Lab 2 – Ad-hoc Commands

Introduction

In this lab we're going to start by running ad-hoc commands in Ansible. Ad-hoc is a great to become familiar with certain modules prior to adding them into your playbooks (discussed later). We'll be using a variety of modules geared towards Arista, Cisco IOS and Cisco NXOS.

Please refer to the **Ansible-pod-info.docx** file for information on connecting to your Ansible host.

1. Ad-hoc setup

1.1 Let's create a workspace for our Ad-hoc commands. Please note, some of the things we're going to create will be discussed later in this course.

Make sure you are in the **lab2-adhoc folder** for this lab.

```
cd ~/ansible_labs/lab2-ad-hoc
```

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

Copy and paste the following to create an Ansible Config file. You can also create this on your own in a file editor like vim.

```
cat > ansible.cfg <<EOF

[defaults]

hostfile = inventory

host_key_checking = False

deprecation_warnings=False

EOF
```

Copy and paste the following to create an inventory file. “XX” in the below file needs to be modified based on your Pod number. Also, update the IPs with the information from your Pod sheet.

```
cat > inventory <<EOF

n9k-standalone-XX.localdomain ansible_ssh_host=10.1.150.13 ansible_ssh_user=a
dmin ansible_ssh_pass=Cisco123

csr1000v-pod-XX.localdomain ansible_ssh_host=172.16.15.218 ansible_ssh_user=a
dmin ansible_ssh_pass=Cisco123

veos-pod-XX.localdomain ansible_ssh_host=172.16.15.209 ansible_ssh_user=admin
ansible_ssh_pass=Cisco123

[network]

n9k-standalone-XX.localdomain

csr1000v-pod-XX.localdomain

veos-pod-XX.localdomain

[9k]

n9k-standalone-XX.localdomain

[csr]

csr1000v-pod-XX.localdomain



[arista]

veos-pod-XX.localdomain

[datacenter:children]

network

EOF
```

Create Group_vars directory

```
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
```

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. Ad-hoc Commands on NXOS

2.1 Now that we have our inventory, config and variables set up, lets start running some ad-hoc commands. Ad-hoc is a great way to test out an ansible module prior to creating a playbook. We have 3 devices in the lab that we'll be running commands against. We have a Cisco Nexus 9K (NXOS), a Cisco CSR (IOS) and a Arista Router (EOS). We'll start by running commands against our NXOS device.

Execute Ad-Hoc commands against NXOS device

```
ansible -m nxos_command  -u admin -k -a "commands='show vlan'" 9k
```

Example of output:

```
n9k-standalone-01.localdomain | SUCCESS => {

    "changed": false,

    "stdout": [

        "VLAN Name                             Status    Ports\n---- --------
------------------------ --------- ------------------------------\n1    defa
ult                          active    Eth1/4, Eth1/5, Eth1/6, Eth1/7\n
Eth1/8, Eth1/9, Eth1/10, Eth1/11\n
Eth1/12, Eth1/13, Eth1/14\n                                              Et
h1/15, Eth1/16, Eth1/17\n                                               Eth1
/18, Eth1/19, Eth1/20\n                                                Eth1/2
1, Eth1/22, Eth1/23\n                                                 Eth1/24,
Eth1/25, Eth1/26\n                                                   Eth1/27, Et
```

```
h1/28, Eth1/29\n                                                    Eth1/33, Eth1
/34, Eth1/35\n                                                    Eth1/36, Eth1/3
7, Eth1/38\n                                                    Eth1/39, Eth1/40,
Eth1/41\n                                                    Eth1/42, Eth1/43, Et
h1/44\n                                                    Eth1/45, Eth1/46, Eth1
/47\n                                                    Eth1/48, Eth1/49, Eth1/5
0\n                                                    Eth1/51, Eth1/54\n100  web
active    Eth1/30\n101  app                                    active    Eth1/31\n1
02  storage                         active    Eth1/32\n1001 VLAN1001
active    \n2000 VLAN2000                            active    \n\nVLAN Type
Vlan-mode\n---- -----        ----------\n1    enet         CE     \n100  enet
CE     \n101  enet         CE     \n102  enet         CE     \n1001 enet
CE     \n2000 enet         CE     \n\nRemote SPAN VLANs\n-------------------
-------------------------------------------------------\n\nPrimary  Secon
dary  Type             Ports\n-------  ---------  ---------------  ----------
---------------------------------"

    ],

    "stdout_lines": [

        [

            "VLAN Name                             Status    Ports",

            "---- -------------------------------- --------- ----------------
---------------",

            "1    default                          active    Eth1/4, Eth1/5,
Eth1/6, Eth1/7",

            "                                                Eth1/8, Eth1/9,
Eth1/10, Eth1/11",

            "                                                Eth1/12, Eth1/13
, Eth1/14",
```


Create a loopback interface and bring it up

```
ansible -m nxos_interface -u admin -a "name='loopback100',admin_state='up'" 9
k
```

Example output:

```
n9k-standalone-01.localdomain | SUCCESS => {

    "changed": true,

    "commands": [

        "interface loopback100",

        "no shutdown"

    ]

}
```

Configure OSPF on NXOS

```
ansible -m nxos_ospf -u admin -c nxapi -a "ospf='10'" 9k
```

Example Output:

```
n9k-standalone-01.localdomain | SUCCESS => {

    "changed": false,

    "commands": []

}
```

3. Ad-hoc Commands on IOS

3.1 Now let's run a few commands on an IOS device. In this case, this will be the Cisco CSR in our lab.

Execute Ad-Hoc commands against IOS device

Show Version command on IOS

```
ansible -m ios_command  -u admin -a "commands='show version'" csr
```

Example Output:

```
csr1000v-pod-00.localdomain | SUCCESS => {

    "changed": false,

    "stdout": [

        "Cisco IOS XE Software, Version 03.16.06b.S - Extended Support Releas
e\nCisco IOS Software, CSR1000V Software (X86_64_LINUX_IOSD-UNIVERSALK9-M), V
ersion 15.5(3)S6b, RELEASE SOFTWARE (fc4)\nTechnical Support: http://www.cisc
o.com/techsupport\nCopyright (c) 1986-2017 by Cisco Systems, Inc.\nCompiled T
hu 02-Nov-17 10:49 by mcpre\n\n\nCisco IOS-XE software, Copyright (c) 2005-20
17 by cisco Systems, Inc.\nAll rights reserved.  Certain components of Cisco
IOS-XE software are\nlicensed under the GNU General Public License (\"GPL\")
Version 2.0.  The\nsoftware code licensed under GPL Version 2.0 is free softw
are that comes\nwith ABSOLUTELY NO WARRANTY.  You can redistribute and/or mod
ify such\nGPL code under the terms of GPL Version 2.0.  For more details, see
the\ndocumentation or \"License Notice\" file accompanying the IOS-XE softwar
e,\nor the applicable URL provided on the flyer accompanying the IOS-XE\nsoft
ware.\n\n\nROM: IOS-XE ROMMON\n\nCSR1000v-Pod-00 uptime is 2 hours, 14 minute
s\nUptime for this control processor is 2 hours, 15 minutes\nSystem returned
to ROM by reload\nSystem image file is \"bootflash:packages.conf\"\nLast relo
ad reason: <NULL>\n\n\n\nThis product contains cryptographic features and is
subject to United\nStates and local country laws governing import, export, tr
ansfer and\nuse. Delivery of Cisco cryptographic products does not imply\nthi
rd-party authority to import, export, distribute or use encryption.\nImporter
s, exporters, distributors and users are responsible for\ncompliance with U.S
. and local country laws. By using this product you\nagree to comply with app
licable laws and regulations. If you are unable\nto comply with U.S. and loca
l laws, return this product immediately.\n\nA summary of U.S. laws governing
Cisco cryptographic products may be found at:\nhttp://www.cisco.com/wwl/expor
t/crypto/tool/stqrg.html\n\nIf you require further assistance please contact
us by sending email to\nexport@cisco.com.\n\nLicense Level: ax\nLicense Type:
Default. No valid license found.\nNext reload license Level: ax\n\ncisco CSR1
000V (VXE) processor (revision VXE) with 2055936K/6147K bytes of memory.\nPro
cessor board ID 96NCEYKC3PH\n3 Gigabit Ethernet interfaces\n32768K bytes of n
on-volatile configuration memory.\n3988292K bytes of physical memory.\n777420
7K bytes of virtual hard disk at bootflash:.\n\nConfiguration register is 0x2
102"
```

```
    ],

    "stdout_lines": [
```

Gather Facts on IOS Device

```
ansible -m ios_facts  -u admin -a "gather_subset='all'" csr
```

Example Output:

```
 csr1000v-pod-00.localdomain | SUCCESS => {

    "ansible_facts": {

        "ansible_net_all_ipv4_addresses": [

            "172.16.15.218"

        ],

        "ansible_net_all_ipv6_addresses": [],

        "ansible_net_config": "Building configuration...\n\nCurrent configura
tion : 1379 bytes\n!\n! Last configuration change at 19:01:30 UTC Tue Sep 4 2
018\n!\nversion 15.5\nservice timestamps debug datetime msec\nservice timesta
mps log datetime msec\nno platform punt-keepalive disable-kernel-core\nplatfo
rm console virtual\n!\nhostname CSR1000v-Pod-00\n!\nboot-start-marker\nboot-e
nd-marker\n!\n!\nenable secret 5 $1$yi5n$DdPlzdtgxHTm3bJmuTPkf0\nenable passw
ord !Cisco123\n!\nno aaa new-model\n!\n!\n!\n!\n!\n!\n!\n!\n!\n!\n!\n!\n!\n\n\nip
domain name cisco\n!\n!\n!\n!\n!\n!\n!\n!\n!\n!\nsubscriber templating\n!\nmu
ltilink bundle-name authenticated\n!\n!\n!\n!\n!\n!\n!\n!\n!\n!\n!\n!\n!\n!\nlic
ense udi pid CSR1000V sn 96NCEYKC3PH\n!\nspanning-tree extend system-id\n!\nu
sername admin secret 5 $1$z1pJ$YixfpjjFUE68QsjIDImid1\n!\nredundancy\n!\n!\n!
\n!\n!\n!\n! \n!\n!\n!\n!\n!\n!\n!\n!\n!\n!\n!\n!\n! \n! \n! \n! \n! \n!\n!\n!\
ninterface GigabitEthernet1\n ip address 172.16.15.218 255.255.0.0\n negotiat
ion auto\n!\ninterface GigabitEthernet2\n no ip address\n shutdown\n negotiat
ion auto\n!\ninterface GigabitEthernet3\n no ip address\n shutdown\n negotiat
ion auto\n!\n!\nvirtual-service csr_mgmt\n!\nip forward-protocol nd\n!\nno ip
http server\nno ip http secure-server\nip tftp source-interface GigabitEthern
et1\nip route 0.0.0.0 0.0.0.0 GigabitEthernet1 172.16.0.1\nip ssh version 2\n
!\n!\n!\n!\ncontrol-plane\n!\n !\n !\n !\n !\n!\n!\n!\n!\nline con 0\n sto
```

```
pbits 1\nline vty 0\n login local\n transport input ssh\nline vty 1 4\n login
local\n length 0\n transport input ssh\n!\nntp server pool.ntp.org\n!\nend",

        "ansible_net_filesystems": [

            "bootflash:"

        ],
```

Perform Basic Ping Test

```
ansible -m ios_ping  -u admin -a "dest='10.1.1.1'" csr
```

Example Output:

```
csr1000v-pod-00.localdomain | SUCCESS => {

    "changed": false,

    "commands": [

        "ping 10.1.1.1"

    ],

    "packet_loss": "20%",

    "packets_rx": 4,

    "packets_tx": 5,

    "rtt": {

        "avg": 1,

        "max": 2,

        "min": 1

    }
```

```
}
```

4. Ad-hoc Commands on EOS

4.1 Now we'll run a few commands on an Arista EOS device

Execute Ad-Hoc commands against EOS device

Show Version command on EOS

```
ansible -m ios_command  -u admin -a "commands='show version'" arista
```

Example Output:

```
veos-pod-00.localdomain | SUCCESS => {

    "changed": false,

    "stdout": [

        "Arista vEOS\nHardware version:    \nSerial number:       \nSystem MA
C address:  000c.29dd.cd6b\n\nSoftware image version: 4.20.7M\nArchitecture:
i386\nInternal build version: 4.20.7M-8944203.4207M\nInternal build ID:
d28d91e2-20a0-4846-91c7-f3c2158211e9\n\nUptime:                 6 weeks, 4 da
ys, 17 hours and 27 minutes\nTotal memory:           4010988 kB\nFree memory:
3137020 kB"

    ],

    "stdout_lines": [

        [

            "Arista vEOS",

            "Hardware version:    ",

            "Serial number:       ",
```

```
            "System MAC address:  000c.29dd.cd6b",
            "",
            "Software image version: 4.20.7M",
            "Architecture:           i386",
            "Internal build version: 4.20.7M-8944203.4207M",
            "Internal build ID:      d28d91e2-20a0-4846-91c7-f3c2158211e9",
            "",
            "Uptime:                 6 weeks, 4 days, 17 hours and 27 minutes
",
            "Total memory:           4010988 kB",
            "Free memory:            3137020 kB"
        ]
    ]
```

Create Loopback interface on EOS Device

```
ansible -m eos_interface -a "name='loopback 100'" arista
```

Example Output:

```
veos-pod-00.localdomain | SUCCESS => {
    "changed": true,
    "commands": [
        "interface loopback 100"
    ],
```

```
    "session_name": "ansible_1536097252"

}
```

Now commit your files to your repo. Reference the Git lab if you are unsure on the process for this.

More information on help with parameters that can be used with specific modules can be found using the below.

ansible-doc <module_name>

Challenge question:

Go to Ansible documentation (https://docs.ansible.com) find the modules eos_vlan.