Network Automation:

Do I Need Expensive Vendor Tools To Do Meaningful Automation?

P. Moore

NANOG 72

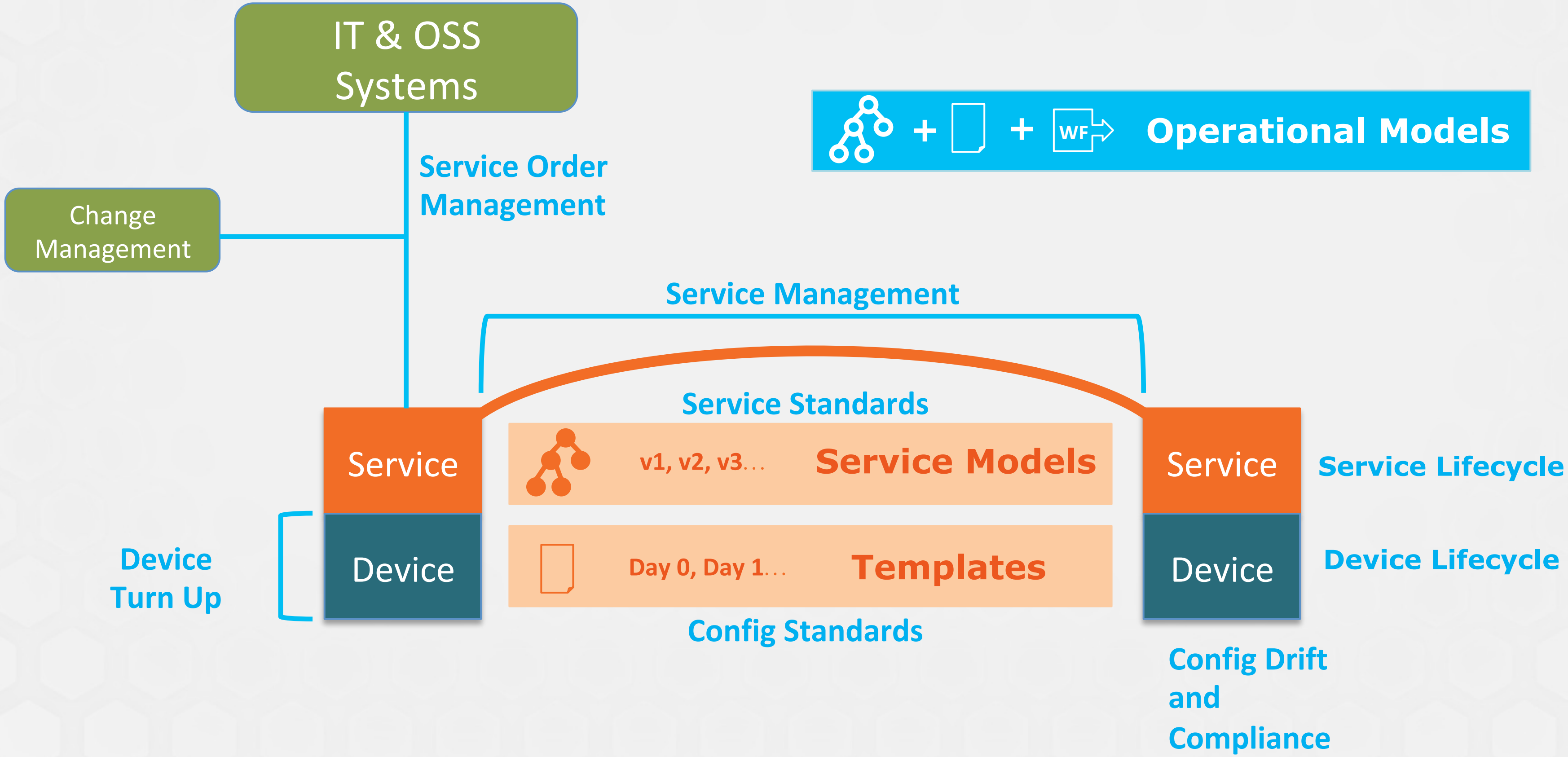
February 20, 2018

itential

Agenda

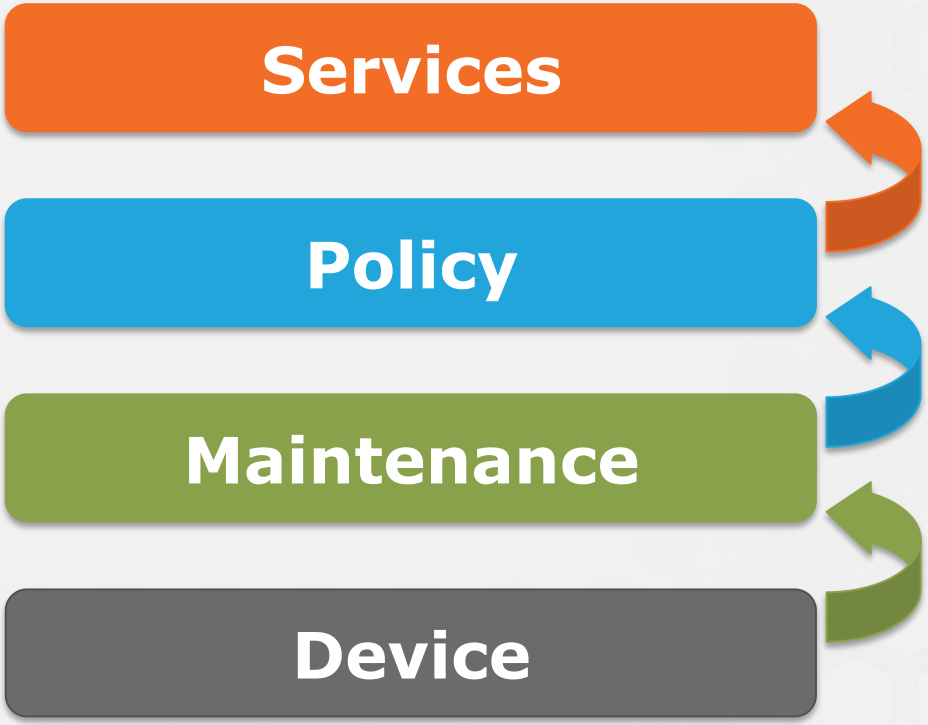
- Orchestration Domains
- An Onramp To Automation
- Open Source Tools
- Use Cases Considered
- Case Studies
- Conclusions

Orchestration Domains

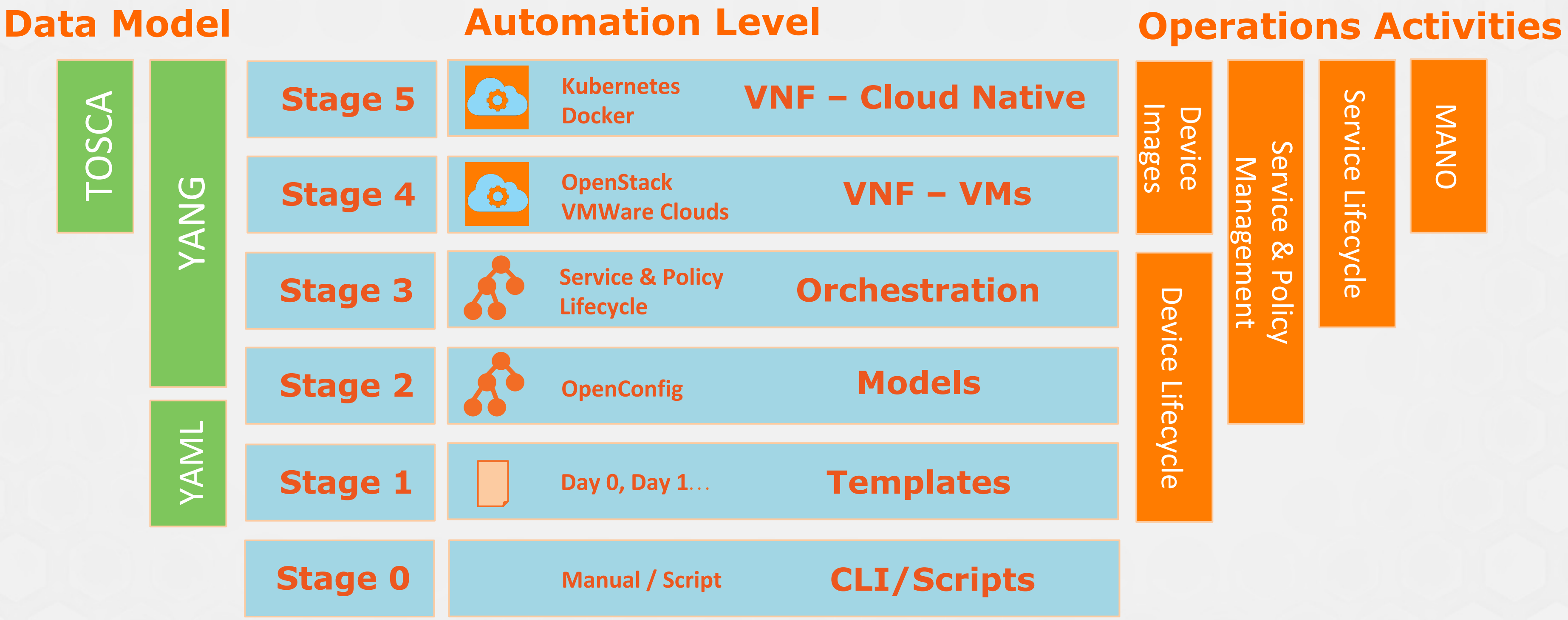


Domains Build Upon Each Other

4. Services
   - Model-based Service Management
3. Policy
   - Model-based Policy Management
2. Maintenance
   - Leverage Device Management to automate MOPs
1. Device (Foundational)
   - Configuration Management



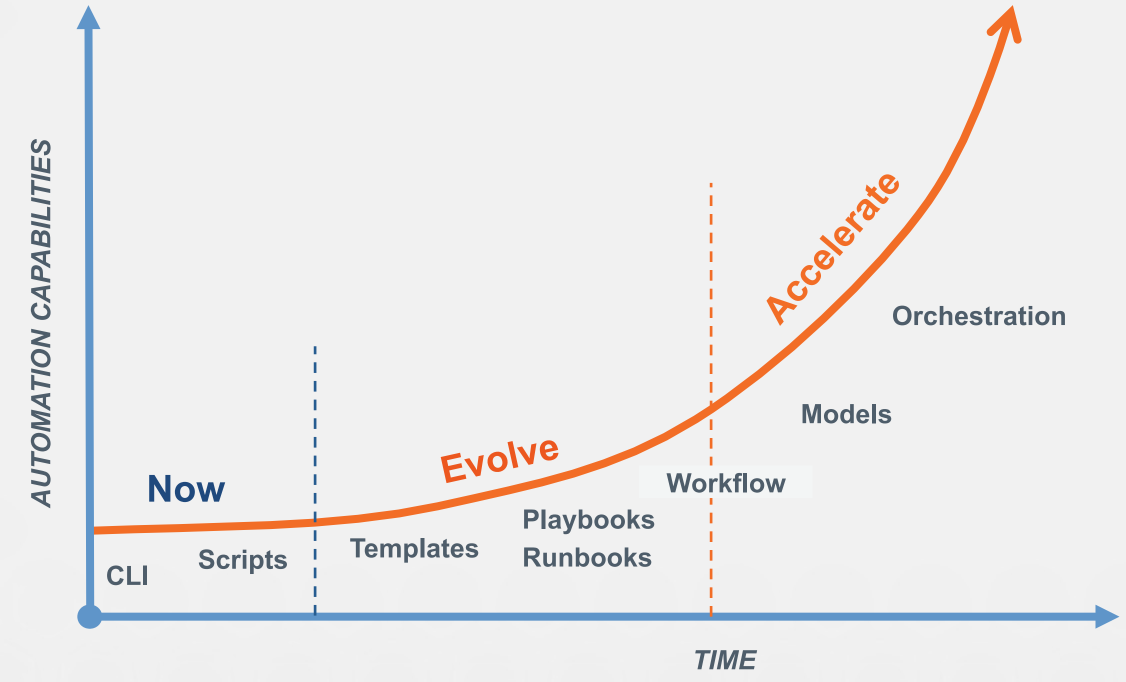
Domains Build Upon Each Other



Automation Onramp

- Address this on 3 fronts:
  - **People**: who will own network automation?
  - **Process**: define how you will manage the automation work
  - **Platform**: define the tools you will use
- Define your use cases thoroughly – "It is ALL about the use case!"
- Crawl > Walk > Run – start simple and expand
- "Evolve and Accelerate!"

Evolve then Accelerate



People

Who will own network automation?

- Dedicated Group? Not necessary unless you are looking to put a very formal program in place
- Roles Required:
  - **Network Automation Lead** – owns the automation efforts and works to remove roadblocks with other departments, vendors, etc.
  - **Automation Designer** – defines the work to be done, tools to use, workflow/steps of automation, and acts as technical lead
  - **Engineer** – works with the Designer to build the automation
  - **Subject Matter Experts (SME)** – provides knowledge in specific technology areas
- All roles may be filled by a single person in some cases, or may be 4 or more people in larger operations

Process: The Automation Factory

Define how you will manage the automation work

- Submission of automation requests
- Prioritization of which efforts to undertake
- Execution of automation efforts

Platform

Define the tools you will use

- Use tools you already have
- Leverage open source tools
  - Ansible & AWX
  - OpenDaylight, ONAP, etc.
  - Puppet, Chef, Salt, etc.
- Leverage vendors where the value makes sense

Examples: Tools

- **Ansible, Salt, Chef, Puppet, OpenDaylight**
  - Playbook scripting
  - YAML, YANG, NETCONF
- **AWX, Tower, ONAP**
  - Playbook Management
  - Workflow
- **Bitbucket, Github, etc.**
  - Playbook versioning
  - Config versioning (including diff)

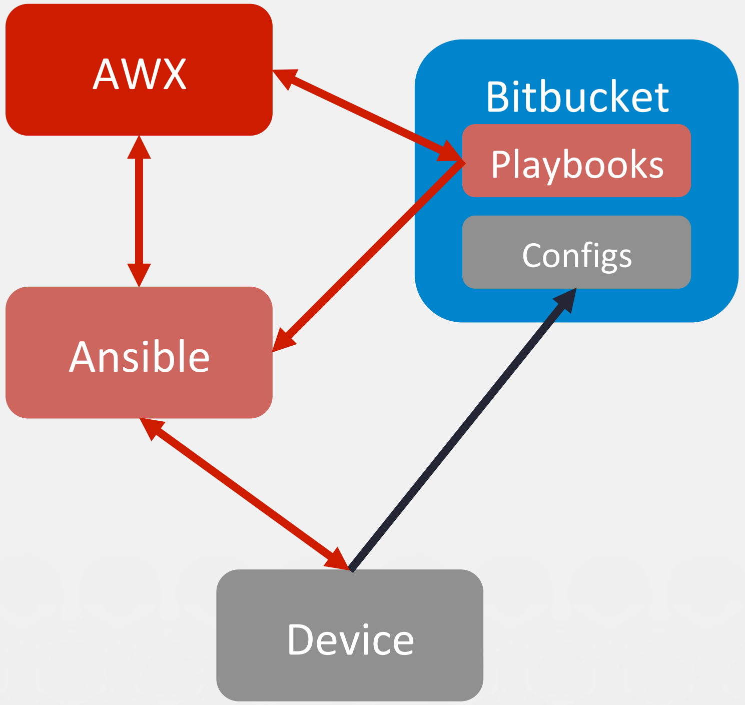
Use Cases for Examples

- **Config Management**
  - Backup
  - Config Diff
- **MOP Automation**
  - Sub interface turn up
  - OS Upgrade

Examples: Tool Architecture

AWX provides GUI based:

- Simple Workflow
- Playbook Management
- Job Management
- Simple Inventory



Bitbucket manages:

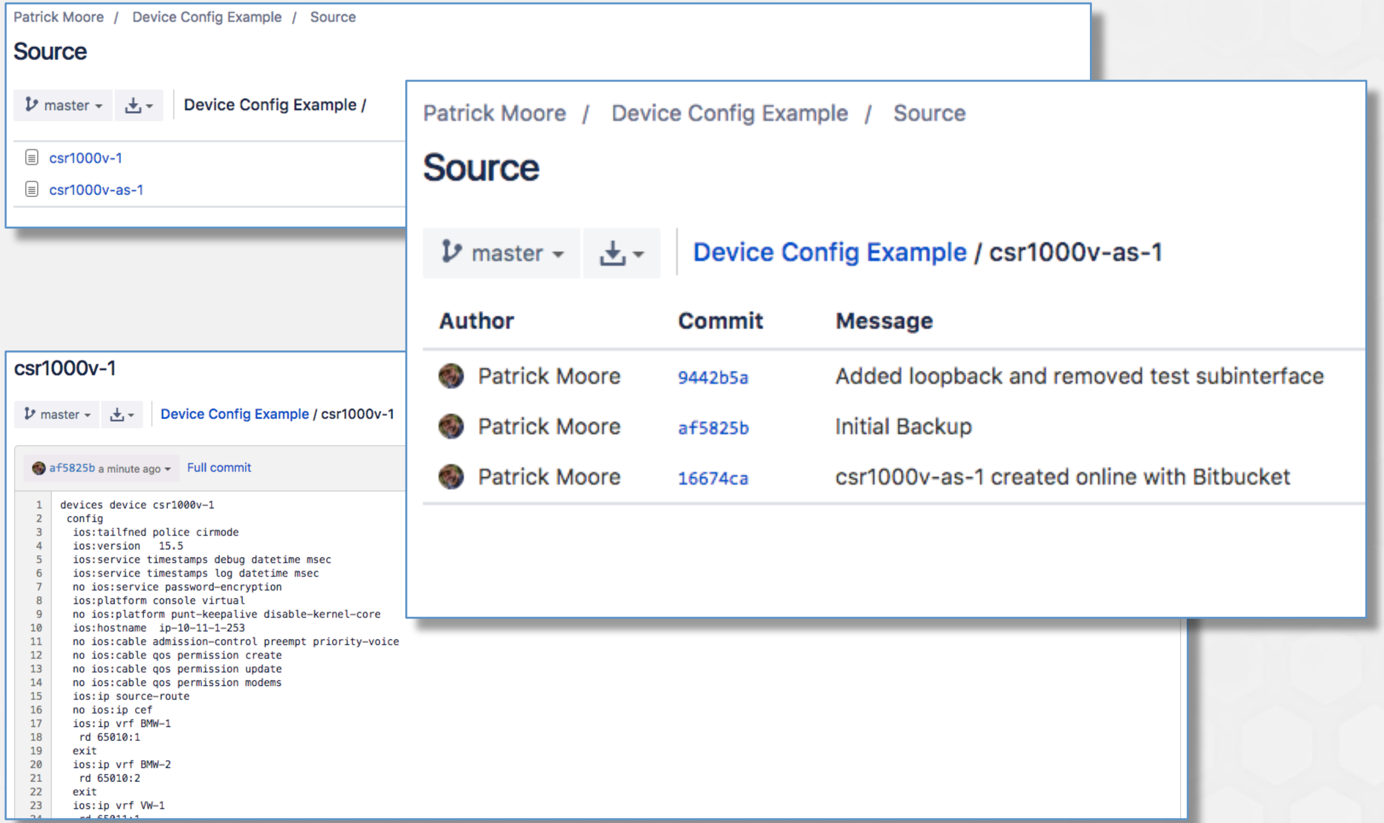
- Playbooks
- Configs

Ansible is the execution engine underneath AWX to communicate with devices

Example: Config Backup

Leveraging Bitbucket:

- Repository for configs
- History of changes to configs
- Ability to view previous configs
- Ability to see diffs between current version and previous versions



Patrick Moore / Device Config Example / Source

Source

csr1000v-1
csr1000v-as-1

Patrick Moore / Device Config Example / Source

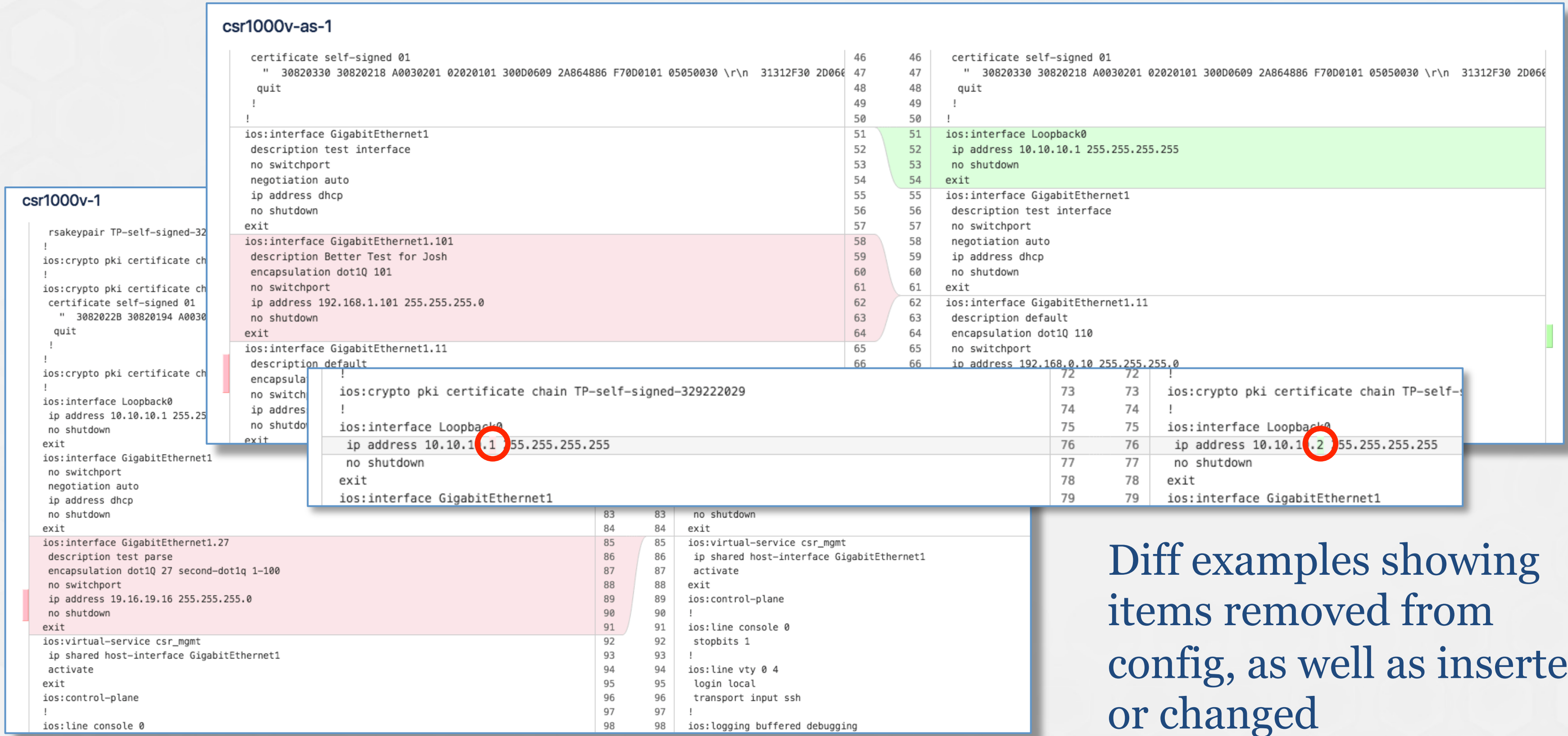
Source

Author	Commit	Message
Patrick Moore	9442b5a	Added loopback and removed test subinterface
Patrick Moore	af5825b	Initial Backup
Patrick Moore	16674ca	csr1000v-as-1 created online with Bitbucket

csr1000v-1

```
devices device csr1000v-1
 config
  ios:tailfned police cirmode
  ios:version   15.5
  ios:service timestamps debug datetime msec
  ios:service timestamps log datetime msec
  no ios:service password-encryption
  ios:platform console virtual
  no ios:platform punt-keepalive disable-kernel-core
  ios:hostname  ip-10-11-1-253
  no ios:cable admission-control preempt priority-voice
  no ios:cable qos permission create
  no ios:cable qos permission update
  no ios:cable qos permission modems
  ios:ip source-route
  no ios:ip cef
  ios:ip vrf BMW-1
   rd 65010:1
  exit
  ios:ip vrf BMW-2
   rd 65010:2
  exit
  ios:ip vrf VW-1
```

Example: Config Diff



csr1000v-as-1

```
 certificate self-signed 01
  "  30820330 30820218 A0030201 02020101 300D0609 2A864886 F70D0101 05050030 \r\n  31312F30 2D06
 quit
 !
!
ios:interface GigabitEthernet1
 description test interface
 no switchport
 negotiation auto
 ip address dhcp
 no shutdown
exit
ios:interface GigabitEthernet1.101
 description Better Test for Josh
 encapsulation dot1Q 101
 no switchport
 ip address 192.168.1.101 255.255.255.0
 no shutdown
exit
ios:interface GigabitEthernet1.11
 description default
 encapsula
 no switch
 ip addres
 no shutdo
exit
```

```
 certificate self-signed 01
  "  30820330 30820218 A0030201 02020101 300D0609 2A864886 F70D0101 05050030 \r\n  31312F30 2D06
 quit
 !
!
ios:interface Loopback0
 ip address 10.10.10.1 255.255.255.255
 no shutdown
exit
ios:interface GigabitEthernet1
 description test interface
 no switchport
 negotiation auto
 ip address dhcp
 no shutdown
exit
ios:interface GigabitEthernet1.11
 description default
 encapsulation dot1Q 110
 no switchport
 ip address 192.168.0.10 255.255.255.0
```

csr1000v-1

```
 rsakeypair TP-self-signed-32
!
ios:crypto pki certificate ch
!
ios:crypto pki certificate ch
 certificate self-signed 01
  "  3082022B 30820194 A0030
 quit
 !
!
ios:crypto pki certificate ch
!
ios:interface Loopback0
 ip address 10.10.10.1 255.25
 no shutdown
exit
ios:interface GigabitEthernet1
 no switchport
 negotiation auto
 ip address dhcp
 no shutdown
exit
ios:interface GigabitEthernet1.27
 description test parse
 encapsulation dot1Q 27 second-dot1q 1-100
 no switchport
 ip address 19.16.19.16 255.255.255.0
 no shutdown
exit
ios:virtual-service csr_mgmt
 ip shared host-interface GigabitEthernet1
 activate
exit
ios:control-plane
!
ios:line console 0
```

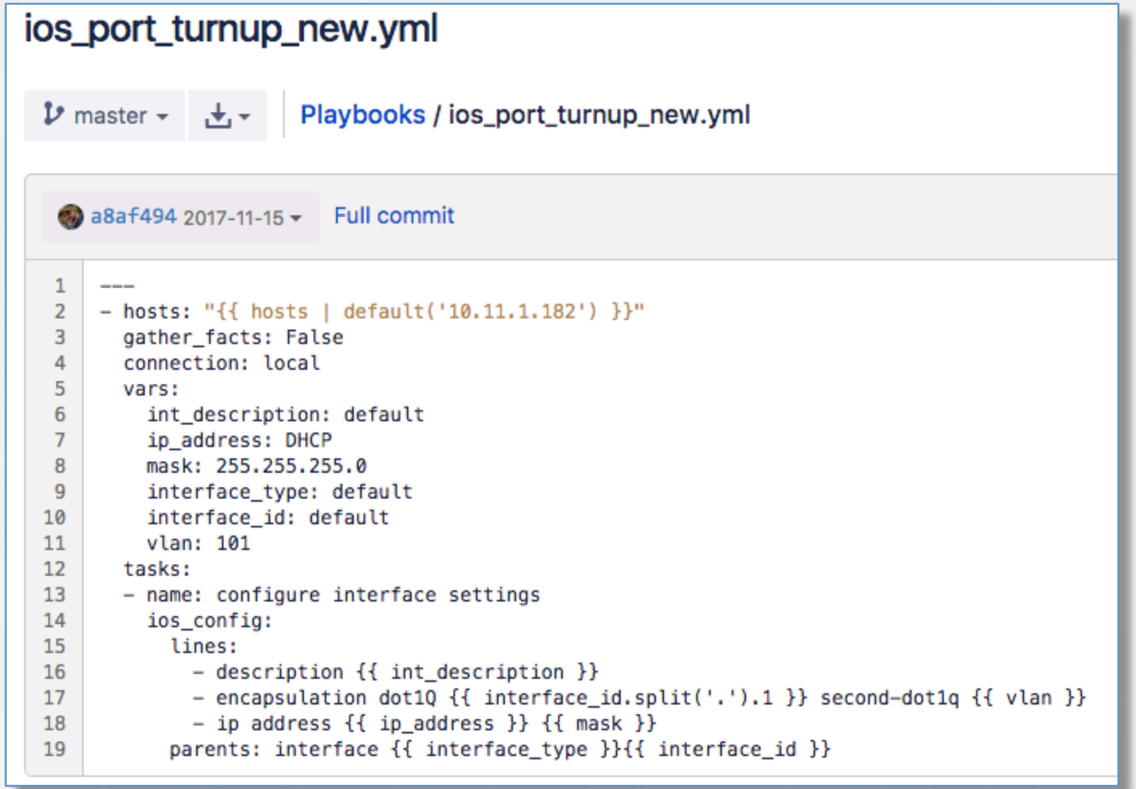
```
 no shutdown
exit
ios:virtual-service csr_mgmt
 ip shared host-interface GigabitEthernet1
 activate
exit
ios:control-plane
!
ios:line console 0
 stopbits 1
!
ios:line vty 0 4
 login local
 transport input ssh
!
ios:logging buffered debugging
```

```
!
ios:crypto pki certificate chain TP-self-signed-329222029
!
ios:interface Loopback0
 ip address 10.10.1.1 255.255.255.255
 no shutdown
exit
ios:interface GigabitEthernet1
```

```
!
ios:crypto pki certificate chain TP-self-s
!
ios:interface Loopback0
 ip address 10.10.1.2 255.255.255.255
 no shutdown
exit
ios:interface GigabitEthernet1
```

Diff examples showing items removed from config, as well as inserted or changed

Sub-interface Turn Up: Playbook


ios_port_turnup_new.yml

master | Playbooks / ios_port_turnup_new.yml

a8af494 2017-11-15 Full commit

```yaml
---
- hosts: "{{ hosts | default('10.11.1.182') }}"
  gather_facts: False
  connection: local
  vars:
    int_description: default
    ip_address: DHCP
    mask: 255.255.255.0
    interface_type: default
    interface_id: default
    vlan: 101
  tasks:
  - name: configure interface settings
    ios_config:
      lines:
        - description {{ int_description }}
        - encapsulation dot1Q {{ interface_id.split('.').1 }} second-dot1q {{ vlan }}
        - ip address {{ ip_address }} {{ mask }}
      parents: interface {{ interface_type }}{{ interface_id }}
```

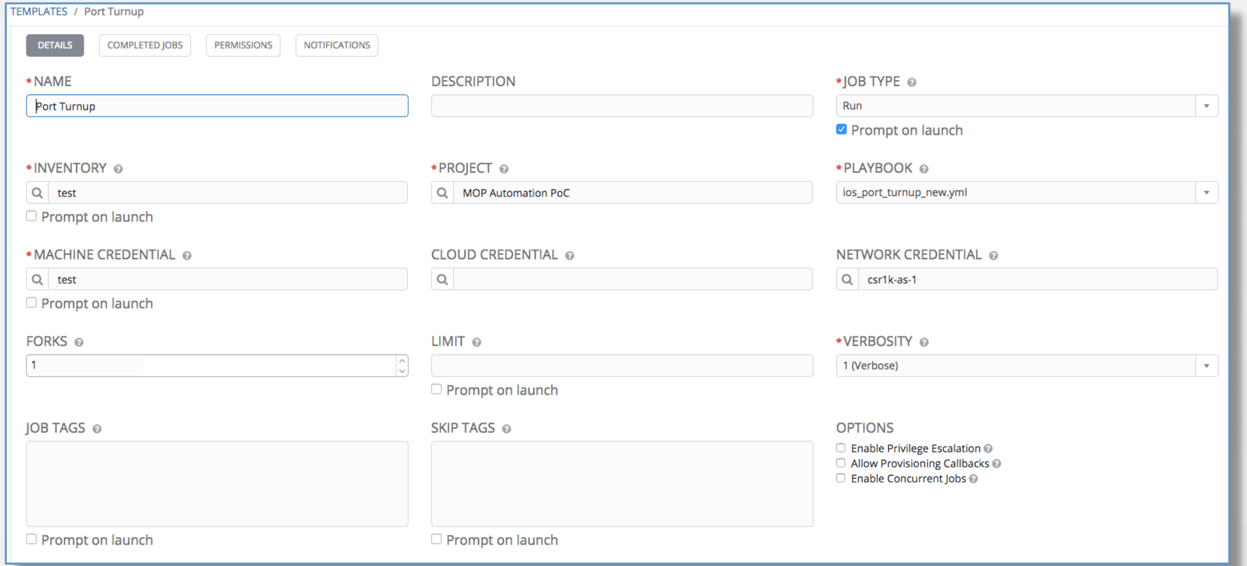
Your Playbooks should be:

- Variablized for **reuse** purposes
- Specific to a use case
- Broken into smaller executable “chunks” – even if you could combine more functions into the single playbook – for **reuse** purposes

Sub-interface Turn Up: Job Template

AWX allows for:

- Definition of Templates for jobs
- Management of credentials for network access
- Management of simple inventory of devices

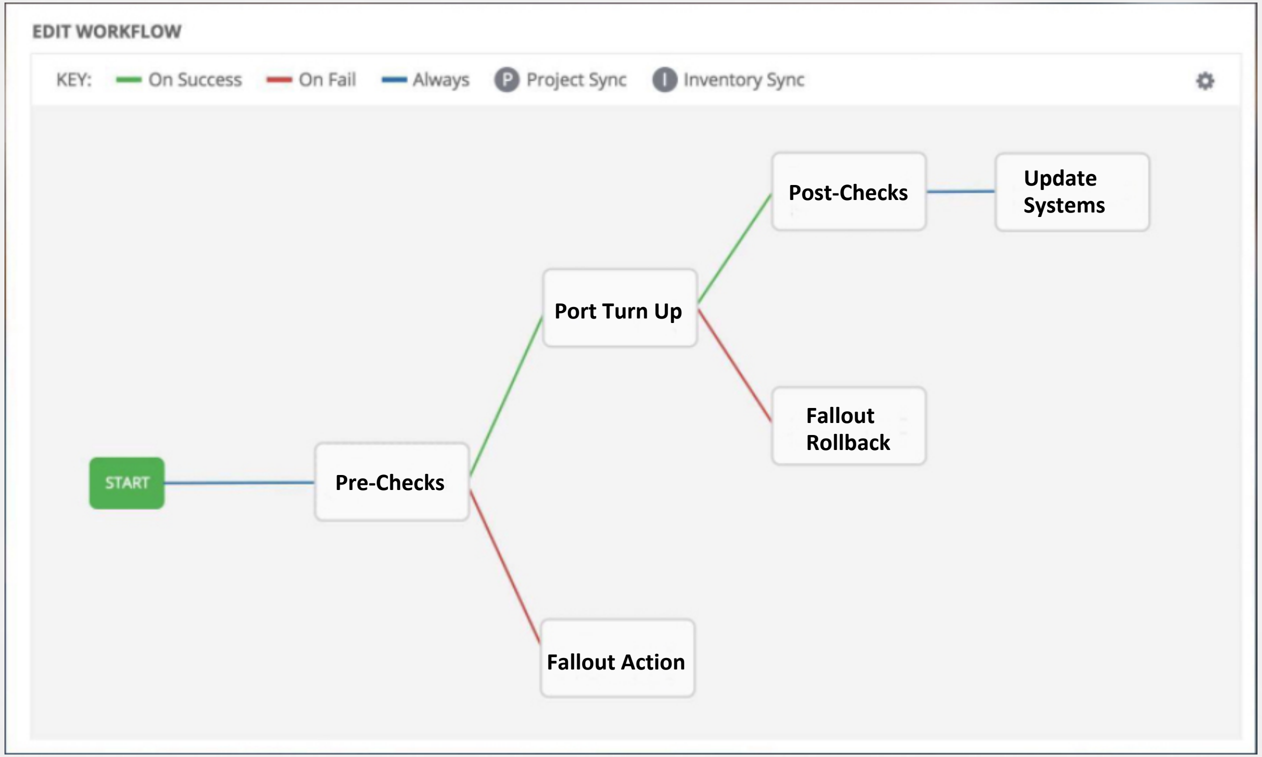


TEMPLATES / Port Turnup

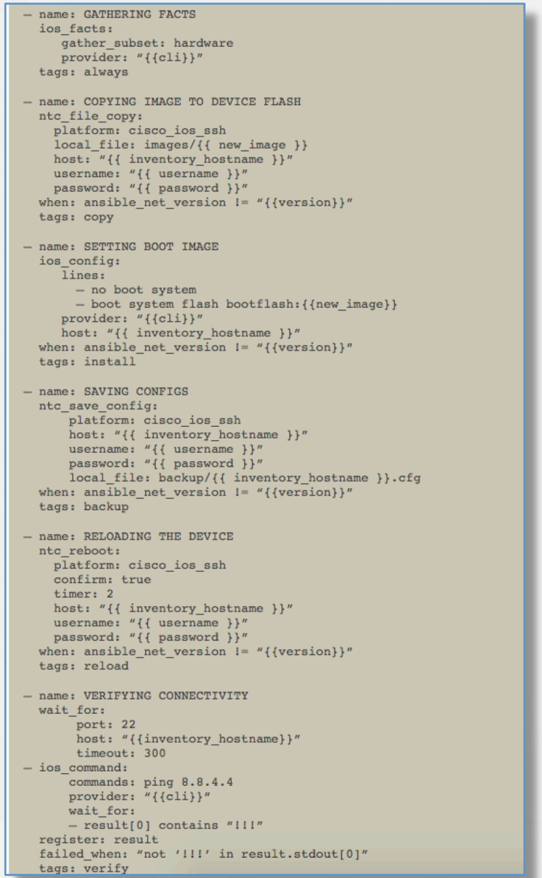
DETAILS | COMPLETED JOBS | PERMISSIONS | NOTIFICATIONS

*NAME: Port Turnup	DESCRIPTION:	*JOB TYPE: Run [✓ Prompt on launch]
*INVENTORY: test ☐ Prompt on launch	*PROJECT: MOP Automation PoC	*PLAYBOOK: ios_port_turnup_new.yml
*MACHINE CREDENTIAL: test ☐ Prompt on launch	CLOUD CREDENTIAL:	NETWORK CREDENTIAL: csr1k-as-1
FORKS: 1	LIMIT: ☐ Prompt on launch	*VERBOSITY: 1 (Verbose)
JOB TAGS: ☐ Prompt on launch	SKIP TAGS: ☐ Prompt on launch	OPTIONS: ☐ Enable Privilege Escalation ☐ Allow Provisioning Callbacks ☐ Enable Concurrent Jobs

Sub-interface Turn Up: Workflow



Device OS Upgrade: Playbook

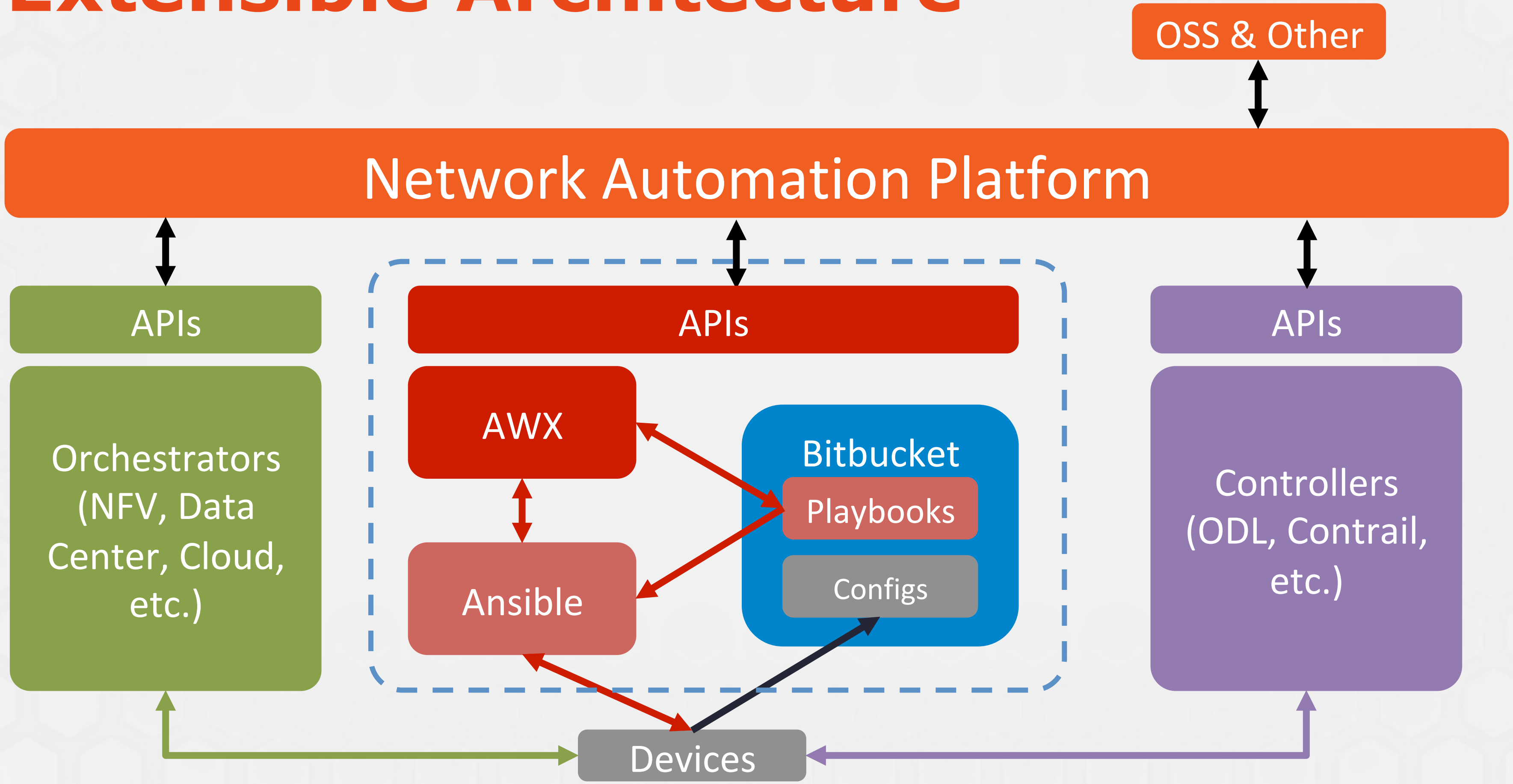


```
– name: GATHERING FACTS
  ios_facts:
     gather_subset: hardware
     provider: “{{cli}}”
  tags: always

– name: COPYING IMAGE TO DEVICE FLASH
  ntc_file_copy:
    platform: cisco_ios_ssh
    local_file: images/{{ new_image }}
    host: “{{ inventory_hostname }}”
    username: “{{ username }}”
    password: “{{ password }}”
  when: ansible_net_version != “{{version}}”
  tags: copy

– name: SETTING BOOT IMAGE
  ios_config:
     lines:
       – no boot system
       – boot system flash bootflash:{{new_image}}
     provider: “{{cli}}”
     host: “{{ inventory_hostname }}”
  when: ansible_net_version != “{{version}}”
  tags: install

– name: SAVING CONFIGS
  ntc_save_config:
      platform: cisco_ios_ssh
      host: “{{ inventory_hostname }}”
      username: “{{ username }}”
      password: “{{ password }}”
      local_file: backup/{{ inventory_hostname }}.cfg
  when: ansible_net_version != “{{version}}”
  tags: backup

– name: RELOADING THE DEVICE
  ntc_reboot:
    platform: cisco_ios_ssh
    confirm: true
    timer: 2
    host: “{{ inventory_hostname }}”
    username: “{{ username }}”
    password: “{{ password }}”
  when: ansible_net_version != “{{version}}”
  tags: reload

– name: VERIFYING CONNECTIVITY
  wait_for:
       port: 22
       host: “{{inventory_hostname}}”
       timeout: 300
– ios_command:
      commands: ping 8.8.4.4
      provider: “{{cli}}”
      wait_for:
      – result[0] contains “!!!”
  register: result
  failed_when: “not ‘!!!’ in result.stdout[0]”
  tags: verify
```

Example of a Playbook for OS Upgrade:

- This Playbook leverages the NTC-Ansible module that can be found at:
  - https://github.com/networktocode/ntc-ansible
- The example Playbook, and more detail, can be found at:
  - http://anastarsha.com/automating-cisco-device-upgrades-with-ansible/

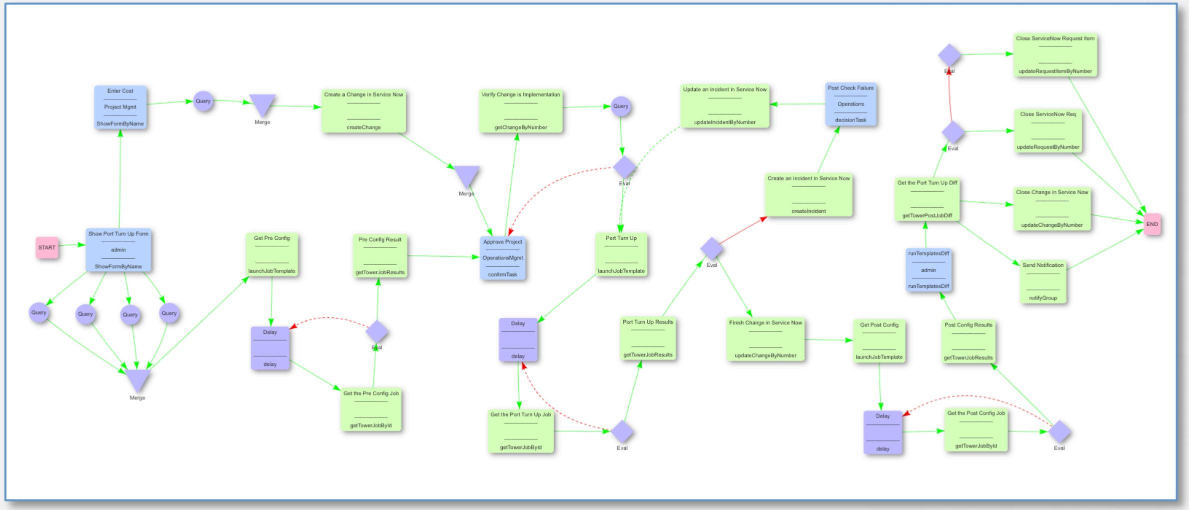
Extensible Architecture


OSS & Other
Network Automation Platform
APIs
Orchestrators (NFV, Data Center, Cloud, etc.)
APIs
AWX
Ansible
Bitbucket
Playbooks
Configs
APIs
Controllers (ODL, Contrail, etc.)
Devices

More Sophisticated Workflow

More advanced automation platforms allow for:

- Sophisticated workflows
- Cross tool workflows (e.g. Ansible, Puppet, Chef, Cisco NSO, etc.)
- Custom forms and apps to enable more complex automations



Questions?

References

- Network to Code Slack Channel: https://networktocode.herokuapp.com/
- Network To Code Ansible Module: https://github.com/networktocode/ntc-ansible
- Automating IOS Upgrades with Ansible: http://anastarsha.com/automating-cisco-device-upgrades-with-ansible/