TECDEV-4500

Automating Network Operations with Ansible

Gopal Naganaboyina

Gowtham Tamilselvan

Muthuraja Ayyanar

Yogi Raghunathan

Cisco live!

Agenda

- Introduction
- Lab Setup
- Ansible Overview
- YAML and Playbooks
- Ansible Variables, Loops, and Conditionals
- Automating Network Operations Tasks

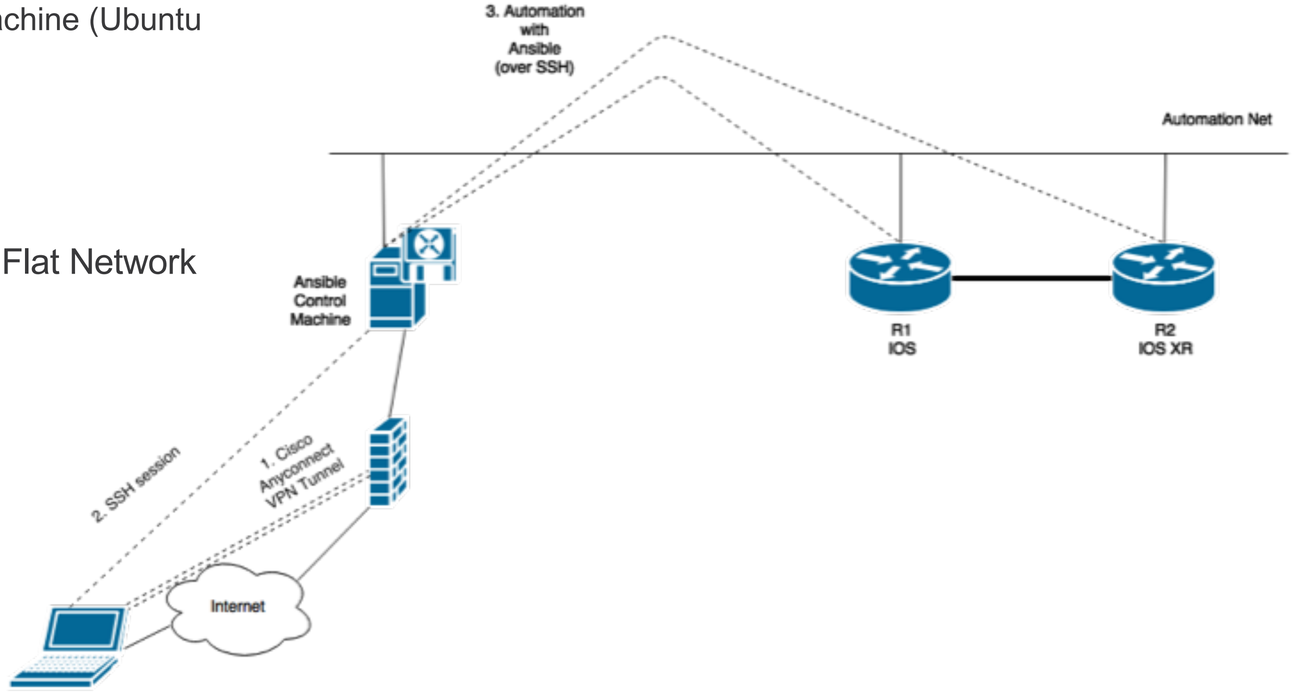
Lab Environment

Lab Topology

3 Node Topology:

1. Ansible Control Machine (Ubuntu server)
2. CSR1Kv Router
3. XRv Router

Mgmt. IP connectivity on Flat Network



Lab Device Access

- Step 1: Use Cisco ANYCONNECT to VPN to the PSL DMZ LAB
  - IP: 152.22.242.56
  - Username: cisco-ansible
  - Password: Cisco.123}
- Step 2: SSH to Ansible-Controller, no need to login to the Routers.
  - Ssh cisco@172.16.101.X (refer to pod sign up sheet)
  - Username: cisco
  - Password: cisco

Ansible Overview

Why Network Automation?

- Paradigm shift in the industry away from traditional network deployment and management lifecycles to a more agile model

- Companies want to reduce deployment time but increase network scale faster, all while reducing OPEX cost

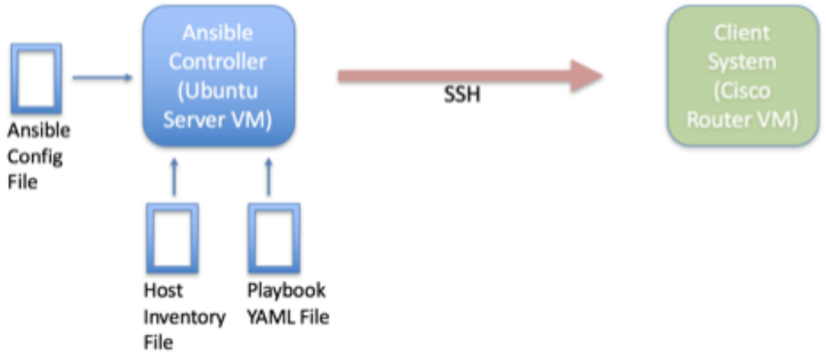
- Automation can improve the efficiency and reliability of the networks

Introducing Ansible

- Ansible is a simple open source IT automation platform created by Michael DeHaan and now owned by Red Hat.
- Ansible can be used for provisioning, configuration management, deployment and orchestration.
- Why Ansible over other automation tools?
  - Ansible is secure, communicates over SSH
  - Ansible is Agentless, no daemons running on the client devices
    - Unlike Puppet or Chef which require agents/daemons to be installed on client devices
  - Ansible is simple and human-readable, no expert-level coding skills needed

Ansible Requirements

- Two components involved in the Ansible environment:
  - Ansible-Controller - server running Ansible
  - Client/Target system
- Ansible-Controller requirements:
  - Any device non-windows device that has Python version 2.6 or higher installed
- Client System requirements:
  - Must have SSH enabled
  - Must have Python version 2.6 or higher installed if using remote execution*
  - Can run any operating system (Linux, OSX, Windows, Cisco OS, etc…)


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

Ansible Files

- Ansible uses two keys of files to execute an action on the client device: config file and the inventory file.
- Ansible Config files contain parameters defined/enabled to be used by Ansible when it gets executed.
- Inventory file contains information about the client/target systems such as their IP address/hostnames
- Another file commonly used with Ansible is the playbook, which is a script that contains a series of tasks set to be executed by Ansible.
    - Playbooks are not a required file, as Ansible can also be run in an ad-hoc mode to execute simple commands.

Ansible Config File

- Ansible Config files contain parameters defined for Ansible.
- Ansible will first look for a config file in the current working directory, if it doesn't find one then it will use the default file.
- Default Ansible config file can be found under */etc/ansible/ansible.cfg*

```
cisco@Ansible-Controller:~/project1$ vi ansible.cfg
[defaults]
inventory = ./inventory.txt

#Turn off ssh key checking
host_key_checking = False

#Turn set SSH Timeout to 10 Seconds
timeout = 10

retry_files_enabled = False
```

Ansible Inventory File

- Inventory file contains a list of Ansible managed-hosts.
- Ansible will first look for the inventory file in the current working directory, if it doesn’t find one then it will use the default file.
- Default Ansible inventory file can be found under */etc/ansible/hosts*
- Hosts are grouped using keys in **[ ]**
- Create groups of groups using **:children** suffix

```
[web_server]
Server1 ansible_host=192.168.0.1
Server2 ansible_host=192.168.0.2

[db_server]
Server3 ansible_host=192.168.0.1
Server4 ansible_host=192.168.0.2

[usa:children]
web_server
db_server
```

Complete Ansible Concepts – Inventory File Section

Ansible Modules

- Ansible modules are reusable, standalone python scripts that can be used by Ansible to complete a task.
- Users do not have to understand the underlying python code to execute a module.
- Ansible modules are **idempotent**, meaning running a module multiple times without any changes will result in the same effect as running it once.

- Ansible has 100s of modules published and available for use.
- Ansible allows for custom modules to be created.
- Example of Cisco Modules:
  - ios_command – to execute show commands on Cisco IOS routers
  - ios_config – to configure Cisco IOS routers
  - Similar modules available for Cisco IOS-XR and NXOS

Ansible Ad hoc command

- Ansible allows for modules to be executed inside of playbooks or individually from the command line.
- Using the “ansible” command, with inputs defining the client device and a module with arguments, you can perform a simple task on the host device.
- Ad hoc commands are used to execute a single action on the client device.

```
cisco@Ansible-Controller:~/project1$ ansible csr -m raw -a "show ip int bri" -u cisco -k -v
Using /home/cisco/project1/ansible.cfg as config file
SSH password:
csr | SUCCESS | rc=0 >>

Interface              IP-Address      OK? Method Status                Protocol
GigabitEthernet1       172.16.1.214    YES TFTP   up                    up
GigabitEthernet2       10.0.0.9        YES TFTP   up                    up
GigabitEthernet3       10.0.0.17       YES TFTP   up                    up
Loopback0              192.168.0.1     YES TFTP   up                    up
Connection to 172.16.1.214 closed by remote host.
Shared connection to 172.16.1.214 closed.
```

Complete Ansible Concepts – Ansible Modules Section

YAML and Playbooks

YAML Introduction

- YAML is a data serialization language designed to be human-friendly and work well with modern programming languages (Python, Perl, PHP, etc…)
- YAML is similar in format to JSON or XML but its easier to read and write due to minimal syntax restrictions
- Ansible Playbooks are written in YAML format

YAML Syntax

- YAML files begin with three dashes ("---") to indicate the start of the document.
- YAML files end with three dots ("..."), this is optional to include. The playbook will also end where there are no more actions to be performed.
- Comments begin with the number sign ("#") and can start anywhere on a line and continue until the end of the line.

```
---
# comments
...
```

YAML Syntax (Continued)

- **Lists** – are an array/sequence of items
- Conventional Block: Each entry begin with hyphen+space ("- ") and contain one item per line.

```
---
# List of Employees
- Adam
- John
- Zach
```

- Indented Block: Items can be enclosed inside square brackets and separated by a comma+ space.

```
---
# List of Employees
- [Adam, John, Zach]
```

YAML Syntax (Continued)

- **Dictionaries** – are a mapping of a key with a value.
- Keys are separated from values by a colon+space (“: ”).
- Indented Block: Use indentation and new lines to separate key/value pairs.

```
---
# Key: value
Name: Adam
Occupation: Engineer
```

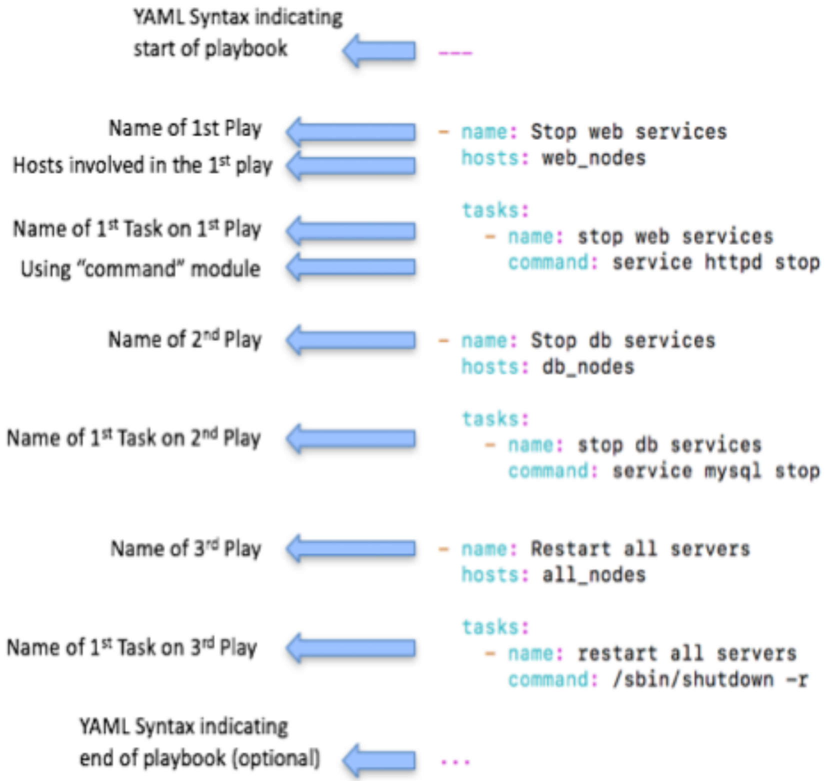
- Inline Block: Use comma+space and new lines to separate key/vaule pairs.

```
---
# Key: Value
{Name: Adam, Occupation: Engineer}
```

Ansible Playbooks

- Ansible Playbooks are a series of “plays” in a list.
- The goal of a play is to map a selection of hosts to some well-defined tasks.
- A task is an execution of an Ansible module with specific arguments (inputs).
- The tasks listed inside a playbook is executed in order, one at a time, against all hosts defined in the play.

Sample Playbook


YAML Syntax indicating start of playbook
Name of 1st Play
Hosts involved in the 1st play
Name of 1st Task on 1st Play
Using "command" module
Name of 2nd Play
Name of 1st Task on 2nd Play
Name of 3rd Play
Name of 1st Task on 3rd Play
YAML Syntax indicating end of playbook (optional)

```yaml
---

- name: Stop web services
  hosts: web_nodes

  tasks:
    - name: stop web services
      command: service httpd stop

- name: Stop db services
  hosts: db_nodes

  tasks:
    - name: stop db services
      command: service mysql stop

- name: Restart all servers
  hosts: all_nodes

  tasks:
    - name: restart all servers
      command: /sbin/shutdown -r

...
```

Complete Basic Playbooks Section

Ansible Variables, Loops, and Conditionals

Ansible Variables

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

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

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

Ansible 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().

```
tasks:
  - name: Collect Router Version and Config
    ios_command:
       authorize: yes
       commands: "{{ item }}"

    with_items:
         - show version
         - show run
```

Ansible Conditionals

- Ansible conditionals are used in a statement to decide whether to run the task or now.
- 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: inventory_hostname == "csr"
```

Automating Network Operation 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.

Complete Automating Network Operations Tasks Section

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.

References

- Ansible Documentation: http://docs.ansible.com/ansible/latest/index.html
- YAML Version 1.2 Specs: http://www.yaml.org/spec/1.2/spec.html
- Jinjia2 Templating: http://jinja.pocoo.org/docs/dev/templates/#
- Ansible Up and Running – Lorin Hochstein
- Python Configuration generation:
  - Kirk Byers Blog – Network configuration using Ansible

Thank you