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

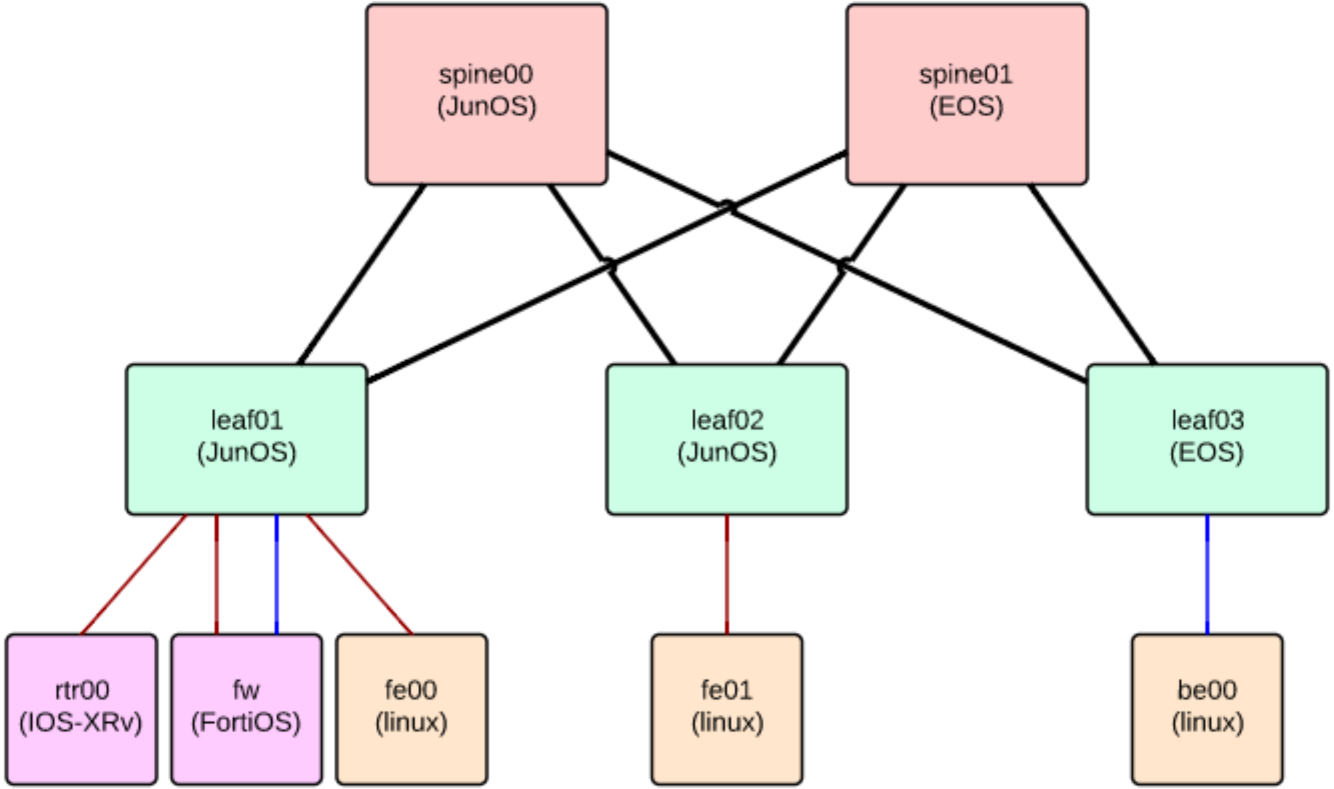
  ```
  napalm_install_config
  ```

- Module to get facts

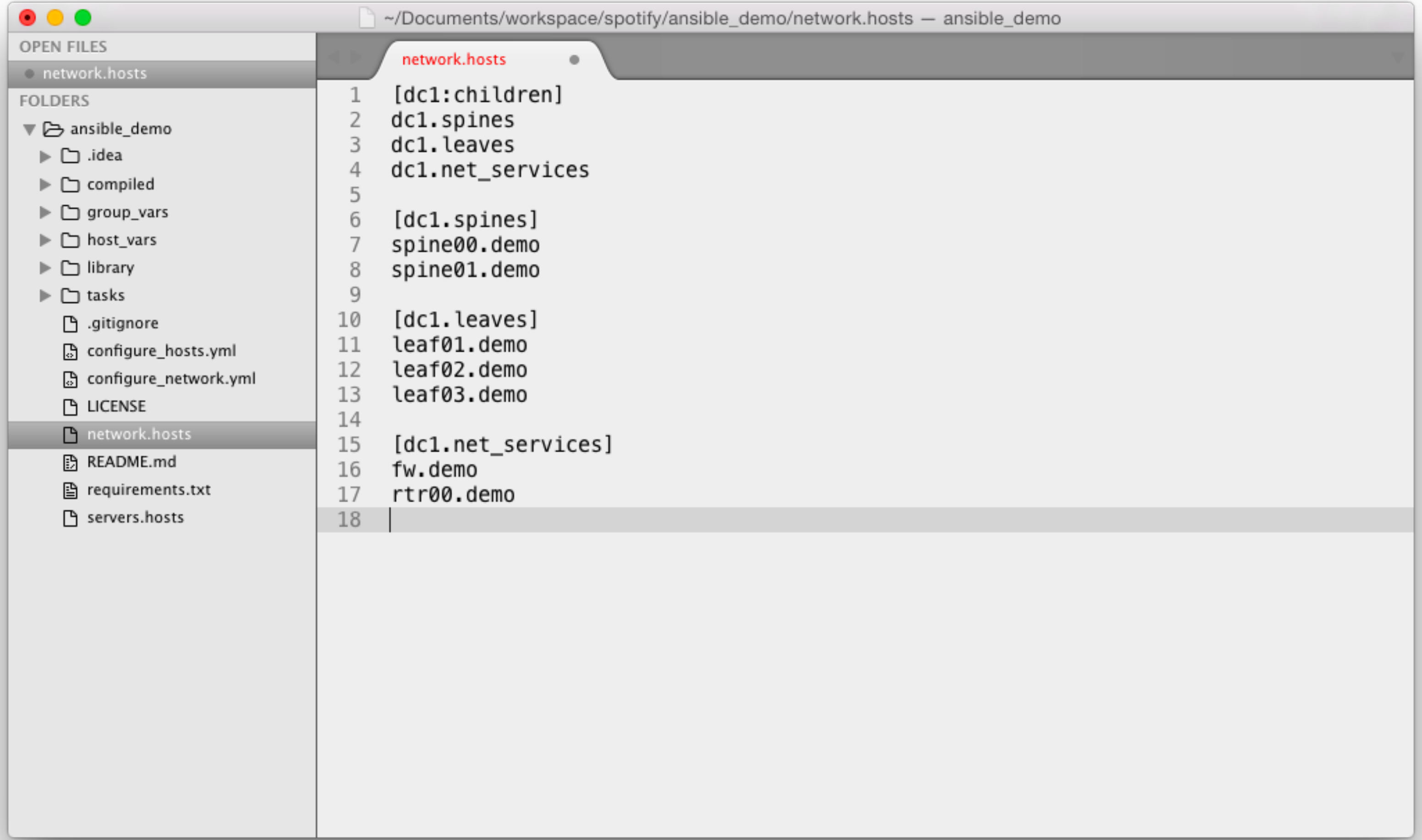
  ```
  napalm_get_facts
  ```

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

{{ DEMO }}

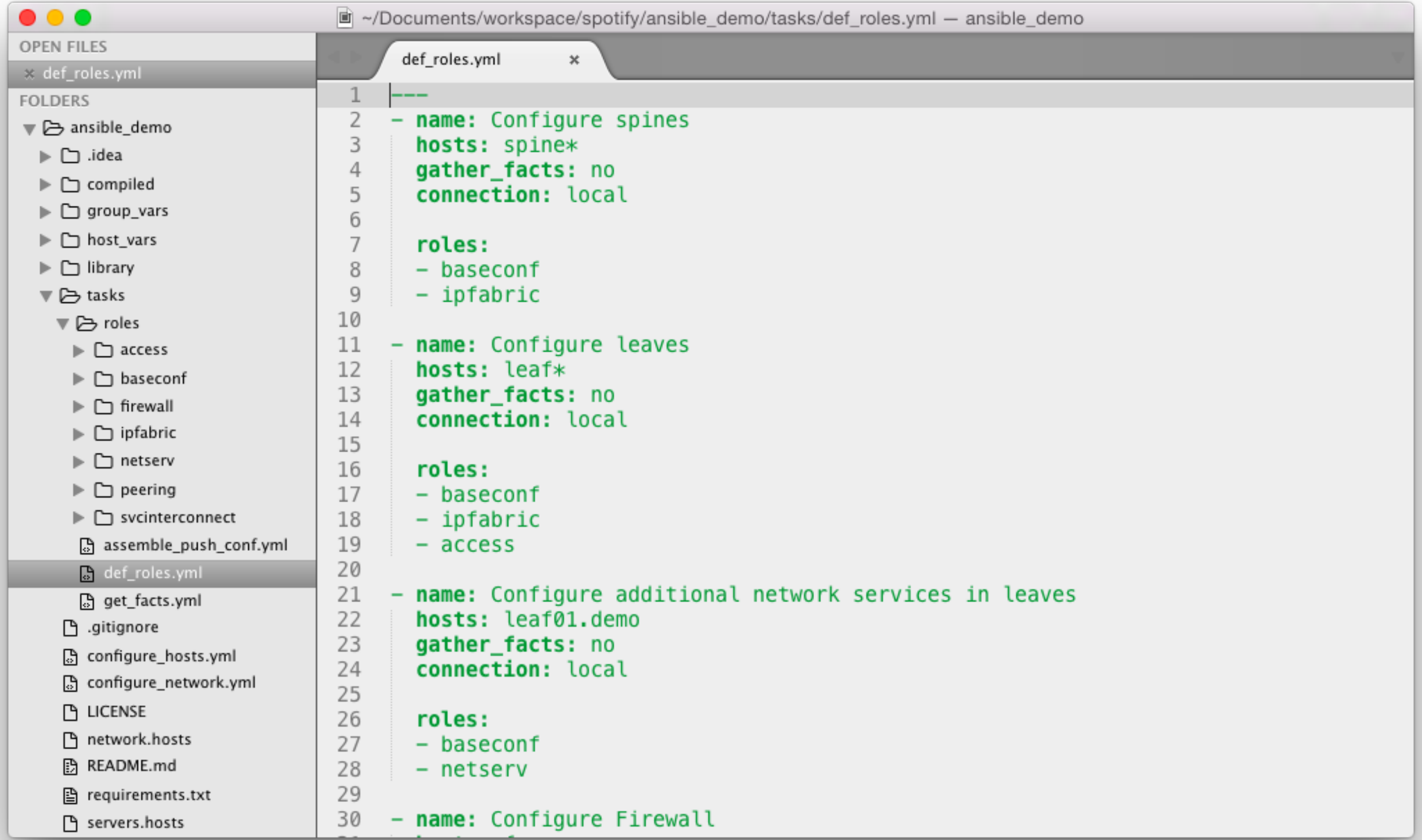


Network Diagram



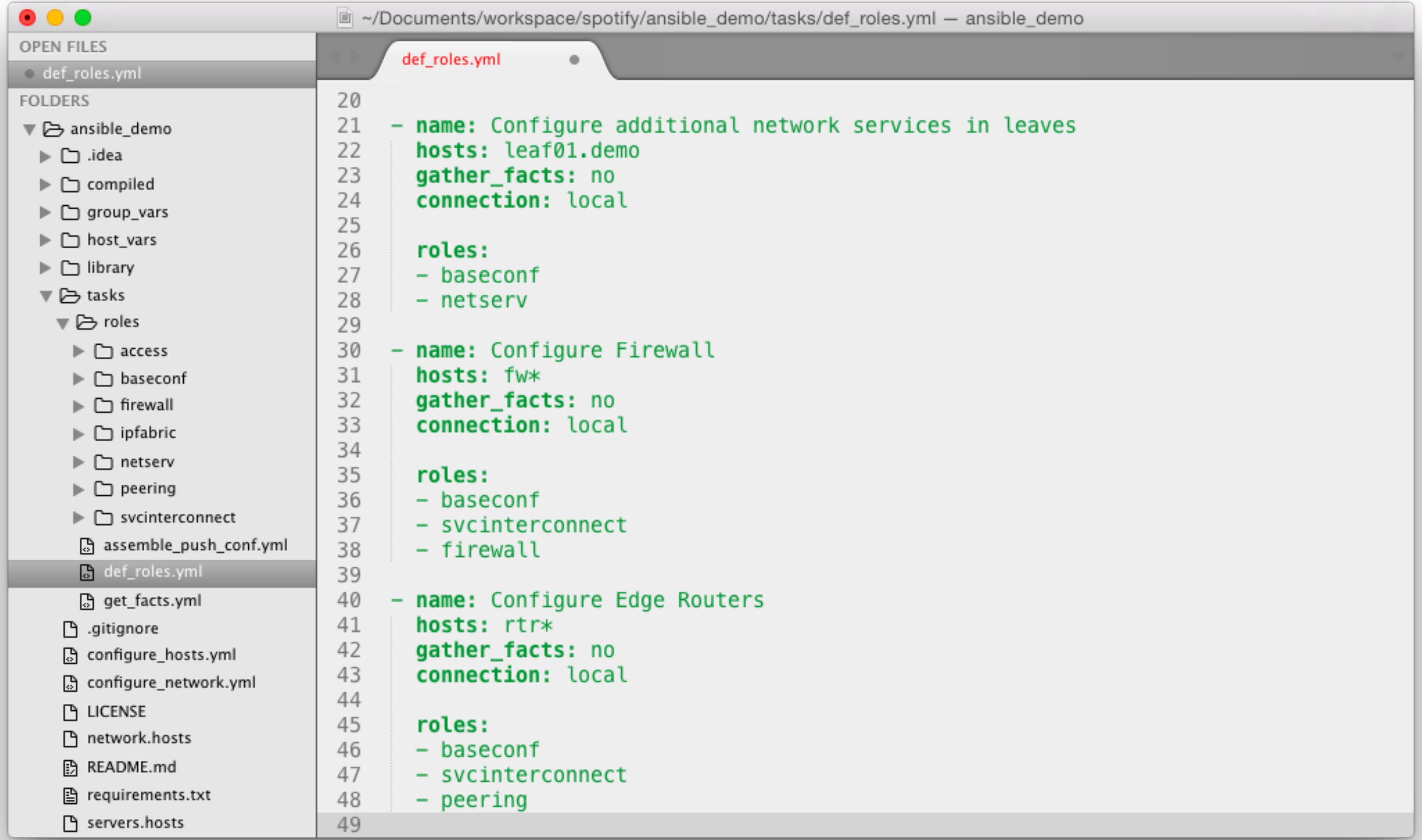
```
[dc1:children]
dc1.spines
dc1.leaves
dc1.net_services

[dc1.spines]
spine00.demo
spine01.demo

[dc1.leaves]
leaf01.demo
leaf02.demo
leaf03.demo

[dc1.net_services]
fw.demo
rtr00.demo

```

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



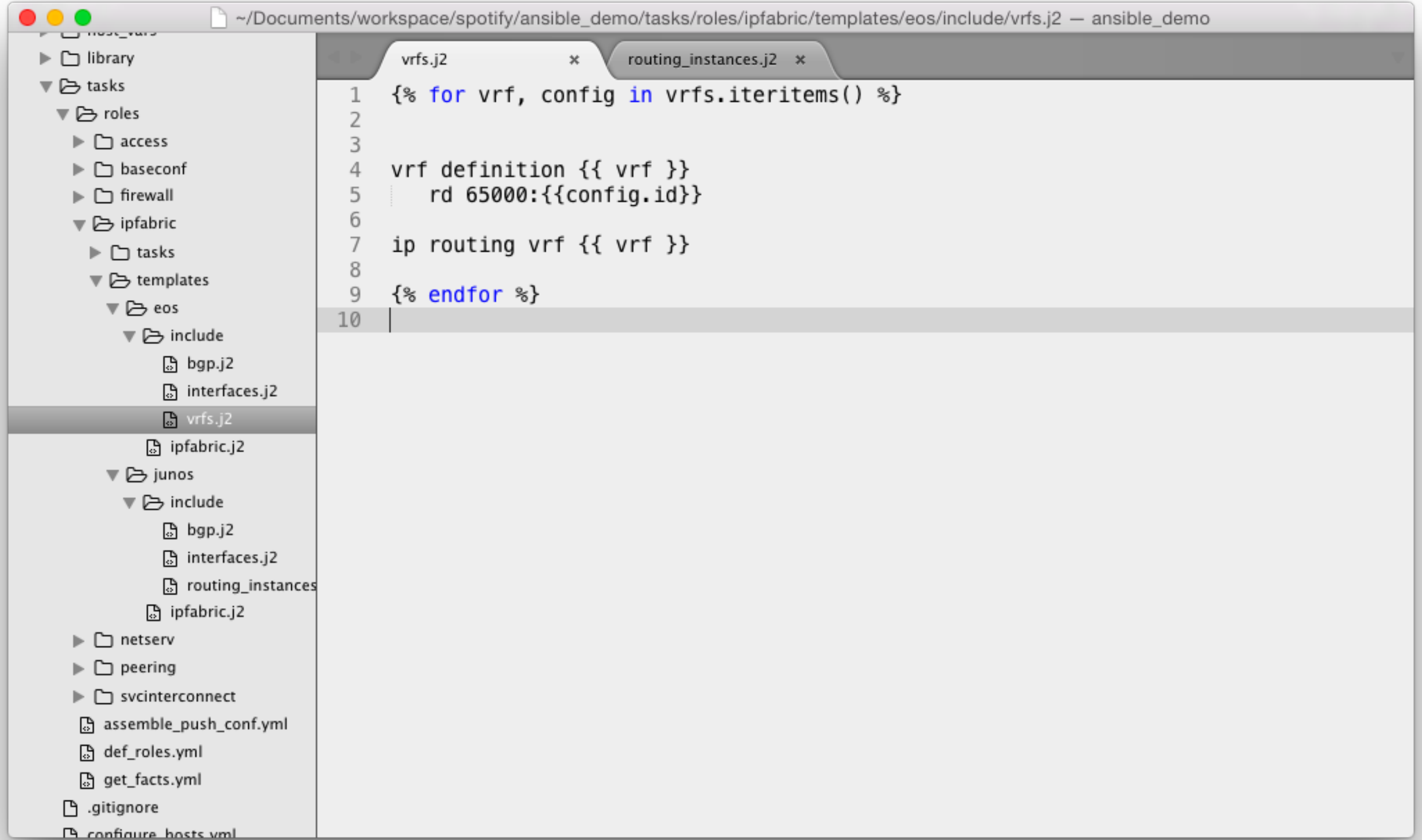
```yaml
---
- name: Configure spines
  hosts: spine*
  gather_facts: no
  connection: local

  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
```

Roles are “Services”



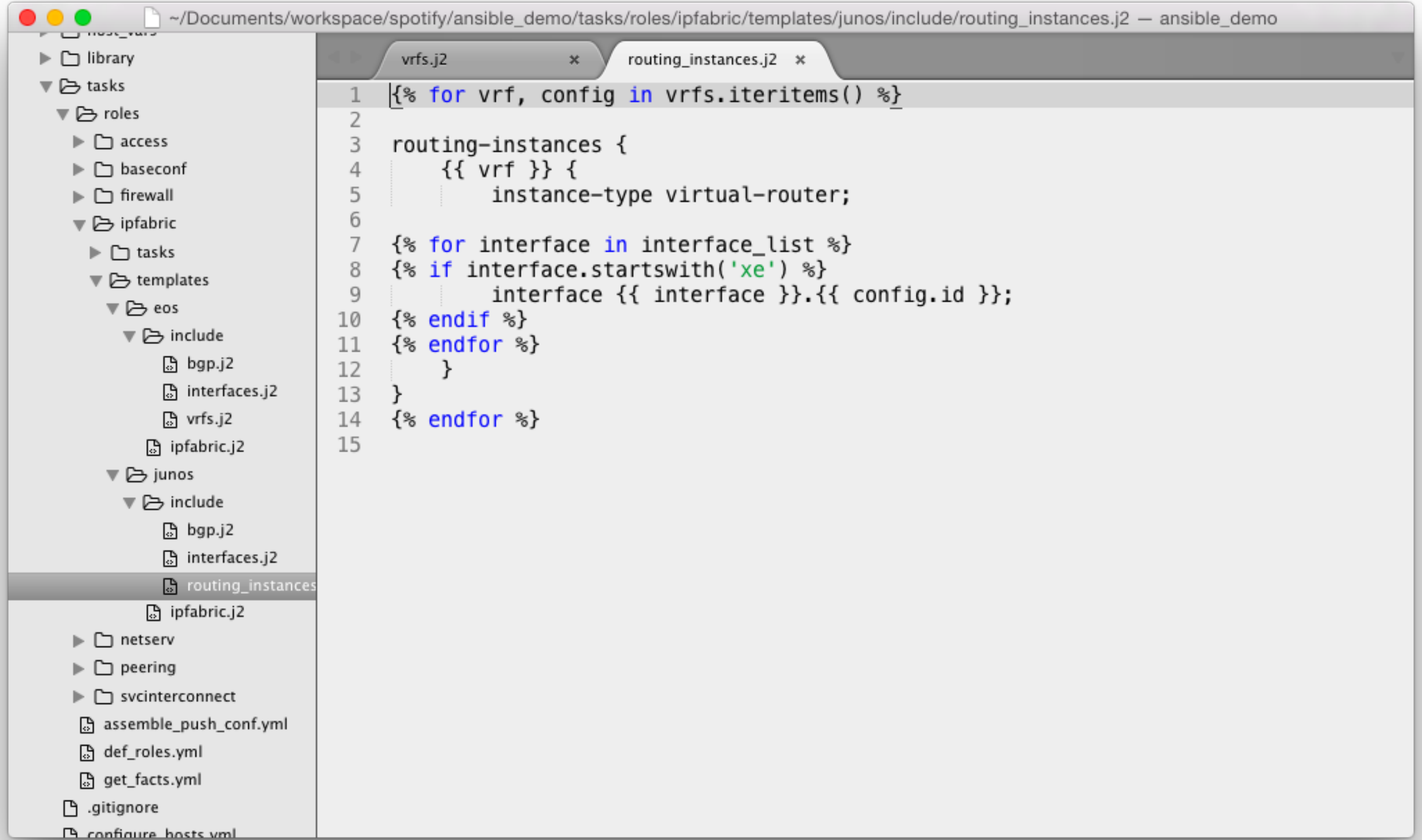
```yaml

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

  roles:
  - baseconf
  - netserv

- name: Configure Firewall
  hosts: fw*
  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)



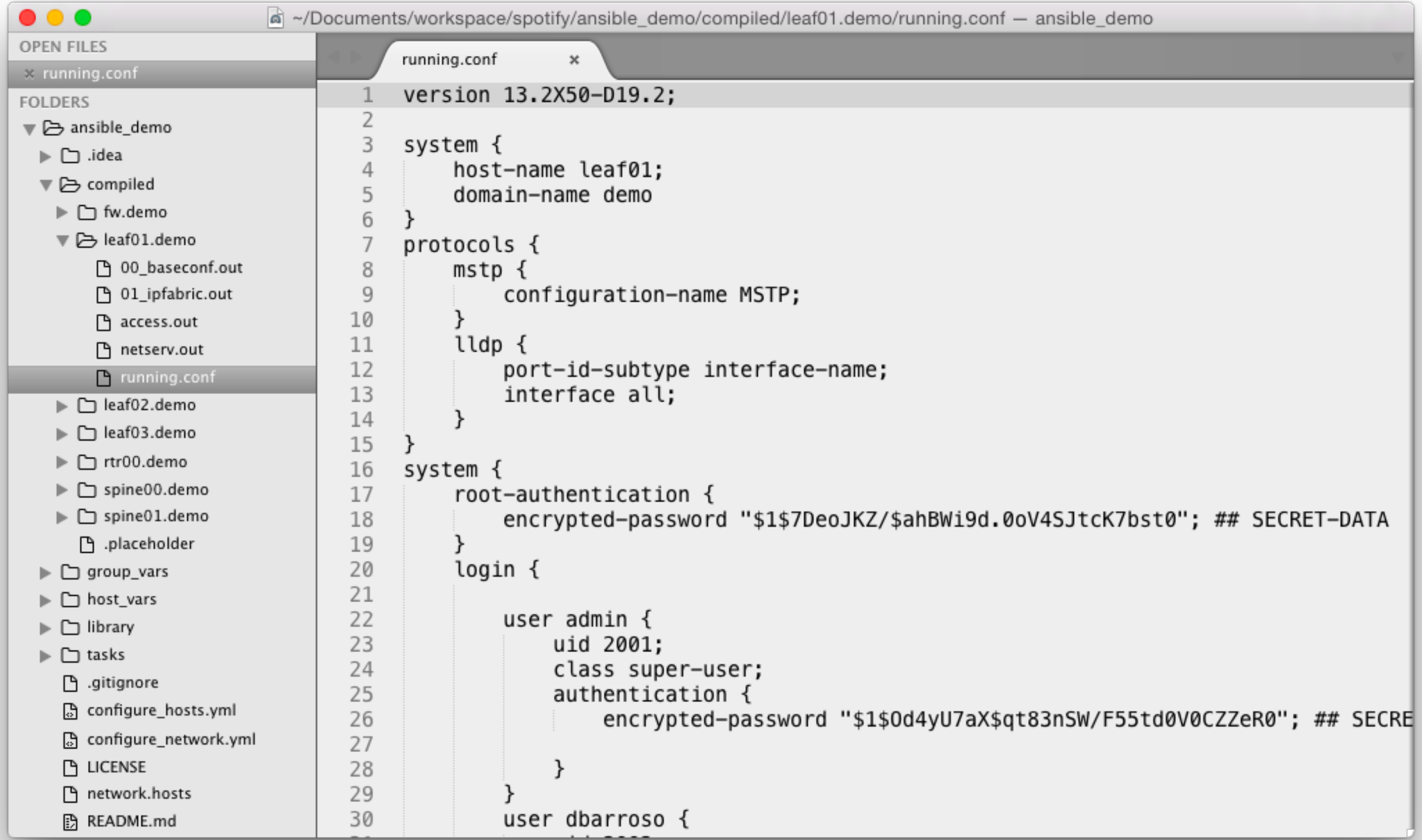
```
{% for vrf, config in vrfs.iteritems() %}


vrf definition {{ vrf }}
   rd 65000:{{config.id}}

ip routing vrf {{ vrf }}

{% endfor %}

```

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



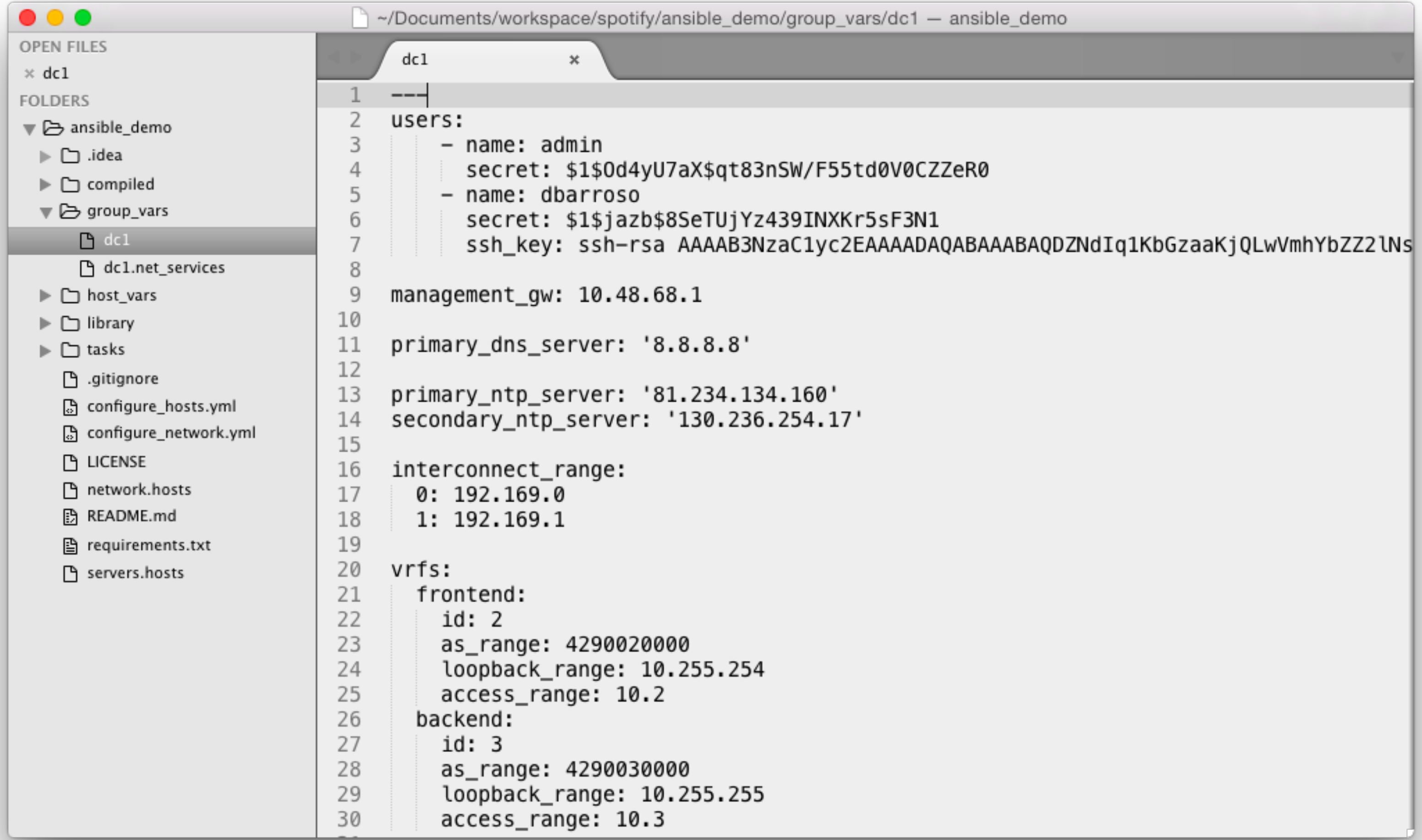
```
{% for vrf, config in vrfs.iteritems() %}

routing-instances {
    {{ vrf }} {
        instance-type virtual-router;

{% for interface in interface_list %}
{% if interface.startswith('xe') %}
        interface {{ interface }}.{{ config.id }};
{% endif %}
{% endfor %}
    }
}
{% endfor %}

```

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



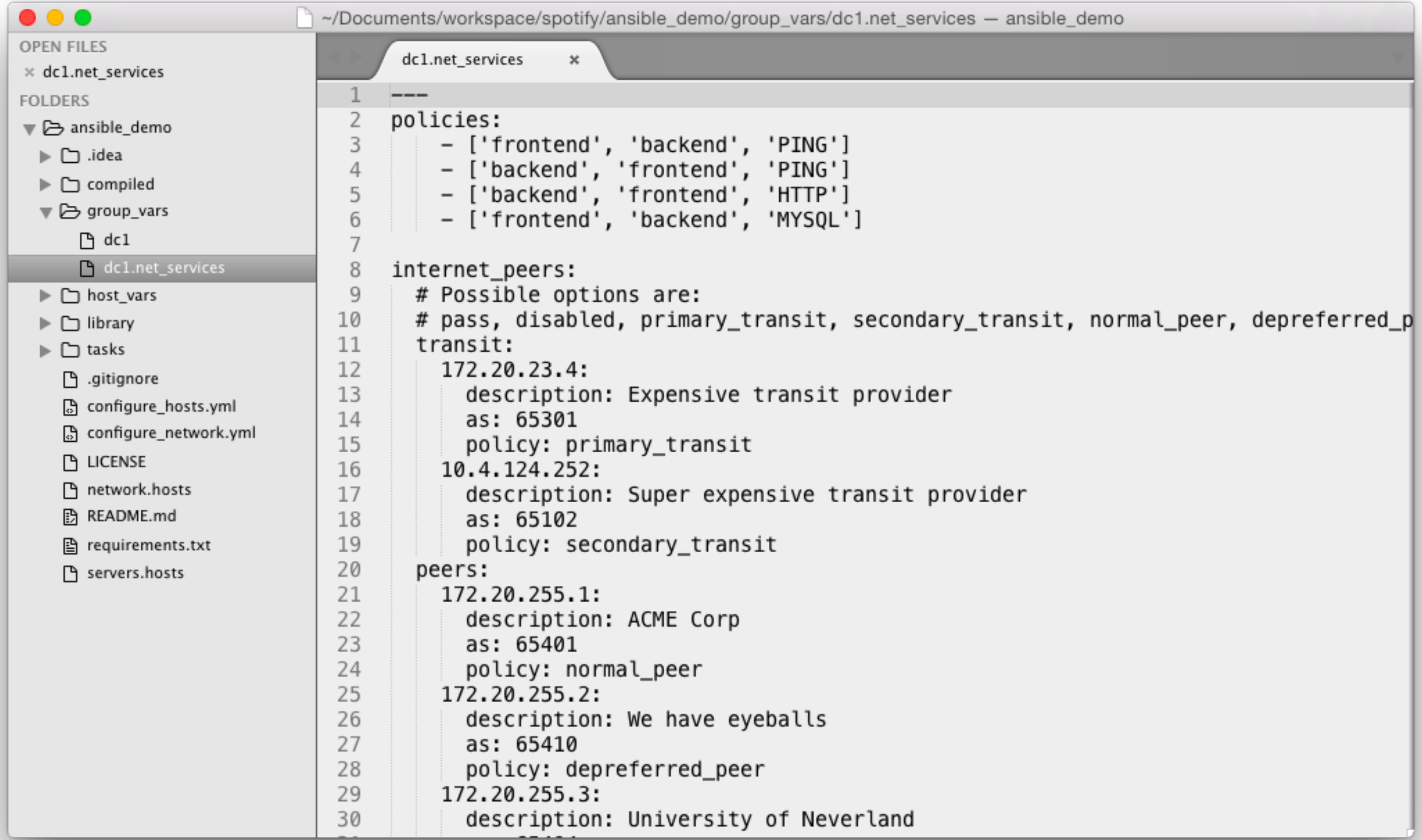
```
version 13.2X50-D19.2;

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$7DeoJKZ/$ahBWi9d.0oV4SJtcK7bst0"; ## SECRET-DATA
    }
    login {

        user admin {
            uid 2001;
            class super-user;
            authentication {
                encrypted-password "$1$Od4yU7aX$qt83nSW/F55td0V0CZZeR0"; ## SECRE

            }
        }
        user dbarroso {
```

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



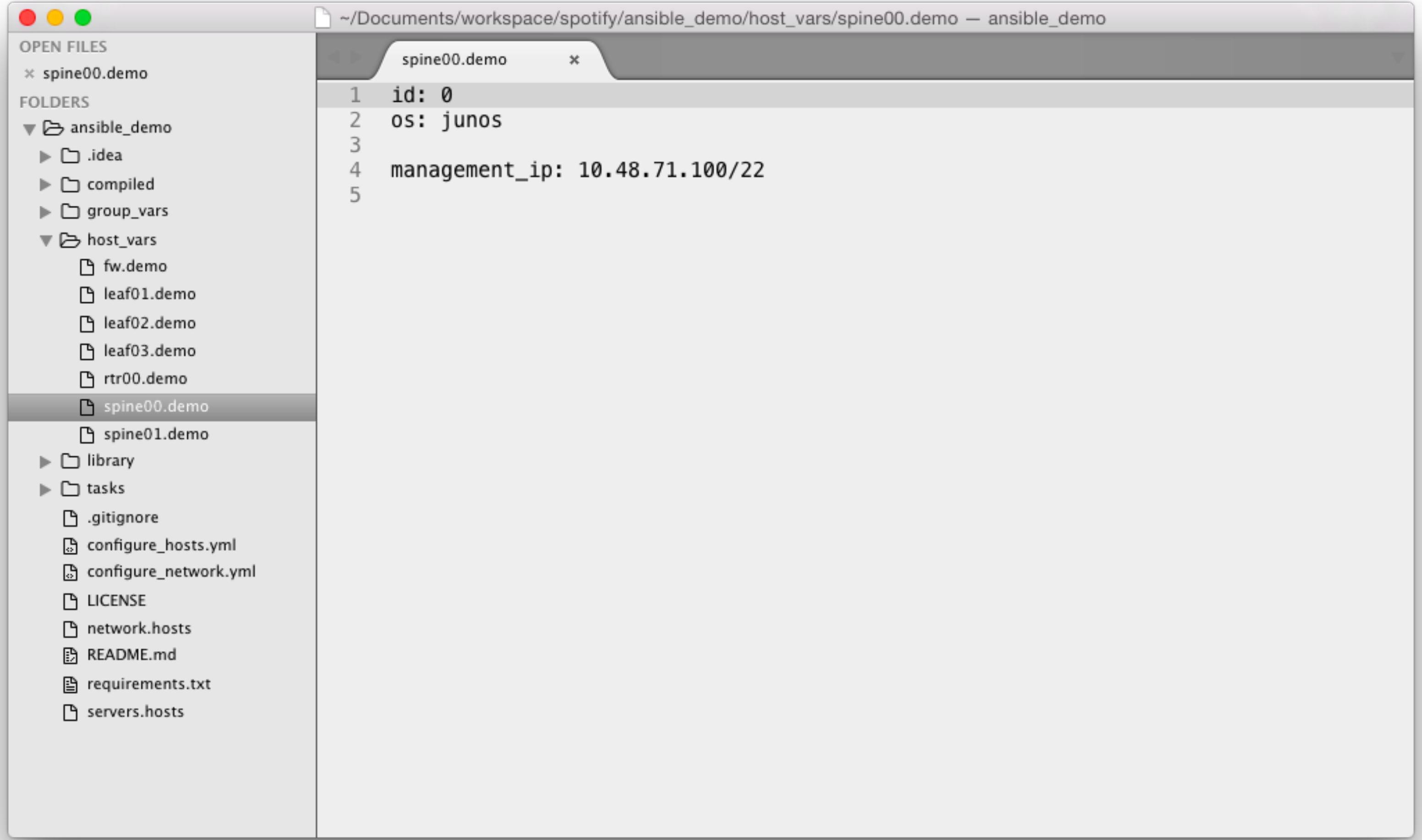
```
---
users:
    - name: admin
      secret: $1$Od4yU7aX$qt83nSW/F55td0V0CZZeR0
    - name: dbarroso
      secret: $1$jazb$8SeTUjYz439INXKr5sF3N1
      ssh_key: ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDZNdIq1KbGzaaKjQLwVmhYbZZ2lNs

management_gw: 10.48.68.1

primary_dns_server: '8.8.8.8'

primary_ntp_server: '81.234.134.160'
secondary_ntp_server: '130.236.254.17'

interconnect_range:
  0: 192.169.0
  1: 192.169.1

vrfs:
  frontend:
    id: 2
    as_range: 4290020000
    loopback_range: 10.255.254
    access_range: 10.2
  backend:
    id: 3
    as_range: 4290030000
    loopback_range: 10.255.255
    access_range: 10.3
```

Some variables are defined at the DC1 level



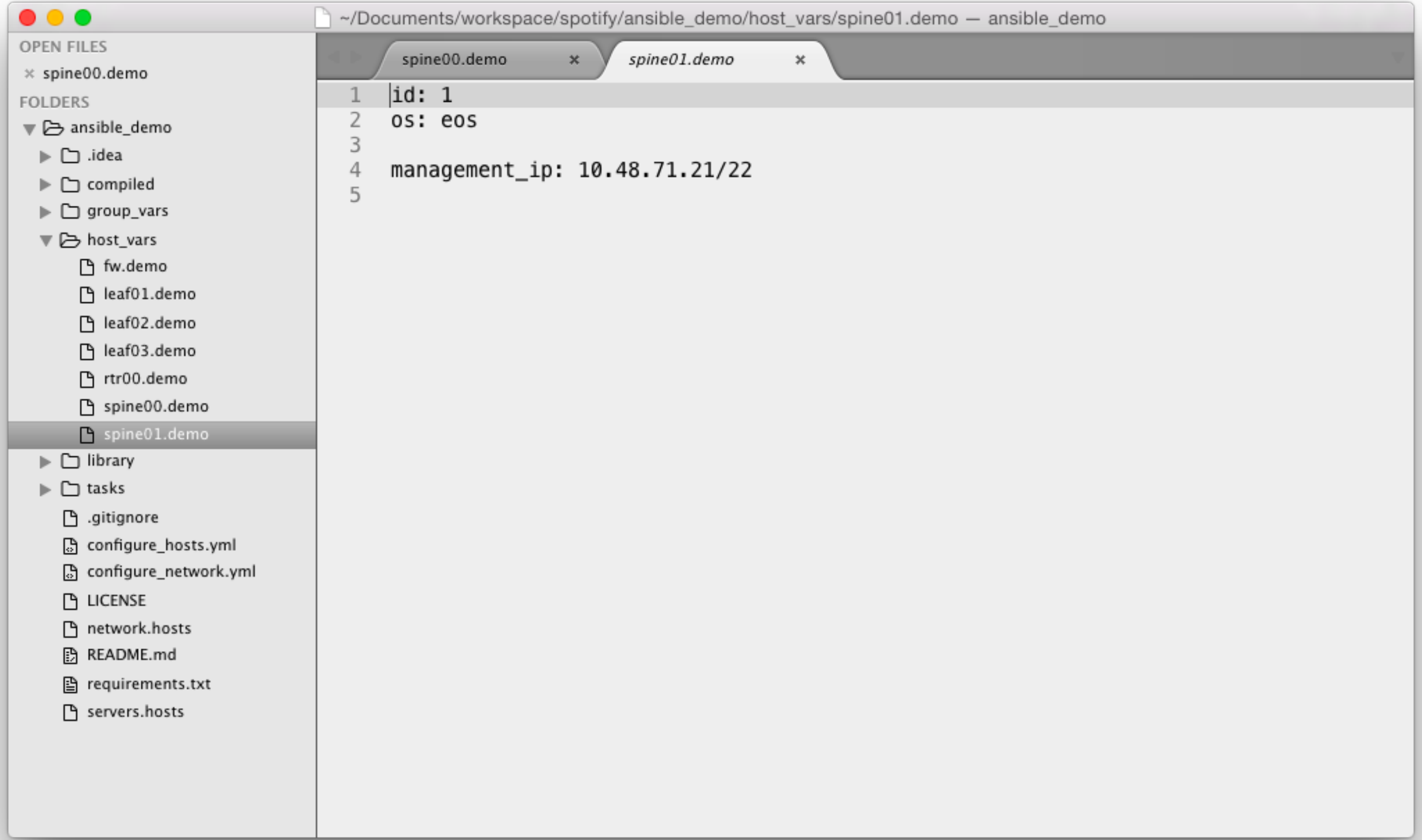
```
---
policies:
    - ['frontend', 'backend', 'PING']
    - ['backend', 'frontend', 'PING']
    - ['backend', 'frontend', 'HTTP']
    - ['frontend', 'backend', 'MYSQL']

internet_peers:
  # Possible options are:
  # pass, disabled, primary_transit, secondary_transit, normal_peer, depreferred_p
  transit:
    172.20.23.4:
      description: Expensive transit provider
      as: 65301
      policy: primary_transit
    10.4.124.252:
      description: Super expensive transit provider
      as: 65102
      policy: secondary_transit
  peers:
    172.20.255.1:
      description: ACME Corp
      as: 65401
      policy: normal_peer
    172.20.255.2:
      description: We have eyeballs
      as: 65410
      policy: depreferred_peer
    172.20.255.3:
      description: University of Neverland
```

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



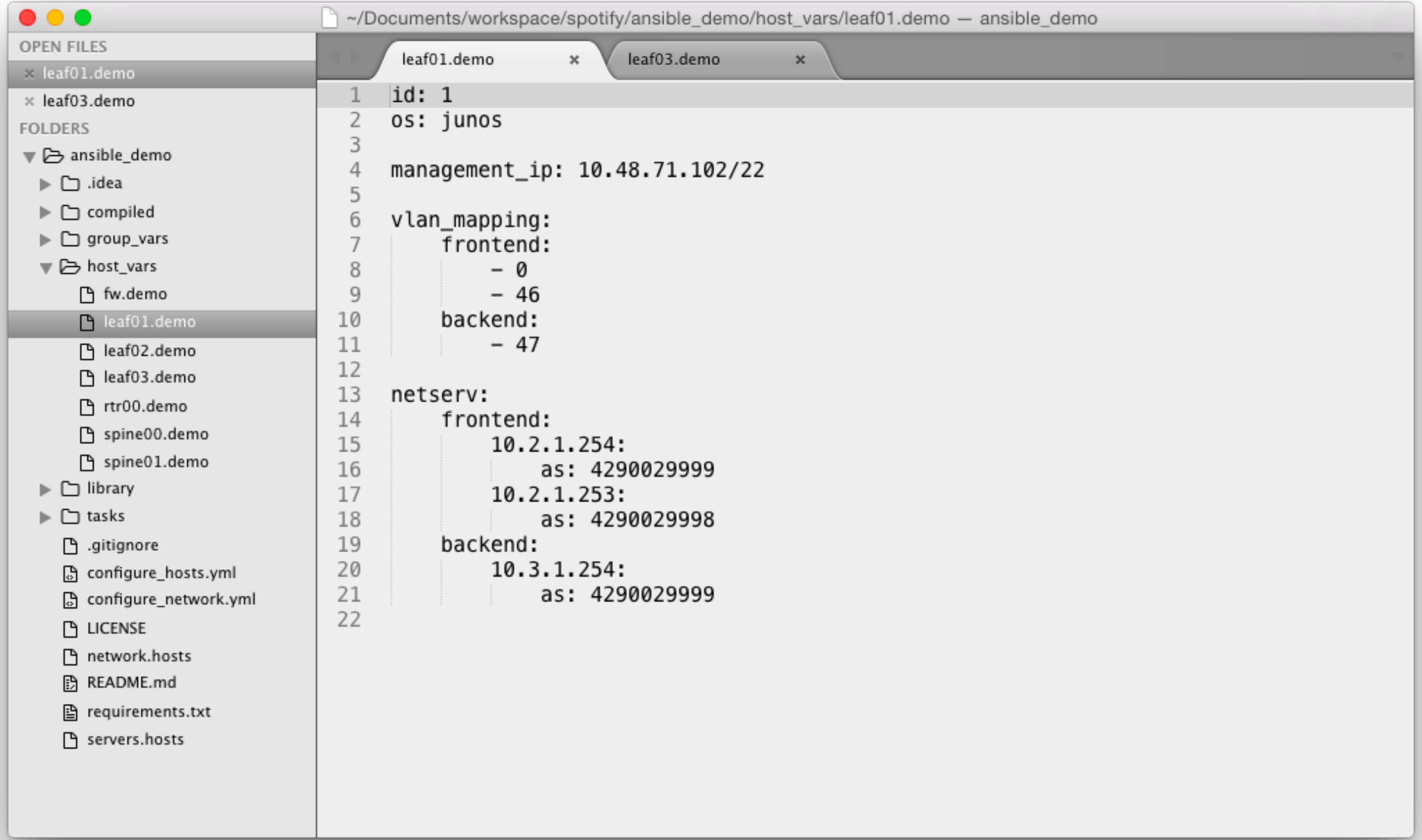
```
id: 0
os: junos

management_ip: 10.48.71.100/22

```

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



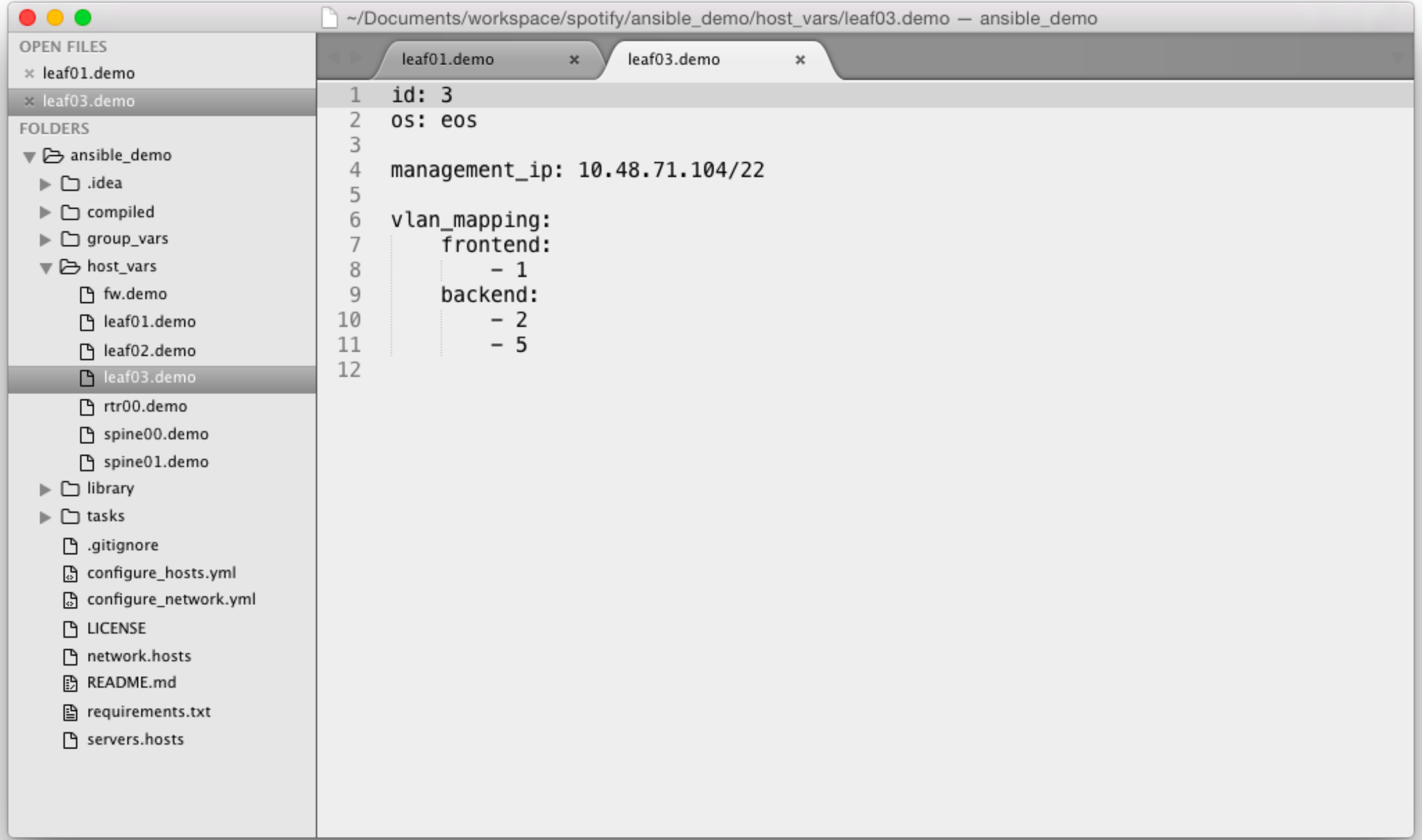
```
id: 1
os: eos

management_ip: 10.48.71.21/22

```

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



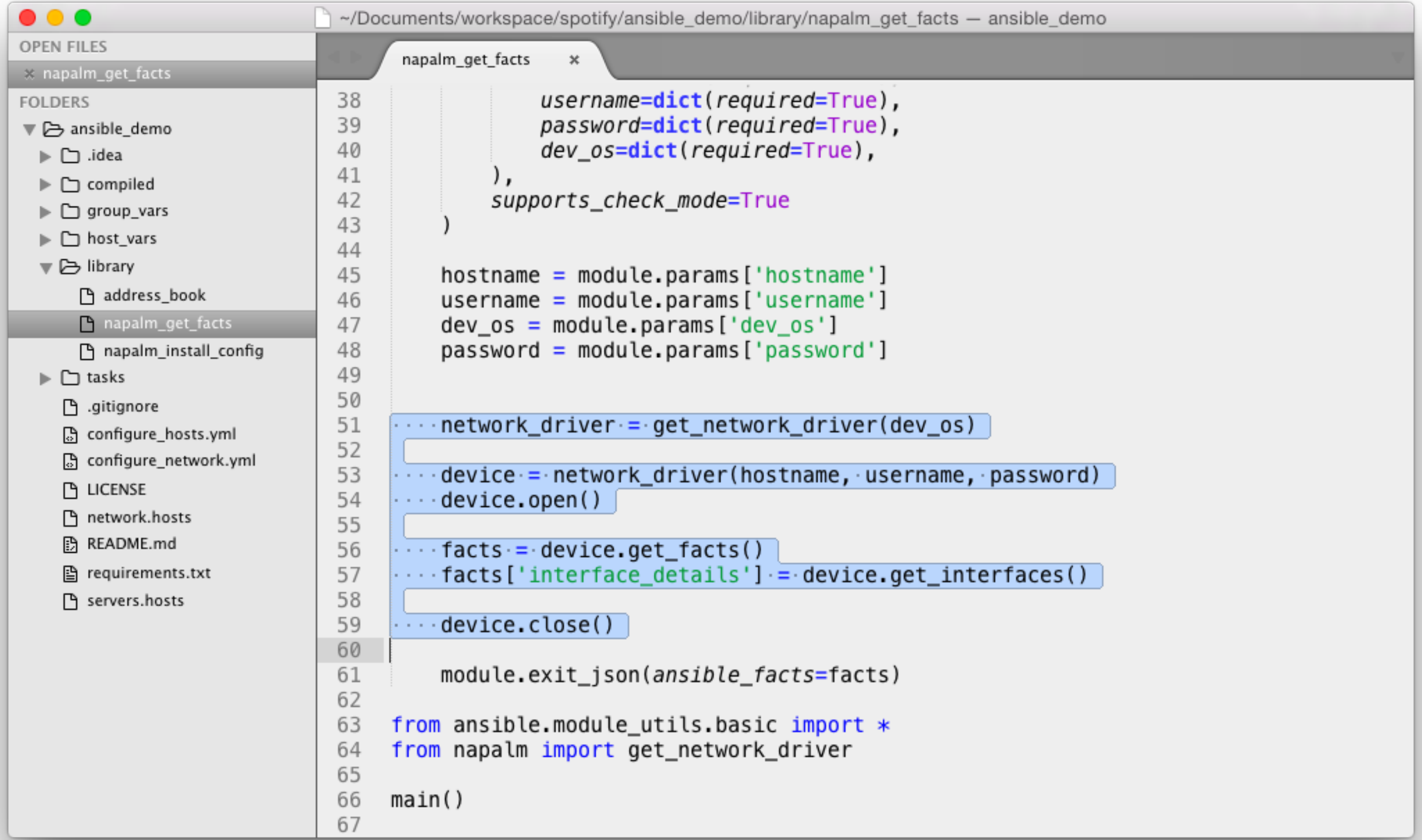
```
id: 1
os: junos

management_ip: 10.48.71.102/22

vlan_mapping:
    frontend:
        - 0
        - 46
    backend:
        - 47

netserv:
    frontend:
        10.2.1.254:
            as: 4290029999
        10.2.1.253:
            as: 4290029998
    backend:
        10.3.1.254:
            as: 4290029999

```

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



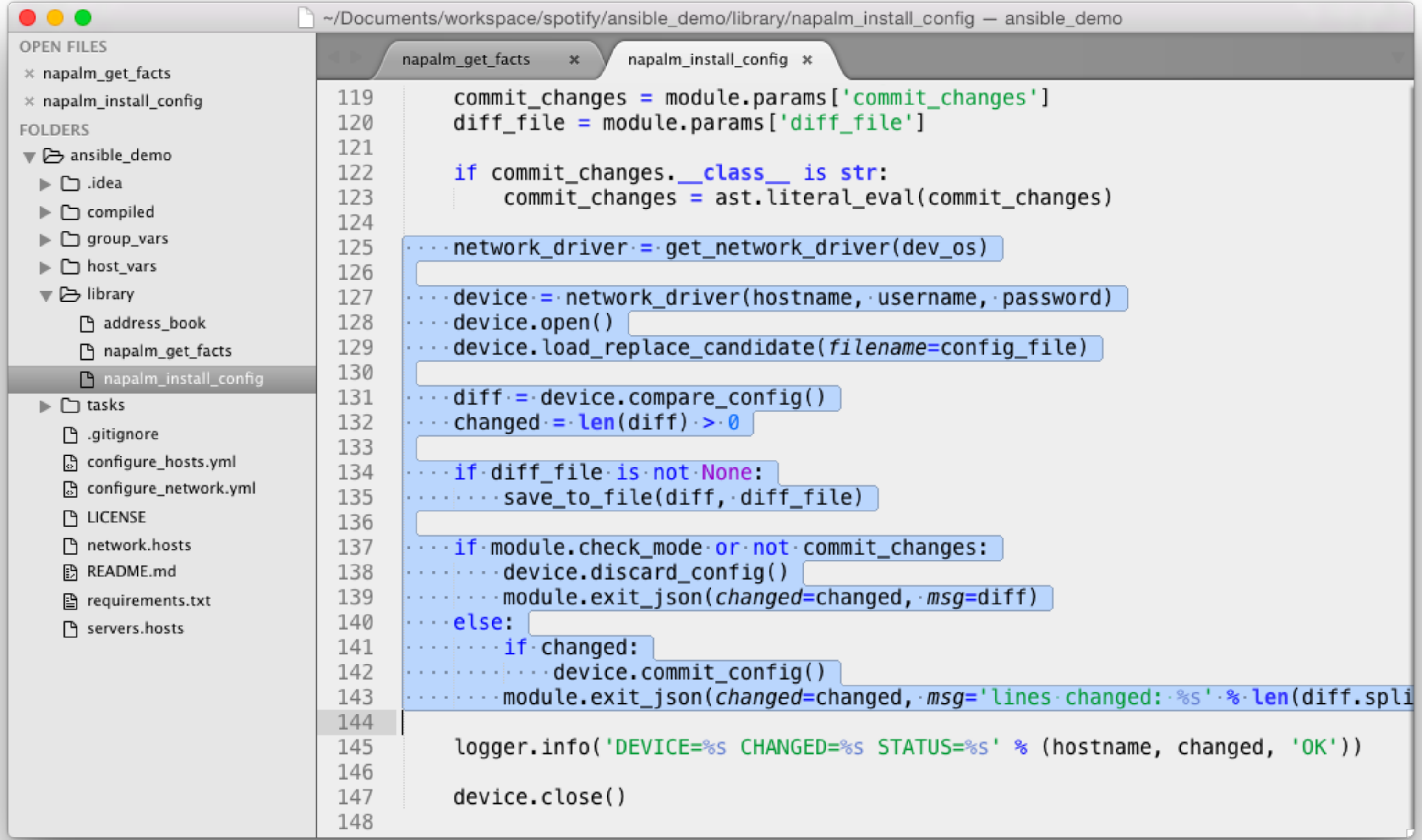
```yaml
id: 3
os: eos

management_ip: 10.48.71.104/22

vlan_mapping:
    frontend:
        - 1
    backend:
        - 2
        - 5

```

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



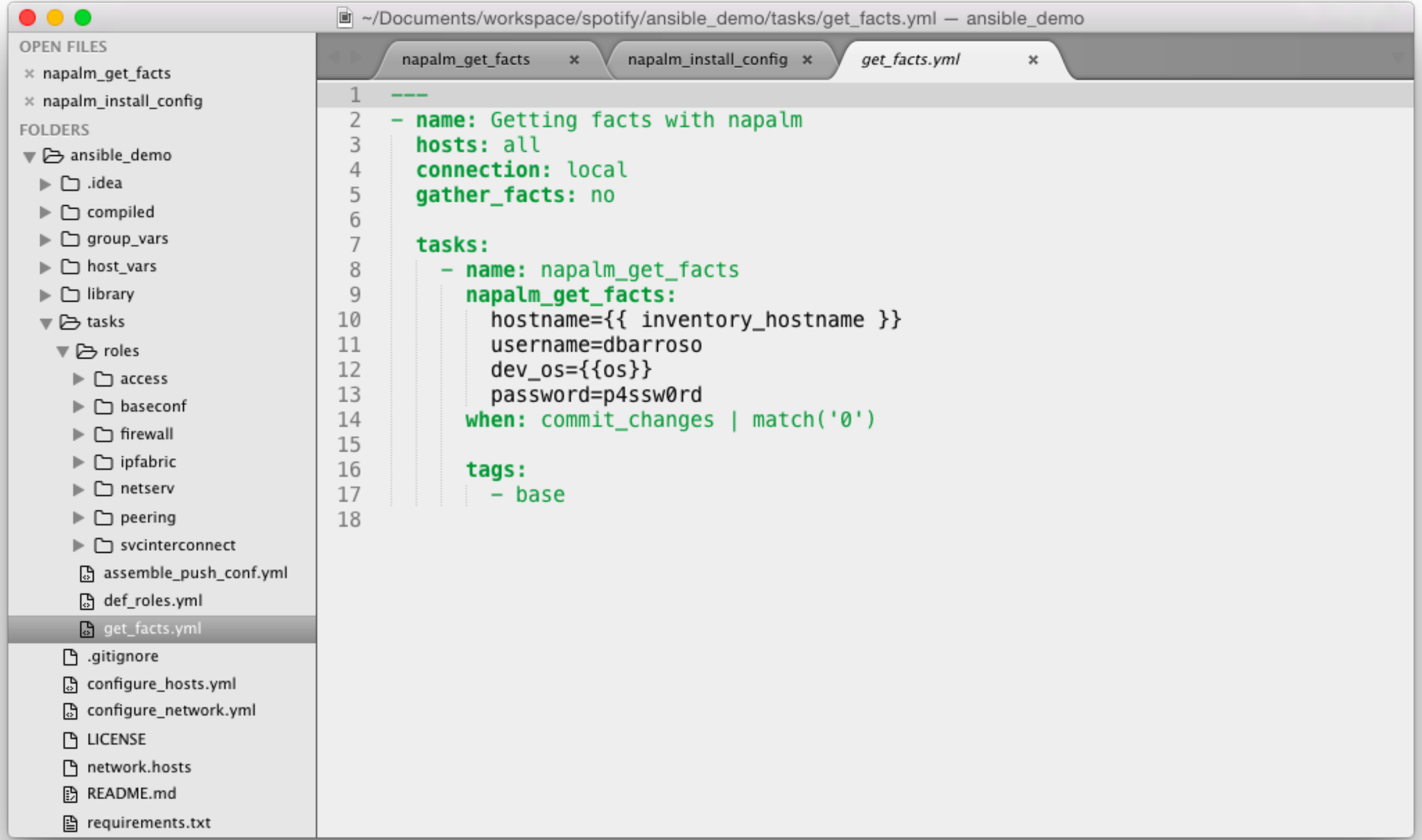
```
            username=dict(required=True),
            password=dict(required=True),
            dev_os=dict(required=True),
        ),
        supports_check_mode=True
    )

    hostname = module.params['hostname']
    username = module.params['username']
    dev_os = module.params['dev_os']
    password = module.params['password']


    network_driver = get_network_driver(dev_os)

    device = network_driver(hostname, username, password)
    device.open()

    facts = device.get_facts()
    facts['interface_details'] = device.get_interfaces()

    device.close()

    module.exit_json(ansible_facts=facts)

from ansible.module_utils.basic import *
from napalm import get_network_driver

main()

```

NAPALM plugins are vendor agnostic (get_facts)



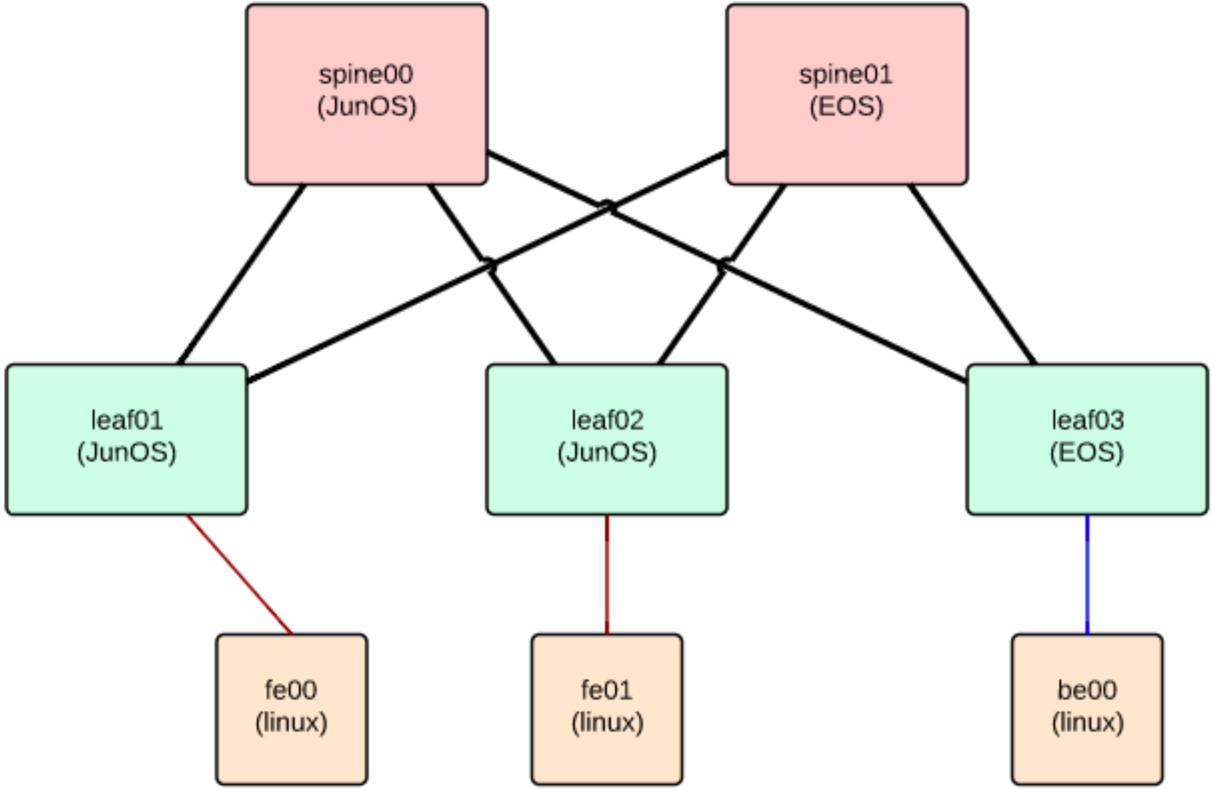
```
    commit_changes = module.params['commit_changes']
    diff_file = module.params['diff_file']

    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.spli

    logger.info('DEVICE=%s CHANGED=%s STATUS=%s' % (hostname, changed, 'OK'))

    device.close()

```

NAPALM plugins are vendor agnostic (napalm_install_config)



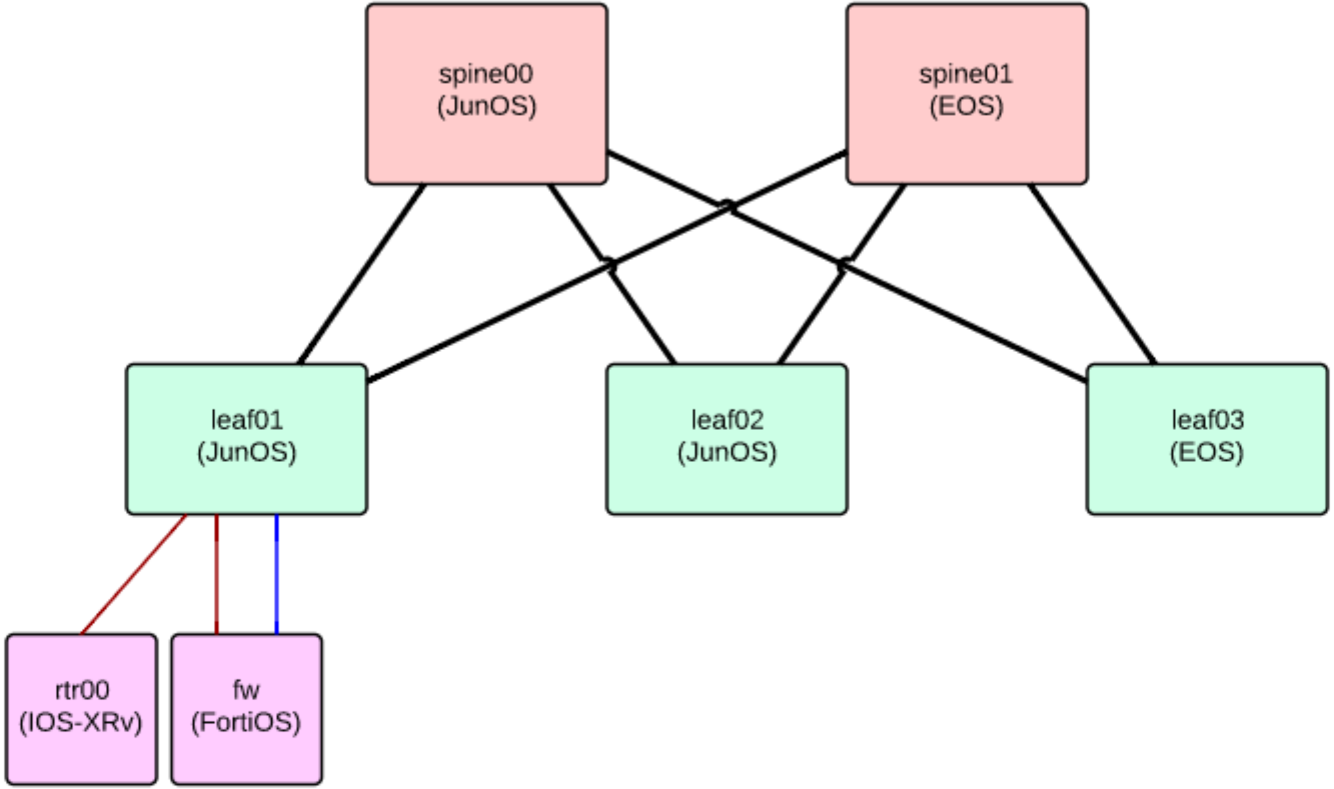
```yaml
---
- name: Getting facts with napalm
  hosts: all
  connection: local
  gather_facts: no

  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



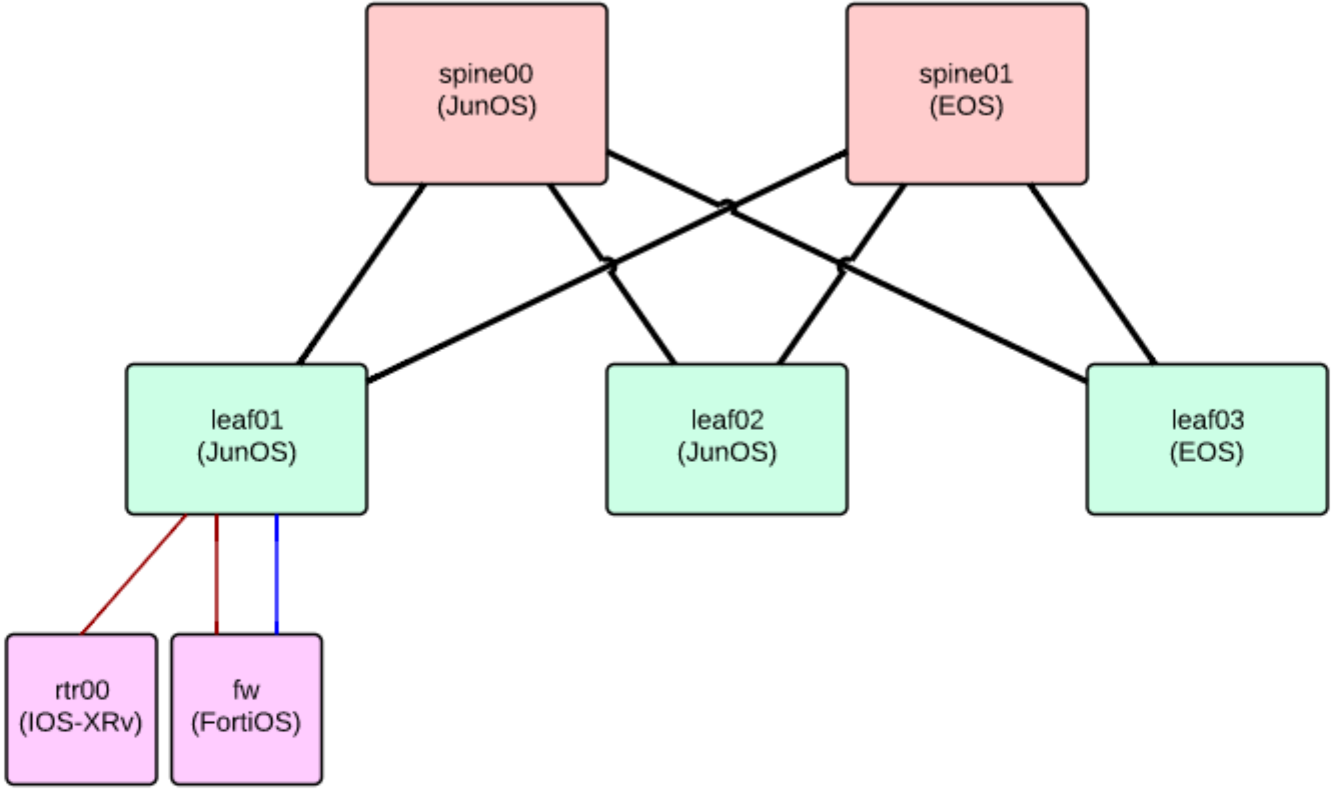
Building the IP Fabric and the Access layer

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



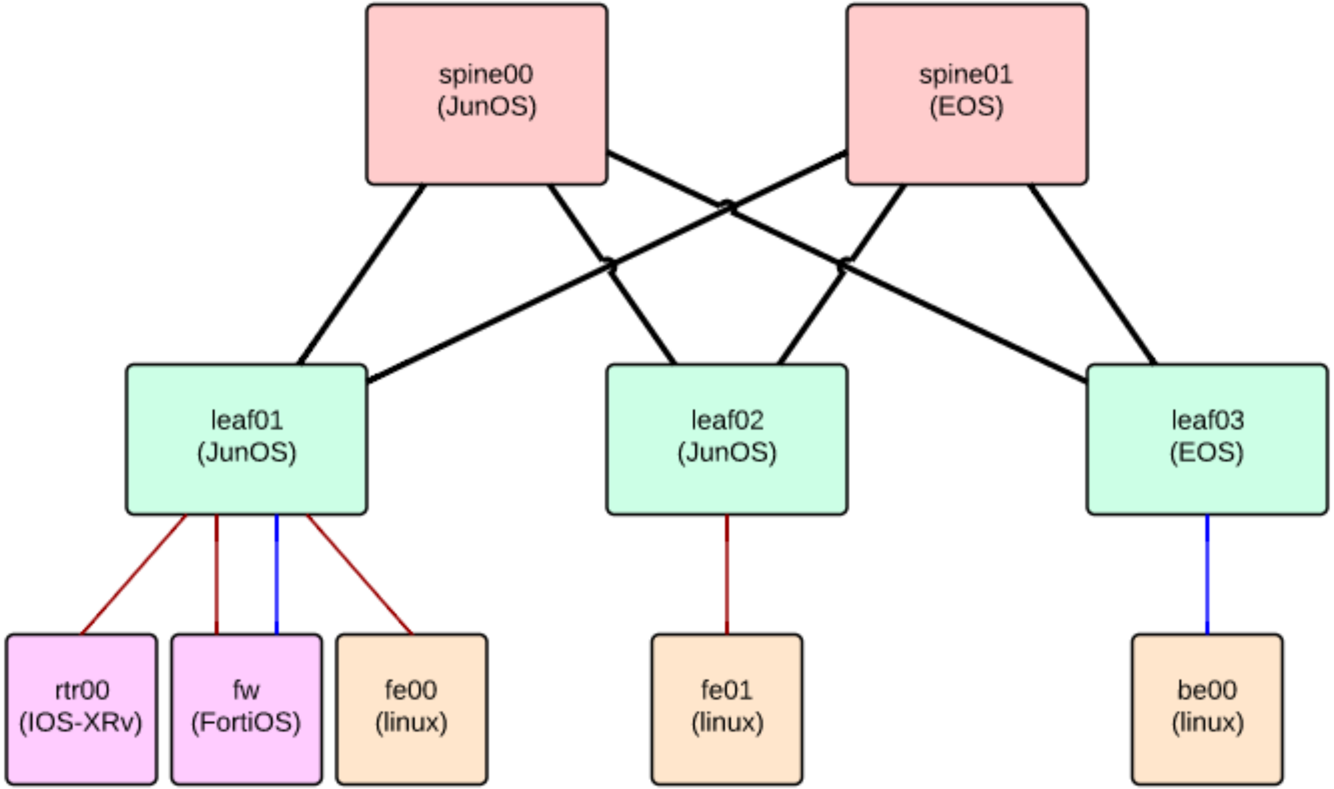
Connecting the network services

```
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"
```



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