Network Automation with Ansible

TECDEV-4500

Gopal Naganaboyina | Gowtham Tamilselvan | Muthuraja Ayyanar | Yogi Raghunathan

Network Automation ?

- Becoming agile and move at scale
- Reduce deployment time while reducing OPEX cost
- Reduce human error; improve the efficiency and reliability of the networks

Session Objective

- Get started with Ansible
- Learn to read and write playbooks
- Automate simple tasks for IOS and XR devices

Time Schedule

- Lecture Session 1- 30 Mins
- Playbook Exercise – 60 Mins
- Lecture Session 2 – 20 Mins
- Conclusion – 10 mins
- Automating Exercises - Offline

Agenda Session 1

- Introduction to Ansible
- Yaml, Modules and Playbooks
- Variables, Loops and Conditionals
- Lab Exercises
  - ➢Ansible Introduction
  - ➢Playbook Primer

Agenda Session 2

- Automating Common Scenarios
- Lab Exercise
  - Automating Common Tasks
- Conclusion

Introduction to Ansible

Ansible

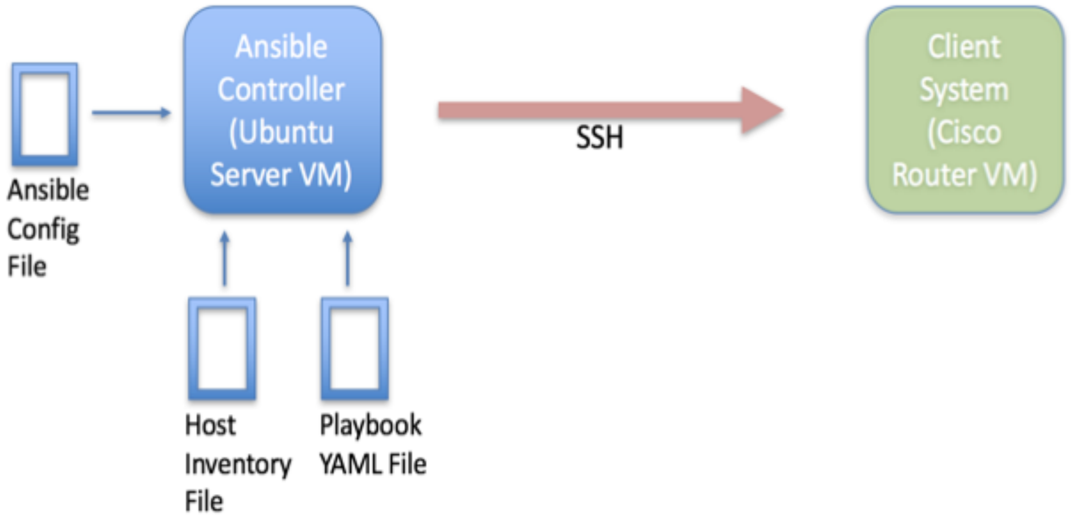
Simple Open-Source Automation Platform

Ansible

1. Agentless
2. Core module support for network devices
3. Provisioning, Configuration Management and Orchestration
4. Wide adoption

Ansible Overview

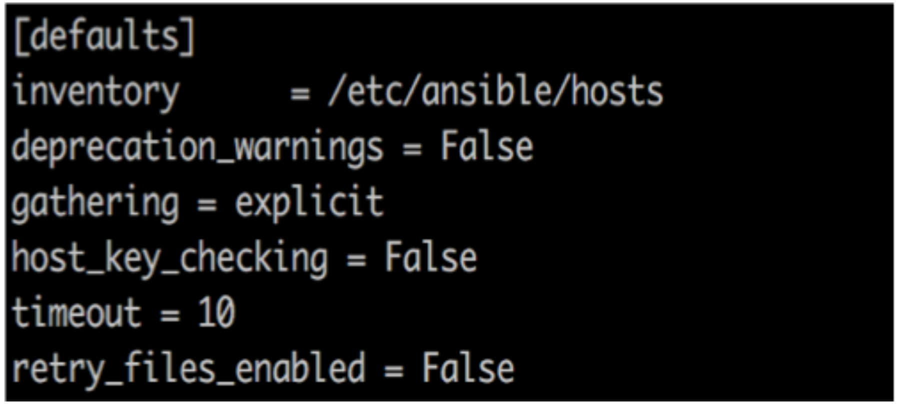
Ansible


Ansible Config File
Ansible Controller (Ubuntu Server VM)
SSH
Client System (Cisco Router VM)
Host Inventory File
Playbook YAML File

Ansible Configuration File – Ansible.cfg

Ansible

- Contain setting parameters defined to be used by Ansible.
- Ansible config file can be edited for customization
- To find your Ansible config file, do:
  - `$ ansible --version`



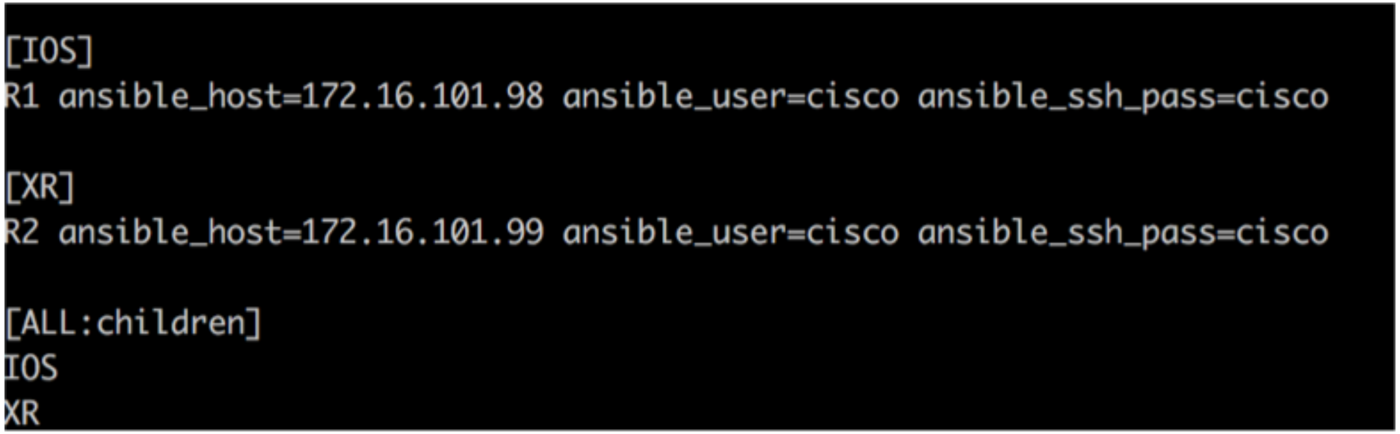
```
[defaults]
inventory       = /etc/ansible/hosts
deprecation_warnings = False
gathering = explicit
host_key_checking = False
timeout = 10
retry_files_enabled = False
```

Ansible Inventory File – Hosts

Ansible

- Contains information about the managed device (ex: IP address)
- Can hold variables
- Group hosts under []

- Default groups:
  - all
  - ungrouped



```
[IOS]
R1 ansible_host=172.16.101.98 ansible_user=cisco ansible_ssh_pass=cisco

[XR]
R2 ansible_host=172.16.101.99 ansible_user=cisco ansible_ssh_pass=cisco

[ALL:children]
IOS
XR
```

YAML, Modules and Playbooks

Yaml

Ansible

- Playbooks are written in yaml
- Intuitive and human readable
- Space indentation is important
- List
  - Always starts with "-"
  - Ordered Data
- Dictionary
  - Key: Value pairs
  - Unordered Data

List

```
    - show ip int brief
    - show ip route summary
```

Dictionary

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

Modules

Ansible

- Playbooks use Modules to execute tasks on the managed devices
- Standalone scripts
- Access from command line, playbook or API
  - ➢ios_command, ios_config
  - ➢iosxr_command, iosxr_config
- You can build your modules

```
cisco@ansible-controller:~$ ansible-doc -l | grep ^ios

ios_banner : Manage multiline banners on Cisco IOS
devices

ios_command : Run commands on remote devices
running Cisco IOS

ios_config : Manage Cisco IOS configuration sections
```

Ad-hoc Command

Ansible

- Allows to execute a single action on the managed device.

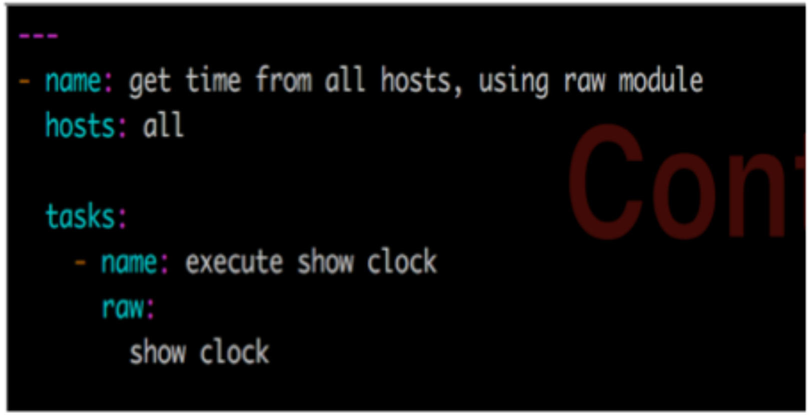
Syntax: ansible <devices> -m <module> -a <command>

- Devices must exist in the hosts file

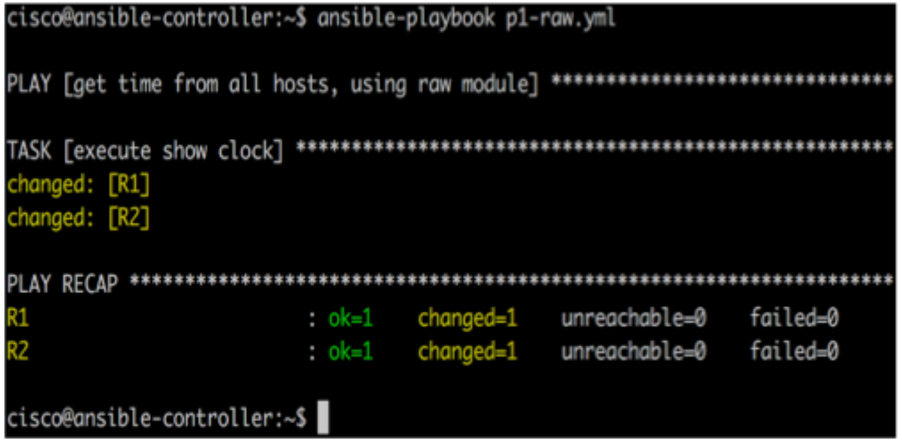
```
$ ansible IOS -m raw -a "show ip int brief”
R1 | SUCCESS | rc=0 >>Interface
IP-Address      OK? Method Status                Protocol
GigabitEthernet1       172.16.101.98   YES TFTP   up                    up
GigabitEthernet2       10.0.0.5        YES TFTP   up                    up
Loopback0              192.168.0.1     YES TFTP   up                    up
Loopback101            1.1.1.101       YES manual administratively down down
Shared connection to 172.16.101.98 closed.
Connection to 172.16.101.98 closed by remote host.
cisco@ansible-controller:~$
```

Playbooks

- Main means of Ansible automation.
- Collections of plays
- Each play is a collection of tasks
- Each task is a collection of modules



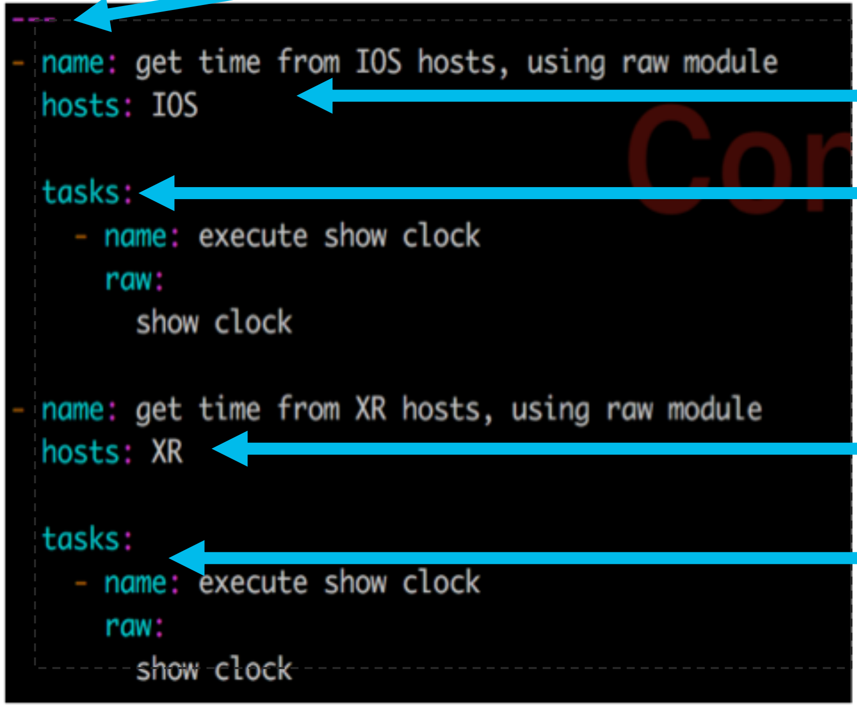
```
---
- name: get time from all hosts, using raw module
  hosts: all

  tasks:
    - name: execute show clock
      raw:
        show clock
```



```
cisco@ansible-controller:~$ ansible-playbook p1-raw.yml

PLAY [get time from all hosts, using raw module] *******************************

TASK [execute show clock] ******************************************************
changed: [R1]
changed: [R2]

PLAY RECAP *********************************************************************
R1                         : ok=1    changed=1    unreachable=0    failed=0
R2                         : ok=1    changed=1    unreachable=0    failed=0

cisco@ansible-controller:~$
```

Playbooks

Yaml files starts with ---



```
---
- name: get time from IOS hosts, using raw module
  hosts: IOS

  tasks:
    - name: execute show clock
      raw:
        show clock

- name: get time from XR hosts, using raw module
  hosts: XR

  tasks:
    - name: execute show clock
      raw:
        show clock
```

Ansible

1ST Play against target IOS

1st Task using Raw Module

2nd Play against target XR

2nd Task using Raw Module

Ansible Variables, Loops and Conditionals

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

```
---
- name: Backup IOS-XR Config
  hosts: csr
  gather_facts: false
  connection: local

  vars:
      host: "{{ ansible_host }}"
      username: "{{ ansible_user }}"
      password: "{{ ansible_ssh_pass }}"
```

Loops

Ansible

- Use when repeatedly performing the same task
- Can be used with value or Variables
- Used “with_items”

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

      with_items:
           - show version
           - show run
```

Conditionals

Ansible

- Use to run a task when a condition is met
- Uses “when” clause condition
- Should be true for task to run

```
tasks:
   – name: Collect Router Version
     ios_command:
       authorize: yes
        commands:
         - show ip int bri
     when: inventory_hostname == " R1"
```

Basic Playbook to retrieve output

Ansible

- Create a playbook using Yaml
- Identify the module to be used
  - Ios_command
  - Iosxr_command
  - Use register to capture the output
  - Use debug module to print on screen

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


tasks:
    - name: Collect Router Version and Config
      iosxr_command:
         commands:
            - show version
            - show run


      register: value

    - debug: var=value.stdout_lines
```

Reusing Playbooks

Ansible

- Playbooks can be reused inside other playbooks
- Use import_playbook or import_tasks modules to repurpose existing playbooks

```
---
- name: route summary from IOS routers
  import_playbook: p2-ioscmd.yml

 - name: route summary from XR routers
   import_playbook: p3-xrcmd.ymli
```

Session 1 – Lab Exercises

Complete Ansible Introduction & Playbook Primer Section

Automating Network Operations

Automation

- Commonly used tasks
  - Config backup
  - Health check
  - MOP
  - IBGP config generation
  - Bulk config generation

- Main tasks
  - Fetch running config (raw)
  - Save it in a file (copy)
  - Run Daily at 3AM, automatically

Automation

- Commonly used tasks
  - Config backup
  - Health check
  - MOP
  - IBGP config generation
  - Bulk config generation

Main tasks:

- Fetch critical data (iosxr_command)
- If specific metrics (when)
- Message accordingly

Automation

- Commonly used tasks
  - Config backup
  - Health check
  - MOP $\longrightarrow$
  - IBGP config generation
  - Bulk config generation

Main tasks:

- Capture data before change
  - ios/iosxr_command
  - Copy
  - tags
- Make config change
  - Ios/iosxr_config
  - meta
- Capture data after change
  - ios/iosxr_command
  - Copy
  - tags

Automation

- Commonly used tasks
  - config backup
  - Health check
  - MOP
  - IBGP config generation →
  - Bulk config generation

Main tasks:

- Generate config
  - Define template (j2)
  - Define variables (vars)
  - Execute template (tasks, template)
- Upload config
  - ios/xr_config, src
- A playbook to execute the Roles

All the above using Roles

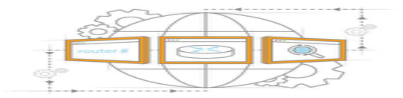
Automation

- Commonly used tasks
  - Config backup
  - Health check
  - MOP
  - IBGP config generation
  - Bulk config generation →

Main tasks:

- Create roles structure
  - ansible-galaxy
- Generate config
  - Define template (j2)
  - Define variables (vars)
  - Execute template (tasks, template)
- All the above using Roles

Network Configuration Management Software



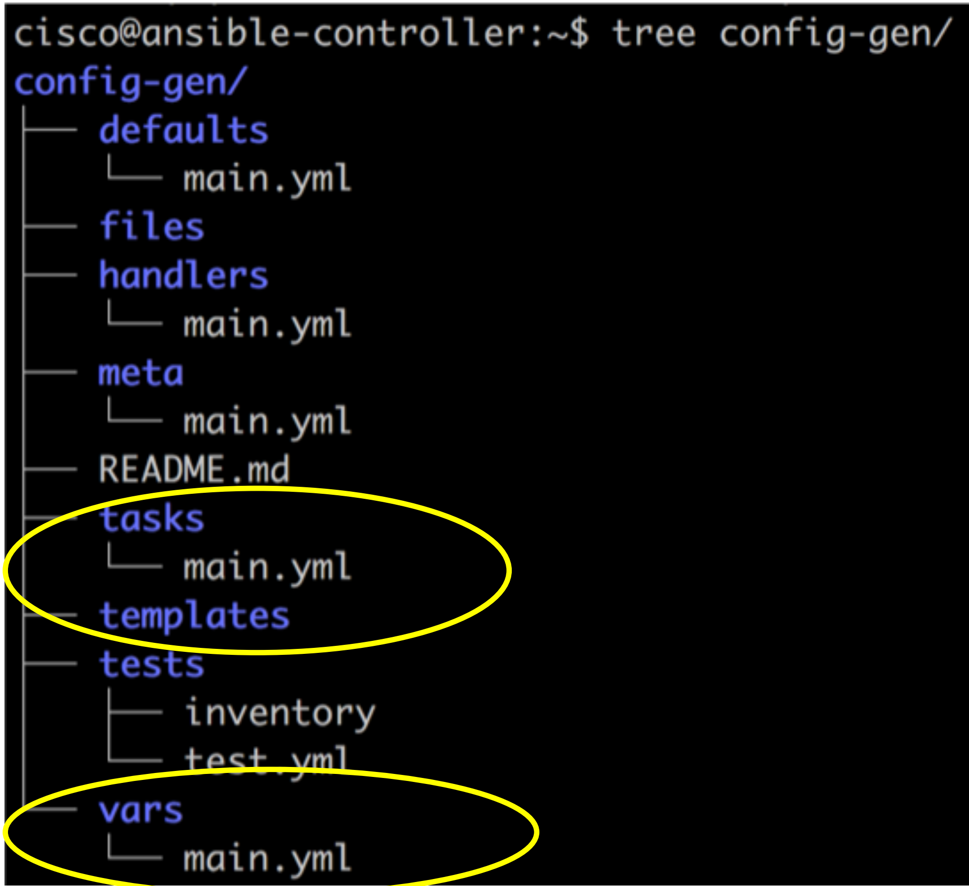
Ansible Roles and Jinja2 Templates

Roles

- Roles helps to organize complex and large playbooks into reusable components (file structures)
- Roles allows for separating components of playbooks: variables, tasks, & templates etc into unique directories
- Grouping contents using Ansible roles makes code/automation script sharing easier

Roles Directory Structure

- File structure can be created manually or automatically via ansible CLI – “**ansible-galaxy**”
- Directory structure contains multiple folders with sub-files (main.yml)
- 3 directories to focus on:
  - Tasks dir: used to host playbook tasks; defined in the main.yml
  - Vars dir: used to host playbook variable; defined in main.yml
  - Templates dir: used to host templates (j2 files) that are to be executed



```
cisco@ansible-controller:~$ tree config-gen/
config-gen/
├── defaults
│   └── main.yml
├── files
├── handlers
│   └── main.yml
├── meta
│   └── main.yml
├── README.md
├── tasks
│   └── main.yml
├── templates
├── tests
│   ├── inventory
│   └── test.yml
└── vars
    └── main.yml
```

Jinja Template

- Jinja is a template engine for the Python programming language
- Template engine contains variables and logic expression, which when evaluated are replaced with **actual** values.
- The variables are placed between tags or delimiters.
  - Jinja templates represents variables using {{ }} expressions
  - Jinja templates represents for loops using {% ... %} expressions

Config Generation Using Templates

- Green box shows sample router config
- Identify the elements that are static and the ones that are “dynamic”
- Use “tags” (with double curly braces)to identify the “dynamic” elements
- Blue box show how to change config into template
- Orange box shows how values can be passed to the variables inside the template. {key:value pairs}

```
username alpha secret beta
ntp server 9.9.9.9
logging host 9.9.9.10
```

```
username {{ user_name }} secret {{ password }}
ntp server {{ ntp_server }}
logging host {{ syslog_server }}
```

```
---
user_name: alpha
password: beta
ntp_server: 9.9.9.9
syslog_server: 9.9.9.10
```

Another example of Templating

- Example demonstrates how to use a for loop with Jinja templates
- Goal is to create 3 loopback interface configuration with unique IPv4 address
- Identify items to be parametrized
- {% ... %} is used for loop and {{... }} is used for variables
- For loop runs thrice since the variables file contains a list with three elements

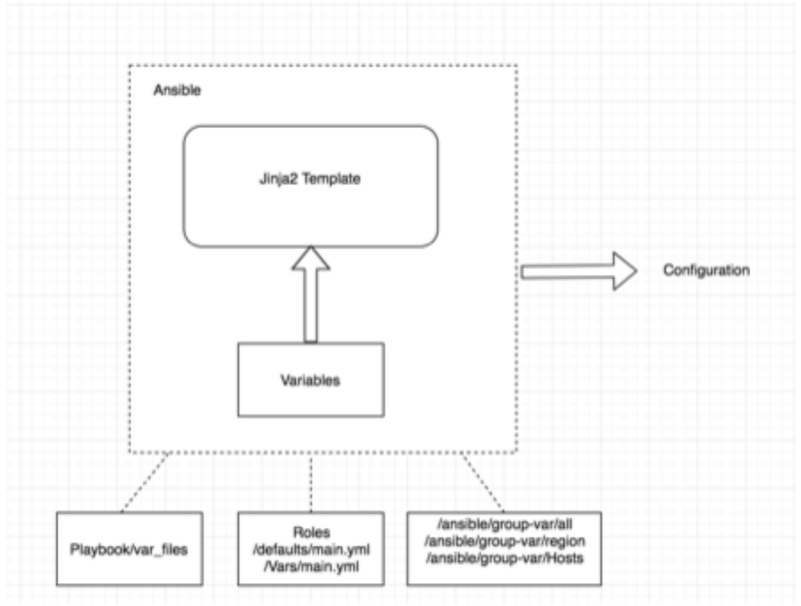
```
interface loopback 11
description Loopback 11 interface
ip address 11.11.11.11 255.255.255.255
!
interface loopback 12
description Loopback 12 interface
ip address 12.12.12.12 255.255.255.255
!
interface loopback 13
description Loopback 13 interface
ip address 13.13.13.13 255.255.255.255
!
```

```
{% for lb_list in interface_list %}
interface loopback {{lb_list.LB_IF_ID}}
description {{lb_list.DESC}}
ip address {{lb_list.IP_ADD}} {{lb_list.MASK}}
!
{% endfor %}
```

```
---
# vars file for Loopback interfaces
interface_list:
  - {LB_IF_ID: 11, IP_ADD: 11.11.11.11, MASK: 255.255.255.255,
DESC: Loopback 11 interface}
  - {LB_IF_ID: 12, IP_ADD: 12.12.12.12, MASK: 255.255.255.255,
DESC: Loopback 12 interface}
  - {LB_IF_ID: 13, IP_ADD: 13.13.13.13, MASK: 255.255.255.255,
DESC: Loopback 13 interface}
```

Config Generation Using Templates

- Templates contain common and device/role specific elements that are ”tagged” or parametrized
- Ansible roles when ran in a playbook combines Jinja 2 templating language and variables
- 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

- Creating a role to generate configuration across multiple devices

Roles Playbook:

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

  roles:
    - xr-config

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

/xr-config/tasks/main.yml:

```
# 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
```

/xr-config/vars/main.yml:

```
# Variable defined in xr-config/vars/main.yml
---
router_hostname:
   - { hostname: router1 }
   - { hostname: router2 }
   - { hostname: router3 }
...
```

/xr-config/templates/xr-config-template.j2:

```
# 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
```

Roles Summary

- Roles allows for separating components of playbooks:
  - tasks
  - vars
  - Templates and other components
- Roles help to modularize playbooks, increase reusability, and improve scalability

Session 2 – Lab Exercises

Complete Automating Common Tasks Section

Conclusion

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.

Reference

Reference

- Ansible 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

Thank you