Contents

[Basic Automation using Telnet (telnetlib) 2](#_Toc524612506)

[Section 1 (telnetlib) - Script 1 4](#_Toc524612507)

[Section 1 (telnetlib) - Script 2 6](#_Toc524612508)

[Section 1 (telnetlib) - Script 3 8](#_Toc524612509)

[Basic Automation using SSH (netmiko) 10](#_Toc524612510)

[Netmiko basics 10](#_Toc524612511)

[**Netmiko Library** 10](#_Toc524612512)

[Section 2 (netmiko) - Script 1 - configuring a single switch 13](#_Toc524612513)

[Section 2 (netmiko) - Script 2 - configuring multiple switches 15](#_Toc524612514)

[Section 2 (netmiko) - Script 3 - configuring multiple switches using a file config 18](#_Toc524612515)

[Section 2 (netmiko) - Script 4 - configuring multiple switches using a 2 files with config 20](#_Toc524612516)

[Network Automation with NAPALM 24](#_Toc524612517)

[Section 3 (napalm) - Script 1 - Get remote device info using NAPALM 27](#_Toc524612518)

[Section 3 (napalm) - Script 2 - Get remote device info using NAPALM (improved output) 28](#_Toc524612519)

[Section 3 (napalm) - Script 3 - Get remote device info using NAPALM (improved output using JSON) 29](#_Toc524612520)

[Section 3 (napalm) - Script 4 - Get more remote device info using NAPALM (improved output using JSON) 31](#_Toc524612521)

[Section 3 (napalm) - Script 5 - Get BGP neighbor info using NAPALM and print the output using JSON 34](#_Toc524612522)

[Section 3 (napalm) - Script 6 - Making config changes to remote devices using NAPALM 36](#_Toc524612523)

[Section 3 (napalm) - Script 6 - Check for diff and make config changes to remote devices using NAPALM 38](#_Toc524612524)

[Section 3 (napalm) - Script 7 - Check for config diff multiple devices using NAPALM 40](#_Toc524612525)

[Section 8 - Iterations with Netmiko 42](#_Toc524612526)

[Section 8 (iterations with Netmiko) - Iteration 1 - Send a 'show' command 42](#_Toc524612527)

[Section 8 (iterations with Netmiko) - Iteration 2 - Send commands from a file 44](#_Toc524612528)

[Section 8 (iterations with Netmiko) - Iteration 3 - Send commands from a file to multiple devices 46](#_Toc524612529)

[Section 8 (iterations with Netmiko) - Iteration 4 - Send commands from a file to multiple devices with user prompt 48](#_Toc524612530)

[Section 8 (iterations with Netmiko) - Iteration 5 - Send commands from a file to multiple devices with user prompt and error handling 50](#_Toc524612531)

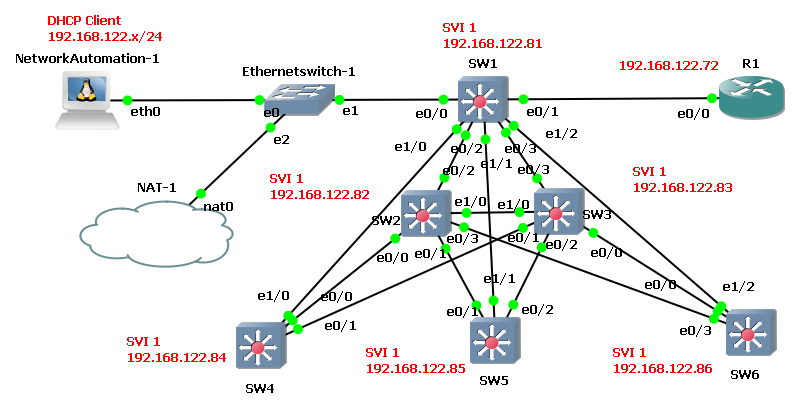
[Section 9 - Scaling Python scripts 53](#_Toc524612532)

[Section 9 (scaling Netmiko) - Encrypt a passwords file using simple-crypt 53](#_Toc524612533)

[Section 9 (scaling Netmiko) - Measuring the script execution time in Netmiko sequential deployment 55](#_Toc524612534)

## Basic Automation using Telnet (telnetlib)

**Topology**



**Network Automation-1 pc DHCP config:**

**Step 1**

Change the following:

*root@NetworkAutomation-1:/etc/network#* ***cat interfaces***

*#*

*# This is a sample network config uncomment lines to configure the network*

*#*

*# Static config for eth0*

*#auto eth0*

*#iface eth0 inet static*

*# address 192.168.0.2*

*# netmask 255.255.255.0*

*# gateway 192.168.0.1*

*# up echo nameserver 192.168.0.1 > /etc/resolv.conf*

*# DHCP config for eth0*

*# auto eth0*

*# iface eth0 inet dhcp*

to:

*root@NetworkAutomation-1:/etc/network#* ***cat interfaces***

*#*

*# This is a sample network config uncomment lines to configure the network*

*#*

*# Static config for eth0*

*#auto eth0*

*#iface eth0 inet static*

*# address 192.168.0.2*

*# netmask 255.255.255.0*

*# gateway 192.168.0.1*

*# up echo nameserver 192.168.0.1 > /etc/resolv.conf*

*# DHCP config for eth0*

*auto eth0*

*iface eth0 inet dhcp*

**Step 2**

Reboot the VM (right-click from GNS3)

**Verification**

root@NetworkAutomation-1:~# **ifconfig**

eth0 Link encap:Ethernet HWaddr d6:17:0f:b9:04:21

inet addr:192.168.122.225 Bcast:192.168.122.255 Mask:255.255.255.0

UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1

RX packets:170875 errors:0 dropped:53 overruns:0 frame:0

TX packets:168894 errors:0 dropped:0 overruns:0 carrier:0

collisions:0 txqueuelen:1000

RX bytes:7197250 (7.1 MB) TX bytes:7097415 (7.0 MB)

lo Link encap:Local Loopback

inet addr:127.0.0.1 Mask:255.0.0.0

inet6 addr: ::1/128 Scope:Host

UP LOOPBACK RUNNING MTU:65536 Metric:1

RX packets:0 errors:0 dropped:0 overruns:0 frame:0

TX packets:0 errors:0 dropped:0 overruns:0 carrier:0

collisions:0 txqueuelen:1

RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)

**Switch and Router configuration**

Here is a sample config that allows Telnet reachability:

SW2(config)#**int range e0/0 - 3 , e1/0**

SW2(config-if-range)#**switchport trunk encapsulation dot1q**

SW2(config-if-range)#**switchport mode trunk**

SW2(config-if-range)#!

SW2(config-if-range)#**int vlan 1**

SW2(config-if)#**no shut**

SW2(config-if)#**ip add 192.168.122.82 255.255.255.0**

SW2(config-if)#!

SW2(config-if)#**line vty 0 4**

SW2(config-line)#**login local**

SW2(config-line)#**transport input all**

SW2(config-line)#**exit**

SW2(config)#**username cisco password cisco**

SW2(config)#**username mikis password cisco**

SW2(config)#**enable password cisco**

**Verification**

root@NetworkAutomation-1:~# **telnet 192.168.122.81**

Trying 192.168.122.81...

Connected to 192.168.122.81.

Escape character is '^]'.

User Access Verification

Username: **cisco**

Password:

SW1>**enable**

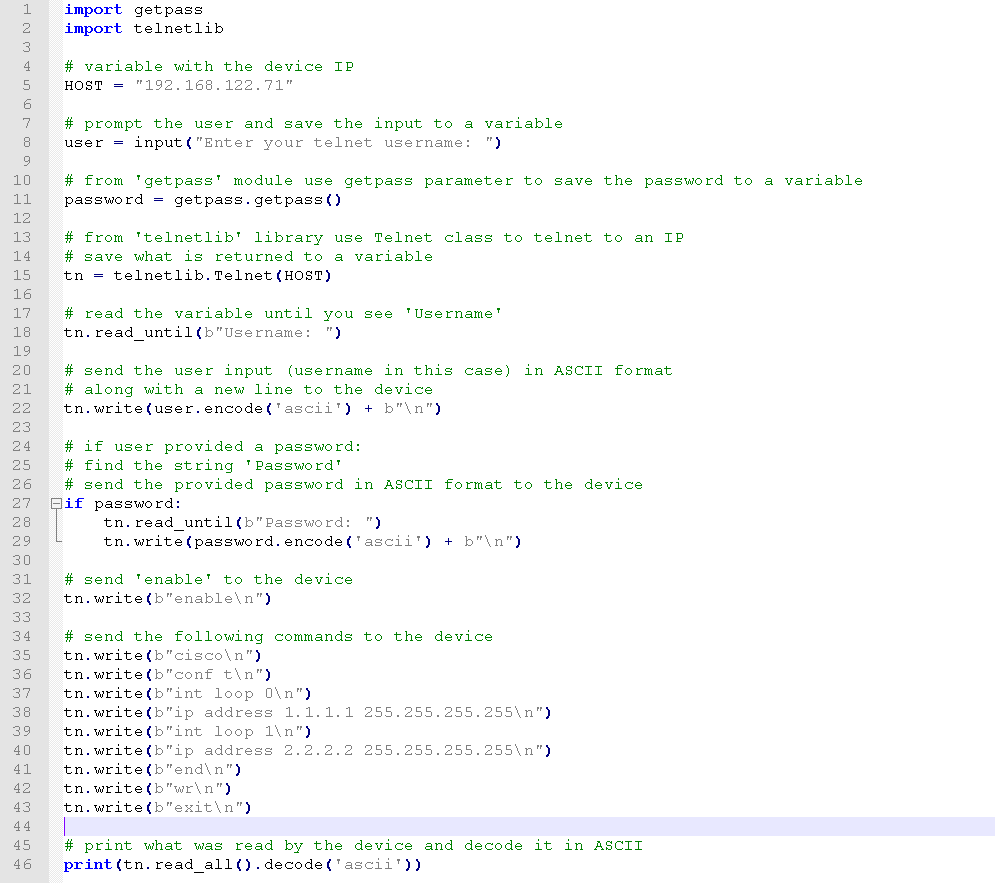
Password:

SW1#

### Section 1 (telnetlib) - Script 1

The following script:

* Connects via telnet to a device
* The IP of the device is specified manually (the HOST variable)
* The user is prompted for user and password
* The provided credentials are used by the script in the telnet connection
* The script creates 2 loopback interfaces



import getpass

import telnetlib

# variable with the device IP

HOST = "192.168.122.71"

# prompt the user and save the input to a variable

user = input("Enter your telnet username: ")

# from 'getpass' module use getpass parameter to save the password to a variable

password = getpass.getpass()

# from 'telnetlib' library use Telnet class to telnet to an IP

# save what is returned to a variable

tn = telnetlib.Telnet(HOST)

# read the variable until you see 'Username'

tn.read\_until(b"Username: ")

# send the user input (username in this case) in ASCII format

# along with a new line to the device

tn.write(user.encode('ascii') + b"\n")

# if user provided a password:

# find the string 'Password'

# send the provided password in ASCII format to the device

if password:

tn.read\_until(b"Password: ")

tn.write(password.encode('ascii') + b"\n")

# send 'enable' to the device

tn.write(b"enable\n")

# send the following commands to the device

tn.write(b"cisco\n")

tn.write(b"conf t\n")

tn.write(b"int loop 0\n")

tn.write(b"ip address 1.1.1.1 255.255.255.255\n")

tn.write(b"int loop 1\n")

tn.write(b"ip address 2.2.2.2 255.255.255.255\n")

tn.write(b"end\n")

tn.write(b"wr\n")

tn.write(b"exit\n")

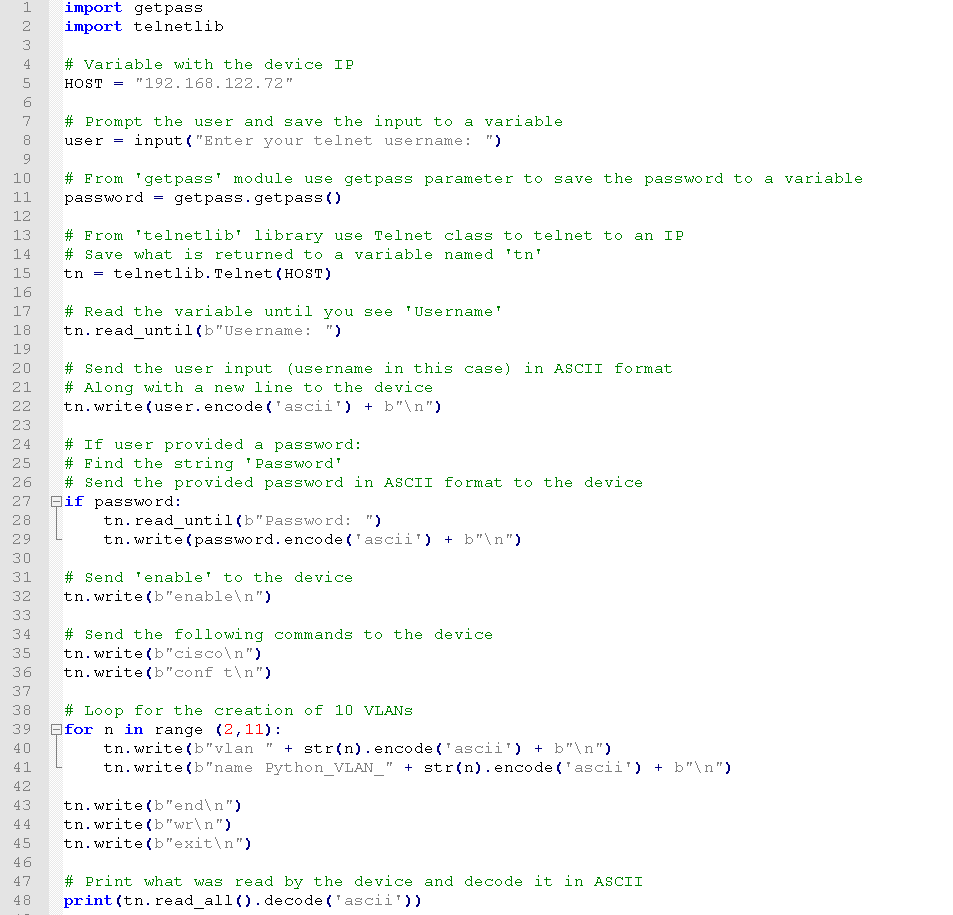
# print what was read by the device and decode it in ASCII

print(tn.read\_all().decode('ascii'))

### Section 1 (telnetlib) - Script 2

The following script:

* Connects via telnet to a device
* The IP of the device is specified manually (the HOST variable)
* The user is prompted for user and password
* The provided credentials are used by the script in the telnet connection
* The script uses an **if loop** to create 10 VLANs



import getpass

import telnetlib

# Variable with the device IP

HOST = "192.168.122.72"

# Prompt the user and save the input to a variable

user = input("Enter your telnet username: ")

# From 'getpass' module use getpass parameter to save the password to a variable

password = getpass.getpass()

# From 'telnetlib' library use Telnet class to telnet to an IP

# Save what is returned to a variable named 'tn'

tn = telnetlib.Telnet(HOST)

# Read the variable until you see 'Username'

tn.read\_until(b"Username: ")

# Send the user input (username in this case) in ASCII format

# Along with a new line to the device

tn.write(user.encode('ascii') + b"\n")

# If user provided a password:

# Find the string 'Password'

# Send the provided password in ASCII format to the device

if password:

tn.read\_until(b"Password: ")

tn.write(password.encode('ascii') + b"\n")

# Send 'enable' to the device

tn.write(b"enable\n")

# Send the following commands to the device

tn.write(b"cisco\n")

tn.write(b"conf t\n")

# Loop for the creation of 10 VLANs

for n in range (2,11):

tn.write(b"vlan " + str(n).encode('ascii') + b"\n")

tn.write(b"name Python\_VLAN\_" + str(n).encode('ascii') + b"\n")

tn.write(b"end\n")

tn.write(b"wr\n")

tn.write(b"exit\n")

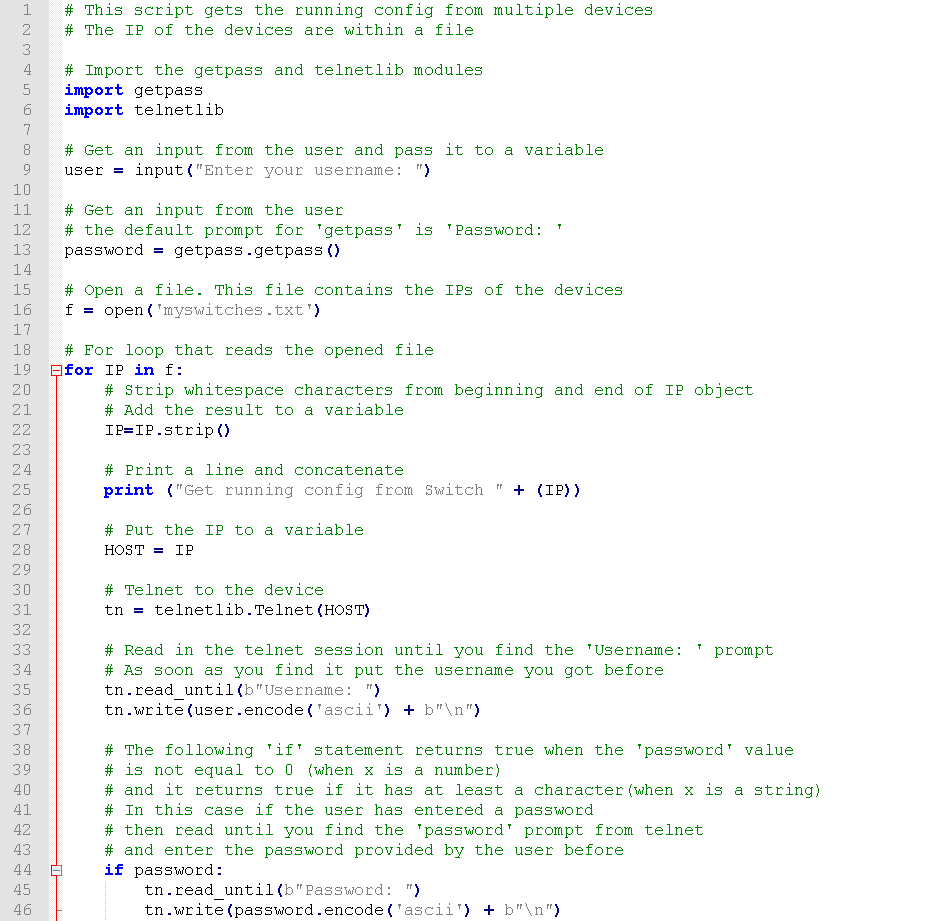
# Print what was read by the device and decode it in ASCII

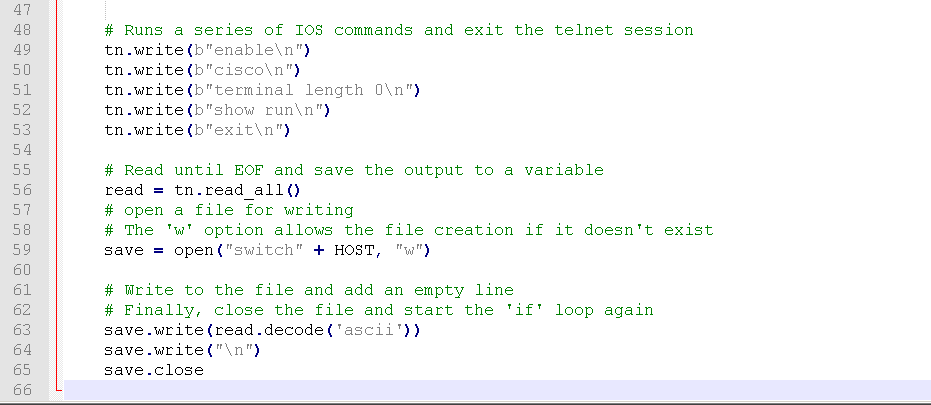
print(tn.read\_all().decode('ascii'))

### Section 1 (telnetlib) - Script 3

The following script:

* Connects via telnet to multiple devices
* The IP of the device are read by the script from a file
* The user is prompted for user and password
* The provided credentials are used by the script in the telnet connection
* The script saves the running config of the remote devices to separate files





# This script gets the running config from multiple devices

# The IP of the devices are within a file

# Import the getpass and telnetlib modules

import getpass

import telnetlib

# Get an input from the user and pass it to a variable

user = input("Enter your username: ")

# Get an input from the user

# the default prompt for 'getpass' is 'Password: '

password = getpass.getpass()

# Open a file. This file contains the IPs of the devices

f = open('myswitches.txt')

# For loop that reads the opened file

for IP in f:

# Strip whitespace characters from beginning and end of IP object

# Add the result to a variable

IP=IP.strip()

# Print a line and concatenate

print ("Get running config from Switch " + (IP))

# Put the IP to a variable

HOST = IP

# Telnet to the device

tn = telnetlib.Telnet(HOST)

# Read in the telnet session until you find the 'Username: ' prompt

# As soon as you find it put the username you got before

tn.read\_until(b"Username: ")

tn.write(user.encode('ascii') + b"\n")

# The following 'if' statement returns true when the 'password' value

# is not equal to 0 (when x is a number)

# and it returns true if it has at least a character(when x is a string)

# In this case if the user has entered a password

# then read until you find the 'password' prompt from telnet

# and enter the password provided by the user before

if password:

tn.read\_until(b"Password: ")

tn.write(password.encode('ascii') + b"\n")

# Runs a series of IOS commands and exit the telnet session

tn.write(b"enable\n")

tn.write(b"cisco\n")

tn.write(b"terminal length 0\n")

tn.write(b"show run\n")

tn.write(b"exit\n")

# Read until EOF and save the output to a variable

read = tn.read\_all()

# open a file for writing

# The 'w' option allows the file creation if it doesn't exist

save = open("switch" + HOST, "w")

# Write to the file and add an empty line

# Finally, close the file and start the 'if' loop again

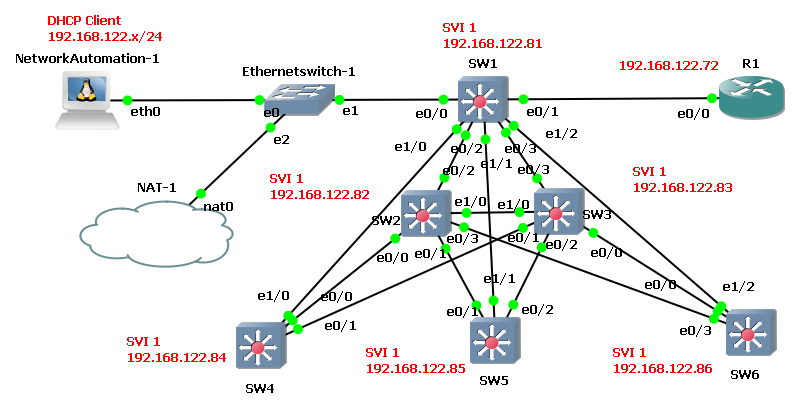
save.write(read.decode('ascii'))

save.write("\n")

save.close

## Basic Automation using SSH (netmiko)

**Topology**



**Step 1**

Prepare the devices for SSH access

SW1(config)#**hostname SW1**

SW1(config)#**ip domain-name mikis.lab**

SW1(config)#**crypto key generate rsa**

The name for the keys will be: SW1.mikis.lab

Choose the size of the key modulus in the range of 360 to 4096 for your

General Purpose Keys. Choosing a key modulus greater than 512 may take

a few minutes.

How many bits in the modulus [512]: **1024**

% Generating 1024 bit RSA keys, keys will be non-exportable...

[OK] (elapsed time was 0 seconds)

SW1(config)#**line vty 0 4**

SW1(config-line)#**transport input ssh**

SW1(config-line)#**login local**

SW1(config)#**username cisco password cisco**

SW1(config)#**username cisco privilege 15**

SW1(config)#**enable secret cisco**

**Verification**

root@NetworkAutomation-1:~# **ssh -l cisco 192.168.122.81**

Bad owner or permissions on /root/.ssh/config

root@NetworkAutomation-1:~# **chmod 600 /root/.ssh/config**

root@NetworkAutomation-1:~# **chown -R root /root/.ssh/config**

root@NetworkAutomation-1:~# **ssh -l cisco 192.168.122.81**

Password:

SW1>**enable**

Password:

SW1#

### Netmiko basics

From https://pynet.twb-tech.com/blog/automation/netmiko.html:

**Netmiko Library**

**By Kirk Byers**

**2015-06-19**

Since late 2014, I have been working on an open-source Python library that simplifies SSH management to network devices. The library is based on the Paramiko SSH library and is named Netmiko.

You can find the library at <https://github.com/ktbyers/netmiko> and the latest released version of the software can be [downloaded here](https://github.com/ktbyers/netmiko/releases).

The purposes of the library are the following:

* Successfully establish an SSH connection to the device
* Simplify the execution of show commands and the retrieval of output data
* Simplify execution of configuration commands including possibly commit actions
* Do the above across a broad set of networking vendors and platforms

I have observed across time that many people run into similar issues managing Python SSH sessions to network devices. For example, HP ProCurve switches have ANSI escape codes in the output or the Cisco WLC has an extra 'login as:' message. These types of issues can soak up a lot of development and troubleshooting time and what is worse people keep solving the same issues over and over again (including sometimes not solving them and giving up).

So Netmiko intends to simplify this lower-level SSH management across a wide set of networking vendors and platforms.

As of June 2015, Netmiko has support for the following platforms:

* Cisco IOS
* Cisco IOS-XE
* Cisco ASA
* Cisco NX-OS
* Cisco IOS-XR
* Cisco WLC (limited testing)
* Arista EOS
* HP ProCurve
* HP Comware (limited testing)
* Juniper Junos
* Brocade VDX (limited testing)
* F5 LTM (experimental)
* Huawei (limited testing)

**As with anything involving programming, test extensively in your own environment and for your own specific use cases. It is your responsibility to know what you are doing and to have good testing processes in place.**

If I try to run a python program that uses Netmiko I get an error:

root@NetworkAutomation-1:~# **python3 netmiko1.py**

Traceback (most recent call last):

File "netmiko1.py", line 2, in <module>

from netmiko import ConnectHandler

ImportError: No module named 'netmiko'

To fix the above I have to do the following:

root@NetworkAutomation-1:~# **apt-get update**

Get:1 http://security.ubuntu.com/ubuntu xenial-security InRelease [107 kB]

Get:2 http://archive.ubuntu.com/ubuntu xenial InRelease [247 kB]

Get:3 http://ppa.launchpad.net/ansible/ansible-2.5/ubuntu xenial InRelease [18.0 kB]

Get:4 http://archive.ubuntu.com/ubuntu xenial-updates InRelease [109 kB]

Get:5 http://archive.ubuntu.com/ubuntu xenial-backports InRelease [107 kB]

Get:6 http://ppa.launchpad.net/ansible/ansible-2.5/ubuntu xenial/main amd64 Packages [469 B]

Get:7 http://security.ubuntu.com/ubuntu xenial-security/universe Sources [90.3 kB]

Get:8 http://security.ubuntu.com/ubuntu xenial-security/main amd64 Packages [703 kB]

Get:9 http://archive.ubuntu.com/ubuntu xenial/universe Sources [9802 kB]

Get:10 http://security.ubuntu.com/ubuntu xenial-security/restricted amd64 Packages [12.7 kB]

Get:11 http://security.ubuntu.com/ubuntu xenial-security/universe amd64 Packages [468 kB]

Get:12 http://security.ubuntu.com/ubuntu xenial-security/multiverse amd64 Packages [3748 B]

Get:13 http://archive.ubuntu.com/ubuntu xenial/main amd64 Packages [1558 kB]

Get:14 http://archive.ubuntu.com/ubuntu xenial/restricted amd64 Packages [14.1 kB]

Get:15 http://archive.ubuntu.com/ubuntu xenial/universe amd64 Packages [9827 kB]

Get:16 http://archive.ubuntu.com/ubuntu xenial/multiverse amd64 Packages [176 kB]

Get:17 http://archive.ubuntu.com/ubuntu xenial-updates/universe Sources [277 kB]

Get:18 http://archive.ubuntu.com/ubuntu xenial-updates/main amd64 Packages [1088 kB]

Get:19 http://archive.ubuntu.com/ubuntu xenial-updates/restricted amd64 Packages [13.1 kB]

Get:20 http://archive.ubuntu.com/ubuntu xenial-updates/universe amd64 Packages [878 kB]

Get:21 http://archive.ubuntu.com/ubuntu xenial-updates/multiverse amd64 Packages [18.8 kB]

Get:22 http://archive.ubuntu.com/ubuntu xenial-backports/main amd64 Packages [7343 B]

Get:23 http://archive.ubuntu.com/ubuntu xenial-backports/universe amd64 Packages [8086 B]

Fetched 25.5 MB in 30s (850 kB/s)

Reading package lists... Done

root@NetworkAutomation-1:~# **apt-get install python3-pip**

Reading package lists... Done

Building dependency tree

Reading state information... Done

python3-pip is already the newest version (8.1.1-2ubuntu0.4).

0 upgraded, 0 newly installed, 0 to remove and 55 not upgraded.

root@NetworkAutomation-1:~# **apt-get install python3-dev**

root@NetworkAutomation-1:~# **pip3 install -U netmiko**

Collecting netmiko

Using cached https://files.pythonhosted.org/packages/1c/82/9aef0c1b4fca782fd18bee44c9ed18404060f1c04ecbf1386594f5f41834/netmiko-2.2.2.tar.gz

Collecting paramiko>=2.0.0 (from netmiko)

Using cached https://files.pythonhosted.org/packages/3e/db/cb7b6656e0e7387637ce850689084dc0b94b44df31cc52e5fc5c2c4fd2c1/paramiko-2.4.1-py2.py3-none-any.whl

Collecting scp>=0.10.0 (from netmiko)

Using cached https://files.pythonhosted.org/packages/ef/8c/399c5ddffc8c1bd2593ee6c8937ed31f005dcfa0c591da149f7b4b993f14/scp-0.11.0-py2.py3-none-any.whl

Collecting pyyaml (from netmiko)

Using cached https://files.pythonhosted.org/packages/9e/a3/1d13970c3f36777c583f136c136f804d70f500168edc1edea6daa7200769/PyYAML-3.13.tar.gz

Collecting pyserial (from netmiko)

Using cached https://files.pythonhosted.org/packages/0d/e4/2a744dd9e3be04a0c0907414e2a01a7c88bb3915cbe3c8cc06e209f59c30/pyserial-3.4-py2.py3-none-any.whl

Collecting textfsm (from netmiko)

Using cached https://files.pythonhosted.org/packages/a1/0d/a1b490503545b3b4600b965eae5d44cc2b6ce27cfb44f4debc563dbb56d3/textfsm-0.4.1.tar.gz

Requirement already up-to-date: pynacl>=1.0.1 in /usr/local/lib/python3.5/dist-packages (from paramiko>=2.0.0->netmiko)

Collecting cryptography>=1.5 (from paramiko>=2.0.0->netmiko)

Using cached https://files.pythonhosted.org/packages/22/21/233e38f74188db94e8451ef6385754a98f3cad9b59bedf3a8e8b14988be4/cryptography-2.3.1.tar.gz

Requirement already up-to-date: bcrypt>=3.1.3 in /usr/local/lib/python3.5/dist-packages (from paramiko>=2.0.0->netmiko)

Requirement already up-to-date: pyasn1>=0.1.7 in /usr/local/lib/python3.5/dist-packages (from paramiko>=2.0.0->netmiko)

Requirement already up-to-date: six in /usr/local/lib/python3.5/dist-packages (from pynacl>=1.0.1->paramiko>=2.0.0->netmiko)

Requirement already up-to-date: cffi>=1.4.1 in /usr/local/lib/python3.5/dist-packages (from pynacl>=1.0.1->paramiko>=2.0.0->netmiko)

Requirement already up-to-date: idna>=2.1 in /usr/local/lib/python3.5/dist-packages (from cryptography>=1.5->paramiko>=2.0.0->netmiko)

Requirement already up-to-date: asn1crypto>=0.21.0 in /usr/local/lib/python3.5/dist-packages (from cryptography>=1.5->paramiko>=2.0.0->netmiko)

Requirement already up-to-date: pycparser in /usr/local/lib/python3.5/dist-packages (from cffi>=1.4.1->pynacl>=1.0.1->paramiko>=2.0.0->netmiko)

Building wheels for collected packages: netmiko, pyyaml, textfsm, cryptography

Running setup.py bdist\_wheel for netmiko ... error

Complete output from command /usr/bin/python3 -u -c "import setuptools, tokenize;\_\_file\_\_='/tmp/pip-build-ajv85x1q/netmiko/setup.py';exec(compile(getattr(tokenize, 'open', open)(\_\_file\_\_).read().replace('\r\n', '\n'), \_\_file\_\_, 'exec'))" bdist\_wheel -d /tmp/tmpx8aep4i7pip-wheel- --python-tag cp35:

/usr/lib/python3.5/distutils/dist.py:261: UserWarning: Unknown distribution option: 'long\_description\_content\_type'

warnings.warn(msg)

usage: -c [global\_opts] cmd1 [cmd1\_opts] [cmd2 [cmd2\_opts] ...]

or: -c --help [cmd1 cmd2 ...]

or: -c --help-commands

or: -c cmd --help

error: invalid command 'bdist\_wheel'

----------------------------------------

Failed building wheel for netmiko

Running setup.py clean for netmiko

Running setup.py bdist\_wheel for pyyaml ... error

Complete output from command /usr/bin/python3 -u -c "import setuptools, tokenize;\_\_file\_\_='/tmp/pip-build-ajv85x1q/pyyaml/setup.py';exec(compile(getattr(tokenize, 'open', open)(\_\_file\_\_).read().replace('\r\n', '\n'), \_\_file\_\_, 'exec'))" bdist\_wheel -d /tmp/tmp8tt0aplmpip-wheel- --python-tag cp35:

usage: -c [global\_opts] cmd1 [cmd1\_opts] [cmd2 [cmd2\_opts] ...]

or: -c --help [cmd1 cmd2 ...]

or: -c --help-commands

or: -c cmd --help

error: invalid command 'bdist\_wheel'

----------------------------------------

Failed building wheel for pyyaml

Running setup.py clean for pyyaml

Running setup.py bdist\_wheel for textfsm ... error

Complete output from command /usr/bin/python3 -u -c "import setuptools, tokenize;\_\_file\_\_='/tmp/pip-build-ajv85x1q/textfsm/setup.py';exec(compile(getattr(tokenize, 'open', open)(\_\_file\_\_).read().replace('\r\n', '\n'), \_\_file\_\_, 'exec'))" bdist\_wheel -d /tmp/tmp34iogmyupip-wheel- --python-tag cp35:

/usr/lib/python3.5/distutils/dist.py:261: UserWarning: Unknown distribution option: 'long\_description\_content\_type'

warnings.warn(msg)

usage: -c [global\_opts] cmd1 [cmd1\_opts] [cmd2 [cmd2\_opts] ...]

or: -c --help [cmd1 cmd2 ...]

or: -c --help-commands

or: -c cmd --help

error: invalid command 'bdist\_wheel'

----------------------------------------

Failed building wheel for textfsm

Running setup.py clean for textfsm

Running setup.py bdist\_wheel for cryptography ... error

Complete output from command /usr/bin/python3 -u -c "import setuptools, tokenize;\_\_file\_\_='/tmp/pip-build-ajv85x1q/cryptography/setup.py';exec(compile(getattr(tokenize, 'open', open)(\_\_file\_\_).read().replace('\r\n', '\n'), \_\_file\_\_, 'exec'))" bdist\_wheel -d /tmp/tmphhryp5q9pip-wheel- --python-tag cp35:

/usr/lib/python3.5/distutils/dist.py:261: UserWarning: Unknown distribution option: 'python\_requires'

warnings.warn(msg)

usage: -c [global\_opts] cmd1 [cmd1\_opts] [cmd2 [cmd2\_opts] ...]

or: -c --help [cmd1 cmd2 ...]

or: -c --help-commands

or: -c cmd --help

error: invalid command 'bdist\_wheel'

----------------------------------------

Failed building wheel for cryptography

Running setup.py clean for cryptography

Failed to build netmiko pyyaml textfsm cryptography

Installing collected packages: cryptography, paramiko, scp, pyyaml, pyserial, textfsm, netmiko

Running setup.py install for cryptography ... / done

Running setup.py install for pyyaml ... done

Running setup.py install for textfsm ... done

Running setup.py install for netmiko ... done

Successfully installed cryptography-2.3.1 netmiko-2.2.2 paramiko-2.4.1 pyserial-3.4 pyyaml-3.13 scp-0.11.0 textfsm-0.4.1

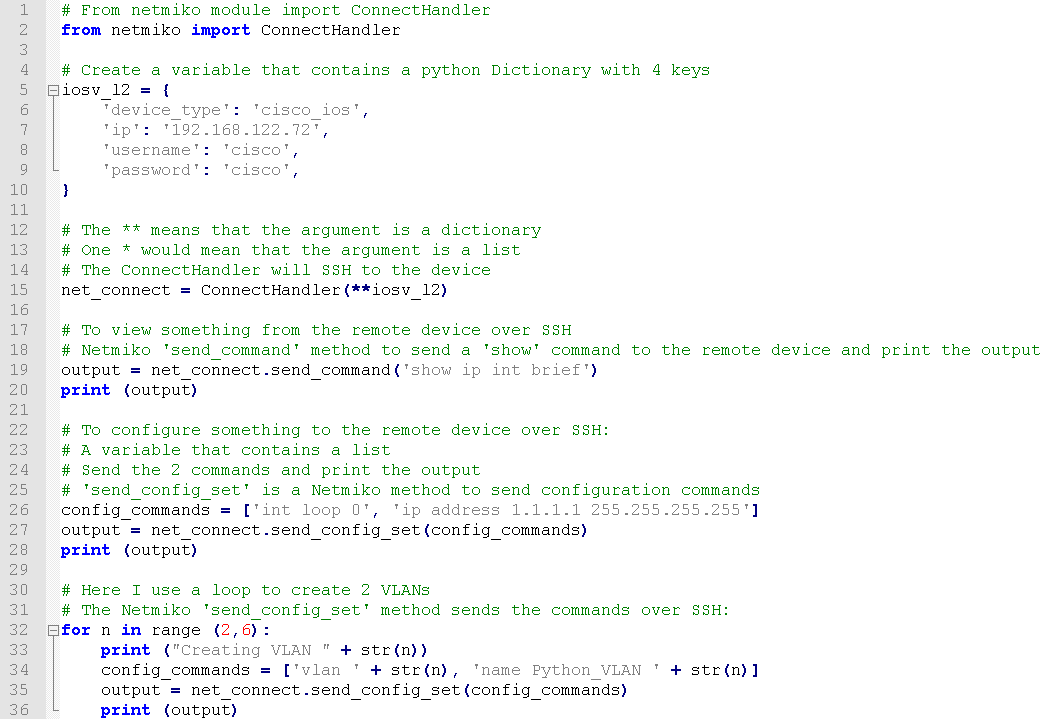
You are using pip version 8.1.1, however version 18.0 is available.

You should consider upgrading via the 'pip install --upgrade pip' command.

### Section 2 (netmiko) - Script 1 - configuring a single switch

The following script:

* Connects to one device via SSH
* The IP of the device is manually specified in the script
* It runs a 'show' command
* It runs few 'config' comands



# From netmiko module import ConnectHandler

from netmiko import ConnectHandler

# Create a variable that contains a python Dictionary with 4 keys

iosv\_l2 = {

'device\_type': 'cisco\_ios',

'ip': '192.168.122.72',

'username': 'cisco',

'password': 'cisco',

}

# The \*\* means that the argument is a dictionary

# One \* would mean that the argument is a list

# The ConnectHandler will SSH to the device

net\_connect = ConnectHandler(\*\*iosv\_l2)

# To view something from the remote device over SSH

# Netmiko 'send\_command' method to send a 'show' command to the remote device and print the output

output = net\_connect.send\_command('show ip int brief')

print (output)

# To configure something to the remote device over SSH:

# A variable that contains a list

# Send the 2 commands and print the output

# 'send\_config\_set' is a Netmiko method to send configuration commands

config\_commands = ['int loop 0', 'ip address 1.1.1.1 255.255.255.255']

output = net\_connect.send\_config\_set(config\_commands)

print (output)

# Here I use a loop to create 2 VLANs

# The Netmiko 'send\_config\_set' method sends the commands over SSH:

for n in range (2,6):

print ("Creating VLAN " + str(n))

config\_commands = ['vlan ' + str(n), 'name Python\_VLAN ' + str(n)]

output = net\_connect.send\_config\_set(config\_commands)

print (output)

**Running the script**

root@NetworkAutomation-1:~# **python3 netmiko1.py**

Interface IP-Address OK? Method Status Protocol

Ethernet0/0 unassigned YES unset up up

Ethernet0/1 unassigned YES unset up up

Ethernet0/2 unassigned YES unset up up

Ethernet0/3 unassigned YES unset up up

Ethernet1/0 unassigned YES unset up up

Ethernet1/1 unassigned YES unset up up

Ethernet1/2 unassigned YES unset up up

Ethernet1/3 unassigned YES unset up up

...

Loopback0 1.1.1.1 YES manual up up

Vlan1 192.168.122.72 YES NVRAM up up

config term

Enter configuration commands, one per line. End with CNTL/Z.

SW1(config)#int loop 0

SW1(config-if)#ip address 1.1.1.1 255.255.255.255

SW1(config-if)#end

SW1#

Creating VLAN 2

config term

Enter configuration commands, one per line. End with CNTL/Z.

SW1(config)#vlan 2

SW1(config-vlan)#name Python\_VLAN 2

SW1(config-vlan)#end

SW1#

Creating VLAN 3

config term

Enter configuration commands, one per line. End with CNTL/Z.

SW1(config)#vlan 3

SW1(config-vlan)#name Python\_VLAN 3

SW1(config-vlan)#end

SW1#

Creating VLAN 4

config term

Enter configuration commands, one per line. End with CNTL/Z.

SW1(config)#vlan 4

SW1(config-vlan)#name Python\_VLAN 4

SW1(config-vlan)#end

SW1#

Creating VLAN 5

config term

Enter configuration commands, one per line. End with CNTL/Z.

SW1(config)#vlan 5

SW1(config-vlan)#name Python\_VLAN 5

SW1(config-vlan)#end

SW1#

root@NetworkAutomation-1:~#

**Verification**

SW1#**show vlan brief**

VLAN Name Status Ports

---- -------------------------------- --------- -------------------------------

1 default active Et0/0, Et0/1, Et0/2, Et0/3

Et1/2, Et1/3, Et2/0, Et2/1

Et2/2, Et2/3, Et3/0, Et3/1

Et3/2, Et3/3

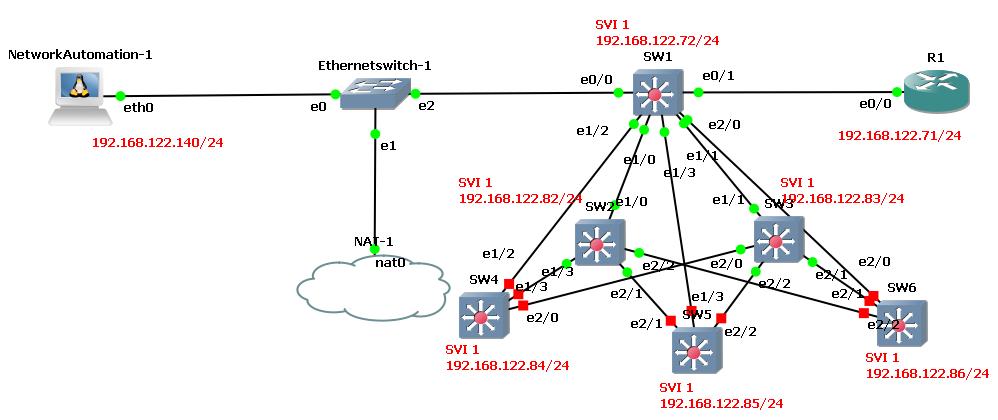
2 Python\_VLAN 2 active

3 Python\_VLAN 3 active

4 Python\_VLAN 4 active

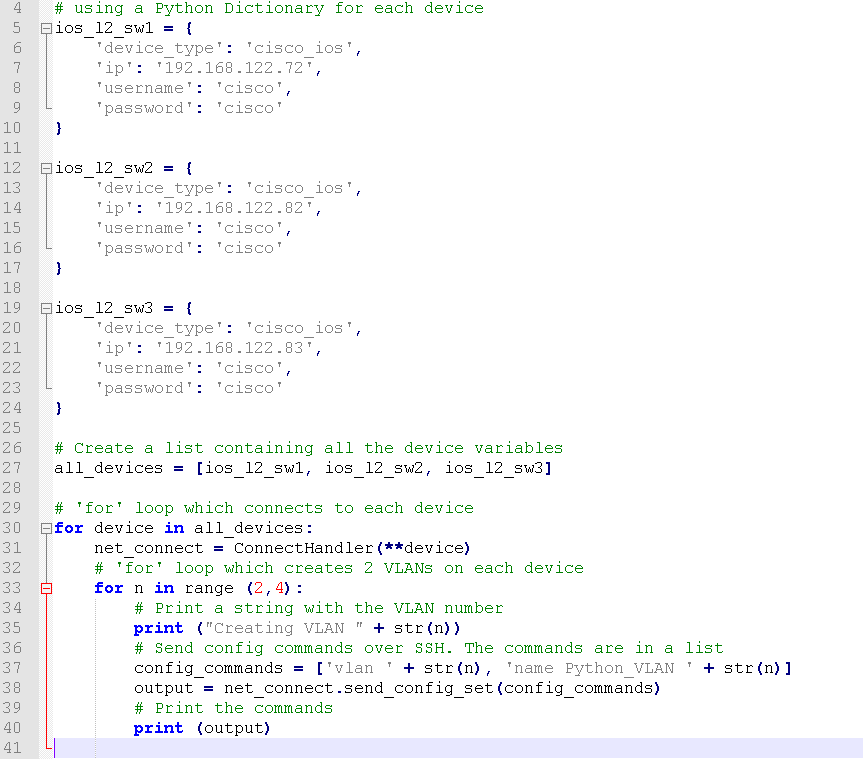
5 Python\_VLAN 5 active

### Section 2 (netmiko) - Script 2 - configuring multiple switches



The following script:

* Connects to multiple devices via SSH
* The IP of the devices is manually specified in the script
* It runs few 'config' comands



from netmiko import ConnectHandler

# Create one variable for each remove device

# using a Python Dictionary for each device

ios\_l2\_sw1 = {

'device\_type': 'cisco\_ios',

'ip': '192.168.122.72',

'username': 'cisco',

'password': 'cisco'

}

ios\_l2\_sw2 = {

'device\_type': 'cisco\_ios',

'ip': '192.168.122.82',

'username': 'cisco',

'password': 'cisco'

}

ios\_l2\_sw3 = {

'device\_type': 'cisco\_ios',

'ip': '192.168.122.83',

'username': 'cisco',

'password': 'cisco'

}

# Create a list containing all the device variables

all\_devices = [ios\_l2\_sw1, ios\_l2\_sw2, ios\_l2\_sw3]

# 'for' loop which connects to each device

for device in all\_devices:

net\_connect = ConnectHandler(\*\*device)

# 'for' loop which creates 2 VLANs on each device

for n in range (2,4):

# Print a string with the VLAN number

print ("Creating VLAN " + str(n))

# Send config commands over SSH. The commands are in a list

config\_commands = ['vlan ' + str(n), 'name Python\_VLAN ' + str(n)]

output = net\_connect.send\_config\_set(config\_commands)

# Print the commands

print (output)

**Running the script**

root@NetworkAutomation-1:~# **python3 netmiko2.py**

Creating VLAN 2

config term

Enter configuration commands, one per line. End with CNTL/Z.

SW1(config)#vlan 2

SW1(config-vlan)#name Python\_VLAN 2

SW1(config-vlan)#end

SW1#

Creating VLAN 3

config term

Enter configuration commands, one per line. End with CNTL/Z.

SW1(config)#vlan 3

SW1(config-vlan)#name Python\_VLAN 3

SW1(config-vlan)#end

SW1#

Creating VLAN 2

config term

Enter configuration commands, one per line. End with CNTL/Z.

SW2(config)#vlan 2

SW2(config-vlan)#name Python\_VLAN 2

SW2(config-vlan)#end

SW2#

Creating VLAN 3

config term

Enter configuration commands, one per line. End with CNTL/Z.

SW2(config)#vlan 3

SW2(config-vlan)#name Python\_VLAN 3

SW2(config-vlan)#end

SW2#

Creating VLAN 2

config term

Enter configuration commands, one per line. End with CNTL/Z.

SW3(config)#vlan 2

SW3(config-vlan)#name Python\_VLAN 2

SW3(config-vlan)#end

SW3#

Creating VLAN 3

config term

Enter configuration commands, one per line. End with CNTL/Z.

SW3(config)#vlan 3

SW3(config-vlan)#name Python\_VLAN 3

SW3(config-vlan)#end

SW3#

root@NetworkAutomation-1:~#

**Verification**

When the script is not connected to the device:

SW3#**show ssh**

%No SSHv2 server connections running.

%No SSHv1 server connections running.

When it is connected:

SW3#**show ssh**

Connection Version Mode Encryption Hmac State Username

0 2.0 IN aes128-cbc hmac-sha1 Session started cisco

0 2.0 OUT aes128-cbc hmac-sha1 Session started cisco

SW3#**show vlan brief**

VLAN Name Status Ports

---- -------------------------------- --------- -------------------------------

1 default active Et0/0, Et0/1, Et0/2, Et0/3

Et1/0, Et1/2, Et1/3, Et2/0

Et2/1, Et2/2, Et2/3, Et3/0

Et3/1, Et3/2, Et3/3

2 Python\_VLAN 2 active

3 Python\_VLAN 3 active

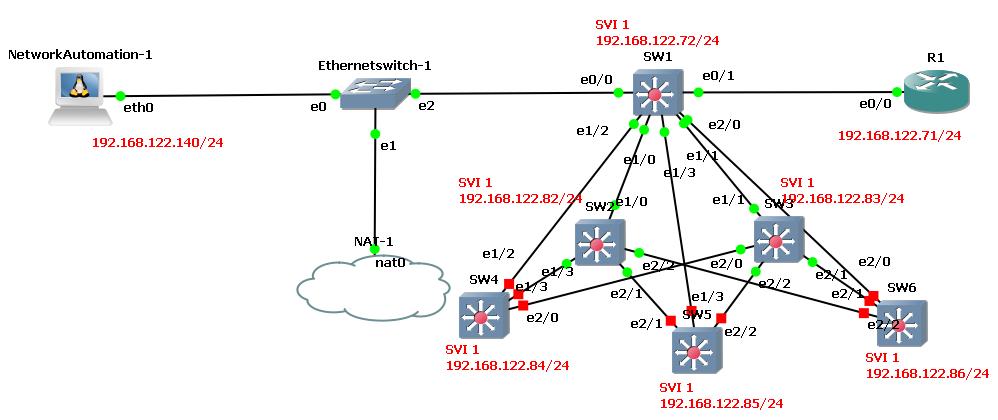
1002 fddi-default act/unsup

1003 token-ring-default act/unsup

1004 fddinet-default act/unsup

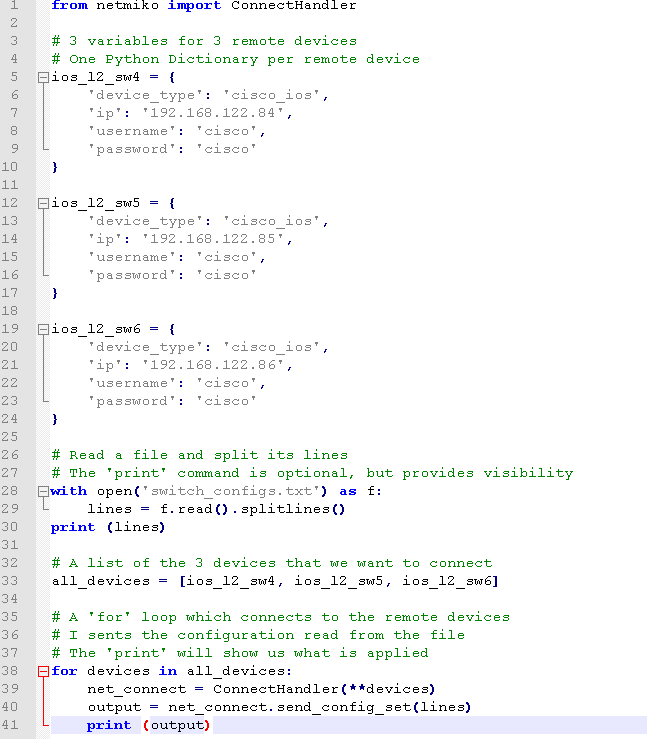
1005 trnet-default act/unsup

### Section 2 (netmiko) - Script 3 - configuring multiple switches using a file config



The following script:

* Connects to multiple devices via SSH
* The IP of the devices is manually specified in the script
* It reads a file which contains configuration and applies it



from netmiko import ConnectHandler

# 3 variables for 3 remote devices

# One Python Dictionary per remote device

ios\_l2\_sw4 = {

'device\_type': 'cisco\_ios',

'ip': '192.168.122.84',

'username': 'cisco',

'password': 'cisco'

}

ios\_l2\_sw5 = {

'device\_type': 'cisco\_ios',

'ip': '192.168.122.85',

'username': 'cisco',

'password': 'cisco'

}

ios\_l2\_sw6 = {

'device\_type': 'cisco\_ios',

'ip': '192.168.122.86',

'username': 'cisco',

'password': 'cisco'

}

# Read a file and split its lines

# The 'print' command is optional, but provides visibility

with open('switch\_configs.txt') as f:

lines = f.read().splitlines()

print (lines)

# A list of the 3 devices that we want to connect

all\_devices = [ios\_l2\_sw4, ios\_l2\_sw5, ios\_l2\_sw6]

# A 'for' loop which connects to the remote devices

# I sents the configuration read from the file

# The 'print' will show us what is applied

for devices in all\_devices:

net\_connect = ConnectHandler(\*\*devices)

output = net\_connect.send\_config\_set(lines)

print (output)

**Running the script**

root@NetworkAutomation-1:~# **python3 netmiko3.py**

['vlan 55', ' name VLAN\_55', '!', 'router ospf 1', ' network 0.0.0.0 255.255.255.255 area 0 ', '!']

config term

Enter configuration commands, one per line. End with CNTL/Z.

SW4(config)#vlan 55

SW4(config-vlan)# name VLAN\_55

SW4(config-vlan)#!

SW4(config-vlan)#router ospf 1

SW4(config-router)# network 0.0.0.0 255.255.255.255 area 0

SW4(config-router)#!

SW4(config-router)#end

SW4#

config term

Enter configuration commands, one per line. End with CNTL/Z.

SW5(config)#vlan 55

SW5(config-vlan)# name VLAN\_55

SW5(config-vlan)#!

SW5(config-vlan)#router ospf 1

SW5(config-router)# network 0.0.0.0 255.255.255.255 area 0

SW5(config-router)#!

SW5(config-router)#end

SW5#

config term

Enter configuration commands, one per line. End with CNTL/Z.

SW6(config)#vlan 55

SW6(config-vlan)# name VLAN\_55

SW6(config-vlan)#!

SW6(config-vlan)#router ospf 1

SW6(config-router)# network 0.0.0.0 255.255.255.255 area 0

SW6(config-router)#!

SW6(config-router)#end

SW6#

root@NetworkAutomation-1:~#

**Verification**

SW4#**sh run | s router**

router ospf 1

network 0.0.0.0 255.255.255.255 area 0

SW4#**sh ip ospf int br**

Interface PID Area IP Address/Mask Cost State Nbrs F/C

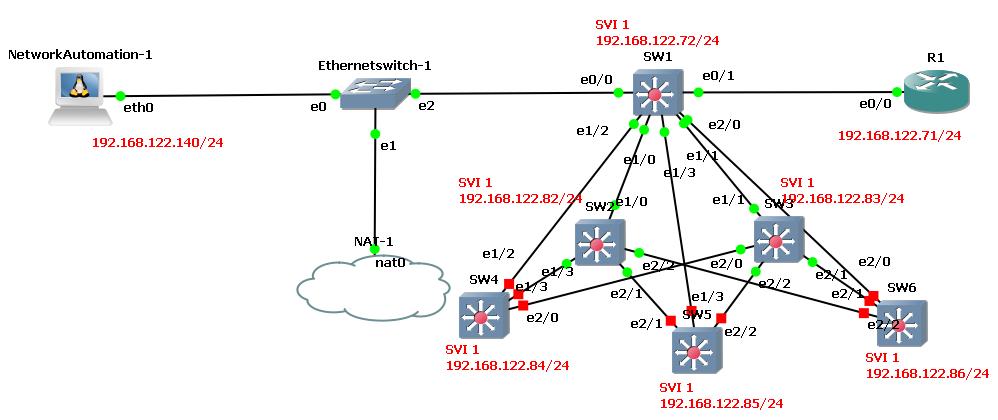
Vl1 1 0 192.168.122.84/24 1 DROTH 0/2

SW4#

\*Sep 10 18:02:51.524: %OSPF-5-ADJCHG: Process 1, Nbr 192.168.122.86 on Vlan1 from LOADING to FULL, Loading Done

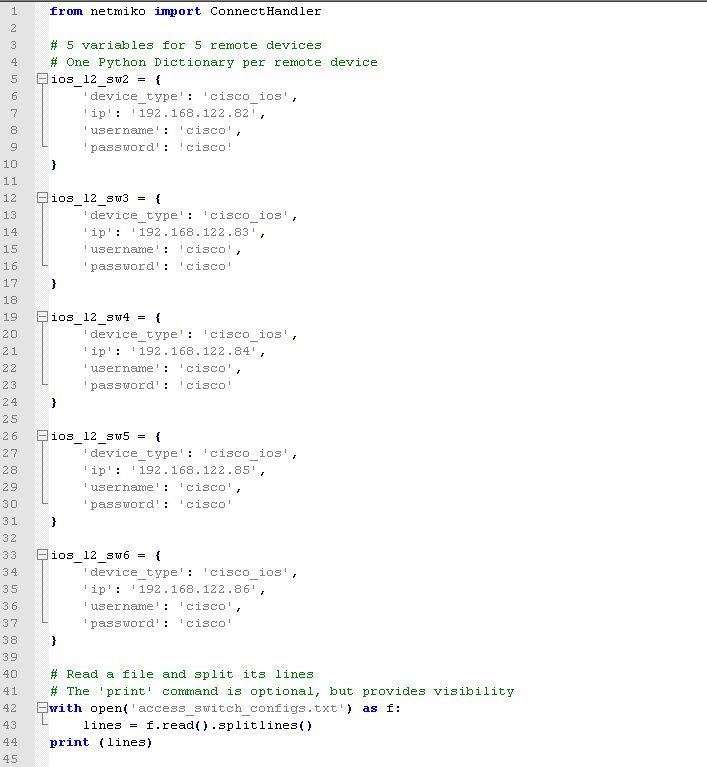
\*Sep 10 18:02:59.016: %OSPF-5-ADJCHG: Process 1, Nbr 192.168.122.85 on Vlan1 from LOADING to FULL, Loading Done

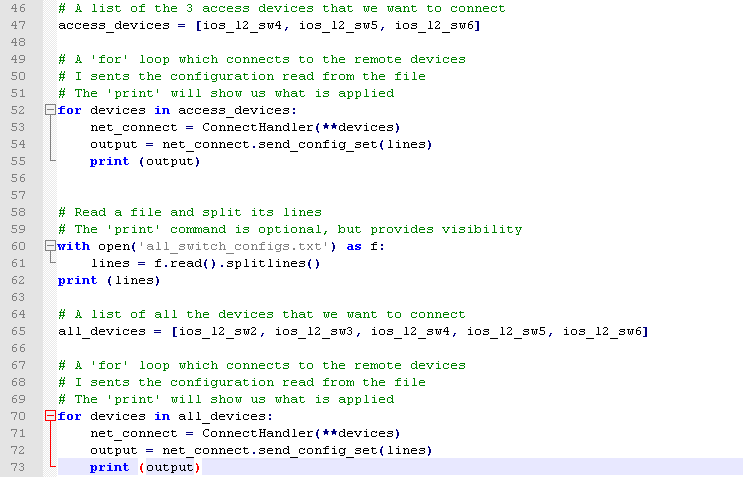
### Section 2 (netmiko) - Script 4 - configuring multiple switches using a 2 files with config



The following script:

* Connects to multiple devices via SSH
* The IP of the devices is manually specified in the script
* It reads a file which contains configuration and applies it to some switches
* It reads another file and applies the configuration to all switches





from netmiko import ConnectHandler

# 5 variables for 5 remote devices

# One Python Dictionary per remote device

ios\_l2\_sw2 = {

'device\_type': 'cisco\_ios',

'ip': '192.168.122.82',

'username': 'cisco',

'password': 'cisco'

}

ios\_l2\_sw3 = {

'device\_type': 'cisco\_ios',

'ip': '192.168.122.83',

'username': 'cisco',

'password': 'cisco'

}

ios\_l2\_sw4 = {

'device\_type': 'cisco\_ios',

'ip': '192.168.122.84',

'username': 'cisco',

'password': 'cisco'

}

ios\_l2\_sw5 = {

'device\_type': 'cisco\_ios',

'ip': '192.168.122.85',

'username': 'cisco',

'password': 'cisco'

}

ios\_l2\_sw6 = {

'device\_type': 'cisco\_ios',

'ip': '192.168.122.86',

'username': 'cisco',

'password': 'cisco'

}

# Read a file and split its lines

# The 'print' command is optional, but provides visibility

with open('access\_switch\_configs.txt') as f:

lines = f.read().splitlines()

print (lines)

# A list of the 3 access devices that we want to connect

access\_devices = [ios\_l2\_sw4, ios\_l2\_sw5, ios\_l2\_sw6]

# A 'for' loop which connects to the remote devices

# I sents the configuration read from the file

# The 'print' will show us what is applied

for devices in access\_devices:

net\_connect = ConnectHandler(\*\*devices)

output = net\_connect.send\_config\_set(lines)

print (output)

# Read a file and split its lines

# The 'print' command is optional, but provides visibility

with open('all\_switch\_configs.txt') as f:

lines = f.read().splitlines()

print (lines)

# A list of all the devices that we want to connect

all\_devices = [ios\_l2\_sw2, ios\_l2\_sw3, ios\_l2\_sw4, ios\_l2\_sw5, ios\_l2\_sw6]

# A 'for' loop which connects to the remote devices

# I sents the configuration read from the file

# The 'print' will show us what is applied

for devices in all\_devices:

net\_connect = ConnectHandler(\*\*devices)

output = net\_connect.send\_config\_set(lines)

print (output)

**Running the script**

root@NetworkAutomation-1:~# **python3 netmiko4.py**

['vlan 55', ' name VLAN\_55', '!', 'router ospf 1', ' network 0.0.0.0 255.255.255.255 area 0 ', '!']

config term

Enter configuration commands, one per line. End with CNTL/Z.

SW4(config)#vlan 55

SW4(config-vlan)# name VLAN\_55

SW4(config-vlan)#!

SW4(config-vlan)#router ospf 1

SW4(config-router)# network 0.0.0.0 255.255.255.255 area 0

SW4(config-router)#!

SW4(config-router)#end

SW4#

config term

Enter configuration commands, one per line. End with CNTL/Z.

SW5(config)#vlan 55

SW5(config-vlan)# name VLAN\_55

SW5(config-vlan)#!

SW5(config-vlan)#router ospf 1

SW5(config-router)# network 0.0.0.0 255.255.255.255 area 0

SW5(config-router)#!

SW5(config-router)#end

SW5#

config term

Enter configuration commands, one per line. End with CNTL/Z.

SW6(config)#vlan 55

SW6(config-vlan)# name VLAN\_55

SW6(config-vlan)#!

SW6(config-vlan)#router ospf 1

SW6(config-router)# network 0.0.0.0 255.255.255.255 area 0

SW6(config-router)#!

SW6(config-router)#end

SW6#

['vlan 66', ' name VLAN\_66', '!', 'router eigrp 1', ' network 0.0.0.0 255.255.255.255', '!']

config term

Enter configuration commands, one per line. End with CNTL/Z.

SW2(config)#vlan 66

SW2(config-vlan)# name VLAN\_66

SW2(config-vlan)#!

SW2(config-vlan)#router eigrp 1

SW2(config-router)# network 0.0.0.0 255.255.255.255

SW2(config-router)#!

SW2(config-router)#end

SW2#

config term

Enter configuration commands, one per line. End with CNTL/Z.

SW3(config)#vlan 66

SW3(config-vlan)# name VLAN\_66

SW3(config-vlan)#!

SW3(config-vlan)#router eigrp 1

SW3(config-router)# network 0.0.0.0 255.255.255.255

SW3(config-router)#!

SW3(config-router)#end

SW3#

config term

Enter configuration commands, one per line. End with CNTL/Z.

SW4(config)#vlan 66

SW4(config-vlan)# name VLAN\_66

SW4(config-vlan)#!

SW4(config-vlan)#router eigrp 1

SW4(config-router)# network 0.0.0.0 255.255.255.255

SW4(config-router)#!

SW4(config-router)#end

SW4#

config term

Enter configuration commands, one per line. End with CNTL/Z.

SW5(config)#vlan 66

SW5(config-vlan)# name VLAN\_66

SW5(config-vlan)#!

SW5(config-vlan)#router eigrp 1

SW5(config-router)# network 0.0.0.0 255.255.255.255

SW5(config-router)#!

SW5(config-router)#end

SW5#

config term

Enter configuration commands, one per line. End with CNTL/Z.

SW6(config)#vlan 66

SW6(config-vlan)# name VLAN\_66

SW6(config-vlan)#!

SW6(config-vlan)#router eigrp 1

SW6(config-router)# network 0.0.0.0 255.255.255.255

SW6(config-router)#!

SW6(config-router)#end

SW6#

root@NetworkAutomation-1:~#

**Verification**

SW2#

\*Sep 10 19:24:52.859: %SYS-5-CONFIG\_I: Configured from console by cisco on vty0 (192.168.122.220)

SW2#

\*Sep 10 19:25:04.520: %DUAL-5-NBRCHANGE: EIGRP-IPv4 1: Neighbor 192.168.122.83 (Vlan1) is up: new adjacency

SW2#

\*Sep 10 19:25:20.502: %DUAL-5-NBRCHANGE: EIGRP-IPv4 1: Neighbor 192.168.122.84 (Vlan1) is up: new adjacency

SW2#

\*Sep 10 19:25:36.795: %DUAL-5-NBRCHANGE: EIGRP-IPv4 1: Neighbor 192.168.122.85 (Vlan1) is up: new adjacency

SW2#

\*Sep 10 19:25:52.974: %DUAL-5-NBRCHANGE: EIGRP-IPv4 1: Neighbor 192.168.122.86 (Vlan1) is up: new adjacency

SW3#show vlan brief

VLAN Name Status Ports

---- -------------------------------- --------- -------------------------------

1 default active Et0/0, Et0/1, Et0/2, Et0/3

Et1/0, Et1/2, Et1/3, Et2/0

Et2/1, Et2/2, Et2/3, Et3/0

Et3/1, Et3/2, Et3/3

2 Python\_VLAN 2 active

3 Python\_VLAN 3 active

66 VLAN\_66 active

SW4#**show vlan brief**

VLAN Name Status Ports

---- -------------------------------- --------- -------------------------------

1 default active Et0/0, Et0/1, Et0/2, Et0/3

Et1/0, Et1/1, Et1/3, Et2/0

Et2/1, Et2/2, Et2/3, Et3/0

Et3/1, Et3/2, Et3/3

2 Python\_VLAN\_2 active

3 Python\_VLAN\_3 active

4 Python\_VLAN\_4 active

5 Python\_VLAN\_5 active

6 Python\_VLAN\_6 active

7 Python\_VLAN\_7 active

8 Python\_VLAN\_8 active

9 Python\_VLAN\_9 active

10 Python\_VLAN\_10 active

55 VLAN\_55 active

66 VLAN\_66 active

## Network Automation with NAPALM

From https://napalm-automation.net/

**What is NAPALM (Network Automation and Programmability Abstraction)?**

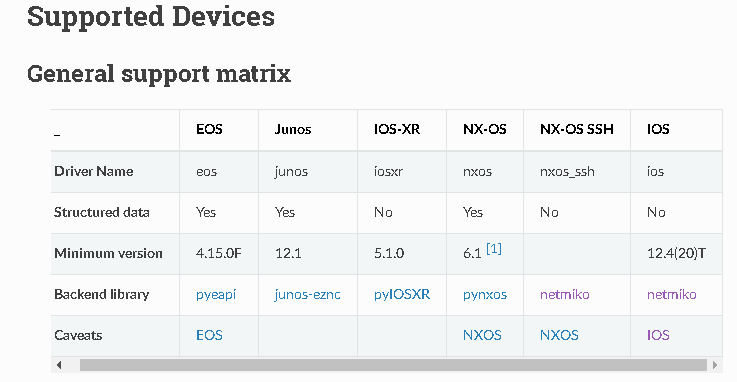
Napalm is a vendor neutral, cross-platform open source project that provides a unified API to network devices.

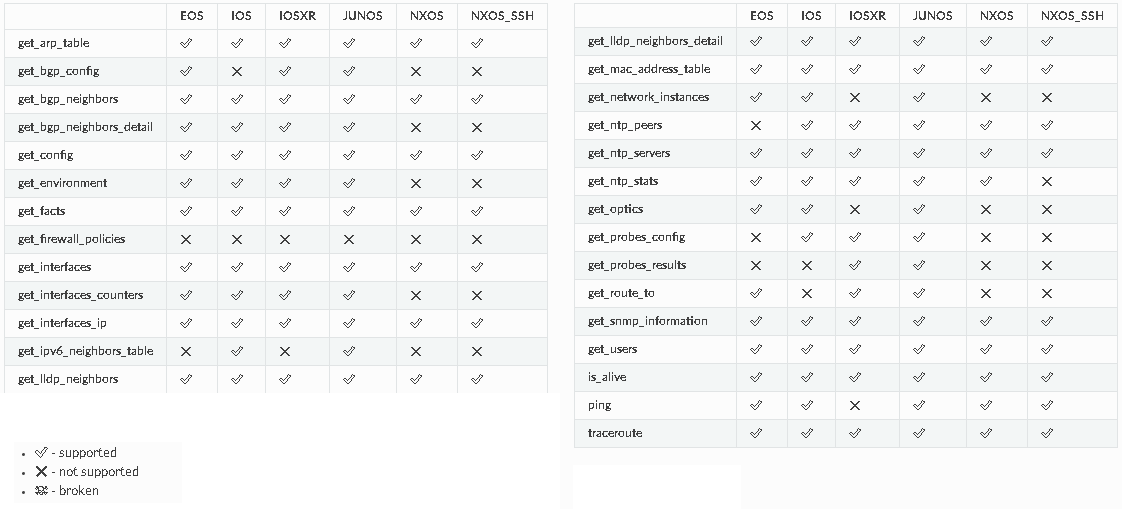
Napalm is written in Python and already works with the most popular automation frameworks.

**IOS Prerequisites**

IOS has no native API to play with, that’s the reason why we used the Netmiko library to interact with it. Having Netmiko installed in your working box is a prerequisite.

Supported Devices (Sep 2018)

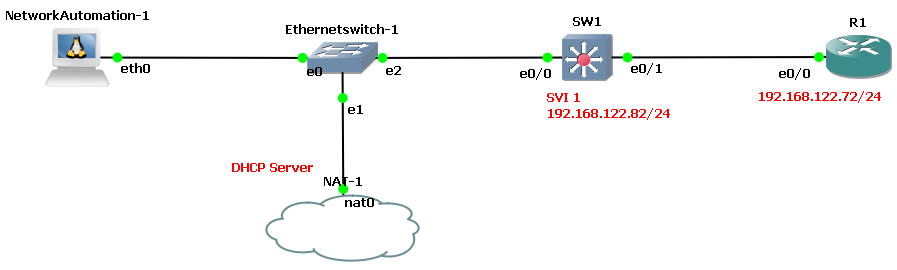




A sample from the past:



**Topology**



**Initial config**

R1/SW1:

ip domain-nam mikis.lab

username cisco password cisco

username cisco privilege 15

username mikis password cisco

username mikis privilege 15

crypto key generate rsa

1024

! % Generating 1024 bit RSA keys, keys will be non-exportable...

line vty 0 4

transport input ssh

login local

!

! for the switch will be SVI 1

int e0/0

ip add 192.168.122.72 255.255.255.0

no shut

**On NetworkAautomation-1 station enable DHCP Client on eth0 by un-commenting the following:**

root@NetworkAutomation-1:/etc/network# **nano interfaces**

*# DHCP config for eth0*

*auto eth0*

*iface eth0 inet dhcp*

**Note about nano: To save press CTRL-X, then 'Y' then Enter**

root@NetworkAutomation-1:/etc/network#

NetworkAutomation-1 console is now available... Press RETURN to get started.

udhcpc (v1.24.2) started

Sending discover...

Sending discover...

Sending select for 192.168.122.120...

Lease of 192.168.122.120 obtained, lease time 3600

**Verification**

root@NetworkAutomation-1:~# **ssh -l mikis 192.168.122.72**

The authenticity of host '192.168.122.72 (192.168.122.72)' can't be established.

RSA key fingerprint is SHA256:qYzhY6wCSH4TF3/G2gB7+8EPLyyAgcBwyXBCQEOQSCI.

Are you sure you want to continue connecting (yes/no)? **yes**

Warning: Permanently added '192.168.122.72' (RSA) to the list of known hosts.

Password:

R1#

**Install NAPALM**

**IMPORTANT**: Please also make sure you use the following commands to install Python3 Netmiko and NAPALM on the Network Automation Container:

**apt-get update**

**apt-get install python3-pip**

**apt-get install python3-dev**

**pip3 install -U netmiko**

At this point test netmiko functionality using the following script which runs a 'show' command on a remote device:

root@NetworkAutomation-1:~# **cat > netmiko-test.py**

*# From netmiko module import ConnectHandler*

*from netmiko import ConnectHandler*

*# Create a variable that contains a python Dictionary with 4 keys*

*iosv\_l2 = {*

*'device\_type': 'cisco\_ios',*

*'ip': '192.168.122.72',*

*'username': 'cisco',*

*'password': 'cisco',*

*}*

*# The \*\* means that the argument is a dictionary*

*# One \* would mean that the argument is a list*

*# The ConnectHandler will SSH to the device*

*net\_connect = ConnectHandler(\*\*iosv\_l2)*

*# To view something from the remote device over SSH*

*# Netmiko 'send\_command' method to send a 'show' command to the remote device and print the output*

*output = net\_connect.send\_command('show ip int brief')*

*print (output)*

root@NetworkAutomation-1:~#

root@NetworkAutomation-1:~# **python3 netmiko-test.py**

Interface IP-Address OK? Method Status Protocol

Ethernet0/0 192.168.122.72 YES manual up up

Ethernet0/1 unassigned YES NVRAM administratively down down

Ethernet0/2 unassigned YES NVRAM administratively down down

Ethernet0/3 unassigned YES NVRAM administratively down down

Ethernet1/0 unassigned YES NVRAM administratively down down

Ethernet1/1 unassigned YES NVRAM administratively down down

Ethernet1/2 unassigned YES NVRAM administratively down down

Ethernet1/3 unassigned YES NVRAM administratively down down

Serial2/0 unassigned YES NVRAM administratively down down

Serial2/1 unassigned YES NVRAM administratively down down

Serial2/2 unassigned YES NVRAM administratively down down

Serial2/3 unassigned YES NVRAM administratively down down

Serial3/0 unassigned YES NVRAM administratively down down

Serial3/1 unassigned YES NVRAM administratively down down

Serial3/2 unassigned YES NVRAM administratively down down

Serial3/3 unassigned YES NVRAM administratively down down

Loopback0 1.1.1.1 YES manual up up

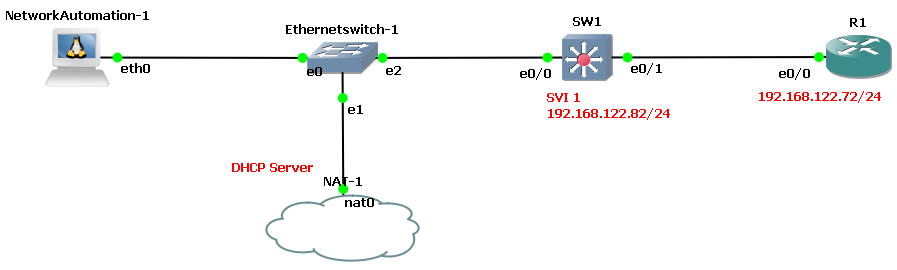
root@NetworkAutomation-1:~#

**pip3 install -U napalm**

**pip3 install -U simplejson**

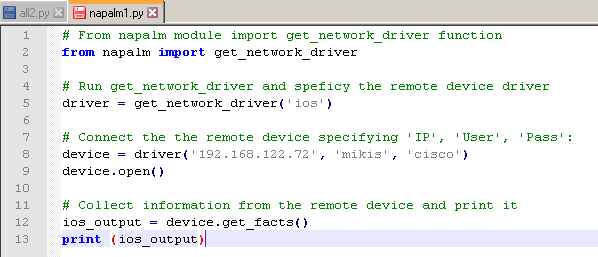
### Section 3 (napalm) - Script 1 - Get remote device info using NAPALM

**Topology**



The following script:

* Connects to a remote IOS device using SSH (Netmiko)
* Gets generic info from the remote device
* It prints the output as a Python Dictionary



root@NetworkAutomation-1:~# **cat > netmiko-test.py**

*# From napalm module import get\_network\_driver function*

*from napalm import get\_network\_driver*

*# Run get\_network\_driver and speficy the remote device driver*

*driver = get\_network\_driver('ios')*

*# Connect the the remote device specifying 'IP', 'User', 'Pass':*

*device = driver('192.168.122.72', 'mikis', 'cisco')*

*device.open()*

*# Collect information from the remote device and print it*

*ios\_output = device.get\_facts()*

*print (ios\_output)*

**Running the script**

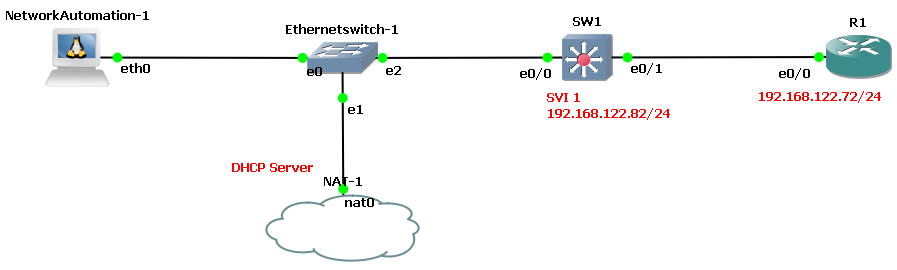
root@NetworkAutomation-1:~# **python3 napalm1.py**

{'serial\_number': '2048002', 'os\_version': 'Linux Software (I86BI\_LINUX-ADVENTERPRISEK9-M), Version 15.4(3)M9, DEVELOPMENT TEST SOFTWARE', 'fqdn': 'R1.mikis.lab', 'model': 'Unknown', 'interface\_list': ['Ethernet0/0', 'Ethernet0/1', 'Ethernet0/2', 'Ethernet0/3', 'Ethernet1/0', 'Ethernet1/1', 'Ethernet1/2', 'Ethernet1/3', 'Serial2/0', 'Serial2/1', 'Serial2/2', 'Serial2/3', 'Serial3/0', 'Serial3/1', 'Serial3/2', 'Serial3/3', 'Loopback0'], 'hostname': 'R1', 'uptime': 2340, 'vendor': 'Cisco'}

root@NetworkAutomation-1:~#

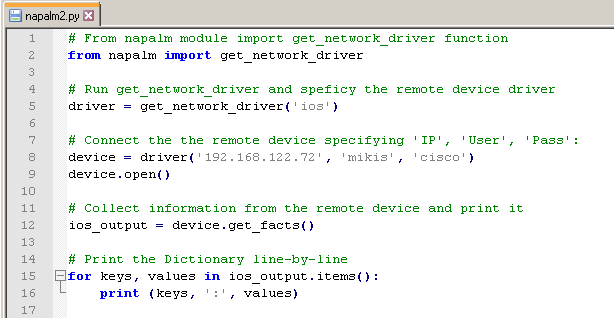
### Section 3 (napalm) - Script 2 - Get remote device info using NAPALM (improved output)

**Topology**



The following script:

* Connects to a remote IOS device using SSH (Netmiko)
* Gets generic info from the remote device
* It gets the remote output as a Python Dictionary, but it prints it line-by-line



root@NetworkAutomation-1:~# **cat > napalm2.py**

*# From napalm module import get\_network\_driver function*

*from napalm import get\_network\_driver*

*# Run get\_network\_driver and speficy the remote device driver*

*driver = get\_network\_driver('ios')*

*# Connect the the remote device specifying 'IP', 'User', 'Pass':*

*device = driver('192.168.122.72', 'mikis', 'cisco')*

*device.open()*

*# Collect information from the remote device and print it*

*ios\_output = device.get\_facts()*

*# Print the Dictionary line-by-line*

*for keys, values in ios\_output.items():*

*print (keys, ':', values)*

**Running the script**

root@NetworkAutomation-1:~# **python3 napalm2.py**

uptime : 9480

fqdn : R1.mikis.lab

vendor : Cisco

model : Unknown

serial\_number : 2048002

hostname : R1

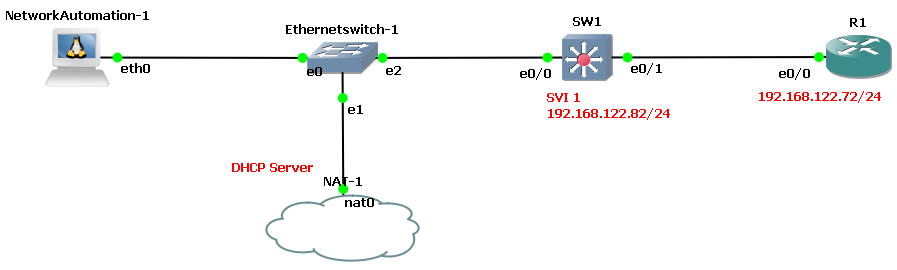
interface\_list : ['Ethernet0/0', 'Ethernet0/1', 'Ethernet0/2', 'Ethernet0/3', 'Ethernet1/0', 'Ethernet1/1', 'Ethernet1/2', 'Ethernet1/3', 'Serial2/0', 'Serial2/1', 'Serial2/2', 'Serial2/3', 'Serial3/0', 'Serial3/1', 'Serial3/2', 'Serial3/3', 'Loopback0']

os\_version : Linux Software (I86BI\_LINUX-ADVENTERPRISEK9-M), Version 15.4(3)M9, DEVELOPMENT TEST SOFTWARE

root@NetworkAutomation-1:~#

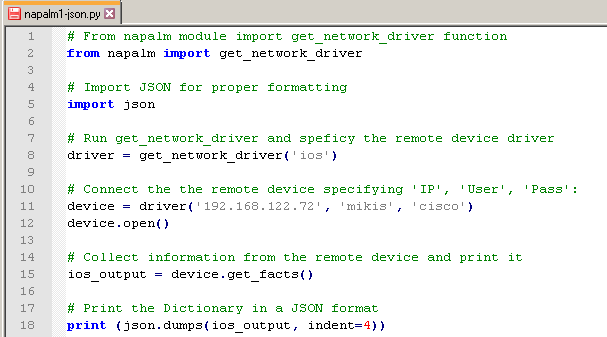
### Section 3 (napalm) - Script 3 - Get remote device info using NAPALM (improved output using JSON)

**Topology**



The following script:

* Connects to a remote IOS device using SSH (Netmiko)
* Gets generic info from the remote device
* It gets the remote output as a Python Dictionary and it prints it using JSON formatting



root@NetworkAutomation-1:~# **cat > napalm1-json.py**

*# From napalm module import get\_network\_driver function*

*from napalm import get\_network\_driver*

*# Import JSON for proper formatting*

*import json*

*# Run get\_network\_driver and speficy the remote device driver*

*driver = get\_network\_driver('ios')*

*# Connect the the remote device specifying 'IP', 'User', 'Pass':*

*device = driver('192.168.122.72', 'mikis', 'cisco')*

*device.open()*

*# Collect information from the remote device and print it*

*ios\_output = device.get\_facts()*

*# Print the Dictionary in a JSON format*

*print (json.dumps(ios\_output, indent=4))*

**Running the script**

root@NetworkAutomation-1:~# **python3 napalm1-json.py**

{

"vendor": "Cisco",

"interface\_list": [

"Ethernet0/0",

"Ethernet0/1",

"Ethernet0/2",

"Ethernet0/3",

"Ethernet1/0",

"Ethernet1/1",

"Ethernet1/2",

"Ethernet1/3",

"Serial2/0",

"Serial2/1",

"Serial2/2",

"Serial2/3",

"Serial3/0",

"Serial3/1",

"Serial3/2",

"Serial3/3",

"Loopback0"

],

"fqdn": "R1.mikis.lab",

"os\_version": "Linux Software (I86BI\_LINUX-ADVENTERPRISEK9-M), Version 15.4(3)M9, DEVELOPMENT TEST SOFTWARE",

"serial\_number": "2048002",

"hostname": "R1",

"model": "Unknown",

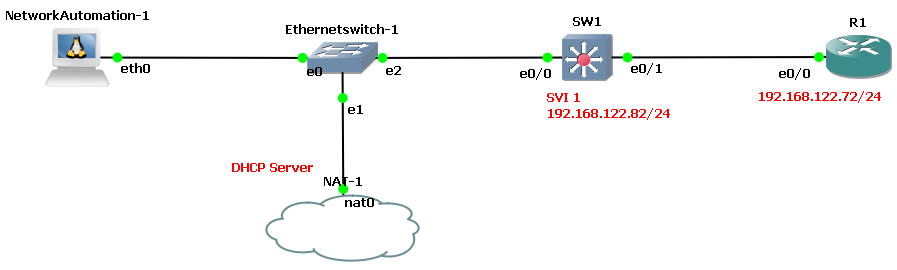
"uptime": 10080

}

root@NetworkAutomation-1:~#

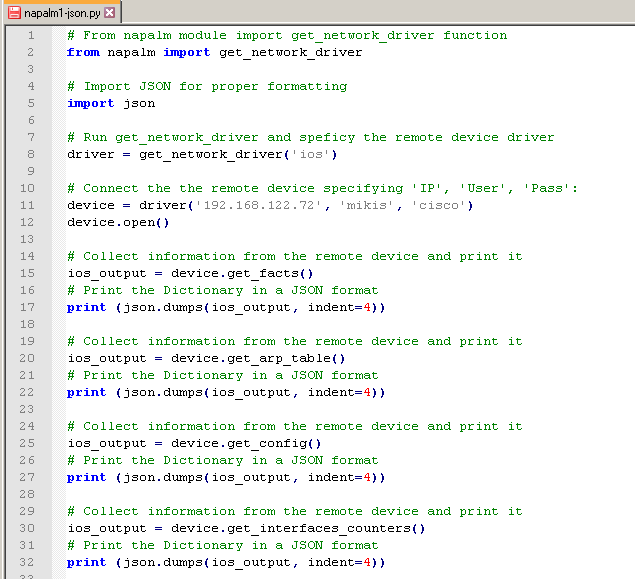
### Section 3 (napalm) - Script 4 - Get more remote device info using NAPALM (improved output using JSON)

**Topology**



The following script:

* Connects to a remote IOS device using SSH (Netmiko)
* Gets generic info from the remote device
* It gets the remote output as a Python Dictionary and it prints it using JSON formatting



root@NetworkAutomation-1:~# **cat > napalm1-json2.py**

*# From napalm module import get\_network\_driver function*

*from napalm import get\_network\_driver*

*# Import JSON for proper formatting*

*import json*

*# Run get\_network\_driver and speficy the remote device driver*

*driver = get\_network\_driver('ios')*

*# Connect the the remote device specifying 'IP', 'User', 'Pass':*

*device = driver('192.168.122.72', 'mikis', 'cisco')*

*device.open()*

*# Collect information from the remote device and print it*

*ios\_output = device.get\_facts()*

*# Print the Dictionary in a JSON format*

*print (json.dumps(ios\_output, indent=4))*

*# Collect information from the remote device and print it*

*ios\_output = device.get\_arp\_table()*

*# Print the Dictionary in a JSON format*

*print (json.dumps(ios\_output, indent=4))*

*# Collect information from the remote device and print it*

*ios\_output = device.get\_config()*

*# Print the Dictionary in a JSON format*

*print (json.dumps(ios\_output, indent=4))*

*# Collect information from the remote device and print it*

*ios\_output = device.get\_interfaces\_counters()*

*# Print the Dictionary in a JSON format*

*print (json.dumps(ios\_output, indent=4))*

**Running the script**

root@NetworkAutomation-1:~# **python3 napalm1-json2.py**

{

"os\_version": "Linux Software (I86BI\_LINUX-ADVENTERPRISEK9-M), Version 15.4(3)M9, DEVELOPMENT TEST SOFTWARE",

"serial\_number": "2048002",

"vendor": "Cisco",

"uptime": 10380,

"fqdn": "R1.mikis.lab",

"interface\_list": [

"Ethernet0/0",

"Ethernet0/1",

"Ethernet0/2",

"Ethernet0/3",

"Ethernet1/0",

"Ethernet1/1",

"Ethernet1/2",

"Ethernet1/3",

"Serial2/0",

"Serial2/1",

"Serial2/2",

"Serial2/3",

"Serial3/0",

"Serial3/1",

"Serial3/2",

"Serial3/3",

"Loopback0"

],

"model": "Unknown",

"hostname": "R1"

}

[

{

"age": 0.0,

"interface": "Ethernet0/0",

"ip": "192.168.122.72",

"mac": "AA:BB:CC:00:02:00"

},

{

"age": 158.0,

"interface": "Ethernet0/0",

"ip": "192.168.122.120",

"mac": "92:39:C3:42:6A:4C"

}

]

{

"startup": "Using 1985 out of 32768 bytes\n!\n! Last configuration change at 08:45:37 UTC Tue Sep 11 2018\n!\nversion 15.4\nservice timestamps debug datetime msec\nservice timestamps log datetime msec\nno service password-encryption\n!\nhostname R1\n!\nboot-start-marker\nboot-end-marker\n!\n!\n!\nno aaa new-model\n!\n!\n!\nmmi polling-interval 60\nno mmi auto-configure\nno mmi pvc\nmmi snmp-timeout 180\n!\n!\n!\n!\n!\nno ip icmp rate-limit unreachable\n!\n!\n!\n!\n!\n!\n!\n!\n\n\n!\n!\n!\n!\nno ip domain lookup\nip domain name mikis.lab\nip cef\nno ipv6 cef\n!\nmultilink bundle-name authenticated\n!\n!\n!\n!\n!\n!\n!\ncts logging verbose\n!\n!\nusername cisco privilege 15 password 0 cisco\nusername mikis privilege 15 password 0 cisco\n!\nredundancy\n!\n!\nip tcp synwait-time 5\n! \n!\n!\n!\n!\n!\n!\n!\n!\n!\n!\n!\ninterface Ethernet0/0\n ip address 192.168.122.72 255.255.255.0\n!\ninterface Ethernet0/1\n no ip address\n shutdown\n!\ninterface Ethernet0/2\n no ip address\n shutdown\n!\ninterface Ethernet0/3\n no ip address\n shutdown\n!\ninterface Ethernet1/0\n no ip address\n shutdown\n!\ninterface Ethernet1/1\n no ip address\n shutdown\n!\ninterface Ethernet1/2\n no ip address\n shutdown\n!\ninterface Ethernet1/3\n no ip address\n shutdown\n!\ninterface Serial2/0\n no ip address\n shutdown\n serial restart-delay 0\n!\ninterface Serial2/1\n no ip address\n shutdown\n serial restart-delay 0\n!\ninterface Serial2/2\n no ip address\n shutdown\n serial restart-delay 0\n!\ninterface Serial2/3\n no ip address\n shutdown\n serial restart-delay 0\n!\ninterface Serial3/0\n no ip address\n shutdown\n serial restart-delay 0\n!\ninterface Serial3/1\n no ip address\n shutdown\n serial restart-delay 0\n!\ninterface Serial3/2\n no ip address\n shutdown\n serial restart-delay 0\n!\ninterface Serial3/3\n no ip address\n shutdown\n serial restart-delay 0\n!\nip forward-protocol nd\n!\n!\nno ip http server\nno ip http secure-server\n!\n!\n!\n!\ncontrol-plane\n!\n!\n!\n!\n!\n!\n!\n!\nline con 0\n exec-timeout 0 0\n privilege level 15\n logging synchronous\nline aux 0\n exec-timeout 0 0\n privilege level 15\n logging synchronous\nline vty 0 4\n login local\n transport input ssh\n!\n!\nend",

"running": "Building configuration...\n\nCurrent configuration : 2052 bytes\n!\n! Last configuration change at 08:54:25 UTC Tue Sep 11 2018 by cisco\n!\nversion 15.4\nservice timestamps debug datetime msec\nservice timestamps log datetime msec\nno service password-encryption\n!\nhostname R1\n!\nboot-start-marker\nboot-end-marker\n!\n!\n!\nno aaa new-model\n!\n!\n!\nmmi polling-interval 60\nno mmi auto-configure\nno mmi pvc\nmmi snmp-timeout 180\n!\n!\n!\n!\n!\nno ip icmp rate-limit unreachable\n!\n!\n!\n!\n!\n!\n!\n!\n\n\n!\n!\n!\n!\nno ip domain lookup\nip domain name mikis.lab\nip cef\nno ipv6 cef\n!\nmultilink bundle-name authenticated\n!\n!\n!\n!\n!\n!\n!\ncts logging verbose\n!\n!\nusername cisco privilege 15 password 0 cisco\nusername mikis privilege 15 password 0 cisco\n!\nredundancy\n!\n!\nip tcp synwait-time 5\n! \n!\n!\n!\n!\n!\n!\n!\n!\n!\n!\n!\ninterface Loopback0\n ip address 1.1.1.1 255.255.255.255\n!\ninterface Ethernet0/0\n ip address 192.168.122.72 255.255.255.0\n!\ninterface Ethernet0/1\n no ip address\n shutdown\n!\ninterface Ethernet0/2\n no ip address\n shutdown\n!\ninterface Ethernet0/3\n no ip address\n shutdown\n!\ninterface Ethernet1/0\n no ip address\n shutdown\n!\ninterface Ethernet1/1\n no ip address\n shutdown\n!\ninterface Ethernet1/2\n no ip address\n shutdown\n!\ninterface Ethernet1/3\n no ip address\n shutdown\n!\ninterface Serial2/0\n no ip address\n shutdown\n serial restart-delay 0\n!\ninterface Serial2/1\n no ip address\n shutdown\n serial restart-delay 0\n!\ninterface Serial2/2\n no ip address\n shutdown\n serial restart-delay 0\n!\ninterface Serial2/3\n no ip address\n shutdown\n serial restart-delay 0\n!\ninterface Serial3/0\n no ip address\n shutdown\n serial restart-delay 0\n!\ninterface Serial3/1\n no ip address\n shutdown\n serial restart-delay 0\n!\ninterface Serial3/2\n no ip address\n shutdown\n serial restart-delay 0\n!\ninterface Serial3/3\n no ip address\n shutdown\n serial restart-delay 0\n!\nip forward-protocol nd\n!\n!\nno ip http server\nno ip http secure-server\n!\n!\n!\n!\ncontrol-plane\n!\n!\n!\n!\n!\n!\n!\n!\nline con 0\n exec-timeout 0 0\n privilege level 15\n logging synchronous\nline aux 0\n exec-timeout 0 0\n privilege level 15\n logging synchronous\nline vty 0 4\n login local\n transport input ssh\n!\n!\nend",

"candidate": ""

}

{

"Serial2/1": {

"tx\_unicast\_packets": 0,

"tx\_errors": 0,

"rx\_broadcast\_packets": 0,

"rx\_multicast\_packets": 0,

"rx\_discards": 0,

"tx\_discards": 0,

"rx\_errors": 0,

"rx\_unicast\_packets": 0,

"tx\_multicast\_packets": -1,

"tx\_octets": 0,

"tx\_broadcast\_packets": -1,

"rx\_octets": 0

},

...

"Serial3/3": {

"tx\_unicast\_packets": 0,

"tx\_errors": 0,

"rx\_broadcast\_packets": 0,

"rx\_multicast\_packets": 0,

"rx\_discards": 0,

"tx\_discards": 0,

"rx\_errors": 0,

"rx\_unicast\_packets": 0,

"tx\_multicast\_packets": -1,

"tx\_octets": 0,

"tx\_broadcast\_packets": -1,

"rx\_octets": 0

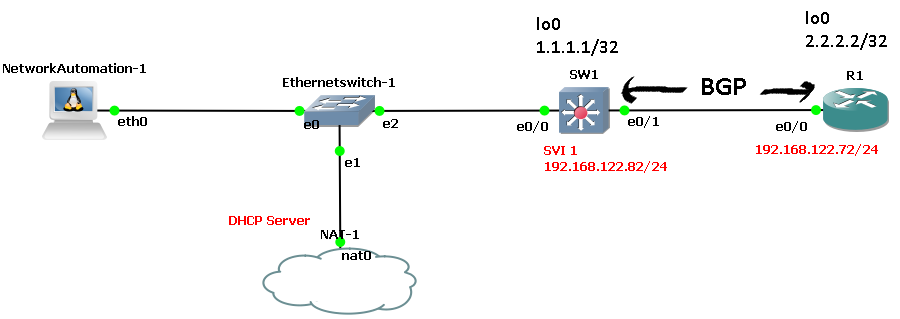
}

}

root@NetworkAutomation-1:~#

### Section 3 (napalm) - Script 5 - Get BGP neighbor info using NAPALM and print the output using JSON

**Topology**



SW1#**show ip bgp summary | b Nei**

Neighbor V AS MsgRcvd MsgSent TblVer InQ OutQ Up/Down State/PfxRcd

192.168.122.72 4 65000 18 18 3 0 0 00:12:08 1

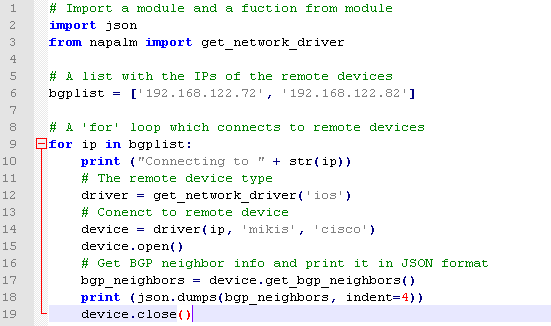
R1#**show ip bgp summary | b Nei**

Neighbor V AS MsgRcvd MsgSent TblVer InQ OutQ Up/Down State/PfxRcd

192.168.122.82 4 65000 17 17 5 0 0 00:11:43 1

The following script:

* Connects to multiple remote IOS device using SSH (Netmiko)
* Gets BGP info from the remote devices
* It gets the remote outputs as a Python Dictionaries and prints it using JSON formatting



root@NetworkAutomation-1:~# **cat > napalm3.py**

*# Import a module and a fuction from module*

*import json*

*from napalm import get\_network\_driver*

*# A list with the IPs of the remote devices*

*bgplist = ['192.168.122.72', '192.168.122.82']*

*# A 'for' loop which connects to remote devices*

*for ip in bgplist:*

*print ("Connecting to " + str(ip))*

*# The remote device type*

*driver = get\_network\_driver('ios')*

*# Conenct to remote device*

*device = driver(ip, 'mikis', 'cisco')*

*device.open()*

*# Get BGP neighbor info and print it in JSON format*

*bgp\_neighbors = device.get\_bgp\_neighbors()*

*print (json.dumps(bgp\_neighbors, indent=4))*

*device.close()*

**Running the script**

root@NetworkAutomation-1:~# **python3 napalm3.py**

Connecting to 192.168.122.72

{

"global": {

"router\_id": "2.2.2.2",

"peers": {

"192.168.122.82": {

"description": "",

"is\_up": true,

"local\_as": 65000,

"is\_enabled": true,

"remote\_id": "1.1.1.1",

"address\_family": {

"ipv4": {

"received\_prefixes": 1,

"sent\_prefixes": 1,

"accepted\_prefixes": 1

}

},

"remote\_as": 65000,

"uptime": 15

}

}

}

}

Connecting to 192.168.122.82

{

"global": {

"router\_id": "1.1.1.1",

"peers": {

"192.168.122.72": {

"description": "",

"is\_up": true,

"local\_as": 65000,

"is\_enabled": true,

"remote\_id": "2.2.2.2",

"address\_family": {

"ipv4": {

"received\_prefixes": 1,

"sent\_prefixes": 1,

"accepted\_prefixes": 1

}

},

"remote\_as": 65000,

"uptime": 24

}

}

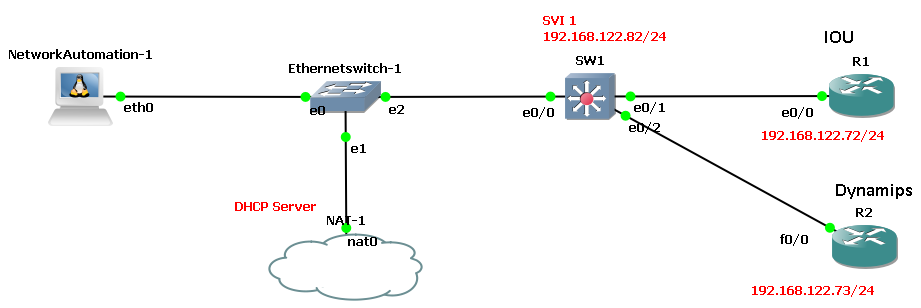
}

}

root@NetworkAutomation-1:~#

### Section 3 (napalm) - Script 6 - Making config changes to remote devices using NAPALM

**Topology**



The following script:

* Connects to a device using SSH (Netmiko) and sends configuration using Napalm

The config we want to apply (merge):

root@NetworkAutomation-1:~# **cat R1.cfg**

router rip

no auto-summary

version 2

network 192.168.122.0

!

ip name-server 8.8.8.8

!

root@NetworkAutomation-1:~#

**Note**

I tried to run a config script to an IOS device and got:

root@NetworkAutomation-1:~# **python3 napalm\_config1.py**

Accessing 192.168.122.72

Traceback (most recent call last):

File "napalm\_config1.py", line 12, in <module>

device.load\_merge\_candidate(filename='R1.cfg')

File "/usr/local/lib/python3.5/dist-packages/napalm/ios/ios.py", line 268, in load\_merge\_candidate

file\_system=self.dest\_file\_system)

File "/usr/local/lib/python3.5/dist-packages/napalm/ios/ios.py", line 238, in \_load\_candidate\_wrapper

file\_system=file\_system)

File "/usr/local/lib/python3.5/dist-packages/napalm/ios/ios.py", line 513, in \_scp\_file

file\_system=file\_system, TransferClass=FileTransfer)

File "/usr/local/lib/python3.5/dist-packages/napalm/ios/ios.py", line 548, in \_xfer\_file

if not transfer.verify\_space\_available():

File "/usr/local/lib/python3.5/dist-packages/netmiko/scp\_handler.py", line 146, in verify\_space\_available

space\_avail = self.remote\_space\_available(search\_pattern=search\_pattern)

File "/usr/local/lib/python3.5/dist-packages/netmiko/scp\_handler.py", line 105, **in remote\_space\_available**

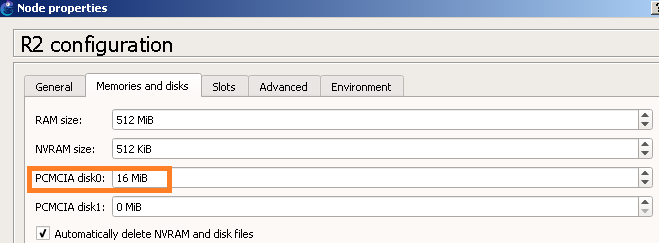
return int(match.group(1))

AttributeError: 'NoneType' object has no attribute 'group'

From:

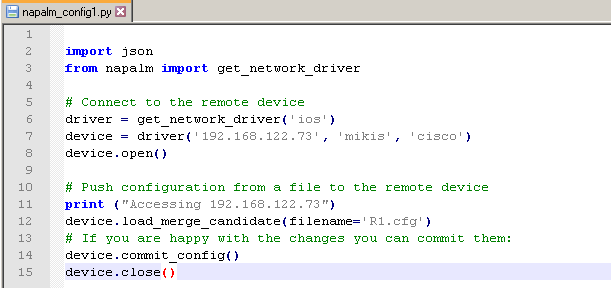
https://github.com/napalm-automation/napalm/issues/612

**napalm-ios requires a file system to transfer a file into for any of the configuration operations.**



and

R2(config)#**ip scp server enable**



root@NetworkAutomation-1:~# **cat > napalm\_config1.py**

*import json*

*from napalm import get\_network\_driver*

*# Connect to the remote device*

*driver = get\_network\_driver('ios')*

*device = driver('192.168.122.73', 'mikis', 'cisco')*

*device.open()*

*# Push configuration from a file to the remote device*

*print ("Accessing 192.168.122.73")*

*device.load\_merge\_candidate(filename='R1.cfg')*

*# If you are happy with the changes you can commit them:*

*device.commit\_config()*

*device.close()*

**Running the script**

root@NetworkAutomation-1:~# **python3 napalm\_config1.py**

Accessing 192.168.122.73

root@NetworkAutomation-1:~#

**Verification**

R2#**sh run | s router**

router rip

version 2

network 192.168.122.0

no auto-summary

R2#

R2#

R2#**sh run | s name**

hostname R2

ip domain name mikis.lab

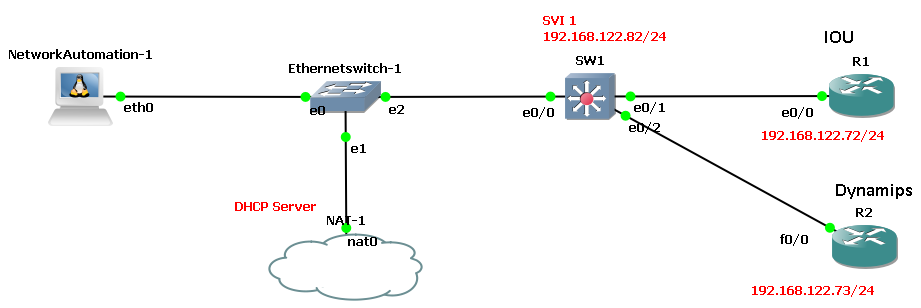
ip name-server 8.8.8.8

multilink bundle-name authenticated

username mikis privilege 15 password 0 cisco

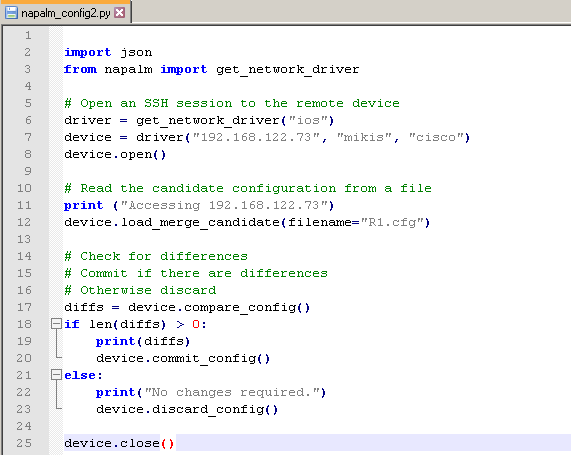
### Section 3 (napalm) - Script 6 - Check for diff and make config changes to remote devices using NAPALM

**Topology**



The following script:

* Connects to a device using SSH (Netmiko) and sends configuration using Napalm
* If the provided config is different than the existing it commits it
* If the provided config is the same as the existing it discards the changes



root@NetworkAutomation-1:~# **cat > napalm\_config2.py**

*import json*

*from napalm import get\_network\_driver*

*# Open an SSH session to the remote device*

*driver = get\_network\_driver("ios")*

*device = driver("192.168.122.73", "mikis", "cisco")*

*device.open()*

*# Read the candidate configuration from a file*

*print ("Accessing 192.168.122.73")*

*device.load\_merge\_candidate(filename="R1.cfg")*

*# Check for differences*

*# Commit if there are differences*

*# Otherwise discard*

*diffs = device.compare\_config()*

*if len(diffs) > 0:*

*print(diffs)*

*device.commit\_config()*

*else:*

*print("No changes required.")*

*device.discard\_config()*

*device.close()*

root@NetworkAutomation-1:~#

**Running the script**

If there are differences the script shows:

root@NetworkAutomation-1:~# **python3 napalm\_config2.py**

Accessing 192.168.122.73

+router rip

- no auto

root@NetworkAutomation-1:~#

If there are no differences the script shows:

root@NetworkAutomation-1:~# **python3 napalm\_config2.py**

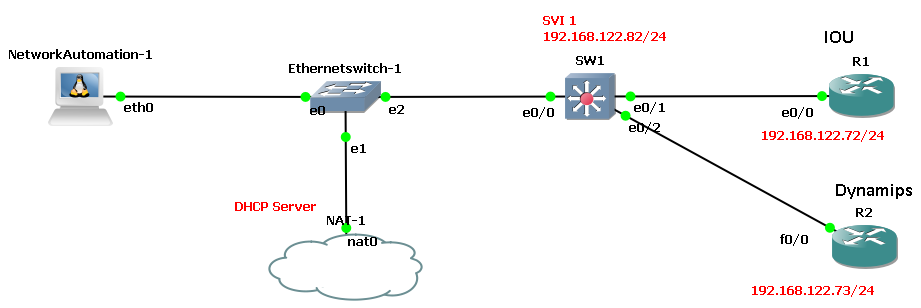
Accessing 192.168.122.73

No changes required.

root@NetworkAutomation-1:~#

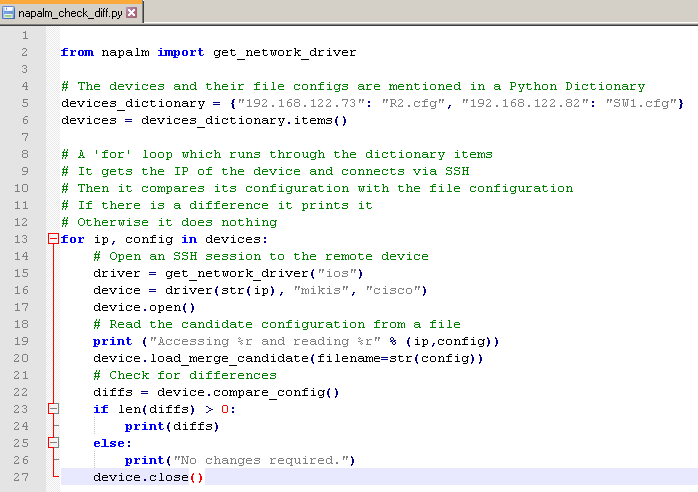
### Section 3 (napalm) - Script 7 - Check for config diff multiple devices using NAPALM

**Topology**



The following script:

* Connects to remote devices using SSH (Netmiko)
* Compares the remote device config with their file configs
* If the file config has config that does not exist in the remote device it prints the differences



root@NetworkAutomation-1:~# **cat > napalm\_check\_diff.py**

*from napalm import get\_network\_driver*

*# The devices and their file configs are mentioned in a Python Dictionary*

*devices\_dictionary = {"192.168.122.73": "R2.cfg", "192.168.122.82": "SW1.cfg"}*

*devices = devices\_dictionary.items()*

*# A 'for' loop which runs through the dictionary items*

*# It gets the IP of the device and connects via SSH*

*# Then it compares its configuration with the file configuration*

*# If there is a difference it prints it*

*# Otherwise it does nothing*

*for ip, config in devices:*

*# Open an SSH session to the remote device*

*driver = get\_network\_driver("ios")*

*device = driver(str(ip), "mikis", "cisco")*

*device.open()*

*# Read the candidate configuration from a file*

*print ("Accessing %r and reading %r" % (ip,config))*

*device.load\_merge\_candidate(filename=str(config))*

*# Check for differences*

*diffs = device.compare\_config()*

*if len(diffs) > 0:*

*print(diffs)*

*else:*

*print("No changes required.")*

*device.close()*

**Running the script**

root@NetworkAutomation-1:~# **python3 napalm\_check\_diff.py**

Accessing '192.168.122.73' and reading 'R2.cfg'

+interface Loopback2

+ ip address 2.2.2.2 255.255.255.255

Accessing '192.168.122.82' and reading 'SW1.cfg'

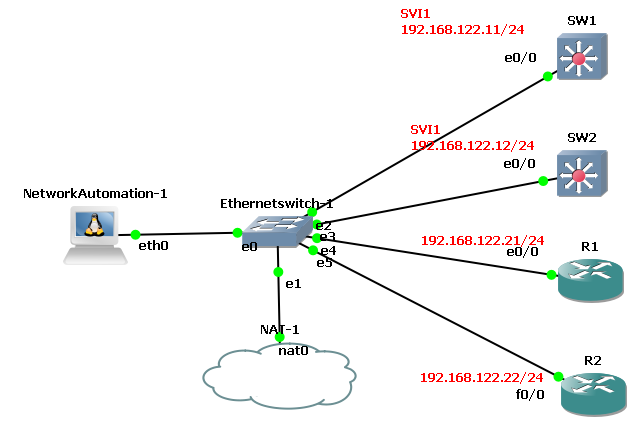
No changes required.

root@NetworkAutomation-1:~#

## Section 8 - Iterations with Netmiko

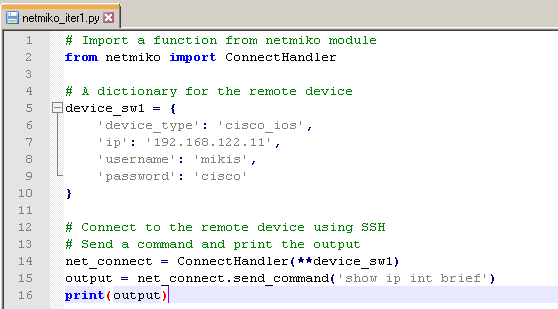
### Section 8 (iterations with Netmiko) - Iteration 1 - Send a 'show' command

**Topology**



The following script:

* Connects to a single remote device using SSH (Netmiko)
* Runs a 'show' command and prints the output



root@NetworkAutomation-1:~# **cat > netmiko\_iter1.py**

*# Import a function from netmiko module*

*from netmiko import ConnectHandler*

*# A dictionary for the remote device*

*device\_sw1 = {*

*'device\_type': 'cisco\_ios',*

*'ip': '192.168.122.11',*

*'username': 'mikis',*

*'password': 'cisco'*

*}*

*# Connect to the remote device using SSH*

*# Send a command and print the output*

*net\_connect = ConnectHandler(\*\*device\_sw1)*

*output = net\_connect.send\_command('show ip int brief')*

*print(output)*

**Running the script**

root@NetworkAutomation-1:~# **python3 netmiko\_iter1.py**

Interface IP-Address OK? Method Status Protocol

Ethernet0/0 unassigned YES unset up up

Ethernet0/1 unassigned YES unset up up

Ethernet0/2 unassigned YES unset up up

Ethernet0/3 unassigned YES unset up up

Ethernet1/0 unassigned YES unset up up

Ethernet1/1 unassigned YES unset up up

Ethernet1/2 unassigned YES unset up up

Ethernet1/3 unassigned YES unset up up

Ethernet2/0 unassigned YES unset up up

Ethernet2/1 unassigned YES unset up up

Ethernet2/2 unassigned YES unset up up

Ethernet2/3 unassigned YES unset up up

Ethernet3/0 unassigned YES unset up up

Ethernet3/1 unassigned YES unset up up

Ethernet3/2 unassigned YES unset up up

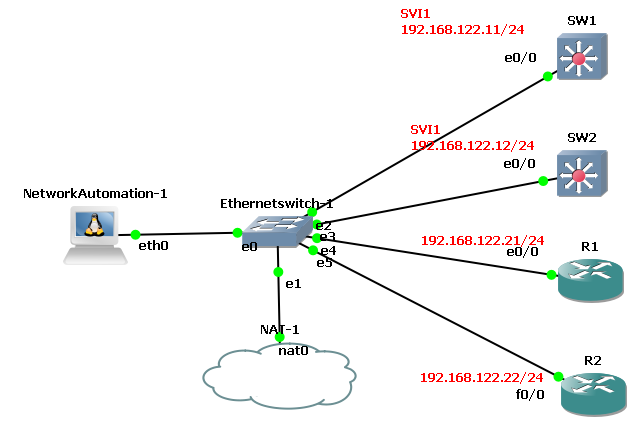
Ethernet3/3 unassigned YES unset up up

Vlan1 192.168.122.11 YES NVRAM up up

root@NetworkAutomation-1:~#

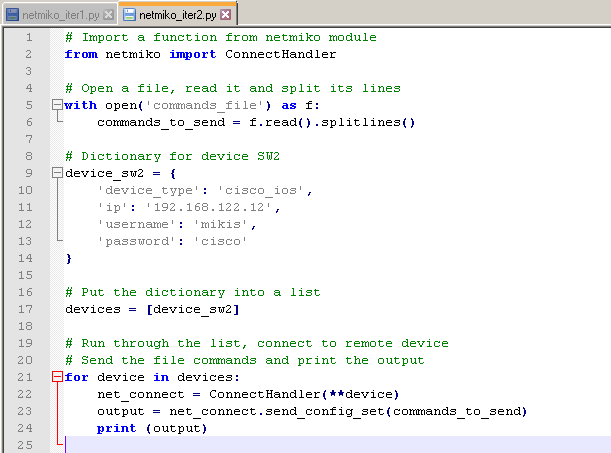
### Section 8 (iterations with Netmiko) - Iteration 2 - Send commands from a file

**Topology**



The following script:

* Connects to a single remote device using SSH (Netmiko)
* Reads commands from a file and sends them to the remote device



root@NetworkAutomation-1:~# **cat > netmiko\_iter2.py**

*# Import a function from netmiko module*

*from netmiko import ConnectHandler*

*# Open a file, read it and split its lines*

*with open('commands\_file') as f:*

*commands\_to\_send = f.read().splitlines()*

*# Dictionary for device SW2*

*device\_sw2 = {*

*'device\_type': 'cisco\_ios',*

*'ip': '192.168.122.12',*

*'username': 'mikis',*

*'password': 'cisco'*

*}*

*devices = [device\_sw2]*

*for device in devices:*

*net\_connect = ConnectHandler(\*\*device)*

*output = net\_connect.send\_config\_set(commands\_to\_send)*

*print (output)*

**Running the script**

root@NetworkAutomation-1:~# **python3 netmiko\_iter2.py**

config term

Enter configuration commands, one per line. End with CNTL/Z.

SW2(config)#router eigrp 1

SW2(config-router)# network 0.0.0.0 255.255.255.255

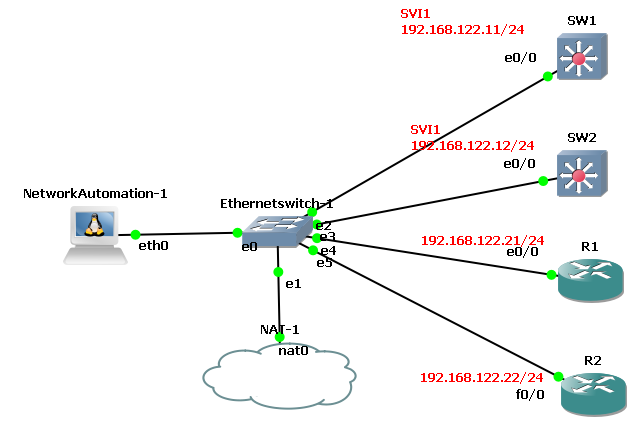
SW2(config-router)#end

SW2#

root@NetworkAutomation-1:~#

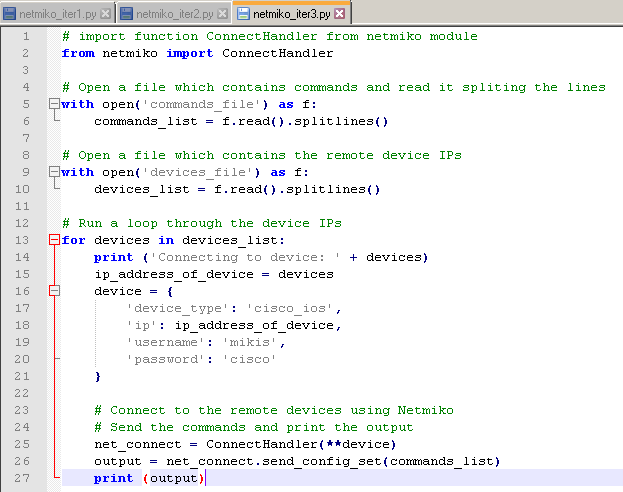
### Section 8 (iterations with Netmiko) - Iteration 3 - Send commands from a file to multiple devices

**Topology**



The following script:

* Reads the IPs of devices from a file and connects using SSH (Netmiko)
* Reads commands from a file and sends them to the remote devices



root@NetworkAutomation-1:~# **cat > netmiko\_iter3.py**

*# import function ConnectHandler from netmiko module*

*from netmiko import ConnectHandler*

*# Open a file which contains commands and read it spliting the lines*

*with open('commands\_file') as f:*

*commands\_list = f.read().splitlines()*

*# Open a file which contains the remote device IPs*

*with open('devices\_file') as f:*

*devices\_list = f.read().splitlines()*

*# Run a loop through the device IPs*

*for devices in devices\_list:*

*print ('Connecting to device: ' + devices)*

*ip\_address\_of\_device = devices*

*device = {*

*'device\_type': 'cisco\_ios',*

*'ip': ip\_address\_of\_device,*

*'username': 'mikis',*

*'password': 'cisco'*

*}*

*# Connect to the remote devices using Netmiko*

*# Send the commands and print the output*

*net\_connect = ConnectHandler(\*\*device)*

*output = net\_connect.send\_config\_set(commands\_list)*

*print (output)*

**Running the script**

root@NetworkAutomation-1:~# **python3 netmiko\_iter3.py**

Connecting to device: 192.168.122.21

config term

Enter configuration commands, one per line. End with CNTL/Z.

R1(config)#router ospf 1

R1(config-router)# network 0.0.0.0 255.255.255.255 area 0

R1(config-router)#end

R1#

Connecting to device: 192.168.122.22

config term

Enter configuration commands, one per line. End with CNTL/Z.

R2(config)#router ospf 1

R2(config-router)# network 0.0.0.0 255.255.255.255 area 0

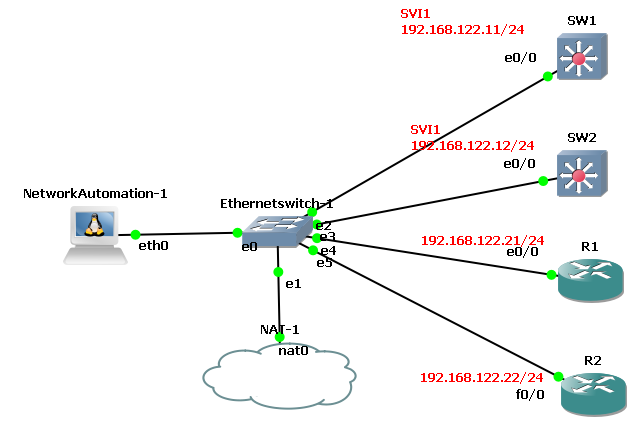
R2(config-router)#end

R2#

root@NetworkAutomation-1:~#

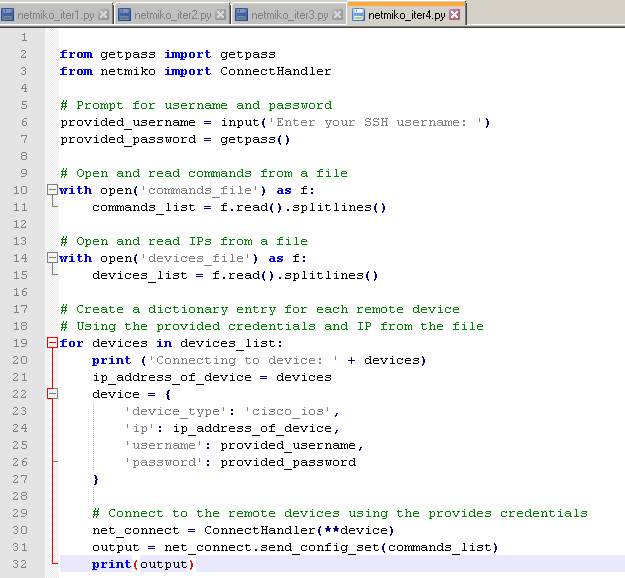
### Section 8 (iterations with Netmiko) - Iteration 4 - Send commands from a file to multiple devices with user prompt

**Topology**



The following script:

* Reads the IPs of devices from a file and connects using SSH (Netmiko)
* Reads commands from a file and sends them to the remote devices
* The user is prompted for SSH username and password



root@NetworkAutomation-1:~# **cat > netmiko\_iter4.py**

*from getpass import getpass*

*from netmiko import ConnectHandler*

*# Prompt for username and password*

*provided\_username = input('Enter your SSH username: ')*

*provided\_password = getpass()*

*# Open and read commands from a file*

*with open('commands\_file') as f:*

*commands\_list = f.read().splitlines()*

*# Open and read IPs from a file*

*with open('devices\_file') as f:*

*devices\_list = f.read().splitlines()*

*# Create a dictionary entry for each remote device*

*# Using the provided credentials and IP from the file*

*for devices in devices\_list:*

*print ('Connecting to device: ' + devices)*

*ip\_address\_of\_device = devices*

*device = {*

*'device\_type': 'cisco\_ios',*

*'ip': ip\_address\_of\_device,*

*'username': provided\_username,*

*'password': provided\_password*

*}*

*# Connect to the remote devices using the provides credentials*

*net\_connect = ConnectHandler(\*\*device)*

*output = net\_connect.send\_config\_set(commands\_list)*

*print(output)*

**Running the script**

root@NetworkAutomation-1:~# **python3 netmiko\_iter4.py**

Enter your SSH username: mikis

Password:

Connecting to device: 192.168.122.21

config term

Enter configuration commands, one per line. End with CNTL/Z.

R1(config)#router ospf 1

R1(config-router)# network 0.0.0.0 255.255.255.255 area 0

R1(config-router)#end

R1#

Connecting to device: 192.168.122.22

config term

Enter configuration commands, one per line. End with CNTL/Z.

R2(config)#router ospf 1

R2(config-router)# network 0.0.0.0 255.255.255.255 area 0

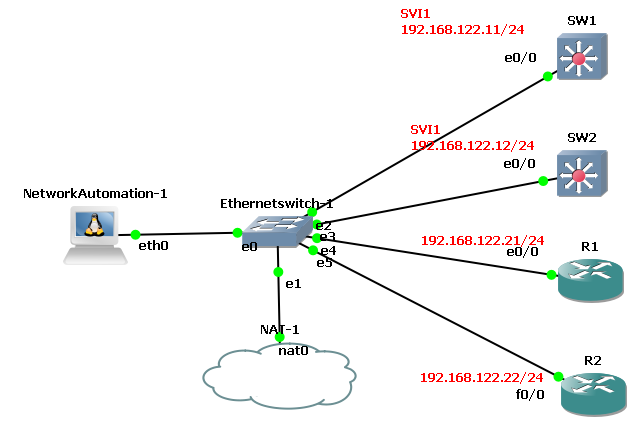
R2(config-router)#end

R2#

root@NetworkAutomation-1:~#

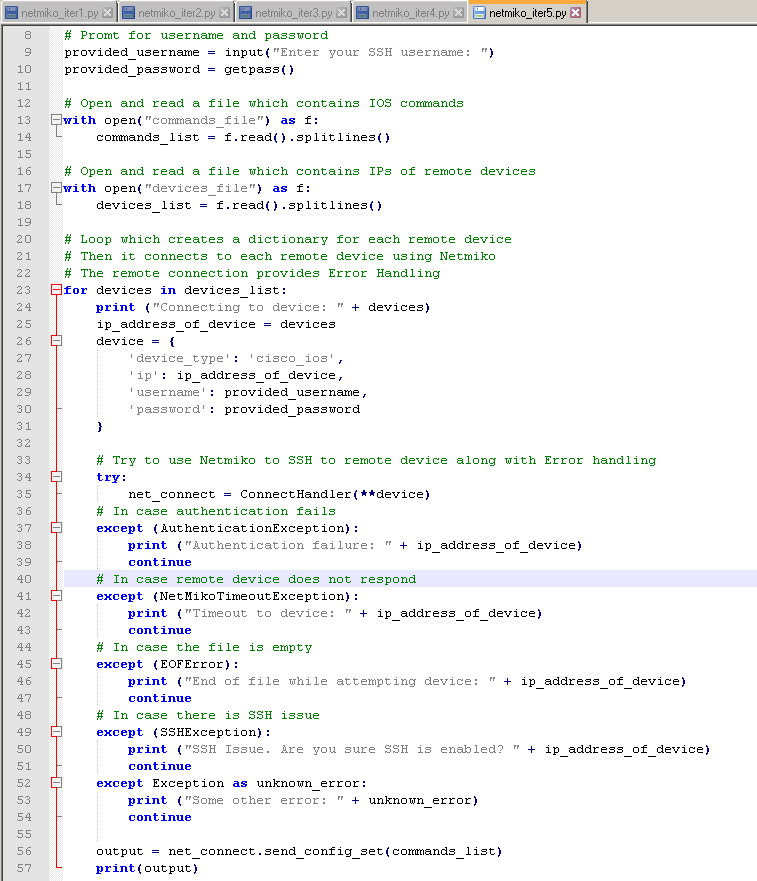
### Section 8 (iterations with Netmiko) - Iteration 5 - Send commands from a file to multiple devices with user prompt and error handling

**Topology**



The following script:

* Reads the IPs of devices from a file and connects using SSH (Netmiko)
* Reads commands from a file and sends them to the remote devices
* The user is prompted for SSH username and password and there is error handling



root@NetworkAutomation-1:~# **cat > netmiko\_iter5.py**

*from getpass import getpass*

*from netmiko import ConnectHandler*

*from netmiko.ssh\_exception import NetMikoTimeoutException*

*from netmiko.ssh\_exception import AuthenticationException*

*from paramiko.ssh\_exception import SSHException*

*# Promt for username and password*

*provided\_username = input("Enter your SSH username: ")*

*provided\_password = getpass()*

*# Open and read a file which contains IOS commands*

*with open("commands\_file") as f:*

*commands\_list = f.read().splitlines()*

*# Open and read a file which contains IPs of remote devices*

*with open("devices\_file") as f:*

*devices\_list = f.read().splitlines()*

*# Loop which creates a dictionary for each remote device*

*# Then it connects to each remote device using Netmiko*

*# The remote connection provides Error Handling*

*for devices in devices\_list:*

*print ("Connecting to device: " + devices)*

*ip\_address\_of\_device = devices*

*device = {*

*'device\_type': 'cisco\_ios',*

*'ip': ip\_address\_of\_device,*

*'username': provided\_username,*

*'password': provided\_password*

*}*

*# Try to use Netmiko to SSH to remote device along with Error handling*

*try:*

*net\_connect = ConnectHandler(\*\*device)*

*# In case authentication fails*

*except (AuthenticationException):*

*print ("Authentication failure: " + ip\_address\_of\_device)*

*continue*

*# In case remote device does not respond*

*except (NetMikoTimeoutException):*

*print ("Timeout to device: " + ip\_address\_of\_device)*

*continue*

*# In case the file is empty*

*except (EOFError):*

*print ("End of file while attempting device: " + ip\_address\_of\_device)*

*continue*

*# In case there is SSH issue*

*except (SSHException):*

*print ("SSH Issue. Are you sure SSH is enabled? " + ip\_address\_of\_device)*

*continue*

*except Exception as unknown\_error:*

*print ("Some other error: " + unknown\_error)*

*continue*

*output = net\_connect.send\_config\_set(commands\_list)*

*print(output)*

**Running the script**

**Attempt 1 - all good**

root@NetworkAutomation-1:~# **python3 netmiko\_iter5.py**

Enter your SSH username: mikis

Password:

Connecting to device: 192.168.122.21

config term

Enter configuration commands, one per line. End with CNTL/Z.

R1(config)#router ospf 1

R1(config-router)# network 0.0.0.0 255.255.255.255 area 0

R1(config-router)#end

R1#

Connecting to device: 192.168.122.22

config term

Enter configuration commands, one per line. End with CNTL/Z.

R2(config)#router ospf 1

R2(config-router)# network 0.0.0.0 255.255.255.255 area 0

R2(config-router)#end

R2#

root@NetworkAutomation-1:~#

**Attempt 2 - Wrong password**

root@NetworkAutomation-1:~# **python3 netmiko\_iter5.py**

Enter your SSH username: mikis

Password:

Connecting to device: 192.168.122.21

Authentication failure: 192.168.122.21

Connecting to device: 192.168.122.22

Authentication failure: 192.168.122.22

**Attempt 3 - Wrong IPs in the device file**

root@NetworkAutomation-1:~# **python3 netmiko\_iter5.py**

Enter your SSH username: mikis

Password:

Connecting to device: #192.168.122.21

Timeout to device: #192.168.122.21

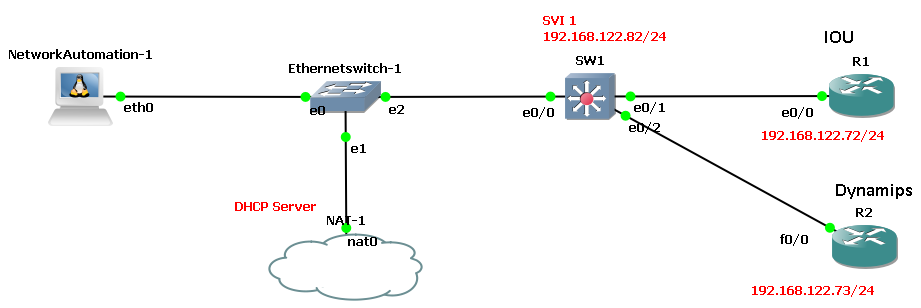
Connecting to device: #192.168.122.22

Timeout to device: #192.168.122.22

## Section 9 - Scaling Python scripts

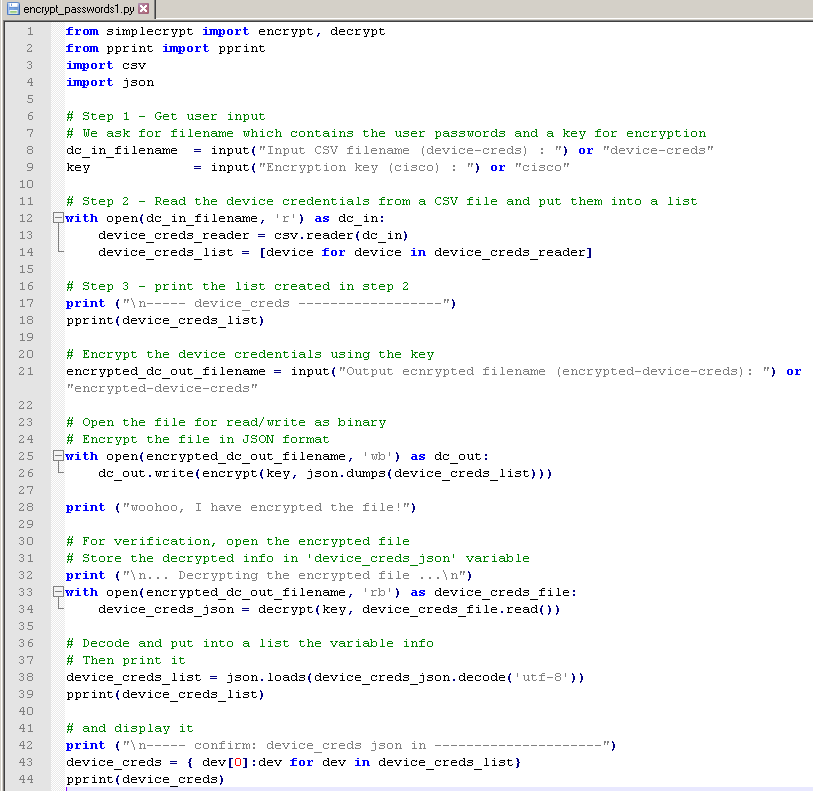
### Section 9 (scaling Netmiko) - Encrypt a passwords file using simple-crypt

**Topology**



The following script:

* Reads a CSV file that contains credentials and ecnrypts it



root@NetworkAutomation-1:~# **cat > encrypt\_passwords1.py**

*from simplecrypt import encrypt, decrypt*

*from pprint import pprint*

*import csv*

*import json*

*# Step 1 - Get user input*

*# We ask for filename which contains the user passwords and a key for encryption*

*dc\_in\_filename = input("Input CSV filename (device-creds) : ") or "device-creds"*

*key = input("Encryption key (cisco) : ") or "cisco"*

*# Step 2 - Read the device credentials from a CSV file and put them into a list*

*with open(dc\_in\_filename, 'r') as dc\_in:*

*device\_creds\_reader = csv.reader(dc\_in)*

*device\_creds\_list = [device for device in device\_creds\_reader]*

*# Step 3 - print the list created in step 2*

*print ("\n----- device\_creds ------------------")*

*pprint(device\_creds\_list)*

*# Encrypt the device credentials using the key*

*encrypted\_dc\_out\_filename = input("Output ecnrypted filename (encrypted-device-creds): ") or "encrypted-device-creds"*

*# Open the file for read/write as binary*

*# Encrypt the file in JSON format*

*with open(encrypted\_dc\_out\_filename, 'wb') as dc\_out:*

*dc\_out.write(encrypt(key, json.dumps(device\_creds\_list)))*

*print ("woohoo, I have encrypted the file!")*

*# For verification, open the encrypted file*

*# Store the decrypted info in 'device\_creds\_json' variable*

*print ("\n... Decrypting the encrypted file ...\n")*

*with open(encrypted\_dc\_out\_filename, 'rb') as device\_creds\_file:*

*device\_creds\_json = decrypt(key, device\_creds\_file.read())*

*# Decode and put into a list the variable info*

*# Then print it*

*device\_creds\_list = json.loads(device\_creds\_json.decode('utf-8'))*

*pprint(device\_creds\_list)*

*# and display it*

*print ("\n----- confirm: device\_creds json in ---------------------")*

*device\_creds = { dev[0]:dev for dev in device\_creds\_list}*

*pprint(device\_creds)*

**Running the script**

root@NetworkAutomation-1:~# **python3 encrypt\_passwords1.py**

Input CSV filename (device-creds) : **pass**

Encryption key (cisco) : **key2**

----- device\_creds ------------------

[['192.168.122.82', 'mikis', 'cisco'],

['192.168.122.72', 'mikis', 'cisco'],

['192.168.122.73', 'mikis', 'cisco']]

Output ecnrypted filename (encrypted-device-creds): encrypted3

woohoo, I have encrypted the file!

... Decrypting the encrypted file ...

[['192.168.122.82', 'mikis', 'cisco'],

['192.168.122.72', 'mikis', 'cisco'],

['192.168.122.73', 'mikis', 'cisco']]

----- confirm: device\_creds json in ---------------------

{'192.168.122.72': ['192.168.122.72', 'mikis', 'cisco'],

'192.168.122.73': ['192.168.122.73', 'mikis', 'cisco'],

'192.168.122.82': ['192.168.122.82', 'mikis', 'cisco']}

root@NetworkAutomation-1:~#

root@NetworkAutomation-1:~# **ls -alt | grep encr**

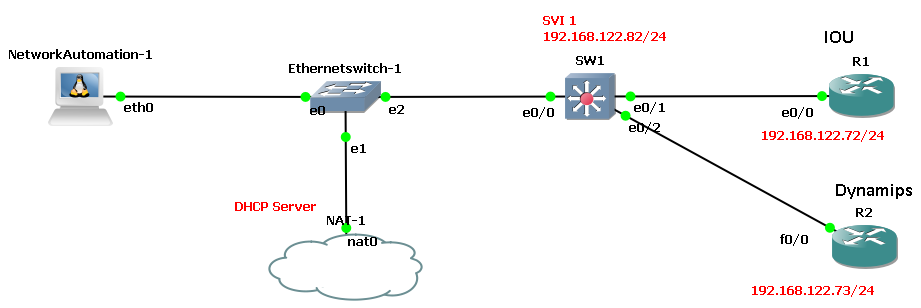
-rw-r--r-- 1 root root 182 Sep 13 12:05 encrypted3

-rw-r--r-- 1 root root 1691 Sep 13 12:05 encrypt\_passwords1.py

If I try to see the contents of the encrypted3 file **the VM crashes**

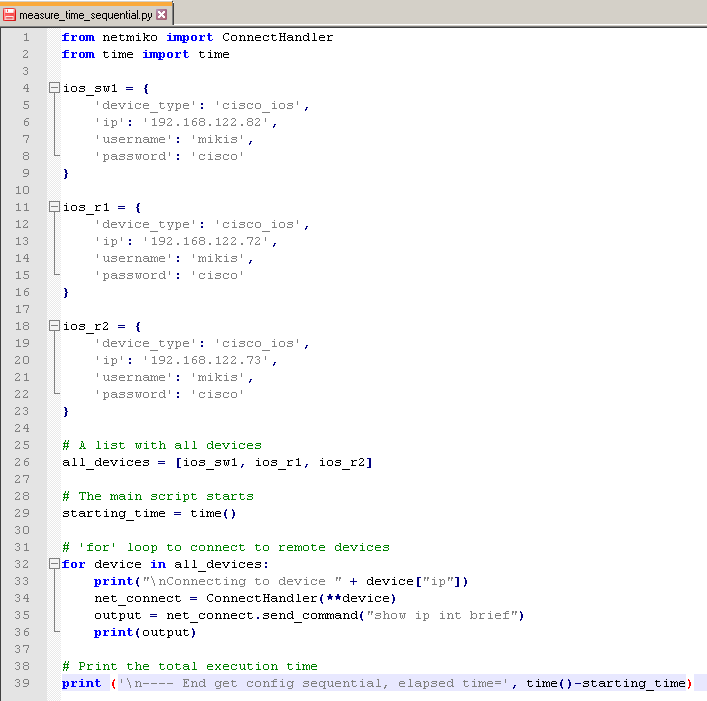
### Section 9 (scaling Netmiko) - Measuring the script execution time in Netmiko sequential deployment

**Topology**



The following script:

* Connects to remote devices sequentially using SSH (Netmiko)
* Measures the script total execution time



root@NetworkAutomation-1:~# **cat > measure\_time\_sequential.py**

*from netmiko import ConnectHandler*

*from time import time*

*ios\_sw1 = {*

*'device\_type': 'cisco\_ios',*

*'ip': '192.168.122.82',*

*'username': 'mikis',*

*'password': 'cisco'*

*}*

*ios\_r1 = {*

*'device\_type': 'cisco\_ios',*

*'ip': '192.168.122.72',*

*'username': 'mikis',*

*'password': 'cisco'*

*}*

*ios\_r2 = {*

*'device\_type': 'cisco\_ios',*

*'ip': '192.168.122.73',*

*'username': 'mikis',*

*'password': 'cisco'*

*}*

*# A list with all devices*

*all\_devices = [ios\_sw1, ios\_r1, ios\_r2]*

*# The main script starts*

*starting\_time = time()*

*# 'for' loop to connect to remote devices*

*for device in all\_devices:*

*print("\nConnecting to device " + device["ip"])*

*net\_connect = ConnectHandler(\*\*device)*

*output = net\_connect.send\_command("show ip int brief")*

*print(output)*

*# Print the total execution time*

*print ('\n---- End get config sequential, elapsed time=', time()-starting\_time)*

**Running the script**

root@NetworkAutomation-1:~# **python3 measure\_time\_sequential.py**

Connecting to device 192.168.122.82

Interface IP-Address OK? Method Status Protocol

Ethernet0/0 unassigned YES unset up up

Ethernet0/1 unassigned YES unset up up

Ethernet0/2 unassigned YES unset up up

Ethernet0/3 unassigned YES unset up up

Ethernet1/0 unassigned YES unset up up

Ethernet1/1 unassigned YES unset up up

Ethernet1/2 unassigned YES unset up up

Ethernet1/3 unassigned YES unset up up

Ethernet2/0 unassigned YES unset up up

Ethernet2/1 unassigned YES unset up up

Ethernet2/2 unassigned YES unset up up

Ethernet2/3 unassigned YES unset up up

...

Loopback1 1.1.1.1 YES NVRAM up up

Vlan1 192.168.122.82 YES NVRAM up up

Connecting to device 192.168.122.72

Interface IP-Address OK? Method Status Protocol

Ethernet0/0 192.168.122.72 YES NVRAM up up

Ethernet0/1 unassigned YES NVRAM administratively down down

Ethernet0/2 unassigned YES NVRAM administratively down down

Ethernet0/3 unassigned YES NVRAM administratively down down

Ethernet1/0 unassigned YES NVRAM administratively down down

Ethernet1/1 unassigned YES NVRAM administratively down down

Ethernet1/2 unassigned YES NVRAM administratively down down

Ethernet1/3 unassigned YES NVRAM administratively down down

Serial2/0 unassigned YES NVRAM administratively down down

Serial2/1 unassigned YES NVRAM administratively down down

Serial2/2 unassigned YES NVRAM administratively down down

Serial2/3 unassigned YES NVRAM administratively down down

...

Loopback0 2.2.2.2 YES NVRAM up up

Loopback22 22.22.22.22 YES NVRAM administratively down down

Connecting to device 192.168.122.73

Interface IP-Address OK? Method Status Protocol

FastEthernet0/0 192.168.122.73 YES NVRAM up up

FastEthernet1/0 unassigned YES NVRAM administratively down down

FastEthernet2/0 unassigned YES NVRAM administratively down down

Serial3/0 unassigned YES NVRAM administratively down down

Serial3/1 unassigned YES NVRAM administratively down down

Serial3/2 unassigned YES NVRAM administratively down down

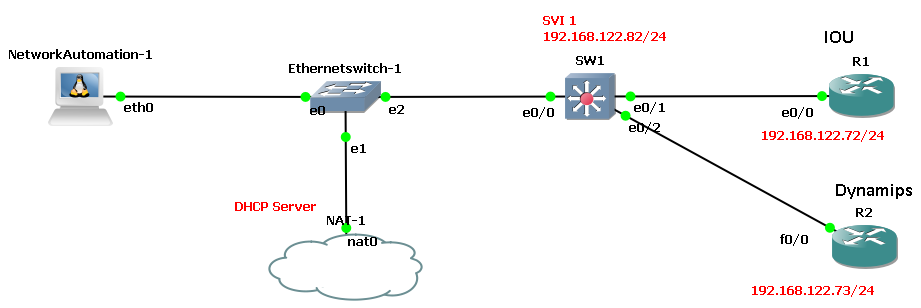
Serial3/3 unassigned YES NVRAM administratively down down

---- End get config sequential, elapsed time= 15.61446762084961

root@NetworkAutomation-1:~#

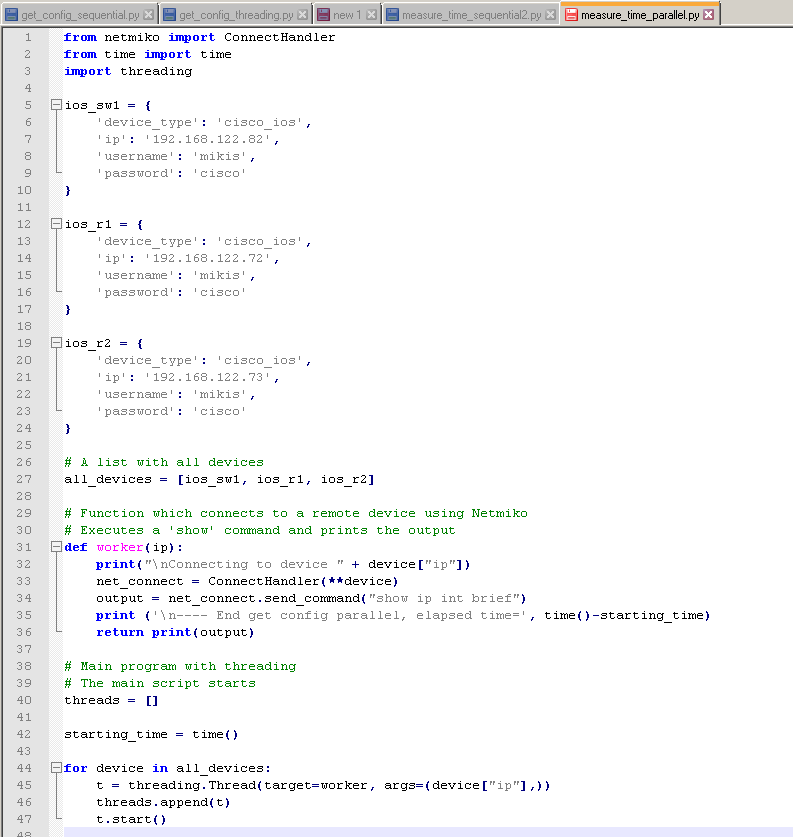
### Section 9 (scaling Netmiko) - Measuring the script execution time in Netmiko parallel deployment

**Topology**



The following script:

* Connects to remote devices in parallel using SSH (Netmiko)
* Measures the script total execution time



root@NetworkAutomation-1:~# **cat > measure\_time\_parallel.py**

*from netmiko import ConnectHandler*

*from time import time*

*import threading*

*ios\_sw1 = {*

*'device\_type': 'cisco\_ios',*

*'ip': '192.168.122.82',*

*'username': 'mikis',*

*'password': 'cisco'*

*}*

*ios\_r1 = {*

*'device\_type': 'cisco\_ios',*

*'ip': '192.168.122.72',*

*'username': 'mikis',*

*'password': 'cisco'*

*}*

*ios\_r2 = {*

*'device\_type': 'cisco\_ios',*

*'ip': '192.168.122.73',*

*'username': 'mikis',*

*'password': 'cisco'*

*}*

*# A list with all devices*

*all\_devices = [ios\_sw1, ios\_r1, ios\_r2]*

*# Function which connects to a remote device using Netmiko*

*# Executes a 'show' command and prints the output*

*def worker(ip):*

*print("\nConnecting to device " + device["ip"])*

*net\_connect = ConnectHandler(\*\*device)*

*output = net\_connect.send\_command("show ip int brief")*

*print ('\n---- End get config parallel, elapsed time=', time()-starting\_time)*

*return print(output)*

*# Main program with threading*

*# The main script starts*

*threads = []*

*starting\_time = time()*

*for device in all\_devices:*

*t = threading.Thread(target=worker, args=(device["ip"],))*

*threads.append(t)*

*t.start()*

**Running the script**

root@NetworkAutomation-1:~# **python3 measure\_time\_parallel.py**

Connecting to device 192.168.122.82

Connecting to device 192.168.122.72

Connecting to device 192.168.122.73

---- End get config parallel, elapsed time= 5.510881662368774

Interface IP-Address OK? Method Status Protocol

Ethernet0/0 192.168.122.72 YES NVRAM up up

Ethernet0/1 unassigned YES NVRAM administratively down down

Ethernet0/2 unassigned YES NVRAM administratively down down

Ethernet0/3 unassigned YES NVRAM administratively down down

Ethernet1/0 unassigned YES NVRAM administratively down down

Ethernet1/1 unassigned YES NVRAM administratively down down

Ethernet1/2 unassigned YES NVRAM administratively down down

Ethernet1/3 unassigned YES NVRAM administratively down down

Serial2/0 unassigned YES NVRAM administratively down down

Serial2/1 unassigned YES NVRAM administratively down down

Serial2/2 unassigned YES NVRAM administratively down down

Serial2/3 unassigned YES NVRAM administratively down down

Serial3/0 unassigned YES NVRAM administratively down down

Serial3/1 unassigned YES NVRAM administratively down down

Serial3/2 unassigned YES NVRAM administratively down down

Serial3/3 unassigned YES NVRAM administratively down down

Loopback0 2.2.2.2 YES NVRAM up up

Loopback22 22.22.22.22 YES NVRAM administratively down down

---- End get config parallel, elapsed time= 5.514587163925171

Interface IP-Address OK? Method Status Protocol

Ethernet0/0 192.168.122.72 YES NVRAM up up

Ethernet0/1 unassigned YES NVRAM administratively down down

Ethernet0/2 unassigned YES NVRAM administratively down down

Ethernet0/3 unassigned YES NVRAM administratively down down

Ethernet1/0 unassigned YES NVRAM administratively down down

Ethernet1/1 unassigned YES NVRAM administratively down down

Ethernet1/2 unassigned YES NVRAM administratively down down

Ethernet1/3 unassigned YES NVRAM administratively down down

Serial2/0 unassigned YES NVRAM administratively down down

Serial2/1 unassigned YES NVRAM administratively down down

Serial2/2 unassigned YES NVRAM administratively down down

Serial2/3 unassigned YES NVRAM administratively down down

Serial3/0 unassigned YES NVRAM administratively down down

Serial3/1 unassigned YES NVRAM administratively down down

Serial3/2 unassigned YES NVRAM administratively down down

Serial3/3 unassigned YES NVRAM administratively down down

Loopback0 2.2.2.2 YES NVRAM up up

Loopback22 22.22.22.22 YES NVRAM administratively down down

---- End get config parallel, elapsed time= 5.747814893722534

Interface IP-Address OK? Method Status Protocol

FastEthernet0/0 192.168.122.73 YES NVRAM up up

FastEthernet1/0 unassigned YES NVRAM administratively down down

FastEthernet2/0 unassigned YES NVRAM administratively down down

Serial3/0 unassigned YES NVRAM administratively down down

Serial3/1 unassigned YES NVRAM administratively down down

Serial3/2 unassigned YES NVRAM administratively down down

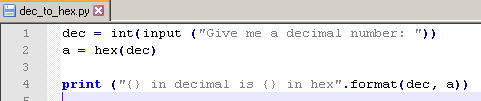
Serial3/3 unassigned YES NVRAM administratively down down

root@NetworkAutomation-1:~#

### Lecture 105 - Python3 - new way of working with strings and variables

The following script:

* Converts decimal to hex



root@NetworkAutomation-2:~# **cat dec\_to\_hex.py**

*dec = int(input ("Give me a decimal number: "))*

*a = hex(dec)*

*print ("{} in decimal is {} in hex".format(dec, a))*

**Running the script**

root@NetworkAutomation-2:~# **python3 dec\_to\_hex.py**

Give me a decimal number: 10

10 in decimal is 0xa in hex

root@NetworkAutomation-2:~# **python3 dec\_to\_hex.py**

Give me a decimal number: 100

100 in decimal is 0x64 in hex

### Lecture 106 - Check built-in methods for Python

Use the dir function:

root@NetworkAutomation-2:~# **python3**

Python 3.5.2 (default, Nov 23 2017, 16:37:01)

[GCC 5.4.0 20160609] on linux

Type "help", "copyright", "credits" or "license" for more information.

>>>

>>> **dir()**

['\_\_builtins\_\_', '\_\_doc\_\_', '\_\_loader\_\_', '\_\_name\_\_', '\_\_package\_\_', '\_\_spec\_\_']

>>>

>>>

>>> **dir(str)**

['\_\_add\_\_', '\_\_class\_\_', '\_\_contains\_\_', '\_\_delattr\_\_', '\_\_dir\_\_', '\_\_doc\_\_', '\_\_eq\_\_', '\_\_format\_\_', '\_\_ge\_\_', '\_\_getattribute\_\_', '\_\_getitem\_\_', '\_\_getnewargs\_\_', '\_\_gt\_\_', '\_\_hash\_\_', '\_\_init\_\_', '\_\_iter\_\_', '\_\_le\_\_', '\_\_len\_\_', '\_\_lt\_\_', '\_\_mod\_\_', '\_\_mul\_\_', '\_\_ne\_\_', '\_\_new\_\_', '\_\_reduce\_\_', '\_\_reduce\_ex\_\_', '\_\_repr\_\_', '\_\_rmod\_\_', '\_\_rmul\_\_', '\_\_setattr\_\_', '\_\_sizeof\_\_', '\_\_str\_\_', '\_\_subclasshook\_\_', 'capitalize', 'casefold', 'center', 'count', 'encode', 'endswith', 'expandtabs', 'find', 'format', 'format\_map', 'index', 'isalnum', 'isalpha', 'isdecimal', 'isdigit', 'isidentifier', 'islower', 'isnumeric', 'isprintable', 'isspace', 'istitle', 'isupper', 'join', 'ljust', 'lower', 'lstrip', 'maketrans', 'partition', 'replace', 'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip', 'split', 'splitlines', 'startswith', 'strip', 'swapcase', 'title', 'translate', 'upper', 'zfill']

>>>

>>> **a = 'test'**

>>> **dir(a)**

['\_\_add\_\_', '\_\_class\_\_', '\_\_contains\_\_', '\_\_delattr\_\_', '\_\_dir\_\_', '\_\_doc\_\_', '\_\_eq\_\_', '\_\_format\_\_', '\_\_ge\_\_', '\_\_getattribute\_\_', '\_\_getitem\_\_', '\_\_getnewargs\_\_', '\_\_gt\_\_', '\_\_hash\_\_', '\_\_init\_\_', '\_\_iter\_\_', '\_\_le\_\_', '\_\_len\_\_', '\_\_lt\_\_', '\_\_mod\_\_', '\_\_mul\_\_', '\_\_ne\_\_', '\_\_new\_\_', '\_\_reduce\_\_', '\_\_reduce\_ex\_\_', '\_\_repr\_\_', '\_\_rmod\_\_', '\_\_rmul\_\_', '\_\_setattr\_\_', '\_\_sizeof\_\_', '\_\_str\_\_', '\_\_subclasshook\_\_', 'capitalize', 'casefold', 'center', 'count', 'encode', 'endswith', 'expandtabs', 'find', 'format', 'format\_map', 'index', 'isalnum', 'isalpha', 'isdecimal', 'isdigit', 'isidentifier', 'islower', 'isnumeric', 'isprintable', 'isspace', 'istitle', 'isupper', 'join', 'ljust', 'lower', 'lstrip', 'maketrans', 'partition', 'replace', 'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip', 'split', 'splitlines', 'startswith', 'strip', 'swapcase', 'title', 'translate', 'upper', 'zfill']

>>>

>>>

>>> **a.capitalize()**

'Test'

>>>

>>> **a.casefold()**

'test'

>>>

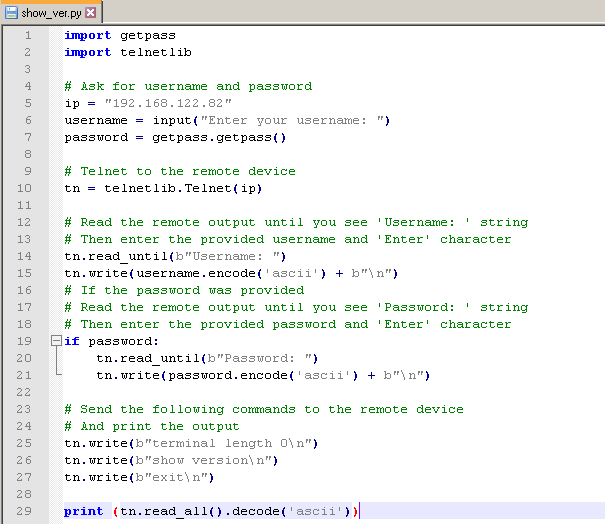
>>> **a.count('t')**

2

### Section 14 (telnetlib) - Script 1

The following script:

* Connects via telnet to a device
* The IP of the device is specified manually (the HOST variable)
* The user is prompted for user and password
* The provided credentials are used by the script in the telnet connection
* The script collects 'show version' output from the remote device



root@NetworkAutomation-1:~# **cat > show\_ver.py**

*import getpass*

*import telnetlib*

*# Ask for username and password*

*ip = "192.168.122.82"*

*username = input("Enter your username: ")*

*password = getpass.getpass()*

*tn = telnetlib.Telnet(ip)*

*tn.read\_until(b"Username: ")*

*tn.write(username.encode('ascii') + b"\n")*

*if password:*

*tn.read\_until(b"Password: ")*

*tn.write(password.encode('ascii') + b"\n")*

*tn.write(b"terminal length 0\n")*

*tn.write(b"show version\n")*

*tn.write(b"exit\n")*

*print (tn.read\_all().decode('ascii'))*

**Running the script**

root@NetworkAutomation-1:~# **python3 show\_ver.py**

Enter your username: **mikis**

Password:

SW1#**terminal length 0**

SW1#**show version**

Cisco IOS Software, Solaris Software (I86BI\_LINUXL2-ADVENTERPRISEK9-M), Experimental Version 15.1(20130726:213425) [dstivers-july26-2013-team\_track 104]

Copyright (c) 1986-2013 by Cisco Systems, Inc.

Compiled Fri 26-Jul-13 15:56 by dstivers

ROM: Bootstrap program is Linux

SW1 uptime is 38 minutes

System returned to ROM by reload at 0

System image file is "unix:/opt/gns3/images/IOU/i86bi-linux-l2-adventerprisek9-15.1a.bin"

Last reload reason: Unknown reason

This product contains cryptographic features and is subject to United

States and local country laws governing import, export, transfer and

use. Delivery of Cisco cryptographic products does not imply

third-party authority to import, export, distribute or use encryption.

Importers, exporters, distributors and users are responsible for

compliance with U.S. and local country laws. By using this product you

agree to comply with applicable laws and regulations. If you are unable

to comply with U.S. and local laws, return this product immediately.

A summary of U.S. laws governing Cisco cryptographic products may be found at:

http://www.cisco.com/wwl/export/crypto/tool/stqrg.html

If you require further assistance please contact us by sending email to

export@cisco.com.

Linux Unix (Intel-x86) processor with 189092K bytes of memory.

Processor board ID 2048001

16 Ethernet interfaces

1 Virtual Ethernet interface

16K bytes of NVRAM.

Configuration register is 0x0

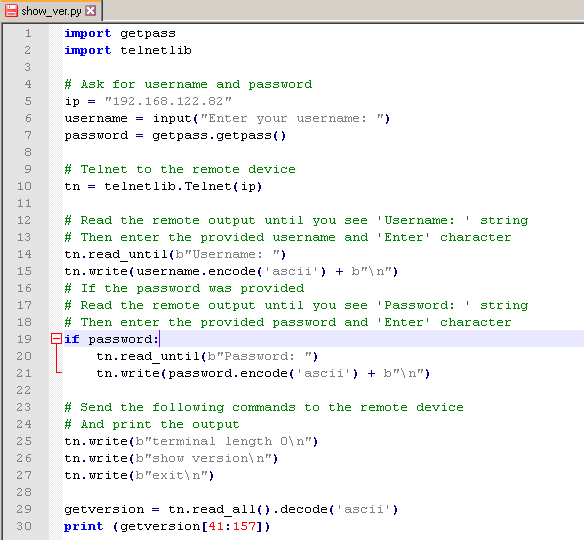
SW1#**exit**

root@NetworkAutomation-1:~#

### Section 14 (telnetlib) - Script 2

The following script:

* Connects via telnet to a device
* The IP of the device is specified manually (the 'ip' variable)
* The user is prompted for user and password
* The provided credentials are used by the script in the telnet connection
* The script collects 'show version' output from the remote device and shows a specific part



root@NetworkAutomation-1:~# **cat > show\_ver2.py**

*import getpass*

*import telnetlib*

*# Ask for username and password*

*ip = "192.168.122.82"*

*username = input("Enter your username: ")*

*password = getpass.getpass()*

*# Telnet to the remote device*

*tn = telnetlib.Telnet(ip)*

*# Read the remote output until you see 'Username: ' string*

*# Then enter the provided username and 'Enter' character*

*tn.read\_until(b"Username: ")*

*tn.write(username.encode('ascii') + b"\n")*

*# If the password was provided*

*# Read the remote output until you see 'Password: ' string*

*# Then enter the provided password and 'Enter' character*

*if password:*

*tn.read\_until(b"Password: ")*

*tn.write(password.encode('ascii') + b"\n")*

*# Send the following commands to the remote device*

*# And print the output*

*tn.write(b"terminal length 0\n")*

*tn.write(b"show version\n")*

*tn.write(b"exit\n")*

*getversion = tn.read\_all().decode('ascii')*

*print (getversion[41:157])*

**Running the script**

root@NetworkAutomation-1:~# **python3 show\_ver2.py**

Enter your username: mikis

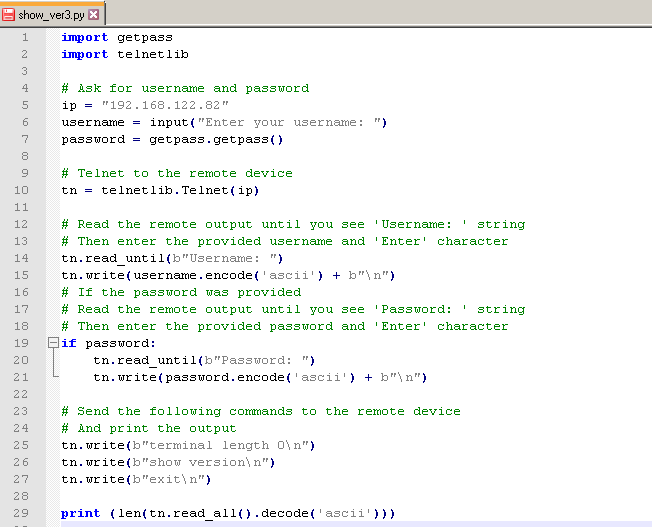
Password:

Cisco IOS Software, Solaris Software (I86BI\_LINUXL2-ADVENTERPRISEK9-M), Experimental Version 15.1(20130726:213425)

### Section 14 (telnetlib) - Script 3 - print length in bytes of specific part of output

The following script:

* Connects via telnet to a device
* The IP of the device is specified manually (the 'ip' variable)
* The user is prompted for user and password
* The provided credentials are used by the script in the telnet connection
* The script collects 'show version' output from the remote device and shows the length in bytes (characters)



root@NetworkAutomation-1:~# **cat > show\_ver3.py**

*import getpass*

*import telnetlib*

*# Ask for username and password*

*ip = "192.168.122.82"*

*username = input("Enter your username: ")*

*password = getpass.getpass()*

*# Telnet to the remote device*

*tn = telnetlib.Telnet(ip)*

*# Read the remote output until you see 'Username: ' string*

*# Then enter the provided username and 'Enter' character*

*tn.read\_until(b"Username: ")*

*tn.write(username.encode('ascii') + b"\n")*

*# If the password was provided*

*# Read the remote output until you see 'Password: ' string*

*# Then enter the provided password and 'Enter' character*

*if password:*

*tn.read\_until(b"Password: ")*

*tn.write(password.encode('ascii') + b"\n")*

*# Send the following commands to the remote device*

*# And print the output*

*tn.write(b"terminal length 0\n")*

*tn.write(b"show version\n")*

*tn.write(b"exit\n")*

*print (len(tn.read\_all().decode('ascii')))*

**Running the script**

root@NetworkAutomation-1:~# **python3 show\_ver3.py**

Enter your username: mikis

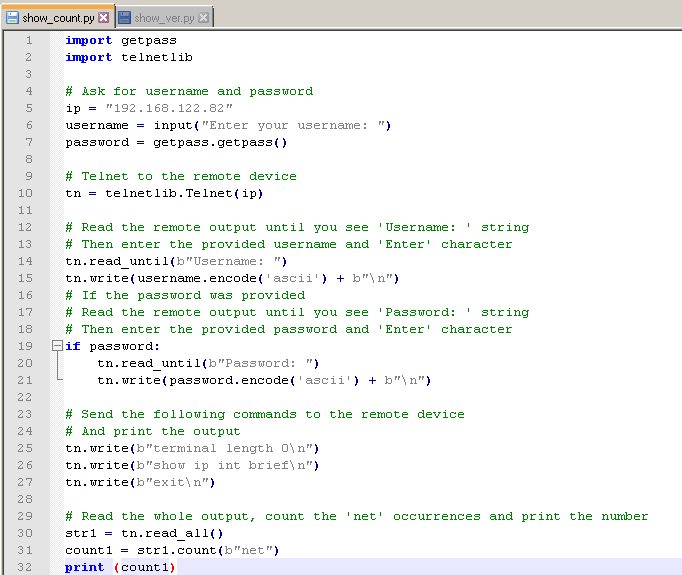
Password:

1533

### Section 14 (telnetlib) - Script 4 - print number of occurrences

The following script:

* Connects via telnet to a device
* The user is prompted for user and password
* The provided credentials are used by the script in the telnet connection
* The script collects 'show ip int brief' output from the remote device and shows the number of occurrences of specific string



root@NetworkAutomation-1:~# **cat > show\_count.py**

*import getpass*

*import telnetlib*

*# Ask for username and password*

*ip = "192.168.122.82"*

*username = input("Enter your username: ")*

*password = getpass.getpass()*

*# Telnet to the remote device*

*tn = telnetlib.Telnet(ip)*

*# Read the remote output until you see 'Username: ' string*

*# Then enter the provided username and 'Enter' character*

*tn.read\_until(b"Username: ")*

*tn.write(username.encode('ascii') + b"\n")*

*# If the password was provided*

*# Read the remote output until you see 'Password: ' string*

*# Then enter the provided password and 'Enter' character*

*if password:*

*tn.read\_until(b"Password: ")*

*tn.write(password.encode('ascii') + b"\n")*

*# Send the following commands to the remote device*

*# And print the output*

*tn.write(b"terminal length 0\n")*

*tn.write(b"show ip int brief\n")*

*tn.write(b"exit\n")*

*# Read the whole output, count the 'net' occurrences and print the number*

*str1 = tn.read\_all()*

*count1 = str1.count(b"net")*

*print (count1)*

**Running the script**

root@NetworkAutomation-1:~# **python3 show\_count.py**

Enter your username: **mikis**

Password:

16

### Section 20 - While loop 1

**The goal**

I want to do this:  
root@NetworkAutomation-1:~# **python3 whileloop1.py**

0

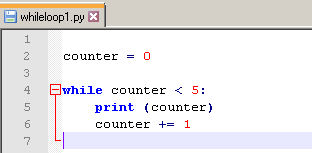
1

2

3

4

**The solution**



root@NetworkAutomation-1:~# **cat > whileloop1.py**

*counter = 0*

*while counter < 5:*

*print (counter)*

*counter += 1*

### Section 20 - While loop 2 - Infinite while with a break

**The goal**

I want to do this:

root@NetworkAutomation-1:~# **python3 whileloop2.py**

String to make lowercase [type "q" t quit]: **Test**

test

String to make lowercase [type "q" t quit]: **1q**

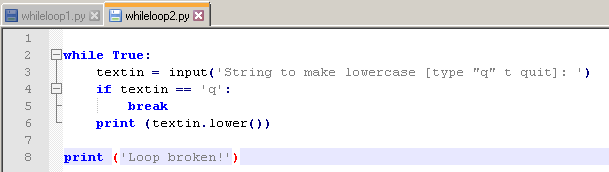
1q

String to make lowercase [type "q" t quit]: **q**

Loop broken!

root@NetworkAutomation-1:~#

**The solution**



root@NetworkAutomation-1:~# **cat > whileloop2.py**

*while True:*

*textin = input('String to make lowercase [type "q" t quit]: ')*

*if textin == 'q':*

*break*

*print (textin.lower())*

*print ('Loop broken!')*

### Section 20 - While loop 3 - Infinite while with continue

**The goal**

I want to do this:

root@NetworkAutomation-1:~# **python3 whileloop3.py**

Enter an integer value [type 'q' to quite]: **1**

The number 1 is odd. Its square is: 1

Enter an integer value [type 'q' to quite]: **2**

This was an Even number, the loop continues

Enter an integer value [type 'q' to quite]: **3**

The number 3 is odd. Its square is: 9

Enter an integer value [type 'q' to quite]: **4**

This was an Even number, the loop continues

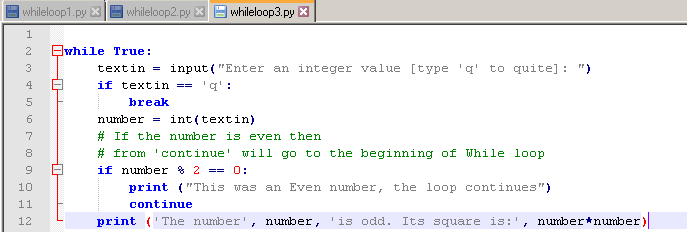
Enter an integer value [type 'q' to quite]: **5**

The number 5 is odd. Its square is: 25

Enter an integer value [type 'q' to quite]: **q**

root@NetworkAutomation-1:~#

**The solution**



root@NetworkAutomation-1:~# **cat > whileloop3.py**

*while True:*

*textin = input("Enter an integer value [type 'q' to quite]: ")*

*if textin == 'q':*

*break*

*number = int(textin)*

*# If the number is even then*

*# from 'continue' will go to the beginning of While loop*

*if number % 2 == 0:*

*print ("This was an Even number, the loop continues")*

*continue*

*print ('The number', number, 'is odd. Its square is:', number\*number)*

### Section 20 - While loop 4 - Infinite while with a list

**The goal**

I want to do this:

root@NetworkAutomation-1:~# **python3 whileloop4.py**

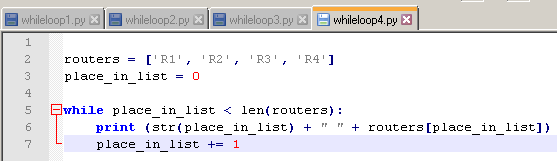
0 R1

1 R2

2 R3

3 R4

**The solution**



root@NetworkAutomation-1:~# **cat > whileloop4.py**

*routers = ['R1', 'R2', 'R3', 'R4']*

*place\_in\_list = 0*

*while place\_in\_list < len(routers):*

*print (str(place\_in\_list) + " " + routers[place\_in\_list])*

*place\_in\_list += 1*

### Section 20 - While loop 5 - Infinite while with else

**The goal**

I want to do this:

root@NetworkAutomation-1:~# **python3 whileloop5.py**

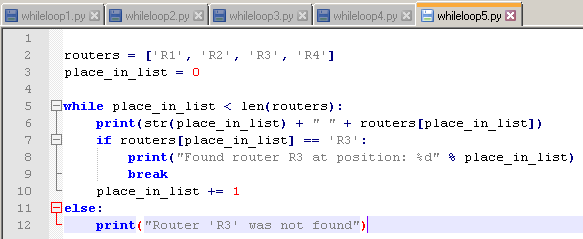
0 R1

1 R2

2 R3

Found router R3 at position: 2

**The solution**



root@NetworkAutomation-1:~# **cat > whileloop5.py**

*routers = ['R1', 'R2', 'R3', 'R4']*

*place\_in\_list = 0*

*while place\_in\_list < len(routers):*

*print(str(place\_in\_list) + " " + routers[place\_in\_list])*

*if routers[place\_in\_list] == 'R3':*

*print("Found router R3 at position: %d" % place\_in\_list)*

*break*

*place\_in\_list += 1*

*else:*

*print("Router 'R3' was not found")*

### Section 21 - For loop 1a

**The goal**

I want to do this:

root@NetworkAutomation-1:~# **python3 forloop1.py**

Vendor is: Cisco

Vendor is: Arista

Vendor is: PAN

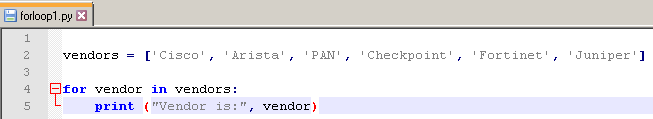
Vendor is: Checkpoint

Vendor is: Fortinet

Vendor is: Juniper

root@NetworkAutomation-1:~#

**The solution**



root@NetworkAutomation-1:~# cat > forloop1.py

*vendors = ['Cisco', 'Arista', 'PAN', 'Checkpoint', 'Fortinet', 'Juniper']*

*for vendor in vendors:*

*print ("Vendor is:", vendor)*

### Section 21 - For loop 1b

**The goal**

I want to do this:

root@NetworkAutomation-1:~# **python3 forloop1b.py**

R

o

u

t

e

r

s

a

n

d

S

w

i

t

c

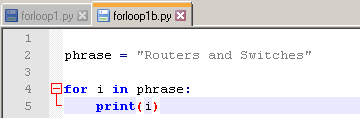
h

e

s

root@NetworkAutomation-1:~#

**The solution**



root@NetworkAutomation-1:~# **cat > forloop1b.py**

*phrase = "Routers and Switches"*

*for i in phrase:*

*print(i)*

### Section 21 - For loop 2 - 'for' loop with Dictionary - print Keys

Consider this Python dictionary:

SW1 = {

'device\_type': 'cisco\_ios',

'ip': '192.168.122.72',

'username': 'mikis',

'password': 'cisco'

}

**The goal**

I want to print its keys e.g.

root@NetworkAutomation-1:~# **python3 forloop2.py**

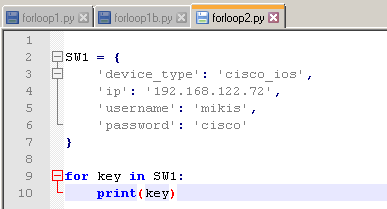
ip

password

username

device\_type

**The solution**



root@NetworkAutomation-1:~# **cat > forloop2.py**

SW1 = {

'device\_type': 'cisco\_ios',

'ip': '192.168.122.72',

'username': 'mikis',

'password': 'cisco'

}

for key in SW1:

print(key)

### Section 21 - For loop 2b - 'for' loop with Dictionary - print Values Way 1

**The goal**

I have a Dictionary and I want to print its values e.g.

root@NetworkAutomation-1:~# **python3 forloop2b.py**

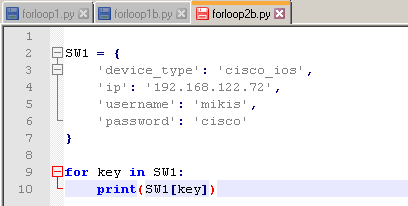
192.168.122.72

cisco

mikis

cisco\_ios

**The solution**

**

root@NetworkAutomation-1:~# **cat > forloop2b.py**

*SW1 = {*

*'device\_type': 'cisco\_ios',*

*'ip': '192.168.122.72',*

*'username': 'mikis',*

*'password': 'cisco'*

*}*

*for key in SW1:*

*print(SW1[key])*

### Section 21 - For loop 2b - 'for' loop with Dictionary - print Values Way 2

**The goal**

I have a Dictionary and I want to print its values e.g.

root@NetworkAutomation-1:~# **python3 forloop2b.py**

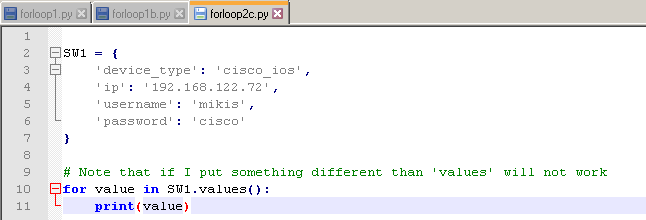
192.168.122.72

cisco

mikis

cisco\_ios

**The solution**

**

root@NetworkAutomation-1:~# **cat > forloop2c.py**

*SW1 = {*

*'device\_type': 'cisco\_ios',*

*'ip': '192.168.122.72',*

*'username': 'mikis',*

*'password': 'cisco'*

*}*

*# Note that if I put something different than 'values' will not work*

*for value in SW1.values():*

*print(value)*

### Section 21 - For loop 2b - 'for' loop with Dictionary - print Keys and Values

**The goal**

I have a Dictionary and I want to print its keys and values e.g.

root@NetworkAutomation-1:~# **python3 forloop2d.py**

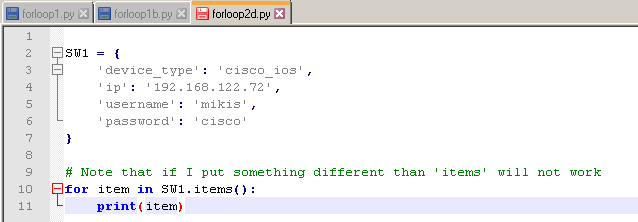
('username', 'mikis')

('device\_type', 'cisco\_ios')

('password', 'cisco')

('ip', '192.168.122.72')

**The solution**



root@NetworkAutomation-1:~# **cat > forloop2d.py**

*SW1 = {*

*'device\_type': 'cisco\_ios',*

*'ip': '192.168.122.72',*

*'username': 'mikis',*

*'password': 'cisco'*

*}*

*# Note that if I put something different than 'items' will not work*

*for item in SW1.items():*

*print(item)*

### Section 21 - For loop 3 - 'for' loop with Lists and If

**The goal**

I have 2 lists and I want to print their differences e.g.

root@NetworkAutomation-1:~# **python3 forloop3.py**

Device mikis\_R1 is an approved one!

Device mikis\_R2 is an approved one!

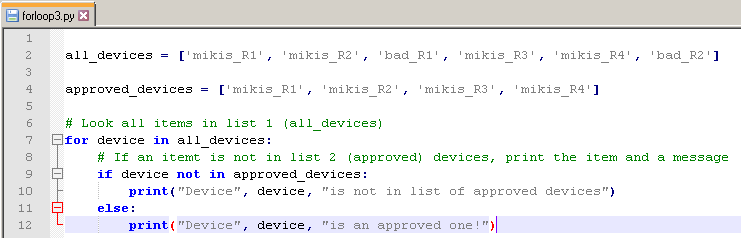
Device bad\_R1 is not in list of approved devices

Device mikis\_R3 is an approved one!

Device mikis\_R4 is an approved one!

Device bad\_R2 is not in list of approved devices

**The solution**

**

root@NetworkAutomation-1:~# **cat > forloop3.py**

*all\_devices = ['mikis\_R1', 'mikis\_R2', 'bad\_R1', 'mikis\_R3', 'mikis\_R4', 'bad\_R2']*

*approved\_devices = ['mikis\_R1', 'mikis\_R2', 'mikis\_R3', 'mikis\_R4']*

*# Look all items in list 1 (all\_devices)*

*for device in all\_devices:*

*# If an itemt is not in list 2 (approved) devices, print the item and a message*

*if device not in approved\_devices:*

*print("Device", device, "is not in list of approved devices")*

*else:*

*print("Device", device, "is an approved one!")*

### Section 22 - Ranges 1 - SSH to a device and create multiple loopbacks

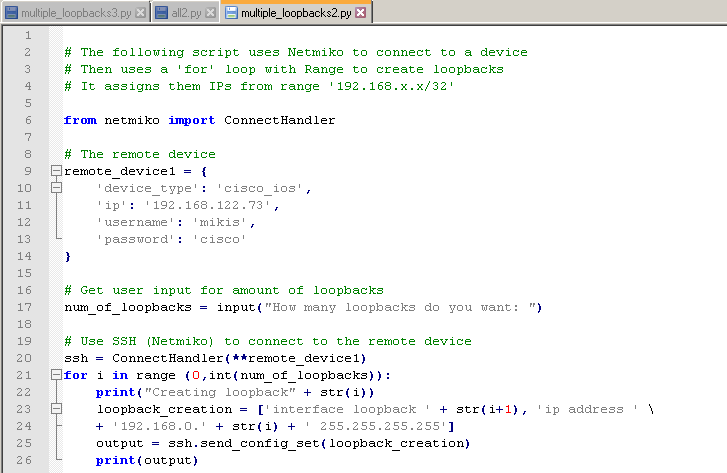
**The goal**

I have a remote device

I want to connect using SSH and create multiple loopbacks

The user specifies the amount interactively

**The solution**



root@NetworkAutomation-1:~# **cat > multiple\_loopbacks2.py**

*# The following script uses Netmiko to connect to a device*

*# Then uses a 'for' loop with Range to create loopbacks*

*# It assigns them IPs from range '192.168.x.x/32'*

*from netmiko import ConnectHandler*

*# The remote device*

*remote\_device1 = {*

*'device\_type': 'cisco\_ios',*

*'ip': '192.168.122.73',*

*'username': 'mikis',*

*'password': 'cisco'*

*}*

*# Get user input for amount of loopbacks*

*num\_of\_loopbacks = input("How many loopbacks do you want: ")*

*# Use SSH (Netmiko) to connect to the remote device*

*ssh = ConnectHandler(\*\*remote\_device1)*

*for i in range (0,int(num\_of\_loopbacks)):*

*print("Creating loopback" + str(i))*

*loopback\_creation = ['interface loopback ' + str(i), 'ip address ' \*

*+ '192.168.0.' + str(i+1) + ' 255.255.255.255']*

*output = ssh.send\_config\_set(loopback\_creation)*

*print(output)*

**Running the script**

root@NetworkAutomation-1:~# **python3 multiple\_loopbacks2.py**

How many loopbacks do you want: 3

Creating loopback0

config term

Enter configuration commands, one per line. End with CNTL/Z.

R2(config)#interface loopback 0

R2(config-if)#ip address 192.168.0.1 255.255.255.255

R2(config-if)#end

R2#

Creating loopback1

config term

Enter configuration commands, one per line. End with CNTL/Z.

R2(config)#interface loopback 1

R2(config-if)#ip address 192.168.0.2 255.255.255.255

R2(config-if)#end

R2#

Creating loopback2

config term

Enter configuration commands, one per line. End with CNTL/Z.

R2(config)#interface loopback 2

R2(config-if)#ip address 192.168.0.3 255.255.255.255

R2(config-if)#end

R2#

root@NetworkAutomation-1:~#

**The result**

R2#**show ip int br | e un**

Interface IP-Address OK? Method Status Protocol

FastEthernet0/0 192.168.122.73 YES NVRAM up up

Loopback0 192.168.0.1 YES manual up up

Loopback1 192.168.0.2 YES manual up up

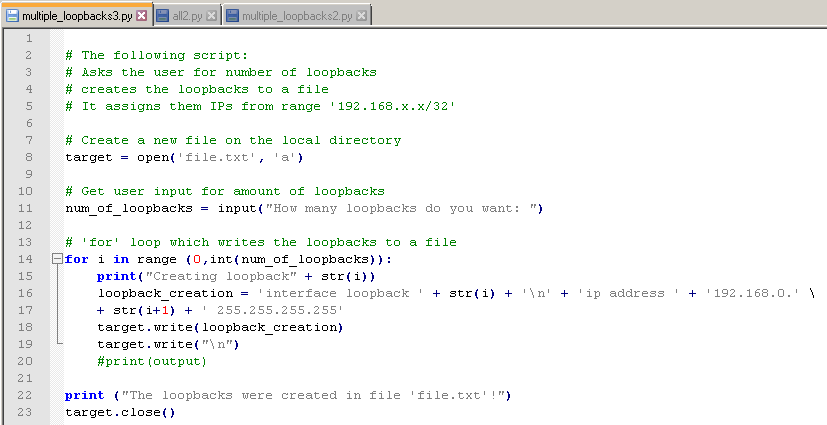
Loopback2 192.168.0.3 YES manual up up

### Section 22 - Ranges 2 - Create multiple loopbacks (<254) in a file

**The goal**

The user specifies the amount interactively. The loopbacks are written to a file

**The solution**

**

root@NetworkAutomation-1:~# **cat > multiple\_loopbacks3.py**

*# The following script:*

*# Asks the user for number of loopbacks*

*# creates the loopbacks to a file*

*# It assigns them IPs from range '192.168.x.x/32'*

*# Create a new file on the local directory*

*target = open('file.txt', 'a')*

*# Get user input for amount of loopbacks*

*num\_of\_loopbacks = input("How many loopbacks do you want: ")*

*# 'for' loop which writes the loopbacks to a file*

*for i in range (0,int(num\_of\_loopbacks)):*

*print("Creating loopback" + str(i))*

*loopback\_creation = 'interface loopback ' + str(i) + '\n' + 'ip address ' + '192.168.0.' \*

*+ str(i+1) + ' 255.255.255.255'*

*target.write(loopback\_creation)*

*target.write("\n")*

*#print(output)*

*print ("The loopbacks were created in file 'file.txt'!")*

*target.close()*

**Running the script**

root@NetworkAutomation-1:~# **python3 multiple\_loopbacks3.py**

How many loopbacks do you want: **5**

Creating loopback0

Creating loopback1

Creating loopback2

Creating loopback3

Creating loopback4

The loopbacks were created in file 'file.txt'!

**The result**

root@NetworkAutomation-1:~# **cat file.txt**

interface loopback 0

ip address 192.168.0.1 255.255.255.255

interface loopback 1

ip address 192.168.0.2 255.255.255.255

interface loopback 2

ip address 192.168.0.3 255.255.255.255

interface loopback 3

ip address 192.168.0.4 255.255.255.255

interface loopback 4

ip address 192.168.0.5 255.255.255.255

root@NetworkAutomation-1:~#

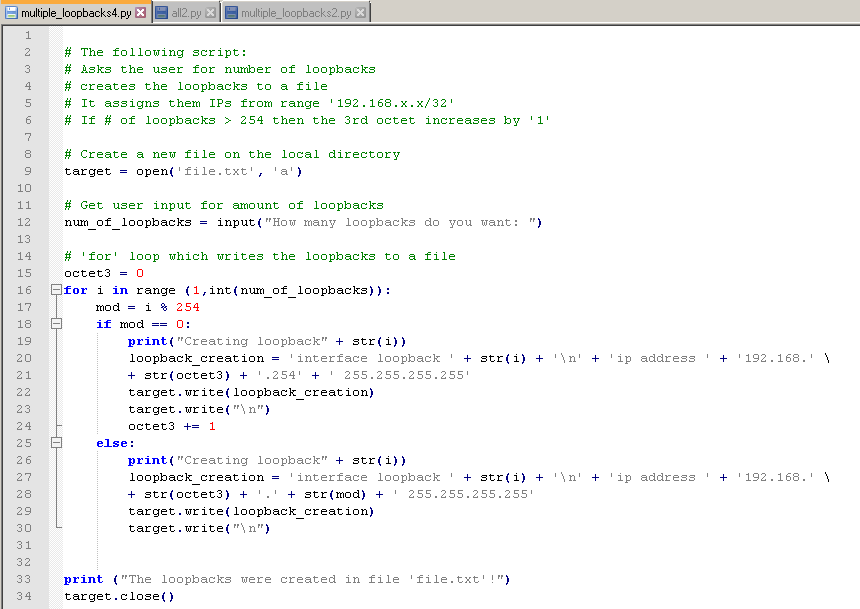
### Section 22 - Ranges 2b - Create multiple loopbacks in a file

**The goal**

The user specifies the amount interactively. The loopbacks are written to a file

If the loopbacks are > 254 then the third octet increases by 1

**The solution**



root@NetworkAutomation-1:~# **cat > multiple\_loopbacks4.py**

*# The following script:*

*# Asks the user for number of loopbacks*

*# creates the loopbacks to a file*

*# It assigns them IPs from range '192.168.x.x/32'*

*# If # of loopbacks > 254 then the 3rd octet increases by '1'*

*# Create a new file on the local directory*

*target = open('file.txt', 'a')*

*# Get user input for amount of loopbacks*

*num\_of\_loopbacks = input("How many loopbacks do you want: ")*

*# 'for' loop which writes the loopbacks to a file*

*octet3 = 0*

*for i in range (1,int(num\_of\_loopbacks)):*

*mod = i % 254*

*if mod == 0:*

*print("Creating loopback" + str(i))*

*loopback\_creation = 'interface loopback ' + str(i) + '\n' + 'ip address ' + '192.168.' \*

*+ str(octet3) + '.254' + ' 255.255.255.255'*

*target.write(loopback\_creation)*

*target.write("\n")*

*octet3 += 1*

*else:*

*print("Creating loopback" + str(i))*

*loopback\_creation = 'interface loopback ' + str(i) + '\n' + 'ip address ' + '192.168.' \*

*+ str(octet3) + '.' + str(mod) + ' 255.255.255.255'*

*target.write(loopback\_creation)*

*target.write("\n")*

*print ("The loopbacks were created in file 'file.txt'!")*

*target.close()*

**Running the script**

root@NetworkAutomation-1:~# **python3 multiple\_loopbacks4.py**

How many loopbacks do you want: **500**

Creating loopback1

Creating loopback2

Creating loopback3

Creating loopback4

Creating loopback5

Creating loopback6

...

Creating loopback497

Creating loopback498

Creating loopback499

The loopbacks were created in file 'file.txt'!

**The result**

root@NetworkAutomation-1:~# **cat file.txt**

interface loopback 1

ip address 192.168.0.1 255.255.255.255

interface loopback 2

ip address 192.168.0.2 255.255.255.255

interface loopback 3

ip address 192.168.0.3 255.255.255.255

interface loopback 4

...

interface loopback 254

ip address 192.168.0.254 255.255.255.255

interface loopback 255

ip address 192.168.1.1 255.255.255.255

interface loopback 256

ip address 192.168.1.2 255.255.255.255