CNI Powered by Cilium for Linux data planes

5.25 Manage TLS certificate issuance and renewal

High

Inventory external and internal certificates, automate issuance and renewal through a supported integration, and alert before expiry. Validate trust chains, TLS policy, secret rotation, and application reload behavior without relying on manual certificate replacement.

READ MORE

Manage TLS certificate issuance and renewal

REVIEW AREA 06 / 15

Storage & data

Choose durable storage deliberately and test how data survives disruption.

6.01 Choose storage by durability, access mode, and performance

High

Choose Azure Disk, Azure Files, object storage, or an external data service based on latency, throughput, IOPS, concurrent access, consistency, backup, and recovery requirements. Validate the selected CSI driver and storage tier under realistic load.

READ MORE

AKS storage concepts

6.02 Size the nodes for storage needs

Medium

Each node size supports a maximum number of disks. Different node sizes also provide different amounts of local storage and network bandwidth. Plan for your application demands to deploy the appropriate size of nodes.

Different types and sizes of nodes are available. Each node (underlying VM) size provides a different amount of core resources such as CPU and memory. These VM sizes have a maximum number of disks that can be attached. Storage performance also varies between VM sizes for the maximum local and attached disk IOPS (input/output operations per second). If your applications require Azure Disks as their storage solution, plan for and choose an appropriate node VM size. The amount of CPU and memory isn't the only factor when you choose a VM size. The storage capabilities are also important.

READ MORE

Size the nodes for storage needs

6.03 Use CSI storage classes with deliberate reclaim policies

Low

Use supported CSI drivers and dynamic provisioning where appropriate. Set access modes, expansion, binding mode, encryption, and Delete versus Retain reclaim policy deliberately; deleting a PVC can delete the underlying volume under a Delete policy.

READ MORE

Use CSI storage classes with deliberate reclaim policies

6.04 Back up persistent application data with a supported solution

Medium

Use Azure Backup for AKS or another supported, workload-aware backup solution. Confirm supported volume types, access modes, region, retention, and restore paths. Azure Site Recovery is not a general AKS cluster backup solution; snapshots alone may not be application-consistent.

READ MORE

Back up persistent application data with a supported solution

6.05 Avoid keeping state in the cluster, and store data outside (AzStorage, AzSQL, Cosmos, etc)

High

Where possible, don't store service state inside the container. Instead, use an Azure platform as a service (PaaS) that supports multiregion replication.

Service state refers to the in-memory or on-disk data that a service requires to function. State includes the data structures and member variables that the service reads and writes. Depending on how the service is architected, the state might also include files or other resources that are stored on the disk. For example, the state might include the files a database uses to store data and transaction logs.

READ MORE

Remove service state from inside containers

6.06 Use ephemeral OS disks only for disposable node state

High

Evaluate ephemeral OS disks for supported VM sizes and node-image requirements. Local OS-disk data is disposable across reimage or node replacement; keep durable workload data on persistent storage or external services and validate capacity for images and logs.

READ MORE

Use ephemeral OS disks only for disposable node state

6.07 For hyper performance storage option use Ultra Disks on AKS

Low

Azure ultra disks offer high throughput, high IOPS, and consistent low latency disk storage for your stateful applications.

One major benefit of ultra disks is the ability to dynamically change the performance of the SSD along with your workloads without the need to restart your agent nodes. Ultra disks are suited for data-intensive workloads.

READ MORE

Use Azure ultra disks on AKS

6.08 Align persistent volumes with availability-zone placement

Medium

Use WaitForFirstConsumer where appropriate so volume provisioning follows scheduling constraints. LRS disks are zonal; evaluate supported ZRS disks for cross-zone attachment requirements. Test failover and attach times, because a replicated disk alone does not guarantee application availability.

READ MORE

Align persistent volumes with availability-zone placement

6.09 Plan managed OS disk capacity and performance

High

Size managed OS disks and node storage for image layers, writable layers, temporary files, and logs. Monitor disk pressure and log rotation. Larger disk size alone is not a universal performance fix; validate the selected disk tier and VM I/O limits.

READ MORE

Azure disk availability zone support

6.10 Migrate deprecated in-tree storage drivers to CSI

High

Inventory PVs, StorageClasses, and automation for in-tree Azure Disk and Azure File provisioners. Use supported CSI drivers, test snapshots and restores, and review migration behavior before a Kubernetes upgrade.

READ MORE

Migrate deprecated in-tree storage drivers to CSI

REVIEW AREA 07 / 15

Resources & scaling

Right-size workloads and scale nodes and applications without surprises.

7.01 Evaluate virtual nodes only for compatible burst workloads

Low

Virtual nodes use Azure Container Instances and have different networking, identity, storage, OS, and feature constraints from regular AKS nodes. Verify the current support matrix, cold-start behavior, regional quota, and cost before adopting this burst path.

READ MORE

Burst to Azure Container Instances

7.02 Size nodes against measured workload and failure capacity

Medium

Choose node sizes and counts using CPU, memory, pod density, network throughput, storage limits, and failure-domain needs. Reserve capacity for system components and upgrades, and benchmark cost and latency rather than optimizing only VM price.

READ MORE

Choosing a worker node size

Choose the right storage type

Kubernetes instance calculator

7.03 Evaluate Arm64 with workload and add-on compatibility tests

Medium

Benchmark supported Arm64 VM sizes for price/performance. Publish multi-architecture images and verify every sidecar, DaemonSet, driver, and security agent supports the target architecture before scheduling workloads there.

READ MORE

Evaluate Arm64 with workload and add-on compatibility tests

7.04 Enforce resource quotas for namespaces

Low

Plan and apply resource quotas at the namespace level. If pods don't define resource requests and limits, reject the deployment. Monitor resource usage and adjust quotas as needed.

Resource requests and limits are placed in the pod specification. These limits are used by the Kubernetes scheduler at deployment time to find an available node in the cluster. But developers can forget them and thus impact other applications by over-consuming resources of the cluster

READ MORE

Enforce resource quotas

Resources quotas

7.05 Namespaces should have LimitRange

Low

A LimitRange is a policy to constrain resource allocations (to Pods or Containers) in a namespace. It's useful to ensure that pods don't forget to declare request limits

READ MORE

LimitRange

7.06 Set measured resource requests and deliberate limits

Medium

Set CPU and memory requests based on observed demand; requests drive scheduling and autoscaling. Set memory limits to bound risk and choose CPU limits deliberately because throttling can increase latency. Limits do not prevent memory leaks, and containers exceeding memory limits can be OOM-killed.

READ MORE

Set measured resource requests and deliberate limits

7.07 Define disruption budgets that permit safe maintenance

Low

Use policy/v1 PodDisruptionBudgets for replicated workloads and ensure replicas can satisfy them during drains and upgrades. PDBs constrain voluntary evictions, not node failures or every rollout; overly strict budgets can block upgrades and scale-down.

READ MORE

Define disruption budgets that permit safe maintenance

7.08 Use supported tooling for workload cost allocation

Low

Use AKS cost analysis, OpenCost, or an appropriately operated cost-allocation tool to attribute spend to teams and workloads. Account for shared capacity, storage, networking, and monitoring costs; validate labels and allocation assumptions.

READ MORE

AKS cost analysis

7.09 When required use multi-instance partitioning GPU on AKS Clusters

Medium

Nvidia's A100 GPU can be divided in up to seven independent instances. Each instance has their own memory and Stream Multiprocessor (SM).

READ MORE

Multi-instance GPU Node pool

7.10 If running a Dev/Test cluster use NodePool Start/Stop

Medium

Your AKS workloads may not need to run continuously, for example a development cluster that has node pools running specific workloads.

READ MORE

Start and stop an Azure Kubernetes Service (AKS) node pool

7.11 Use node pool snapshots for configuration reuse, not data backup

Low

Node pool snapshots capture supported node pool configuration for reuse. They do not back up application data, Kubernetes resources, or an entire cluster; ensure the referenced Kubernetes version and image remain supported.

READ MORE

Use node pool snapshots for configuration reuse, not data backup

7.12 Ensure your subscription has enough quota to scale out your nodepools

High

Azure VMs/nodes have quotas per subscription. Ensure that you have planned for these quota and proactively prepare to request for more if needed

READ MORE

Azure subscription and service limits, quotas, and constraints

7.13 Use scale down mode to delete/deallocate nodes

Low

Scale-down Mode allows you to decide whether you would like to delete or deallocate the nodes in your Azure Kubernetes Service (AKS) cluster upon scaling down.

READ MORE

Use Scale-down Mode to delete/deallocate nodes

7.14 Choose cluster autoscaler or node auto-provisioning intentionally

Medium

Evaluate node auto-provisioning for supported workloads and regions, or configure the cluster autoscaler for explicit node pools. Respect coexistence restrictions, quota, disruption settings, consolidation behavior, and capacity limits; do not enable competing node scalers blindly.

READ MORE

Choose cluster autoscaler or node auto-provisioning intentionally

7.15 Coordinate HPA, VPA, and KEDA scaling signals

Medium

Set realistic requests, min/max replicas, stabilization windows, and metric availability. Avoid competing controllers for the same resource signal. Use VPA recommendations or supported update modes deliberately and load-test burst and scale-to-zero behavior.

READ MORE

Coordinate HPA, VPA, and KEDA scaling signals

REVIEW AREA 08 / 15

Cluster operations

Automate day-two work and make changes repeatable and recoverable.

8.01 Use proximity placement only for measured latency requirements

Low

Benchmark before using proximity placement groups. Review compatibility with availability zones, capacity, and node-pool constraints; tighter placement can reduce placement flexibility and resilience. Do not trade away failure-domain separation without a documented reason.

READ MORE

Use proximity placement only for measured latency requirements

8.02 Govern cluster extensions and managed add-ons

Low

Inventory installed extensions and add-ons, their supported Kubernetes versions, permissions, upgrade policies, costs, and operational owners. Remove unused components and validate lifecycle compatibility before upgrades.

READ MORE

Govern cluster extensions and managed add-ons

8.03 Choose a governed node resource group name at creation

Low

Set a node resource group name if required by naming policy, and document AKS ownership. Its name is not a security control; protect access and avoid manually modifying AKS-managed infrastructure.

READ MORE

AKS node resource group

8.04 Use approved private paths for node troubleshooting

Medium

Keep node SSH/RDP endpoints off the public internet. Use supported AKS debug access or approved private management connectivity with time-bound permissions, audit logs, and incident ownership. Do not expose nodes merely to make troubleshooting easier.

READ MORE

Securely connect to nodes through a bastion host

8.05 Run supported diagnostics and configuration checks

Medium

Use AKS diagnostics, Azure Monitor, and maintained troubleshooting tools. Validate tool provenance and permissions before execution. Use policy and manifest validation to detect missing requests and limits; do not install an unmaintained scanner as a production prerequisite.

READ MORE

Run supported diagnostics and configuration checks

8.06 Enable cluster autoscaling

Medium

To keep up with application demands in Azure Kubernetes Service (AKS), you may need to adjust the number of nodes that run your workloads. The cluster autoscaler component can watch for pods in your cluster that can't be scheduled because of resource constraints.

You can enable autoscaling module per node pool but only create one mutual autoscale profile

READ MORE

AKS Autoscaler

Node autoprovisioning (karpenter)

Cluster autoscaler

8.07 Use GitOps to deploy workloads in your cluster

High

GitOps works by using Git as a single source of truth for declarative infrastructure and applications

On Azure you can for instance use Azure Arc for Kubernetes but also directly GitOps addon for AKS

READ MORE

Use Flux v2 addon for AKS

Use ArgoCD with AKS

What is Azure Arc enabled Kubernetes?

8.08 Implement CI/CD to deploy workloads in your cluster

High

Instead of GitOps (or in addition) you can use CI/CD to deploy workloads/tools in your cluster

READ MORE

Automated Deployments for Azure Kubernetes Service (Preview)

Build and deploy to Azure Kubernetes Service with Azure Pipelines

Tutorial: Deploy from GitHub to Azure Kubernetes Service using Jenkins

8.09 Improve efficiency with K8S Tools

Medium

The Kubernetes ecosystem is strengthened by many tools that make operating it easier. Here's a few

READ MORE

OpenLens

OpenLens

Helm

kubectl aliases

kubectx

k9s

8.10 Don't use the default namespace

Medium

It's recommended to keep all applications in a namespace other than default

READ MORE

Using multiple namespaces

8.11 Apply different types of labels to all resources

Low

A common set of labels allows tools to work interoperably, describing objects in a common manner that all tools can understand. For instance, resources should have technical, business and security labels.

READ MORE

Recommended labels

8.12 Fine tune your node configuration

Low

Customizing your node configuration allows you to configure or tune your operating system (OS) settings, node pools or the kubelet parameters to match the needs of the workloads.

When you create an AKS cluster or add a node pool to your cluster, you can customize a subset of commonly used OS and kubelet settings.

READ MORE

Customize node configuration for Azure Kubernetes Service (AKS) node pools

8.13 Restrict command invoke on private clusters

Low

AKS command invoke runs through the Azure management API and can reach a private cluster without direct network connectivity. Restrict its Azure RBAC actions, audit use, and disable it when policy requires all administration through approved private paths.

READ MORE

Restrict command invoke on private clusters

8.14 Consider an appropriate node size, not too large or too small

High

Optimizing node size can lead to lots of benefits including cost optimization, optimized performance and proper resource allocation

READ MORE

Which VM size should I choose as AKS node?

8.15 Regularly check Azure Advisor for recommendations on your cluster

Medium

Azure advisor can enhance environment monitoring and can help improve the operations of your cluster by reducing costs, etc

READ MORE

Azure advisor get started

8.16 Write manifests following good practices using dedicated tools

Low

There are various tools that can make creating k8s manifest files much easier and can help with operational efficiency

READ MORE

Helm chart best practices

8.17 Monitor cluster certificates and verify supported rotation

Low

Understand AKS-managed certificate rotation and monitor cluster health and certificate expiry. Use the documented certificate rotation procedure when necessary and assess its disruption. Workload ingress certificates need a separate renewal process.

READ MORE

Monitor cluster certificates and verify supported rotation

8.18 Test operational constraints before using virtual nodes

Low

If using ACI-backed virtual nodes, validate monitoring, security policy, service connectivity, storage, scheduling, and application recovery end to end. Do not assume virtual nodes are equivalent to ordinary VM-backed node pools.

READ MORE

Create and configure an AKS cluster to use virtual nodes

8.19 Consider spot node pools for non time-sensitive workloads

Low

Spot nodepools can help with cost management

READ MORE

Add an Azure Spot node pool to an AKS cluster

8.20 Consider subscribing to EventGrid Events for AKS automation

Low

AKS emits certain events that enable it to subscribe to EventGrid which can be used for automation

READ MORE

AKS as an Event Grid source

8.21 Protect the AKS-managed node resource group

High

Manage nodes through supported AKS APIs and infrastructure as code, not by manually editing VM scale sets, load balancers, or networking in the node resource group. Evaluate node resource group lockdown and restrict unnecessary write access.

READ MORE

Protect the AKS-managed node resource group

8.22 Prepare a least-privilege incident troubleshooting workflow

High

Document diagnostics, ephemeral debug containers, log access, escalation contacts, and time-bound privileged access. Use approved images and audit actions; do not permanently install privileged debugging toolboxes across production nodes.

READ MORE

Prepare a least-privilege incident troubleshooting workflow

8.23 Test upgrade surge, drain timeouts, and rollback procedures

High

Before upgrading, validate regional VM quota, subnet IP space, maxSurge, PDBs, drain timeout, soak time, and deprecated APIs. Rehearse workload rollback or cluster replacement; Kubernetes control-plane downgrades are not a supported rollback strategy.

READ MORE

Test upgrade surge, drain timeouts, and rollback procedures

8.24 Use supported declarative APIs and validate infrastructure drift

Medium

Pin supported resource-provider API versions and provider/module versions. Review plan or what-if output, apply changes through code review, and detect drift. Use deployment identities scoped to the environment rather than persistent cluster-admin credentials.

READ MORE

Use supported declarative APIs and validate infrastructure drift

REVIEW AREA 09 / 15

Lifecycle & retirements

Stay supported. Track Kubernetes, node operating systems, APIs, and add-ons.

9.01 Use a supported and workload-compatible node operating system

Medium

Select a supported Ubuntu or Azure Linux release after testing drivers, agents, and application dependencies. Azure Linux 2.0 support ended on 30 November 2025 and its node images were removed starting 31 March 2026; migrate to Azure Linux 3.0 or another supported OS.

READ MORE

Use a supported and workload-compatible node operating system

9.02 Keep Kubernetes within its supported lifecycle

High

Track the AKS release calendar and the versions available in each deployment region with az aks get-versions. Standard support follows the supported minor-version window; eligible LTS versions require the Premium tier. Platform support is reduced support, not a substitute for an upgrade plan.

READ MORE

Keep Kubernetes within its supported lifecycle

9.03 Configure node OS upgrades separately from Kubernetes upgrades

High

Choose a supported node OS upgrade channel, such as NodeImage, and test it with a dedicated maintenance window. Node OS upgrades and cluster version upgrades are separate controls. Track image rollout availability, disruption budgets, surge capacity, and completion.

READ MORE

Configure node OS upgrades separately from Kubernetes upgrades

9.04 Configure automatic Kubernetes upgrades and maintenance windows

Medium

Select a supported cluster auto-upgrade channel and planned maintenance windows with enough duration. Validate application compatibility in lower environments first. Maintenance windows are not an absolute guarantee that every emergency platform action is deferred.

READ MORE

Configure automatic Kubernetes upgrades and maintenance windows

9.05 Evaluate Premium tier and LTS against the actual release calendar

Medium

Use AKS Premium tier with the AKSLongTermSupport plan for eligible versions when extended support is required. Check the published support end date and OS/add-on compatibility; LTS does not extend an independently retired node OS or remove the need for patches and upgrades.

READ MORE

Evaluate Premium tier and LTS against the actual release calendar

9.06 Replace retired pod-managed identity with Workload ID

High

The open-source pod-managed identity project was deprecated in 2022 and archived in 2023; AKS managed add-on patch support ended in September 2025. Migrate applications and SDKs to Microsoft Entra Workload ID, test token refresh, then remove obsolete identity bindings and components.

READ MORE

Replace retired pod-managed identity with Workload ID

9.07 Upgrade retired Basic Load Balancer and Availability Sets

High

AKS support for Basic Load Balancer and Availability Sets ended on 30 September 2025. Follow the documented migration paths to Standard Load Balancer and supported node-pool infrastructure; validate public IPs, DNS, routes, downtime, and recovery.

READ MORE

Upgrade retired Basic Load Balancer and Availability Sets

9.08 Migrate kubenet before its retirement

High

AKS kubenet retirement is scheduled for 31 March 2028. Inventory affected clusters, validate the supported migration path to Azure CNI Overlay, and rehearse changes to pod addressing, network policies, routes, and external dependencies.

READ MORE

Migrate kubenet before its retirement

9.09 Complete the Ingress NGINX migration before support ends

High

Upstream Ingress NGINX maintenance ended in March 2026, and AKS application routing NGINX critical-security-patch support runs through November 2026. Test a supported Gateway API replacement in parallel, translate annotations and policy, validate TLS and traffic behavior, and plan DNS cutover and rollback.

READ MORE

Complete the Ingress NGINX migration before support ends

9.10 Track release notes and future platform-initiated upgrades

High

Subscribe to AKS releases, retirement notices, Service Health, and the regional release tracker. Microsoft documents platform-initiated upgrades beginning 1 June 2027 after the unsupported-version grace period; these do not follow your maintenance window. Keep clusters supported rather than relying on that safeguard.

READ MORE

Track release notes and future platform-initiated upgrades

REVIEW AREA 10 / 15

Observability & incident response

Connect metrics, logs, traces, and actionable alerts to service objectives.

10.01 Collect and govern workload logs

High

Enable Container insights or a supported log platform with an explicit data collection rule. Define namespace filters, sensitive-data handling, retention, access control, and a cost budget. Confirm that logs are queryable during node and application failure.

READ MORE

Collect and govern workload logs

10.02 Collect Prometheus metrics with a supported monitoring stack

High

Use Azure Monitor managed service for Prometheus or an operated Prometheus deployment. Configure scraping, recording and alerting rules, cardinality limits, retention, and Grafana dashboards. Container log collection is separate from Prometheus metric ingestion.

READ MORE

Collect Prometheus metrics with a supported monitoring stack

10.03 Define a distributed tracing strategy

Low

Choose propagation standards, instrumentation ownership, sampling, retention, and sensitive-data controls for distributed traces. Verify that representative requests can be correlated across services and dependencies before relying on tracing in an incident.

READ MORE

Application Insights OpenTelemetry overview

10.04 Monitor CPU and memory utilization of the nodes

Medium

With Container insights, you can use the performance charts and health status to monitor the workload of Kubernetes clusters hosted on AKS

READ MORE

Monitor your Kubernetes cluster performance

10.05 Enable control-plane diagnostic and audit logs

High

Configure AKS diagnostic settings for the required control-plane and audit categories. Choose audit versus audit-admin deliberately, route logs to an approved destination, and validate access, retention, query availability, and collection costs.

READ MORE

AKS monitoring reference

10.06 Query workload logs through a governed log platform

High

Provide documented queries and dashboards for application failures, restarts, and dependency issues. Use supported Container insights or another operated platform, define retention and access controls, and verify diagnostic data is available during an incident.

READ MORE

Container insights overview

10.07 Use ContainerLogV2 and supported Azure Monitor agents

Medium

Migrate legacy ContainerLog collection and the retired Log Analytics agent to supported Container insights collection using Azure Monitor Agent and ContainerLogV2. Validate queries, alerts, data collection rules, multiline logging, retention, and ingestion costs.

READ MORE

Use ContainerLogV2 and supported Azure Monitor agents

10.08 Configure alerts on the most critical metrics (see Container Insights for recommendations)

High

Alerts can provide early warnings of issues, etc in your cluster

READ MORE

Metric alert rules in Container insights

10.09 Monitor outbound SNAT capacity and connection failures

Medium

Monitor SNAT utilization, dropped connections, and outbound failure rates for the actual egress path. Tune connection reuse, idle timeouts, outbound IPs, and NAT gateway capacity; SNAT monitoring is primarily an availability and capacity control, not a replacement for egress filtering.

READ MORE

Monitor outbound SNAT capacity and connection failures

10.10 Subscribe to resource health notifications for your AKS cluster

High

Learn how to troubleshoot resource health issues

READ MORE

Azure Kubernetes Services troubleshooting documentation

10.11 Define SLIs, SLOs, and actionable error-budget alerts

High

Measure user-visible availability and latency, set service objectives, and page on actionable symptoms or error-budget burn. Include pending pods, failed scheduling, persistent-volume attachment, DNS, certificate expiry, and failed upgrades in platform alerts.

READ MORE

Define SLIs, SLOs, and actionable error-budget alerts

10.12 Test incident response and diagnostic-data retention

High

Maintain dashboards, runbooks, escalation paths, audit-log access, and diagnostic retention for the investigation window. Run an incident exercise and verify alerts reach an accountable responder. Protect logs from leaking secrets and excessive collection costs.

READ MORE

Test incident response and diagnostic-data retention

10.13 Correlate OpenTelemetry traces with workload metrics and logs

Medium

Instrument application request paths with OpenTelemetry or a supported Application Insights integration. Preserve correlation IDs, control sampling and cardinality, and validate that dependency failures can be traced across service boundaries without exposing sensitive data.

READ MORE

Correlate OpenTelemetry traces with workload metrics and logs

REVIEW AREA 11 / 15

Reliability & disaster recovery

Validate availability, recovery objectives, backups, and regional failover.

11.01 Define non-functional requirements such as SLAs, RTO (Recovery Time Objective) and RPO (Recovery Point Objective)

High

Before going to production, it is very important to decide the target of availability and recovery time you are looking for. This will help you to decide the right backup strategy but also the architecture of your solution.

READ MORE

Define the BCDR strategy

11.02 Rehearse recovery into a clean, isolated environment

High

Regularly rebuild the platform from infrastructure as code in an isolated recovery environment. Restore application configuration and data, measure RTO/RPO, and test dependencies and access. Never delete production merely to conduct a recovery rehearsal.

READ MORE

Rehearse recovery into a clean, isolated environment

11.03 Distribute production workloads across availability zones

High

Use supported zonal node pools and topology spread constraints to separate replicas across nodes and zones. Validate sufficient surviving capacity, zonal storage behavior, load balancing, and failure recovery; availability zones are separate datacenter failure domains within a region.

READ MORE

Distribute production workloads across availability zones

11.04 Design multiregion recovery around business requirements

Medium

Choose active-active or active-passive regions based on RTO/RPO, data residency, latency, dependencies, and available capacity. Regions need not be an Azure region pair; validate service support, replication, quota, and independence explicitly.

READ MORE

Design multiregion recovery around business requirements

11.05 Test global traffic failover and failback

Low

Select Front Door or Traffic Manager according to protocol and routing needs. Test health probes, DNS caching, TLS, session behavior, regional dependencies, and failback. Routing changes do not recover or synchronize application data.

READ MORE

Test global traffic failover and failback

11.06 Create a storage migration plan

Low

Your applications might use Azure Storage for their data. Because your applications are spread across multiple AKS clusters in different regions, you need to keep the storage synchronized

READ MORE

Create a storage migration plan

Backup, restore and migrate Kubernetes resources including state to another AKS cluster with Velero

11.07 Select an AKS tier with the required control-plane SLA

High

Use Standard or Premium tier for the financially backed Kubernetes API-server uptime SLA. Microsoft documents 99.95% with availability zones and 99.9% without; Free tier has no financially backed SLA. These are control-plane commitments, not an end-to-end application SLA; verify current contractual terms.

READ MORE

Select an AKS tier with the required control-plane SLA

11.08 Spread replicas across nodes and zones

Low

Use topology spread constraints or appropriate anti-affinity to keep replicas in separate failure domains. Balance strict scheduling rules against available recovery capacity, and test both node loss and zone loss under representative load.

READ MORE

Spread replicas across nodes and zones

11.09 If using a private registry (ACR), configure region replication to store images in multiple regions

Medium

Companies that want a local presence, or a hot backup, choose to run services from multiple Azure regions.

As a best practice, placing a container registry in each region where images are run allows network-close operations, enabling fast, reliable image layer transfers. Geo-replication enables an Azure container registry to function as a single registry, serving multiple regions with multi-master regional registries.

READ MORE

Enable geo-replication for container images

11.10 Configure your registry to use zone redundancy

Medium

Use Azure availability zones to create a resilient and high availability Azure container registry within an Azure region.

READ MORE

Enable zone redundancy in Azure Container Registry for resiliency and high availability

11.11 Move Container Registry to a dedicated resource group

Low

Because container registries are resources that are used across multiple container hosts, a registry should reside in its own resource group.

Although you might experiment with a specific host type, such as Azure Container Instances, you'll likely want to delete the container instance when you're done. However, you might also want to keep the collection of images you pushed to Azure Container Registry. By placing your registry in its own resource group, you minimize the risk of accidentally deleting the collection of images in the registry when you delete the container instance resource group.

READ MORE

Registry best practices - Use dedicated resource group

11.12 Evaluate ACR soft delete against support and retention requirements

Low

Where supported, evaluate ACR soft delete to recover accidentally deleted artifacts within the configured retention period. Check current feature status, limitations, and costs; soft delete does not replace immutable release retention, replication, or tested disaster recovery.

READ MORE

Enable soft delete policy

11.13 Back up supported cluster resources and persistent volumes

High

Configure Azure Backup for AKS or a supported alternative with explicit scope, retention, and recovery objectives. Check the current support matrix for Azure Disk/Azure Files, snapshot versus vaulted backup, regions, and cross-region restore. Back up external databases separately.

Perform full restore drills, including identities, CRDs, secrets, application dependencies, and data consistency. A successful backup job does not prove that the application can be recovered.

READ MORE

Back up supported cluster resources and persistent volumes

11.14 Exercise failure scenarios under representative load

High

Rehearse pod, node, zone, DNS, registry, egress, and dependency failures in a controlled environment. Use fault injection only with explicit scope and safety controls, verify service objectives, and track remediation actions from each exercise.

READ MORE

Exercise failure scenarios under representative load

REVIEW AREA 12 / 15

Windows workloads

Account for Windows-specific compatibility, identity, and operational constraints.

12.01 Match Windows base images to a supported node OS

High

Validate the supported Windows host/container compatibility matrix. Windows Server 2019 node-pool support ended on 1 March 2026 and images are scheduled for removal on 1 April 2027. Migrate to a supported LTSC release such as Windows Server 2022 or 2025 after compatibility testing.

READ MORE

Match Windows base images to a supported node OS

12.02 Test graceful shutdown and recovery on Windows

Medium

Test SIGTERM-equivalent Windows shutdown behavior, preStop hooks, terminationGracePeriodSeconds, and actual application cleanup on supported Windows images. Avoid blanket claims that graceful termination is unavailable; design for abrupt failure and verify behavior during upgrades.

READ MORE

Test graceful shutdown and recovery on Windows

12.03 Restrict Windows HostProcess and administrative privileges

Medium

Run applications as a non-administrative identity where supported. Windows does not use Linux privileged-container semantics, but HostProcess containers can access the host and require tightly restricted admission, identities, and namespaces.

READ MORE

Restrict Windows HostProcess and administrative privileges

12.04 Set and monitor Windows memory requests and limits

Medium

Size Windows nodes and container memory using measured working sets and system reservations. Validate eviction and OOM behavior for the actual Windows/runtime version, alert on pressure, and test recovery rather than assuming Linux memory-management behavior.

READ MORE

Set and monitor Windows memory requests and limits

12.05 Choose a supported Azure CNI model for Windows

High

Use a Windows-supported Azure CNI configuration, including Overlay where supported for the selected versions. Kubenet and the Linux Cilium data plane are not Windows networking options. Validate policy-engine and IPAM support before provisioning.

READ MORE

Choose a supported Azure CNI model for Windows

12.06 Keep windows containers patch level in sync with host patch level

High

Windows Server nodes in AKS must be upgraded to get the latest patch fixes and updates. Windows Updates are not enabled on nodes in AKS. AKS releases new node pool images as soon as patches are available, it is the customers responsibility to upgrade node pools to stay current on patches and hotfix.

READ MORE

Windows container version compatibility

How do patch my Windows nodes?

12.07 Validate network policy enforcement on Windows

High

Do not assume Linux and Windows share the same network policy capabilities. Azure NPM on Windows is unsupported from 30 September 2026; select a supported alternative such as Calico after validating the exact OS, network model, and version matrix.

READ MORE

Validate network policy enforcement on Windows

12.08 Enable Group Managed Service Accounts (GMSA) for your Windows Server nodes

Medium

Group Managed Service Accounts (GMSA) is a managed domain account for multiple servers that provides automatic password management, simplified service principal name (SPN) management and the ability to delegate the management to other administrators.

READ MORE

Enable Group Managed Service Accounts (GMSA) for your Windows Server nodes

12.09 If required for AKS Windows workloads, HostProcess containers can be used

Medium

HostProcess / Privileged containers extend the Windows container model to enable a wider range of Kubernetes cluster management scenarios.

READ MORE

Use Windows HostProcess containers

12.10 Taint Windows nodes

Low

Adding taints, tolerations and node selectors will help avoid having pods scheduled in the wrong OS node pool

READ MORE

Scheduling Windows containers

12.11 Migrate Windows Azure NPM policy before relying on enforcement

Medium

Inventory Windows pools using Azure Network Policy Manager, whose AKS Windows support ended on 30 September 2026. Follow supported migration guidance and test positive and negative connectivity cases with the replacement engine.

READ MORE

Migrate Windows Azure NPM policy before relying on enforcement

12.12 Validate accelerated networking for Windows node SKUs

Medium

Select a supported Windows VM size and verify accelerated networking availability and actual enablement. Benchmark throughput and latency, and include driver and node-image compatibility in upgrade testing.

READ MORE

Accelerated networking overview

12.13 Track Windows LTSC and Annual Channel retirement dates

High

Windows Server Annual Channel preview support ended on 15 May 2026, with image removal scheduled for 15 May 2027. Windows Server 2022 AKS support is scheduled to end on 30 June 2028, with image removal on 30 June 2029. Plan LTSC migration against current version and OS support matrices.

READ MORE

Track Windows LTSC and Annual Channel retirement dates

Application delivery

Ship resilient workloads with safe rollouts and explicit runtime contracts.

13.01 Implement liveness probes that detect unrecoverable local failure

High

Use a liveness probe only when restarting the container can resolve a failure. Avoid coupling liveness to a shared downstream dependency, which can trigger restart storms during an outage. Test thresholds under load.

READ MORE

Implement liveness probes that detect unrecoverable local failure

13.02 Protect slow startup with a startup probe

Medium

Configure a startup probe with a failure budget covering the measured worst-case startup. Liveness and readiness probes do not run until the startup probe succeeds; test image pulls and cold initialization separately.

READ MORE

Protect slow startup with a startup probe

13.03 Implement a proper Readiness probe

Medium

Sometimes, applications are temporarily unable to serve traffic. For example, an application might need to load large data or configuration files during startup, or depend on external services after startup. In such cases, you don't want to kill the application, but you don't want to send it requests either. Kubernetes provides readiness probes to detect and mitigate these situations. A pod with containers reporting that they are not ready does not receive traffic through Kubernetes Services.

READ MORE

Configure Liveness, Readiness and Startup Probes

13.04 Implement graceful termination and connection draining

Low

Handle termination signals, configure terminationGracePeriodSeconds, and use preStop only where needed. Allow in-flight requests to drain and test rolling updates and node drains. Hooks are not guaranteed during every failure, so persist critical state safely.

READ MORE

Implement graceful termination and connection draining

13.05 Run enough replicas for the availability target

Medium

Use multiple replicas for services that require high availability, and spread them across nodes and zones. Test that the remaining replicas can serve peak demand during failure or maintenance; replica count alone does not establish resilience.

READ MORE

Pod replicas

13.06 Apply tags/labels to all resources

Medium

Ensure that your components are tagged, it could be business, security or technical tags and these tags will help to assess or apply relevant policies.

READ MORE

Recommended Labels

13.07 Autoscale applications using measured demand

Low

Configure HPA or KEDA with appropriate metrics, requests, replica bounds, and stabilization windows. Test metric outages, burst traffic, dependency limits, and scale-down; ensure node scaling and regional capacity can satisfy demand.

READ MORE

Horizontal Pod Autoscaler

Kubernetes Event-driven Autoscaling (KEDA)

KEDA as AKS addon

13.08 Keep secrets out of images and use a managed secret store

High

Store application secrets in Key Vault and retrieve them with a supported integration and least-privilege identity. AKS provides encryption at rest, but Kubernetes Secrets are still accessible to authorized API clients and are not an external vault. Never commit secrets to source or bake them into images.

READ MORE

Keep secrets out of images and use a managed secret store

13.09 Use Microsoft Entra Workload ID for application access

Medium

Enable the OIDC issuer and Workload ID, then federate a dedicated Kubernetes service account to a least-privilege managed identity. Use supported SDKs and the required workload identity label. Replace the retired pod-managed identity add-on; do not give applications node identity permissions.

READ MORE

Use Microsoft Entra Workload ID for application access

13.10 Use namespaces with explicit access and policy boundaries

Medium

Use dedicated namespaces for workloads and apply scoped RBAC, quotas, and network/admission policy. Avoid deploying applications into default or system namespaces. Namespaces provide logical separation, not a complete security boundary.

READ MORE

Namespaces

13.11 Set up requests and limits on your containers

High

When Containers have resource requests specified, the scheduler can make better decisions about which nodes to place Pods on. And when Containers have their limits specified, contention for resources on a node can be handled in a specified manner.

READ MORE

Managing Compute Resources for Containers

Take benefit of the Quality of Service

13.12 Declare an explicit workload security context

Medium

Use platform-appropriate least-privilege settings. For Linux, prefer non-root, RuntimeDefault seccomp, dropped capabilities, no privilege escalation, and a read-only root filesystem. Disable automatic service account token mounting where the workload does not need it; validate projected tokens required by Workload ID.

READ MORE

Declare an explicit workload security context

13.13 Ensure your manifests respect good practices

Medium

Best practices inside the cluster start with the configuration of your manifests. Ensure that they respect good practices

READ MORE

Kubernetes application security checklist

13.14 Lint Dockerfiles and enforce an image-build baseline

High

Validate base-image provenance, pinned versions, non-root execution where supported, build secrets, dependency installation, and unnecessary packages. Use maintained linters and policy checks without exposing build credentials in layers or logs.

READ MORE

SNYK 10 Docker Image Security Best Practices

21 Best Practises in 2021 for Dockerfile

Dockle

Hadolint

13.15 Scan container images during the build

High

Scan OS packages and application dependencies during image builds with maintained tools. Retain reports linked to immutable digests, define remediation targets, and rescan deployed versions as new vulnerabilities are disclosed.

READ MORE

Container vulnerability assessment

13.16 Enforce image vulnerability thresholds and exception expiry

Medium

Define policy using exploitability, exposure, severity, and available fixes. Block unacceptable risk, record owners and expiry for exceptions, and rescan continuously. An unfixed critical vulnerability is not automatically acceptable simply because the vendor has not published a fix.

READ MORE

Enforce image vulnerability thresholds and exception expiry

13.17 Validate container build and runtime policy

High

Validate approved base images, non-root execution where supported, exposed services, secret handling, and required labels. Apply rules based on the threat model; opening port 80 alone is not proof of noncompliance.

READ MORE

Validate container build and runtime policy

13.18 Assess application modernization before containerization

Low

Validate runtime support, licensing, state, external dependencies, health probes, and operational ownership before moving an application to AKS. Choose supported migration tooling based on the workload rather than assuming a point-and-containerize conversion produces a production-ready service.

READ MORE

Assess application modernization before containerization

13.19 Apply the right deployment type to your application

Low

There are a variety of techniques to deploy new applications to production so choosing the right strategy is an important decision that needs to be made to leverage the impact of change on the consumer.

READ MORE

Kubernetes Deployment Strategies

13.20 Use workload controllers instead of standalone pods

Low

Manage long-running applications through Deployments or StatefulSets, and finite work through Jobs or CronJobs. A standalone pod is not recreated after node loss by a workload controller; test replacement and application recovery.

READ MORE

Use workload controllers instead of standalone pods

13.21 Set image pull policy and deploy immutable image digests

Low

Use immutable digests for reproducible deployments and choose imagePullPolicy deliberately. Always checks registry resolution but can reuse cached layers; IfNotPresent reuses cached images. Do not rely on mutable latest tags, and test registry failure behavior.

READ MORE

Set image pull policy and deploy immutable image digests

13.22 Provision Azure resources with reviewed infrastructure as code

Medium

Use Bicep, ARM templates, or Terraform modules with pinned versions, environment-specific configuration, least-privilege deployment identities, and reviewed changes. Keep state protected, detect drift, and rehearse rebuilding the platform.

READ MORE

Provision Azure resources with reviewed infrastructure as code

13.23 Use canary or blue/green deployments

Medium

This will help ensure your workloads remain online during cluster upgrades.

READ MORE

Blue-green deployment for AKS

13.24 Evaluate Dapr only when its application capabilities are needed

Low

Dapr can provide service invocation, pub/sub, state management, and other distributed application APIs. Adopt it when those capabilities justify additional sidecars, control-plane components, security configuration, and lifecycle management; it is not an upgrade-availability mechanism.

READ MORE

Evaluate Dapr only when its application capabilities are needed

13.25 Validate deprecated Kubernetes APIs before every upgrade

High

Check release notes, API-server warnings, manifests, CRDs, Helm charts, operators, and admission webhooks for APIs removed in the target version. Upgrade dependencies first and test conversion and rollback paths in a representative environment.

READ MORE

Validate deprecated Kubernetes APIs before every upgrade

13.26 Secure CI/CD with federation and scoped deployment permissions

High

Use OIDC federation from the CI platform to Azure, protect release environments, pin reviewed actions and build dependencies, and scope deployment identities to the required namespaces and resources. Keep production credentials and kubeconfig files out of logs and artifacts.

READ MORE

Secure CI/CD with federation and scoped deployment permissions

Images & supply chain

Build, verify, patch, and distribute trusted container artifacts.

14.01 Scan the container image against vulnerabilities

High

Scan operating-system packages and application dependencies in container images during build and continuously after publication. Triage findings using exploitability and exposure, enforce remediation targets, and verify deployed image digests. A clean scan does not prove that an image is free of vulnerabilities.

READ MORE

- Azure Security Center : scanning feature (Qualys)
- Identify vulnerable container images in your CI/CD workflows
- Prisma (ex Twistlock)
- Anchore
- Clair

14.02 Allow deploying containers only from known registries

Medium

One of the most common custom policies that you might want to consider is to restrict the images that can be deployed in your cluster. But it can also be addressed with a proper egress lockdown or using an admission controller

READ MORE

- Use the Azure Policy : Ensure only allowed container images in AKS
- Using ImagePolicyWebhook
- Using egress lockdown and authorizing only the URL of your registry

14.03 Detect and respond to container runtime threats

High

Enable supported runtime threat detection, such as Microsoft Defender for Containers, and route alerts into incident response. Validate required sensors and permissions, investigate suspicious process and network activity, and rehearse containment without assuming a sensor prevents every attack.

READ MORE

- Runtime threat protection

14.04 Prevent promotion of images that violate current policy

High

Continuously rescan stored images and block release promotion or admission when they violate approved risk policy. Verify that quarantine tooling is supported and enforced by every pull path; a registry scan alone does not block deployment.

READ MORE

Prevent promotion of images that violate current policy

14.05 Scope registry roles to the required repositories and actions

High

Separate image push, pull, delete, signing, and administrative permissions. For ABAC-enabled registries, use supported repository roles and conditions rather than assuming AcrPull/AcrPush applies. Audit assignments and keep the registry admin user disabled.

READ MORE

Scope registry roles to the required repositories and actions

14.06 Network Segmentation of Docker Registries

High

Limit access to a registry by assigning virtual network private IP addresses to the registry endpoints and using Azure Private Link. Network traffic between the clients on the virtual network and the registry's private endpoints traverses the virtual network and a private link on the Microsoft backbone network, eliminating exposure from the public internet. Private Link also enables enables private registry access from on-premises through Azure ExpressRoute private peering or a VPN gateway.

READ MORE

Azure Container Registry Private Link

14.07 Use minimal, maintained base images

Medium

Choose supported, regularly patched minimal or distroless images compatible with the application. Do not use end-of-life Debian 10 examples. Removing a shell reduces attack surface but does not prevent exploitation; retain a controlled debug workflow.

READ MORE

Use minimal, maintained base images

14.08 Refresh container when base image is updated

Low

As you use base images for application images, use automation to build new images when the base image is updated. As those base images typically include security fixes, update any downstream application container images.

READ MORE

Automatically build new images on base image update

Azure DevOps - Trigger pipeline from Docker image update

14.09 Generate SBOMs and verify artifact provenance

High

Generate an SBOM for each release, retain build provenance, and sign immutable artifacts using supported OCI signing tools. Verify signatures and trusted identities during release or admission; signing without verification does not establish a supply-chain control.

READ MORE

Generate SBOMs and verify artifact provenance

14.10 Retain and replicate the artifacts required for recovery

Medium

Keep known-good image digests, Helm charts, manifests, and signatures available for the recovery window. Test pulls from the recovery region, registry private DNS, retention rules, and access identities; avoid deleting an image still referenced by production or rollback plans.

READ MORE

Retain and replicate the artifacts required for recovery

REVIEW AREA 15 / 15

Cost & sustainability

Measure unit costs and balance efficiency with reliability and operational effort.

15.01 Allocate costs to teams and workloads

High

Use AKS cost analysis where supported or a maintained allocation tool. Combine Azure tags and Kubernetes labels with ownership, budgets, and chargeback rules. Include idle capacity, monitoring ingestion, network egress, storage, licenses, and managed add-ons.

READ MORE

Allocate costs to teams and workloads

15.02 Right-size from measured demand and unit economics

Medium

Review requests, limits, node utilization, autoscaler behavior, and cost per business transaction. Reduce waste without removing failure headroom, disrupting latency objectives, or creating unschedulable workloads.

READ MORE

Right-size from measured demand and unit economics

15.03 Use reservations and savings plans only for stable baseline demand

Medium

Model measured compute commitments against expected VM families, regions, and utilization. Use Spot capacity only for interruption-tolerant workloads with checkpointing and fallback; never rely on Spot for essential system components.

READ MORE

Use reservations and savings plans only for stable baseline demand

15.04 Stop disposable environments and clean orphaned resources safely

Medium

Schedule nonproduction shutdown or scale-down when supported and acceptable. Review unused disks, public IPs, snapshots, images, and workspaces with owners and retention policies before deletion; never treat an unattached disk as automatically disposable.

READ MORE

Stop disposable environments and clean orphaned resources safely

15.05 Evaluate sustainability without compromising service objectives

Low

Measure utilization and workload efficiency, evaluate compatible Arm64 compute, and select regions within latency and residency constraints. Consider the total operational footprint rather than assuming one VM architecture is always cheaper or lower-carbon.

READ MORE

Evaluate sustainability without compromising service objectives

Keep the review alive.

Revisit this assessment after platform upgrades, architecture changes, incidents, and retirement announcements. The linked documentation is authoritative for implementation and support terms.

Created by Louis-Guillaume Morand and the AKS Checklist contributors. This is an independent community project, not an official Microsoft product. Original code is based on David Dias's Front-End Checklist. Licensed under MPL-2.0.

github.com/lgmorand/the-aks-checklist