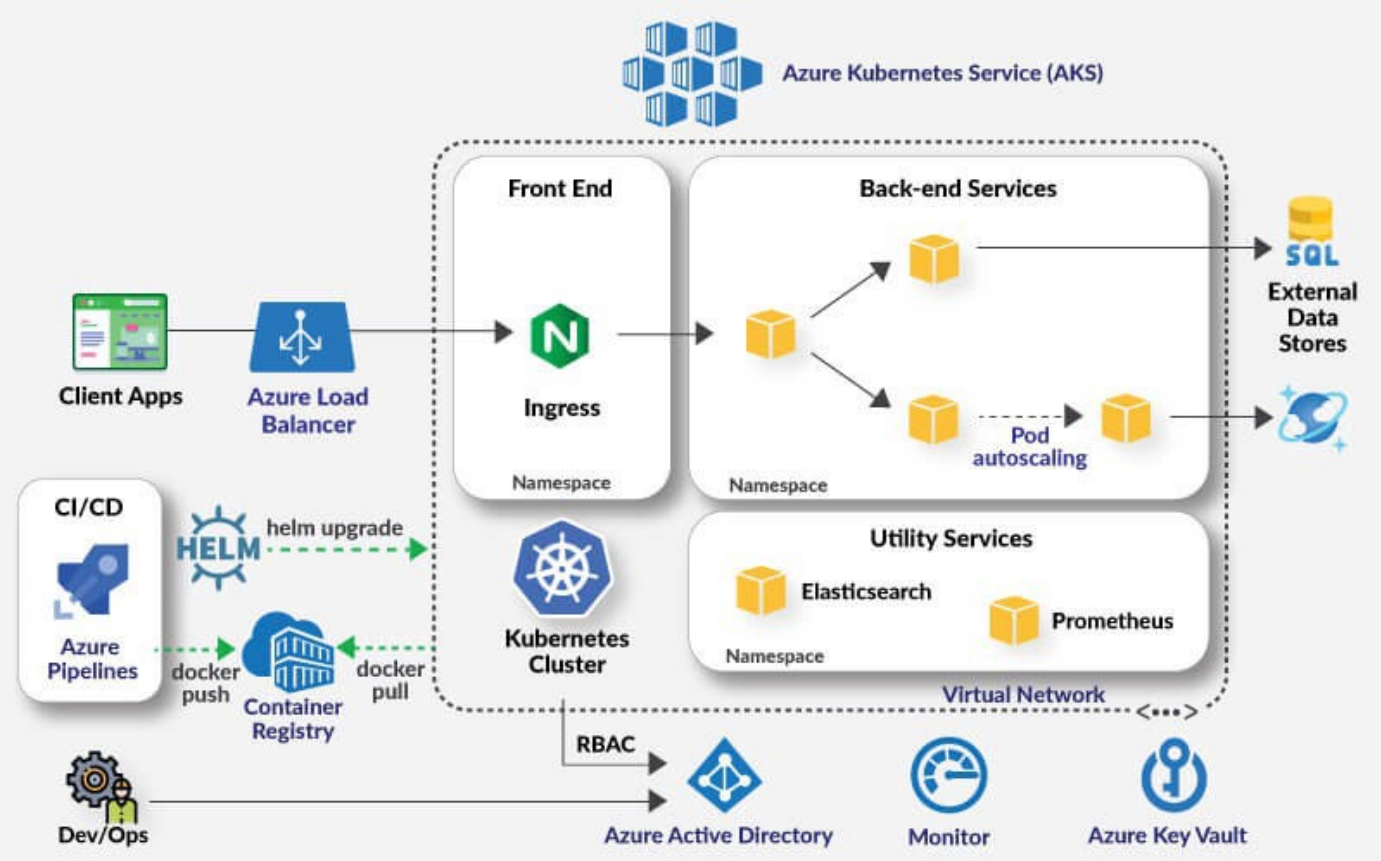
Azure Kubernetes Service (AKS)

What is Azure Kubernetes Service?

AKS is an open-source fully managed container orchestration service that became available in June 2018 and is available on the Microsoft Azure public cloud that can be used to deploy, scale and manage Docker containers and container-based applications in a cluster environment.

Azure Kubernetes Service offers provisioning, scaling, and upgrades of resources as per requirement or demand without any downtime in the Kubernetes cluster and the best thing about AKS is that you don’t require deep knowledge and expertise in container orchestration to manage AKS.

AKS is certainly an ideal platform for developers to develop their modern applications using Kubernetes on the Azure architecture where Azure Container Instances are the pretty right choice to deploy containers on the public cloud. The Azure Container Instances help in reducing the stress on developers to deploy and run their applications on Kubernetes architecture.



Features of Azure Kubernetes Service (AKS)

Managed AKS:

- AKS simplifies the creation and delivery of cloud-native/container-based applications, which falls between the IaaS and PaaS level. Built-in features enable less administrative work while providing an option for deploying the serverless Azure function to the AKS cluster.
- The AKS Master Node (API Server, Scheduler, Controller Manager, etc.) is completely managed by Microsoft
- AKS supports both Windows and Linux based container deployment

Networking:

- AKS is deployed into a virtual network subnet with several Azure network services enabled
- VNET is completely supported by AKS
- Azure Load Balancer Secures AKS cluster environment in a private cluster and balances traffic within a virtual network
- Ingress controller: For reverse-proxy feature, routing traffic, etc.
- Network model: Based on the network model requirement for IP address pace, pod communication, Azure network policy and virtual notes, you can pick Kubernetes or Azure CNI (recommended by Azure CNI)

Storage:

- Azure Kubernetes supports multiple storage solutions (persistent volume) such as Azure Disk Storage and Azure File for concurrent data access, NFA and Azure NetApp Files

Scalability:

- Azure Kubernetes Service (AKS) is a leading Kubernetes managed solution
- It quickly helps to spin-up new clusters and it is very easy to maintain and run the clusters
- AKS completely supports a container's scalability

Hybrid Cloud:

To support the hybrid cloud deployment model, Kubernetes uses Azure Arc and Azure Stack to provide a more stable and faster environment

Azure DevOps:

- DevOps provides an agile board, repository, AKS container application pipeline and release
- Azure Monitor and AKS container monitoring provide information for monitoring containers such as processor, memory, log, etc.

Security:

- Key security options are available to secure the AKS environment. AKS supports Azure Active Directory (AD), Service Principal and Role-Based Access Control (RBAC) authentication.
- Azure Application Gateway configuration option in Ingress
- Images are scanned using third-party tools and these scanned images are securely stored in the container registry
- Setup network policy to secure communication paths between namespaces and nodes
- Restricted access to the configuration of VNET, whitelist and blocklist in AKS
- Compliance with certifications such as SOC, HIPAA, PCI, etc.

Cost:

- The Azure Master node is fully run by Microsoft for free
- AKS pricing models are VM Node and Scale Set

Azure Kubernetes Service networking and security

Networking and Multi Tenancy

In this section we will discuss the below topics:

- Which network plugin to choose?
- Pros and Cons of network models
- Setting up ingress
- Securing traffic with a web application firewall (WAF)
- Control traffic flow with network policies
- Securely connect to nodes through a bastion host
- Private and Public Clusters

Which network plugin to choose?

While deploying AKS, one of the most asked questions is which networking model to use, Kubenet or Azure CNI(Container Networking Interface). It is an important decision to make as once AKS cluster is created with one of the networking models then there is no turning back as it can't be changed later. AKS offers 2 network plugins Azure CNI and Kubenet

When it comes to multi-tenancy, both options work and will deliver the same outcomes, however, in the spirit of completeness below is an easy criteria to decide which one to use:

- If you don't have any shortage with IPv4 allocation, use Azure CNI, it offers an out of the box experience and pretty easy to manage.
- If the workload isn't performance sensitive in terms of throughput, i.e. the pod will be moving large data sets to external services like a database. Use Azure CNI as it offers VM like performance. While Kubenet offers great performance, you won't be getting the a VM like network performance from your pods.
- If you have problems with IP allocation i.e. network team only provides small subnets, then Kubenet would be ideal solution as the pods will be assigned non-routable IP space.

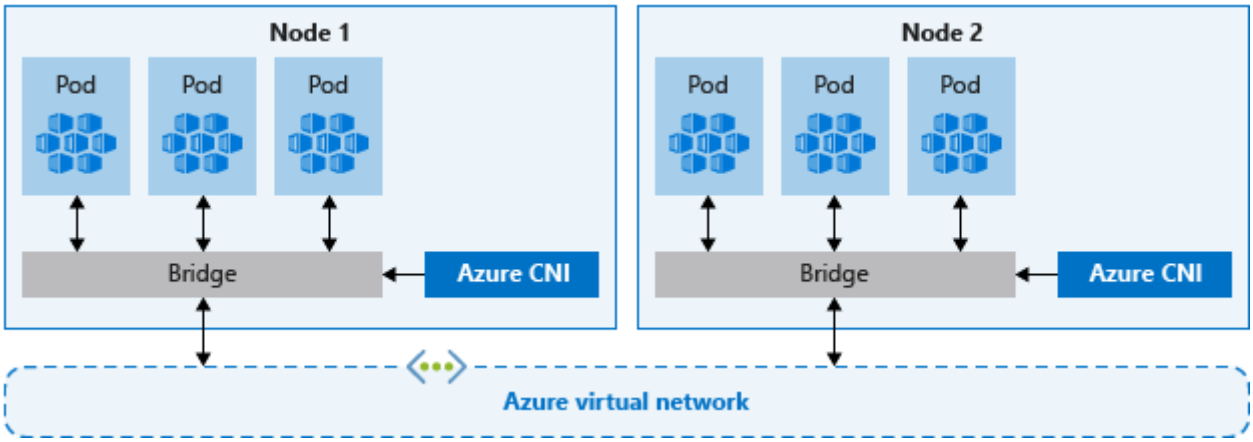
Azure CNI networking

Azure CNI is a vendor-neutral protocol that lets the container runtime make requests to a network provider. It 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 - no need for extra routing to communicate with other resources or services.
In case of Azure CNI, pods get IP addresses from the subnet and can be accessed directly. These IP addresses are always unique across the network space and are always planned in advance. There is a maximum limit for the number of pods that a node can support. Hence, number of IP addresses per node is then reserved upfront for that node.
When you use Azure CNI networking, the virtual network resource is in a separate resource

group to the AKS cluster. Delegate permissions for the AKS cluster identity to access and manage these resources. The cluster identity used by the AKS cluster must have at least Network Contributor (https://docs.microsoft.com/en-us/azure/role-based-access-control/built-in-roles#network-contributor) permissions on the subnet within your virtual network.

Each node and pod receives its own IP address,so plan out the address ranges for the AKS subnets. In this case:

- The subnet must be large enough to provide IP addresses for every node, pods, and network resource that you deploy.
- Each AKS cluster must be placed in its own subnet.
- Avoid using IP address ranges that overlap with existing network resources.
- To handle scale out events or cluster upgrades, you need extra IP addresses available in the assigned subnet.



Kubenet networking

Kubenet networking option is the default configuration for AKS cluster creation. In which, nodes get an IP address from the Azure virtual network subnet and pods receive IP addresses from a logically different address space to the Azure virtual network subnet of the nodes. For the reachabilty of the pods to Azure Vnet, Network address translation (NAT) is configured in the background.

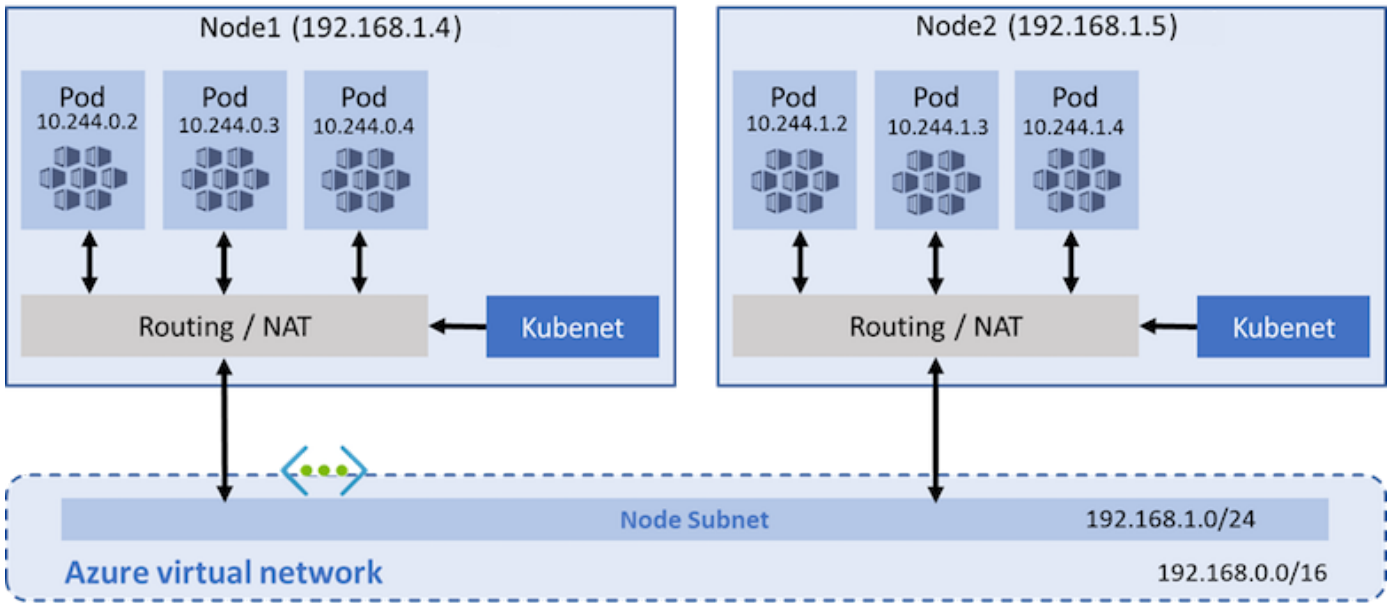
Although kubenet doesn't require you to set up the virtual networks before the cluster is deployed, there are disadvantages to waiting:

- Since nodes and pods are placed on different IP subnets, User Defined Routing (UDR) and IP forwarding routes traffic between pods and nodes. This extra routing may reduce network performance.
- Connections to existing on-premises networks or peering to other Azure virtual networks can be complex.

In most cases, Kubenet is using for:

- Small development or test workloads.

- Simple websites with low traffic.
- Lifting and shifting workloads into containers.



Pros and Cons of network models

Azure CNI

Pros:

- Pods can be reached to connected networks by their private IP addresses.
- Azure network policy is supported.
- Limitation of 250 pods per node.
- Supports windows server node pools.
- Expose pods using cluster subnet IPs. This can be negative when you are connecting through pod IPs internally.
- VM to pod and on-prem to pod using VPN or express route is supported both ways.

Cons:

- Require bigger IP address space as it uses at least 30 IP addresses per node.
- Doesn’t support CIDR and advance customisations.

Kubenet networking

Pros:

- Conserve IP address spaces.
- Requires Kubernetes services and load balancer to expose Pods.
- User defined routes(UDR) are created and maintained manually.
- Maximum of 400 nodes per cluster.

- Support both basic and standard Azure load balancers.
- Calico network policy is supported.
- Both new or existing subnets are supported.

Cons:

- VM to pod and on-prem to pod using VPN or express route is not supported, however, other way round is supported.
- Doesn’t support Azure network policies.
- Limitation of 110 pods per node.
- Doesn’t support windows server node pools.
- Doesn’t support CIDR and advance customisations.

Ingress setup

Normally any Kubernetes cluster will make use on an Ingress Controller of sort, and there are many options in the market, we won’t be discussing which one is best but we will cover the different deployment options to achieve safe multi-tenancy.

1. Single Shared Ingress across all tenants, in this model one ingress controller “deployment” will be used to serve all tenants. important things to consider

- Use dedicated node pool for ingress controller, ingress controller is normally one of the most heavily used resources in your cluster, dedicating a node pool for the ingress is a good practice to achieve good level of availability and scaling, consider this guidance for any shared system tool that you will be using i.e. Monitoring.
- Ensure proper scaling is taking place for your ingress controller
- If charge back is a requirement then the easiest option is to charge a flat rate per tenant for the shared ingress.

2. Dedicated ingress controller per tenant in this model each tenant will get its own ingress controller dedicated to its workloads

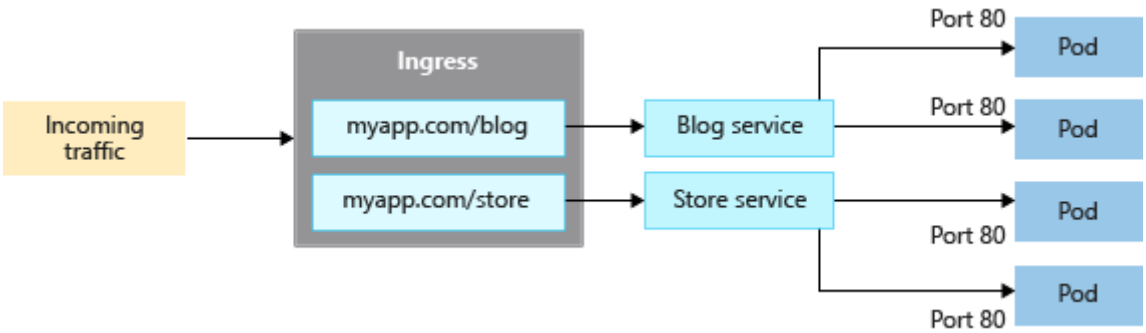
- The ingress controller can live in the tenant namespace
- The ingress controller can be deployed in the tenant node pool assuming node pools is your tenancy model of choice
- You need to make use of the class object in order to ensure traffic is handled by the correct ingress
- You can still deploy all the different ingresses in one dedicated node pool similar to the previous model

To distribute HTTP or HTTPS traffic to your applications, use ingress resources and controllers. Compared to an Azure load balancer, ingress controllers provide extra features and can be managed as native Kubernetes resources.

While an Azure load balancer can distribute customer traffic to applications in your AKS cluster, it's limited in understanding that traffic. A load balancer resource works at layer 4, and distributes traffic based on protocol or ports.

Most web applications using HTTP or HTTPS should use Kubernetes ingress resources and controllers, which work at layer 7. Ingress can distribute traffic based on the URL of the application and handle TLS/SSL termination. Ingress also reduces the number of IP addresses you expose and map.

With a load balancer, each application typically needs a public IP address assigned and mapped to the service in the AKS cluster. With an ingress resource, a single IP address can distribute traffic to multiple applications.



There are two components of ingress:

- An ingress *resource*
- An ingress *controller*

Ingress resource

The ingress resource is a YAML manifest of kind: Ingress. It defines the host, certificates, and rules to route traffic to services running in your AKS cluster.

```
metadata:
 name: myapp-ingress
   annotations: kubernetes.io/ingress.class: "PublicIngress"
spec:
 tls:
 - hosts:
   - myapp.com
   secretName: myapp-secret
 rules:
   - host: myapp.com
     http:
      paths:
      - path: /blog
        backend:
         serviceName: blogservice
         servicePort: 80
      - path: /store
        backend:
         serviceName: storeservice
         servicePort: 80
```

Ingress controller

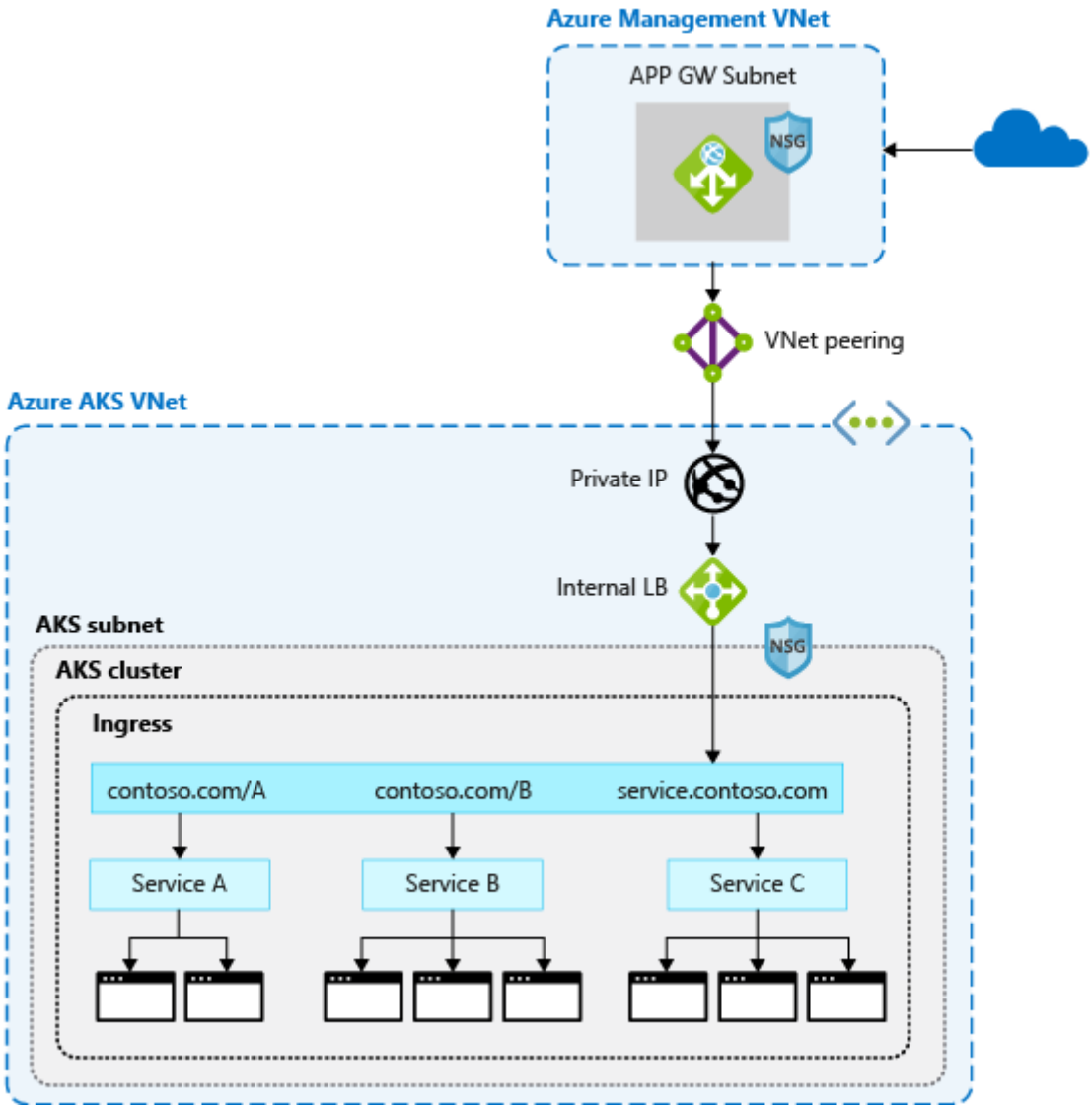
An ingress controller is a daemon that runs on an AKS node and watches for incoming requests. Traffic is then distributed based on the rules defined in the ingress resource. The most common ingress controller is based on NGINX (https://www.nginx.com/products/nginx-ingress-controller).

Ingress controllers must be scheduled on a Linux node. Indicate that the resource should run on a Linux-based node using a node selector in your YAML manifest or Helm chart deployment.

Secure traffic with a web application firewall (WAF)

An ingress controller is a Kubernetes resource in your AKS cluster that distributes traffic to services and applications. The controller runs as a daemon on an AKS node, and consumes some of the node's resources, like CPU, memory, and network bandwidth. In larger environments, you'll want to:

- Offload some of this traffic routing or TLS termination to a network resource outside of the AKS cluster.
- Scan incoming traffic for potential attacks.



Web application firewall (WAF) is used to scan incoming traffic for potential attacks. By default it is Azure Application Gateway. These more advanced network resources can also route traffic beyond just HTTP and HTTPS connections or basic TLS termination.

For that extra layer of security, a web application firewall (WAF) filters the incoming traffic. With a set of rules, the Open Web Application Security Project (OWASP) watches for attacks like cross-site scripting or cookie poisoning. Azure Application Gateway (currently in preview in AKS) is a WAF that integrates with AKS clusters, locking in these security features before the traffic reaches your AKS cluster and applications.

Control traffic flow with network policies

Network policy is a Kubernetes feature available in AKS that lets you control the traffic flow between pods. You allow or deny traffic to the pod based on settings such as assigned labels, namespace, or traffic port. Network policies are a cloud-native way to control the flow of traffic for pods. As pods are dynamically created in an AKS cluster, required network policies can be automatically applied.

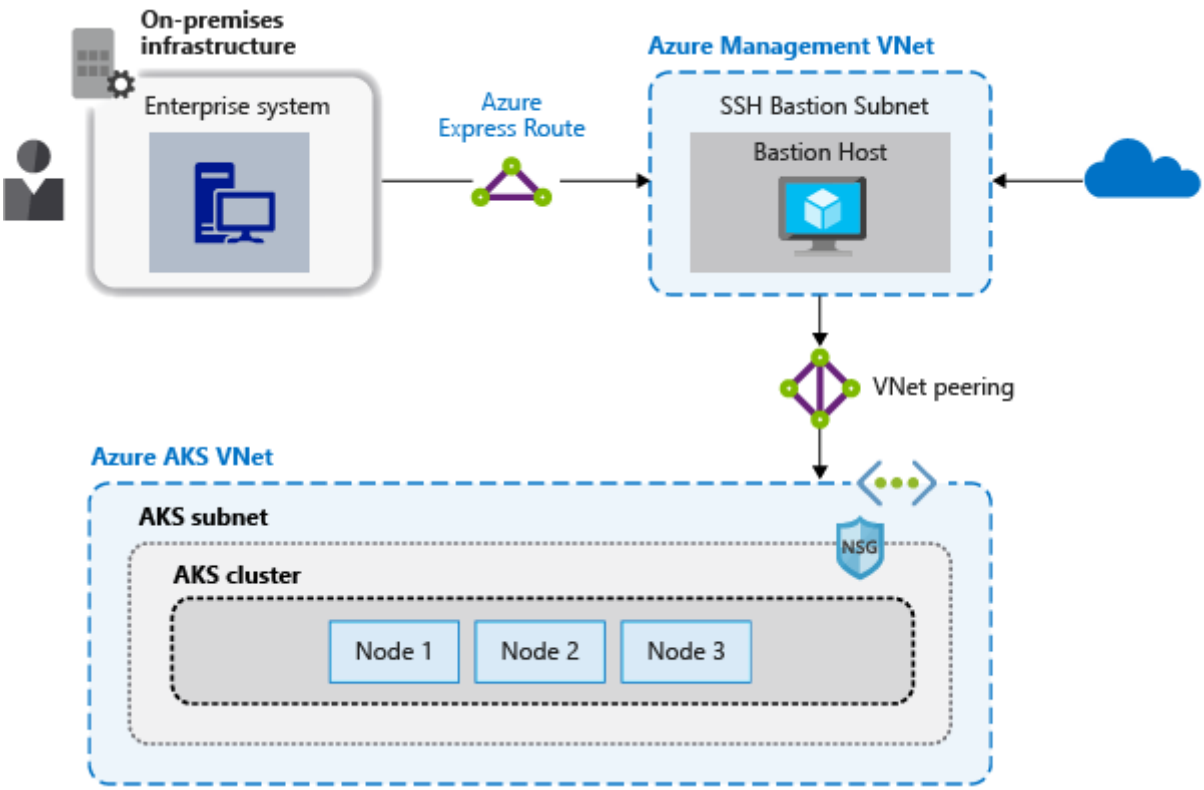
To use network policy, enable the feature when you create a new AKS cluster. You can't enable network policy on an existing AKS cluster. Plan ahead to enable network policy on the necessary clusters.

> Network policy should only be used for Linux-based nodes and pods in AKS.

You create a network policy as a Kubernetes resource using a YAML manifest. Policies are applied to defined pods, with ingress or egress rules defining traffic flow.
The following example applies a network policy to pods with the app: backend label applied to them. The ingress rule only allows traffic from pods with the app: frontend label:

```
apiVersion: networking.k8s.io/v1
metadata:
  name: backend-policy
spec:
  podSelector:
    matchLabels:
      app: backend
  ingress:
  - from:
    - podSelector:
        matchLabels:
          app: frontend
```

Securely connect to nodes through a bastion host



You can complete most operations in AKS using the Azure management tools or through the Kubernetes API server. AKS nodes are only available on a private network and aren't connected to the public internet. To connect to nodes and provide maintenance and support, route your connections through a bastion host, or jump box. Verify this host lives in a separate, securely-peered management virtual network to the AKS cluster virtual network.

Private and Public Clusters and Services

Clusters

By private or public cluster we refer to the API server being assigned either a public IP which is the default behavior in AKS or Private IP using private link in AKS in both cases worker nodes are always assigned Private IP address.

In relation to multi-tenancy the choice won't make much difference, the only difference is that with a multi-tenant cluster you will have many personnel authorized to access the API server so the surface is a bit large, but then even in public API server you can still use authorized ip ranges to ensure access is only grated to your offices or VPN IPs.