Lab 1 – CLI Tools & Tricks

Introduction

Ansible comes with some pretty handy tools out of the box besides just playbooks and other things we will learn about soon enough. Let’s walk through some different CLI tools and commands we have at our disposal. You can also see more information on the ansible command itself at the documentation located at <https://docs.ansible.com/ansible/latest/cli/ansible.html>

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

1. Ansible-Config

**1.1** The first CLI command we are going to talk about and interact with is ansible-config so we can take a look at the pre-configured ansible.cfg file we have setup.

Make sure you are in the **lab1-cli-tools folder** for this lab.

cd ~/ansible\_lab/lab1-cli-tools

ansible-config view

**Output:** We have host\_key\_checking setup to False so that way we can get around a few things in the lab setup. Basically this is normally on by default and if a host is not in the known\_hosts file it can cause some issues.

We are also telling it to use the inventory file located in this folder instead of having to specify it. By default ansible looks in the /etc/ansible/hosts for the inventory.

[defaults]

host\_key\_checking = False

inventory = ./inventory

**1.2** There is also a dump option and we can use the –only-changed to see what is unique about our ansible.cfg file compared to the default one.

ansible-config dump --only-changed

**Output:** Here we can see we have the default\_host\_list(the inventory) and the host\_key\_checking that are unique and changed in our ansible.cfg file compared to the default.

DEFAULT\_HOST\_LIST(/root/ansible\_lab\_files/lab1-cli-tools/ansible.cfg) = [u'/root/ansible\_lab\_files/lab1-cli-tools/inventory']

HOST\_KEY\_CHECKING(/root/ansible\_lab\_files/lab1-cli-tools/ansible.cfg) = False

**Note:** Press “**q**” to exit the command.

2. Ansible Command

**2.1** The first CLI command we are going to talk about and interact with is ansible itself. If we just run the ansible command it will give us back a list of all the different options we have.

ansible

**2.2** Notice all the options we have for this output we will focus on a few of these, there is ansible –version which gives us our version running.

ansible --version

**Output:**

ansible 2.6.2

config file = /etc/ansible/ansible.cfg

configured module search path = [u'/root/.ansible/plugins/modules', u'/usr/share/ansible/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 Hat 4.8.5-28)]

**2.3** We can also use the ansible –doc to look at the ansible documentation live from the CLI and see what our options are.

Copy

ansible-doc

**Output:** Here we can see some of the options the ansible-doc gives us. So we could use the –all or –a to see everything. The –l is handy and allows us to see all the modules available in the Ansible platform.

Usage: ansible-doc [-l|-F|-s] [options] [-t <plugin type> ] [plugin]

plugin documentation tool

Options:

-a, --all \*\*For internal testing only\*\* Show documentation for

all plugins.

-h, --help show this help message and exit

-j, --json \*\*For internal testing only\*\* Dump json metadata for

all plugins.

-l, --list List available plugins

-F, --list\_files Show plugin names and their source files without

summaries (implies --list)

-M MODULE\_PATH, --module-path=MODULE\_PATH

prepend colon-separated path(s) to module library

(default=[u'/root/.ansible/plugins/modules',

u'/usr/share/ansible/plugins/modules'])

-s, --snippet Show playbook snippet for specified plugin(s)

-t TYPE, --type=TYPE Choose which plugin type (defaults to "module")

-v, --verbose verbose mode (-vvv for more, -vvvv to enable

connection debugging)

--version show program's version number and exit

See man pages for Ansible CLI options or website for tutorials

https://docs.ansible.com

Now let’s use the ansible-doc –l command and see what nxos plugin options are available for us to use.

Copy

ansible-doc –l | grep nxos

**Output:** Here we can see a small subset of all the data we got back and shows us what options the nxos module has.

nxos\_aaa\_server

Manages AAA server global configuration.

nxos\_aaa\_server\_host

Manages AAA server host-specific configuration.

nxos\_acl

Manages access list entries for ACLs.

nxos\_acl\_interface

Manages applying ACLs to interfaces.

nxos\_banner

Manage multiline banners on Cisco NXOS devices

nxos\_bgp

Manages BGP configuration.

nxos\_bgp\_af

Manages BGP Address-family configuration.

nxos\_bgp\_neighbor

Manages BGP neighbors configurations.

nxos\_bgp\_neighbor\_af

Manages BGP address-family's neighbors configuration.

nxos\_command

Run arbitrary command on Cisco NXOS devices

nxos\_config

Manage Cisco NXOS configuration sections

**2.4** We can also use the **ansible –list-hosts** to see what hosts are a part of a certain group.

There are several ways to view the inventory file. We will explore these next. The first is a straight forward cat of the file

Copy

cat inventory

**Output:** Here we can see what our current inventory file looks like that we are working with. We have 5 groups here, server, switches, eos, nxos and routers with devices in each group.

[server]

localhost

[switches]

veos-pod-XX.localdomain

[eos]

veos-pod-XX.localdomain

[nxos]

n9k-standalone-XX.localdomain

[routers]

csr1000v-pod-XX.localdomain

**Take the time right now to modify the inventory file so that the XX’s are replaced with the proper numbers of your devices from your lab sheet.**

**With vi you can enter “:” to get to command mode and then use %s/XX/YY/g where YY is your pod number**

**2.5**  Verify that switches group from the host inventory

ansible switches --list-hosts

**Output:** You can see here we looked up the “switches” group and it told us that it found the below host in that group.

hosts (1):

veos-pod-XX.localdomain

**2.6**  We can also use the group all to see all of our hosts in our playbook file

Copy

ansible all --list-hosts

**Output:** So here since we gave it the all group we have both of our switches from our inventory file.

# ansible all --list-hosts

hosts (4):

n9k-standalone-XX.localdomain

veos-pod-XX.localdomain

csr1000v-pod-XX.localdomain

localhost

**2.7**  Now utilizing what we have already done we can run a ansible module called ping against the server group. This is a way to verify that the server can run python modules and such, it does not mean that its actually pinging it.

**Be sure to modify the inventory file so that the XX’s are replaced with the proper numbers of your devices from your lab sheet.**

We are supplying –m to specify the ping module and giving it –u to tell it to use the root user and –k to ask us for the password.

ansible server -m ping -u root -k

**Output:** So here since we gave it the server group we have only one response from the localhost server

SSH password:

localhost | SUCCESS => {

"changed": false,

"ping": "pong"

}

**2.8**  There is also another command to help with the configuration and it is called ansible-inventory. It can be used to display or dump the inventory how Ansible currently sees it. By using the --list it works like an inventory script.

ansible-inventory --list

**Output:** Here you can see we have a json output, we can also give it the flag -y to get the output in YAML, feel free to try that on your own.

{

"\_meta": {

"hostvars": {

"csr1000v-pod-00.localdomain": {},

"localhost": {},

"n9k-standalone-01.localdomain": {},

"veos-pod-00.localdomain": {}

}

},

"all": {

"children": [

"eos",

"nxos",

"routers",

"server",

"switches",

"ungrouped"

]

},

"eos": {

"hosts": [

"veos-pod-00.localdomain"

]

},

"nxos": {

"hosts": [

"n9k-standalone-01.localdomain"

]

},

"routers": {

"hosts": [

"csr1000v-pod-00.localdomain"

]

},

"server": {

"hosts": [

"localhost"

]

},

"switches": {

"hosts": [

"veos-pod-00.localdomain"

]

},

"ungrouped": {}

}

3. Testing Connectivity

**3.1** Now that we have run through using ansible to check our version and see what our hosts are in our files, let’s verify via a simple CLI command we can connect and interact with our devices. In this example some switches. We have an inventory file already pre-configured in this lab directory. Feel free to **cat inventory** to look further at it.

Copy

ansible switches -m raw -a "show version" -u admin -k

Now what we are doing in this command is telling ansible to run this command against only switches group hosts in our inventory file which we don’t have to specify since we are already doing that in the ansible.cfg file in our directory. Feel free to **“ansible-config view”** to see the current configuration.

The next option –m is stating we want to use the module named raw. The raw module is a way of executing what Ansible refers to as a ***“low-down and dirty”*** SSH command. You typically only use this on devices like routers and switches that don’t have python installed, if python can be installed on the node you are interacting with then you would use the **shell** or **command** module.

The –a is the way we tell ansible the module args to pass, in this case it’s the command we want to run on the ssh session which is **“show version”.**

The –u is the way we specify the user account to use, in our case we want to ssh in with user admin. The –k is how we tell it to ask us for the password to the device, later on we will discuss using ssh keys as well. When it asks for the ssh password please refer to your lab guide information for the switch password.

**Output:** Here we can see the output of us running the show version against only the switches group. Feel free to try running it against the group all or routers.

veos-pod-XX.localdomain | SUCCESS | rc=0 >>

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: 3 weeks, 0 days, 11 hours and 18 minutes

Total memory: 4010988 kB

Free memory: 654600 kB

Shared connection to veos-pod-XX.localdomain closed.

4. Ansible-console

**4.1** There is also a really cool command called ansible-console which allows you to run interactive commands from the CLI sort of like a REPL. Below we will test running some interactive commands against a server

ansible-console server -u root -k

**Output:** Now we should be presented with an option sort of like you see below. This is the interactive CLI where we can run module commands and such within reason. We will be interacting by using the raw module for now.

SSH password:

Welcome to the ansible console.

Type help or ? to list commands.

admin@server (1)[f:5]$

**Note :** You can type "**exit**" to exit fom the server

Now we are going to test running a command on the CLI

raw hostname

**Output:** Now we should see the output below of the “hostname” command

localhost | SUCCESS | rc=0 >>

Ansible-CentOS-POD01

Shared connection to localhost closed.

**Note :** You can type "**exit**" to exit from the server