Network Plugins

Kubernetes 1.26 supports Container Network Interface (CNI) plugins for cluster networking. You must use a CNI plugin that is compatible with your cluster and that suits your needs. Different plugins are available (both open- and closed- source) in the wider Kubernetes ecosystem.

A CNI plugin is required to implement the Kubernetes network model.

You must use a CNI plugin that is compatible with the v0.4.0 or later releases of the CNI specification. The Kubernetes project recommends using a plugin that is compatible with the v1.0.0 CNI specification (plugins can be compatible with multiple spec versions).

Installation

A Container Runtime, in the networking context, is a daemon on a node configured to provide CRI Services for kubelet. In particular, the Container Runtime must be configured to load the CNI plugins required to implement the Kubernetes network model.

> **Note:**
> Prior to Kubernetes 1.24, the CNI plugins could also be managed by the kubelet using the `cni-bin-dir` and `network-plugin` command-line parameters. These command-line parameters were removed in Kubernetes 1.24, with management of the CNI no longer in scope for kubelet.
>
> See Troubleshooting CNI plugin-related errors if you are facing issues following the removal of dockershim.

For specific information about how a Container Runtime manages the CNI plugins, see the documentation for that Container Runtime, for example:

- containerd
- CRI-O

For specific information about how to install and manage a CNI plugin, see the documentation for that plugin or networking provider.

Network Plugin Requirements

For plugin developers and users who regularly build or deploy Kubernetes, the plugin may also need specific configuration to support kube-proxy. The iptables proxy depends on iptables, and the plugin may need to ensure that container traffic is made available to iptables. For example, if the plugin connects containers to a Linux bridge, the plugin must set the `net/bridge/bridge-nf-call-iptables` sysctl to `1` to ensure that the iptables proxy functions correctly. If the plugin does not use a Linux bridge, but uses something like Open vSwitch or some other mechanism instead, it should ensure container traffic is appropriately routed for the proxy.

By default, if no kubelet network plugin is specified, the `noop` plugin is used, which sets `net/bridge/bridge-nf-call-iptables=1` to ensure simple configurations (like Docker with a bridge) work correctly with the iptables proxy.

Loopback CNI

In addition to the CNI plugin installed on the nodes for implementing the Kubernetes network model, Kubernetes also requires the container runtimes to provide a loopback interface `lo`, which is used for each sandbox (pod sandboxes, vm sandboxes, ...). Implementing the loopback interface can be accomplished by re-using the CNI loopback plugin. or by developing your own code to achieve this (see this example from CRI-O).

Support hostPort

The CNI networking plugin supports `hostPort`. You can use the official portmap plugin offered by the CNI plugin team or use your own plugin with portMapping functionality.

If you want to enable `hostPort` support, you must specify `portMappings capability` in your `cni-conf-dir`. For example:

```
{
  "name": "k8s-pod-network",
  "cniVersion": "0.4.0",
  "plugins": [
    {
      "type": "calico",
      "log_level": "info",
      "datastore_type": "kubernetes",
      "nodename": "127.0.0.1",
      "ipam": {
        "type": "host-local",
        "subnet": "usePodCidr"
      },
      "policy": {
        "type": "k8s"
      },
      "kubernetes": {
        "kubeconfig": "/etc/cni/net.d/calico-kubeconfig"
      }
    },
    {
      "type": "portmap",
      "capabilities": {"portMappings": true},
      "externalSetMarkChain": "KUBE-MARK-MASQ"
    }
  ]
}
```

Support traffic shaping

Experimental Feature

The CNI networking plugin also supports pod ingress and egress traffic shaping. You can use the official bandwidth plugin offered by the CNI plugin team or use your own plugin with bandwidth control functionality.

If you want to enable traffic shaping support, you must add the `bandwidth` plugin to your CNI configuration file (default `/etc/cni/net.d`) and ensure that the binary is included in your CNI bin dir (default `/opt/cni/bin`).

```
{
  "name": "k8s-pod-network",
  "cniVersion": "0.4.0",
  "plugins": [
    {
      "type": "calico",
      "log_level": "info",
      "datastore_type": "kubernetes",
      "nodename": "127.0.0.1",
      "ipam": {
        "type": "host-local",
        "subnet": "usePodCidr"
      },
      "policy": {
        "type": "k8s"
      },
      "kubernetes": {
        "kubeconfig": "/etc/cni/net.d/calico-kubeconfig"
      }
    },
    {
      "type": "bandwidth",
      "capabilities": {"bandwidth": true}
    }
  ]
}
```

Now you can add the `kubernetes.io/ingress-bandwidth` and `kubernetes.io/egress-bandwidth` annotations to your Pod. For example:

```
apiVersion: v1
kind: Pod
metadata:
  annotations:
    kubernetes.io/ingress-bandwidth: 1M
    kubernetes.io/egress-bandwidth: 1M
...
```

What's next

Feedback

Was this page helpful?

Yes No

Last modified October 08, 2022 at 4:55 PM PST: Tweak line wrappings in the network-plugins page (7242d4158)