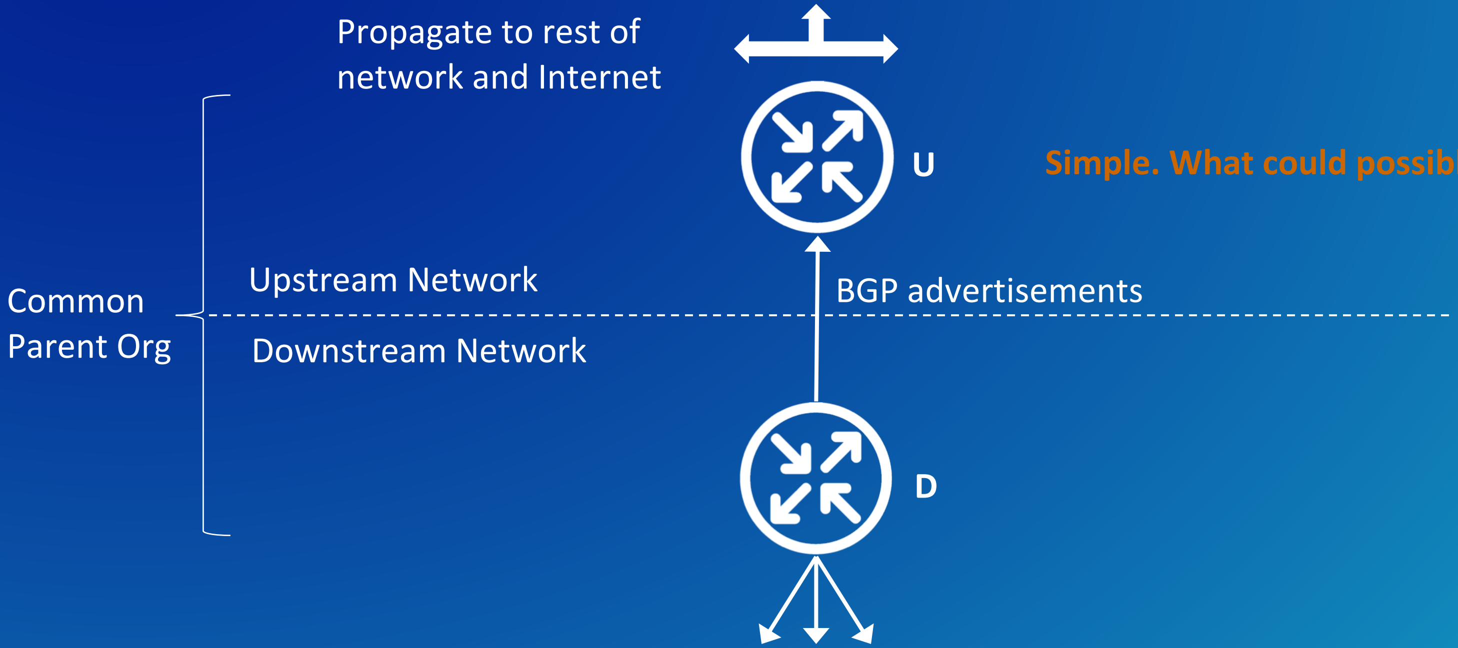
Query Your Network Like a Database

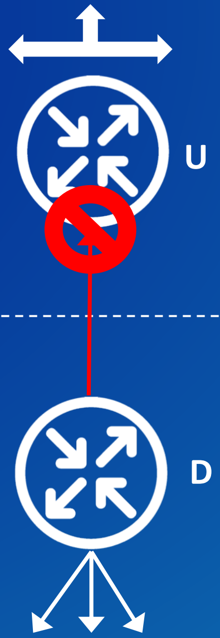
Andreas Voellmy, PhD
Forward Networks
Andreasvoellmy@forwardnetworks.com
@AndreasVoellmy

Once Upon a Time, a Service Provider Came To Us …



Simple. What could possibly go wrong?

BGP Import Policy Filters



Upstream

Downstream

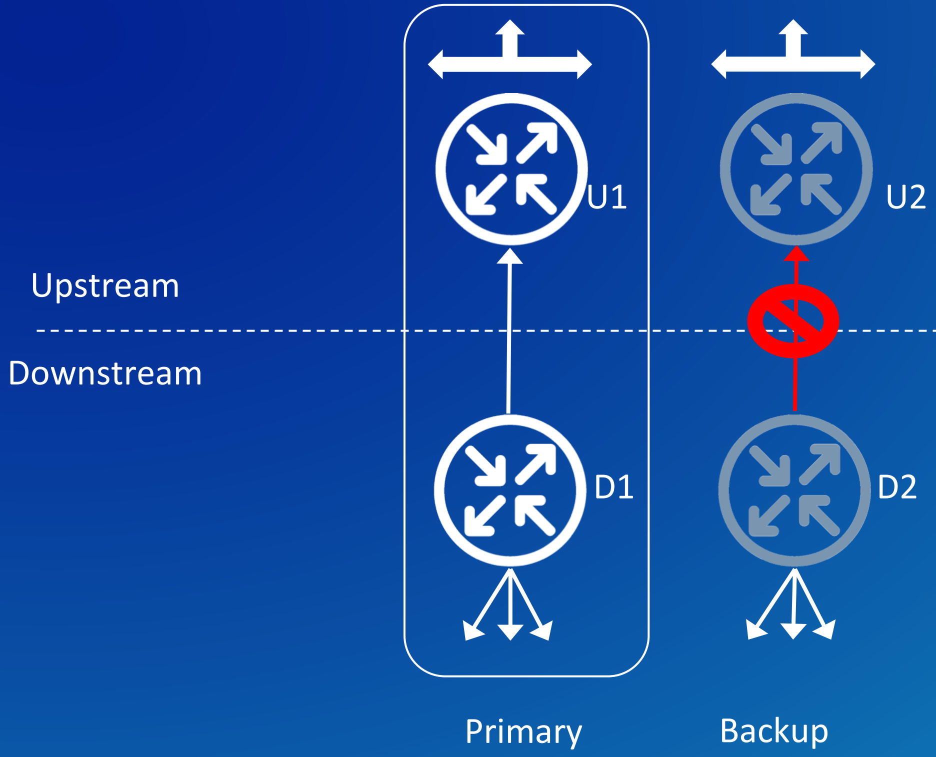
Import policy:

- filter if community X present
- filter if AS path matches some pattern
- routes introduced by other protocols

Policy change

- starts using community X
- starts prepending and triggers pattern
- starts announcing new route

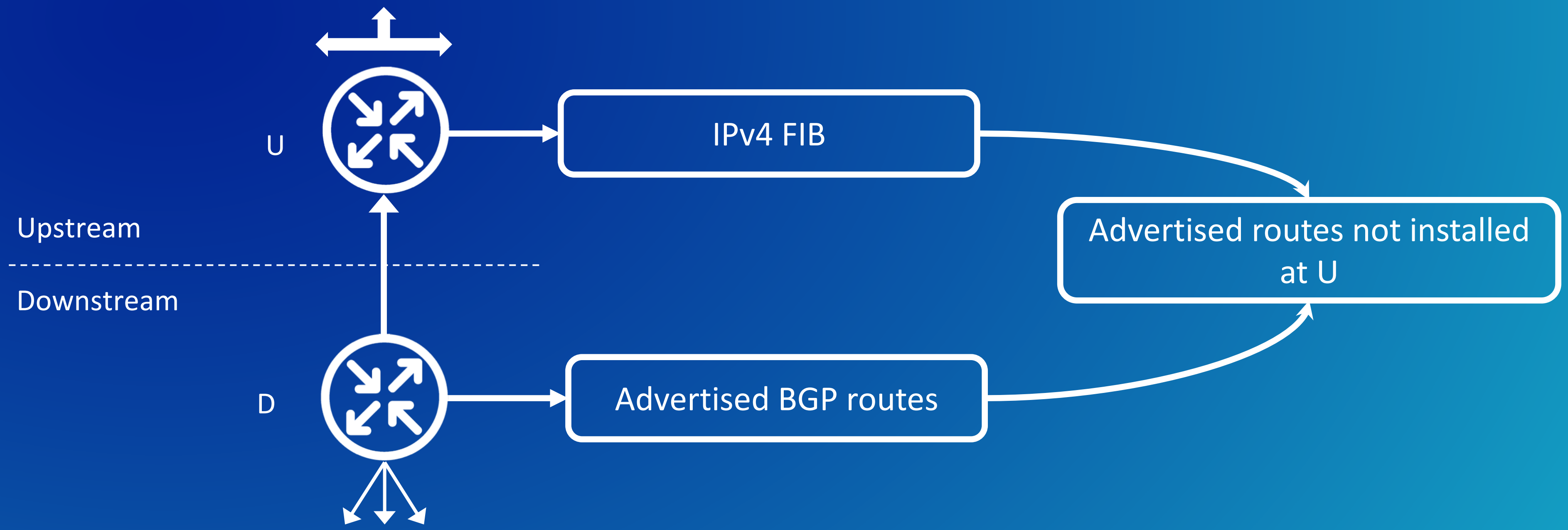
Policy Problem May Be Latent



Problem only becomes apparent when failover occurs

SLAs mean this is an expensive outage

Plan: Proactively Detect The Problem



Simple, Important, but Hard to Answer

- Hard to answer these questions on a large (10K+ devices), heterogeneous (100s of vendor/OS combinations) network.
- NetConf and other APIs are not widely supported on today's networks.
- The only workable solution for network operators today:
  - SSH and grab text
  - Parse poorly-documented, unstructured outputs
  - Organize the data set …

Example: Get interface status on two devices

Cisco NX-OS: two commands needed

```
interface Ethernet1/3
 shutdown
 switchport mode private-vlan host
 switchport access vlan 50
 speed 1000
 switchport private-vlan host-association 50 2000

interface Ethernet1/4
 switchport mode private-vlan trunk secondary
 speed 1000
 no shutdown
 switchport private-vlan trunk native vlan 2000
 switchport private-vlan trunk allowed vlan 1000,2000
 switchport private-vlan association trunk 50 1000
```

```
Ethernet        VLAN    Type Mode    Status Reason                   Speed    Port
Interface                                                                     Ch #
--------------------------------------------------------------------------------------
Eth1/1          1       eth  access  up     none                     1000(D)  --
Eth1/2          1       eth  access  up     none                     1000(D)  --
Eth1/3          1       eth  access  down   Administratively down    auto(D)  --
Eth1/4          1       eth  access  down   Administratively down    auto(D)  --
```

A10: one command

```
Ethernet 10 is up, line protocol is up
Hardware is 10Gig, Address is 001f.a011.8dde
Member of L2 Vlan 601, Port is Tagged
Flow Control is disabled, IP MTU is 9216 bytes
Member of trunk group 1

Trunk ID           : 1    Member Count: 2
Trunk Name         : None
Trunk Status       : Up
Trunk Type         : Dynamic (LACP)
Admin Key          : 1001
Members            : 9    10
Cfg Status         : Enb Enb
Oper Status        : Up   Up
Ports-Threshold    : None
Working Lead       : 9

Trunk ID           : 16         Member Count: 2
Trunk Name         : None
Trunk Status       : Up
Trunk Type         : Dynamic (LACP)
Admin Key          : 1016
Members            : 1    2
Cfg Status         : Enb Enb
Oper Status        : Up   Up
Ports-Threshold    : None
Working Lead       : 2
```

Different commands to run and formats to parse, even for the most basic data.

Simple, Important, but Hard to Answer

- *"interface status, BGP session, … we could deploy a person for 6 months to do this…. 80% of the effort is collecting and parsing..."*
- The work would be duplicative
- … but in fact mostly does not get done: operators are mostly not programmers and are otherwise busy fixing stuff.

Let’s Rethink This

What if we had a database of network information, and we could just query it?

SP Network Query

"Are there any BGP routes advertised by my downstream BGP routers that are not installed in their upstream router's FIB?"

Many Other Examples

- Ad-hoc questions:
  - *Where have we defined VLAN 100?*
- Desired invariants:
  - *Do all my connected interfaces use the same MTU?*
- Bad states:
  - *Are any of my expected BGP sessions in a bad state?*

Forward Network Query Engine (NQE)

Forward provides **access to structured, normalized data** about the network, so that users can query their network like a database.

NQE: Query Your Network Like a Database!

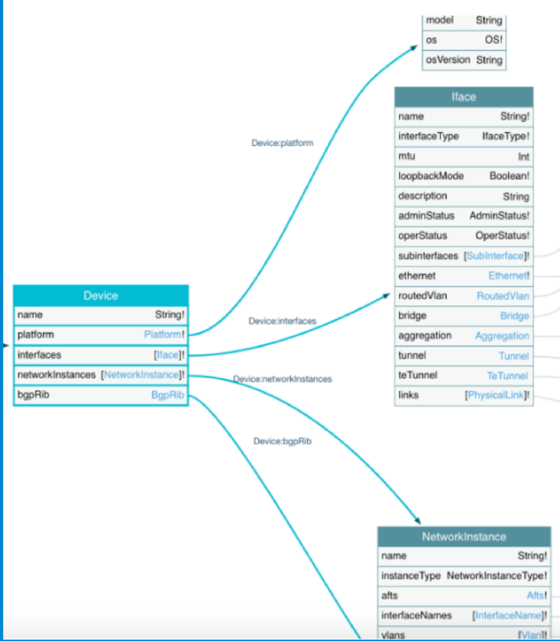
Vendor-specific configuration files and state information

```
Physical interface: ge-0/0/0, Enabled, Physical link is Up
  Interface index: 139, SNMP ifIndex: 513
  Link-level type: Ethernet, MTU: 1514, MRU: 1522, Speed: 1000
  Pad to minimum frame size: Disabled
  Device flags   : Present Running
  Interface flags: SNMP-Traps Internal: 0x4000
  Link flags     : None
  CoS queues     : 8 supported, 8 maximum usable queues
  Current address: 00:15:00:00:00:04, Hardware address: 00:15:
  Last flapped   : 2017-08-18 00:53:01 UTC (1w6d 22:04 ago)
  Input rate     : 1456 bps (1 pps)
  Output rate    : 224 bps (0 pps)
  Active alarms  : None
  Active defects : None
  Interface transmit statistics: Disabled

  Logical interface ge-0/0/0.0 (Index 329) (SNMP ifIndex 525)
    Flags: Up SNMP-Traps 0x4004000 Encapsulation: ENET2
    Input packets : 59606
    Output packets: 279170
    Protocol inet, MTU: 1500
      Flags: Sendbcast-pkt-to-re, Is-Primary
      Addresses, Flags: Is-Preferred Is-Primary
        Destination: 10.100.0.78/31, Local: 10.100.0.78
    Protocol iso, MTU: 1497
      Flags: Is-Primary
    Protocol multiservice, MTU: Unlimited
      Flags: Is-Primary

Physical interface: lc-0/0/0, Enabled, Physical link is Up
  Interface index: 136, SNMP ifIndex: 507
  Speed: 800mbps
  Device flags   : Present Running
  Link flags     : None
  Last flapped   : Never
    Input packets : 0
    Output packets: 0
```

Structured, normalized schema based on Forward data model



Possible query of network schema and results

Sample query (concept):

"Which devices have interfaces with different operational and admin states?"

Formatted results:

Device	Interface	Admin Status	Oper Status
atl-lb01	1.3	UP	DOWN
atl-lb01	1.4	UP	DOWN
sjc-ce01	ge-0/0/1	UP	DOWN
sjc-ce02	ge-0/0/5	UP	DOWN

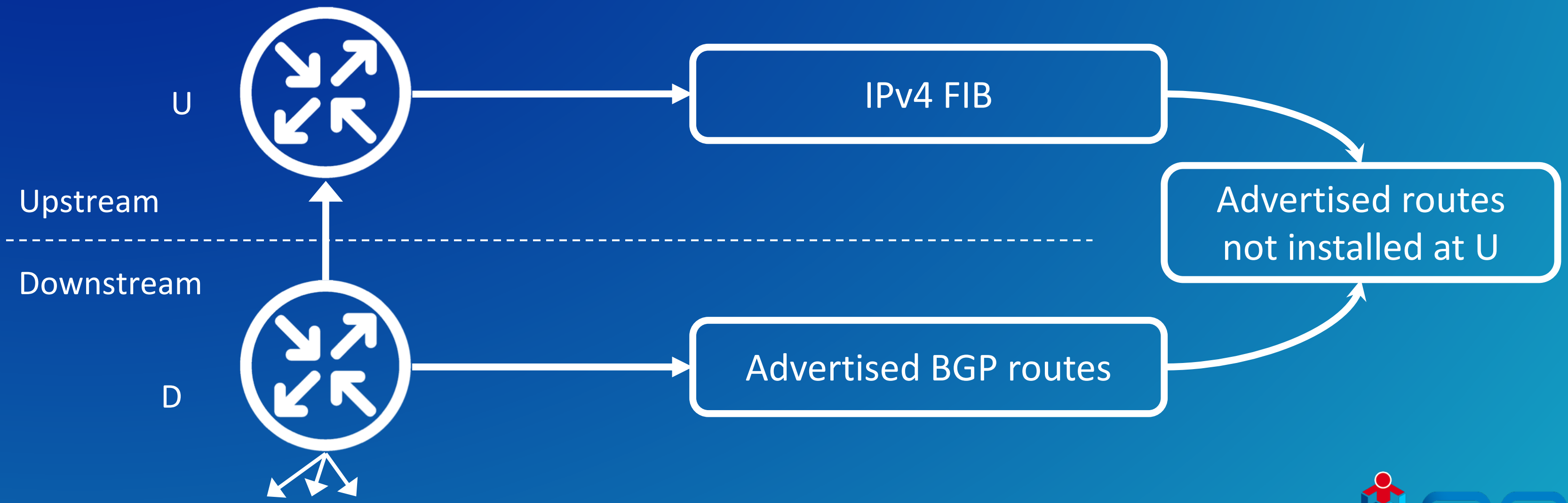
NQE: Single query works on all supported devices

Vendors

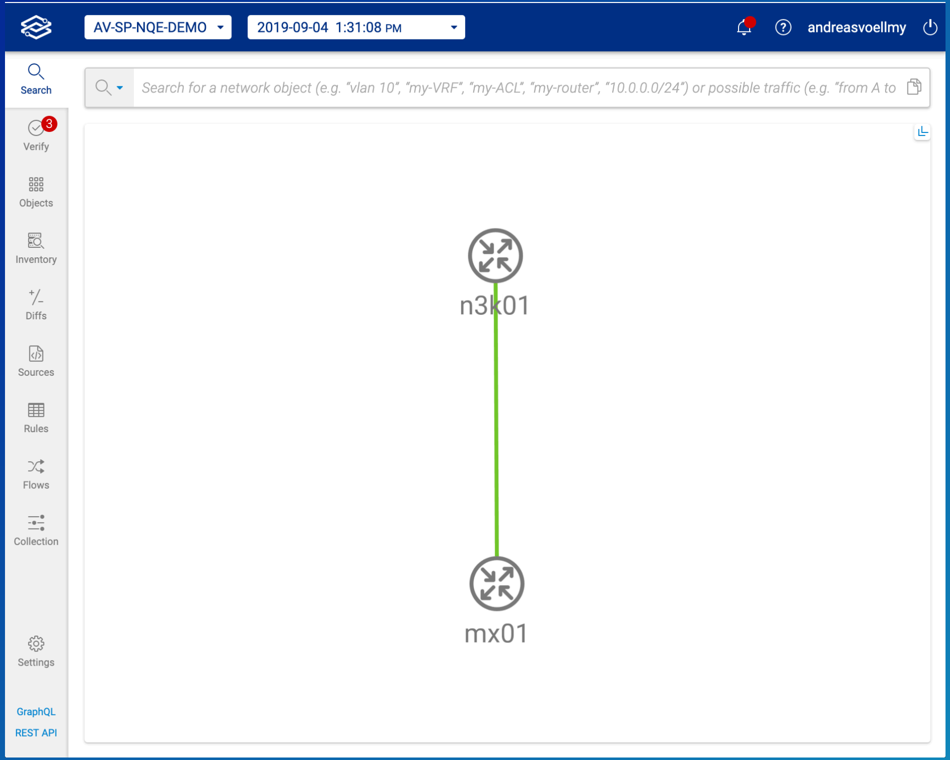
A10	ARISTA	AVI Networks	aws	Symantec. BLUE COAT	Check Point SOFTWARE TECHNOLOGIES LTD	CISCO	CITRIX
CUMULUS	f5	FORTINET	Hewlett Packard Enterprise	JUNIPER NETWORKS	paloalto NETWORKS	PICA8	vmware

NQE Walkthrough: SP Use Case

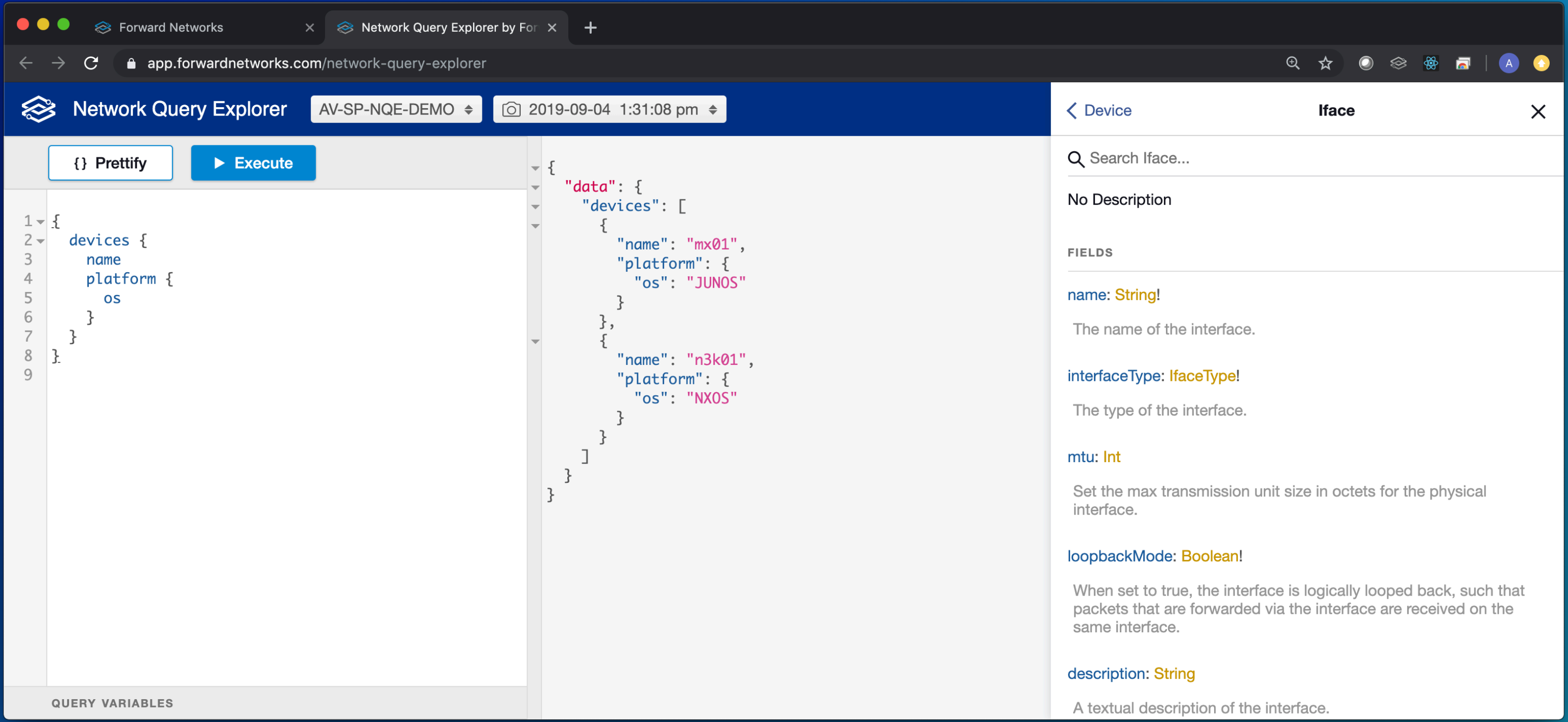
"Are there any BGP routes advertised by my downstream BGP routers that are not installed in their upstream router's FIB?"



NQE Walkthrough: Forward UI



NQE Walkthrough: Query Editor

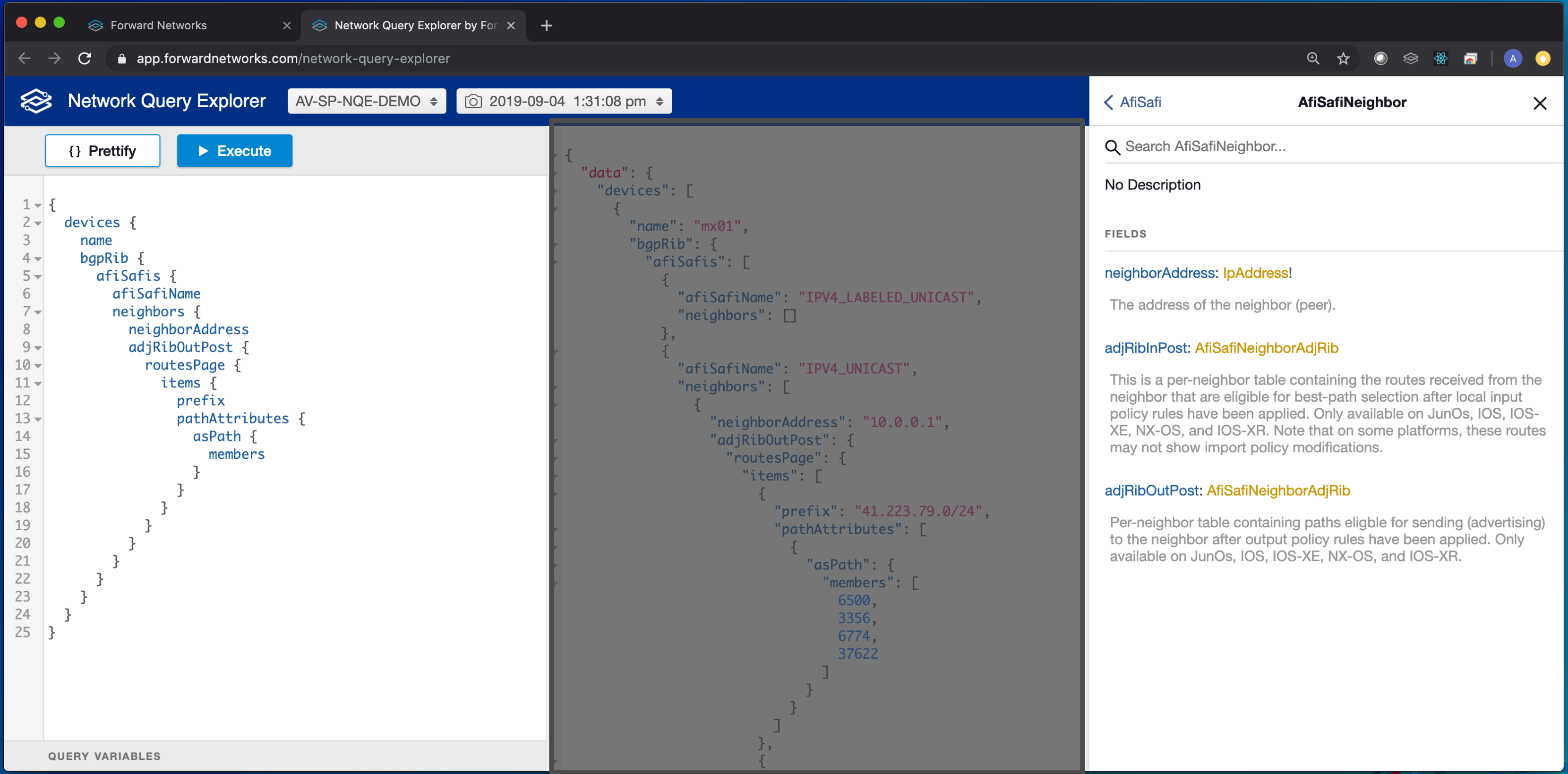


Editor

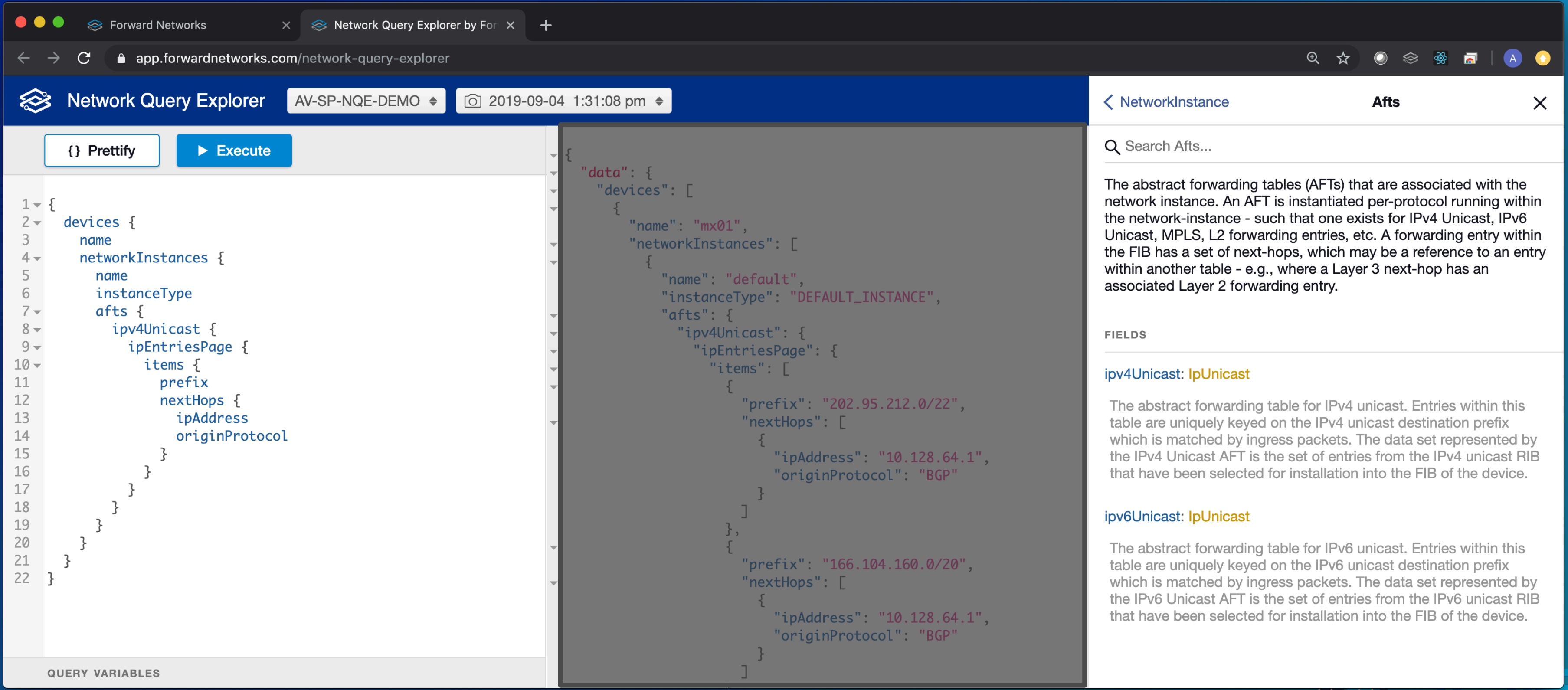
Results

Schema Docs

NQE Walkthrough: BGP advertised routes



NQE Walkthrough: Get IPv4 Routes



NQE Walkthrough: Query Script

+ Simple, small script:
  + *Runs both queries, compares routes, prints violations.*

Found the following violations:

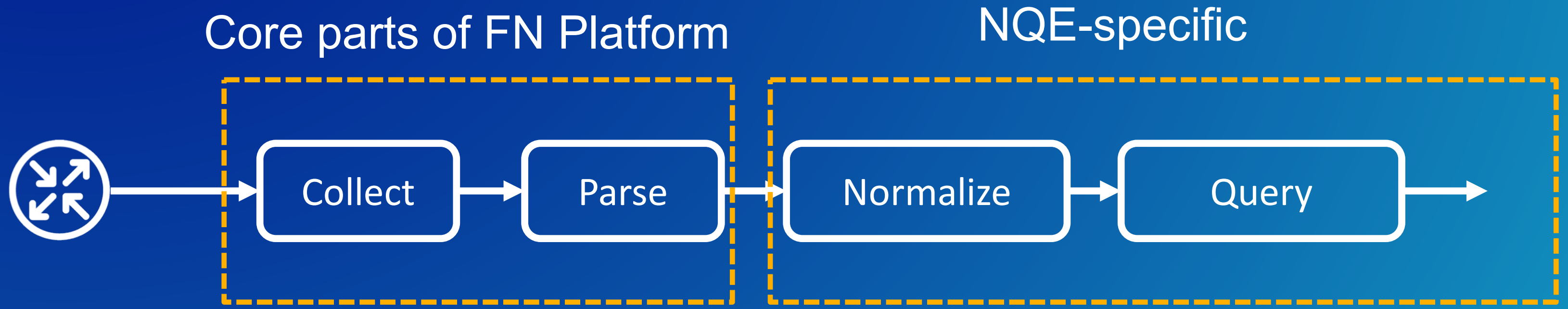
Prefix	AS Path	Diagnosis
192.121.121.0/24	[1, 1, 1, 1, 6500, 3356, 42708, 30893]	Filtered by upstream BGP import policy
31.168.160.0/20	[6500, 3356, 8551, 8551, 8551, 8551]	Upstream selects different route: STATIC DROP
204.235.115.0/24	[1, 1, 1, 1, 6500, 3356, 4323, 3456, 3456, 3456, 3456, 3456, 3456]	Filtered by upstream BGP import policy
216.206.127.0/24	[1, 1, 1, 1, 6500, 3356, 54114]	Filtered by upstream BGP import policy
37.46.200.0/21	[6500, 3356, 39326]	Upstream selects different route: STATIC DROP
202.46.240.0/22	[1, 1, 1, 1, 6500, 3356, 2914, 58463, 18059, 3583]	Filtered by upstream BGP import policy
23.200.16.0/20	[6500, 3356, 2914]	Upstream selects different route: STATIC DROP
203.13.35.0/24	[1, 1, 1, 1, 6500, 3356, 4637, 1221, 38285, 10113]	Filtered by upstream BGP import policy
211.118.176.0/24	[1, 1, 1, 1, 6500, 3356, 3491, 9848, 18305, 18305, 18305]	Filtered by upstream BGP import policy
192.251.17.0/24	[1, 1, 1, 1, 6500, 3356, 22773]	Filtered by upstream BGP import policy
23.252.160.0/21	[6500, 3356, 4134, 36678, 26484]	Upstream selects different route: STATIC DROP
202.95.212.0/22	[1, 1, 1, 1, 6500, 3356, 2516, 10021, 10021]	Filtered by upstream BGP import policy
198.178.192.0/24	[1, 1, 1, 1, 6500, 3356, 701, 702]	Filtered by upstream BGP import policy
24.142.176.0/24	[6500, 3356, 19009, 53432]	Upstream selects different route: STATIC DROP
216.57.121.0/24	[1, 1, 1, 1, 6500, 3356, 5738, 26082]	Filtered by upstream BGP import policy
198.136.250.0/24	[1, 1, 1, 1, 6500, 3356, 7018, 2386]	Filtered by upstream BGP import policy

16 rows

Time to implement: 6 months → 1 hour

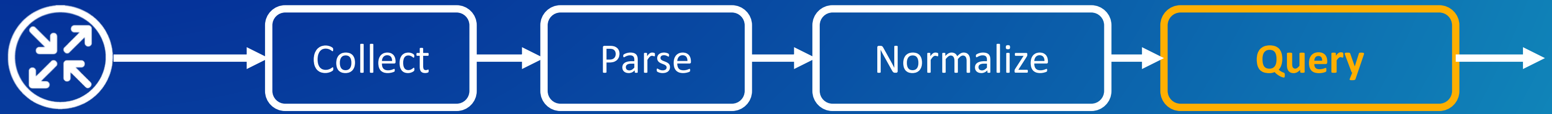
How to Implement a Normalized Network Database?

In theory, this is simple



In practice: challenges in every step of the process.

How to Implement a Normalized Network Database?



Query API

- Operators are not professional programmers; we wanted a query API that was easy to use and required minimal learning.
- While other choices may also have worked, GraphQL was a great fit.
- "Query language for your API"

Describe your data

```
type Project {
  name: String
  tagline: String
  contributors: [User]
}
```

Ask for what you want

```
{
  project(name: "GraphQL") {
    tagline
  }
}
```

Get predictable results

```
{
  "project": {
    "tagline": "A query language for APIs"
  }
}
```

GraphQL: Schemas

- Network data model is elaborate; users need clear definitions and help navigating this.

- GraphQL schema language enables us to describe the model simply and clearly.

```
type Ethernet {
   # MAC Address of the Ethernet interface
   macAddress: MACAddress
   # The duplex mode that has been negotiated.
   negotiatedDuplexMode: DuplexMode
   # The interface speed that has been negotiated.
   negotiatedPortSpeed: PortSpeed
   # The logical aggregate interface to which this interface belongs.
   aggregateId: String
   # MAC Address of the Ethernet interface
    switchedVlan: SwitchedVlan
}
```

- Great tooling around the schema.

GraphQL: Easy to Query

+ Queries are simple: they just follow the data organization

```
{
  devices {
    interfaces {
      name
      operStatus
      adminStatus
    }
  }
}
```

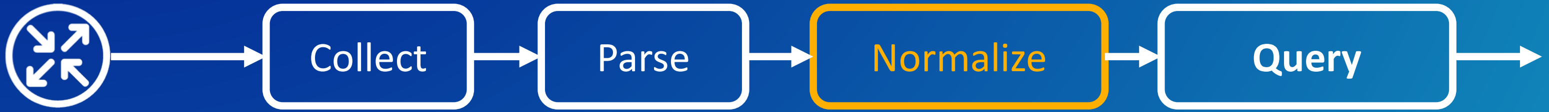
+ Output is JSON and follows the data organization, with values filled in.

```
{
  “data”: {
    “devices”: [
      {
        “name”: “gi0/0/0/0”,
        “operStatus”: “UP”,
        “adminStatus”: “UP”
      },
      {
        “name”: “gi0/0/0/1”,
        “operStatus”: “UP”,
        “adminStatus”: “DOWN”
      }, … ]
  }
}
```

GraphQL: Easy to Implement

- The largest networks present large datasets:
  - 2M+ routes on a single device
  - 600K+ ACLs on a single device
- To handle this, Forward implements custom storage formats and data structures.
- GraphQL is agnostic to storage format; allows us to implement queries with custom logic.

How to Implement a Normalized Network Database?



Normalization: How to Organize Data?

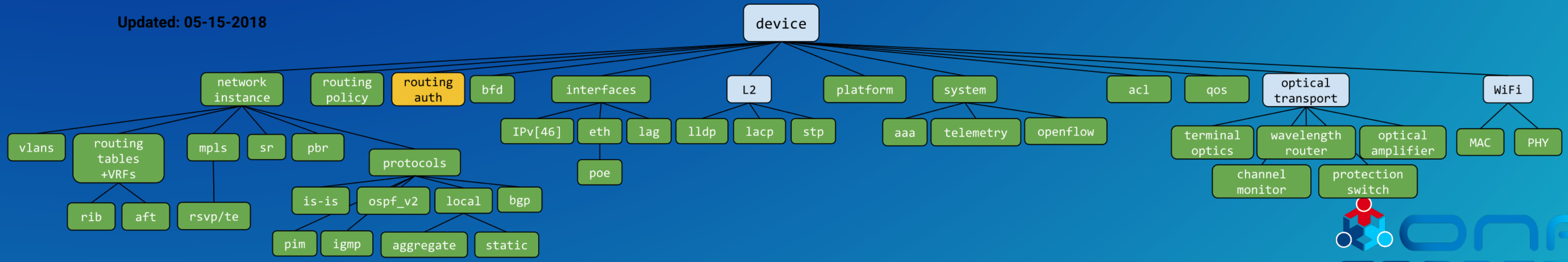
+ **We are not interested in re-inventing the wheel here.**

+ We based our schemas on OpenConfig YANG models.

OPENCONFIG

Vendor-neutral, model-driven network management designed by users

+ Operator-driven community, with operator-vetted models, with broad coverage.

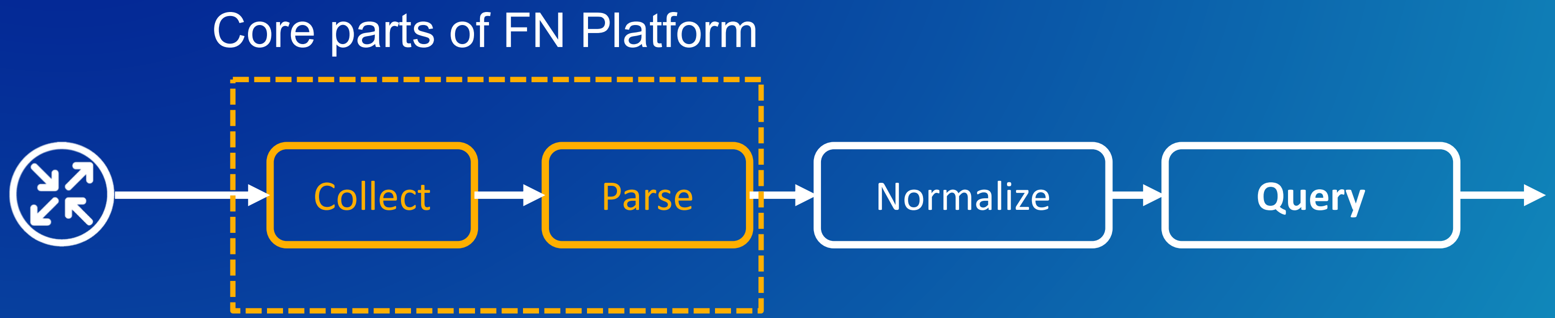


Marrying OpenConfig with GraphQL

There are some mismatches

- OpenConfig vs GraphQL naming requirements.
- Simplified for read-only use case.
- Leverage GraphQL's graph database facilities to enable easier linking between objects.
- Expose paging over large collections.

How to implement a normalized network database?



Parsing: Millions of Patterns

Scale: 16 vendors, 23 Oses, 242 OS versions.

Example: On just a single device OS (Cisco NX-OS), there are 120k ways of combining keywords into valid top-level config commands.

Critical: streamlined way of ingesting text-based data into the model.

One of the major focus areas at Forward.

CONNECT

Collection: Getting the Data in the Real World

All sorts of surprising challenges lurk here.

+ No inventory, no topology
+ Complex infrastructure slows down collection
+ Device failures are common

The Road Ahead

NQE announced in January this year.

We continue to evolve and improve:

+ Continue to expand the data set

+ Explore ways to simplify and make it easier to query without dropping into scripting.

Thank You

andreasvoellmy@forwardnetworks.com @AndreasVoellmy

We're excited to see what the community does with NQE.

+ Blog post: https://forwardnetworks.com/blog/network-query-engine
+ Github repo: https://github.com/forwardnetworks/network-query-engine-examples