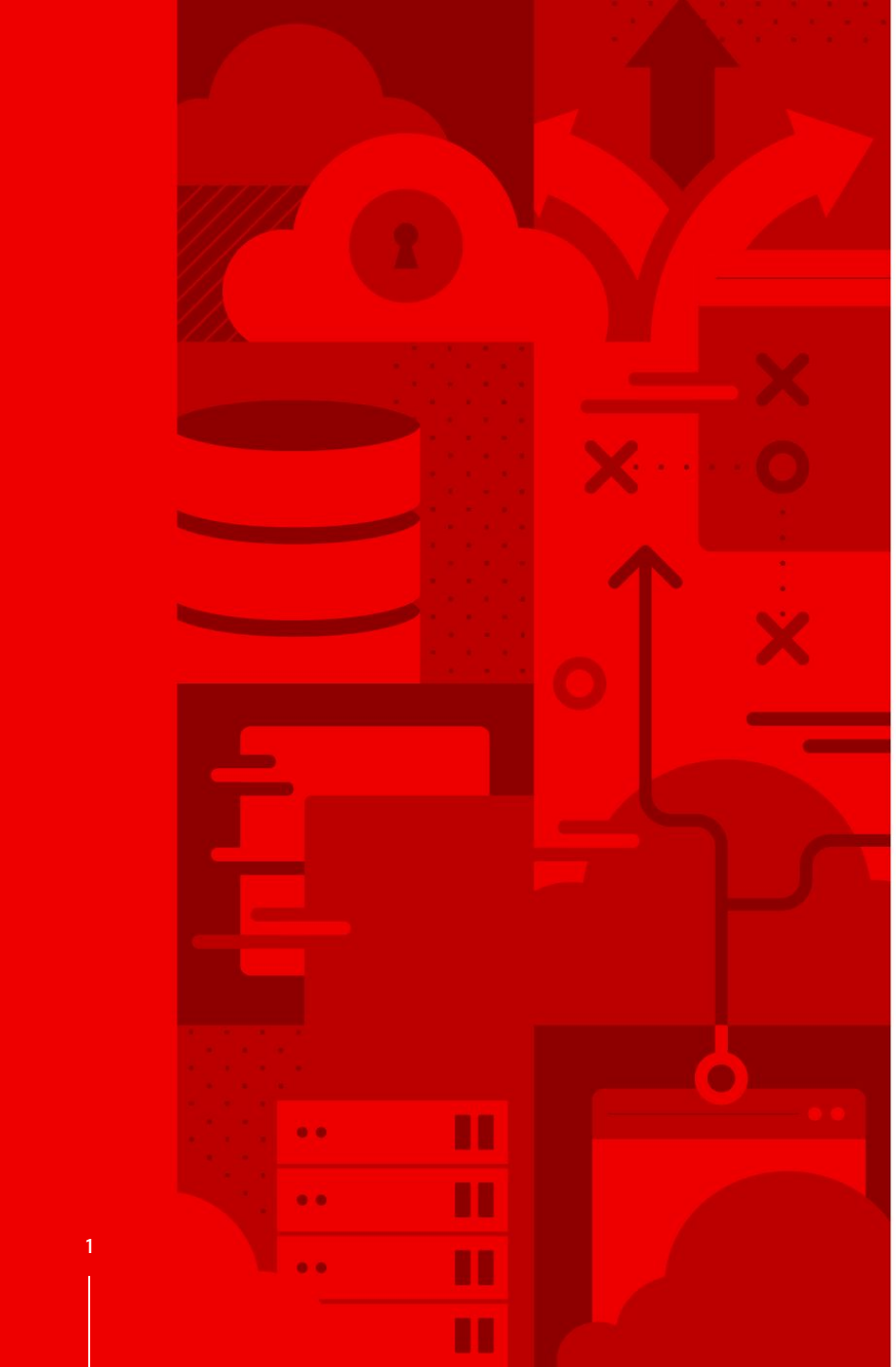


CNI Automagic: Device Discovery for semantic network attachment in Kubernetes

FOSDEM 2023

Douglas Smith

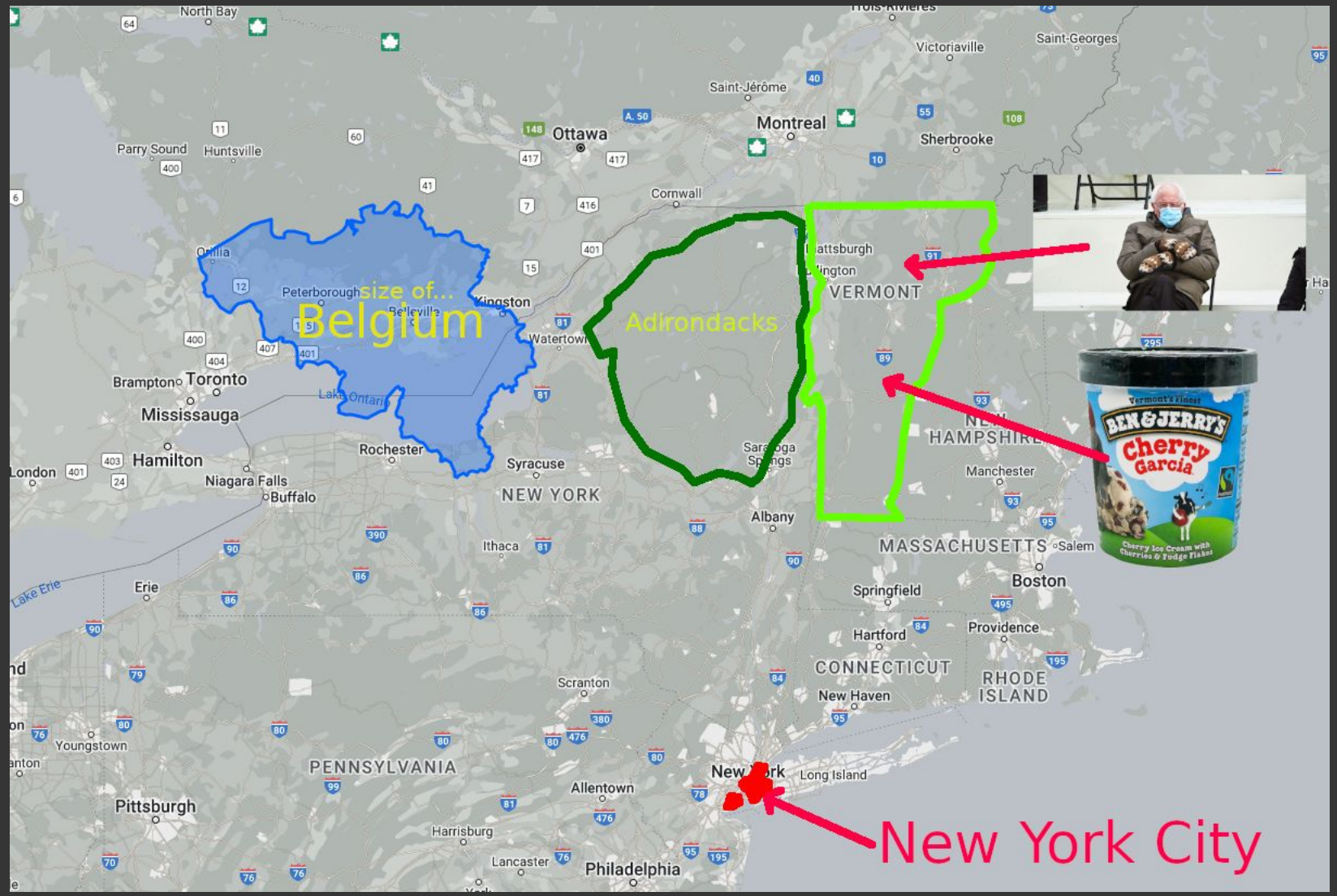
Principal Software Engineer, OpenShift Engineering @ Red Hat, Inc.

@dougbtv

Doug Smith

- Technical lead for OpenShift Network Plumbing Team in OpenShift Engineering
- Multus CNI maintainer
- Network Plumbing Working Group member
- Blog: https://dougbtv.com

From Vermont!



Belgium: 30.6k km^2
Adirondacks: 24.3k km^2
Vermont: 24.9k km^2
New York City: 0.8k km^2

Agenda

- Problem Statement
- Tour of Surveyor CNI
- Known issues / Possible Solutions
- Impact on CNI 2.0!
- Call to action!

Problem Statement.

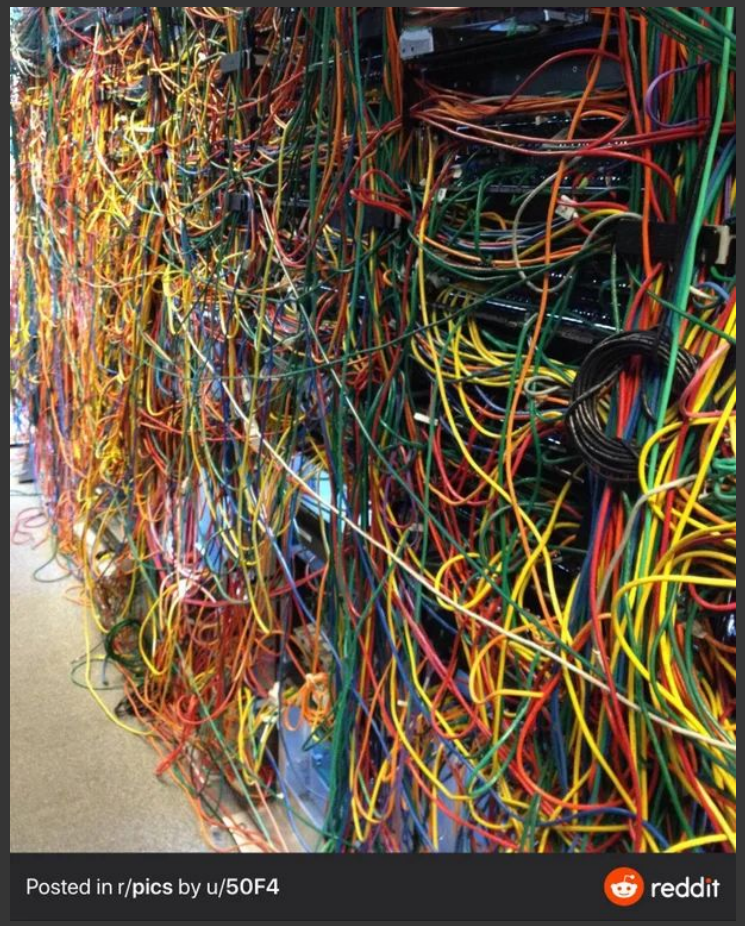
First things first!

- This is a problem and it's somewhat of an interesting solution
- But! More-so, I want to look at the approach that I took and how that plays into CNI 2.0, so keep that in mind the whole time.

How networks look in diagrams.

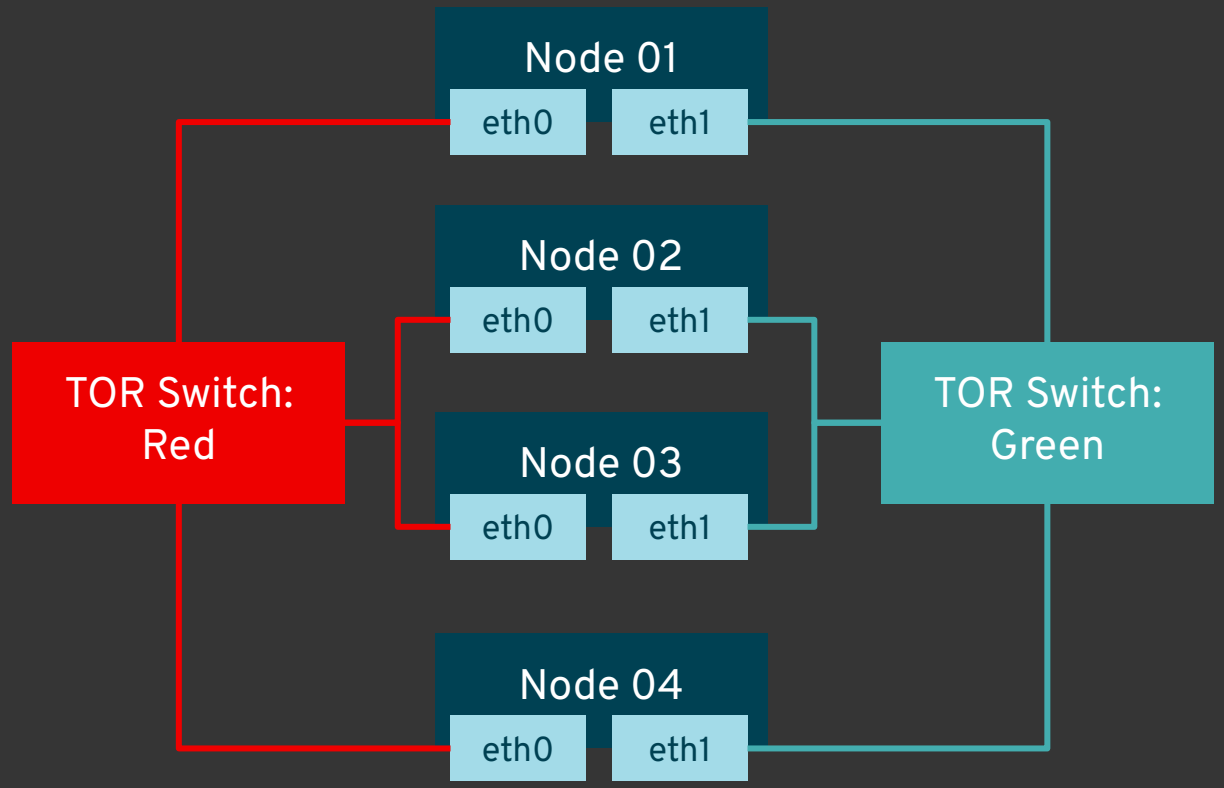
Source: Reddit

How networks look in real life...

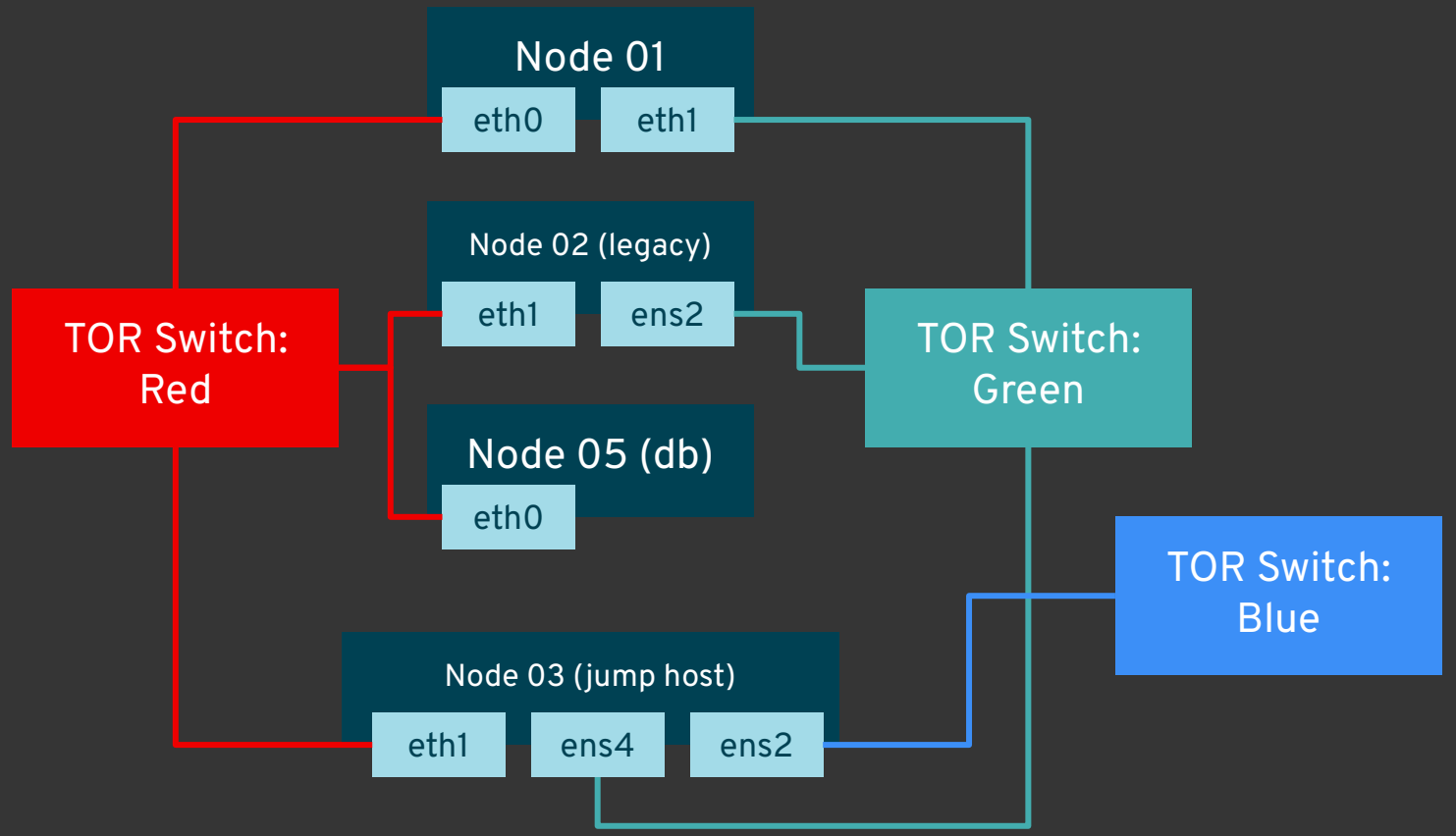


Source: Reddit

In an ideal world, all things are uniform.



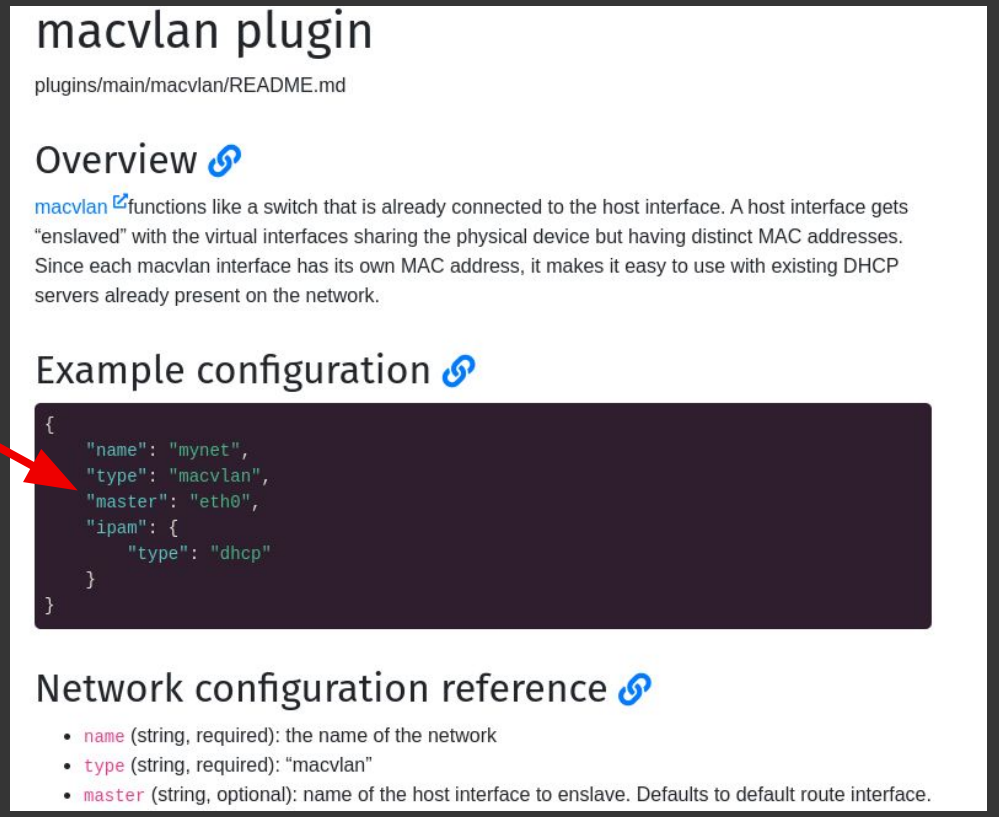
The real world breeds nonuniformity.



In a nutshell…

CNI plugins that refer to specific interfaces aren’t scaleable for non-uniform environments.

Often our CNI configurations want to know which interface... but which network does that attach to?



macvlan plugin

plugins/main/macvlan/README.md

Overview

macvlan functions like a switch that is already connected to the host interface. A host interface gets "enslaved" with the virtual interfaces sharing the physical device but having distinct MAC addresses. Since each macvlan interface has its own MAC address, it makes it easy to use with existing DHCP servers already present on the network.

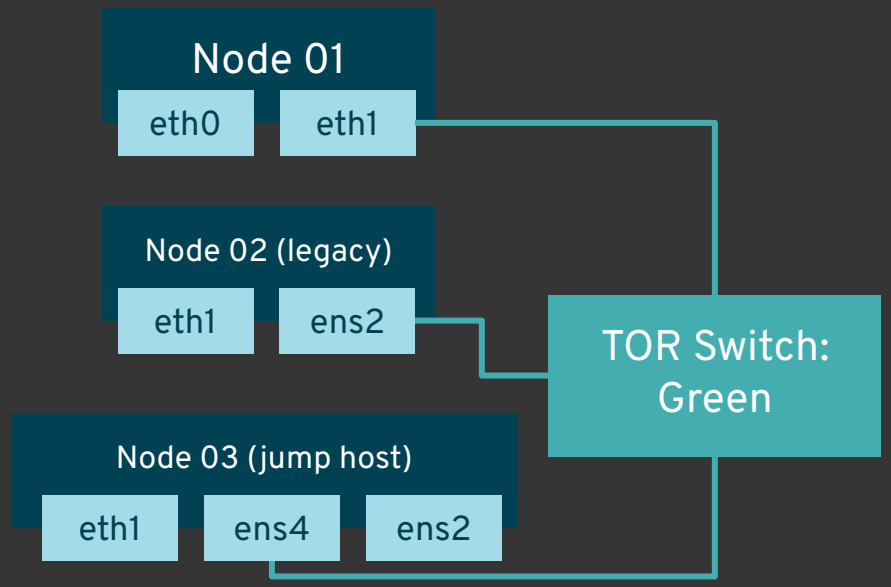
Example configuration

```
{
    "name": "mynet",
    "type": "macvlan",
    "master": "eth0",
    "ipam": {
        "type": "dhcp"
    }
}
```

Network configuration reference

- `name` (string, required): the name of the network
- `type` (string, required): "macvlan"
- `master` (string, optional): name of the host interface to enslave. Defaults to default route interface.

Great, now I need a CNI configuration for each node, and a node selector for each pod. I don't wish anyone to scale this from 3 nodes to 30.



CNI config...

```
{
    "name": "mynet",
    "type": "macvlan",
    "master": "eth1",
    "ipam": {
        "type": "dhcp"
    }
}
```

pod...

```
apiVersion: v1
kind: Pod
metadata:
  annotations:
    k8s.v1.cni.cncf.io/networks: macvlan-eth1-node01
spec:
  containers:
  - name: [...]
  nodeSelector:
    eth1-on-green: "true"
```

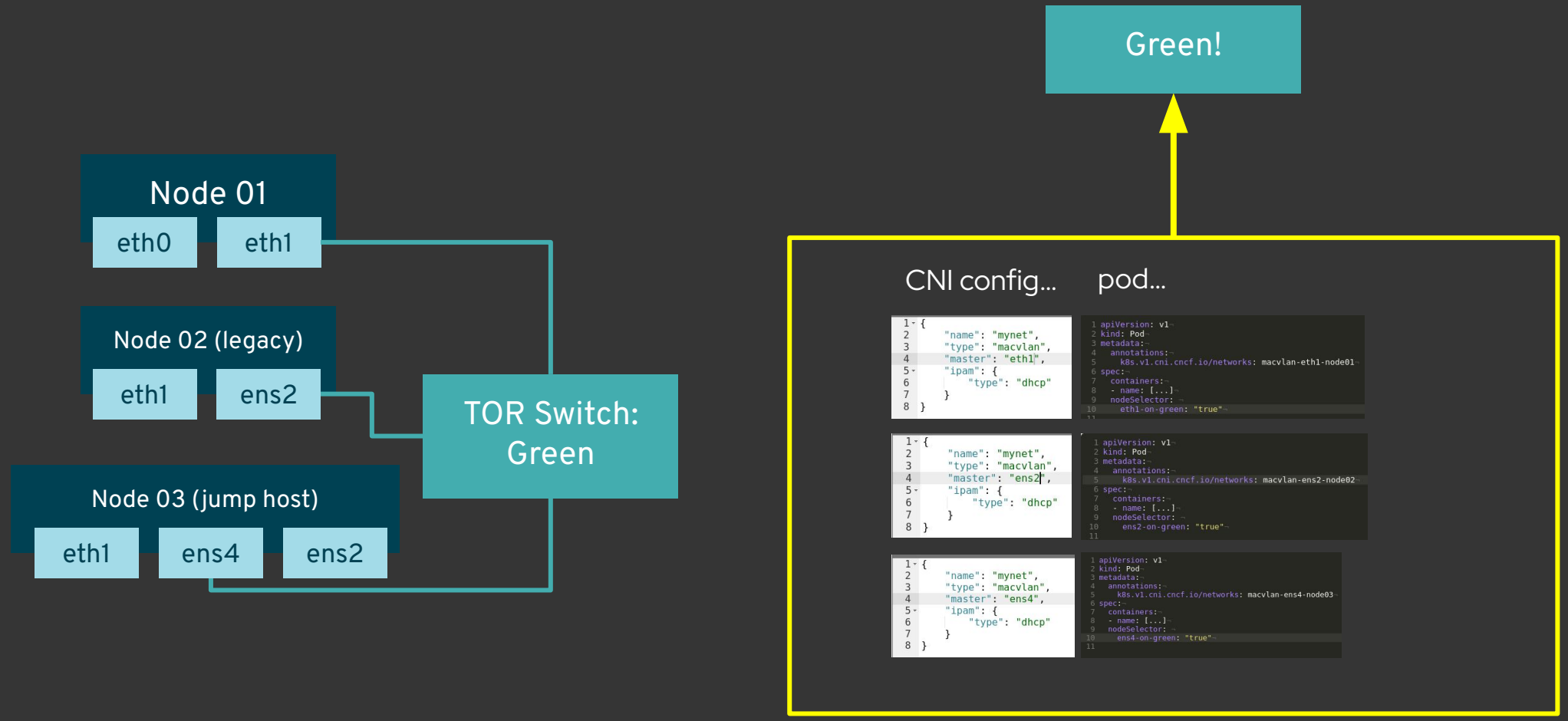
```
{
    "name": "mynet",
    "type": "macvlan",
    "master": "ens2",
    "ipam": {
        "type": "dhcp"
    }
}
```

```
apiVersion: v1
kind: Pod
metadata:
  annotations:
    k8s.v1.cni.cncf.io/networks: macvlan-ens2-node02
spec:
  containers:
  - name: [...]
  nodeSelector:
    ens2-on-green: "true"
```

```
{
    "name": "mynet",
    "type": "macvlan",
    "master": "ens4",
    "ipam": {
        "type": "dhcp"
    }
}
```

```
apiVersion: v1
kind: Pod
metadata:
  annotations:
    k8s.v1.cni.cncf.io/networks: macvlan-ens4-node03
spec:
  containers:
  - name: [...]
  nodeSelector:
    ens4-on-green: "true"
```

I want to take all this information and loft it up so it can be stored in the Kubernetes API, and it's abstracted so I can just say the network I want is "GREEN!"



We need to add some meaning to our network interfaces, so we can scale our setups.

We use Kubernetes for scale.
We use CNI for plumbing network interfaces.

Surveyor CNI!

Surveyor CNI

- Maps devices to network names using CRDs (custom resource definitions)
- Name inspired by my favorite explorer and topographic engineer of the Adirondacks, Verplanck Colvin

Surveyor CNI works in two phases:

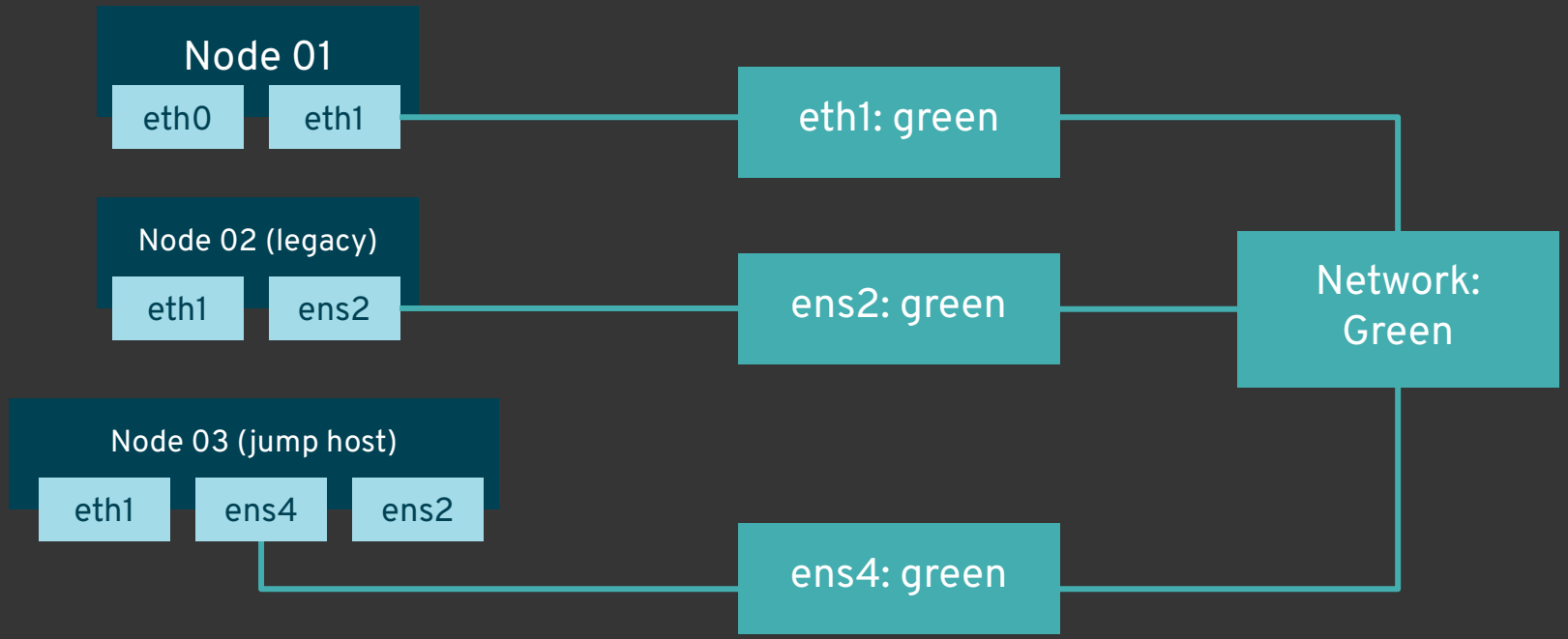
1. 🔍 Inspection

   Surveyor initializes a custom resource with an empty map of each of your host interfaces.

2. Mapping association

   Then, went executed as a CNI plugin, Surveyor looks up the mapping from interface name to network.

Surveyor provides a mapping CRD that allows you to associate interfaces with networks.



Surveyor CNI is just an example, something that REALLY scales might make these associations using something smarter...

- A K8s controller that knows some rules about your network

- A listener on NETLINK
  - Maybe you could detect when an physical cable is plugged in? (Or when a VPC is created)
    - Or What if the cable was moved...
  - Macvtap CNI does have a netlink listener!

- Maybe you can train an AI to know something about your network (ChatGPT will confidently make it up)

Why not just apply aliases to the network interfaces?

Sure! You should absolutely do that today, extremely reliable.

But as we more deeply integrate our network-centric applications at scale, we want finer control of them using Kubernetes and Kubernetes controllers, and we don’t want to have to fish that information out of configuration files.

And it’s harder to add intelligence to this configuration if and when our network changes on the fly.

I would say we could loft it to Kubernetes, or lower it to machine configurations. I think K8s developers want to loft it. But it’s a worthwhile consideration in your clusters.

Known Issues & Possible Solutions

Pitfalls!

- Something that people bring up is the problems that Neutron brought to OpenStack.
- Having a kind of “interface manager” for Kubernetes might bring this same suite of problems.
- On the flip side: Maybe a modular piece like this is more fitting to a Unix philosophy.
- Device plugins are necessary to properly scale.

Source: Activision Games Blog

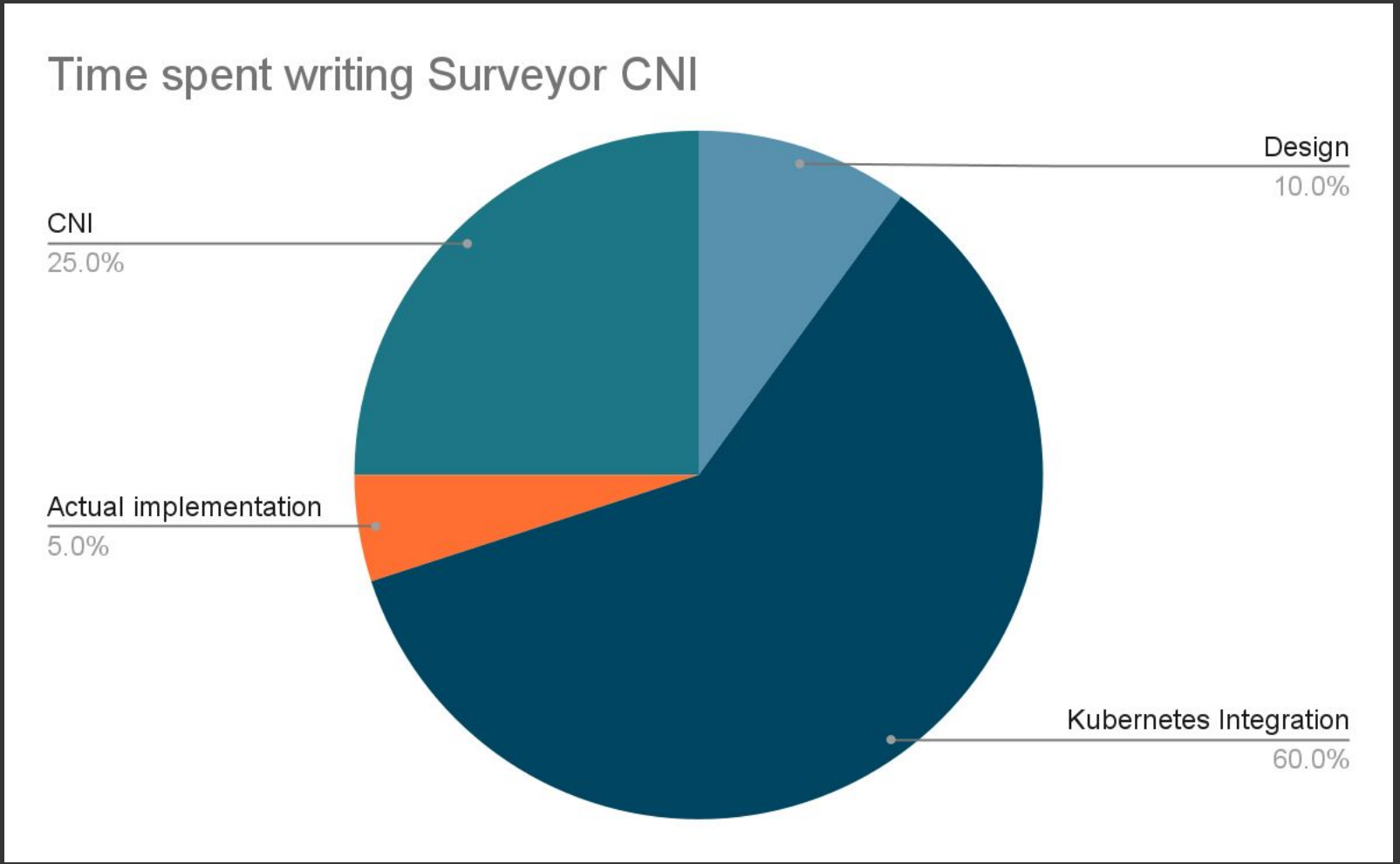
Impact on CNI 2.0

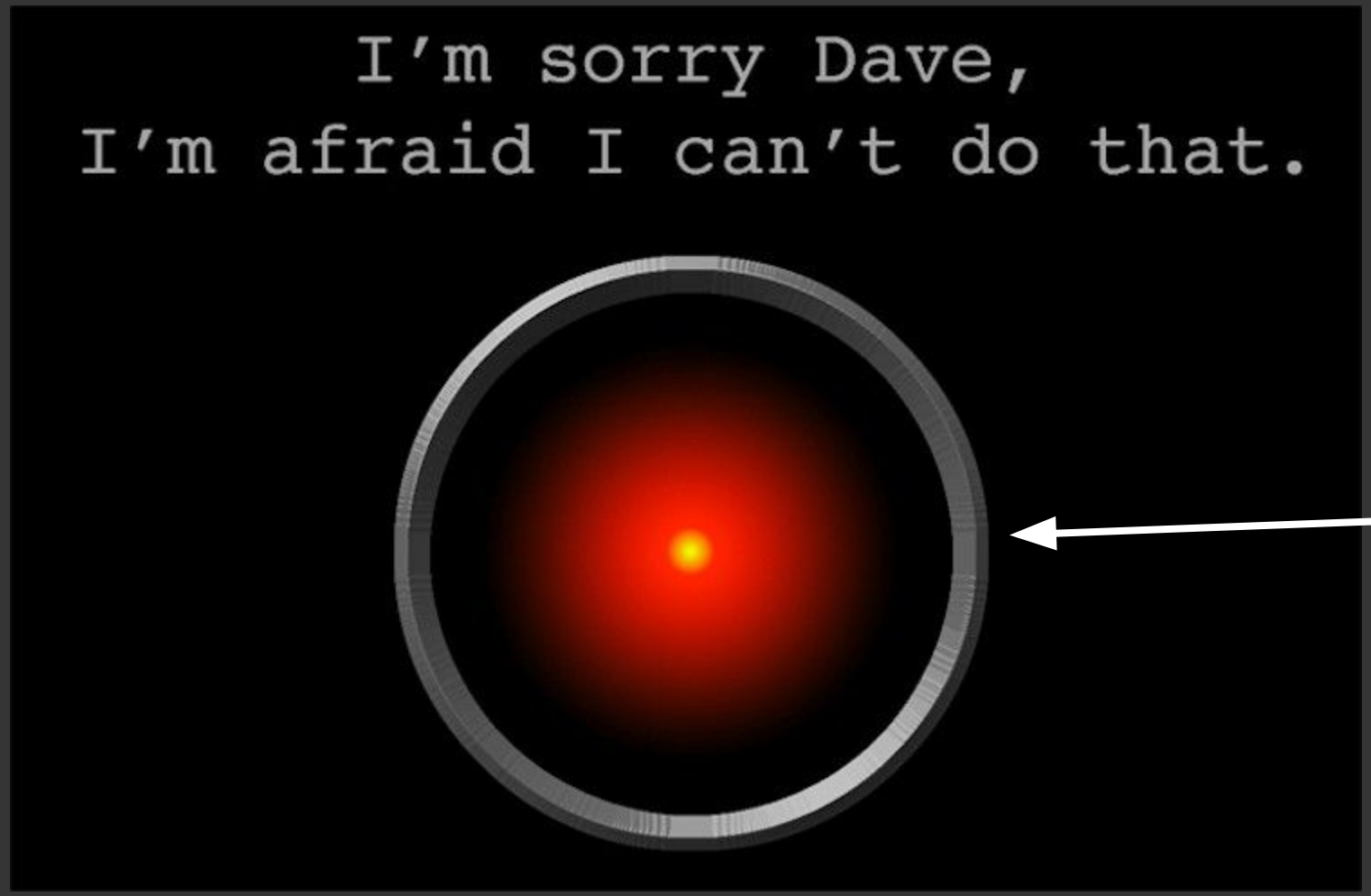
As both a CNI developer, and a Kubernetes developer, the first thing I went to use was...

Kubernetes.

I needed to figure out how to interact with the Kubernetes objects (again!)

Where did I spend my time?




I'm sorry Dave,
I'm afraid I can't do that.
ChatGPT

CNI 2.0 needs to be able to communicate with Kubernetes to evolve.

We want YOU to be involved in the evolution.

(That's more important than this demo!)

Try it out!

https://github.com/dougbtv/surveyor-cni

(intentionally blank)