MANAGING NETWORKS HASN'T CHANGED IN 30 YEARS.
WHAT IS THE PRIMARY METHOD OF MAKING NETWORK CHANGES IN YOUR ENVIRONMENT?

<table>
<thead>
<tr>
<th>Method</th>
<th>Percentage of Respondents</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLI on individual devices</td>
<td>71</td>
</tr>
<tr>
<td>GUI on individual devices</td>
<td>8</td>
</tr>
<tr>
<td>Vendor’s network management system</td>
<td>10</td>
</tr>
<tr>
<td>Network automation tool</td>
<td>6</td>
</tr>
<tr>
<td>API</td>
<td>3</td>
</tr>
<tr>
<td>Others</td>
<td>2</td>
</tr>
</tbody>
</table>

**Figure 1**
Primary Method for Making Network Changes

Systems Thinking

- Manufacturing (and IT) move only as fast as the slowest component
- Optimization anywhere but the bottleneck is an illusion
Why Does This Matter?

\[ Cycle\ Time = \frac{WIP}{Throughput} \]
Why Does This Matter?

Backlog \rightarrow \text{Work in Progress}

Bob

\text{Cycle Time} = \frac{\text{WIP}}{\text{Throughput}}
Bob is your Bottleneck.

Bob is slowing you down!
Wait...
Bob is your HERO!
Bob is your HERO!

1) Bob designs the network.
Bob is your HERO!

1) Bob designs the network.
2) Bob builds the network.
Bob is your HERO!

1) Bob designs the network.
2) Bob builds the network.
3) Bob fixes the network.
Bob is your HERO!

1) Bob designs the network.
2) Bob builds the network.
3) Bob fixes the network.
4) Bob deploys WIFI at the VP’s lake house.
Bob is your HERO!

1) Bob designs the network
2) Bob builds the network
3) Bob fixes the network
4) Bob deploys WIFI at the VP’s lake house.

Bob does EVERYTHING!
So... what’s the problem?
Heroes are good, right?
Heroes are in high demand

The most important person in the World

Hero

I need this now!
Heroes are in high demand

The most important person in the World
Heroes are in high demand

The most important person in the World
Buses (and Recruiters) Like Heroes

Hero

(red hat logo)
What do you do?
Option #1: Reduce WIP

Increase throughput by decreasing demand on Bob
Option #2: Increase Throughput
Automation: Hero as Code

- Leverages Human Experience
- Reduce Repetition

- Reduce Variability
- Reduce Isolation
Automation: Hero as Code

- Leverages Human Experience
- Reduce Repetition

- Reduce Variability
- Reduce Isolation
Convert Procedures to Playbooks

1. Create VLAN
2. Add port to VLAN
3. Address Interface

Method of Procedure
- Define Intent, Policy, Architecture
- Apply across device type, vendor
Manage Lifecycle with Process & Playbooks

- Revision control, configuration management
- Ensure an ongoing steady-state
- Automated testing, reduce human error
Communicate with Playbooks

- DEVELOPMENT
- SECURITY
- OPERATIONS
- BUSINESS (ARCHITECTS)
What is Ansible?

**Ansible** is a simple automation language that can perfectly describe an IT application infrastructure in Ansible Playbooks.

**Ansible Engine** is an automation engine that runs Ansible Playbooks.

**Ansible Tower** is an enterprise framework for controlling, securing and managing your Ansible automation with a UI and RESTful API.
Why Ansible?

**SIMPLE**
- Human readable automation
- No special coding skills needed
- Tasks executed in order
- Get productive quickly

**POWERFUL**
- Image updates
- Configuration management
- Compliance
- Orchestrate the network lifecycle

**AGENTLESS**
- Agentless architecture
- Uses OpenSSH & WinRM
- No agents to exploit or update
- More efficient & more secure
- Build & manage dynamic inventory
- Roles-Based Access Control
- Workflows
- Ongoing Compliance
- Running Playbooks at Scale
- RESTful API
API-Driven Infrastructure

Well Defined, Role Based API

Easily Customizable Back End
The Flexibility of Choice

Business Requirements

Abstraction Through Automation

BGP  LB  OSPF  VLAN  ACL  QOS  EVPN  AAA

Cisco  f5  Infoblox  Arista  Palo Alto  Juniper Networks
The Road To Automation

**STANDARDIZE**
- Snapshot State
- Detect Unauthorized Change
- Standardize Existing Configs
- Standardize New Deployments

**AUTOMATE**
- Automate common tasks
- Make changes across any set of network devices
- Validate that changes were successful

**SCALE**
- Automated deployment from Services Catalogue
- Automated compliance checking & enforcement
- API-Driven Integration with Application Development

Organize the Chaos → Optimize your Infrastructure → Stop Logging Into Devices
Improved Outcomes with Automation

**Time to Value**
Configuration & Change Automation
- Faster Customer Service On-boarding
- Faster Execution of Change Requests

**Time to Remediation**
Automated Fault Remediation
- Faster Execution of Maintenance
- Faster Troubleshooting and Remediation
Playbooks & Network Modules
Under the Hood

ANSIBLE’S AUTOMATION ENGINE

PUBLIC / PRIVATE CLOUD

CMDB

USERS

ANSIBLE PLAYBOOK

INVENTORY

MODULES

API

PLUGINS

HOSTS

NETWORKING
Connection Plugins

Python code is executed locally on the control node.

Python code is copied to the managed node, executed, then removed.

NETWORKING DEVICES

LINUX HOSTS
### Anatomy of a Playbook

<table>
<thead>
<tr>
<th>Inventory: The devices to configure</th>
</tr>
</thead>
<tbody>
<tr>
<td>hosts: network</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Variables: The key/value pairs that change from device to device</th>
</tr>
</thead>
<tbody>
<tr>
<td>vars:</td>
</tr>
<tr>
<td>site_domain_name: 'example.net'</td>
</tr>
<tr>
<td>network_name_servers:</td>
</tr>
<tr>
<td>- 8.8.8.8</td>
</tr>
<tr>
<td>- 8.8.4.4</td>
</tr>
<tr>
<td>log_host: 10.2.2.3</td>
</tr>
<tr>
<td>tasks:</td>
</tr>
<tr>
<td>- name: Configure the hostname and domain name</td>
</tr>
<tr>
<td>net_system:</td>
</tr>
<tr>
<td>hostname: &quot;{{ inventory_hostname }}&quot;</td>
</tr>
<tr>
<td>domain_name: &quot;{{ site_domain_name }}&quot;</td>
</tr>
<tr>
<td>name_servers: &quot;{{ network_name_servers }}&quot;</td>
</tr>
<tr>
<td>- name: configure host logging</td>
</tr>
<tr>
<td>net_logging:</td>
</tr>
<tr>
<td>dest: host</td>
</tr>
<tr>
<td>name: &quot;{{ log_host }}&quot;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tasks: The tasks to perform on those devices</th>
</tr>
</thead>
</table>
Network Functional Modules

Building Blocks

**command**
(e.g. `ios_command`)
- Executes command on device
- Provides output for further processing

**config**
(e.g. `ios_config`)
- Manipulates the config of the device
- Idempotent

**facts**
(e.g. `ios_facts`)
- Collects facts from the device
Network Functional Module: Command

- hosts: network
gather_facts: no
connection: local
tasks:
  - name: show version
    ios_command:
      commands:
        - show version
    wait_for:
      - result[0] contains Version
    register: results

- set_fact:
  ver: "{{ results.stdout[0]|regex_search('Version ([0-9.]+)','\1') }}"

- debug: var=ver
Network Functional Module: Command

PLAY [network]
**************************************************************************
TASK [show version and show interfaces]
**************************************************************************
ok: [rtr1]

TASK [set_fact]
**************************************************************************
ok: [rtr1]

TASK [debug] **************************************************************************
ok: [rtr1] => {
  "ver": [
    "16.06.01"
  ]
}

PLAY RECAP **************************************************************************
 rtr1          : ok=3  changed=0  unreachable=0  failed=0
Network Functional Module: Config

- hosts: network
  gather_facts: no
  connection: local
  tasks:
    - name: configure hostname
      ios_config:
        lines:
        - "hostname {{ inventory_hostname }}"
First Run:
PLAY [network]
**********************************************************************************************
TASK [configure hostname]
**********************************************************************************************
changed:  [rtr1]

PLAY RECAP
**********************************************************************************************

rtr1:  ok=1  changed=1  unreachable=0  failed=0

Second Run:
PLAY [network]
**********************************************************************************************
TASK [configure hostname]
**********************************************************************************************

ok:  [rtr1]

PLAY RECAP
**********************************************************************************************

rtr1:  ok=1  changed=0  unreachable=0  failed=0

Network Functional Module: Config
Network Functional Module: Facts

- hosts: network
  connection: local
  gather_facts: False
  tasks:

    - name: Get facts
      ios_facts:
        gather_subset: all

    - debug: msg="Serial Number is {{ ansible_net_serialnum }}"
Network Functional Module: Facts

PLAY [network]

TASK [Get facts]

ok: [rtr1]

TASK [debug]

ok: [rtr1] => {
  "msg": "Serial Number is 9G2O4MxLVP"
}

PLAY RECAP

rtr1 : ok=2  changed=0  unreachable=0  failed=0
Network Resource Modules

- name: configure eos system properties
  eos_system:
    domain_name: ansible.com
    vrf: management
  when: ansible_network_os == 'eos'

- name: configure nxos system properties
  nxos_system:
    domain_name: ansible.com
    vrf: management
  when: ansible_network_os == 'nxos'

- name: configure ios system properties
  ios_system:
    domain_name: ansible.com
    lookup_enabled: yes
  when: ansible_network_os == 'ios'

- Per Platform Implementation
- Focused on managing a resource
- Declarative by design
- Handles complexity
- name: configure network interface
  net_interface
    name: "{{ interface_name }}"
    description: "{{ interface_description }}"
    enabled: yes
    mtu: 9000
    state: up

- name: configure VLAN ID and name
  net_vlan:
    vlan_id: 20
    name: test-vlan
Declarative Intent

- **name**: configure interface
  - **net_interface**:
    - **name**: GigabitEthernet0/2
    - **description**: public interface configuration
    - **enabled**: yes
    - **state**: connected
    - **neighbors**:
      - **host**: core-01
      - **port**: Ethernet5/2/6
Aggregate Resources

Loop entries

- name: Configure VLANs

  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 }

Bulk entries

- name: Configure VLANs and Purge

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

Multiple Operations

Single Operation
Applications Roles

- Focused on addressing operational use cases
- Approved and opinionated methods
- Developed, tested, and distributed by Ansible
- Agile development with gated release process
Software Supply Chain

Network Operators aren’t programmers, need one-stop for “approved” content

<table>
<thead>
<tr>
<th>Community</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>Where to obtain playbooks, roles, modules?</td>
<td>Trusted Distribution:</td>
</tr>
<tr>
<td>Who wrote them?</td>
<td>● Development: GitHub/ansible-network</td>
</tr>
<tr>
<td>Are they tested?</td>
<td>● Released: Ansible Galaxy</td>
</tr>
<tr>
<td>Who supports them?</td>
<td>Distributed CI test system</td>
</tr>
<tr>
<td></td>
<td>Supported by Red Hat</td>
</tr>
</tbody>
</table>
Core Concepts & Best Practices
## Layered Implementation

Simplifies playbooks, limits blast radius, and facilitates RBAC

<table>
<thead>
<tr>
<th>Cluster 1</th>
<th>App A</th>
<th>Tenant 1</th>
<th>App B</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overlays</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OSPF</td>
<td>EIGRP</td>
<td>BGP</td>
<td></td>
</tr>
<tr>
<td>Interconnects, MLAG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STP</td>
<td>VLANs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AAA</td>
<td>NTP</td>
<td>Logging</td>
<td>Banners</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DNS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ACLs</td>
</tr>
</tbody>
</table>

### Access

### Core

### System
Manage Applications, not Devices
Key/Value Pairs

Abstraction Through Data Models

**Cisco IOS**

```bash
router bgp 65082
no synchronization
bgp log-neighbor-changes
neighbor 10.11.12.2 remote-as 65086
no auto-summary
```

**Juniper JunOS**

```bash
bgp {
  local-as 65082;
  group TST {
    peer-as 65086;
    neighbor 10.11.12.2;
  }
}
```
Key/Value Pairs

Abstraction Through Data Models

Cisco IOS

```bash
router bgp 65082
no synchronization
bgp log-neighbor-changes
neighbor 10.11.12.2 remote-as 65086
no auto-summary
```

Juniper JunOS

```bash
bgp {
  local-as 65082;
  group TST {
    peer-as 65086;
    neighbor 10.11.12.2;
  }
}
```
Key/Value Pairs

Abstraction Through Data Models

```
bgp:
  global:
    config:
      as: 65082
  neighbors:
    neighbor:
      - neighbor_address: 10.11.12.2
        config:
          peer_group: TST
          peer_as: 65086

router bgp 65082
  no synchronization
  bgp log-neighbor-changes
  neighbor 10.11.12.2
  remote-as 65086
  no auto-summary
```

YANG OC Data Model

Vendor-Specific Rendering
The Flexibility of Ansible + Data Models

Any Model, Any Encoding, Any Transport

- Model
  - Vendor
  - OpenConfig
  - Custom

- Encoding
  - XML
  - JSON
  - CLI

- Transport
  - Netconf
  - API
  - SSH
Implementation

- system:
  - hostname: "{{ inventory_hostname }}"
  - domain_name: eng.ansible.com
- source_interface:
  - name: Management1
  - vrf: default
- domain_lookup: no
- name_servers:
  - 1.1.1.1
  - 2.2.2.2
- vlan_data:
  - { id: 600, name: management }
  - { id: 601, name: users }

Source of Truth (a.k.a. Key/Value Pairs)

Operations

- Implementation

- Engineering

- Production

Desired State

Deploys

Infrastructure

- Applications
- Servers
- Storage
- Network

Feeds

- Engineering
- Production

- Operations

+
## Facts: Loading and Using

### Load SoT from Inventory:

```
host_vars\switch1\interfaces.yml
```

### Or

### CMDB

### Or

### Manually load w/Playbook:

- include_role:
  - name: load_interface_data

### Available for Playbooks to reference:

```yaml
- name: Set Interface Attributes
  net_interface:
    name: "{{ item }}"
    description: "{{ item.description }}"
    enabled: {{ item.enabled }}
  with_items: "{{ interfaces.keys() }}"
```

### Per-Inventory Item

### Facts Cache
The Automated Enterprise

SMEs
Operators
Developers

Develop
Plan
Deploy
Test
Operate

SoT

ANSIBLE TOWER by Red Hat