Networking on Windows

Kubernetes supports running nodes on either Linux or Windows. You can mix both kinds of node within a single cluster. This page provides an overview to networking specific to the Windows operating system.

Container networking on Windows

Networking for Windows containers is exposed through CNI plugins. Windows containers function similarly to virtual machines in regards to networking. Each container has a virtual network adapter (vNIC) which is connected to a Hyper-V virtual switch (vSwitch). The Host Networking Service (HNS) and the Host Compute Service (HCS) work together to create containers and attach container vNICs to networks. HCS is responsible for the management of containers whereas HNS is responsible for the management of networking resources such as:

- Virtual networks (including creation of vSwitches)
- Endpoints / vNICs
- Namespaces
- Policies including packet encapsulations, load-balancing rules, ACLs, and NAT rules.

The Windows HNS and vSwitch implement namespacing and can create virtual NICs as needed for a pod or container. However, many configurations such as DNS, routes, and metrics are stored in the Windows registry database rather than as files inside `/etc`, which is how Linux stores those configurations. The Windows registry for the container is separate from that of the host, so concepts like mapping `/etc/resolv.conf` from the host into a container don't have the same effect they would on Linux. These must be configured using Windows APIs run in the context of that container. Therefore CNI implementations need to call the HNS instead of relying on file mappings to pass network details into the pod or container.

Network modes

Windows supports five different networking drivers/modes: L2bridge, L2tunnel, Overlay (Beta), Transparent, and NAT. In a heterogeneous cluster with Windows and Linux worker nodes, you need to select a networking solution that is compatible on both Windows and Linux. The following table lists the out-of-tree plugins are supported on Windows, with recommendations on when to use each CNI:

Network Driver	Description	Container Packet Modifications	Network Plugins	Network Plugin Characteristics
L2bridge	Containers are attached to an external vSwitch. Containers are attached to the underlay network, although the physical network doesn't need to learn the container MACs because they are rewritten on ingress/egress.	MAC is rewritten to host MAC, IP may be rewritten to host IP using HNS OutboundNAT policy.	win-bridge, Azure-CNI, Flannel host-gateway uses win-bridge	win-bridge uses L2bridge network mode, connects containers to the underlay of hosts, offering best performance. Requires user-defined routes (UDR) for inter-node connectivity.

Network Driver	Description	Container Packet Modifications	Network Plugins	Network Plugin Characteristics
L2Tunnel	This is a special case of l2bridge, but only used on Azure. All packets are sent to the virtualization host where SDN policy is applied.	MAC rewritten, IP visible on the underlay network	Azure-CNI	Azure-CNI allows integration of containers with Azure vNET, and allows them to leverage the set of capabilities that Azure Virtual Network provides. For example, securely connect to Azure services or use Azure NSGs. See azure-cni for some examples
Overlay	Containers are given a vNIC connected to an external vSwitch. Each overlay network gets its own IP subnet, defined by a custom IP prefix.The overlay network driver uses VXLAN encapsulation.	Encapsulated with an outer header.	win-overlay, Flannel VXLAN (uses win-overlay)	win-overlay should be used when virtual container networks are desired to be isolated from underlay of hosts (e.g. for security reasons). Allows for IPs to be re-used for different overlay networks (which have different VNID tags) if you are restricted on IPs in your datacenter. This option requires KB4489899 on Windows Server 2019.
Transparent (special use case for ovn-kubernetes)	Requires an external vSwitch. Containers are attached to an external vSwitch which enables intra-pod communication via logical networks (logical switches and routers).	Packet is encapsulated either via GENEVE or STT tunneling to reach pods which are not on the same host. Packets are forwarded or dropped via the tunnel metadata information supplied by the ovn network controller. NAT is done for north-south communication.	ovn-kubernetes	Deploy via ansible. Distributed ACLs can be applied via Kubernetes policies. IPAM support. Load-balancing can be achieved without kube-proxy. NATing is done without using iptables/netsh.
NAT (*not used in Kubernetes*)	Containers are given a vNIC connected to an internal vSwitch. DNS/DHCP is provided using an internal component called WinNAT	MAC and IP is rewritten to host MAC/IP.	nat	Included here for completeness

As outlined above, the Flannel CNI plugin is also supported on Windows via the VXLAN network backend (**Beta support** ; delegates to win-overlay) and host-gateway network backend (stable support; delegates to win-bridge).

This plugin supports delegating to one of the reference CNI plugins (win-overlay, win-bridge), to work in conjunction with Flannel daemon on Windows (Flanneld) for automatic node subnet lease assignment and HNS network creation. This plugin reads in its own configuration file (cni.conf), and aggregates it with the environment variables from the FlannelD generated subnet.env file. It then delegates to one of the reference CNI plugins for network plumbing, and sends the correct configuration containing the node-assigned subnet to the IPAM plugin (for example: `host-local`).

For Node, Pod, and Service objects, the following network flows are supported for TCP/UDP traffic:

- Pod → Pod (IP)
- Pod → Pod (Name)
- Pod → Service (Cluster IP)
- Pod → Service (PQDN, but only if there are no ".")
- Pod → Service (FQDN)

- Pod → external (IP)
- Pod → external (DNS)
- Node → Pod
- Pod → Node

IP address management (IPAM)

The following IPAM options are supported on Windows:

- host-local
- azure-vnet-ipam (for azure-cni only)
- Windows Server IPAM (fallback option if no IPAM is set)

Load balancing and Services

A Kubernetes Service is an abstraction that defines a logical set of Pods and a means to access them over a network. In a cluster that includes Windows nodes, you can use the following types of Service:

- `NodePort`
- `ClusterIP`
- `LoadBalancer`
- `ExternalName`

Windows container networking differs in some important ways from Linux networking. The Microsoft documentation for Windows Container Networking provides additional details and background.

On Windows, you can use the following settings to configure Services and load balancing behavior:

Feature	Description	Minimum Supported Windows OS build	How to enable
Session affinity	Ensures that connections from a particular client are passed to the same Pod each time.	Windows Server 2022	Set `service.spec.sessionAffinity` to "ClientIP"
Direct Server Return (DSR)	Load balancing mode where the IP address fixups and the LBNAT occurs at the container vSwitch port directly; service traffic arrives with the source IP set as the originating pod IP.	Windows Server 2019	Set the following flags in kube-proxy: `--feature-gates="WinDSR=true" --enable-dsr=true`
Preserve-Destination	Skips DNAT of service traffic, thereby preserving the virtual IP of the target service in packets reaching the backend Pod. Also disables node-node forwarding.	Windows Server, version 1903	Set `"preserve-destination": "true"` in service annotations and enable DSR in kube-proxy.
IPv4/IPv6 dual-stack networking	Native IPv4-to-IPv4 in parallel with IPv6-to-IPv6 communications to, from, and within a cluster	Windows Server 2019	See IPv4/IPv6 dual-stack
Client IP preservation	Ensures that source IP of incoming ingress traffic gets preserved. Also disables node-node forwarding.	Windows Server 2019	Set `service.spec.externalTrafficPolicy` to "Local" and enable DSR in kube-proxy

Warning:

There are known issue with NodePort Services on overlay networking, if the destination node is running Windows Server 2022. To avoid the issue entirely, you can configure the service with `externalTrafficPolicy: Local`.

There are known issues with Pod to Pod connectivity on l2bridge network on Windows Server 2022 with KB5005619 or higher installed. To workaround the issue and restore Pod to Pod connectivity, you can disable the WinDSR feature in kube-proxy.

These issues require OS fixes. Please follow https://github.com/microsoft/Windows-Containers/issues/204 for updates.

Limitations

The following networking functionality is *not* supported on Windows nodes:

- Host networking mode
- Local NodePort access from the node itself (works for other nodes or external clients)
- More than 64 backend pods (or unique destination addresses) for a single Service
- IPv6 communication between Windows pods connected to overlay networks
- Local Traffic Policy in non-DSR mode
- Outbound communication using the ICMP protocol via the `win-overlay`, `win-bridge`, or using the Azure-CNI plugin. Specifically, the Windows data plane (VFP) doesn't support ICMP packet transpositions, and this means:
  - ICMP packets directed to destinations within the same network (such as pod to pod communication via ping) work as expected;
  - TCP/UDP packets work as expected;
  - ICMP packets directed to pass through a remote network (e.g. pod to external internet communication via ping) cannot be transposed and thus will not be routed back to their source;
  - Since TCP/UDP packets can still be transposed, you can substitute `ping <destination>` with `curl <destination>` when debugging connectivity with the outside world.

Other limitations:

- Windows reference network plugins win-bridge and win-overlay do not implement CNI spec v0.4.0, due to a missing `CHECK` implementation.
- The Flannel VXLAN CNI plugin has the following limitations on Windows:
  - Node-pod connectivity is only possible for local pods with Flannel v0.12.0 (or higher).
  - Flannel is restricted to using VNI 4096 and UDP port 4789. See the official Flannel VXLAN backend docs for more details on these parameters.

Feedback

Was this page helpful?

Yes No

Last modified May 29, 2024 at 4:41 PM PST: Updated and added links in 'Networking on Windows' doc (d501dbe174)