A Practical Look at Network Automation

Jason Edelman
@jedelman8
jedelman.com
jason@networktocode.com
AGENDA

• Why Are We Here?
• SDN Alongside Network Automation
• Use Cases
• Action Plan
AGENDA

• Why Are We Here?
• SDN Alongside Network Automation
• Use Cases
• Action Plan
WHY ARE WE HERE

- OpenFlow
- Software Defined Networking
- APIs
- DevOps
- Network Automation
- Learn how to Program?
THE REALITY

It’s 2014 on highway 101 from San Francisco to San Jose, some cars are driving themselves. Around the world there are military aircraft flying around with no pilot, being controlled by remotely from another country. In your data center there is an engineer/admin configuring a switch on a CLI. What’s wrong with this picture?

Joe Onisick – Principal Engineer Cisco Systems
PROBLEM: NETWORK AGILITY

1994

Router> enable
Router# configure terminal
Router(config)# enable secret cisco
Router(config)# ip route 0.0.0.0 0.0.0.0 20.2.2.3
Router(config)# interface ethernet0
Router(config-if)# ip address 10.1.1.1 255.0.0.0
Router(config-if)# no shutdown
Router(config-if)# exit
Router(config)# interface serial0
Router(config-if)# ip address 20.2.2.2 255.0.0.0
Router(config-if)# no shutdown
Router(config-if)# exit
Router(config)# router rip
Router(config-router)# network 10.0.0.0
Router(config-router)# network 20.0.0.0
Router(config-router)# exit
Router(config)# exit
Router# copy run start

Terminal Protocol: Telnet

2014

Router> enable
Router# configure terminal
Router(config)# enable secret cisco
Router(config)# ip route 0.0.0.0 0.0.0.0 20.2.2.3
Router(config)# interface ethernet0
Router(config-if)# ip address 10.1.1.1 255.0.0.0
Router(config-if)# no shutdown
Router(config-if)# exit
Router(config)# interface serial0
Router(config-if)# ip address 20.2.2.2 255.0.0.0
Router(config-if)# no shutdown
Router(config-if)# exit
Router(config)# router rip
Router(config-router)# network 10.0.0.0
Router(config-router)# network 20.0.0.0
Router(config-router)# exit
Router(config)# exit
Router# copy run start

Terminal Protocol: SSH
LOOKING AHEAD

• Network Operations does in fact need to be improved, but there is more…

• Need to embrace the people that embrace the culture, process, and technology that adapt to change

• Re-think: Engineer for Change
AGENDA

• Why Are We Here?
• SDN Alongside Network Automation
• Use Cases
• Action Plan
EVOLVING ECOSYSTEMS

Cloud Management Platforms

Controller Based Networking Fabrics

ACI | NSX | Plexxi | OpenDaylight
REST | Big Cloud Fabric | Nuage

Network Automation Tools

Programmable Network Devices

NX-API | onePK | eAPI | REST
Cumulus | JunOS XML | API de jour

Are These Divergent Paths?
EVOLVING ECOSYSTEMS

Cloud Management Platforms

Controller Based Networking Fabrics

Network Automation Tools

Programmable Network Devices

Programmability and Platform Extensibility Should be Key Decision Making Criteria
CONSISTENCY

Cloud Management Platforms

Network Automation Platforms & Tool Chains

Controller Based Networking Fabrics

Programmable Network Devices

ACI  NSX  Plexxi  OpenDaylight

REST  Big Cloud Fabric  Nuage

NX-API  onePK  eAPI  REST

Cumulus  JunOS XML  API de jour

Consistent policy, configuration, tools, and common languages and interfaces

DESIGN FOR CHANGE
AGENDA

• Why Are We Here?
• SDN alongside Network Automation
• Use Cases
• Action Plan
### Let’s Get Practical

<table>
<thead>
<tr>
<th>Template Building</th>
<th>Device Configurations, Vendor Migrations, IPv4 to IPv6 Migration, Site Rollouts, Office/DC Relocations, BYOD configs for switches</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Collection</td>
<td>Cabling Check, Neighbors, Serial Numbers (support contracts?), Linecards, Modules, Audit Checks, PSIRT checks</td>
</tr>
<tr>
<td>Super Commands</td>
<td>Wireless Client to AP to Switchport, Phone to switchport, BGP Table + Routing Table, Integrate to UC, WLAN, IPAM</td>
</tr>
<tr>
<td>Troubleshooting (Ops)</td>
<td>Cabling, L2 neighbors, L3 adjacencies (have it tell you WHY the neighbor relationship failed), Interface Errors, ACLs</td>
</tr>
<tr>
<td>Source Control</td>
<td>Configuration, Templates, Dynamic state stored in central repositories. Re-deploy infrastructure → DR/BCP, Relocations</td>
</tr>
<tr>
<td>Provisioning</td>
<td>The Scary Part? Configs, config snippets, one-off changes</td>
</tr>
</tbody>
</table>

SO MUCH CAN BE DONE WITHOUT “PUSHING” CONFGS
TEMPLATE BUILDING
WHERE TO BEGIN?
TEMPLATIZE CONFIGS

Jinja
DE-COUPLE THE VARIABLES
RENDER THE TEMPLATE

Great way to get started, but for more robust templates, a “real” tool should be used.

Render script available on my GitHub page.

Think config snippets:
- v4 to v6
- BYOD
- one-offs

https://github.com/jedelman8/interop-nyc-2014
CABLE VERIFICATION

• Is the cabling accurate?

• How do you know if something is mis-cabled?

• Ever work 3rd party contractors that cable based on your patch schedule and somehow it doesn’t come out right?
DEFINE THE DESIRED STATE

1. Define the Desired Cabling Scheme

(only showing portion of YAML file)
OBTAIN THE ACTUAL STATE

1. Define the Desired Cabling Scheme
2. Get the actual (run time) topology via CDP/LLDP

Multiple methods available
This example uses a Python script and gets neighbor info using NX-API on the Nexus 9000

(only showing portion of YAML file)
DESIRED VS. ACTUAL

1. Define the Desired Cabling Scheme
2. Get the actual (run time) topology via CDP/LLDP
3. Examine Desired vs. Actual

(only showing portion of YAML file)
PAUSE: SAMPLE NX-API OUTPUT

cisco@edelman:/apps/nxapi/library/pyfiles$ python interop-cdp.py

 Neighbor: c3550
Local Interface: mgmt0
Neighbor Interface: FastEthernet0/22

 Neighbor: N9K2.cisconxapi.com(SAL1819S6LU)
Local Interface: Ethernet1/48
Neighbor Interface: Ethernet1/1

 Neighbor: N9K2.cisconxapi.com(SAL1819S6LU)
Local Interface: Ethernet2/11
Neighbor Interface: Ethernet2/2

 Neighbor: N9K2.cisconxapi.com(SAL1819S6LU)
Local Interface: Ethernet2/12
Neighbor Interface: Ethernet2/1
NX-API
< 20 Lines of Code

1. Connect to Device
2. Wrap CLI and get return data
3. Convert XML to dict (JSON)
4. Extract CDP information
5. Print CDP information
TROUBLESHOOTING OSPF

• Remember how neighbors are formed in OSPF?

• Do you remember at 3am on a Saturday?

• Does the junior network engineer remember when you’re on vacation?
TROUBLESHOOTING OSPF

• Remember how neighbors are formed in OSPF?

• Do you remember at 3am on a Saturday?

• Does the junior network engineer remember when you’re on vacation?

• How about we automate the process of a neighbor check?

• Do we really enjoy bouncing back and forth between routers?

• Let’s get to it!
cisco@onepk:~/apps/a4nS ansible-playbook ospfops.yml

PLAY [collecting ospf data] **************************************************************

TASK: [get ospf facts] ********************************************************************
ok: [10.1.1.110]
ok: [10.1.1.110]

TASK: [interface ip addresses used for OSPF peering] *************************************
ok: [10.1.1.110] =>
  "msg": "Local router interface IP address- 10.1.1.110/24 on gig0/2"
ok: [10.1.1.120] =>
  "msg": "Local router interface IP address- 10.1.1.120/24 on gig0/1"

TASK: [is ospf active on interface?] ****************************************************
ok: [10.1.1.110] =>
  "msg": "ospf active on interface = True"
ok: [10.1.1.120] =>
  "msg": "ospf active on interface = True"

TASK: [process id check] ****************************************************************
ok: [10.1.1.110] =>
  "msg": "at least one ospf process configurd on router = True"
ok: [10.1.1.120] =>
  "msg": "at least one ospf process configurd on router = True"

TASK: [MTUs of interfaces] **************************************************************
ok: [10.1.1.110] =>
  "msg": "MTU = 1500"
ok: [10.1.1.120] =>
  "msg": "MTU = 1500"

TASK: [ospf network type] **************************************************************
ok: [10.1.1.110] =>
  "msg": "network type = BROADCAST"
ok: [10.1.1.120] =>
  "msg": "network type = BROADCAST"

TASK: [ospf timers on interface] ****************************************************
ok: [10.1.1.110] =>
  "msg": "{"hello": u'10', "dead": u'40"}"
ok: [10.1.1.120] =>
  "msg": "{"hello": u'10', "dead": u'40"}"

TASK: [interface status] **************************************************************
ok: [10.1.1.110] =>
  "msg": "interface status=up and line protocol = up"
ok: [10.1.1.120] =>
  "msg": "interface status=up and line protocol = up"

TASK: [display neighbors and state] ***************************************************
ok: [10.1.1.110] =>
  "msg": "{"": ["FULL", "BDR"]}"
ok: [10.1.1.120] =>
  "msg": "{"": ["FULL", "BDR"]}"
GET FACTS AND ANALYZE

Everything else is just printing data from facts.
THE ANSIBLE PLAYBOOK

```yaml
---
# PLAY 1

- name: collecting ospf data
  hosts: routers
  connection: local
  gather_facts: no

  tasks:
    - name: get ospf facts
      ospf_facts: device={{inventory_hostname}} interface={{ ospf_interface }}
      register: ospf_data
    - name: interface ip addresses used for OSPF peering
      debug: msg="local router interface IP address- {{ ospf_data.ofacts.oif_ip }}"

## BOTTOM DEBUG / PRINTS HAVE BEEN REMOVED ##
```
THE ANSIBLE PLAYBOOK

• `ospf_facts` is an Ansible module
• Ansible modules can be written in Python

• BUT, WHO WRITES THEM?
WHAT OPTIONS DO WE HAVE FOR TOOLS?

DIY

Do we need to be able to program?
Less as time goes on.

Closed Turnkey Solution

Sweet spot.
AGENDA

• Why Are We Here?
• SDN alongside Network Automation
• Use Cases
• Action Plan
ACTION PLAN

• Dedicate time, maybe lots…
  – Remember how much time it took to get your existing certifications or learn any new skill?

• Document existing workflow and processes
  – Start with small tasks
  – You can’t automate what you don’t know

• Research DevOps Culture
ACTION PLAN

• Dedicate time, maybe lots…
  – Remember how much time it took to get your existing certifications or learn any new skill?

• Document existing workflow and processes
  – Start with small tasks
  – You can’t automate what you don’t know

• Research DevOps Culture

• Templating
  – Jinja2/YAML

• Scripting
  – Not building applications!
  – Python

• Try out a Device API

• Explore automation tools
  – Ansible (even if it’s to see what can be done with servers)
THANK YOU

“There may be people that have more talent than you, but there’s no excuse for anyone to work harder than you do.”

-Derek Jeter