Red Hat
Ansible Automation
Platform

ANSIBLE AUTOMATES

Network Resource Modules

Sean Cavanaugh
@IPvSean

Red Hat

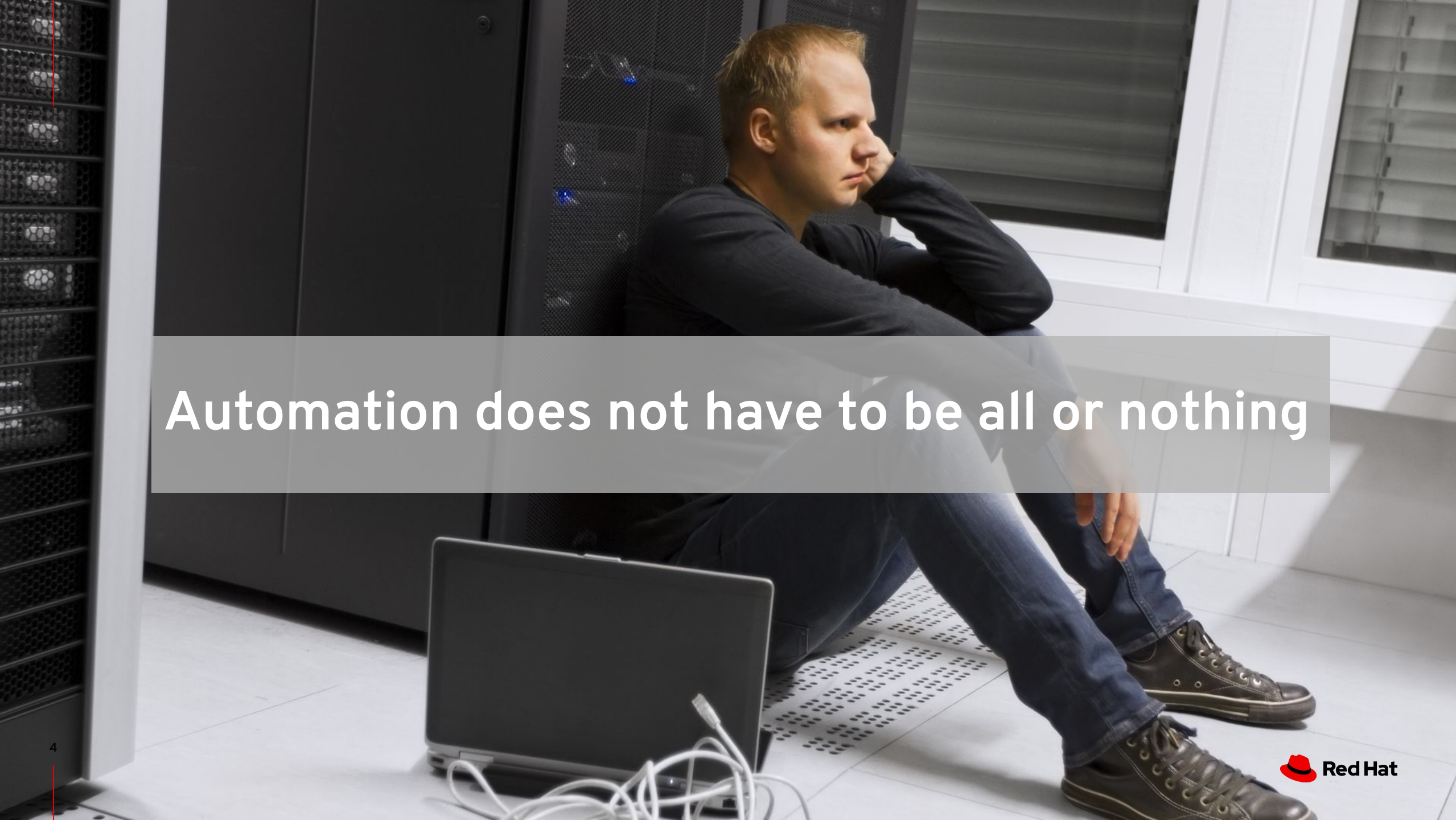
Agenda

- Setting the narrative
- About Network Modules
- Config Management
- Third Party Integrations
- Try it now

Obligatory Gartner Self-Validation Quote

"Automation is perhaps one of the highest yielding investments an entity running its network can make."

Gartner, Oct 2019



Automation does not have to be all or nothing

Red Hat

Automation is a cultural journey

Automation for an individual

Automation for an organization

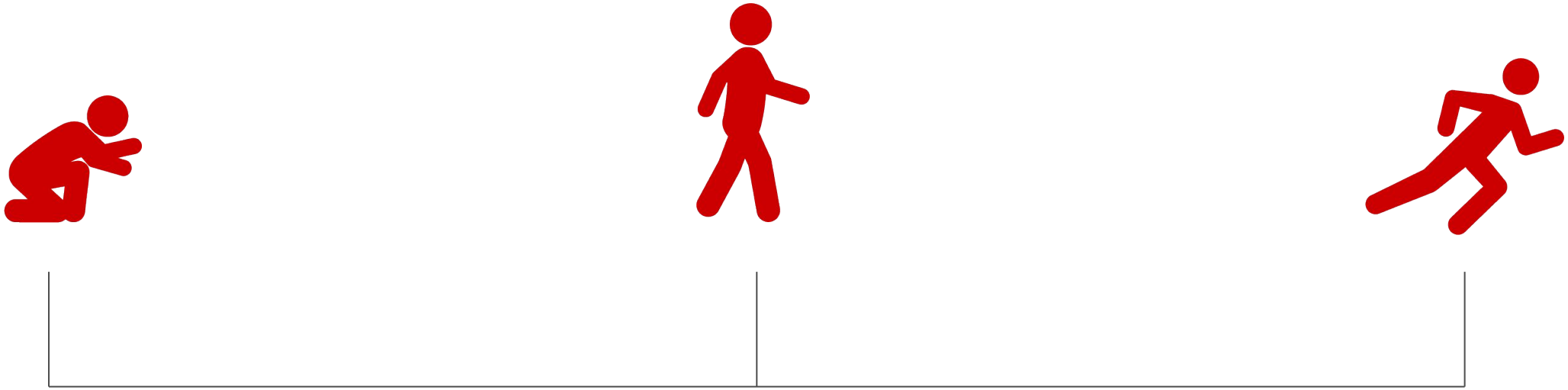
Start small, think big

Start with read-only
Get quick automation wins with no risk

Automate discrete configs
Enforce config policy for small pieces of config

Incrementally add automation
Don’t over complicate it on day 1

Crawl, Walk, Run



- “Manual” Automation
- Back up discrete inventory
- Use `command` and `config` modules

Days

- “DevOps Aware”
- CI/CD
- Use Resource Modules

Months

- “DevOps Essential”
- SoT
- Jinja2 Templating

Years

Red Hat

Network Modules

What are they and why do I care?

- Each module represents a task
- Easy integrations for multiple vendors
- Human readable

Preface: Four types of network modules

command

Command modules run arbitrary commands on a network device

config

Config modules allow configuration on the network device in a stateful (idempotent) way

facts

Fact modules return structured data about the network device

resource

Resource modules can read and configure a specific resource (e.g. vlans) on a network device

Red Hat

Preface: Four types of network modules

command

Command modules run arbitrary commands on a network device

Examples:
Arista EOS - eos_command
Cisco IOS - ios_command
Cisco IOS-XR - iosxr_command
Cisco NX-OS - nxos_command
Juniper Junos - junos_command
VyOS - vyos_command

config

Config modules allow configuration on the network device in a stateful (idempotent) way

Examples:
Arista EOS - eos_config
Cisco IOS - ios_config
Cisco IOS-XR - iosxr_config
Cisco NX-OS - nxos_config
Juniper Junos - junos_config
VyOS - vyos_config

facts

Fact modules return structured data about the network device

Examples:
Arista EOS - eos_facts
Cisco IOS - ios_facts
Cisco IOS-XR - iosxr_facts
Cisco NX-OS - nxos_facts
Juniper Junos - junos_facts
VyOS - vyos_facts

resource

Resource modules can read and configure a specific resource (e.g. vlans) on a network device

Examples:
Arista EOS - eos_vlans
Cisco IOS - ios_interfaces
Cisco IOS-XR - iosxr_lacp
Cisco NX-OS - nxos_vlans
Juniper Junos - junos_lldp_global
VyOS - vyos_lag_interfaces

Red Hat

Configuration Facts

Network Native Configuration (On box)

Convert to structured data

Example facts parameter
gather_network_resources: interfaces

```
ansible_facts:
  ansible_net_api: cliconf
  ansible_net_fqdn: rtr2
  ansible_net_gather_network_resources:
  - interfaces
  ansible_net_gather_subset:
  - default
  ansible_net_hostname: rtr2
  ansible_net_image: flash:EOS.swi
  ansible_net_model: vEOS
  ansible_net_python_version: 2.7.5
  ansible_net_serialnum:
D00E130991A37B49F970714D8CCF7FCB
  ansible_net_system: eos
  ansible_net_version: 4.22.0F
  ansible_network_resources:
    interfaces:
    - enabled: true
      name: Ethernet1
    - enabled: true
      name: Loopback0
```

<<rest of output removed for slide brevity>>

Red Hat

Resource Modules

```yaml
interfaces:
- enabled: true
  name: Ethernet1
  mtu: '1476'
- enabled: true
  name: Loopback0
- enabled: true
  name: Loopback1
- enabled: true
  mtu: '1476'
  name: Tunnel0
- enabled: true
  name: Ethernet1
- enabled: true
  name: Tunnel1
- enabled: true
  name: Ethernet1
```

Example structured data
interfaces

Resource Module

Network Native Configuration (On box)

Example resource module
eos_interfaces

Example Rendered configuration

```
interface Tunnel0
   mtu 1476
!
```

Red Hat

Resource modules have corresponding facts

resource module	gather_network_resources
*os_interfaces (e.g. ios_interfaces for Cisco IOS-XE)	interfaces
*os_l2_interfaces	l2_interfaces
*os_l3_interfaces	l3_interfaces
*os_lacp	lacp
*os_vlans	vlans

Please refer to *os_facts module documentation for full list of gather_network_resources
e.g. for Arista eos refer to
https://docs.ansible.com/ansible/devel/modules/eos_facts_module.html

Red Hat

Resource modules have corresponding facts

resource

eos_interfaces

```yaml
---
- name: convert interface to structured data
  hosts: arista
  gather_facts: false
  tasks:

   - name: grab arista eos info
     eos_facts:
       gather_subset: min
       gather_network_resources: interfaces
```

Structured data is malleable

```yaml
interfaces:
- enabled: true
  name: Ethernet1
  mtu: '1476'
- enabled: true
  name: Loopback0
- enabled: true
  name: Loopback1
- enabled: true
  mtu: '1476'
  name: Tunnel0
```

Jinja2 Templating

Customized Report

Example structured data
interfaces

Example resource module
eos_interfaces

Examples reports
HTML, CSV, markdown

Ansible Network Automation Example Report

Hostname	Transport	Platform	Code Version	Serial Number
rtr1	network_cli	Cisco ios	16.09.02	9I23FI7K2S7
rtr2	network_cli	Arista eos	4.22.0F	D00E130991A37B49F970714D8CCF7FCB

rtr1

>Interfaces Facts - Click to see details

Interface Name	Description	Duplex	Enabled	MTU	Speed
loopback0	none	default	True	default	default
loopback1	none	default	True	default	default
loopback100	none	default	True	default	default
GigabitEthernet1	none	default	True	default	default

>Interfaces L3 Facts - Click to see details

Interface Name	IPv4	IPv6
loopback0	192.168.1.101 255.255.255.0	
loopback1	10.1.1.101 255.255.255.0	
loopback100	5.6.8.1 255.255.255.0 secondary 5.6.7.8 255.255.255.0	fc00::2/64 fec0::3/64
GigabitEthernet1	dhcp	

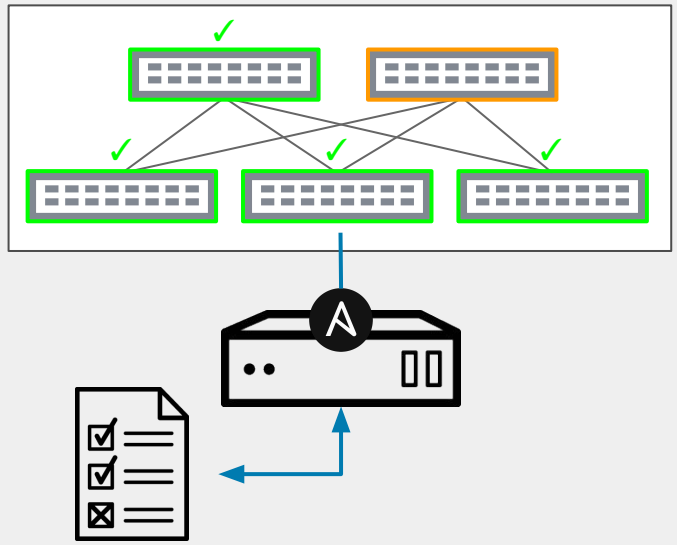
rtr2

>Interfaces Facts - Click to see details

Config Management

Why is it important?

- Enforce configuration policy
- Correct configuration drift
- Force multiplier for config changes
- Lockdown configs to best practices



Red Hat

Configuration Management Methods

- Config modules with raw configs

```
- name: ensure that the desired snmp strings are present
  ios_config:
    commands:
      - snmp-server community ansible-public RO
      - snmp-server community ansible-private RW
```

- Config modules with jinja2 templating

```
- name: ensure that the desired snmp strings are present
  ios_config:
    config: "{{ lookup('template', 'template.j2') }}"
```

- Resource modules

```
- name: ensure that the IP address information is accurate
  ios_l3_interfaces:
    config: "{{ ip_address_info }}"
```

Red Hat

Resource module config management

YAML variables

```yaml
ip_address_info:
  - name: Loopback100
    ipv4:
    - address: 10.10.10.1/24
    ipv6:
      - address: fc00::100/64
      - address: fc00::101/64
  - name: Loopback200
    ipv4:
    - address: 10.10.20.1/24
```

Ansible Playbook Task

```yaml
- name: ensure that the IP address information is accurate
  ios_l3_interfaces:
    config: "{{ ip_address_info }}"
    state: merged
```

Ansible Playbook output

```
[student1@ansible ~]$ ansible-playbook ip_address.yml

PLAY [rtr2] ***********************************************************************

TASK [ensure that the IP address information is accurate] ***********************
changed: [rtr2]

PLAY RECAP *********************
rtr2                       ok=1    changed=1    unreachable=0    failed=1    skipped=0    rescued=0    ignored=0
```

Red Hat

Resources are bidirectional capable

Facts will always be returned in the data model the corresponding module expects

```yaml
---
- name: example of facts being pushed right back to device
  hosts: arista
  gather_facts: false
  tasks:

   - name: grab arista eos facts
     eos_facts:
       gather_subset: min
       gather_network_resources: l3_interfaces

   - name: ensure that the IP address information is accurate
     eos_l3_interfaces:
       config: "{{ansible_network_resources['l3_interfaces']}}"
     register: result

   - name: ensure config did not change
     assert:
       that: not result.changed
```

Red Hat

Resource modules - **state** parameter

- **`merged`**: configuration merged with the provided configuration (<u>default</u>)
- **`replaced`**: configuration of provided resources will be replaced with the provided configuration
- **`overridden`**: The configuration of the provided resources will be replaced with the provided configuration, extraneous resource instances will be removed
- **`deleted`**: The configuration of the provided resources will be deleted/defaulted

Resource modules - **state** parameter

Partial running Configuration on Cisco IOS-XE rtr1

```
interface Loopback100
 ip address 10.10.1.100 255.255.255.0
 ipv6 address FC00::100/64
```

YAML SOT (Source of Truth)

```yaml
config:
- name: loopback100
  ipv6:
  - address: fc00::100/64
  - address: fc00::101/64
```

Ansible Task

```yaml
- name: configure l3 interface
  ios_l3_interfaces:
    config: "{{config}}"
    state: merged
```

merged →

```
interface Loopback100
 ip address 10.10.1.100 255.255.255.0
 ipv6 address FC00::100/64
 ipv6 address FC00::101/64
```

replaced →

```
interface Loopback100
 no ip address
 ipv6 address FC00::100/64
 ipv6 address FC00::101/64
```

overridden →

inappropriate use case, would remove all interfaces from configuration except Loopback100 config provided

deleted →

```
interface Loopback100
 no ip address
```

Resource modules - return values

- **`before`**
  The configuration prior to module execution is always returned.

- **`commands`**
  delta command set for the device

- **`after`**
  the configuration post module execution

Resource modules - `return` parameters

Running Configuration on Cisco IOS-XE rtr1

```
interface Loopback100
 ip address 10.10.10.1 255.255.255.0
 ipv6 address FC00::100/64
```

YAML SOT (Source of Truth)

```yaml
config:
- name: loopback100
  ipv6:
  - address: fc00::100/64
  - address: fc00::101/64
```

Ansible Task

```yaml
- name: configure l3 interface
  ios_l3_interfaces:
    config: "{{config}}"
    state: merged
```

before

```yaml
- name: Loopback100
  ipv4:
  - address: 10.10.10.1/24
  ipv6:
  - address: fc00::100/64
```

commands

```yaml
commands:
  - interface Loopback100
  - ipv6 address fc00::101/64
```

after

```yaml
- name: Loopback100
  ipv4:
  - address: 10.10.10.1/24
  ipv6:
  - address: fc00::100/64
  - address: fc00::101/64
```

Resource modules enable platform agnosticism

YAML SOT (Source of Truth)

```yaml
config:
- name: lo0
  ipv4:
  - address: 10.10.10.1/24
- ipv6:
  - address: fc00::100/64
  - address: fc00::101/64
```

Ansible Task

```yaml
- name: configure l3 interface
  junos_l3_interfaces:
    config: "{{config}}"
    state: merged
```

commands

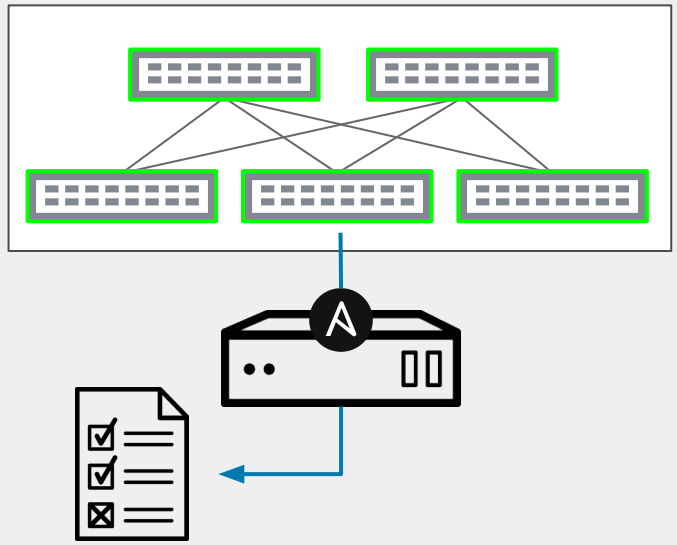
Implementation details are taken care of by the resource module

```
commands:
 <nc:interfaces
xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"><nc:interface><nc:name>l
o0</nc:name><nc:unit><nc:name>0</nc:name><nc:family><nc:inet><nc:address><n
c:name>10.10.10.1/24</nc:name></nc:address></nc:inet></nc:family><nc:family
><nc:inet6><nc:address><nc:name>fc00::100/64</nc:name></nc:address><nc:addr
ess><nc:name>fc00::101/64</nc:name></nc:address></nc:inet6></nc:family></nc
:unit></nc:interface></nc:interfaces>
```

Creating a Source of Truth

Why is it important?

- Incremental steps to infrastructure as code
- Structured variables for network config
- Simple and agnostic data model

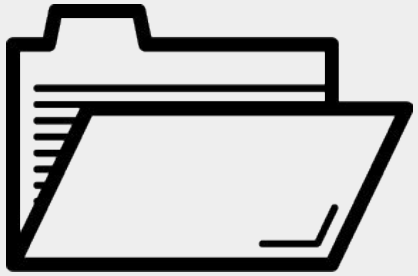


Red Hat

Create a structured SOT (source of truth)

Network Native Configuration (on box)

Convert to structured data



Store in host_vars

Convert facts into flat-file variables

```yaml
---
- hosts: rtr2
  gather_facts: false
  tasks:

  - name: grab info
    eos_facts:
      gather_subset: min
      gather_network_resources: all

  - name: push structured data to hostvars
    copy:
      content: "{{ansible_network_resources | to_nice_yaml}}"
      dest: "{{playbook_dir}}/host_vars/{{inventory_hostname}}"
```

Red Hat

Convert facts into flat-file variables

```
[user@ansible ~]$ cat host_vars/rtr2
```

```yaml
interfaces:
- enabled: true
  name: Ethernet1
  mtu: '1476'
- enabled: true
  name: Loopback0
- enabled: true
  name: Loopback1
- enabled: true
  mtu: '1476'
  name: Tunnel0
- enabled: true
  name: Ethernet1
- enabled: true
  name: Tunnel1
- enabled: true
  name: Ethernet1
vlans:
- name: None
  vlan_id: 2
- name: None
  vlan_id: 3
- name: None
  vlan_id: 4
- name: None
  vlan_id: 100
- name: None
  state: suspend
  vlan_id: 5
<... rest of output removed for brevity...>
```

- Each resource is a list of dicts
- The key is the resource (e.g. interfaces,vlans)
- Resources that are not used will show empty (e.g. `lacp: {}`)

Red Hat

Each resource can be treated individually

```yaml
---
- hosts: rtr2
  gather_facts: false
  tasks:
    - name: push interfaces
      eos_interfaces:
        config: "{{interfaces}}"

    - name: push l2 interfaces
      eos_l2_interfaces:
        config: "{{l2_interfaces}}"

    - name: push l3 interfaces
      eos_l3_interfaces:
        config: "{{l3_interfaces}}"

    - name: push VLANs
      eos_vlans:
        config: "{{vlans}}"
```

Red Hat

Ansible Playbook output

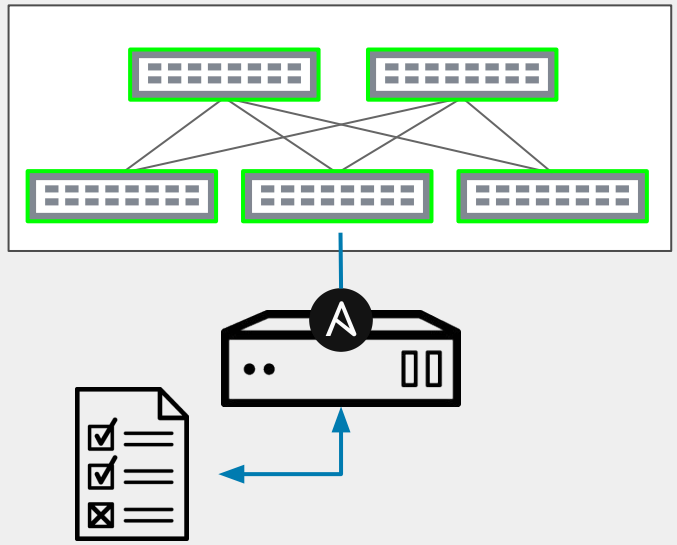
```
[student1@ansible ~]$ ansible-playbook eos_all.yml

PLAY [rtr2] ********************************************************************************

TASK [push interfaces] *********************************************************************
ok: [rtr2]

TASK [push l2 interfaces] ******************************************************************
ok: [rtr2]

TASK [push l3 interfaces] ******************************************************************
ok: [rtr2]

TASK [push l3 interfaces] ******************************************************************
ok: [rtr2]

PLAY RECAP *********************************************************************************
rtr2                       ok=4    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

Red Hat

Third-party Integrations

Why is it important?

- Brownfield networks have brownfield tools
- Tower's API
- Ansible Tower has "in the box" integrations



Types of Integrations

Automate the automators

SDN and other Controllers via API

CI/CD

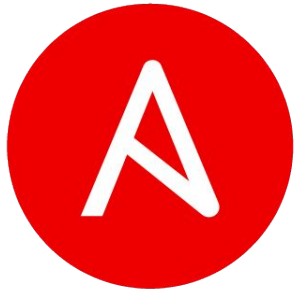
SCM, stage/prod pipelines, outage windows

Business Workflow

Ticketing, process approvals, dashboards


WIRELESS
aws
Microsoft Azure
UCS:
LINUX
WINDOWS
SWITCHES
IOS
IOS-XR
NX-OS
ROUTERS
ASA FIREWALL
CISCO ACI
LOAD BALANCERS
CISCO NSO

Red Hat



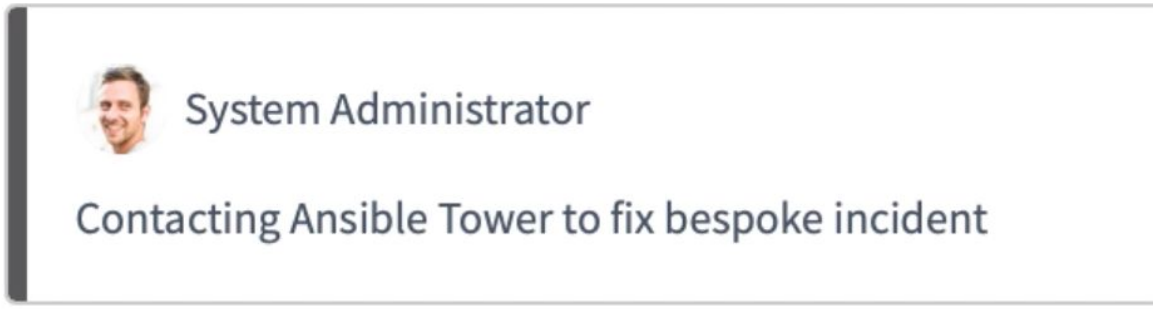
+ servicenow™

https://ansible.github.io/workshops/demos/

Ansible Tower uses `snow_record` module

```
TASK [MARK THE TICKET AS RESOLVED] ***********
******************
changed: [rtr1]

TASK [MARK THE TICKET AS CLOSED] *************
******************
changed: [rtr1]
```

ServiceNow uses the Ansible Tower API



Red Hat

ANSIBLE AUTOMATES

THANK YOU

youtube.com/AnsibleAutomation

facebook.com/ansibleautomation

linkedin.com/company/Red-Hat

twitter.com/ansible

Thank you

Red Hat is the world's leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.

linkedin.com/company/red-hat

youtube.com/user/AnsibleAutomation

facebook.com/AnsibleAutomation

twitter.com/Ansible

Red Hat