Lab 12- End to End Configuration

Introduction

In this lab we're going to walk through a creating a playbook that will configure OSPF between two devices. In this case, we'll be configuring two Nexus 9000 switches that are connected back to back. Each student will be tasked with configuring a single interface on both devices. We're going to use a combination of methods you've learned so far to complete this playbook. At the end of this lab, we'll have a few challenges to further enhance your playbook.

1. Environment setup

1.1 Let's create a workspace for our end to end playbook.

Create a directory for end2end

```
mkdir ~/ansible_labs/lab12_end2end && cd ~/ansible_labs/lab12_end2end
```

1.2 Now let's create an ansible configuration and Inventory file

```
cat > ansible.cfg <<EOF

[defaults]

hostfile = inventory

host_key_checking = False

deprecation_warnings=False

EOF
```

1.3 Create an inventory file by adding different device
NOTE: Under the interface and IP address and OPSF ID variables, please replace with the correct variables for your POD.

POD	Interfaces	IP Addresses
1	Ethernet1/1	1.1.1.1/24 & 1.1.1.2/24
2	Ethernet1/2	2.2.2.1/24 & 2.2.2.2/24
3	Ethernet1/3	3.3.3.1/24 & 3.3.3.2/24
4	Ethernet1/4	4.4.4.1/24 & 4.4.4.2/24
5	Ethernet1/5	5.5.5.1/24 & 1.1.1.2/24
6	Ethernet1/6	6.6.6.1/24 & 6.6.6.2/24
7	Ethernet1/7	7.7.7.1/24 & 7.7.7.2/24
8	Ethernet1/8	8.8.8.1/24 & 8.8.8.2/24
9	Ethernet1/9	9.9.9.1/24 & 9.9.9.2/24
10	Ethernet1/10	10.10.10.1/24 & 10.10.10.2/24

```
cat > inventory <<EOF

n9k-standalone-02.localdomain ansible_ssh_host=10.1.150.14 a
nsible_ssh_user=admin ansible_ssh_pass=#cisco123

n9k-standalone-03.localdomain ansible_ssh_host=10.1.150.15 a
nsible_ssh_user=admin ansible_ssh_pass=#cisco123

csr1000v-pod-00.localdomain ansible_ssh_host=172.16.15.218 a
nsible_ssh_user=admin ansible_ssh_pass=!Cisco123
```

```
veos-pod-00.localdomain ansible_ssh_host=172.16.15.209 ansib
le_ssh_user=admin ansible_ssh_pass=!Cisco123

[network]

n9k-standalone-02.localdomain

n9k-standalone-03.localdomain

csr1000v-pod-00.localdomain

veos-pod-00.localdomain

[nxos]

n9k-standalone-02.localdomain

n9k-standalone-03.localdomain

[end2end]

n9k-standalone-02.localdomain interface=<Interface> ip_addre
ss=<IP Address> ospfid=<POD #>

n9k-standalone-03.localdomain interface=<Interface> ip_addre
ss=<IP Address> ospfid=<POD #>



 [csr]

csr1000v-pod-00.localdomain



[arista]

veos-pod-00.localdomain
```

```
[datacenter:children]
network

[datacenter:vars]
EOF
```

1.4 Create a **group_vars** directory and Create a different network yaml files for

```
mkdir group_vars && cd group_vars

cat > network.yml <<EOF
---
ansible_connection: network_cli
EOF

cat > arista.yml <<EOF
---
ansible_become: yes
ansible_network_os: eos
ansible_become_method: enable
ansible_become_pass: "!Cisco123"
```

```
EOF


cat > csr.yml <<EOF
---
ansible_become: yes
ansible_network_os: ios
ansible_become_method: enable
ansible_become_pass: "!Cisco123"
EOF


cat > 9k.yml <<EOF
---
ansible_network_os: nxos
EOF


cat > end2end.yml <<EOF
---
ansible_network_os: nxos
EOF
```

NOTE: Since we're connecting to different devices, we are using the group variables file to identify nuances for each. For example, the 9k group we're identifying what OS we are connecting to. For the CSR and Arista devices however, we need to also tell ansible that we need to connect to the device and enter enable mode before running any commands. The Ansible become command is used to accomplish this.

2. Build Playbook

2.1 Now that we have our inventory, config and variables set up, Let's build a playbook to configure two interfaces with an IP address, and then configure OSPF on two Nexus 9Ks

Create an **end2end.yaml** file

```
cd /~/ansible_labs/lab11_end2end



cat > end2end.yml <<EOF

---

- hosts: end2end



  tasks:

   - debug:

       msg: "{{ hostvars[inventory_hostname].interface }} {{
hostvars[inventory_hostname].ip_address }}"



   - name: Set IP Address on Interfaces

     nxos_l3_interface:

      name: "{{ hostvars[inventory_hostname].interface }}"
```

```
      ipv4: "{{ hostvars[inventory_hostname].ip_address }}"
      host: inventory_hostname

  - name: Bring up interfaces
    nxos_interface:
      name: "{{ hostvars[inventory_hostname].interface }}"
      admin_state: up

  - name: Test reachability to IP Addresses
    nxos_ping:
      dest: "{{ hostvars[inventory_hostname].ip_address[:-3] }}"
      state: present

  - name: Ensure ospf feature is turned on
    nxos_feature:
      feature: ospf
      state: enabled

  - name: Configure OSPF Instance
```

```
     nxos_ospf:
      ospf: "{{ hostvars[inventory_hostname].ospfid }}"
      state: present


   - name: push OSPF config to devices
     nxos_config:
        src: ./ospf-template.j2
        save_when: changed
EOF
```

2.2 Build the **ospf-template.j2** jinja template file

```
cat > ospf-template.j2 <<EOF
interface {{ hostvars[inventory_hostname].interface }}
{% if "Loopback" not in interface %}
    no switchport
    ip ospf network point-to-point
{% endif %}
    description OSPF connection pod 1
!
!
```

```
router ospf {{ hostvars[inventory_hostname].ospfid }}

    router-id {{ hostvars[inventory_hostname].ip_address[:-3
] }}

    network {{ hostvars[inventory_hostname].ip_address[:-3]
}} 0.0.0.0 area 0.0.0.0

EOF
```

3. Execute Playbook

3.1 Run the playbook by verify that syntax for **ansible-playbook end2end.yaml** file by running below command.

```
ansible-playbook –-syntax-check end2end.yml

ansible-playbook end2end.yml
```

Output:

```
PLAY [end2end] ***************************************************
**************************************************************
*********************

TASK [Gathering Facts] *******************************************
**************************************************************
*********************
ok: [n9k-standalone-03.localdomain]
ok: [n9k-standalone-02.localdomain]

TASK [debug] *****************************************************
**************************************************************
*********************
ok: [n9k-standalone-02.localdomain] => {
```

```
    "msg": "Ethernet1/2 2.2.2.1/24"
}
ok: [n9k-standalone-03.localdomain] => {
    "msg": "Ethernet1/2 2.2.2.2/24"
}

TASK [Set IP Address on Interfaces] ********************
**********************************************************
*********************
ok: [n9k-standalone-03.localdomain]
ok: [n9k-standalone-02.localdomain]

TASK [Test reachability to IP Addresses] ***************
**********************************************************
*********************
ok: [n9k-standalone-03.localdomain]
ok: [n9k-standalone-02.localdomain]

TASK [Ensure ospf feature is turned on] ****************
**********************************************************
*********************
ok: [n9k-standalone-02.localdomain]
ok: [n9k-standalone-03.localdomain]

TASK [Configure OSPF Instance] *************************
**********************************************************
*********************
changed: [n9k-standalone-03.localdomain]
changed: [n9k-standalone-02.localdomain]

TASK [push OSPF config to devices] *********************
**********************************************************
*********************
changed: [n9k-standalone-02.localdomain]
changed: [n9k-standalone-03.localdomain]
```

```
PLAY RECAP *********************************************
**************************************************
*********************
n9k-standalone-02.localdomain : ok=7    changed=2    unre
achable=0    failed=0
n9k-standalone-03.localdomain : ok=7    changed=2    unre
achable=0    failed=0
```

4. Validate Commands

SSH into the Nexus 9Ks and validate the configuration was pushed to each device. Please be aware that you are sharing these two devices with the rest of the students do you may see their configuration

```
ssh admin@n9k-standalone-02.localdomain

Show run
```

```
Interface Ethernet1/1

Description OSPF Connection to POD 1

IP Address 1.1.1.1/24

Ip ospf network point-to-point

Router ospf 1

  Router-id 1.1.1.1

  Network 1.1.1.1/32 area 0.0.0.0
```

```
ssh admin@n9k-standalone-03.localdomain

Show run
```

```
Interface Ethernet1/1

Description OSPF Connection to POD 1

IP Address 1.1.1.2/24

Ip ospf network point-to-point

Router ospf 1

  Router-id 1.1.1.2

  Network 1.1.1.2/32 area 0.0.0.0
```

5. Challenge

Feeling comfortable with playbooks? Lets see what you've learned so far! Now create a playbook to do the following:

Challenge 1:

Create a new playbook called challenge1.yml. Feel free to copy end2end.yml file to help you get started. This playbook should do the following:

1. Remove the interface IP from the interface assigned to you.
2. Administratively bring the interface down.
3. Remove the OSPF instance configuration.

Note: Do not disable the OSPF feature! This will affect other students doing the lab.

Once you finish this playbook, execute it and test it.

Challenge 2:

Create a new playbook called challenge2.yml. Feel free to copy end2end.yml file to help you get started. This playbook should do the following:

1. Create a VLAN 100X (X = POD number). For example, POD 3 is 1003
2. Configure your interface to be a switchport
3. Add this VLAN to your interface
4. Create an SVI interface for this VLAN.
5. Assign the IP address previously assigned to this interface. (HINT: the interface variable can be used in your inventory file).
6. Ping and validate connectivity

Challenge 3:

Create a new playbook called challenge3.yml. Feel free to copy end2end.yml file to help you get started. This playbook should do the following:

1. Enable HSRP between your newly created SVIs. Make sure the HSRP is authenticated (clear text is ok) and you have a VIP configured. Please use .10 for your VIP address (for example: 3.3.3.10/24). Make sure n9k-standalone-02.localdomain becomes active! (hint: you can create new variables in your inventory file).
2. Ping the VIP and validate connectivity (a delay may be necessary. Add a wait timer OR rerun the playbook)
3. Create a new OSPF instance but this time create a VRF. Use the same OSPF instance ID with a VRF of PODX (X = POD number)
4. Add authentication to the SVI for OSPF. Clear text is ok.
5. Use a similar jinja template to configure the OSPF adjacency