IT – NSO Training

Brandon Black, Elliot Wise, Patrick Huynh, Jason Belk

Agenda

- Day 1: What is NSO?
- Day 2: Using NSO
- Day 3: XML & Yang
- Day 4: Services
- Day 5: Complex Services and Python

WELCOME TO DAY ONE!

Agenda

- CONTEXT
- WHY AUTOMATE? WHY NSO?
- THE WHAT & WHY OF NSO
- NSO COMPONENTS
- BASH BASICS

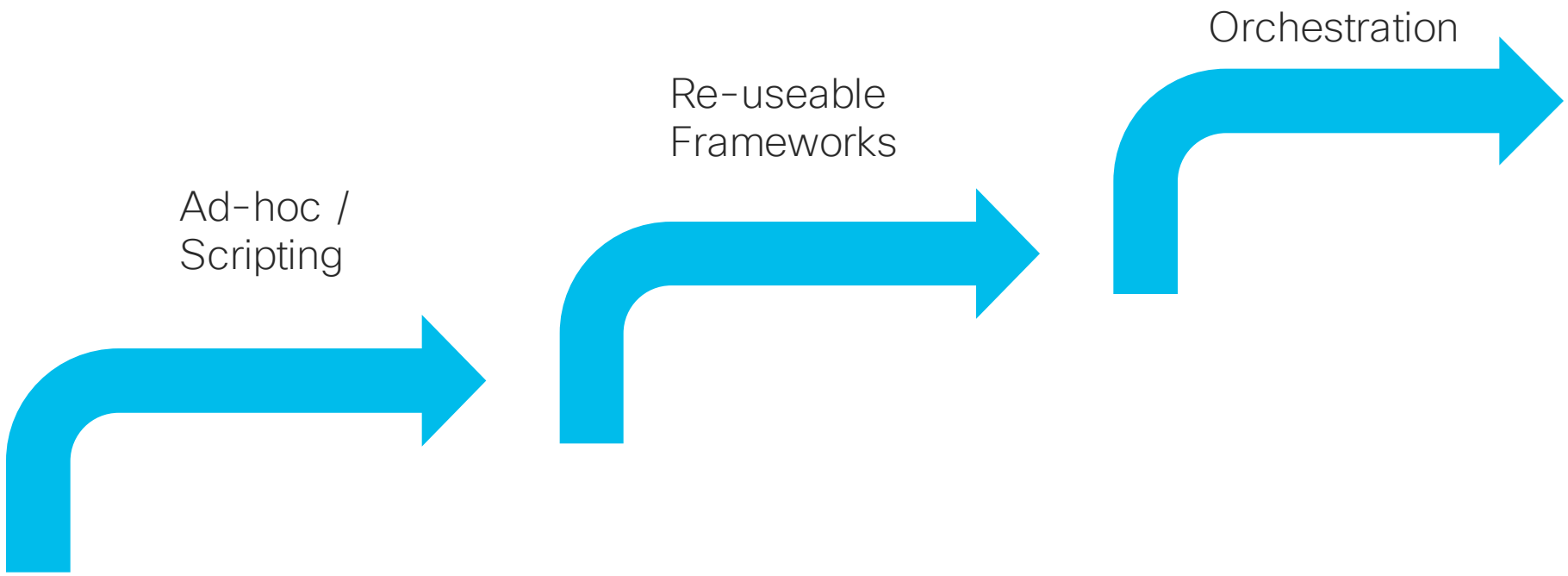
Training Expectations

- Learning is best when focused, laptops down
- Ask questions
- If unclear, please ask for clarification
- Spark room for question (1 trainer presenting, 1 on Spark)
- Labs are for learning, not merely tasks to be done

Context

Automation Maturity Model

NSO: Python API | Python Templates | Service Models



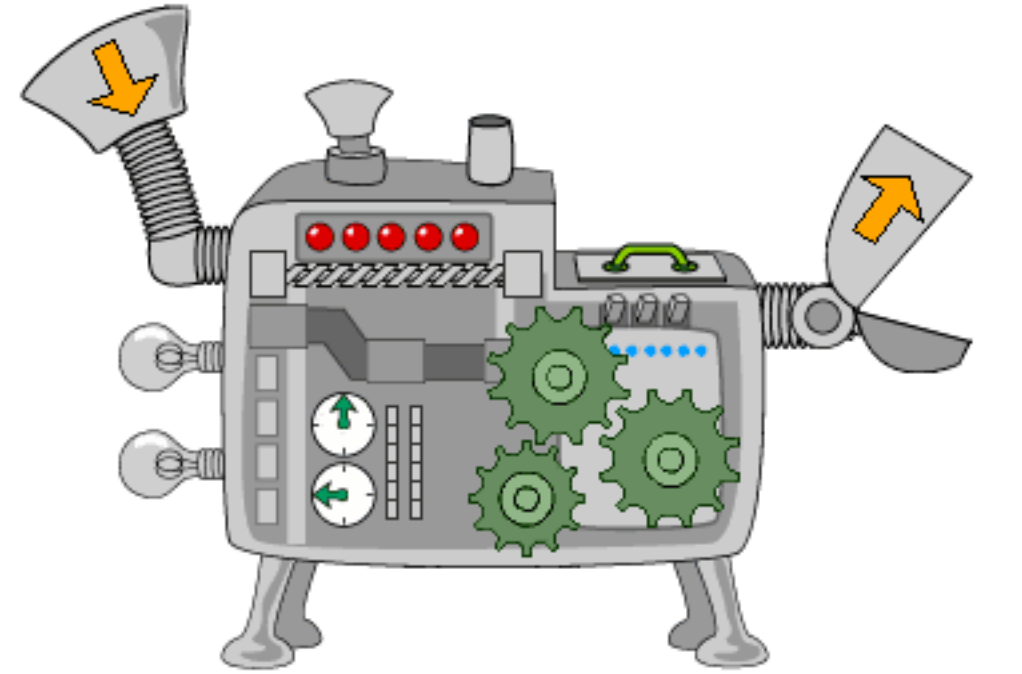
Engineer run one-off scripts and tools

Centrally managed frameworks & templates for faster development

Automated configuration lifecycle. Creation, Modification and Removal automated in one place

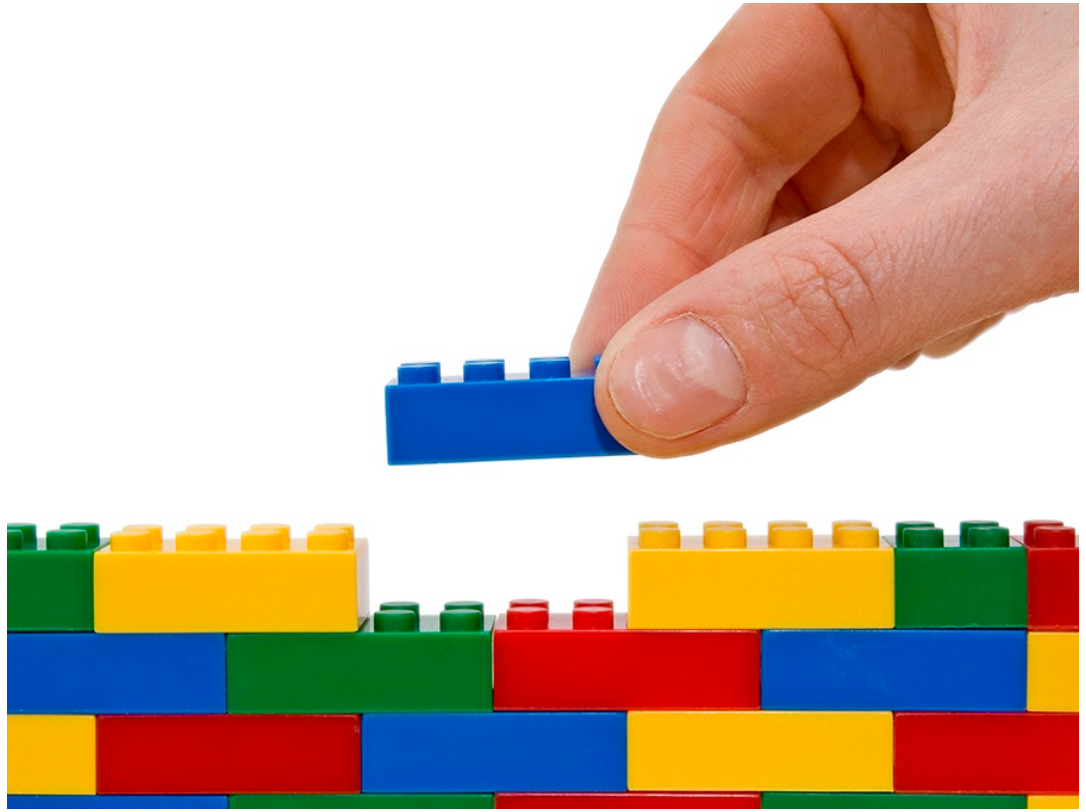
Automation Maturity Model – Ad Hoc / Scripting

- 1 slide ad-hoc slide example
- 802.1X Script used for EM/MM rollout, generating config
- N2i script
- Most python scripts (using Paramiko, Netmiko, Cisco-Connect)
- Built for a specific purpose, with some minor variations.



Automation Maturity Model – Re-Usable Frameworks

- Auto-config
- Templates
- Code scaffolds (insert your code here), NO2 modules, Cisco-Connect
- Ansible



Automation Maturity Model – Orchestration

- DNA-C
- NSO
- ACI APIC

What is one example from each of the 3 categories in the Automation Maturity Model ?

Demo – Script vs Orchestration

WHY AUTOMATE? WHY NSO?

Why Automate?

- Changing Skillset of a Network Engineer (everyone automates, not just a dedicated automation engineer)
- Network Infrastructure moving to Software-Defined Everything (NFV, SDA, etc.)
- Maintain and Create a predictable and uniform network, making operations much easier
- Speed to delivery at scale (making changes to 100s of devices)

Common Objections

- I am too busy to automate, or learn/practice automation skills
- I do not know where to start
- I get stuck setting up my environment
- I get interrupted when I finally get time to try
- I don’t see how I can automate this task, it is just easier to do it manually, since I know that will work
- There seems to be too many competing platforms (ansible, NSO, python, NO2, etc.), I don’t know which one to use
- I know how to use bash/SNMP/python/perl scripts, I don’t need anything else

Engineers getting started in automation

How to fail at network automation?

1. Start with high-risk, difficult problems.
2. Assume an all-or-nothing mindset (everything has to be automated or nothing can be automated).
3. Try to reinvent everything yourself.
4. Superficially copy code and patterns without comprehension.
5. Fail to learn good debugging processes.
6. [Hugely] over-engineering the solution.

Engineers getting started in automation

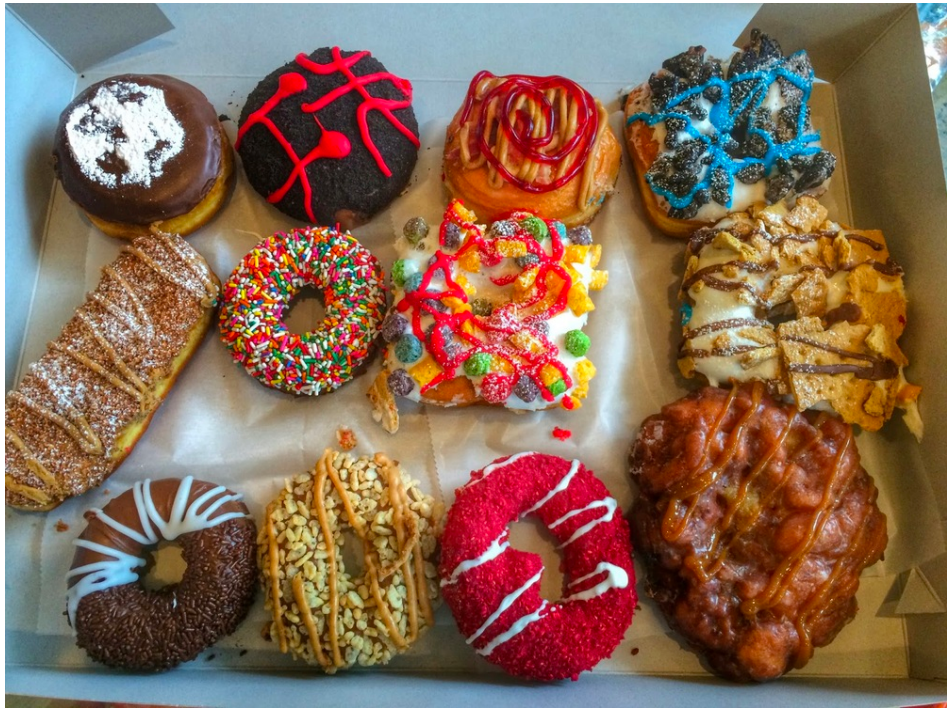
How to fail at network automation?

7. Fail to apply things that you learned on a small scale.
8. Being too busy to automate.
9. Fail to learn how to reuse your code [longer term].
10. Fail to use available developer tools: Git, linters, unit-testing, CI-tools [longer term].

How to Succeed at Network Automation

- Carve out time in your week where you can learn and practice the skills without interruption (no cell phone, no jabber/spark/email, etc.)
- Have a humble and teachable attitude, being patient while adjusting to the learning curve
- Teach others what you are learning to reinforce the concepts
- Pick a very small task to start with
- Read lots of examples and follow the logic
- Consider not only the problem at hand, but the business process around it
- Don't be afraid to ask others for help if you get stuck
- Use version control (git) and do not hard code passwords in code
- Actively seek to apply your skills to your daily job
- Write code / templates with readability and reusability in mind

Every Network is Unique vs. Predictable



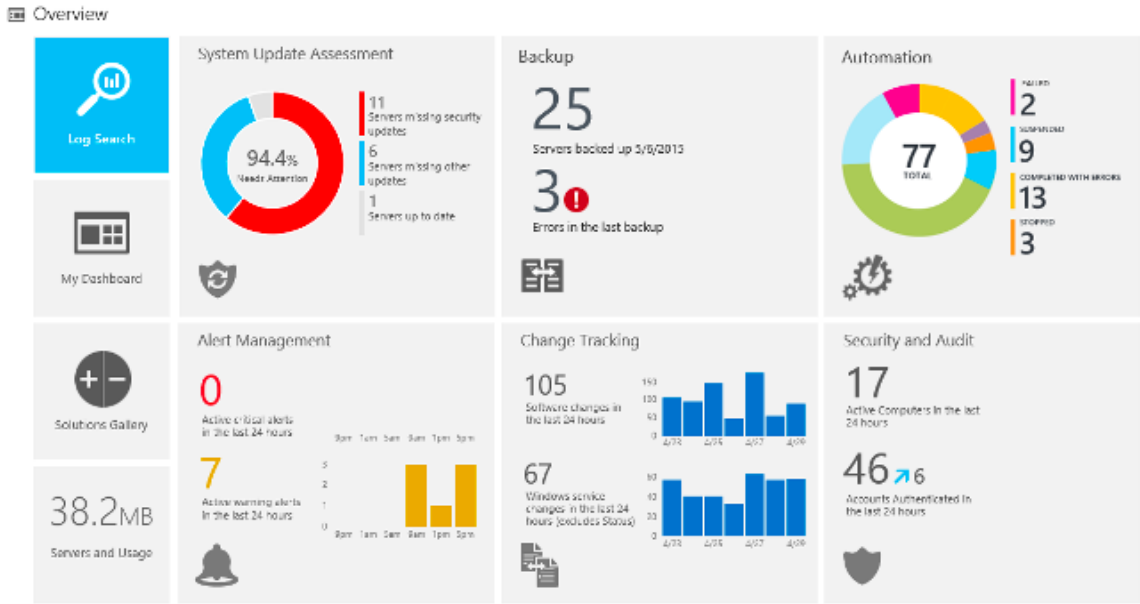
VS

(at least) Two Types of Automation

Configuration Management (auto-config for example)

```
R1(config-ext-nacl)#do sh access-list OutBoundAccess
Extended IP access list OutBoundAccess
    10 permit ip 192.168.1.0 0.0.0.255 any
    11 deny tcp 192.168.2.0 0.0.0.127 any eq smtp
    12 deny tcp 192.168.2.0 0.0.0.127 any eq sunrpc
    13 deny tcp 192.168.2.0 0.0.0.127 any eq pop2
    14 deny tcp 192.168.2.0 0.0.0.127 any eq nntp
    15 deny tcp 192.168.2.0 0.0.0.127 any eq ftp
    16 deny tcp 192.168.2.0 0.0.0.127 any eq ftp-data
    17 deny tcp 192.168.2.0 0.0.0.127 any eq telnet
    18 deny tcp 192.168.2.0 0.0.0.127 any eq cmd
    19 deny tcp 192.168.2.0 0.0.0.127 any eq irc
    20 permit ip 192.168.2.0 0.0.0.255 any
    30 permit ip 192.168.3.0 0.0.0.255 any
    40 permit ip 192.168.4.0 0.0.0.255 any
    50 permit ip 192.168.5.0 0.0.0.255 any
R1(config-ext-nacl)#
```

Operational Alerting (EMAN Monitoring, Science Logic, etc)



WHAT IS NSO?

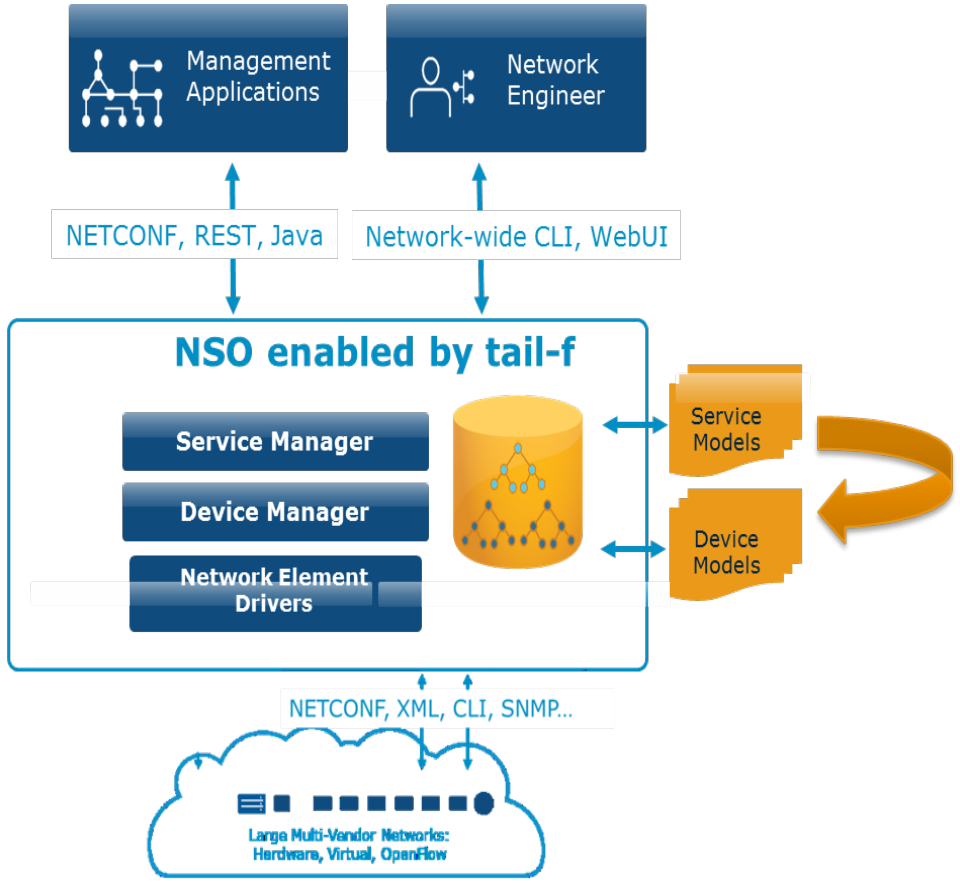
The Flexibility of NSO

Do you want to build a house or a boat?

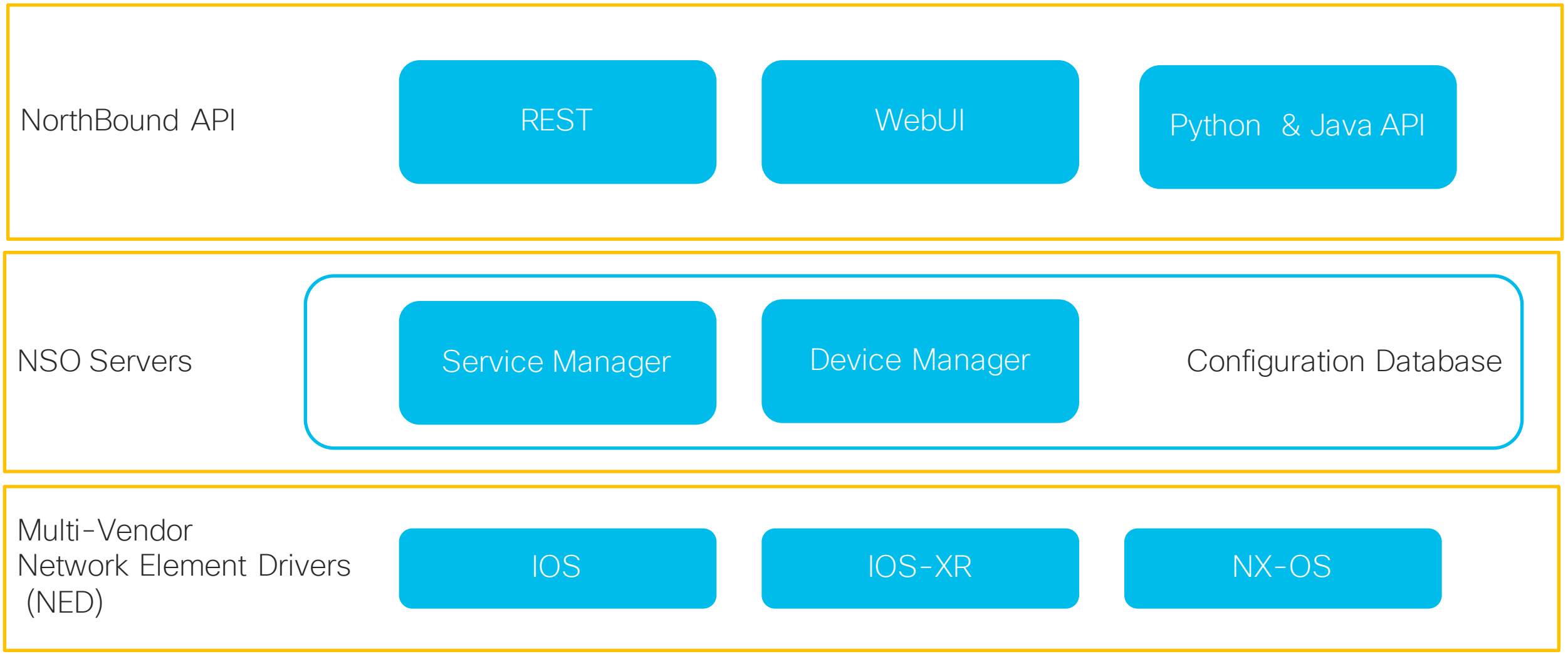
NSO gives the tools to do either (metaphorically).

Network Service Orchestrator Overview

Model Driven
Transactional
Automation Engine
Multi-Vendor
Built for Network engineers
API into the Network



Network Service Orchestrator Components



The “Service” in Network Service Orchestrator

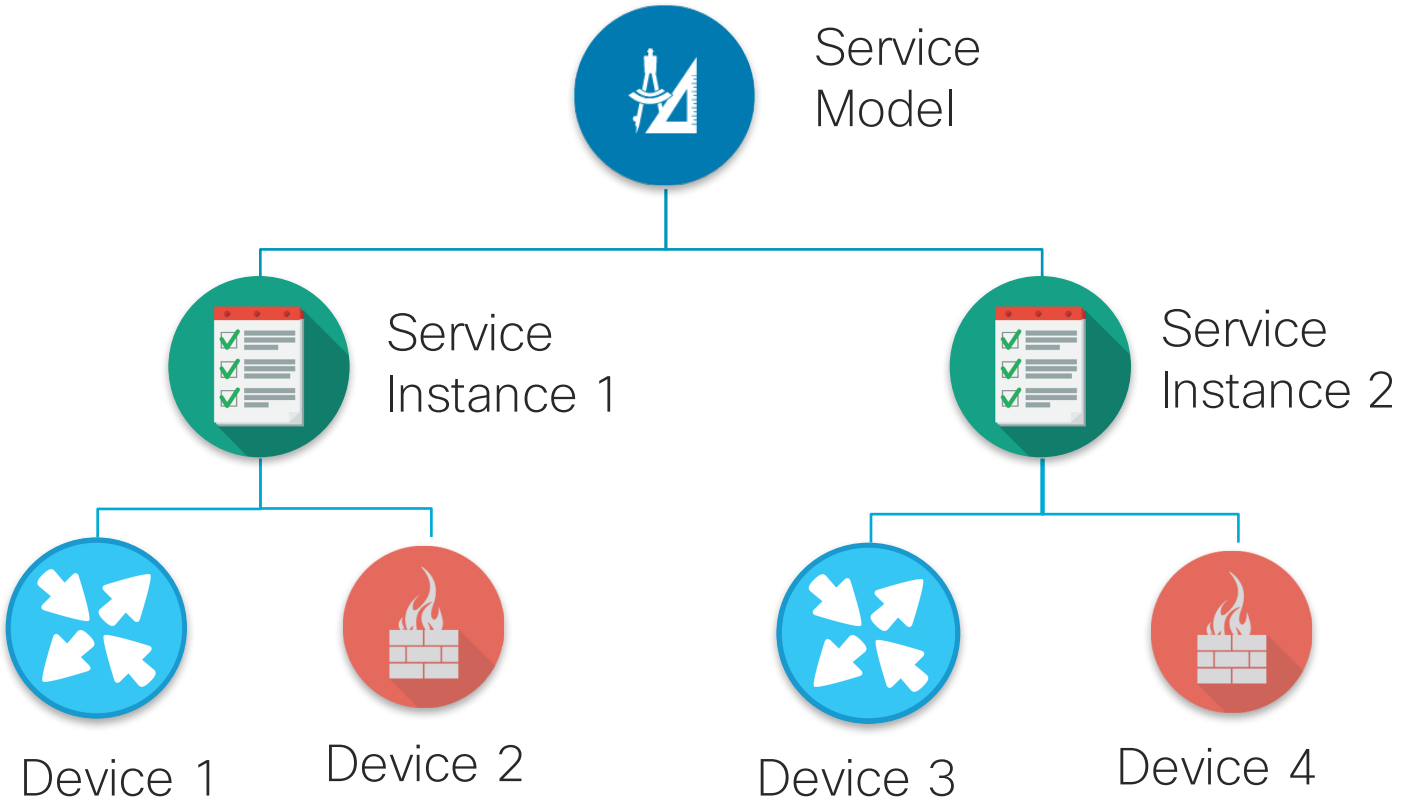
A Network Service is a collection of configurations across one or multiple devices and servers that enable a capability.

An example is Basic Wireless Service:

Why is the ‘S’ in NSO so useful? And what does it stand for?

NSO Service Manager & Models

NSO Manages Network Services through the Service Model Construct:



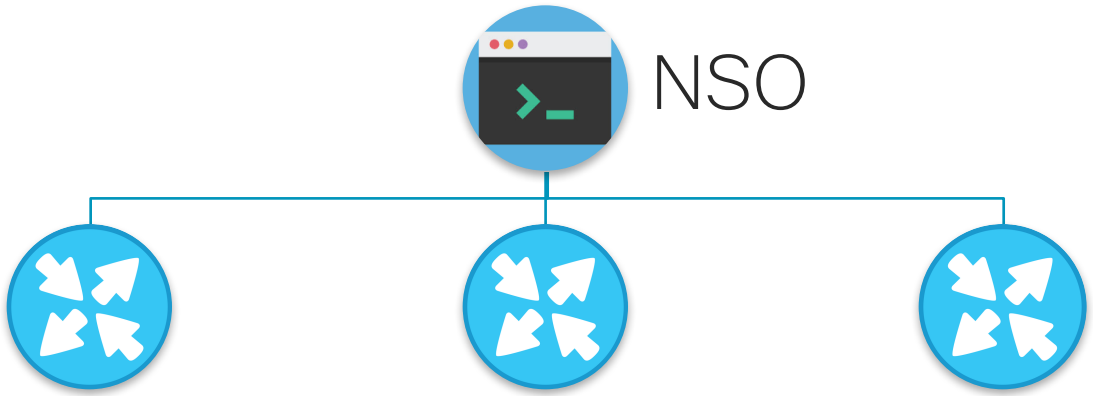
Designed in YANG and modeled with XML templates, code or both

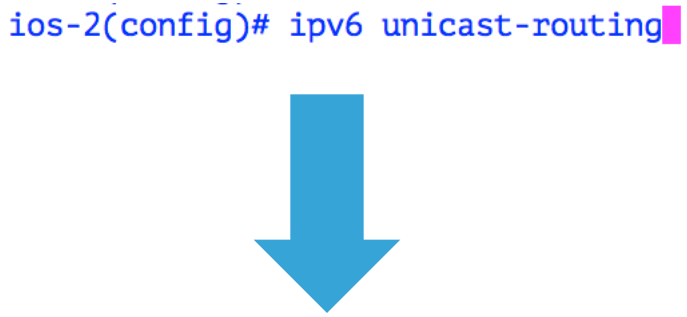
Instantiated in NSO with input parameters
Service's lifecycle is managed by NSO
(Deployment, Compliance and Removal)

NSO Pushes configuration based upon service logic
And input parameters to the devices.

NSO Device Manager

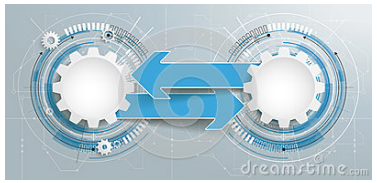
- Devices are added to NSO to be managed
- NSO Syncs the devices running config into the NSO config Database
- Devices can be bundled into groups and managed collectively
- Makes simple large scale changes trivial (example: ipv6 unicast-routing)





Network Element Drivers (NED)

```
<spanning-tree xmlns="urn:ios" >
        <extend>
                <system-id/>
        </extend>
        <mode>
                pvst
        </mode>
</spanning-tree>
```



```
spanning-tree extend system-id
spanning-tree mode pvst
```

Configuration Database

- Keeps copy of managed devices configuration
- Maintains Configurations for network services across instances and devices

Northbound APIs

REST API
GET,PUT,POST, DELETE
High Level API into:
NSO Service and Device Manager

Python API
Low Level and High Level APIs into:
NSO cDB
NSO Transaction Engine
NSO Service and Device Mangers

Java API
Low Level and High Level APIs into:
NSO cDB
NSO Transaction Engine
NSO Service and Device Mangers

Building Blocks of NSO Devices

- NSO can connect to any device assuming it has the IP, login credentials, NED (CLI type)
- NSO device names are independent of DNS lookups, the IP is the key piece of info to connect to the device.
- Devices can be in device groups, you can device groups of device groups
- To make changes to a device NSO needs a local copy of the Config and the device needs to be unlocked. By default a device is locked when added.

Building Blocks of NSO Devices

- Typical flow of adding a device:
  - Add required information for device.
  - Unlock device to be able to pull config (default is admin-state locked)
  - Commit changes to NSO
  - Fetch the ssh keys from the device
  - sync-from the device the config, so NSO has local copy of running-config
  - Admin-state lock the device

Building Blocks of NSO Devices

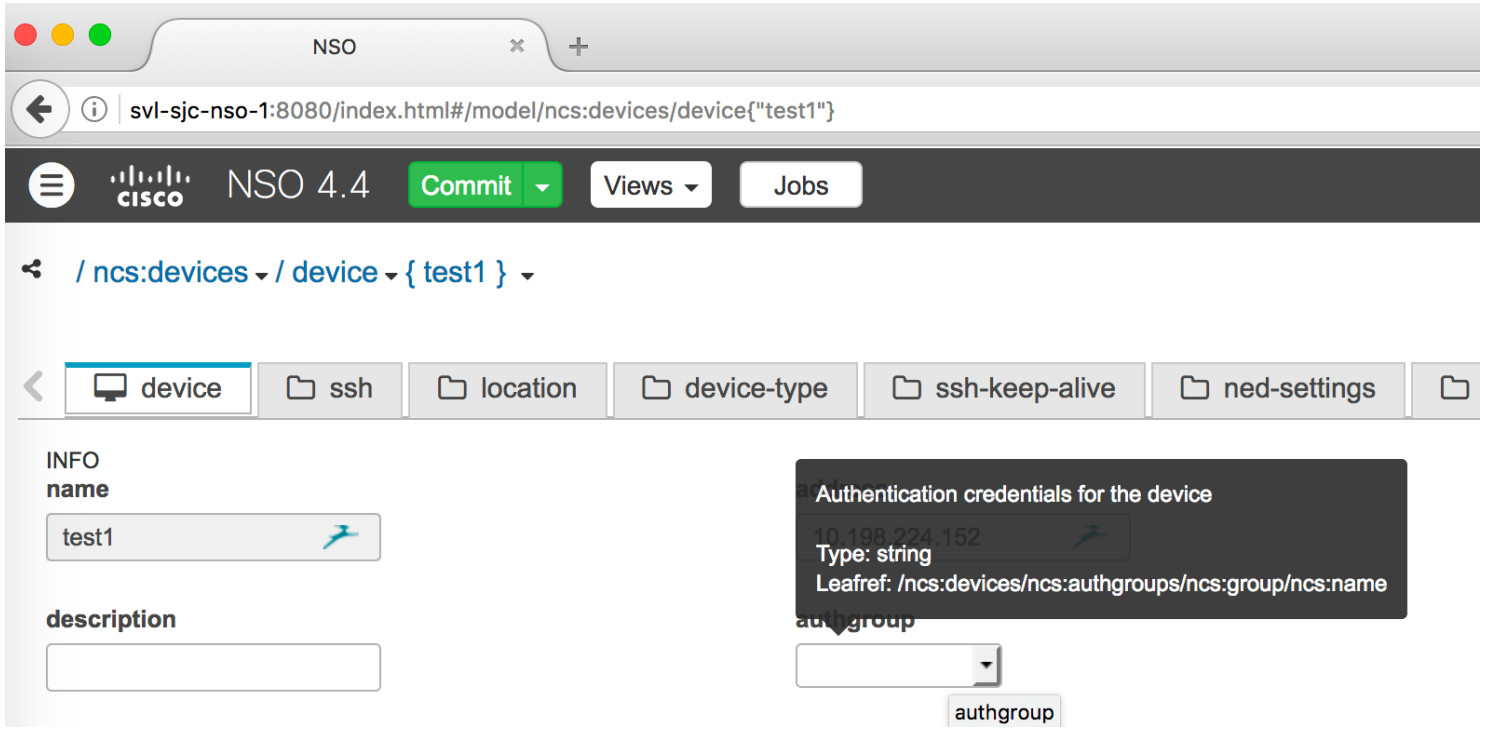
- Typical flow of configuring a device:
  - Enter Config Mode (‘config’)
  - Tell NSO which device or device-group to configure (‘devices device DEVICENAME config’)
  - Use the **NED prefix (‘ios:’)** for commands, **use ‘?’ to verify options**. Some commands have prefix, some don’t depending on how the data is structured in the XML. (’devices device DEVICENAME config ios:interface GigabitEthernet 1/47’)
  - **Don’t use shorthand commands** like gig 1/47 instead of GigabitEthernet 1/47. Use Tab completion instead.

Building Blocks of NSO CLI

- Everything is within a hierarchy
- Most common commands are under the ‘devices device’ or ‘devices …’ options.
- Use ‘?’ to explore options
- Use ’|’ to change output format, debug, or to save output to file. File will be saved to ncs-run directory.
- All changes in config mode need to be commited, all changes can be previewed with commit dry-run

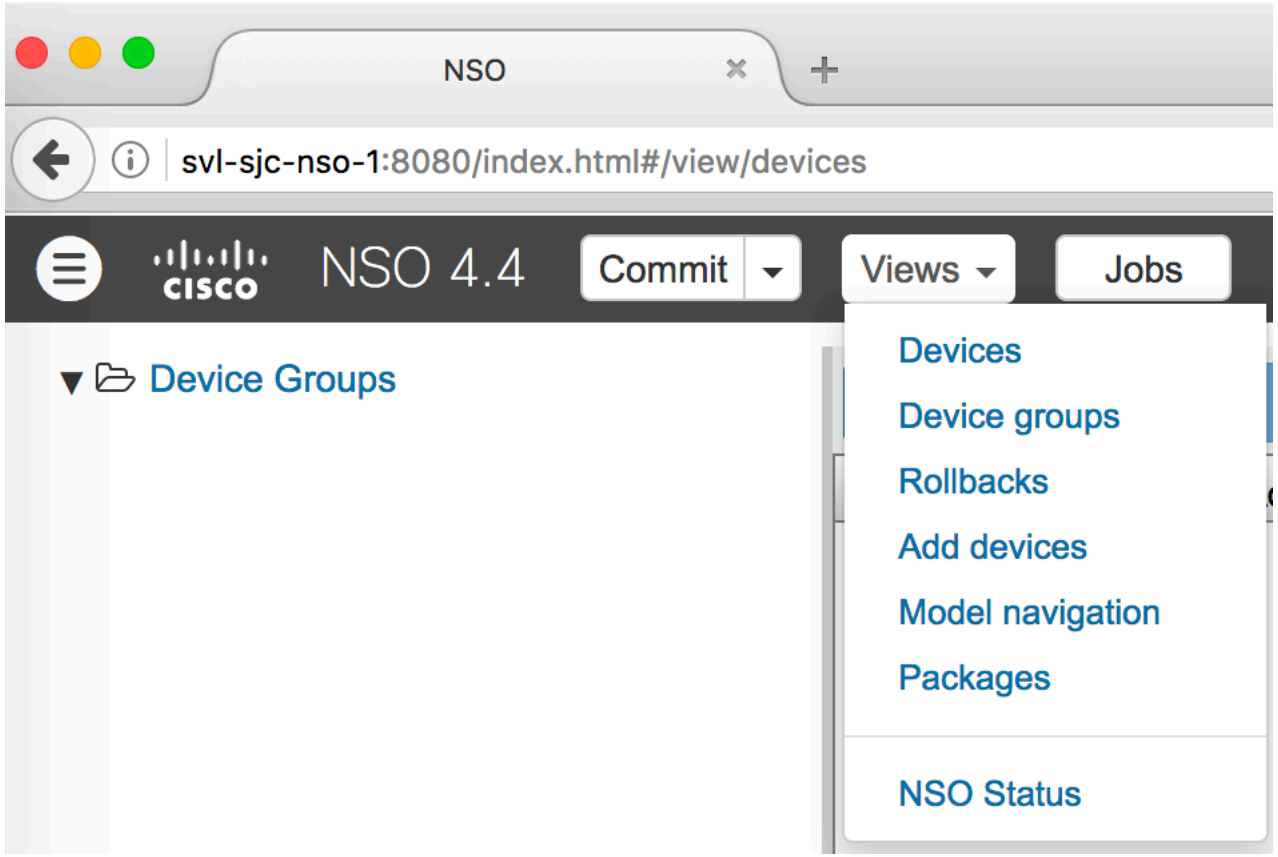
Demo NSO GUI

- Devices View
- Adding a Device
- Commit
- Commit Dry Run
- Alarms
- Rollback
- Xpath Mouse Hover



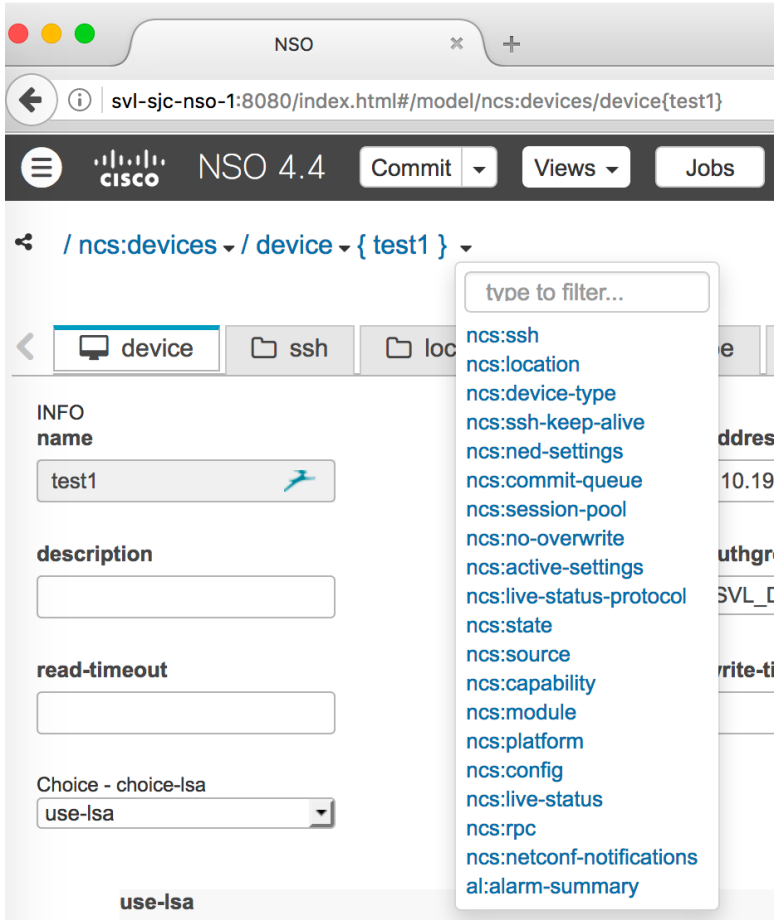
Building Blocks of NSO GUI

- Everything is within a hierarchy
- Views are good entry points to explore.
- Add devices View allows netsim devices to be added



NSO GUI Tips

- Use GUI for visualizing the CLI and understanding data model better
- Use arrows at end of breadcrumbs for more options
- Go to Devices, then click on the device name as a hyperlink to go to details



Lab 1 – Creating Your First Network

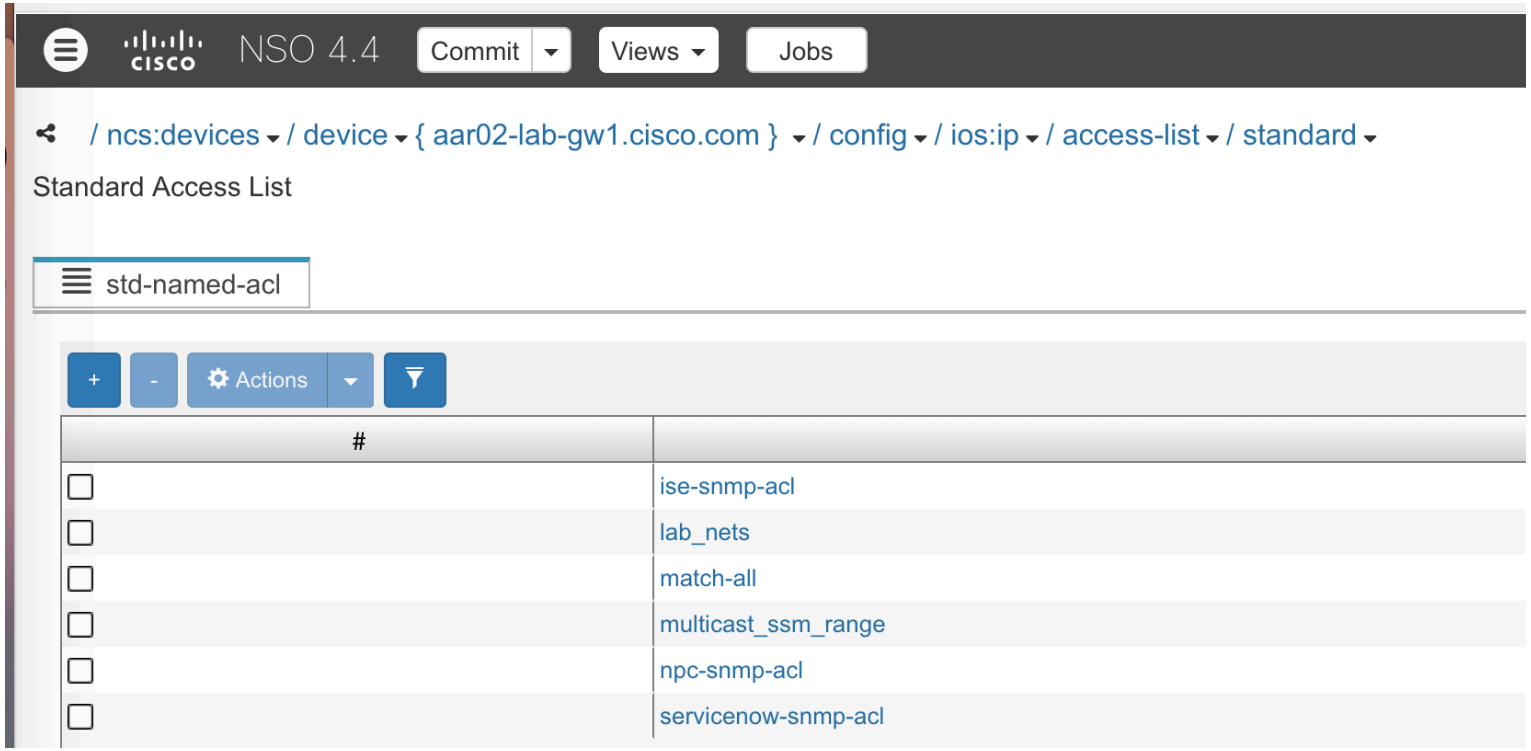

BRACE YOURSELF
LUNCH IS COMING
made on imgur

NSO Components: Device Manager

```
admin@ncs# show devices list
NAME                          ADDRESS                          DESCRIPTION  NED ID     ADMIN STATE
-------------------------------------------------------------------------------------------------
aaaaaaaaNETSIM                127.0.0.1                        -            cisco-ios  unlocked
aaaaaaaaa3850LABBY            rtp5-itnlab-3850-sw1.cisco.com   -            cisco-ios  unlocked
aar02-lab-gw1.cisco.com       10.49.95.153                     -            cisco-ios  locked
aar02-sw1.cisco.com           10.49.95.2                       -            cisco-ios  locked
aar02-wan-gw1.cisco.com       10.49.68.17                      -            cisco-ios  locked
```

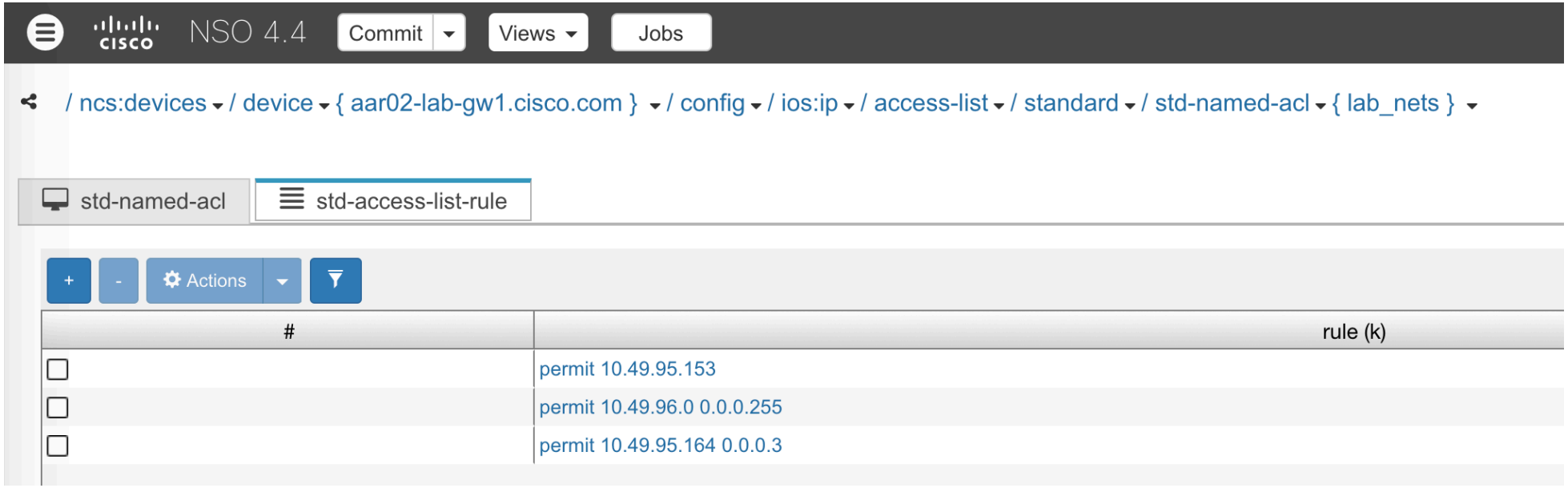
#	Name	Address	Port	Description	Authgroup	Oper State	Admin State
☐	aaaaaaaaNETSIM	127.0.0.1	10022		default	unknown	unlocked
☐	aaaaaaaaa3850LABBY	rtp5-itnlab-385(	22		jabelk.web.auth	unknown	unlocked
☐	aar02-lab-gw1.cisco.com	10.49.95.153	22		jabelk.web.auth	unknown	locked
☐	aar02-sw1.cisco.com	10.49.95.2	22		jabelk.web.auth	unknown	locked
☐	aar02-wan-gw1.cisco.com	10.49.68.17	22		jabelk.web.auth	unknown	locked

NSO Components: Config Management




```
admin@ncs# show running-config devices device aar02-lab-gw1.cisco.com config ios:ip access-list standard | include ios:
  ios:ip access-list standard ise-snmp-acl
  ios:ip access-list standard lab_nets
  ios:ip access-list standard match-all
  ios:ip access-list standard multicast_ssm_range
  ios:ip access-list standard npc-snmp-acl
  ios:ip access-list standard servicenow-snmp-acl
admin@ncs#
```

NSO Components: Config Management




```
admin@ncs# show running-config devices device aar02-lab-gw1.cisco.com config ios:ip access-list standard lab_nets
devices device aar02-lab-gw1.cisco.com
 config
  ios:ip access-list standard lab_nets
   permit 10.49.95.153
   permit 10.49.96.0 0.0.0.255
   permit 10.49.95.164 0.0.0.3
  !
 !
!
admin@ncs#
```

Individual or Group device Configuration

```
admin@ncs(config)# devices device aar02-lab-gw1.cisco.com config
admin@ncs(config-config)# ios:ip access-list standard lab_nets
admin@ncs(config-std-nacl)# permit 127.0.0.1
admin@ncs(config-std-nacl)# permit 172.0.0.2
admin@ncs(config-std-nacl)# commit dry-run outformat native
native {
    device {
        name aar02-lab-gw1.cisco.com
        data ip access-list standard lab_nets
              permit 127.0.0.1
              permit 172.0.0.2
             !
    }
}
admin@ncs(config-std-nacl)#
```

Static Templates

```
admin@ncs(config)# devices template "lab_nets template example"
admin@ncs(config-template-lab_nets template example)# config
admin@ncs(config-config)# ios:ip access-list standard std-named-acl lab_nets
admin@ncs(config-std-named-acl-lab_nets)# std-access-list-rule "permit 127.0.0.1"
admin@ncs(config-std-access-list-rule-permit 127.0.0.1)# exit
admin@ncs(config-std-named-acl-lab_nets)# std-access-list-rule "permit 127.0.0.2"
admin@ncs(config-std-access-list-rule-permit 127.0.0.2)# commit dry-run outformat native
native {
}
```

Static template creation

```
admin@ncs(config-std-access-list-rule-permit 127.0.0.2)# commit dry-run
cli {
    local-node {
        data  devices {
             +    template "lab_nets template example" {
             +        config {
             +            ios:ip {
             +                access-list {
             +                    standard {
             +                        std-named-acl lab_nets {
             +                            std-access-list-rule "permit 127.0.0.1";
             +                            std-access-list-rule "permit 127.0.0.2";
             +                        }
             +                    }
             +                }
             +            }
             +        }
             +    }
              }
    }
}
admin@ncs(config-std-access-list-rule-permit 127.0.0.2)#
```

Static template applied to a device

```
admin@ncs(config)# devices device aar02-lab-gw1.cisco.com apply-template template-name ?
Possible completions:
  lab_nets template example
admin@ncs(config)# devices device aar02-lab-gw1.cisco.com apply-template template-name lab_nets\ template\ example
apply-template-result {
    device aar02-lab-gw1.cisco.com
    result ok
}
admin@ncs(config)# commit dry-run outformat native
native {
    device {
        name aar02-lab-gw1.cisco.com
        data ip access-list standard lab_nets
              permit 127.0.0.1
              permit 127.0.0.2
             !
    }
}
admin@ncs(config)#
```

Template with Variables

Template with variables

```
admin@ncs(config)# devices template template_with_vars
admin@ncs(config-template-template_with_vars)# config
admin@ncs(config-config)# ios:ip access-list standard std-named-acl {$LIST-NAME}
admin@ncs(config-std-named-acl-{$LIST-NAME})# std-access-list-rule {$ACL-RULE}
admin@ncs(config-std-access-list-rule-{$ACL-RULE})# exit
admin@ncs(config-std-named-acl-{$LIST-NAME})# std-access-list-rule {$ACL-RULE-2}
admin@ncs(config-std-access-list-rule-{$ACL-RULE-2})# commit
Commit complete.
admin@ncs(config-std-access-list-rule-{$ACL-RULE-2})#
```

Original Static template without variables

```
admin@ncs(config)# devices template "lab_nets template example"
admin@ncs(config-template-lab_nets template example)# config
admin@ncs(config-config)# ios:ip access-list standard std-named-acl lab_nets
admin@ncs(config-std-named-acl-lab_nets)# std-access-list-rule "permit 127.0.0.1"
admin@ncs(config-std-access-list-rule-permit 127.0.0.1)# exit
admin@ncs(config-std-named-acl-lab_nets)# std-access-list-rule "permit 127.0.0.2"
admin@ncs(config-std-access-list-rule-permit 127.0.0.2)# commit dry-run outformat native
native {
}
```

- Variable using syntax {$somename} (upper or lower case
- In this case list-name replaces lab_nets
- Acl-rule replaces “permit 127.0.0.1”
- Acl-rule-2 replaces “permit 127.0.0.2”
- Note that NSO IOS NED represents the ACL syntax as an entire string unit (includes permit + rule within the variable, for example, Nexus NED has separate values for each part of the rule

Template with variables application

```
admin@ncs(config-device-aar02-lab-gw1.cisco.com)# apply-template template-name template_with_var
variable { name LIST-NAME value 'lab_nets' } variable { name ACL-RULE value 'permit\ 127.0.0.1'
variable { name ACL-RULE-2 value 'permit\ 127.0.0.2' }
apply-template-result {
    device aar02-lab-gw1.cisco.com
    result ok
}
admin@ncs(config-device-aar02-lab-gw1.cisco.com)# commit dry-run outformat native
native {
    device {
        name aar02-lab-gw1.cisco.com
        data ip access-list standard lab_nets
              permit 127.0.0.1
              permit 127.0.0.2
             !
    }
}
admin@ncs(config-device-aar02-lab-gw1.cisco.com)#
```

- Note that the variables are inclosed in single quotes and words with spaces need an escape character '\'

Check Sync and out of band changes

```
admin@ncs# devices device rtp5-itnlab-dt-sw1.cisco.com check-sync
result in-sync
```

Now I logged in a separate ssh session and made a change (without NSO knowing it)

```
rtp5-itnlab-dt-sw1(config-if)#int gigabitEthernet 1/5
rtp5-itnlab-dt-sw1(config-if)#desc
rtp5-itnlab-dt-sw1(config-if)#description automation_test
rtp5-itnlab-dt-sw1(config-if)#end
rtp5-itnlab-dt-sw1#wr
Building configuration...
Compressed configuration from 138460 bytes to 37538 bytes[OK]
rtp5-itnlab-dt-sw1#
```

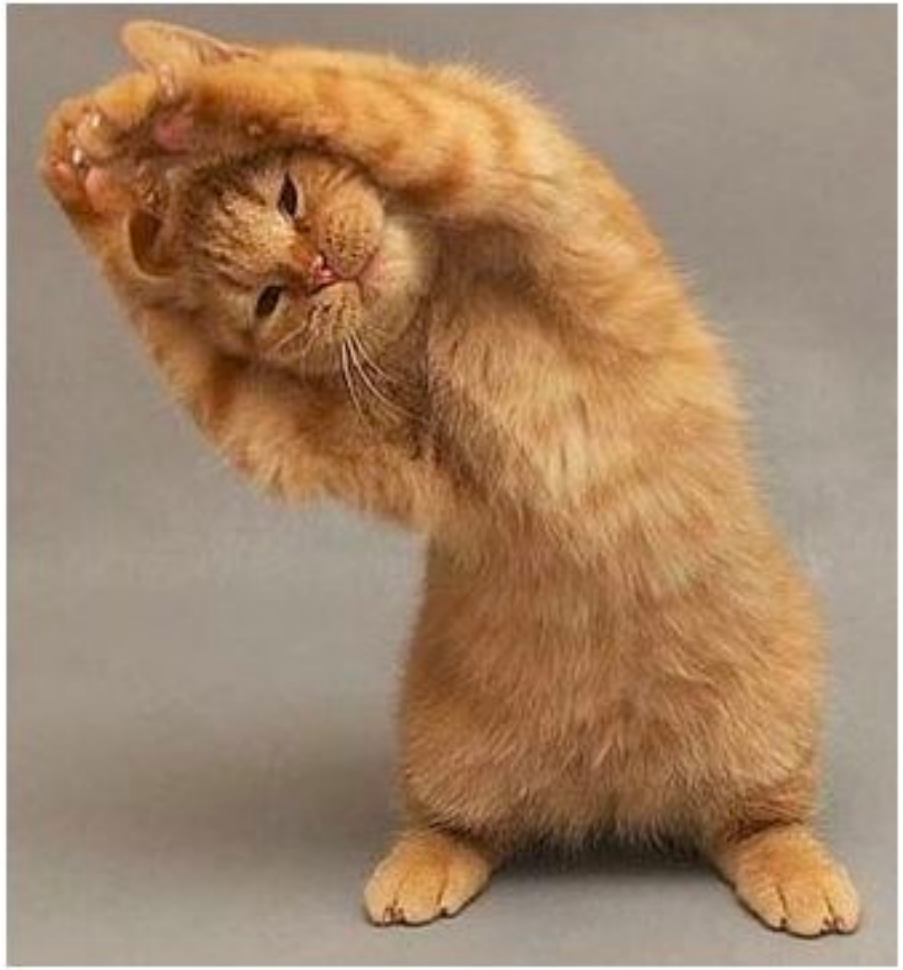
Check-Sync/Compare-Config with difference

```
admin@ncs# devices device rtp5-itnlab-dt-sw1.cisco.com check-sync
result out-of-sync
info got: 9effb7a4329ebeb30f95cb91f46d7026 expected: 710cd4880d4214b48de34f937ff5ec02

admin@ncs# *** ALARM out-of-sync: got: 9effb7a4329ebeb30f95cb91f46d7026 expected: 710cd4880d4214b48de34f937ff5ec02

admin@ncs# devices device rtp5-itnlab-dt-sw1.cisco.com compare-config
diff
 devices {
     device rtp5-itnlab-dt-sw1.cisco.com {
         config {
             ios:interface {
                 GigabitEthernet 1/5 {
-                    description "applying demosite acl";
+                    description automation_test;
                 }
             }
         }
     }
 }
```

Use Sync-from to make local NSO DB in sync

```
admin@ncs# devices device rtp5-itnlab-dt-sw1.cisco.com sync-from
result true
admin@ncs# devices device rtp5-itnlab-dt-sw1.cisco.com check-sync
result in-sync
admin@ncs#
```

Stretch Break!



Device Groups, Templates and Rollback

- A device can be added to one or many device group(s), a device group can be nested into another device group
- Templates can either have static configs enforced every time, or variables put in to be declared at the time of usage. Service templates will be covered later.
- You can rollback any change, or part of a change, in NSO through the Rollback in GUI or on CLI (config -> rollback)

Lab 2 – Configuring Device IOS Config & Templates & Device Groups

Show Commands

```
admin@ncs# devices device rtp5-itnlab-dt-sw1.cisco.com live-status ios-stats:exec ?
Possible completions:
  any           Execute any command on device
  clear         Reset functions
  copy          Copy from one file to another
  license       Smart licensing Commands
  ping          Send echo messages
  reload        Halt and perform a cold restart
  show          Execute show commands
  traceroute    Trace route to destination
  verify        Verify a file
```

```
admin@ncs# devices device rtp5-itnlab-dt-sw1.cisco.com live-status ios-stats:exec any ?
Possible completions:
  WORD    any "<cmd> [option(s)]", e.g: any "show ppp summary"
  |       Output modifiers
  <cr>
```

```
admin@ncs# devices device rtp5-itnlab-dt-sw1.cisco.com live-status ios-stats:exec any "show ip int br"
result
> show ip int br
Interface              IP-Address      OK? Method Status                Protocol
FastEthernet1          unassigned      YES unset  down                  down
GigabitEthernet1/1     unassigned      YES unset  down                  down
GigabitEthernet1/2     unassigned      YES unset  down                  down
GigabitEthernet1/3     unassigned      YES unset  down                  down
```

Lab

Open Discussion – Feedback, Demos, Deep Dives?