CLOUD NATIVE COMPUTING FOUNDATION

Webinar Series

Introduction to CNI

August 24, 2017

Your Presenters

Ken Owens
VP Digital Native Architecture
Mastercard

Bryan Boreham
Director of Engineering,
WeaveWorks
@bboreham

Agenda

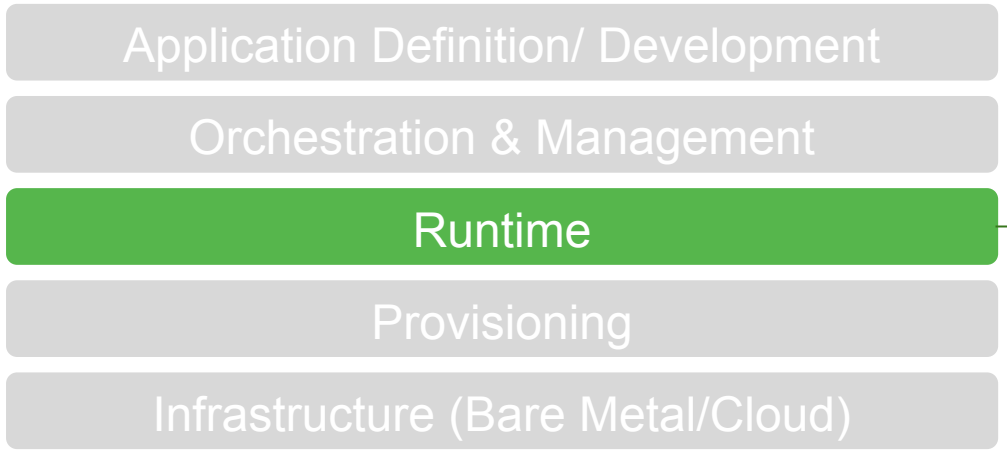
What is CNI?

Who uses CNI - runtimes and plugins

Recent developments

The Future

CNI in the CNCF Reference Architecture

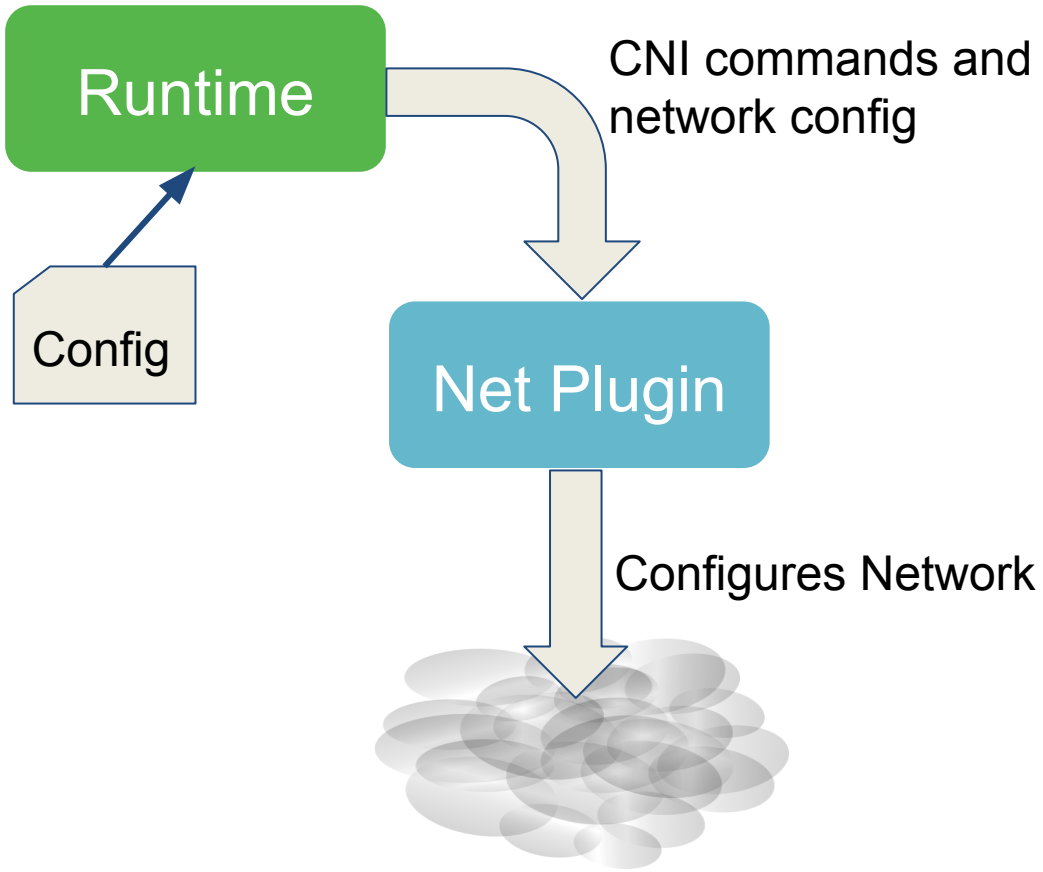


- Resource Management
  - Image Management
  - Container Management
  - Compute Resources
- **Cloud Native – Network**
  - **Network Segmentation and Policy**
  - **SDN & APIs (eg CNI, libnetwork)**
- Cloud Native- Storage
  - Volume Drivers/Plugins
  - Local Storage Management
  - Remote Storage Access

What is CNI?

- The simplest possible interface between container runtime and network implementation
- Originated at CoreOS as part of Rkt
- Now a CNCF project

What is CNI?



What does a CNI call look like?

Set some environment variables and go!

```
CNI_COMMAND=ADD \
CNI_CONTAINERID=$id \
CNI_NETNS=/proc/$pid/ns/net \
CNI_PATH=/opt/cni/bin \
CNI_IFNAME=eth0 \
  my-plugin < my-config
```

Can be either `ADD`, `DEL` or `VERSION`

A JSON document defining the network

What does a CNI config look like?

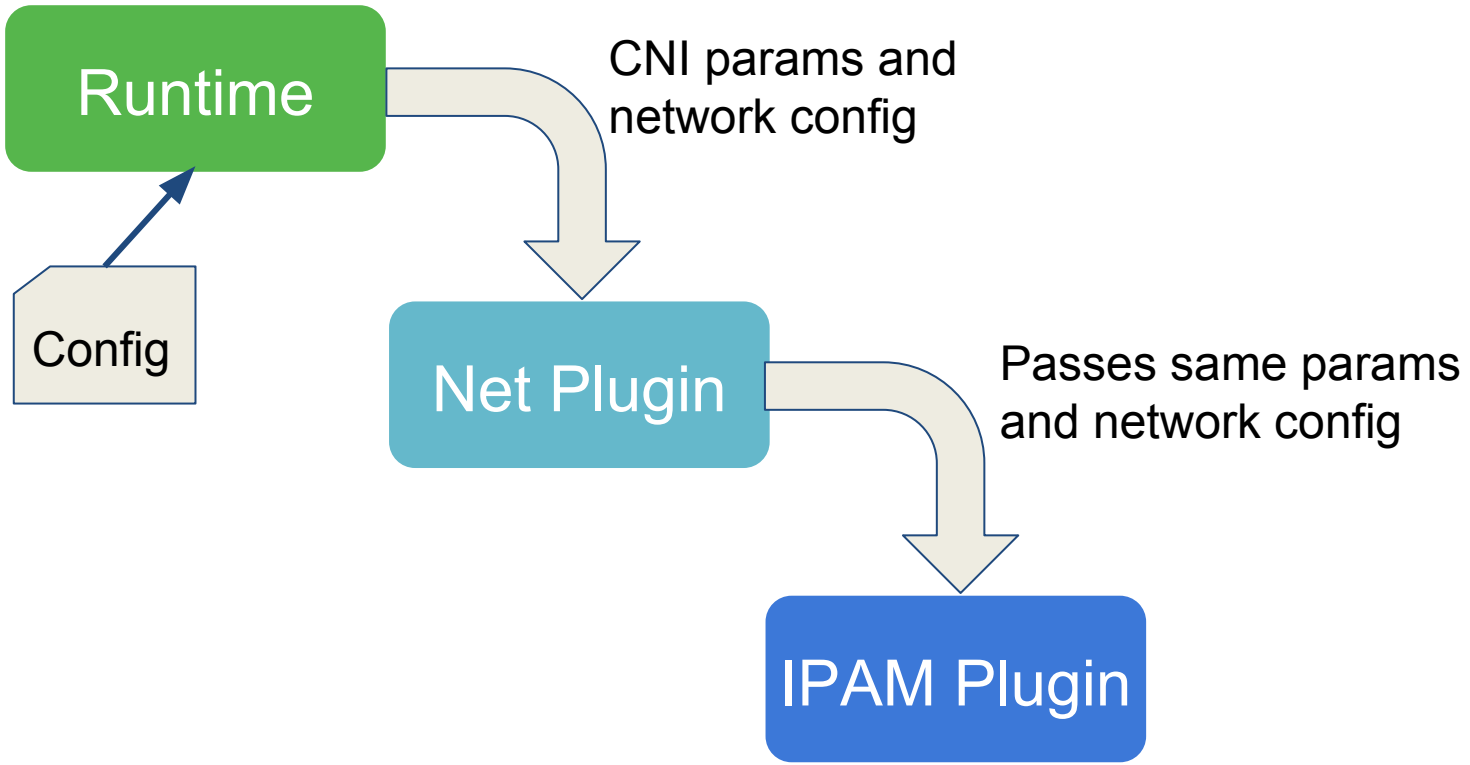
```
{
  "cniVersion": "0.3.0",
  "name": "mynet",
  "type": "my-plugin",
  "some-parameter": "foo",
  "ipam": {
     "type": "host-local",
     "subnet": "10.42.0.0/24",
  }
}
```

Tells the plugin what version the caller is using

The caller should look for a plugin with this name

First plugin will call a second plugin with this name

Network plugin calls IPAM plugin



CNI project repo

https://github.com/containernetworking/cni

- CNI Specification: the API between runtimes and network plugins
- Conventions: extensions to the API that are not required for all plugins
- Library: a Go implementation of the CNI specification that plugins and runtimes can use

- 5 maintainers
- 63 contributors from 10+ companies
- 785 stars

CNI plugins repo

https://github.com/containernetworking/plugins

Main: interface-creating

- `bridge`
- `ipvlan`
- `loopback`
- `macvlan`
- `ptp`
- `vlan`

IPAM: IP address allocation

- `dhcp`
- `host-local`

Meta: other plugins

- `flannel`
- `tuning`
- `portmap`

Sample

- The `sample` plugin provides an example for building your own plugin.

Ecosystem

Container runtimes

- rkt - container engine
- Kubernetes
- Kurma - container runtime
- Cloud Foundry - a platform for cloud applications
- Mesos - a distributed systems kernel

3rd party plugins

- Weave Net
- Project Calico
- Contiv
- SR-IOV
- Cilium
- Infoblox
- Multus
- Romana
- CNI-Genie

Quote

"Our forthcoming ECS Task Networking capabilities are written as a CNI plugin, and we expect CNI to be the basis for all container-based networking on AWS."

- Adrian Cockroft, VP of Cloud Architecture, AWS

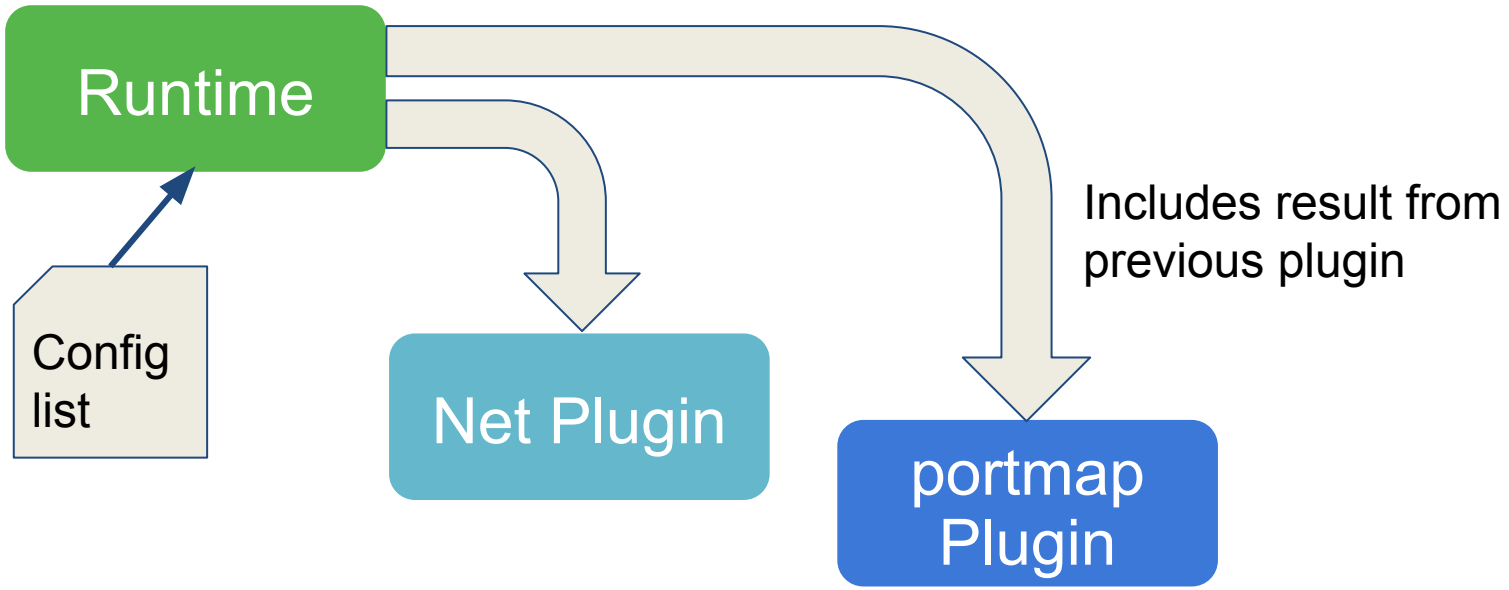
Recent developments

- Chaining (new in 0.5)
  - Configure a list of plugins, not just one
  - Runtime will call each one in turn
  - Capability arguments give extra info to the runtime
- IPv6 (completed in 0.6)
  - Spec now allows multiple addresses to be returned
  - All base plugins support IPv6

Chaining config list example

```
{ "cniVersion": "0.3.0",
  "name": "weave",
  "plugins": [
      {   "type": "weave-net",
          "hairpinMode": true
      },
      {   "type": "portmap",
          "capabilities": { "portMappings": true },
          "snat": true
      }
  ] }
```

Chaining multiple plugins



Looking forward

`GET` command

- For runtime to query the status of an interface

Kubernetes `kubenet` as pure CNI

- Currently part CNI and part embedded inside kubelet

CNI v1.0

- Stable spec, with strategy and tooling for backwards compatibility
- Complete test coverage
- Release from CI

Summary

CNI is a simple interface based on environment variables and JSON

Open Source, lots of runtimes and plugins use it

We welcome new implementers

Please comment on the spec *before* 1.0!

Thank You

- https://github.com/containernetworking/cni
- Email: cni-dev@googlegroups.com
- IRC: #containernetworking channel on freenode.org
- Slack: containernetworking.slack.com