Python for Network Engineers

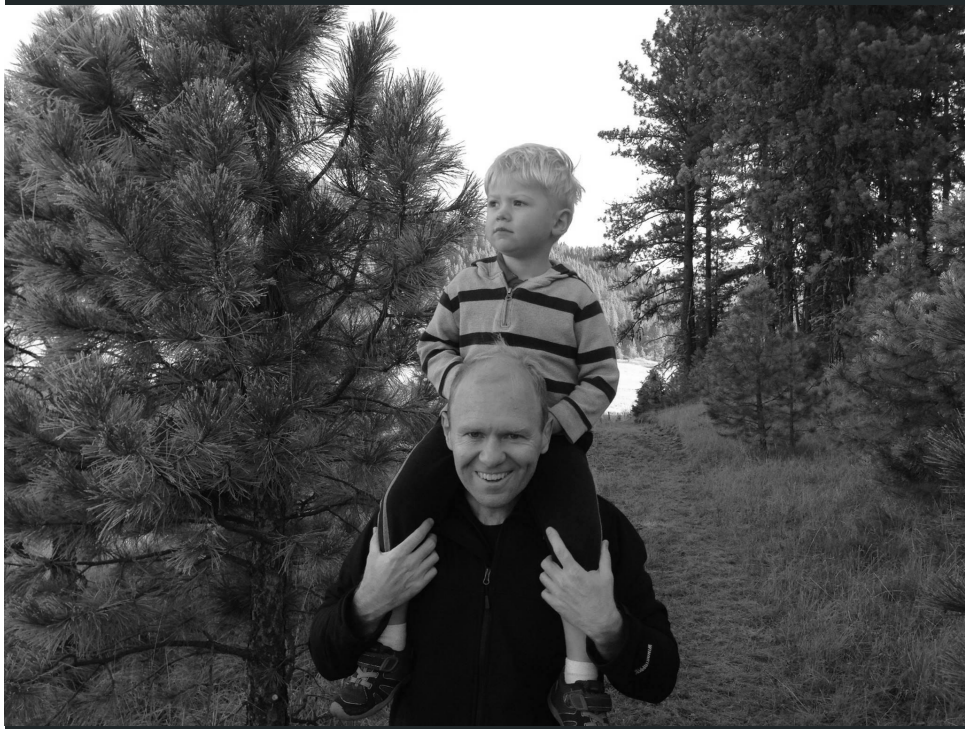
Onsite Training Session

$ whoami

Kirk Byers
Network Engineer:
CCIE #6243 (emeritus)

Programmer:
Netmiko
NAPALM

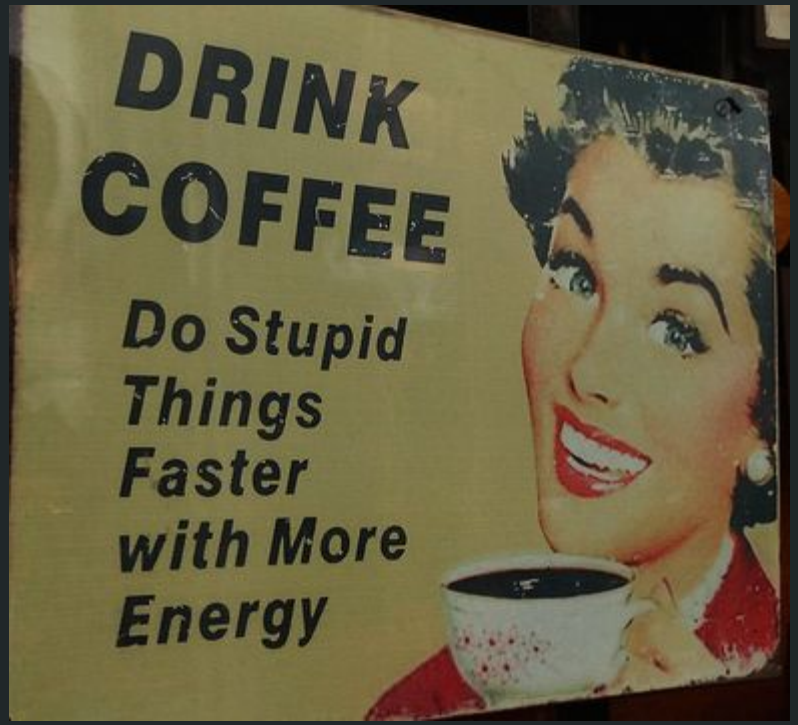
Teach Python and Ansible
SF Network Automation Meetup



General:

8:30AM - 4:45PM
Lunch
Some breaks

Focused
Minimize Distractions
Exercises and Examples
Examples in the Python Shell



Flickr: Ben Sutherland

Day1 Schedule

1. Intro
2. GIT
3. Python Fundamentals - General
4. Strings
5. Numbers
6. Files
7. Lists
7. Booleans / None
8. Conditionals
9. Loops
10. Dictionaries
11. Exceptions
12. Functions
13. Python Code Structure
14. Classes and Objects

Git

- Why care about Git?
- Git and GitHub
- Cloning a Project
- git init / git add / git rm / git commit
- git pull / git push
- Managing Git branches
- Making a Pull Request
- Git Rebase

Why Python?

- Widely supported (meaning lots of library support)
- Easily available on systems
- Language accommodates beginners through advanced
- Maintainable
- Allows for easy code reuse
- High-level

Python Characteristics

Indentation matters.

Use spaces not tabs.

Python programmers are particular.

Py2 or Py3.

General Items

The Python interpreter shell

Assignment and variable names

Python naming conventions

Printing to standard out/reading from standard in

Quotes, double quotes, triple quotes

comments

dir() and help()

Strings

- String methods
- Chaining
- split()
- strip()
- substr in string
- unicode
- raw strings
- re

Numbers

Integers
Floats
Math Operators (+, -, *, /, **, %)
Strange Behavior of Integer Division

Writing to a file/reading from a file:

```
with open(file_name, "w") as f:
    f.write(output)
```

```
with open(file_name) as f:
    output = f.read()
```

Lists

Zero-based indices

.append()

List slices

Tuple

Copying a list

.join()

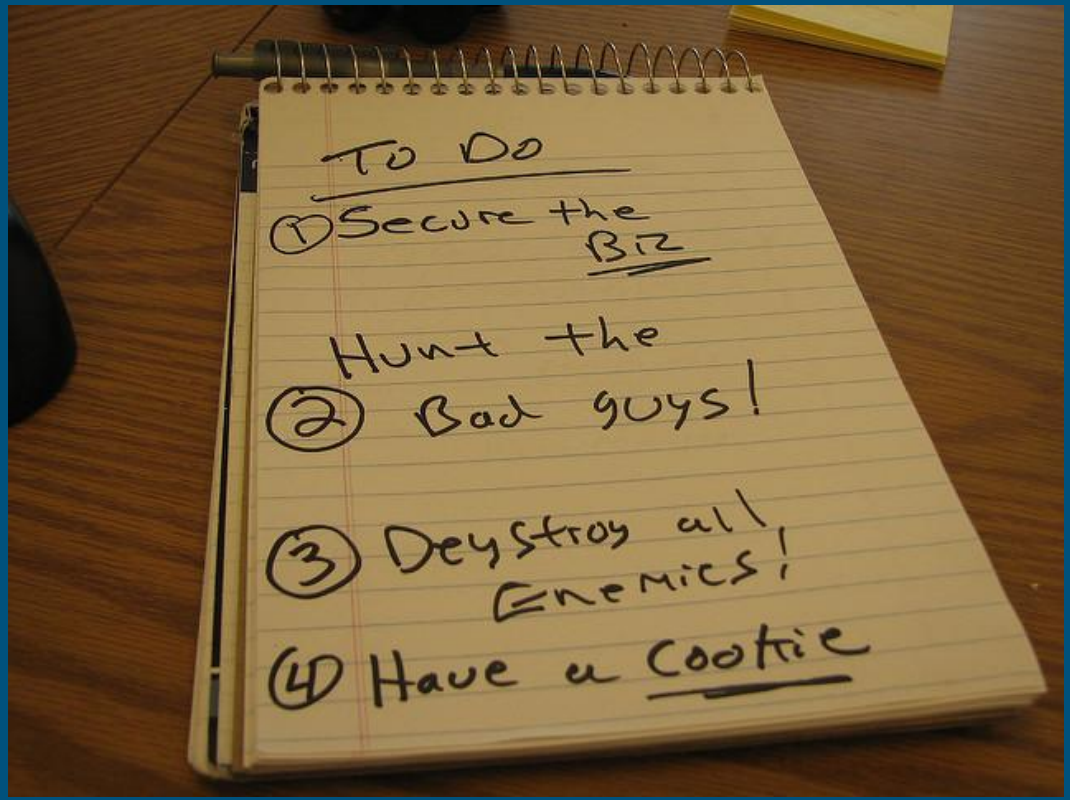


Photo: Purple Slog (Flickr)

Booleans and None

Boolean operators (and, or, not)

is

Truish

Comparison operators (==, !=, <, >, >=, <=)

None

Conditionals

```
if a == 15:
    print "Hello"
elif a >= 7:
    print "Something"
else:
    print "Nada"
```

Loops

- for
- while
- break
- continue
- range(len())
- enumerate

Photo: Mário Monte Filho (Flickr)

For/while syntax

```
for my_var in iterable:
    print my_var


i = 0
while i < 10:
    print i
    i += 1
```

Dictionaries

- Creating
- Updating
- get()
- Iterating over keys
- Iterating over keys and values

Photo: Holger Zscheyge (Flickr)

Exception Handling

```
try:
    my_dict[‘missing_key’]
except KeyError:
    do_something
```

- Trying to gracefully handle errors.
- finally: - always ran if you have a cleanup condition.

Functions:

- Defining a function
- Positional arguments
- Named arguments
- Mixing positional and named arguments
- Default values
- Passing in *args, **kwargs
- Functions and promoting the reuse of code

Classes and Objects

```
class NetDevice(object):
    def __init__(self, ip_addr, username, password):
        self.ip_addr = ip_addr
        self.username = username
        self.password = password

    def test_method(self):
        print "Device IP is: {}".format(self.ip_addr)
        print "Username is: {}".format(self.username)


    rtr1 = NetDevice('10.22.1.1', 'admin', 'passw')
    rtr1.test_method()
```

Writing reusable code

Basic Building Blocks (functions/classes)
Python Modules
if __name__
Python Packages
Don’t repeat yourself

Flickr: Koka Sexton

Day2 Schedule: Python Applied to Networking

1. Review Day1
2. Managing Python Libraries
3. Modules and Packages
4. Namespaces
5. Regular expressions
6. SNMP
7. Email notifications
8. CiscoConfParse
9. Telnetlib (optional)
10. Python and SSH (Netmiko)

Libraries

sys.path

PYTHONPATH

Installing packages (pip)

import x

from x import y

Photo: Viva Vivanista (Flickr)

Modules/Packages and Namespaces

Modules and Packages

- Reusing code across programs
- __init__.py
- Integrating a package together

Namespaces

Python Regular Expresions

import re

Other re methods

re.split()

re.sub()

re.findall()

re.search(pattern, string)

- always use raw strings
- re.M/re.MULTILINE
- re.DOTALL
- re.I
- Parenthesis to retain patterns
- greedy/not greedy (.*?)

SNMP

```
#!/usr/bin/env python
import getpass
import snmp_helper

SYS_DESCR = '1.3.6.1.2.1.1.1.0'
ip_addr1 = raw_input("pynet-rtr1 IP address: ")
community_string = getpass.getpass(prompt="Community string: ")
pynet_rtr1 = (ip_addr1, community_string, 161)

snmp_data = snmp_helper.snmp_get_oid(pynet_rtr1, oid=SYS_DESCR)
output = snmp_helper.snmp_extract(snmp_data)
print output
```

Email Notifications

```
#!/usr/bin/env python
from email_helper import send_mail

sender = 'twb@twb-tech.com'
recipient = 'ktbyersx@gmail.com'
subject = 'This is a test message.'
message = '''Whatever'''

send_mail(recipient, subject, message, sender)
```

CiscoConfParse

```
#!/usr/bin/env python
from ciscoconfparse import CiscoConfParse

cisco_file = 'cisco_config.txt'
cisco_cfg = CiscoConfParse(cisco_file)
intf_obj = cisco_cfg.find_objects(r"^interf")
print
for intf in intf_obj:
    print intf.text
    for child in intf.children:
        print child.text
    print
```

Telnetlib

```
import telnetlib
import time
from getpass import getpass
TELNET_PORT = 23
TELNET_TIMEOUT = 6

password = getpass()
remote_conn = telnetlib.Telnet('184.105.247.70', TELNET_PORT, TELNET_TIMEOUT)
output = remote_conn.read_until("sername:", TELNET_TIMEOUT)
remote_conn.write('pyclass\n')
output += remote_conn.read_until("ssword:", TELNET_TIMEOUT)
remote_conn.write(password + '\n')
remote_conn.write('show ip int brief\n')
time.sleep(1)
print remote_conn.read_very_eager()
```

Paramiko & Netmiko

Paramiko is the standard Python SSH library.

Netmiko is a multi-vendor networking library based on Paramiko.

Netmiko Vendors

Regularly tested
Arista vEOS
Cisco ASA
Cisco IOS
Cisco IOS-XR
Cisco SG300
HP Comware7
HP ProCurve
Juniper Junos
Linux

Limited testing
Avaya ERS
Avaya VSP
Brocade VDX
Brocade ICX/FastIron
Brocade MLX/NetIron
Cisco IOS-XE
Cisco NX-OS
Cisco WLC
Dell-Force10 DNOS9
Huawei
Palo Alto PAN-OS
Vyatta VyOS

Experimental
A10
Alcatel-Lucent SR-OS
Enterasys
Extreme
F5 LTM
Fortinet

Netmiko example

```
#!/usr/bin/env python
from getpass import getpass
from netmiko import ConnectHandler

if __name__ == "__main__":
    password = getpass("Enter router password: ")
    pynet_rtr1 = {
        'device_type': 'cisco_ios',
        'ip':   '184.105.247.70',
        'username': 'pyclass',
        'password': password
    }

    net_connect = ConnectHandler(**pynet_rtr1)
    print net_connect.find_prompt()
```

Key Netmiko Methods

.send_command()
.send_command_timing()

.send_config_set()
.send_config_from_file()

.commit()
.enable()
.disconnect()

.write_channel()
.read_channel()

FileTransfer Class

Netmiko Tools

git clone https://github.com/ktbyers/netmiko_tools

In your .bashrc file if you want to retain it
export PATH=~/netmiko_tools/netmiko_tools:$PATH

~/.netmiko.yml

netmiko-grep
netmiko-show
netmiko-cfg

Day3 Schedule

1. Serialization JSON and YAML
2. Concurrency
- Threads
- Processes
3. Cisco NX-API
4. Juniper, NETCONF, and PyEZ
- What is NETCONF
- PyEZ
- PyEZ get operations
- PyEZ config operations
5. Integrating to a Database
- Using Django's ORM
- Basic CRUD
- Primary and foreign Keys

Flickr: Pierre-Olivier Carles

APIs and Data Serialization

Why do we need data serialization?

Characteristics of JSON

Characteristics of YAML

Threads/Processes

- Concurrency
- Python and the GIL
- Example with threads
- Example with processes
- Example with a queue

NX-API

- NX-API Sandbox

- Interfacing to NX-API programmatically

- Using pynxos for NX-API

- Data Gathering with NX-API

- Configuration Automation with NX-API

Interfacing to NX-API programmatically

```
from pprint import pprint
from getpass import getpass
from pynxos.device import Device

nexus_ip = "1.1.1.1"
nxs_test = Device(host=nexus_ip, username="pyclass", password=getpass(),
                  transport='https', port=8443)

my_facts = nxs_test.facts
pprint(nxs_test.facts)
```

Data Gathering with NX-API

nxs_test.facts
nxs_test.show("show hostname")
nxs_test.show("show ip arp vrf management", raw_text=True)
nxs_test.show("show ip arp vrf management", raw_text=False)
nxs_test.show("show ip int brief vrf management")
nxs_test.show("show lldp neighbors")
nxs_test.running_config

Juniper, NETCONF, and PyEZ

- What is NETCONF?
- PyEZ
- PyEZ get operations
- PyEZ config operations

PyEZ simple connect / facts

```
from jnpr.junos import Device
from getpass import getpass
from pprint import pprint

juniper_srx = {
        "host": “184.105.247.76”,
        "user": "pyclass",
        "password": getpass(),
    }

a_device = Device(**juniper_srx)
a_device.open()
pprint(a_device.facts)
```

PyEZ table operations

```
from jnpr.junos import Device
from jnpr.junos.op.ethport import EthPortTable
from getpass import getpass

juniper_srx = {
    "host": "184.105.247.76",
    "user": "pyclass",
    "password": getpass(),
}
a_device = Device(**juniper_srx)
a_device.open()
eth_ports = EthPortTable(a_device)
eth_ports.get()
```

PyEZ config operations

```
#!/usr/bin/env python
from jnpr.junos import Device
from jnpr.junos.utils.config import Config
from getpass import getpass

juniper_srx = {
    "host": "184.105.247.76",
    "user": "pyclass",
    "password": getpass(),
}
a_device = Device(**juniper_srx)
a_device.open()
cfg = Config(a_device)
```

```
cfg.load("set system host-name test1", format="set", merge=True)
cfg.load(path="load_hostname.conf", format="text", merge=True)
cfg.load(path="load_hostname.xml", format="xml", merge=True)

cfg.diff()
cfg.rollback(0)
cfg.commit()
```

Integrating to a DB

- Django ORM
- Defining the DB
- Creating the DB
- Primary Keys, Foreign Keys
- CRUD Operations

Day4

Review
NAPALM
Comware API example
BigSwitch Rest API
Argparse
Subprocess

Ansible Overview
Ansible Config Templating
More Config Templating
Ansible 2.1 Modules
Ansible + Comware7

Flickr: au_tiger01

NAPALM

Purpose of NAPALM: create a standard set of operations across a range of platforms.

Operations fall into two general categories: Config Operations + Getter Operations.

NAPALM Vendors

eos
junos
iosxr
fortios
nxos
ios
pluribus
panos

NAPALM Getters

get_facts
get_environment
get_snmp_information
get_ntp_peers
get_ntp_stats
get_mac_address_table
get_arp_table
get_interfaces
get_interfaces_ip
get_lldp_neighbors
get_lldp_neighbors_detail
get_bgp_neighbors
get_bgp_neighbors_detail
get_bgp_config
get_route_to
get_probes_config
get_probes_results
get_users
get_optics

NAPALM Config Operations

device.load_merge_candidate()
device.load_replace_candidate()

device.compare_config()
device.discard_config()

device.commit_config()

device.rollback()

Comware NETCONF API

https://www.youtube.com/watch?v=_ADooXuSNpA

Comware Config

```
netconf ssh server port 830       # Default
netconf ssh server enable
```

```
from ncclient import manager
my_conn = manager.connect(host='1.1.1.1'', port=830, username='admin',
                          password=getpass(), device_params={'name': 'hpcomware'},
                          hostkey_verify=False, allow_agent=False, look_for_keys=False,
                          timeout=30)
```

Comware NETCONF API

```
my_conn.cli_display("display ip interface brief")
my_conn.client_capabilities

my_conn.get('subtree', my_xml)
my_conn.get()

my_conn.cli_config()
my_conn.edit_config(target="running", config=my_xml)
```

Comware NETCONF API

https://github.com/HPENetworking/pyhpecw7

```
from pyhpecw7.comware import HPCOM7
from pyhpecw7.features.vlan import Vlan
from pyhpecw7.features.interface import Interface

device = HPCOM7(host='1.1.1.1', username='admin',
password='admin')
print device.open()

vlan = Vlan(device, '1')
print vlan.get_config()
print vlan.get_vlan_list()

interface = Interface(device, 'FortyGigE1/0/50')
print interface.get_config()
```

```
cleanerase    # Factory default
config        # manage comware configs
file_copy
install_os
ipinterface
irf
```

BigSwitch REST API

https://github.com/bigswitch/sample-scripts/blob/master/bcf/bsnlabs/show_switch.py

```
# debug rest
# show version

import json
import requests
controller_url = "https://{}:8443".format(controller_ip)
path = "/api/v1/auth/login"
url = controller_url + path

data = '{"user": "admin", "password": "bsn123"}'
headers = {"content-type": "application/json"}
response = requests.request('POST', url, data=data, headers=headers, verify=False)
cookie = json.loads(response.content)['session_cookie']
session_cookie = 'session_cookie=%s' % cookie
```

Ansible Overview

- Ansible Introduction
- Ansible Terminology: Playbook, Play, Task
- Ansible Inventory
  - /etc/ansible/hosts
  - Overridden with -i option
  - host_vars
  - group_vars

Ansible Config Templating

```
---
- name: Configuration templating
  hosts: localhost
  tasks:
  - name: Generate configuration files
    template: src=access_switch.j2 dest=CFGS/{{ item.hostname }}.txt
    with_items:
      - {hostname: pynet-sw1, ip_addr: 10.10.10.20}
      - {hostname: pynet-sw2, ip_addr: 10.10.20.20}
```

Ansible Config Templating

access_switch.j2

```
!
!
service timestamps debug datetime msec localtime show-timezone
service timestamps log datetime msec localtime show-timezone
!
hostname {{ item.hostname }}
!
logging buffered 32000
no logging console
```

Ansible Config Templating

Jinja2

```
{% if item.field %}
ip access-list extended TEST-ACL
  permit ip host 1.1.1.1 any log
  permit ip host 2.2.2.2 any log
{% elif item.otherfield %}
ip access-list extended TEST-ACL
  permit ip host 3.3.3.3 any log
{% else %}
ip access-list extended TEST-ACL
  permit ip host 4.4.4.4 any log
{% endif %}
```

Jinja2

```
{% for port_number in range(1,25) %}
interface FastEthernet0/{{ port_number }}
 switchport access vlan {{ item.access_vlan }}
!
{% endfor %}
```

Ansible Config Templating Exercises

Jinja2 Includes

```
######### Jinja2 template #######
service timestamps debug datetime msec localtime
show-timezone
service timestamps log datetime msec localtime
show-timezone
!
hostname {{ item.hostname }}
!
!
{% include item.model_interfaces' %}
!
!
############ End Template ########
```

Jinja2 Macros

Creating functions inside of template.

```
{% macro intf_trunk(native_vlan=1, trunk_allowed_vlans=1) -%}
 switchport mode trunk
 switchport trunk native vlan {{ native_vlan }}
 switchport trunk allowed vlan {{ trunk_allowed_vlans }}
{%- endmacro %}
```

Note, the extra '-' in the macro syntax i.e. "-%}"

The macro name above is 'intf_trunk'
It takes two arguments "native_vlan" and "trunk_allowed_vlans". Each of these arguments has a default value.

```
interface FastEthernet0
 no ip address
 {{ intf_trunk(native_vlan=1, trunk_allowed_vlans="1,100") }}
```

Ansible 2.1 Networking Modules

platform_command
platform_config
platform_template

match: line/strict/exact

replace: line/block

parents
before

Comware7 Ansible Modules

https://github.com/HPENetworking/ansible-hpe-cw7

git clone https://github.com/HPENetworking/ansible-hpe-cw7

-M ~/ansible-hpe-cw7/library

Comware7 Ansible Modules

comware_clean_erase.py
comware_command.py
comware_facts.py
comware_file_copy.py
comware_install_config.py
comware_install_os.py
comware_interface.py
comware_ipinterface.py
comware_irf_members.py
comware_irf_ports.py
comware_l2vpn_global.py
comware_neighbors.py
comware_ping.py
comware_portchannel.py
comware_reboot.py
comware_save.py
comware_switchport.py
comware_vlan.py
comware_vrrp_global.py
comware_vrrp.py
comware_vxlan.py
comware_vxlan_svc_instance.py
comware_vxlan_tunnel.py

Day5 Schedule

Day5

1. Ansible Overview Review

2. More Playbook Topics
   a. tags
   b. when
   c. with_items
   d. limit
   e. register
   f. notify

3. NAPALM + Ansible
   - Configuration Merge Operations
   - Pushing Full Configuration Files

4. Using Cisco specific Ansible Modules

5. Using Juniper specific Ansible Modules

6. Ansible Roles

7. Dynamic Inventory

8. Writing custom Ansible Modules

9. ntc-ansible Modules

Ansible ‘with_items and ‘when’

```
with_items:
  - element1
  - element2
  - element3
  - element4
```

```
- name: Create Vlan 999
    ntc_config_command:
      host: "{{ host }}"
      username: "{{ username }}"
      password: "{{ password }}"
      platform: arista_eos
      commands:
        - 'vlan 999'
        - 'name BLUE'
    when: vlan_999 == false
```

tags, limit, register, and notify

ansible-playbook test_ans.yml --tags logging
ansible-playbook test_ans.yml --limit pynet-sw1

```
  register: result
- debug: var=result

  notify:
    - write mem

  handlers:
    - name: write mem
```

Cisco NXOS Ansible modules

nxos_aaa_server
nxos_aaa_server_host
nxos_acl
nxos_acl_interface
nxos_bgp
nxos_bgp_af
nxos_bgp_neighbor
nxos_bgp_neighbor_af
nxos_command
nxos_config
nxos_evpn_global
nxos_evpn_vni
nxos_facts
nxos_feature
nxos_file_copy
nxos_gir
nxos_gir_profile_management
nxos_hsrp
nxos_igmp
nxos_igmp_interface
nxos_igmp_snooping
nxos_install_os
nxos_interface
nxos_interface_ospf
nxos_ip_interface
nxos_mtu
nxos_ntp
nxos_ntp_auth
nxos_ntp_options
nxos_nxapi
nxos_ospf
nxos_ospf_vrf
nxos_overlay_global
nxos_pim
nxos_pim_interface
nxos_pim_rp_address
nxos_ping
nxos_portchannel
nxos_reboot
nxos_rollback
nxos_smu
nxos_snapshot
nxos_snmp_community
nxos_snmp_contact
nxos_snmp_host
nxos_snmp_location
nxos_snmp_traps
nxos_snmp_user
nxos_static_route
nxos_switchport
nxos_template (D)
nxos_udld
nxos_udld_interface
nxos_vlan
nxos_vpc
nxos_vpc_interface
nxos_vrf
nxos_vrf_af
nxos_vrf_interface
nxos_vrrp
nxos_vtp_domain
nxos_vtp_password
nxos_vtp_version
nxos_vxlan_vtep
nxos_vxlan_vtep_vni

Juniper Ansible Modules

https://github.com/Juniper/ansible-junos-stdlib
junos_cli
junos_commit
junos_get_config
junos_get_facts
junos_get_table
junos_install_config
junos_install_os
junos_jsnapy
junos_ping
junos_rollback
junos_rpc
junos_shutdown
junos_srx_cluster
junos_zeroize

Ansible roles / adding structure

```
- name: Build Python + Ansible (Group A)
  hosts: pylab9a
  sudo: yes

  roles:
    - server
    - applied_python
    - netmiko
    - arista
    - django
    - juniper
```

Directories

./roles/access_switch/files
./roles/access_switch/handlers
./roles/access_switch/tasks
./roles/access_switch/templates
./roles/access_switch/vars

Ansible Dynamic Inventory

$ ansible-playbook my_playbook.yml -i ./dyn_inv.py

The --list option must list out all of the groups and the associated hosts and group variables.
The --host option must either return an empty dictionary or a dictionary of variables relevant to that host.

https://github.com/ktbyers/pynet/blob/master/ansible/dyn_inv_v1.py

http://docs.ansible.com/ansible/dev_guide/developing_inventory.html

Ansible Dynamic Inventory

```
$   ./dyn_inv.py --list

{
    'arista': {
        'hosts': ['pynet-sw1', 'pynet-sw2', 'pynet-sw3', 'pynet-sw4'],
        'vars': {
            'ansible_connection': 'local',
            'eapi_hostname': '10.10.10.227',
            'eapi_password': 'password',
            'eapi_username': 'admin1'
        }
    },
    'local': {
        'hosts': ['localhost'],
        'vars': {'ansible_connection': 'local'}
    }
}
```

```
$ ./dyn_inv.py --host pynet-sw1
{"eapi_port": 8243}
```

Creating an Ansible Module

```
from ansible.module_utils.basic import AnsibleModule

def main():
    module = AnsibleModule(
        argument_spec = dict(
            state     = dict(default='present', choices=['present', 'absent']),
            name      = dict(required=True),
            enabled   = dict(required=True, type='bool'),
            something = dict(aliases=['whatever'])
        )
    )

module.exit_json(changed=True, something_else=12345)

module.fail_json(msg="Something fatal happened")
```

NTC-Ansible Modules

ntc_config_command.py
ntc_file_copy.py
ntc_get_facts.py
ntc_install_os.py
ntc_reboot.py
ntc_rollback.py
ntc_save_config.py
ntc_show_command.py

TextFSM Templates

cisco_ios_show_access-list.template
cisco_ios_show_aliases.template
cisco_ios_show_archive.template
cisco_ios_show_capability_feature_routing.template
cisco_ios_show_cdp_neighbors_detail.template
cisco_ios_show_cdp_neighbors.template
cisco_ios_show_clock.template
cisco_ios_show_interfaces_status.template
cisco_ios_show_interfaces.template
cisco_ios_show_interface_transceiver.template
cisco_ios_show_inventory.template
cisco_ios_show_ip_arp.template
cisco_ios_show_ip_bgp_summary.template
cisco_ios_show_ip_bgp.template
cisco_ios_show_ip_int_brief.template
cisco_ios_show_ip_ospf_neighbor.template
cisco_ios_show_ip_route.template
cisco_ios_show_lldp_neighbors.template
cisco_ios_show_mac-address-table.template
cisco_ios_show_processes_cpu.template
cisco_ios_show_snmp_community.template
cisco_ios_show_spanning-tree.template
cisco_ios_show_standby_brief.template
cisco_ios_show_version.template
cisco_ios_show_vlan.template
cisco_ios_show_vtp_status.template

Ansible 2.2 Networking Features

Template deprecated; *_template integrated into *_config
Added *_facts module
Expanded platform support.
Handle prompts in *_command modules.
Can set the output format in *_command modules (on some platforms)
Full config replace is also now supported (on some platforms)

The end...

Questions?

ktbyers@twb-tech.com
Twitter: @kirkbyers