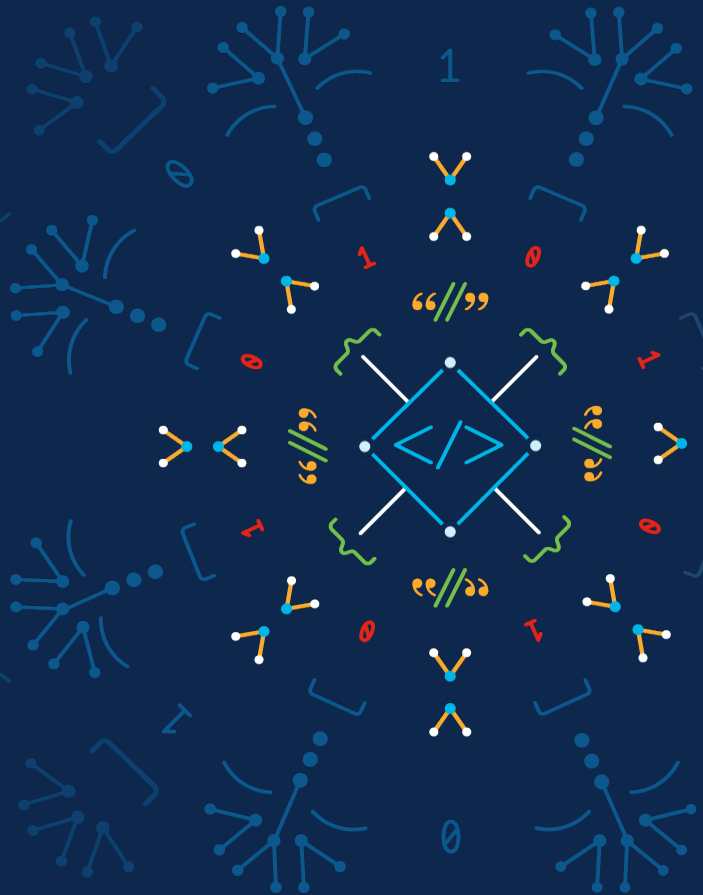
Developer Days
Automation

cisco
The bridge to possible

Stop the Chaos, Organize Your Network with NSO and Netbox

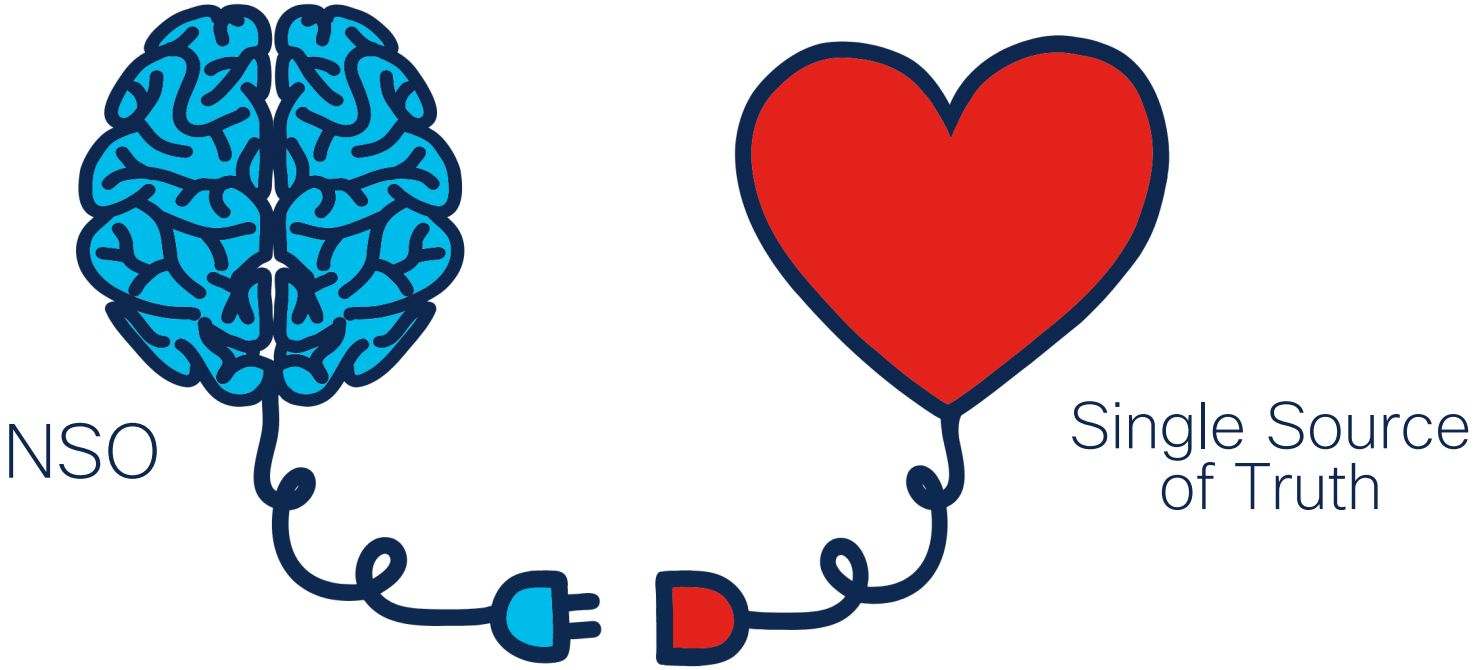


Anna Wójcik
Software Engineer Infrastructure – NSO BU
May 9th 2023

“In the Beginning, There was Chaos”

Agenda

1. Single Source of Truth
2. Single Network Interface
3. NSO and Netbox meet
4. What’s next?



Network Automation

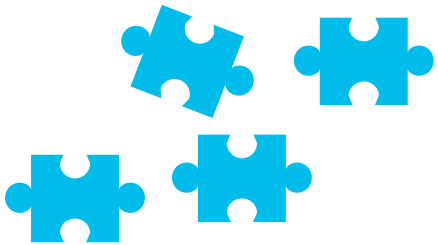
Chapter 1:

Single source of truth –
But what does it mean?

Where do we keep information about the network?

- HLD, LLD
- Spreadsheets
- IPAM
- Git repository
- ...ohhhh, there is this one guy, that should know that...

- Personal notes
- Wiki
- Databases
- NSO
- ...



What information do we want to keep?

- Inventory (racks, devices)
- IP address management
- Cabling and connections
- Sites and locations
- Virtual machines
- Vlans, VRFs, AS
- Anything else? – Sure

How about Netbox?

Take advantage of Netbox

Base

- Data Model
- Customization
- Open Source
- Dockerized

Integrations

- Scripts
- Webhooks
- Plugins

API

- REST
- Swagger
- GraphQL

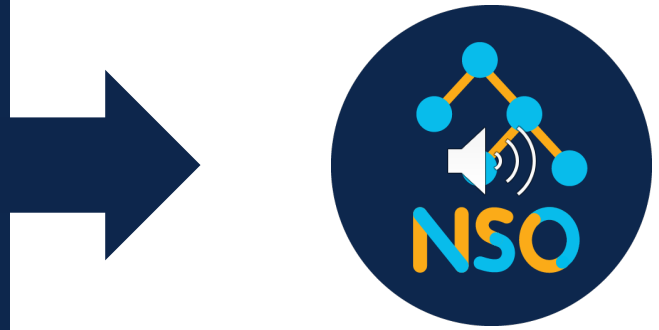
DEMO

Chapter 2:

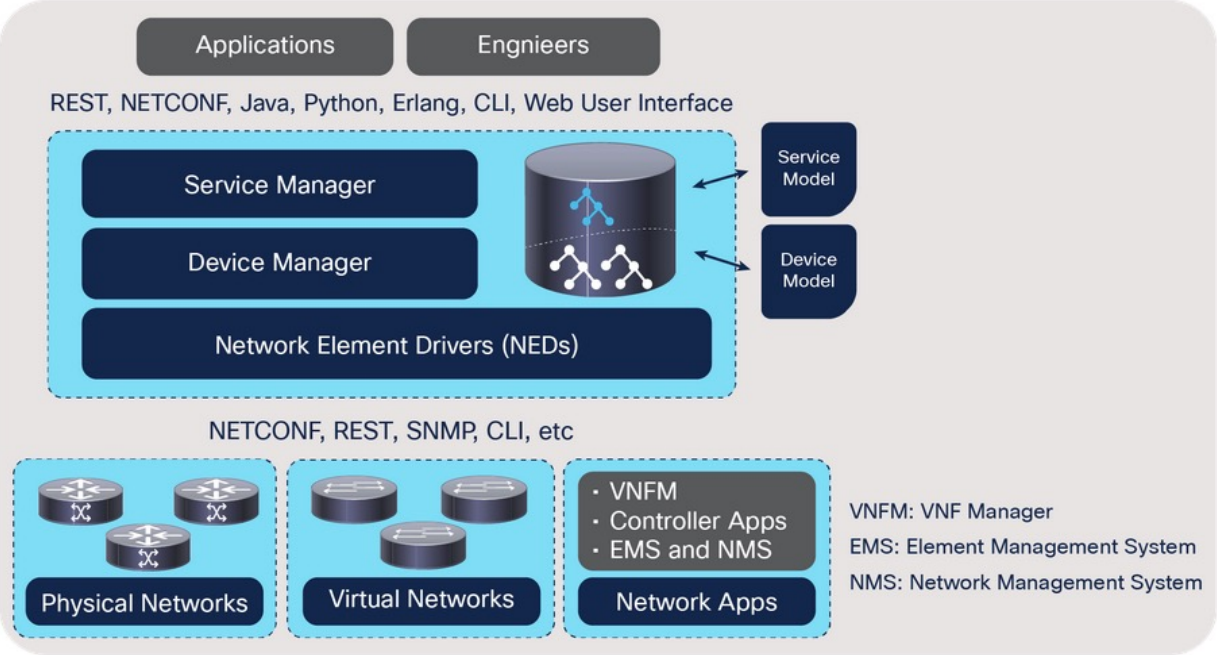
Single Network Interface

How do we talk to our devices?

- CLI
- CLI != CLI
- SNMP
- XML
- NETCONF
- RESTCONF
- Web Interface
- API

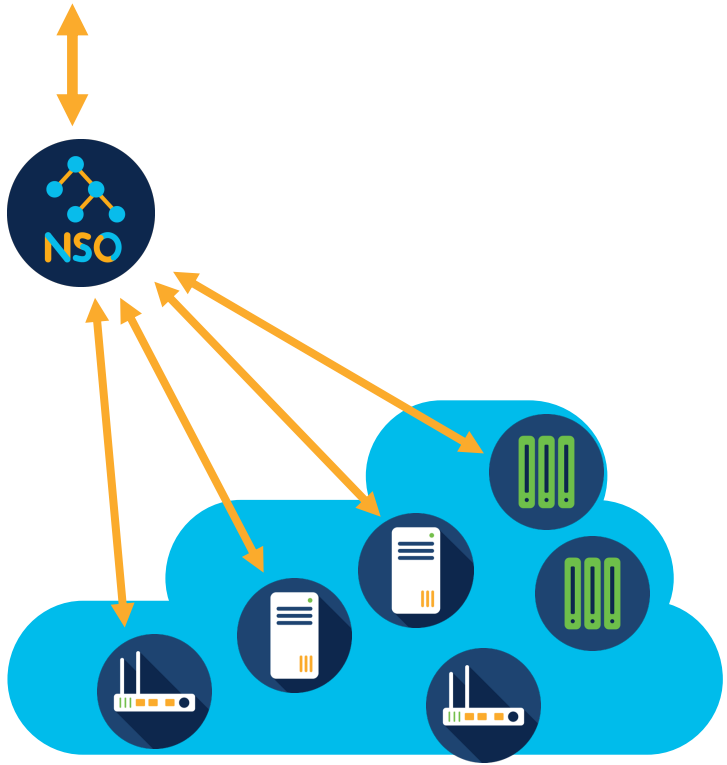


Network Element Driver

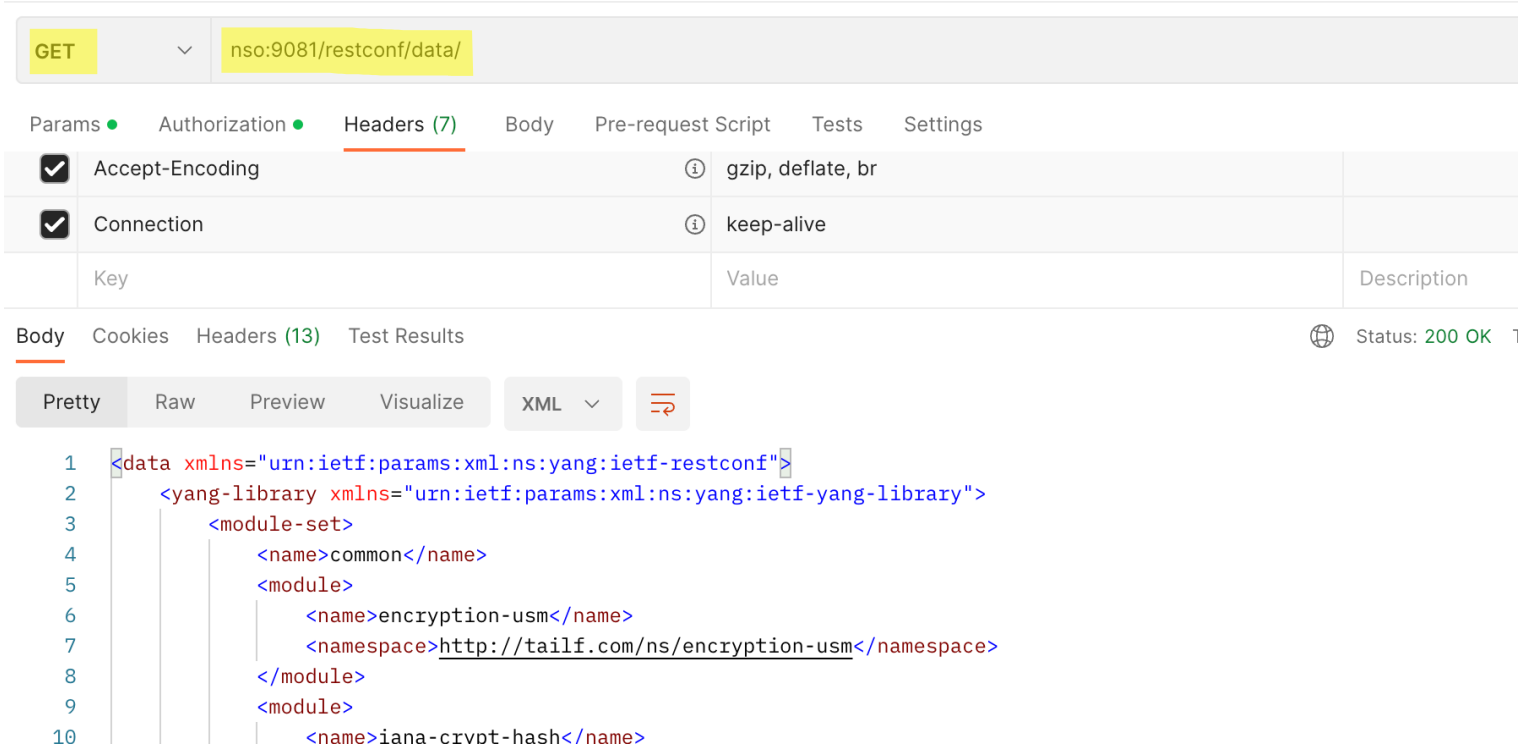


NSO

- Multivendor orchestration platform (NED)
- Configuration synchronization
- Configuration templates
- Services (YANG)
- Single API to entire network

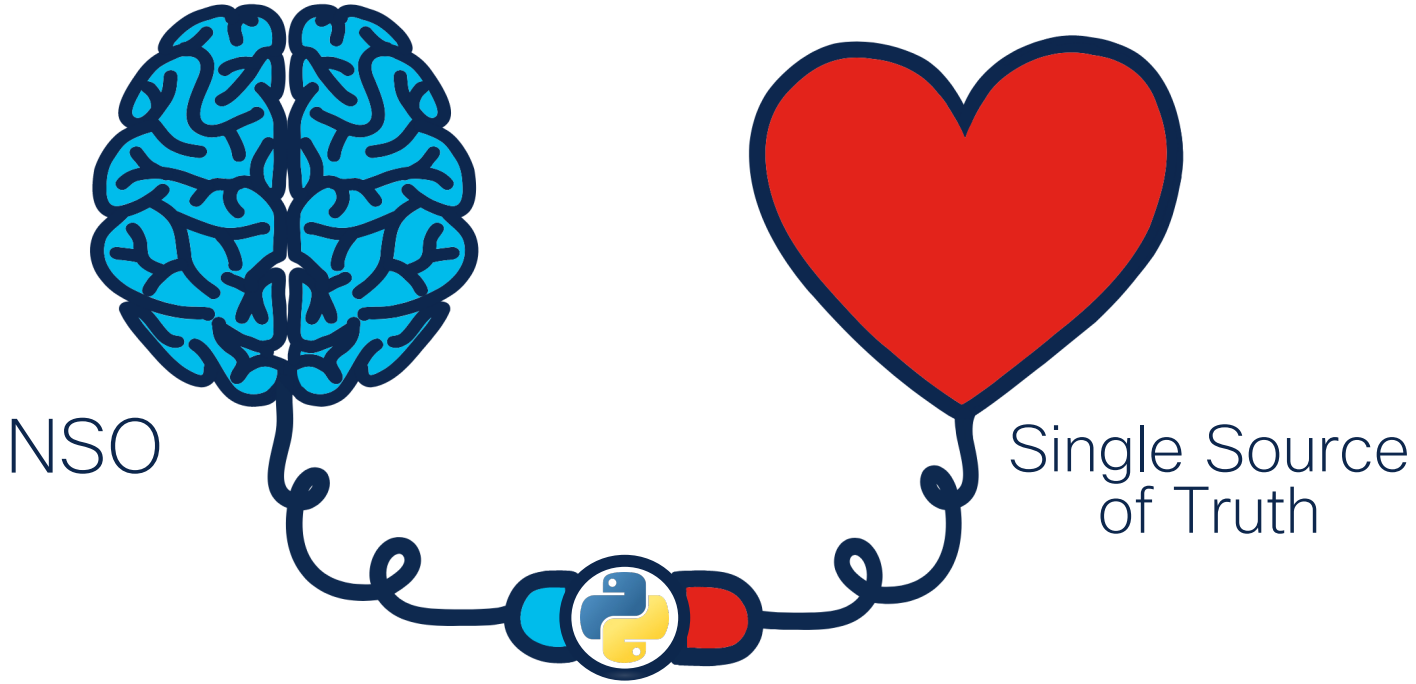


RESTCONF API



Chapter 3

Let’s make them talk to each other



Network Automation

Part 1 – New branch office

1. Create:
   Site, Rack, Devices, IP subnet

2. Create associations

3. Add devices to NSO

How to create Netbox script?

Define input data

```
class NewSiteScript(Script):
    class Meta:
        name = "New Site"
        description = "Provision a new site"

    site_codename = StringVar(
        description="Short name of the new site",
    )
    switch_count = IntegerVar(
        description="Number of access switches in the site",
        default=1
    )
    switch_model = ObjectVar(
        description="Switch model",
        model=DeviceType
    )
    add_to_nso = BooleanVar(
        description="Add devices to NSO inventory"
    )
```

Define logic

```
def run(self, data, commit):
    site_codename = data['site_codename']
    switch_count = data['switch_count']
    switch_model = data['switch_model']
    add_to_nso = data['add_to_nso']


    site = self.create_new_site(site_codename)

    mgmt_id = self.find_next_free_mgmt_id()
    prefix = self.create_mgmt_prefix(site, mgmt_id)
    dns_results = self.dns_allocations(site, mgmt_id, switch_count)

    rack = self.create_rack(site)
    self.create_switch(site, switch_model, switch_count, mgmt_id,
rack)


    if add_to_nso:
        self.add_devices_to_nso(site, dns_results)
```

DEMO

Means to interact

1) Netbox with native python packages

```
def create_new_site(self, site_codename, site_name):
    site = Site(
        name=site_codename,
        slug=slugify(site_codename),
        description=site_name,
        status=SiteStatusChoices.STATUS_PLANNED
    )

    site.save()
    self.log_success("New site %s (%s) created" % (site_codename,
site_name))
    return site
```

Means to interact

1) Netbox with native python packages

2) NSO with requests

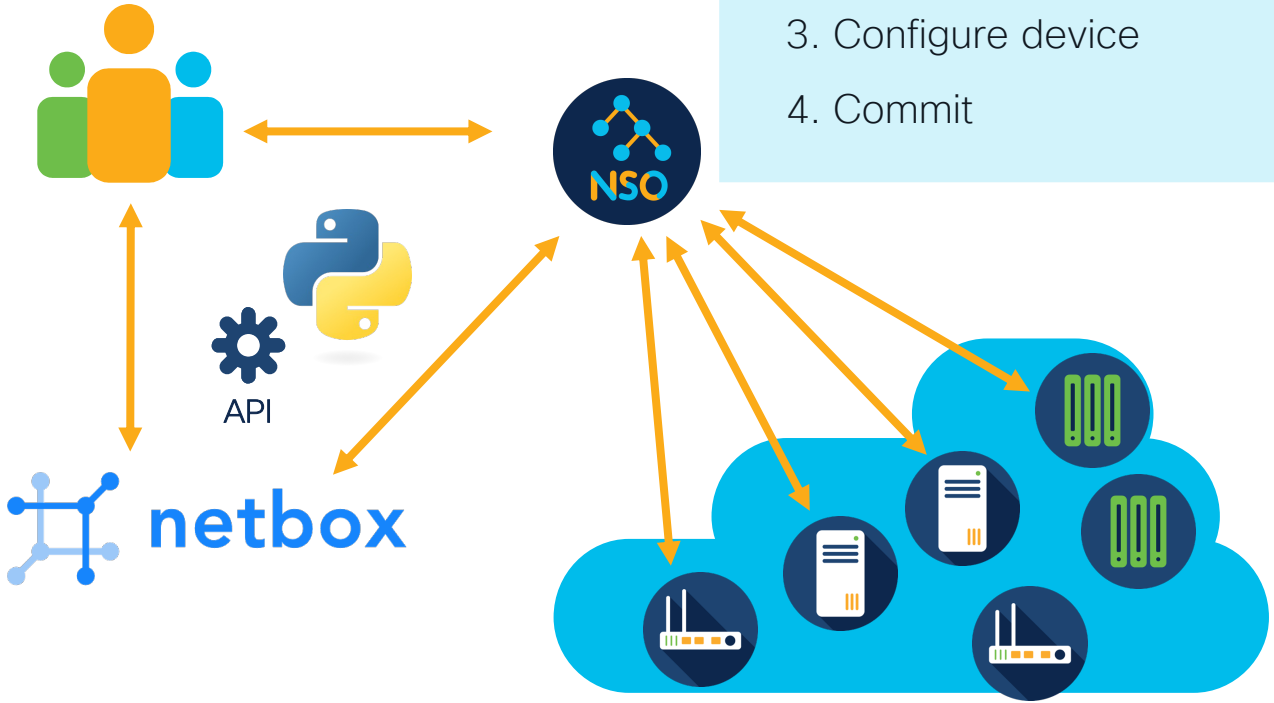
```
headers = {
    'Accept': 'application/yang-data+json',
    'Content-Type': 'application/yang-data+json',
    'Authorization': 'Basic YWRtaW46YWRtaW4='
}

payload = json.dumps({
    "tailf-ncs:device": [
        {
            "name": name,
            "address": ip,
            "port": 22,
            "authgroup": "mygroup",
            "device-type": {
                "cli": {
                    "ned-id": "cisco-ios-cli-3.8:cisco-ios-cli-3.8"
                }
            },
            "state": {
                "admin-state": "unlocked"
            }
        }
    ]
})

response = requests.request("POST", url, headers=headers, data=payload)
resp = response.status_code
```

Part 2 – DHCP server

1. GET subnet information from Netbox
2. Reserve IP range for DHCP pool
3. Configure device
4. Commit



How to create NSO python script?

Talk to Netbox

```
def get_subnet():
    url = netbox + "ipam/prefixes/?site=" + site
    response = requests.request("GET", url, headers=headers, data={}).json()
    subnet = response["results"][0]["prefix"]
    return subnet


def reserve_dhcp_pool(subnet):
    start_address = subnet.replace(".0/24", ".129/24")
    end_address = subnet.replace(".0/24", ".254/24")

    url = netbox + "ipam/ip-ranges/"
    payload = json.dumps({
        "start_address": start_address,
        "end_address": end_address,
        "description": pool_name
    })
    response = requests.request("POST", url, headers=headers, data=payload)
    return response
```

Talk to device

```
def configure_device(subnet):
    with ncs.maapi.single_write_trans('admin', 'python', groups=['ncsadmin']) as t:
        root = ncs.maagic.get_root(t)
        device_cdb = root.devices.device[device_name]
        device_cdb.config.ios__ip.dhcp.pool.create(pool_name)
        device_cdb.config.ios__ip.dhcp.pool[pool_name].default_router.create(default_router)
        device_cdb.config.ios__ip.dhcp.pool[pool_name].network.network_number =
network_number
        device_cdb.config.ios__ip.dhcp.pool[pool_name].network.mask = "255.255.255.128"
        t.apply()
```

DEMO

Means to interact

1) Netbox with native python packages

2) NSO with requests

3) NSO data with built in API

```
with ncs.maapi.single_write_trans('admin', 'python', groups=['ncsadmin']) as t:
    root = ncs.maagic.get_root(t)
    device_cdb = root.devices.device[device_name]
    device_cdb.config.ios__ip.dhcp.pool.create(pool_name)
    device_cdb.config.ios__ip.dhcp.pool[pool_name].default_router.create(default_router)
    device_cdb.config.ios__ip.dhcp.pool[pool_name].network.network_number = network_number
    device_cdb.config.ios__ip.dhcp.pool[pool_name].network.mask = "255.255.255.128"
    params = t.get_params()
    params.dry_run_native()
    result = t.apply_params(True, params)
    t.apply_params(True, t.get_params())
```

Means to interact

1) Netbox with native python packages

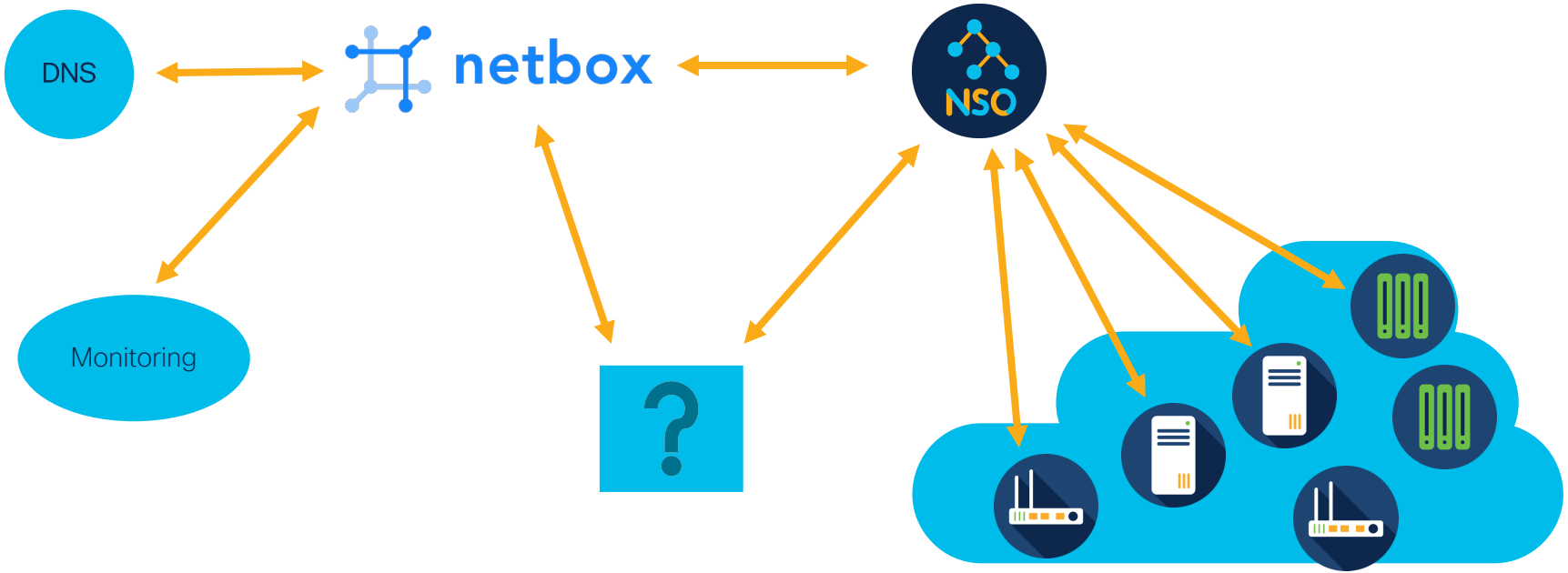
2) NSO with requests

3) NSO data with built in API

4) Netbox data with requests

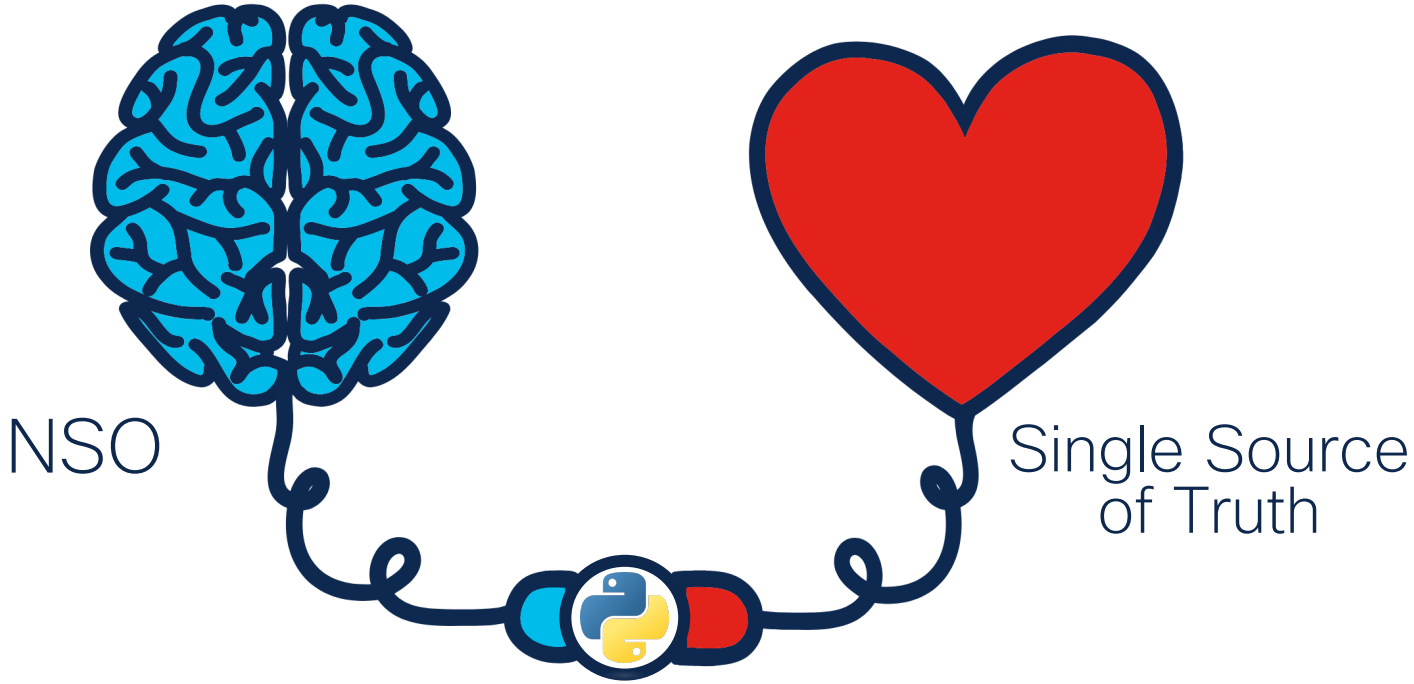
```
url = netbox + "ipam/ip-ranges/"
payload = json.dumps({
    "start_address": start_address,
    "end_address": end_address,
    "description": pool_name
})
response = requests.request("POST", url, headers=headers, data=payload)
```

What can happen next?



Q&A

Moral of the Story



Network Automation

Add NSO and Netbox to your Automation Toolbox

… and experiment with them

cisco.com/go/nsohub

docs.netbox.dev

github.com/annately

Thank you

The bridge to possible