containernetworking / cni Public

Container Network Interface - networking for Linux containers

cni.dev

Apache-2.0 license

4.6k stars 991 forks

Star Watch

Code Issues 97 Pull requests 10 Actions Projects Wiki Security Insights

main ...

squeed ... ✓ last week

View code

CNI

CNI - the Container Network Interface

What is CNI?

CNI (*Container Network Interface*), a Cloud Native Computing Foundation project, consists of a specification and libraries for writing plugins to configure network interfaces in Linux containers, along with a number of supported plugins. CNI concerns itself only with network connectivity of containers and removing allocated resources when the container is deleted. Because of this focus, CNI has a wide range of support and the specification is simple to implement.

As well as the specification, this repository contains the Go source code of a library for integrating CNI into applications and an example command-line tool for executing CNI plugins. A separate repository contains reference plugins and a template for making new plugins.

The template code makes it straight-forward to create a CNI plugin for an existing container networking project. CNI also makes a good framework for creating a new container networking project from scratch.

Here are the recordings of two sessions that the CNI maintainers hosted at KubeCon/CloudNativeCon 2019:

- Introduction to CNI
- CNI deep dive

README.md

Why develop CNI?

Application containers on Linux are a rapidly evolving area, and within this area networking is not well addressed as it is highly environment-specific. We believe that many container runtimes and orchestrators will seek to solve the same problem of making the network layer pluggable.

To avoid duplication, we think it is prudent to define a common interface between the network plugins and container execution: hence we put forward this specification, along with libraries for Go and a set of plugins.

Who is using CNI?

Container runtimes

- rkt - container engine
- Kubernetes - a system to simplify container operations
- OpenShift - Kubernetes with additional enterprise features
- Cloud Foundry - a platform for cloud applications
- Apache Mesos - a distributed systems kernel
- Amazon ECS - a highly scalable, high performance container management service
- Singularity - container platform optimized for HPC, EPC, and AI
- OpenSVC - orchestrator for legacy and containerized application stacks

3rd party plugins

- Project Calico - a layer 3 virtual network
- Weave - a multi-host Docker network
- Contiv Networking - policy networking for various use cases
- SR-IOV
- Cilium - BPF & XDP for containers
- Infoblox - enterprise IP address management for containers
- Multus - a Multi plugin
- Romana - Layer 3 CNI plugin supporting network policy for Kubernetes
- CNI-Genie - generic CNI network plugin
- Nuage CNI - Nuage Networks SDN plugin for network policy kubernetes support
- Silk - a CNI plugin designed for Cloud Foundry
- Linen - a CNI plugin designed for overlay networks with Open vSwitch and fit in SDN/OpenFlow network environment
- Vhostuser - a Dataplane network plugin - Supports OVS-DPDK & VPP
- Amazon ECS CNI Plugins - a collection of CNI Plugins to configure containers with Amazon EC2 elastic network interfaces (ENIs)
- Bonding CNI - a Link aggregating plugin to address failover and high availability network
- ovn-kubernetes - an container network plugin built on Open vSwitch (OVS) and Open Virtual Networking (OVN) with support for both Linux and Windows
- Juniper Contrail / TungstenFabric - Provides overlay SDN solution, delivering multicloud networking, hybrid cloud networking, simultaneous overlay-underlay support, network policy enforcement, network isolation, service chaining and flexible load balancing
- Knitter - a CNI plugin supporting multiple networking for Kubernetes
- DANM - a CNI-compliant networking solution for TelCo workloads running on Kubernetes

- VMware NSX – a CNI plugin that enables automated NSX L2/L3 networking and L4/L7 Load Balancing; network isolation at the pod, node, and cluster level; and zero-trust security policy for your Kubernetes cluster.
- cni-route-override - a meta CNI plugin that override route information
- Terway - a collection of CNI Plugins based on alibaba cloud VPC/ECS network product
- Cisco ACI CNI - for on-prem and cloud container networking with consistent policy and security model.
- Kube-OVN - a CNI plugin that bases on OVN/OVS and provides advanced features like subnet, static ip, ACL, QoS, etc.
- Project Antrea - an Open vSwitch k8s CNI
- OVN4NFV-K8S-Plugin - a OVN based CNI controller plugin to provide cloud native based Service function chaining (SFC), Multiple OVN overlay networking
- Azure CNI - a CNI plugin that natively extends Azure Virtual Networks to containers
- Hybridnet - a CNI plugin designed for hybrid clouds which provides both overlay and underlay networking for containers in one or more clusters. Overlay and underlay containers can run on the same node and have cluster-wide bidirectional network connectivity.

The CNI team also maintains some core plugins in a separate repository.

Contributing to CNI

We welcome contributions, including bug reports, and code and documentation improvements. If you intend to contribute to code or documentation, please read CONTRIBUTING.md. Also see the contact section in this README.

How do I use CNI?

Requirements

The CNI spec is language agnostic. To use the Go language libraries in this repository, you'll need a recent version of Go. You can find the Go versions covered by our automated tests in .travis.yaml.

Reference Plugins

The CNI project maintains a set of reference plugins that implement the CNI specification. NOTE: the reference plugins used to live in this repository but have been split out into a separate repository as of May 2017.

Running the plugins

After building and installing the reference plugins, you can use the `priv-net-run.sh` and `docker-run.sh` scripts in the `scripts/` directory to exercise the plugins.

note - priv-net-run.sh depends on `jq`

Start out by creating a netconf file to describe a network:

```
$ mkdir -p /etc/cni/net.d
$ cat >/etc/cni/net.d/10-mynet.conf <<EOF
{
        "cniVersion": "0.2.0",
        "name": "mynet",
        "type": "bridge",
        "bridge": "cni0",
        "isGateway": true,
        "ipMasq": true,
        "ipam": {
                "type": "host-local",
                "subnet": "10.22.0.0/16",
                "routes": [
                        { "dst": "0.0.0.0/0" }
                ]
        }
}
EOF
$ cat >/etc/cni/net.d/99-loopback.conf <<EOF
{
        "cniVersion": "0.2.0",
        "name": "lo",
        "type": "loopback"
}
EOF
```

The directory `/etc/cni/net.d` is the default location in which the scripts will look for net configurations.

Next, build the plugins:

```
$ cd $GOPATH/src/github.com/containernetworking/plugins
$ ./build_linux.sh # or build_windows.sh
```

Finally, execute a command (`ifconfig` in this example) in a private network namespace that has joined the `mynet` network:

```
$ CNI_PATH=$GOPATH/src/github.com/containernetworking/plugins/bin
$ cd $GOPATH/src/github.com/containernetworking/cni/scripts
$ sudo CNI_PATH=$CNI_PATH ./priv-net-run.sh ifconfig
eth0      Link encap:Ethernet  HWaddr f2:c2:6f:54:b8:2b
          inet addr:10.22.0.2  Bcast:0.0.0.0  Mask:255.255.0.0
          inet6 addr: fe80::f0c2:6fff:fe54:b82b/64 Scope:Link
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:1 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:1 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:90 (90.0 B)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

The environment variable `CNI_PATH` tells the scripts and library where to look for plugin executables.

Running a Docker container with network namespace set up by CNI plugins

Use the instructions in the previous section to define a netconf and build the plugins. Next, docker-run.sh script wraps `docker run`, to execute the plugins prior to entering the container:

```
$ CNI_PATH=$GOPATH/src/github.com/containernetworking/plugins/bin
$ cd $GOPATH/src/github.com/containernetworking/cni/scripts
$ sudo CNI_PATH=$CNI_PATH ./docker-run.sh --rm busybox:latest ifconfig
eth0      Link encap:Ethernet  HWaddr fa:60:70:aa:07:d1
          inet addr:10.22.0.2  Bcast:0.0.0.0  Mask:255.255.0.0
          inet6 addr: fe80::f860:70ff:feaa:7d1/64 Scope:Link
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:1 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:1 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:90 (90.0 B)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

What might CNI do in the future?

CNI currently covers a wide range of needs for network configuration due to its simple model and API. However, in the future CNI might want to branch out into other directions:

- Dynamic updates to existing network configuration
- Dynamic policies for network bandwidth and firewall rules

If these topics are of interest, please contact the team via the mailing list or IRC and find some like-minded people in the community to put a proposal together.

Where are the binaries?

The plugins moved to a separate repo: https://github.com/containernetworking/plugins, and the releases there include binaries and checksums.

Prior to release 0.7.0 the `cni` release also included a `cnitool` binary; as this is a developer tool we suggest you build it yourself.

Contact

For any questions about CNI, please reach out via:

- Email: cni-dev
- IRC: #containernetworking channel on freenode.net
- Slack: #cni on the CNCF slack. NOTE: the previous CNI Slack (containernetworking.slack.com) has been sunsetted.

Security

If you have a *security* issue to report, please do so privately to the email addresses listed in the MAINTAINERS file.

Releases 20

CNI v1.1.2 Latest
on Jul 27, 2022

+ 19 releases

Contributors 137

+ 126 contributors

Languages

● **Go** 98.5% ● **Shell** 1.5%