

NetDevOps: Automation on Bare Metal Switches

- Intros
- What is Open Networking? (Kevin)
 - Open hardware, ONIE, APD
- Automation (Kevin)
 - Ansible playbooks, puppet manifests, etc for network configuration
- CI/CD in a Network context (Russ)
 - Full stack testing
- Demo (Russ)

Open Networking?

Open Hardware

ONIE

APD

Open Networking: What is It?

- Is it?
 - Linux Foundation
 - ONF
 - SDN
 - Openflow
 - API
- Or is it?
 - Freedom
 - Choice
 - Flexibility
 - Community



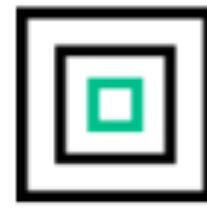
Open Standards



Open APIs



Open Ecosystem



Open Source

Open Networking: Why do I care?

- Operational Flexibility
- Innovate Faster
- Control your back office destiny
- Simple
- Did I mention Cost?

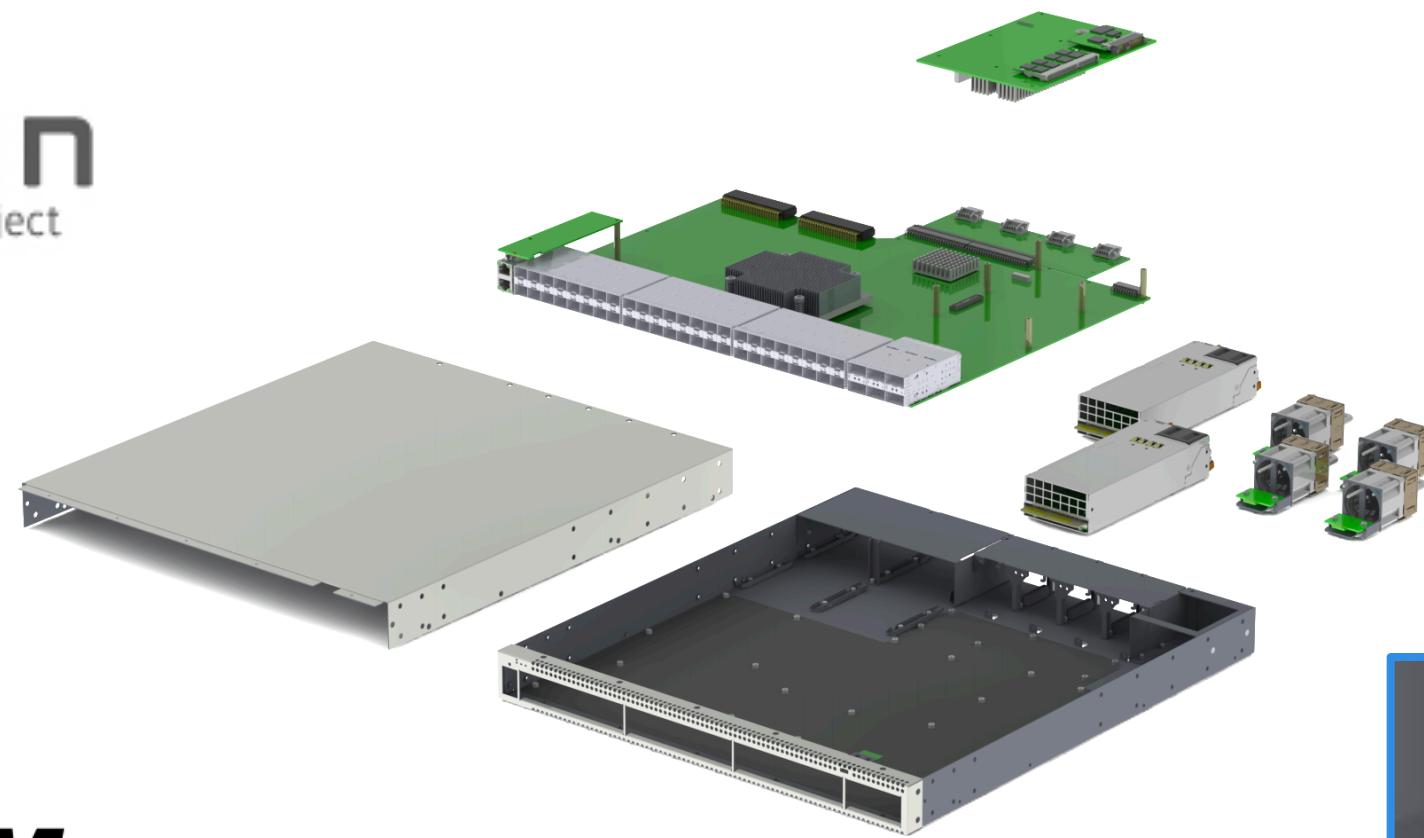


Open Hardware

Open Networking: “Whitebox switch”



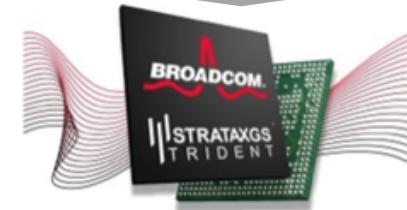
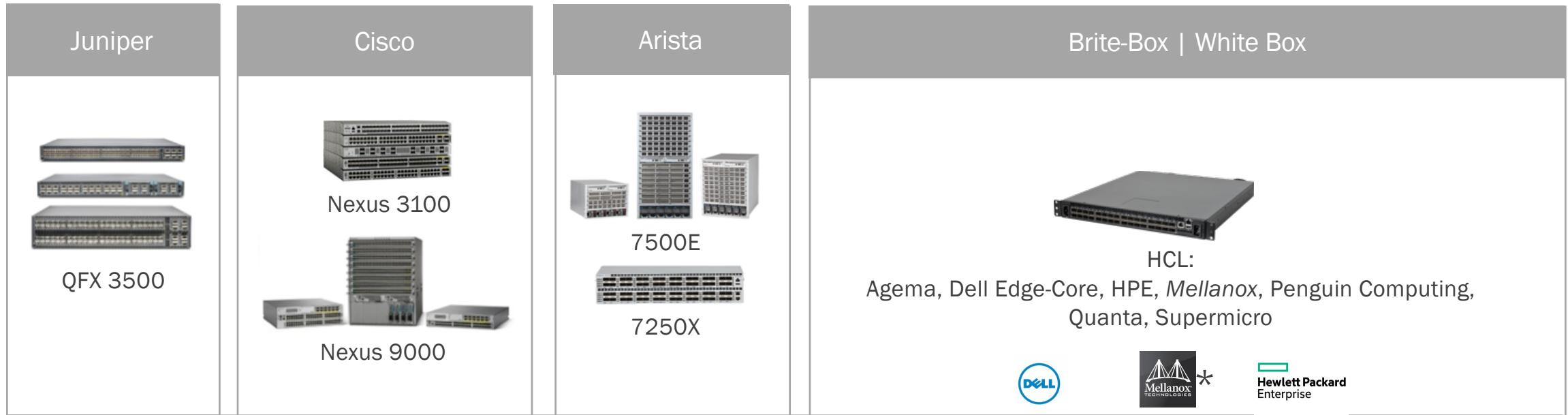
OPEN
Compute Project



BROADCOM®

**Mellanox®
TECHNOLOGIES**

Industry-Standard Network Hardware



Trident II

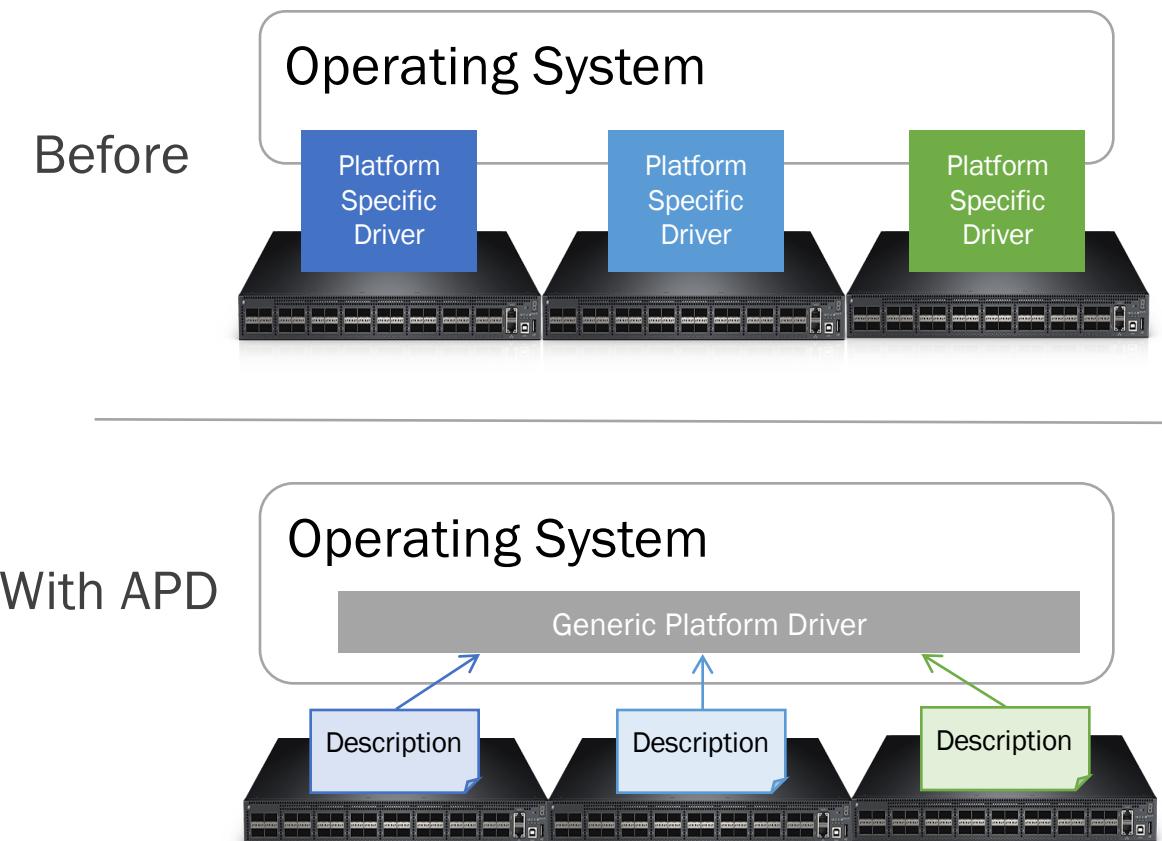
* Spectrum not T2

APD

ACPI Platform Description (APD)

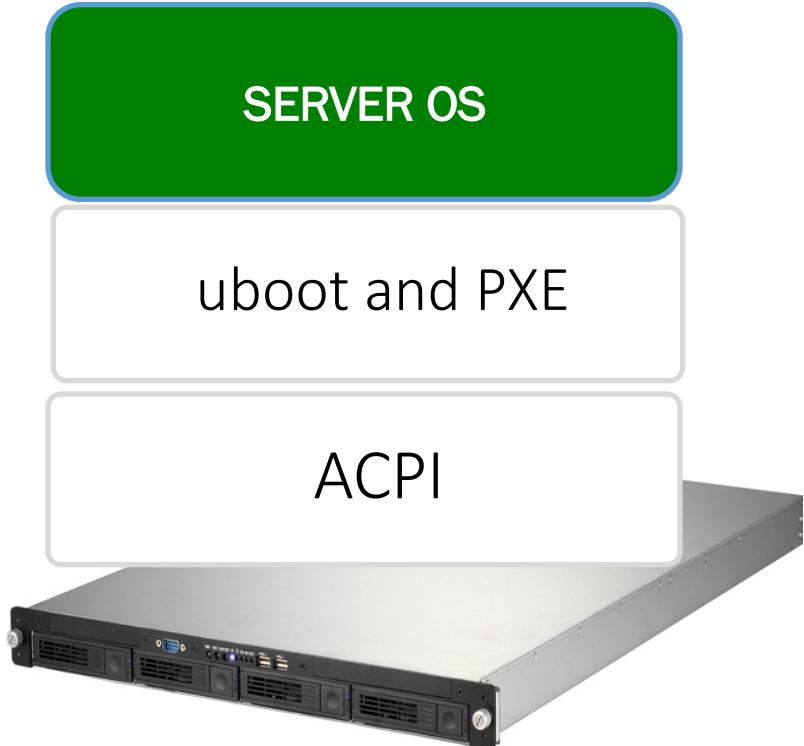
- Standardizing Platform Description

- ACPI extension for networking
 - used for PCs/servers
- Cuts Platform integration time
 - months to days
- Accelerate platform availability



Open ecosystem for open networking to thrive

Servers



Switches

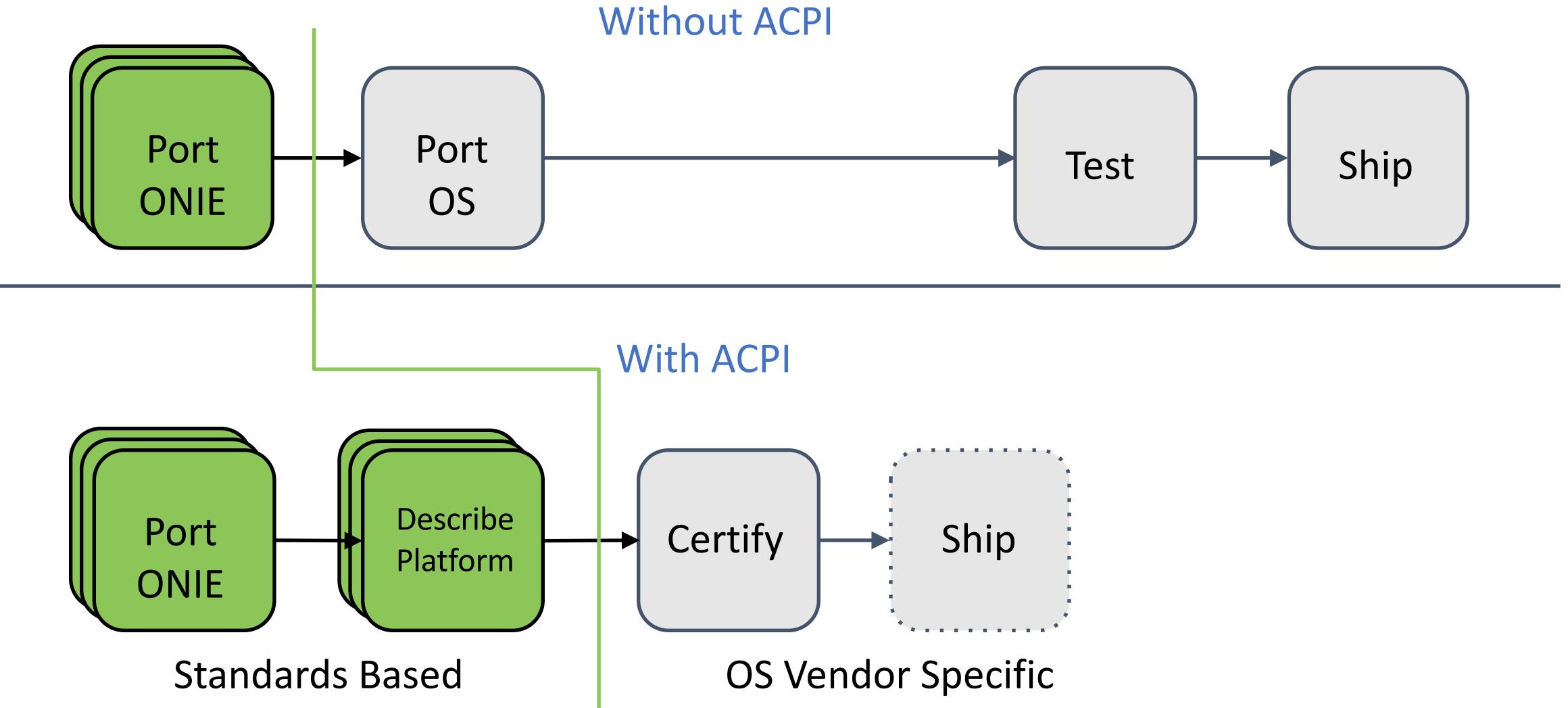


Linux Networking Model

Load any OS on any
platform

Self-described
platform to any OS

ACPI



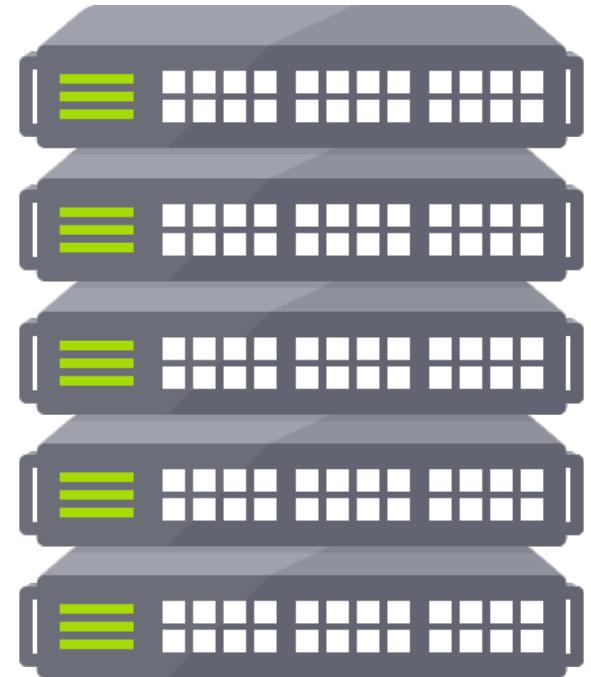
ONIE

Open Networking: What's an ONIE?

“PXE that doesn’t suck”

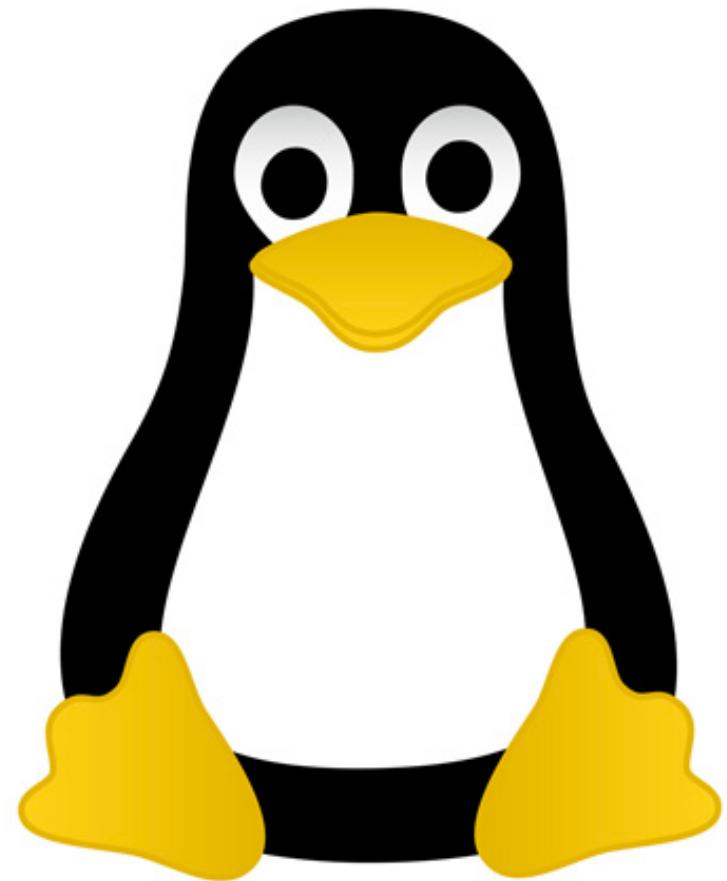


- Cumulus Networks contribution to OCP
- Enhanced Boot loader of Linux Kernel + BusyBox
- Enables install NOS of your choice



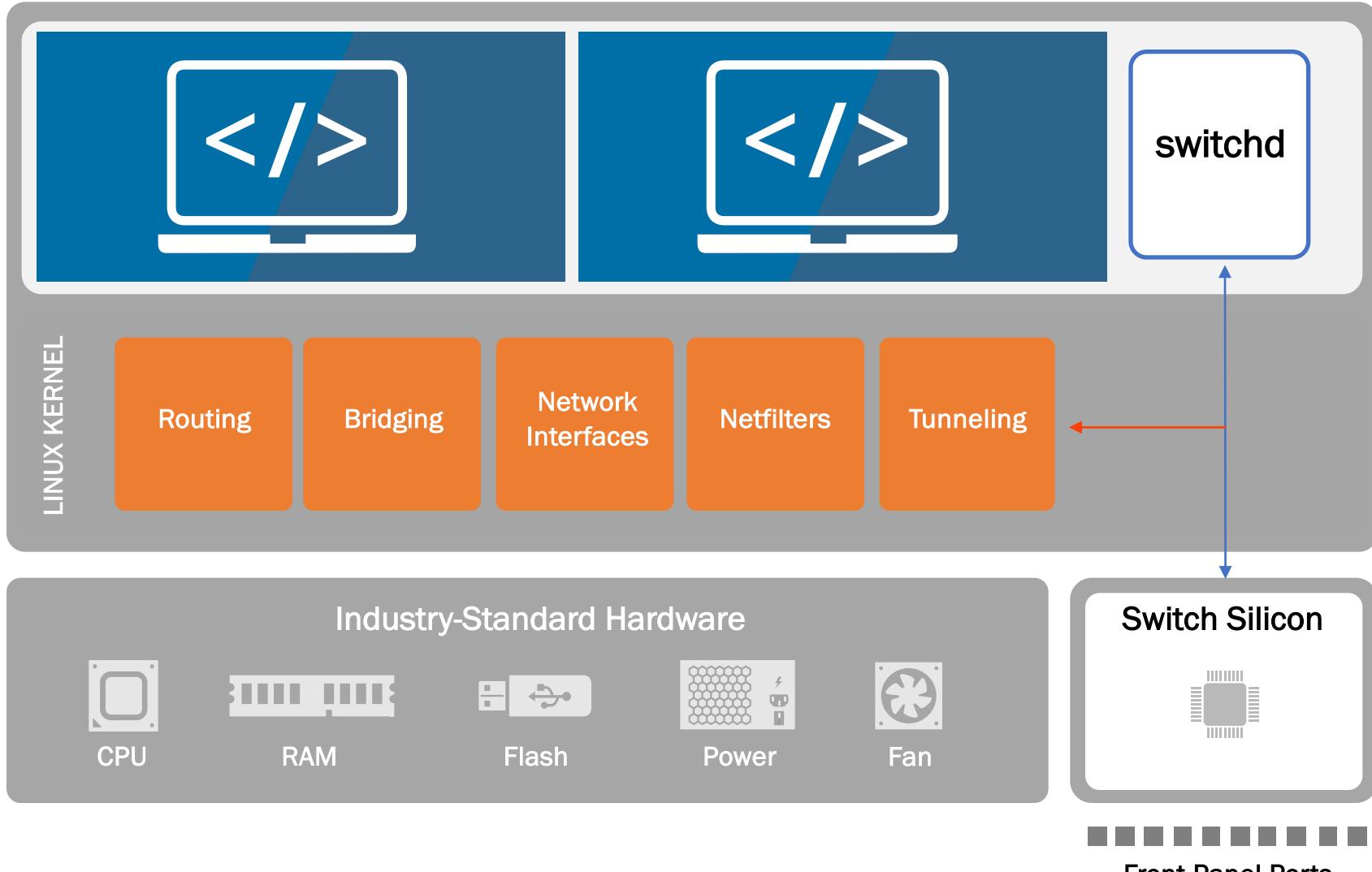


Just Add Network OS



Bare No More: The Emperor has Cloths!

Hmm... a Server
with 32 100G LINE-
RATE NIC's?





Open Networking Forecast

Snowflakes are hard to manage!

- Where we've been



DAVE GRANLUND © www.davegranlund.com

Transition feels like.



What took so long?



No More Snowflakes!

Automation

Basics of booting a switch

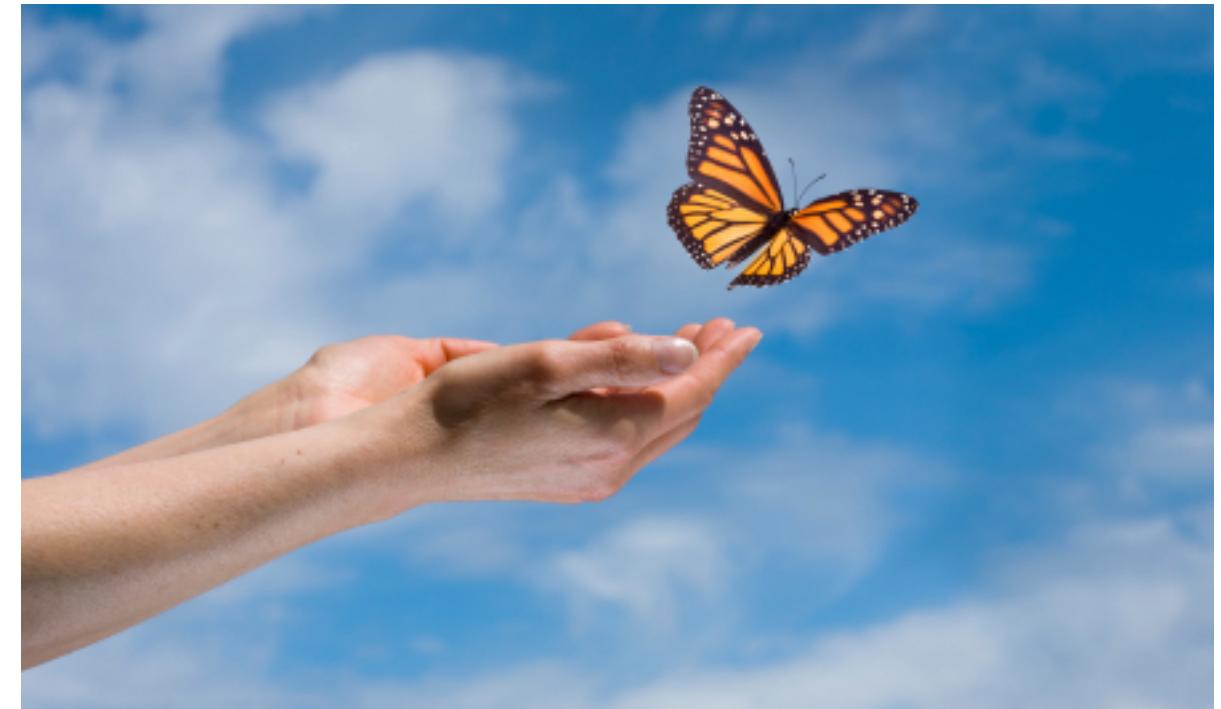
Ansible Playbooks

Puppet manifests

- Traditional Networking
Dragging you Down?

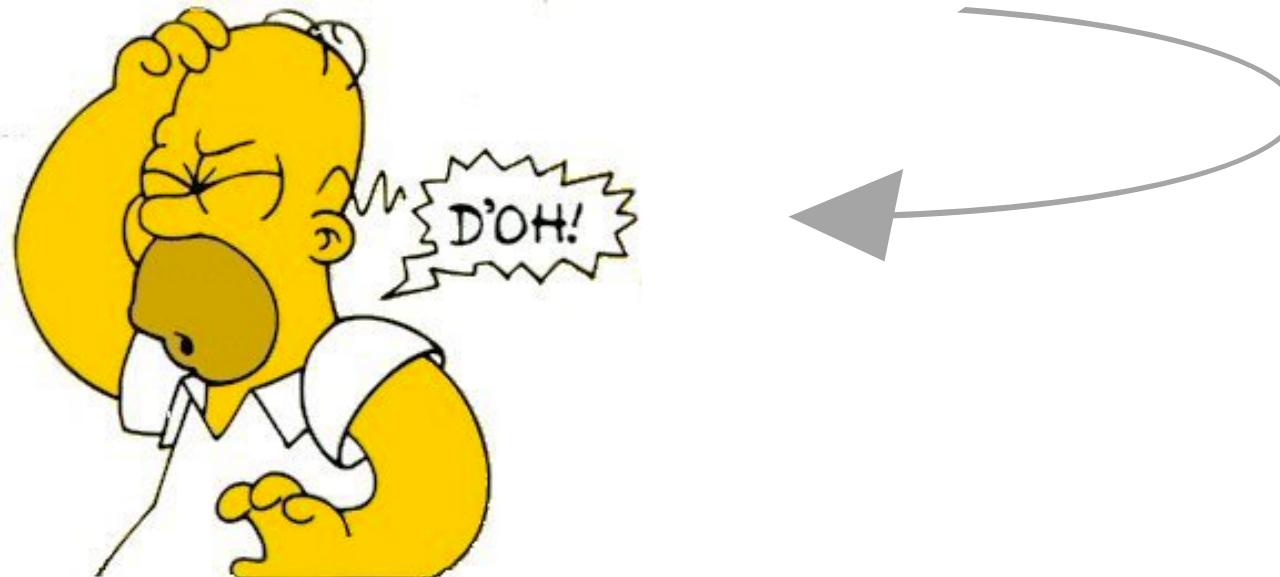


- Open Networking Free Fly Zone!

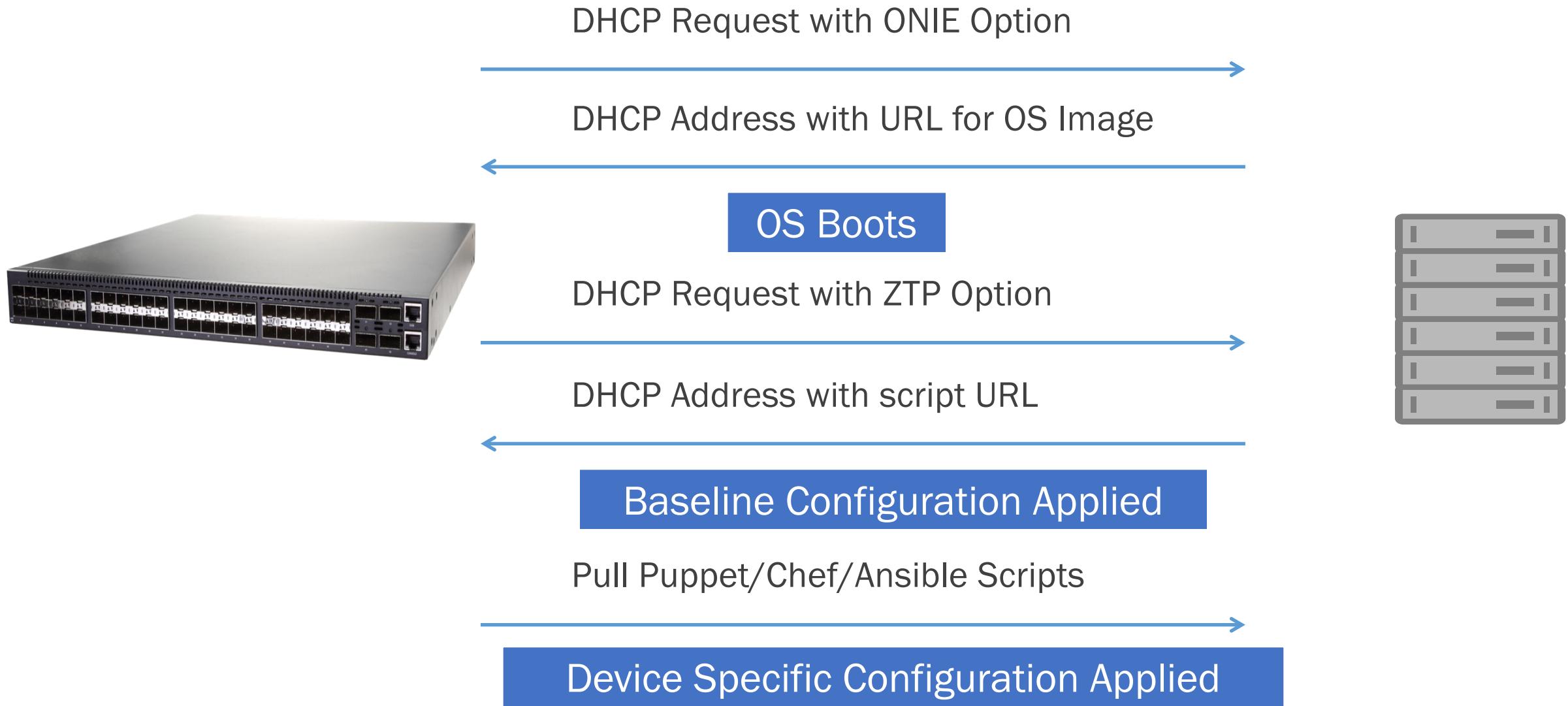


Configuration errors.... That never happens!

- Human Errors
 - Inconsistent
 - Unpredictable
 - Difficult to find
- Machine Errors
 - Consistent
 - [typically] easy fix
 - Caused by {see sample A}



ONIE and ZTP – The basics



ZTP with DHCP - example dhcp.conf

```
ddns-update-style none;
default-lease-time 4320;
max-lease-time 8640;
authoritative;

option cumulus-provision-url code 239 = text;

subnet 192.168.0.0 netmask 255.255.255.0 {
    range 192.168.0.100 192.168.0.200;
    option routers 192.168.0.1;
    option domain-name-servers 192.168.0.1;
    option domain-name "lab.mycompany.com";
    option cumulus-provision-url "http://192.168.0.1/provision.sh";
}
```

ZTP Scripts can be Powerful and/or Simple

```
#!/bin/bash

# Upgrade and install Puppet
apt-get update -y
apt-get upgrade -y
apt-get install puppet -y

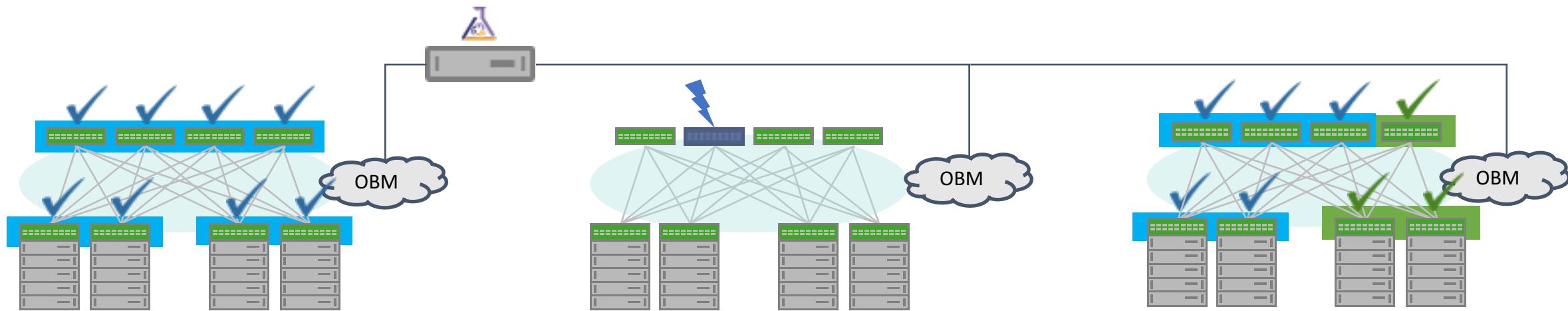
echo "Configuring puppet" | wall -n
sed -i /etc/default/puppet -e 's/START=no/START=yes/'
service puppet restart
# CUMULUS-AUTOPROVISIONING
exit 0
```

```
#!/bin/bash

mkdir -p /root/.ssh
/usr/bin/wget -O /root/.ssh/authorized_keys http://192.168.0.1/ssh.keys

#CUMULUS-AUTOPROVISIONING
```

NetDevOps: Automation Use Cases



Rapid Provisioning

- Weeks and months now take seconds
- Pods of equipment can be stamped out in multiple locations

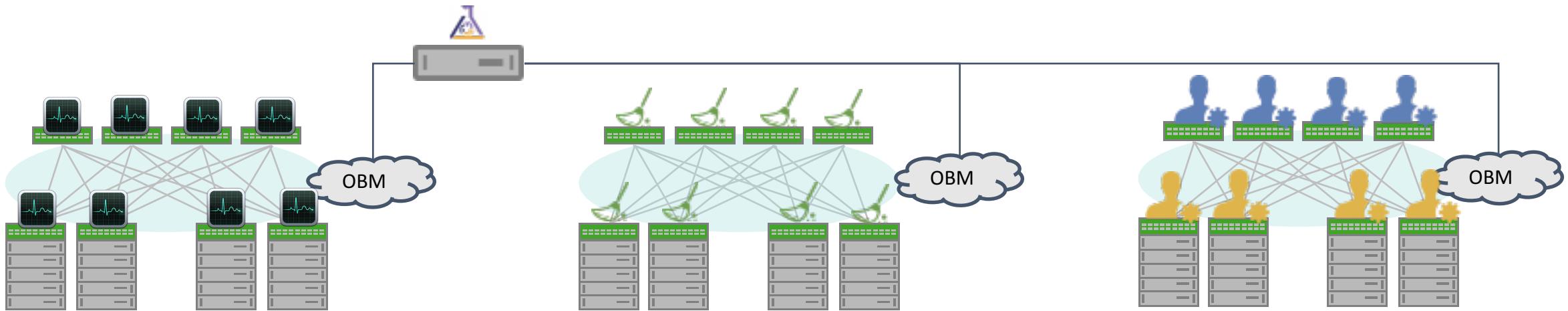
Hot Swap the Whole Switch

- With automation the whole switch can be replaced and provisioned
- High capacity, fine-grained failure domain

Configuration Management

- Infrastructure as code
- Enforcement from central location
- Easy change control

NetDevOps: Automation Use Cases



Easy Application Deployment

- No need to know how to install an App on Linux
- Apps can be installed, configured and up and running with one command

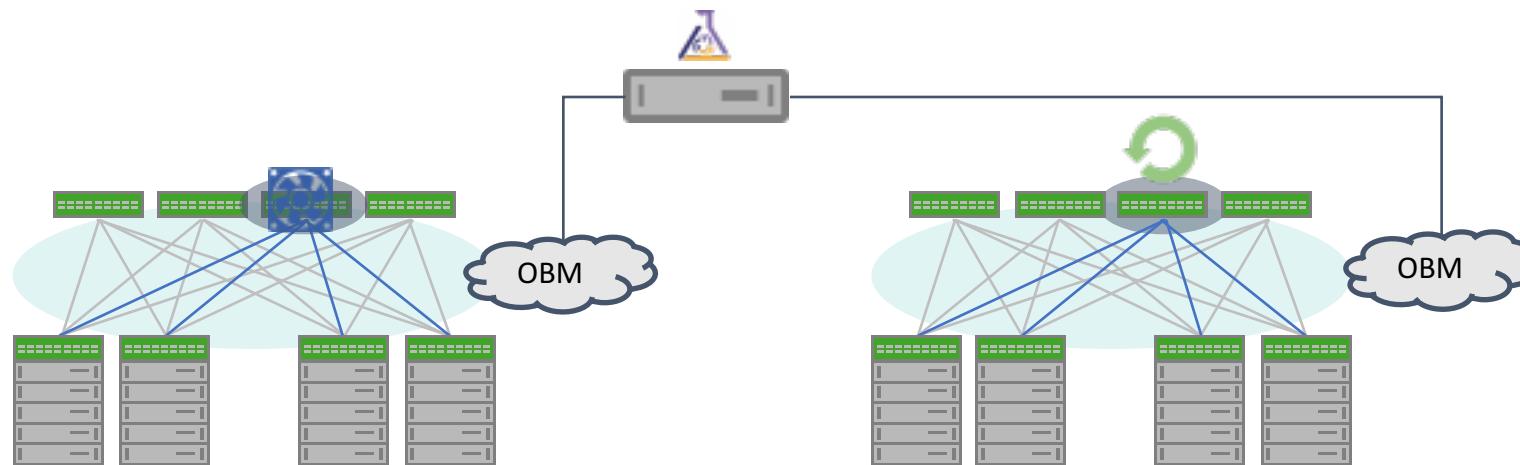
Maintenance Tasks

- Automate maintenance tasks such as log and space cleanup, patching applications, etc
- Shortened change-windows and less down-time

User Administration

- Kick off user customization for new hires
- Revoke user permissions as they change job roles

NetDevOps: Automation Use Cases

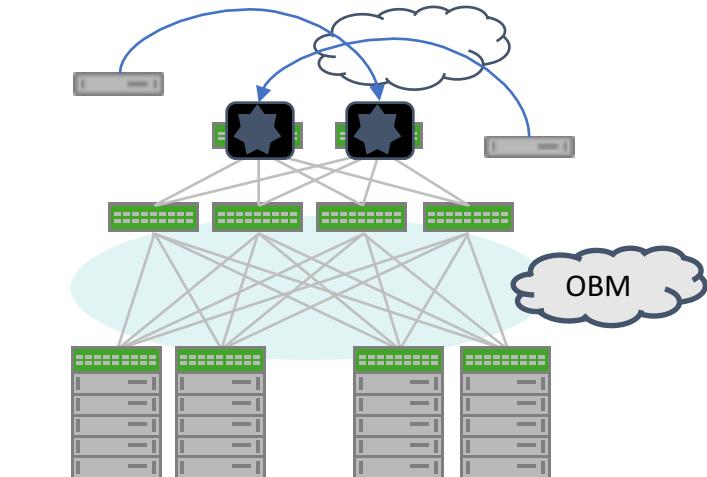


Reactive Network Changes

- Bad fan? High Temperature?
Don't just know when it happens but **automatically** respond

Hitless Upgrades

- Bring switches offline automatically without affecting the network
- Check surrounding devices and check route tables to know network state



Automatic Threat Response

- Deter threats as they happen rather than 'when' you notice
- Grab logging information and make it easy to see the attack vector

Provisioning Playbooks

```
---
```

```
# leaf-upgrade.yml
```

```
- hosts: a-leafs
become: yes
tasks:
  - include:
    roles/leafs/tasks/upgrade.yml
handlers:
  - include: handlers/main.yml
```



```
- hosts: b-leafs
become: yes
tasks:
  - include:
    roles/leafs/tasks/upgrade.yml
handlers:
  - include: handlers/main.yml
```

```
---
```

```
# handlers/main.yml (partial)
```

```
- name: reload networking
shell: "service networking reload"
register: networking_reload
notify: print networking reload error
changed_when: "networking_reload.stderr != """

- name: reload quagga
shell: "service quagga reload"
register: quagga_reload
notify: print quagga reload error
changed_when: "quagga_reload.stderr != ""

- name: print networking reload error
debug: msg="{{networking_reload.stderr}}"
when: networking_reload.stderr is defined
failed_when: "networking_reload.stderr is defined"

- name: print quagga reload error
debug: msg="{{quagga_reload.stderr}}"
when: quagga_reload.stderr is defined
failed_when: "quagga_reload.stderr is defined"

- name: restart apache
service: name=apache2 state=restarted

- name: start clcmd_server
service: name=clcmd_server state=started enabled=yes
```

```
---
```

```
# roles/leafs/tasks/upgrade.yml
```

```
- name: fetch ports.conf
fetch:
  dest=roles/leafs/{{ansible_hostname}}/ports.conf
  src=/etc/cumulus/ports.conf
  flat=yes

- name: copy quagga daemons file
fetch:
  dest=roles/leafs/{{ansible_hostname}}/daemons
  src=/etc/quagga/daemons
  flat=yes

- name: copy Quagga.conf
fetch:
  dest=roles/leafs/{{ansible_hostname}}/Quagga.conf
  src=/etc/quagga/Quagga.conf
  flat=yes

- name: List files in interfaces.d directory
shell: ls /etc/network/interfaces.d/
register: interfaces_files

- name: Fetch interfaces_files
fetch:
  dest=roles/leafs/
  src=/etc/network/interfaces.d/{{item}}
  with_items: interfaces_files.stdout_lines
```

Project Tree

```
$ tree
.
├── L2Topology.png
├── README.md
├── Vagrantfile
├── ansible.cfg
├── callback_plugins
│   ├── human_log.py
│   └── human_log.pyc
├── configs-backup.yml
├── handlers
│   └── main.yml
├── install_fping
├── I2-troubleshooting-all.yml
├── I2-troubleshooting-bridges.yml
├── I2-troubleshooting-clag.yml
├── I2-troubleshooting-links.yml
├── leaf-config-restore.yml
├── leaf-image-rollback.yml
├── leaf-upgrade.yml
├── provision.retry
├── provision.yml
└── roles
    ├── common
    │   ├── files
    │   │   ├── daemons
    │   │   └── interface_stats.py
    │   └── tasks
    │       └── main.yml
    └── leafs
        ├── tasks
        │   ├── configs-restore.yml
        │   ├── main.yml
        │   ├── rollback.yml
        │   └── upgrade.yml
        └── templates
            └── Quagga.conf.j2
```

Troubleshooting

```
---
# to run this playbook, use the following command:
# "ansible-playbook I2-troubleshooting-all.yml --verbose"
#
# to run commands individually, use the following:
# ansible network -a "clagctl"
- hosts: network
  tasks:
    - name: Show MLAG
      command: sudo clagctl

    - name: Show Bridges
      command: sudo brctl show

    - name: Show Bridge MACs
      command: sudo brctl showmacs br0

    - name: Show Bridge STP
      command: sudo brctl showstp br0

    - name: Show links
      command: sudo ip link show

    - name: Show Interfaces
      command: sudo ifquery -a
```

Infrastructure as Code

Where are my configs?

- Configuration for network nodes is stored:

- Cookbooks
- Manifests
- Playbooks

- Revision control tools:

- Git (Stash, Github)
- Subversion
- Mercurial
- Clearcase
- CVS

The screenshot shows a GitHub repository page for 'example-ospfunnum-chef'. At the top, there's a header with a search bar, pull requests, issues, and gist links. Below the header, the repository name 'CumulusNetworks / example-ospfunnum-chef' is displayed along with statistics: 26 commits, 1 branch, 0 releases, and 3 contributors. A 'Branch: master' dropdown is shown. The main content area displays a list of commits from 'Vanders' dated July 1. The commits are:

File	Description	Date
data_bags/interfaces	Configure interfaces using cumulus_* LWRPs	3 months ago
nodes	Re-organise everything to make it easier to install	5 months ago
roles	Start making use of the Cumulus cookbook	3 months ago
site-cookbooks	Remove redundant template	3 months ago
Cheffile	Start making use of the Cumulus cookbook	3 months ago
Gemfile	Import Cookbooks from cldemo packages	5 months ago
LICENSE	Adding MIT license	5 months ago
README.md	Add description to README	5 months ago
.chefignore	Import Cookbooks from cldemo packages	5 months ago

On the right side, there's a sidebar with links for Code, Issues, Pull requests, Wiki, Pulse, and Graphs. It also includes an HTTPS clone URL and options to Clone in Desktop or Download ZIP.

Pick'em - Automation Tools

Agentless	Agent-based
Requires no additional application software	Requires installation of an application (e.g., a daemon) to communicate with the control server
Relies on existing processes (such as SSH) to communicate with the control server	May also use existing processes to perform its functions

Want to learn more?

Community

<http://community.cumulusnetworks.com/>

Slack Channel



cumulusnetworks.slack.com



Download Cumulus VM's

<https://cumulusnetworks.com/cumulus-vx/>

Demo's and Lab's:

<https://support.cumulusnetworks.com/hc/en-us/articles/201787686>

<https://support.cumulusnetworks.com/hc/en-us/sections/200398866>

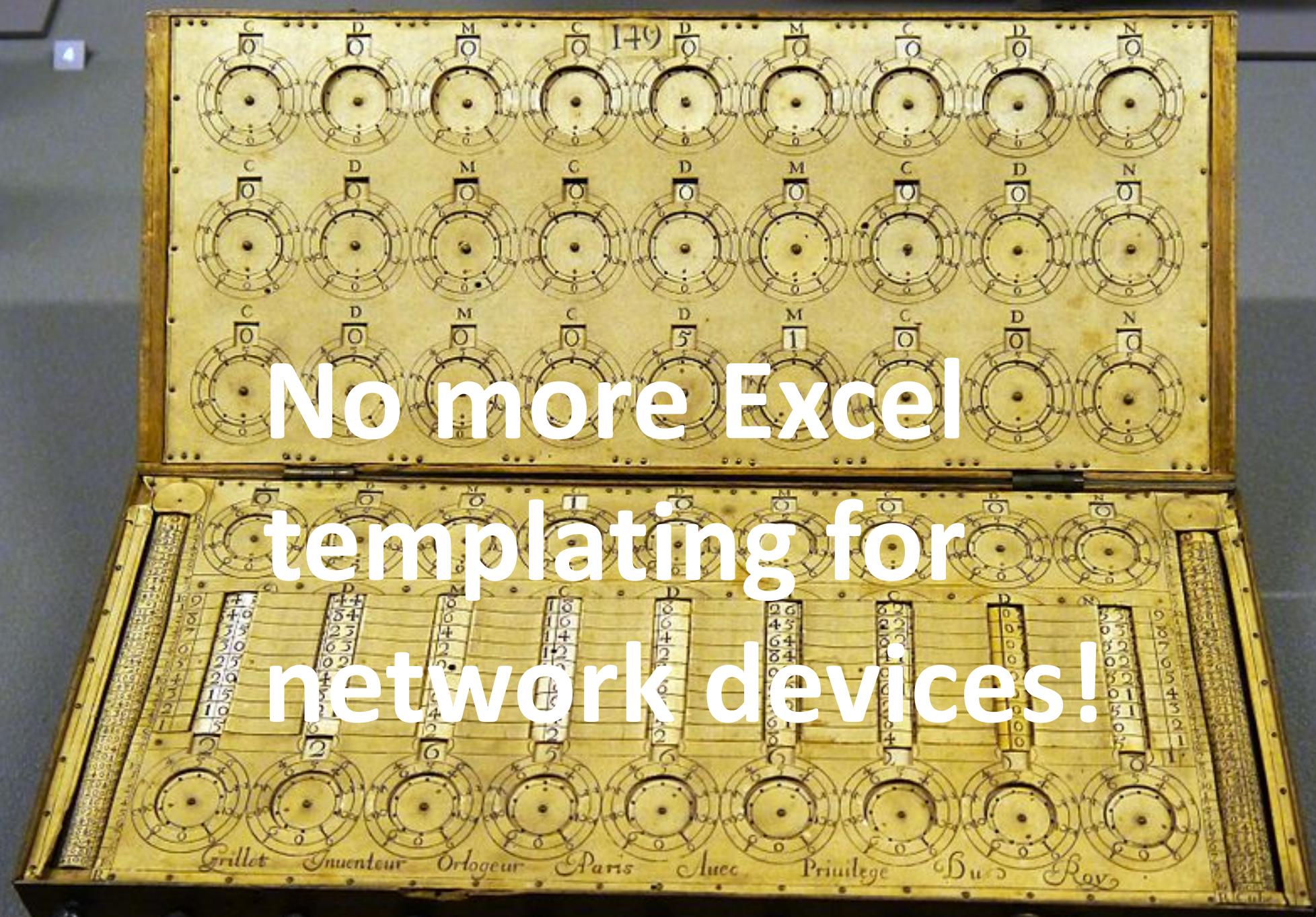


"Artisan working meticulously on a wooden sculpture" by Piseth1 is licensed under CC-BY-SA-3.0

CI/CD in a Network Context

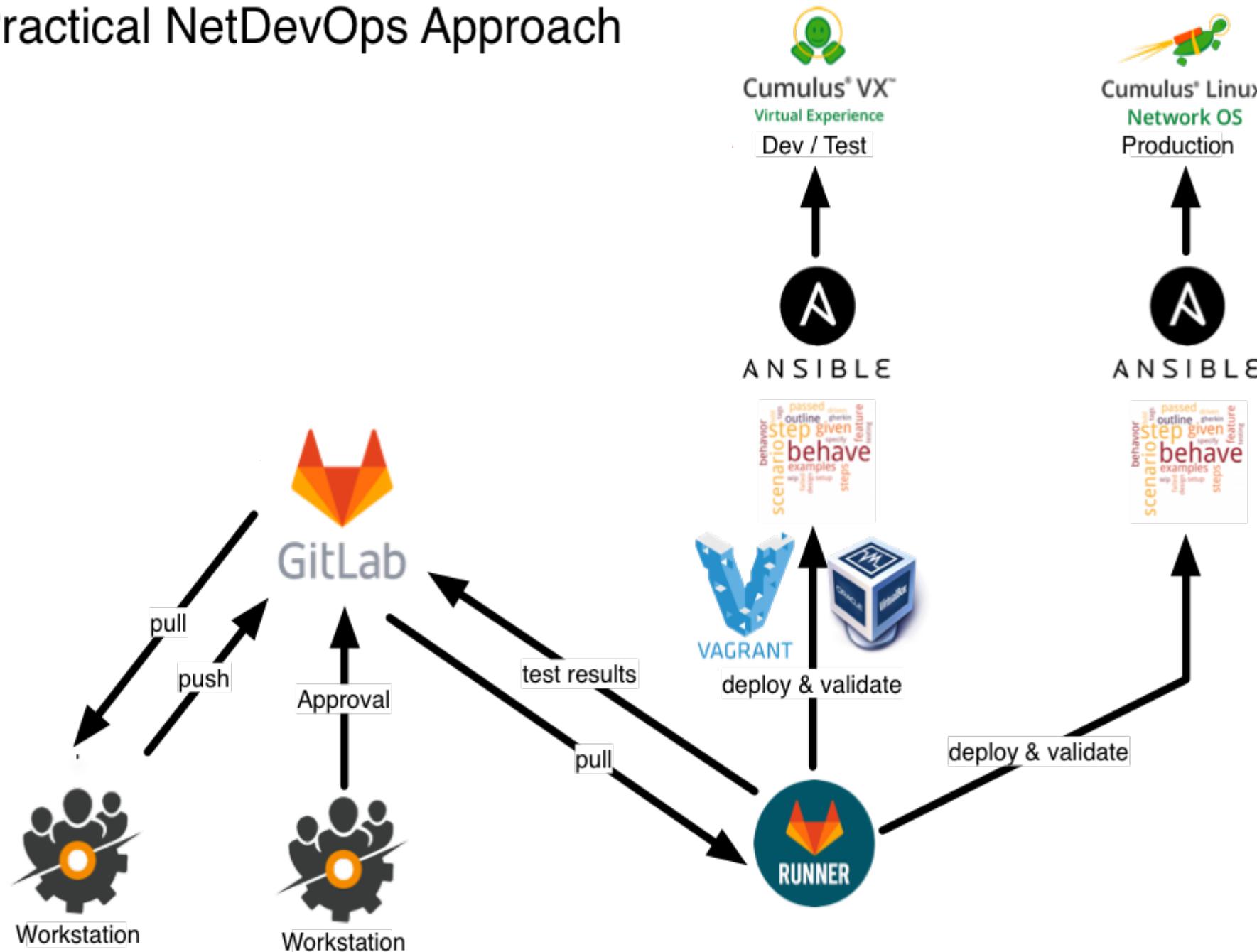
- CI/CD was originally intended for software
- CI/CD evolves and matures into DevOps
- Network Configuration is now code
- Desired outcomes for NetDevOps:
 - Raise quality and value of the Infrastructure AND the services on top of it
 - Spend less time on unplanned work & rework [1]
 - Stop creating snowflakes
 - Automate repetitive tasks
 - All changes go through Code Control
- Operational Improvement with NetDevOps
 - Capacity Planning
 - Automate troubleshooting steps / information gathering
 - Automate maintenance events and engaging “Smart Hands”

[1] “2016 State of DevOps” Report from Puppet



No more Excel
templating for
network devices!

A Practical NetDevOps Approach



GitLab CI Anatomy

📄 .gitlab-ci.yml 544 Bytes

```
1 ---  
2 before_script:  
3   - cd $CI_PROJECT_DIR/../../../../..  
4   - vagrant destroy -f  
5  
6 stages:  
7   - envcheck  
8   - lint  
9   - behave  
10  - cleanup  
11  
12 lint:  
13   stage: lint  
14   script:  
15     - bash linter.sh
```

```
17 envcheck:  
18   stage: envcheck  
19   script:  
20     - bash envcheck.sh  
21  
22 launch_lab:  
23   stage: behave  
24   script:  
25     - cd $CI_PROJECT_DIR/../../../../../../validation  
26     - vagrant up --color  
27     - sleep 10  
28     - behave --force-color  
29     - behave --version  
30     - sleep 10  
31  
32 flush_vagrant:  
33   stage: cleanup  
34   script:  
35     - vagrant destroy -f
```

Behave

- Behavior-Driven Development (BDD)
- Natural Language Tests

“feature” code

./validation/bgp.feature

1 Feature: Validate BGP

2

3 Scenario: Check BGP Neighbors

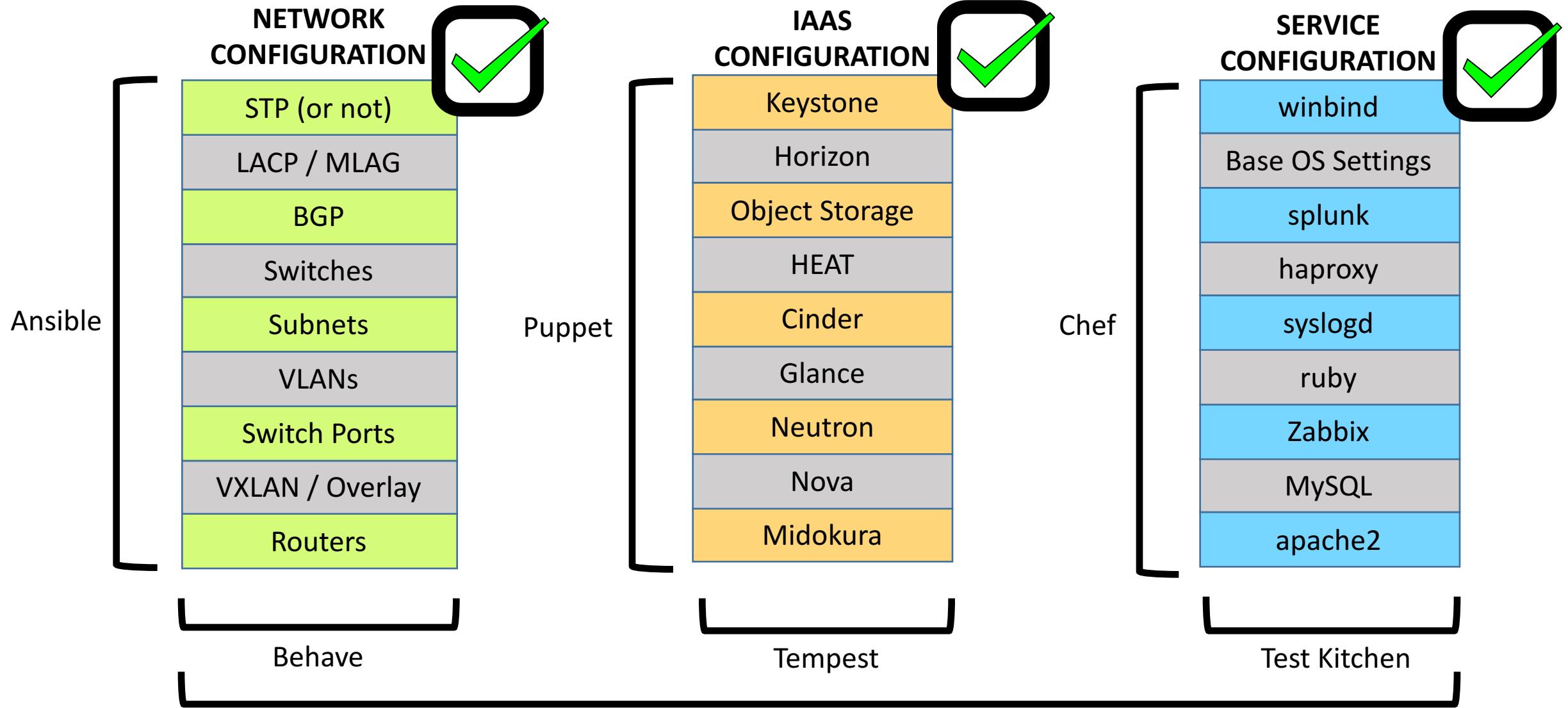
4 Given BGP is enabled
5 when neighbors are configured
6 then the neighbors should be up

Backing Python code

./validation/steps/bgp.py

```
317 @then('the neighbors should be up')
318 def step_implementation(context):
319     ...
320     Validate that the BGP state from Ansible is "Established"
321     ...
322
323     global spine_bgp_neighbor_config, list_of_spines
324     global leaf_bgp_neighbor_config, list_of_leafs
325     global server_bgp_neighbor_config, list_of_servers
326
327     for spine in list_of_spines:
328         json_data = json.loads(spine_bgp_neighbor_config[spine]["stdout"])
329
330         neighbor_list = json_data["peers"].keys()
331
332         for neighbor in neighbor_list:
333             if not json_data["peers"][neighbor]["state"] == "Established":
334                 assert False, spine + " peer " + neighbor + " not Established. Current state: " + j
335
336     for leaf in list_of_leafs:
337         json_data = json.loads(leaf_bgp_neighbor_config[leaf]["stdout"])
338
339         neighbor_list = json_data["peers"].keys()
340
341         for neighbor in neighbor_list:
342             if not json_data["peers"][neighbor]["state"] == "Established":
343                 assert False, leaf + " peer " + neighbor + " not Established. Current state: " + js
344
345     for server in list_of_servers:
346         json_data = json.loads(server_bgp_neighbor_config[server]["stdout"])
347
348         neighbor_list = json_data["peers"].keys()
349
350         for neighbor in neighbor_list:
351             if not json_data["peers"][neighbor]["state"] == "Established":
352                 assert False, server + " peer " + neighbor + " not Established. Current state: " +
353
354 ---
```

Full Stack Testing



GitLab CI Runner

Demo

- GitLab project overview
 - Repo, Pipelines, and Runners
- GitLab Runner overview
- Good change: Update MOTD
- Bad change: Routing typo / lint failure

```
$ behave --force-color
config.force_color True
Feature: Validate BGP # bgp.feature:1

Scenario: Check BGP Neighbors # bgp.feature:3
  Given BGP is enabled # steps/bgp.py:196
  Given BGP is enabled # steps/bgp.py:196 14.240s
  When neighbors are configured # steps/bgp.py:220
  When neighbors are configured # steps/bgp.py:220 0.001s
  Then the neighbors should be up # steps/bgp.py:317
  Then the neighbors should be up # steps/bgp.py:317 0.000s
Assertion Failed: spine1 peer swp2 not Established. Current state: Idle

Feature: Validate Interfaces are up and IPs are applied # interfaces.feature:1

Scenario: Check interfaces are up # interfaces.feature:3
  Given an interface is configured # steps/interfaces.py:138
  Given an interface is configured # steps/interfaces.py:138 3.052s
  Then the interfaces should be up # steps/interfaces.py:162
  Then the interfaces should be up # steps/interfaces.py:162 0.004s
```

```
Feature: Validate the webservers can be reached. # website.feature:1
  This will validate each server has apache2 configured and running.
  Then each server will try to reach every other server and fetch the index page
Scenario: Validate Web Server Access # website.feature:5
  Given a webserver is configured # steps/website.py:124
  Given a webserver is configured # steps/website.py:124 0.609s
  When apache is running # steps/website.py:136
  When apache is running # steps/website.py:136 0.510s
  Then the website should be accessible # steps/website.py:144
  Then the website should be accessible # steps/website.py:144 30.594s
Assertion Failed: Ansible Error: Socket error: timed out to http://10.0.0.2
```

Failing scenarios:

- bgp.feature:3 Check BGP Neighbors
- website.feature:5 Validate Web Server Access

1 feature passed, 2 failed, 0 skipped
1 scenario passed, 2 failed, 0 skipped
6 steps passed, 2 failed, 0 skipped, 0 undefined
Took 0m49.010s

ERROR: Build failed: exit status 1