Lab 3 – Ansible Troubleshooting & Tools

Introduction

There are some handy tools you can use to help with figuring out syntax issues as well as general troubleshooting of failed playbooks and such.

Please refer to the **Lab Connection Information.docx** file for information on connecting to your Ansible host.

Don't forget to change your inventory file and replace the XX with your pod information.

1. Use Case 1 - Broken Playbook

1.1 There are a few options out there for YAML Linters but for a GUI www.yamllint.com I find to be one of the better ones and there is also a python package we will talk about.

We are going to cat this broken playbook then take this file output and check it against the linter site to see what is wrong.

Make sure you are in the **lab3-ansible-troubleshooting folder** for this lab.

```
cd ~/ansible_labs/lab3-ansible-troubleshooting
```

Review the broken_playbook.yaml file

```
cat broken_playbook.yaml
```

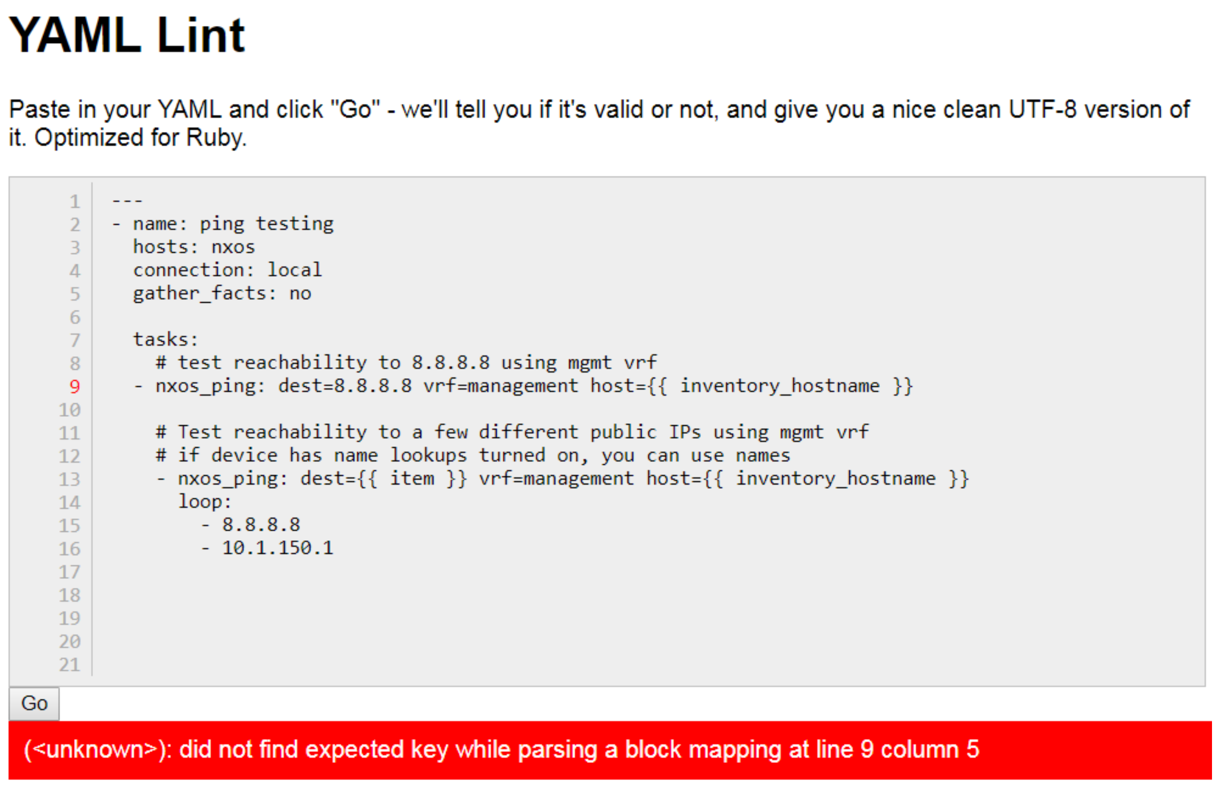
Output:

```
---
```

```
- name: ping testing

  hosts: nxos

  connection: local

  gather_facts: no



  tasks:

    # test reachability to 8.8.8.8 using mgmt vrf

  - nxos_ping: dest=8.8.8.8 vrf=management host={{ inventory_hostname }}



    # Test reachability to a few different public IPs using mgmt vrf

    # if device has name lookups turned on, you can use names

    - nxos_ping: dest={{ item }} vrf=management host={{ inventory_hostname }}

      loop:

        - 8.8.8.8

        - 10.1.150.1
```

Now let’s take the output from that broken file and input it into our YAML Lint site and see if you can fix the file, if you have issues check the working_playbook.yaml file. One thing to know about this site is it after hitting go and it saying its valid yaml it could throw the - formatting off a bit so just be mindful.

So as we can see in the screenshot the issue appears to be with line 9, can you figure out what is wrong? Compare line 9 and line 13 and see if you can figure out the issue.



YAML Lint

Paste in your YAML and click "Go" - we'll tell you if it's valid or not, and give you a nice clean UTF-8 version of it. Optimized for Ruby.

```
 1  ---
 2  - name: ping testing
 3    hosts: nxos
 4    connection: local
 5    gather_facts: no
 6
 7    tasks:
 8      # test reachability to 8.8.8.8 using mgmt vrf
 9    - nxos_ping: dest=8.8.8.8 vrf=management host={{ inventory_hostname }}
10
11      # Test reachability to a few different public IPs using mgmt vrf
12      # if device has name lookups turned on, you can use names
13      - nxos_ping: dest={{ item }} vrf=management host={{ inventory_hostname }}
14        loop:
15          - 8.8.8.8
16          - 10.1.150.1
17
18
19
20
21
```

```
(<unknown>): did not find expected key while parsing a block mapping at line 9 column 5
```

Solution Answer

The issue was an indentation problem with our first nxos_ping task. By looking at the working_playbook.yaml file we can see how it should look.

```
---



- name: ping  testing

  hosts: nxos

  connection: local

  gather_facts: no



  tasks:

    # test reachability to 8.8.8.8 using mgmt vrf

    - nxos_ping: dest=8.8.8.8 vrf=management host={{ inventory_hostname }}



    # Test reachability to a few different public IPs using mgmt vrf

    # if device has name lookups turned on, you can use names

    - nxos_ping: dest={{ item }} vrf=management host={{ inventory_hostname }}

      loop:

        - 8.8.8.8

        - 10.1.150.1
```

2. YAML Lint CLI & Playbook Checker

2.1 We also have options to use ansible-playbook with the --syntax-check option to check our playbooks for errors. As well there is a CLI linter similar to the GUI, we will be looking at both.

Let’s take our same example as before and use the ansible-playbook option to check the syntax

```
ansible-playbook --syntax-check broken_playbook.yaml
```

Output:

```
ERROR! Syntax Error while loading YAML.

  did not find expected key



The error appears to have been in '/root/ansible_lab_files/lab3-troubleshooting/broke
n_playbook.yaml': line 15, column 5, but maybe elsewhere in the file depending on the
exact syntax problem.

The offending line appears to be:

    # if device has name lookups turned on, you can use names

    - nxos_ping: dest={{ item }} vrf=management host={{ inventory_hostname }}

    ^ here

We could be wrong, but this one looks like it might be an issue with

missing quotes.  Always quote template expression brackets when they

start a value. For instance:

    with_items:

      - {{ foo }}

Should be written as:

    with_items:

      - "{{ foo }}"
```

So we can see it found the same line issue and told us something about missing quotes so it didn’t actually realize there was a spacing issue per say.

A successful output for a working playbook should look like below

```
# ansible-playbook --syntax-check working_playbook.yaml

playbook: working_playbook.yaml

#
```

2.2 As previously mentioned there is also a CLI linter option. It is called ansible-lint and can be easily installed with pip which we have already done for you.

```
ansible-lint broken_playbook.yaml
```

As we can see from the output it is basically the same output as the ansible-playbook --syntax-check command.

```
ERROR! Syntax Error while loading YAML.

  did not find expected key



The error appears to have been in '/root/ansible_lab_files/lab3-troubleshooting/broke
n_playbook.yaml': line 15, column 5, but maybe elsewhere in the file depending on the
exact syntax problem.

The offending line appears to be:

    # if device has name lookups turned on, you can use names

    - nxos_ping: dest={{ item }} vrf=management host={{ inventory_hostname }}

    ^ here

We could be wrong, but this one looks like it might be an issue with

missing quotes.  Always quote template expression brackets when they
```

```
start a value. For instance:

    with_items:

      - {{ foo }}

Should be written as:

    with_items:

      - "{{ foo }}"
```

3. Use Case Failing SSH - Using Assert & Debugging

3.1 In ansible we have options to get more debugs as well as we can use the assert module to ensure our commands made the proper changes or to verify certain versions are in use, etc.

Let's start with a simple example to just see more debug on a show version raw output.

Here we are specifying a group called nxos that is in our inventory file. From the previous lab we did how might you see what is in our inventory file?

```
ansible nxos -m raw -a "show version" -u admin -k -v
```

Output: I have removed the actual show version output but notice because we used the -v we see this in blue in the output at the beginning.

```
Using /root/ansible_lab_files/lab3-ansible-troubleshooting/ansible.cfg as config file
```

3.2 Now let's try with --vv and see what other information we see.

```
ansible nxos -m raw -a "show version" -u admin -k -vv
```

Output: I have removed the actual show version output again and notice what debug output we got this time. Notice we get more information about the ansible version and our config file and such. It also tells us when it ran each handler or task in this example as well.

```
ansible 2.6.2

  config file = /root/ansible_lab_files/lab3-ansible-troubleshooting/ansible.cfg

  configured module search path = [u'/root/.ansible/plugins/modules', u'/usr/share/an
sible/plugins/modules']

  ansible python module location = /usr/lib/python2.7/site-packages/ansible

  executable location = /usr/bin/ansible

  python version = 2.7.5 (default, Apr 11 2018, 07:36:10) [GCC 4.8.5 20150623 (Red Ha
t 4.8.5-28)]

Using /root/ansible_lab_files/lab3-ansible-troubleshooting/ansible.cfg as config file

META: ran handlers

META: ran handlers

META: ran handlers
```

3.3 Now let’s try with --vvv and see what other information we see.

```
ansible nxos -m raw -a "show version" -u admin -k -vvv
```

Output: I have removed the actual show version output again and notice what debug output we got this time compared to the -vv.

Notice how we see a lot more debug information about the ssh connection? This is very handy when you are having auth issues or ssh issues when connecting to devices via playbooks or ad-hoc commands.

```
Parsed /root/ansible_lab_files/lab3-ansible-troubleshooting/inventory inventory sourc
e with ini plugin

META: ran handlers

<n9k-standalone-01.localdomain> ESTABLISH SSH CONNECTION FOR USER: admin

<n9k-standalone-01.localdomain> SSH: EXEC sshpass -d12 ssh -C -o ControlMaster=auto -
o ControlPersist=60s -o StrictHostKeyChecking=no -o User=admin -o ConnectTimeout=10 -
o ControlPath=/root/.ansible/cp/bf041968a8 -tt n9k-standalone-01.localdomain 'show ve
rsion'

<n9k-standalone-01.localdomain> (0, 'Cisco Nexus Operating System (NX-OS) Software\r\
nTAC support: http://www.cisco.com/tac\r\nCopyright (C) 2002-2015, Cisco and/or its a
ffiliates.\r\nAll rights reserved.\r\nThe copyrights to certain works contained in th
is software are\r\nowned by other third parties and used and distributed under their 
own\r\nlicenses, such as open source.  This software is provided "as is," and unless\
r\notherwise stated, there is no warranty, express or implied, including but not\r\nl
imited to warranties of merchantability and fitness for a particular purpose.\r\nCert
ain components of this software are licensed under\r\nthe GNU General Public License 
(GPL) version 2.0 or \r\nGNU General Public License (GPL) version 3.0  or the GNU\r\n
Lesser General Public License (LGPL) Version 2.1 or \r\nLesser General Public License
(LGPL) Version 2.0. \r\nA copy of each such license is available at\r\nhttp://www.ope
nsource.org/licenses/gpl-2.0.php and\r\nhttp://opensource.org/licenses/gpl-3.0.html a
nd\r\nhttp://www.opensource.org/licenses/lgpl-2.1.php and\r\nhttp://www.gnu.org/licen
ses/old-licenses/library.txt.\r\n\r\nSoftware\r\n  BIOS: version 07.34\r\n  NXOS: ver
sion 7.0(3)I2(1)\r\n  BIOS compile time:  08/11/2015\r\n  NXOS image file is: bootfla
sh:///nxos.7.0.3.I2.1.bin\r\n  NXOS compile time:  9/3/2015 16:00:00 [09/04/2015 00:1
8:15]\r\n\r\n\r\nHardware\r\n  cisco Nexus9000 C9372PX chassis \r\n  Intel(R) Core(TM
) i3- CPU @ 2.50GHz with 16402008 kB of memory.\r\n  Processor Board ID SAL1947TD7N\r
\n\r\n  Device name: N9k-Standalone-Pod-1\r\n  bootflash:   51496280 kB\r\nKernel upt
ime is 99 day(s), 13 hour(s), 16 minute(s), 35 second(s)\r\n\r\nLast reset at 908987 
usecs after  Tue May  8 01:34:41 2018\r\n\r\n  Reason: Reset Requested by CLI command
reload\r\n  System version: 7.0(3)I2(1)\r\n  Service: \r\n\r\nplugin\r\n  Core Plugin
, Ethernet Plugin\r\n\r\nActive Package(s):\r\n', '\n  _   _                        _
__   ___   ___   ___\n | \\ | | _____  ___   _ ___   / _ \\ / _ \\ / _ \\ / _ \\\n |
\\| |/ _ \\ \\/ / | | / __| | (_) | | | | | | | | | |\n | |\\  |  __/>  <| |_| \\__ \
\  \\__, | |_| | |_| | |_| |\n |_| \\_|\\___/_/\\_\\\\__,_|___/    /_/ \\___/ \\___/ 
\\___/\n\n Welcome to the Nexus 9000 Programmability Lab\n\n This device has been res
erved for training purposes.\n Please contact davidclin@onecloudinc.com for any quest
ions.\n\nShared connection to n9k-standalone-01.localdomain closed.\r\n')
```

The final -vvvv just gives even more information.

Now let’s try giving it an incorrect username and see what our debug output looks like, try this command on your own and review the output. We will use -vvvv.

```
ansible nxos -m raw -a "show version" -u admin -k -vvvv
```

3.4 Now using the assert module and a little regex let’s see if our Nexus switch is running a 7.x version.

This would be useful if we were trying to make sure all our switches were on a certain version or if we were doing a upgrade and wanted to assert if after the upgrade the version didn’t match our upgrade version.

Let’s take a look first at this playbook **assert_example_solution.yaml** and what we are doing here:

```
---

- name: verify nxos version

  hosts: nxos

  connection: network_cli

  gather_facts: yes



  tasks:

    - nxos_facts:

    - name: Current NXOS Version

      debug: var=ansible_net_version

    - name: Verify Version is 7.x

      assert:

        that:
          - "'{{ ansible_net_version }}' is match('^7')"
```

From the tasks section you can see we are using a variable that is a part of the nxos module called **ansible_net_version** which is the version of the switch that was collected from the **gather_facts: yes** part of the playbook.

Then we use the assert module with a regex to ensure the version output starts with a 7 which is the is **match('^7')** part at the bottom.

We will specify the **-e ansible_network_os=nxos** per the NXOS connection requirements, later on we will move this to a group_var

Please write this playbook yourself and then lets run it below to see what our results are. If you get stuck or have issues please see the **assert_example_solution.yaml** file in the lab3 directory.

```
ansible-playbook assert_example_solution.yaml -u admin -k -e ansible_network_os=nxos
```

Output: Here we can see it passed if the version was say a 6.x it would come back as a failure.

```
PLAY [verify nxos version] **********************************************************
*************************************************



TASK [Gathering Facts] **************************************************************
*************************************************


ok: [n9k-standalone-XX.localdomain]



TASK [nxos_facts] *******************************************************************
*************************************************


ok: [n9k-standalone-XX.localdomain]



TASK [Current NXOS Version] *********************************************************
*************************************************
```

```
ok: [n9k-standalone-XX.localdomain] => {

    "ansible_net_version": "7.0(3)I2(1)"

}



TASK [Verify Version is 7.x] ********************************************************
*************************************************

ok: [n9k-standalone-XX.localdomain] => {

    "changed": false,

    "msg": "All assertions passed"

}



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

n9k-standalone-XX.localdomain : ok=4    changed=0    unreachable=0    failed=0
```

Try modifying the regex from a 7 to a 6 and re-run the playbook and see what type of result you get, compare the result to the successful playbook.

4. Test Your Knowledge

4.1 In this lab you will test what we have already taught you and walked you through to build your own playbook from the template (**assert_example.yaml**) using the NXOS command parameters.

Below you will configure a VLAN on your NXOS group in the inventory file provided to you then verify and assert if it wasn't created correctly.

Remember what you have learned and the **ansible-doc -l** command

Example Information

We want the following task completed on our group nxos in our playbook.

1. Create VLAN 2000
2. Verify using debug module in the playbook to verify that VLAN 2000 is found on the switch after creation.

Since we haven't fully discussed playbooks all the way yet I have provided a starter playbook so all you have to add are the required tasks for this example.

I have provided you the first **- nxos_facts** task to start off and get the switch facts. Remember after making our VLAN change we will need to run facts again to check and see what the values are after our change.

What module would you use to output the values?

Where could we find what the nxos_facts variables are to find our vlan?

What module would we use in order to add a vlan 2000?

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

```
---



- name: verify nxos version

  hosts: nxos

  connection: network_cli

  gather_facts: yes



  tasks:

    - nxos_facts:
```

Helpful Information

ansible-doc -s nxos_vlan

https://docs.ansible.com/ansible/devel/modules/nxos_vlan_module.html#nxos-vlan-module

https://docs.ansible.com/ansible/devel/modules/debug_module.html#debug-module

https://docs.ansible.com/ansible/latest/network/user_guide/platform_nxos.html