Networking concepts for applications in Azure Kubernetes Service (AKS)

Article • 02/27/2024

In a container-based, microservices approach to application development, application components work together to process their tasks. Kubernetes provides various resources enabling this cooperation:

- You can connect to and expose applications internally or externally.
- You can build highly available applications by load balancing your applications.
- You can restrict the flow of network traffic into or between pods and nodes to improve security.
- You can configure Ingress traffic for SSL/TLS termination or routing of multiple components for your more complex applications.

This article introduces the core concepts that provide networking to your applications in AKS:

- Services and ServiceTypes
- Azure virtual networks
- Ingress controllers
- Network policies

Kubernetes networking basics

Kubernetes employs a virtual networking layer to manage access within and between your applications or their components:

- **Kubernetes nodes and virtual network**: Kubernetes nodes are connected to a virtual network. This setup enables pods (basic units of deployment in Kubernetes) to have both inbound and outbound connectivity.

- **Kube-proxy component**: kube-proxy runs on each node and is responsible for providing the necessary network features.

Regarding specific Kubernetes functionalities:

- **Services**: Services is used to logically group pods, allowing direct access to them through a specific IP address or DNS name on a designated port.
- **Service types**: Specifies the kind of Service you wish to create.

- **Load balancer**: You can use a load balancer to distribute network traffic evenly across various resources.
- **Ingress controllers**: These facilitate Layer 7 routing, which is essential for directing application traffic.
- **Egress traffic control**: Kubernetes allows you to manage and control outbound traffic from cluster nodes.
- **Network policies**: These policies enable security measures and filtering for network traffic in pods.

In the context of the Azure platform:

- Azure streamlines virtual networking for AKS (Azure Kubernetes Service) clusters.
- Creating a Kubernetes load balancer on Azure simultaneously sets up the corresponding Azure load balancer resource.
- As you open network ports to pods, Azure automatically configures the necessary network security group rules.
- Azure can also manage external DNS configurations for HTTP application routing as new Ingress routes are established.

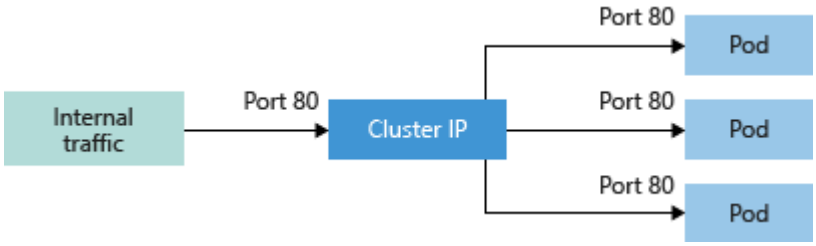
Services

To simplify the network configuration for application workloads, Kubernetes uses *Services* to logically group a set of pods together and provide network connectivity. You can specify a Kubernetes *ServiceType* to define the type of Service you want. For example, if you want to expose a Service on an external IP address outside of your cluster. For more information, see the Kubernetes documentation on Publishing Services (ServiceTypes) .

The following ServiceTypes are available:

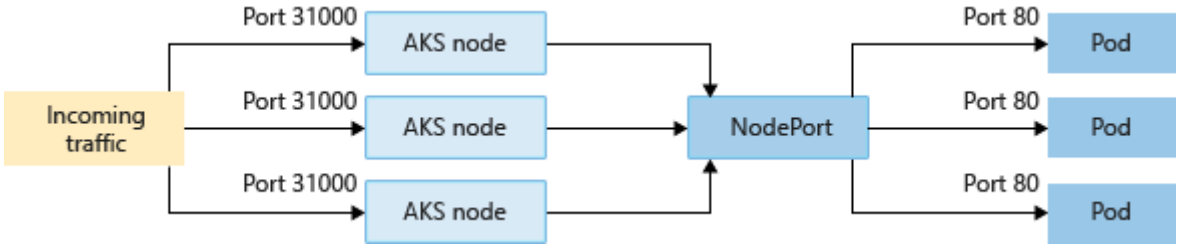
- **ClusterIP**

  ClusterIP creates an internal IP address for use within the AKS cluster. The ClusterIP Service is good for *internal-only applications* that support other workloads within the cluster. ClusterIP is the default used if you don't explicitly specify a type for a Service.



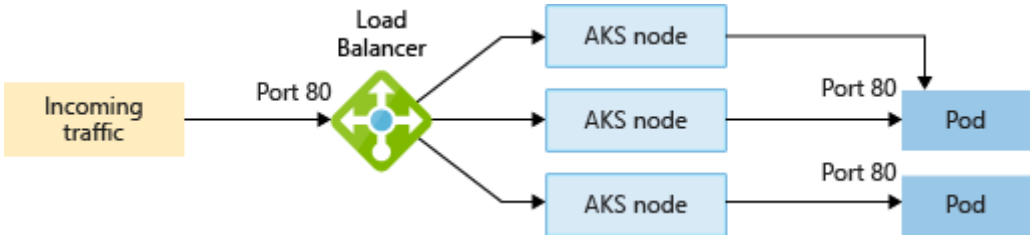
- **NodePort**

  NodePort creates a port mapping on the underlying node that allows the application to be accessed directly with the node IP address and port.



- **LoadBalancer**

  LoadBalancer creates an Azure load balancer resource, configures an external IP address, and connects the requested pods to the load balancer backend pool. To allow customers' traffic to reach the application, load balancing rules are created on the desired ports.



  For HTTP load balancing of inbound traffic, another option is to use an Ingress controller.

- **ExternalName**

  Creates a specific DNS entry for easier application access.

Either the load balancers and services IP address can be dynamically assigned, or you can specify an existing static IP address. You can assign both internal and external static IP addresses. Existing static IP addresses are often tied to a DNS entry.

You can create both *internal* and *external* load balancers. Internal load balancers are only assigned a private IP address, so they can't be accessed from the Internet.

Learn more about Services in the Kubernetes docs .

Azure virtual networks

In AKS, you can deploy a cluster that uses one of the following network models:

- ***Kubenet*** **networking**

The network resources are typically created and configured as the AKS cluster is deployed.

- ***Azure Container Networking Interface (CNI)*** **networking**

  The AKS cluster is connected to existing virtual network resources and configurations.

Kubenet (basic) networking

The *kubenet* networking option is the default configuration for AKS cluster creation. With *kubenet*:

1. Nodes receive an IP address from the Azure virtual network subnet.
2. Pods receive an IP address from a logically different address space than the nodes' Azure virtual network subnet.
3. Network address translation (NAT) is then configured so that the pods can reach resources on the Azure virtual network.
4. The source IP address of the traffic is translated to the node's primary IP address.

Nodes use the kubenet Kubernetes plugin. You can let the Azure platform create and configure the virtual networks for you, or choose to deploy your AKS cluster into an existing virtual network subnet.

Only the nodes receive a routable IP address. The pods use NAT to communicate with other resources outside the AKS cluster. This approach reduces the number of IP addresses you need to reserve in your network space for pods to use.

> **Note**
>
> While kubenet is the default networking option for an AKS cluster to create a virtual network and subnet, it isn't recommended for production deployments. For most production deployments, you should plan for and use Azure CNI networking due to its superior scalability and performance characteristics.

For more information, see Configure kubenet networking for an AKS cluster.

Azure CNI (advanced) networking

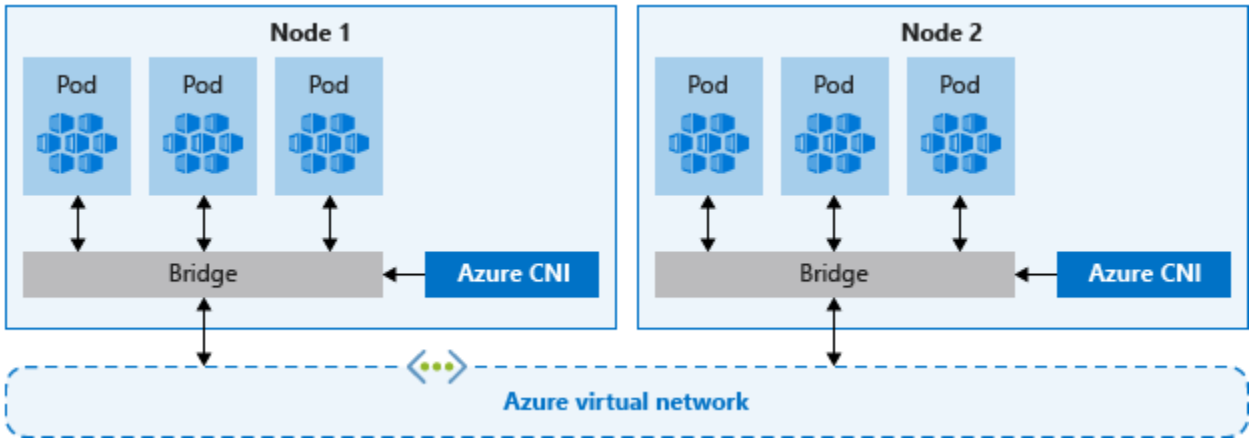
With Azure CNI, every pod gets an IP address from the subnet and can be accessed directly. These IP addresses must be planned in advance and unique across your network space. Each node has a configuration parameter for the maximum number of

pods it supports. The equivalent number of IP addresses per node are then reserved up front. This approach can lead to IP address exhaustion or the need to rebuild clusters in a larger subnet as your application demands grow, so it's important to plan properly. To avoid these planning challenges, it's possible to enable the feature Azure CNI networking for dynamic allocation of IPs and enhanced subnet support.

> **Note**
>
> Due to Kubernetes limitations, the Resource Group name, the Virtual Network name and the subnet name must be 63 characters or less.

Unlike kubenet, traffic to endpoints in the same virtual network isn't translated (NAT) to the node's primary IP. The source address for traffic inside the virtual network is the pod IP. Traffic that's external to the virtual network still NATs to the node's primary IP.

Nodes use the Azure CNI Kubernetes plugin.



For more information, see Configure Azure CNI for an AKS cluster.

Azure CNI Overlay networking

Azure CNI Overlay represents an evolution of Azure CNI, addressing scalability and planning challenges arising from the assignment of virtual network IPs to pods. Azure CNI Overlay assigns private CIDR IPs to pods. The private IPs are separate from the virtual network and can be reused across multiple clusters. Azure CNI Overlay can scale beyond the 400 node limit enforced in Kubenet clusters. Azure CNI Overlay is the recommended option for most clusters.

Azure CNI Powered by Cilium

Azure CNI Powered by Cilium uses Cilium to provide high-performance networking, observability, and network policy enforcement. It integrates natively with Azure CNI Overlay for scalable IP address management (IPAM).

Additionally, Cilium enforces network policies by default, without requiring a separate network policy engine. Azure CNI Powered by Cilium can scale beyond Azure Network Policy Manager's limits of 250 nodes / 20-K pod by using ePBF programs and a more efficient API object structure.

Azure CNI Powered by Cilium is the recommended option for clusters that require network policy enforcement.

Bring your own CNI

It's possible to install in AKS a non-Microsoft CNI using the Bring your own CNI feature.

Compare network models

Both kubenet and Azure CNI provide network connectivity for your AKS clusters. However, there are advantages and disadvantages to each. At a high level, the following considerations apply:

- **kubenet**
  - Conserves IP address space.
  - Uses Kubernetes internal or external load balancers to reach pods from outside of the cluster.
  - You manually manage and maintain user-defined routes (UDRs).
  - Maximum of 400 nodes per cluster.
- **Azure CNI**
  - Pods get full virtual network connectivity and can be directly reached via their private IP address from connected networks.
  - Requires more IP address space.

Expand table

Network model	When to use
Kubenet	• IP address space conversation is a priority. • Simple configuration. • Fewer than 400 nodes per cluster. • Kubernetes internal or external load balancers are sufficient for reaching pods

Network model	When to use
	from outside the cluster. • Manually managing and maintaining user defined routes is acceptable.
Azure CNI	• Full virtual network connectivity is required for pods. • Advanced AKS features (such as virtual nodes) are needed. • Sufficient IP address space is available. • Pod to pod and pod to virtual machine connectivity is needed. • External resources need to reach pods directly. • AKS network policies are required.
Azure CNI Overlay	• IP address shortage is a concern. • Scaling up to 1,000 nodes and 250 pods per node is sufficient. • Extra hop for pod connectivity is acceptable. • Simpler network configuration. • AKS egress requirements can be met.

The following behavior differences exist between kubenet and Azure CNI:

Expand table

Capability	Kubenet	Azure CNI	Azure CNI Overlay	Azure CNI Powered by Cilium
Deploy cluster in existing or new virtual network	Supported - UDRs manually applied	Supported	Supported	Supported
Pod-pod connectivity	Supported	Supported	Supported	Supported
Pod-VM connectivity; VM in the same virtual network	Works when initiated by pod	Works both ways	Works when initiated by pod	Works when initiated by pod
Pod-VM connectivity; VM in peered virtual network	Works when initiated by pod	Works both ways	Works when initiated by pod	Works when initiated by pod
On-premises access using VPN or Express Route	Works when initiated by pod	Works both ways	Works when initiated by pod	Works when initiated by pod
Access to resources secured by service endpoints	Supported	Supported	Supported	
Expose Kubernetes services using a load balancer	Supported	Supported	Supported	Same limitations when using

Capability	Kubenet	Azure CNI	Azure CNI Overlay	Azure CNI Powered by Cilium
service, App Gateway, or ingress controller				Overlay mode
Support for Windows node pools	Not Supported	Supported	Supported	Available only for Linux and not for Windows.
Default Azure DNS and Private Zones	Supported	Supported	Supported	

For more information on Azure CNI and kubenet and to help determine which option is best for you, see Configure Azure CNI networking in AKS and Use kubenet networking in AKS.

Support scope between network models

Whatever network model you use, both kubenet and Azure CNI can be deployed in one of the following ways:

- The Azure platform can automatically create and configure the virtual network resources when you create an AKS cluster.
- You can manually create and configure the virtual network resources and attach to those resources when you create your AKS cluster.

Although capabilities like service endpoints or UDRs are supported with both kubenet and Azure CNI, the support policies for AKS define what changes you can make. For example:

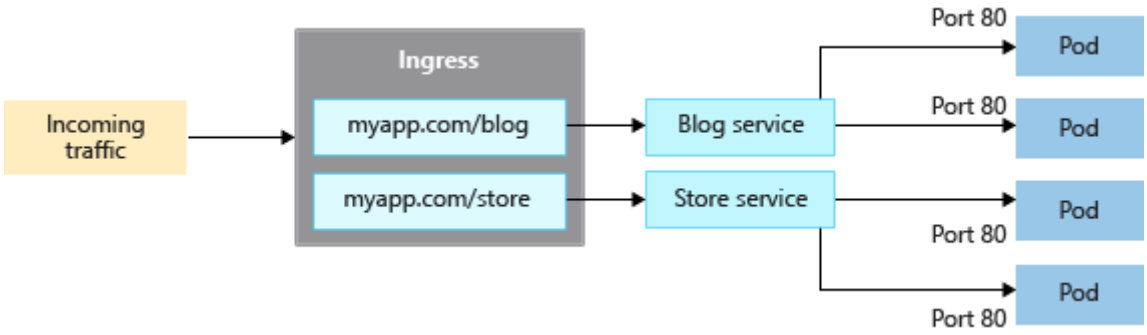
- If you manually create the virtual network resources for an AKS cluster, you're supported when configuring your own UDRs or service endpoints.
- If the Azure platform automatically creates the virtual network resources for your AKS cluster, you can't manually change those AKS-managed resources to configure your own UDRs or service endpoints.

Ingress controllers

When you create a LoadBalancer-type Service, you also create an underlying Azure load balancer resource. The load balancer is configured to distribute traffic to the pods in your Service on a given port.

The *LoadBalancer* only works at layer 4. At layer 4, the Service is unaware of the actual applications, and can't make any more routing considerations.

Ingress controllers work at layer 7 and can use more intelligent rules to distribute application traffic. Ingress controllers typically route HTTP traffic to different applications based on the inbound URL.



Create an Ingress resource

The application routing addon is the recommended way to configure an Ingress controller in AKS. The application routing addon is a fully managed, ingress controller for Azure Kubernetes Service (AKS) that provides the following features:

- Easy configuration of managed NGINX Ingress controllers based on Kubernetes NGINX Ingress controller.
- Integration with Azure DNS for public and private zone management.
- SSL termination with certificates stored in Azure Key Vault.

For more information about the application routing addon, see Managed NGINX ingress with the application routing add-on.

Client source IP preservation

Configure your ingress controller to preserve the client source IP on requests to containers in your AKS cluster. When your ingress controller routes a client's request to a container in your AKS cluster, the original source IP of that request is unavailable to the target container. When you enable *client source IP preservation*, the source IP for the client is available in the request header under *X-Forwarded-For*.

If you're using client source IP preservation on your ingress controller, you can't use TLS pass-through. Client source IP preservation and TLS pass-through can be used with other services, such as the *LoadBalancer* type.

To learn more about client source IP preservation, see How client source IP preservation works for LoadBalancer Services in AKS .

Control outbound (egress) traffic

AKS clusters are deployed on a virtual network and have outbound dependencies on services outside of that virtual network. These outbound dependencies are almost entirely defined with fully qualified domain names (FQDNs). By default, AKS clusters have unrestricted outbound (egress) Internet access, which allows the nodes and services you run to access external resources as needed. If desired, you can restrict outbound traffic.

For more information, see Control egress traffic for cluster nodes in AKS.

Network security groups

A network security group filters traffic for VMs like the AKS nodes. As you create Services, such as a *LoadBalancer*, the Azure platform automatically configures any necessary network security group rules.

You don't need to manually configure network security group rules to filter traffic for pods in an AKS cluster. You can define any required ports and forwarding as part of your Kubernetes Service manifests and let the Azure platform create or update the appropriate rules.

You can also use network policies to automatically apply traffic filter rules to pods.

For more information, see How network security groups filter network traffic.

Network policies

By default, all pods in an AKS cluster can send and receive traffic without limitations. For improved security, define rules that control the flow of traffic, like:

- Back-end applications are only exposed to required frontend services.
- Database components are only accessible to the application tiers that connect to them.

Network policy is a Kubernetes feature available in AKS that lets you control the traffic flow between pods. You can allow or deny traffic to the pod based on settings such as assigned labels, namespace, or traffic port. While network security groups are better for AKS nodes, network policies are a more suited, 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.

For more information, see Secure traffic between pods using network policies in Azure Kubernetes Service (AKS).

Next steps

To get started with AKS networking, create and configure an AKS cluster with your own IP address ranges using kubenet or Azure CNI.

For associated best practices, see Best practices for network connectivity and security in AKS.

For more information on core Kubernetes and AKS concepts, see the following articles:

- Kubernetes / AKS clusters and workloads
- Kubernetes / AKS access and identity
- Kubernetes / AKS security
- Kubernetes / AKS storage
- Kubernetes / AKS scale