Network Automation with Ansible

April 13, 2018

Objective

After this session, you will be able to:

- Write Ansible playbooks to perform simple tasks
- Read and understand basic Ansible playbooks
- Research on your own to read and understand complex playbooks
- Research on your own to create playbooks to perform complex tasks

Fact-check

- Limited time (=4 hr.)
- Unfamiliar area
- Topic requires practice

Agenda

- Speakers Introduction
- Attendees Introduction
- Break
- History of Ansible
- Motivation
- Advantages of Ansible
- Ansible Market share
- Ansible in Devops
- Adopting Ansible for Operations
- Lessons learned

Agenda

Ansible Concepts

Lab

Basic Playbooks

Lab

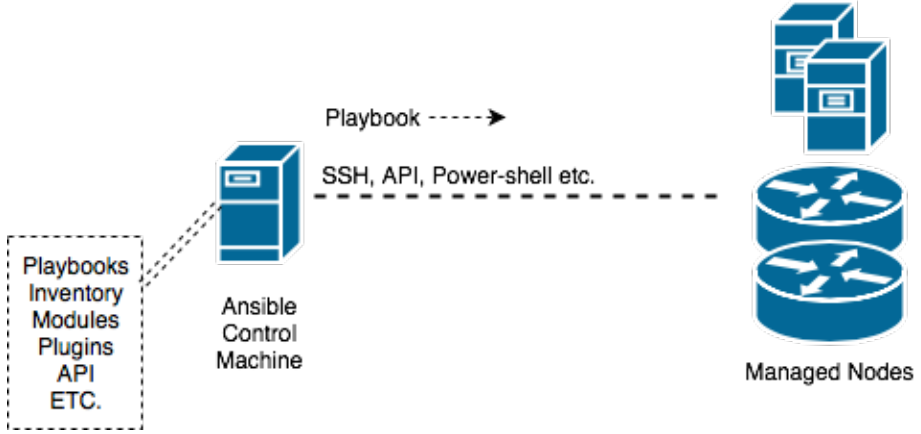
Break

Lab

Roles

Lab

What is Ansible



- Platform that can automate:
  - Software provisioning
  - Configuration management
  - Application deployment.
- Automation model: 2 components
  - Ansible Control Machine: Linux server with Ansible SW
  - Managed devices: Devices that are being automated by Ansible.
- Ansible Control Machine talks to devices over SSH (and other transports)
- Only requirement on on network devices = enable SSH

Ansible Concepts: Config File

- Ansible config file can be edited for customization
- To find your Ansible config file, do:
  - `$ ansible --version`
- Other methods to customize:
  - Multiple config files
  - Environmental settings
  - Command line options
  - Roles

- In this session, we will configure the below (uncomment):
  - inventory = /etc/ansible/hosts
  - host_key_checking = False
  - timeout = 10
  - retry_files_enabled = False

Ansible Concepts: Inventory File

- Managed device info is saved in inventory file
- Inventory file path should be uncommented in the config file
- Device info can be listed as individual hosts or groups
- Variables can be assigned to hosts or groups
- Default groups:
  - all
  - ungrouped

Example:

```
$ cat /etc/ansible/hosts

[IOS]
172.16.101.91 ansible_user=cisco ansible_ssh_pass=cisco

[XR]
172.16.101.92 ansible_user=cisco ansible_ssh_pass=cisco

[ALL:children]
IOS
XR
```

Ansible Concepts: Modules

- Modules are the nuts and bolts of Ansible automation tasks
- Playbooks use Modules to execute tasks on the managed devices
- Modules are Operating System specific.
- Example modules:
  - raw
  - ios_command
  - ios_config

```
$ ansible IOS -m raw -a "show ip route sum"
```

```
$ cat ios_sh_ip_route_sum.yml
---
- name: route summary from IOS devices
  hosts: IOS
  gather_facts: false
  tasks:
    - raw: sho ip route summary
```

Ansible Concepts: YAML

- YAML = YAML Ain't Markup Language
- YAML is a data serialization language
- Ansible playbooks are written in YAML
- YAML is intuitive, human readable
- Space indentation is important
- Tab invalid. Use “space” key.
- Check out Youtube links in the reference section

Ansible Concepts: YAML

- Key-value-pair is represented as:
  - <key><colon><space><value>
- List: “ordered data”, represented as:
  - <dash><space><data>
- Dictionary: “non-ordered data”
  - Bunch of key-value pairs
- There can be lists of dictionaries and dictionary of lists

- Key-value pair

```
platform: ASR9K
```

- List

```
  - show ip int brief
  - show ip route summ
```

- Dictionary

```
name: Verify Router OS
hosts: IOS
gather_facts: false
connection: local
```

Ansible Concepts: Playbooks

- Playbooks are the main means of Ansible automation.
- Playbook is a collection of plays
- Each play is a collection tasks
- Each task is a collection of modules

Example structure:

```
- name: play1
  hosts: group1
  tasks:
    - module1: parameters
    - module2: parameters

- name: play2
  hosts: group2
  tasks:
    - module1: parameters
    - module2: parameters
```

Ansible Concepts: Playbooks

Example:

- Capture the below data from IOS and XR devices
  - Interface list
  - Route summary
- Playbook:
  - play1
  - play2

```
- name: play1
  hosts: IOS
  tasks:
    - raw: show ip int br
    - raw: sho ip route summ
```

```
- name: play2
  hosts: XR
  tasks:
    - raw: show ipv4 int br
    - raw: sho route summ
```

Basic Playbooks

Basic Playbooks: ios & xr command module

- Module Names: ios_command & iosxr_command
- Module sends exec command to remote devices and returns the results
- Both modules require local connection execution method
- Required Parameters for ios & xr command module:
  - commands option : specify router command to retrieve data

ios_command

```
---
- name: IOS Module Router Config
  hosts: IOS
  gather_facts: false
  connection: local

  tasks:
    - name: Collect Router Version and Config
      ios_command:
        authorize: yes
        commands:
          - show version
          - show run

      register: value

    - debug: var=value.stdout_lines
```

iosxr_command

```
---
- name: XR Module Router Config
  hosts: XR
  gather_facts: false
  connection: local

  tasks:
    - name: Collect Router Version and Config
      iosxr_command:
        commands:
          - show version
          - show ip int bri

      register: value

    - debug: var=value.stdout_lines
```

Basic Playbooks: Register & Debug

- Basic Playbooks contain register and debug commands.
- Register
  - The “register” statement is used to capture the output of a task into a variable.
  - In previous example, we are saving the output of the show commands to the variable value.
  - Refer: http://docs.ansible.com/ansible/latest/playbooks_conditionals.html#register-variables
- Debug
  - The “debug” module prints statements during playbook execution.
  - The ”debug” modules takes in a var parameter, which is the variable you want to print.
  - Refer: http://docs.ansible.com/ansible/latest/debug_module.html

Basic Playbooks: ios & xr config module

- Module Names: ios_config & iosxr_config
- The config modules are used to configure the cisco routers.
- The modules uses parent and line options to structure the configuration in a hierarchical way.
- Both modules require local connection execution method.

ios_config

```
---
- name: IOS Module Router Config
  hosts: IOS
  gather_facts: false
  connection: local

 tasks:
    - name: Configure Interface Setting
      ios_config:
        parents: "interface Ethernet1”
        lines:
          - "description test”
          - "ip address 172.31.1.1 255.255.255.0”
```

iosxr_config

```
---
- name: XR Module Router Config
  hosts: XR
  gather_facts: false
  connection: local

  tasks:
    - name: Configure Interface Setting
      iosxr_config:
        parents: "interface GigabitEthernet0/0/0/0”
        lines:
          - "description test”
          - "ip address 172.31.1.1 255.255.255.0”
```

Basic Playbooks: Variables

- Ansible variables are used to store information that will change with each host.
- Variable can be defined:
  - inventory file (ansible_host)
  - created directly in the playbook
  - created in a separate file and included within the playbook.
- Variables are defined in playbooks
  - Using“{{ }}” the single/double quotes around double curly brackets
  - Using {{ }} the double curly brackets if its part of a sentence/string

```
---
- name: Play 1
  hosts: IOS
  gather_facts: false
  vars:
      host: "{{ ansible_host }}"
      username: ”This variable is {{ ansible_user }}"
      password: "{{ ansible_ssh_pass }}"
```

Basic Playbooks: Loops

- Ansible loops are used when repeatedly performing the same task with a set of different items.
- Ansible with_items loop is a combination of with_ and lookup().

With_items before Ansible Ver 2.5

```
tasks:
  - name: Collect Rtr Ver and Cfg
    ios_command:
       authorize: yes
       commands: "{{ item }}"

    with_items:
         - show version
         - show run
```

Updated to loop in Ansible Ver 2.5

```
tasks:
  - name: Collect Rtr Ver and Cfg
    ios_command:
       authorize: yes
       commands: "{{ item }}"

    loop:
         - show version
         - show run
```

Basic Playbooks: Conditionals

- Ansible conditionals are used in a statement to decide whether to run the task or not.
- Ansible uses a **when** clause to dictate a conditional which needs to be true in order for the task to be performed.

```
tasks:
   - name: Collect Router Version
     ios_command:
        authorize: yes
       commands:
         - show ip int bri
     when: ansible_user == "cisco"
```

Automating Network Operations Tasks

Network Automation Exercises

- Exercise 1 – Configure OSPF on all routers
  - Create Ansible playbook to configure OSPF on both IOS and XR router
  - Setup pre and post checks to ensure OSPF is working correctly
- Exercise 2 – Automatically backup router's config to server daily
  - Create Ansible playbook to capture router's running config from all Cisco device types.
  - Setup cron job to execute playbook daily and backup router's config on server.
- Exercise 3 – Create a playbook to compare two files to find the differences
  - Create Ansible playbook to find differences between two files.
  - Ex: comparing pre-upgrade and post-upgrade router captures.
- Exercise 4 – Ansible Vault
  - Use ansible-vault feature to encrypt sensitive data in inventory files.
  - Ex: comparing pre-upgrade and post-upgrade router captures.

Conclusion

- Ansible is an open-source, agentless automation tool that can be leveraged for networks configuration management functions.
- Ansible-playbooks provides capabilities to automate daily operations tasks.
- Automating repetitive tasks with Ansible can reduce OPEX costs and improve efficiency.
- With increasing support of modules, it is possible to automate even more network functions through Ansible.

Ansible Roles

Playbook

```
---
- name: output from IOS routers
  hosts: XR
  gather_facts: false
  connection: local
  vars:
    INTF: loopback1
  tasks:
    - name: read config
      iosxr_command:
        commands: show run int {{INTF}}
      register: DATA
    - name: print output
      debug: var=DATA.stdout_lines
```

Efficient Usage

- We can simplify
  - Playbook of playbooks
- We can modularize
  - vars
  - tasks
  - And other components
- We can reuse
  - Modularize components that are used repeatedly

Playbook of playbooks

- We can call playbooks within a playbook

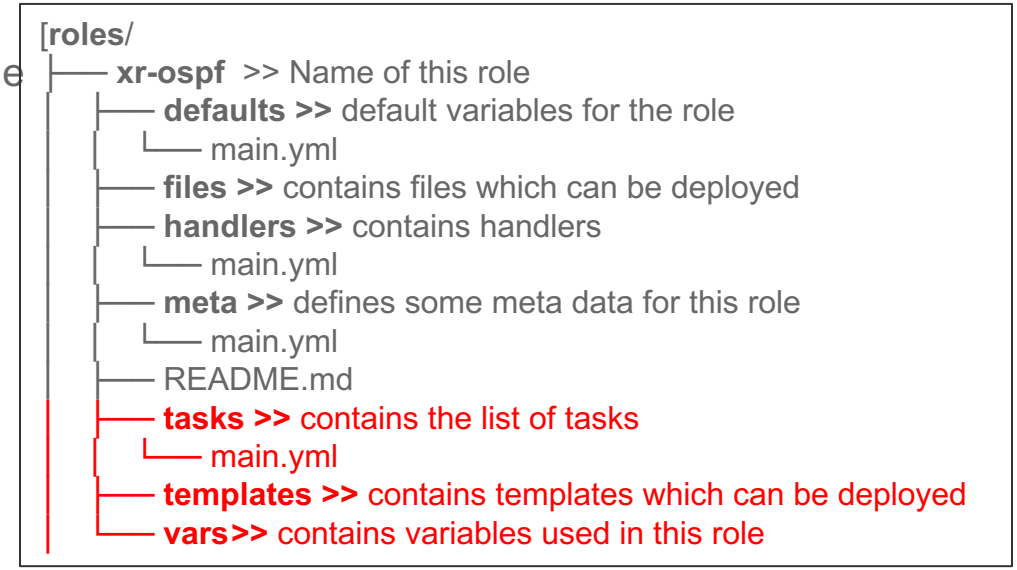
Playbook to acquire data from IOS and XR devices:

```
---
- name: ios config
  import_playbook: basic_ios_cmd.yml

- name: xr config
  import_playbook: basic_xr_cmd.yml
```

Roles

- Organize a large playbook into reusable file structures
- Creates a separation of functions; variables, tasks, & templates in unique directories
- Expects files main.yml, and .j2 files in respective folders
- File structure can be created manually or automatically via ansible CLI – "**ansible-galaxy**"

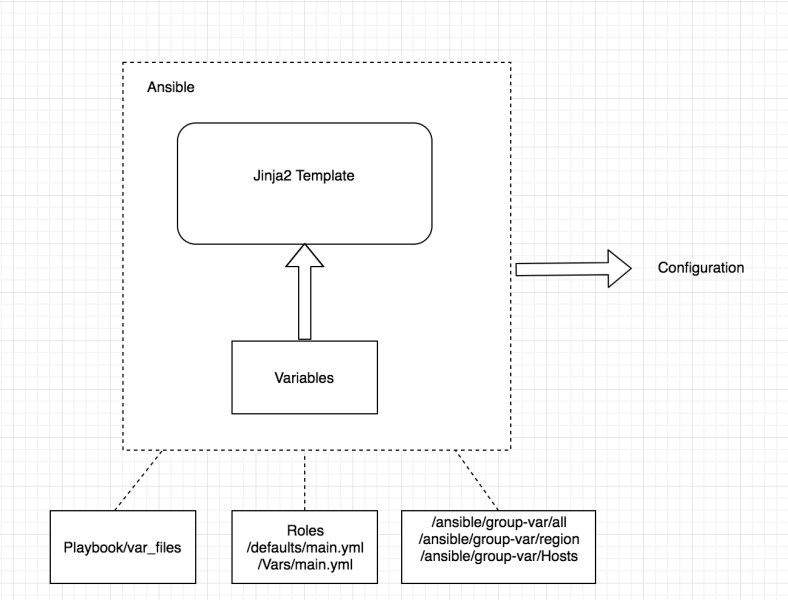


Roles Style Config

```
name: read config
iosxr_command:
commands: show run int {{INTF}}
register: OUT
name: print output debug: var=OUT.stdout_lines
```

Config Generation Using Templates

- Templates contain common and device/role specific elements
- Ansible uses Jinja 2 templating language for access to variables and logic/dynamic expression
- Jinja 2 template files end with .j2 ext
- Ansible can automatically access the Jinja2 templates through its Python API



Role with lists with single variables – Example 1

- Creating a role to generate configuration across multiple devices

```
- name: execute xr-config role
  hosts: localhost
  gather_facts: no

  roles:
    - xr-config

# playbook for executing role of xr-config
```

```
# Executes main.yml in xr-config/tasks/main.yml
- name: Generate the configuration from templates
  template: src=xr-config-template.j2
dest=/home/cisco/{{item.hostname}}.txt
  with_items:
    - "{{ router_hostname }}"

# tasks file for xr
```

```
# Variable defined in xr-config/vars/main.yml

---
router_hostname:
   - { hostname: router1 }
   - { hostname: router2 }
   - { hostname: router3 }
...
```

```
# Leverages j2 template for standard and variable config
hostname {{item.hostname}}
service timestamps log datetime msec
service timestamps debug datetime msec
clock timezone {{item.timezone}} {{item.timezone_offset}}
clock summer-time {{item.timezone_dst}} recurring
```

Jinja2 Template – For loop

- For Loop is a continuous loop until it runs out of inputs variables
- For Loop is invoked using {% for x in y %} syntax and ends with {% endfor %} syntax

```
# /template/template.j2
{% for INTF in interface_list %}
interface {{INTF}}
cost 1
!
{% endfor %}
  !

# /vars/main.yml
Interface_list:
  - GigabitEthernet0/0/0/0
  - GigabitEthernet0/0/0/1
```

Hierarchical templates and Block configs

- Base template *.J2 is pulled to specific template through {% extends ”base_config_template.j2" %} knob
- Configurations from specific template are inserted through block configs that being with { % block x %} and end with { % endblock % }

```
## Config lines from lsr_config referring base
template

{% extends "ler_lsr_config_template.j2" %}
```

```
#/templates/ ler_lsr_config_template.j2
hostname {{item.hostname}}
service timestamps log datetime msecservice
timestamps debug datetime msectelnet vrf default
ipv4 server max-servers 10telnet vrf Mgmt-intf
ipv4 server max-servers 10domain name
virl.infodomain lookup disablecdp
{% block rsvp %}
{% endblock %}
!,,
```

```
#/templates/ lsr__config.j2

{% block rsvp %}
!
rsvp
{% for interface in interface_list_ler %}
 interface {{interface}}
  bandwidth percentage 100
 !
{% endfor %}
{% endblock %}
```

Lab Exercises

- Exercise A – Create a playbook using role and Jinja2 template
  - Utilize roles to generate simple config by passing template and variable
- Exercise B – Create a playbook utilizing looping function
  - Utilize roles and Jinja2 template to create a config with looping function
- Exercise C – Create BGP generation for different device types
  - Utilize the templates and variables for config generation for different OS type
- Exercise D - Hierarchical Template
  - Utilize Hierarchical Template model for config generation

Reference

Reference

- Ansbile user guide URL
- Ansible installation URL
- YAML resources
  - http://docs.ansible.com/ansible/latest/YAMLSyntax.html
  - http://www.yaml.org
  - https://www.youtube.com/watch?v=cdLNKUoMc6c
  - https://www.youtube.com/watch?v=U9_gfT0n_5Q
- Ansible Training
  - Ansible for the Absolute Beginner @Udemy Click here
  - Ansible for Network Engineers @Udemy Click here
  - Kirk Byers Ansible training Jive page
  - Dcloud lab Ansible for Cisco Nexus Switches v1

Acknowledgement

Acknowledgements

- Some material in this session are sourced from Ansible docs
  - http://docs.ansible.com/ansible/latest/index.html

Lab Access

Step-1: SSH to hop-on-server

- IP address: 152.22.242.56
- Port: 8080
- Username: att-ansible
- Password: ansible@ATT18

Step-2: ssh to Ansible server

- $ ssh 172.16.101.X

TOMORROW starts here.