Configure Azure CNI networking in Azure Kubernetes Service (AKS)

06/03/2019 • 13 minutes to read • +9

In this article

By default, AKS clusters use kubenet, and a virtual network and subnet are created for you. With *kubenet*, nodes get an IP address from a virtual network subnet. Network address translation (NAT) is then configured on the nodes, and pods receive an IP address "hidden" behind the node IP. This approach reduces the number of IP addresses that you need to reserve in your network space for pods to use.

With Azure Container Networking Interface (CNI), every pod gets an IP address from the subnet and can be accessed directly. These IP addresses must be unique across your network space, and must be planned in advance. Each node has a configuration parameter for the maximum number of pods that it supports. The equivalent number of IP addresses per node are then reserved up front for that node. This approach requires more planning, and often leads to IP address exhaustion or the need to rebuild clusters in a larger subnet as your application demands grow.

This article shows you how to use *Azure CNI* networking to create and use a virtual network subnet for an AKS cluster. For more information on network options and considerations, see Network concepts for Kubernetes and AKS.

Prerequisites

- The virtual network for the AKS cluster must allow outbound internet connectivity.
- AKS clusters may not use `169.254.0.0/16`, `172.30.0.0/16`, `172.31.0.0/16`, or `192.0.2.0/24` for the Kubernetes service address range.
- The service principal used by the AKS cluster must have at least Network Contributor permissions on the subnet within your virtual network. If you wish to define a custom role instead of using the built-in Network Contributor role, the following permissions are required:
  - `Microsoft.Network/virtualNetworks/subnets/join/action`
  - `Microsoft.Network/virtualNetworks/subnets/read`
- Instead of a service principal, you can use the system assigned managed identity for permissions. For more information, see Use managed identities.
- The subnet assigned to the AKS node pool cannot be a delegated subnet.

Plan IP addressing for your cluster

Clusters configured with Azure CNI networking require additional planning. 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.

IP addresses for the pods and the cluster's nodes are assigned from the specified subnet within the virtual network. Each node is configured with a primary IP address. By default, 30 additional IP addresses are pre-configured by Azure CNI that are assigned to pods scheduled on the node. When you scale out your cluster, each node is similarly configured with IP addresses from the subnet. You can also view the maximum pods per node.

> **ⓘ Important**
>
> The number of IP addresses required should include considerations for upgrade and scaling operations. If you set the IP address range to only support a fixed number of nodes, you cannot upgrade or scale your cluster.

- When you **upgrade** your AKS cluster, a new node is deployed into the cluster. Services and workloads begin to run on the new node, and an older node is removed from the cluster. This rolling upgrade process requires a minimum of one additional block of IP addresses to be available. Your node count is then `n + 1`.
  - This consideration is particularly important when you use Windows Server node pools. Windows Server nodes in AKS do not automatically apply Windows Updates, instead you perform an upgrade on the node pool. This upgrade deploys new nodes with the latest Window Server 2019 base node image and security patches. For more information on upgrading a Windows Server node pool, see **Upgrade a node pool in AKS**.

- When you **scale** an AKS cluster, a new node is deployed into the cluster. Services and workloads begin to run on the new node. Your IP address range needs to take into considerations how you may want to scale up the number of nodes and pods your cluster can support. One additional node for upgrade operations should also be included. Your node count is then `n + number-of-additional-scaled-nodes-you-anticipate + 1`.

If you expect your nodes to run the maximum number of pods, and regularly destroy and deploy pods, you should also factor in some additional IP addresses per node. These additional IP addresses take into consideration it may take a few seconds for a service to be deleted and the IP address released for a new service to be deployed and acquire the address.

The IP address plan for an AKS cluster consists of a virtual network, at least one subnet for nodes and pods, and a Kubernetes service address range.

Address range / Azure resource	Limits and sizing
Virtual network	The Azure virtual network can be as large as /8, but is limited to 65,536 configured IP addresses.

Address range / Azure resource	Limits and sizing
Subnet	Must be large enough to accommodate the nodes, pods, and all Kubernetes and Azure resources that might be provisioned in your cluster. For example, if you deploy an internal Azure Load Balancer, its front-end IPs are allocated from the cluster subnet, not public IPs. The subnet size should also take into account upgrade operations or future scaling needs. To calculate the *minimum* subnet size including an additional node for upgrade operations: `(number of nodes + 1) + ((number of nodes + 1) * maximum pods per node that you configure)` Example for a 50 node cluster: `(51) + (51 * 30 (default)) = 1,581` (/21 or larger) Example for a 50 node cluster that also includes provision to scale up an additional 10 nodes: `(61) + (61 * 30 (default)) = 1,891` (/21 or larger) If you don't specify a maximum number of pods per node when you create your cluster, the maximum number of pods per node is set to *30*. The minimum number of IP addresses required is based on that value. If you calculate your minimum IP address requirements on a different maximum value, see how to configure the maximum number of pods per node to set this value when you deploy your cluster.
Kubernetes service address range	This range should not be used by any network element on or connected to this virtual network. Service address CIDR must be smaller than /12. You can reuse this range across different AKS clusters.
Kubernetes DNS service IP address	IP address within the Kubernetes service address range that will be used by cluster service discovery (kube-dns). Don't use the first IP address in your address range, such as .1. The first address in your subnet range is used for the *kubernetes.default.svc.cluster.local* address.

Address range / Azure resource	Limits and sizing
Docker bridge address	The Docker bridge network address represents the default *docker0* bridge network address present in all Docker installations. While *docker0* bridge is not used by AKS clusters or the pods themselves, you must set this address to continue to support scenarios such as *docker build* within the AKS cluster. It is required to select a CIDR for the Docker bridge network address because otherwise Docker will pick a subnet automatically which could conflict with other CIDRs. You must pick an address space that does not collide with the rest of the CIDRs on your networks, including the cluster's service CIDR and pod CIDR. Default of 172.17.0.1/16. You can reuse this range across different AKS clusters.

Maximum pods per node

The maximum number of pods per node in an AKS cluster is 250. The *default* maximum number of pods per node varies between *kubenet* and *Azure CNI* networking, and the method of cluster deployment.

Deployment method	Kubenet default	Azure CNI default	Configurable at deployment
Azure CLI	110	30	Yes (up to 250)
Resource Manager template	110	30	Yes (up to 250)
Portal	110	30	No

Configure maximum - new clusters

You're able to configure the maximum number of pods per node at cluster deployment time or as you add new node pools. If you deploy with the Azure CLI or with a Resource Manager template, you can set the maximum pods per node value as high as 250.

If you don't specify maxPods when creating new node pools, you receive a default value of 30 for Azure CNI.

A minimum value for maximum pods per node is enforced to guarantee space for system pods critical to cluster health. The minimum value that can be set for maximum pods per node is 10 if and only if the configuration of each node pool has space for a minimum of 30 pods. For example, setting the maximum pods per node to the minimum of 10 requires each individual node pool to have a minimum of 3 nodes. This requirement applies for each new node pool created as well, so if 10 is defined as maximum pods per node each subsequent node pool added must have at least 3 nodes.

Networking	Minimum	Maximum
Azure CNI	10	250
Kubenet	10	110

> **Note**
>
> The minimum value in the table above is strictly enforced by the AKS service. You can not set a maxPods value lower than the minimum shown as doing so can prevent the cluster from starting.

- **Azure CLI**: Specify the `--max-pods` argument when you deploy a cluster with the az aks create command. The maximum value is 250.
- **Resource Manager template**: Specify the `maxPods` property in the ManagedClusterAgentPoolProfile object when you deploy a cluster with a Resource Manager template. The maximum value is 250.
- **Azure portal**: You can't change the maximum number of pods per node when you deploy a cluster with the Azure portal. Azure CNI networking clusters are limited to 30 pods per node when you deploy using the Azure portal.

Configure maximum - existing clusters

The maxPod per node setting can be defined when you create a new node pool. If you need to increase the maxPod per node setting on an existing cluster, add a new node pool with the new desired maxPod count. After migrating your pods to the new pool, delete the older pool. To delete any older pool in a cluster, ensure you are setting node pool modes as defined in the [system node pool documentsystem-node-pools.

Deployment parameters

When you create an AKS cluster, the following parameters are configurable for Azure CNI networking:

Virtual network: The virtual network into which you want to deploy the Kubernetes cluster. If you want to create a new virtual network for your cluster, select *Create new* and follow the steps in the *Create virtual network* section. For information about the limits and quotas for an Azure virtual network, see Azure subscription and service limits, quotas, and constraints.

Subnet: The subnet within the virtual network where you want to deploy the cluster. If you want to create a new subnet in the virtual network for your cluster, select *Create new* and follow the steps in the *Create subnet* section. For hybrid connectivity, the address range shouldn't overlap with any other virtual networks in your environment.

Kubernetes service address range: This is the set of virtual IPs that Kubernetes assigns to internal services in your cluster. You can use any private address range that satisfies the following requirements:

- Must not be within the virtual network IP address range of your cluster
- Must not overlap with any other virtual networks with which the cluster virtual network peers
- Must not overlap with any on-premises IPs
- Must not be within the ranges `169.254.0.0/16`, `172.30.0.0/16`, `172.31.0.0/16`, or `192.0.2.0/24`

Although it's technically possible to specify a service address range within the same virtual network as your cluster, doing so is not recommended. Unpredictable behavior can result if overlapping IP ranges are used. For more information, see the FAQ section of this article. For more information on Kubernetes services, see Services in the Kubernetes documentation.

Kubernetes DNS service IP address: The IP address for the cluster's DNS service. This address must be within the *Kubernetes service address range*. Don't use the first IP address in your address range, such as .1. The first address in your

subnet range is used for the *kubernetes.default.svc.cluster.local* address.

Docker Bridge address: The Docker bridge network address represents the default *docker0* bridge network address present in all Docker installations. While *docker0* bridge is not used by AKS clusters or the pods themselves, you must set this address to continue to support scenarios such as *docker build* within the AKS cluster. It is required to select a CIDR for the Docker bridge network address because otherwise Docker will pick a subnet automatically which could conflict with other CIDRs. You must pick an address space that does not collide with the rest of the CIDRs on your networks, including the cluster's service CIDR and pod CIDR.

Configure networking - CLI

When you create an AKS cluster with the Azure CLI, you can also configure Azure CNI networking. Use the following commands to create a new AKS cluster with Azure CNI networking enabled.

First, get the subnet resource ID for the existing subnet into which the AKS cluster will be joined:

Azure CLI Copy Try It

```
$ az network vnet subnet list \
    --resource-group myVnet \
    --vnet-name myVnet \
    --query "[0].id" --output tsv

/subscriptions/<guid>/resourceGroups/myVnet/providers/Microsoft.Network/virtualNetworks/myVnet/subnets/default
```

Use the az aks create command with the `--network-plugin azure` argument to create a cluster with advanced networking. Update the `--vnet-subnet-id` value with the subnet ID collected in the previous step:

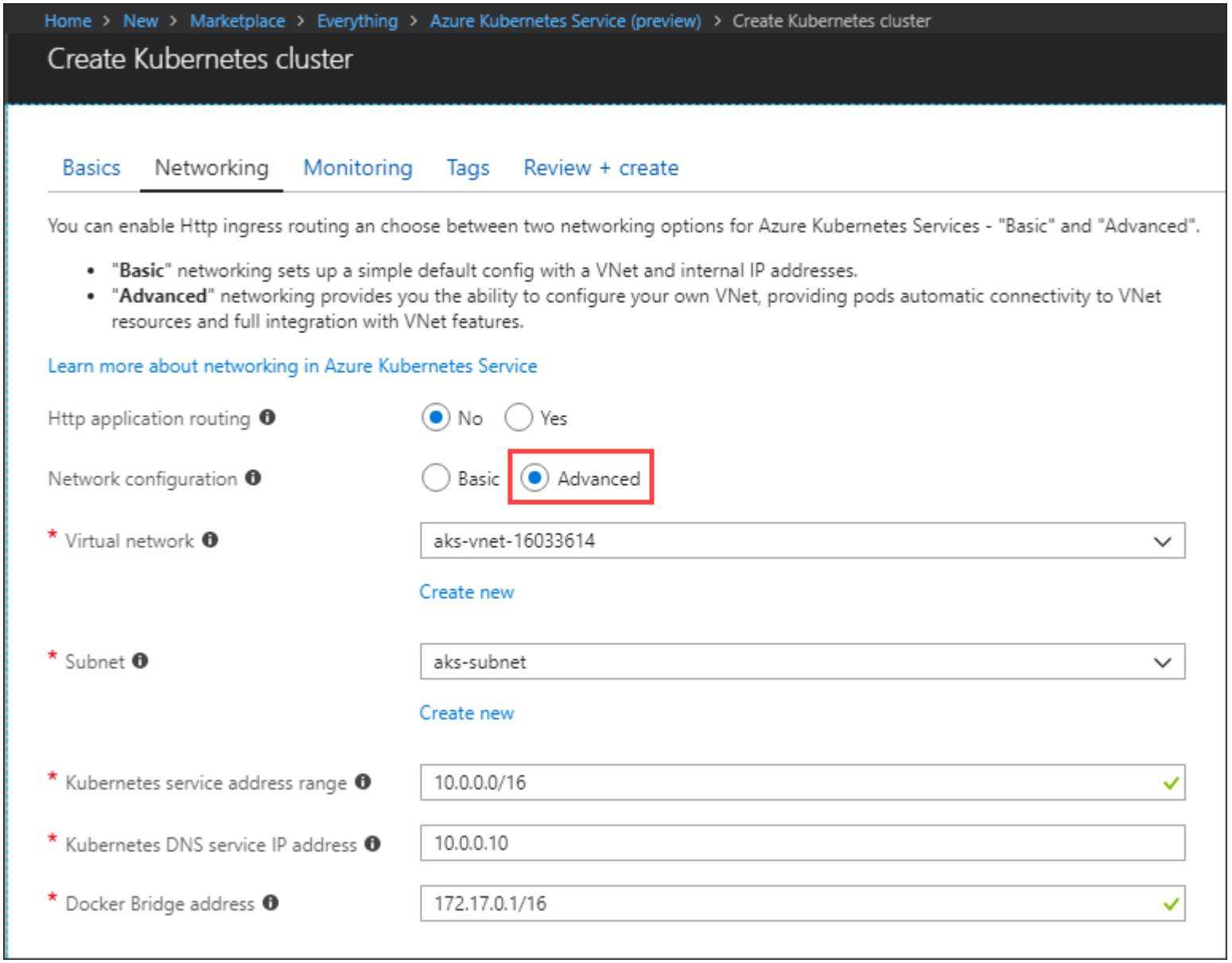
Azure CLI Copy Try It

```
az aks create \
    --resource-group myResourceGroup \
```

```
    --name myAKSCluster \
    --network-plugin azure \
    --vnet-subnet-id <subnet-id> \
    --docker-bridge-address 172.17.0.1/16 \
    --dns-service-ip 10.2.0.10 \
    --service-cidr 10.2.0.0/24 \
    --generate-ssh-keys
```

Configure networking - portal

The following screenshot from the Azure portal shows an example of configuring these settings during AKS cluster creation:



Frequently asked questions

The following questions and answers apply to the **Azure CNI** networking configuration.

- *Can I deploy VMs in my cluster subnet?*

  Yes.

- *Can I configure per-pod network policies?*

  Yes, Kubernetes network policy is available in AKS. To get started, see Secure traffic between pods by using network policies in AKS.

- *Is the maximum number of pods deployable to a node configurable?*

  Yes, when you deploy a cluster with the Azure CLI or a Resource Manager template. See Maximum pods per node.

  You can't change the maximum number of pods per node on an existing cluster.

- *How do I configure additional properties for the subnet that I created during AKS cluster creation? For example, service endpoints.*

  The complete list of properties for the virtual network and subnets that you create during AKS cluster creation can be configured in the standard virtual network configuration page in the Azure portal.

- *Can I use a different subnet within my cluster virtual network for the* **Kubernetes service address range**?

  It's not recommended, but this configuration is possible. The service address range is a set of virtual IPs (VIPs) that Kubernetes assigns to internal services in your cluster. Azure Networking has no visibility into the service IP range of the Kubernetes cluster. Because of the lack of visibility into the cluster's service address range, it's possible to later create a new subnet in the cluster virtual network that overlaps with the service address range. If such an overlap occurs, Kubernetes could assign a service an IP that's already in use by another resource in the subnet, causing unpredictable behavior or failures. By ensuring you use an address range outside the cluster's virtual network, you can avoid this overlap risk.

Next steps

Learn more about networking in AKS in the following articles:

- Use a static IP address with the Azure Kubernetes Service (AKS) load balancer
- Use an internal load balancer with Azure Container Service (AKS)
- Create a basic ingress controller with external network connectivity
- Enable the HTTP application routing add-on
- Create an ingress controller that uses an internal, private network and IP address
- Create an ingress controller with a dynamic public IP and configure Let's Encrypt to automatically generate TLS certificates
- Create an ingress controller with a static public IP and configure Let's Encrypt to automatically generate TLS certificates

AKS Engine

Azure Kubernetes Service Engine (AKS Engine) is an open-source project that generates Azure Resource Manager templates you can use for deploying Kubernetes clusters on Azure.

Kubernetes clusters created with AKS Engine support both the kubenet and Azure CNI plugins. As such, both networking scenarios are supported by AKS Engine.