CISCO Connect Phoenix

October 17, 2019

You make **possible**

cisco Connect Phoenix

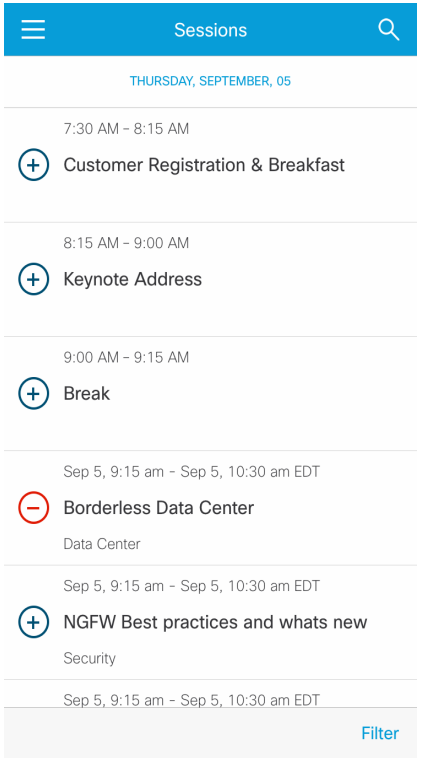
Getting Started with DC Network Automation

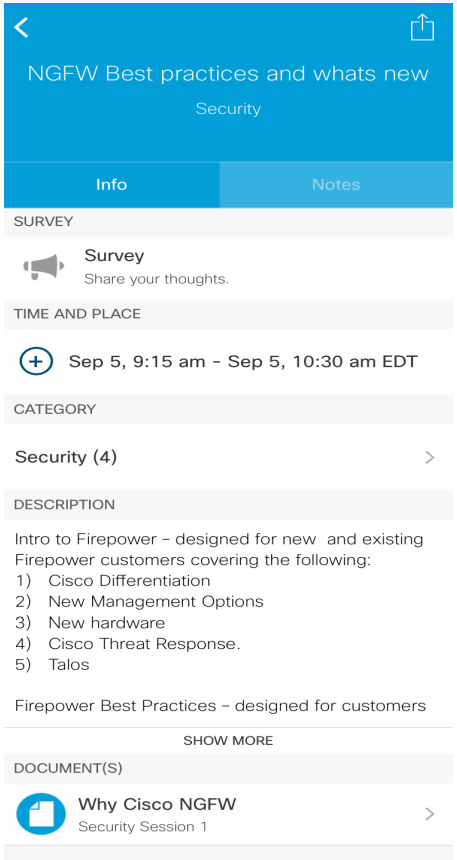
Taking the Fear Out of NetDevOps

Quinn Snyder
Janitor
October 17, 2019

Thank You, Sponsors

Session Presentations





Disclaimer

I'm not an expert. My code will reflect this. There's lots of things "not to do" in there.

You won't be an expert after sitting through this – its all about muscle memory. Practice!

I'm not advocating for one technology over another, just demonstrating simple ways to get started

What Do I Bring to this Presentation?

- 2.5 years at Cisco
  - Working on driving automation within my customers (Ansible, NSO, Puppet, etc)
  - Always around for DevNet
- 10.5 years at Cisco partner
  - Post-sales practice management around SP, NFV, automation, NetDevOps
  - Mostly focused on open-source toolchains and non-Cisco hardware

Quinn Snyder

Systems Architect

qsnyder@cisco.com

http://github.com/qsnyder

http://www.linkedin.com/in/qsnyder

http://www.twitter.com/qsnyder

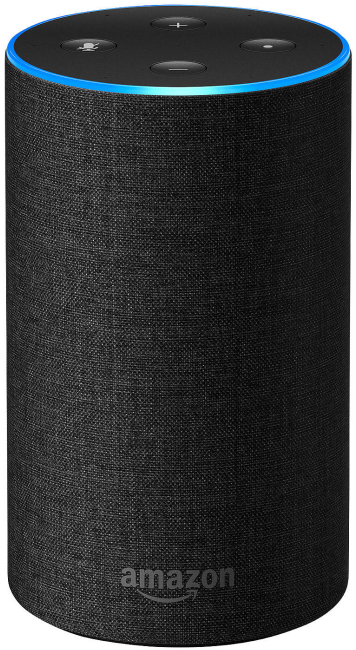
Rough Agenda
(subject to change)

1. Why Automate?
2. Review (RESTful) APIs
3. Vagrant anyone?
4. Vagrant and NXAPI Sandbox *(DEMO)*
5. Review Ansible concepts
6. Ansible with Jinja2 for template creation *(DEMO)*
7. Using Ansible to automate Nexus configurations *(DEMO)*
8. Moving beyond Ansible
9. Automation Nirvana *(DEMO)*

Personal Automation Journey

(not related to networks)


POWER
TV
P.STD
OK
MENU
VOLUME
CHANNEL
DISPLAY
TV/VIDEO
POWER



Automation is a journey. Don't let your vision of Alexa stop you from programming that remote.

The Problem with Learning Automation

This is a lot of code to see if BGP is configured


```
#!/usr/bin/env python

from datetime import datetime

from netmiko import ConnectHandler
from my_devices import cisco_ios, cisco_xr, arista_veos


def check_bgp(net_connect, cmd='show run | inc router bgp'):
    """Check whether BGP is currently configured on device. Return boolean"""
    output = net_connect.send_command_expect(cmd)
    return 'bgp' in output

def main():
    device_list = [cisco_ios, cisco_xr, arista_veos]
    start_time = datetime.now()
    print

    for a_device in device_list:
        net_connect = ConnectHandler(**a_device)
        net_connect.enable()
        print "{}: {}".format(net_connect.device_type, net_connect.find_prompt())
        if check_bgp(net_connect):
            print "BGP currently configured"
        else:
            print "No BGP"
        print

    print "Time elapsed: {}\n".format(datetime.now() - start_time)


if __name__ == "__main__":
    main()
```

“I’m pretty sure you can’t just buy quantity four of ‘automations’ from your vendor…”*

**Cisco NSO excepted (more on that later)*

Definitions

(just so we're on the same page)

Automation

the act of removing human action in a single task or process

Orchestration

the act of chaining several automated tasks together in order to form a process or workflow that does not require human interaction

APIs

REST Web Service

What is REST?

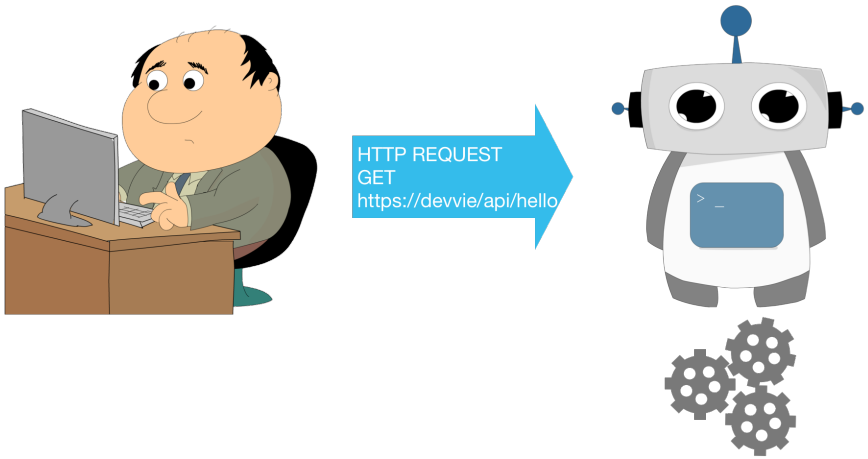
- REpresentational State Transfer
- API framework built on HTTP
- Stateless

What is a REST Web Service?

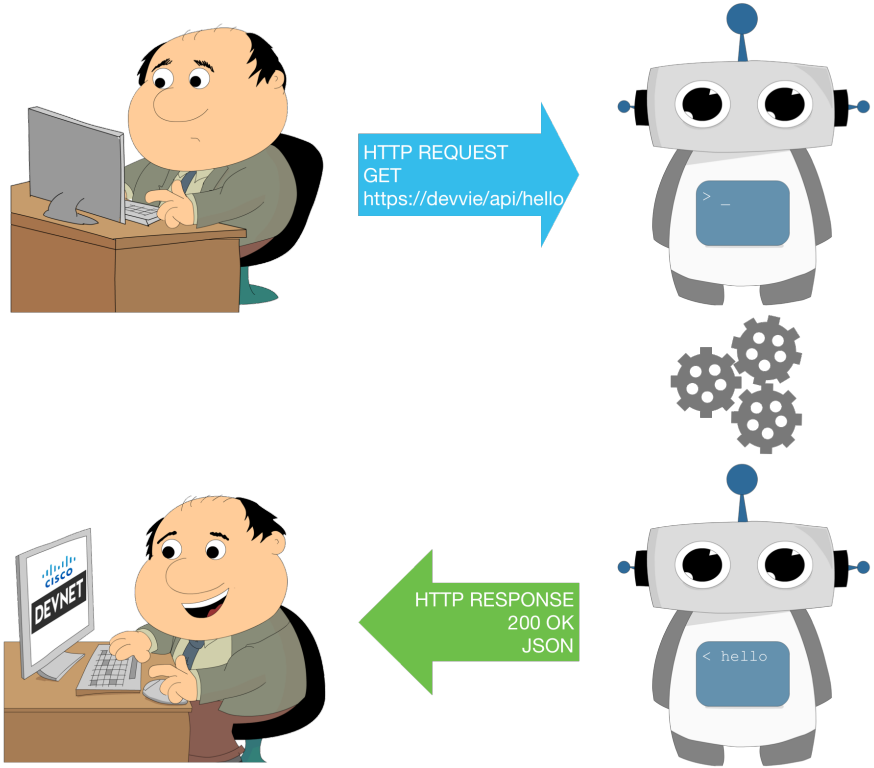
- REST is an architecture style for designing networked applications.
- Popular due to performance, scale, simplicity, and reliability

The REST API Flow

The REST API Flow



The REST API Flow



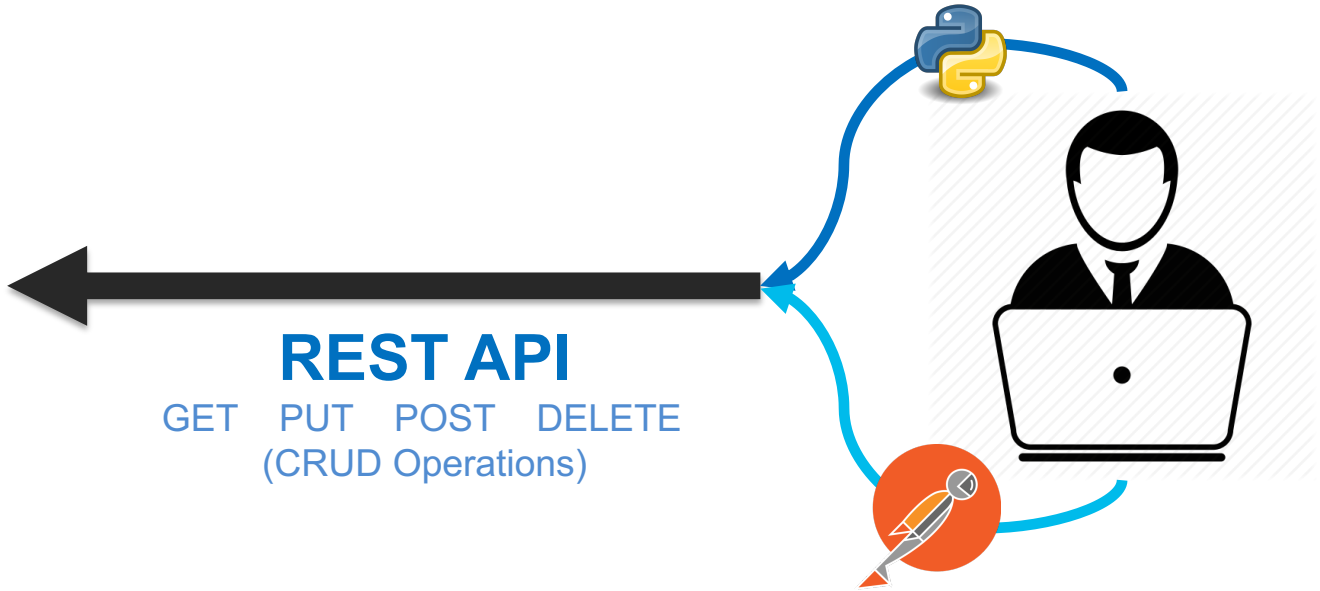
REST

Transfer service that allows for sending of data in structured format (JSON, XML, etc) towards a URI

RESTCONF

REST-like protocol used to access YANG datastores defined in NETCONF over HTTP

Managing Nexus Switches With NXAPI



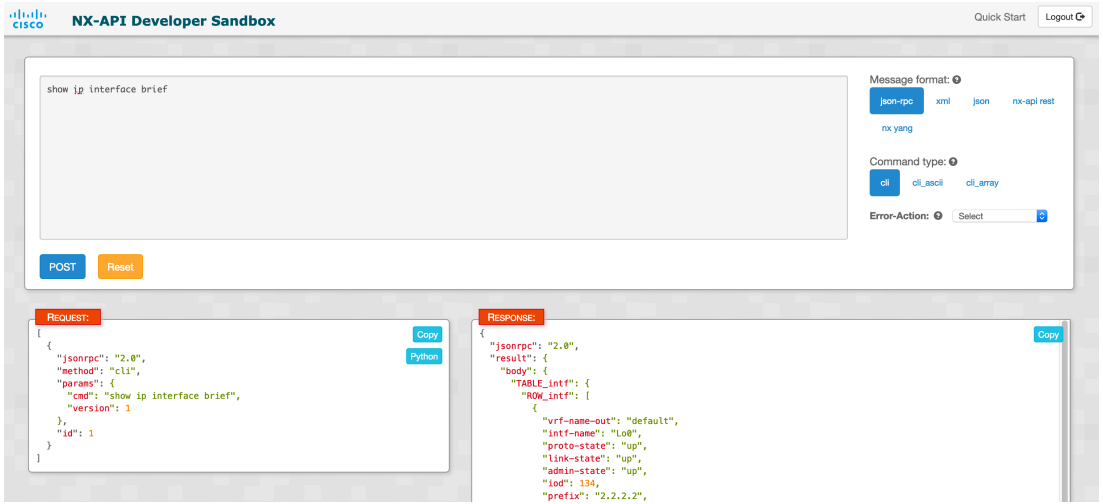
...use the sandbox to explore...

NXOS APIs

- NXOS supports full RESTful API for config and operational data
- Accessible through in-band or mgmt0 interface
- Supported on N3K, N5K, N7K, and N9K
    - Code restrictions apply

There's even a sandbox to explore the API

All changes are made after POST is made!



So how do I get started…?

Accessible Dev Environments

- Vagrant
- VIRL/CML
- ncs-netsim
  (more on this later)

“There’s no sense in being precise when you don’t even know what you’re talking about”

~ John von Neumann

We Can All Be Vagrants

VM created with all my tools. I used Fusion 10, Ubuntu guest, bridged networking. Works great!

10GB OVA? This will take forever.

And the bridged networking doesn't work, which breaks half of the tools.

We've all been here.

```
apt-get install
http://cyberelk.net/tim/data/xmlto/

Err http://cyberelk.net/tim/data/xmlto/
Could not open file - open
-  (2 No such file or directory)
-  Failed to fetch
   http://cyberelk.net/tim/data/xmlto/
   Could not open file - open
-  (2 No such file or directory)
-  E: Failed to fetch some archives.
Welcome to dependency hell.

#       ###### #       ###### ###
#       #     # #           #  ###
#       #     # #          #   ###
#       #     # #         #     #
#       #     # #        #
#       #     # #       #      ###
####### ###### ####### ###### ###
```

I'll make my own VM. With all the tools he added, and it will work!

We've all been here, too

There’s a
better way...

Enter Vagrant

- Define a standard build for all machines, using provisioners for customization
- Simple CLI for machine/network instantiation
- Uses standard virtualization applications (VBox, KVM, etc)
- Open-source
- Network and server VMs
- Easy to use

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

Vagrant.configure("2") do |config|

  config.vm.define "n9kv1" do |n9kv1|
    n9kv1.vm.box = "nxos/7.0.3.I7.2" #change to your own box name on import
    n9kv1.ssh.insert_key = false
    n9kv1.vm.boot_timeout = 600
    n9kv1.vm.synced_folder '.', '/vagrant', disabled: true
    n9kv1.vm.network "forwarded_port", guest: 8080, host: 8080
    n9kv1.vm.network "forwarded_port", guest: 8443, host: 8443
    n9kv1.vm.network "private_network", ip: "1.1.1.1", virtualbox__intnet: "vagrant-net", auto_config: false
    n9kv1.vm.provider "virtualbox" do |vb|
     vb.memory = "4096"
     vb.customize ['modifyvm',:id,'--nicpromisc2','allow-all']
     end
    end

   config.vm.define "n9kv2" do |n9kv2|
    n9kv2.vm.box = "nxos/7.0.3.I7.2"
    n9kv2.ssh.insert_key = false
    n9kv2.vm.boot_timeout = 600
    n9kv2.vm.synced_folder '.', '/vagrant', disabled: true
    n9kv2.vm.network "forwarded_port", guest: 8081, host: 8081
    n9kv2.vm.network "forwarded_port", guest: 8444, host: 8444
    n9kv2.vm.network "private_network", ip: "1.1.1.2", virtualbox__intnet: "vagrant-net", auto_config: false
    n9kv2.vm.provider "virtualbox" do |vb|
      vb.memory = "4096"
      vb.customize ['modifyvm',:id,'--nicpromisc2','allow-all']
      end
    end
  end
```

Vagrantfiles are written in Ruby – but you don't need to learn the language

Vagrant CLI

Tool	Description
`destroy`	*stops and deletes all traces of the vagrant machine*
`halt`	*stops the vagrant machine*
`init`	*initializes a new Vagrant environment by creating a Vagrantfile*
`reload`	*restarts vagrant machine, loads new Vagrantfile configuration*
`resume`	*resume a suspended vagrant machine*
`ssh`	*connects to machine via SSH*
`suspend`	*suspends the machine*
`up`	*starts and provisions the vagrant environment*

Software Download

Downloads Home / Switches / Data Center Switches / Nexus 9000 Series Switches / Nexus 9000v Switch / NX-OS System Software- 9.2(2)

Search...

Expand All | Collapse All

Latest Release

9.2(2)

7.0(3)I7(5a)

All Release

9

7

Nexus 9000v Switch

Release 9.2(2)

Notifications

Related Links and Documentation

Release Notes for 9.2(2) N3K

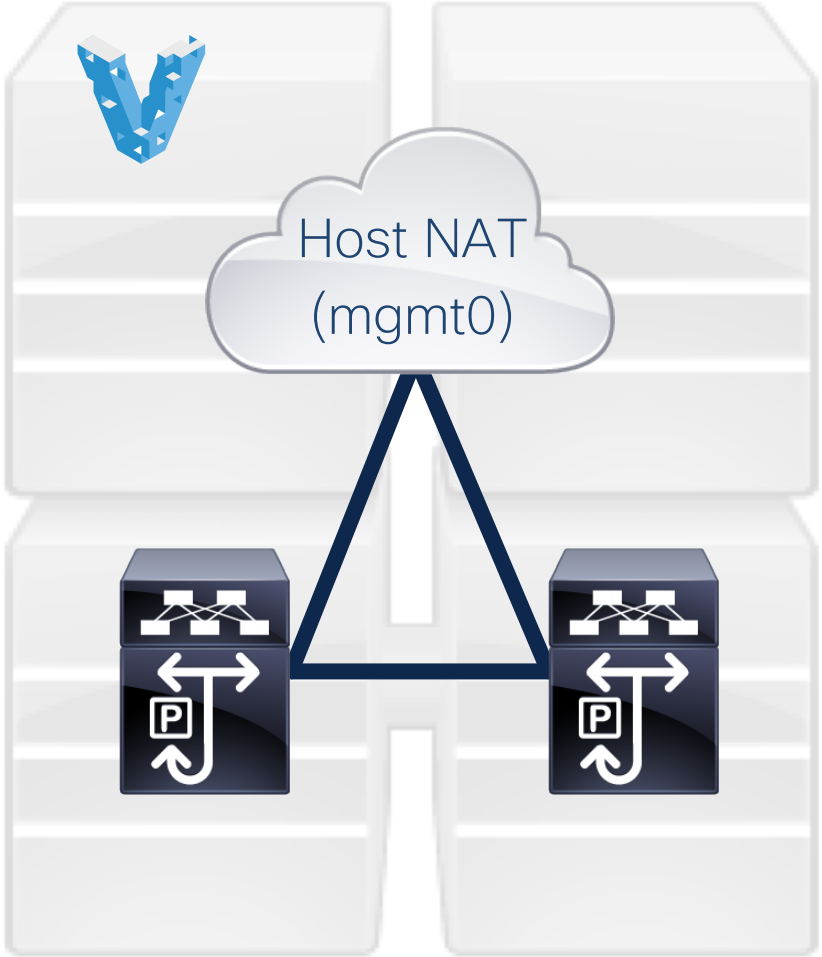
Release Notes for 9.2(2) N9K

Cisco Nexus 9000/3000 Virtual Switch for Vagrant nxosv.9.2.2.box	5-Nov-2018	1282.33 MB
Cisco Nexus 9000/3000 Virtual Switch for ESXi or Fusion nxosv.9.2.2.ova	5-Nov-2018	1273.05 MB
Cisco Nexus 9000/3000 Virtual Switch for KVM nxosv.9.2.2.qcow2	5-Nov-2018	1281.81 MB
Cisco Nexus 9000/3000 Virtual Switch for ESXi or Fusion or Vagrant nxosv.9.2.2.vmdk	5-Nov-2018	1281.63 MB

Cisco packages boxes!

Additional Benefits

- Because of the packaging (.box) and the provisioners – resulting image will always be the same
- Synched folders allow file movement between host and guest easily
- Cross-platform support (Windows, macOS, Linux), as well as cloud (AWS, etc)
- Ability to define “multi-machine” environments in single Vagrantfile



The environment doesn't need to be complex.

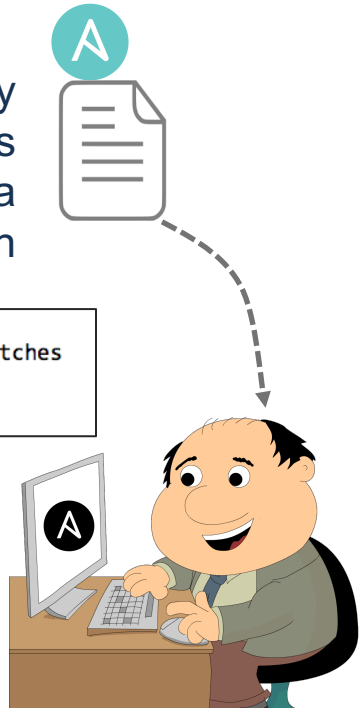
NXAPI and Postman Demo within Vagrant *(DEMO)*

Ansible and Ansibility

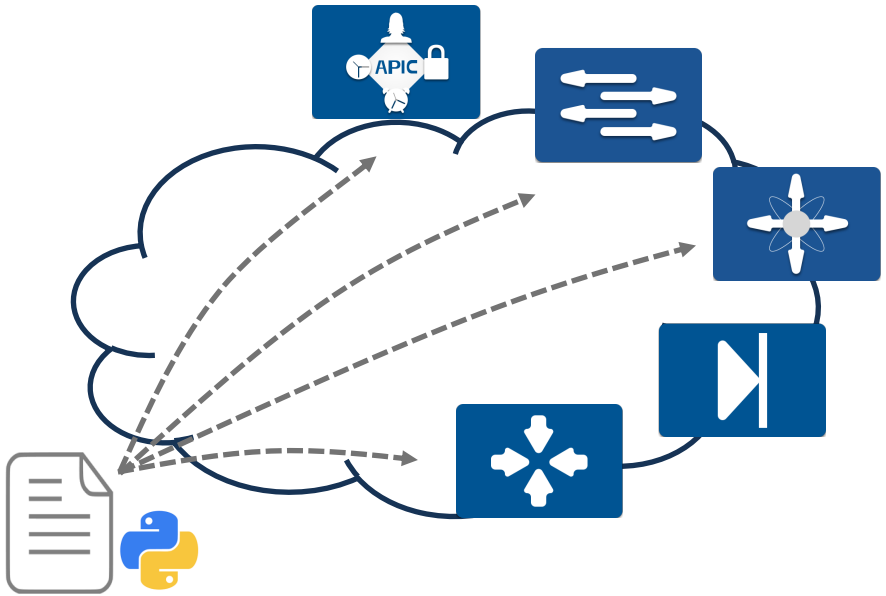
Ansible and Networking

1. Engineers deploy Ansible playbooks written in YAML to a control station

```
---
- name: Retrieve facts from Switches
  hosts: switches
  connection: local
```



Ansible Control Station



2. Ansible executes modules locally using APIs (or SSH) to interface with devices (different than server admin)

Inside the Ansible Control Station

- Linux (or Mac) host with Python and Ansible installed
- Support transport to remote hosts
  - Traditionally done with SSH
  - New modules can leverage API
- Ansible Components
  - Ansible configuration file
  - Inventory files
  - Ansible modules
  - Playbooks

YAML Overview

- What is YAML?
  - “YAML Ain’t Markup Language”
  - YAML is a human readable data serialization language
    - YAML files are easily parsed into software data structures
  - YAML is a common basis for many domain specific languages
    - Ansible
    - Heat
    - Saltstack
    - docker-compose
    - Nornir (Python)
    - pyATS/Genie
    - cloud-init
    - Many more!

Ansible Terms

Tool	Description
`module`	Code, typically written in Python, that will perform some action on a host. *Example: yum - Manages packages with the yum package manager*
`task`	A single action that references a module to run along with any input arguments and actions
`play`	Matching a set of tasks to a host or group of hosts
`playbook`	A YAML file that includes one or more plays
`role`	A pre-built set of playbooks designed to perform some standard configuration in a repeatable fashion. A play could leverage a role rather than tasks. *Example: A role to configure a web server would install Apache, configure the firewall, and copy application files.*

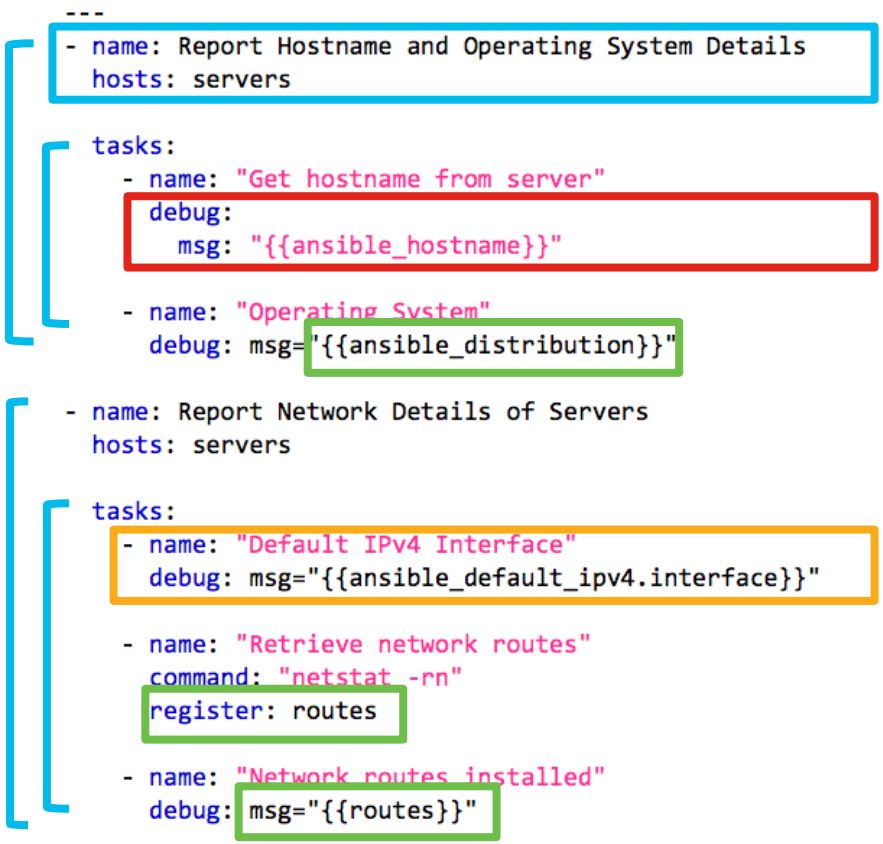
http://docs.ansible.com/ansible/latest/list_of_all_modules.html
http://docs.ansible.com/ansible/latest/playbooks.html

Ansible CLI Tool Overview

Tool	Description
`ansible`	Executes modules against targeted hosts without creating playbooks.
`ansible-playbook`	Run playbooks against targeted hosts.
`ansible-vault`	Encrypt sensitive data into an encrypted YAML file.
`ansible-pull`	Reverses the normal “push” model and lets clients "pull" from a centralized server for execution.
`ansible-docs`	Parses the docstrings of Ansible modules to see example syntax and the parameters modules require.
`ansible-galaxy`	Creates or downloads roles from the Ansible community.

Ansible Playbooks

- Written in YAML
- One or more plays that contain hosts and tasks
- Tasks have a name & module keys.
- Modules have parameters
- Variables referenced with {{name}}
  - Ansible gathers “facts”*
  - Create your own by `register`-ing output from another task



http://docs.ansible.com/ansible/latest/YAMLSyntax.html

Ansible Playbooks

```
DevNet$ ansible-playbook -u root example1.yaml

PLAY [Report Hostname and Operating System Details]
*********************************************************************************************
TASK [Gathering Facts]
*********************************************************************************************
ok: [10.10.20.20]

TASK [Get hostname from server]
*********************************************************************************************
ok: [10.10.20.20] => {
    "msg": "localhost"
}

PLAY [Report Network Details of Servers]
*********************************************************************************************
TASK [Network routes installed]
*********************************************************************************************
ok: [10.10.20.20] => {
        "stdout_lines": [
            "Kernel IP routing table",
            "Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface",
            "0.0.0.0         10.10.20.254    0.0.0.0         UG        0 0          0 ens160",
            "10.10.20.0      0.0.0.0         255.255.255.0   U         0 0          0 ens160",
            "172.16.30.0     10.10.20.160    255.255.255.0   UG        0 0          0 ens160",
        ]

PLAY RECAP
*********************************************************************************************
*******************
10.10.20.20                : ok=7    changed=1    unreachable=0    failed=0
```

Jinja2 Templating – Build Configs the Easy Way!

- Not just for Ansible templates
- Powerful templating language
  - Loops, conditionals and more supported
- Leverage `template` module
  - Attributes
    - `src:` The template file
    - `dest:` Where to save generated template

example3.j2

```
feature {{feature}}
router bgp {{asn}}
  router-id {{router_id}}
```

```
---
- name: Generate Configuration from Template
  hosts: localhost
  gather_facts: false
  vars:
    feature: bgp
    asn: 65001
    router_id: 10.10.10.1

  tasks:
    - name: "Generate config"
      template:
        src: "example3.j2"
        dest: "./example3.conf"
```

http://docs.ansible.com/ansible/latest/playbooks_templating.html

Jinja2 Templating – Build Configs the Easy Way!

```
DevNet$ ansible-playbook -u root example3.yaml

PLAY [Generate Configuration from Template] ********************************

TASK [Generate config] *****************************************************
changed: [localhost]

PLAY RECAP *****************************************************************
localhost                  : ok=1    changed=1    unreachable=0    failed=0

DevNet$ cat example3.conf
feature bgp
router bgp 65001
  router-id 10.10.10.1
```

Modules exist for almost every vendor

Nxos

- nxos_aaa_server - Manages AAA server global configuration.
- nxos_aaa_server_host - Manages AAA server host-specific configuration.
- nxos_acl - Manages access list entries for ACLs.
- nxos_acl_interface - Manages applying ACLs to interfaces.
- nxos_banner - Manage multiline banners on Cisco NXOS devices
- nxos_bgp - Manages BGP configuration.
- nxos_bgp_af - Manages BGP Address-family configuration.
- nxos_bgp_neighbor - Manages BGP neighbors configurations.
- nxos_bgp_neighbor_af - Manages BGP address-family's neighbors configuration.
- nxos_command - Run arbitrary command on Cisco NXOS devices
- nxos_config - Manage Cisco NXOS configuration sections
- nxos_evpn_global - Handles the EVPN control plane for VXLAN.
- nxos_evpn_vni - Manages Cisco EVPN VXLAN Network Identifier (VNI).
- nxos_facts - Gets facts about NX-OS switches

- (68) modules for NXOS
- (8) modules for IOS-XR
- (11) modules for IOS
- (3) modules for ASA
- (32) modules for ACI
- (2) modules for AireOS

Ansible NX-OS Integration

- NX-OS modules included in Ansible install
  - Maintained by Ansible Network Team
  - Robust module list across features
- Transport API Options
  - `network_cli` – default*
  - `nxapi`
- To Use NX-API must enable feature

Nxos

- nxos_aaa_server - Manages AAA server global configuration.
- nxos_aaa_server_host - Manages AAA server host-specific configuration.
- nxos_acl - Manages access list entries for ACLs.
- nxos_acl_interface - Manages applying ACLs to interfaces.
- nxos_bgp - Manages BGP configuration.
- nxos_bgp_af - Manages BGP Address-family configuration.
- nxos_bgp_neighbor - Manages BGP neighbors configurations.
- nxos_bgp_neighbor_af - Manages BGP address-family's neighbors configuration.
- nxos_command - Run arbitrary command on Cisco NXOS devices

```
nx-osv9000-1# conf t
nx-osv9000-1(config)# feature nxapi
nx-osv9000-1(config)# exit
nx-osv9000-1#
```

http://docs.ansible.com/ansible/latest/list_of_network_modules.html#nxos

Synopsis

- Manages Layer 3 attributes for IPv4 and IPv6 interfaces.

Each module has a similar page. Version dependent!

Requirements (on host that executes module)

- ipaddress

Options

parameter	required	default	choices	comments
addr	no			IPv4 or IPv6 Address.
allow_secondary (added in 2.4)	no			Allow to configure IPv4 secondary addresses on interface.
host	yes			Specifies the DNS host name or address for connecting to the remote device over the specified transport. The value of host is used as the destination address for the transport.
interface	yes			Full name of interface, i.e. Ethernet1/1, vlan10.
mask	no			Subnet mask for IPv4 or IPv6 Address in decimal format.
password	no			Specifies the password to use to authenticate the connection to the remote device. This is a common argument used for either *cli* or *nxapi* transports. If the value is not specified in the task, the value of environment variable `ANSIBLE_NET_PASSWORD` will be used instead.
port	no	0 (use common port)		Specifies the port to use when building the connection to the remote device. This value applies to either *cli* or *nxapi*. The port value will default to the appropriate transport common port if none is provided in the task. (cli=22, http=80, https=443).

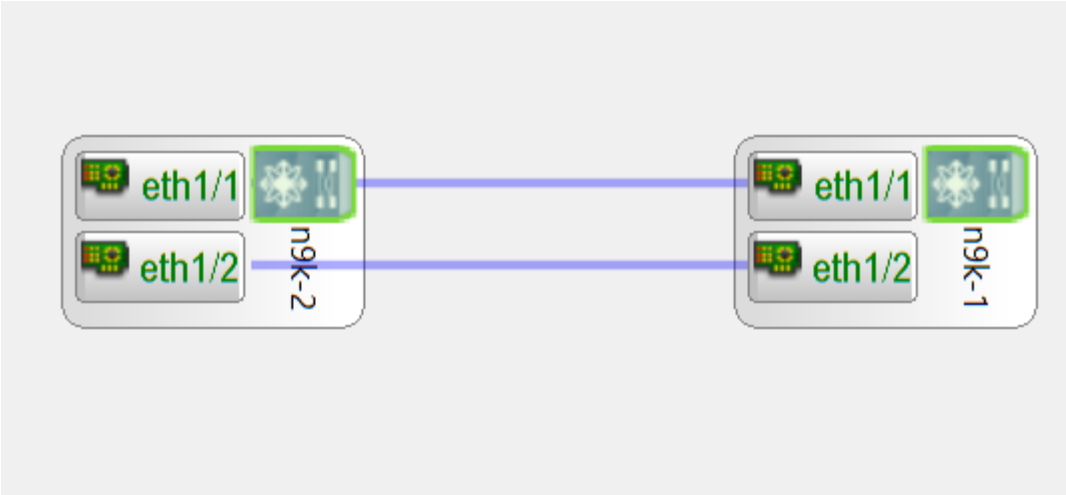
Examples

```
- name: Ensure ipv4 address is configured on Ethernet1/32
  nxos_ip_interface:
    interface: Ethernet1/32
    transport: nxapi
    version: v4
    state: present
    addr: 20.20.20.20
    mask: 24

- name: Ensure ipv6 address is configured on Ethernet1/31
  nxos_ip_interface:
    interface: Ethernet1/31
    transport: cli
    version: v6
    state: present
    addr: '2001::db8:800:200c:cccb'
    mask: 64

- name: Ensure ipv4 address is configured with tag
  nxos_ip_interface:
    interface: Ethernet1/32
    transport: nxapi
    version: v4
    state: present
    tag: 100
    addr: 20.20.20.20
    mask: 24
```

With examples!



The environment doesn't need to be complex.

Ansible Jinja2 Templates and Playbook Progression *(DEMO)*

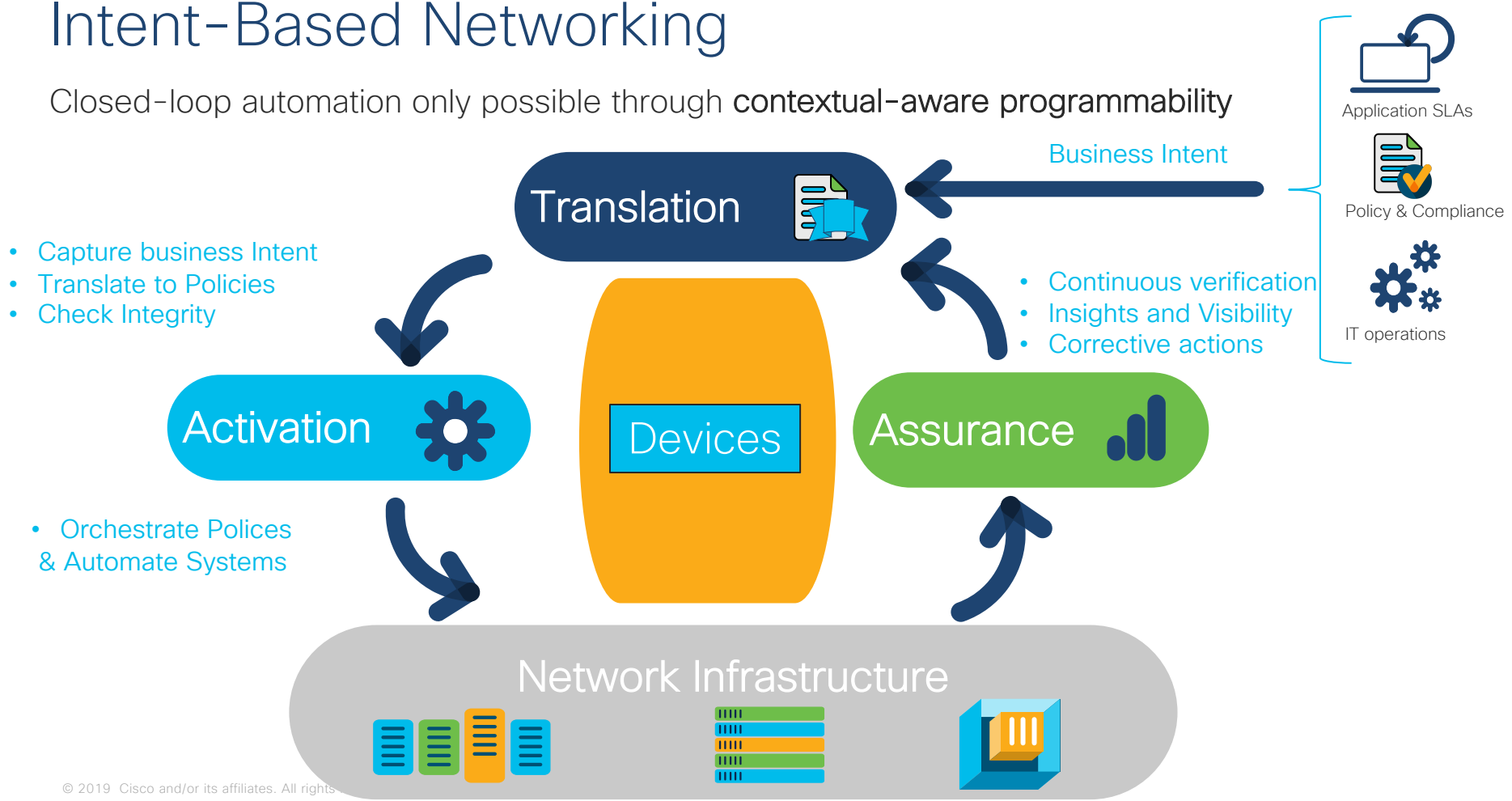
Moving Beyond Ansible

Ansible Gaps

- Designed for stateless server automation
- Complex automations with programmable constructs are limited
- No “service-level” assurance; the state is the network
- Module coverage can be hit-or-miss
- Architectural changes may require rework to playbooks (2.5+)

Intent-Based Networking

Closed-loop automation only possible through **contextual-aware programmability**



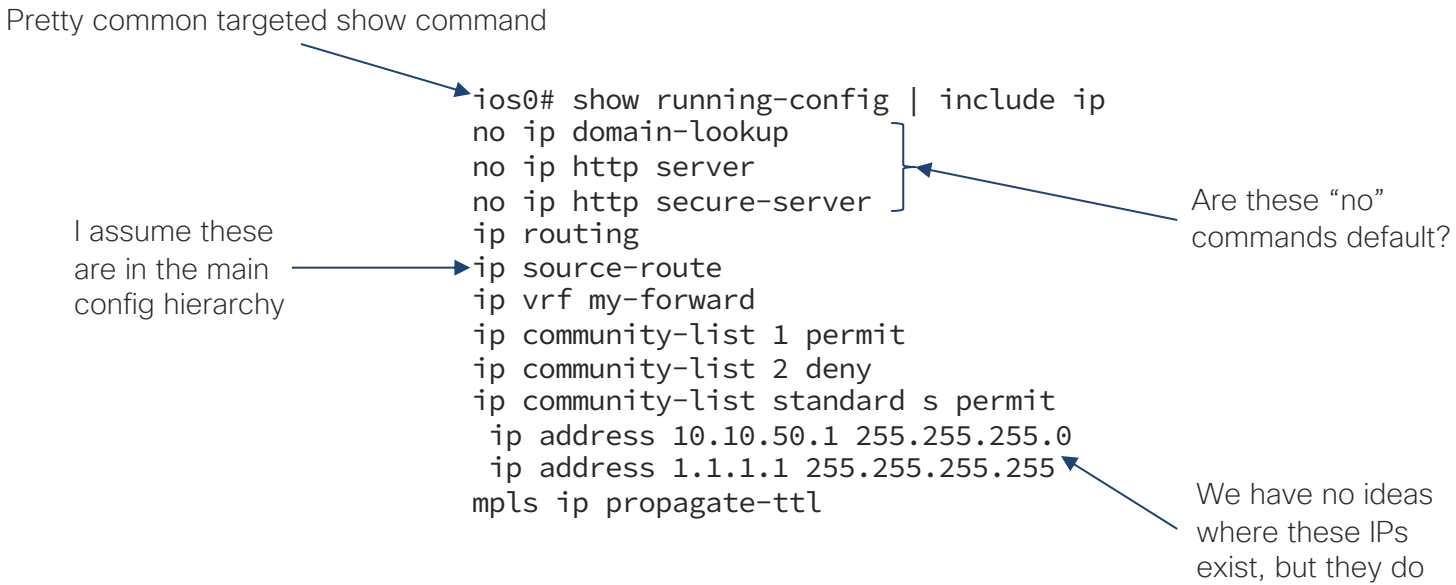
IBN Requires Context *AND* Assurance

> "To assure what is orchestrated, we must orchestrate assurance"
>
> -- Wise Person

Device-centric Automation

Service-centric Automation
(assurance included)

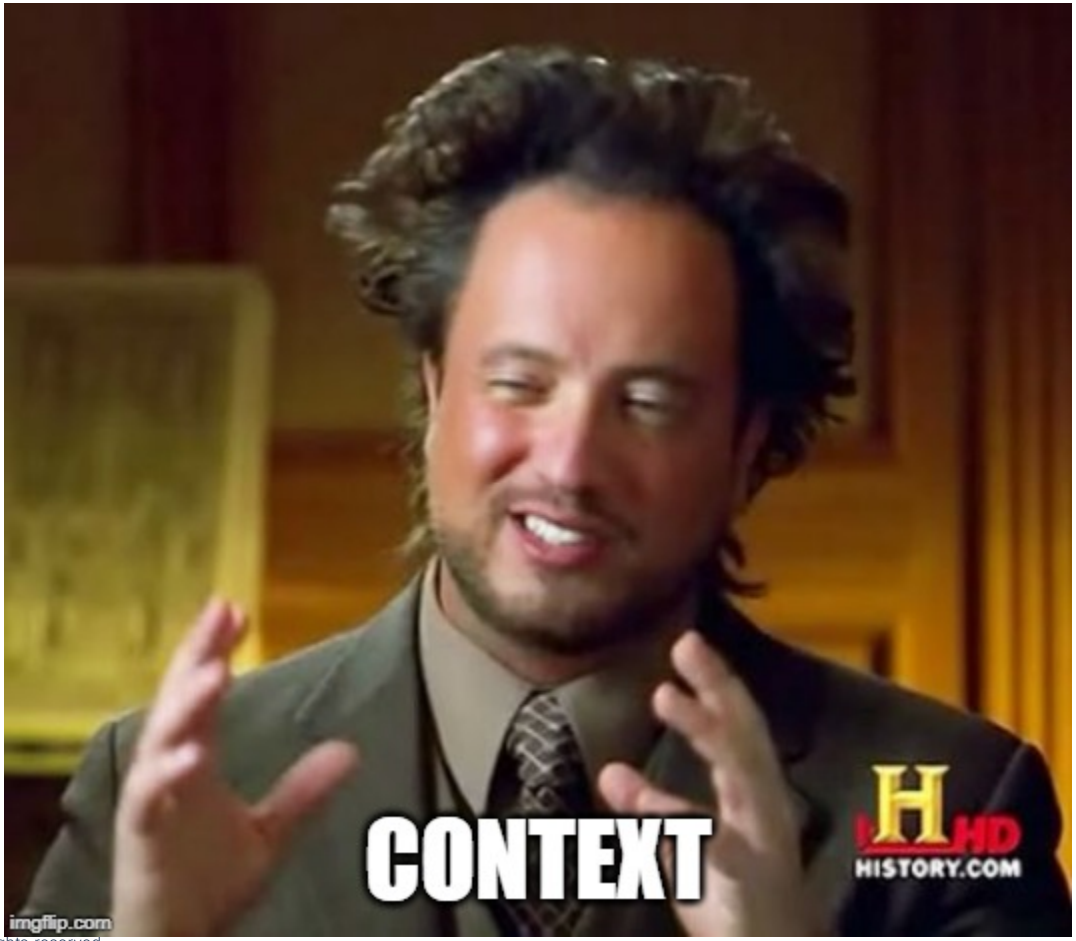
"Network programmability uptake is drastically slowed by legacy, CLI-only infrastructure. We need a way to contextualize existing configurations and provide a network-wide API."



Without data structures – we are left in a world without context

Not all heroes wear capes…


```
admin@ncs# show running-config devices device ios0 | include ip | display xpath
/devices/device[name='ios0']/config/ios:ip/vrf[name='my-forward']/bgp/next-hop/Loopback 1
/devices/device[name='ios0']/config/ios:ip/community-list/number-standard[no='1']/permit
/devices/device[name='ios0']/config/ios:ip/community-list/number-standard[no='2']/deny
/devices/device[name='ios0']/config/ios:ip/community-list/standard[name='s']/permit
/devices/device[name='ios0']/config/ios:ip/http/server false
/devices/device[name='ios0']/config/ios:ip/http/secure-server false
/devices/device[name='ios0']/config/ios:ip/routing
/devices/device[name='ios0']/config/ios:ip/source-route true
/devices/device[name='ios0']/config/ios:interface/FastEthernet[name='0/1']/ip/address/primary/address 10.10.50.1
/devices/device[name='ios0']/config/ios:interface/FastEthernet[name='0/1']/ip/address/primary/mask 255.255.255.0
/devices/device[name='ios0']/config/ios:interface/Loopback[name='0']/ip/address/primary/address 1.1.1.1
/devices/device[name='ios0']/config/ios:interface/Loopback[name='0']/ip/address/primary/mask 255.255.255.255
/devices/device[name='ios0']/config/ios:mpls/ip/conf/propagate-ttl true
```



APIs and Language Bindings

- NSO allows for “API normalization” and “API gateway” functionality based on exposed service
- **Remember:** Northbound APIs are all clients to the same YANG-based datastore
- Many customers start (and build trust) using the CLI, but gradually introduce REST for scripting trivial tasks

ncs-netsim

ncs-netsim

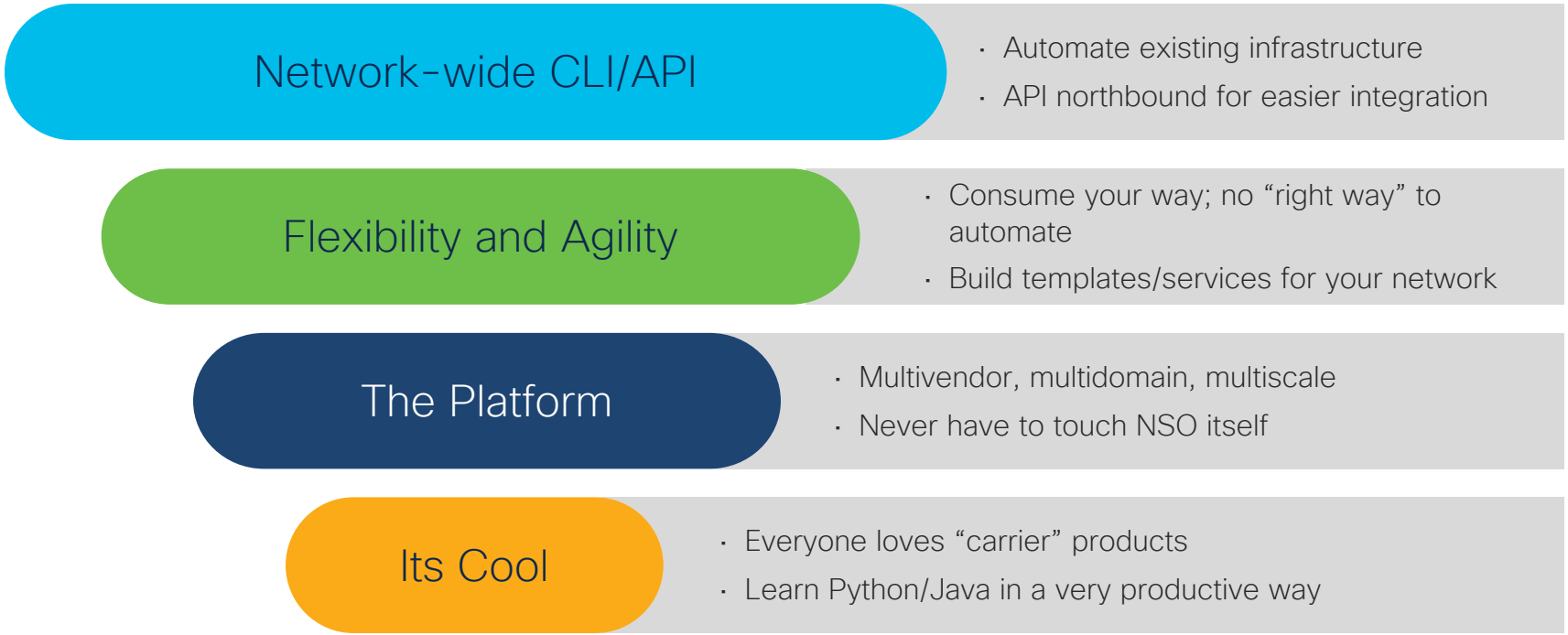
Batteries included programmability simulations

- Need something to automate “against”
- Control-plane only; still needs knowledge of “doing the right thing”
- Devices not “tied together”
- Arbitrary config to explore data model, can add config
- Device-type requires NED
- Snaps-in to NSO CLI, scales bigly

“There’s no sense in being precise when you don’t even know what you’re talking about”

~ John von Neumann

Why Look at NSO?



IaC CI/CD Pipeline with EVPN-VXLAN ("Alexa") *(DEMO)*

Wrapping it all up

GitHub all the things…

https://github.com/qsnyder/

connect-2019: complete repository of everything presented (except CI/CD)

ansible-docker_vagrant: Vagrant machine with scripts to install Ansible, Docker-CE, AWX with exposed ports for AWX access on localhost

GitHub all the things…

https://github.com/qsnyder/september-pcug

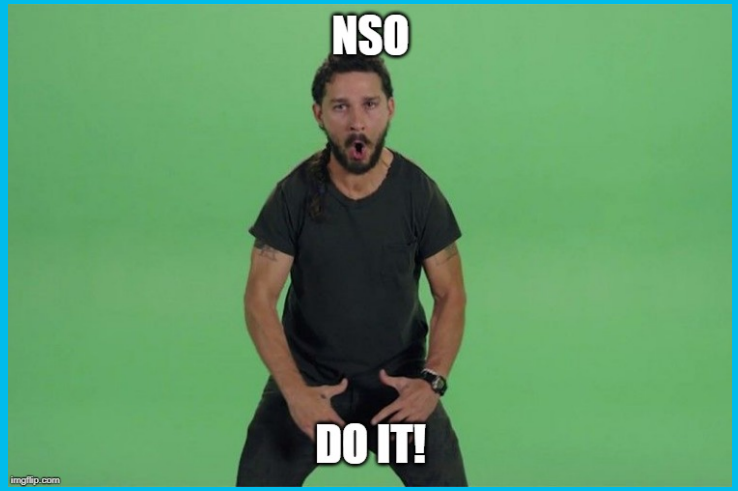
Makefile: Batteries included approach to starting NSO without syntactical barriers

arbitrary-yang: Syntactically correct YANG model to explore using `pyang`, with sample outputs

packages: Contains all of the service-package files demonstrated during the presentation

xr-skeleton-data: Configuration skeleton in XML format for ncs-netsim when invoking XR simulations (there is none in default NSO install)

NSO | Get your copy!



https://developer.cisco.com/docs/nso/

NSO has been released to the general public for free (as in beer) consumption

Package includes NSO software and sample NEDS to work through `examples.ncs` within $NCS_HOME

NSO DevNet

One-Stop sharing, finding, and collaboration

Light start through DevNet content page and Learning-Labs

Constant news and updates to keep you up to date

Code sharing through public GitHub

Large, searchable content pool

Access for Cisco customers, partners, and employees

Got a question? Ask! We will help ensure a fast response

Easy to share and find public content

www.cisco/go/nsodevnet

There's also dCloud

https://dcloud2-sjc.cisco.com/content/demo/26391

Leverage the DevNet Express DCv2 Pod.

- Cisco APIC 2.1(1h)
- VMWare Vcenter 6.0
- VMWare ESXi 6.0
- UCS Central 1.5(1b)
- UCM Manager 3.1(2b)
- UCS Director 6.5.0
- CIMC Emulator 1.0.0.11
- NetApp Edge Emulator 3.1(2h)
- NX-OSv 7.03(I5.2)

There's also dCloud

https://devnetsandbox.cisco.com/RM/Diagram/Index/6b023525-4e7f-4755-81ae-05ac500d464a?diagramType=Topology

Leverage the Multi-IOS Sandbox

https://github.com/DevNetSandbox/sbx_multi_ios/tree/master/cicd-3tier

GitHub repository to work with a CI/CD pipeline and NSO automation


CISCO