

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 pyIOSXR
- 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

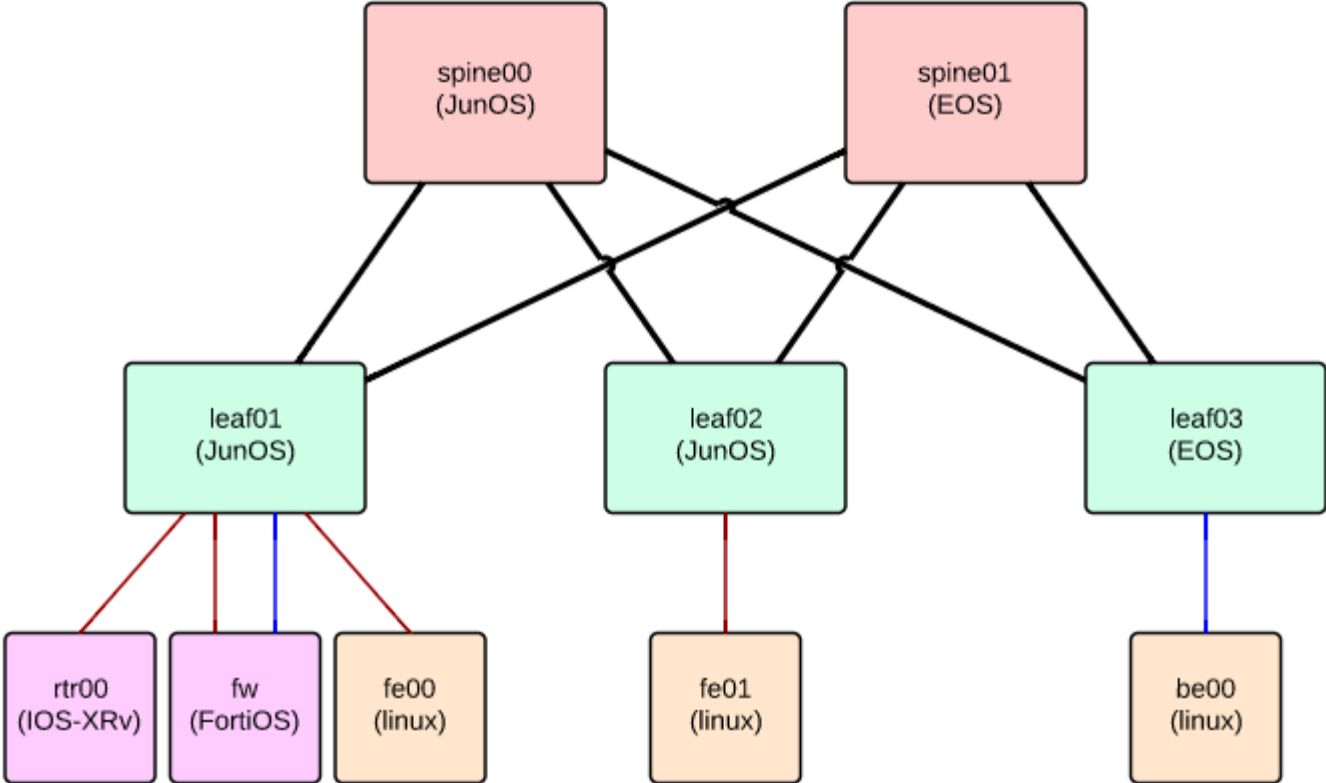
`napalm_install_config`

- Module to get facts

`napalm_get_facts`

N.A.P.A.L.M. + ANSIBLE

{ { DEMO } }



Network Diagram


```
1 [dc1:children]
2 dc1.spines
3 dc1.leaves
4 dc1.net_services
5
6 [dc1.spines]
7 spine00.demo
8 spine01.demo
9
10 [dc1.leaves]
11 leaf01.demo
12 leaf02.demo
13 leaf03.demo
14
15 [dc1.net_services]
16 fw.demo
17 rtr00.demo
18
```

Inventory File - We can group devices per type and/or location

The image shows a code editor window with a sidebar on the left and a main editor area on the right. The sidebar is titled "OPEN FILES" and "FOLDERS". Under "FOLDERS", there is a tree view showing a directory structure: "ansible_demo" containing ".idea", "compiled", "group_vars", "host_vars", "library", "tasks", and "roles". The "roles" folder is expanded, showing subfolders: "access", "baseconf", "firewall", "ipfabric", "netserv", "peering", and "svcinterconnect". Below the folders, several files are listed, including "assemble_push_conf.yml", "def_roles.yml", "get_facts.yml", ".gitignore", "configure_hosts.yml", "configure_network.yml", "LICENSE", "network.hosts", "README.md", "requirements.txt", and "servers.hosts". The main editor area shows the content of "def_roles.yml" with line numbers 1 through 30. The code is written in a green monospace font on a light background. It defines three tasks, each using a role. The first task is "Configure spines" for hosts "spine*", using roles "baseconf" and "ipfabric". The second task is "Configure leaves" for hosts "leaf*", using roles "baseconf", "ipfabric", and "access". The third task is "Configure additional network services in leaves" for hosts "leaf01.demo", using roles "baseconf" and "netserv". The fourth task is "Configure Firewall" (partially visible).

```
1 ---
2 - name: Configure spines
3   hosts: spine*
4   gather_facts: no
5   connection: local
6
7   roles:
8     - baseconf
9     - ipfabric
10
11 - name: Configure leaves
12   hosts: leaf*
13   gather_facts: no
14   connection: local
15
16   roles:
17     - baseconf
18     - ipfabric
19     - access
20
21 - name: Configure additional network services in leaves
22   hosts: leaf01.demo
23   gather_facts: no
24   connection: local
25
26   roles:
27     - baseconf
28     - netserv
29
30 - name: Configure Firewall
```

Roles are “Services”

The image shows a screenshot of a code editor window. The title bar indicates the file path is `~/Documents/workspace/spotify/ansible_demo/tasks/def_roles.yml` and the editor is titled `def_roles.yml`. On the left, a sidebar shows a file tree with folders like `ansible_demo`, `tasks`, and `roles`. The main editor area displays the following Ansible playbook content:

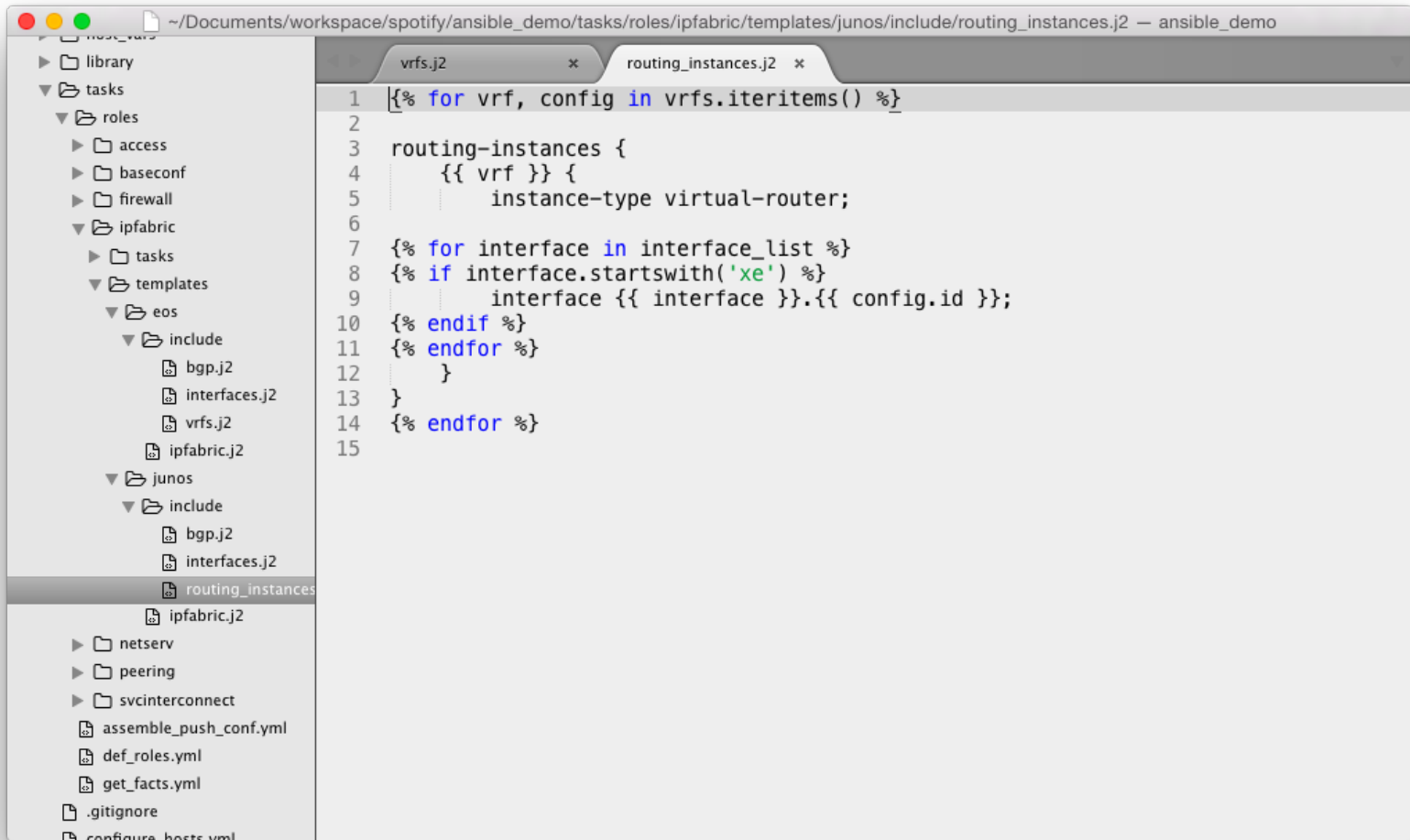
```
20
21 - name: Configure additional network services in leaves
22   hosts: leaf01.demo
23   gather_facts: no
24   connection: local
25
26   roles:
27     - baseconf
28     - netserv
29
30 - name: Configure Firewall
31   hosts: fw*
32   gather_facts: no
33   connection: local
34
35   roles:
36     - baseconf
37     - svcinterconnect
38     - firewall
39
40 - name: Configure Edge Routers
41   hosts: rtr*
42   gather_facts: no
43   connection: local
44
45   roles:
46     - baseconf
47     - svcinterconnect
48     - peering
49
```

Roles are “Services” (cont’d)

The image shows a code editor window with a file explorer on the left and a code editor on the right. The file explorer shows a directory structure for an Ansible role, with the file `vrfs.j2` selected under the path `roles/ipfabric/templates/eos/include`. The code editor displays the content of `vrfs.j2`, which is a Jinja2 template for configuring VRFs on EOS devices. The code is as follows:

```
1 {% for vrf, config in vrfs.iteritems() %}
2
3
4 vrf definition {{ vrf }}
5     rd 65000:{{config.id}}
6
7 ip routing vrf {{ vrf }}
8
9 {% endfor %}
10
```

Services are templated for every vendor (EOS example for ipfabric service)



The image shows a code editor window with a file explorer on the left and a code editor on the right. The file explorer shows a directory structure for an Ansible demo, with the file `routing_instances.j2` selected under the path `tasks/roles/ipfabric/templates/junos/include/`. The code editor displays the content of `routing_instances.j2`, which is an Ansible Jinja2 template for configuring routing instances on a JunOS device. The template uses a `for` loop to iterate over VRFs and another `for` loop to iterate over interfaces within each VRF. The code is as follows:

```
1 {% for vrf, config in vrfs.iteritems() %}
2
3 routing-instances {
4     {{ vrf }} {
5         instance-type virtual-router;
6
7     {% for interface in interface_list %}
8     {% if interface.startswith('xe') %}
9         interface {{ interface }}.{{ config.id }};
10    {% endif %}
11    {% endfor %}
12    }
13 }
14 {% endfor %}
15
```

Services are templated for every vendor (JunOS example for ipfabric service)

```
~/Documents/workspace/spotify/ansible_demo/compiled/leaf01.demo/running.conf — ansible_demo
OPEN FILES
x running.conf
FOLDERS
  ansible_demo
    .idea
    compiled
      fw.demo
      leaf01.demo
        00_baseconf.out
        01_ipfabric.out
        access.out
        netserv.out
        running.conf
      leaf02.demo
      leaf03.demo
      rtr00.demo
      spine00.demo
      spine01.demo
        .placeholder
    group_vars
    host_vars
    library
    tasks
    .gitignore
    configure_hosts.yml
    configure_network.yml
    LICENSE
    network.hosts
    README.md
running.conf
1 version 13.2X50-D19.2;
2
3 system {
4     host-name leaf01;
5     domain-name demo
6 }
7 protocols {
8     mstp {
9         configuration-name MSTP;
10    }
11    lldp {
12        port-id-subtype interface-name;
13        interface all;
14    }
15 }
16 system {
17     root-authentication {
18         encrypted-password "$1$7DeoJKZ/$ahBW19d.0oV4SJtcK7bst0"; ## SECRET-DATA
19     }
20     login {
21
22         user admin {
23             uid 2001;
24             class super-user;
25             authentication {
26                 encrypted-password "$1$0d4yU7aX$qt83nSW/F55td0V0CZZeR0"; ## SECRE
27             }
28         }
29     }
30     user dbarroso {
```

The combination of all the services is the complete “running” configuration

```
~/Documents/workspace/spotify/ansible_demo/group_vars/dc1 — ansible_demo
dc1
1 ---
2 users:
3     - name: admin
4       secret: $1$0d4yU7aX$qt83nSW/F55td0V0CZZeR0
5     - name: dbarroso
6       secret: $1$jazb$8SeTUjYz439INXKr5sF3N1
7       ssh_key: ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQDZnIq1KbGzaaSjQLwVmhYbZZ2Lns
8
9 management_gw: 10.48.68.1
10
11 primary_dns_server: '8.8.8.8'
12
13 primary_ntp_server: '81.234.134.160'
14 secondary_ntp_server: '130.236.254.17'
15
16 interconnect_range:
17     0: 192.169.0
18     1: 192.169.1
19
20 vrfs:
21     frontend:
22         id: 2
23         as_range: 4290020000
24         loopback_range: 10.255.254
25         access_range: 10.2
26     backend:
27         id: 3
28         as_range: 4290030000
29         loopback_range: 10.255.255
30         access_range: 10.3
```

Some variables are defined at the DC1 level

```
1 ---
2 policies:
3     - ['frontend', 'backend', 'PING']
4     - ['backend', 'frontend', 'PING']
5     - ['backend', 'frontend', 'HTTP']
6     - ['frontend', 'backend', 'MYSQL']
7
8 internet_peers:
9     # Possible options are:
10    # pass, disabled, primary_transit, secondary_transit, normal_peer, depreffered_p
11    transit:
12        172.20.23.4:
13            description: Expensive transit provider
14            as: 65301
15            policy: primary_transit
16        10.4.124.252:
17            description: Super expensive transit provider
18            as: 65102
19            policy: secondary_transit
20    peers:
21        172.20.255.1:
22            description: ACME Corp
23            as: 65401
24            policy: normal_peer
25        172.20.255.2:
26            description: We have eyeballs
27            as: 65410
28            policy: depreffered_peer
29        172.20.255.3:
30            description: University of Neverland
```

Some variables are defined per type of devices/location (i.e. net_services @DC1)


```
~/Documents/workspace/spotify/ansible_demo/host_vars/spine00.demo -- ansible_demo
spine00.demo x
1 id: 0
2 os: junos
3
4 management_ip: 10.48.71.100/22
5
```

OPEN FILES
x spine00.demo

FOLDERS
▼ ansible_demo
▶ .idea
▶ compiled
▶ group_vars
▼ host_vars
fw.demo
leaf01.demo
leaf02.demo
leaf03.demo
rtr00.demo
spine00.demo
spine01.demo
▶ library
▶ tasks
.gitignore
configure_hosts.yml
configure_network.yml
LICENSE
network.hosts
README.md
requirements.txt
servers.hosts

Per host variables are define according to their services (vendor agnostic)

The screenshot shows a code editor window with the following content:

```
~/Documents/workspace/spotify/ansible_demo/host_vars/spine01.demo -- ansible_demo
```

spine00.demo x spine01.demo x

```
1 id: 1
2 os: eos
3
4 management_ip: 10.48.71.21/22
5
```

The sidebar on the left shows the file structure:

- OPEN FILES
 - spine00.demo
- FOLDERS
 - ansible_demo
 - .idea
 - compiled
 - group_vars
 - host_vars
 - fw.demo
 - leaf01.demo
 - leaf02.demo
 - leaf03.demo
 - rtr00.demo
 - spine00.demo
 - spine01.demo
 - library
 - tasks
 - .gitignore
 - configure_hosts.yml
 - configure_network.yml
 - LICENSE
 - network.hosts
 - README.md
 - requirements.txt
 - servers.hosts

Per host variables are define according to their services (vendor agnostic)

The screenshot shows a code editor window with the following content:

```
~/Documents/workspace/spotify/ansible_demo/host_vars/leaf01.demo — ansible_demo  
leaf01.demo * leaf03.demo *  
1 id: 1  
2 os: junos  
3  
4 management_ip: 10.48.71.102/22  
5  
6 vlan_mapping:  
7   frontend:  
8     - 0  
9     - 46  
10  backend:  
11    - 47  
12  
13 netserv:  
14   frontend:  
15     10.2.1.254:  
16       as: 4290029999  
17     10.2.1.253:  
18       as: 4290029998  
19   backend:  
20     10.3.1.254:  
21       as: 4290029999  
22
```

The left sidebar shows the file explorer with the following structure:

- OPEN FILES
 - leaf01.demo
 - leaf03.demo
- FOLDERS
 - ansible_demo
 - .idea
 - compiled
 - group_vars
 - host_vars
 - fw.demo
 - leaf01.demo
 - leaf02.demo
 - leaf03.demo
 - rtr00.demo
 - spine00.demo
 - spine01.demo
 - library
 - tasks
- FILES
 - .gitignore
 - configure_hosts.yml
 - configure_network.yml
 - LICENSE
 - network.hosts
 - README.md
 - requirements.txt
 - servers.hosts

Per host variables are define according to their services (vendor agnostic)

The image shows a code editor window with a sidebar on the left and a main editing area on the right. The sidebar is titled "OPEN FILES" and "FOLDERS". Under "OPEN FILES", there are two files: leaf01.demo and leaf03.demo. Under "FOLDERS", there is a folder named "ansible_demo" which contains several sub-folders and files. The main editing area shows the content of leaf03.demo, which is an Ansible host variable file. The content is as follows:

```
1 id: 3
2 os: eos
3
4 management_ip: 10.48.71.104/22
5
6 vlan_mapping:
7     frontend:
8         - 1
9     backend:
10        - 2
11        - 5
12
```

Per host variables are define according to their services (vendor agnostic)

The image shows a code editor window with a sidebar on the left and a main code area on the right. The sidebar, titled "OPEN FILES", shows a file tree for a project named "ansible_demo". The "library" folder is expanded, showing "address_book", "napalm_get_facts" (selected), "napalm_install_config", "tasks", ".gitignore", "configure_hosts.yml", "configure_network.yml", "LICENSE", "network.hosts", "README.md", "requirements.txt", and "servers.hosts". The main code area shows the Python code for the "napalm_get_facts" module. The code is as follows:

```
38     username=dict(required=True),
39     password=dict(required=True),
40     dev_os=dict(required=True),
41 ),
42     supports_check_mode=True
43 )
44
45 hostname = module.params['hostname']
46 username = module.params['username']
47 dev_os = module.params['dev_os']
48 password = module.params['password']
49
50
51 ... network_driver = get_network_driver(dev_os)
52
53 ... device = network_driver(hostname, username, password)
54 ... device.open()
55
56 ... facts = device.get_facts()
57 ... facts['interface_details'] = device.get_interfaces()
58
59 ... device.close()
60
61 module.exit_json(ansible_facts=facts)
62
63 from ansible.module_utils.basic import *
64 from napalm import get_network_driver
65
66 main()
67
```

NAPALM plugins are vendor agnostic (get_facts)

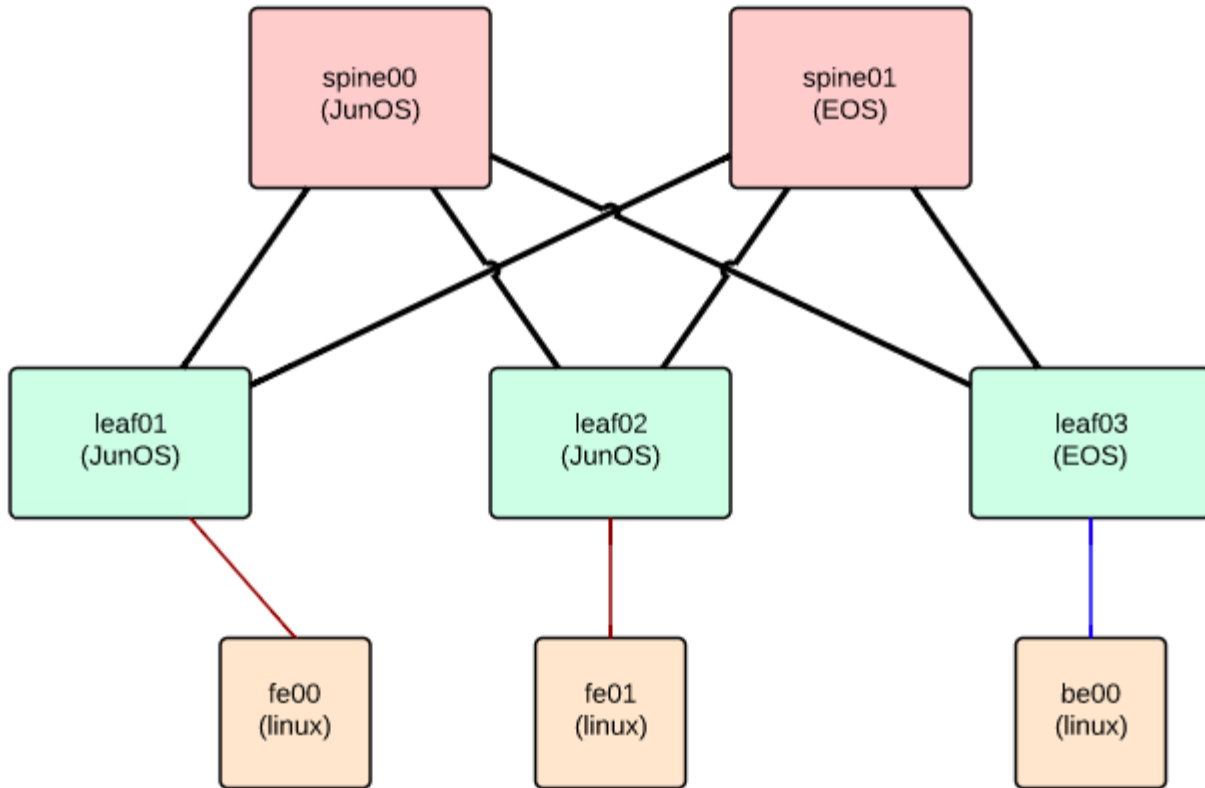
```
~/Documents/workspace/spotify/ansible_demo/library/napalm_install_config -- ansible_demo
napalm_get_facts x napalm_install_config x
OPEN FILES
x napalm_get_facts
x napalm_install_config
FOLDERS
▼ ansible_demo
  ► .idea
  ► compiled
  ► group_vars
  ► host_vars
  ▼ library
    address_book
    napalm_get_facts
    napalm_install_config
  ► tasks
  .gitignore
  configure_hosts.yml
  configure_network.yml
  LICENSE
  network.hosts
  README.md
  requirements.txt
  servers.hosts
119     commit_changes = module.params['commit_changes']
120     diff_file = module.params['diff_file']
121
122     if commit_changes.__class__ is str:
123         commit_changes = ast.literal_eval(commit_changes)
124
125     ...network_driver = get_network_driver(dev_os)
126
127     ...device = network_driver(hostname, username, password)
128     ...device.open()
129     ...device.load_replace_candidate(filename=config_file)
130
131     ...diff = device.compare_config()
132     ...changed = len(diff) > 0
133
134     ...if diff_file is not None:
135         ...save_to_file(diff, diff_file)
136
137     ...if module.check_mode or not commit_changes:
138         ...device.discard_config()
139         ...module.exit_json(changed=changed, msg=diff)
140     ...else:
141         ...if changed:
142             ...device.commit_config()
143             ...module.exit_json(changed=changed, msg='lines changed: %s' % len(diff.splitlines()))
144
145     logger.info('DEVICE=%s CHANGED=%s STATUS=%s' % (hostname, changed, 'OK'))
146
147     device.close()
148
```

NAPALM plugins are vendor agnostic (napalm_install_config)

The image shows a code editor window with a file explorer on the left and a code editor on the right. The file explorer shows a project structure for 'ansible_demo' with various folders and files. The code editor displays an Ansible playbook named 'get_facts.yml' with the following content:

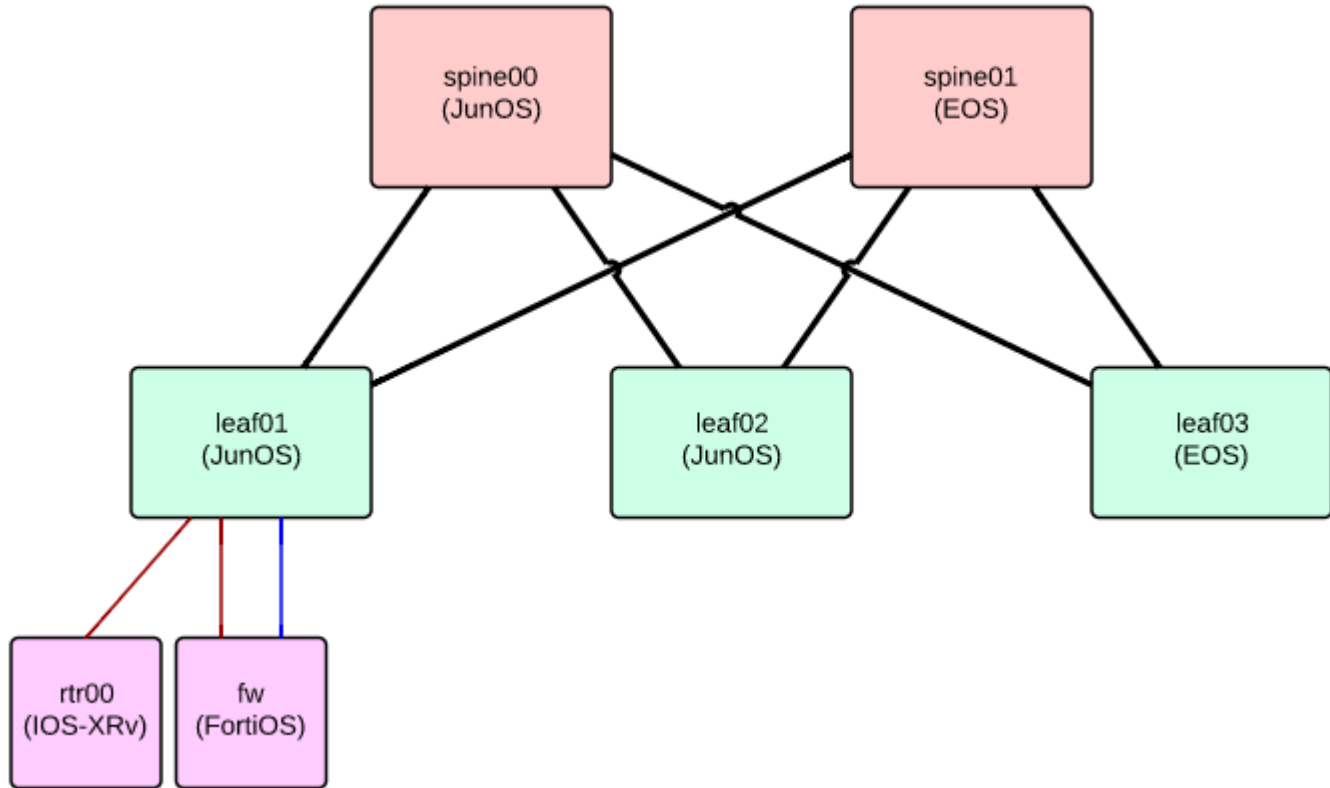
```
1 ---
2 - name: Getting facts with napalm
3   hosts: all
4   connection: local
5   gather_facts: no
6
7   tasks:
8     - name: napalm_get_facts
9       napalm_get_facts:
10        hostname={{ inventory_hostname }}
11        username=dbarroso
12        dev_os={{os}}
13        password=p4ssw0rd
14        when: commit_changes | match('0')
15
16      tags:
17        - base
18
```

Plays are also vendor agnostic



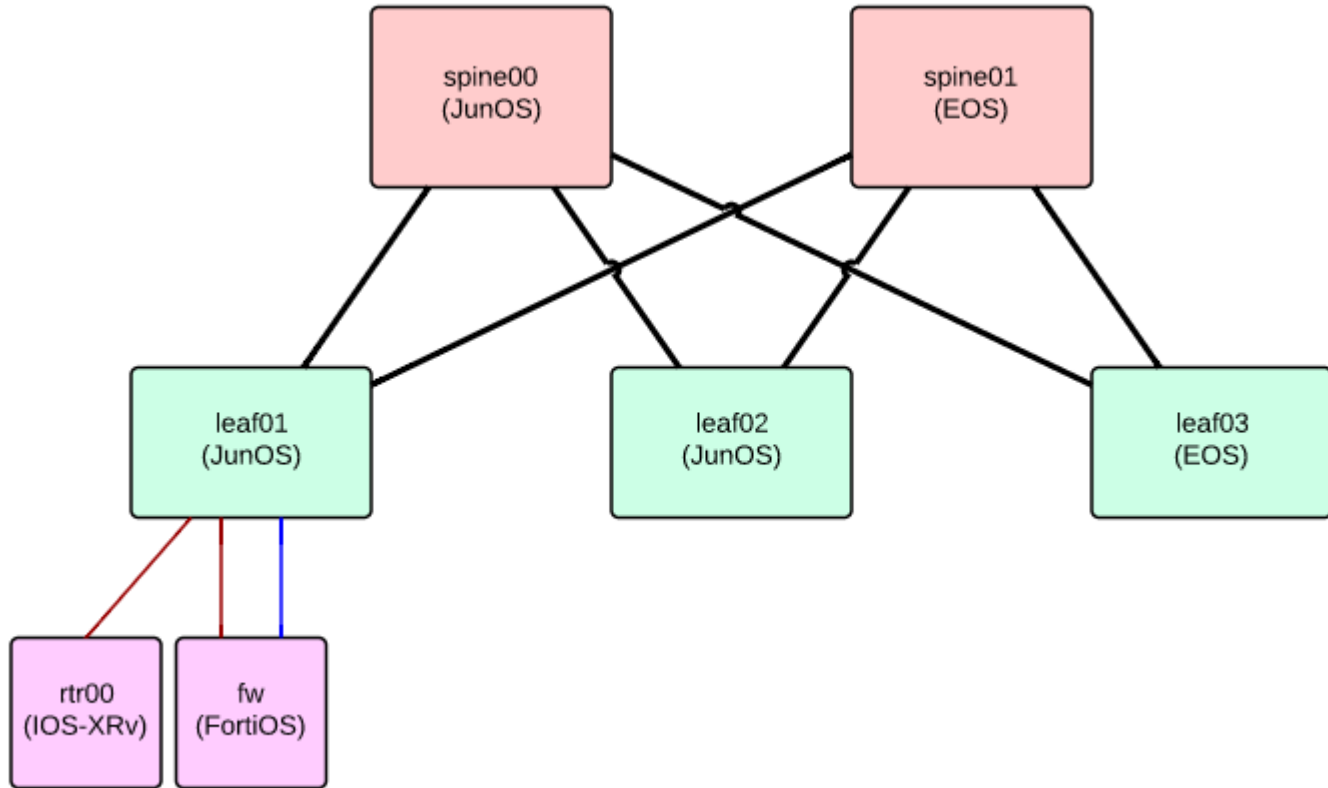
Building the IP Fabric and the Access layer

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

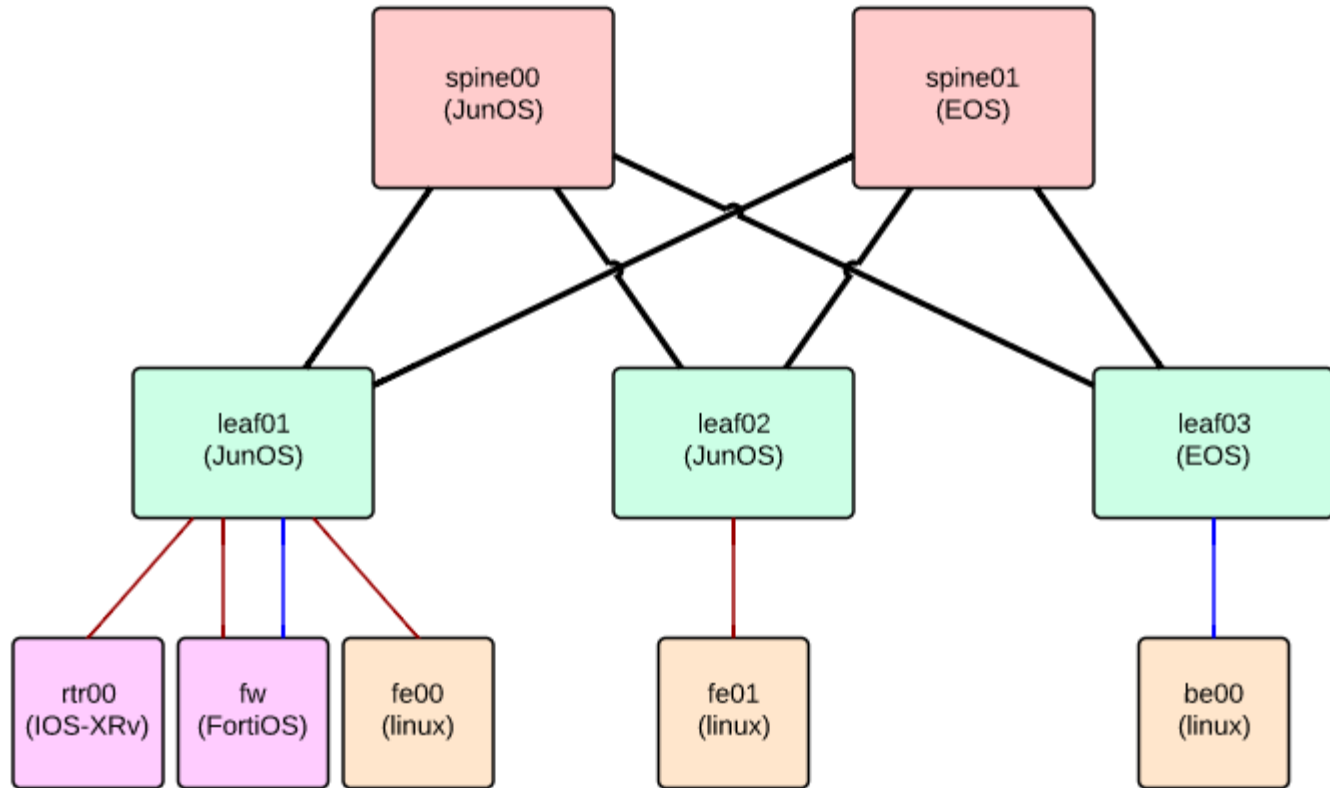
Connecting the network services

```
ansible-playbook -i network.hosts configure_network.yml --tags base,fabric,access,net serv,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"
```



Unified deployment

```
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

- N.A.P.A.L.M. - <https://github.com/spotify/napalm>
- Mailing List - napalm-automation@googlegroups.com
- Ansible Demo - https://github.com/dbarrosop/ansible_demo