ANSIBLE

AUTOMATION FOR NETWORK INFRASTRUCTURE

David Clauvel
Specialist Solutions Architect
@automaticdavid

January 28, 2019

redhat

Why Automation for Networks?


MANAGING NETWORKS
HASN’T CHANGED
IN 30 YEARS.

WHAT IS THE PRIMARY METHOD OF MAKING NETWORK CHANGES IN YOUR ENVIRONMENT?

ANSIBLE

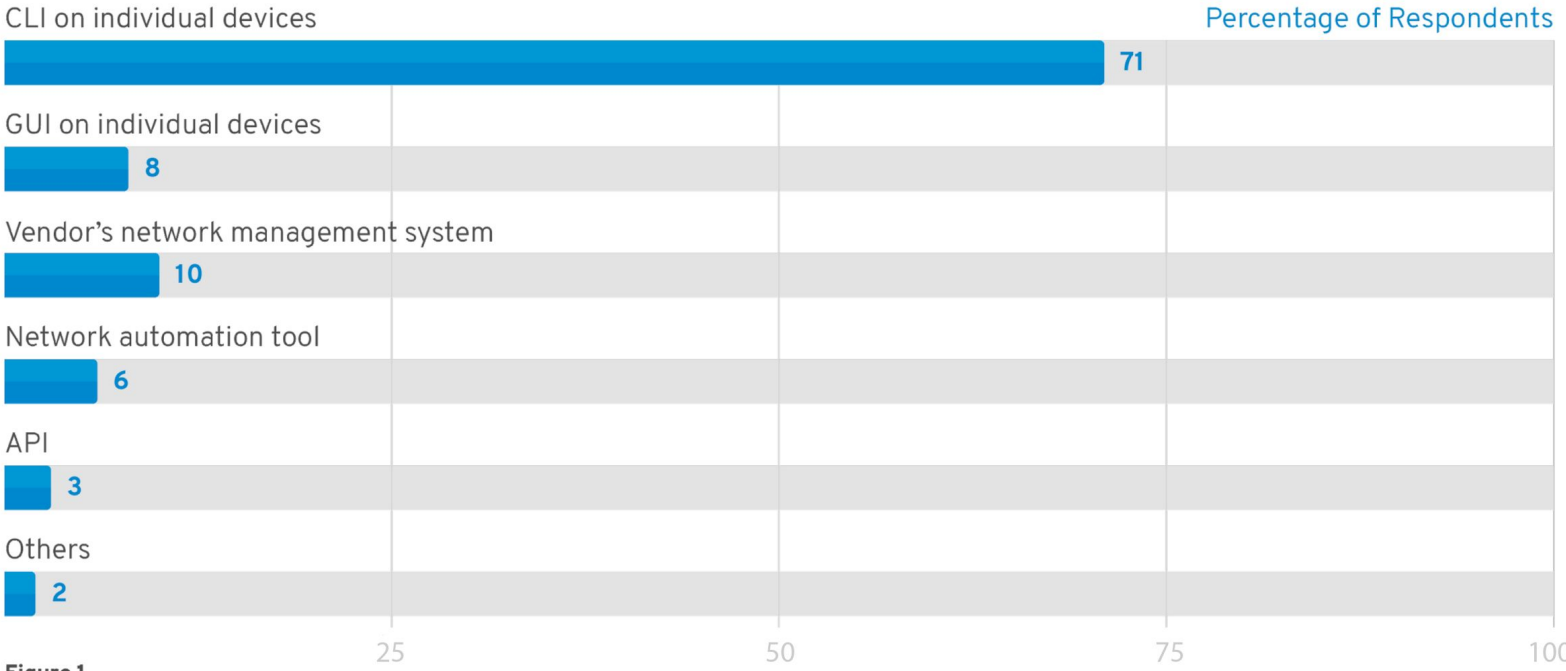


Figure 1
Primary Method for Making Network Changes

Source: Gartner, *Look Beyond Network Vendors for Network Innovation.* January 2018. Gartner ID: G00349636. (n=64)

redhat.

WHY HASN'T NETWORKING CHANGED?

PEOPLE

Domain specific skillsets

Vendor oriented experience

Siloed organizations

Legacy operational practices

PRODUCTS

Infrastructure-focused features

Baroque, CLI-only methodologies

Siloed technologies

Monolithic, proprietary platforms

BIGGEST CHALLENGE FOR ENTERPRISES: CULTURE!

Traditional Network Ops

- Legacy culture
- Risk averse
- Proprietary solutions
- Siloed from others
- “Paper” practices, MOPs
- “Artisanal” networks

Next-Gen Network Ops

- Community culture
- Risk aware
- Open solutions
- Teams of heroes
- Infrastructure as code
- Virtual prototyping / DevOps

WHY ANSIBLE FOR NETWORK AUTOMATION?

ANSIBLE

Configure, validate, & ensure continuous compliance for physical network devices

BUILD
with Red Hat Ansible Engine

- Get automating quickly.
- Integrate multi-vendor configurations.
- Ideal for both existing and greenfield.

Automate discrete tasks

MANAGE
with Red Hat Ansible Engine

- Methodically track configuration drift.
- Make changes across any set of network devices.
- Validate that changes were successful.

Automate business processes

SCALE
with Red Hat Ansible Tower

- Ensure ongoing steadystate on a schedule.
- Use role-based access controls with specific teams.
- Integrate external third-party solutions with RESTful API.

Orchestrate & operationalize automation

RED HAT ANSIBLE Engine

RED HAT ANSIBLE Engine

RED HAT ANSIBLE Tower

redhat.

HOW ARE THEY GETTING THERE?

"Start small, Think big!"

Infrastructure as YAML

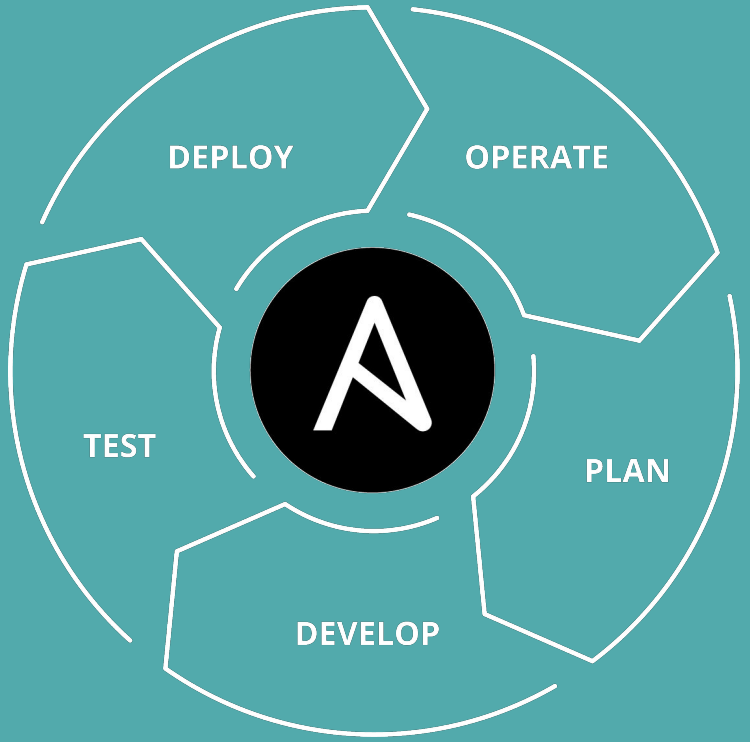
- Backups/restores can be automated
- Manage "golden" versions of configurations

Configuration management

- Changes can be incremental or wholesale
- Make it part of the process: agile, waterfall, etc.

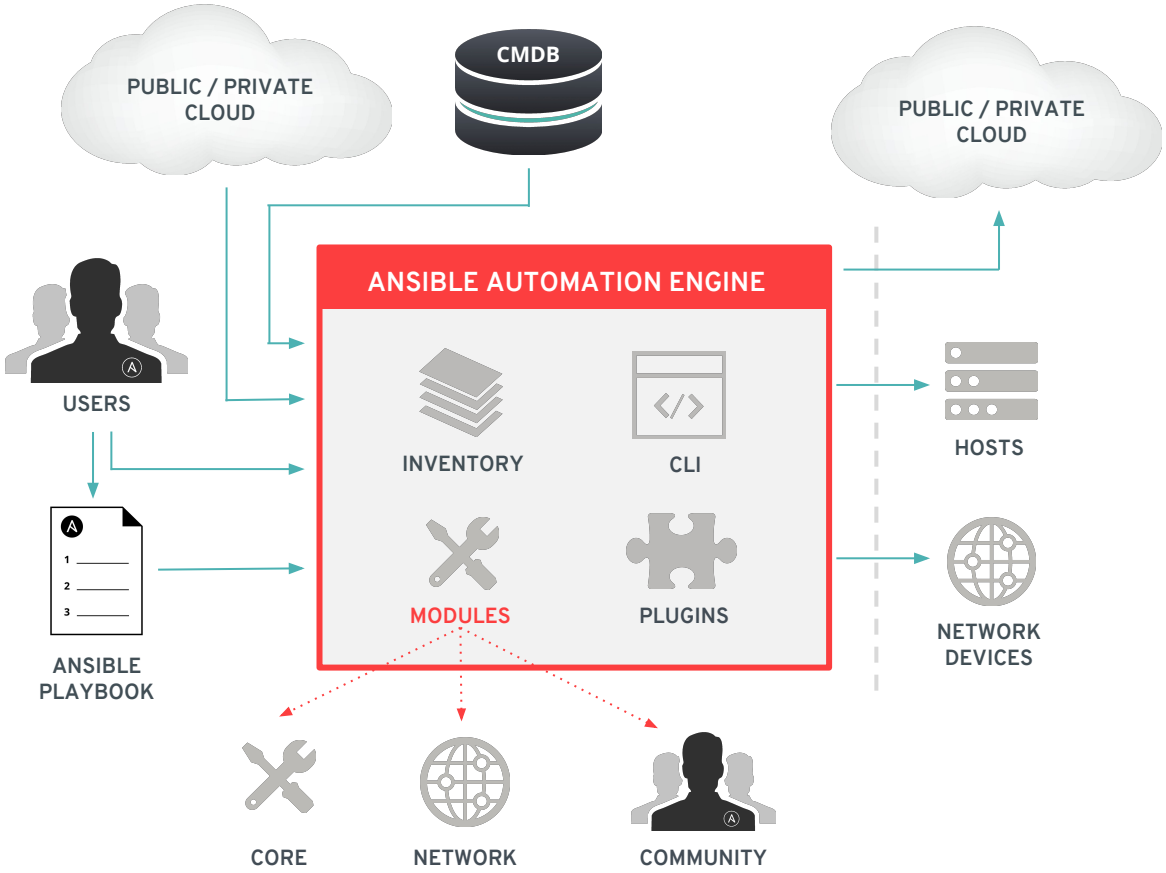
Ensure an on-going steady-state

- Daily, weekly, monthly scheduled tasks
- State checking and validation



HOW ANSIBLE DOES WORK ?

ANSIBLE

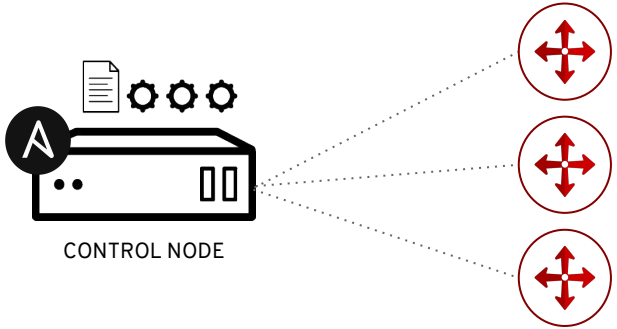


redhat.

HOW DOES NETWORK AUTOMATION WORK?

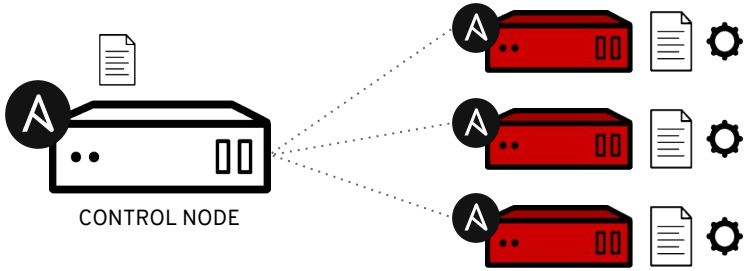
ANSIBLE

Module code is executed locally on the control node



NETWORKING DEVICES

Module code is copied to the managed node, executed, then removed



LINUX HOSTS

redhat.

ANSIBLE NETWORK AUTOMATION

50
Network Platforms

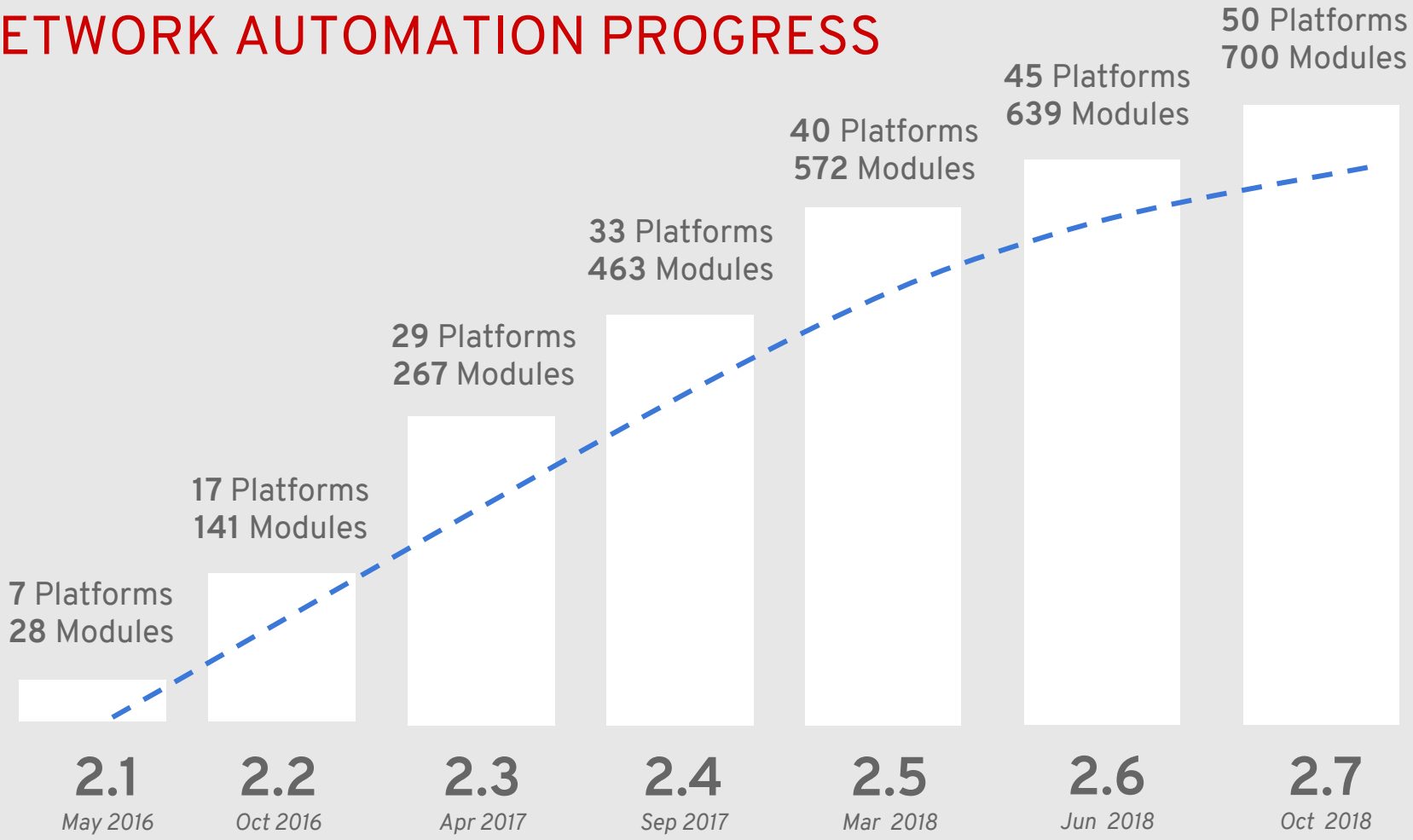
700+
Network Modules

12*
Galaxy Network Roles

ansible.com/for/networks
galaxy.ansible.com/ansible-network

Ansible Network modules comprise 1/3 of all modules that ship with Ansible Engine

NETWORK AUTOMATION PROGRESS



NETWORK MODULES: BUILT-IN DEVICE ENABLEMENT

A10

Apstra AOS

Arista EOS, CVP

Aruba Networks

AVI Networks

Big Switch Networks

Brocade Ironware

Cisco ACI, AireOS, ASA, Firepower, IOS, IOS-XR, Meraki, NSO, NX-OS

Citrix Netscaler

Cumulus Linux

Dell OS6, OS9, OS10

Exoscale

Extreme EX-OS, NOS, SLX-OS, VOSS

F5 BIG-IP, BIG-IQ

Fortinet FortIOS, FMGR

Huawei CloudEngine

Illumos

Infoblox NIOS

Juniper JunOS

Lenovo CNOS, ENOS

Mellanox ONYX

MikroTik RouterOS

Openswitch (OPX)

Ordnance

NETCONF

Netvisor

OpenSwitch

Open vSwitch (OVS)

Palo Alto PAN-OS

Nokia NetAct, SR OS

Ubiquiti EdgeOS

VyOS

RED HAT ANSIBLE NETWORK AUTOMATION

NETWORK MODULES

- **Developed, maintained, tested, and supported** by Red Hat, includes Ansible Engine, Ansible Tower
- **140+ supported modules** and growing*
- Red Hat **reports and fixes problems**
- **Networking Modules and Roles included**

*take special note of the specific supported platforms

INCLUDED SUPPORT:

Arista EOS

Cisco IOS

Cisco IOS XR

Cisco NX-OS

Infoblox NIOS

Juniper Junos

Open vSwitch

VyOS

PLAYBOOK EXAMPLE

```
- name: run multiple commands and evaluate the output
  hosts: cisco
  gather_facts: no
  connection: network_cli
  tasks:
  - name: show version and show interfaces
    ios_command:
      commands:
         - show version
         - show interfaces
      wait_for:
         - result[0] contains IOS
         - result[1] contains Loopback0
```

NETWORK MODULES

`*_command:`
Run command get/use output

`*_config:`
Make a change to the config with context

`*_facts:`
Get information (e.g. OS version, interfaces, etc.)

Ios

- ios_command - Run commands on remote devices running Cisco IOS
- ios_config - Manage Cisco IOS configuration sections
- ios_facts - Collect facts from remote devices running IOS
- ios_template (D) - Manage Cisco IOS device configurations over SSH

Iosxr

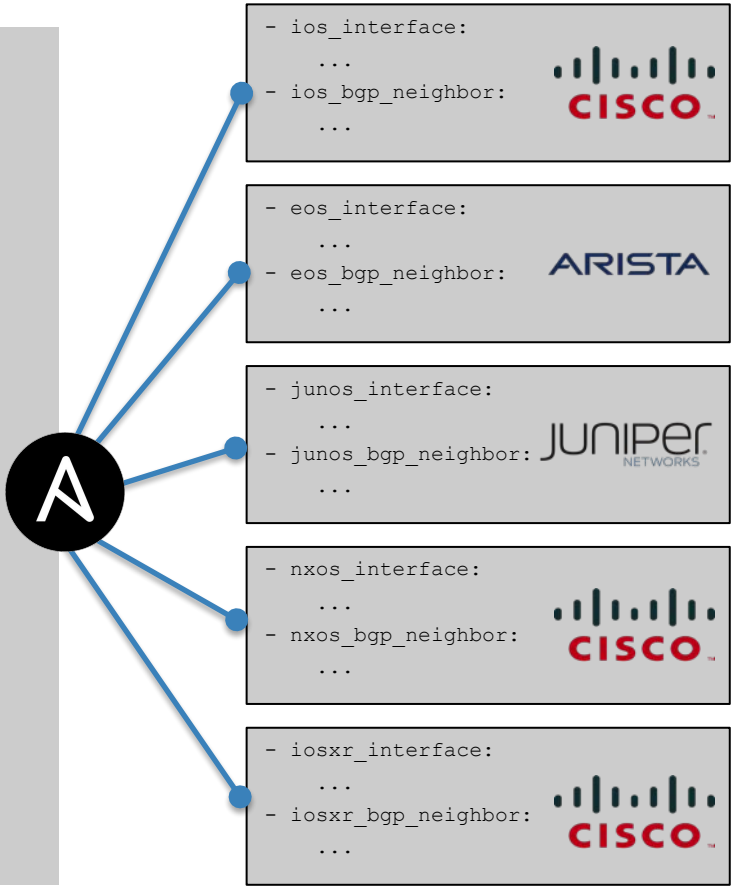
- iosxr_command - Run commands on remote devices running Cisco iosxr
- iosxr_config - Manage Cisco IOS XR configuration sections
- iosxr_facts - Collect facts from remote devices running IOS-XR
- iosxr_template (D) - Manage Cisco IOSXR device configurations over SSH

Junos

- junos_command - Execute arbitrary commands on a remote device running Junos
- junos_config - Manage configuration on devices running Juniper JUNOS
- junos_facts - Collect facts from remote device running Junos
- junos_netconf - Configures the Junos Netconf system service
- junos_package - Installs packages on remote devices running Junos
- junos_template (D) - Manage configuration on remote devices running Junos

PLATFORM AGNOSTIC MODULES

```
- name: configure network interface
  net_interface:
    name: "{{ if_name }}"
    description: "{{ if_description }}"
    enabled: yes
    mtu: 9000
    state: up

- name: configure bgp neighbors
  net_bgp_neighbor:
    peers: "{{ item.peer }}"
    remote_as: "{{ item.remote_as }}"
    update_source: Loopback0
    send_community: both
    enabled: yes
    state: present
```



redhat.

AGGREGATE RESOURCES

```
- name: configure vlans neighbor
  net_vlan:
    vlan_id: "{{ item.vlan_id }}"
    name: "{{ item.name }}"
    state: "{{ item.state | default('active') }}"
  with_items:
    - { vlan_id: 1, name: default }
    - { vlan_id: 2, name: Vl2 }
    - { vlan_id: 3, state: suspend }
```

```
- name: configure vlans neighbor
  net_vlan:
    aggregate:
      - { vlan_id: 1, name: default }
      - { vlan_id: 2, name: Vl2 }
      - { vlan_id: 3, state: suspend }
    state: active
    purge: yes
```

DECOUPLE IT!

ANSIBLE

Definition

```
project_tag: foo
tenant_nets:
  - 192.133.157.0/24

fw_outside_ip: 192.133.159.73
fw_inside_ip: 192.133.159.137

vlan_data:
  - { vlan_id: 1, name: default }
  - { vlan_id: 2, name: Vl2 }
  - { vlan_id: 3, state: suspend }

interface_data:
  - { name: Ethernet0/1, mtu: 256, description: test-interface-1 }
  - { name: Ethernet0/2, mtu: 516, description: test-interface-2 }

port_data:
  - { desc: "mcp1.titan1", switch: "aa17-n9k-1", interface: "Ethern
  - { desc: "mcp1.titan1", switch: "aa17-n9k-2", interface: "Ethern
```

Implementation

```
- name: Creating aggregate of vlans
  nxos_vlan:
    aggregate: "{{ vlan_data }}"
    state: active
    purge: yes


- name: Add interface using aggregate
  nxos_interface:
    aggregate: "{{ interface_data }}"
    duplex: full
    speed: 100
    state: present
```

Define Once

Apply Many

redhat.

NETWORK ENGINE & NETWORK FUNCTIONS

NETWORK ENGINE

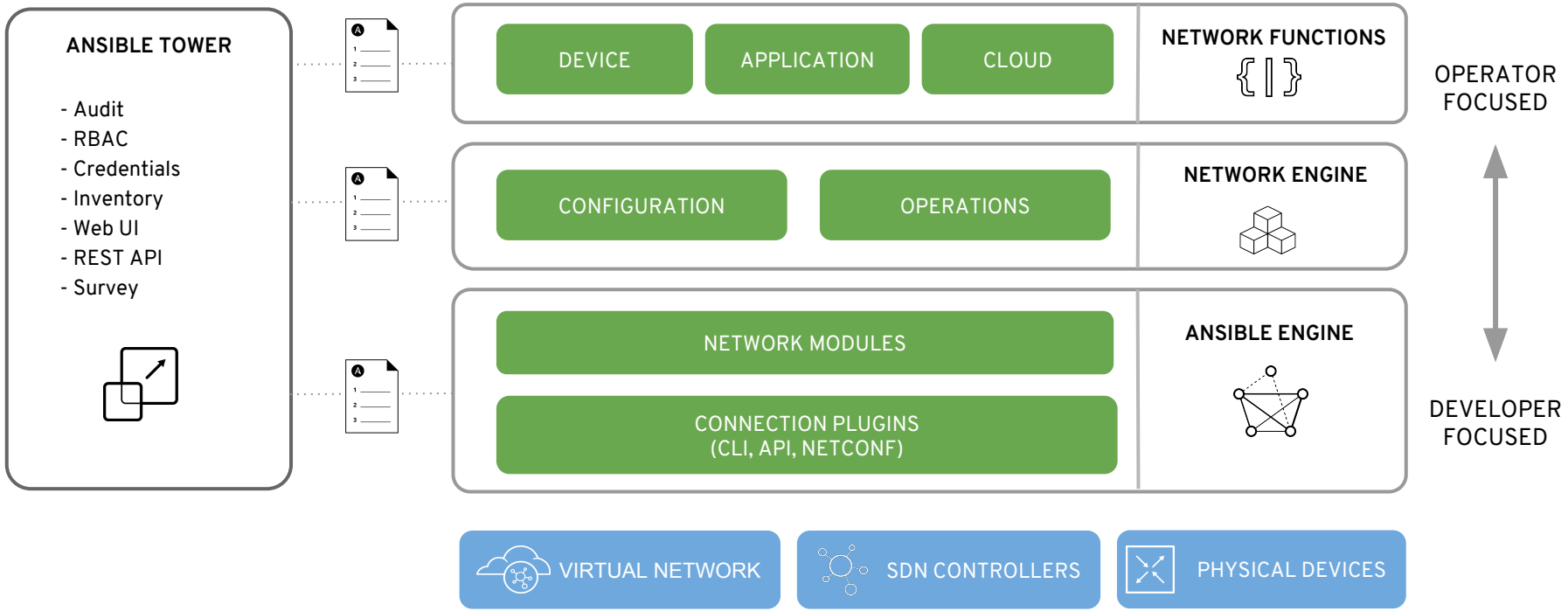
- Ansible Role decoupled from mainline development branch
- Incubate new capabilities and accelerate automation adoption
- Biweekly release cycle

NETWORK FUNCTIONS

- Data driven workflows for performing network operator tasks
- Extensible and adaptable for any platform, any device
- DIY => MIY

Deliver FRICTIONLESS network automation for SEAMLESS orchestration of workloads

ANSIBLE NETWORK STACK ARCHITECTURE



AUTOMATION USE CASE EXAMPLES

- **Information / Inventory Retrieval and Configuration**
  - Ad hoc or bulk, generate reports
  - Iteration over specific network segments, VLANs, VRFs
- **State Checking and Validation**
  - Compare running configs to desired configs
- **Run & Delegate discrete commands**
  - Manually, API via Tower, Scheduled via Tower

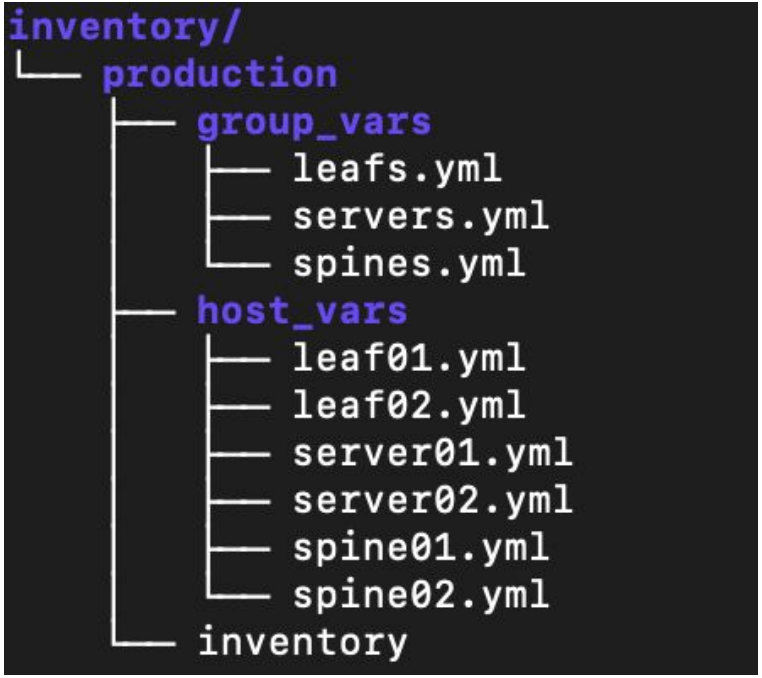
AUTOMATION USE CASE EXAMPLES

- **Continuous Compliance**
  - Combining stateful validation with schedules
  - Logging and Aggregation
  - Configuration backup

- **Integrations**
  - End to end automation
  - Zero touch provisioning using a callback

VERSION CONTROL YOUR TOPOLOGY & DATA MODEL

```
[leafs]
leaf01 ansible_host=10.16.10.11
leaf02 ansible_host=10.16.10.12

[leafs:vars]
ansible_network_os=vyos
ansible_user=my_vyos_user

[spines]
spine01 ansible_host=10.16.10.13
spine02 ansible_host=10.16.10.14

[spines:vars]
ansible_network_os=vyos
ansible_user=my_vyos_user

[network:children]
leafs
spines

[servers]
server01 ansible_host=10.16.10.15
server02 ansible_host=10.16.10.16

[datacenter:children]
leafs
spines
servers
```



NETWORK CI WORKFLOW