Table of Contents

Module 2

NAPALM INTRODUCTION

Module Contents

- What is NAPALM
- Getting Operational data using NAPALM
- Configuration Deployment using NAPALM

Module Contents

- What is NAPALM
- Getting Operational data using NAPALM
- Configuration Deployment using NAPALM

NAPALM Intro – What is NAPALM

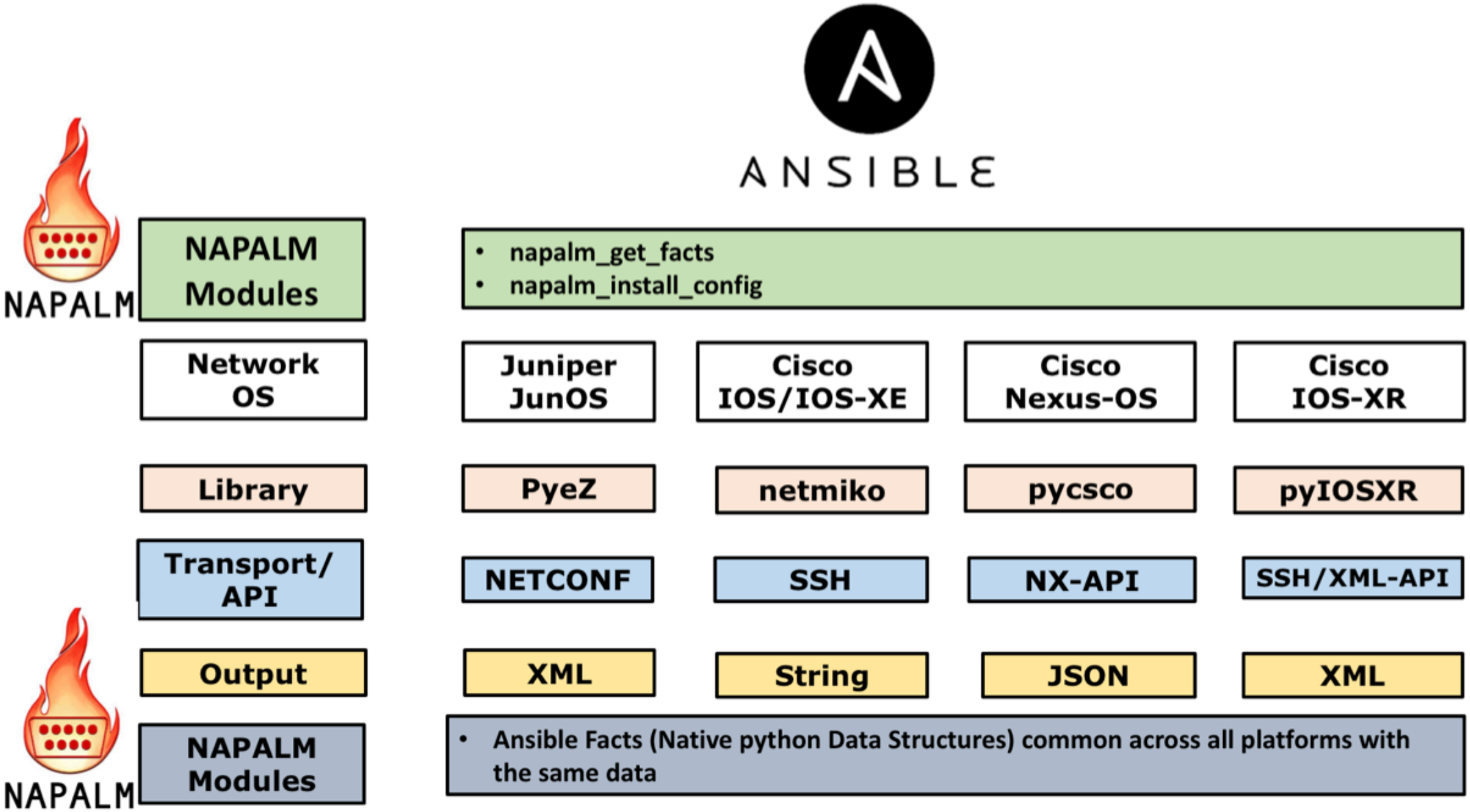
- NAPALM is an Open Source project for Network device managment.
- NAPALM (**N**etwork **A**utomation and **P**rogrammability **A**bstraction **L**ayer with **M**ultivendor support) tries to deliver a common API to manage different vendor OS.
- It is a **python** library that utilizes existing python libraries for each vendor (like Juniper Pyez) to communicate with each vendor Nodes.
- It abstracts these different libraries into a command API libraries which based on the device OS being managed triggers the correct API call to the target node.

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

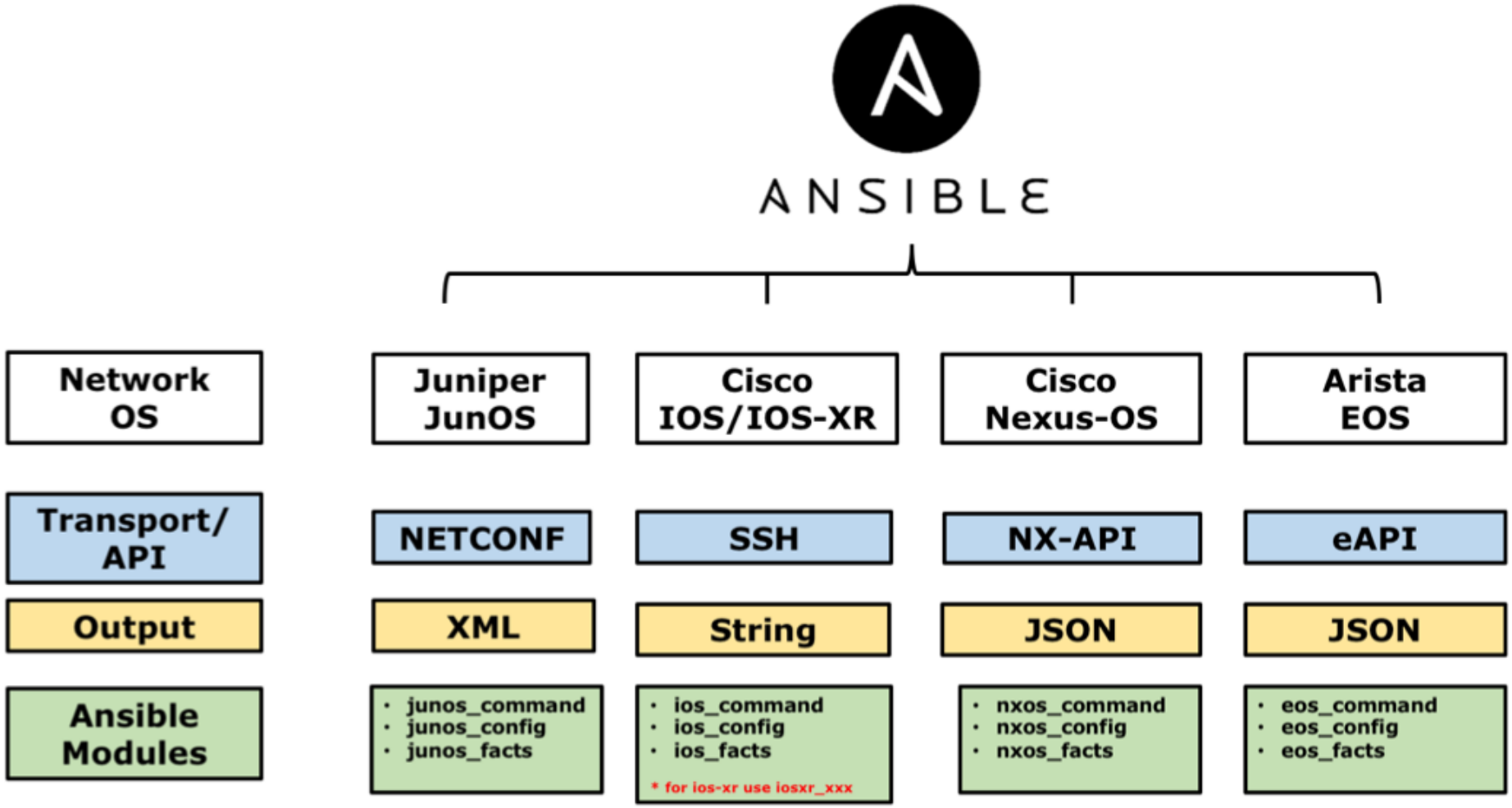
NAPALM Intro – NAPALM Device Support

- NAPALM Support Only the following devices
  - ✓Arista EOS
  - ✓Cisco IOS
  - ✓Cisco IOS-XR
  - ✓Cisco NX-OS
  - ✓Juniper JunOS

NAPALM Intro – NAPALM and Ansible



NAPALM Intro – Ansible with no NAPALM



NAPALM Intro – NAPALM Modules

- NAPALM provides two main functions to manage network devices.
  - ✓**napalm_get_facts**, this is used to get different output from the devices and return a common data structure.
  - ✓**napalm_install_config**, this load the configuration into the managed nodes.
- NAPALM emulate operation like **compare Config** and **commit Config** which is not available natively by some Networking OS like Cisco-IOS. Thus, it delivers a common API to interact with different vendors Network OS.
- Other Modules supported are in the below link

http://napalm.readthedocs.io/en/latest/integrations/ansible/modules/index.html

NAPALM Intro – Why NAPALM

NOT Consistent Data Structure


```
[
    {
        "rpc-reply": {
            "bgp-information": {
                "bgp-peer": [
                    {
                        "bgp-rib": [
                            {
                                "accepted-prefix-count": "1",
                                "active-prefix-count": "1",
                                "name": "inet.0",
                                "received-prefix-count": "1",
                                "suppressed-prefix-count": "0"
                            },
                            {
                                "accepted-prefix-count": "0",
                                "active-prefix-count": "0",
                                "name": "bgp.l3vpn.0",
                                "received-prefix-count": "0",
                                "suppressed-prefix-count": "0"
                            }
                        ],
                        "elapsed-time": "9:49:06",
                        "flap-count": "0",
                        "input-messages": "1298",
                        "output-messages": "1302",
                        "peer-address": "10.100.2.2",
                        "peer-as": "65000",
                        "peer-state": "Established",
                        "route-queue-count": "0"
                    }
                ],
                "down-peer-count": "0",
                "group-count": "1",
                "peer-count": "3"
            }
        }
    }
]
```


JunOS Output

```
BGP router identifier 10.100.4.4, local AS number 65000
BGP generic scan interval 60 secs
Non-stop routing is enabled
BGP table state: Active
Table ID: 0xe0000000   RD version: 4
BGP main routing table version 4
BGP NSR Initial initsync version 3 (Reached)
BGP NSR/ISSU Sync-Group versions 0/0
BGP scan interval 60 secs

BGP is operating in STANDALONE mode.


Process       RcvTblVer   bRIB/RIB   LabelVer  ImportVer  SendTblVer  StandbyVer
Speaker               4          4          4          4           4           0

Neighbor        Spk    AS MsgRcvd MsgSent   TblVer  InQ OutQ  Up/Down  St/PfxRcd
10.100.1.1        0 65000    1353    1226        4    0    0 10:11:35        4
```

IOS-XR Output

NAPALM Intro – Why NAPALM

Consistent Data Structure


```
"napalm_bgp_neighbors": {
    "global": {
        "peers": {
            "10.100.2.2": {
                "address_family": {
                    "ipv4": {
                        "accepted_prefixes": 1,
                        "received_prefixes": 1,
                        "sent_prefixes": 3
                    },
                    "ipv6": {
                        "accepted_prefixes": -1,
                        "received_prefixes": -1,
                        "sent_prefixes": -1
                    },
                    "l3vpn": {
                        "accepted_prefixes": 0,
                        "received_prefixes": 0,
                        "sent_prefixes": 0
                    }
                },
                "description": "",
                "is_enabled": true,
                "is_up": true,
                "local_as": 65000,
                "remote_as": 65000,
                "remote_id": "10.100.2.2",
                "uptime": 37317
            }
        },
        "router_id": "10.100.1.1"
    }
}
```


JunOS Output


```
"napalm_bgp_neighbors": {
    "global": {
        "peers": {
            "10.100.1.1": {
                "address_family": {
                    "ipv4": {
                        "accepted_prefixes": 4,
                        "received_prefixes": 4,
                        "sent_prefixes": 0
                    }
                },
                "description": "",
                "is_enabled": false,
                "is_up": true,
                "local_as": 65000,
                "remote_as": 65000,
                "remote_id": "10.100.1.1",
                "uptime": 37198
            }
        },
        "router_id": "10.100.4.4"
    }
}
```


IOS-XR Output

Module Contents

- What is NAPALM
- Getting Operational data using NAPALM
- Configuration Deployment using NAPALM

NAPALM Intro – Operational Data

- NAPALM use Ansible module **napalm_get_facts**, to execute some pre-defined show commands on the networking node.
- ONLY pre defined show commands can be executed on the remote nodes.
- The required show commands are passed as filter to the module.
- Below is sample of the filters
  - **get_bgp_neighbors**
  - **get_interfaces**
  - **get_lldp_neighbors**
- Full list of all supported filters

  https://napalm.readthedocs.io/en/latest/support/index.html

NAPALM Intro – Operational Data


```
"napalm_bgp_neighbors": {
    "global": {
        "peers": {
            "10.100.1.1": {
                "address_family": {
                    "ipv4": {
                        "accepted_prefixes": -1,
                        "received_prefixes": -1,
                        "sent_prefixes": -1
                    },
                    "ipv6": {
                        "accepted_prefixes": -1,
                        "received_prefixes": -1,
                        "sent_prefixes": -1
                    }
                },
                "description": "",
                "is_enabled": true,
                "is_up": false,
                "local_as": 100,
                "remote_as": 100,
                "remote_id": "",
                "uptime": 2944
            }
        },
```

NAPALM Intro – Operational Data Validation

```yaml
---

- name: GET NAPALM Facts
  hosts: all
  vars:
    BGP_setup:
      vMX1:
        peers: [10.100.2.2, 10.100.4.4, 10.100.5.5]
      vMX2:
        peers: [10.100.1.1]
      R4:
        peers: [10.100.1.1]
      XR5:
        peers: [10.100.1.1]
  tasks:
  - name: GET BGP Peers
    napalm_get_facts:
      hostname: "{{ ansible_host }}"
      username: "{{ ansible_user }}"
      dev_os: "{{dev_os}}"
      password: "{{ ansible_ssh_pass }}"
      optional_args:
        port: "{{ ansible_port }}"
      filter:
        - bgp_neighbors

  - name: BGP Validation
    assert:
        that: hostvars[inventory_hostname].napalm_bgp_neighbors.global.peers[item].is_up == true
        msg: "{{inventory_hostname}} BGP Peer {{item}} is down"
    with_items: "{{BGP_setup[inventory_hostname].peers}}"
    ignore_errors: yes
```

Module Contents

- What is NAPALM
- Getting Operational data using NAPALM
- Configuration Deployment using NAPALM

NAPALM Intro – Configuration

```
---
- name: push the configuration to the devices
  hosts: all
  vars:
    mdt_nodes: [vMX1, vMX2,XR5]
  tasks:
    - file: path=diff state=directory
      run_once: true
    - name: load the configuration to the devices
      napalm_install_config:
        hostname: "{{ ansible_host }}"
        username: "{{ ansible_user }}"
        dev_os: "{{dev_os}}"
        password: "{{ ansible_ssh_pass }}"
        optional_args:
          port: "{{ ansible_port }}"
        config_file: ./{{inventory_hostname}}-config.txt
        commit_changes: "{{commit| default('no')}}"
        diff_file: diff/{{inventory_hostname}}-diff.txt
      when: "inventory_hostname in mdt_nodes"
```