Ansible Roles and Jinja2 Templating

Objective

In the previous session, you used ad hoc commands and playbooks to execute a particular set of tasks on the target devices. Ansible works well, when you want to scale both up and down. Easy to implement simple jobs as well as complex by breaking them into smaller ones.

In this session, you will be introduced to the concept of roles and Jinja2 templating. For complex scenarios, you can leverage roles to break a large playbook or building configuration across multiple OS/Devices into multiple files.

Roles:

Roles are ways of automatically loading tasks, templates, variables, and handlers into a project based on a known file structure. Unlike a playbook which contains all tasks and variables in a single file, roles will use multiple subdirectories and YAML files to separate the various actions. The roles directory structure is listed below. Roles must include at least one of these directories, however it is perfectly fine to exclude any which are not being used. When in use, each directory must contain a main.yml file, which contains the relevant content.

Roles Directory Structure:

```
roles/
├── xr        >> Name of this role
│   ├── defaults    >> default variables for the role
│   │   └── main.yml
│   ├── files       >> contains files which can be deployed via this role
│   ├── handlers    >> contains handlers, can used by this role or anywhere outside
this role
│   │   └── main.yml
│   ├── meta        >> defines some meta data for this role
│   │   └── main.yml
│   ├── README.md
│   ├── tasks       >> contains the main list of tasks to be executed by the role
│   │   └── main.yml
│   ├── templates   >> contains templates which can be deployed via this role
│   └── vars        >> contains variables used in this role
│       └── main.yml
```

Grouping roles based on platform type or platform function, will make it easier to build configurations for network devices.

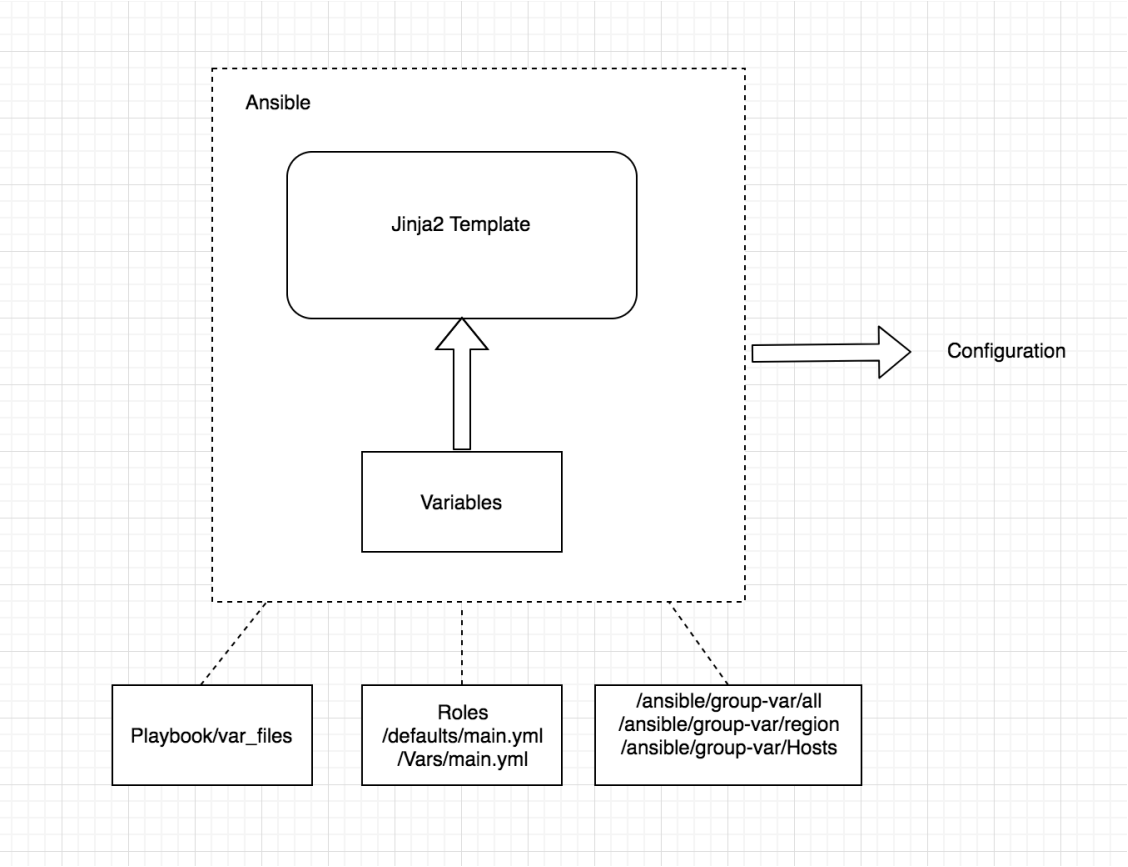
Note: The role's directory structure can be created manually or automatically using the ansible-galaxy command.

Jinja2 Templating

Network operators use templates for generating configurations for network devices. A template contains configs which are common across devices and some part of configs which are device specific. Jinja2 is a templating language for Python which can be accessed through the python API. Jinja2 files are text file that ends with *.j2 extensions. With Jinja2, a template contains logic for the variables and commands (loops etc) which on rendering can generate specific parameters.

Ansible provides separation between the playbooks and template logic. When playbooks are run, Ansible leverages the python code/modules/libraries to extract the specific variables from the logic (Jinja2 templates) and the input file. Variables are inserted into templates, and the final configs are generated by Ansible.

The following diagram shows a high-level flow of config generation using Jinja2 template.



Variables are defined with double brackets {{ Variables }} and their arguments are passed with single brackets { arguments }. Variables and logic commands can be used in both playbooks and in Jinja2 templates.

Syntax for Jinja2 templates:

1. Passing variable - **{{ variable }}**

2. For Loops - **{ % for interface in interface_list % } … { % endfor % }**
3. Calling template - **{ % extends source-template.j2 % }**
4. Sending block config **{ % block ospf % } {end block }**

Passing Variables:

Refer to Ansible documentation for order of precedence for variable
http://docs.ansible.com/ansible/latest/playbooks_variables.html

By using the key "with_items" we leverage iterations and loops. In the example below, we pass parameters for variable hostname; there are two values defined. On the first iteration, Ansible will render the item.hostname to xr-router-rt1 and the file will be stored as defined in the destination folder within the playbook.

```
cisco@Ansible-Controller:~/project1$ vi xr/tasks/main.yml

---
- name: Generate the configuration for xr-router1
  template: src=xr-template-part1.j2
dest=/home/cisco/project1/{{item.hostname}}.txt
  with_items:
  - { hostname: xr-router-rtr1 }
  - { hostname: xr-router-rtr2 }
...
# tasks file for xr
```

The variable coming through iteration with_items will also be passed to the Jinja2 template, and Ansible will substitute this parameter on executing the playbook.

Another example, for variable under with_items, the loop below refers to a different variable under " {{ xr hostnames }}". This variable is further defined inside vars/main.yml file. When using the roles directory structure, Ansible will know to expand under the /vars/main.yml and render the necessary arguments. Make note the variables are defined under **{{ variables }}** and values are defined as list with – { arguments } syntax.

```
cisco@Ansible-Controller:~/project1$ vi xr/tasks/main.yml

---
- name: Generate the configuration for xr-router1
  template: src=xr-template-part1.j2
dest=/home/cisco/project1/{{item.hostname}}.txt
  with_items:
    - "{{ xr_hostnames }}"

...
# tasks file for xr

cisco@Ansible-Controller:~/project1$ vi xr/vars/main.yml
---
xr_hostnames:
   - { hostname: xr-router-rtr1 }
   - { hostname: xr-router-rtr2 }
   - { hostname: xr-router-rtr3 }
```

```
  - { hostname: xr-router-rtr4 }
  - { hostname: xr-router-rtr5 }
...

# vars file for xr
```

For loops:

In jinja2 templating language, loop logic is invoked using {% for x in y %} syntax. A For loop is a continuous loop until it runs out of inputs coming from the variable y. Arguments for variable y will come from /vars/main.yml file. For loops are ended with syntax {% endfor %}

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

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

In above example, when an Ansible playbook is executed there will be two iterations of the for loop with values gig 0/0/0/0 & gig 0/0/0/0. Similarly, conditional loops can be built using **if** and **endif** statements, and they follow a similar syntax for starting a loop: **{% if x== “” %}** and end with **{% endif %}** syntax.

Refer to jinja2 documentation for additional looping logics and exact syntax.

http://jinja.pocoo.org/docs/2.10/templates/

Calling templates:

In Jinja2 you can call one template inside of another template to generate a full config. In exercise D, you will leverage hierarchical templates logic, by first creating one template for configurations that are common across roles and then creating another template for specific to some roles. Finally the role specific template will call the common template to build a full router config.

Within Jinja2, you will invoke the other template file by using the syntax **{ % extends xxx.j2}**

```
cisco@Ansible-Controller:~/project1$ more pe-p/templates/ler_config-template.j2

{% extends "ler_lsr_config_template.j2" %} #>>> name of source template.j2 file
```

Within configuration logics, you can define a block of configs by using the syntax **{ % block %}** This block of config will usually be defined under the role specific template. Each role can have its own unique set of configurations.

```
#/templates/ ler_config_template.j2

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

This block of configuration can be referred in base template through syntax **{ % block %}** **{ % endblock %}**

Notice all commands and logics leverage the **{ % xxx % }** syntax

```
#/templates/ ler_config_template.j2
{% block rsvp %}
{% endblock %}
!
```

In summary, Ansible uses Jinja2 templating to enable dynamic expressions and access to variables. All templating happens on the Ansible controller before the task is sent and executed on the target machine. This is done to minimize the requirements on the target (Jinja2 is only required on the controller) and to be able to pass the minimal information needed for the task.

Ansible-Galaxy

The ansible-galaxy command comes bundled with Ansible, and you will use it to install roles from Galaxy. Galaxy, is a free site for searching, downloading, and sharing community developed roles. You can create your own roles and publish them on the ansible-galaxy website, allowing other users to download and use your role. The ansible-galaxy command can be used it to create a new role, remove existing roles, or perform tasks on the Galaxy website.

Config Generation using roles and jinja2 templates

In the following sections, you will gain an understanding of how to use roles and Jinja2 templates for generating router configs. You will leverage the previous concepts learned through module 1, and 2. The following are the goals of each exercises:

Exercise A – In this section, you will generate a full XR router config for multiple routers using Jinja2 template format while utilizing the roles directory structure. You will create one role for a single device type (XR), and generate multiple device configs.

Exercise B – In this section, you will generate full router configs for different device types: IOSXE, IOSXR, and NXOS, using Jinja2 templates and Ansible roles. You will create one role to be used with three different device types and generate 3 unique device configs.

Exercise D - In this section, you will create a hierarchical template using nested templates. You will generate configurations for two different roles – LSR/LER using single device type (XR).

Exercise A – Role use with list and dictionaries – Passing multiple variable in Jinja2

In this exercise you will work towards building full router configuration for N (in this case 5) number of devices. You will again leverage the variables & loop concepts to generate configs based on templates.

Step 1: Create a new role xr2 using ansible-galaxy

```
cisco@Ansible-Controller:~/project1$ ansible-galaxy init xr-config
- xr2 was created successfully
```

Step 2: Create a playbook xr2-router-1.yml that will use the xr2 role. This file is hosted on the parent directory of xr2 role folder.

```
cisco@Ansible-Controller:~/project1$ vi xr-router.yml

---
  - name: Create a config for router1 from template XR
    hosts: localhost

    roles:
      - xr-config
...
```

Step 3: Edit the main.yml file under the xr2/tasks folder with information regarding the source location of the template, the location of output destination files, and the hostname variables to be used.

```
cisco@Ansible-Controller:~/project1$ vi xr-config/tasks/main.yml
---
- name: Generate the configuration for xr-router
  template:
     src=xr-config-template.j2
     dest=/home/cisco/project1/{{item.hostname}}.txt
  with_items:
     - "{{ xr_hostnames }}"

...
# tasks file for xr-config
```

Step 4: Create a template **(full-xr-config-template.j2)** with the common xr router configuration, and store it as a J2 template under the /xr2/template/ folder.

```
cisco@Ansible-Controller:~/project1$ vi xr-config/templates/xr-config-template.j2

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
 no ipv4 address
 cdp
 no shutdown
!
!
router ospf 1
 log adjacency changes
 area 0
  !
  interface Loopback0
   passive enable
  !
{% for interface in xr_interfaces %}
interface {{interface}}
cost 1
!
{% endfor %}
 !
!
!
```

You will notice in config, there are additional variables and for loops being called while generating the config.

Step 5: Define the variables needed to generate the template in the /xr2/vars/main.yml file. Each host will need to contain values for all the variables highlighted in the template file. The var file will contain the variable inputs needed both the with_items loop and the for loop.

```
cisco@Ansible-Controller:~/project1$ vi xr-config/vars/main.yml

---
xr_hostnames:
   - { hostname: xr-router-rtr11, timezone: EST, timezone_dst: EDT,
timezone_offset: -5, loopback0_ip: 192.168.1.11, loopback0_mask: 255.255.255.255,
```

```
gig0000_ip: 10.1.11.1, gig0000_mask: 255.255.255.252, gig0001_ip: 10.1.11.4 ,
gig0001_mask: 255.255.255.252,}
   - { hostname: xr-router-rtr12, timezone: MST, timezone_dst: MDT,
timezone_offset: -7, loopback0_ip: 192.168.1.12, loopback0_mask: 255.255.255.255,
gig0000_ip: 10.1.12.1, gig0000_mask: 255.255.255.252, gig0001_ip: 10.1.12.4 ,
gig0001_mask: 255.255.255.252,}
   - { hostname: xr-router-rtr13, timezone: MST, timezone_dst: MDT,
timezone_offset: -7, loopback0_ip: 192.168.1.13, loopback0_mask: 255.255.255.255,
gig0000_ip: 10.1.13.1, gig0000_mask: 255.255.255.252, gig0001_ip: 10.1.13.4,
gig0001_mask: 255.255.255.252,}
   - { hostname: xr-router-rtr14, timezone: PST, timezone_dst: PDT,
timezone_offset: -8, loopback0_ip: 192.168.1.14, loopback0_mask: 255.255.255.255,
gig0000_ip: 10.1.14.1, gig0000_mask: 255.255.255.252, gig0001_ip: 10.1.14.4 ,
gig0001_mask: 255.255.255.252,}
   - { hostname: xr-router-rtr15, timezone: MST, timezone_dst: MDT,
timezone_offset: -7, loopback0_ip: 192.168.1.15, loopback0_mask: 255.255.255.255,
gig0000_ip: 10.1.15.1, gig0000_mask: 255.255.255.252, gig0001_ip: 10.1.15.4 ,
gig0001_mask: 255.255.255.252,}

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

# vars file for xr2
```

Step 7: Execute the xr-router-1.yml playbook.

```
cisco@Ansible-Controller:~/project1$ ansible-playbook xr-router.yml

[WARNING]: log file at /var/log/ansible.log is not writeable and we cannot
create it, aborting

[DEPRECATION WARNING]: DEFAULT_SUDO_USER option, In favor of become which is a
generic framework . This feature will be removed in version 2.8. Deprecation
warnings can be disabled by setting
deprecation_warnings=False in ansible.cfg.

PLAY [Create a config for router1 from template XR]
********************************************************************************
*********************************************************************

TASK [Gathering Facts]
********************************************************************************
********************************************************************************
***********************
ok: [localhost]

TASK [xr2 : Generate the configuration for xr-router1]
********************************************************************************
*********************************************************************
changed: [localhost] => (item={u'timezone_dst': u'EDT', u'gig0000_mask':
u'255.255.255.252', u'timezone_offset': -5, u'hostname': u'xr-router-rtr11',
u'loopback0_ip': u'192.168.1.11', u'timezone': u'EST', u'gig0001_mask':
u'255.255.255.252', u'gig0000_ip': u'10.1.11.1', u'gig0001_ip': u'10.1.11.4',
u'loopback0_mask': u'255.255.255.255'})
changed: [localhost] => (item={u'timezone_dst': u'MDT', u'gig0000_mask':
u'255.255.255.252', u'timezone_offset': -7, u'hostname': u'xr-router-rtr12',
u'loopback0_ip': u'192.168.1.12', u'timezone': u'MST', u'gig0001_mask':
u'255.255.255.252', u'gig0000_ip': u'10.1.12.1', u'gig0001_ip': u'10.1.12.4',
u'loopback0_mask': u'255.255.255.255'})
changed: [localhost] => (item={u'timezone_dst': u'MDT', u'gig0000_mask':
u'255.255.255.252', u'timezone_offset': -7, u'hostname': u'xr-router-rtr13',
```

```
u'loopback0_ip': u'192.168.1.13', u'timezone': u'MST', u'gig0001_mask':
u'255.255.255.252', u'gig0000_ip': u'10.1.13.1', u'gig0001_ip': u'10.1.13.4',
u'loopback0_mask': u'255.255.255.255'})
changed: [localhost] => (item={u'timezone_dst': u'PDT', u'gig0000_mask':
u'255.255.255.252', u'timezone_offset': -8, u'hostname': u'xr-router-rtr14',
u'loopback0_ip': u'192.168.1.14', u'timezone': u'PST', u'gig0001_mask':
u'255.255.255.252', u'gig0000_ip': u'10.1.14.1', u'gig0001_ip': u'10.1.14.4',
u'loopback0_mask': u'255.255.255.255'})
changed: [localhost] => (item={u'timezone_dst': u'MDT', u'gig0000_mask':
u'255.255.255.252', u'timezone_offset': -7, u'hostname': u'xr-router-rtr15',
u'loopback0_ip': u'192.168.1.15', u'timezone': u'MST', u'gig0001_mask':
u'255.255.255.252', u'gig0000_ip': u'10.1.15.1', u'gig0001_ip': u'10.1.15.4',
u'loopback0_mask': u'255.255.255.255'})

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

Step 6: Validate the router configs are generated in the desired folder and explore the generated files.

```
cisco@Ansible-Controller:~/project1$ ls -al *rtr1?.txt

-rw-rw-r-- 1 cisco cisco 1317 Apr 26 01:59 xr-router-rtr11.txt
-rw-rw-r-- 1 cisco cisco 1317 Apr 26 01:59 xr-router-rtr12.txt
-rw-rw-r-- 1 cisco cisco 1317 Apr 26 01:59 xr-router-rtr13.txt
-rw-rw-r-- 1 cisco cisco 1317 Apr 26 01:59 xr-router-rtr14.txt
-rw-rw-r-- 1 cisco cisco 1317 Apr 26 01:59 xr-router-rtr15.txt
```

In the above section, you were exposed passing device and role specific variables. Notice how easy and quickly you were able to generate full XR router configs for five different routers. By adding more variables to the device template, you can easily scale to higher number of nodes.

Exercise B – Generate full router config files for multiple device types using Ansible Roles and Jinja2 template

In exercise A and B, you were exposed to the usage of roles and their file structures within the use of a single role and single device type (XR routers). Now, let us switch from creating roles based on device types to creating roles base on usage/feature. In this section, you will create a standard role called core-config, and in this role you will create the core configs that will be used by different types of routers (IOS, XR, and NXOS). To do this, you will need to create different template files for each router type but still associate the templates to one role.

Step 1: Create a new role called **core-config**.

```
cisco@Ansible-Controller:~/project1$ ansible-galaxy init core-config
- core-config was created successfully
```

Step 2: Create a playbook core-router-config that will use the core-config role. This file is hosted on the parent directory of the core-config role folder.

```
cisco@Ansible-Controller:~/project1$ vi core-router-config.yml

---
  - name: Playbook to create core router configs for IOSXE, IOSXR, & NXOS Routers
    hosts: localhost

    roles:
      - core-config
...
```

Step 3: Edit the main.yml file under the core-config/tasks folder with information regarding the source location of the template, the location of output destination files, and the hostname variables to be used.

```
cisco@Ansible-Controller:~/project1$ vi core-config/tasks/main.yml

---
- name: Generate the configuration for xr-router1
  template:
     src=full-xr-config-template.j2
     dest=/home/cisco/project1/{{item.hostname}}.txt
  with_items:
     - "{{ xr_hostnames }}"

- name: Generate the configuration for nxos-router2
  template:
     src=full-nxos-config-template.j2
     dest=/home/cisco/project1/{{item.hostname}}.txt
  with_items:
     - "{{ nxos_hostnames }}"

- name: Generate the configuration for iosxe-router3
  template:
     src=full-ios-config-template.j2
     dest=/home/cisco/project1/{{item.hostname}}.txt
  with_items:
     - "{{ ios_hostnames }}"
...

# tasks file for core-config
```

We are generating configuration for IOS-XE, NXOS and IOS-XR routers. Each device type, leverages its own source template from the templates folder and its respective variables from the vars folder.

Step 4: Create the platform specific configuration template as J2 template and save them under the core-config/templates folder.

Template configuration for IOS Config:

```
cisco@Ansible-Controller:~/project1$ vi core-config/templates/full-ios-config-template.j2

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
!
!
!
license accept end user agreement
license boot level premium
!
!
no aaa new-model
!
!
ipv6 unicast-routing
!
!
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
no service config
enable password cisco
enable secret 4 tnhtc92DXBhelxjYk8LWJrPV36S2i4ntXrpb4RFmfqY
ip classless
ip subnet-zero
no ip domain lookup
ip domain name virl.info
crypto key generate rsa modulus 768
ip ssh server algorithm authentication password
username cisco privilege 15 secret cisco
line vty 0 4
 transport input ssh telnet
 exec-timeout 720 0
 password cisco
 login local
line con 0
 password cisco
!
cdp run
!
!
interface Loopback0
  description Loopback
  ip address {{item.loopback0_ip}} {{item.loopback0_mask}}
!
interface GigabitEthernet1
  description OOB Management
```

```
 vrf forwarding Mgmt-intf
 ! Configured on launch
 no ip address
 cdp enable
 no shutdown
!
interface GigabitEthernet2
 description to Ansible-Controller
 ip address {{item.gigaethernet2_ip}} {{item.gigaethernet2_mask}}
 cdp enable
 ip ospf cost 1
 no shutdown
!
interface GigabitEthernet3
 description to R2-XRv
 ip address {{item.gigaethernet2_ip}} {{item.gigaethernet3_mask}}
 cdp enable
 ip ospf cost 1
 no shutdown
!
!
!
router ospf 1
 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}}
!
!
```

Template configuration for NXOS Config:

```
cisco@Ansible-Controller:~/project1$ vi core-config/templates/full-nxos-config-template.j2
!
hostname {{item.hostname}}
vdc {{item.hostname}} id 1
 allocate interface {{item.allocate_int_2}}
 allocate interface {{item.allocate_int_3}}
 limit-resource vlan minimum {{item.vlan_min}} maximum {{item.vlan_max}}
 limit-resource vrf minimum {{item.vrf_min}} maximum {{item.vrf_max}}
 limit-resource port-channel minimum 0 maximum 768
 limit-resource u4route-mem minimum 96 maximum 96
 limit-resource u6route-mem minimum 24 maximum 24
 limit-resource m4route-mem minimum 58 maximum 58
 limit-resource m6route-mem minimum 8 maximum 8

{% for feature in feature_list %}
```

```
feature {{feature}}
{% endfor %}

username adminbackup password 5 ! role {{item.role2}}
username {{item.user1}} password 5 {{item.password1}} role {{item.role1}}
username {{item.user2}} password 5 {{item.password2}} role {{item.role2}}
username {{item.user2}} role {{item.role1}}
username lab password 5 {{item.password1}} role {{item.role1}}


no password strength-check
ip domain-lookup
copp profile strict
nmp-server user {{item.user1}} {{item.role1}}
snmp-server user {{item.user2}} {{item.role2}}
snmp-server user {{item.user2}} {{item.role1}}
rmon event 1 log trap public description FATAL(1) owner PMON@FATAL
rmon event 2 log trap public description CRITICAL(2) owner PMON@CRITICAL
rmon event 3 log trap public description ERROR(3) owner PMON@ERROR
rmon event 4 log trap public description WARNING(4) owner PMON@WARNING
rmon event 5 log trap public description INFORMATION(5) owner PMON@INFO

vlan 1

vrf context management
hardware forwarding unicast trace

interface Loopback0
  description Loopback
  ip address {{item.loopback0_ip}} {{item.loopback0_mask}}
  ip router ospf 1 area 0

interface Ethernet2/1
  description to R2-XRv
  ip address {{item.ethernet21_ip}} {{item.ethernet21_mask}}
  ip router ospf 1 area 0
  duplex full
  mac-address fa16.3e00.0001
  no shutdown

interface Ethernet2/2
  description to Ansible-Controller
  ip address {{item.ethernet22_ip}} {{item.ethernet22_mask}}
  ip router ospf 1 area 0
  duplex full
  mac-address fa16.3e00.0002
  no shutdown
```

```
interface mgmt0
 description OOB Management
 ! Configured on launch
 no ip address
 duplex full
 mac-address fa16.3e00.0003
 no shutdown
 vrf member management

line console
line vty
router ospf 1
 router-id {{item.loopback0_ip}}
!
```

Template configuration for XR Config:

```
cisco@Ansible-Controller:~/project1$ vi core-config/templates/full-xr-config-template.j2

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
  no ipv4 address
  cdp
  no shutdown
!
!
router ospf 1
  log adjacency changes
  area 0
    !
    interface Loopback0
      passive enable
    !
{% for interface in xr_interfaces %}
interface {{interface}}
cost 1
```

```
!
{% endfor %}
 !
!
!
```

Verify the templates have been created and are under the core-config/templates directory.

```
cisco@Ansible-Controller:~/project1$ ls -ltr core-config/templates/
total 12
-rw-rw-r-- 1 cisco cisco 1750 Jan 21 00:18 full-ios-config-template.j2
-rw-rw-r-- 1 cisco cisco 1421 Jan 21 00:27 full-xr-config-template.j2
-rw-rw-r-- 1 cisco cisco 2306 Jan 21 00:27 full-nxos-config-template.j2
```

Step 5: Define the variables needed to generate the template in the /core-config/vars/main.yml file. Each host will need to contain values for all the variables highlighted in the template file.

```
cisco@Ansible-Controller:~/project1$ vi core-config/vars/main.yml

---
xr_hostnames:
   - { hostname: xr-router-rtr111, timezone: EST, timezone_dst: EDT,
timezone_offset: -5, loopback0_ip: 192.168.1.111, loopback0_mask: 255.255.255.255,
gig0000_ip: 10.1.111.1, gig0000_mask: 255.255.255.252, gig0001_ip: 10.1.111.4 ,
gig0001_mask: 255.255.255.252,}

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

nxos_hostnames:
   - { hostname: nxos-router-rtr112, allocate_int_2: Ethernet2/1-48,
allocate_int_3: Ethernet3/1-48, vlan_min: 16, vlan_max: 4094, vrf_min: 16, vrf_max:
4096, timezone: EST, timezone_dst: EDT, timezone_offset: -5, user1: admin, user2:
cisco, password1: cisco123, password2: cisco1234, role1: network-admin, role2:
network-operator, loopback0_ip: 192.168.1.112, loopback0_mask: 255.255.255.255,
ethernet21_ip: 10.1.11.1, ethernet21_mask: 255.255.255.252, ethernet22_ip:
10.1.11.4, ethernet22_mask: 255.255.255.252,}

feature_list:
  - telnet
  - ospf
  - bgp

nxos_interfaces:
  - Ethernet2/1
  - Ethernet2/2

ios_hostnames:
   - { hostname: ios-router-rtr113, timezone: EST, timezone_dst: EDT,
timezone_offset: -5, loopback0_ip: 192.168.1.113, loopback0_mask: 255.255.255.255,
gigaethernet2_ip: 10.1.113.1, gigaethernet2_mask: 255.255.255.252,
gigaethernet3_ip: 10.1.113.4 , gigaethernet3_mask: 255.255.255.252, ospf_network1:
192.168.1.113, ospf_network_mask1: 0.0.0.0, ospf_network2: 10.1.113.0,
ospf_network_mask2: 0.0.0.3, ospf_network3: 10.2.113.0, ospf_network_mask3:
0.0.0.3, areaid: 0 }

ios_interfaces:
```

```
  - GigabitEthernet0/0/0/0
  - GigabitEthernet0/0/0/1

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

Platform specific information is defined within each of its variable. For example, within NXOS, it has commands specific to the platform, such as features, interface numbers, and specific CLI format for configuring.

Step 6: Execute the playbook core-router-config.yml

You will notice three different configurations are generated and stored in target location.

```
cisco@Ansible-Controller:~/project1$ ansible-playbook core-router-config.yml

PLAY [Playbook to create core router configs for IOSXE, IOSXR, & NXOS Routers]
*********************************************************************************
*****************************************

TASK [Gathering Facts]
*********************************************************************************
*********************************************************************************
**********************
ok: [localhost]

TASK [core-config : Generate the configuration for xr-router1]
*********************************************************************************
************************************************************
changed: [localhost] => (item={u'timezone_dst': u'EDT', u'gig0000_mask':
u'255.255.255.252', u'timezone_offset': -5, u'hostname': u'xr-router-rtr111',
u'loopback0_ip': u'192.168.1.111', u'timezone': u'EST', u'gig0001_mask':
u'255.255.255.252', u'gig0000_ip': u'10.1.111.1', u'gig0001_ip':
u'10.1.111.4', u'loopback0_mask': u'255.255.255.255'})

TASK [core-config : Generate the configuration for nxos-router2]
*********************************************************************************
************************************************************
changed: [localhost] => (item={u'password1': u'cisco123', u'password2':
u'cisco1234', u'user2': u'cisco', u'user1': u'admin', u'ethernet22_mask':
u'255.255.255.252', u'timezone': u'EST', u'allocate_int_2': u'Ethernet2/1-48',
u'allocate_int_3': u'Ethernet3/1-48', u'role1': u'network-admin', u'role2':
u'network-operator', u'hostname': u'nxos-router-rtr112', u'vlan_min': 16,
u'loopback0_mask': u'255.255.255.255', u'vlan_max': 4094, u'timezone_dst':
u'EDT', u'timezone_offset': -5, u'ethernet21_ip': u'10.1.11.1',
u'loopback0_ip': u'192.168.1.112', u'vrf_max': 4096, u'ethernet21_mask':
u'255.255.255.252', u'ethernet22_ip': u'10.1.11.4', u'vrf_min': 16})

TASK [core-config : Generate the configuration for iosxe-router3]
*********************************************************************************
************************************************************
changed: [localhost] => (item={u'timezone_dst': u'EDT', u'areaid': 0,
u'ospf_network3': u'10.2.113.0', u'gigaethernet2_mask': u'255.255.255.252',
u'gigaethernet3_mask': u'255.255.255.252', u'ospf_network1': u'192.168.1.113',
u'timezone_offset': -5, u'hostname': u'ios-router-rtr113',
u'ospf_network_mask1': u'0.0.0.0', u'ospf_network_mask2': u'0.0.0.3',
u'loopback0_ip': u'192.168.1.113', u'gigaethernet3_ip': u'10.1.113.4',
u'timezone': u'EST', u'ospf_network_mask3': u'0.0.0.3', u'gigaethernet2_ip':
```

```
u'10.1.113.1', u'ospf_network2': u'10.1.113.0', u'loopback0_mask':
u'255.255.255.255'})

PLAY RECAP
********************************************************************************
********************************************************************************
*********************************
localhost                  : ok=4    changed=3    unreachable=0    failed=0
```

Step 7: Validate the router configurations were generated in the correct folder and explore the generated files.

```
cisco@Ansible-Controller:~/project1$ pwd
/home/cisco/project1
cisco@Ansible-Controller:~/project1$ ls -al *rtr11?.txt
-rw-rw-r-- 1 cisco cisco 1545 Jan 21 01:10 ios-router-rtr113.txt
-rw-rw-r-- 1 cisco cisco 2063 Jan 21 01:10 nxos-router-rtr112.txt
-rw-rw-r-- 1 cisco cisco 1321 Jan 21 01:10 xr-router-rtr111.txt
```

In this section, you used full config templates for three different types of routers to generate the router configuration files.

Exercise C – Create a playbook that runs multiple role

Step 1: Create an playbook that invokes multiple roles in single play

```
cisco@Ansible-Controller:~/project1$ vi multirole.yml

---
 - name: Test case to run multiple roles in single play
   hosts: localhost

   roles:
       - xr-config
       - core-config

...
```

Step 2: Execute the playbook and observe both the roles being executed

```
cisco@Ansible-Controller:~/project1$  ansible-playbook multirole.yml
[WARNING]: log file at /var/log/ansible.log is not writeable and we cannot create it, aborting

[DEPRECATION WARNING]: DEFAULT_SUDO_USER option, In favor of become which is a
generic framework . This feature will be removed in version 2.8. Deprecation warnings can be
disabled by setting
deprecation_warnings=False in ansible.cfg.
```

```
PLAY [Test case to run multiple roles in single play]
*****************************************************************************
*********************************************************************

TASK [Gathering Facts]
*****************************************************************************
*****************************************************************************
************************
ok: [localhost]

TASK [xr-config : Generate the configuration for xr-router]
*****************************************************************************
*************************************************************
ok: [localhost] => (item={u'timezone_dst': u'EDT', u'gig0000_mask': u'255.255.255.252',
u'timezone_offset': -5, u'hostname': u'xr-router-rtr11', u'loopback0_ip': u'192.168.1.11',
u'timezone': u'EST', u'gig0001_mask': u'255.255.255.252', u'gig0000_ip': u'10.1.11.1',
u'gig0001_ip': u'10.1.11.4', u'loopback0_mask': u'255.255.255.255'})
ok: [localhost] => (item={u'timezone_dst': u'MDT', u'gig0000_mask': u'255.255.255.252',
u'timezone_offset': -7, u'hostname': u'xr-router-rtr12', u'loopback0_ip': u'192.168.1.12',
u'timezone': u'MST', u'gig0001_mask': u'255.255.255.252', u'gig0000_ip': u'10.1.12.1',
u'gig0001_ip': u'10.1.12.4', u'loopback0_mask': u'255.255.255.255'})
ok: [localhost] => (item={u'timezone_dst': u'MDT', u'gig0000_mask': u'255.255.255.252',
u'timezone_offset': -7, u'hostname': u'xr-router-rtr13', u'loopback0_ip': u'192.168.1.13',
u'timezone': u'MST', u'gig0001_mask': u'255.255.255.252', u'gig0000_ip': u'10.1.13.1',
u'gig0001_ip': u'10.1.13.4', u'loopback0_mask': u'255.255.255.255'})
ok: [localhost] => (item={u'timezone_dst': u'PDT', u'gig0000_mask': u'255.255.255.252',
u'timezone_offset': -8, u'hostname': u'xr-router-rtr14', u'loopback0_ip': u'192.168.1.14',
u'timezone': u'PST', u'gig0001_mask': u'255.255.255.252', u'gig0000_ip': u'10.1.14.1',
u'gig0001_ip': u'10.1.14.4', u'loopback0_mask': u'255.255.255.255'})
ok: [localhost] => (item={u'timezone_dst': u'MDT', u'gig0000_mask': u'255.255.255.252',
u'timezone_offset': -7, u'hostname': u'xr-router-rtr15', u'loopback0_ip': u'192.168.1.15',
u'timezone': u'MST', u'gig0001_mask': u'255.255.255.252', u'gig0000_ip': u'10.1.15.1',
u'gig0001_ip': u'10.1.15.4', u'loopback0_mask': u'255.255.255.255'})

TASK [core-config : Generate the configuration for xr-router1]
*****************************************************************************
*************************************************************
ok: [localhost] => (item={u'timezone_dst': u'EDT', u'gig0000_mask': u'255.255.255.252',
u'timezone_offset': -5, u'hostname': u'xr-router-rtr111', u'loopback0_ip': u'192.168.1.111',
u'timezone': u'EST', u'gig0001_mask': u'255.255.255.252', u'gig0000_ip': u'10.1.111.1',
u'gig0001_ip': u'10.1.111.4', u'loopback0_mask': u'255.255.255.255'})

TASK [core-config : Generate the configuration for nxos-router2]
*****************************************************************************
*************************************************************
ok: [localhost] => (item={u'password1': u'cisco123', u'password2': u'cisco1234', u'user2':
u'cisco', u'user1': u'admin', u'ethernet22_mask': u'255.255.255.252', u'timezone': u'EST',
u'allocate_int_2': u'Ethernet2/1-48', u'allocate_int_3': u'Ethernet3/1-48', u'role1': u'network-
```

```
admin', u'role2': u'network-operator', u'hostname': u'nxos-router-rtr112', u'vlan_min': 16,
u'loopback0_mask': u'255.255.255.255', u'vlan_max': 4094, u'timezone_dst': u'EDT',
u'timezone_offset': -5, u'ethernet21_ip': u'10.1.11.1', u'loopback0_ip': u'192.168.1.112',
u'vrf_max': 4096, u'ethernet21_mask': u'255.255.255.252', u'ethernet22_ip': u'10.1.11.4',
u'vrf_min': 16})

TASK [core-config : Generate the configuration for iosxe-router3]
*********************************************************************************
*************************************************************
ok: [localhost] => (item={u'timezone_dst': u'EDT', u'areaid': 0, u'ospf_network3': u'10.2.113.0',
u'gigaethernet2_mask': u'255.255.255.252', u'gigaethernet3_mask': u'255.255.255.252',
u'ospf_network1': u'192.168.1.113', u'timezone_offset': -5, u'hostname': u'ios-router-rtr113',
u'ospf_network_mask1': u'0.0.0.0', u'ospf_network_mask2': u'0.0.0.3', u'loopback0_ip':
u'192.168.1.113', u'gigaethernet3_ip': u'10.1.113.4', u'timezone': u'EST',
u'ospf_network_mask3': u'0.0.0.3', u'gigaethernet2_ip': u'10.1.113.1', u'ospf_network2':
u'10.1.113.0', u'loopback0_mask': u'255.255.255.255'})

PLAY RECAP
*********************************************************************************
*********************************************************************************
************************************
localhost                  : ok=5    changed=0    unreachable=0    failed=0
```

Exercise D – Generate router config files using a Hierarchical Template

One of the drawbacks of the previous approach is that when using full router configs, if there is any change in the common configuration, the end user has to go back and update all the base templates for each device. In this section, you will create a hierarchical template which consists of one common template that can get called by other templates, creating a nested configuration.

Step 1: Create a new role called **pe-p**.

```
cisco@Ansible-Controller:~/project1$ ansible-galaxy init ler-lsr
- pe-p was created successfully
```

Step 2: Create a playbook xr-hier that will use the pe-p role. This file is hosted on the parent directory of the pe-p role folder.

```
cisco@Ansible-Controller:~/project1$ vi xr-hier.yml

---
 - name: Create a config for ler-lsr config from template XR
   hosts: localhost

   roles:
     - ler-lsr
```

```
...
```

Step 3: Edit the main.yml file under the ler-lsr/tasks folder with information regarding the source location of the template, the location of the output destination files, and the hostname variables to be used.

```
cisco@Ansible-Controller:~/project1$ vi ler-lsr/tasks/main.yml

---
- name: Generate the configuration for LER/PE Router1
  template:
    src=ler_config_template.j2
    dest=/home/cisco/project1/{{item.hostname}}.txt
  with_items:
    - "{{ ler_routers }}"

- name: Generate the configuration for LSR/P Router2
  template:
    src=lsr_config_template.j2
    dest=/home/cisco/project1/{{item.hostname}}.txt
  with_items:
    - "{{ lsr_routers }}"
...
```

Step 4: Create the three different templates listed below and save them under the /ler-lsr/templates/ folder.

ler_lsr_config_template.j2 - File contains the common configuration used for LSR/LER.

ler_config_template.j2 - File contains configuration specific to LER/PE Role.

lsr_config_template.j2 - File contains configuration specific to LSR/P Role.

Note - We use a combination of { extend/block } function to pass arguments and configuration across the templates.

The following are the common configuration used across ler-lsr device.

```
cisco@Ansible-Controller:~/project1$ vi ler-lsr/templates/ler_lsr_config_template.j2

hostname {{item.hostname}}
service timestamps log datetime msec
service timestamps debug datetime msec
telnet vrf default ipv4 server max-servers 10
telnet vrf Mgmt-intf ipv4 server max-servers 10
domain name virl.info
domain lookup disable
cdp
vrf Mgmt-intf
 address-family ipv4 unicast
 !
 address-family ipv6 unicast
```

```
 !
!
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
   !
  !
 !
!
interface Loopback0
 description Loopback
 ipv4 address {{item.loopback0_ip}} {{item.loopback0_mask}}
!
interface Loopback1
 ipv4 address {{item.loopback1_ip}} {{item.loopback1_mask}}

!
interface Loopback2
 ipv4 address {{item.loopback2_ip}} {{item.loopback2_mask}}
!
{% block interface_ip %}
{% endblock%}
!
interface MgmtEth0/0/CPU0/0
 description OOB Management
 cdp
 ! Configured on launch
 vrf Mgmt-intf
 ipv4 address {{item.mgmt_ip}} {{item.mgmt_mask}}
!
route-policy bgp_in
  pass
end-policy
!
route-policy bgp_out
```

```
  pass
end-policy
!
{% block isis %}
{% endblock %}
!
{% block bgp %}
{% endblock %}
!
!
mpls oam
!

{% block rsvp %}
{% endblock %}
!
{% block mpls_traffic_eng %}
{% endblock %}
 !
 bandwidth-accounting
  application
   enable
   interval 180
  !
  sampling-interval 60
  adjustment-factor 100
  flooding threshold up 10 down 10
 !
!
!
ssh server v2
ssh server vrf Mgmt-intf
end
```

The following are the Ler specific configuration:

```
cisco@as-rtp-cisco-st31:~/playbooks/roles/htemp$ vi ler-
lsr/templates/ler_config_template.j2

{% extends "ler_lsr_config_template.j2" %}
{% block interface_ip %}
!
interface tunnel-te1
 bandwidth 100
 ipv4 unnumbered Loopback0
 shutdown
 autoroute announce
```

```
  exclude-traffic segment-routing
 !
 destination 192.168.0.5
 path-option 1 dynamic
!
interface tunnel-te1000
 bandwidth 100
 ipv4 unnumbered Loopback0
 shutdown
 autoroute announce
  exclude-traffic segment-routing
 !
 destination 192.168.0.6
 path-option 1 dynamic
 !

interface GigabitEthernet0/0/0/0
 description to LER-01
 cdp
 ipv4 address {{item.gig0000_ip}} {{item.gig0000_mask}}
!
interface GigabitEthernet0/0/0/1
 description to LSR-02
 cdp
 ipv4 address {{item.gig0001_ip}} {{item.gig0001_mask}}
!
interface GigabitEthernet0/0/0/2
 description to LSR-03
 cdp
 ipv4 address {{item.gig0002_ip}} {{item.gig0002_mask}}
!
{% endblock %}

{% block isis %}
!
router isis {{item.isisno}}
 net 49.1921.6800.0001.00
 segment-routing global-block 400000 464000
 address-family ipv4 unicast
  metric-style wide
  mpls traffic-eng level-2-only
  mpls traffic-eng router-id Loopback0
  segment-routing mpls
 !
 interface Loopback0
  passive
  circuit-type level-2-only
  address-family ipv4 unicast
```

```
 !
 !
 interface Loopback1
  passive
  circuit-type level-2-only
  address-family ipv4 unicast
   prefix-sid index 11
 !
 {% for interface in interface_list_ler %}
 !
 interface {{interface}}
  circuit-type level-2-only
  point-to-point
  address-family ipv4 unicast
   metric 10
  !
  address-family ipv6 unicast
   metric 10
  !
 {% endfor %}
 {% endblock %}

 {% block bgp %}
!
router bgp {{item.bgpas}}
 bgp router-id {{item.loopback0_ip}}
 address-family ipv4 unicast
  network {{item.loopback0_ip}}
 !

 ! iBGP
 ! iBGP peers
 neighbor {{item.peerip}}
  remote-as {{item.peeras}}
  description iBGP peer BB03
  update-source Loopback0
  address-family ipv4 unicast
  !
!
{% endblock %}
!

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

```
 !
{% endfor %}
{% endblock %}
!
{% block mpls_traffic_eng %}
mpls traffic-eng
{% for interface in interface_list_ler %}
!
interface {{interface}}
!
{% endfor %}
{% endblock %}
 !
```

The following are LSR specific device configuration:

```
cisco@Ansible-Controller:~/project1$ vi ler-lsr/templates/lsr_config_template.j2

{% extends "ler_lsr_config_template.j2" %}
{% block interface_ip %}
!
interface GigabitEthernet0/0/0/0
 description to LER01
 cdp
 ipv4 address {{item.gig0000_ip}} {{item.gig0000_mask}}
!
interface GigabitEthernet0/0/0/1
 description to LSR03
 cdp
 ipv4 address {{item.gig0001_ip}} {{item.gig0001_mask}}
!
{% endblock%}

{% block isis %}
!
router isis 123
 net 49.1921.6800.0006.00
 segment-routing global-block 400000 464000
 address-family ipv4 unicast
  metric-style wide
  mpls traffic-eng level-2-only
  mpls traffic-eng router-id Loopback0
  segment-routing mpls
  !
 interface Loopback0
  passive
  circuit-type level-2-only
```

```
 address-family ipv4 unicast
 !
interface Loopback1
 passive
 circuit-type level-2-only
 address-family ipv4 unicast
  prefix-sid index 66
 !
!
{% for interface in interface_list_lsr %}
!
interface {{interface}}
 circuit-type level-2-only
 point-to-point
 address-family ipv4 unicast
  metric 10
 !
 address-family ipv6 unicast
  metric 10
 !
{% endfor %}
 {% endblock %}
!
{% block rsvp %}
rsvp
 {% for interface in interface_list_lsr %}
 interface {{interface}}
  bandwidth percentage 100
 !
{% endfor %}
{% endblock %}
!
{% block mpls_traffic_eng %}
mpls traffic-eng
{% for interface in interface_list_lsr %}
!
interface {{interface}}
!
{% endfor %}
{% endblock %}
!
```

Verify the templates have been created and are under the core-config/templates directory.

```
cisco@Ansible-Controller:~/project1$ ls -ltr ler-lsr/templates/
total 12
-rw-rw-r-- 1 cisco cisco 2161 Jan 21 01:57 ler_config_template.j2
-rw-rw-r-- 1 cisco cisco 1456 Jan 21 02:31 ler_lsr_config_template.j2
```

```
-rw-rw-r-- 1 cisco cisco 1323 Jan 21 02:31 lsr_config_template.j2
```

Step 5: Define the variables needed to generate the template in the /pe-p/vars/main.yml file. Each host will need to contain values for all the variables highlighted in the template file.

```
cisco@Ansible-Controller:~/project1$ vi ler-lsr/vars/main.yml

---
ler_routers:
    - { hostname: ler1_xr_cl2018, loopback0_ip: 192.168.0.1, loopback0_mask:
255.255.255.255, loopback1_ip: 192.168.1.1, loopback1_mask: 255.255.255.255,
loopback2_ip: 192.168.2.1, loopback2_mask: 255.255.255.255, mgmt_ip: 172.16.1.1,
mgmt_mask: 255.255.255.0, gig0000_ip: 10.1.0.1, gig0000_mask: 255.255.255.0,
gig0001_ip: 10.1.1.1, gig0001_mask: 255.255.255.0, gig0002_ip: 10.1.2.1,
gig0002_mask: 255.255.255.0, isisno: 12345, bgpas: 65123, peerip: 192.168.0.123,
peeras: 65321 }

interface_list_ler:
    - GigabitEthernet0/0/0/0
    - GigabitEthernet0/0/0/1
    - GigabitEthernet0/0/0/2

lsr_routers:
    - { hostname: lsr1_xr_cl2018, loopback0_ip: 192.168.0.2, loopback0_mask:
255.255.255.255, loopback1_ip: 192.168.1.2, loopback1_mask: 255.255.255.255,
loopback2_ip: 192.168.2.2, loopback2_mask: 255.255.255.255, mgmt_ip: 172.16.2.1,
mgmt_mask: 255.255.255.0, gig0000_ip: 10.1.0.2, gig0000_mask: 255.255.255.0,
gig0001_ip: 10.2.1.1, gig0001_mask: 255.255.255.0, gig0002_ip: 10.2.2.1,
gig0002_mask: 255.255.255.0, isisno: 12345 }

interface_list_lsr:
    - GigabitEthernet0/0/0/0
    - GigabitEthernet0/0/0/1
...
# vars file for ler-lsr
```

In the above exercise, you have two different variables defined for two roles LSR and LER but both are for a single router type. For expanding into larger scale, you can add more dictionaries under LSR or LER router for each device type.

Step 6: Execute the Playbook xr-hier.yml

```
cisco@Ansible-Controller:~/project1$ ansible-playbook xr-hier.yml

PLAY [Create a config for router1 from template XR]
****************************************************************************
*********************************************************************

TASK [Gathering Facts]
****************************************************************************
****************************************************************************
**********************
ok: [localhost]

TASK [pe-p : Generate the configuration for LER/PE Router1]
****************************************************************************
*************************************************************
changed: [localhost] => (item={u'gig0002_mask': u'255.255.255.0', u'peerip':
u'192.168.0.123', u'gig0000_mask': u'255.255.255.0', u'isisno': 12345,
```

```
u'loopback1_ip': u'192.168.1.1', u'mgmt_ip': u'172.16.1.1', u'loopback1_mask':
u'255.255.255.255', u'loopback2_mask': u'255.255.255.255', u'peeras': 65321,
u'hostname': u'ler1_xr_cl2018', u'gig0002_ip': u'10.1.2.1', u'loopback0_ip':
u'192.168.0.1', u'loopback2_ip': u'192.168.2.1', u'loopback0_mask':
u'255.255.255.255', u'mgmt_mask': u'255.255.255.0', u'gig0001_mask':
u'255.255.255.0', u'gig0000_ip': u'10.1.0.1', u'gig0001_ip': u'10.1.1.1',
u'bgpas': 65123})

TASK [pe-p : Generate the configuration for LSR/P Router2]
*********************************************************************
*****************************************************************
changed: [localhost] => (item={u'gig0002_mask': u'255.255.255.0',
u'gig0000_mask': u'255.255.255.0', u'isisno': 12345, u'loopback1_ip':
u'192.168.1.2', u'mgmt_ip': u'172.16.2.1', u'loopback1_mask':
u'255.255.255.255', u'loopback2_mask': u'255.255.255.255', u'hostname':
u'lsr1_xr_cl2018', u'gig0002_ip': u'10.2.2.1', u'loopback0_ip':
u'192.168.0.2', u'loopback2_ip': u'192.168.2.2', u'mgmt_mask':
u'255.255.255.0', u'gig0001_mask': u'255.255.255.0', u'gig0000_ip':
u'10.1.0.2', u'gig0001_ip': u'10.2.1.1', u'loopback0_mask':
u'255.255.255.255'})

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

You will observe that two different configurations for LSR/LER device types have been generated.

Step 7: Validate the router configurations are generated in the correct folder and explore the generated files.

```
cisco@Ansible-Controller:~/project1$ ls -al l?r*.txt
-rw-rw-r-- 1 cisco cisco 3440 Apr 26 02:25 ler1_xr_cl2018.txt
-rw-rw-r-- 1 cisco cisco 2439 Apr 26 02:25 lsr1_xr_cl2018.txt
```

Key Takeaways

- The objectives of this session are to introduce you to roles.
- Roles allow you to break a complex playbook into smaller files.
- Utilizing roles and Jinja J2 templates allows for a simple and quick way to generate configurations for large scale networks.
- The modularity of the hierarchical template prevents changes across templates for common configurations. As a result, a change in the base config will only need to be done once and it will be propagated to any other template which calls the base template.