Navigating Automata

Without a map

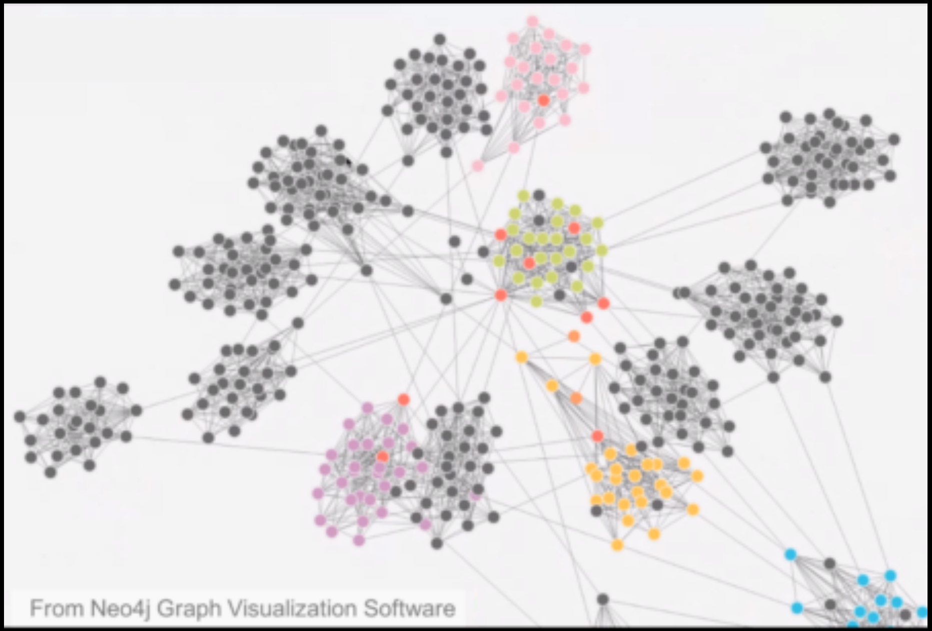
David Gee
@_ipengineer
dave.dev / ipengineer.net
me@dave.dev

What starts as frustration, will either kill you or turn into something

Networking – is a graph
Everything in networking is a graph
Graphs are everywhere in networking.
Everywhere, there are graphs in networking…I'll stop now…

Automation is ultimately a labelled transition system (graph) that allows us to model things like finite state machines.

Manifestations of such things we/I call stateful **workflows**



What starts as frustration, will either kill you or turn into something (right?)

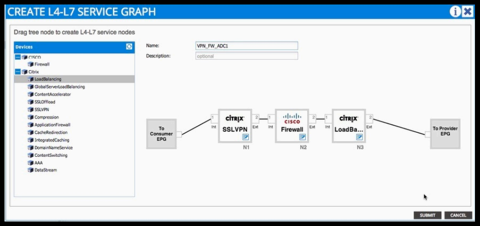
Workflows: Directed Acyclic Graphs (sorry)

Workflows are your **map** to your intended moment in time desired state.

- *End-state is never a thing; networks are without an 'end-state'*

Workflows mechanized, come in two distinct flavours:

- Productized automation like ACI and Contrail (do graph things)
- Generic workflow engine (graph engine)


CREATE L4-L7 SERVICE GRAPH

Drag tree node to create L4-L7 service nodes

Name: VPN_FW_ADC1

Description: optional

The frustration: Workflows (graphs) react to external impulses (graph) to mutate network (graph) state (graph)…**can you guess what they need?**

Where are we as an industry?

Automate the keyboard

- First generate config
- Throw 💩 at a box
- Hope for the best ¯_(ツ)_/¯

Separation of host variables from input, then render via actuation…

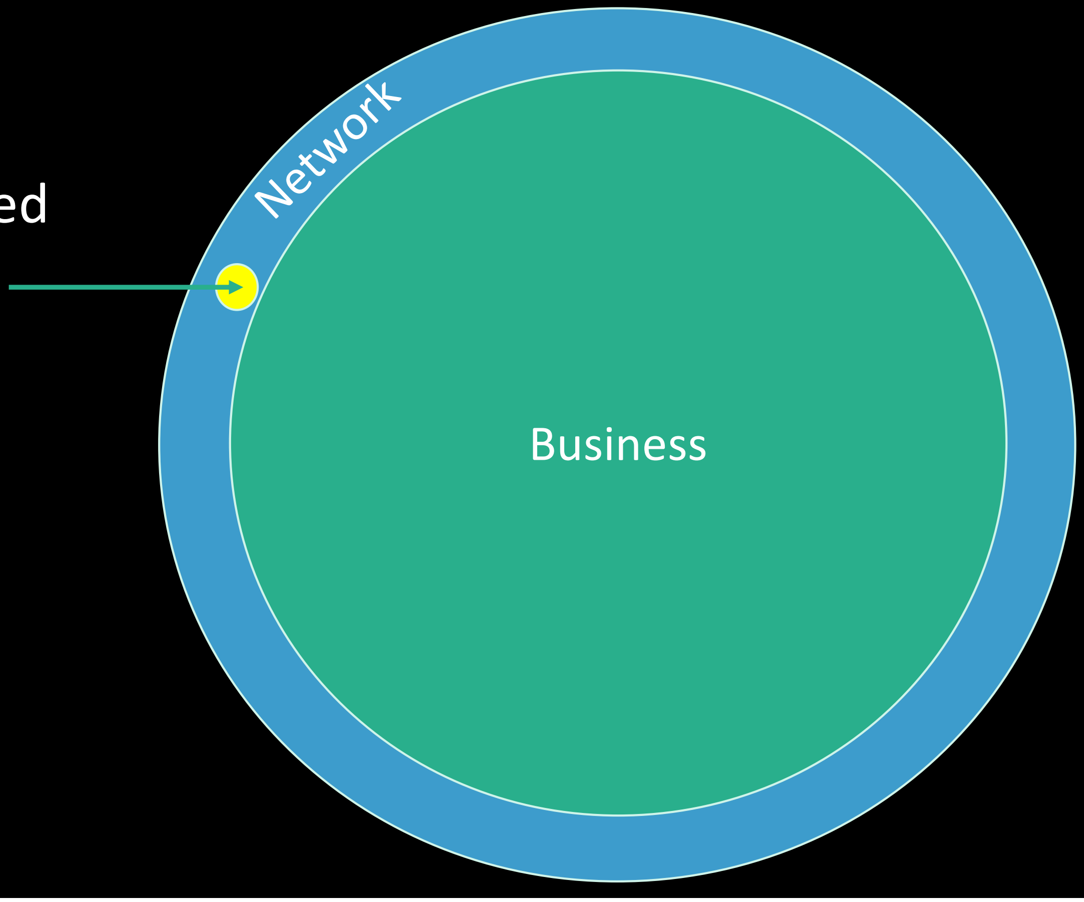
Then came unit tests

- But what if we test first for pre-state and potential collisions…like a human does?
- Then what if we check for post state to make sure the management plane is in the state we want?

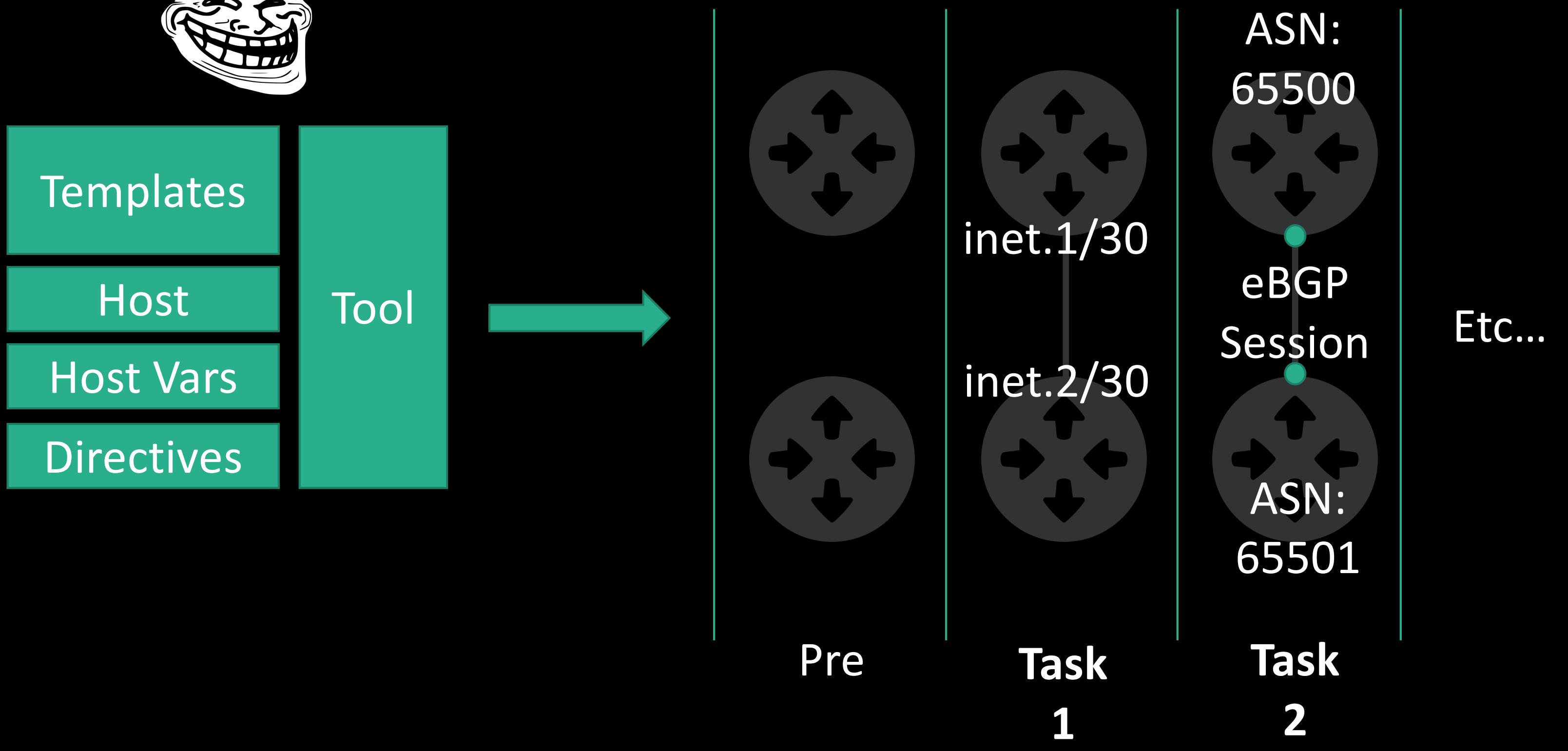
Keyword tests for cfg & ops (if you're skilled enough to use Robot)

Where are we as an industry?

Totally templated the 💩 out of that network man



Where are we as an industry?



Where are we as an industry?

Explosion of automation

Ansible, RunDeck, Salt, Puppet, Chef, StableNet, Apstra, Tungsten Fabric – all human based ☹

If we use these for networking, do you KNOW WHAT'S MISSING?

You guessed it…a b***dy graph

Where are we as an industry?

DEV ?

Productised Automata

OPS ?

Workflow Engine based or Scripts

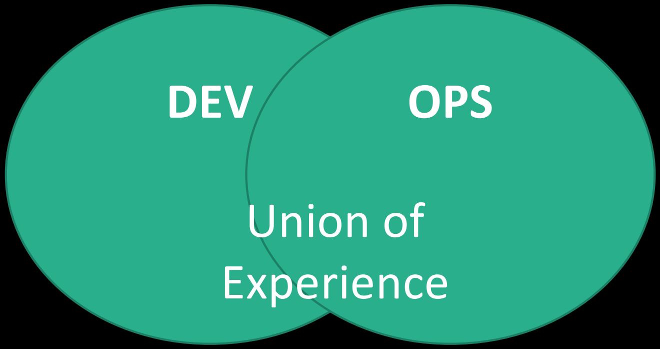
Every tool is heroic, in its own special snowflake way

The industry is still trying to make DevOps fit bottom-up to an organic graph, one that you can't treat as immutable with blast-radius 1 intent

Where are we as an industry?

%#*(**(#%*Q*&*#&*&*(#@&$

The Panacea?

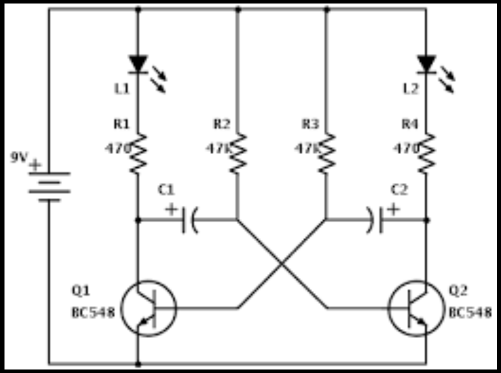


Every tool is heroic, in its own special snowflake way

The industry is still trying to make DevOps fit bottom-up (blast radius 1)

There is a blackhole in the middle of every automation system sucking the life out of possibility (and my soul)

Where are we as an industry?



Network Control

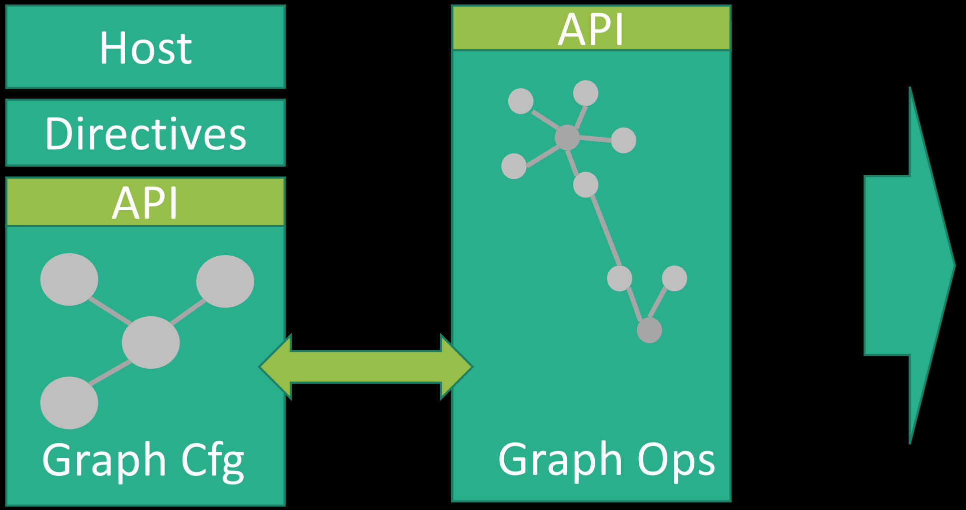
>>>

We shall use a tool.

The kingdom shall rejoice and pay many taxes.

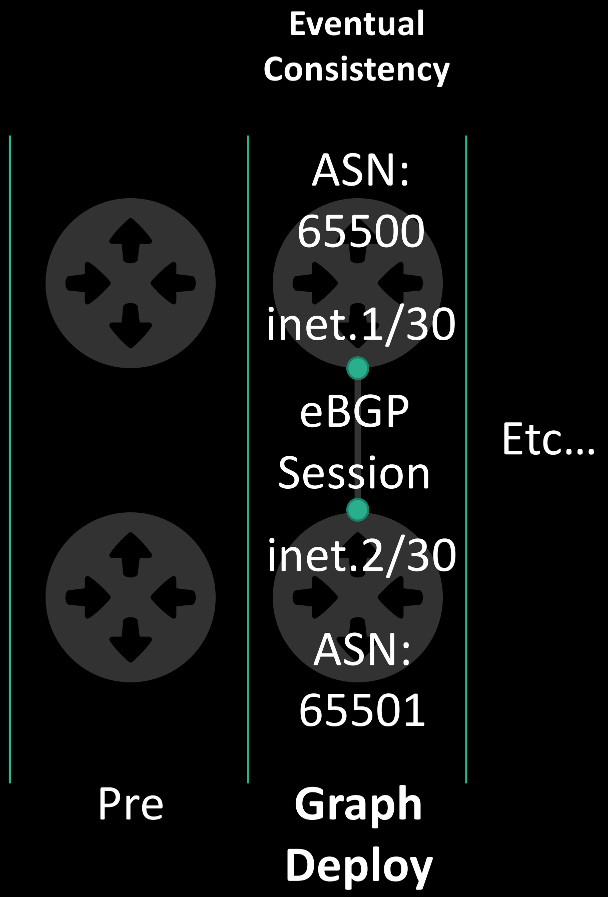
- We don’t design, nor model and rarely verify
- We don’t have a formal language (even structured programming ~70yrs old now)
- Python/Ansible blinding the industry
- Automata is more complex than if-else statements (it is, honest governor)

Where could we be?



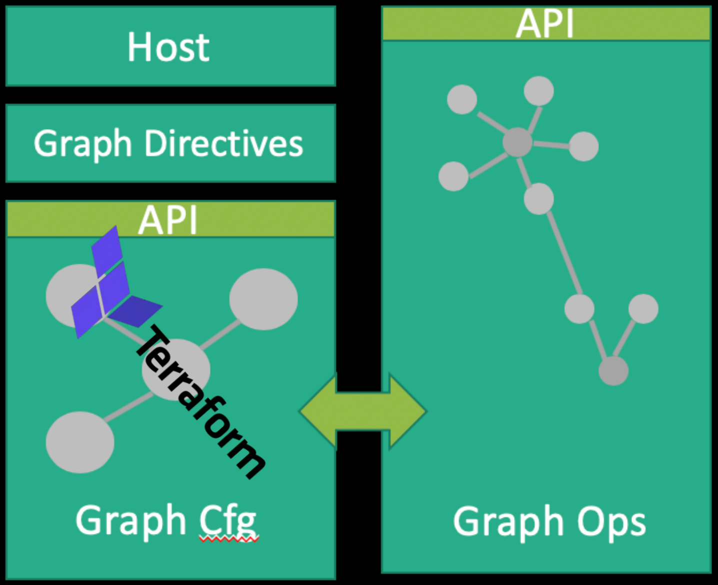
Declarative
Ordered
Atomic

Physical / Cabling / Logical
Constant Discovery
Absolute Knowledge
Handles Dynamism
DNA System



What I’m trying to do about it

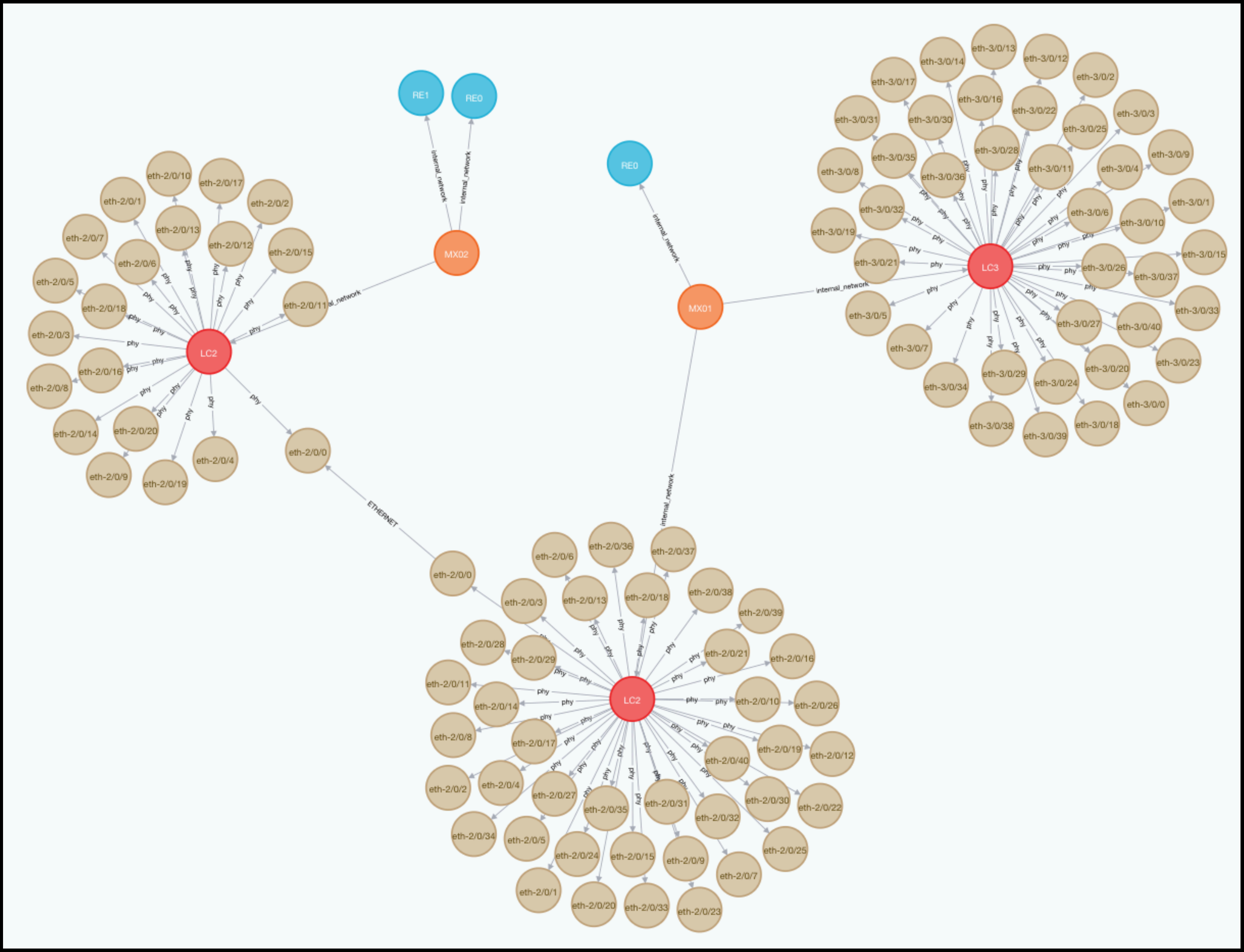
1. **Release Terraform Providers Network Operating Systems (NOS) that don’t just pass CLI**
2. **Terraform has a graph engine for configuration changes, that deals with CRUD (properly)**
3. **Build or open source a graph ops engine**
4. **You can extract schemas from TF, great for m2m**

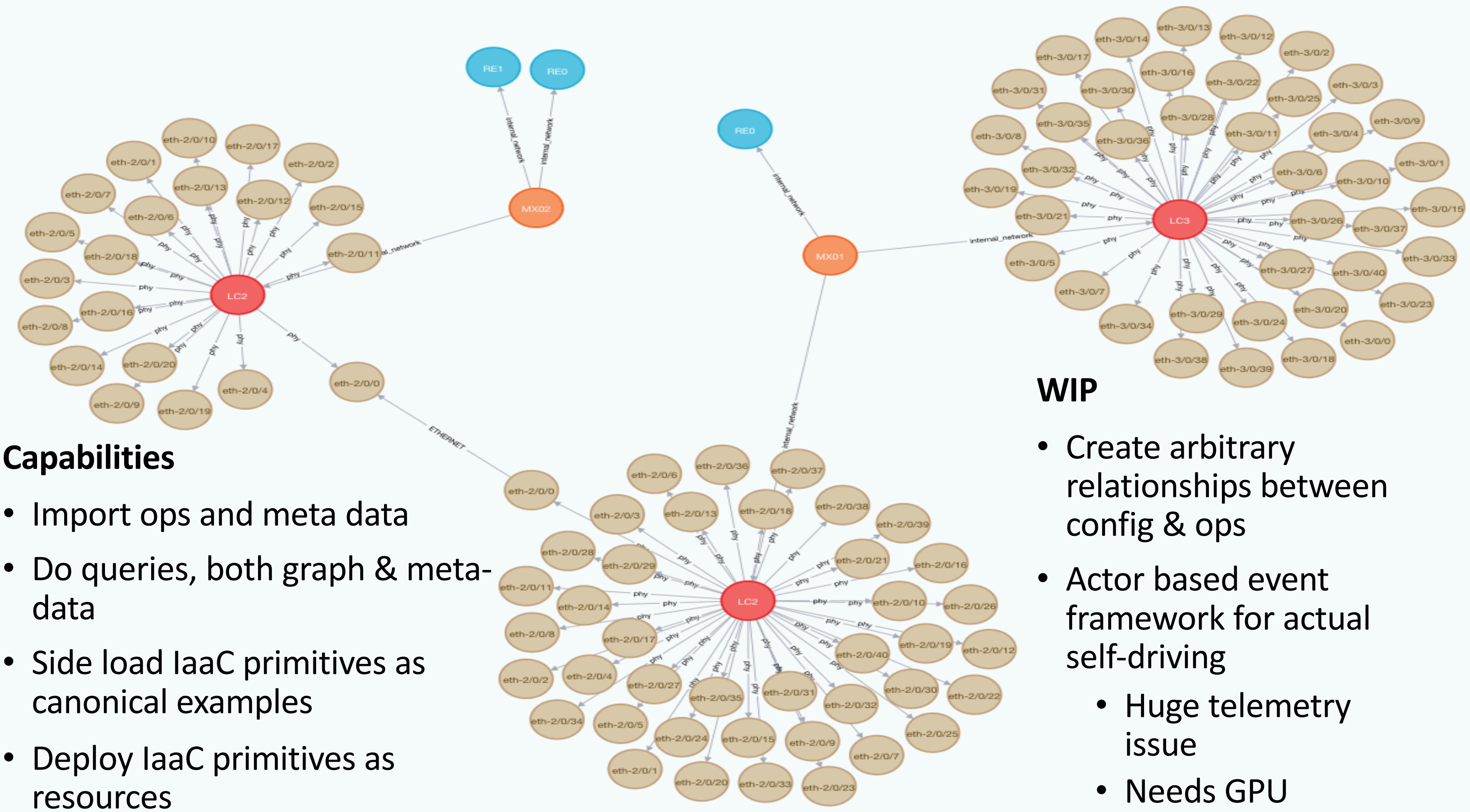


Early experimental work but showing promise.

Built on Neo4j, with 99% of code in Go, 1% Python, GPB and NATS to glue things together.

Lots of known unknowns yet to solve but progressing.





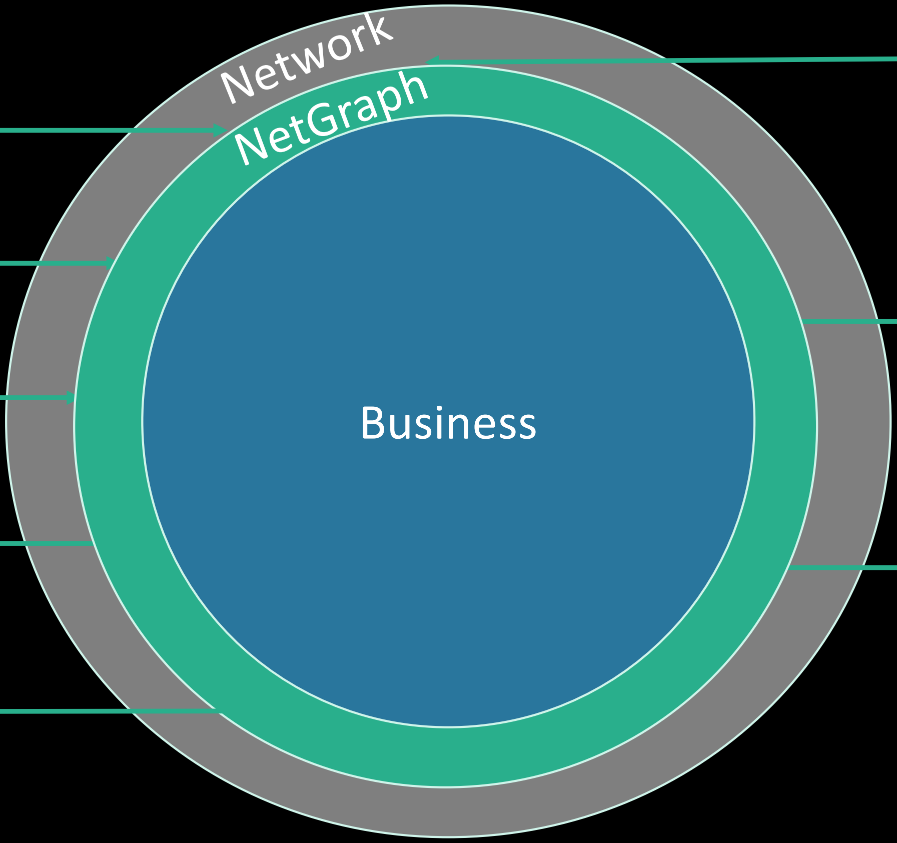
Capabilities

- Import ops and meta data
- Do queries, both graph & meta-data
- Side load IaaC primitives as canonical examples
- Deploy IaaC primitives as resources

WIP

- Create arbitrary relationships between config & ops
- Actor based event framework for actual self-driving
  - Huge telemetry issue
  - Needs GPU

Where we could be….



Fed by protocols like BMP or emerging telemetry

Template-less

Machine friendly data

Remove CLI graph checks entirely

Eyes wide open

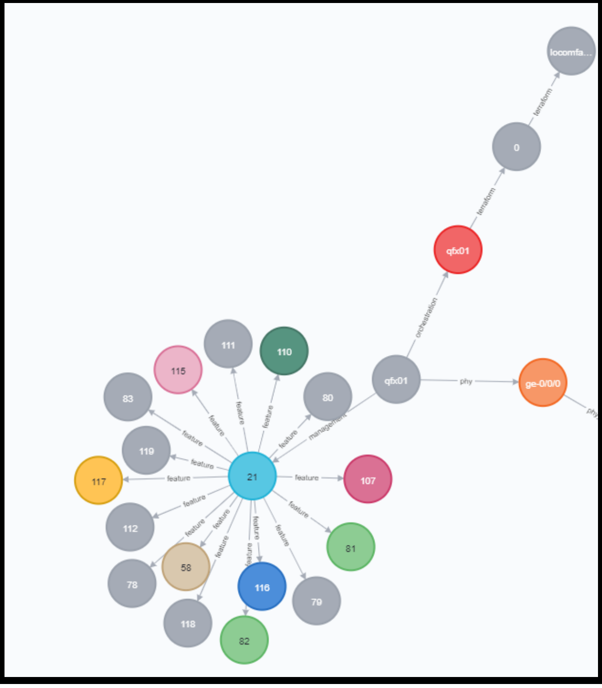
Tight control loop between what we want and what we have

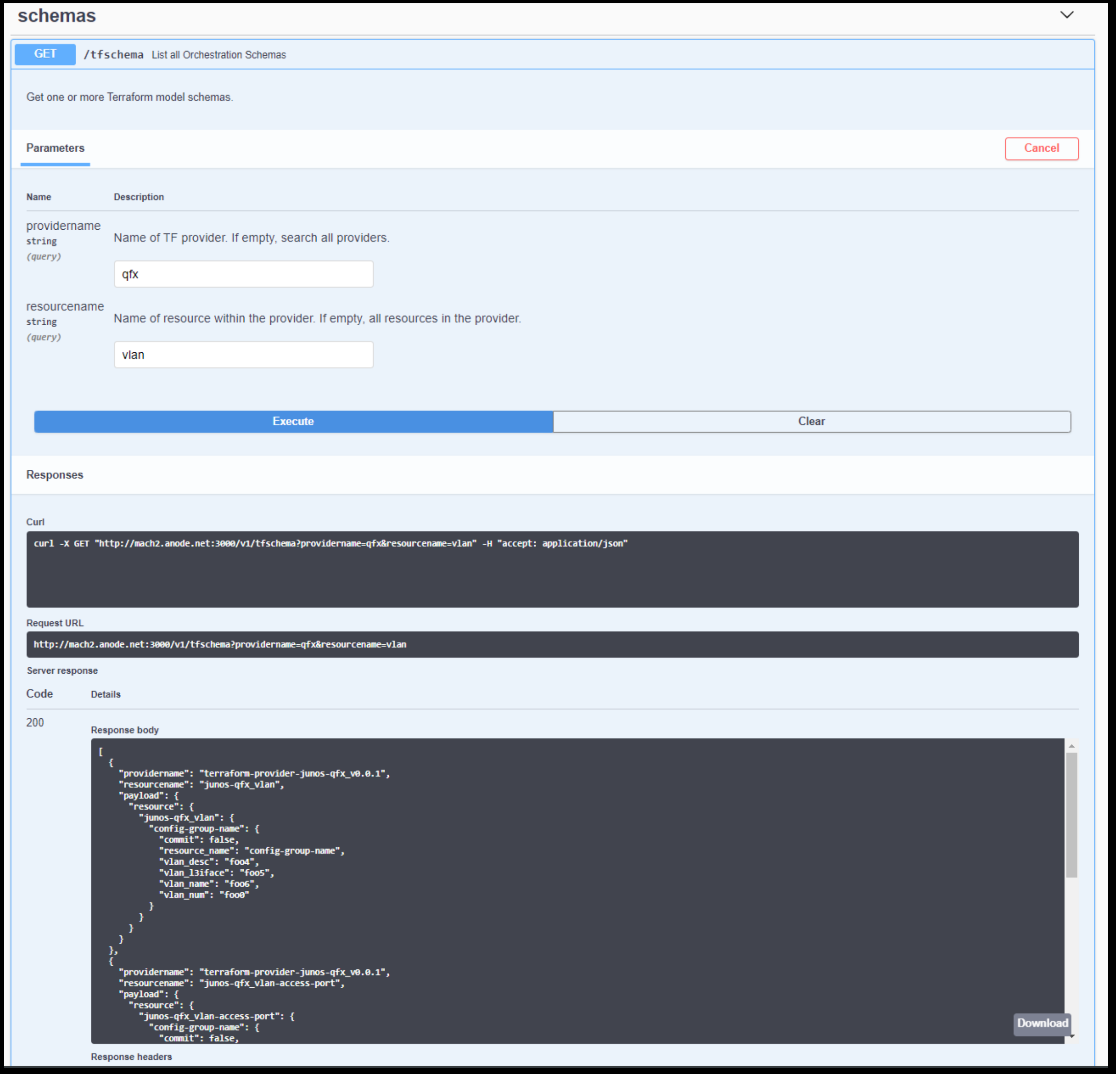
A means to do "event driven" and "self-driving" by actor based graph queries

Services can be rendered with automatic checks of constraints like placement rules

Machine Interaction of IaaC

- Software reads machine friendly schemas from IaaC and creates canonical examples of use
- Once invoked or posted, the appear on the graph as desired state





schemas

GET `/tfschema` List all Orchestration Schemas

Get one or more Terraform model schemas.

Parameters

Name	Description
providername string (query)	Name of TF provider. If empty, search all providers. qfx
resourcename string (query)	Name of resource within the provider. If empty, all resources in the provider. vlan

Responses

Curl

```
curl -X GET "http://mach2.anode.net:3000/v1/tfschema?providername=qfx&resourcename=vlan" -H "accept: application/json"
```

Request URL

```
http://mach2.anode.net:3000/v1/tfschema?providername=qfx&resourcename=vlan
```

Server response

Code Details

200

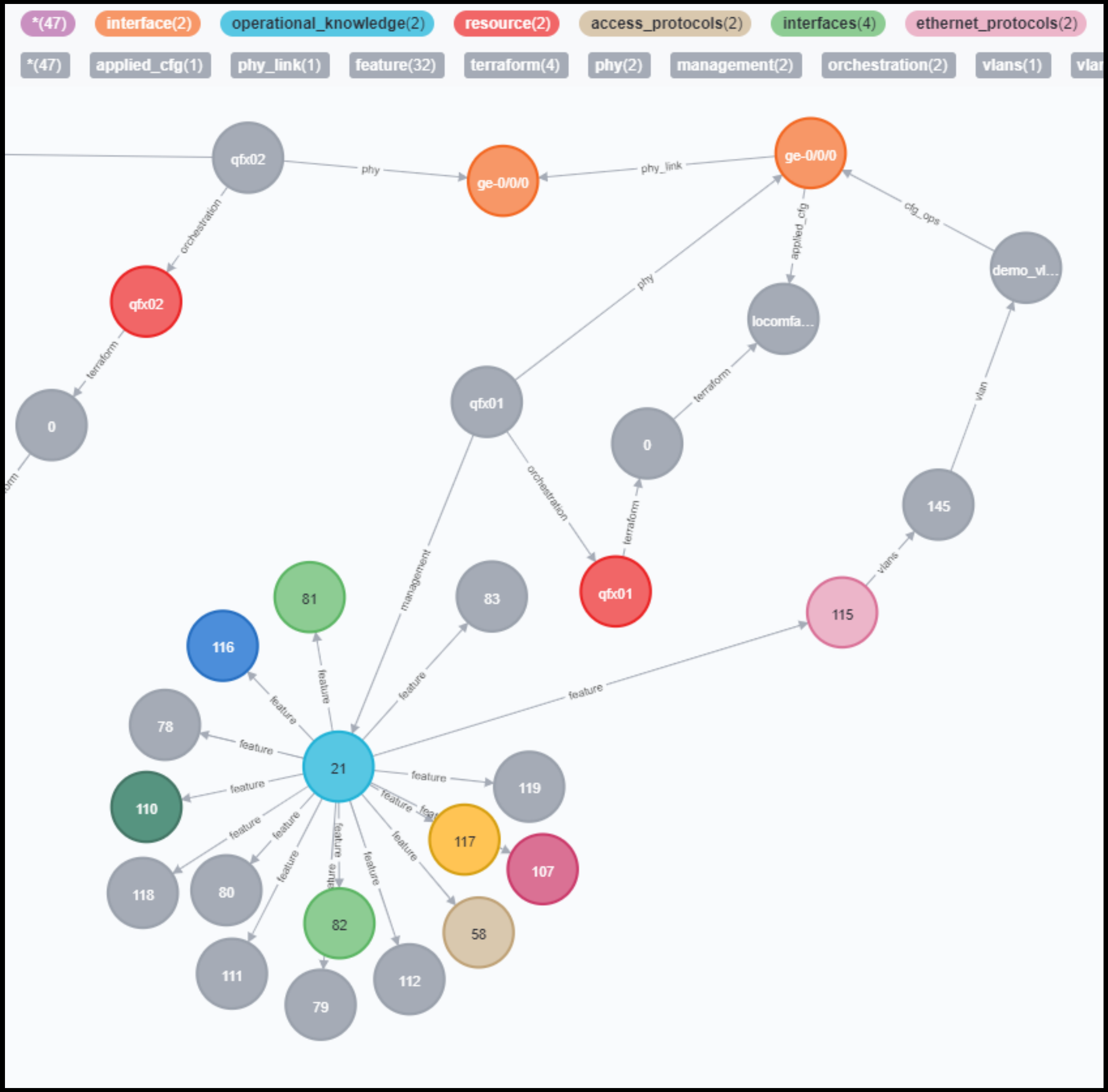
Response body

```
[
  {
    "providername": "terraform-provider-junos-qfx_v0.0.1",
    "resourcename": "junos-qfx_vlan",
    "payload": {
      "resource": {
        "junos-qfx_vlan": {
          "config-group-name": {
            "commit": false,
            "resource_name": "config-group-name",
            "vlan_desc": "foo4",
            "vlan_l3iface": "foo5",
            "vlan_name": "foo6",
            "vlan_num": "foo0"
          }
        }
      }
    }
  },
  {
    "providername": "terraform-provider-junos-qfx_v0.0.1",
    "resourcename": "junos-qfx_vlan-access-port",
    "payload": {
      "resource": {
        "junos-qfx_vlan-access-port": {
          "config-group-name": {
            "commit": false,
```

Response headers

Linkage Between Ops & Config

- Graph can get messy…first and foremost
- Graph views derived by queries (so don't panic!)
- When viewing everything, it's easy to see how config maps to operational info, like VLAN mapping to an interface (cliché, yes)
- We also see config anchored to a resources node from IaaC
- Possible to see operational data relationship to config data and config data to physical interfaces (useful for troubleshooting, validation etc)



Thank you!

David Gee

@_ipengineer

dave.dev / ipengineer.net

me@dave.dev