N.A.P.A.L.M.
Network Automation and Programmability Abstraction Layer with Multivendor support

dbarroso@spotify.com
elisa@bigwaveit.org
N.A.P.A.L.M.

- Python library
- Open source
- Unified API for multiple vendors
- Methods to manipulate configs
- Methods to retrieve data
Supported Vendors

- Arista EOS
  Using pyEOS (you will need EOS version 4.14.6M or superior)

- Juniper JunOS
  Using junos-eznc

- Cisco IOS-XR
  Using pylOSXR

- Fortigate FortiOS
  Using pyFG
Supported Methods v0.1

- `load_replace_config`
  full configuration “override” (load override in junos terms)
- `load_merge_config`
  partial configuration merge
- `diff_config`
  return a diff of the “candidate” and the “running” config
- `discard`
  discard candidate
- `commit`
  commit changes
- `rollback`
  rollback last commit
Supported Methods v0.2 (beta)

- `get_facts`
  retrieve basic facts from the device
- `get_interfaces`
  get info per interface
- `get_bgp_neighbors`
  BGP session information
- `get_lldp_neighbors`
  details about LLDP neighbors
Ansible Modules

- Module to push configurations
  napalm_install_config
- Module to get facts
  napalm_get_facts
N.A.P.A.L.M. + ANSIBLE

{{ DEMO }}
Inventory File - We can group devices per type and/or location
---
- name: Configure spines
  hosts: spine*
  gather_facts: no
  connection: local
  roles:
    - baseconf
    - ipfabric

- name: Configure leaves
  hosts: leaf*
  gather_facts: no
  connection: local
  roles:
    - baseconf
    - ipfabric
    - access

- name: Configure additional network services in leaves
  hosts: leaf01.demo
  gather_facts: no
  connection: local
  roles:
    - baseconf
    - netserv

- name: Configure Firewall
  hosts: leaf01.demo
  gather_facts: no
  connection: local
  roles:
    - baseconf
    - netserv

Roles are “Services”
- name: Configure additional network services in leaves
  hosts: leaf01.demo
  gather_facts: no
  connection: local

  roles:
  - baseconf
  - netserv

- name: Configure Firewall
  hosts: fWk*
  gather_facts: no
  connection: local

  roles:
  - baseconf
  - svcinterconnect
  - firewall

- name: Configure Edge Routers
  hosts: rtr*
  gather_facts: no
  connection: local

  roles:
  - baseconf
  - svcinterconnect
  - peering

Roles are “Services” (cont’d)
Services are templated for every vendor (EOS example for ipfabric service)
Services are templated for every vendor (JunOS example for ipfabric service)
version 13.2X50-D19.2;

system {
    host-name leaf01;
    domain-name demo
}

protocols {
    mstp {
        configuration-name MSTP;
    }
    lldp {
        port-id-subtype interface-name;
        interface all;
    }
}

system {
    root-authentication {
        encrypted-password "$1$s7DeoJKZ/$ahBWi9d.00V4SJtcK7bst0"; ## SECRET-DATA
    }
    login {
        user admin {
            uid 2001;
            class super-user;
            authentication {
                encrypted-password "$1$s0d4yU7aX$qt83nSW/F55td0V0CZZeR0"; ## SECRET
            }
        }
        user dbarroso {
        }
    }
}
Some variables are defined at the DC1 level
Some variables are defined per type of devices/location (i.e. net_services @DC1)
Per host variables are define according to their services (vendor agnostic)
Per host variables are defined according to their services (vendor agnostic)
Per host variables are define according to their services (vendor agnostic)
Per host variables are defined according to their services (vendor agnostic)
NAPALM plugins are vendor agnostic (get_facts)
commit_changes = module.params['commit_changes']
diff_file = module.params['diff_file']

if commit_changes.__class__ is str:
    commit_changes = ast.literal_eval(commit_changes)

    network_driver = get_network_driver(dev_os)

    device = network_driver(hostname, username, password)
    device.open()
    device.load_replace_candidate(filename=config_file)

    diff = device.compare_config()
    changed = len(diff) > 0

    if diff_file is not None:
        save_to_file(diff, diff_file)

    if module.check_mode or not commit_changes:
        device.discard_config()
        module.exit_json(changed=changed, msg=diff)
    else:
        if changed:
            device.commit_config()
            module.exit_json(changed=changed, msg='lines changed: %s' % len(diff.split()))

logger.info('DEVICE=%s CHANGED=%s STATUS=%s' % (hostname, changed, 'OK'))
device.close()
---
- name: Getting facts with napalm
  hosts: all
  connection: local
  gather_facts: no

  tasks:
    - name: napalm_get_facts
      napalm_get_facts:
        hostname={{ inventory_hostname }}
        username=dbarroso
        dev_os={{ os }}
        password=p4ssw0rd
        when: commit_changes | match('0')

      tags:
        - base

Plays are also vendor agnostic
Building the IP Fabric and the Access layer

ansible-playbook -i network.hosts configure_network.yml --tags base,fabric,access,deploy --limit "dc1.spines,dc1.leaves" -e "commit_changes=0"
Connecting the network services

ansible-playbook -i network.hosts configure_network.yml --tags base,fabric,access,netserv,deploy --limit "dc1.net_services,dc1.spines,dc1.leaves" -e "commit_changes=0"
Deploying Network services

ansible-playbook -i network.hosts configure_network.yml --limit dc1.net_services -e "commit_changes=0"
ansible-playbook -i network.hosts configure_network.yml -e "commit_changes=0"
Summary

- Devices are broken down into different services
- Services are templated per vendor
- The combination of all services builds the full configuration of the devices
- The full configuration is pushed to the device, although only the delta is applied.
- Plays, playbooks and data is vendor agnostic
- N.A.P.A.L.M. allows you to have vendor agnostic workflows
Questions?

- David Barroso - dbarroso@spotify.com
- Elisa Jasinska - elisa@bigwaveit.org

Resources

- Mailing List - napalm-automation@googlegroups.com
- Ansible Demo - https://github.com/dbarrosop/ansible_demo