Network Automation using Ansible

Version 1.0

July 2018

Contents

[Environment setup 3](#_Toc520475848)

[Network Automation using Ansible raw module 8](#_Toc520475849)

[Automation Cisco Nexus devices using Ansible 11](#_Toc520475850)

[Perform Clean up activity on nexus device 15](#_Toc520475851)

[Automating Cisco CSR1000v using Ansible 17](#_Toc520475852)

[Automation Arista EOS using Ansible 21](#_Toc520475853)

[Perform Cleanup activity on the EOS Device using Ansible 26](#_Toc520475854)

[Appendix A 30](#_Toc520475855)

[Running Ansible environment setup script 30](#_Toc520475856)

**Note:** We will be using Ansible version 2.4.2.0 for this LAB due to some existing bugs on network\_cli module. We will be leveraging one of the Managed Nodes VM to install the Ansible 2.4

We are not going to install Ansible as package instead we will be cloning the Ansible github repository and switching out to the desired version.

# Environment setup

1. Login as root use on ssh [root@podx-node1.origin.com](mailto:root@podx-node1.origin.com)
2. Execute the following command to Install Python-pip

# subscription-manager repos --enable rhel-server-rhscl-7-rpms

# yum install python27-python-pip -y

# scl enable python27 bash

**Note:** Since our LAB RHEL subscription is expired, we cannot use the Yum to install the python-pip, we can use a workaround to install the python pip

Please follow the steps from this link – <https://linuxconfig.org/installation-of-pip-the-python-packaging-tool-on-rhel-7-linux>

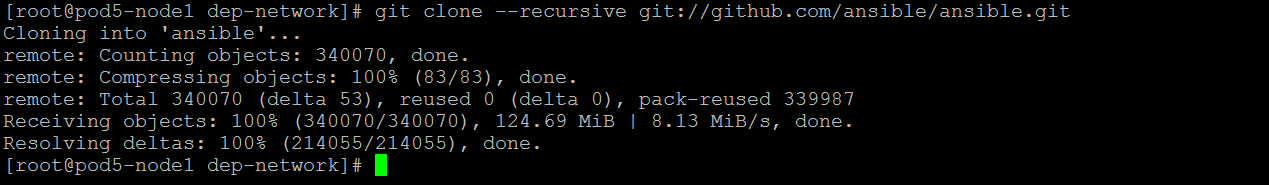
For the actual Training we will use the above Yum install method to install the packages

1. Create a directory for setting up Ansible

mkdir ~/lab11-dep-network && cd ~/lab11-dep-network

1. Clone the latest ansible repo from GitHub

git clone --recursive git://github.com/ansible/ansible.git



1. Cd to ansible directory

cd ./ansible

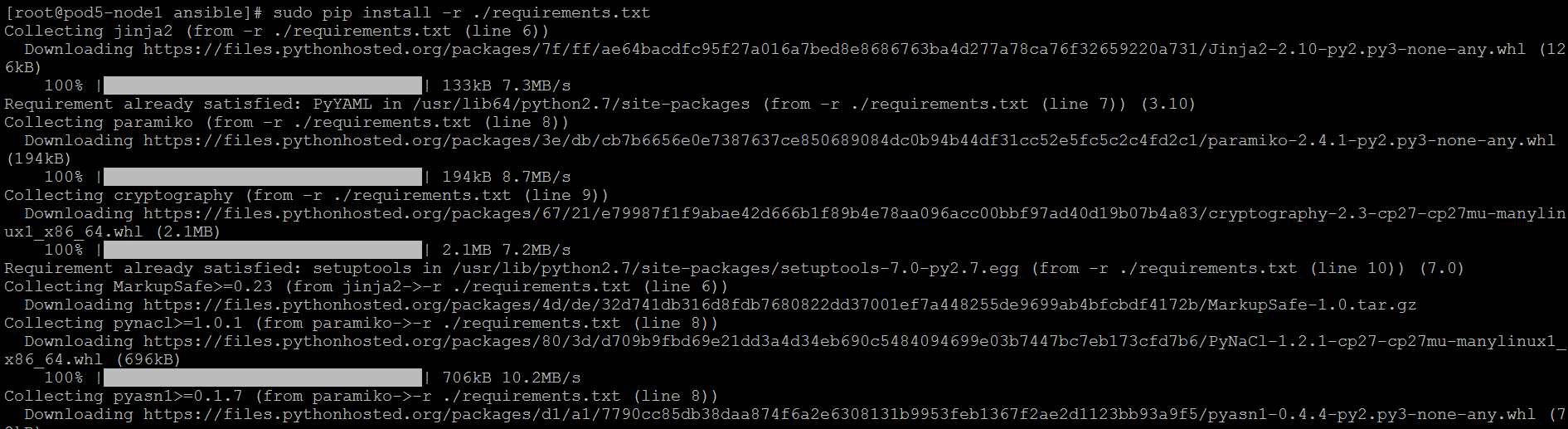
1. Execute the env-setup script

source ./hacking/env-setup



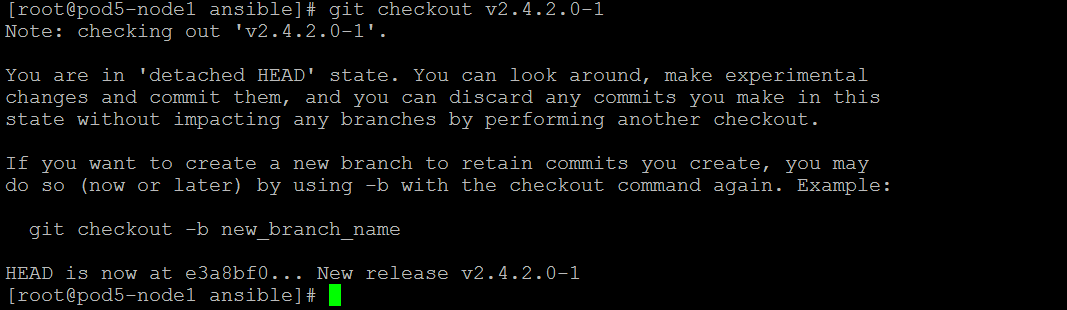
1. Install the Requirements

sudo pip install -r ./requirements.txt



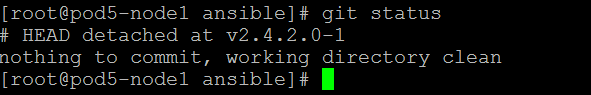
1. Use Git tag to Checkout to the branch v2.4.2.0-1

git checkout v2.4.2.0-1



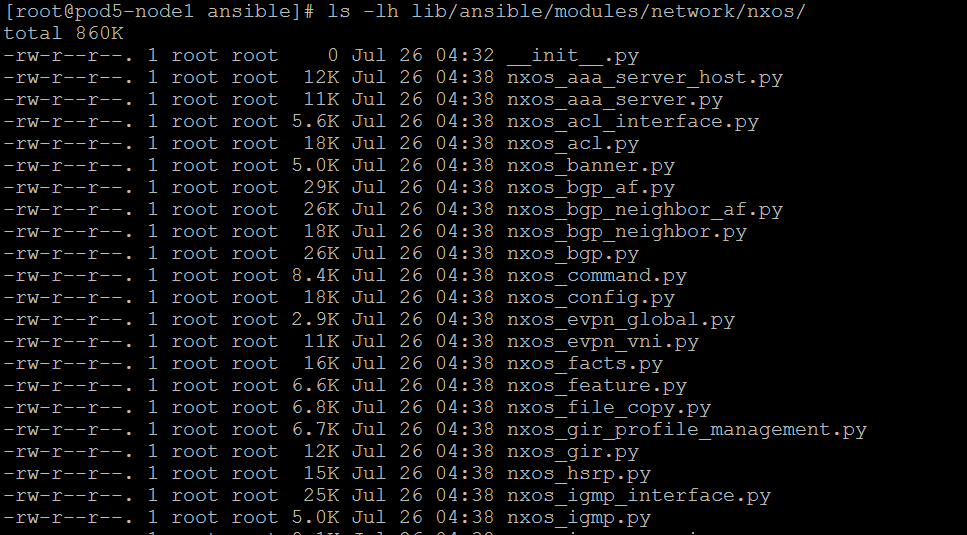
1. Use git status command to ensure you in the correct version

git status



1. List available Ansible NXOS modules.

ls -lh lib/**ansible**/modules/network/nxos/



**Note:** In the Environment setup lab 1 we need to add the POD specific network devices IP address and hostnames to add to the host file 🡪 I will let Naushad and Nagaveni to add this

For this test, let’s add the following entries manually in the host file

cat > /etc/hosts <<EOF

127.0.0.1  localhost

172.16.120.15 pod5-master.origin.com pod5-master

172.16.120.25 pod5-node1.origin.com pod5-node1

172.16.120.35 pod5-node2.origin.com pod5-node2

10.1.150.95 pod5-nxos1

172.16.15.218 pod5-ios1

172.16.15.209 pod5-eos1

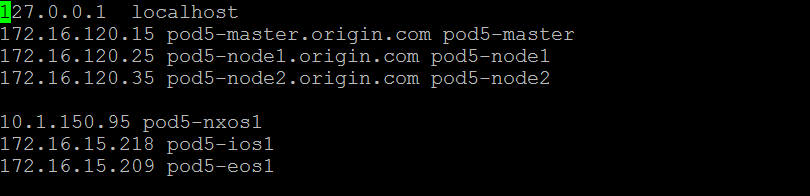
EOF

**POD0 network device**

10.1.150.95 pod5-nxos1

172.16.15.218 pod5-ios1

172.16.15.209 pod5-eos1



1. Create the managed nodes inventory file for the network devices as follows

cat >> ./inventory <<EOF

[all:vars]

un\_nxos = admin

pwd\_nxos = #cisco123

un\_eos = admin

pwd\_eos = !Cisco123

un\_ios = admin

pwd\_ios = !Cisco123

[nxos]

pod5-nxos1

[eos]

pod5-eos1

[ios]

pod5-ios1

EOF

1. Create the **ansible.cfg** file in the present working directory with the following content

cat > ansible.cfg <<EOF

[defaults]

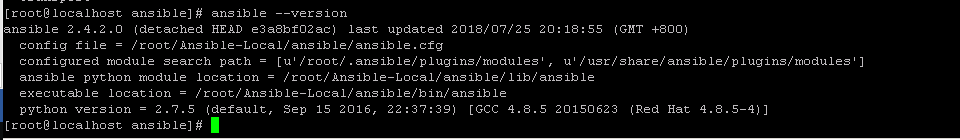
host\_key\_checking = False

log\_path = /var/log/ansible.log

inventory = ./inventory

EOF

1. Verify the configuration file by running “ansible –version”



# Network Automation using Ansible raw module

Using the **raw\_module** we can execute a direct SSH commands on a target device without going through the module subsystem of ansible.

1. Here is an example of ansible raw command to get the running configuration of a network device

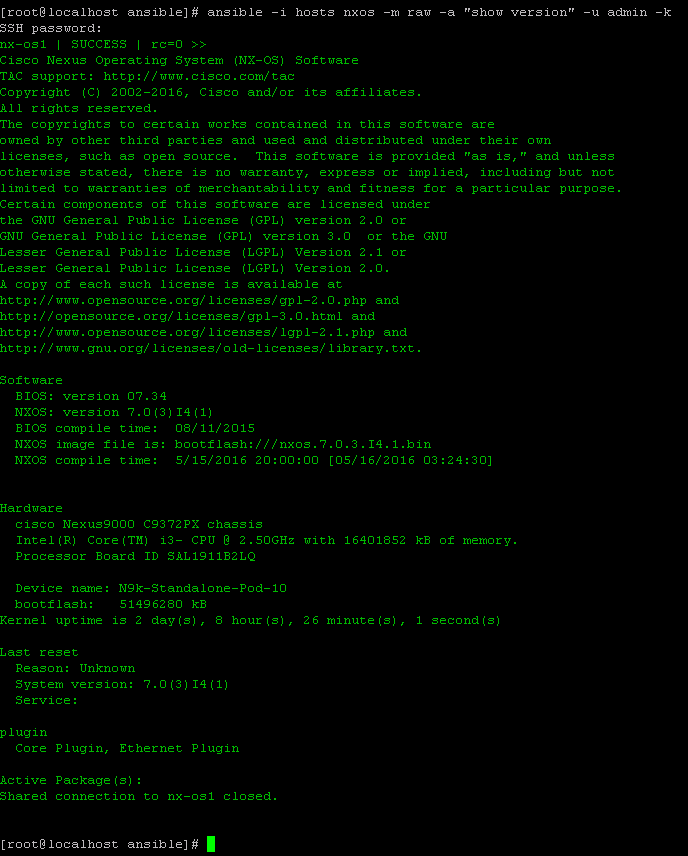
#ansible -i hosts nxos -m raw -a "show version" -u admin -k

1. Make sure the **sshpass** package is installed on the node, if it’s not installed please the steps below to install it

# wget <http://download.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm>

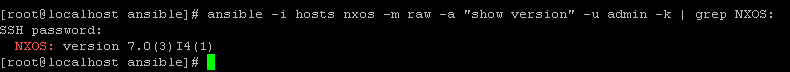
# rpm –ivh epel-release-6-8.noarch.rpm

# yum –enablerepo=epel –y install sshpass



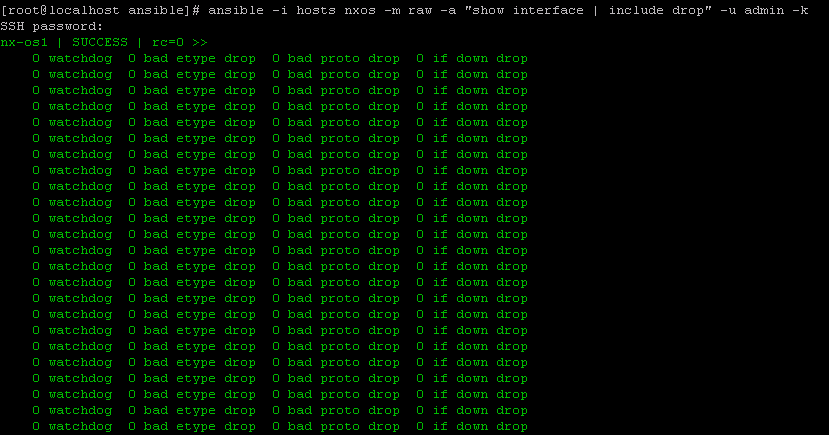
1. You can manipulate the output with the grep commands and redirect it to a file etc.
2. Execute the followong command to grep only the **NXOS version**

ansible -i hosts nxos -m raw -a "show version" -u admin -k | grep NXOS:



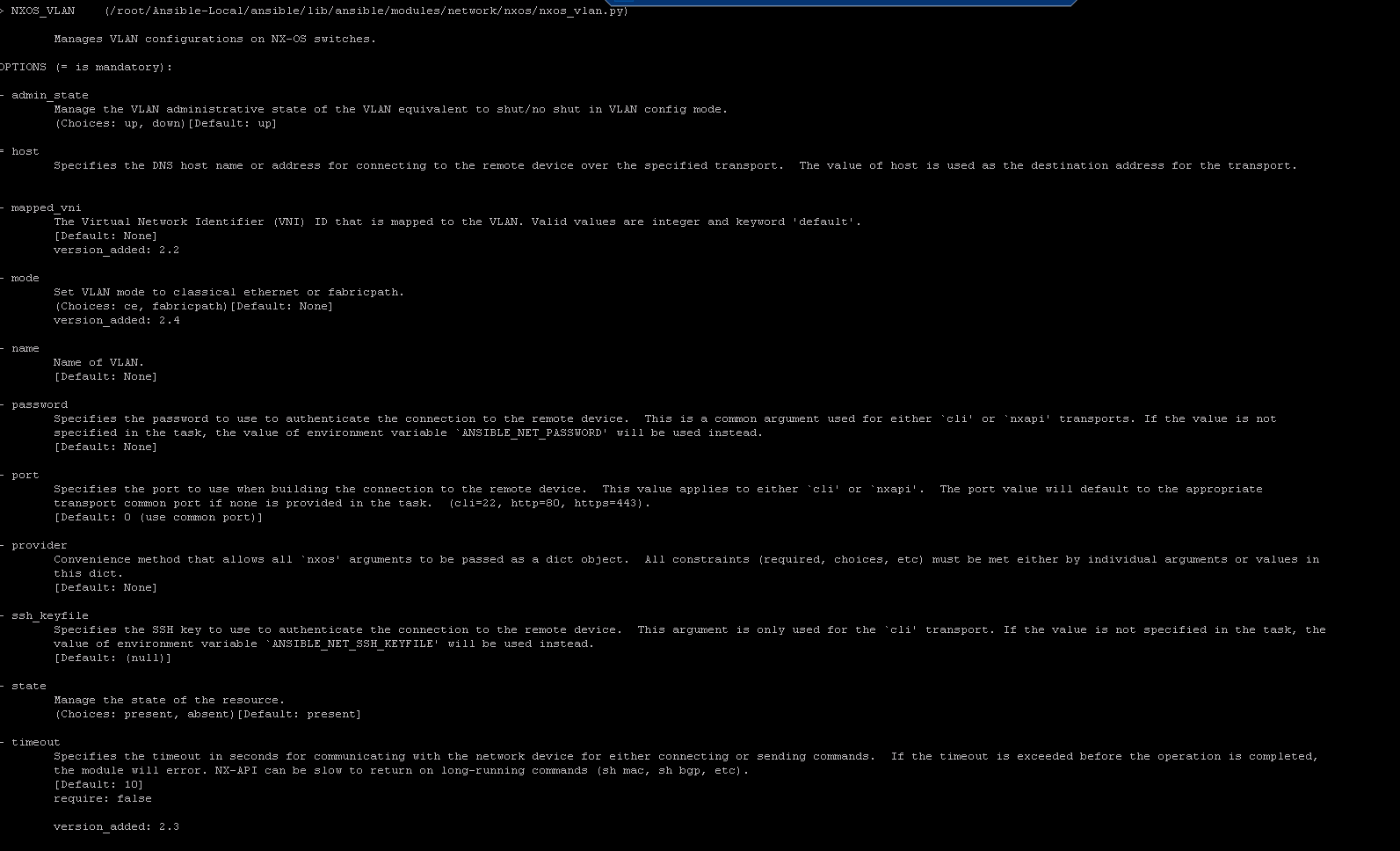
1. You also use the pipe | statement in the command to get some useful network stats.
2. Execute the following command to get the **list of packet drops**

ansible -i hosts nxos -m raw -a "show interface | include drop" -u admin -k



# Automation Cisco Nexus devices using Ansible

1. We will be using the **nxos\_vlan** module for this activity. Use **ansible-doc nxos\_vlan** to explore the available parameters and options.



1. Press letter “Q” to exit the documentation page
2. Create a Playbook to configure the following on the Nexus device

* Check if the vlan doesn’t exists
* Create the vlan
* Verify the VLAN is created

1. You can manually create the playbook by referring the ansible-doc or use the following command to create the **nxos\_vlan\_add.yml**

cat > /root/dep-network/nxos\_vlan\_add.yml <<EOF

---

- name: Create VLAN's across NX-OS based switches

hosts: nxos

connection: local

gather\_facts: no

vars:

provider:

username: "{{ un\_nxos }}"

password: "{{ pwd\_nxos }}"

transport: nxapi

host: "{{ inventory\_hostname }}"

tasks:

- name: Adding VLAN using NXOS module "nxos\_vlan"

nxos\_vlan:

vlan\_id: 210

name: Ansible-Added-VLAN-POD5

provider: "{{ provider }}"

- name: Ensure the vlan is created

nxos\_vlan:

vlan\_id: 210

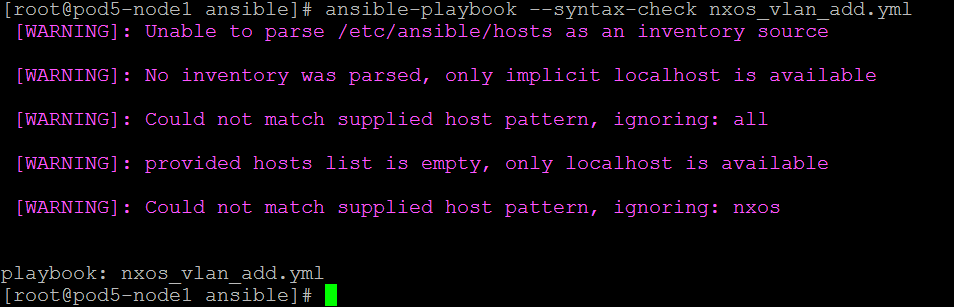
state: present

provider: "{{ provider }}"

EOF

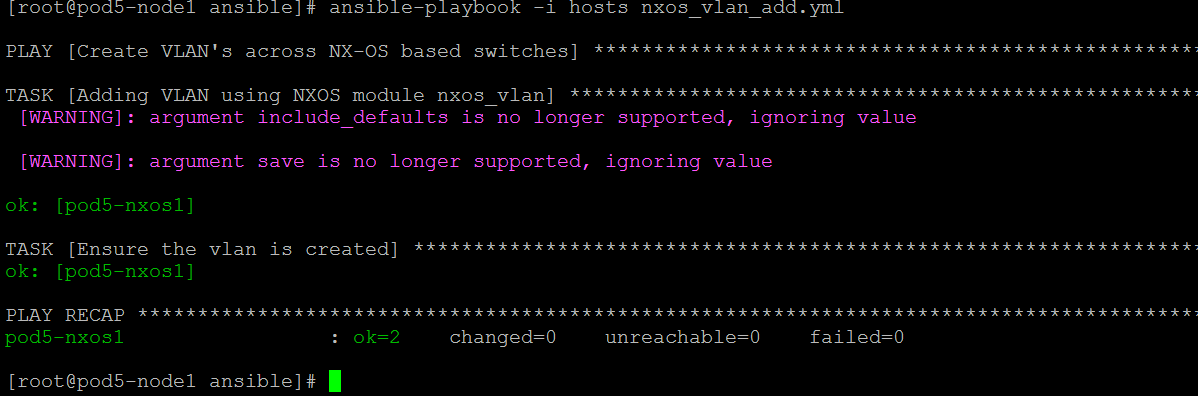
1. Validate the playbook syntax by executing the following command

ansible-playbook --syntax-check nxos\_vlan\_add.yml



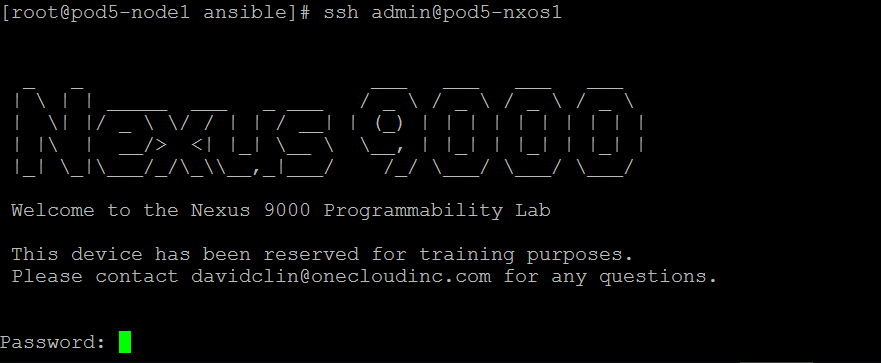
1. Ignore the warnings
2. Execute the playbook and verify the output

ansible-playbook -i hosts nxos\_vlan\_add.yml



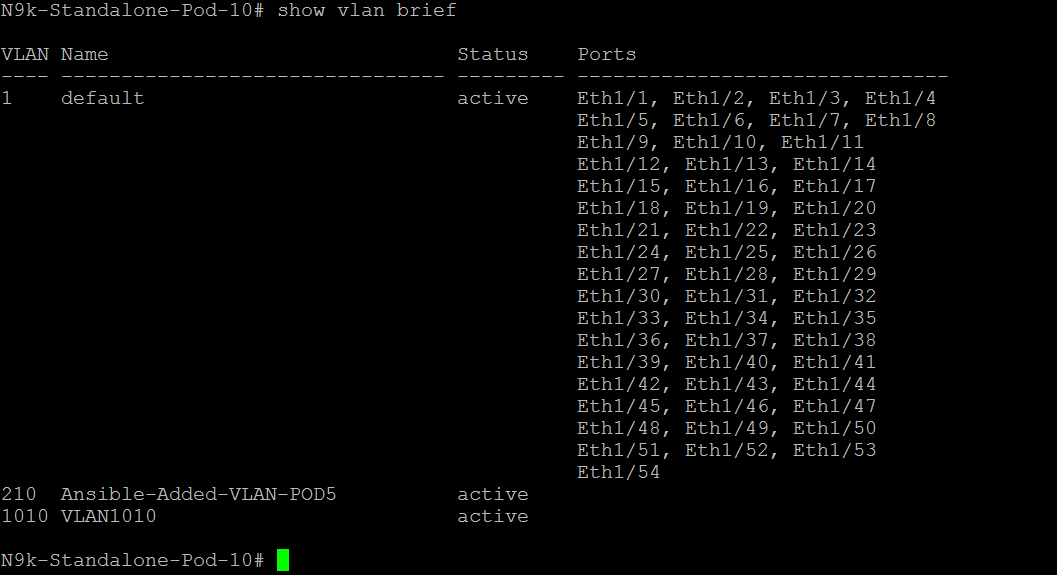
1. Login to the Nexus device as **admin/#cisco123** and verify the configuration

#ssh admin@pod5-nxos1



1. Execute **“show vlan brief”** to list the VLANS

N9k-Standalone-Pod-10# show vlan brief



1. Verify the VLAN is created and exit from the nexus device shell
2. Type “exit” to exit the device ssh shell
3. Explore the other available **nxos** modules and create your own playbook to **configure interface, login banner etc.**

## Perform Clean up activity on nexus device

1. Create a play book to delete the VLAN added in the previous activity, use the nxos\_vlan to create the paly,
2. Refer the **ansible-doc nxos\_vlan** documentation to get the parameter and options
3. You can manually create the playbook by referring the ansible-doc or use the following command to create the **nxos\_vlan\_remove.yml**

cat > /root/dep-network/nxos\_vlan\_remove.yml <<EOF

---

- name: Delete VLAN's across NX-OS based switches

hosts: nxos

connection: local

gather\_facts: no

vars:

provider:

username: "{{ un\_nxos }}"

password: "{{ pwd\_nxos }}"

transport: nxapi

host: "{{ inventory\_hostname }}"

tasks:

- name: Deleting VLAN using NXOS module "nxos\_vlan"

nxos\_vlan:

provider: "{{ provider }}"

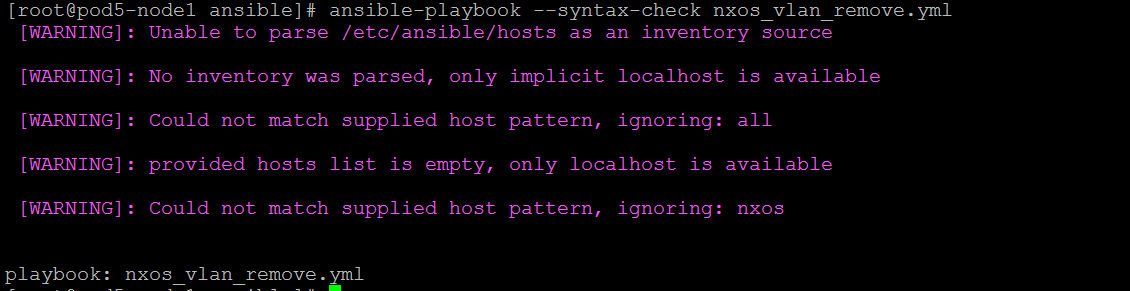
vlan\_id: 210

state: absent

EOF

1. Validate the syntax and execute the playbook

ansible-playbook --syntax-check nxos\_vlan\_remove.yml



1. Ignore the warnings and execute the playbook

ansible-playbook -i hosts nxos\_vlan\_remove.yml

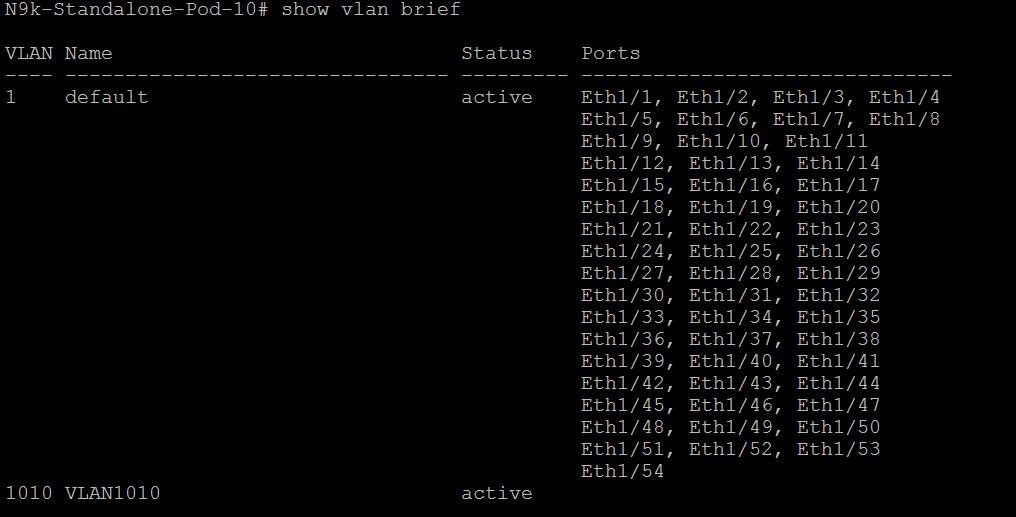


1. SSH to the network device and verify the VLAN is removed

ssh admin@pod5-nxos1

1. Execute show vlan brief and verify the VLAN is removed from the device

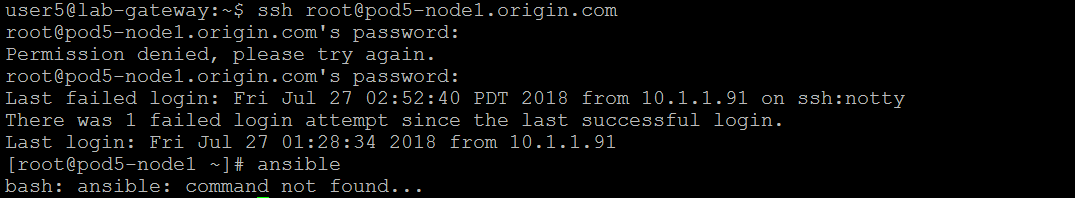
N9k-Standalone-Pod-10# show vlan brief



1. Verify the VLAN is removed and exit from the device SSH shell
2. Type “exit” to exit from the device SSH shell

# Automating Cisco CSR1000v using Ansible

1. Login to the POD5 node1 VM from the SSH gateway
2. Login as root use on ssh [root@podx-node1.origin.com](mailto:root@podx-node1.origin.com) with password !cisco123



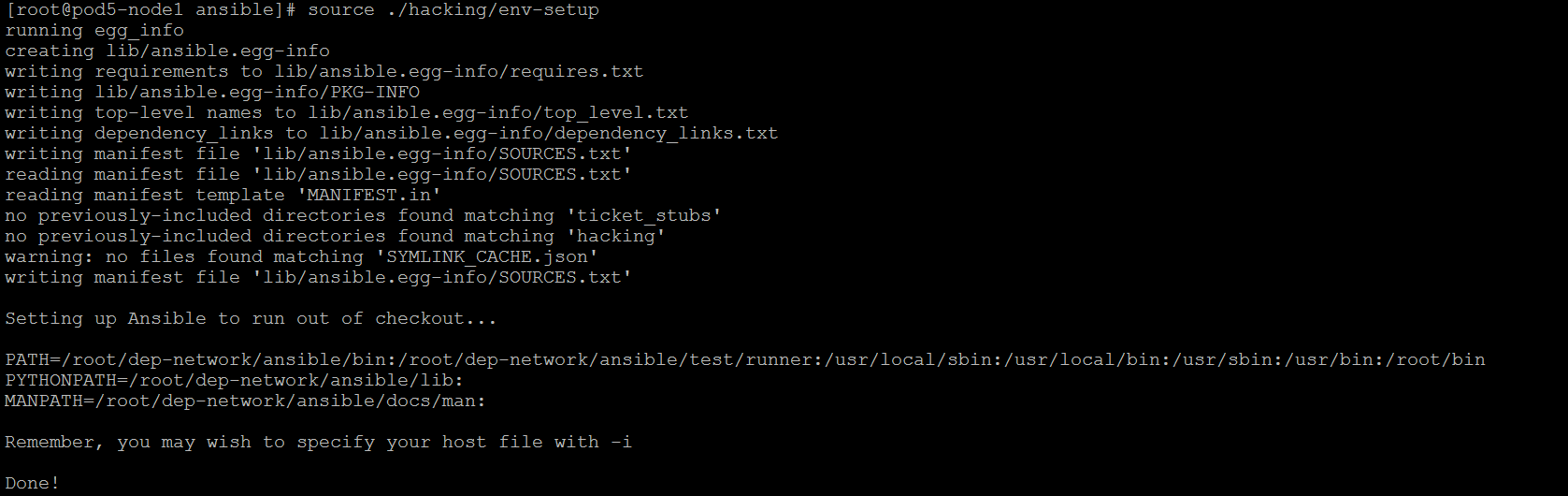
**Note:** Since we have not installed the Ansible as package on the Node VM, the executable will not work when the SSH session is re-established. You need to run the environment setup script on every login.

1. Change to **“dep-network/ansible/”** directory

cd dep-network/ansible/

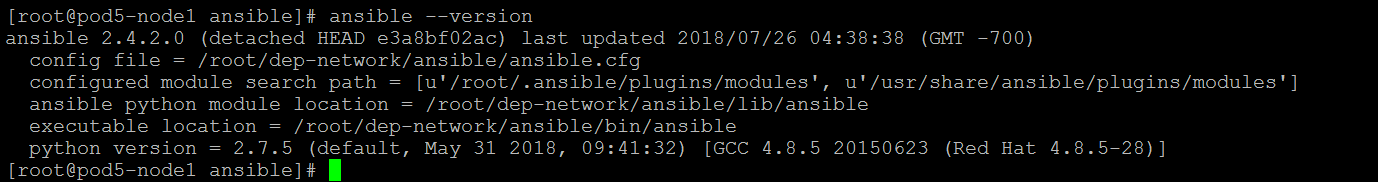
1. Execute the **env-setup script** to setup the environment

[root@pod5-node1 ansible]# source ./hacking/env-setup



1. Verify the ansible binary is executable, Enter **ansible --version** to verify

[root@pod5-node1 ansible]# ansible --version



1. Create a playbook to configure the following on the CSR100v

* Add a ACL “access-list 99 permit 172.16.0.254”
* Verify the ACL is added
* Save the configuration

1. We will use **ios\_command** and **ios\_config** modules to automation the above mentioned configuration
2. Explore the module parameters and option using ansible-doc <module name> command
3. You can manually create the playbook by referring the ansible-doc or use the following command to create the playbook

cat > /root/dep-network/ios\_add\_acl.yml <<EOF

---

- name: Manage ISO Device

hosts: ios

connection: local

gather\_facts: no

vars:

provider:

username: "{{ un\_ios }}"

password: "{{ pwd\_ios }}"

host: "{{ inventory\_hostname }}"

tasks:

- name: Configure ACL on Cisco CSR 1000v

ios\_config:

auth\_pass: "{{ pwd\_ios }}"

authorize: yes

provider: "{{ provider }}"

lines:

- access-list 99 permit 172.16.0.254

- name: Verify ACL is present

ios\_command:

auth\_pass: "{{ pwd\_ios }}"

authorize: yes

provider: "{{ provider }}"

commands:

- sh access-l

waitfor:

- result[0] contains 'permit 172.16.0.254'

- name: Save config

ios\_config:

auth\_pass: "{{ pwd\_ios }}"

authorize: yes

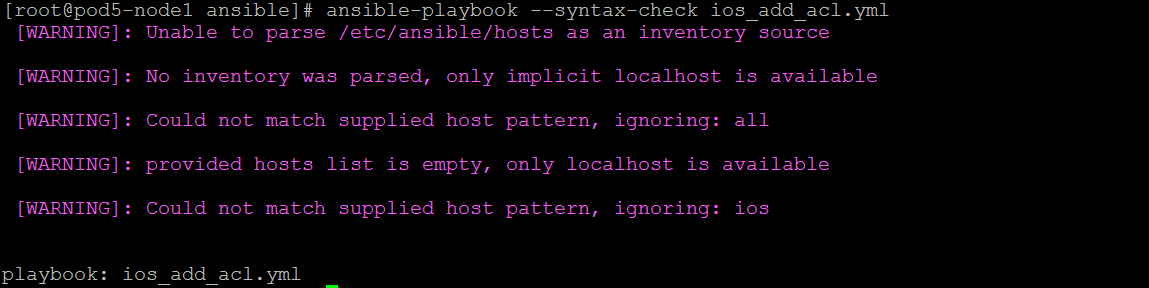
provider: "{{ provider }}"

save\_when: modified

EOF

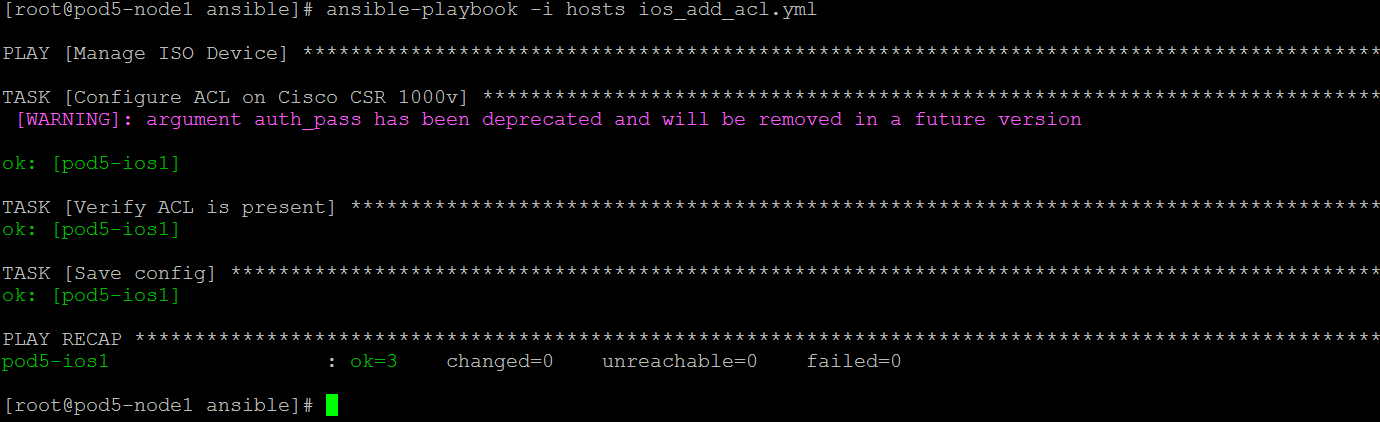
1. Validate the syntax

ansible-playbook --syntax-check ios\_add\_acl.yml



1. Ignore the warnings and execute the playbook

ansible-playbook -i hosts ios\_add\_acl.yml



1. Login to the CSR1000v ssh console as (admin/!Cisco123) and verify the ACL are configured

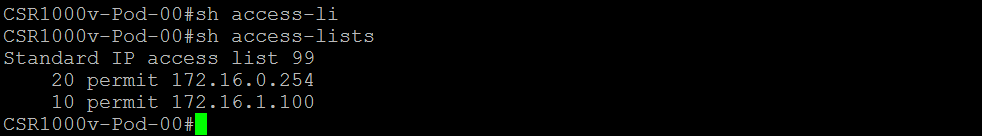
[root@pod5-node1 ansible]# ssh admin@pod5-ios1

1. Accept the fingerprint warning and press enter
2. Enter into the privileged mode with password “!Cisco123”

CSR1000v-Pod-00>enable

Password:

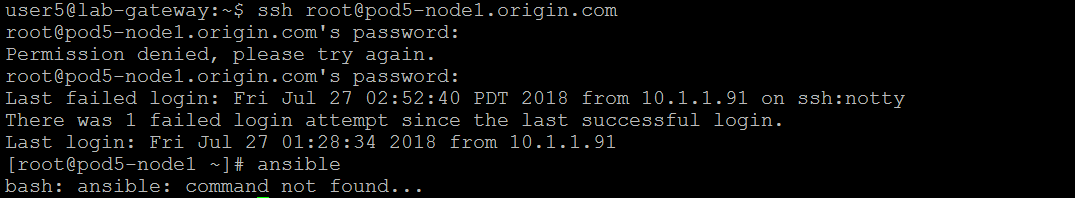
1. Execute the “show access-list” command and verify the ACL is present



1. Exit from the device SSH shell, Type “exit” to exit the shell

# Automation Arista EOS using Ansible

1. Login to the POD5 node1 from the SSH gateway
2. Login as root use on ssh [root@podx-node1.origin.com](mailto:root@podx-node1.origin.com) with password !cisco123



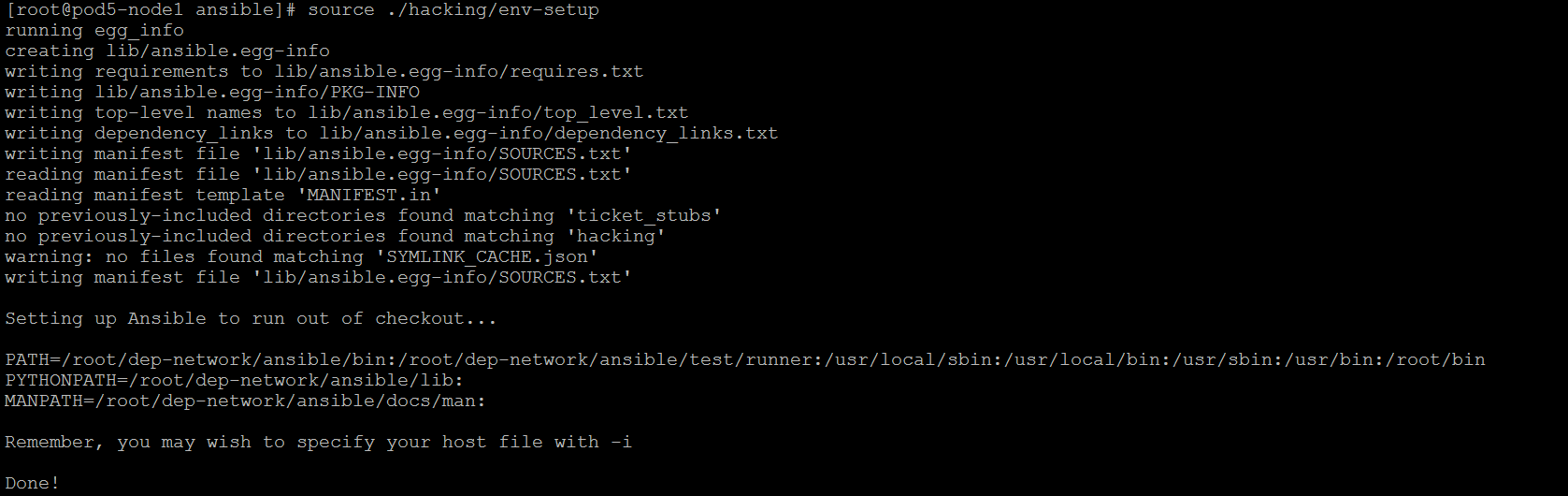
**Note:** Since we have not installed the Ansible as package on the Node VM, the executable will not work when the SSH session is re-established. You need to run the environment setup script on every login.

1. Change to “dep-network/ansible/” directory

cd dep-network/ansible/

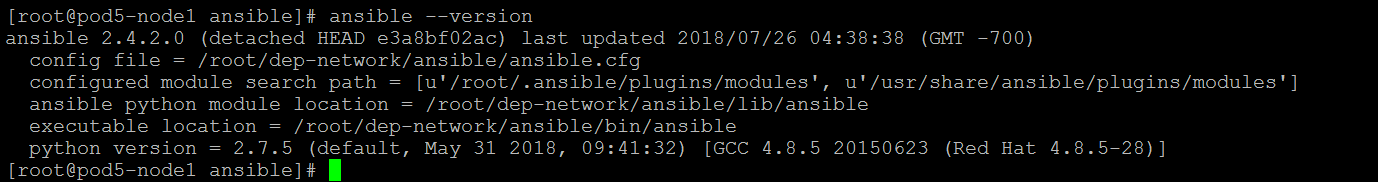
1. Execute the env-setup script to setup the environment

[root@pod5-node1 ansible]# source ./hacking/env-setup



1. Verify the ansible command is executable, Enter ansible --version to verify

[root@pod5-node1 ansible]# ansible --version



1. Create a playbook to configure the following on the Arista vEOS

* Set the **hostname** as **“vEOS-POD5”**
* Configure the **dns name** as **“veos-pod5.onecloudinc.com”**
* Configure the **name servers** as **“8.8.8.8 and 8.8.4.4”**
* Configure the **Login banner** as **“Managed by Ansible”**

1. We will be using the **eos\_system** and **eos\_banner** modules to automate the above configuration
2. Explore the module parameters and options using the ansible-doc <module name>
3. You can manually create the playbook by referring the ansible-doc or use the following command to create the playbook

cat > /root/dep-network/eos\_configure\_device.yml <<EOF

---

- name: Configure EOS switches

hosts: eos

connection: local

gather\_facts: no

vars:

provider:

host: "{{ inventory\_hostname }}"

username: "{{ un\_eos }}"

password: "{{ pwd\_eos }}"

transport: eapi

use\_ssl: false

tasks:

- name: configure hostname and domain-name

eos\_system:

authorize: yes

auth\_pass: "{{ pwd\_eos }}"

hostname: vEOS-POD5

domain\_name: veos-pod5.onecloudinc.com

provider: "{{ provider }}"

- name: configure name servers

eos\_system:

authorize: yes

auth\_pass: "{{ pwd\_eos }}"

name\_servers:

- 8.8.8.8

- 8.8.4.4

provider: "{{ provider }}"

- name: configure the login banner

eos\_banner:

authorize: yes

auth\_pass: "{{ pwd\_eos }}"

banner: login

text: Managed by Ansible

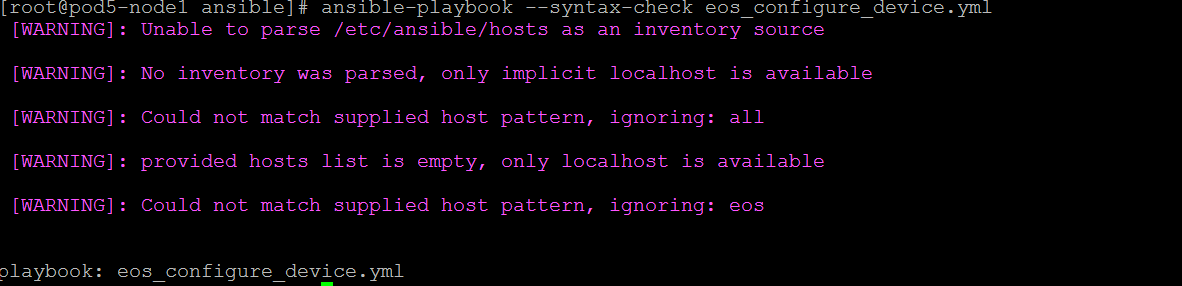
state: present

provider: "{{ provider }}"

EOF

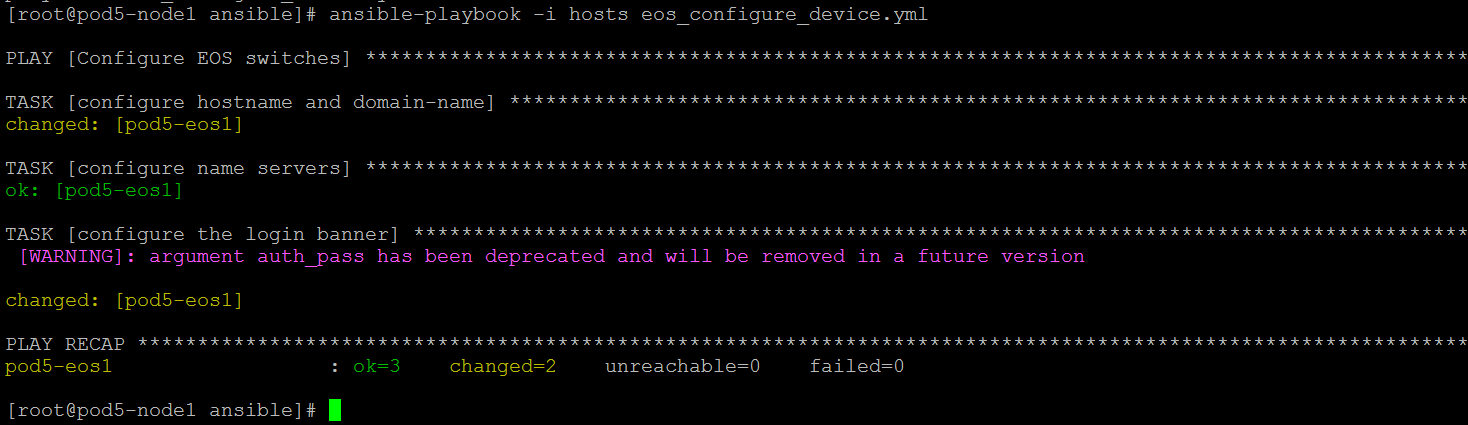
1. Validate the syntax

ansible-playbook --syntax-check eos\_configure\_device.yml



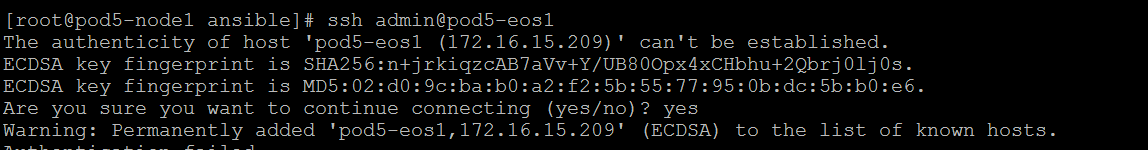
1. Ignore the warnings and execute the playbook

ansible-playbook -i hosts eos\_configure\_device.yml

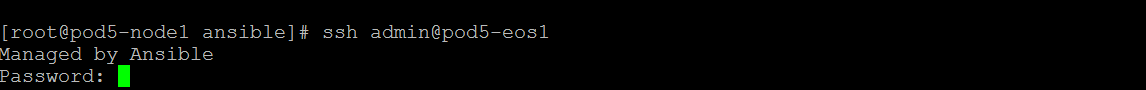


1. Login to the device and verify the configuration
2. SSH to device as **admin/!Cisco123**

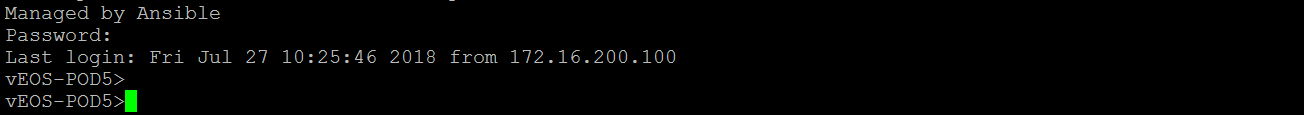
ssh admin@pod5-eos1



1. Verify the configuration
2. Login banner

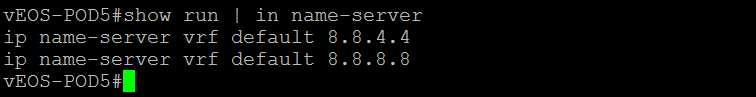


1. Hostname , Name servers



1. Enter to privileged mode with password - **!Cisco123** and execute **“show run | in name-server”**

vEOS-POD5#show run | in name-server



1. Exit from the device ssh shell, Type **“exit”** to exit from the SSH shell

## Perform Cleanup activity on the EOS Device using Ansible

1. Create a playbook to **remove the EOS configuration created during the previous activity**
2. We will be the same **eos\_system** and **eos\_banner** to remove the configuration, Explore the module options and parameters using ansible-doc <module name>
3. You can manually create the playbook by referring the ansible-doc or use the following command create the playbook

cat > /root/dep-network/eos\_remove\_device\_config.yml <<EOF

---

- name: Create VLAN's across NX-OS based switches

hosts: eos

connection: local

gather\_facts: no

vars:

provider:

host: "{{ inventory\_hostname }}"

username: "{{ un\_eos }}"

password: "{{ pwd\_eos }}"

transport: eapi

use\_ssl: false

tasks:

- name: Remove configuration

eos\_system:

authorize: yes

auth\_pass: "{{ pwd\_eos }}"

state: absent

provider: "{{ provider }}"

- name: Remove login banner

eos\_banner:

authorize: yes

auth\_pass: "{{ pwd\_eos }}"

banner: login

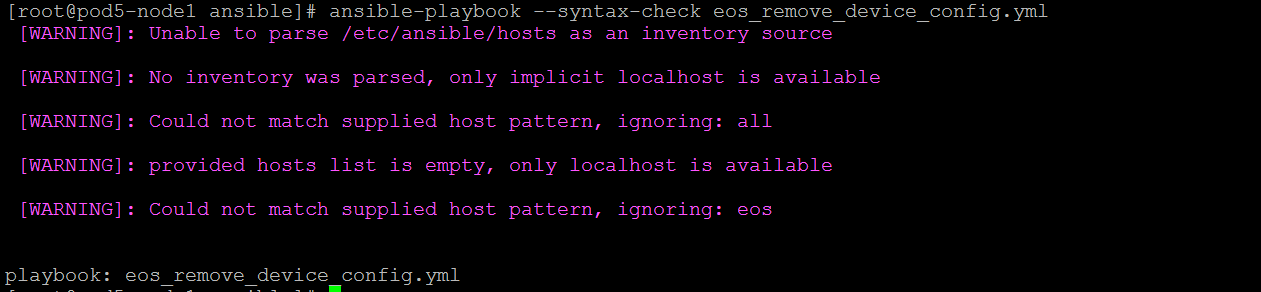
state: absent

provider: "{{ provider }}"

EOF

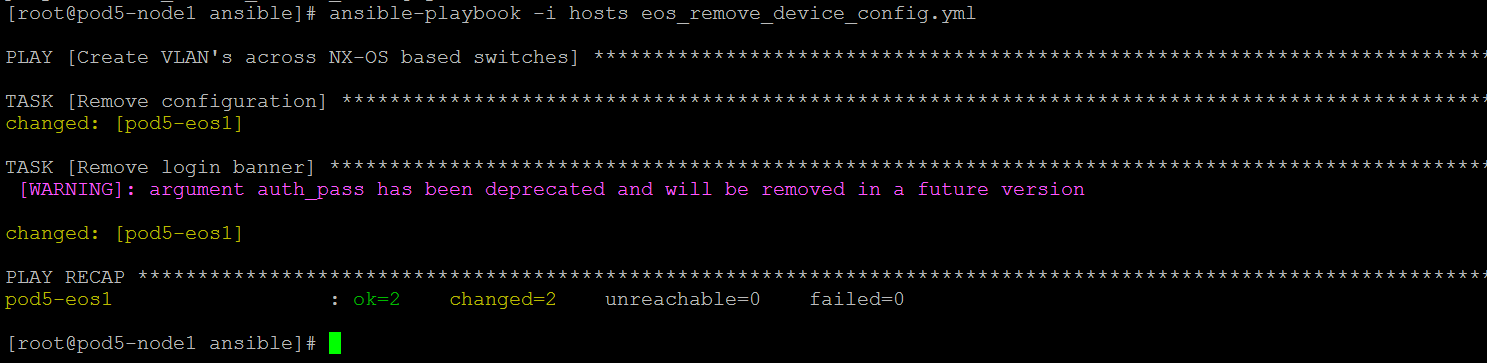
1. Validate the syntax

ansible-playbook --syntax-check eos\_remove\_device\_config.yml



1. Ignore the warnings and execute the playbook

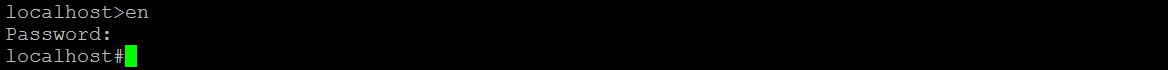
ansible-playbook -i hosts eos\_remove\_device\_config.yml



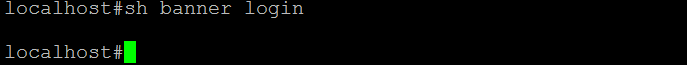
1. Login to the device and verify the configuration are removed
2. SSH to deivce as **admin/!Cisco123**

ssh admin@pod5-eos1

1. Enter into privileged mode with password **!Cisco123**



1. Verify the **Hostname, login banner** configuration are removed.



# Appendix A

## Running Ansible environment setup script

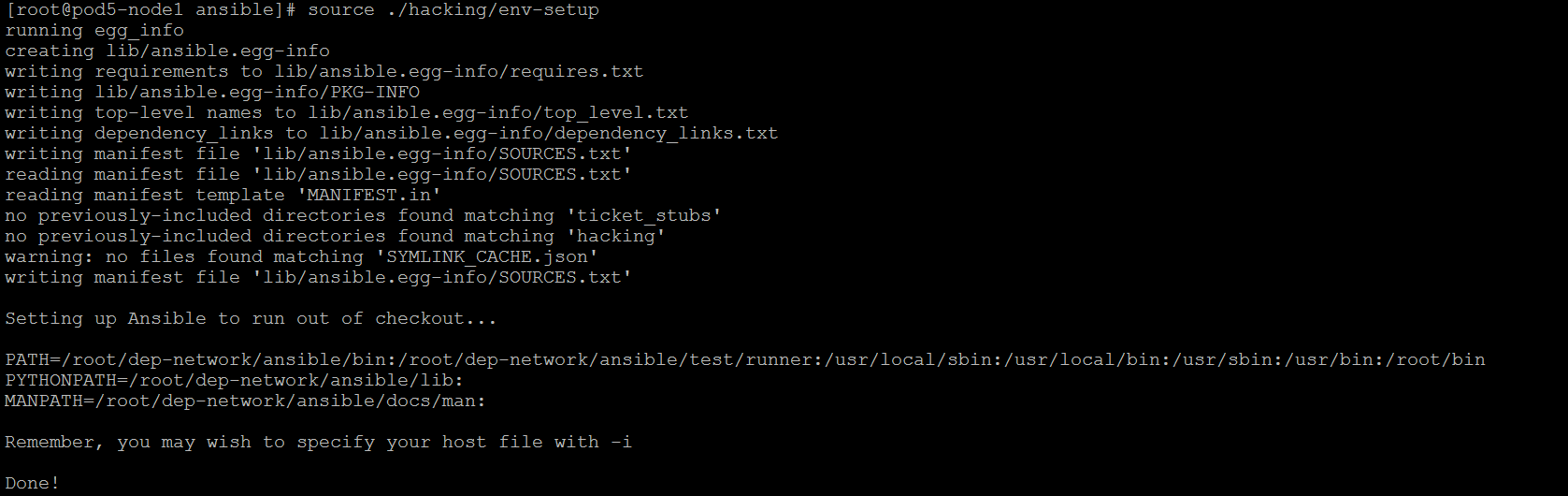
**Note:** Since we have not installed the Ansible as package on the Node VM, the executable will not work if the SSH session is re-established. You need to run the environment setup script on every new login.

1. Change to “dep-network/ansible/” directory

cd dep-network/ansible/

1. Execute the env-setup script to setup the environment

[root@pod5-node1 ansible]# source ./hacking/env-setup



1. Verify the ansible command is executable, Enter ansible --version to verify

[root@pod5-node1 ansible]# ansible --version

