CNI Intro

Bryan Boreham, Weaveworks
Casey Callendrello, Red Hat

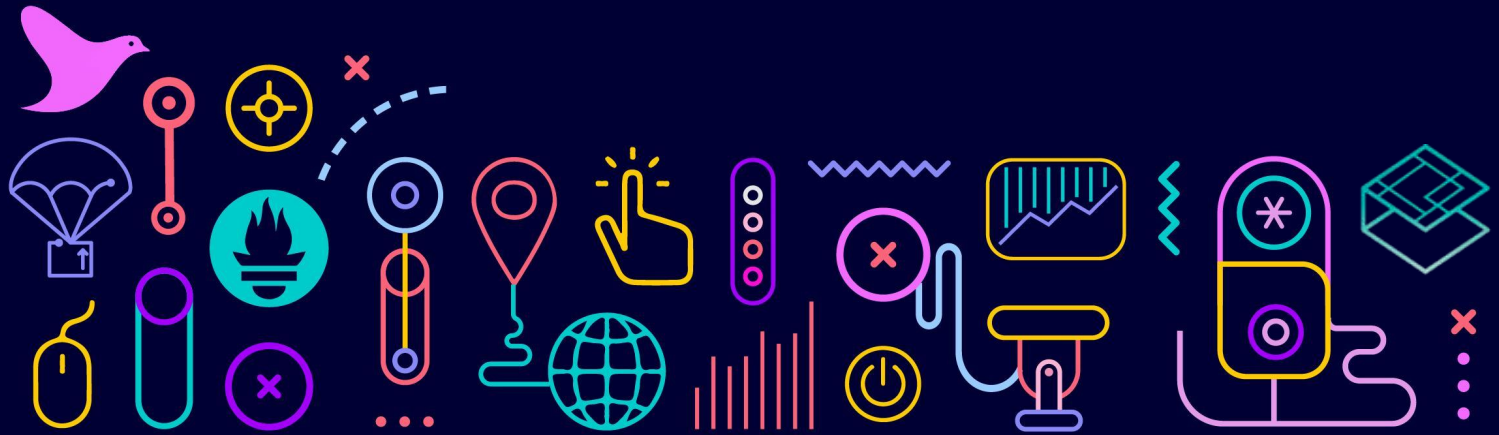
Outline

- Introductions
- What is CNI?
- What is the CNI project?
- How does CNI work?
- FAQ

Introductions...

KubeCon | CloudNativeCon Europe 2020

CNI Deep Dive



Who are we?

Bryan Boreham / @bboreham

Distinguished Engineer, Weaveworks

Maintainer on CNI, Weave Net and Cortex

Likes: helicopters

Dislikes: iptables

Casey Callendrello / @squeed

Principal Engineer & Team Lead, Openshift Networking

Maintainer of CNI, OVN-Kubernetes, and go-iptables (sorry)

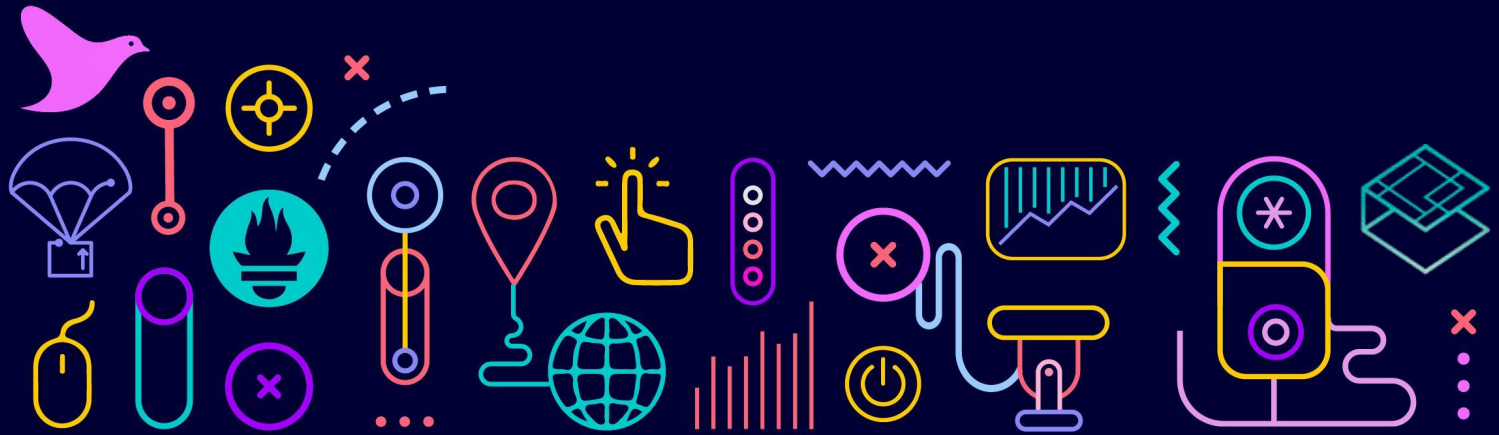
Likes: IPv6, jumbo frames, Informers

Dislikes: NAT, Informers

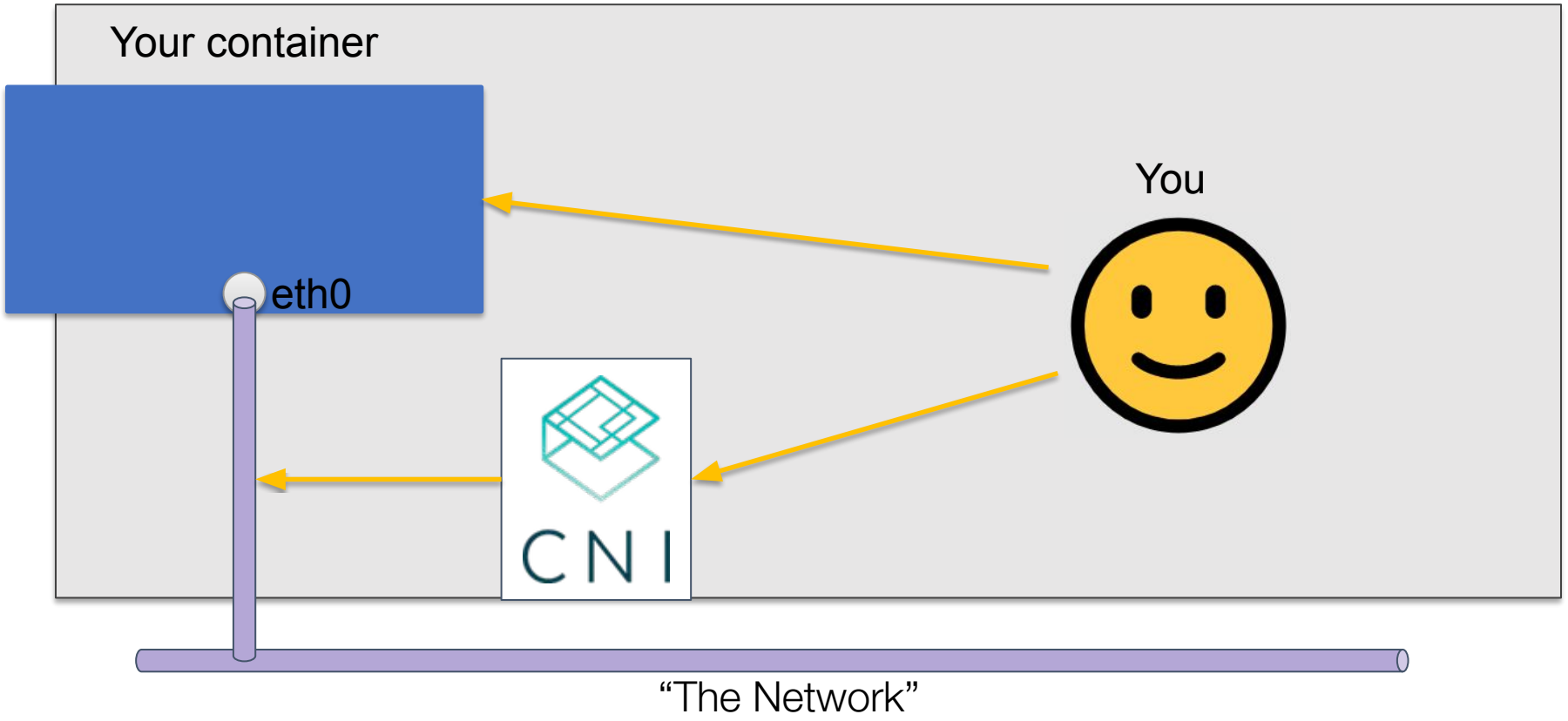
KubeCon | CloudNativeCon Europe 2020

CNI Deep Dive

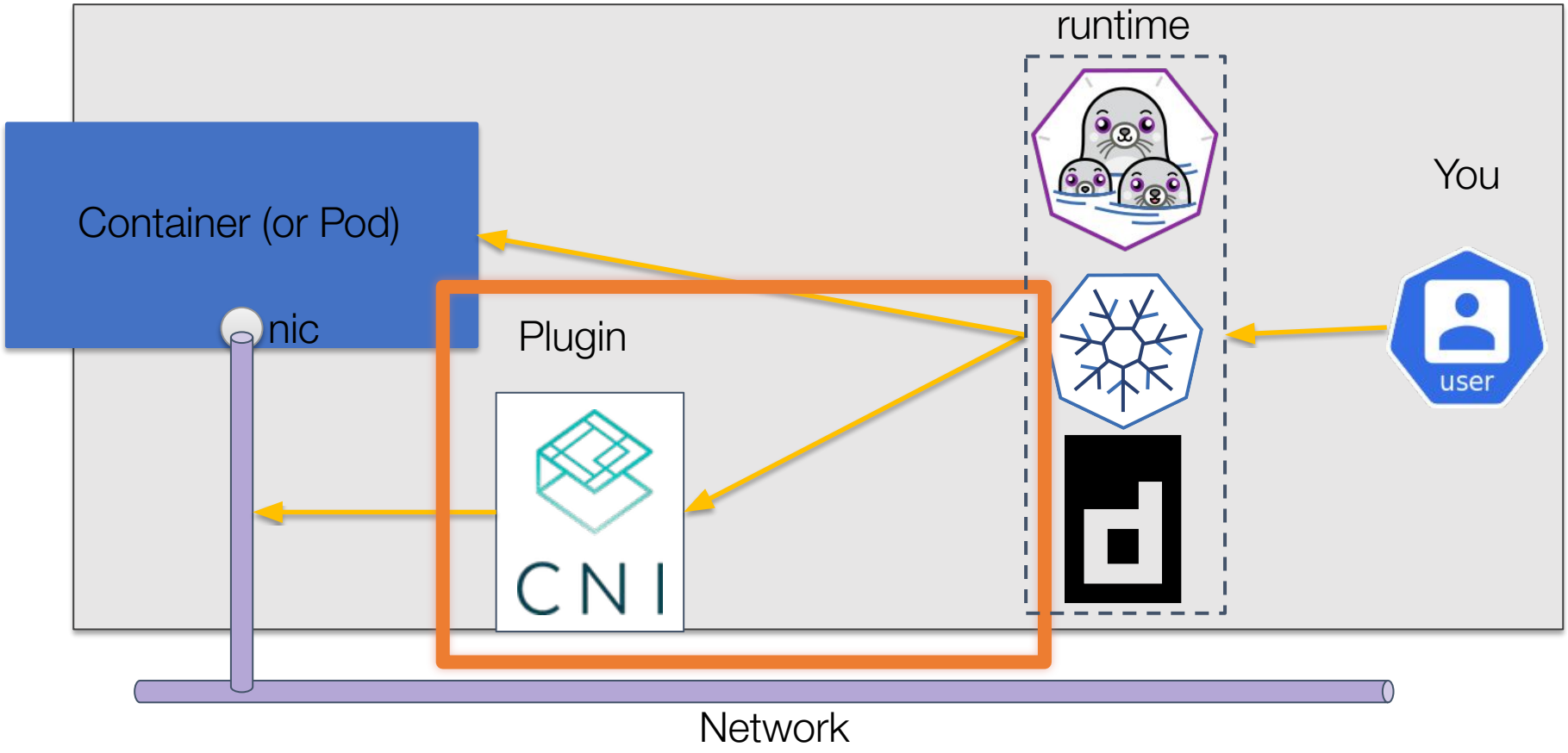
What is CNI?



What is CNI?



What is CNI?



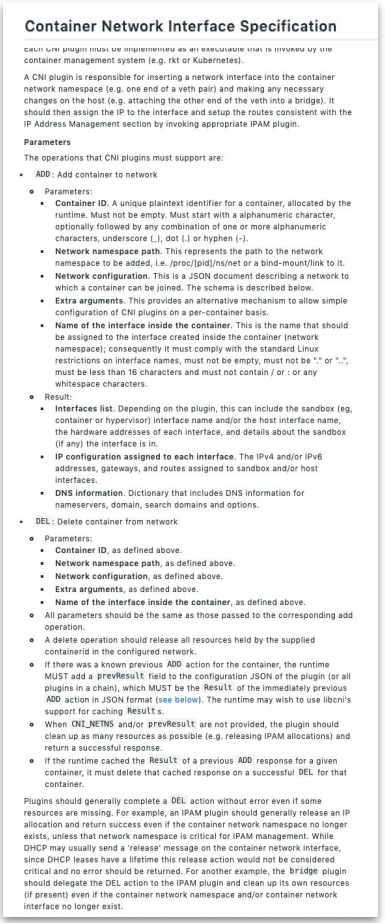
What is the CNI project?

The CNI project has two major parts:

1. The CNI specification documents & reference implementation
   - libcni: Go code to help implement runtime and plugins
   - github.com/containernetworking/cni
2. A set of reference and example plugins:
   - ptp, bridge, macvlan, portmap, bandwidth, tuning, …
   - Linux and Windows
   - github.com/containernetworking/plugins

Specification

- A vendor-neutral specification - not just for Kubernetes
- Also used by Mesos, CloudFoundry, podman, CRI-O, AWS Fargate, ~~rkt~~
- Protocol defines
  a. Plugin discovery
  b. Execution flow
  c. Configuration format
  d. Response struct
- Attempts to keep things simple and backwards compatible



Container Network Interface Specification

Each CNI plugin must be implemented as an executable that is invoked by the container management system (e.g. rkt or Kubernetes).

A CNI plugin is responsible for inserting a network interface into the container network namespace (e.g. one end of a veth pair) and making any necessary changes on the host (e.g. attaching the other end of the veth into a bridge). It should then assign the IP to the interface and setup the routes consistent with the IP Address Management section by invoking appropriate IPAM plugin.

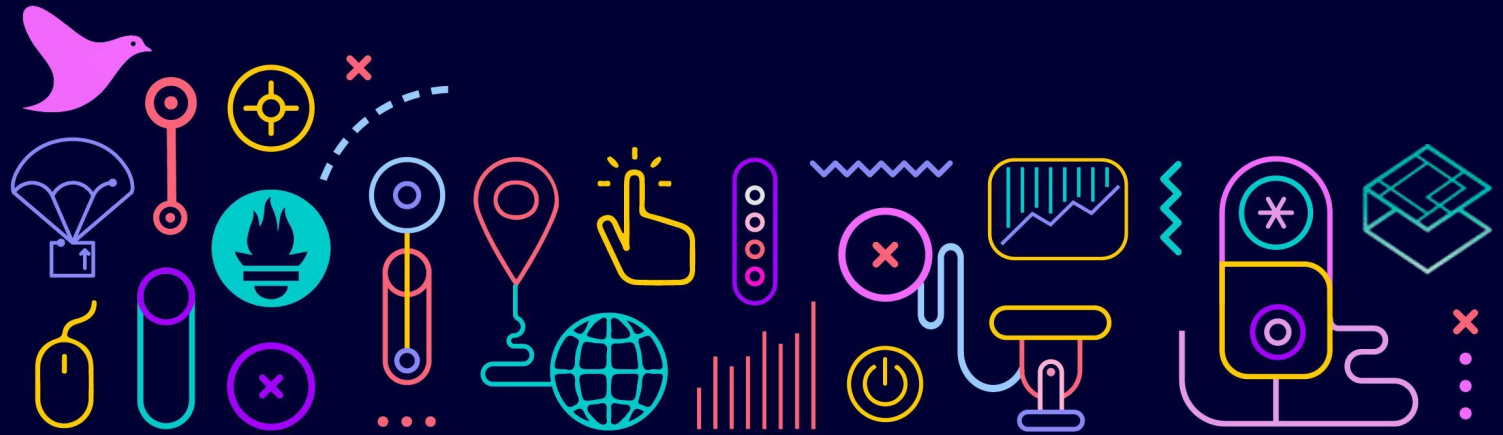
Parameters

The operations that CNI plugins must support are:

- `ADD`: Add container to network
  - Parameters:
    - **Container ID**. A unique plaintext identifier for a container, allocated by the runtime. Must not be empty. Must start with a alphanumeric character, optionally followed by any combination of one or more alphanumeric characters, underscore (_), dot (.) or hyphen (-).
    - **Network namespace path**. This represents the path to the network namespace to be added, i.e. /proc/[pid]/ns/net or a bind-mount/link to it.
    - **Network configuration**. This is a JSON document describing a network to which a container can be joined. The schema is described below.
    - **Extra arguments**. This provides an alternative mechanism to allow simple configuration of CNI plugins on a per-container basis.
    - **Name of the interface inside the container**. This is the name that should be assigned to the interface created inside the container (network namespace); consequently it must comply with the standard Linux restrictions on interface names, must not be empty, must not be "." or "..", must be less than 16 characters and must not contain / or : or any whitespace characters.
  - Result:
    - **Interfaces list**. Depending on the plugin, this can include the sandbox (eg, container or hypervisor) interface name and/or the host interface name, the hardware addresses of each interface, and details about the sandbox (if any) the interface is in.
    - **IP configuration assigned to each interface**. The IPv4 and/or IPv6 addresses, gateways, and routes assigned to sandbox and/or host interfaces.
    - **DNS information**. Dictionary that includes DNS information for nameservers, domain, search domains and options.
- `DEL`: Delete container from network
  - Parameters:
    - **Container ID**, as defined above.
    - **Network namespace path**, as defined above.
    - **Network configuration**, as defined above.
    - **Extra arguments**, as defined above.
    - **Name of the interface inside the container**, as defined above.
  - All parameters should be the same as those passed to the corresponding add operation.
  - A delete operation should release all resources held by the supplied containerid in the configured network.
  - If there was a known previous `ADD` action for the container, the runtime MUST add a `prevResult` field to the configuration JSON of the plugin (or all plugins in a chain), which MUST be the `Result` of the immediately previous `ADD` action in JSON format (see below). The runtime may wish to use libcni's support for caching `Result`s.
  - When `CNI_NETNS` and/or `prevResult` are not provided, the plugin should clean up as many resources as possible (e.g. releasing IPAM allocations) and return a successful response.
  - If the runtime cached the `Result` of a previous `ADD` response for a given container, it must delete that cached response on a successful `DEL` for that container.

Plugins should generally complete a `DEL` action without error even if some resources are missing. For example, an IPAM plugin should generally release an IP allocation and return success even if the container network namespace no longer exists, unless that network namespace is critical for IPAM management. While DHCP may usually send a 'release' message on the container network interface, since DHCP leases have a lifetime this release action would not be considered critical and no error should be returned. For another example, the `bridge` plugin should delegate the DEL action to the IPAM plugin and clean up its own resources (if present) even if the container network namespace and/or container network interface no longer exist.

What *isn't* CNI?



KubeCon | CloudNativeCon Europe 2020

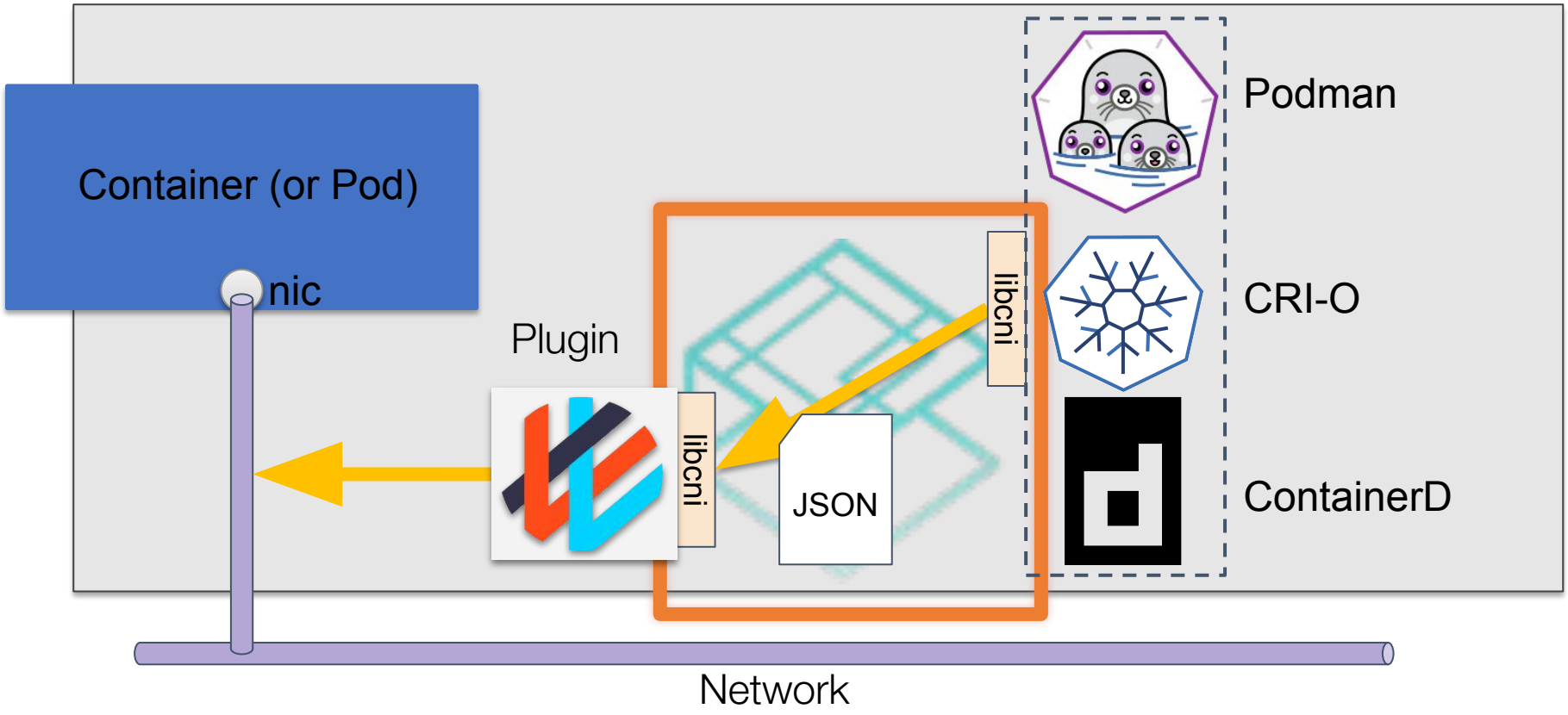
CNI Deep Dive

What isn’t the CNI project?

- All of Kubernetes networking
  - CNI only covers container connectivity, not load-balancing, service discovery, policy, etc.
- Kubernetes-specific
  - Used by podman, AWS Fargate, Mesos…
- The only set of plugins you can use
  - Check out Weave Net, Calico, etc...


KubeCon
CloudNativeCon
Europe 2020
How does it all work?
CNI Deep Dive

What is CNI?


Container (or Pod)
nic
Plugin
libcni
JSON
libcni
Podman
CRI-O
ContainerD
Network

Execution Flow

- Four commands: ADD, DEL, CHECK, and VERSION
- **Plugins are executables**
- Spawned by the runtime when network operations are desired
- Fed JSON configuration via stdin
- Report versioned JSON result via stdout

Simple!

Configuration Format

- JSON-based configuration. Usually lives on disk.
- Consumed by both runtime **and** plugin

```
{
    "cniVersion": "0.4.0",
    "name": "mynet",
    "type": "bridge",
    "bridge": "mynet0",
    "isDefaultGateway": true,
    "forceAddress": false,
    "ipMasq": true,
    "hairpinMode": true,
    "ipam": {
     "type": "host-local",
     "subnet": "10.10.0.0/16"
    }
}
```

Dynamic configuration

Spec v0.3+ defines substitution points for dynamic configuration.

Allows for mix of static, on-disk & dynamic, per-container settings.

```
{
  "name" : "ExamplePlugin",
  "type" : "port-mapper",
  "capabilities": {
    "portMappings": true
  }
}
```

```
{
  "name" : "ExamplePlugin",
  "type" : "port-mapper",
  "runtimeConfig": {
    "portMappings": [
      {
        "hostPort": 8080,
        "containerPort": 80,
        "protocol": "tcp"
      }
    ]
  }
}
```

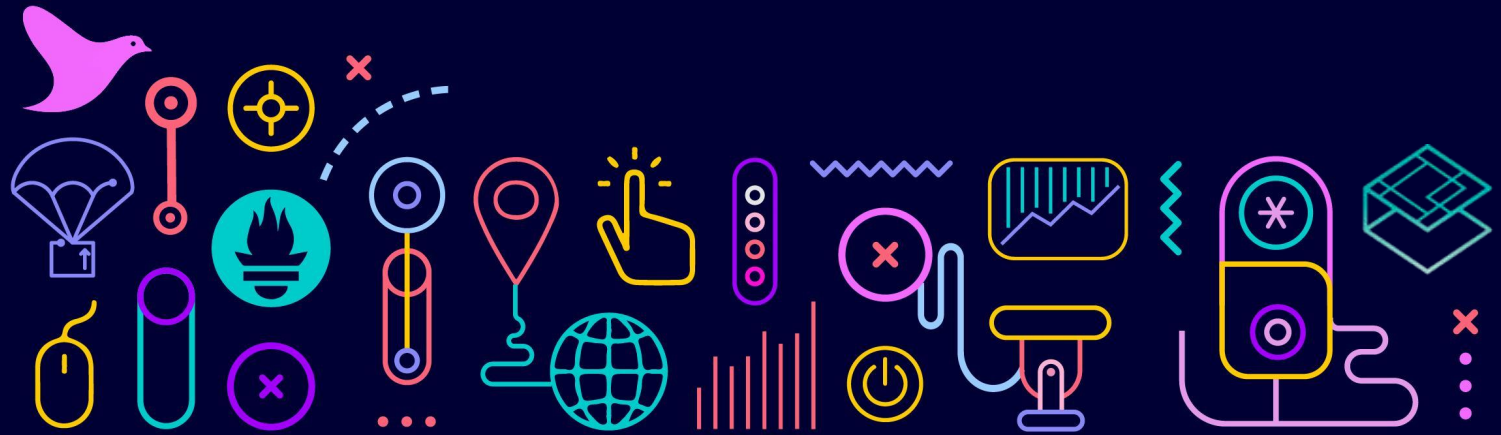
Plugin Chaining

CNI also allows for multiple plugins to be executed in a row. (Spec v0.3+)

Used for tweaking / tuning, **not** multiple interfaces.

```
{
  "name" : "mynet",
  "cniVersion" : "0.3.1",
  "plugins": [
    {
      “type”: “bridge”, “foo”: “bar”, ...
    },
    {
      “type”: “portforward”, ...
    }
  ]
}
```

How does the CNI project work?



KubeCon | CloudNativeCon Europe 2020

CNI Deep Dive

How does the CNI project work?

Spec:

- Actively maintained, but slow cadence
- Trying to hit 1.0 in a month.

Plugins:

- Faster release cadence
- Lots of contributors

Kubenet, kube-proxy, etc:

- Are in a different project. CNI works with, but is independent of Kubernetes

Who is the CNI project?

Seven maintainers:

- Bruce Ma (Alibaba)
- Bryan Boreham (Weaveworks)
- Casey Callendrello (Red Hat)
- Dan Williams (Red Hat)
- Gabe Rosenhouse (Pivotal)
- Matt Dupre (Tigera)
- Piotr Skamruk (CodiLime)

Lots of contributors!

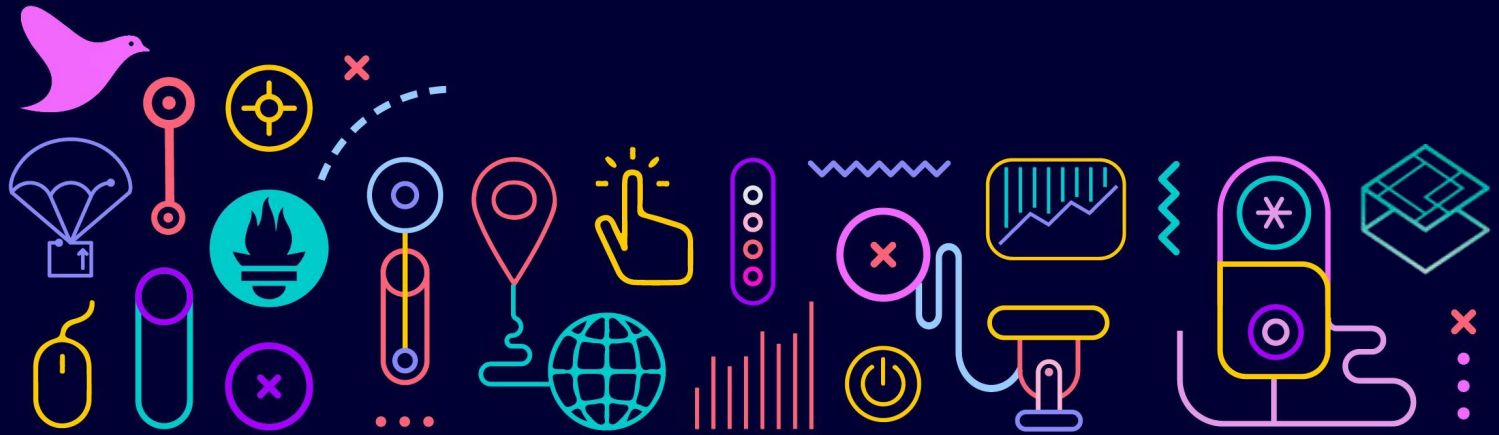
What happened recently?

- Spec v0.4: CHECK support
- Lots of new features
  - Better static IP & MAC support
  - Layer 2 bridges
  - Host-device plugin plays nicely with Kubernetes device plugins
- CRI-O implements configuration checkpointing
- And a CVE
  - Come to deep-dive to learn more!

What's next?

- 1.0 is now feature-complete; the spec is stable
- We still need:
  - More precise specification
  - Signed release binaries
- Dreaming about 2.0
  - gRPC? More on this later...
  - Richer lifecycle?
  - What do you want?

FAQs



KubeCon | CloudNativeCon Europe 2020

CNI Deep Dive

FAQs

Q: Why do some CNI plugins come with a host-side daemon?

A: CNI is only executed on pod creation and deletion. Need a long-running process to maintain routing tables.

In other words, the Kubernetes and CNI lifecycles are different.

For example, this daemon might need to react to node events.

FAQs

Q: Can I have multiple interfaces in a pod?

A: Sure! Just use a runtime that supports multiple CNI attachments.

Real answer: Sure! Just use a multiplexing CNI plugin like Multus, cni-Genie, or DANM.

FAQs

Q: How should I deploy my plugins and configuration on Kubernetes?

A: Complicated, best practice is to use a DaemonSet to “leak” files to the host.

We’d like to come up with a better story for this. Sig-cluster-lifecycle has a cluster-addons sub-project, maybe they will come up with one.

https://github.com/kubernetes-sigs/cluster-addons

FAQs

Q: Do you support IPv6?

A: Yes! Spec has supported it for years, plugins have since v0.6.0.

Kubernetes supports single-stack IPv6. Dual-stack works, though still has some pain points.

FAQs

Q: Why executing binaries? Why not gRPC?

A: Long story. CNI came out of rkt, which is daemonless. Executing a program is a really simple API.

We'd like to offer gRPC for CNI v2.0.

FAQs

Q: What are some cool chained plugins?

A: I'm glad you asked!

portmap: set up port mappings from host to container

tuning: tweak sysctls, mac, etc.

bandwidth: apply bandwidth limitations

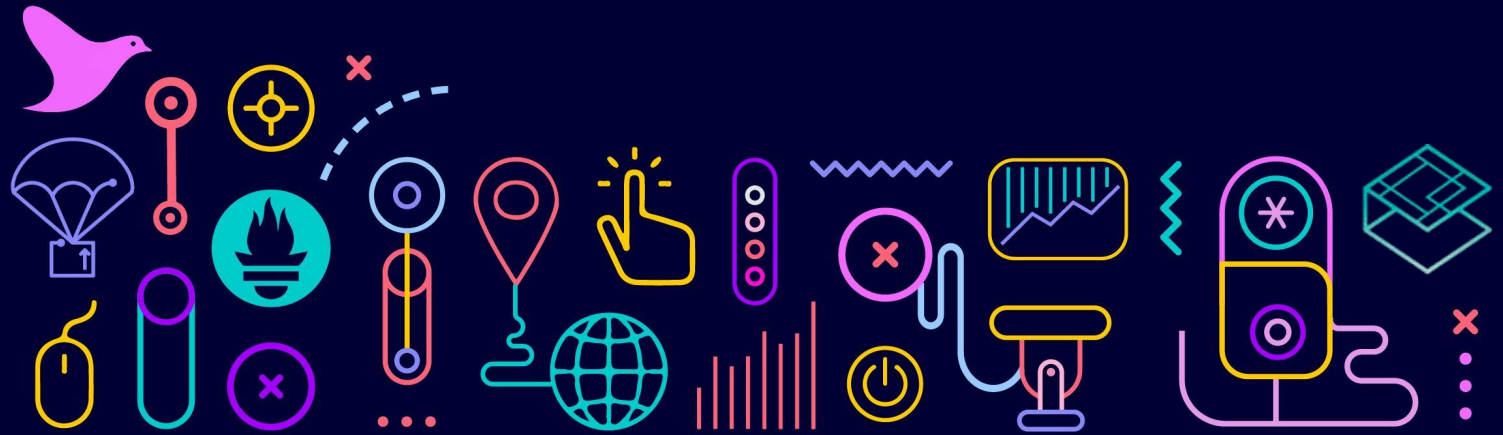
firewall: Add your container to iptables or firewalld

sbr: source-based routing

Information and links

- CNI Deep Dive session https://sched.co/ZexP / Aug 19 13:00 CEST
- GitHub repos:
  - https://github.com/containernetworking/cni
  - https://github.com/containernetworking/plugins
- Slack - https://slack.cncf.io - topic #cni
  - https://slack.k8s.io/ #sig-network for k8s-specific topics.

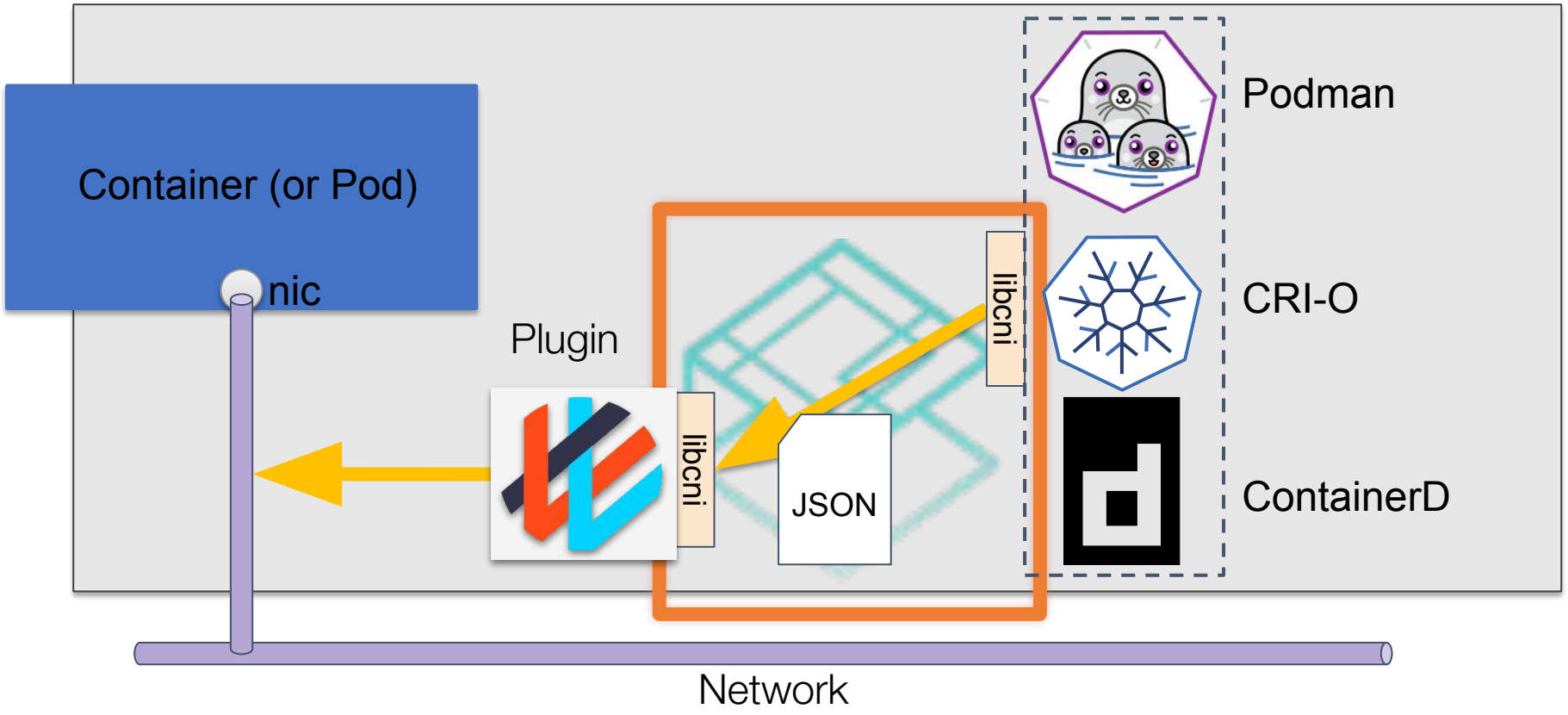
Thanks!



KubeCon | CloudNativeCon Europe 2020

CNI Deep Dive

Questions?


Container (or Pod)
nic
Plugin
libcni
JSON
libcni
Podman
CRI-O
ContainerD
Network


KubeCon
CloudNativeCon
Europe 2020
Virtual
HELM
KEEP CLOUD NATIVE
CONNECTED