

---

# The AKS Checklist

[www.the-aks-checklist.com](http://www.the-aks-checklist.com)

May 6, 2021

- Cluster Setup
  - Logically isolate cluster
  - Physically isolate cluster
  - IP Range authorization
  - AAD Integration
  - Use System Node Pools
  - AKS Managed Identity
  - VM Sizing
  - Set Upgrade Channel
  - K8S RBAC + AAD Integration
  - Private cluster
  - Enable cluster autoscaling
  - Sizing of the nodes
  - Refresh container when base image is updated
  - Check if you need the Kubernetes dashboard
  - Use AKS and ACR integration without password
  - Use placement proximity group to improve performance
- Container
  - Scan the container image against vulnerabilities
  - Allow deploying containers only from known registries
  - Runtime Security of Applications
  - Quarantine of Docker Images in Docker Registries that have discovered issues
  - Role-Based Access Control (RBAC) to Docker Registries
  - Network Segmentation of Docker Registries
  - Prefer distroless images
- Development
  - Implement a proper Liveness probe
  - Implement a proper Startup probe
  - Implement a proper Readiness probe
  - Implement a proper prestop hook
  - Run more than one replica for your Deployment
  - Apply tags to all resources
  - Implement autoscaling of your applications
  - Store your secrets in Azure Key Vault, don't inject passwords in Docker Images
  - Implement Pod Identity
  - Use Kubernetes namespaces to properly isolate your Kubernetes resources
  - Set up requests and limits on your containers
  - Specify the security context of your pod/container

- Conduct Docker Image Builds using Docker Image Security Best Practices
- Static Analysis of Docker Images on Build
- Threshold enforcement of Docker Image Builds that contain vulnerabilities
- Compliance enforcement of Docker Image Builds
- Disaster Recovery
  - Ensure you can perform a whitespace deployment
  - Use availability zones
  - Plan for multiregion deployment
  - Use Azure Traffic Manager to route traffic
  - Create a storage migration plan
  - Guaranty SLA for the control plane
- Multi Tenancy
  - Enable geo-replication for container images
  - Enable traffic management
- Network
  - Choose the appropriate network model
  - Plan IP addressing carefully
  - Distribute ingress traffic
  - Secure your exposed applications with a web application firewall (WAF)
  - Don't expose your load-balancer on Internet if not necessary
  - Control traffic flow with network policies
  - Configure default network policies in each namespace
  - Prevent data-leaking with egress lockdown
- Operations
  - Maintain kubernetes version up to date
  - Keep nodes up to date and patched
  - Securely connect to nodes through a bastion host
  - Regularly check for cluster issues
  - Monitor the security of your cluster with Azure Security Center
  - Provision a log aggregation tool
  - Enable master node logs
  - Collect metrics
  - Configure distributed tracing
  - Control the compliance with Azure Policies
  - Read The Definitive Guide to Securing Kubernetes
  - Enable Azure Defender for Kubernetes
  - Use Azure Key Vault

- Use GitOps
  - Make your life easier with K8S Tools
  - Don't use the default namespace
- Resource Management
  - Enforce resource quotas
  - Set memory limits and requests for all containers
  - Configure pod disruption budgets
  - Set up cluster auto-scaling
- Storage
  - Choose the right storage type
  - Size the nodes for storage needs
  - Dynamically provision volumes
  - Secure and back up your data
  - Make your storage resilient
- Windows
  - Map the base image to node OS
  - Prepare your application for an abrupt kill
  - Don't use privileged containers
  - Watch for memory usage
  - Implement CNI network mode
  - Patch your nodes yourself
  - Secure the traffic of your containers

## Cluster Setup

### Logically isolate cluster

Use logical isolation to separate teams and projects. Try to minimize the number of physical AKS clusters you deploy to isolate teams or applications

#### Documentation

- [Isolating cluster](#)

### Physically isolate cluster

Minimize the use of physical isolation for each separate team or application deployment

## Documentation

- [Isolating cluster](#)

## IP Range authorization

The API server is the central way to interact with your cluster. To improve cluster security and minimize attacks, the API server should only be accessible from a limited set of IP address ranges. By using a private cluster, you can ensure that network traffic between your API server and your node pools remains on the private network only. Because the API server has a private address, it means that to access it for administration or for deployment, you need to set up a private connection, like using a “jumpbox” (i.e.: Azure Bastion)

## Documentation

- [Secure access to the API server using authorized IP address ranges](#)

## AAD Integration

Azure Kubernetes Service (AKS) can be configured to use Azure Active Directory (Azure AD) for user authentication. In this configuration, you can sign in to an AKS cluster by using your Azure AD authentication token.

## Documentation

- [AKS-managed Azure Active Directory integration](#)

## Use System Node Pools

Manage system node pools in Azure Kubernetes Service (AKS)

## Documentation

- [AKS System Pools](#)

## AKS Managed Identity

Each AKS cluster needs either a Managed Identity or Service Principal. We recommend using Managed Identity in AKS

## Documentation

- [Use managed identities in Azure Kubernetes Service](#)

## VM Sizing

Prefer fewer, larger clusters over of many, smaller clusters

## Set Upgrade Channel

In addition to manually upgrading a cluster, you can set an auto-upgrade channel on your cluster

### Documentation

- [Set AKS auto-upgrade channel](#)

## K8S RBAC + AAD Integration

Control access to cluster resources using Kubernetes role-based access control and Azure Active Directory identities in Azure Kubernetes Service

### Documentation

- [Limit cluster access via K8S RBAC for users & workloads](#)

## Private cluster

In a private cluster, the control plane or API server has internal IP addresses and is not exposed to Internet. By using a private cluster, you can ensure that network traffic between your API server and your node pools remains on the private network only. Because the API server has a private address, it means that to access it for administration or for deployment, you need to set up private connection, like using a “jumpbox” (i.e.: Azure Bastion)

### Documentation

- [Create a private cluster](#)

## Enable cluster autoscaling

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.

### Documentation

- [AKS Autoscaler](#)

## Sizing of the nodes

what type of worker nodes should I use, and how many of them is a critical question which requires the analysis of the workloads deployed on your cluster to get the best values of it

### Documentation

- [Choosing a worker node size](#)
- [Choose the right storage type](#)
- [How to size Kubernetes clusters](#)

## Refresh container when base image is updated

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.

### Documentation

- [Automatically build new images on base image update](#)
- [Azure DevOps - Trigger pipeline from Docker image update](#)

## Check if you need the Kubernetes dashboard

By default Kubernetes in AKS deploys and exposes the default dashboard for clusters running a Kubernetes version less than 1.18. If you don't plan to use it to pilot your cluster, you should disable it.

## Use AKS and ACR integration without password

AKS can authenticate to ACR without using any password, but by using either Service Principal or Managed Identity.

### Documentation

- [Authenticate with Azure Container Registry from AKS](#)

## Use placement proximity group to improve performance

When deploying your application in Azure, spreading Virtual Machine (VM) instances across regions or availability zones creates network latency, which may impact the overall performance of your application. A proximity placement group is a logical grouping used to make sure Azure compute resources are physically located close to each other

### Documentation

- [Reduce latency with proximity placement groups](#)

## Container

### Scan the container image against vulnerabilities

Scan your container images to ensure there are no vulnerabilities in it

### Documentation

- [Azure Security Center : scanning feature \(Qualys\)](#)
- [Introduction to Azure Defender for container registries](#)

### Tools

- [Prisma \(ex Twistlock\)](#)
- [Aqua](#)
- [Anchore](#)
- [Clair](#)

### Allow deploying containers only from known registries

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

### Documentation

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



## Runtime Security of Applications

Integrate Runtime Security for your pods. To complete the defense in depth structure, ensure Runtime protection is in place to protect from process, network, storage and system call attacks.

### Tools

- [Prisma Runtime defense](#)
- [Aqua Runtime Policies Overview](#)
- [StackRox Threat Detection](#)
- [Falco](#)

## Quarantine of Docker Images in Docker Registries that have discovered issues

Use policy to protect images from drift while in the registry, on both push and pull. On build, the image is secured based on the threshold set, but now while in the registry a new issue is discovered. You need to ensure that the image can not be deployed until the issue is remediated.

### Documentation

- [ACR Quarantine](#)

## Role-Based Access Control (RBAC) to Docker Registries

The Azure Container Registry service supports a set of built-in Azure roles that provide different levels of permissions to an Azure container registry. Use Azure role-based access control (RBAC) to assign specific permissions to users, service principals, or other identities that need to interact with a registry.

### Documentation

- [Azure Container Registry roles and permissions](#)

## Network Segmentation of Docker Registries

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 private registry access from on-premises through Azure ExpressRoute private peering or a VPN gateway.

### Documentation

- [Azure Container Registry Private Link](#)

## Prefer distroless images

When building a docker image, try to use the distroless version of the base OS image, to reduce the risk of vulnerabilities with preinstalled but unused tools. For example, use base-debian10 instead of debian10

### Documentation

- [Google distroless images](#)

## Development

### Implement a proper Liveness probe

Many applications running for long periods of time eventually transition to broken states, and cannot recover except by being restarted. Kubernetes provides liveness probes to detect and remedy such situations. The probe is here to tell Kubernetes to restart your pod when it is not responding anymore

### Documentation

- [Configure Liveness, Readiness and Startup Probes](#)

### Implement a proper Startup probe

Protect slow starting containers with startup probes. Startup probe allow to delay the initial check by liveness which could cause deadlock or wrong result

### Documentation

- [Configure Liveness, Readiness and Startup Probes](#)

## Implement a proper Readiness probe

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.

### Documentation

- [Configure Liveness, Readiness and Startup Probes](#)

## Implement a proper prestop hook

This hook is called immediately before a container is terminated due to an API request or management event such as liveness probe failure, preemption, resource contention and others. It can be used when you have critical process you want to finish or save when your pod is destroyed for any reason

### Documentation

- [Container Lifecycle Hooks](#)

## Run more than one replica for your Deployment

Ensure that your application always configure proper replicas to ensure resiliency.

## Apply tags to all resources

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.

## Implement autoscaling of your applications

Automatically scale your application to the number of pods required to handle the current load. This can be achieved by using Horizontal Pod Autoscaler for CPU & Memory or by using KEDA for scaling based on other sources

### Documentation

- [Horizontal Pod Autoscaler](#)

## Tools

- [Kubernetes Event-driven Autoscaling \(KEDA\)](#)

## Store your secrets in Azure Key Vault, don't inject passwords in Docker Images

Secrets are not encrypted in etcd, prefer to store your secrets in a proper HSM like Azure Key Vault. You can then inject secrets using CSI provider or continue to use FlexVolume if kubernetes < 1.16.

### Documentation

- [Azure Key Vault Provider for Secret Store CSI Driver](#)
- [AKV2K8S](#)
- [Key Vault and flex volumes \(Deprecated\)](#)

## Implement Pod Identity

Don't use fixed credentials within pods or container images, as they are at risk of exposure or abuse. Instead, use pod identities to automatically request access using a central Azure AD identity solution. When pods need access to other Azure services, such as Cosmos DB, Key Vault, or Blob Storage, the pod needs access credentials. These access credentials could be defined with the container image or injected as a Kubernetes secret, but need to be manually created and assigned. Often, the credentials are reused across pods, and aren't regularly rotated. Managed identities for Azure resources (currently implemented as an associated AKS open source project) let you automatically request access to services through Azure AD. You don't manually define credentials for pods, instead they request an access token in real time, and can use it to access only their assigned services.

### Documentation

- [Use pod identity](#)
- [Do not use Pod Identity with kubenet !](#)

## Use Kubernetes namespaces to properly isolate your Kubernetes resources

Namespaces give you the ability to create logical partitions and enforce separation of your resources as well as limit the scope of user permissions. Don't forget not to use the Default namespace

## Documentation

- [Namespaces](#)

## Set up requests and limits on your containers

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.

## Documentation

- [Managing Compute Resources for Containers](#)

## Specify the security context of your pod/container

A security context defines privilege and access control settings for a Pod or Container. Control the capabilities and the rights your container can have. If you don't specify the security context, the pod get the "default" one which may have more rights that it should

## Documentation

- [Configure a Security Context for a Pod or Container](#)

## Conduct Docker Image Builds using Docker Image Security Best Practices

Define a Image build security baseline for your developers to follow.

## Documentation

- [SNYK 10 Docker Image Security Best Practices](#)

## Static Analysis of Docker Images on Build

Introduction of DevSecOps into the environment to promote a proactive security model that starts to shift the responsibility left Ensure static analysis on build is taking place as a starting point

## Documentation

- [Introduction to Azure Defender for container registries](#)

- [Palo Alto CI/CD Integration \(twistcli\)](#)
- [Aquasec CI/CD Integration](#)
- [Qualys CI/CD Integration](#)
- [StackRox CI/CD Integration](#)
- [Clair CI/CD Integration](#)

## Threshold enforcement of Docker Image Builds that contain vulnerabilities

Restrict builds with identified issues. Use a tool that allows for the restriction of builds with enough granularity to not break development. All Critical CVE's are not the same, so being able to restrict builds based on Critical or High vulnerabilities with a Vendor fix, but allowing builds to continue if that Critical vulnerability is "Open"

### Documentation

- [Prisma Threshold enforcement](#)
- [StackRox Advanced Policy Enforcement](#)
- [AquaSec Proactive Risk Management](#)

## Compliance enforcement of Docker Image Builds

Being able to assess and restrict the Compliance state of an image on build. Identifying an image running as "root" before it get deployed, or opening up port 80 or 22

### Documentation

- [Azure Built-In Policy](#)
- [Prisma Managing Compliance](#)
- [Aqua Image Assurance Policies](#)
- [StackRox Compliance](#)

## Disaster Recovery

### Ensure you can perform a whitespace deployment

A whitespace (greenfield) deployment is the exercise to delete everything and to redeploy the whole platform in an automated way.

## Use availability zones

An Azure Kubernetes Service (AKS) cluster distributes resources such as the nodes and storage across logical sections of the underlying Azure compute infrastructure. This deployment model makes sure that the nodes run across separate update and fault domains in a single Azure datacenter.

### Documentation

- [Create an AKS cluster across availability zones](#)

## Plan for multiregion deployment

When you deploy multiple AKS clusters, choose regions where AKS is available, and use paired regions.

### Documentation

- [Plan for multiregion deployment](#)

## Use Azure Traffic Manager to route traffic

Azure Traffic Manager can direct customers to their closest AKS cluster and application instance. For the best performance and redundancy, direct all application traffic through Traffic Manager before it goes to your AKS cluster.

### Documentation

- [Traffic Manager and AKS](#)

## Create a storage migration plan

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

### Documentation

- [Create a storage migration plan](#)
- [Backup, restore and migrate Kubernetes resources including state to another AKS cluster with Velero](#)

## Guaranty SLA for the control plane

Uptime SLA is an optional feature to enable a financially backed, higher SLA for a cluster. It provides you a 99,95% SLA instead of the 99,5% SLO and is relevant for your production clusters

### Documentation

- [Azure Kubernetes Service \(AKS\) Uptime SLA](#)

## Multi Tenancy

### Enable geo-replication for container images

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

### Documentation

- [Enable geo-replication for container images](#)

### Enable traffic management

Azure Traffic Manager can direct customers to their closest AKS cluster and application instance. For the best performance and redundancy, direct all application traffic through Traffic Manager before it goes to your AKS cluster.

### Documentation

- [Use Azure Traffic Manager to route traffic](#)

## Network

### Choose the appropriate network model

For integration with existing virtual networks or on-premises networks, use Azure CNI networking in AKS. This network model also allows greater separation of resources and controls in an enterprise environment but be aware of the impact on the network topology/IP ranges While Kubenet is the default Kubernetes network plugin, the Container Networking Interface (CNI) is a vendor-neutral protocol that lets the container runtime make requests to a network provider. The Azure CNI assigns IP addresses to



Pods and nodes, and provides IP address management (IPAM) features as you connect to existing Azure virtual networks. Each node and pod resource receives an IP address in the Azure virtual network, and no additional routing is needed to communicate with other resources or services.

### Documentation

- [Kubenet vs CNI](#)

## Plan IP addressing carefully

The size of your virtual network and its subnet must accommodate the number of pods you plan to run and the number of nodes for the cluster. As an example, using CNI, you need one IP for each node + one spare for a new node in case of cluster upgrade, and you need an IP for each pod which can represent hundred of IP addresses

### Documentation

- [Plan IP addressing for your cluster](#)

## Distribute ingress traffic

To distribute HTTP or HTTPS traffic to your applications, use ingress resources and controllers. Ingress controllers provide additional features over a regular Azure load balancer, and can be managed as native Kubernetes resources.

### Documentation

- [Distribute ingress traffic](#)

## Secure your exposed applications with a web application firewall (WAF)

If you plan to host exposed applications, to scan incoming traffic for potential attacks, use a web application firewall (WAF) such as Barracuda WAF for Azure or Azure Application Gateway. These more advanced network resources can also route traffic beyond just HTTP and HTTPS connections or basic SSL termination.

### Documentation

- [Secure traffic with a web application firewall \(WAF\)](#)

## Don't expose your load-balancer on Internet if not necessary

There is almost no reason to directly expose the ingress entry point to Internet but by default AKS create a public one. Tell him to create an internal one only.

### Documentation

- [Create an ingress controller to an internal virtual network](#)

## Control traffic flow with network policies

Use network policies to allow or deny traffic to pods. By default, all traffic is allowed between pods within a cluster. For improved security, define rules that limit pod communication. Network policy is a Kubernetes feature that lets you control the traffic flow between pods. You can choose to allow or deny traffic based on settings such as assigned labels, namespace, or traffic port. The use of network policies gives a cloud-native way to control the flow of traffic. As pods are dynamically created in an AKS cluster, the required network policies can be automatically applied. Don't use Azure network security groups to control pod-to-pod traffic, use network policies.

### Documentation

- [Control traffic flow with network policies](#)

### Tools

- [Calico](#)
- [Cilium](#)

## Configure default network policies in each namespace

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

### Documentation

- [Recipes of best default network policies](#)

## Prevent data-leaking with egress lockdown

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

### Documentation

- [Egress traffic requirements](#)

## Operations

### Maintain kubernetes version up to date

To stay current on new features and bug fixes, regularly upgrade to the Kubernetes version in your AKS cluster. Support for kubernetes is current and N-2 versions only

#### Documentation

- [Regularly update to the latest version of Kubernetes](#)

### Keep nodes up to date and patched

AKS supports upgrading the images on a node so you're up to date with the newest OS and runtime updates. AKS provides one new image per week with the latest updates, so it's beneficial to upgrade your node's images regularly for the latest features, including Linux or Windows patches

#### Documentation

- [Azure Kubernetes Service \(AKS\) node image upgrades](#)
- [Process Linux node updates and reboots using Kured](#)

#### Tools

- [Kured \(KUbernetes REboot Daemon\)](#)

### Securely connect to nodes through a bastion host

Don't expose remote connectivity to your AKS nodes. Create a bastion host, or jump box, in a management virtual network. Use the bastion host to securely route traffic into your AKS cluster to remote management tasks.

#### Documentation

- [Securely connect to nodes through a bastion host](#)

### Regularly check for cluster issues

Regularly run the latest version of kube-advisor open source tool to detect issues in your cluster. If you apply resource quotas on an existing AKS cluster, run kube-advisor first to find pods that don't have resource requests and limits defined.

#### Documentation

- [Regularly check for cluster issues with kube-advisor](#)

### Tools

- [Kube-advisor](#)
- [Kube-advisor by Alcide.io](#)

## Monitor the security of your cluster with Azure Security Center

Security Center brings security benefits to your AKS clusters using data already gathered by the AKS master node.

### Documentation

- [Azure Kubernetes Services integration with Security Center](#)

## Provision a log aggregation tool

Ensure that you are always aware of what happens in your cluster. Monitor the health of the cluster (nodes, server) but also the pods

### Documentation

- [Azure Monitor for AKS](#)

### Tools

- [Elastic Cloud](#)
- [Datadog](#)

## Enable master node logs

To help troubleshoot your application and services, you may need to view the logs generated by the master components. Be aware that if you don't enable these logs, there is no way for Microsoft to retrieve them for you

### Documentation

- [Enable and review Kubernetes master node logs](#)

## Collect metrics

If default integration can collect telemetry data and basic metrics (CPU/Memory), they don't collect custom metrics and more detailed information. It's often necessary to install a 3rd party software (prometheus is recommend within Kubernetes) and they store these metrics to exploit them.

### Documentation

- [Configure scraping of Prometheus metrics](#)
- [Deploying ELK](#)

## Configure distributed tracing

Distributed tracing, also called distributed request tracing, is a method used to profile and monitor applications, especially those built using a microservices architecture. Distributed tracing helps pinpoint where failures occur and what causes poor performance.

### Documentation

- [Solution for onboarding Kubernetes/AKS workloads onto Application Insights monitoring.](#)
- [Zero instrumentation application monitoring for Kubernetes hosted applications \(deprecated\)](#)

## Control the compliance with Azure Policies

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

### Documentation

- [Azure Policies for AKS](#)

### Tools

- [Gatekeeper](#)

## Read The Definitive Guide to Securing Kubernetes

This whitepaper provides an overview of key aspects and best practices of K8s security

### Documentation

- [The Definitive Guide to Securing Kubernetes](#)

## Tools

- [Gatekeeper](#)

## Enable Azure Defender for Kubernetes

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

### Documentation

- [Introduction to Azure Defender for Kubernetes](#)

## Use Azure Key Vault

Use Azure Key Vault to store Secrets and Certificates

### Documentation

- [Tutorial: Configure and run the Azure Key Vault provider for the Secrets Store CSI driver on Kubernetes](#)

## Use GitOps

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

### Documentation

- [Guide To GitOps](#)

## Make your life easier with K8S Tools

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

## Tools

- [Helm](#)
- [kubectl aliases](#)
- [kubectx](#)
- [k9s](#)

## Don't use the default namespace

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

## Resource Management

### Enforce resource quotas

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.

#### Documentation

- [Enforce resource quotas](#)
- [Resources quotas](#)

### Set memory limits and requests for all containers

Set CPU and memory limits and requests to all the containers. It prevents memory leaks and CPU over-usage and protects the whole platform

#### Documentation

- [Assign Memory Resources to container](#)

### Configure pod disruption budgets

To maintain the availability of applications, define Pod Disruption Budgets (PDBs) to make sure that a minimum number of pods are available in the cluster.

#### Documentation

- [Plan for availability using pod disruption budgets](#)
- [Specifying a Disruption Budget for your Application](#)

### Set up cluster auto-scaling

To maintain the availability of applications and guaranty available resources, set up cluster auto-scaling

#### Documentation

- [Use AKS cluster auto-scale](#)

## Storage

### Choose the right storage type

Understand the needs of your application to pick the right storage. Use high performance, SSD-backed storage for production workloads. Plan for network-based storage when there is a need for multiple concurrent connections.

#### Documentation

- [Choose the right storage type](#)

### Size the nodes for storage needs

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.

#### Documentation

- [Size the nodes for storage needs](#)

### Dynamically provision volumes

To reduce management overhead and let you scale, don't statically create and assign persistent volumes. Use dynamic provisioning. In your storage classes, define the appropriate reclaim policy to minimize unneeded storage costs once pods are deleted.

#### Documentation

- [Dynamically provision volumes](#)

### Secure and back up your data

Back up your data using an appropriate tool for your storage type, such as Velero or Azure Site Recovery Understand the limitations of the different approaches to data



backups and if you need to quiesce your data prior to snapshot. Data backups don't necessarily let you restore your application environment of cluster deployment.

### Documentation

- [Secure and back up your data](#)

## Make your storage resilient

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.

### Documentation

- [Remove service state from inside containers](#)

## Windows

### Map the base image to node OS

Windows has strict compatibility rules, where the host OS version must match the container base image OS version. Only Windows containers with a container operating system of Windows Server 2019 are supported.

### Documentation

- [Windows container version compatibility](#)
- [Limitations of Windows containers](#)

### Prepare your application for an abrupt kill

TerminationGracePeriod is not implemented on Windows containers

### Documentation

- [Understand pod lifecycle](#)
- [Limitations of Windows containers](#)

## Don't use privileged containers

Windows containers do not support elevation of privilege

### Documentation

- [Limitations of Windows containers](#)

## Watch for memory usage

There are no OOM eviction actions taken by the kubelet. Windows always treats all user-mode memory allocations as virtual, and pagefiles are mandatory. If memory is over-provisioned and all physical memory is exhausted, then paging can slow down performance.

### Documentation

- [Limitations of Windows containers](#)

## Implement CNI network mode

AKS clusters with Windows node pools must use the Azure CNI (advanced) networking model. Kubenet (basic) networking is not supported.

### Documentation

- [What network plug-ins are supported?](#)

## Patch your nodes yourself

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.

### Documentation

- [How do patch my Windows nodes?](#)

## Secure the traffic of your containers

Network policies are currently not supported, ensure that the containerized applications have a layer of protection like authentication

### Documentation

- [Limitations of Windows containers](#)