AUTOMATING DATACENTER NETWORKS

Will Tome
Sr. Solutions Architect, Ansible
@willtome
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

**Source:** Gartner, *Look Beyond Network Vendors for Network Innovation*. January 2018. Gartner ID: G00349636. (n=64)
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
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
Example: Backup Config & Run Commands

---

- name: Backup and Troubleshoot
  hosts: all
  connection: local

  tasks:
  - name: Backup and Save Running Config
    nxos_config:
      backup: yes
      save_when: always

  - name: Run Show Commands
    nxos_command:
      commands:
        - show ip interface brief
        - show ip interface description
        - show ip route
      register: show_commands_out
Example: Configure a Port

```yaml
---
- hosts: leaf
  connection: local
  vars:
    vlan_id: 20
    interface: Ethernet2/1

  tasks:
  - name: "Configure vlan {{ vlan_id }}"
    nxos_vlan:
      vlan_id: "{{ vlan_id }}"
  - name: "Configure interface {{ interface }}"
    nxos_config:
      before: "default interface {{ interface }}"
      lines:
      - description Access Port
      - switchport mode access
      - "switchport access vlan {{ vlan_id }}"
      parents: "interface {{ interface }}"
      save_when: modified
```
Access Port Survey

**LAUNCH JOB** | **DC_DEMO / Access Port**

- **INTERFACE NAME**
  Which interface would you like to configure? (eg. Eth3/1)

- **DESCRIPTION**
  Provide the interface description

- **VLAN ID**
  Which VLAN is the port configured for?

**INVENTORY**
- DC Demo

**CREDENTIAL**
- Machine: Local
- Network: VRL

**CANCEL** | **LAUNCH**
**Access Port Output**

**Details**
- **Status**: Successful
- **Started**: 4/10/2018 3:41:40 PM
- **Finished**: 4/10/2018 3:42:21 PM
- **Template**: DC_DEMO / Access Port
- **Job Type**: Run
- **Launched By**: neteng
- **Inventory**: DC Demo
- **Project**: DC DEMO
- **Revision**: 509dee5
- **Playbook**: access_port.yml
- **Machine Credential**: Local
- **Extra Credentials**: VIRL
- **Forks**: 0
- **Limit**: leaf2
- **Verbosity**: 0 (Normal)
- **Instance Group**: tower

**DC_DEMO / Access Port**

```
17  future version
18
19  ok: [leaf2]
20
21  TASK [netops : assert]  **********************
22
23  ok: [leaf2] => {
24  "changed": false,
25  "msg": "All assertions passed"
26  }
27
28  TASK [netops : access VLAN 10 on Eth2/3]  15:42:08
29  changed: [leaf2]
30
31  PLAY RECAP  15:42:16
32  leaf2    : ok=3  changed=1  unreachable=0  failed=0
```

Event ID: 1614046  Status: Host OK  Click for details
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

Data Model

Vendor-Specific Rendering
How do we go the other way?

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

bgp {
  local-as 65082;
  group TST {
    peer-as 65086;
    neighbor 10.11.12.2;
  }
}
```
edge1# sho cdp neighbors detail
----------------------------------------
Device ID: spine1(TB00000000B)
System Name: spine1
Interface address(es):
   IPv4 Address: 172.16.1.112
Platform: N7K-C7018, Capabilities: Router Switch
Interface: Ethernet2/1, Port ID (outgoing port):
   Ethernet2/1
Holdtime: 148 sec
Version:
Cisco Nexus Operating System (NX-OS) Software,
Version 7.3(0)D1(1)
Duplex: full
MTU: 1500
Mgmt address(es):
   IPv4 Address: 172.16.1.112
----------------------------------------
Device ID: spine2(TB00030000B)
System Name: spine2 ...

"neighbors_list": [
  {
    "local_port": "mgmt0",
    "remote_device": "spine1(TB00000000B)",
    "mgmt_ip": "172.16.1.112",
    "remote_port": "mgmt0"
  },
  {
    "local_port": "Ethernet2/1",
    "remote_device": "spine1(TB00000000B)",
    "mgmt_ip": "172.16.1.112",
    "remote_port": "Ethernet2/2"
  },
  {
    "local_port": "Ethernet2/2",
    "remote_device": "spine2(TB00030000B)",
    "mgmt_ip": "172.16.1.113",
    "remote_port": "Ethernet2/2"
  }
]
Ansible Network Roles (preview)

Connection Plug-ins
(NETCONF/SSH, CLI/SSH, API/SSH)

Functional, Declarative Modules
(40+ Platform Specific)

Playbooks / Roles

INCLUDED IN DISTRO

NOT INCLUDED IN DISTRO

WRITTEN BY USERS

WRITTEN BY ANSIBLE*

*Can also be created and maintained by the community
edge1# sho cdp neighbors detail
----------------------------------------
Device ID: spine1(TB00000000B)
System Name: spine1
Interface address(es):
  IPv4 Address: 172.16.1.112
Platform: N7K-C7018, Capabilities: Router Switch
Interface: Ethernet2/1, Port ID (outgoing port): Ethernet2/1
Holdtime: 148 sec
Version:
Cisco Nexus Operating System (NX-OS) Software,
  Version 7.3(0)D1(1)
Duplex: full
MTU: 1500
Mgmt address(es):
  IPv4 Address: 172.16.1.112
----------------------------------------
Device ID: spine2(TB00030000B)
System Name: spine2
...
Data Structure

---

pod_interconnects:

Ethernet2/2:
  lines: |
    description default trunk
    switchport
    switchport mode trunk
    switchport trunk allowed vlan 10,20
    no shutdown
    description: default trunk
    neighbor: leaf1

Ethernet2/1:
  lines: |
    description default trunk
    switchport
    switchport mode trunk
    switchport trunk allowed vlan 10,20
    no shutdown
    description: default trunk
    neighbor: edge1

...
## Manage Inventory

### Network

#### Search

<table>
<thead>
<tr>
<th>Network</th>
<th>Permissions</th>
<th>Groups</th>
<th>Hosts</th>
<th>Sources</th>
<th>Completed Jobs</th>
</tr>
</thead>
</table>

#### Hosts

<table>
<thead>
<tr>
<th>ON</th>
<th>Host</th>
<th>Related Groups</th>
</tr>
</thead>
<tbody>
<tr>
<td>ON</td>
<td>edge1</td>
<td>edge</td>
</tr>
<tr>
<td>ON</td>
<td>leaf1</td>
<td>leaf</td>
</tr>
<tr>
<td>ON</td>
<td>leaf2</td>
<td>leaf</td>
</tr>
<tr>
<td>ON</td>
<td>spine1</td>
<td>spine</td>
</tr>
<tr>
<td>ON</td>
<td>spine2</td>
<td>spine</td>
</tr>
</tbody>
</table>

#### Spine1

<table>
<thead>
<tr>
<th>Details</th>
<th>Facts</th>
<th>Groups</th>
</tr>
</thead>
</table>

### Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pod_interconnects</td>
<td></td>
</tr>
<tr>
<td>othernets</td>
<td></td>
</tr>
<tr>
<td>lines</td>
<td>default trunk</td>
</tr>
<tr>
<td>switchport</td>
<td></td>
</tr>
<tr>
<td>switchport_mode_trunk</td>
<td>allowed vlan 10,20</td>
</tr>
</tbody>
</table>
Example: Deploy VLAN

---

- hosts: leaf:spine:edge
  connection: local
vars:
  vlan_id: 20
  vlan_description: DATA_NET

tasks:
- name: "Configure vlan {{ vlan_id }}"
  nxos_vlan:
    vlan_id: "{{ vlan_id }}"
    name: "{{ vlan_description }}"

- name: "Add vlan {{ vlan_id }} to interconnects"
  nxos_config:
    lines:
      - "switchport trunk allowed vlan add {{ vlan_id }}"
    parents: "interface {{ item }}"
    with_items: "{{ pod_interconnects }}"
Deploy VLAN Survey

LAUNCH JOB | DC DEMO / Deploy VLAN

OTHER PROMPTS  SURVEY

* VLAN ID

30

* DESCRIPTION

DEMO_VLAN

INVENTORY
DC Demo

CREDENTIAL
Machine: Local
Network: VIRL

CANCEL  LAUNCH
Deploy VLAN Output

DC DEMO / Deploy VLAN

PLAY RECAP

- edge1: ok=6 changed=4 unreachable=0 failed=0
- leaf1: ok=4 changed=3 unreachable=0 failed=0
- leaf2: ok=4 changed=3 unreachable=0 failed=0
- spine1: ok=6 changed=4 unreachable=0 failed=0
- spine2: ok=6 changed=4 unreachable=0 failed=0
RESOURCES

Ansible Networking Homepage: ansible.com/networking

Would you like to learn Ansible? It’s easy to get started: ansible.com/get-started

Want to learn more? Videos, webinars, case studies, whitepapers: ansible.com/resources

Network Automation Examples: https://github.com/network-automation/

Playbooks from this Demo: https://github.com/willtome/dcdemo/

Github: willtome | Twitter: @willtome