**Design Considerations & Hardening Azure Kubernetes Services**

Whenever we like to setup Azure Kubernetes environment, we must consider the usability within our organization. Within our organization we will probably have to deal with compliancy and security guidelines, usually for good reasons.

I have chosen solutions based on my own experience and knowledge. This document can be used as a tool for communication when addressing security on AKS.

This Document is to provide an initial guide for hardening our managed Kubernetes environment on Azure (AKS), based on following Content.

1) [AKS Security](https://dev.azure.com/dude-projects/compliant-aks/_wiki/wikis/compliant-aks.wiki/84/Setup-a-secure-AKS-environment?anchor=azure-networking#aks-security)

2) [Networking](https://dev.azure.com/dude-projects/compliant-aks/_wiki/wikis/compliant-aks.wiki/84/Setup-a-secure-AKS-environment?anchor=azure-networking#networking)

3) [Monitoring & Alerting](https://dev.azure.com/dude-projects/compliant-aks/_wiki/wikis/compliant-aks.wiki/84/Setup-a-secure-AKS-environment?anchor=azure-networking#monitoring-%26-alering)

4) [AKS\_RACI](https://dev.azure.com/dude-projects/compliant-aks/_wiki/wikis/compliant-aks.wiki/84/Setup-a-secure-AKS-environment?anchor=azure-networking#policies)

1. [AKS Security](https://dev.azure.com/dude-projects/compliant-aks/_wiki/wikis/compliant-aks.wiki/84/Setup-a-secure-AKS-environment?anchor=azure-networking#aks-security)

**Authentication and Authorization**

1. Cluster Level –Identity and Access Management through AAD and Kubernetes RBAC

• Kubernetes Developer authenticates with AAD.

• The AAD token issuance endpoint issues the access token.

• Developer performs action w/ AAD token. E.g. Kubectl create pod.

• Kubernetes validates token with AAD and fetches the Developer’s AAD Groups for example: Dev Team A, App Group B.

• Kubernetes RBAC and cluster policies are applied • Request is successful or not based on the previous validation.

1. Azure Level –Identity and Access Management through AAD and RBAC

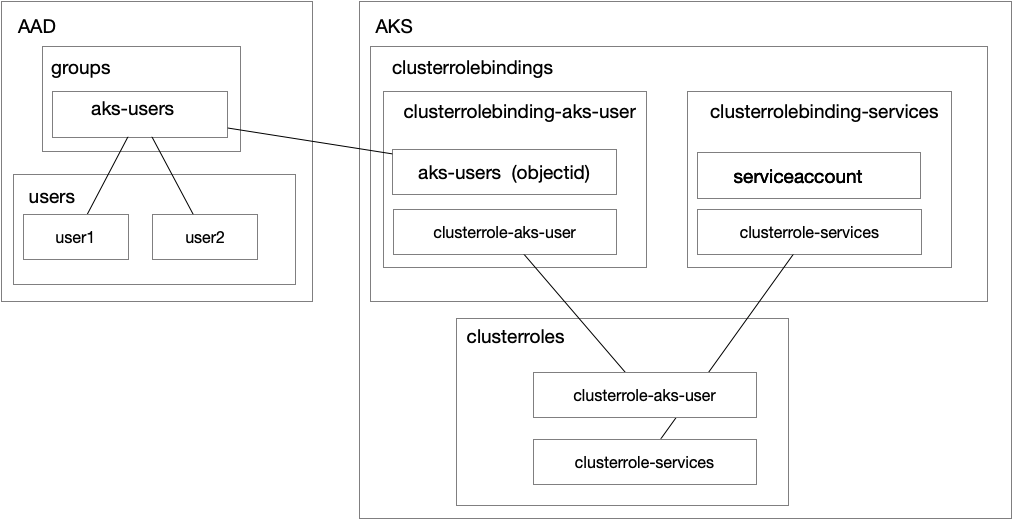
• Kubernetes Administrator authenticates with AAD.

• The AAD token issuance endpoint issues the access token.

• Administrator fetches the admin kubeconfig and configures RBAC roles and bindings.

• Kubernetes Developer fetches the user kubeconfig.

**AKS AD Integration and Kubernetes RBAC**



Note: Please refer for AKD AD Integration to:

<https://docs.microsoft.com/en-us/azure/aks/azure-ad-integration>

**Admission Controllers**

After we authenticated and Authorized we have the ability to enforce rules/ policies in Kubernetes using [Admission Controllers](https://kubernetes.io/docs/reference/access-authn-authz/extensible-admission-controllers/).

An admission controller is a piece of code that intercepts requests to the Kubernetes API server prior to persistence of the object, but after the request is authenticated and authorized. AKS supports the following admission controllers (as of now it is not modifiable):

* NamespaceLifecycle
* LimitRanger
* ServiceAccount
* DefaultStorageClass
* DefaultTolerationSeconds
* MutatingAdmissionWebhook
* ValidatingAdmissionWebhook
* ResourceQuota
* DenyEscalatingExec
* AlwaysPullImages

1. [Networking](https://dev.azure.com/dude-projects/compliant-aks/_wiki/wikis/compliant-aks.wiki/84/Setup-a-secure-AKS-environment?anchor=azure-networking#networking)

AKS clusters can be deployed with one of the following two network models:

• Kubenet networking (Basic) -The network resources are typically created and configured as the AKS cluster is deployed.

• Azure Container Networking Interface (CNI) networking (Advanced) -The AKS cluster is connected to existing virtual network resources and configurations

## **Preparation**

### AKS subnet size

AKS uses a subnet to host nodes, pods, and any other Kubernetes and Azure resources that are created for the AKS cluster. As such, it is extremely important that the subnet is appropriately sized, to ensure it can accommodate the resources that will be initially created, and still have enough room for any future updates.

There are two networking methods available when deploying an Azure Kubernetes Service cluster

* Kubenet
* Azure Container Networking Interface (CNI)

AKS uses kubnet by default, and in doing so, it automatically creates a virtual network and subnets that are required to host the pods in. This is a great solution if you are learning about AKS, however if you need more control, it is better to go with Azure CNI. With Azure CNI, you get the option to use an existing virtual network and subnet or you can create a custom one. This is a much better option, especially when deploying into a production environment.

The formula below provides a good estimate on how large your subnet must be, in order to accommodate your AKS resources.

*Subnet size = (number of nodes + 1) + ((number of nodes + 1) \* maximum number of pods per node that you configure)*

When using Azure CNI, by default each node is setup to run 30 pods. If you need to change this limit, you will have to deploy your AKS cluster using Azure CLI or Azure Resource Manager templates.

Just as an example, for a default AKS cluster deployment, using Azure CNI with 4 nodes, the subnet size at a minimum must be

IPs required = (4 + 1) + ((4+ 1) \* (30 pods per node)) = 5 + (5 \* 30) = 155

This means that the subnet must be at least a /24.

For this blog, create a new resource group called **myAKS-resourcegroup**. Within this new resource group, create a virtual network called **AKSVNet**with an address space of **10.1.0.0/16**. Inside this virtual network, create a subnet called **AKSSubnet1** with an address range of **10.1.3.0/24**.

Network Design to Secure Applications and APIs

Define Requirements for example:

* Advanced/Basic Networking
* Do not expose any Azure Public IPs and force all Internet Ingress and Egress through Azure Firewall or NVAs such as Palo Alto
* Azure Application Gateway Web Access Firewall or third party WAF such as Akamai for all External Web Apps hosted at AKS
* Azure API Management Gateway or other Third-Party API Management Gateway such as SAP API Management Gateway for Externally accessible APIs hosted at AKS
* NGINX Ingress Controllers
* High Availability
* Azure Firewall or Palo Alto NVA for North South Traffic Management
* AKS Network Policies for Intra AKS Cluster east west traffic
* NSG to control AKS VNET traffic
* Ideally NVAs and AKS in different subscriptions, resource groups and VNETs as per Azure Architecture
* Do not expose any Azure Public IPs
* Internal applications shall only be accessible internally without exposing them to public

Azure Limitations:

* AKS by default creates Azure Load Balancer for Ingress Controllers or any Public Facing AKS Services with Basic SKU
* Azure has a limitation that Basic SKU Private ALB Frontend IP is not reachable by other peered VNETs
* As a result, Azure Firewall or Palo Alto NVAs running in other VNETs and Subscription can’t reach to Basic SKU Private ALB Frontend IP
* Azure AKS Preview mode allows to create ALB with Standard SKU, but by default create Public ALB defeating the purpose of using private IP Addresses.

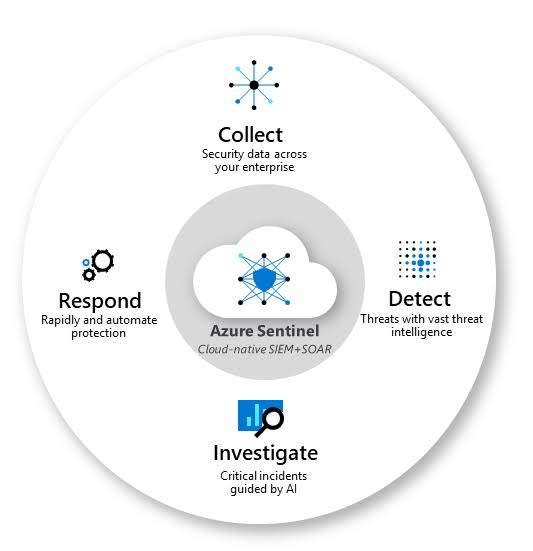
1. [Monitoring & Alerting](https://dev.azure.com/dude-projects/compliant-aks/_wiki/wikis/compliant-aks.wiki/84/Setup-a-secure-AKS-environment?anchor=azure-networking#monitoring-%26-alering)

AKS Monitoring through Azure Monitor workspace and develop a notification and alert process based on performance counters and other desired parameters:

* Key Metrics:
* Node metrics (CPU Usage, Memory Usage, Disk Usage, Network Usage)
* Kube\_node\_status\_condition
* Pod memory usage / limit; memory\_failures\_total
* container\_memory\_working\_set\_bytes
* Pod CPU usage average / limit
* Filesystem Usage
* Network receive / transmit errors
* Azure Monitor Metrics available today:
* Kube-controller-manager
* Kube-api-server
* Kube-scheduler
* Audit logs on the roadmap

### Azure Security Center

### [Azure Security Center](https://docs.microsoft.com/en-us/azure/security-center/security-center-intro)  or [Azure Sentinel](https://azure.microsoft.com/nl-nl/services/azure-sentinel/)  (for more SEAM like behavior) are great offering to consider continuous monitoring of security related threats



1. [AKS\_RACI](https://dev.azure.com/dude-projects/compliant-aks/_wiki/wikis/compliant-aks.wiki/84/Setup-a-secure-AKS-environment?anchor=azure-networking#policies)

The **RACI matrix** is a responsibility assignment chart that maps out every task, milestone or key decision involved in completing a project and assigns which roles are Responsible for each action item, which personnel are Accountable, and, where appropriate, who needs to be Consulted or Informed:

