Two-node k3s distribution with NATS

David Gee and Byron Ruth, Synadia
Mark Abrams and Andrew Gracey, SUSE

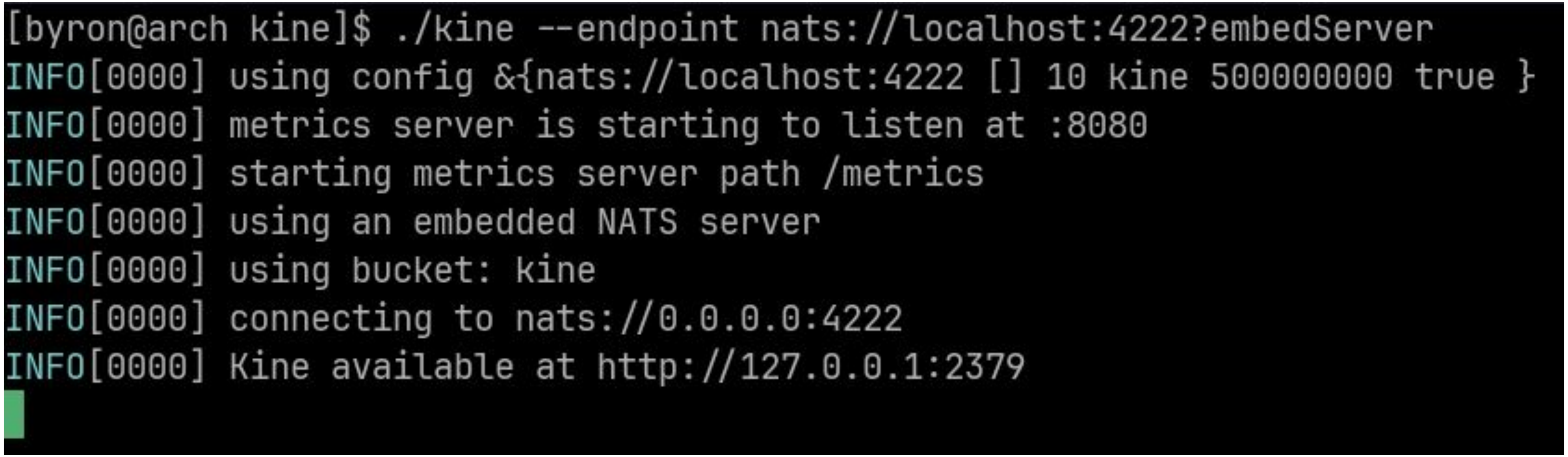
Goal

Distribution of k3s supporting a two-node setup with NATS as an embedded replacement for etcd.

This is a RAFT**less** implementation of a two-node NATS powered cluster!

Embedded NATS

Extend and Refactor



```
[byron@arch kine]$ ./kine --endpoint nats://localhost:4222?embedServer
INFO[0000] using config &{nats://localhost:4222 [] 10 kine 500000000 true }
INFO[0000] metrics server is starting to listen at :8080
INFO[0000] starting metrics server path /metrics
INFO[0000] using an embedded NATS server
INFO[0000] using bucket: kine
INFO[0000] connecting to nats://0.0.0.0:4222
INFO[0000] Kine available at http://127.0.0.1:2379
```

https://github.com/k3s-io/kine/pull/158

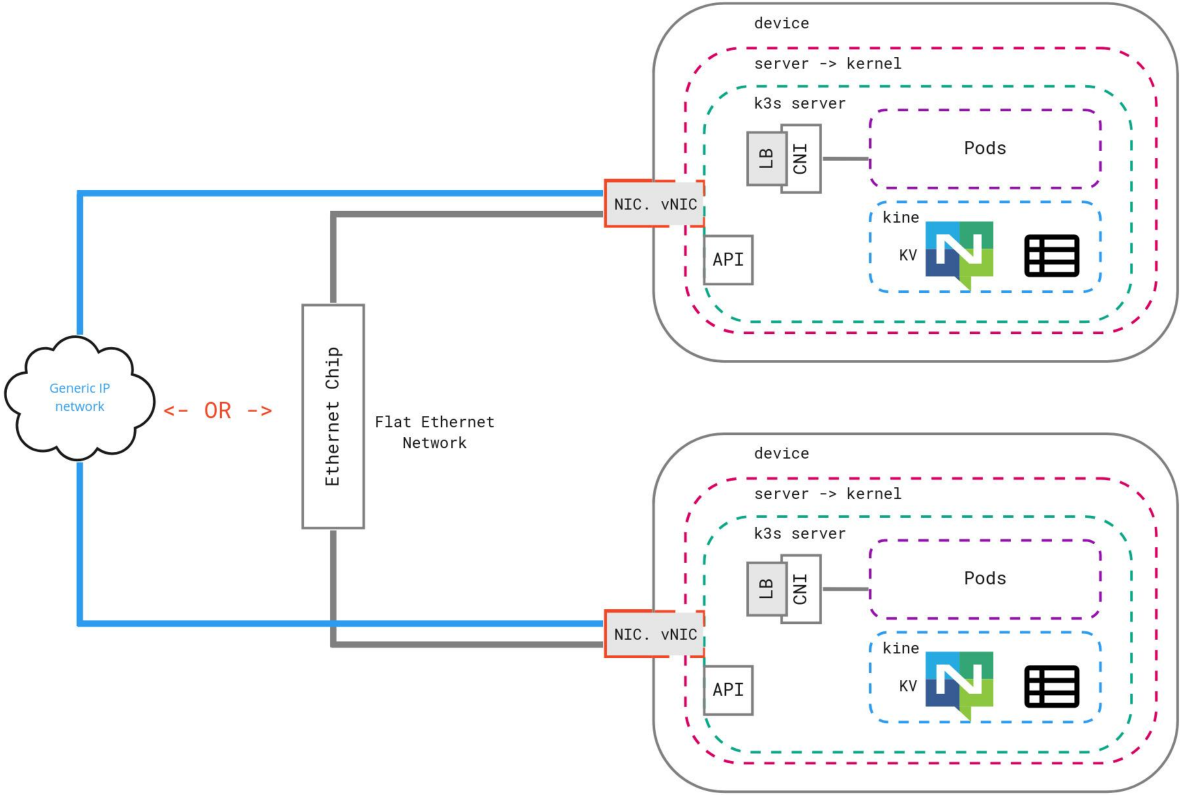
`go build -tags nats`

Two-node Problem

Constraints

- Strictly two k3s servers and no additional agents
- No load balancer required
- Unreliable east-west link between two nodes
- No required human intervention
- Raft protocol not in use

Network Schematic



Concrete Assumptions

- No shared storage
- Single NIC machines
- Nodes assume both server and agent roles
- Leader is explicitly configured

Loose Assumptions

- The network switch might be physically external, embedded in a hypervisor, a Linux bridge or OVS running on the OS (on a single machine) or a separate virtual-machine such as on OVS
- Power source is assumed to be shared, so if one node fails, they both fail
    - Not vital, just calling it out for coverage
- **If architects or designers start thinking about this for cloud, then shame on them! Stop being cheap and buy a third node and make use of RAFT**
- A physical network interface card (NIC) can be sliced into virtual functions, such as with SR-IOV.A network SR-IOV slice might fail (weird, but ok?). It could be possible that a pod connects directly to a network slice, but it's imperative that the API server and host communications colocate on a single NIC or vNIC (SR-IOV slice).

Loose Assumptions

- When a node is impaired, the resource set should NOT be changed. As with an Akri use case, this behaviour might need overriding with something like an *impairedMutationList*, but it should be noted that the changes MIGHT be temporary until a re-convergence event happens
- The concept of Leadership is a configuration input and not changeable on the fly??? TBC
- Configuration is static from an external perspective, with only operators making changes post deployment
- Where the nodes are on separate networks, the logical tests still apply
- When the leader is impaired, prioritisation and preemption could be enabled so the leaderRX might need to mutate to cope with the prioritised workload. That means the *approvedMutationList* must contain the prioritised list of allowed mutations

Failure Modes

Failure States

- Calculated through signals derived from local and cross-network tests
- Simple state machine to provide these states:
    - **Leader**
    - **LeaderRX**
        - Restricted Leader mode with approved mutations through an *approvedMutationList* which works in concert with preemption and prioritisation settings
    - **Impaired**
        - Broken but…*impairedMutationList* which is a “limp home” equivalent mode

Local Failures (Impaired State)

- Physical network uplink failure
- NIC failure
- vNIC failure
    - Due to cross pod and internal server connectivity failure
- CNI failure
- API failure

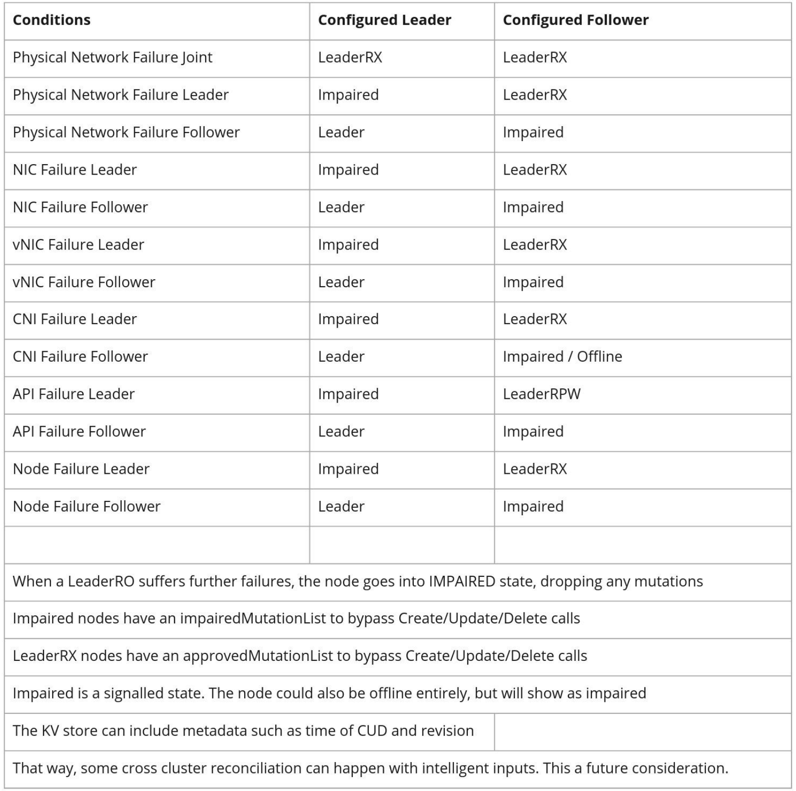
Remote Failures (Restricted Mode)

- *Assumes all local failure tests to be OK*
- API failure
- Node failure
- [an approvedMutationList will be permitted which allows creates/updates/deletions to a constrained set of resources. On a re-convergence event after split-brain, the operator will regulate the deployments]

Impaired/Read-only State

- Prevent state mutations
    - Proposal of an ***impairedMutationList*** to bypass the drop
        - Might be required but comes with a warning label!
- Create, Update, and Delete commands
- Opt-in “allowed mutation list” as configuration

Logic Condition Table



Conditions	**Configured Leader**	**Configured Follower**
Physical Network Failure Joint	LeaderRX	LeaderRX
Physical Network Failure Leader	Impaired	LeaderRX
Physical Network Failure Follower	Leader	Impaired
NIC Failure Leader	Impaired	LeaderRX
NIC Failure Follower	Leader	Impaired
vNIC Failure Leader	Impaired	LeaderRX
vNIC Failure Follower	Leader	Impaired
CNI Failure Leader	Impaired	LeaderRX
CNI Failure Follower	Leader	Impaired / Offline
API Failure Leader	Impaired	LeaderRPW
API Failure Follower	Leader	Impaired
Node Failure Leader	Impaired	LeaderRX
Node Failure Follower	Leader	Impaired
When a LeaderRO suffers further failures, the node goes into IMPAIRED state, dropping any mutations		
Impaired nodes have an impairedMutationList to bypass Create/Update/Delete calls		
LeaderRX nodes have an approvedMutationList to bypass Create/Update/Delete calls		
Impaired is a signalled state. The node could also be offline entirely, but will show as impaired		
The KV store can include metadata such as time of CUD and revision		
That way, some cross cluster reconciliation can happen with intelligent inputs. This a future consideration.		

Health Probes

East-to-West

- API HTTP probe
- ICMP Echo Request

North-to-South

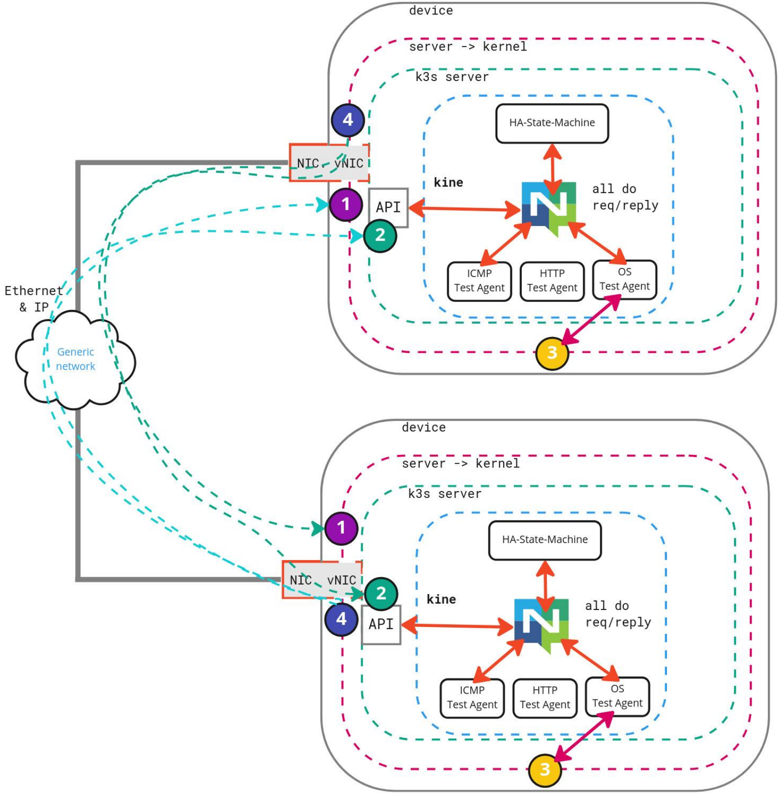
- One or more ICMP echo request test points to assert environment awareness
  - Gateway ping suggests up to first hop connectivity
  - WAN gateway/s suggests domain connectivity

Health Probes

Local (state of self)

- NIC test
  - Admin and Operational state
- Storage Check
  - Free space check
  - NOTE: Suggested for NATS KV data store backing
- CPU
  - Load
- Memory
  - Usage & Pressure

Healthcheck Schematic



Plan

Next Steps

- Finish up KINE PR
- Implement health checks within KINE
  - Test agents as Go routines (HTTP, ICMP, local CLI)
- Define “approved mutation list” and “impaired mutation list” configuration options
- Update k3s to expose relevant CLI options
- TESTING!

Thanks!