ABSTRACT

This is the talk that one engineer wished she heard three years ago: a “What-Have-I-Learned” about starting from ground zero in network automation. From testing to production in a brownfield campus environment with over 80 model types, topics include network automation possibilities at different maturities, the need for spinning up a network "Source of Truth", the minimum data and effort required to get started, losing sleep trying to figure out the best approach to start, and popular tooling. This is a discussion of lessons learned and how those lessons carried over to Internet2’s latest efforts.

INTERNET2

2022 TECHNOLOGY exchange

Beginning Automation:
An Autobiography

Shannon Byrnes
Sr. NetDevOps Engineer, Internet2

INTERNET2

A to Z

The Talk I Wish I Heard

Setting the Stage

- ❏ One bright-eyed engineer with their whole career ahead of them.
- ❏ Knows enough Python to be dangerous.
- ❏ Read a lot of posts by Kirk Byers.
- ❏ 100+ Cisco Catalyst switches to upgrade.

Enter: IOS Upgrade Script

- An amalgamation that ultimately went on to be worked on by 3 different network engineers.
- Procedural using Netmiko and Nornir.
- List of hostnames pulled from monitoring software.
- No validation or md5 checks in the first iteration.
- What's a git? What's a pip?
- "Mega-Main".

Lesson 1

Engineers are wonderfully creative and talented people who *will* find a solution.

Lesson 2

Network engineers are not necessarily software engineers.

Lesson 3

Accidentally DoSing your TACACS server may lead to unexpected results.

Enter: Debug Website

- In addition to other miscellaneous, ad-hoc scripts, we created a website!
- Flask/Gunicorn App.
- Takes the visitor’s IP address and returns all kinds of stuff.
    - Logged-in User and IP
    - Connection type (Wired, Wireless, VPN)
    - Connected switch/interface or connected AP.
    - VLAN ID and client MAC address, if applicable
    - Client browser information
- We stole a lot of HTML and boilerplate code.

Lesson 4

While a Python Flask app helps to make a website simple,

Now you need to maintain a person on your team with a basic understanding of web development.

We Need a Source of Truth!

- Inventory management was through monitoring software.
  - Trust issues
  - Inaccurate PIDs and serial numbers
  - Difficult to depict stacks
- A need to associate additional information with network components.
  - People/Groups
  - Broadcast Domains
  - And more

(Note: Nautobot did not exist yet)*

Lesson 5: “Source of Truth” is Overloaded

(According to Shannon)*

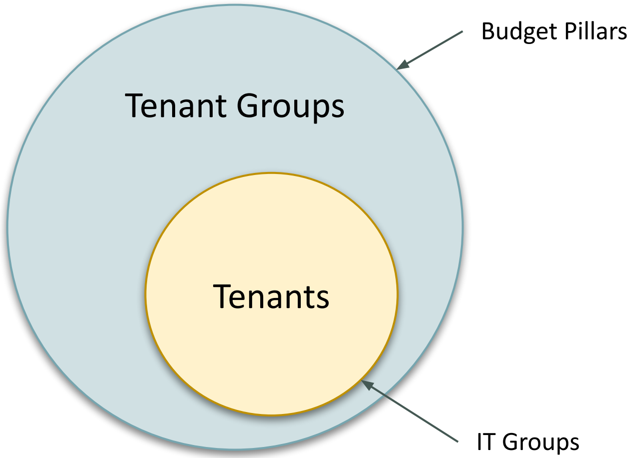
Source of Truth Responsibility	Not a Source of Truth Responsibility	Example
An API-driven and *trusted* repository containing the intended state of the network.	Repository of the current or operational state of the network.	***Optimal:*** Diffs between the network and the SoT can show issues or pending work. ***Optimal***: Trusted to help drive reporting. ***Less-Optimal***: Network assets, like devices, “only exist if they are in monitoring.”
Leveraged by external or decoupled tooling for purposes of network automation or orchestration.	Automating or orchestrating the network.	***Optimal***: The SoT exists independently of automation tooling. ***Less-Optimal***: Changes to automation tooling require changes to the SoT due to coupling.
Provide references or act as a “key-store” for external data that is authoritative elsewhere.	Duplication of data that is authoritative elsewhere, leading to potential loss of trust and data integrity.	***Optimal***: Tenant names in the SoT exclusively dictate names in other systems, like Grouper groups. ***Less-Optimal***: SoT stores the timestamp for when an interface was last “up”. ***Less-Optimal***: SoT tracks DHCP pool usage.

(This is not how everyone starts, and that’s okay and expected.)*

First Steps

A lot of research and design decisions on schema

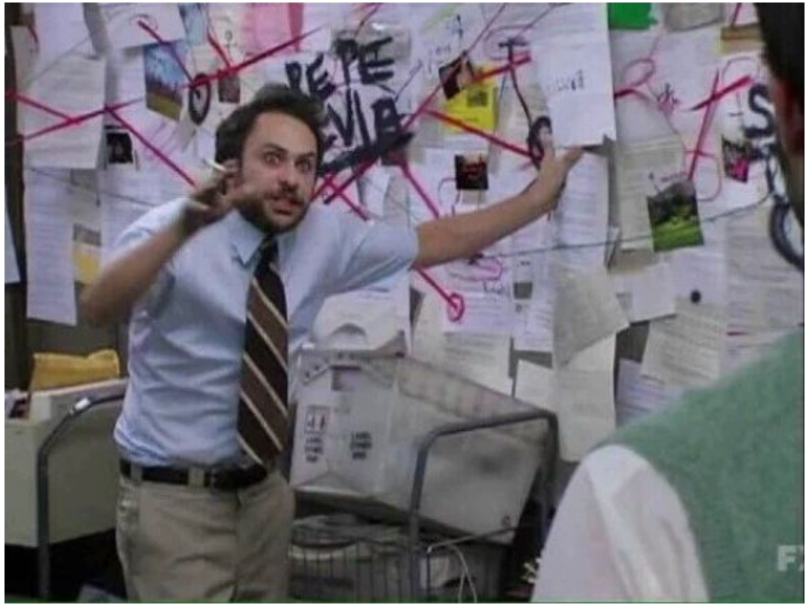
- How do we represent distributed IT groups and their subnets?
- Should a device represent a stack or just a stack member?
- How do we represent chassis and their line cards?
- Should we represent interface state or configuration? Up/down, assigned VLANs, trunking, etc.
- How do we associate VLAN IDs with the sites they're spanned between? What if there is more than one domain within the same site?
- Should a conjoined campus building, with two physical addresses, be a single site?



Lesson 6

Allow plenty of time to simply model your network in a Source of Truth.

Do not be afraid to pivot.

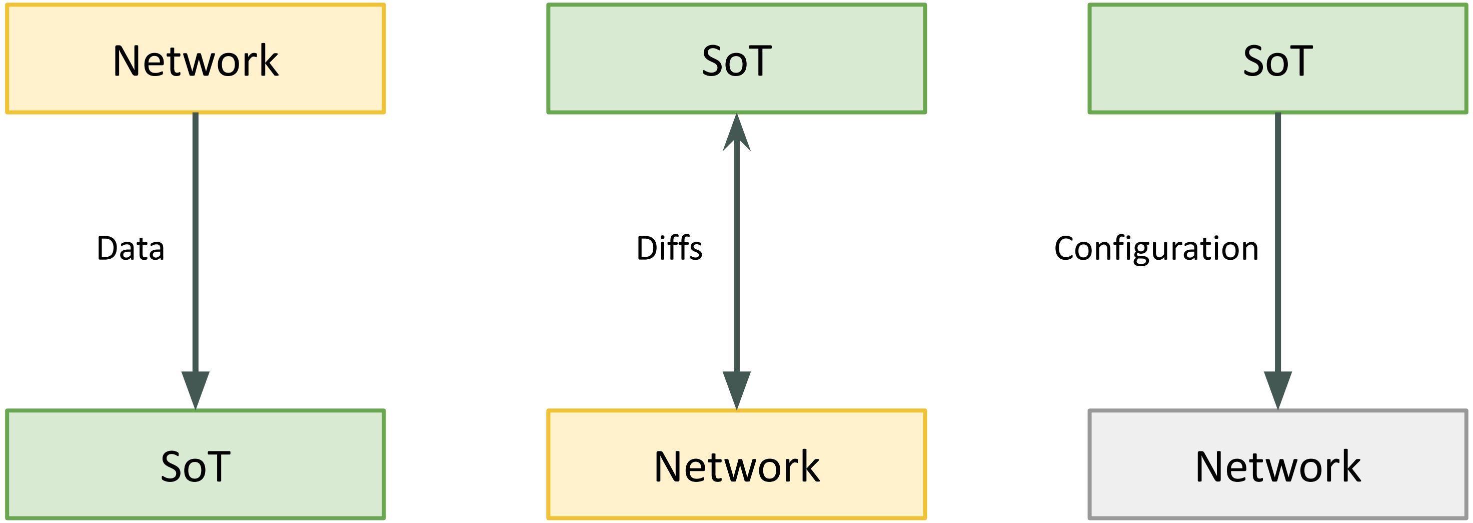


Getting Data In

- To lay groundwork for automation, need to populate basic infrastructure data.
    - Devices and PIDs
    - Device Roles
    - Virtual Chassis for stacks, keep stack members as Devices
    - Interfaces
    - Inventory Items like line cards and optics

- … and IPAM data.
    - Prefixes and Aggregates
    - VLANs and VLAN Groups

A to Z?


Network
Data
SoT
SoT
Diffs
Network
SoT
Configuration
Network

Lesson 7

Vigilance against scope creep is <u>very</u> important.

“Separation of Concerns”

LEFT
EXIT 12
Intended State of the Network
Adding operational data for reports
Me, with not enough hours in the day.
imgflip.com

Enter: NetBox Onboarder

First Iteration

- Pull all hostnames from monitoring software.
- Monitoring software knows their platforms. I.e., IOS, NXOS, ASA.
- Use Nornir, Netmiko, and ntc-templates (Jinja2) to CLI-scrape.

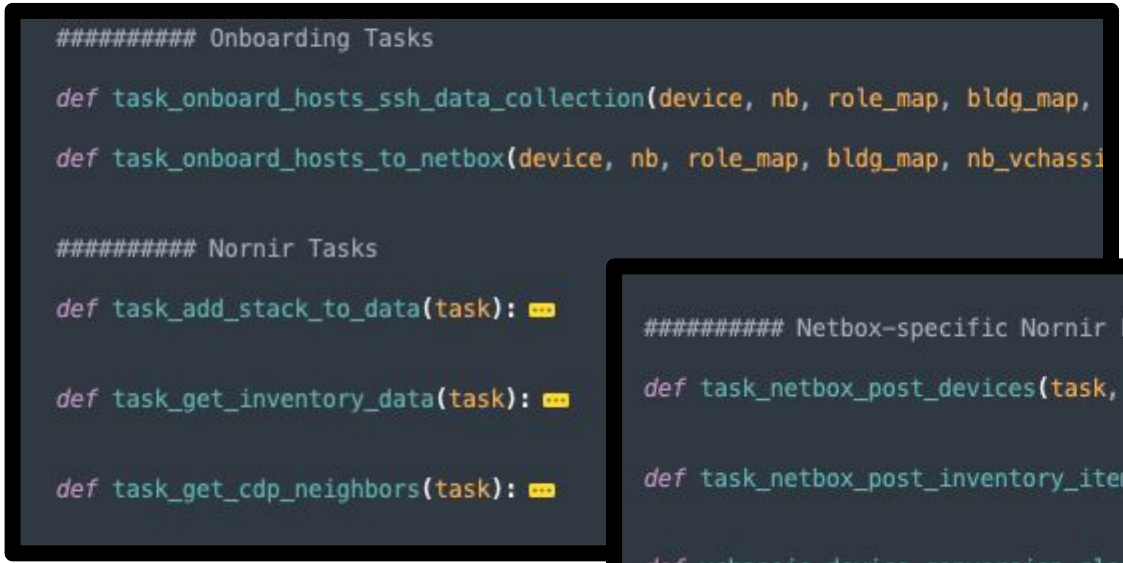
Second Iteration

- General refactoring.
- Now can handle manual input instead of exclusively using monitoring software dumps.

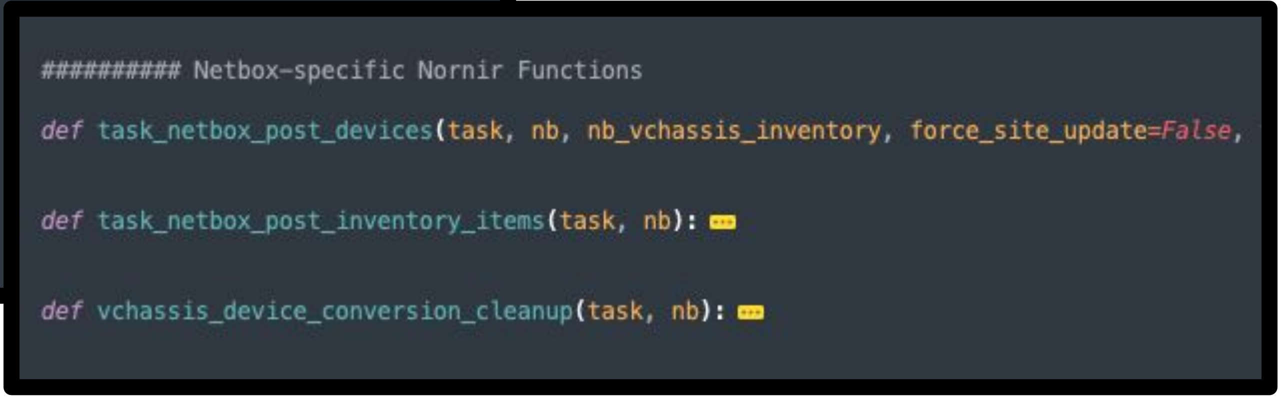
Third Iteration

- Create CLI tool that imports the onboarder to onboard devices by name.

Example Function Definitions



```
########## Onboarding Tasks

def task_onboard_hosts_ssh_data_collection(device, nb, role_map, bldg_map,

def task_onboard_hosts_to_netbox(device, nb, role_map, bldg_map, nb_vchassi

########## Nornir Tasks

def task_add_stack_to_data(task): ...

def task_get_inventory_data(task): ...

def task_get_cdp_neighbors(task): ...
```



```
########## Netbox-specific Nornir Functions

def task_netbox_post_devices(task, nb, nb_vchassis_inventory, force_site_update=False,

def task_netbox_post_inventory_items(task, nb): ...

def vchassis_device_conversion_cleanup(task, nb): ...
```

Lesson 8

Plan ahead on how you want to handle differing syntaxes among different device types.

Plan for trial and error.

Depending on the number of platforms and device types you have, make sure to prioritize good mental health practices.

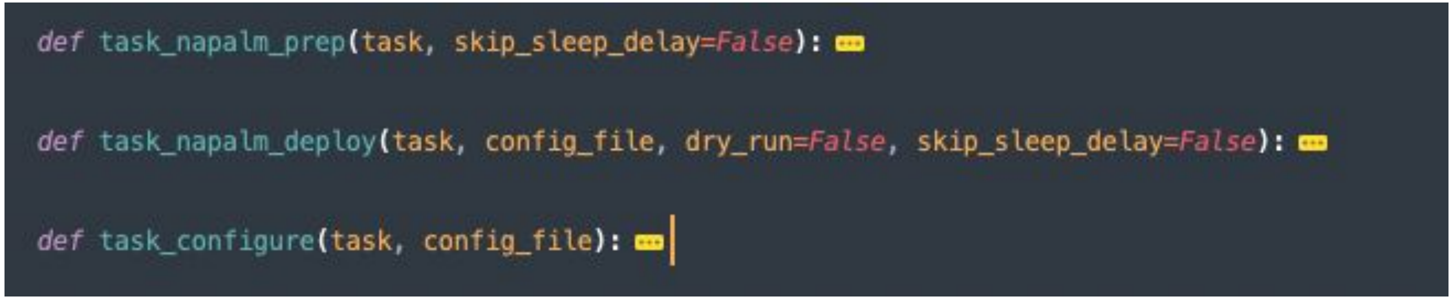
Lesson 9

Implementing good logging should be a priority.

- More than just print statements.
- LOG_LEVEL should be easily modifiable.
- Remember that you can capture logs from your imported packages in Python, such as Netmiko logs.

Enter: NetBox Config Deployer

- Used the NetBox inventory plugin with Nornir
- NAPALM tasks with a sprinkle of Netmiko tasks
    - At the time, NAPALM did not have an official ASA driver.
- Deploy arbitrary configuration.
- Wrote additional custom modules for specific projects.

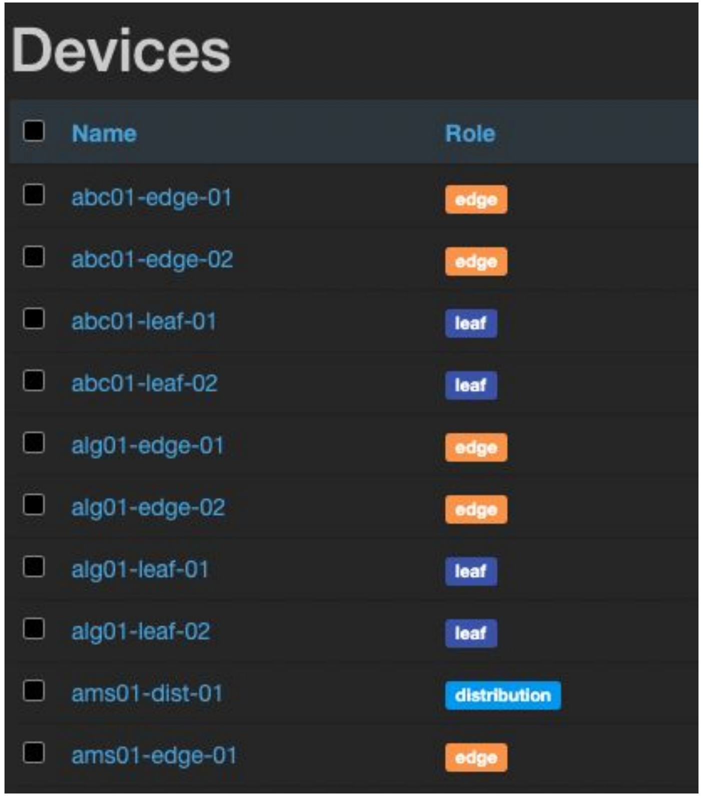


```
def task_napalm_prep(task, skip_sleep_delay=False): ...

def task_napalm_deploy(task, config_file, dry_run=False, skip_sleep_delay=False): ...

def task_configure(task, config_file): ...
```

Lesson 10

Device Roles are powerful and worth the effort.

- Access Layer L2 Switch
- Data Center Access Layer L2 Switch
- Building Aggregation L3 Switch
- SDMZ L2 Switch
- Etc



Devices

Name	Role
abc01-edge-01	edge
abc01-edge-02	edge
abc01-leaf-01	leaf
abc01-leaf-02	leaf
alg01-edge-01	edge
alg01-edge-02	edge
alg01-leaf-01	leaf
alg01-leaf-02	leaf
ams01-dist-01	distribution
ams01-edge-01	edge

(Source: https://demo.nautobot.com/dcim/devices/)*

General Automated Project Examples

- Image upgrades.
- Emergency password rotation.
- Enable LLDP across the access layer for new E911 implementation.
- Update IP Helpers that use centralized DHCP across all edge SVIs.
- Routine onboarding of APs via Cisco Prime and WLC CLI.
- Conditionally update device and interface configuration for campus-wide deployment of closed dot1x.
- Detect newly plugged-in dot1x supplicant switches via CDP and run configuration tasks.
- …a lot of reporting.

Lesson 11

Every new deliverable that the automation team provides is now a permanent responsibility that must be supported for the foreseeable future.

“That cool ad-hoc report you did last year for some one-off, ASAP thing? Yeah, we need it every year now.”

Takeaways

Even more lesson slides??

Many more lessons learned.

- Few engineers on the team could run the code, and fewer understood it.
- High-stakes software was being developed by green developers.
- Inexperience often means additional issues with estimating time and effort.
- Management is inexperienced with managing developers.
- Little documentation.
- High-risk of technical debt.

Enter: A Pretend Time Machine… and a Bag of Money

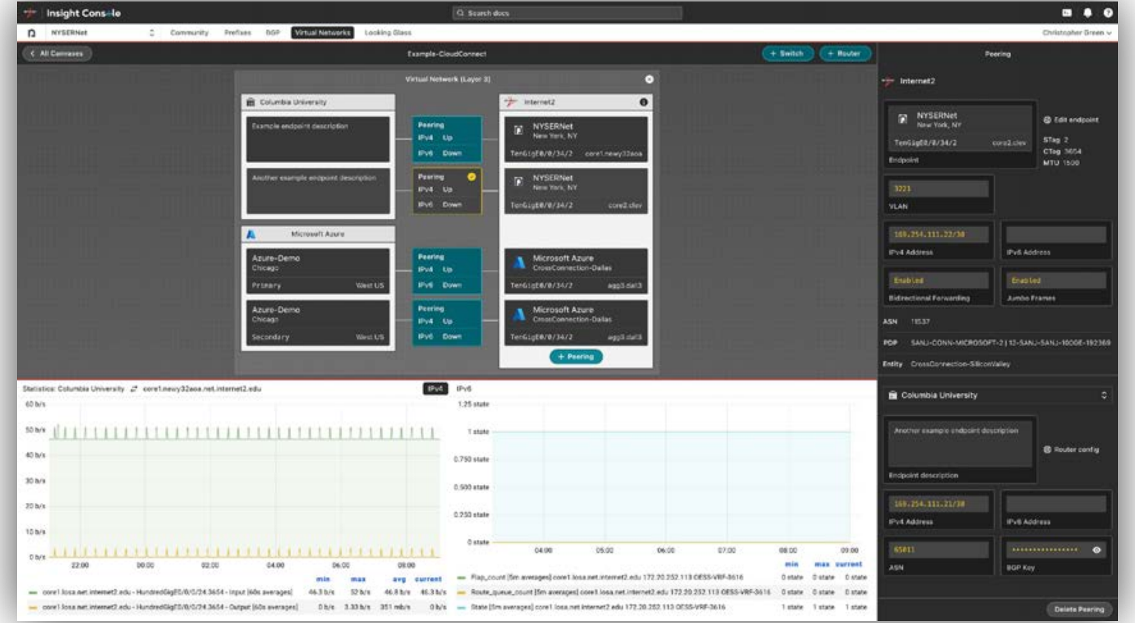
- Twice as much time. Getting and estimating time was the biggest challenge.
  - Document everything.
  - Take less shortcuts; reduce technical debt
  - More active overlap with other engineers.
  - More code polishing to make it accessible to more team members.

- Treated as a large, multi-year project that requires dedicated team members and accounts for turnover.
  - You're developing a new team.
  - Demand is high, and universities are up against for-profit companies.

Thank you!

Take Insight Console for a test drive

usability.ns.internet2.edu



Columbia University
- Example endpoint description — Peering: IPv4 Up, IPv6 Down — NYSERNet, New York, NY, TenGigE0/0/34/2, core1.newy32aoa
- Another example endpoint description — Peering: IPv4 Up, IPv6 Down — NYSERNet, New York, NY, TenGigE0/0/34/2, core2.clev

Microsoft Azure
- Azure-Demo, Chicago, Primary, West US — Peering: IPv4 Up, IPv6 Down — Microsoft Azure, CrossConnection-Dallas, TenGigE0/0/34/2, agg3.dal3
- Azure-Demo, Chicago, Secondary, West US — Peering: IPv4 Up, IPv6 Down — Microsoft Azure, CrossConnection-Dallas, TenGigE0/0/34/2, agg3.dal3

Peering

Internet2

NYSERNet, New York, NY, TenGigE0/0/34/2, core2.clev — Endpoint
STag 2
CTag 3654
MTU 1500
VLAN: 3221
IPv4 Address: 169.254.111.22/30
IPv6 Address:
Bidirectional Forwarding: Enabled
Jumbo Frames: Enabled
ASN 11537
PDP SANJ-CONN-MICROSOFT-2 | 12-SANJ-SANJ-100GE-192369
Entity CrossConnection-SiliconValley

Columbia University

Endpoint description: Another example endpoint description
IPv4 Address: 169.254.111.21/30
IPv6 Address:
ASN: 65011
BGP Key: ****************

Statistics: Columbia University ⇄ core1.newy32aoa.net.internet2.edu — IPv4 IPv6

	min	max	avg	current
core1.losa.net.internet2.edu - HundredGigE0/0/0/24.3654 - Input [60s averages]	46.3 b/s	52 b/s	46.8 b/s	46.3 b/s
core1.losa.net.internet2.edu - HundredGigE0/0/0/24.3654 - Output [60s averages]	0 b/s	3.33 b/s	351 mb/s	0 b/s

	min	max	current
Flap_count [5m averages] core1.losa.net.internet2.edu 172.20.252.113 OESS-VRF-3616	0 state	0 state	0 state
Route_queue_count [5m averages] core1.losa.net.internet2.edu 172.20.252.113 OESS-VRF-3616	0 state	0 state	0 state
State [5m averages] core1.losa.net.internet2.edu 172.20.252.113 OESS-VRF-3616	1 state	1 state	1 state

Plaza Room 5
Drop-in or schedule in advance

INTERNET2 2022TECHNOLOGYEXCHANGE