# Mayday ! CNI overboard

June 16th 2023, DevConf CZ

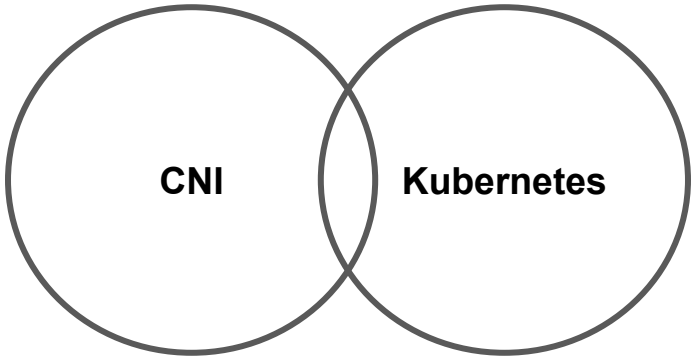
**Daniel Mellado**
dmellado@redhat.com
https://github.com/danielmellado

**Miguel Duarte Barroso**
mdbarroso@redhat.com
https://github.com/maiqueb

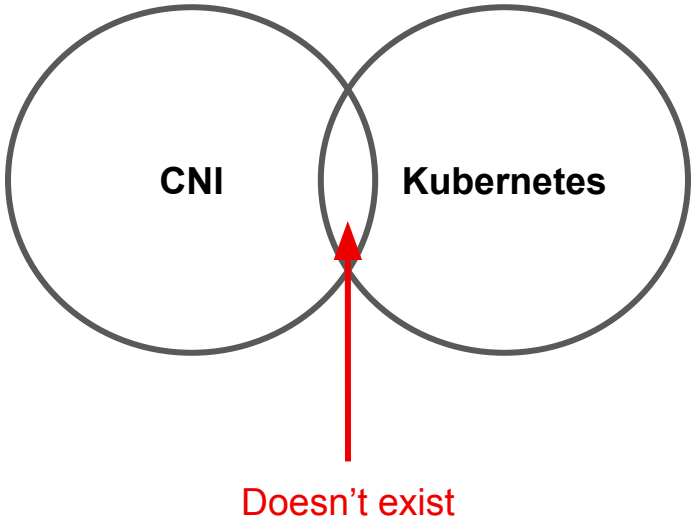
# Agenda

- Motivation
- CNI Introduction
- Problem specification
- Multi-network specification
- Upcoming CNI 2.0
- Conclusions

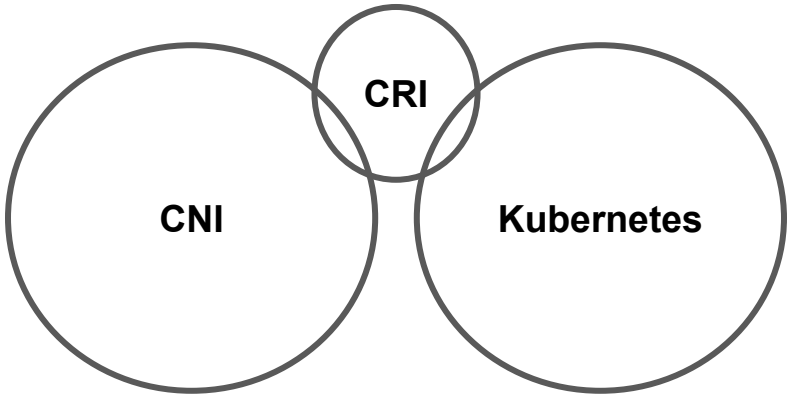
# Motivation

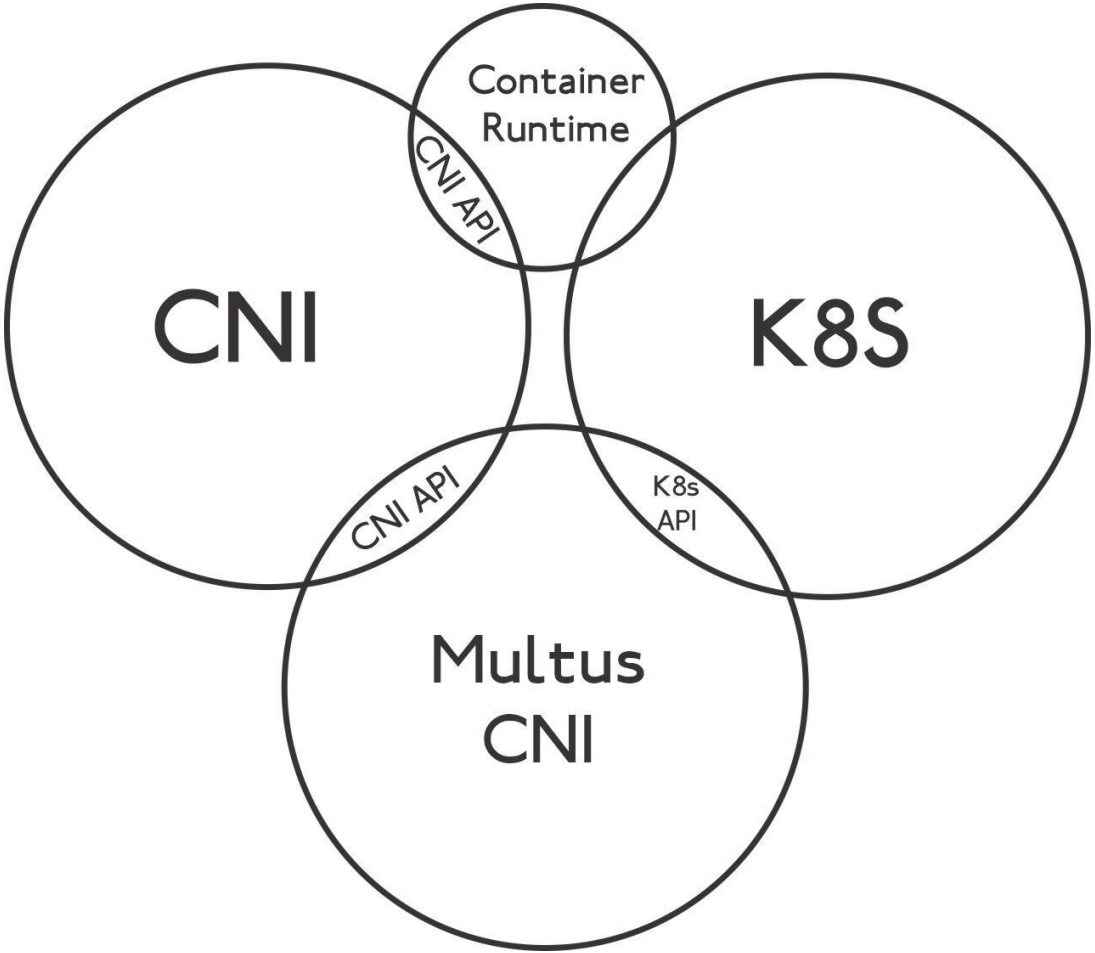


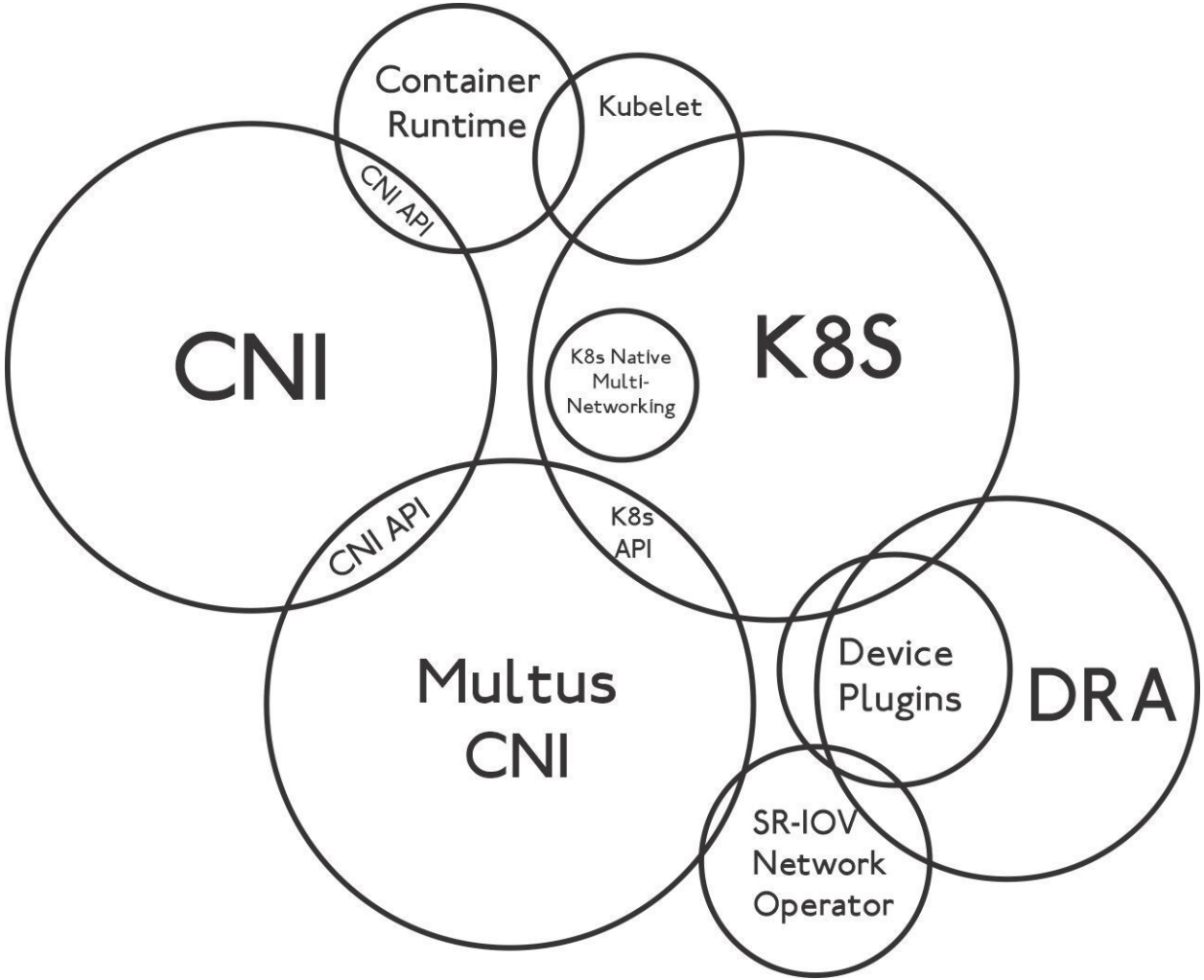
# Motivation



# Motivation




Container
Runtime
CNI API
CNI
K8S
CNI API
K8s
API
Multus
CNI

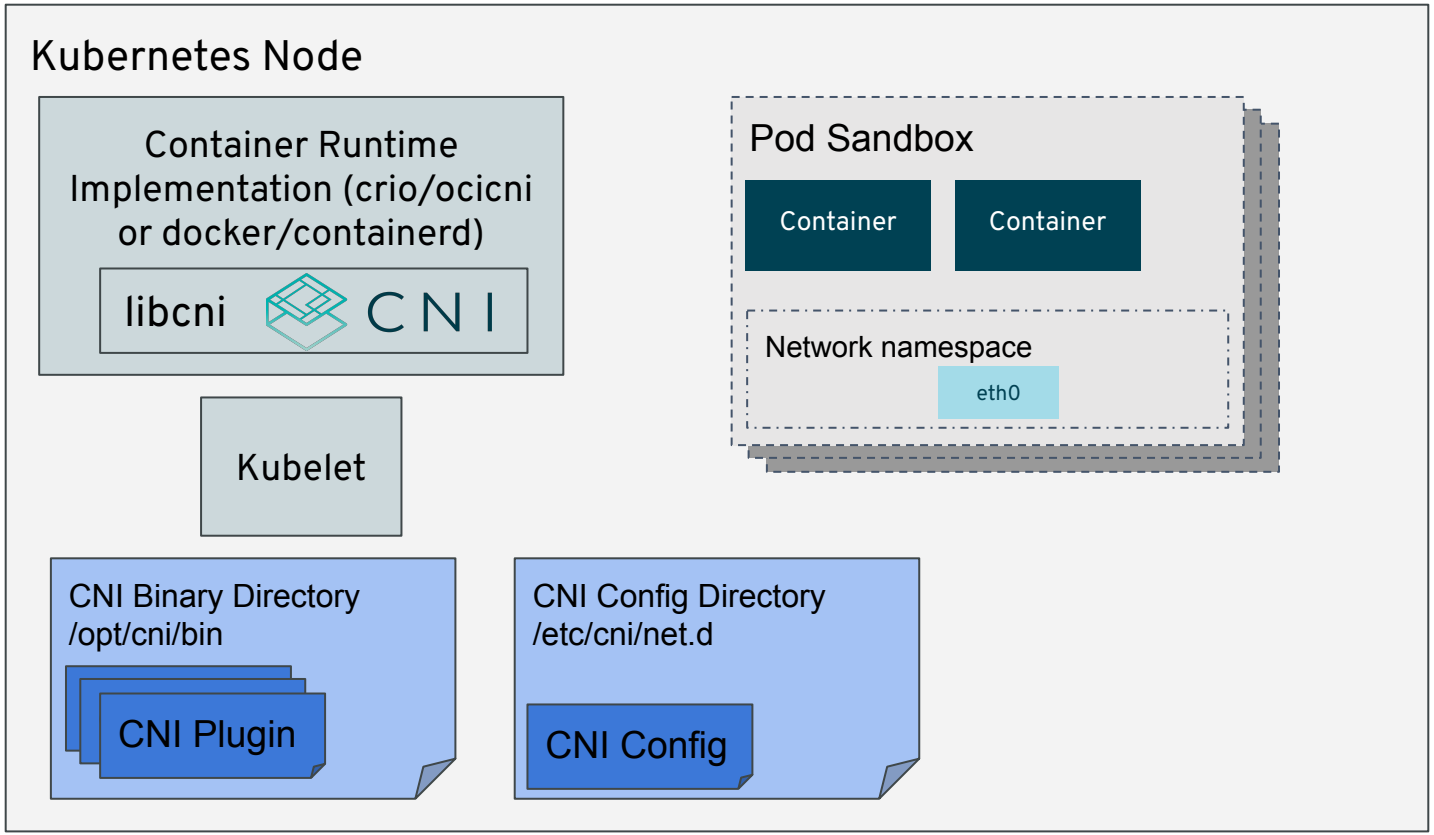

Container Runtime
Kubelet
CNI API
CNI
K8s Native Multi-Networking
K8S
CNI API
K8s API
Multus CNI
Device Plugins
DRA
SR-IOV Network Operator

# So what’s the deal?

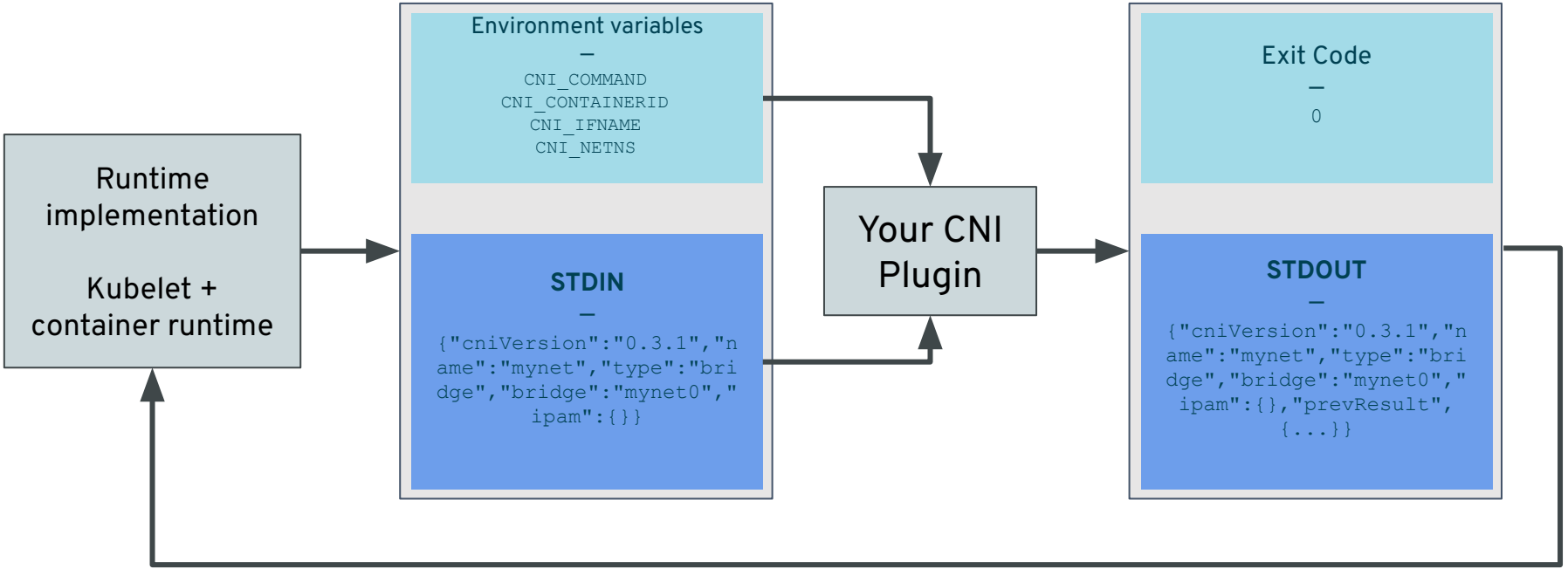
- Adding multiple interfaces in Kubernetes is difficult, and is done out of tree.
- CNI is the common language for speaking about networking in K8s
- There isn’t a K8s-native way to interact with CNI
- There’s a lot of moving parts

# Current state of CNI

# CNI anatomy: from a Kubernetes perspective



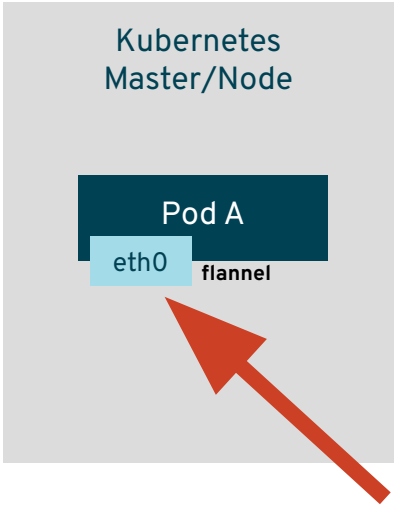
# CNI specification



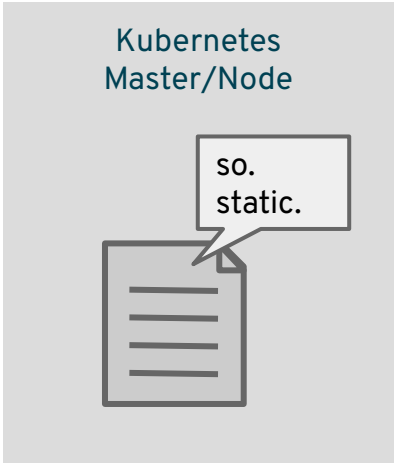
# Multus

# THE PROBLEM (As it were...)

#1 Each pod only has one network interface

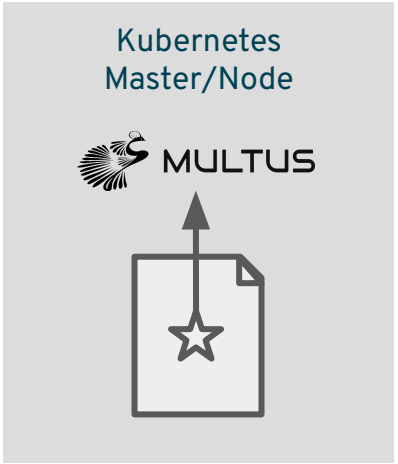



#2 Each master/node has only one static CNI configuration

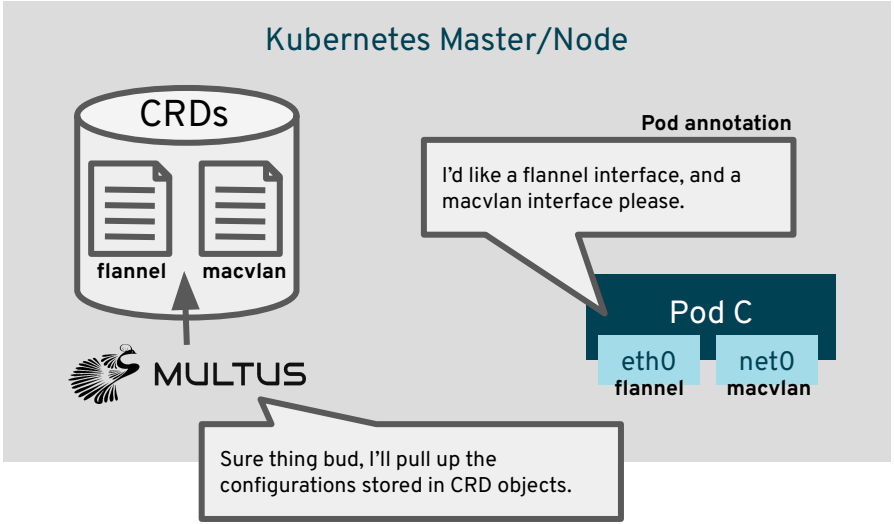


# HOW THE PROBLEM IS APPROACHED.

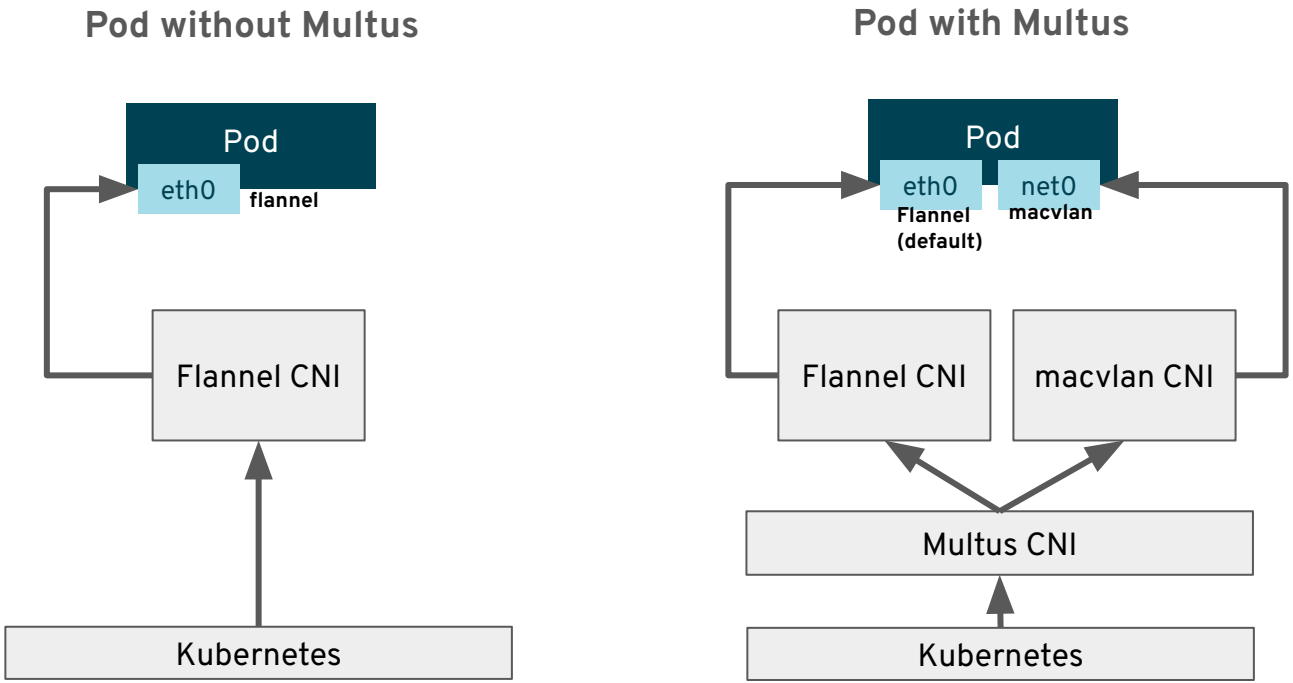
Static CNI configuration points to Multus




Each subsequent CNI plugin, as called by Multus, has configurations which are defined in CRD objects



# WHAT MULTUS DOES



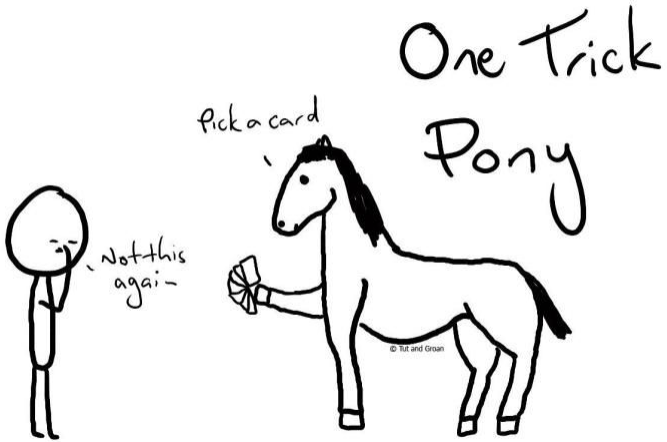
# Problem Statement

# Current CNI issues (an editorial)

- CNI is simple (good!)
  - … but developers need to reconcile the resources their CNI creates
  - … but lifecycle is limited (no garbage collection, only reacts to pod add / delete … )
- CNI user base is getting smaller (baaaaad !!!!)
  - Podman is replacing it with netavark
- CNI does not have native K8s integration (kinda bad)
  - Kubernetes developers don’t necessarily want to have integrate kind of a “second API” to manage networking.

# So what's the deal?

- CNI is modular, powerful, and simple
- … but also has cumbersome common flows (garbage collection)
- … isn't flexible enough (only reacts to pod add / del)




**src**: https://tutandgroan.com/one-trick-pony/

# Where is this going ?

- Existing proposal for a Kubernetes native multi-networking
  - Ignores CNI and puts the ecosystem at risk
- … maybe towards extinction ?

# Current state of K8s native multi-network

# Current state

- Breaking it into 3 KEPs / phases
  - Use cases
  - Object/API definition
  - Logical core k8s changes (services, networkpolicy, etc)
- Currently in API definition
- Implementation agnostic
  - You could ignore CNI entirely
  - Multi-net KEP is cloud focused
    - … what about “your” stuff running on a typical data-center ?...

# Advantages

- You have a k8s native way to write code
- Workloads with multiple interfaces
- Dynamic interfaces are captured in the requirements KEP proposal
  - ADD / DEL from a running pod
- HW based interface sharing / passthrough to the workloads are captured in the requirements KEP proposal
  - Has potential to reduce complexity for hardware device handling
- Native integration with network policies / services / …

# What we have now vs the in-tree proposal

```yaml
apiVersion: k8s.cni.cncf.io/v1
kind: NetworkAttachmentDefinition
metadata:
  name: tenantred
spec:
  config: |2
    {
            "cniVersion": "0.3.1",
            "name": "tenantred",
            "type": "ovn-k8s-cni-overlay",
            "topology": "localnet",
            "subnets": "192.168.123.0/24",
            "excludeSubnets": "192.168.123.1/32",
            "vlanID": 10,
            "netAttachDefName": "default/tenantred"
    }
```

```yaml
apiVersion: v1
kind: Network
metadata:
  name: dataplane
spec:
  ipam: "kubernetes"
  provider: "my-cni"
  routes:
  - dst: "42.42.42.0/24"
    gw: "14.0.1.1"
status:
  capabilities:
    networkPolicy: true
    service: false
```

# Overcoming CNI limitations

- Done case by case
  - Reconcile your allocated resources (IPs, etc)
  - Dynamic requests via multus-dynamic-networks
  - SLAAC => no answer for this use case yet (sounds like a particular type of interface update)

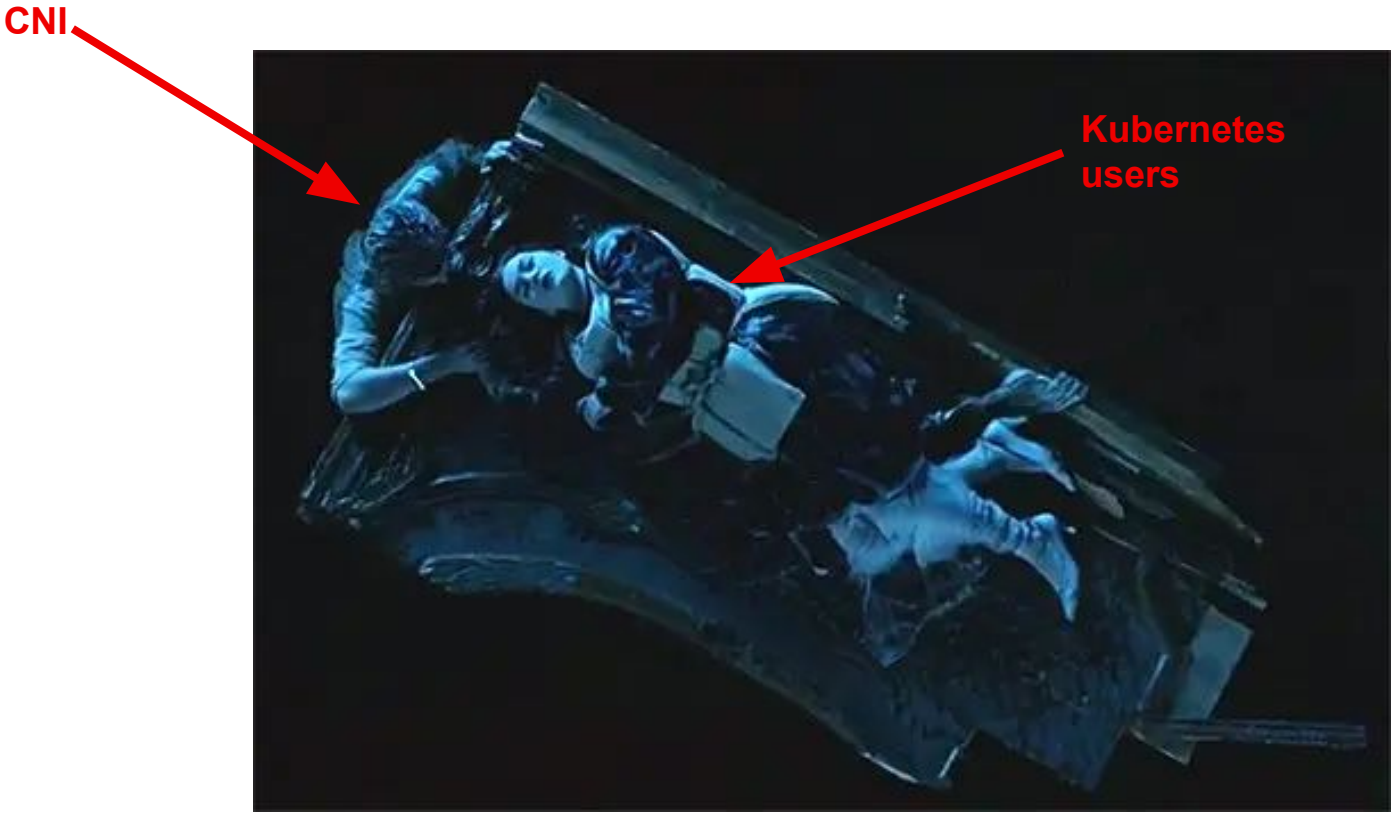
# CNI 2.0

- Issues
  - Consider daemonization: move away from an `exec` based API into a daemon
    - Additional lifecycle methods
    - Easier to deploy
    - Kubernetes native
  - Enhanced lifecycle: more than ADD / DEL / CHECK / VERSION
    - Garbage collection / INIT / Reconfiguration
  - Plugin events: plugins to CRI notifications
    - New interfaces / iface goes up / down
    - New IPs (SLAAC, …)
  - Device interactions: standardize HW device allocation
    - Currently relying on multus
    - DRA - Device Resource Allocation

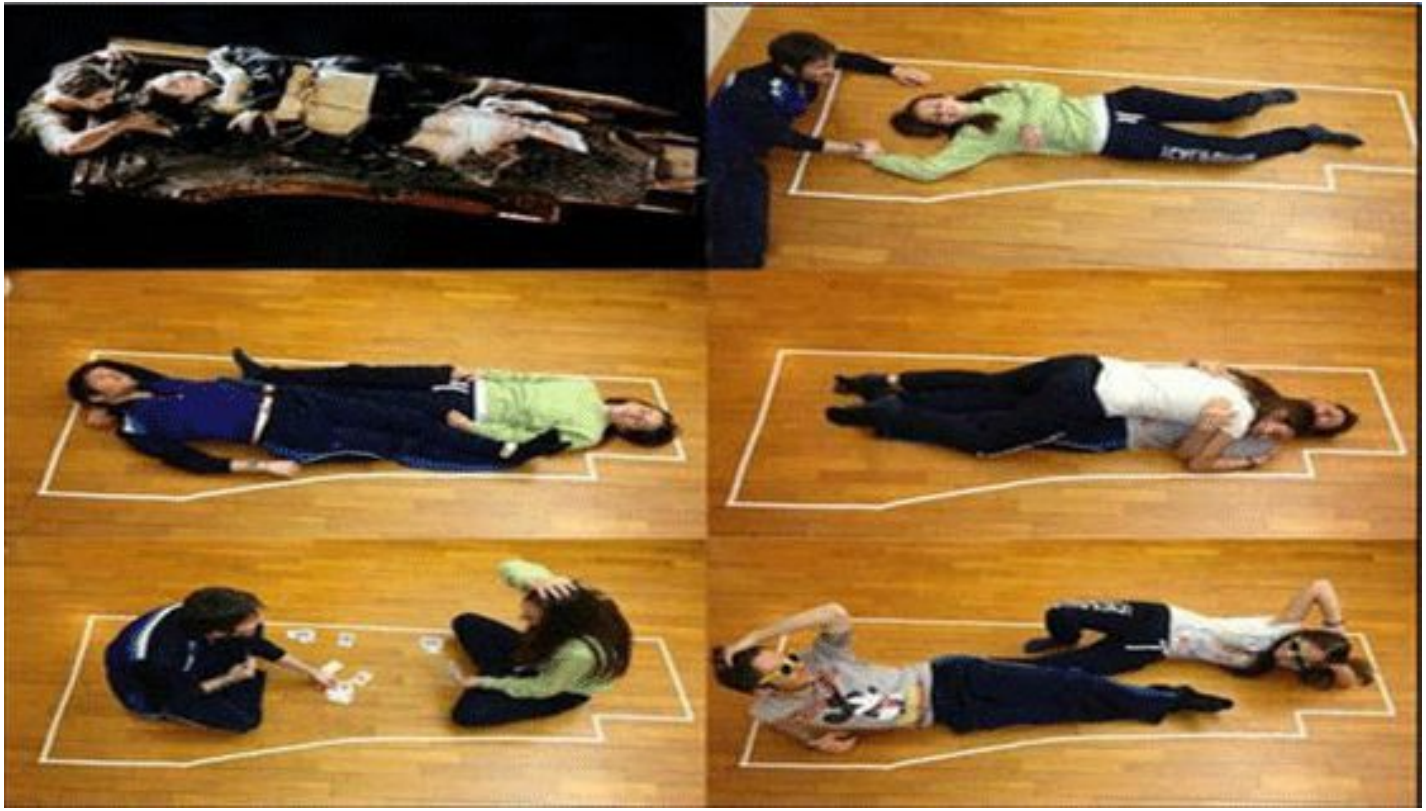
# Conclusions

# Conclusions

- The multi-net KEP is very strong in terms of use cases
- Addresses pretty much all current limitations of CNI
- Addresses pretty much all multus features
- CNI 2.0 better be good …
- You (users) can help !!!
  - Give feedback on the opened CNI 2.0 issues
  - Request missing features
  - …




Copyright: 20th century fox



Copied from: https://c.tribune.com.pk/2016/02/jack-and-kate1.gif
Original source unknown

Thank you !!!
Questions ?..