Ansible

Introduction for HBONE Workshop'2019

Szabó Gábor
(gabszabo@cisco.com)
Systems Engineer
02/14/2019

Introduction to Ansible

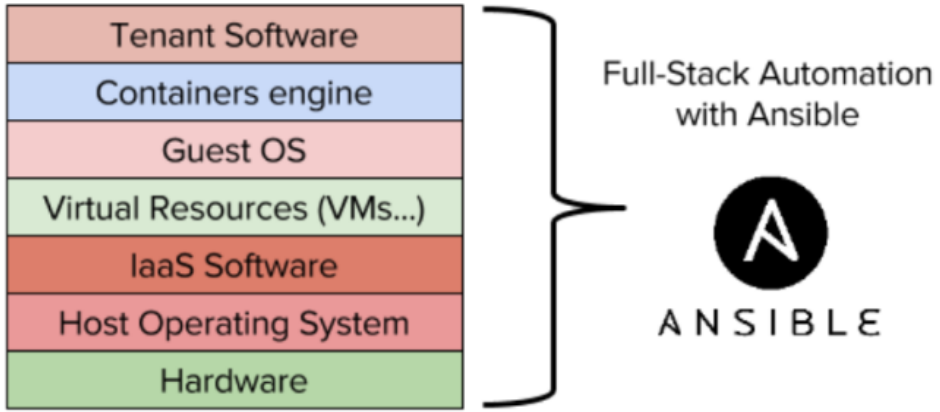
Ansible 101

Brief history

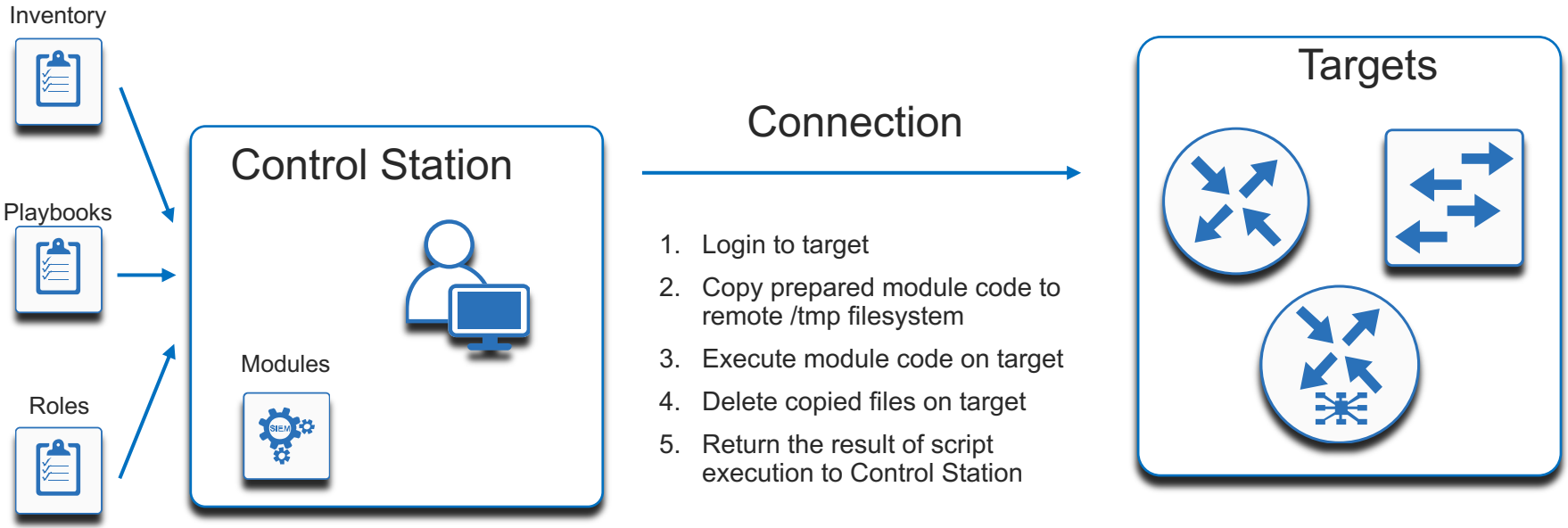
- A configuration management tool, deployment tool, and ad-hoc task execution tool all in one.
- Initially developed by Michael DeHaan
  - Author of Cobbler and co-author of Func
- Ansible is open-source project:
  - https://github.com/ansible
- Acquired by Red Hat in 2015 for $150m
- Red Hat offers support for Ansible with Red Hat Ansible Engine product

Ansible Use Cases

- Provisioning
- Configuration Management
- Application Deployment
- Continuous Delivery
- Security & Compliance
- Automation



Ansible – Helicopter View



Ansible Highlights

- By default works in a push mode
  - Ansible Control Station connects to the managed nodes and does what it's supposed to do
  - Pull mode is also available
- By default uses SSH for transport
  - It can be changed using connection plugins
- Agentless
  - It is expected that managed node only has SSH daemon and Python interpreter
- Runs tasks in a sequential manner
- Expects idempotent modules:
  - An operation is idempotent if the result of performing it once is exactly the same as the result of performing it repeatedly

Ansible Default Connection Mode

- Connects to the target node:
  - Typically it is the SSH connection to the target node
  - Instead of SSH, another protocol can be used (e.g. PowerShell for Windows)
- Copies and then executes module code (typically written in Python) on the target node
- Removes copied files
- Return result of the execution in JSON format
- Optimal for managing servers/applications

Ansible Connection Modes for Network Devices

- Network devices usually don't have Linux-like filesystem and Python interpreter - so default mode is not applicable here
- Connection Modes for Network Devices:
  - Executes module code on the local server
  - The code has to establish a connection to the target node itself.
  - Return result of the execution in JSON format
- Most common connection modes:
  - network_cli
  - netconf
  - local (for nxapi)

New Way of Interacting With Systems: REST API

- REST API: use HTTP requests to Create/Read/Update/Delete (CRUD) operations on resources of remote systems
- General approach, applicable to different kind of objects (applications, network devices, cloud resources etc.)
- No need to login to remote system using SSH
- Can Ansible handle it? YES, similarly how it handles network devices
  - Local connection mode: the relevant module is executed on Control Station
  - The module must deal with the HTTP request

Ansible Components

YAML

YAML Overview

- YAML = “YAML Ain’t Markup Language”
- Human readable data serialization language
- Common basis for a number of domain specific languages:
  - Ansible
  - OpenStack Heat
  - Saltstack
  - cloud-init
  - etc.

YAML File Format

- All YAML files:
  - begin with “---” (start of a document)
  - may end with “....” (end of a document)
- # character marks the beginning of a comment in the line (if not enclosed single/double quotes):
- Structure is determined by indentation:
  - indentation = zero or more space characters at the start of a line.
  - tab characters must not be used in indentation

YAML Basic Primitives

- Scalars (strings/numbers)
- Mappings (hashes/dictionaries)
- Sequences (arrays/lists)

YAML Mappings (Dictionaries)

- YAML keeps data stored as a map containing keys and values associated to those keys:
  - Commonly called a “hash” or a “dictionary”
  - No order
- A dictionary is represented in a “<key>: <value>” form (note that colon must be followed by a space)

```
my_key: my_value
```

- Alternatively, a value can be associated with a key through indentation

```
my_key:
  my_value
```

YAML Sequences (Lists)

- Lists are used to store a collection of ordered items:
  - Scalar
  - Key/value pair
- All items start with a "- " (a dash and a space) on the same indentation level:
  - Item can be in the new line

```
---
- 1st_item
- 2nd_item_key: 2nd_item_value
-
  3rd_item
```

Example Ansible Playbook from YAML Perspective

List with single item

First-level keys in the list items

List of 2 items as value of the “tasks” key

```
---
- name: Techtorial example - Add VLAN to L2 trunk interface
  hosts: Site-1-Leafs

  vars:
    vlan: 123
    interface: Ethernet1/11

  tasks:
    - name: Configure interface to Layer2 (switchport)
      nxos_interface:
        name: "{{ interface }}"
        description: 'Configured by Ansible'
        mode: layer2
        state: present

    - name: Configure interface to trunk and add vlan to allowed vlans
      nxos_l2_interface:
        name: "{{ interface }}"
        mode: trunk
        trunk_vlans: "{{ vlan }}”
        state: present
```

Ansible Components

Configuration and Inventory

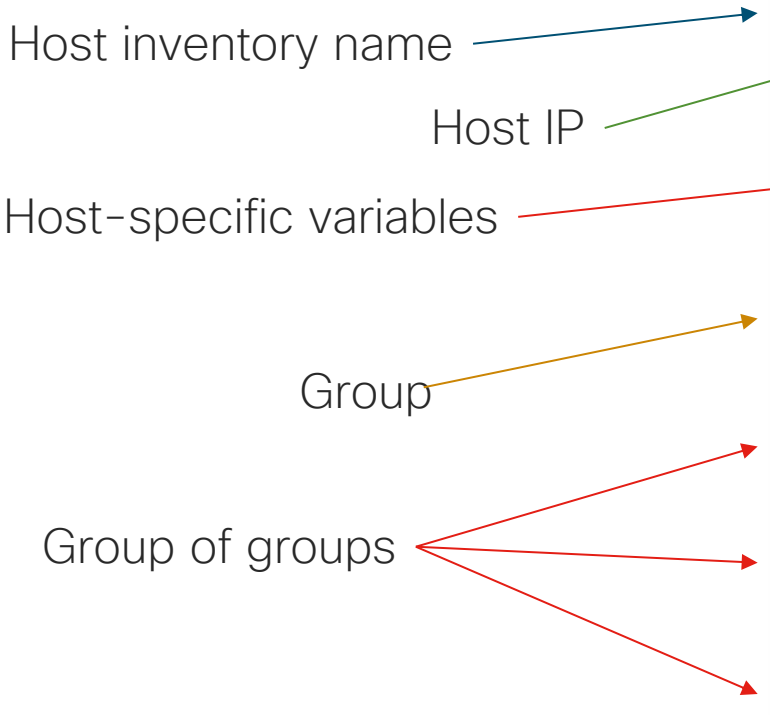
Ansible Configuration File

- Global configuration settings (usually the default values are fine)
- The configuration file is one variant of an INI format
- Sources of Configuration (by priority)
  - ANSIBLE_CONFIG (an environment variable)
  - ./ansible.cfg (in the current directory)
  - ~/ansible.cfg (in the home directory)
  - /etc/ansible/ansible.cfg file

Inventory File

- Inventory file identifies hosts, and groups of hosts which Ansible manages
  - INI format file (can be YAML format as well)
  - Hosts can be IP or FQDN
  - Groups enclosed in []
  - There are two reserved group names: [all] and [ungrouped]
    - Ungrouped: all hosts that don't have group membership other than [all]
  - Ranges or regexes can be used in host definitions
- Can include host-specific variables as well
- Sources (by priority)
  - Command line parameter ('-i <path>')
  - Environment Variable (ANSIBLE_INVENTORY)
  - Default ("/etc/ansible/hosts")

Inventory Files
Example



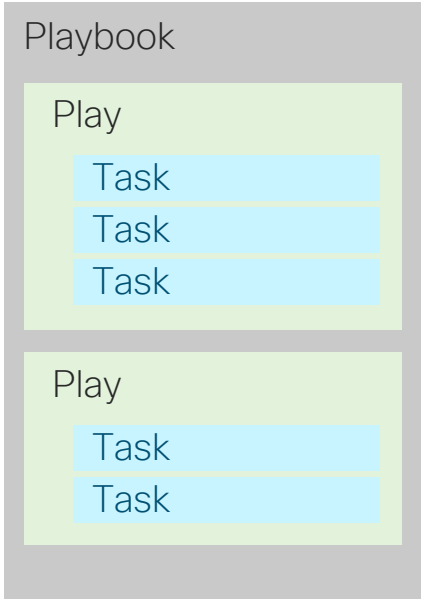

```
[Site-1-Leafs]
L11 ansible_host=<IP address>
L12 ansible_host=<IP address>

[Site-1-BGWs]
BG11 ansible_host=<IP address>
BG12 ansible_host=<IP address>

[Site-2-Leafs]
L21 ansible_host=<IP address>

[Site-2-BGWs]
BG21 ansible_host=<IP address>
BG22 ansible_host=<IP address>

[Site-1:children]
Site-1-Leafs
Site-1-BGWs

[Site-2:children]
Site-2-Leafs
Site-2-BGWs

[Multi-Site-Fabric:children]
Site-1
Site-2
```

Ansible Components

Playbooks and Plays

Playbooks and Plays

- Playbook is series of Ansible commands (tasks) targeted at a specific set of hosts/groups
- Each playbook is composed of one or more 'plays':
  - Play maps a group of hosts to some tasks
  - Play must contain a sequential list of tasks to execute
- Expressed in YAML format

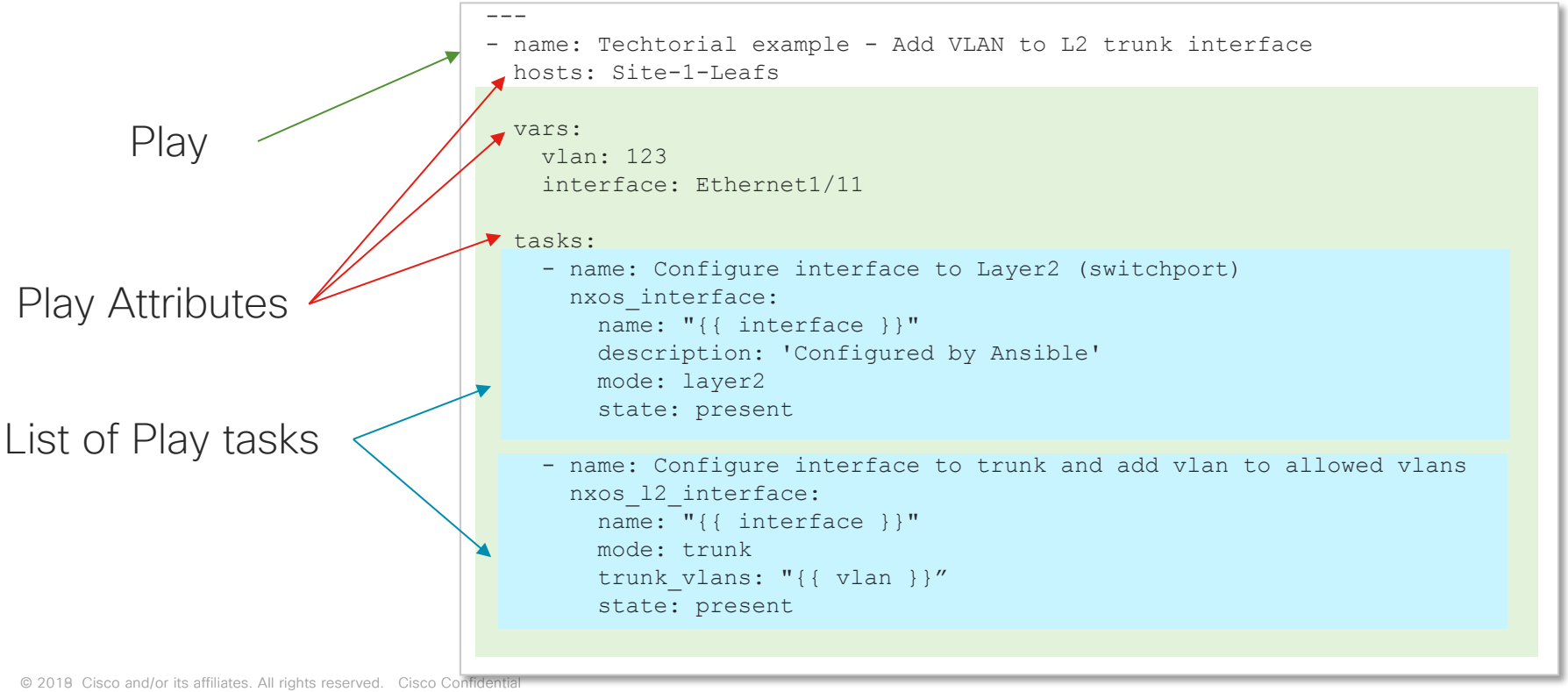


Structure Of a Play

- Host and users
  - Targets and user name to log in
- List of roles
  - Imports tasks, variables for a specific function
- List of tasks
  - Invoke modules with parameters
- List of handlers
  - List of tasks, runs if notified
  - Runs once, after all of the tasks complete in the play

```
---
- name: Play structure demo
  hosts: webservers
  remote_user: root

  roles:
    - common
    - webserver

  tasks:
    - name: Enable Apache module
      apache2_module:
        name:  "{{ module_name }}"
        state: latest
      notify:
        - restart apache2

  handlers:
    - name: restart apache2
      service:
        name: apache2
        state: restarted
```

Ansible Playbook
Example Ansible Playbook



Ansible Components

Tasks, Modules and Roles

Ansible Task
Introduction

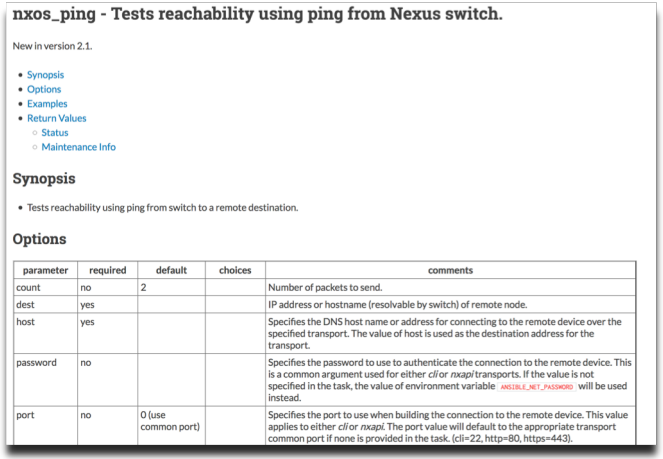
- Task is a call to an Ansible module
  - Formally, task definition is defined as a set of key/value pairs
  - Typical keys are name of the task and a module invocation
- Tasks are executed in order, one at a time
- Each task is executed against all machines matched by the host pattern
  - It is the purpose of a play to map a selection of hosts to tasks.

Ansible Task
Common Tasks Attributes

Name	Description
name	Name of the task
register	Variable which will contain the output of the module
args	Arguments of the module
when	Condition under which the task is executed
notify	Name of the handler to be notified

Modules

- Prepared “Scripts“ performing an action on a host (usually Python)
- They have defined input/output parameters
- Modules are called from Ansible playbook / CLI
- Default module library ships with Ansible (2146 modules in v2.7.6)
  - Cisco Network Modules: ACI, WLC, ASA, IOS / IOS XR, NX-OS
  - New in Ansible 2.5: UCS, NSO
- You can write your own modules



nxos_ping - Tests reachability using ping from Nexus switch.

New in version 2.1.

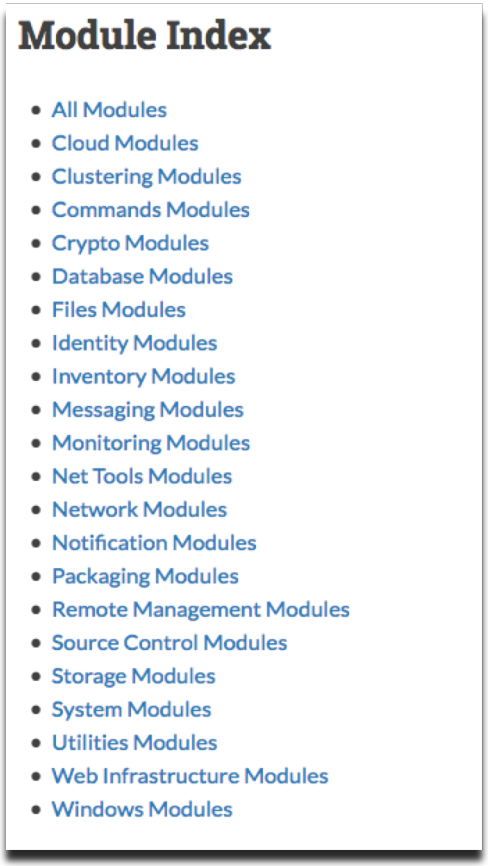
- Synopsis
- Options
- Examples
- Return Values
  - Status
  - Maintenance Info

Synopsis

- Tests reachability using ping from switch to a remote destination.

Options

parameter	required	default	choices	comments
count	no	2		Number of packets to send.
dest	yes			IP address or hostname (resolvable by switch) of remote node.
host	yes			Specifies the DNS host name or address for connecting to the remote device over the specified transport. The value of host is used as the destination address for the transport.
password	no			Specifies the password to use to authenticate the connection to the remote device. This is a common argument used for either *cli* or *nxapi* transports. If the value is not specified in the task, the value of environment variable ANSIBLE_NET_PASSWORD will be used instead.
port	no	0 (use common port)		Specifies the port to use when building the connection to the remote device. This value applies to either *cli* or *nxapi*. The port value will default to the appropriate transport common port if none is provided in the task. (cli=22, http=80, https=443).



Module Index

- All Modules
- Cloud Modules
- Clustering Modules
- Commands Modules
- Crypto Modules
- Database Modules
- Files Modules
- Identity Modules
- Inventory Modules
- Messaging Modules
- Monitoring Modules
- Net Tools Modules
- Network Modules
- Notification Modules
- Packaging Modules
- Remote Management Modules
- Source Control Modules
- Storage Modules
- System Modules
- Utilities Modules
- Web Infrastructure Modules
- Windows Modules

Ansible modules documentation:
http://docs.ansible.com/ansible/latest/modules.html

Example Ansible Playbook from Modules Perspective

Ansible module names

Ansible module input parameters

```
---
- name: Techtorial example - Add VLAN to L2 trunk interface
  hosts: Site-1-Leafs

  vars:
    vlan: 123
    interface: Ethernet1/11

  tasks:
    - name: Configure interface to Layer2 (switchport)
      nxos_interface:
        name: "{{ interface }}"
        description: 'Configured by Ansible'
        mode: layer2
        state: present

    - name: Configure interface to trunk and add vlan to allowed vlans
      nxos_l2_interface:
        name: "{{ interface }}"
        mode: trunk
        trunk_vlans: "{{ vlan }}"
        state: present
```

Ansible Roles

- Roles is comprised from all the tasks and variables needed to complete one specific unit of work
  - Can be used for implement a specific network feature (e.g NTP client, RR client etc.)
  - Use a pre-defined directory structure
- A role is the Ansible way of creating reusable content
- Ansible Galaxy is Ansible's official community hub for sharing Ansible roles.
  - https://galaxy.ansible.com/

Example Ansible Role

```
---
- name: Example for Roles
  hosts: Site-1-Leafs
  roles:
    nxos-ntp-client
    nxos-tacacs
```

```
+-- (directory of the playbook file or the inventory file)
    +-- roles
        +-- nxos-ntp-client
            +-- tasks
            |   +- main.yml
            |
            +-- vars
                +- main.yml
        +-- nxos-tacacs
            +-- tasks
            |   +- main.yml
            |
            +-- vars
                +- main.yml
```

```
---
- name: Ensure NX-OS NTP client is configured
  nxos_ntp:
    server: "{{ NtpServer }}"
    source_int: mgmt0
    vrf_name: management
    state: present
```

```
---
NtpServer: 10.62.45.1
```

```
---
- name: Ensure TACACS NX-OS feature is enabled
  nxos_feature:
    feature: tacacs
    state: enabled
- name: "Tacacs Server Host Configuration"
  nxos_aaa_server_host:
    state: present
    server_type: tacacs
    address: "{{ TacacsServer }}"
    key:  "{{ TacacsSharedSecret }}"
```

```
---
TacacsServer: 10.62.44.20
TacacsSharedSecret: „thisisasecret”
```

Ansible Components

Variables

Ansible Variables
Introduction

- Variable names consist of:
  - letters (should always start with a letter)
  - numbers
  - underscores
- Variable is referenced using the Jinja2 templating system:
  - In the simplest form:

```
{{ variable }}
```

  - If value starts with variable, then variable must be referenced using quotes:

```
"{{ variable }}"
```

Ansible Variables
Definition Sources

- Playbook

```
---
- name: Techtorial example - Add multiple vlan to L2 trunk interface
  vars:
    interface: Ethernet1/11
  vars_files:
    - group_vars/nxos.yml
```

- Inventory file

```
[Site-1-Leafs]
L11 ansible_host=<IP address>
L12 ansible_host=<IP address>

[Site-1-Leafs:vars]
Interface=Ethernet1/11
```

- In a pre-defined structure of folders:

```
+-- (directory of the playbook file or the inventory file)
    +-- host_vars
    |   +- L11.yml
    |
    +- group_vars
        +- Site-1-Leafs.yml
        +- all.yml
```

Ansible Variables
Source Priority

- Highest priority:
  - '--extra-vars' on the command line
- General:
  - 'vars' component of a playbook
  - Files referenced by 'vars_file' in a playbook
  - Included files and roles
  - Parameters passed to includes
  - 'register:' in tasks
- Lower priority:
  - Inventory (set on host or group)
  - Facts
- Lowest priority
  - Role defaults (from defaults/main.yml)

Ansible Components

Conditionals and Loops

Ansible Conditionals

- Task can be conditionally executed i.e. executed only if the conditional expression is true
- Conditional expression is defined as the value of the “when” statement

```
- name: do task if “result” has value “OK”
  when:
      result == “OK”
```

- Complex conditional expression with parentheses:

```
- name: "shut down CentOS 6 and Debian 7 systems”
    command: /sbin/shutdown -t now
    when: (ansible_distribution == "CentOS" and ansible_distribution_major_version == "6") or
          (ansible_distribution == "Debian" and ansible_distribution_major_version == "7")
```

Ansible Loops

- Task can be executed in a loop
- List of items that are iterated over are defined as a value of the “loop” task attribute
- Current item value in the loop is referred through built-in “item” variable
- Nested loops can be used
- Before 2.5 Ansible mainly used the with_<lookup> keywords to create loops
  - with_list, with_item, with_dict etc.
  - loop is equivalent with the with_list keyword

Ansible Loops
Example with Locally-defined List of Items

```
---
- name: Techtorial example - Add multiple vlan to L2 trunk interface
  hosts: Site-1-Leafs

  vars:
    interface: Ethernet1/11

  tasks:
    - name: Configure interface to Layer2 (switchport)
      nxos_interface:
        name: "{{ interface }}"
        description: 'Configured by Ansible'
        mode: layer2

    - name: Configure interface to trunk and add multiple vlans
      nxos_l2_interface:
        name: "{{ interface }}"
        mode: trunk
        trunk_vlans: "{{ item }}"
      loop:
        - 123
        - 456
        - 789
```

Ansible Loops
Example with Globally-defined List of Items

```
---
- name: Techtorial example - Add multiple vlan to L2 trunk interface
  hosts: Site-1-Leafs

  vars:
    interface: Ethernet1/11
    vlans:
      - 123
      - 456
      - 789

  tasks:
    - name: Configure interface to Layer2 (switchport)
      nxos_interface:
        name: "{{ interface }}"
        description: 'Configured by Ansible'
        mode: layer2

    - name: Configure interface to trunk and add multiple vlans
      nxos_l2_interface:
        name: "{{ interface }}"
        mode: trunk
        trunk_vlans: "{{ item }}"
      loop: "{{ vlans }}"
```

Reference to the "vlans" variable (list)

Using Ansible

CLI commands

Ansible CLI Tool Overview

Tool	Description
`ansible`	Executes modules against targeted hosts without creating playbooks.
`ansible-playbook`	Run playbooks against targeted hosts.
`ansible-vault`	Encrypt sensitive data into an encrypted YAML file.
`ansible-pull`	Reverses the normal “push” model and lets clients "pull" from a centralized server for execution.
`ansible-docs`	Parses the docstrings of Ansible modules to see example syntax and the parameters modules require.
`ansible-galaxy`	Creates or downloads roles from the Ansible community.

Ansible CLI Examples
Check Ansible version

```
[gabszabo@budlab-ansible ansible]$ ansible --version
ansible 2.5.0
  config file = /etc/ansible/ansible.cfg
  configured module search path = [u'/home/gabszabo/.ansible/plugins/modules',
u'/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python2.7/site-packages/ansible-2.5.0-py2.7.egg/ansible
  executable location = /usr/bin/ansible
  python version = 2.7.5 (default, Aug  4 2017, 00:39:18) [GCC 4.8.5 20150623 (Red Hat 4.8.5-16)]
```

Ansible and Cisco

Cisco Networking Portfolio

Ansible for Management of Cisco Infrastructure

- Ansible delivers native orchestration support for multiple Cisco platforms (as of Ansible v2.7):
  - NXOS, ACI
  - IOS / IOS-XE, IOS-XR
  - WLC, ASA, FTD
  - NSO, Meraki Cloud, UCS

- Ansible is formally in the Cisco Market Place:
  - https://marketplace.cisco.com/catalog/solution/155963?pid=94919

Ansible in Networking Environment

- Ansible is very well suited for Application and Operating System standup:
  - Particularly in virtualized environments where entire systems can be removed and rebuilt quickly, especially if the automation fails.
- It is well suited for networking systems except where:
  - Devices cannot be wiped and rebuilt
  - Fine grained control of the configuration for different services is required
  - Transactions across network devices with rollback is desired
  - Development of the Update and Deleting Playbooks is onerous.

Useful Links

Useful Links
Generic Ansible

- Ansible Repository (Github)
  - https://github.com/ansible/ansible
- Ansible Documentation
  - http://docs.ansible.com/ansible/latest/index.html
- Ansible Modules for IOS, IOS XR and NX-OS:
  - http://docs.ansible.com/ansible/latest/list_of_network_modules.html#ios
  - http://docs.ansible.com/ansible/latest/list_of_network_modules.html#iosxr
  - http://docs.ansible.com/ansible/latest/list_of_network_modules.html#nxos

Useful Links
Tools

- Ansible Summary (non-official):
  - https://gist.github.com/andreicristianpetcu/b892338de279af9dac067891579cad7d
- YAML Validator
  - https://jsonformatter.org/yaml-validator
- Online YAML Parser (JSON/Python output)
  - http://yaml-online-parser.appspot.com/
- Regular expression debugger
  - https://regex101.com/


CISCO