CISCO LIVE 2018

ORLANDO, FL

Network Automation with Ansible

TECDEV-4500

Lab Guide

Gopal Naganaboyina | Yogi Raghunathan

Table of Contents

1. Ansible introduction

Objective

- Review Ansible customization in your pod
- Run Ansible ad-hoc commands

Lab exercises

- The following topics will be covered:
  - Configuration file
  - Inventory file
  - Ansible modules
  - Ad-hoc commands

1.1 Configuration file

- Ansible configuration file is preconfigured for you.
- We uncommented the following config lines, under [default] section:
  - inventory → use default inventory file, /etc/ansible/hosts
  - gathering: → set to explicit, to not gather facts by default
  - host_key_checking → Do not expect ssh keys for authentication
  - timeout → set timeout to 10 sec.
  - retry_files_enabled → do not create retry files
- Execute the below to review the Ansible installation and the config file:

```
$ which ansible
$ ansible --help
$ ansible --version
$ cat /etc/ansible/ansible.cfg
$ cat /etc/ansible/ansible.cfg | grep -v "#" | grep -v ^$
```

Example output

- Just for reference if you need to compare your lab output to someone else's.
- You don't need to refer to this if your steps go smooth.

```
cisco@ansible-controller:~$ which ansible
/usr/bin/ansible
cisco@server-1:~$ ansible --help
Usage: ansible <host-pattern> [options]
:
Some modules do not make sense in Ad-Hoc (include, meta, etc)
cisco@server-1:~$
cisco@ansible-controller:~$ ansible --version
ansible 2.5.0
  config file = /etc/ansible/ansible.cfg        <<<
  configured module search path = [u'/home/cisco/.ansible/plugins/modules',
u'/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python2.7/dist-packages/ansible
  executable location = /usr/bin/ansible
  python version = 2.7.6 (default, Oct 26 2016, 20:30:19) [GCC 4.8.4]
cisco@ansible-controller:~$ cat /etc/ansible/ansible.cfg | grep -v "#" | grep -v ^$
[defaults]
inventory      = /etc/ansible/hosts
gathering = explicit
host_key_checking = False
timeout = 10
deprecation_warnings = False
retry_files_enabled = False
:
cisco@ansible-controller:~$
```

Reference

> This is for later use; skip for now.
> http://docs.ansible.com/ansible/latest/reference_appendices/config.html#ansible-configuration-settings
>
> - Changes can be made and used in a configuration file which will be searched for in the following order:
>   - ANSIBLE_CONFIG (environment variable if set)
>   - ansible.cfg (in the current directory)
>   - ~/.ansible.cfg (in the home directory)
>   - /etc/ansible/ansible.cfg

> - Ansible will process the above list and use the first file found, all others are ignored.

1.2 Inventory file

- Inventory file is preconfigured for you. We added your router IP addresses in two groups: IOS and XR.
- Review the inventory file:

```
$ grep inventory /etc/ansible/ansible.cfg | grep hosts
$ cat /etc/ansible/hosts
$ cat /etc/ansible/hosts | grep -v ^# | grep -v ^$
```

- Verification:

```
$ ansible --list-hosts IOS
$ ansible --list-hosts XR
$ ansible --list-hosts ALL
$ ansible --list-hosts all
```

Example output

- Reference purpose only. Feel free to skip it.

```
cisco@ansible-controller:~$ grep inventory /etc/ansible/ansible.cfg | grep hosts
inventory      = /etc/ansible/hosts

cisco@ansible-controller:~$ grep -v "#" /etc/ansible/hosts | grep -v ^$
[IOS]
R1      ansible_host=172.16.101.XX ansible_user=cisco ansible_ssh_pass=cisco
[XR]
R2      ansible_host=172.16.101.XX ansible_user=cisco ansible_ssh_pass=cisco
[ALL:children]
IOS
XR
:
cisco@ansible-controller:~$ ansible --list-hosts IOS
  hosts (1):
    R1
cisco@ansible-controller:~$ ansible --list-hosts XR
  hosts (1):
    R2
cisco@ansible-controller:~$ ansible --list-hosts ALL
  hosts (2):
    R1
    R2
cisco@ansible-controller:~$ ansible --list-hosts all
  hosts (2):
    R1
    R2
```

Reference

> - This is for future reference. http://docs.ansible.com/ansible/latest/user_guide/intro_inventory.html
> - In our case, /etc/ansible/hosts is the default inventory file. It is possible to have multiple inventory files.

1.3 Ansible modules

- Ansible ships with several modules.
- Try the below commands for a **quick review**

```
$ ansible-doc --help
$ ansible-doc -l | grep ^ios_
$ ansible-doc -l | grep iosxr
$ ansible-doc -l
$ ansible-doc ios_command
```

Example output

```
cisco@ansible-controller:~$ ansible-doc --help
Usage: ansible-doc [-l|-F|-s] [options] [-t <plugin type> ] [plugin]

plugin documentation tool

Options:
  -a, --all             **For internal testing only** Show documentation for
                        all plugins.
  -h, --help            show this help message and exit
  -l, --list            List available plugins
  -F, --list_files      Show plugin names and their source files without
                        summaries (implies --list)
  -M MODULE_PATH, --module-path=MODULE_PATH
                        prepend colon-separated path(s) to module library
                        (default=[u'/home/cisco/.ansible/plugins/modules',
                        u'/usr/share/ansible/plugins/modules'])
  -s, --snippet         Show playbook snippet for specified plugin(s)
  -t TYPE, --type=TYPE  Choose which plugin type (defaults to "module")
  -v, --verbose         verbose mode (-vvv for more, -vvvv to enable
                        connection debugging)
  --version             show program's version number and exit

See man pages for Ansible CLI options or website for tutorials
```

```
https://docs.ansible.com
cisco@ansible-controller:~$
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
ios_facts                                            Collect facts from remote devices
running Cisco IOS
ios_interface                                        Manage Interface on Cisco IOS
network devices
ios_l2_interface                                     Manage Layer-2 interface on Cisco
IOS devices.
ios_l3_interface                                     Manage L3 interfaces on Cisco IOS
network devices.
ios_linkagg                                          Manage link aggregation groups on
Cisco IOS network devic...
ios_lldp                                             Manage LLDP configuration on Cisco
IOS network devices.
ios_logging                                          Manage logging on network devices
ios_ping                                             Tests reachability using ping from
Cisco IOS network devi...
ios_static_route                                     Manage static IP routes on Cisco IOS
network devices
ios_system                                           Manage the system attributes on
Cisco IOS devices
ios_user                                             Manage the aggregate of local users
on Cisco IOS device
ios_vlan                                             Manage VLANs on IOS network devices
ios_vrf                                              Manage the collection of VRF
definitions on Cisco IOS dev...
cisco@ansible-controller:~$ ansible-doc -l | grep iosxr
iosxr_banner                                         Manage multiline banners on Cisco
IOS XR devices
iosxr_command                                        Run commands on remote devices
running Cisco IOS XR
iosxr_config                                         Manage Cisco IOS XR configuration
sections
iosxr_facts                                          Collect facts from remote devices
running IOS XR
iosxr_interface                                      Manage Interface on Cisco IOS XR
network devices
iosxr_logging                                        Configuration management of system
logging services on ne...
iosxr_netconf                                        Configures NetConf sub-system
service on Cisco IOS-XR dev...
iosxr_system                                         Manage the system attributes on
Cisco IOS XR devices
iosxr_user                                           Manage the aggregate of local users
on Cisco IOS XR devic...
cisco@ansible-controller:~$ ansible-doc -l | grep iosxr | wc -l
9
cisco@ansible-controller:~$ ansible-doc -l | wc -l
1652
cisco@ansible-controller:~$
```

Reference

> For your research later; **not now**
>
> - http://docs.ansible.com/ansible/latest/modules/modules_by_category.html

---

1.4 Ad-hoc commands

- Let us use a few modules in this section: `raw` , `ios_command` , and `iosxr_command`
- Syntax: `ansible <devices> -m <module> -a <command>`
    - devices must exist in the inventory file
- Execute the below ad-hoc commands (from your home directory, /home/cisco)

```
$ ansible ALL -m raw -a "show clock"
$ ansible all -m raw -a "ping vrf Mgmt-intf 172.16.101.93"
$ ansible IOS --connection local -m ios_command -a "commands='show ip route summ'"
$ ansible XR --connection local -m iosxr_command -a "commands='show route summ'"
```

Optional exercises

> - Do this section if you are ahead of schedule, else skip and come back later.
> - This lab will be available to you for a few days after Cisco Live.
> - It is possible to use ad-hoc commands with more arguments, as below:
>
> ```
> $ ansible IOS -c local -m ios_command -a "authorize=true commands='show run
> int gig1'"
> $ ansible IOS -c local -m ios_command -a "username=cisco password=cisco
> auth_pass=cisco authorize=true commands='show run int gig1'"
> $ ansible XR -c local -m iosxr_facts -a "username=cisco password=cisco
> gather_subset=hardware"
> ```

Conclusion

- Review the section and discuss if you have any questions.

Example output

```
cisco@ansible-controller:~$ ansible --list-hosts IOS
```

```
  hosts (1):
    172.16.101.91
cisco@ansible-controller:~$ ansible --list-hosts XR
  hosts (1):
    172.16.101.92
cisco@ansible-controller:~$ ansible --list-hosts ALL
  hosts (2):
    172.16.101.91
    172.16.101.92
cisco@ansible-controller:~$ ansible --list-hosts all
  hosts (2):
    172.16.101.91
    172.16.101.92
cisco@ansible-controller:~$
cisco@ansible-controller:~$ ansible ALL -m raw -a "show clock"
172.16.101.91 | SUCCESS | rc=0 >>

*03:27:05.914 UTC Sun Jun 3 2018Shared connection to 172.16.101.91 closed.
Connection to 172.16.101.91 closed by remote host.


172.16.101.92 | SUCCESS | rc=0 >>


Sun Jun  3 03:28:36.549 UTC
03:28:36.589 UTC Sun Jun 3 2018

:

cisco@ansible-controller:~$ ansible all -m raw -a "ping vrf Mgmt-intf 172.16.101.93"
172.16.101.91 | SUCCESS | rc=0 >>

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.101.93, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/1 msShared connection to
172.16.101.91 closed.
Connection to 172.16.101.91 closed by remote host.


172.16.101.92 | SUCCESS | rc=0 >>


Sun Jun  3 03:28:53.207 UTC
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.101.93, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/1 ms
:
cisco@ansible-controller:~$ ansible IOS --connection local -m ios_command -a
"commands='show ip route summ'"
172.16.101.91 | SUCCESS => {
    "changed": false,
    "stdout": [
        "IP routing table name is default (0x0)\nIP routing table maximum-paths is
32\nRoute Source    Networks    Subnets     Replicates  Overhead    Memory
(bytes)\napplication     0           0           0           0           0\nconnected
0           4           0           384         1216\nstatic          0           0
0           0           0\nospf 1          0           1           0           96
308\n  Intra-area: 1 Inter-area: 0 External-1: 0 External-2: 0\n  NSSA External-1: 0 NSSA
External-2: 0\nbgp 1           0           0           0           0           0\n
External: 0 Internal: 0 Local: 0\ninternal        3
1432\nTotal           3           5           0           480         2956"
    ],
    "stdout_lines": [
        [
            "IP routing table name is default (0x0)",
            "IP routing table maximum-paths is 32",
            "Route Source    Networks    Subnets     Replicates  Overhead    Memory
(bytes)",
            "application     0           0           0           0           0",
            "connected       0           4           0           384         1216",
            "static          0           0           0           0           0",
            "ospf 1          0           1           0           96          308",
            "  Intra-area: 1 Inter-area: 0 External-1: 0 External-2: 0",
            "  NSSA External-1: 0 NSSA External-2: 0",
            "bgp 1           0           0           0           0           0",
            "  External: 0 Internal: 0 Local: 0",
            "internal        3                                               1432",
            "Total           3           5           0           480         2956"
        ]
    ]
}
cisco@ansible-controller:~$ ansible XR --connection local -m iosxr_command -a
"commands='show route summ'"
172.16.101.92 | SUCCESS => {
    "changed": false,
    "stdout": [
        "Route Source                    Routes     Backup     Deleted
Memory(bytes)\nlocal                           5          0          0          800
\nconnected                       1          4          0          800          \ndagr
0          0          0           0            \nospf 1                          1
0          0           160          \nbgp 1                           0          0
0           0            \nTotal                           7          4          0
1760"
    ],
    "stdout_lines": [
        [
            "Route Source                    Routes     Backup     Deleted
Memory(bytes)",
            "local                           5          0          0          800
",
            "connected                       1          4          0          800
",
            "dagr                            0          0          0          0
",
            "ospf 1                          1          0          0          160
",
            "bgp 1                           0          0          0          0
",
            "Total                           7          4          0          1760"
        ]
    ]
}
```

2. Playbook primer

- Ansible playbook exercises require creating YAML style playbook files.
- There are two options to create playbook files.
  - Option-1: Create the file in "Atom" editor, on the laptop. Atom is preconfigured with remote-sync: when you save the file in the default local directory(cl2018), it will automatically get copied to the home directory of your Ansible controller (/home/cisco).
  - Option-2: Use Ubuntu's "vi" or "vim" editor, which is included in your Ansible controller.

- The following topics are covered:
  - Raw Module
  - IOS Command Module
  - XR Command Module
  - IOS Config Module
  - XR Config Module
  - Variables
  - Loops
  - Conditionals
  - Importing Playbooks

2.1 Raw module

Lab exercise

- Let us use "raw" module in a playbook
- Create a playbook file (p1-raw.yml) with the below content, in your home directory, /home/cisco

```
---
- name: get interface info from all hosts
  hosts: ALL

  tasks:
    - name: execute show ip interface brief
      raw:
        show ip interface brief
```

- Predict the outcome of this playbook.
- Execute the play book:

```
$ ansible-playbook p1-raw.yml --syntax-check

$ ansible-playbook p1-raw.yml --step

$ ansible-playbook p1-raw.yml
$ ansible-playbook p1-raw.yml -v
```

- As you may have noticed, command output is not displayed.
- Create another playbook p1a-raw.yml, to print the output from routers

```
---
- name: get interface info from all hosts
  hosts: ALL

  tasks:
    - name: execute show intf
      raw:
        show ip int br

      register: P1A_RAW_OUTPUT

    - name: print data saved in the variable
      debug:
        var: P1A_RAW_OUTPUT.stdout_lines
```

- Predict the outcome of this playbook.
- Execute the playbook

```
$ ansible-playbook p1a-raw.yml --syntax-check

$ ansible-playbook p1a-raw.yml

$ ansible-playbook p1a-raw.yml -v
```

Conclusion

- In this section you used the raw module to collect and display command output from devices that are in the group, named ALL.
- Review the section and discuss if you have any questions.

Example output

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

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

TASK [execute show clock]
*****************************************************************************************
changed: [172.16.101.91]
changed: [172.16.101.92]
```

```
PLAY RECAP
*****************************************************************************************
***************
172.16.101.91              : ok=1    changed=1    unreachable=0    failed=0
172.16.101.92              : ok=1    changed=1    unreachable=0    failed=0

cisco@ansible-controller:~$
cisco@ansible-controller:~$ ansible-playbook p1a-raw --syntax-check

playbook: p1a-raw
cisco@ansible-controller:~$ ansible-playbook p1a-raw.yml

PLAY [get interface info from all hosts]
*************************************************************

TASK [execute show intf]
*****************************************************************************
changed: [172.16.101.91]
changed: [172.16.101.92]

TASK [print data saved in the variable]
**************************************************************
ok: [172.16.101.91] => {
    "P3_RAW_OUTPUT.stdout_lines": [
        "",
        "Interface              IP-Address      OK? Method Status
Protocol",
        "GigabitEthernet1       172.16.101.91   YES TFTP   up                    up
",
        "GigabitEthernet2       10.0.0.5        YES TFTP   up                    up
",
        "Loopback0              192.168.0.1     YES TFTP   up                    up
",
        "Loopback1              1.1.1.1         YES manual up                    up
"
    ]
}
ok: [172.16.101.92] => {
    "P3_RAW_OUTPUT.stdout_lines": [
        "",
        "",
        "Sun Jun  3 19:07:57.578 UTC",
        "",
        "Interface                      IP-Address      Status          Protocol Vrf-
Name",
        "Loopback0                      192.168.0.2     Up              Up       default
",
        "Loopback1                      1.1.1.2         Up              Up       default
",
        "Loopback99                     1.1.1.99        Up              Up       default
",
        "Loopback102                    1.1.1.102       Up              Up       default
",
        "MgmtEth0/0/CPU0/0              172.16.101.92   Up              Up       Mgmt-
intf",
        "GigabitEthernet0/0/0/0         10.0.0.6        Up              Up       default
"
    ]
}

PLAY RECAP
*****************************************************************************************
**
172.16.101.91              : ok=2    changed=1    unreachable=0    failed=0
172.16.101.92              : ok=2    changed=1    unreachable=0    failed=0
```

2.2 IOS command module

Lab exercise

- Use the ios_command to execute some exec level commands on IOS devices.
- Create a playbook, p2-ioscmd.yml, with the below contents:

```
---
- name: collect ip route summary from all IOS devices
  hosts: IOS
  connection: local

  tasks:
    - name: execute route summary command
      ios_command:
        commands:
          show ip route summary

      register: P2_OUTPUT

    - name: print output
      debug:
        var: P2_OUTPUT.stdout_lines
```

- Predict the outcome of this playbook.
- Execute the playbook
- After reviewing the playbook output, try run the playbook in verbose mode: with -v, -vv, or -vvv

```
$ ansible-playbook p2-ioscmd.yml --syntax-check

$ ansible-playbook p2-ioscmd.yml
```

Conclusion

- In this section you used the ios_command module to collect and display command output from an IOS device.
- Review the section and discuss if you have any questions

Reference

> Reference:
>
> - this is for future reference only (don't spend time on this now).
> - Pay attention to sections: parameters and return values.
> - http://docs.ansible.com/ansible/latest/modules/ios_command_module.html
> - http://docs.ansible.com/ansible/latest/modules/modules_by_category.html

Example output

```
cisco@ansible-controller:~$ ansible-playbook p2-ioscmd.yml

PLAY [collect ip route summary from all IOS devices]
**************************************************************

TASK [execute route summary command]
******************************************************************************
ok: [172.16.101.91]

TASK [print output]
*****************************************************************************************
******
ok: [172.16.101.91] => {
    "P2_OUTPUT.stdout_lines": [
        [
            "IP routing table name is default (0x0)",
            "IP routing table maximum-paths is 32",
            "Route Source    Networks    Subnets     Replicates  Overhead    Memory
(bytes)",
            "application     0           0           0           0           0",
            "connected       0           4           0           384         1216",
            "static          0           0           0           0           0",
            "ospf 1          0           1           0           96          308",
            "  Intra-area: 1 Inter-area: 0 External-1: 0 External-2: 0",
            "  NSSA External-1: 0 NSSA External-2: 0",
            "bgp 1           0           0           0           0           0",
            "  External: 0 Internal: 0 Local: 0",
            "internal        3                                               1432",
            "Total           3           5           0           480         2956"
        ]
    ]
}

PLAY RECAP
*****************************************************************************************
***************
172.16.101.91              : ok=2    changed=0    unreachable=0    failed=0
```

2.3 XR command module

Lab exercise

- Use the iosxr_command to execute some exec level commands on XR devices.
- Create a playbook, p3-xrcmd.yml, with the below contents

```
---
- name: collect ip route summary from all XR devices
  hosts: XR
  connection: local

  tasks:
    - name: execute route summary command
      iosxr_command:
        commands:
          show route summary

      register: P3_OUTPUT

    - name: print output
      debug:
        var: P3_OUTPUT.stdout_lines
```

- Predict the outcome of this playbook.
- Execute the playbook

```
$ ansible-playbook p3-xrcmd.yml --syntax-check
$ ansible-playbook p3-xrcmd.yml
```

Conclusion

- In this section you used the xr_command module to collect and display command output from an XR router.
- Review the section and discuss if you have any questions.

Optional exercise

> - Do this section if you are ahead of schedule, else skip and come back later.
> - This lab will be available for you to work for a few days after Cisco Live. You have the option of doing this later as well.
> - Write a playbook, op3-cmd.yml, to meet the below requirements:
>   - Collect output of route summary from both IOS and XR routers
>   - Use the modules, ios_command and iosxr_command
>   - Write 2 plays within one playbook.
> - Playbook solution is in the appendix section (op3-cmd.yml).

Example output

```
cisco@ansible-controller:~$ ansible-playbook p3-xrcmd.yml --syntax-check

playbook: p3-xrcmd.yml
cisco@ansible-controller:~$ ansible-playbook p3-xrcmd.yml

PLAY [collect ip route summary from all XR devices]
***************************************************************

TASK [execute route summary command]
******************************************************************************
ok: [172.16.101.92]

TASK [print output]
*****************************************************************************************
******
ok: [172.16.101.92] => {
    "P3_OUTPUT.stdout_lines": [
        [
            "Route Source                      Routes     Backup     Deleted
Memory(bytes)",
            "local                             4          0          0           640
",
            "connected                         1          3          0           640
",
            "dagr                              0          0          0           0
",
            "ospf 1                            1          0          0           160
",
            "bgp 1                             0          0          0           0
",
            "Total                             6          3          0           1440"
        ]
    ]
}

PLAY RECAP
*****************************************************************************************
***************
172.16.101.92              : ok=2    changed=0    unreachable=0    failed=0
```

2.4 IOS config module

Lab exercise

- Use ios_config module to configure a loopback interface on an IOS router.
- Create a playbook, p4-iosconfig.yml, with the below contents:

```
---
- name: configure loopback1 interface on IOS devices
  hosts: IOS
  connection: local

  tasks:
    - name: configure loopback101 interface
      ios_config:
        parents: interface loopback101
        lines:
          - description test config by p4
          - ip address 1.1.1.101 255.255.255.255
          - shutdown
```

- Execute the playbook
- Check if loopback101 interface is created by p4-iosconfig.yml playbook

```
$ ansible IOS -m raw -a "show run int loop101"

$ ansible-playbook p4-iosconfig.yml

$ ansible IOS -m raw -a "show run int loop101"
```

Conclusion

- In this section you used the ios_config module to configure a loopback interface on an IOS router.
- Review the section and discuss if you have any questions

Example output

```
cisco@ansible-controller:~$ ansible IOS -m raw -a "show run int loop101"
172.16.101.91 | SUCCESS | rc=0 >>

Line has invalid autocommand "show run int loop101"Shared connection to 172.16.101.91
closed.
Connection to 172.16.101.91 closed by remote host.


cisco@ansible-controller:~$ ansible-playbook p4-iosconfig.yml

PLAY [configure loopback1 interface on IOS devices]
*****************************************************************************************
*****

TASK [configure loopback101 interface]
*****************************************************************************************
******************
changed: [172.16.101.91]
```

```
PLAY RECAP
************************************************************************************************************************************
172.16.101.91              : ok=1    changed=1    unreachable=0    failed=0

cisco@ansible-controller:~$ ansible IOS -m raw -a "show run int loop101"
172.16.101.91 | SUCCESS | rc=0 >>

Building configuration...

Current configuration : 98 bytes
!
interface Loopback101
 description test config by p4
 ip address 1.1.1.101 255.255.255.255
end
Shared connection to 172.16.101.91 closed.
Connection to 172.16.101.91 closed by remote host.
```

2.5 XR config module

Lab exercise

- Use xr_config module to configure an access-list on a XR router.
- Create a playbook, p5-xrconfig.yml, with the below contents.

```
---
- name: configure ACL test7 on all XR devices
  hosts: XR
  connection: local

  tasks:
    - name: configure acl test7
      iosxr_config:
        parents: ipv4 access-list test7
        lines:
          - 10 permit ipv4 host 1.1.1.1 any
          - 20 permit ipv4 host 2.2.2.2 any
          - 30 permit ipv4 host 3.3.3.3 any
```

- Predict the outcome of executing the above playbook
- Check the ACL using commands below, before configuring
- Run the playbook
- Check for the post-playbook config

```
$ ansible XR -m raw -a "sho run ipv4 access-list"
$ ansible-playbook p5-xrconfig.yml

$ ansible XR -m raw -a "sho run ipv4 access-list"
```

Conclusion

- In this section you used the xr_config module to configure an access-list on an XR router.
- Review the section and discuss if you have any questions

Example output

```
cisco@ansible-controller:~$ ansible XR -m raw -a "sho run ipv4 access-list"
172.16.101.92 | SUCCESS | rc=0 >>


Mon May 28 16:45:40.825 UTC
% No such configuration item(s)


IMPORTANT:  READ CAREFULLY
:
Shared connection to 172.16.101.92 closed.
Connection to 172.16.101.92 closed by remote host.


cisco@ansible-controller:~$ ansible-playbook p5-xrconfig.yml

PLAY [configure ACL test7 on all XR devices]
*********************************************************

TASK [configure acl test7]
***************************************************************************
changed: [172.16.101.92]

PLAY RECAP
*****************************************************************************************
**
172.16.101.92              : ok=1    changed=1    unreachable=0    failed=0

cisco@ansible-controller:~$ ansible XR -m raw -a "sho run ipv4 access-list"
172.16.101.92 | SUCCESS | rc=0 >>


Mon May 28 16:45:54.404 UTC
ipv4 access-list test7
 1 remark configured by p7
 10 permit ipv4 host 1.1.1.1 any
 20 permit ipv4 host 2.2.2.2 any
 30 permit ipv4 host 3.3.3.3 any
!

IMPORTANT:  READ CAREFULLY
:
```

```
Connection to 172.16.101.92 closed by remote host.
```

2.6 Variables

- Ansible uses variables to enable more flexibility in playbooks.
- Variables are used to store information. This information can be used in a playbook by calling the specific variable.

Lab exercise

- Create custom variables inside a playbook to store an interface name and call the variable to complete the show command execution.
- Create a playbook, p6-vars.yml, with the below contents:

```
---
- name: get config of gig1 and gig2
  hosts: IOS
  connection: local
  vars:
    INTF1: GigabitEthernet1
    INTF2: GigabitEthernet2

  tasks:
    - name: disable proxy-arp
      ios_command:
        commands:
          - sho run int {{INTF1}}
          - sho run int {{INTF2}}

      register: PROX

    - name: print stuff, style-1
      debug:
        var: PROX

    - name: print stuff, style-2
      debug:
        var: PROX.stdout_lines[0]

    - name: print stuff, style-2
      debug:
        var: PROX.stdout_lines[1]
```

- Predict the outcome of executing the above playbook
- Run the playbook

```
$ ansible-playbook p6-vars.yml --syntax-check
$ ansible-playbook p6-vars.yml
```

Conclusion

- In this section you created variables inside a playbook and used the variables to complete a task execution.
- When multiple commands are there, "register" keeps track of output of each command. We used PROX.stdout_lines[n] to print them out separately.
- Do not distract yourself too much on print formatting, now. Stay focused on playbooks.
- Review the section and discuss if you have any questions

Optional exercise

> - Create a playbook to print messages, "This is " and "This is ".
> - Use default variable, "inventory_hostname". The messages will look like: "This is R1" and "This is R2"
> - Solution playbook (op6-vars.yml) is given in the Appendix section

Reference

> - This is a simple exercise on variables; recommend to read below pages for more advanced use cases.
> - http://docs.ansible.com/ansible/latest/user_guide/playbooks_variables.html
> - https://gist.github.com/andreicristianpetcu/b892338de279af9dac067891579cad7d

Example output

```
cisco@ansible-controller:~$ ansible-playbook p6-vars.yml --syntax-check

playbook: p6-vars.yml
cisco@ansible-controller:~$ ansible-playbook p6-vars.yml

PLAY [get config of gig1 and gig2]
*******************************************************************

TASK [disable proxy-arp]
*****************************************************************************
ok: [172.16.101.91]

TASK [print stuff]
***********************************************************************************
ok: [172.16.101.91] => {
    "PROX": {
        "changed": false,
        "failed": false,
        "stdout": [
            "Building configuration...\n\nCurrent configuration : 188 bytes\n!\ninterface
GigabitEthernet1\n description OOB Management\n vrf forwarding Mgmt-intf\n ip address
172.16.101.91 255.255.255.0\n negotiation auto\n cdp enable\n no mop enabled\n no mop
sysid\nend",
            "Building configuration...\n\nCurrent configuration : 177 bytes\n!\ninterface
GigabitEthernet2\n description Connected to XRV\n ip address 10.0.0.5 255.255.255.252\n
ip ospf cost 1\n negotiation auto\n cdp enable\n no mop enabled\n no mop sysid\nend"
        ],
        "stdout_lines": [
            [
                "Building configuration...",
                "",
                "Current configuration : 188 bytes",
                "!",
                "interface GigabitEthernet1",
                " description OOB Management",
                " vrf forwarding Mgmt-intf",
                " ip address 172.16.101.91 255.255.255.0",
                " negotiation auto",
                " cdp enable",
                " no mop enabled",
                " no mop sysid",
                "end"
            ],
            [
                "Building configuration...",
                "",
                "Current configuration : 177 bytes",
                "!",
                "interface GigabitEthernet2",
                " description Connected to XRV",
                " ip address 10.0.0.5 255.255.255.252",
                " ip ospf cost 1",
                " negotiation auto",
                " cdp enable",
                " no mop enabled",
                " no mop sysid",
                "end"
            ]
        ]
    }
}

PLAY RECAP
*****************************************************************************************
**
172.16.101.91              : ok=2    changed=0    unreachable=0    failed=0
```

2.7 Loops

- Loops are used to perform a task repeatedly with a set of different items.

Lab exercise

- Utilize Ansible loops to simplify execution of multiple show commands.
- Create a playbook, p7-loops.yml, with the below contents

```
---
- name: get config of gig1 and gig2 from IOS devices
  hosts: IOS
  connection: local

  tasks:
    - name: get config of gig1 and gig2 and time
      ios_command:
        commands:
          - "{{item}}"

      with_items:
           - show run int gig1
           - show run int gig2
           - show clock

      register: P7_OUT

    - name: print stuff
      debug:
        var: P7_OUT
```

- Predict the outcome of executing the above playbook
- Run the playbook

```
$ ansible-playbook p7-loops.yml --syntax-check

$ ansible-playbook p7-loops.yml
```

Conclusion

- In this section you created a loop to iterate the execution of multiple show commands.
- Review the section and discuss if you have any questions

Reference

> Reference: http://docs.ansible.com/ansible/latest/user_guide/playbooks_loops.html

Example output

```
cisco@ansible-controller:~$ ansible-playbook p7-loops.yml --syntax-check
```

```
playbook: p7-loops.yml
cisco@ansible-controller:~$ ansible-playbook p7-loops.yml

PLAY [get config of gig1 and gig2 from IOS devices]
**************************************************

TASK [get config of gig1 and gig2 and time]
**********************************************************
ok: [172.16.101.91] => (item=show run int gig1)
ok: [172.16.101.91] => (item=show run int gig2)
ok: [172.16.101.91] => (item=show clock)

TASK [print stuff]
***********************************************************************************
ok: [172.16.101.91] => {
    "P7_OUT": {
        "changed": false,
        "msg": "All items completed",
        "results": [
            {
            :
                },
                "item": "show run int gig1",
                "stdout": [
                    "Building configuration...\n\nCurrent configuration : 188
bytes\n!\ninterface GigabitEthernet1\n description OOB Management\n vrf forwarding Mgmt-
intf\n ip address 172.16.101.91 255.255.255.0\n negotiation auto\n cdp enable\n no mop
enabled\n no mop sysid\nend"
                ],
                "stdout_lines": [
                    [
                        "Building configuration...",
                        "",
                        "Current configuration : 188 bytes",
                        "!",
                        "interface GigabitEthernet1",
                        " description OOB Management",
                        " vrf forwarding Mgmt-intf",
                        " ip address 172.16.101.91 255.255.255.0",
                        " negotiation auto",
                        " cdp enable",
                        " no mop enabled",
                        " no mop sysid",
                        "end"
                    ]
                ]
            },
            {
              :
                  }
                },
                "item": "show run int gig2",
                "stdout": [
                    "Building configuration...\n\nCurrent configuration : 177
bytes\n!\ninterface GigabitEthernet2\n description Connected to XRV\n ip address 10.0.0.5
255.255.255.252\n ip ospf cost 1\n negotiation auto\n cdp enable\n no mop enabled\n no
mop sysid\nend"
```

```
                ],
                "stdout_lines": [
                    [
                        "Building configuration...",
                        "",
                        "Current configuration : 177 bytes",
                        "!",
                        "interface GigabitEthernet2",
                        " description Connected to XRV",
                        " ip address 10.0.0.5 255.255.255.252",
                        " ip ospf cost 1",
                        " negotiation auto",
                        " cdp enable",
                        " no mop enabled",
                        " no mop sysid",
                        "end"
                    ]
                ]
            },
            {
            :
                }
                },
                "item": "show clock",
                "stdout": [
                    "*21:40:30.056 UTC Mon May 28 2018"
                ],
                "stdout_lines": [
                    [
                        "*21:40:30.056 UTC Mon May 28 2018"
                    ]
                ]
            }
        ]
    }
}

PLAY RECAP
*****************************************************************************************
**
172.16.101.91              : ok=2    changed=0    unreachable=0    failed=0

cisco@ansible-controller:~$
```

2.8 Conditionals

- Conditionals are used, to decide whether to run a task or not.
- In this section, you will be working on "when" condition.

Lab exercise

- Collect route summary data by using appropriate command based on the router’s OS
- Create a playbook, p8-conditionals.yml, with the below contents

```
---
- name: get route summary from IOS and XR routers
  hosts: ALL

  tasks:
    - name: collect version info
      raw: show version

      register: SHVER

    - name: run "show ip route summ" on IOS routers
      when: SHVER.stdout | join('') is search('IOS XE')
      raw: show ip route summary

      register: IOSRT

    - debug: var=IOSRT.stdout_lines

    - name: run "show route summ" on XR routers
      when: SHVER.stdout | join('') is search('IOS XR')
      raw: show route summary

      register: XRRT

    - debug: var=XRRT.stdout_lines
```

- Predict the outcome of executing the above playbook
- Run the playbook:

```
$ ansible-playbook p8-conditionals.yml --syntax-check

$ ansible-playbook p8-conditionals.yml
```

Conclusion

- In this section you created a conditional statement to use a show command based on the Router OS.
- Review the section and discuss if you have any questions

Optional exercise

> - Create a playbook, which will:
>   - Detect router OS
>   - If a router has IOS, print message, "'hostname' is a IOS router" and if a router has XR, print, "'hostname' is a XR router"
>   - Playbook need to find the router names dynamically from the inventory file.

> - Solution playbook, op8-conditionals.yml is included in the Appendix section.

Reference

> Reference: http://docs.ansible.com/ansible/latest/user_guide/playbooks_conditionals.html#the-when-statement

Example output

```
cisco@ansible-controller:~$ ansible-playbook p8-conditionals.yml --syntax-check

playbook: p8-conditionals.yml
cisco@ansible-controller:~$ ansible-playbook p8-conditionals.yml

PLAY [get route summary from IOS and XR routers]
*****************************************************************************************
********

TASK [collect version info]
*****************************************************************************************
*****************************
changed: [R1]
changed: [R2]

TASK [run "show ip route summ" on IOS routers]
*****************************************************************************************
**********
skipping: [R2]
changed: [R1]

TASK [debug]
*****************************************************************************************
********************************************
ok: [R1] => {
    "IOSRTR.stdout_lines": [
        "",
        "IP routing table name is default (0x0)",
        "IP routing table maximum-paths is 32",
        "Route Source    Networks    Subnets     Replicates  Overhead    Memory (bytes)",
        "application     0           0           0           0           0",
        "connected       0           4           0           384         1216",
        "static          0           0           0           0           0",
        "ospf 1          0           1           0           96          308",
        "  Intra-area: 1 Inter-area: 0 External-1: 0 External-2: 0",
        "  NSSA External-1: 0 NSSA External-2: 0",
        "bgp 1           0           0           0           0           0",
        "  External: 0 Internal: 0 Local: 0",
        "internal        3                                               1432",
        "Total           3           5           0           480         2956"
    ]
}
ok: [R2] => {
    "IOSRTR.stdout_lines": "VARIABLE IS NOT DEFINED!"
}

TASK [run "show route summ" on XR routers]
*****************************************************************************************
**************
skipping: [R1]
changed: [R2]

TASK [debug]
*****************************************************************************************
********************************************
ok: [R1] => {
    "XRRTR.stdout_lines": "VARIABLE IS NOT DEFINED!"
}
ok: [R2] => {
    "XRRTR.stdout_lines": [
        "",
        "",
        "Mon Jun  4 03:09:34.788 UTC",
        "Route Source                    Routes     Backup     Deleted
Memory(bytes)",
        "local                           5          0          0           800
",
        "connected                       1          4          0           800
",
        "dagr                            0          0          0           0
",
        "ospf 1                          1          0          0           160
",
        "bgp 1                           0          0          0           0
",
        "Total                           7          4          0           1760
",
        ""
    ]
}

PLAY RECAP
*****************************************************************************************
**********************************************
R1                         : ok=4    changed=2    unreachable=0    failed=0
R2                         : ok=4    changed=2    unreachable=0    failed=0

cisco@ansible-controller:~$
```

2.9 Importing playbooks

- Ansible allows for one playbook to import and execute another playbook using the import_playbook module.
- We will be calling (or importing) using import_playbook module.

Lab exercise

- Create a playbook, p9-import.yml, with the below contents

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

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

- Read the playbooks p2-ioscmd.yml and p3-xrcmd.yml
- Predict the outcome of executing the playbook, p9-import.yml
- Run the playbook

```
$ cat p2-ioscmd.yml

$ cat p3-xrcmd.yml

$ ansible-playbook p9-import.yml --syntax-check

$ ansible-playbook p9-import.yml
```

Conclusion

- Note that import_play is to be used at **play-level** and not at task-level
- In this section you created a playbook to import and execute two other playbooks, rather than recreating the same playbook content again.
- Review the section and discuss if you have any questions.

Reference

> Details: https://docs.ansible.com/ansible/2.4/import_playbook_module.html

Example output

```
cisco@ansible-controller:~$ ansible-playbook p9-import.yml

PLAY [collect ip route summary from all IOS devices]
*************************************************************************************
****

TASK [execute route summary command]
*************************************************************************************
********************
ok: [R1]
```

```
TASK [print output]
*************************************************************************************************
****************************************
ok: [R1] => {
    "P2_OUTPUT.stdout_lines": [
        [
            "IP routing table name is default (0x0)",
            "IP routing table maximum-paths is 32",
            "Route Source    Networks    Subnets     Replicates  Overhead    Memory
(bytes)",
            "application     0           0           0           0           0",
            "connected       0           4           0           384         1216",
            "static          0           0           0           0           0",
            "ospf 1          0           1           0           96          308",
            "  Intra-area: 1 Inter-area: 0 External-1: 0 External-2: 0",
            "  NSSA External-1: 0 NSSA External-2: 0",
            "bgp 1           0           0           0           0           0",
            "  External: 0 Internal: 0 Local: 0",
            "internal        3                                               1432",
            "Total           3           5           0           480         2956"
        ]
    ]
}

PLAY [collect ip route summary from all XR devices]
*************************************************************************************************
*****

TASK [execute route summary command]
*************************************************************************************************
*********************
ok: [R2]

TASK [print output]
*************************************************************************************************
****************************************
ok: [R2] => {
    "P3_OUTPUT.stdout_lines": [
        [
            "Route Source                     Routes     Backup     Deleted
Memory(bytes)",
            "local                            5          0          0            800
",
            "connected                        1          4          0            800
",
            "dagr                             0          0          0            0
",
            "ospf 1                           1          0          0            160
",
            "bgp 1                            0          0          0            0
",
            "Total                            7          4          0            1760"
        ]
    ]
}

PLAY RECAP
```

```
*****************************************************************************************
**********************************************
R1                          : ok=2    changed=0    unreachable=0    failed=0
R2                          : ok=2    changed=0    unreachable=0    failed=0

cisco@ansible-controller:~$
```

3. Automating common tasks

- The following exercises will use Ansible to automate certain network operations tasks:
  - Router configuration backup
  - Device health monitoring
  - Method of procedure
  - Generate iBGP config using roles
  - Generating bulk configuration

3.1 Router config backup

Objective

- Create a playbook to capture and backup a router’s running config.

Approach

- We can one play with two tasks.
  - Task-1 will collect the config and save it to a variable. (use raw module)
  - Task-2 will save the contents of the variable into a file. (use copy module)

Lab exercise

- Review the contents in the playbook below.
- Task-1 collects and saves running-config in a variable named as RUNCFG
- Task-2 saves the contents of RUNCFG in a file in the home directory.
- Task-2 contains a well known variable (aka default variable) called, inventory_hostname. It means, "current inventory device", the router on which the tasks are run.
- Create the playbook with file name, p31-runcfg-bkup.yml, with the contents below.

```
---
- name: backup all routers config
  hosts: all
  tasks:
    - name: collect config  from all routers
      connection: ssh
      raw:
        show run

      register: RUNCFG

    - name: save output to a file
      connection: local
      copy:
        content: "{{ RUNCFG.stdout }}"
        dest: "/home/cisco/{{ inventory_hostname }}.txt"
```

- Predict the outcome of running this playbook
- Run the playbook
- Look for config files in `/home/cisco` directory

```
$ ansible-playbook p31-runcfg-bkup.yml --syntax-check

$ ansible-playbook p31-runcfg-bkup.yml

$ ls -l R*.txt
```

Conclusion

- You applied two modules, raw and copy, to get the router config.
- Note that the file names may be overwritten when you rerun the playbook. And, you need to manually run the playbook.
- The optional exercise at the end of this section addresses the above tow problems and enhance this playbook. Feel free to follow the optional playbook if you want to review the other playbook.
- Review the section and discuss if you have any questions.

Optional exercise

> - If you are interested, do this after completing all the exercises.
> - Create a playbook to backup routers' config, with the below requirements:
>   - Filename of the config files should include current timestamp to maintain uniqueness in filenames.
>   - Config backup should be taken daily at 03:00 hrs UTC. Use linux cronjob for this task.
> - Execute your playbook and verify if the results meet the requirements.
> - Fyi, a solution playbook file, op31-runcfg-bkup.yml is given in the appendix section for your reference.

Reference

> - Copy module: http://docs.ansible.com/ansible/latest/modules/copy_module.html
> - inventory_hostname: http://docs.ansible.com/ansible/latest/user_guide/playbooks_variables.html

Example output

```
cisco@ansible-controller:~$ cat p31-runcfg-bkup.yml
---
- name: backup all routers config
  hosts: all

  tasks:
    - name: collect config  from all routers
      connection: ssh
      raw:
        show run

      register: RUNCFG

    - name: save output to a file
      connection: local
      copy:
        content: "{{ RUNCFG.stdout }}"
        dest: "/home/cisco/{{ inventory_hostname }}.txt"

cisco@ansible-controller:~$ ansible-playbook p31-runcfg-bkup.yml --syntax-check

playbook: p31-runcfg-bkup.yml

cisco@ansible-controller:~$ ansible-playbook p31-runcfg-bkup.yml

PLAY [backup all routers config]
*********************************************************************

TASK [collect config  from all routers]
**************************************************************
changed: [172.16.101.91]
changed: [172.16.101.92]

TASK [save output to a file]
*************************************************************************
changed: [172.16.101.91]
changed: [172.16.101.92]

PLAY RECAP
*****************************************************************************************
**
172.16.101.91              : ok=2    changed=2    unreachable=0    failed=0
172.16.101.92              : ok=2    changed=2    unreachable=0    failed=0

cisco@ansible-controller:~$ ls -l 172.16.101.*
-rw-rw-r-- 1 cisco cisco 5066 May 31 21:14 172.16.101.91.txt
-rw-rw-r-- 1 cisco cisco 2008 May 31 21:14 172.16.101.92.txt

cisco@ansible-controller:~$ cat 172.16.101.91.txt | more
```

```
Building configuration...

Current configuration : 4768 bytes
!
! Last configuration change at 17:02:00 UTC Mon May 28 2018 by cisco
!
version 16.5
service timestamps debug datetime msec
service timestamps log datetime msec
platform qfp utilization monitor load 80
no platform punt-keepalive disable-kernel-core
platform console serial
:
cisco@ansible-controller:~$ cat 172.16.101.92.txt | more


Thu May 31 21:16:06.567 UTC
Building configuration...
!! IOS XR Configuration 6.2.2.15I
!! Last configuration change at Mon May 28 16:45:47 2018 by cisco
!
!  IOS-XR Config generated on 2018-03-27 20:08
! by autonetkit_0.24.0
hostname R2-XRv
```

3.2 Device health monitoring

Objective

- Create a playbook to collect critical data, process, and report issues.

Approach

- Collect critical data by using iosxr_command module
- Use conditional functions to process the collected data and report any issues.
- Note: The playbook shown here is only applicable for XR devices. It can be modified for any other device by using the appropriate Ansible module.

Lab exercise

- The first task will use the xr_command module to capture show command outputs from R2-XRv router.
- The second task will use the copy module to write the captured data into a file on the server. The inventory_hostname variable used here is a global variable, which means current device (out of the devices referred in "hosts"), that the tasks are run. Output of inventory_hostname will be R2; the hostname of the router defined in the inventory file.
- The tasks under the debug module use a conditional statement to check the captured data for abnormalities.

- Create a playbook, named, p32-xr-health-monitoring.yml

```
---
- name: XR Router Health Monitoring
  hosts: XR
  connection: local

  tasks:
    - name: Router Health Monitoring Commands
      iosxr_command:
        commands:
          - show platform
          - show redundancy
          - show proc cpu | ex "0%      0%       0%"
          - show memory sum location all | in "node|Pyhsical|available"
          - show ipv4 vrf all int bri
          - show route sum
          - show ospf neighbor
          - show mpls ldp neighbor | in "Id|Up"
          - show bgp all all sum | in "Address|^[0-9]+"

      register: iosxr_mon

    - name: save output to a file
      copy:
          content="\n\n ===show platform=== \n\n {{ iosxr_mon.stdout[0] }} \n\n ===show
redundancy=== \n\n {{ iosxr_mon.stdout[1] }} \n\n ===show proc cpu=== \n\n {{
iosxr_mon.stdout[2] }} \n\n ===show memory summary=== \n\n {{ iosxr_mon.stdout[3] }} \n\n
===show ipv4 vrf all int bri=== \n\n {{ iosxr_mon.stdout[4] }} \n\n ===show route sum===
\n\n {{ iosxr_mon.stdout[5] }} \n\n ===show ospf nei=== \n\n {{ iosxr_mon.stdout[6] }}
\n\n ===show mpls ldp neighbor=== \n\n {{ iosxr_mon.stdout[7] }} \n\n ===show bgp sum===
\n\n {{iosxr_mon.stdout[8] }}"
           dest="./{{ inventory_hostname }}_health_check.txt"

    - debug:
        msg: " {{ inventory_hostname }} show_platform indicates card is down"
      when: iosxr_mon.stdout[0] | join('') | search('Down')

    - debug:
        msg: " {{ inventory_hostname }} show_redundancy indicates card is not present"
      when: iosxr_mon.stdout[1] | join('') | search('NSR not ready since Standby is not
Present')

    - debug:
         msg: "{{ inventory_hostname }} CPU Utilization {{ iosxr_mon.stdout[2] }}"

    - debug:
        msg: " {{ inventory_hostname }} Memory Available: {{ iosxr_mon.stdout[3] }}"

    - debug:
        msg: " {{ inventory_hostname }} Interface is Down"
      when: iosxr_mon.stdout[4] | join('') | search('Down')

    - debug:
        msg: " {{ inventory_hostname }} Route Summary: {{iosxr_mon.stdout[5]}}"

    - debug:
        msg: " {{ inventory_hostname }} OSPF Summary: {{iosxr_mon.stdout[6]}}"
      when: iosxr_mon.stdout[6] | join('') | search('FULL')

    - debug:
        msg: " {{ inventory_hostname }} MPLS LDP Summary: {{iosxr_mon.stdout[7]}}"

    - debug:
        msg: " {{ inventory_hostname }} BGP Sessions Down: {{ iosxr_mon.stdout[8] }} "
      when: iosxr_mon.stdout[8] | join('') | search('Active')
```

- Predict the outcome of running this playbook
- Run the playbook

```
$ ansible-playbook p32-xr-health-monitoring.yml --syntax-check

$ ansible-playbook p32-xr-health-monitoring.yml
```

Conclusion

- Ansible playbooks can be leveraged for Proactive health monitoring to identify issues
- Operations can proactively identify faults by periodically running the playbook in network
- Data collected can be used in-line or offline for analysis

Example output

```
cisco@ansible-controller:~$ cp gowtham-playbooks/p32-xr-health-monitoring.yml .
cisco@ansible-controller:~$ ansible-playbook p32-xr-health-monitoring.yml --syntax-check

playbook: p32-xr-health-monitoring.yml
cisco@ansible-controller:~$ ansible-playbook p32-xr-health-monitoring.yml

PLAY [XR Router Health Monitoring]
*******************************************************************************

TASK [Router Health Monitoring Commands]
*************************************************************************
ok: [R2]

TASK [save output to a file]
*********************************************************************************
****
changed: [R2]

TASK [debug]
*********************************************************************************
********************
skipping: [R2]

TASK [debug]
*********************************************************************************
```

```
********************
ok: [R2] => {
    "msg": " R2 show_redundancy indicates card is not present"
}

TASK [debug]
*****************************************************************************************
********************
ok: [R2] => {
    "msg": "R2 CPU Utilization CPU utilization for one minute: 0%; five minutes: 0%;
fifteen minutes: 0%\n \nPID    1Min    5Min    15Min Process"
}

TASK [debug]
*****************************************************************************************
********************
ok: [R2] => {
    "msg": " R2 Memory Available: node:      node0_0_CPU0\n\fPhysical Memory: 3071M total
(1300M available)\n Application Memory : 2868M (1300M available)"
}

TASK [debug]
*****************************************************************************************
********************
skipping: [R2]

TASK [debug]
*****************************************************************************************
********************
ok: [R2] => {
    "msg": " R2 Route Summary: Route Source                    Routes     Backup
Deleted     Memory(bytes)\nlocal                            5          0          0
800          \nconnected                     1          4          0          800
\ndagr                          0          0          0          0           \nospf
1                           1          0          0          160          \nbgp 1
0          0          0          0            \nTotal                            7
4          0           1760"
}

TASK [debug]
*****************************************************************************************
********************
ok: [R2] => {
    "msg": " R2 OSPF Summary: * Indicates MADJ interface\n# Indicates Neighbor awaiting
BFD session up\n\nNeighbors for OSPF 1\n\nNeighbor ID     Pri   State           Dead Time
Address         Interface\n192.168.0.1     1     FULL/BDR        00:00:36    10.0.0.5
GigabitEthernet0/0/0/0\n    Neighbor is up for 15:08:07\n\nTotal neighbor count: 1"
}

TASK [debug]
*****************************************************************************************
********************
ok: [R2] => {
    "msg": " R2 MPLS LDP Summary: "
}

TASK [debug]
*****************************************************************************************
```

```
********************
skipping: [R2]

PLAY RECAP
*****************************************************************************************
**********************
R2                           : ok=8    changed=1    unreachable=0    failed=0

cisco@ansible-controller:~$
```

3.3 Method of Procedure (MOP)

- MOP is a documented step-by-step sequence of tasks executed on network devices to achieve a planned objective. In this case, that objective is provisioning of OSPF.

Objective

- Configure OSPF on both IOS and XR routers and enable OSPF on the loopback and GigE connected interface.
- Do not configure if OSPF is already found in the router.

Approach

- Step-1 : Create playbook to capture ospf data and run it as a pre-check
- Step-2: Configure OSPF on both routers
- Step-3: Run capture playbook again as post-check
- Step-4: Create one playbook to run through the full MOP

Lab exercise

Step-1: OSPF data capture

- Create a playbook to capture OSPF data on both routers.
- Create a playbook called p33-ospf-capture.yml
- This playbook has two plays:
  - first one to capture OSPF data from IOS routers
  - second one to capture OSPF data from XR routers.

```
---
- name: OSPF captures from IOS routers
  hosts: IOS
  connection: local

  tasks:
    - name: Collect IOS OSPF commands
      ios_command:
        authorize: yes
        commands:
           - show run | section ospf
           - show ip ospf interface brief
           - show ip ospf neighbor
           - show ip route ospf
      tags: always

      register: IOSPF

    - name: Save IOS precheck output to a file
      copy:
        content=" \n\n ===show run router ospf=== \n\n {{ IOSPF.stdout[0] }} \n\n ===show
ip ospf int bri=== \n\n {{ IOSPF.stdout[1] }} \n\n ===show ip ospf nei=== \n\n {{
IOSPF.stdout[2] }} \n\n ===show ip route ospf=== \n\n {{ IOSPF.stdout[3] \n\n }} "
        dest="./p33-precheck-ios-ospf.txt"
      tags: PRECHECK

    - name: Save IOS postcheck output to a file
      copy:
        content=" \n\n ===show run router ospf=== \n\n {{ IOSPF.stdout[0] }} \n\n ===show
ip ospf int bri=== \n\n {{ IOSPF.stdout[1] }} \n\n ===show ip ospf nei=== \n\n {{
IOSPF.stdout[2] }} \n\n ===show ip route ospf=== \n\n {{ IOSPF.stdout[3] \n\n }} "
        dest="./p33-postcheck-ios-ospf.txt"
      tags: POSTCHECK

- name: OSPF captures from XR routers
  hosts: XR
  connection: local

  tasks:
    - name: Collect XR OSPF commands
      iosxr_command:
        commands:
           - show run router ospf
           - show ospf interface brief
           - show ospf neighbor
           - show route ospf
      tags: always

      register: XOSPF

    - name: Save XR precheck output to a file
      copy:
        content=" \n\n ===show run router ospf=== \n\n {{ XOSPF.stdout[0] }} \n\n ===show
ospf int bri=== \n\n {{ XOSPF.stdout[1] }} \n\n ===show ospf nei=== \n\n {{
XOSPF.stdout[2] }} \n\n===show route ospf=== \n\n {{ XOSPF.stdout[3] \n\n }} "
        dest="./p33-precheck-xr-ospf.txt"
      tags: PRECHECK

    - name: Save XR postcheck output to a file
      copy:
        content=" \n\n ===show run router ospf=== \n\n {{ XOSPF.stdout[0] }} \n\n ===show
ospf int bri=== \n\n {{ XOSPF.stdout[1] }} \n\n ===show ospf nei=== \n\n {{
XOSPF.stdout[2] }} \n\n===show route ospf=== \n\n {{ XOSPF.stdout[3] \n\n }} "
        dest="./p33-postcheck-xr-ospf.txt"
      tags: POSTCHECK
```

- Predict the outcome of this playbook.
- Run the playbook. Note the **tags** option. We want the files to be saved as precheck since this capture is taken before configuring OSPF.
- Ensure that the playbook runs successfully and verify the existence of the precheck files in the home dir, /home/cisco

```
$ ansible-playbook p33-ospf-capture.yml --syntax-check

$ ansible-playbook p33-ospf-capture.yml --tags PRECHECK

$ ls -l p33-precheck*.txt
```

Step-2: Configure OSPF

- Create a playbook with 2 plays.
  - Play-1 will be to configure OSPF on IOS Routers
  - Play-2 will be to configure OSPF on XR Routers
- One of the requirements as outlined in the objectives above is not to configure OSPF if it is already present. We achieved this using "meta" module.
  - Here, we search for the string, OSPF and abort the play if it found.
- Create playbook, p33-ospf-config.yml, with the below contents:

```
---
- name: configure ospf on IOS routers - play-1
  hosts: IOS
  connection: local

  tasks:
    - name: pre-check for ospf config
      ios_command:
        commands:
          - show run | section ospf

      register: IOS_OSPF

    - meta: end_play
      when: IOS_OSPF.stdout | join('') | search('router ospf')

    - name: configure IOS ospf
      ios_config:
          parents: "router ospf 1"
          lines:
            - "router-id 192.168.0.1"
            - "passive-interface Loopback0"
            - "network 192.168.0.1 0.0.0.0 area 0"
            - "network 10.0.0.4 0.0.0.3 area 0"
          save_when: always

- name: configure ospf on XR routers - play-2
  hosts: XR
  connection: local

  tasks:
    - name: pre-check for ospf config
      iosxr_command:
        commands:
          - show run router ospf

      register: XR_OSPF

    - meta: end_play
      when: XR_OSPF.stdout | join('') | search('router ospf')

    - name: configure XR ospf
      iosxr_config:
        parents: "router ospf 1"
        lines:
          - "router-id 192.168.0.2"
          - "area 0"
          - "interface Loopback0"
          - "passive enable"
          - "exit"
          - "interface GigabitEthernet0/0/0/0"
          - "exit"
```

- Predict the outcome of this playbook.
- Check the existence of OSPF on the routers

```
$ ansible IOS -m raw -a "sho run | sec ospf"

$ ansible XR -m raw -a "sho run router ospf"

$ ansible XR -m raw -a "show route ospf"
```

- If there is ospf config, you may delete it, or simply continue.
- Run the playbook
- Verify that OSPF route exist on the routers.

```
$ ansible-playbook p33-ospf-config.yml --syntax-check

$ ansible-playbook p33-ospf-config.yml

$ ansible IOS -m raw -a "show ip ospf neigh"

$ ansible IOS -m raw -a "show ip route ospf"

$ ansible XR -m raw -a "show ospf neigh"

$ ansible XR -m raw -a "show route ospf"
```

Step-3: Post-config data capture

- Run p33-ospf-capture.yml to collect the post capture data for OSPF on both routers.
- Review the playbook above, p33-ospf-capture.yml. Look for `tags: POSTCHECK`
- Predict the outcome of this playbook, with tags=postcheck.
- Run the playbook:
- Ensure that the playbook runs successfully and verify the existence of the post-check files in the home dir, /home/cisco

```
$ ansible-playbook p33-ospf-capture.yml --syntax-check

$ ansible-playbook p33-ospf-capture.yml --tags POSTCHECK

$ ls -l p33-postcheck*.txt
```

Step-4: Create one playbook to run through the full MOP

- Create a playbook, named p33-mop.yml, with the contents below:

```
---
- name: pre-config captures
  hosts: localhost

  tasks:
    - name: run precheck playbook
      command: ansible-playbook p33-ospf-capture.yml --tags PRECHECK

- name: configure OSPF on both routers
  import_playbook: p33-ospf-config.yml

- name: post-config captures
  hosts: localhost

  tasks:
    - name: run postcheck playbook
      command: ansible-playbook p33-ospf-capture.yml --tags POSTCHECK
```

- Predict the outcome of this playbook
- Run the playbook:

```
$ ansible-playbook p33-mop.yml --syntax-check

$ ansible-playbook p33-mop.yml

$ ls -ltr p33*.txt
```

Conclusion

- Production MOPs require more stringent checks than shown here. This was a basic exercise to show how the 3 phases: Pre-check, Config Change, and Post-Check could be done using Ansible.
- Review the section and discuss if you have any questions.

Optional exercise

> - Add the below requirements to the above MOP playbook:
>   - Insert a delay of 30 seconds before collecting post-config data
>   - Create a file with the differences between pre-config and post-config data
> - Execute your playbook and verify if the results meet the requirements.
> - Solution playbook files (op-33-*.yml) are given in the appendix section for your reference.

Reference

> copy module: http://docs.ansible.com/ansible/latest/modules/copy_module.html meta module: http://docs.ansible.com/ansible/latest/modules/meta_module.html pause: http://docs.ansible.com/ansible/latest/modules/pause_module.html

Example output

```
cisco@ansible-controller:~$ cat p33-ospf-capture.yml
p33-ospf-capture.yml
cisco@ansible-controller:~$ cat p33-ospf-capture.yml
---
- name: OSPF captures from IOS routers
  hosts: IOS
  connection: local

  tasks:
    - name: Collect IOS OSPF commands
      ios_command:
        authorize: yes
        commands:
           - show run | section ospf
           - show ip ospf interface brief
           - show ip ospf neighbor
           - show ip route ospf
      tags: always

      register: IOSPF

    - name: Save IOS precheck output to a file
      copy:
        content=" \n\n ===show run router ospf=== \n\n {{ IOSPF.stdout[0] }} \n\n ===show
ip ospf int bri=== \n\n {{ IOSPF.stdout[1] }} \n\n ===show ip ospf nei=== \n\n {{
IOSPF.stdout[2] }} \n\n ===show ip route ospf=== \n\n {{ IOSPF.stdout[3] \n\n }} "
        dest="./p33-precheck-ios-ospf.txt"
      tags: PRECHECK
    - name: Save IOS postcheck output to a file
      copy:
        content=" \n\n ===show run router ospf=== \n\n {{ IOSPF.stdout[0] }} \n\n ===show
ip ospf int bri=== \n\n {{ IOSPF.stdout[1] }} \n\n ===show ip ospf nei=== \n\n {{
IOSPF.stdout[2] }} \n\n ===show ip route ospf=== \n\n {{ IOSPF.stdout[3] \n\n }} "
        dest="./p33-postcheck-ios-ospf.txt"
      tags: POSTCHECK

- name: OSPF captures from XR routers
  hosts: XR
  connection: local

  tasks:
    - name: Collect XR OSPF commands
      iosxr_command:
        commands:
           - show run router ospf
           - show ospf interface brief
           - show ospf neighbor
           - show route ospf
      tags: always

      register: XOSPF

    - name: Save XR precheck output to a file
      copy:
        content=" \n\n ===show run router ospf=== \n\n {{ XOSPF.stdout[0] }} \n\n ===show
ospf int bri=== \n\n {{ XOSPF.stdout[1] }} \n\n ===show ospf nei=== \n\n {{
XOSPF.stdout[2] }} \n\n===show route ospf=== \n\n {{ XOSPF.stdout[3] \n\n }} "
        dest="./p33-precheck-xr-ospf.txt"
      tags: PRECHECK

    - name: Save XR postcheck output to a file
      copy:
        content=" \n\n ===show run router ospf=== \n\n {{ XOSPF.stdout[0] }} \n\n ===show
ospf int bri=== \n\n {{ XOSPF.stdout[1] }} \n\n ===show ospf nei=== \n\n {{
XOSPF.stdout[2] }} \n\n===show route ospf=== \n\n {{ XOSPF.stdout[3] \n\n }} "
        dest="./p33-postcheck-xr-ospf.txt"
      tags: POSTCHECK
cisco@ansible-controller:~$ ansible-playbook p33-ospf-capture.yml --syntax-check

playbook: p33-ospf-capture.yml
cisco@ansible-controller:~$
cisco@ansible-controller:~$
cisco@ansible-controller:~$ ansible-playbook p33-ospf-capture.yml --tags PRECHECK

PLAY [OSPF captures from IOS routers]
*****************************************************************************************
*******************************************************


TASK [Collect IOS OSPF commands]
*****************************************************************************************
************************************************************
ok: [R1]

TASK [Save IOS precheck output to a file]
*****************************************************************************************
```

```
***************************************************
changed: [R1]

PLAY [OSPF captures from XR routers]
*****************************************************************************************
********************************************************

TASK [Collect XR OSPF commands]
*****************************************************************************************
*************************************************************
ok: [R2]

TASK [Save XR precheck output to a file]
*****************************************************************************************
****************************************************
changed: [R2]

PLAY RECAP
*****************************************************************************************
**********************************************************************************
R1                          : ok=2    changed=1    unreachable=0    failed=0
R2                          : ok=2    changed=1    unreachable=0    failed=0

cisco@ansible-controller:~$ ls -l p33-precheck*.txt
-rw-rw-r-- 1 cisco cisco 1392 Jun  7 20:56 p33-precheck-ios-ospf.txt
-rw-rw-r-- 1 cisco cisco  963 Jun  7 20:56 p33-precheck-xr-ospf.txt
cisco@ansible-controller:~$ cat p33-ospf-config.yml
---
- name: configure ospf on IOS routers - play-1
  hosts: IOS
  connection: local

  tasks:
    - name: pre-check for ospf config
      ios_command:
        commands:
          - show run | section ospf

      register: IOS_OSPF

    - meta: end_play
      when: IOS_OSPF.stdout | join('') | search('router ospf')

    - name: configure IOS ospf
      ios_config:
          parents: "router ospf 1"
          lines:
            - "router-id 192.168.0.1"
            - "passive-interface Loopback0"
            - "network 192.168.0.1 0.0.0.0 area 0"
            - "network 10.0.0.4 0.0.0.3 area 0"
          save_when: always

- name: configure ospf on XR routers - play-2
  hosts: XR
  connection: local

  tasks:
    - name: pre-check for ospf config
      iosxr_command:
        commands:
          - show run router ospf

      register: XR_OSPF

    - meta: end_play
      when: XR_OSPF.stdout | join('') | search('router ospf')

    - name: configure XR ospf
      iosxr_config:
        parents: "router ospf 1"
        lines:
          - "router-id 192.168.0.2"
          - "area 0"
          - "interface Loopback0"
          - "passive enable"
          - "exit"
          - "interface GigabitEthernet0/0/0/0"
          - "exit"

          cisco@ansible-controller:~$ ansible IOS -m raw -a "sho run | sec ospf"
          R1 | SUCCESS | rc=0 >>
          Connection to 172.16.101.91 closed by remote host.
          Shared connection to 172.16.101.91 closed.


          cisco@ansible-controller:~$ ansible XR -m raw -a "sho run router ospf"
          R2 | SUCCESS | rc=0 >>


          Thu Jun  7 22:35:08.331 UTC
          % No such configuration item(s)




          IMPORTANT:  READ CAREFULLY
          Welcome to the Demo Version of Cisco IOS XRv (the "Software").
          :


          Shared connection to 172.16.101.92 closed.
          Connection to 172.16.101.92 closed by remote host.


          cisco@ansible-controller:~$ ansible XR -m raw -a "show route ospf"
          R2 | SUCCESS | rc=0 >>


          Thu Jun  7 22:35:18.030 UTC

          % No matching routes found
          IMPORTANT:  READ CAREFULLY
          Welcome to the Demo Version of Cisco IOS XRv (the "Software").
          :


          Shared connection to 172.16.101.92 closed.
          Connection to 172.16.101.92 closed by remote host
:
cisco@ansible-controller:~$ ansible-playbook p33-ospf-config.yml

PLAY [configure ospf on IOS routers - play-1]
*****************************************************************************************
***********************************************

TASK [pre-check for ospf config]
*****************************************************************************************
************************************************************
ok: [R1]

TASK [configure IOS ospf]
*****************************************************************************************
*******************************************************************
changed: [R1]

PLAY [configure ospf on XR routers - play-2]
*****************************************************************************************
************************************************

TASK [pre-check for ospf config]
*****************************************************************************************
************************************************************
ok: [R2]

TASK [configure XR ospf]
*****************************************************************************************
********************************************************************
changed: [R2]

PLAY RECAP
*****************************************************************************************
**********************************************************************************
R1                         : ok=2    changed=1    unreachable=0    failed=0
R2                         : ok=2    changed=1    unreachable=0    failed=0

cisco@ansible-controller:~$

.
cisco@ansible-controller:~$ ansible IOS -m raw -a "show ip ospf neigh"
R1 | SUCCESS | rc=0 >>


Neighbor ID     Pri   State           Dead Time   Address         Interface
192.168.0.2       1   FULL/DR         00:00:33    10.0.0.6        GigabitEthernet2Shared
connection to 172.16.101.91 closed.
Connection to 172.16.101.91 closed by remote host.


cisco@ansible-controller:~$ ansible IOS -m raw -a "show ip route ospf"
```

```
R1 | SUCCESS | rc=0 >>

Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
       a - application route
       + - replicated route, % - next hop override, p - overrides from PfR

Gateway of last resort is not set

      192.168.0.0/32 is subnetted, 2 subnets
O        192.168.0.2 [110/2] via 10.0.0.6, 00:01:09, GigabitEthernet2Shared connection to
172.16.101.91 closed.
Connection to 172.16.101.91 closed by remote host.


cisco@ansible-controller:~$ ansible XR -m raw -a "show ospf neigh"
R2 | SUCCESS | rc=0 >>


Thu Jun  7 23:17:09.508 UTC

* Indicates MADJ interface
# Indicates Neighbor awaiting BFD session up

Neighbors for OSPF 1

Neighbor ID     Pri   State           Dead Time   Address         Interface
192.168.0.1     1     FULL/BDR        00:00:33    10.0.0.5        GigabitEthernet0/0/0/0
    Neighbor is up for 00:01:57

Total neighbor count: 1



IMPORTANT:  READ CAREFULLY
Welcome to the Demo Version of Cisco IOS XRv (the "Software").
:


Connection to 172.16.101.92 closed by remote host.
Shared connection to 172.16.101.92 closed.


cisco@ansible-controller:~$ ansible XR -m raw -a "show route ospf"
R2 | SUCCESS | rc=0 >>


Thu Jun  7 23:17:15.928 UTC

O    192.168.0.1/32 [110/2] via 10.0.0.5, 00:01:23, GigabitEthernet0/0/0/0
```

```
IMPORTANT:  READ CAREFULLY
Welcome to the Demo Version of Cisco IOS XRv (the "Software").
:


Shared connection to 172.16.101.92 closed.
Connection to 172.16.101.92 closed by remote host.


cisco@ansible-controller:~$ ansible-playbook p33-ospf-capture.yml --syntax-check

playbook: p33-ospf-capture.yml
cisco@ansible-controller:~$ ansible-playbook p33-ospf-capture.yml --tags POSTCHECK

PLAY [OSPF captures from IOS routers]
*****************************************************************************************
*******************************************************

TASK [Collect IOS OSPF commands]
*****************************************************************************************
************************************************************
ok: [R1]

TASK [Save IOS postcheck output to a file]
*****************************************************************************************
**************************************************
changed: [R1]

PLAY [OSPF captures from XR routers]
*****************************************************************************************
********************************************************

TASK [Collect XR OSPF commands]
*****************************************************************************************
*************************************************************
ok: [R2]

TASK [Save XR postcheck output to a file]
*****************************************************************************************
***************************************************
changed: [R2]

PLAY RECAP
*****************************************************************************************
**********************************************************************************
R1                          : ok=2    changed=1    unreachable=0    failed=0
R2                          : ok=2    changed=1    unreachable=0    failed=0

cisco@ansible-controller:~$ ls -l p33-postcheck*.txt
-rw-rw-r-- 1 cisco cisco 1395 Jun  7 23:15 p33-postcheck-ios-ospf.txt
-rw-rw-r-- 1 cisco cisco  969 Jun  7 23:15 p33-postcheck-xr-ospf.txt
cisco@ansible-controller:~$

cisco@ansible-controller:~$ ansible-playbook p33-mop.yml

PLAY [pre-config captures]
*****************************************************************************************
******************************************************************
```

```
TASK [run precheck playbook]
*****************************************************************************************
****************************************************************
changed: [localhost]

PLAY [configure ospf on IOS routers - play-1]
*****************************************************************************************
***********************************************

TASK [pre-check for ospf config]
*****************************************************************************************
************************************************************
ok: [R1]

PLAY [configure ospf on XR routers - play-2]
*****************************************************************************************
************************************************

TASK [pre-check for ospf config]
*****************************************************************************************
************************************************************
ok: [R2]

PLAY [post-config captures]
*****************************************************************************************
*****************************************************************

TASK [run postcheck playbook]
*****************************************************************************************
***************************************************************
changed: [localhost]

PLAY RECAP
*****************************************************************************************
**********************************************************************************
R1                           : ok=1    changed=0    unreachable=0    failed=0
R2                           : ok=1    changed=0    unreachable=0    failed=0
localhost                    : ok=2    changed=2    unreachable=0    failed=0
```

3.4 Generate iBGP Config

- Network configuration and rollouts are critical part of daily network operations.
- This exercise will simulate a network config generation via roles and rollout of configs to network elements.

Objectives

- Create a playbook to generate configuration based on Roles directory structures and Jinja2 Templates.
- The objective is to generate the below config:

```
IOS:
```

```
router bgp 1
 bgp router-id 192.168.0.1
 bgp log-neighbor-changes
 neighbor 192.168.0.2 remote-as 1
 neighbor 192.168.0.2 update-source Loopback0
!

XR:

router bgp 1
 bgp router-id 192.168.0.2
 address-family ipv4 unicast
 !
 neighbor 192.168.0.1
  remote-as 1
  update-source Loopback0
  address-family ipv4 unicast
  !
 !
!
```

Approach

- Step-1: Manually create the roles directory structure for the 2 router types (IOS & XR).
- Step-2: Create a playbook roles-bgy.yml to call the two roles (ios-bgp & xr-bgp) to generate the iBGP config and to upload the generated config to the routers.
- Step-3: Create main.yml files under the tasks and vars folders to setup the tasks to be executed when the roles are called.
- Step-4: Create the iBGP configuration template for csr and xr router using Jinja2 templating format.
- Step-5: Execute the playbook roles-bgp.yml to generate the iBGP config and apply it to both routers.

Lab Exercise

Step-1: Create two sub-folders by the name “ios-bgp” and “xr-bgp” using mkdir command

- Perform these steps from your home directory (or the directory where you are going to save the playbook).

```
$ mkdir ios-bgp xr-bgp
```

Step-2: Create tasks, vars, and templates directories under both ios-bgp and xr-bgp folder.

```
$ mkdir ios-bgp/tasks ios-bgp/vars ios-bgp/templates

$ mkdir xr-bgp/tasks xr-bgp/vars xr-bgp/templates
```

Step-3: Create a playbook called roles-bgp.yml.

- This playbook contains 3 plays:
  - First play will run on the localhost (ansible-controller) and call on the roles to generate IOS and XR iBGP configs.
  - The second and third plays are used to upload the BGP configs to the routers.

```
---
- name: Generate router bgp configuration files using Roles and Jinja2 Templates
  hosts: localhost

  roles:
   - xr-bgp
   - ios-bgp

- name: Task to upload config to CSR
  hosts: IOS
  gather_facts: false
  connection: local
  tasks:
  - name: "Load iBGP configs for CSR1kv router using SRC option using IOS_CONFIG Module"
    ios_config:
      src: "/home/cisco/R1-IOS-BGP.txt"

- name: Task to upload config to R2-XRV
  gather_facts: false
  connection: local
  hosts: XR

  tasks:
  - name: "Load iBGP configs for xr router using SRC option of IOSXR_CONFIG Module "
    iosxr_config:
      src: "/home/cisco/R2-XRv-BGP.txt"
```

Step-4: Create a main.yml file inside the ios-bgp/tasks folder.

- The main.yml file in the tasks sub-folder identifies the Jinja2 Template that contains the iBGP configuration template, specified under the template folder, for this role.

- Note vi users do the following to create the file.

```
$ vi ios-bgp/tasks/main.yml
```

- Note ATOM users do the following once the file has been created on the home directory.

```
$ mv main.yml ios-bgp/tasks/main.yml
```

- Paste the following inside the tasks main.yml file.

```
---
```

```
- name: Generate R1 CSR router iBGP config file
  template: src=IOS-BGP.j2 dest=./{{item.hostname}}-BGP.txt
  with_items: "{{router_list}}"
```

Step-5: Create a main.yml file inside the ios-bgp/vars folder.

- Vars file contains all the values for parameters given in the Jinj2 template file.
- Note vi users do the following to create the file.

```
$ vi ios-bgp/vars/main.yml
```

- Note ATOM users do the following once the file has been created on the home directory.

```
$ mv main.yml ios-bgp/vars/main.yml
```

- Paste the following inside the vars main.yml file.

```
---
router_list:
  -  hostname: R1-IOS
     RID: 192.168.0.1
     PEER_IP: 192.168.0.2
     LCL_ASN: 1
     RMT_ASN: 1
```

Step-6: Create a IOS-BGP.j2 template file inside the ios-bgp/templates folder.

- Note vi users do the following to create the file.

```
$ vi ios-bgp/templates/IOS-BGP.j2
```

- Note ATOM users do the following once the file has been created on the home directory.

```
$ mv IOS-BGP.j2 ios-bgp/templates/IOS-BGP.j2
```

- Paste the following inside the templates IOS-BGP.j2 file.

```
router bgp {{item.LCL_ASN}}
 bgp router-id {{item.RID}}
 bgp log-neighbor-changes
 neighbor {{item.PEER_IP}} remote-as {{item.RMT_ASN}}
 neighbor {{item.PEER_IP}} update-source Loopback0
!
```

Step-7: Same steps will need to be repeated for the xr-bgp role. Create a main.yml file inside the xr-bgp/tasks folder.

- Note vi users do the following to create the file.

```
$ vi xr-bgp/tasks/main.yml
```

- Note ATOM users do the following once the file has been created on the home directory.

```
$ mv main.yml xr-bgp/tasks/main.yml
```

- Paste the following inside the tasks main.yml file.

```
---
- name: Generate R2 XRV router iBGP config file
  template: src=XR-BGP.j2 dest=./{{item.hostname}}-BGP.txt
  with_items: "{{router_list}}"
```

Step-8: Create a main.yml file inside the xr-bgp/vars folder.

- Note vi users do the following to create the file.

```
$ vi xr-bgp/vars/main.yml
```

- Note ATOM users do the following once the file has been created on the home directory.

```
$ mv main.yml xr-bgp/vars/main.yml
```

- Paste the following inside the vars main.yml file.

```
router_list:
  -  hostname: R2-XRv
     RID: 192.168.0.2
     PEER_IP: 192.168.0.1
     LCL_ASN: 1
     RMT_ASN: 1
```

Step-9: Create a XR-BGP.j2 file inside the xr-bgp/templates folder.

- Note vi users do the following to create the file.

```
$ vi xr-bgp/templates/XR-BGP.j2
```

- Note ATOM users do the following once the file has been created on the home directory.

```
$ mv XR-BGP.j2 xr-bgp/templates/XR-BGP.j2
```

- Paste the following inside the templates XR-BGP.j2 file.

```
router bgp {{item.LCL_ASN}}
 bgp router-id {{item.RID}}
 address-family ipv4 unicast
 !
 neighbor {{item.PEER_IP}}
  remote-as {{item.RMT_ASN}}
  update-source Loopback0
  address-family ipv4 unicast
   !
 !
!
```

Step-10: Run the tree command and validate that following files structure is created

```
$ tree ios-bgp
ios-bgp
├── tasks
│   └── main.yml
├── templates
│   └── CSR-BGP.J2
└── vars
    └── main.yml

3 directories, 3 files

$ tree xr-bgp
xr-bgp
├── tasks
│   └── main.yml
├── templates
│   └── XR-BGP.J2
└── vars
    └── main.yml
```

Step-11: Execute the roles-bgp.yml playbook.

```
$ ansible-playbook roles-bgp.yml
```

Step-12: After the playbook is run, there should be 2 files generated on the current working directory (R1-IOS-BGP.txt & R2-XRv-BGP.txt).

```
$ more cfg/R1-IOS-BGP.txt

router bgp 1
 bgp router-id 192.168.0.1
 bgp log-neighbor-changes
 neighbor 192.168.0.2 remote-as 1
 neighbor 192.168.0.2 update-source Loopback0
!

$ more cfg/R2-XRv-BGP.txt

router bgp 1
 bgp router-id 192.168.0.2
 address-family ipv4 unicast
 !
 neighbor 192.168.0.1
  remote-as 1
  update-source Loopback0
  address-family ipv4 unicast
  !
 !
!
```

- Make sure iBGP was configure on both routers.

```
$ ansible IOS -m raw -a "show run | sec bgp"

$ ansible XR -m raw -a "show run router bgp"

$ ansible all -m raw -a "show bgp sum"
```

Conclusion

- Ansible roles can be utilized to organize large playbooks.
- Roles creates a separation of functions: variables, tasks, and templates in unique directories.
- Ansible roles expect a main.yml file under the sub-directory.

Example output:

```
cisco@ansible-controller:~/CLUS18-Lab$ ansible-playbook roles-bgp.yml

PLAY [Generate router bgp configuration files using Roles and Jinja2 Templates]
*****************************************************************

TASK [xr-bgp : Generate R2 XRV router iBGP config file]
*****************************************************************************************
*
ok: [localhost] => (item={u'profile': u'IOS-XR', u'LCL_ASN': 1, u'RMT_ASN': 1,
u'hostname': u'R2-XRv', u'PEER_IP': u'192.168.0.1', u'RID': u'192.168.0.2'})
```

```
TASK [ios-bgp : Generate R1 CSR router iBGP config file]
*****************************************************************************************
ok: [localhost] => (item={u'profile': u'IOS-XE', u'LCL_ASN': 1, u'RMT_ASN': 1,
u'hostname': u'R1-IOS', u'PEER_IP': u'192.168.0.2', u'RID': u'192.168.0.1'})

PLAY [Task to upload config to CSR]
*****************************************************************************************
*********************

TASK [Load iBGP configs for CSR1kv router using SRC option using IOS_CONFIG Module]
**************************************************************

ok: [R1]

PLAY [Task to upload config to R2-XRV]
*****************************************************************************************
******************

TASK [Load iBGP configs for xr router using SRC option of IOSXR_CONFIG Module]
*******************************************************************

ok: [R2]

PLAY RECAP
*****************************************************************************************
**********************************************
R1                           : ok=1    changed=0    unreachable=0    failed=0
R2                           : ok=1    changed=0    unreachable=0    failed=0
localhost                    : ok=2    changed=0    unreachable=0    failed=0
```

3.5 Bulk Config Generation

- By Leveraging Ansible Roles and templates, users can build bulk configurations for deployment at scale.

Objective

- Build configurations for 2 different type of router OS – IOS and XR.

Approach

- Initialize the roles directory and file structure by using ansible-galaxy cli
- Build the playbook, template and variables for bulk config generation

Lab Exercise

Step-1: Create a new role called config-gen by using the ansible-galaxy utility.

- Note: Ansible-Galaxy command is built into Ansible and allows for an automated way to create the Roles directory structure.

```
cisco@ansible-controller:~$ ansible-galaxy init config-gen
- config-gen was created successfully
```

Step-2: Review the tree structure that has been created by galaxy. For this lab, you will be using the templates, vars, and tasks sub-directories to generate a bulk configuration.

```
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
                8 directories, 8 files
                cisco@ansible-controller:~$
```

Step-3: Edit the main.yml file under the config-gen/tasks sub-directory.

- Note vi users do the following to create the file.

```
$ vi config-gen/tasks/main.yml
```

- Note ATOM users do the following once the file has been created on the home directory.

```
$ mv main.yml config-gen/tasks/main.yml
```

- Paste the following inside the tasks main.yml file.

```
---
- name: Generate the configuration for xr-routers
  template:
     src=xr-config-template.j2
     dest=./{{item.hostname}}.txt
  with_items:
     - "{{ xr_hostnames }}"

- name: Generate the configuration for iosxe-routers
```

```
    template:
       src=ios-config-template.j2
       dest=./{{item.hostname}}.txt
    with_items:
       - "{{ ios_hostnames }}"
  ...
  # tasks file for config-gen
```

Step-4: Create the platform specific configuration and save them under the config-gen/templates folder.

- 4a. Create the following template “xr-config-template.j2” under the templates directory.
    - Note vi users do the following to create the file.

```
$ vi config-gen/templates/xr-config-template.j2
```

- Note ATOM users do the following once the file has been created on the home directory.

```
$ mv xr-config-template.j2 config-gen/templates/xr-config-template.j2
```

- Paste the following inside the templates xr-config-template.j2 file.

```
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
telnet vrf default ipv4 server max-servers 10
telnet vrf Mgmt-intf ipv4 server max-servers 10
domain lookup disable
vrf Mgmt-intf
 address-family ipv4 unicast
 !
 address-family ipv6 unicast
 !
!
domain name virl.info
ssh server v2
ssh server vrf Mgmt-intf
!
line template vty
timestamp
exec-timeout 720 0
!
line console
exec-timeout 0 0
!
line default
exec-timeout 720 0
!
vty-pool default 0 50
```

```
control-plane
 management-plane
  inband
   interface all
    allow all
   !
  !
 !
!
!
cdp
!
!
interface Loopback0
  description Loopback
  ipv4 address {{item.loopback0_ip}} {{item.loopback0_mask}}
!
interface GigabitEthernet0/0/0/0
  description to R1-CSR1kv
  ipv4 address {{item.gig0000_ip}} {{item.gig0000_mask}}
  cdp
  no shutdown
!
interface GigabitEthernet0/0/0/1
  description to R3-NXOS
  ipv4 address {{item.gig0001_ip}} {{item.gig0001_mask}}
  cdp
  no shutdown
!
interface mgmteth0/0/CPU0/0
  description OOB Management
  ! Configured on launch
  vrf Mgmt-intf
 ipv4 address 172.16.101.99 255.255.255.0
  cdp
  no shutdown
!
!
router ospf 16509
  log adjacency changes
  router-id {{item.loopback0_ip}}
  address-family ipv4
  area 0
    !
    interface Loopback0
      passive enable
    !
{% for interface in xr_interfaces %}
interface {{interface}}
cost 1
!
{% endfor %}
  !
!
!
```

- 4b. Create the following template “ios-config-template.j2” under the templates directory.
    - Note vi users do the following to create the file.

```
$ vi config-gen/templates/ios-config-template.j2
```

- Note ATOM users do the following once the file has been created on the home directory.

```
$ mv ios-config-template.j2 config-gen/templates/ios-config-template.j2
```

- Paste the following inside the templates ios-config-template.j2 file.

```

clock timezone {{item.timezone}} {{item.timezone_offset}}
clock summer-time {{item.timezone_dst}} recurring

service timestamps debug datetime msec
service timestamps log datetime msec
platform qfp utilization monitor load 80
no platform punt-keepalive disable-kernel-core
platform console serial
!
hostname {{item.hostname}}
!
boot-start-marker
boot-end-marker
!
!
vrf definition Mgmt-intf
 !
 address-family ipv4
 exit-address-family
 !
 address-family ipv6
 exit-address-family
!
enable secret 4 tnhtc92DXBhelxjYk8LWJrPV36S2i4ntXrpb4RFmfqY
enable password cisco
!
no aaa new-model
!
!
!
!
!
!
!
!

no ip domain lookup
ip domain name virl.info
!
```

```
!
!
ipv6 unicast-routing
!
!
!
!
!
!
!
subscriber templating
!
!
!
!
!
!
!
multilink bundle-name authenticated
!
!
!
!
!
crypto pki trustpoint TP-self-signed-35466579
 enrollment selfsigned
 subject-name cn=IOS-Self-Signed-Certificate-35466579
 revocation-check none
 rsakeypair TP-self-signed-35466579
!
!
crypto pki certificate chain TP-self-signed-35466579
 certificate self-signed 01
  3082032C 30820214 A0030201 02020101 300D0609 2A864886 F70D0101 05050030
  2F312D30 2B060355 04031324 494F532D 53656C66 2D536967 6E65642D 43657274
  69666963 6174652D 33353436 36353739 301E170D 31383033 32383136 33343532
  5A170D32 30303130 31303030 3030305A 302F312D 302B0603 55040313 24494F53
  2D53656C 662D5369 676E6564 2D436572 74696669 63617465 2D333534 36363537
  39308201 22300D06 092A8648 86F70D01 01010500 0382010F 00308201 0A028201
  0100C122 3C95D116 714EE581 53539DCE 33BBE636 20BCAB70 B12ECDE8 832DB71C
  F223B066 E3779F87 0BF81EE6 CE6E60EE F471B22F 5ECE57FD 50C7D706 17F3F62D
  4573882F B9B6351F ECDC6192 167D768A DC8B4613 8A2AEB70 1906E49D 0A2734A8
  64C0C7A3 4B6951D2 573AFF96 5682BE7D 305F4351 A5E6A667 DB787283 724AF55F
  3A049F98 57A1C34F CC9B9C24 3056B3DF 11A04AB4 3F051C0D 14D5AACE B7B0D991
  611FE0D6 6B2CC9D2 3F410224 52701D25 135C7BF2 FEDC0BCD F9BD7C10 4B437143
  E38A10E8 F5423F0E BB71A593 AFDBC814 D6DD4ED6 0709FCC5 33F480F0 6389C2AF
  F0C36163 54164A20 541AAA30 EAFDFD2B 35361640 82331C9B F0D97302 B1429508
  87DB0203 010001A3 53305130 0F060355 1D130101 FF040530 030101FF 301F0603
  551D2304 18301680 14B0C21C 0050185E 5D0751E6 6A90DD48 D9157E6B 0E301D06
  03551D0E 04160414 B0C21C00 50185E5D 0751E66A 90DD48D9 157E6B0E 300D0609
  2A864886 F70D0101 05050003 82010100 4E89908F 13A8518B 33D0DC0E 71548510
  7E3285F7 71E4B8A4 2E25FA83 3FD571F5 17D190EA DFC4F076 AF1C3494 17DC54B4
  93A61630 C2D321BE F3D1B9D1 72AA7BE9 D5755FD5 C2330B82 F9DA1B4B 590BBA8A
  0A36758E 22061021 86D03C8B D5877680 954F22E6 3A4F807E 79CA5DB5 F63ECF74
  CA45C80D A8052A3A 48CD69B5 027D66D5 08020FA6 94FCE404 07D12573 590C0D60
  5999C40B FECA7B2D A11FC2B8 21D7A110 E4814E8E 2ED74D9D B22A66DF B9BF8932
  424A5807 AF9A59B5 FB6A7FCE B73E25B8 F937695D 9E15768D 614AA387 0B26B6FA
  C54DF6E2 34E5E803 1123AB24 9CC8F3CF FDBB6B7E CC3FF86C B83C858A 34646F0B
```

```
    0C79ED3D 814ACA2F 3F565B5C BB84FCAA
          quit
  !
  !
  !
  !
  !
  !
  !
  !
  license udi pid CSR1000V sn 9N7CZX65NJ3
  license accept end user agreement
  license boot level ax
  diagnostic bootup level minimal
  !
  spanning-tree extend system-id
  !
  !
  username cisco privilege 15 secret 5 $1$F6GC$L/.gqoiPm0AcItLajjXXJ/
  !
  redundancy
  !
  !
  cdp run
  !

  !
  interface Loopback0
   description Loopback
   ip address {{item.loopback0_ip}} {{item.loopback0_mask}}
  !
  interface GigabitEthernet1
   description OOB Management
   vrf forwarding Mgmt-intf
   ip address {{item.Mgmt_ip}} {{item.Mgmt_mask}}
   negotiation auto
   cdp enable
   no mop enabled
   no mop sysid
  !
  interface GigabitEthernet2
   description to R2-XRv
   ip address {{item.gigaethernet2_ip}} {{item.gigaethernet2_mask}}
   ip ospf cost 1
   negotiation auto
   cdp enable
   no mop enabled
   no mop sysid
  !
  !
  router ospf 16509
    network {{item.ospf_network1}} {{item.ospf_network_mask1}} area {{item.areaid}}
    log-adjacency-changes
    passive-interface Loopback0
    network {{item.ospf_network2}} {{item.ospf_network_mask1}} area {{item.areaid}}
    network {{item.ospf_network3}} {{item.ospf_network_mask1}} area {{item.areaid}}
  !
  !
```

```
threat-visibility
!
virtual-service csr_mgmt
!
ip forward-protocol nd
ip http server
ip http authentication local
ip http secure-server
!
ip ssh server algorithm encryption aes128-ctr aes192-ctr aes256-ctr
ip ssh server algorithm authentication password
ip ssh client algorithm encryption aes128-ctr aes192-ctr aes256-ctr
!
!
control-plane
!
!
line con 0
 password cisco
 stopbits 1
line vty 0 4
 exec-timeout 720 0
 password cisco
 login local
 transport input telnet ssh
!
end
```

Step-5: Define the variable needed to generate the config in the main.yml file under the config-gen/vars sub-directory.

- Note vi users do the following to create the file.

```
$ vi config-gen/vars/main.yml
```

- Note ATOM users do the following once the file has been created on the home directory.

```
$ mv main.yml config-gen/vars/main.yml
```

- Paste the following inside the vars main.yml file.

```
---
xr_hostnames:
   - { hostname: xr-router-rtr1, timezone: EST, timezone_dst: EDT, timezone_offset: -5,
loopback0_ip: 192.168.0.1, loopback0_mask: 255.255.255.255
, gig0000_ip: 10.0.0.5, gig0000_mask: 255.255.255.0, gig0001_ip: 10.1.0.5 , gig0001_mask:
255.255.255.0,}

xr_interfaces:
  - GigabitEthernet0/0/0/0
  - GigabitEthernet0/0/0/1


ios_hostnames:
   - { hostname: ios-router-rtr1, timezone: EST, timezone_dst: EDT, timezone_offset: -5,
loopback0_ip: 192.168.0.2, loopback0_mask: 255.255.255.25
5, Mgmt_ip: 172.16.101.98, Mgmt_mask: 255.255.255.0, gigaethernet2_ip: 10.0.0.6,
gigaethernet2_mask: 255.255.255.0, ospf_network1: 192.168.0.2, os
pf_network_mask1: 0.0.0.0, ospf_network2: 10.0.0.0, ospf_network_mask2: 0.0.0.255,
ospf_network3: 10.1.0.0, ospf_network_mask3: 0.0.0.255, areaid:
 0 }

...
# vars file for config-gen
```

Step-6: Create a playbook config-gen.yml to invoke the role of Bulk config generation. The config-gen.yml playbook has to be in the same directory as the config-gen role.

```
---
  - name: Playbook to generate configuration based on role "config-gen"
    hosts: localhost

    roles:
       - config-gen
```

Step-7: Execute the playbook config-gen.yml. You will see the config files are generated in target location.

```
cisco@ansible-controller:~$ ansible-playbook config-gen.yml
```

Step-8: Validate the files are created

```
cisco@ansible-controller:~$ ls -ltr *BGP.txt
-rw-r--r-- 1 cisco cisco 168 Jun  5 02:40 R2-XRv-BGP.txt
-rw-r--r-- 1 cisco cisco 148 Jun  5 02:41 R1-IOS-BGP.txt
cisco@ansible-controller:~$
```

Conclusion

- You can utilize the concept of roles - predetermined order of directories and files to automate generating bulk tasks.

Example Output

```
cisco@ansible-controller:~/CLUS18-Lab$ ansible-playbook config-gen.yml
```

```
PLAY [Playbook to generate configuration based on role "config-gen"]
*****************************************************************************

TASK [config-gen : Generate the configuration for xr-routers]
************************************************************************************
ok: [localhost] => (item={u'timezone_dst': u'EDT', u'gig0000_mask': u'255.255.255.0',
u'timezone_offset': -5, u'hostname': u'xr-router-rtr1', u'loopback0_ip': u'192.168.0.1',
u'timezone': u'EST', u'gig0001_mask': u'255.255.255.0', u'gig0000_ip': u'10.0.0.5',
u'gig0001_ip': u'10.1.0.5', u'loopback0_mask': u'255.255.255.255'})

TASK [config-gen : Generate the configuration for iosxe-routers]
*********************************************************************************
ok: [localhost] => (item={u'timezone_dst': u'EDT', u'areaid': 0, u'ospf_network3':
u'10.1.0.0', u'Mgmt_ip': u'172.16.101.98', u'gigaethernet2_mask': u'255.255.255.0',
u'ospf_network1': u'192.168.0.2', u'timezone_offset': -5, u'hostname': u'ios-router-
rtr1', u'ospf_network_mask1': u'0.0.0.0', u'ospf_network_mask2': u'0.0.0.255',
u'loopback0_ip': u'192.168.0.2', u'Mgmt_mask': u'255.255.255.0', u'timezone': u'EST',
u'ospf_network_mask3': u'0.0.0.255', u'gigaethernet2_ip': u'10.0.0.6', u'ospf_network2':
u'10.0.0.0', u'loopback0_mask': u'255.255.255.255'})

PLAY RECAP
*****************************************************************************************
**********************************************
localhost                  : ok=2    changed=0    unreachable=0    failed=0
```

4. Appendix

4.1 Ansible Vault

- Ansible Vault is a feature of Ansible that allows you to keep sensitive data such as passwords or keys in encrypted files, rather than as plaintext in playbooks or roles.
- Ansible vault has many security features but this section will cover only the basics of encrypting and decrypting a file.

Objective

- Encrypt the inventory (/etc/hosts) file using Ansible Vault, so router login data is not stored in plain text.

Lab exercise

- Use Ansible vault to encrypt and decrypt the inventory file.

Pre-check

- Pay attention the owner and file privileges (`-rw-r--r--`).
- Make sure the playbook runs without any errors

```
$ ls -l /etc/ansible/hosts

$ cat /etc/ansible/hosts

$ ansible-playbook p2-ioscmd.yml
```

Encrypt inventory file and execute a playbook

- Steps summary
  - Encrypt the inventory file using `ansible-vault` command.
    - sudo password: `cisco`
    - New vault password: `cisco123`
  - Read the encrypted file
  - Playbook execution without vault-key will fail.
  - Read the failure error message about inventory file
  - Execute the playbook with vault-key
  - supply sudo and vault passwords as prompted.
  - The playbook should run fine now.

```
$ ansible-vault --help

$ sudo ansible-vault encrypt /etc/ansible/hosts

$ sudo cat /etc/ansible/hosts

$ sudo ansible-vault view /etc/ansible/hosts

$ sudo ansible-playbook p2-ioscmd.yml

$ sudo ansible-playbook p2-ioscmd.yml --ask-vault-pass
```

Decrypt and restore

- Steps summary
  - Pay attention the owner and file privileges. Notice that the file permissions are not changed back to the original values.
  - Change the file permissions to the previous settings.
  - Make sure that the file permissions are: `-rw-r--r--`
  - Ensure the playbook runs successfully before proceeding to next section.

```
$ sudo ansible-vault decrypt /etc/ansible/hosts

$ ls -l /etc/ansible/hosts

$ sudo chmod 644 /etc/ansible/hosts
$ ls -l /etc/ansible/hosts

$ cat /etc/ansible/hosts

$ ansible-playbook p2-ioscmd.yml
```

Conclusion

- Ansible files can be encrypted and decrypted using Vault.
- It is possible to use the encrypted files in playbooks using the option --ask-vault-pass -Note: When vault encrypted the file it changed the file permissions; but when it decrypted the file it did not restore the old permissions.
- Review the section and discuss if you have any questions.

Example output

```
cisco@ansible-controller:~$ ls -l /etc/ansible/hosts
-rw-r--r-- 1 root root 1333 May 31 15:03 /etc/ansible/hosts
cisco@ansible-controller:~$ cat /etc/ansible/hosts
# This is the default ansible 'hosts' file.
#
# It should live in /etc/ansible/hosts
#
#   - Comments begin with the '#' character
#   - Blank lines are ignored
#   - Groups of hosts are delimited by [header] elements
#   - You can enter hostnames or ip addresses
#   - A hostname/ip can be a member of multiple groups

[IOS]
172.16.101.91

[XR]
172.16.101.92

[ALL:children]
IOS
XR

[ALL:vars]
ansible_user=cisco
ansible_ssh_pass=cisco
:
cisco@ansible-controller:~$ ansible-playbook p2-ioscmd.yml

PLAY [collect ip route summary from all IOS devices]
*************************************************

TASK [execute route summary command]
*****************************************************************
ok: [172.16.101.91]

TASK [print output]
```

```
**********************************************************************************
ok: [172.16.101.91] => {
    "P2_OUTPUT.stdout_lines": [
        [
            "IP routing table name is default (0x0)",
            "IP routing table maximum-paths is 32",
            "Route Source    Networks    Subnets     Replicates  Overhead    Memory
(bytes)",
            "application     0           0           0           0           0",
            "connected       0           4           0           384         1216",
            "static          0           0           0           0           0",
            "ospf 1          0           1           0           96          308",
            "  Intra-area: 1 Inter-area: 0 External-1: 0 External-2: 0",
            "  NSSA External-1: 0 NSSA External-2: 0",
            "bgp 1           0           0           0           0           0",
            "  External: 0 Internal: 0 Local: 0",
            "internal        3                                               1432",
            "Total           3           5           0           480         2956"
        ]
    ]
}

PLAY RECAP
*****************************************************************************************
**
172.16.101.91              : ok=2    changed=0    unreachable=0    failed=0

cisco@ansible-controller:~$
cisco@ansible-controller:~$ ansible-vault --help
Usage: ansible-vault [create|decrypt|edit|encrypt|encrypt_string|rekey|view] [options]
[vaultfile.yml]

:
cisco@ansible-controller:~$ sudo ansible-vault encrypt /etc/ansible/hosts
[sudo] password for cisco:
New Vault password:
Confirm New Vault password:
Encryption successful
cisco@ansible-controller:~$ sudo cat /etc/ansible/hosts
'$ANSIBLE_VAULT;1.1;AES256
66366161326466633561633433343264646366646330633430313061663639333836386531313936
35623333383039366337333966626131663234393936393
90a336661333532646534373138363663
30386630383365386639633461366332363939373465386132373930653235353466333032633663
32623635653435396
50a613264636664336339313532363938393766333130616364336133386531
:
cisco@ansible-controller:~$
cisco@ansible-controller:~$ sudo ansible-vault view /etc/ansible/hosts
Vault password:
# This is the default ansible 'hosts' file.
#
# It should live in /etc/ansible/hosts
#
#   - Comments begin with the '#' character
#   - Blank lines are ignored
#   - Groups of hosts are delimited by [header] elements
#   - You can enter hostnames or ip addresses
#   - A hostname/ip can be a member of multiple groups
```

```
[IOS]
172.16.101.91

[XR]
172.16.101.92

[ALL:children]
IOS
XR

[ALL:vars]
ansible_user=cisco
ansible_ssh_pass=cisco
:
cisco@ansible-controller:~$ sudo ansible-playbook p2-ioscmd.yml
[sudo] password for cisco:
 [WARNING]:  * Failed to parse /etc/ansible/hosts with yaml plugin: Attempting to decrypt
but no
vault secrets found

 [WARNING]:  * Failed to parse /etc/ansible/hosts with ini plugin: Attempting to decrypt
but no vault
secrets found

 [WARNING]: Unable to parse /etc/ansible/hosts as an inventory source

 [WARNING]: No inventory was parsed, only implicit localhost is available

 [WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit
localhost does not match 'all'

 [WARNING]: Could not match supplied host pattern, ignoring: IOS


PLAY [collect ip route summary from all IOS devices]
*************************************************
skipping: no hosts matched

PLAY RECAP
*****************************************************************************************
**

cisco@ansible-controller:~$
cisco@ansible-controller:~$ sudo ansible-playbook p2-ioscmd.yml --ask-vault-pass
Vault password:

PLAY [collect ip route summary from all IOS devices]
*************************************************

TASK [execute route summary command]
*****************************************************************
ok: [172.16.101.91]

TASK [print output]
**********************************************************************************
ok: [172.16.101.91] => {
    "P2_OUTPUT.stdout_lines": [
        [
            "IP routing table name is default (0x0)",
            "IP routing table maximum-paths is 32",
            "Route Source    Networks    Subnets     Replicates  Overhead    Memory
(bytes)",
            "application     0           0           0           0           0",
            "connected       0           4           0           384         1216",
            "static          0           0           0           0           0",
            "ospf 1          0           1           0           96          308",
            "  Intra-area: 1 Inter-area: 0 External-1: 0 External-2: 0",
            "  NSSA External-1: 0 NSSA External-2: 0",
            "bgp 1           0           0           0           0           0",
            "  External: 0 Internal: 0 Local: 0",
            "internal        3                                               1432",
            "Total           3           5           0           480         2956"
        ]
    ]
}

PLAY RECAP
*************************************************************************************
**
172.16.101.91              : ok=2    changed=0    unreachable=0    failed=0

cisco@ansible-controller:~$
cisco@ansible-controller:~$ sudo ansible-playbook p2-ioscmd.yml
[sudo] password for cisco:
 [WARNING]:  * Failed to parse /etc/ansible/hosts with yaml plugin: Attempting to decrypt
but no
vault secrets found

 [WARNING]:  * Failed to parse /etc/ansible/hosts with ini plugin: Attempting to decrypt
but no vault
secrets found

 [WARNING]: Unable to parse /etc/ansible/hosts as an inventory source

 [WARNING]: No inventory was parsed, only implicit localhost is available

 [WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit
localhost does not match 'all'

 [WARNING]: Could not match supplied host pattern, ignoring: IOS


PLAY [collect ip route summary from all IOS devices]
*************************************************
skipping: no hosts matched

PLAY RECAP
*************************************************************************************
**

cisco@ansible-controller:~$
cisco@ansible-controller:~$ ls -l /etc/ansible/hosts
-rw------- 1 root root 1333 May 31 17:26 /etc/ansible/hosts
cisco@ansible-controller:~$
```

```
cisco@ansible-controller:~$ sudo chmod 644 /etc/ansible/hosts
cisco@ansible-controller:~$ ls -l /etc/ansible/hosts
-rw-r--r-- 1 root root 1333 May 31 17:26 /etc/ansible/hosts
cisco@ansible-controller:~$ cat /etc/ansible/hosts
# This is the default ansible 'hosts' file.
#
# It should live in /etc/ansible/hosts
#
#   - Comments begin with the '#' character
#   - Blank lines are ignored
#   - Groups of hosts are delimited by [header] elements
#   - You can enter hostnames or ip addresses
#   - A hostname/ip can be a member of multiple groups

[IOS]
172.16.101.91

[XR]
172.16.101.92

[ALL:children]
IOS
XR

[ALL:vars]
ansible_user=cisco
ansible_ssh_pass=cisco
:
cisco@ansible-controller:~$ ansible-playbook p2-ioscmd.yml


PLAY [collect ip route summary from all IOS devices]
*************************************************

TASK [execute route summary command]
*****************************************************************
ok: [172.16.101.91]

TASK [print output]
**********************************************************************************
ok: [172.16.101.91] => {
    "P2_OUTPUT.stdout_lines": [
        [
            "IP routing table name is default (0x0)",
            "IP routing table maximum-paths is 32",
            "Route Source    Networks    Subnets     Replicates  Overhead    Memory
(bytes)",
            "application     0           0           0           0           0",
            "connected       0           4           0           384         1216",
            "static          0           0           0           0           0",
            "ospf 1          0           1           0           96          308",
            "  Intra-area: 1 Inter-area: 0 External-1: 0 External-2: 0",
            "  NSSA External-1: 0 NSSA External-2: 0",
            "bgp 1           0           0           0           0           0",
            "  External: 0 Internal: 0 Local: 0",
            "internal        3                                               1432",
            "Total           3           5           0           480         2956"
        ]
    ]
}

PLAY RECAP
*********************************************************************************************
**
172.16.101.91              : ok=2    changed=0    unreachable=0    failed=0

cisco@ansible-controller:~$
```

4.2 Optional exercise op3-cmd.yml

Objective

- Write a playbook, op3-cmd.yml, to meet the below requirements:
  - Collect output of route summary from both IOS and XR routers
  - Use the modules, ios_command and iosxr_command
  - Write 2 plays within one playbook.

Lab exercise

- playbook op3-cmd.xml

```
---
  - name: play-1:get route summary from IOS routers
    hosts: IOS
    connection: local

    tasks:
      - name: execute IOS route summary command
        ios_command:
          commands:
            show ip route summary

        register: IOS_OUTPUT

      - name: print output
        debug:
          var: IOS_OUTPUT.stdout_lines

  - name: play-2:get route summary from XR routers
    hosts: XR
    connection: local

    tasks:
      - name: execute XR route summary command
        iosxr_command:
          commands:
            show route summary
        register: XR_OUTPUT

      - name: print output
        debug:
          var: XR_OUTPUT.stdout_lines
```

Example output

```
cisco@ansible-controller:~$ ansible-playbook op3-cmd.yml

PLAY [play-1:get route summary from IOS routers]
*******************************************************************

TASK [execute IOS route summary command]
***************************************************************************
ok: [R1]

TASK [print output]
*****************************************************************************************
*******
ok: [R1] => {
    "IOS_OUTPUT.stdout_lines": [
        [
            "IP routing table name is default (0x0)",
            "IP routing table maximum-paths is 32",
            "Route Source    Networks    Subnets     Replicates  Overhead    Memory
(bytes)",
            "application     0           0           0           0           0",
            "connected       0           4           0           384         1216",
            "static          0           0           0           0           0",
            "ospf 1          0           1           0           96          308",
            "  Intra-area: 1 Inter-area: 0 External-1: 0 External-2: 0",
            "  NSSA External-1: 0 NSSA External-2: 0",
            "bgp 1           0           0           0           0           0",
            "  External: 0 Internal: 0 Local: 0",
            "internal        3                                               1432",
            "Total           3           5           0           480         2956"
        ]
    ]
}

PLAY [play-2:get route summary from XR routers]
********************************************************************

TASK [execute XR route summary command]
****************************************************************************
ok: [R2]

TASK [print output]
*****************************************************************************************
*******
ok: [R2] => {
    "XR_OUTPUT.stdout_lines": [
        [
            "Route Source                      Routes     Backup     Deleted
Memory(bytes)",
            "local                             5          0          0            800
",
            "connected                         1          4          0            800
",
            "dagr                              0          0          0            0
",
            "ospf 1                            1          0          0            160
",
            "bgp 1                             0          0          0            0
",
            "Total                             7          4          0            1760"
        ]
    ]
}

PLAY RECAP
*****************************************************************************************
****************
R1                         : ok=2    changed=0    unreachable=0    failed=0
R2                         : ok=2    changed=0    unreachable=0    failed=0

```

4.3 Optional exercise op6-vars.yml

Objective

- Create a playbook to print messages, "This is " and "This is ".
- Use default variable, "inventory_hostname". The messages should look like: "This is R1" and "This is R2"

Lab exercise

- playbook, op6-vars.yml

```
---
- name: print hostnames of all devices
  hosts: ALL
  connection: local

  tasks:
    - name: print hostname
      debug:
        msg: "This is {{ inventory_hostname }}"
```

Example output

```
cisco@ansible-controller:~$ ansible-playbook op6-vars.yml --syntax-check

playbook: op6-vars.yml
cisco@ansible-controller:~$ ansible-playbook op6-vars.yml

PLAY [print hostnames of all devices]
*****************************************************************************

TASK [print hostname]
****************************************************************************************
*****
ok: [R1] => {
    "msg": "This is R1"
}
ok: [R2] => {
    "msg": "This is R2"
}

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

cisco@ansible-controller:~$
```

4.4 Optional exercise op8-conditionals.yml

Objective

- Create a playbook, which will:
  - Detect router OS
  - If a router has IOS, print message, "'hostname' is a IOS router" and if a router has XR, print, "'hostname' is a XR router"
  - Playbook need to find the router names dynamically from the inventory file.

Lab exercise

- playbook, op8-conditionals.yml

```
---
- name: print each host and its OS
  hosts: ALL

  tasks:
    - name: collect OS
      raw: show ver
      register: OS

    - name: print IOS host
      debug:
        msg: "{{ inventory_hostname }} is an IOS Router."
      when: OS.stdout | join('') | search('IOS XE')

    - name: print XR host
      debug:
        msg: "{{ inventory_hostname }} is an XR Router."
      when: OS.stdout | join('') | search('IOS XR')
```

Example output

```
cisco@ansible-controller:~$ ansible-playbook op8-conditionals.yml --syntax-check

playbook: op8-conditionals.yml
cisco@ansible-controller:~$ ansible-playbook op8-conditionals.yml

PLAY [print each host and its OS]
*********************************************************************************

TASK [collect OS]
*****************************************************************************************
*********
changed: [R1]
changed: [R2]

TASK [print IOS host]
*****************************************************************************************
*****
skipping: [R2]
ok: [R1] => {
    "msg": "R1 is an IOS Router."
}

TASK [print XR host]
*****************************************************************************************
******
skipping: [R1]
ok: [R2] => {
    "msg": "R2 is an XR Router."
}

PLAY RECAP
*****************************************************************************************
****************
R1                          : ok=2    changed=1    unreachable=0    failed=0
R2                          : ok=2    changed=1    unreachable=0    failed=0

cisco@ansible-controller:~$
```

4.5 Optional exercise op31-runcfg-bkup.yml

Objective

- Create a playbook to backup routers' config, with the below additional requirements:
  - Filename of the config files should include current timestamp to maintain uniqueness in filenames.
  - Config backup should be taken daily at 03:00 hrs UTC. Use linux cronjob for this task.
  - Execute your playbook and verify if the results meet the requirements.

Step-1: Playbook, op31-runcfg-bkup.yml

```
---
- name: backup all routers config
  hosts: all

  tasks:
    - name: collect config  from all routers
      connection: ssh
      raw: show run

      register: RUNCFG

    - set_fact: time="{{lookup('pipe','date \"+%Y-%m-%d-%H-%M\"')}}"

    - name: save output to a file
      connection: local
      copy:
        content: "{{ RUNCFG.stdout }}"
        dest: "/home/cisco/{{ inventory_hostname }}_{{ time }}.txt"
```

Step-2: Cronjob

- Append the below entry in /etc/crontab file
- You need sudo permissions (sudo password=cisco): `$ sudo vi /etc/crontab`

```
#Run Ansible Playbook rtr-cfg-bkup everyday at 3:00 am UTC to backup router configs

0 3 * * * cisco /usr/bin/ansible-playbook /home/cisco/op31-runcfg-bkup.yml
```

Reference

> set_fact: http://docs.ansible.com/ansible/latest/modules/set_fact_module.html

Example output

```
cisco@ansible-controller:~$ cat /etc/crontab
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# m h dom mon dow user  command
17 *    * * *   root    cd / && run-parts --report /etc/cron.hourly
25 6    * * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report
/etc/cron.daily )
47 6    * * 7   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report
/etc/cron.weekly )
52 6    1 * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report
/etc/cron.monthly )

#Run Ansible Playbook rtr-cfg-bkup everyday at 3:00 am UTC to backup router configs

0 3 * * * cisco /usr/bin/ansible-playbook /home/cisco/op31-runcfg-bkup.yml

cisco@ansible-controller:~$ ansible-playbook op31-runcfg-bkup.yml --syntax-check

playbook: op31-runcfg-bkup.yml
cisco@ansible-controller:~$ ansible-playbook op31-runcfg-bkup.yml

PLAY [backup all routers config]
*********************************************************************************

TASK [collect config  from all routers]
**************************************************************************
changed: [R1]
changed: [R2]

TASK [set_fact]
*****************************************************************************************
***********
ok: [R1]
ok: [R2]

TASK [save output to a file]
***************************************************************************************
changed: [R1]
changed: [R2]

PLAY RECAP
*****************************************************************************************
****************
R1                         : ok=3    changed=2    unreachable=0    failed=0
R2                         : ok=3    changed=2    unreachable=0    failed=0
```

```
cisco@ansible-controller:~$ ls -ltr R*

-rw-rw-r-- 1 cisco cisco 1973 Jun  5 18:34 R2_2018-06-05-18-34.txt
-rw-rw-r-- 1 cisco cisco 5018 Jun  5 18:34 R1_2018-06-05-18-34.txt
```

4.6 Optional exercise op33-mop.yml (MOP)

Objective

- We have two additional requirements on top of the MOP playbook that we already did.
  - Insert a delay of 30 seconds before collecting post-config data
  - Create a file with the differences between pre-config and post-config data

Approach

- For delay, "pause" module can be used.
- For diff, we can command module with diff argument.

Lab exercise

- Create playbook, op-33.yml, with the contents below:

```
---
- name: pre-config captures - play-1
  hosts: localhost

  tasks:
    - name: run precheck playbook
      command: ansible-playbook p33-ospf-capture.yml --tags PRECHECK

- name: configure OSPF on both routers - play-2
  import_playbook: p33-ospf-config.yml

- name: post-config captures - play-3
  hosts: localhost

  tasks:
    - pause: seconds=30

    - name: run postcheck playbook
      command: ansible-playbook p33-ospf-capture.yml --tags POSTCHECK

- name: Compare pre and post files and create a diff file - play-4
  hosts: localhost
  connection: local

  tasks:
    - set_fact: TIMESTAMP="{{lookup('pipe','date \"+%Y-%m-%d-%H-%M\"')}}"
    - name: diff pre and post IOS files
      command: diff /home/cisco/p33-precheck-ios-ospf.txt /home/cisco/p33-postcheck-ios-
ospf.txt

      register: IOS_DIFF

      failed_when: "IOS_DIFF.rc > 1"

    - name: create diff file with timestamp included in the name
      copy:
        content: "{{ IOS_DIFF.stdout }}"
        dest: "/home/cisco/op33-ios_diff_{{ TIMESTAMP }}.txt"

    - name: diff pre and post XR files
      command: diff /home/cisco/p33-precheck-xr-ospf.txt /home/cisco/p33-postcheck-xr-
ospf.txt

      register: XR_DIFF

      failed_when: "XR_DIFF.rc > 1"

    - name: create diff file with timestamp included in the name
      copy:
        content: "{{ XR_DIFF.stdout }}"
        dest: "/home/cisco/op33-xr_diff_{{ TIMESTAMP }}.txt"
```

Example output

```
cisco@ansible-controller:~$ ansible-playbook op33-mop.yml --syntax-check

playbook: op33-mop.yml
cisco@ansible-controller:~$ ansible-playbook op33-mop.yml

PLAY [pre-config captures, play-1]
*************************************************************************************

TASK [run precheck playbook]
***************************************************************************************
****
changed: [localhost]

PLAY [configure ospf on IOS routers - play-1]
****************************************************************************

TASK [pre-check for ospf config]
***************************************************************************************
ok: [R1]

PLAY [configure ospf on XR routers - play-2]
*****************************************************************************

TASK [pre-check for ospf config]
***************************************************************************************
```

```
ok: [R2]

PLAY [post-config captures, play-3]
**************************************************************************************

TASK [pause]
*****************************************************************************************
********************
Pausing for 30 seconds
(ctrl+C then 'C' = continue early, ctrl+C then 'A' = abort)
ok: [localhost]

TASK [run postcheck playbook]
*****************************************************************************************
***
changed: [localhost]

PLAY [Compare pre and post files and create a diff file, play-4]
*********************************************************

TASK [set_fact]
*****************************************************************************************
*****************
ok: [localhost]

TASK [diff pre and post IOS files]
***************************************************************************************
changed: [localhost]

TASK [create diff file with timestamp included in the name]
**************************************************************
changed: [localhost]

TASK [diff pre and post XR files]
****************************************************************************************
changed: [localhost]

TASK [create diff file with timestamp included in the name]
**************************************************************
changed: [localhost]

PLAY RECAP
*****************************************************************************************
**********************
R1                          : ok=1    changed=0    unreachable=0    failed=0
R2                          : ok=1    changed=0    unreachable=0    failed=0
localhost                   : ok=8    changed=6    unreachable=0    failed=0

cisco@ansible-controller:~$ ls -ltr op33*.txt
-rw-rw-r-- 1 cisco cisco 393 Jun  8 00:46 op33-ios_diff_2018-06-08-00-46.txt
-rw-rw-r-- 1 cisco cisco 489 Jun  8 00:46 op33-xr_diff_2018-06-08-00-46.txt
cisco@ansible-controller:~$
```

4.7 Ansible installation

- Ansible control machine is on Linux based systems with Python 2 (versions 2.6 or higher) or Python 3 (versions 3.5 or higher).
- Red Hat, Debian, CentOS, OS X (MAC OS), Ubuntu, BSDs etc. are supported. MS Windows OS is not supported.
- Installation steps are straight forward. Depending on your OS flavor, pick the steps from the installation guide.
- Ansible installation guide: http://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html
- Just for example, installtion steps for CentOS:

```
$ sudo yum update

$ sudo yum install ansible
```

- Installation steps for Ubuntu:

```
$ sudo apt-get update

$ sudo apt-get install software-properties-common

$ sudo apt-add-repository ppa:ansible/ansible

$ sudo apt-get update

$ sudo apt-get install ansible
```

4.8 Reference

- 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 installation: https://www.youtube.com/watch?v=NIEVaCBGOUc
- Introduction to YAML: https://www.youtube.com/watch?v=o9pT9cWzbnI
- Ansible - Playbooks for beginners: https://www.youtube.com/watch?v=Z01b9QZG0D0
- Python Configuration generation: - Kirk Byers Blog – Network configuration using Ansbile
- Ansible Up and Running – Lorin Hochstein

End of Lab Guide