コンフィグの抽象化

Shishio Tsuchiya
shtsuchi@arista.com

今日話したい事

- YANGの状況
- OpenConfig
- もっと深くAnsible
- NAPALM

自動化の考え方

	大規模クラウド事業者	先端企業 / サービスプロバイダー	従来の企業 / サービスプロバイダー
1人のエンジニアがメンテナンスするネットワーク機器数	10,000	1,000	100
自動化ツール	自前で作成	DevOpsツール	ベンダーソリューション
JANOG40のセッション (個人的な感想)	フロントエンドエンジニアがY!のNW部隊で取り組んだこと	コンフィグの抽象化 OpenConfigを用いたネットワーク機器操作	ShowNetでピアリングを自動化してみた

スタートはともかく移行出来るべき!?

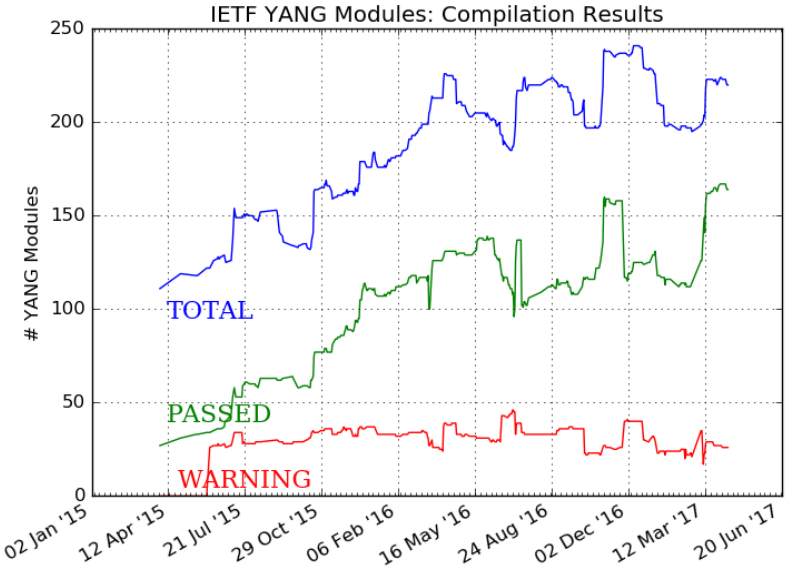
ARISTA

YANGとは？

- Yet Another Next Generationモデリング言語
- NETCONFのRequest/Replyの中のObjectおよびデータを定義
- XMLスキーマーとSMIに類似（より強力）
- コンフィグレーション/オペレーション/RPCコールのモデル化
- RFC 6020 - YANG - A data modeling language for NETCONF
- それぞれのベンターはCLIを持っていて、それをXMLで包むのみでは十分ではない
- YANGは全てのネットワーク機器に共通のデータモデルを提供して運用/設定を行う
- 各々のベンダーは共通のYANGモジュールを実装しなければならない

YANG Status Industry

https://www.ietf.org/blog/2016/03/yang-data-models-in-the-industry-current-state-of-affairs/



- IETF YANG MODELS 236
- MEF YANG MODELS 6
- IEEEStandard YANG MODELS 13
- IEEEExperimental YANG MODELS 2
- BBF YANG MODELS 156
- ONFOpenTransport YANG MODELS 10
- Openconfig YANG MODELS 105
- OPENDAYLIGHT YANG MODELS(Beryllium) 703
- SysrepoInternal YANG MODELS 3
- SysrepoApplication YANG MODELS 4
- OpenROADM20 YANG MODELS 52

IETFのみでは無く多くの標準化団体でも取り組まれ進んでいる

http://www.claise.be/YANGPageMain.html

ARISTA

Githubから見るベンダーのステータス 2017.7.18 現在

- Githubで各標準化団体やベンダーが定義しているYANGモジュールを共有

  https://github.com/YangModels/yang

  https://github.com/YangModels/yang/tree/master/vendor

  を見るとベンダー特有モジュールのステータスがある程度予想出来る

  フォルダがあるのはCiena/Cisco/Juniperのみ

- Cisco
  - IOS-XE 16.3.1/NX-OS 7.0(3)/IOS-XR5.3.1あたりから
  - ベンダー独自のモジュールも積極的に対応。Argumentで連携出来るとの考え方
- Juniper
  - 最初はoperational.yang/configurational.yangという形で全体的なサポートの仕方
  - https://github.com/Juniper/yang に移動し、operational.yangは複数に分離
- Arista
  - 現状Openconfigのみのサポートの為、本フォルダには無い

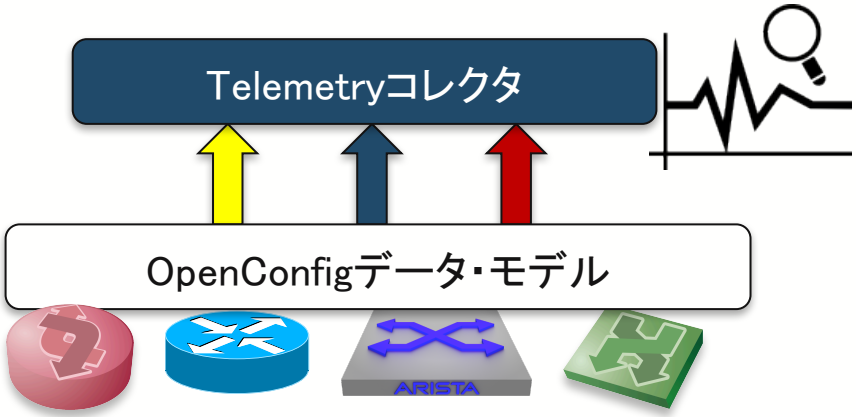
OpenConfig

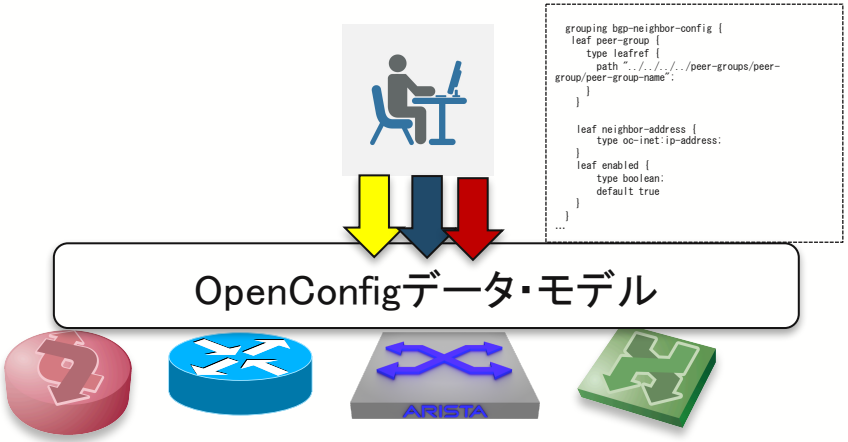
http://openconfig.net/

- Openconfigはよりダイナミックでプログラム可能な運用を目指す非公式なネットワーク運用者のグループ
- 運用者の使用例や要求より、実際の運用に必要なベンダー非依存のデータ・モデルを定義
- BGPやACL設定やテレメトリーをモデル化済み
  - https://github.com/openconfig/public/tree/master/release/models
- トランスポートとRPCプロトコル
  - データ・モデルはトランスポートやRPCとは独立している。任意のプロトコルを使用出来る
  - 参加者の中ではNETCONF/RESTCONFおよびgNMI(gRPC Network Management Interface)を使用してる
  - gNMIはTelemetryとConfigurationで共通のgRPCプロトコルを使用

OpenConfigが目指すもの

- ベンダーや世代に依存しない設定/テレメトリーの為のデータモデルの提供
- トランスポートは運用者が環境に合わせて選べる





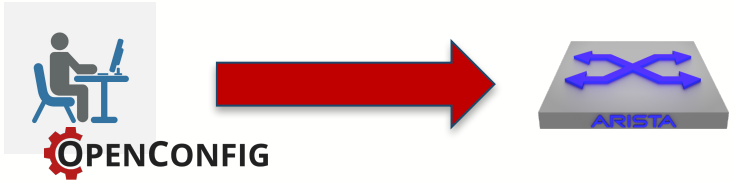
デモ openconfig

- デモ動画

https://youtu.be/satbCzrMjk0

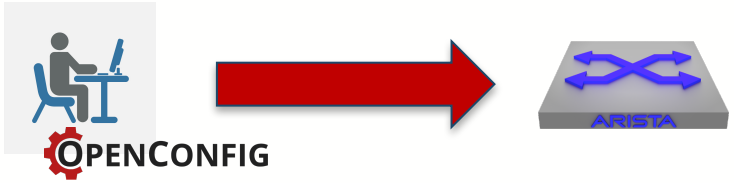
```
vEOS1#show running-config | section management api
management api http-commands
   protocol http
   no shutdown
!
management api openconfig
   transport grpc default
vEOS1#show running-config | section router bgp
router bgp 65000
   bgp listen range 10.100.1.0/24 peer-group DC peer-filter Leaf
   neighbor DC peer-group
   neighbor DC maximum-routes 12000
   network 10.255.255.1/32
```

デモ openconfig



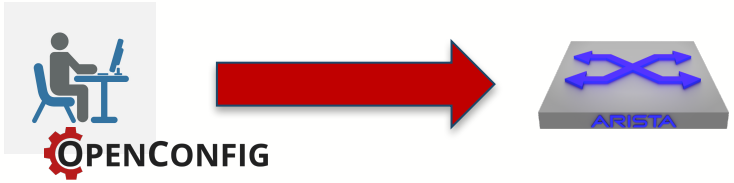

```
arista@ubuntu:~/OpenConfig$ ./grpcli_linux_386 -addr 10.10.1.1:6030  -username "admin" -password "admin" -get
/network-instances/network-instance/default/protocols/protocol/"BGP BGP"/bgp/global
{"openconfig-bgp-global:config": {"as": 65000, "router-id": "0.0.0.0"}, "openconfig-bgp-global:default-route-
distance": {"config": {"external-route-distance": 200, "internal-route-distance": 200}}, "openconfig-bgp-
global:graceful-restart": {"config": {"restart-time": 0, "stale-routes-time": "+Inf.0"}}, "openconfig-bgp-
common:route-selection-options": {"config": {"advertise-inactive-routes": false, "always-compare-med": false,
"external-compare-router-id": true}}}
arista@ubuntu:~/OpenConfig$ ./grpcli_linux_386 -addr 10.10.1.1:6030  -username "admin" -password "admin" -get
/network-instances/network-instance/default/protocols/protocol/"BGP BGP"/bgp/global/config
{"openconfig-bgp-global:as": 65000, "openconfig-bgp-global:router-id": "0.0.0.0"}
```

デモ openconfig




```
arista@ubuntu:~/OpenConfig$ ./grpcli_linux_386 -addr 10.10.1.1:6030  -username "admin" -password "admin" -get
/network-instances/network-instance/default/protocols/protocol/"BGP BGP"/bgp/global/config | python -m json.tool
{
    "openconfig-bgp-global:as": 65000,
    "openconfig-bgp-global:router-id": "0.0.0.0"
}
```

デモ openconfig




```
arista@ubuntu:~/OpenConfig$ ./grpcli_linux_386 -addr 10.10.1.1:6030  -username "admin" -password "admin" -get
/network-instances/network-instance/default/protocols/protocol/"BGP BGP"/bgp/global/config/router-id
"0.0.0.0"
arista@ubuntu:~/OpenConfig$ ./grpcli_linux_386 -addr 10.10.1.1:6030  -username "admin" -password "admin" -set
/network-instances/network-instance/default/protocols/protocol/"BGP BGP"/bgp/global/config/router-
id='"10.255.255.1"'
arista@ubuntu:~/OpenConfig$ ./grpcli_linux_386 -addr 10.10.1.1:6030  -username "admin" -password "admin" -get
/network-instances/network-instance/default/protocols/protocol/"BGP BGP"/bgp/global/config | python -m json.tool
{
    "openconfig-bgp-global:as": 65000,
    "openconfig-bgp-global:router-id": "10.255.255.1"
}
```

デモ openconfig


```
vEOS1#show running-config | section router bgp
router bgp 65000
   router-id 10.255.255.1
   bgp listen range 10.100.1.0/24 peer-group DC peer-filter Leaf
   neighbor DC peer-group
   neighbor DC maximum-routes 12000
   network 10.255.255.1/32
```

問題点!?

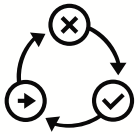
- トランスポートの自由度が高すぎる?
- 各ベンダーのソフトウェアのサポート状況に依存
  - データモデル
  - トランスポート
- 古い機器サポートは?

	大規模クラウド事業者	先端企業 / サービスプロバイダー	従来の企業 / サービスプロバイダー
1人のエンジニアがメンテナンスするネットワーク機器数	10,000	1,000	100
自動化ツール	自前で作成	DevOpsツール	ベンダーソリューション
各ツールのポジション (個人的な感想)			

ARISTA

Ansibleとは何か？

- Red hatのオープンソースの構成管理ツール
- 読みやすい
  - タスク内容を記述するPlaybookはテキスト形式（YAML）



- エージェンスレス
  - Chef/Puppetとは違いクライアントにエージェントが要らない
- 他のオープンソースとの連携が可能

Ansible

- Ansibleは読みやすく、エージェントレス
- ネットワーク機器の対応も幅広くなってきた*
- またサーバーやクラウド対応も増え、連携がしやすい
- 多くの設定変更がある際には再現性が高い為、効力を発揮する

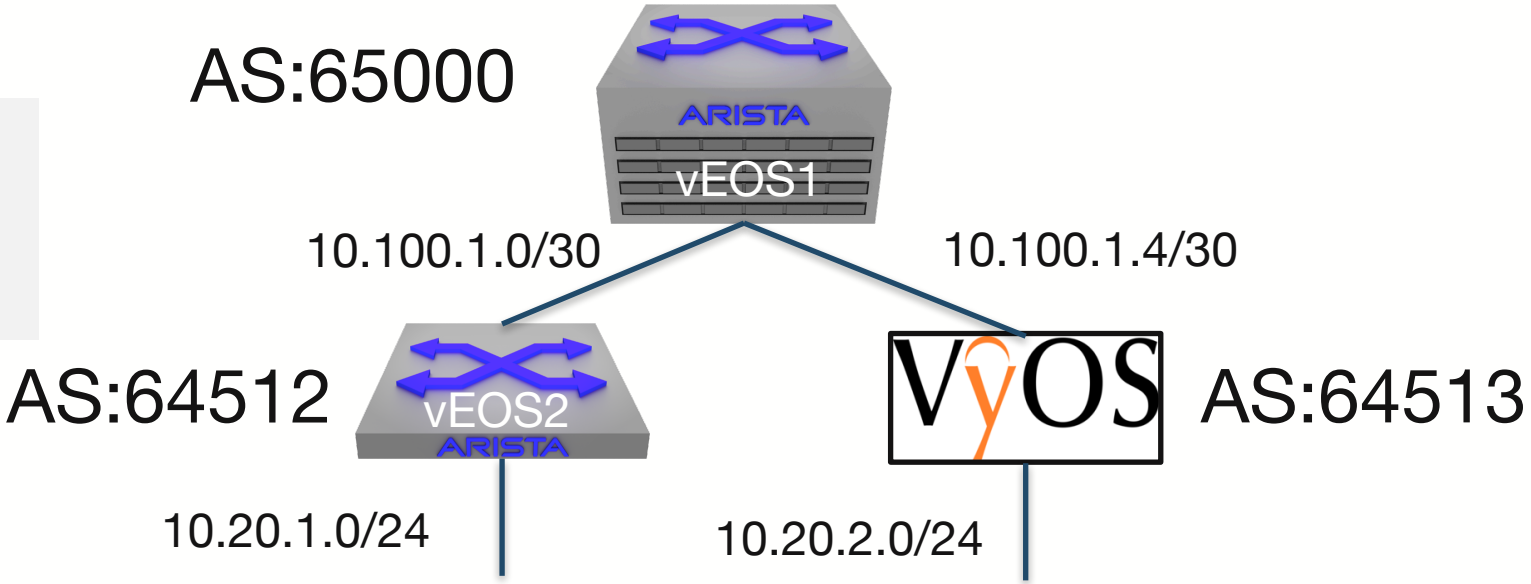
Ansible Network Module

http://docs.ansible.com/ansible/list_of_network_modules.html

Vender	Arista	Cisco	Cisco	Cumulus	Juniper
OS	EOS	IOS/IOS-XE	IOS-XR	Cumulus Linux	JUNOS
メモ	eos_config はauthorize: true が必要 設定を保存するた めにはsave:yes	ios_config authorize: true が 必要 設定を保存するた めにはsave:yes	commit がデフォルトで動く formalコマンドもサポート	インターフェースを変え た時にはrestartが必 要 localアクションではなく、 リモート	NETCONF で動作する ためjunos- ezncが必要 commit が デフォルトで 動く

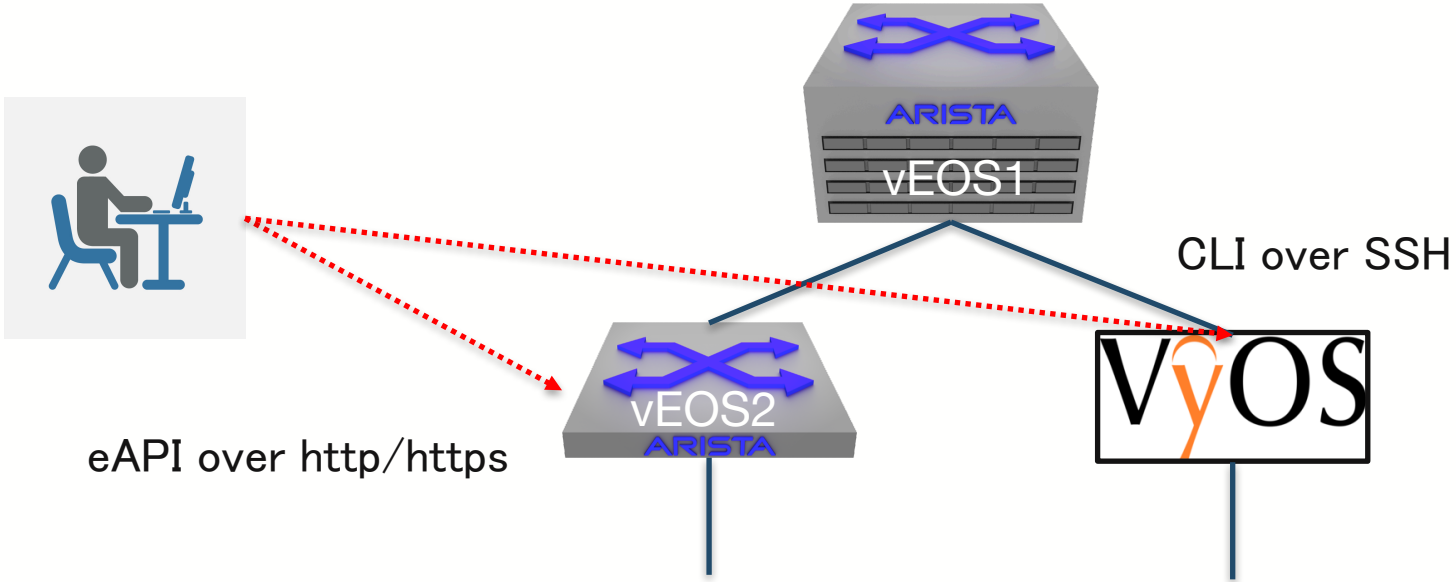
- 沖縄オープンラボでAnsibleハッカソンを実施
- 一通りの機器が動作する事は確認が出来たが
- 機器毎にトランスポートやデータモデル(CLIのフォーマット)に違いがある

ベンダー毎の違いをこんなネットワークで考える

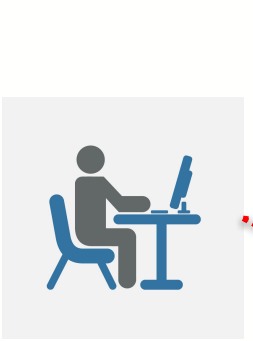


https://youtu.be/7k7hk3Wjm-c

接続方法の違い



設定コマンドの違い



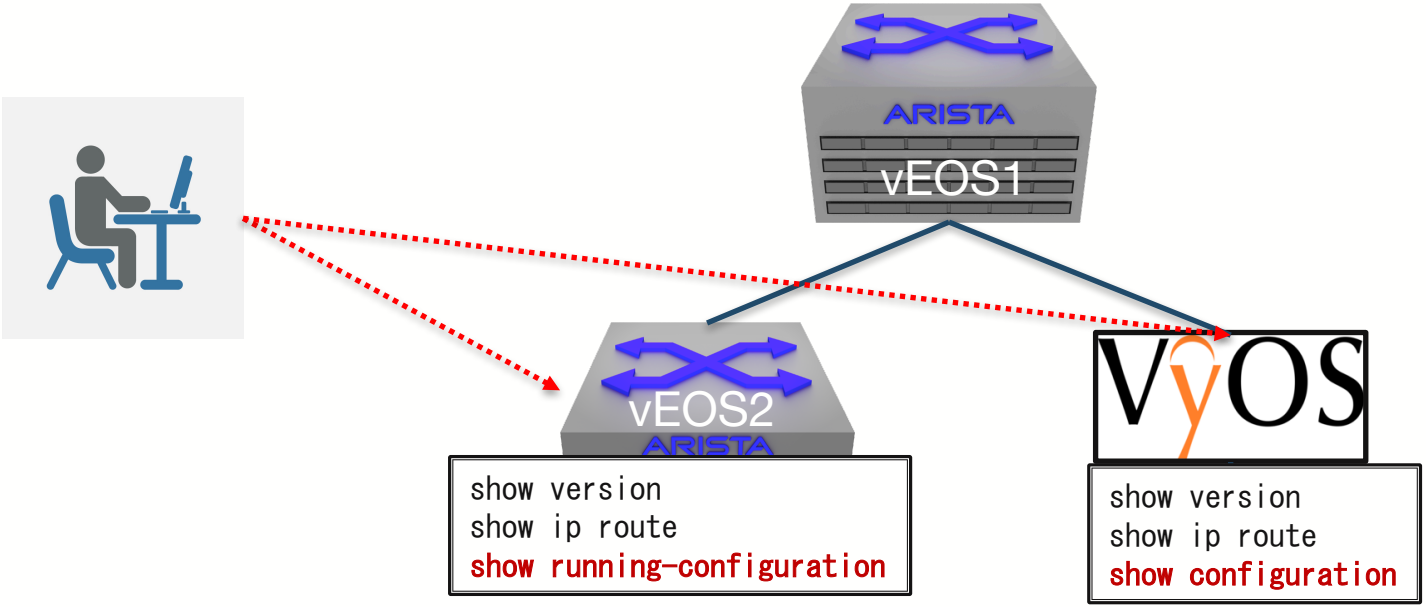
CLI over SSH


```
!
interface Management1
   ip address 10.10.1.2/24
!
router bgp 64512
   neighbor 10.100.1.1 remote-as 65000
!
```

```
set interfaces ethernet eth0 address '10.10.10.5/24'
set protocols bgp 64513 neighbor 10.100.1.5 remote-as '65000'
```

- 階層型コンフィグとワンラインコンフィグ

確認コマンドの違い



- バージョン確認/ルーティングテーブル確認/設定確認

varsとJinja2テンプレート

- インベントリーファイルはいくつかのGroupに分ける事が出来る
- host_vars以下でホスト単体の変数
- group_vars以下でグループの変数
- templates以下ではJinja2テンプレートを使ってコンフィグをテンプレート化出来る

```
arista@ubuntu:~$ cat ansible/hosts
[Spine]
vEOS1
[EOS-leaf]
vEOS2
#vEOS3
[vyos-leaf]
vyos
[leafs:children]
EOS-leaf
vyos-leaf
```

```
arista@ubuntu:~$ tree ansible
ansible
|-- eos-leaf-show.yaml
|-- eos-leaf.yaml
|-- group_vars
|   |-- EOS-leaf
|   |-- all
|   |-- leafs
|   `-- vyos-leaf
|-- host_vars
|   |-- vEOS2
|   |-- vEOS3
|   `-- vyos
|-- hosts
|-- templates
|   |-- eoint.j2
|   |-- eontp.j2
|   |-- eorouting.j2
|   |-- vyint.j2
|   |-- vyntp.j2
|   `-- vyrouting.j2
|-- vyos-disable.yaml
|-- vyos-leaf-show.yaml
`-- vyos-leaf.yaml
```

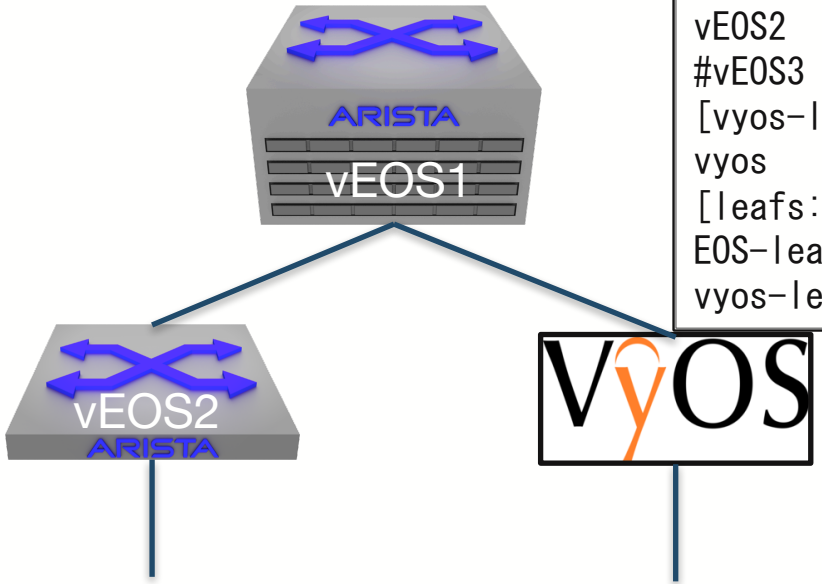
デモ vEOS1設定

AS Ranges for Dynamic BGP Peer Groups

- 設定がシンプルになり尚且つ各Spineで共有化出来るような機能は有意義

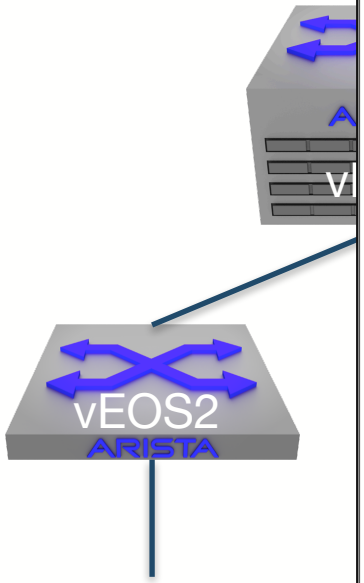
```
peer-filter Leaf
   10 match as-range 64512-64999 result accept
   20 match as-range 65002-65534 result reject
!
router bgp 65000
   bgp listen range 10.100.1.0/24 peer-group DC peer-filter Leaf
   neighbor DC peer-group
   neighbor DC maximum-routes 12000
   network 10.255.255.1/32
!
```

インベントリー確認 hosts




```
arista@ubuntu:~/ansible$ cat hosts
[Spine]
vEOS1
[EOS-leaf]
vEOS2
#vEOS3
[vyos-leaf]
vyos
[leafs:children]
EOS-leaf
vyos-leaf
```

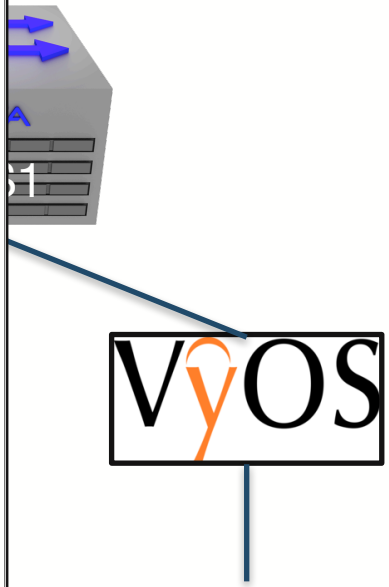
eos確認playbooks



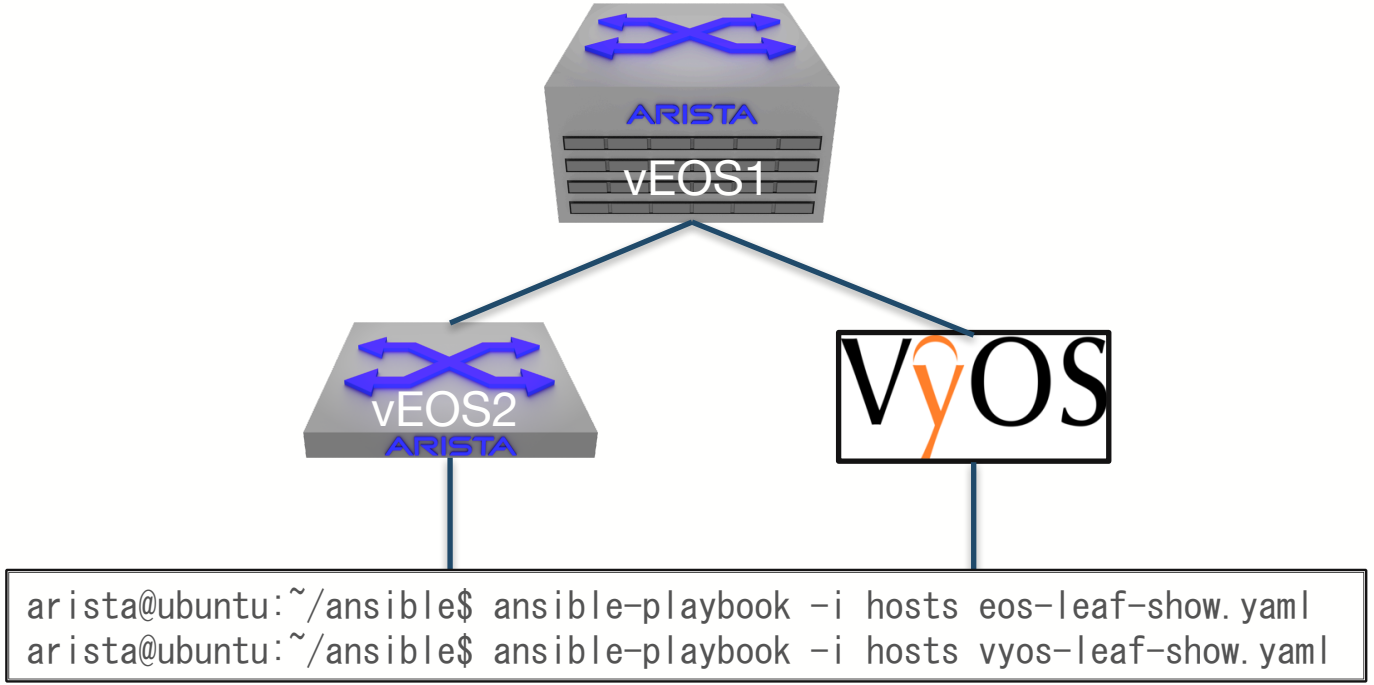

```
arista@ubuntu:~/ansible$ cat eos-leaf-show.yaml
---

- hosts: EOS-leaf
  connection: local
  gather_facts: yes

  tasks:
  - name: Gather info from show Version/show ip route/show config
    eos_command:
     commands:
       - show version
       - show ip route
       - show running-config
     provider: '{{ eos_connection }}'
    register: result
  - debug: var=result
arista@ubuntu:~/ansible$ cat group_vars/EOS-leaf
   eos_connection:
    host: "{{ inventory_hostname }}"
    username: admin
    password: admin
    tranport: eapi
    use_ssl: false
    authorize: true
```

vyos確認playbooks

```
arista@ubuntu:~/ansible$ cat vyos-leaf-show.yaml
---

- hosts: vyos-leaf
  connection: local
  gather_facts: yes

  tasks:
  - name: Gather info from show version/show ip route/show config
    vyos_command:
     commands:
       - show version
       - show ip route
       - show config command
     provider: '{{ vyos_connection }}'
    register: result
  - debug: var=result
  arista@ubuntu:~/ansible$ cat group_vars/vyos-leaf
   vyos_connection:
    host: "{{ inventory_hostname }}"
    username: admin
    password: admin
    use_ssl: false
    authorize: true
```



現状確認



EOS設定変更playbook/jinja2テンプレート

```
arista@ubuntu:~/ansible$ cat eos-leaf.yaml
---

- hosts:  EOS-leaf
  connection: local
  gather_facts: yes

  tasks:
  - debug: var=inventory_hostname
  - name: configure interface
    eos_config:
     src: eoint.j2
     provider: '{{ eos_connection }}'
  - name: configure bgp
    eos_config:
     src: eorouting.j2
     provider: '{{ eos_connection }}'
  - name: configure NTP
    eos_config:
     src: eontp.j2
     provider: '{{ eos_connection }}'
     save: yes
```

```
arista@ubuntu:~/ansible$ cat templates/eoint.j2
{% for eointf in interfaces %}
interface {{ eointf.name }}
   description {{ eointf.description }}
   ip address {{ eointf.address }}/{{ eointf.mask }}
{% endfor %}
arista@ubuntu:~/ansible$ cat templates/eorouting.j2
{% for eorouting in bgpconf  %}
 switchport default mode routed
 ip routing
 router bgp {{ eorouting.localas }}
   neighbor {{ eorouting.nbrip }} remote-as
{{ eorouting.remoteas }}
   redistribute connected
{% endfor %}
arista@ubuntu:~/ansible$ cat templates/eontp.j2
{% for eontp in ntpconf %}
ip name-server {{ eontp.server }}
{% endfor %}
```

EOS設定変更playbook/jinja2テンプレート

```
arista@ubuntu:~/ansible$ ansible-playbook -i hosts eos-leaf.yaml

PLAY [EOS-leaf] ****************************************************************************************************************************************

TASK [Gathering Facts] *********************************************************************************************************************************
ok: [vEOS2]

TASK [debug] *******************************************************************************************************************************************
ok: [vEOS2] => {
    "inventory_hostname": "vEOS2"
}

TASK [configure interface] *****************************************************************************************************************************
 [WARNING]: argument transport has been deprecated and will be removed in a future version

changed: [vEOS2]

TASK [configure bgp] ***********************************************************************************************************************************
changed: [vEOS2]

TASK [configure NTP] ***********************************************************************************************************************************
changed: [vEOS2]

PLAY RECAP *********************************************************************************************************************************************
vEOS2                      : ok=5    changed=3    unreachable=0    failed=0

arista@ubuntu:~/ansible$ ansible-playbook -i hosts eos-leaf-show.yaml
```

VyOS設定変更playbook/jinja2テンプレート

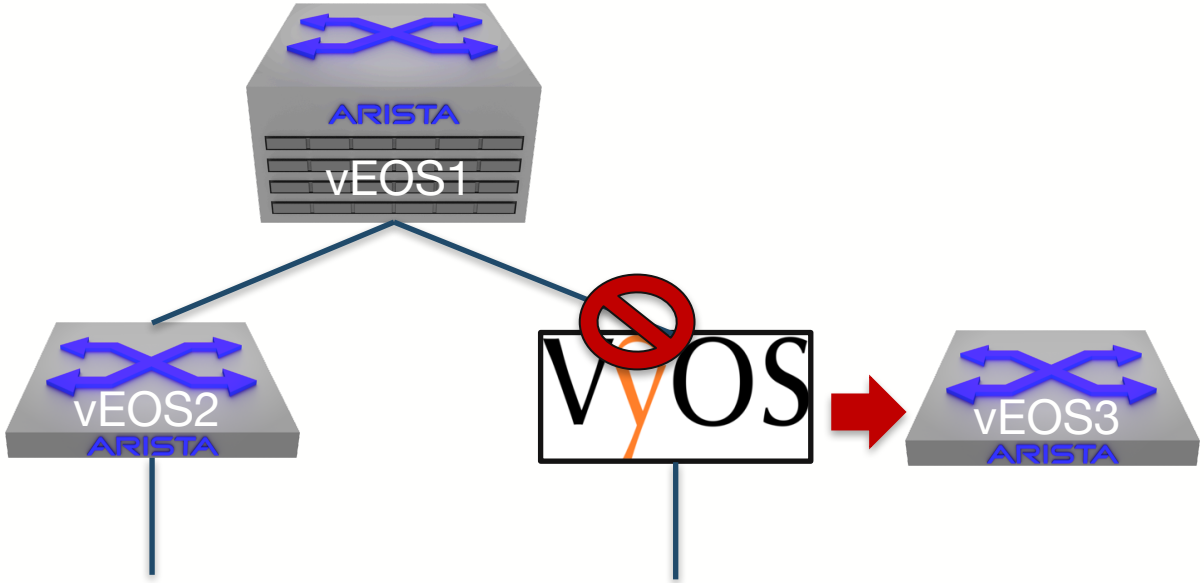
```
arista@ubuntu:~/ansible$ cat vyos-leaf.yaml
---

- hosts: vyos-leaf
  connection: local
  gather_facts: yes

  tasks:
   - debug: var=inventory_hostname
   - name: configure interface
     vyos_config:
      src: vyint.j2
      provider: '{{ vyos_connection }}'
   - name: configure ntp
     vyos_config:
      src: vyntp.j2
      provider: '{{ vyos_connection }}'
   - name: configure bgp
     vyos_config:
      src: vyrouting.j2
      provider: '{{ vyos_connection }}'
      save: yes
     register: result
```

```
arista@ubuntu:~/ansible$ cat templates/vyint.j2
{% for vyint in interfaces %}
set interfaces {{ vyint.name }} address
'{{ vyint.address }}/{{ vyint.mask }}'
set interfaces {{ vyint.name }} description
'{{ vyint.description }}'
{% endfor %}
arista@ubuntu:~/ansible$ cat templates/vyntp.j2
{% for vyntp in ntpconf %}
set system name-server {{ vyntp.server }}
{% endfor %}
arista@ubuntu:~/ansible$ cat templates/vyrouting.j2
{% for vyrouting in bgpconf  %}
set protocols bgp {{ vyrouting.localas }} neighbor
{{ vyrouting.nbrip }} remote-as {{ vyrouting.remoteas }}
set protocols bgp {{ vyrouting.localas }} redistribute
'connected'
{% endfor %}
```

VyOS設定変更playbook/jinja2テンプレート

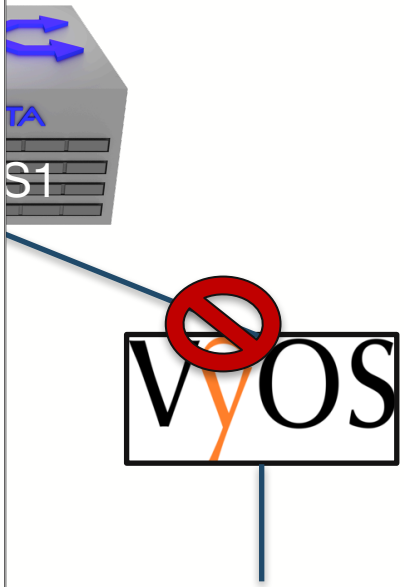
```
arista@ubuntu:~/ansible$ ansible-playbook -i hosts vyos-leaf.yaml

PLAY [vyos-leaf] ***************************************************************************************************************************************

TASK [Gathering Facts] *********************************************************************************************************************************
ok: [vyos]

TASK [debug] *******************************************************************************************************************************************
ok: [vyos] => {
    "inventory_hostname": "vyos"
}

TASK [configure interface] *****************************************************************************************************************************
changed: [vyos]

TASK [configure ntp] ***********************************************************************************************************************************
changed: [vyos]

TASK [configure bgp] ***********************************************************************************************************************************
changed: [vyos]

PLAY RECAP *********************************************************************************************************************************************
vyos                       : ok=5    changed=3    unreachable=0    failed=0

arista@ubuntu:~/ansible$ ansible-playbook -i hosts vyos-leaf-show.yaml
```

VyOS停止->EOS入れ替え



VyOS停止

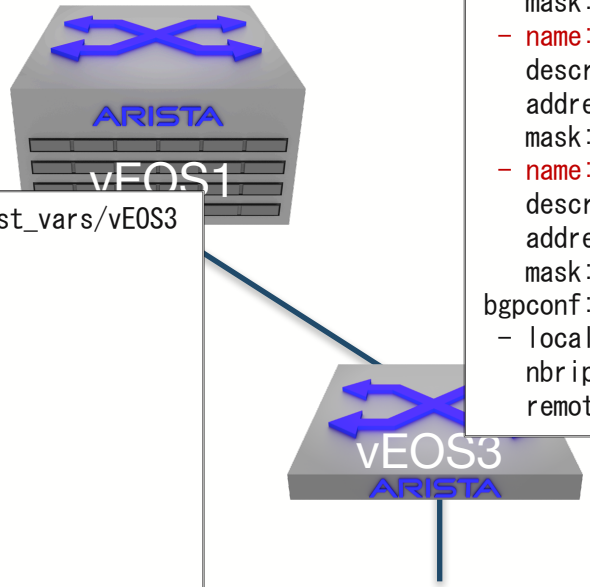
```
arista@ubuntu:~/ansible$ cat vyos-disable.yaml
---

- hosts: vyos-leaf
  connection: local
  gather_facts: yes


  tasks:
   - debug: var=inventory_hostname
   - name: disable vyos
     vyos_config:
      lines: set interfaces ethernet eth1 disable
      provider: '{{ vyos_connection }}'
     register: result
```



VyOS停止playbook/確認

```
arista@ubuntu:~/ansible$ ansible-playbook -i hosts vyos-disable.yaml

PLAY [vyos-leaf] ***************************************************************************************************************

TASK [Gathering Facts] *********************************************************************************************************
ok: [vyos]

TASK [debug] *******************************************************************************************************************
ok: [vyos] => {
    "inventory_hostname": "vyos"
}

TASK [disable vyos] ************************************************************************************************************
changed: [vyos]

PLAY RECAP *********************************************************************************************************************
vyos                       : ok=3    changed=1    unreachable=0    failed=0

arista@ubuntu:~/ansible$ ansible-playbook -i hosts vyos-leaf-show.yaml
```

vyosと新ノードEOSパラメータ比較




```
arista@ubuntu:~/ansible$ cat host_vars/vEOS3
interfaces:
 - name: Ethernet 1
   description: toLeaf
   address: 10.100.1.6
   mask: 30
 - name: Ethernet 3
   description: toServer
   address: 10.20.2.1
   mask: 24
 - name: Loopback 0
   description: router id
   address: 10.255.255.5
   mask: 32
bgpconf:
 - localas: 64513
   nbrip: 10.100.1.5
   remoteas: 65000
```

```
arista@ubuntu:~/ansible$ cat host_vars/vyos
interfaces:
 - name: ethernet eth1
   description: toLeaf
   address: 10.100.1.6
   mask: 30
 - name: ethernet eth2
   description: toServer
   address: 10.20.2.1
   mask: 24
 - name: loopback lo
   description: router id
   address: 10.255.255.5
   mask: 32
bgpconf:
 - localas: 64513
   nbrip: 10.100.1.5
   remoteas: 65000
```

EOS設定追加playbook

```
arista@ubuntu:~/ansible$ vi hosts
arista@ubuntu:~/ansible$ ansible-playbook -i hosts eos-leaf.yaml

PLAY [EOS-leaf] ****************************************************************************************************************************************

TASK [Gathering Facts] *********************************************************************************************************************************
ok: [vEOS2]
ok: [vEOS3]

TASK [debug] *******************************************************************************************************************************************
ok: [vEOS2] => {
    "inventory_hostname": "vEOS2"
}
ok: [vEOS3] => {
    "inventory_hostname": "vEOS3"
}

TASK [configure interface] *****************************************************************************************************************************
 [WARNING]: argument transport has been deprecated and will be removed in a future version

changed: [vEOS3]
changed: [vEOS2]

TASK [configure bgp] ***********************************************************************************************************************************
ok: [vEOS2]
changed: [vEOS3]

TASK [configure NTP] ***********************************************************************************************************************************
changed: [vEOS3]
changed: [vEOS2]

PLAY RECAP *********************************************************************************************************************************************
vEOS2                      : ok=5    changed=2    unreachable=0    failed=0
vEOS3                      : ok=5    changed=3    unreachable=0    failed=0

arista@ubuntu:~/ansible$ ansible-playbook -i hosts eos-leaf-show.yaml
```

ベンダー間差分

```
arista@ubuntu:~/ansible$ cat host_vars/vEOS3
interfaces:
 - name: Ethernet 1
   description: toLeaf
   address: 10.100.1.6
   mask: 30
 - name: Ethernet 3
   description: toServer
   address: 10.20.2.1
   mask: 24
 - name: Loopback 0
   description: router id
   address: 10.255.255.5
   mask: 32
bgpconf:
 - localas: 64513
   nbrip: 10.100.1.5
   remoteas: 65000
```

インターフェース名など
host環境変数で吸収

コマンド違いなど
jinja2テンプレートで吸収

```
arista@ubuntu:~/ansible$ cat host_vars/vyos
interfaces:
 - name: ethernet eth1
   description: toLeaf
   address: 10.100.1.6
   mask: 30
 - name: ethernet eth2
   description: toServer
   address: 10.20.2.1
   mask: 24
 - name: loopback lo
   description: router id
   address: 10.255.255.5
   mask: 32
bgpconf:
 - localas: 64513
   nbrip: 10.100.1.5
   remoteas: 65000
```

```
arista@ubuntu:~/ansible$ cat templates/eoint.j2
{% for eointf in interfaces %}
interface {{ eointf.name }}
   description {{ eointf.description }}
   ip address {{ eointf.address }}/{{ eointf.mask }}
{% endfor %}
arista@ubuntu:~/ansible$ cat templates/eorouting.j2
{% for eorouting in bgpconf  %}
 switchport default mode routed
 ip routing
 router bgp {{ eorouting.localas }}
   neighbor {{ eorouting.nbrip }} remote-as {{ eorouting.remoteas }}
   redistribute connected
{% endfor %}
```

```
arista@ubuntu:~/ansible$ cat templates/vyint.j2
{% for vyint in interfaces %}
set interfaces {{ vyint.name }} address '{{ vyint.address }}/{{ vyint.mask }}'
set interfaces {{ vyint.name }} description '{{ vyint.description }}'
{% endfor %}
arista@ubuntu:~/ansible$ cat templates/vyrouting.j2
{% for vyrouting in bgpconf  %}
set protocols bgp {{ vyrouting.localas }} neighbor {{ vyrouting.nbrip }} remote-as {{ vyrouting.remoteas }}
set protocols bgp {{ vyrouting.localas }} redistribute 'connected'
{% endfor %}
```

Ansible

- 設定コマンドなどはjinjaテンプレートで吸収可能
- トランスポートなどは各group_varsで定義する事が可能
- インターフェース記述違いなどもhost_varsのパラメータとして定義すれば良い

	大規模クラウド事業者	先端企業 / サービスプロバイダー	従来の企業 / サービスプロバイダー
1人のエンジニアがメンテナンスするネットワーク機器数	10,000	1,000	100
自動化ツール	自前で作成	DevOpsツール	ベンダーソリューション
各ツールのポジション (個人的な感想)			

ARISTA

NAPALM

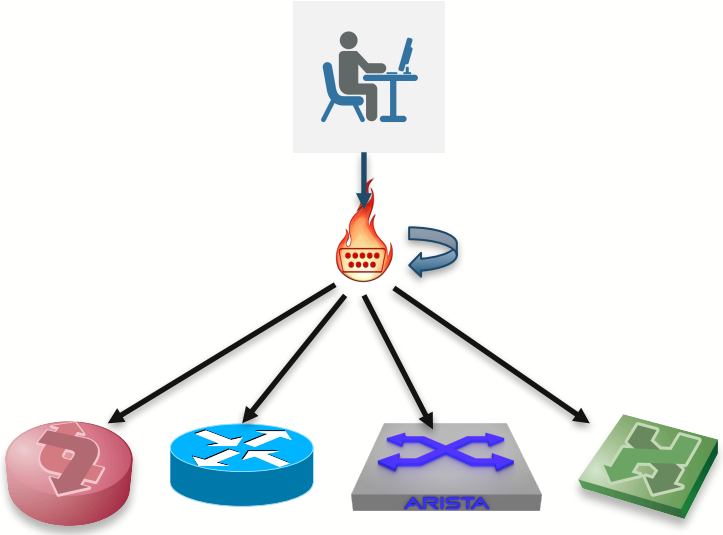
Network Automation and Programmability Abstraction Layer with Multivendor support

- Pythonで書かれたマルチベンダー環境での運用ツール
- NAPALM側で差分を吸収
- マルチベンダー間での統一されたAPI、機器リプレースやコンフィグ入れ替えをスムーズに
- AnsibleやSaltstackに対応<–CloudFlareはSaltstackで運用？
  - https://2017.apricot.net/assets/files/APIC674/apricot%202017%20network%20automation%20with%20salt%20and%20napalm%20mircea%20ulinic%20cloudflare_1488177661.pdf
- サポート機器

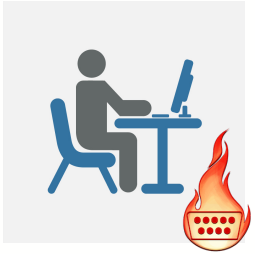
Arista EOS	IBM
Cisco IOS	Juniper JunOS
Cisco IOS-XR	Mikrotik RouterOS
Cisco NX-OS	Palo Alto NOS
Fortinet Fortios	Pluribus
	Vyos

NAPALMが実現してくれる事

- 運用コマンドの共通化
- バックエンドではpyeapi/junos-eznc/netmikoなどを活用
- replace/rollbackなどのコンフィグ操作のコマンドはサポートしているが、コンフィグテンプレートは少ない
  - https://napalm.readthedocs.io/en/latest/support/index.html#available-configuration-templates
  - 実運用には十分かも？
  - NAPALMで作るネットワークオペレーション自動化への道のり



再びデモ環境

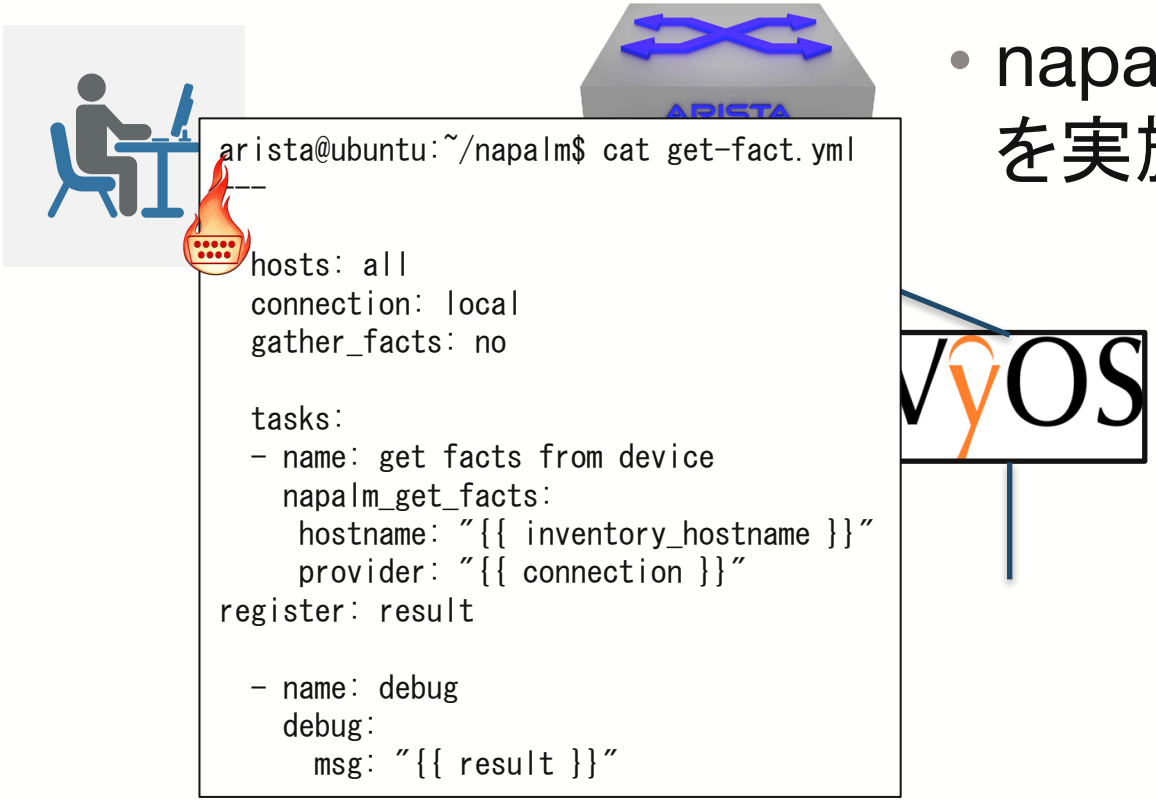


- varでosを指定する
- ビデオ https://youtu.be/PNjYcl4BR3U

```
arista@ubuntu:~/napalm$ more group_vars/EOS-leaf
   connection:
    host: "{{ inventory_hostname }}"
    username: admin
    password: admin
    tranport: eapi
    use_ssl: false
    authorize: true
    dev_os: "eos"
```

```
arista@ubuntu:~/napalm$ more group_vars/vyos-leaf
   connection:
    host: "{{ inventory_hostname }}"
    username: admin
    password: admin
    use_ssl: false
    authorize: true
    dev_os: vyos
```

napalm_get_facts



- napalmのget factsを実施

napalm_get_facts

- 全てのベンダーの機器から情報を収集

```
arista@ubuntu:~/napalm$ ansible-playbook -i hosts get-fact.yml

PLAY [all] *********************************************************************************************************************************

TASK [get facts from device] ***************************************************************************************************************
ok: [vEOS2]
ok: [vEOS1]
ok: [vyos]

TASK [debug] *******************************************************************************************************************************
ok: [vEOS1] => {
    "msg": {
        "ansible_facts": {
            "facts": {
```

napalm_get_facts


```
ok: [vEOS2] => {
    "msg": {
        "ansible_facts": {
            "facts": {
                "fqdn": "vEOS2",
                "hostname": "vEOS2",
                "interface_list": [
                    "Ethernet1",
                    "Ethernet2",
                    "Ethernet3",
                    "Loopback0",
                    "Management1"
                ],
                "model": "vEOS",
                "os_version": "4.18.0F-4224134.4180F",
                "serial_number": "",
                "uptime": 5061,
                "vendor": "Arista"
            }
        },
        "changed": false
    }
}
```



```
ok: [vyos] => {
    "msg": {
        "ansible_facts": {
            "facts": {
                "fqdn": "",
                "hostname": "None",
                "interface_list": [
                    "eth2",
                    "eth1",
                    "eth0",
                    "lo"
                ],
                "model": "VirtualBox",
                "os_version": "1.1.6",
                "serial_number": "0",
                "uptime": 5052,
                "vendor": "VyOS"
            }
        },
        "changed": false
    }
}
```


- 同じデータフォーマットで情報が取れている

arptable

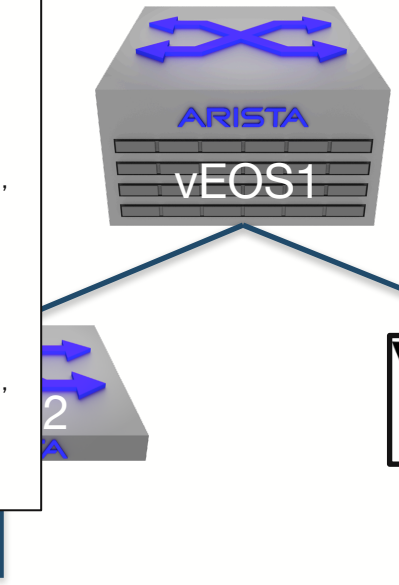

```
ok: [vEOS2] => {
    "msg": {
        "ansible_facts": {
            "arp_table": [
                {
                    "age": 0.0,
                    "interface": "Ethernet1",
                    "ip": "10.100.1.1",
                    "mac": "08:00:27:45:50:AF"
                },
                {
                    "age": 0.0,
                    "interface": "Management1",
                    "ip": "10.10.1.101",
                    "mac": "08:00:27:0A:65:0A"
                },
                {
                    "age": 0.0,
                    "interface": "Management1",
                    "ip": "10.10.1.250",
                    "mac": "0A:00:27:00:00:00"
                }
            ],
```



```
ok: [vyos] => {
    "msg": {
        "ansible_facts": {
            "arp_table": [
                {
                    "age": 0.0,
                    "interface": "eth1",
                    "ip": "10.100.1.5",
                    "mac": "08:00:27:45:50:af"
                },
                {
                    "age": 0.0,
                    "interface": "eth0",
                    "ip": "10.10.1.101",
                    "mac": "08:00:27:0a:65:0a"
                }
            ],
```

interface_counters

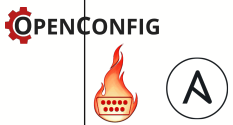

```
"interfaces_counters": {
                "Ethernet1": {
                    "rx_broadcast_packets": 1,
                    "rx_discards": 0,
                    "rx_errors": 0,
                    "rx_multicast_packets": 207,
                    "rx_octets": 52423,
                    "rx_unicast_packets": 209,
                    "tx_broadcast_packets": 1,
                    "tx_discards": 0,
                    "tx_errors": 0,
                    "tx_multicast_packets": 203,
                    "tx_octets": 54469,
                    "tx_unicast_packets": 224
                },
```


```
"interfaces_counters": {
                "eth0": {
                    "rx_broadcast_packets": -1,
                    "rx_discards": 0,
                    "rx_errors": 0,
                    "rx_multicast_packets": 6,
                    "rx_octets": 251141,
                    "rx_unicast_packets": 2541,
                    "tx_broadcast_packets": -1,
                    "tx_discards": 0,
                    "tx_errors": 0,
                    "tx_multicast_packets": -1,
                    "tx_octets": 680406,
                    "tx_unicast_packets": 3078
                },
```

NAPALM

- 多くの機器に対応
- AnsibleやSaltstackと連携する事でワークフローなども実現可能
- 共通のデータフォーマットで情報を取り出す事で管理が容易に？
- ちょっと足りない機能は多い気が。

	大規模クラウド事業者	先端企業 / サービスプロバイダー	従来の企業 / サービスプロバイダー
1人のエンジニアがメンテナンスするネットワーク機器数	10,000	1,000	100
自動化ツール	自前で作成	DevOpsツール	ベンダーソリューション
各ツールのポジション (個人的な感想)			

まとめ

- YANG/Openconfigは実装および標準化など進んできている
- Ansibleはサポート出来るベンダーも豊富で、jinja2テンプレートや環境変数を使う事でかなり抽象化出来る気がする
- 運用コマンドに関しては主要ベンダーはNAPALMで多くサポートされているはず

聞きたい事

- OpenConfig使いたいトランスポートは？
  - セキュリティ機器との連携ならREST
  - Telemetry統合ならgNMIも
- そもそも設定の抽象化ってあんまり要らない？
  - 運用コマンドの抽象化の方が欲しいもの？
- Ansibleもっと上手いやり方ある？
- OSSは自由度が高すぎて、ベンダーツールは完成度が高い・Ansibleはベンダーのガバナンスが上手く効いてるOSSの気がする

ARISTA