

# どう使う？データセンターネットワーク最前線 概要編

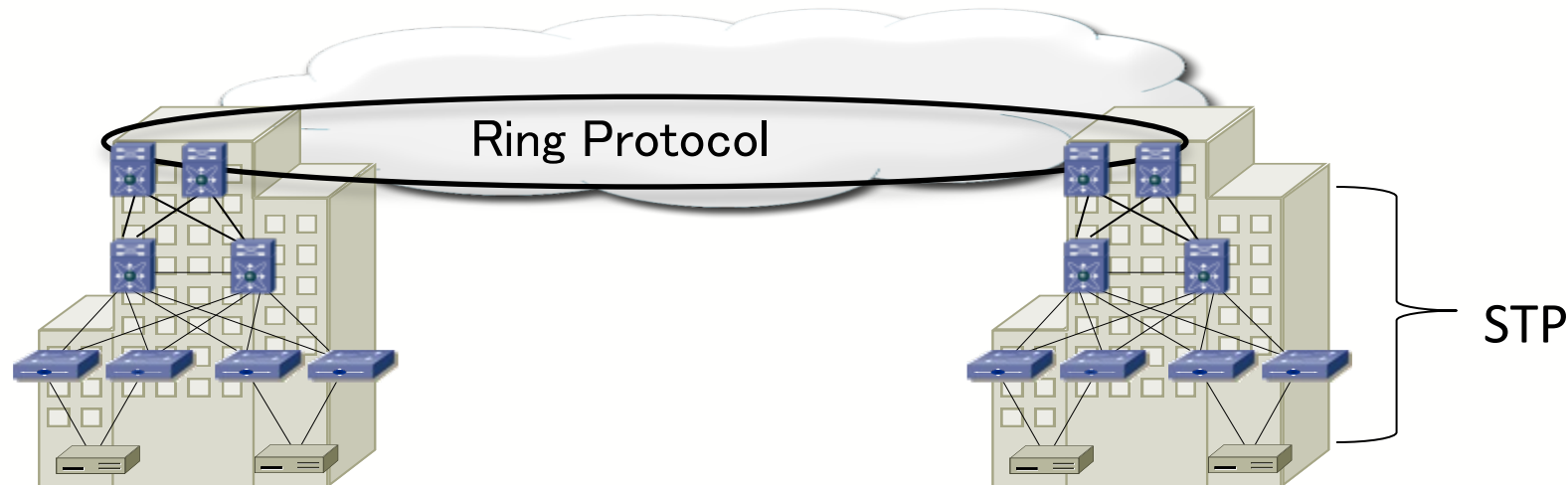
Shishio Tsuchiya  
shtsuchi@arista.com

# Agenda

- **データセンターネットワークデザイン**
- ZTP(ゼロタッチプロビジョニング)
- Ansible-コンフィグ、作業内容の雛形化-
- テレメトリー

# データセンターネットワークデザイン

# 2011年ごろのデータセンターネットワーク

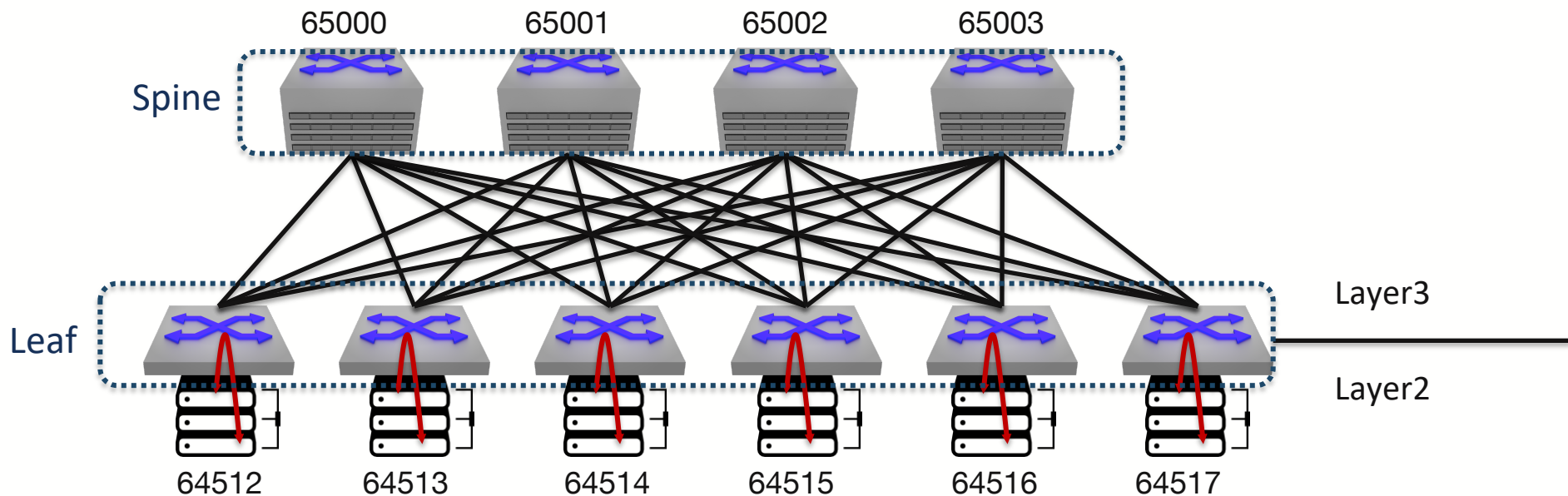


- フラットなレイヤー2のネットワークデザイン
- データセンター内ではスパンニングツリーが使用
- ベンダー独自のリングプロトコル (またはG.8032)がデータセンター間で使用される
- VMライブマイグレーションは必要 (GARPによって移動を通知)
- VLANがユーザ毎にアサインされる

# 問題点

- データセンター間/データセンター内
  - VLANスケーラビリティ > 4K
  - MACアドレステーブルのスケーラビリティ
  - VMライブマイグレーションや簡単に使うためのブロードキャストドメインの拡張
  - East / Westトラフィック帯域の増加
  - 高速収束
  - 自動化/オーケストレーション
- データセンター間
  - 要求に応じた帯域増強
  - ベンダーロックイン技術からの解放
  - 柔軟性のあるトポロジーデザイン
  - トラフィックエンジニアリング
  - BUM(Broadcast/Unknown unicast/Multicast) トラフィックの最適化
- ゲートウェイ
  - ARP/NDPスケーラビリティ
  - IETF ARMD(Address Resolution for Massive numbers of hosts in the Data center) Groupは一つの informational RFCを発行 [RFC6820 Address Resolution Problems in Large Data Center Networks](#)

# 大規模データセンタールーティングでのBGPの使用



- スケールアウトするClosデザイン
- 安定した標準的なBGPプロトコルをToR/Leafスイッチに使用
- 安定性にフォーカスし、VMモビリティはラック内に留める

# このネットワークデザインにおける特別なBGPの要件

- AS\_PATH Multipath Relax

- Allow AS In

- 重複したAS番号を使用する為の機能

- Fast eBGP Fall-over

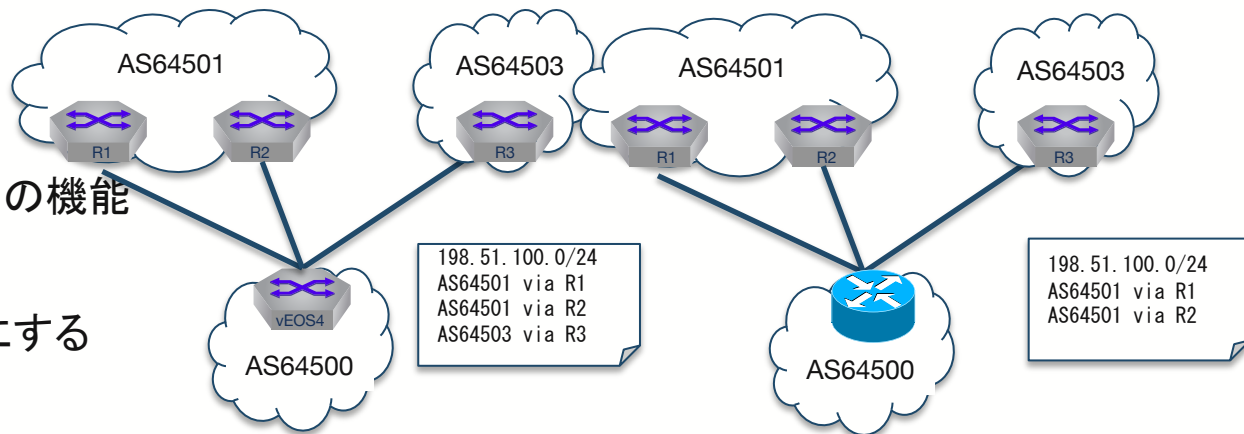
- 高速コンバージェンスを可能にする

- Remove Private AS

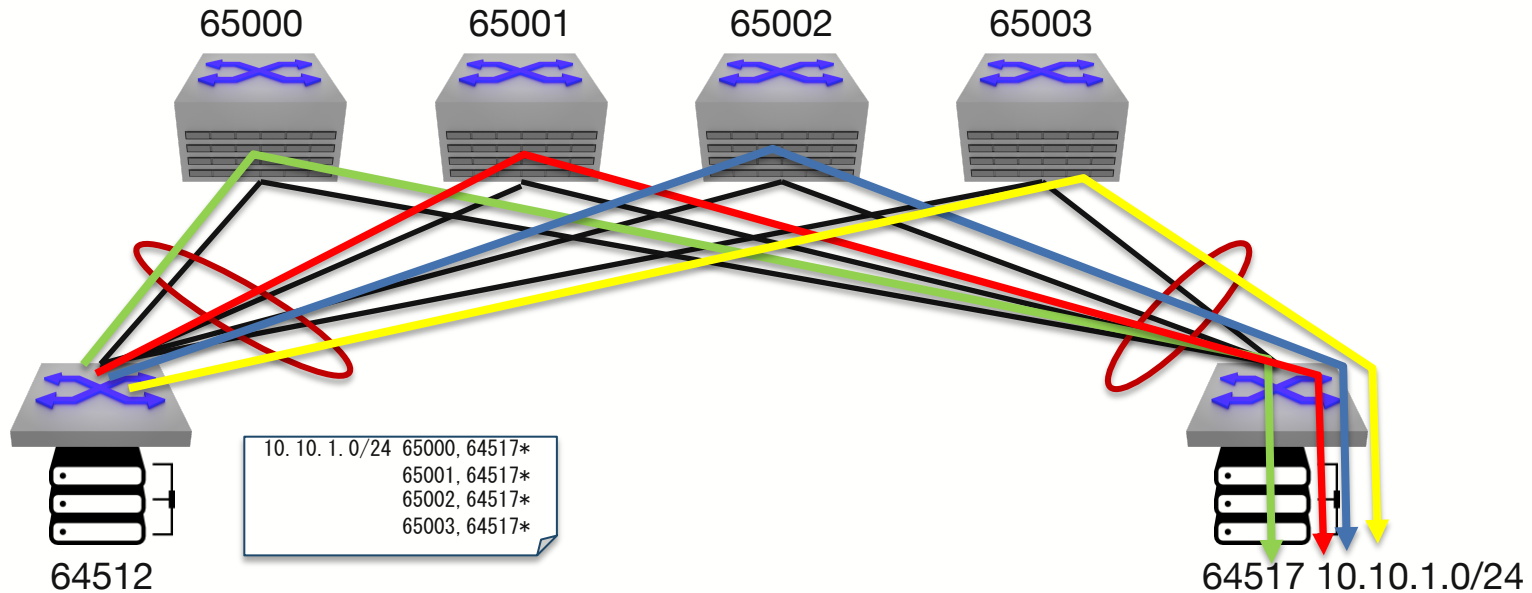
- 2バイトのプライベートAS番号 (64512-65534)だけでなく、4バイトのプライベートAS番号 (4200000000 - 4294967294) もエッジで除去

- [RFC6996 Autonomous System \(AS\) Reservation for Private Use](#)

- 全て既存の技術を使用**



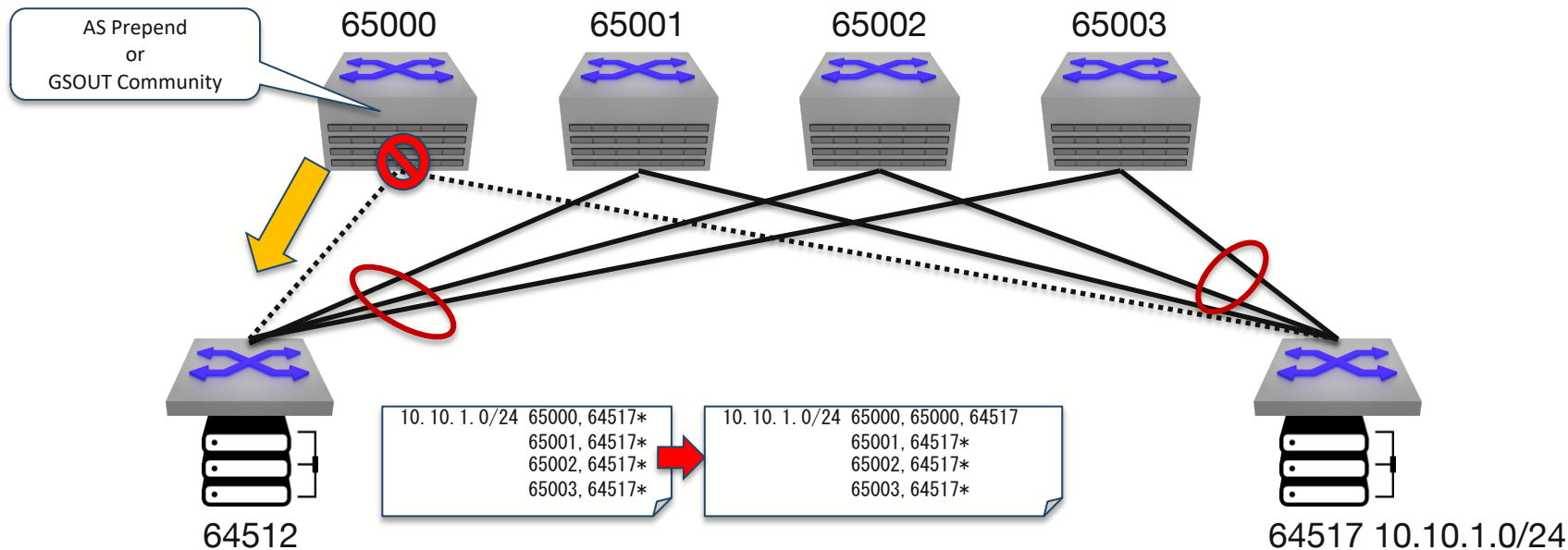
# IP ECMPの利点



- ECMPで全てのLeaf-Spineリンクを使用する
- それぞれのフローはECMPハッシュにてバランスされる(既の実装されている)
- ルーティングパスはBGPパス属性により可視化される

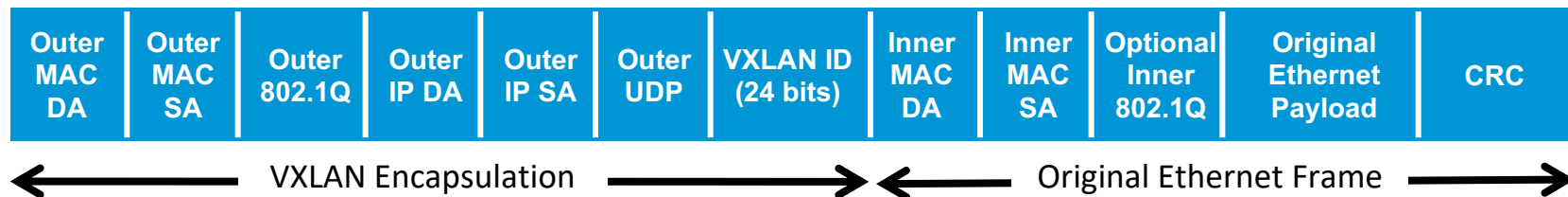


# BGPの利点



- ASパスを追加するもしくはBGPグレースフルシャットダウンコミュニティ ([GSHUT \(0xFFFF0000\) community](#)) を使って簡単にメンテナンス可能

# Virtual eXtensible LAN (VXLAN)



- レイヤー2フレームをIPでカプセルプロトコルを定義
- レイヤー3インフラストラクチャー上でオーバーレイのネットワークを作成する為に使用
- レイヤー2の接続性をユーザに提供

# Virtual eXtensible LAN (VXLAN)フレームフォーマット

- フラグ(8 bits):
  - 有効なVXLANのネットワークIDの場合、Iフラグは1をセットしなければならない。他の7ビット(R)は予約フィールドで送信時に0をセットしなければならない、受信時に無視される
- VXLANセグメントID/VXLANネットワーク識別子 (VNI):
  - これは、通信するVMが配置されている個々のVXLANオーバーレイネットワークを指定するために使用される24ビットの値です。異なるVXLANオーバーレイネットワーク内のVMは互いに通信できません。
- 宛先ポート:
  - IANAはVXLANのポートとして4789をアサインした。このポートをデフォルトの宛先ポートとして使う
- 送信元ポート:
  - UDPソースポート番号は、ロードバランスの際のハッシュの計算に使用される。動的にプライベートポート範囲49152-65535である事が推奨される

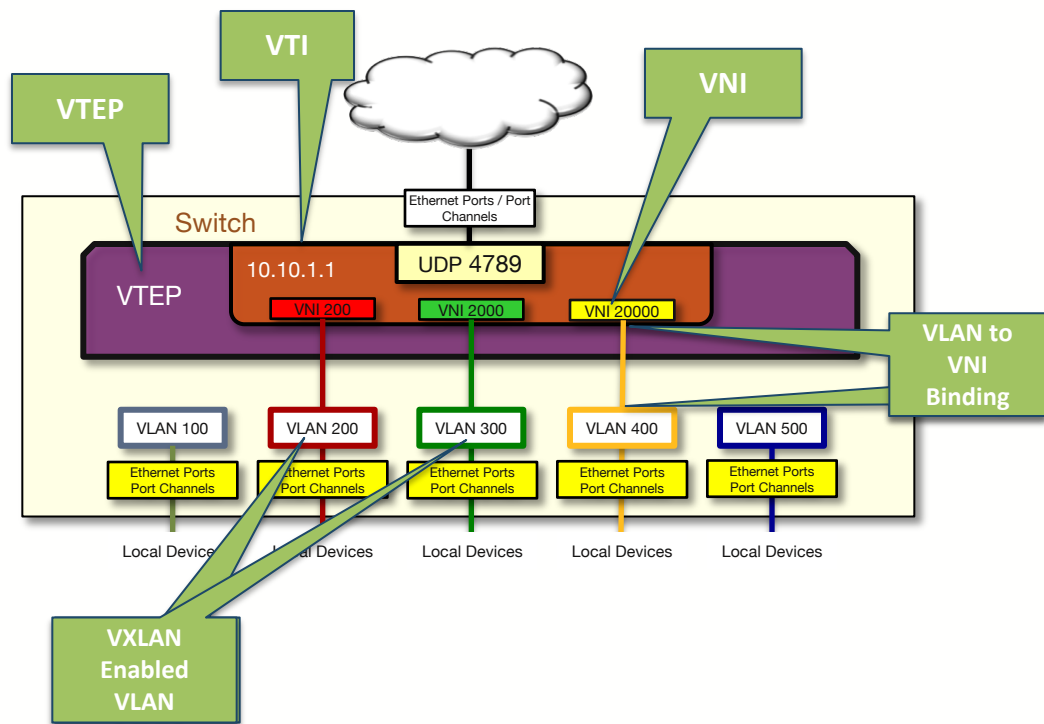
Outer UDP Header:

```
+++++
|           Source Port           |           Dest Port = 4789           |
+++++
|           UDP Length            |           UDP Checksum              |
+++++
```

VXLAN Header:

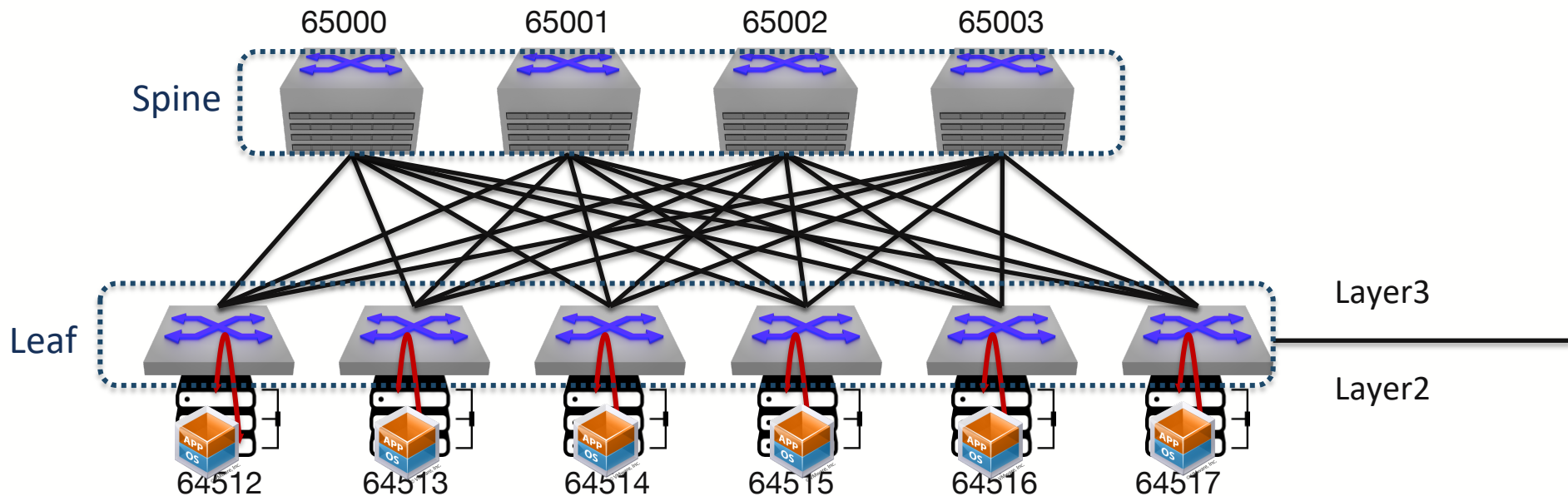
```
+++++
|R|R|R|R|I|R|R|R|           Reserved           |
+++++
|           VXLAN Network Identifier (VNI) |           Reserved           |
+++++
```

# VXLAN用語集



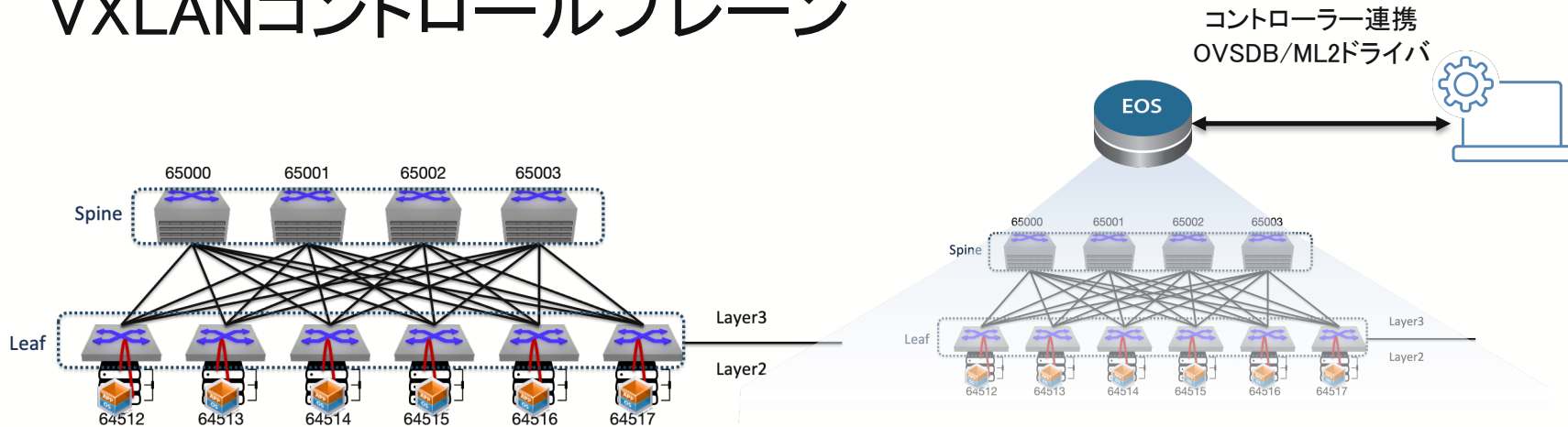
- VTEP – Virtual Tunnel Endpoint  
VXLAN対応のVLANのencap/decapのポイントとして動作するスイッチ
- VTI – Virtual Tunnel Interface  
VXLAN encapされたトラフィックを終端するインターフェース
- VNI – Virtual Network Identifier  
VXLANが通過するVXLANセグメント
- VxLAN Enabled VLAN  
このVLANはVXLANにencapされる  
VLAN IDはローカルだけに意味がある
- VLAN to VNI Bindings  
VXLAN enabled VLANをVNIに紐付けるテーブル

# VXLAN + IP Closデザイン



- データセンターデザインで完全なソリューション
- 標準で安定したBGPプロトコル/広く展開されたVXLANフォーマットを使用
- **特別な要求が中間ノードには存在しない**

# VXLANコントロールプレーン



- MACアドレス配布は?
- VTEPリストは?

## IPマルチキャスト

- VTEPがVNIのIPマルチキャストグループに参加
- Unknown UnicastはIPマルチキャスト経由でVNI内のVTEPに転送
- Flood & Learn/IPマルチキャストのサポートが必要

## Head-End Replication(HER)

- BUMトラフィックはVNI内の各VTEPに複製される
- 複製はIngress VTEPで行われる
- Flood & Learnが行われるがマルチキャストは不要

## HER+SDNコントローラー

- コントローラーは各VTEPにステートを動的に配布
- 動的なMAC配布/VTEPリストの配布
- 冗長性の為のHAクラスタ

## EVPN

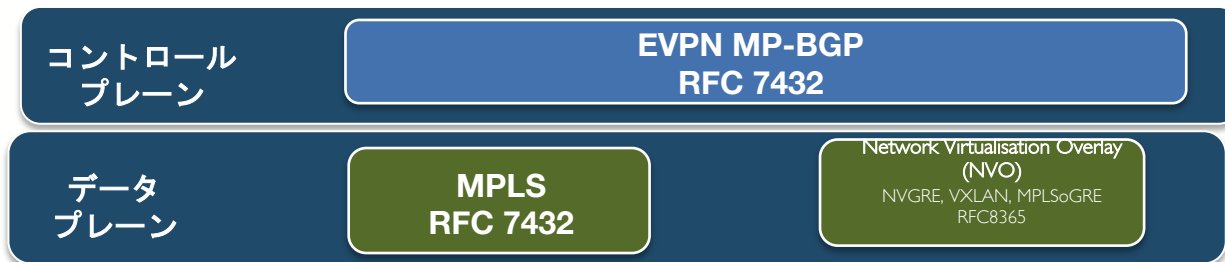
- BGPがローカルなIPとMACのバインディング情報を配布
- HERモデルによるブロードキャストトラフィック処理
- BGPによる動的なMAC配布とVNI学習

# Requirements for Ethernet VPN (EVPN)

- EVPNは既存のイーサネットVPNサービス(VPLS)ではなし得ないオールアクティブのマルチホーミングなどを実現する事を目的に作成された
  - <https://tools.ietf.org/html/rfc7209>
- 1. 冗長性の要求
  1. フローベースのロードバランシング
  2. フローベースのマルチパス
  3. PEノードの地理的冗長性
  4. 最適なトラフィックフォワーディング
  5. 最適なリダンダンシーグループ
  6. マルチホームネットワーク
- 2. マルチキャスト最適化
- 3. プロビジョニングの簡素化
- 4. 新しいサービスインターフェース
- 5. 高速コンバージェンス
- 6. フラッディング抑制
- 7. 柔軟なVPNTポロジーとポリシー
  - L2VPN/L3VPN/VPWS/E-TREE

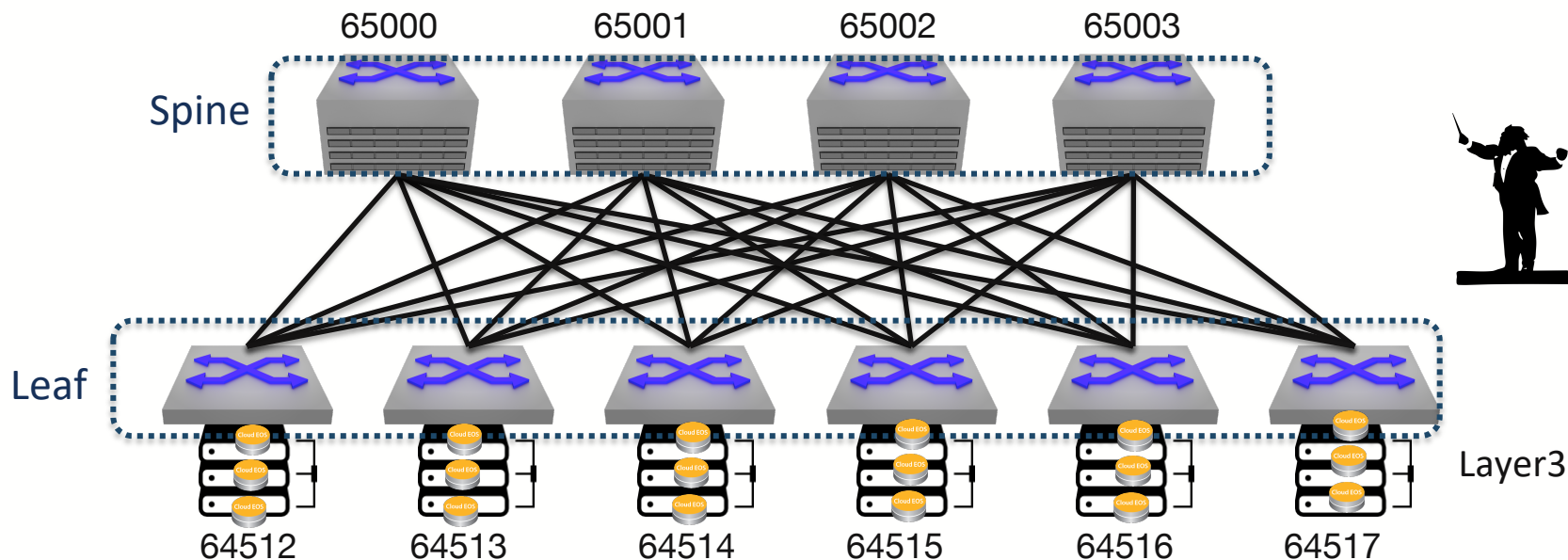
# Ethernet VPN (EVPN) とは？

- **EVPN は RFC 7432で定義される**
  - 同RFC中では、MPLSがデータプレーンとして定義
  - BGPにて新たなアドレスファミリーを設け、MAC/IPとIPプレフィックスを広報可能に
- **データプレーンの選択肢はMPLS と NVO**
  - コントロールプレーンは共有し、ネットワーク仮想化プロトコル等をデータプレーンとして採用





# Extending EVPN and VXLAN to the Host



- さらなるメンテナンス性の向上を求めて、サーバーでBGPを展開する
- EVPN/VXLANなどサーバー上で実現する

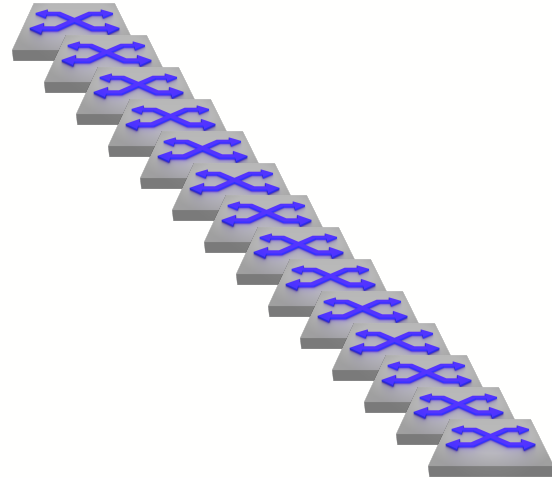
## まとめ

- ネットワークの安定性、柔軟性、拡張性をもとめIP Closデザインが普及した
- レイヤー2は安定したネットワーク上で広く普及したVXLANなどのオーバーレイテクノロジーで構築
- VXLANコントロールプレーンはHERやBGPを使ったEVPNが主流
- サーバーのBGPデーモンを使った展開も増えてきている

# ZTP(ゼロタッチプロビジョニング)

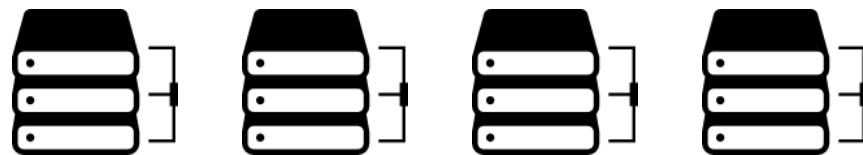
# ZTPとは

- Zero Touch Provisioning
- ユーザが機器にログインせずに設定を完了する事



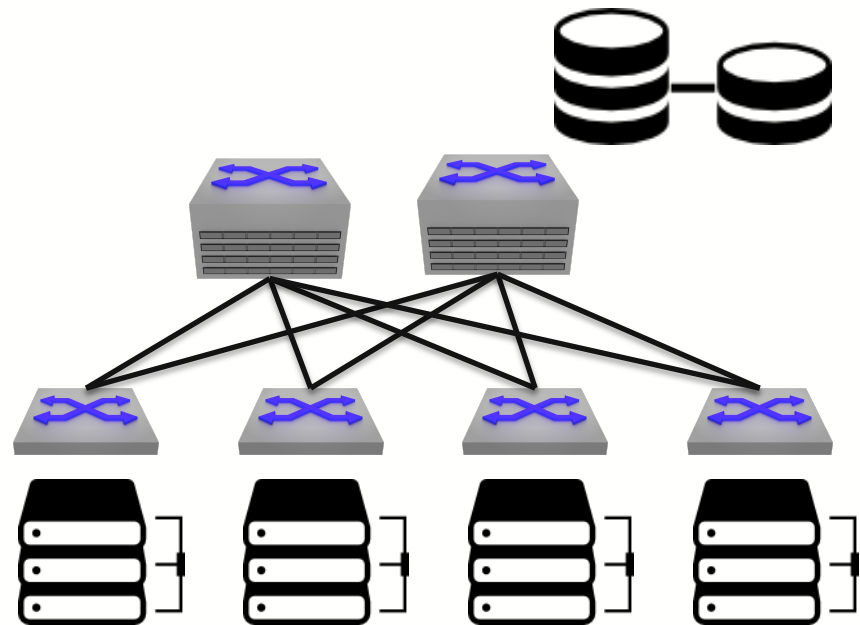
# ネットワークブートの必要性

- 複数のサーバーが存在するデータセンターでは下記の理由でのネットワークブートが有効
  - 物理的に移動する必要が無い
  - HDDが壊れた時のリカバリー
  - 一度に大量にインストールしたい
  - 複数のOSを同じ環境にインストールする



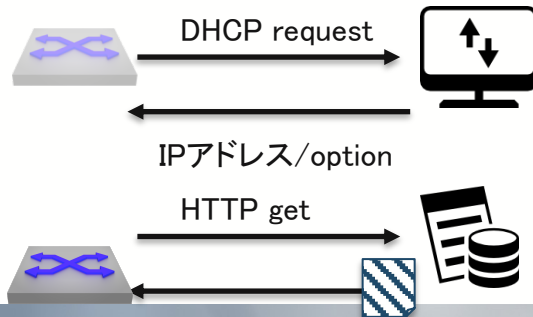
# ネットワーク機器でのネットワークブートの必要性

- サーバーで構築された環境をそのまま使える
- 大量キッティング時間の短縮や現地作業も可能



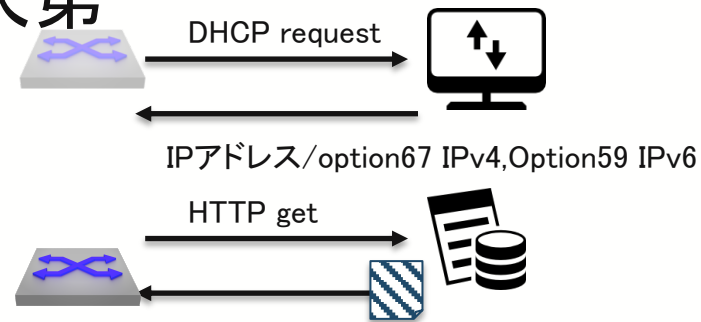
# 必要なもの

- ネットワークブートが可能なブートルoader
  - iPXE(Preboot Execution Environment)
  - ONIE(Open Network Install Environment)
  - Arista EOS(初期状態では全てのポートがDHCPクライアントとして動作する)
- DHCPサーバー
  - IPアドレスおよび起動に必要なパラメーター情報渡す
- TFTP/HTTPサーバー
  - 設定ファイルやイメージを転送する



# ZTPに使えるDHCPオプション

- DHCPv4 Option 67 “BootFile name”
  - <https://www.rfc-editor.org/rfc/rfc2132.html>
- DHCPv6 Option 59 “OPT\_BOOTFILE\_URL”
  - <https://www.rfc-editor.org/rfc/rfc5970.html>
- 使えるオプションはBootローダ次第





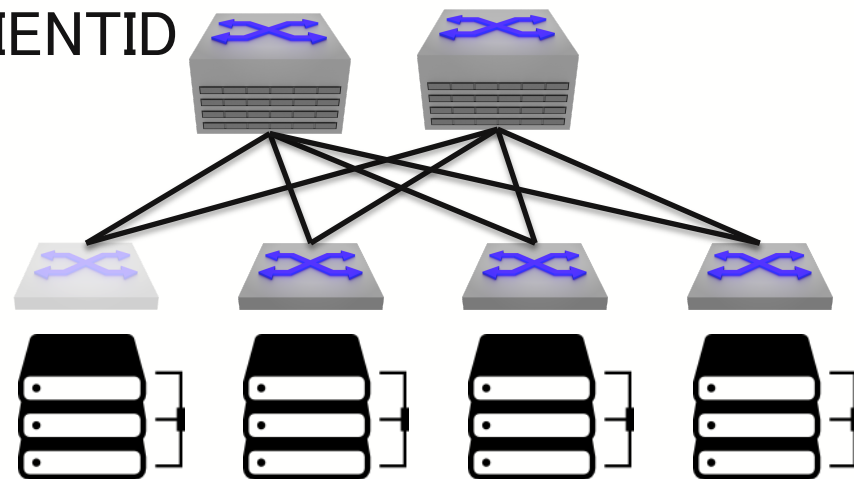
# どの情報で機器固有情報を判断するか

- シリアルナンバー

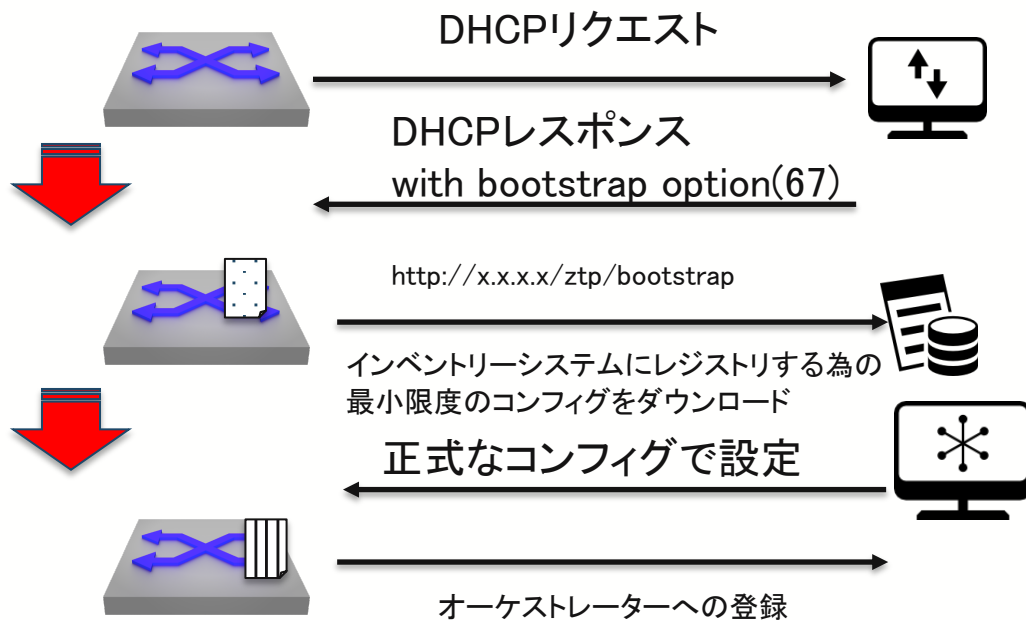
- DHCPv4 Option61 Client-identifier
- MACアドレスが使用される
- <https://www.rfc-editor.org/rfc/rfc2132.html>
- DHCPv6 Option1 OPTION\_CLIENTID
- DUID(DHCP Unique Identifier)
- <https://www.rfc-editor.org/rfc/rfc8415.html>

- LLDPネイバー情報

- <https://ztpserver.readthedocs.io/en/master/config.html#dynamic-provisioning-neighbordb>

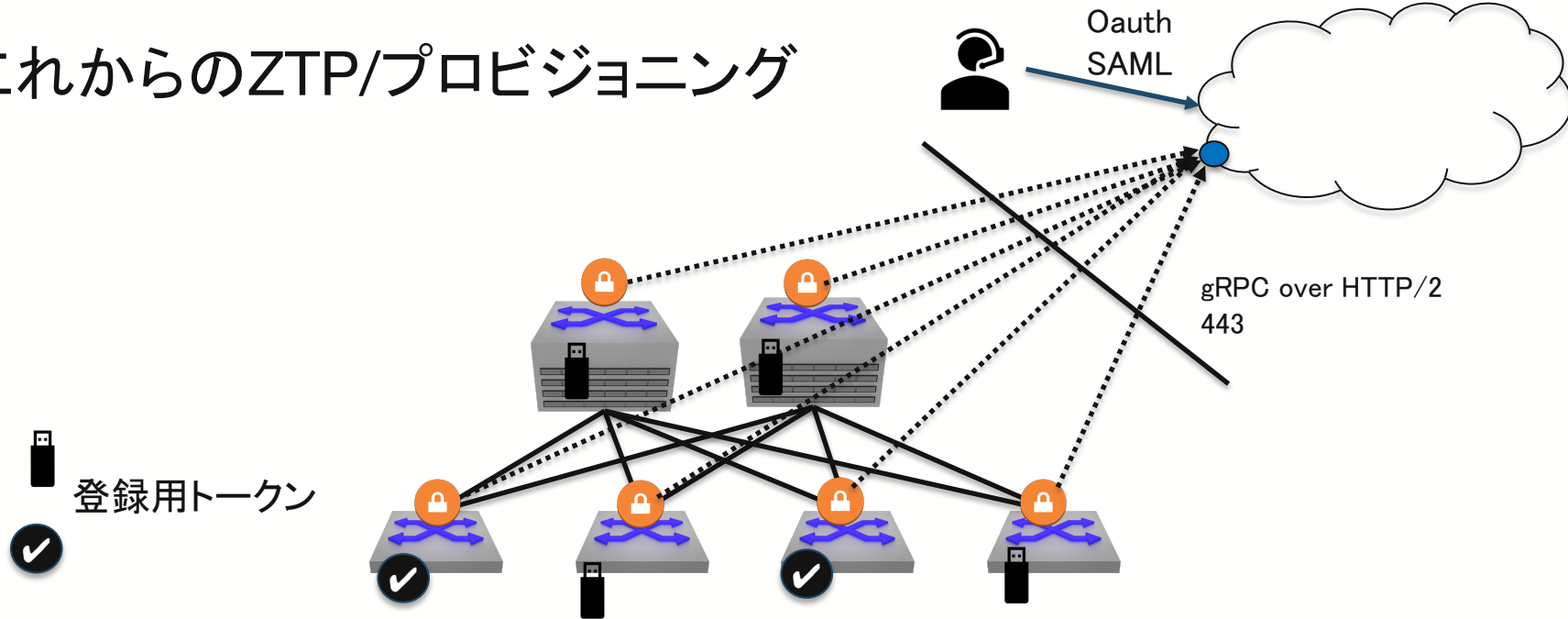


# 実際にクラウドタイタンで使われるZTP



- テンポラリファイルもしくはscriptでインベントリーシステムに登録-ZTP
- 機器の確認や承認など済んだ後に正式なコンフィグを設定
- その後はオーケストレーターに正常登録される

# これからのZTP/プロビジョニング



- 管理システムの運用をクラウドなどに任せる
- gRPC over HTTP/2上でテレメトリーやAPIによる設定が可能
- サービスへのユーザアクセスはOAuthやSAML
- デバイスがサービスに登録する時にトークンを必要
- ZTPを実施する為には機器自体にトークンを出荷時に登録するか、USBにトークン情報を入れる
- トークン情報とともにサービスに登録していく

# Ansible-コンフィグ、作業内容の雛形化-

# Ansibleとは

- Redhatが提供するオープンソースの構成管理ツール
  - <https://docs.ansible.com/>
- Pythonベース
- SSHやAPIを操作
- YAMLで読みやすい
- エージェントレス
- 非常に強力
- 活発なオープンソースコミュニティ
- ネットワーク機器のみならず多くの機器やクラウドに対応
  - <https://docs.ansible.com/ansible/latest/collections/arista/eos/index.html#plugins-in-arista-eos>



ANSIBLE



# YAML(YAML Ain't a Markup Language)

- 人が読みやすいデータ構造
- ファイル拡張子はymlもしくはyaml
- スペースやインデントが特徴
- YAMLで使われるデータ構造
  - List
  - Dictionally
  - Nested

List

```
---  
  
Spine:  
- R1  
- R2
```

Dictionally

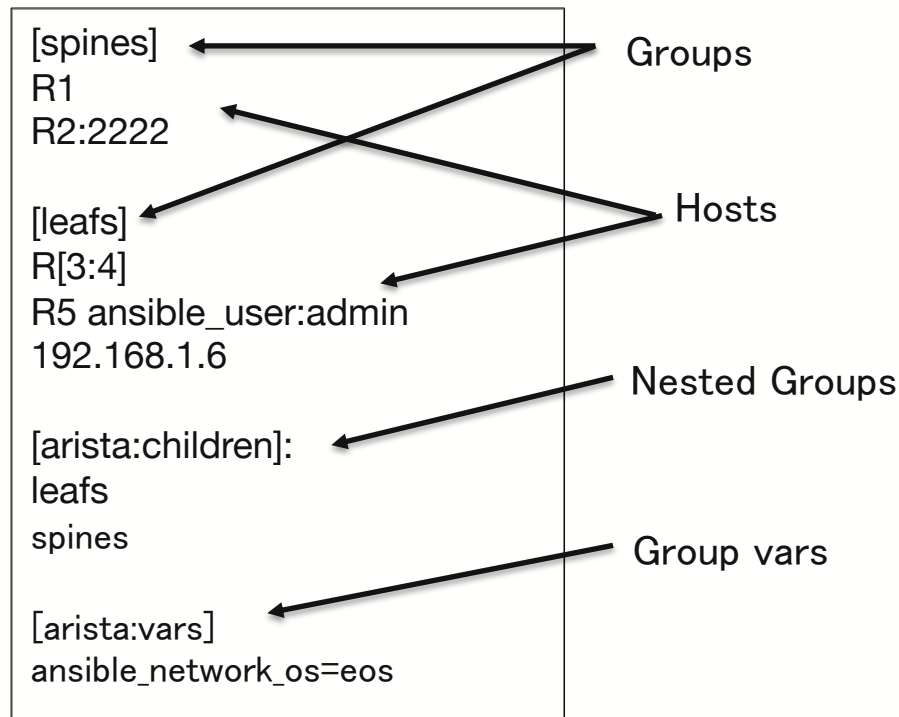
```
---  
  
R1:  
  MgmtIP: 192.168.1.1  
  Loopback: 10.255.255.1  
  AS: 65001
```

List of dictionally

```
---  
  
Spine:  
- R1  
  MgmtIP: 192.168.1.1  
  Loopback: 10.255.255.1  
  AS: 65001  
- R2  
  MgmtIP: 192.168.1.2  
  Loopback: 10.255.255.2  
  AS: 65002
```

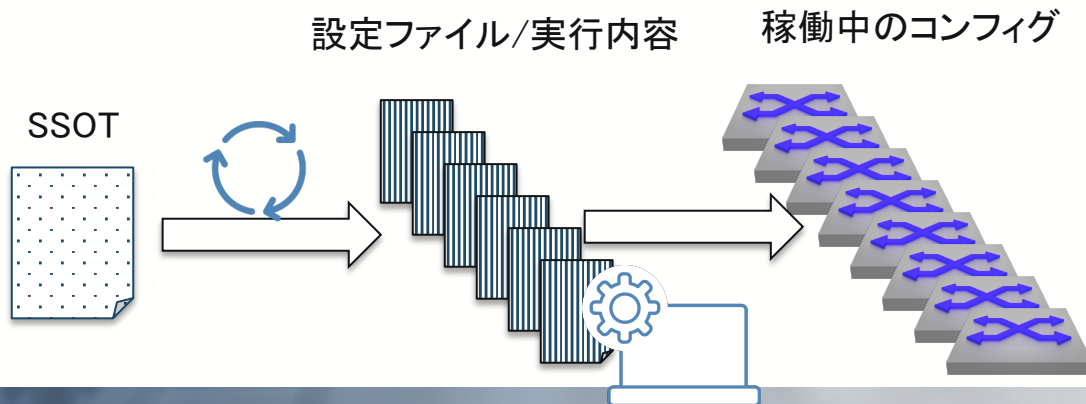
# Inventory(インベントリ)

- Inventoryファイルでホストやグループを指定
- ファイル名は通常hosts
- INI形式
- グループ毎の記載
- 階層化も可能
- 変数も指定可能



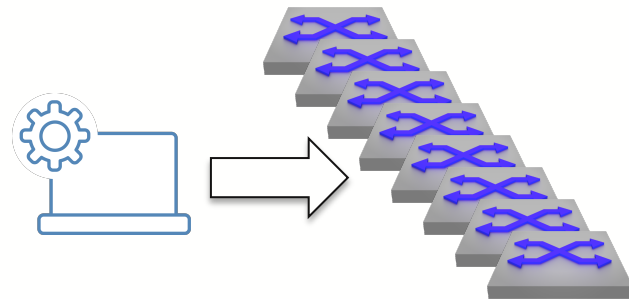
# 信頼できる唯一の情報源 (SSOT: Single Source of Truth)

- 情報システムの設計と理論においては、すべてのデータが1か所でのみ作成、あるいは編集されるように、情報モデルと関連するデータスキーマとを構造化する方法である。<https://ja.wikipedia.org/wiki/信頼できる唯一の情報源>
- 自動化の最初のステップでこれが課題になる。あなたのシステムのSSOTはなんですか？参照可能で構造化されていますか？
- AnsibleではこれがYAMLになる





# Ansible Ad-hocコマンド



```
arista@ip-10-33-6-141:~/ATD$ ansible all -i hosts -m raw -a "show version" -u arista -k
SSH password:
192.168.0.14 | SUCCESS | rc=0 >>
```

- 1つのタスクを1つ以上のノードに対して実行する
- 情報収集などの時には便利

all コマンドを実行するgroupを指定する/allですべてのグループ

-i hosts インベントリーファイルを指定(hosts)

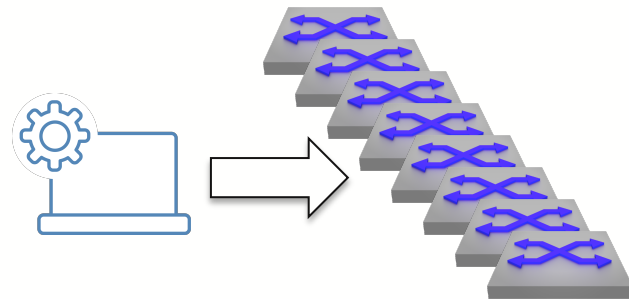
-m raw デバイスとの接続にAnsibleのRaw SSHモジュールを使用する

-a “show version” Ad-Hocコマンドのオプション。この場合は”show version”

-u arista SSHログイン名

-k パスワードプロンプト

# Ansible Ad-hocコマンド[実行結果]



```
arista@ip-10-33-6-141:~/ATD$ ansible all -i hosts -m raw -a "show version" -u arista -k
SSH password:
192.168.0.14 | CHANGED | rc=0 >>
  vEOS
Hardware version:
Serial number:      leaf1
System MAC address: 04a8.0159.c467

Software image version: 4.23.0.1F
Architecture:          i686
Internal build version: 4.23.0.1F-13860745.42301F
Internal build ID:     6ald05a3-2754-4ecf-b553-fc15f98cfe62

Uptime:                0 weeks, 0 days, 1 hours and 13 minutes
Total memory:          4007004 kB
Free memory:           2982560 kB

Shared connection to 192.168.0.14 closed.
```

# Ansible Playbook

- YAMLで記述
- 主な内容
  - Play
  - Task
  - Module
  - パラメータ
- シーケンシャルに実行

# プレイブック

右サイド！すぐ上がって  
10番！裏側に走って

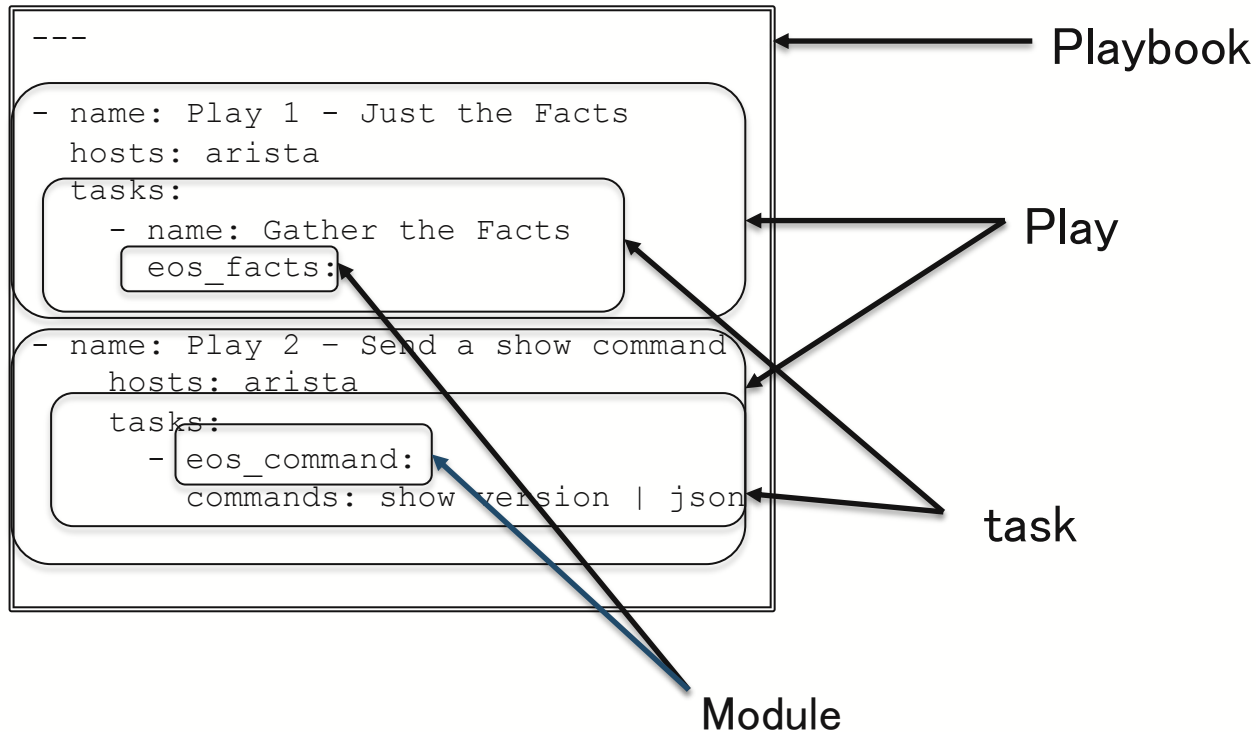
hosts:右サイド、10番

group:右サイド(7番と5番)

tasks:すぐ上がって、裏側走って



# Playbookの中身



# Ansible Network Module

- 標準モジュールとして提供

[http://docs.ansible.com/ansible/list\\_of\\_network\\_modules.html](http://docs.ansible.com/ansible/list_of_network_modules.html)

- 各ベンダー機器毎のモジュールを提供
  - eos\_command – Arista EOSの為の管理コマンド実行
  - eos\_config – Arista EOSの設定コマンド
  - eos\_eapi – Arista EOS eAPI管理と設定コマンド
  - eos\_facts – Arista EOSを実行してるリモートデバイスの情報を収集

# Ansible Playbook

## Leaf1(192.168.0.14)にvlan500を追加するplaybookを作成する

```
arista@ip-10-33-6-141:~/ATD$ vi vlan.yml
---
- name: Add a VLAN
  hosts: 192.168.0.14
  gather_facts: no
  connection: local

  tasks:
    - eos_config:
        lines:
          - name foo
        parents: vlan 500
        provider: "{{ eos_connection }}"
```

# Ansible Playbook

```
---  
- name: Add a VLAN  
  hosts: 192.168.0.14  
  gather_facts: no  
  connection: local
```

name: taskの名前

hosts: Playbookを実行するターゲット

gather\_facts: ターゲット情報を事前を取得するかを指定する

connection: 実行するデバイスから見て、どこでタスクを実行するか



# Ansible Playbook

```
tasks:
  - eos_config:
      lines:
        - name foo
      parents: vlan 500
      provider: "{{ eos_connection }}"
```

task: task設定

eos\_config: Ansible eos\_configモジュールを使用

[https://docs.ansible.com/ansible/latest/modules/eos\\_config\\_module.html#eos-config-module](https://docs.ansible.com/ansible/latest/modules/eos_config_module.html#eos-config-module)

lines: eos\_configの行(ライン)を指定する

-name foo: 実際に設定する行

parents: vlan 500 設定するラインの親の階層

provider: “{{ eos\_connection }}” 接続するため変数

# Jinja2



Jinja2はPythonの為の最新のテンプレートエンジン

継承的なテンプレート使用

デバックが容易

変更可能なSyntaxなどの機能によりWeb開発の場面で良く使われている

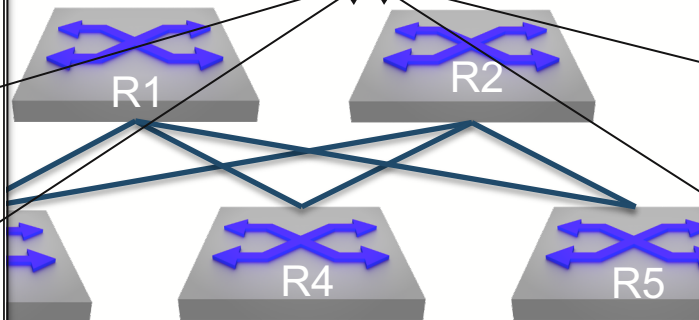
Ansibleでは変数読み込みなど動的な設定表現にJinja2テンプレートが使われる  
ループ処理などが可能なため、設定読み込みなどに便利

```
<title>{% block title %}{% endblock %}</title>
<ul>
{% for user in users %}
  <li><a href="{ { user.url } }">{ { user.username } }</a></li>
{% endfor %}
</ul>
```

[http://docs.ansible.com/ansible/latest/playbooks\\_templating.html](http://docs.ansible.com/ansible/latest/playbooks_templating.html)  
<http://jinja.pocoo.org/docs/2.10/>

# テンプレートの作り方

## グローバルテンプレート



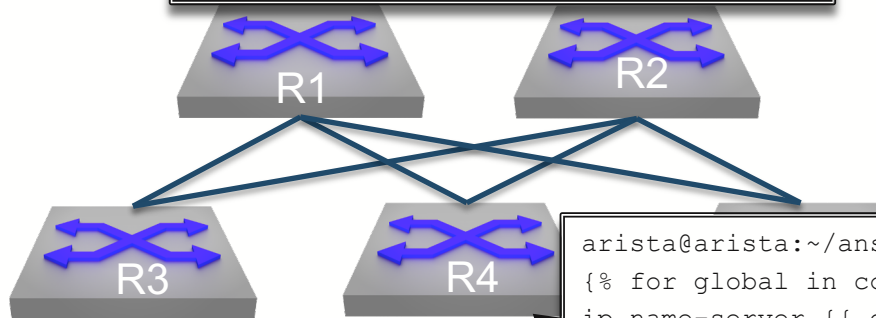
## インターフェース テンプレート

```
R1#show running-config
! Command: show running-config
! device: R1 (vEOS, EOS-4.18.1F)
!
! boot system flash:/vEOS-lab-4.18.1F.swi
!
switchport default mode routed
!
transceiver qsfp default-mode 4x10G
!
hostname R1
ip name-server vrf default 8.8.8.8
!
interface Ethernet1
  description toR3
  no switchport
  ip address 10.10.1.1/30
!
interface Ethernet2
  description toR4
  no switchport
  ip address 10.10.1.5/30
!
interface Ethernet3
  description toR5
  no switchport
  ip address 10.10.1.9/30
!
```

```
R2#show running-config
! Command: show running-config
! device: R2 (vEOS, EOS-4.18.1F)
!
! boot system flash:/vEOS-lab-4.18.1F.swi
!
switchport default mode routed
!
transceiver qsfp default-mode 4x10G
!
hostname R2
ip name-server vrf default 8.8.8.8
!
!
!
interface Ethernet1
  description toR3
  no switchport
  ip address 10.10.1.13/30
!
interface Ethernet2
  description toR4
  no switchport
  ip address 10.10.1.17/30
!
interface Ethernet3
  description toR5
  no switchport
  ip address 10.10.1.21/30
!
```

# グローバルパラメーター

```
switchport default mode routed
!  
ip name-server vrf default 8.8.8.8
```



```
arista@arista:~/ansible$ cat templates/global.j2  
{% for global in common %}  
ip name-server {{ global.dnsserver }}  
switchport default mode {{ global.switchport }}  
{% endfor %}
```

```
tasks:  
- name: configure global parameter  
  eos_config:  
    src: global.j2  
    provider: '{{ eos_connection }}'
```

Jinja2テンプレート呼び出し

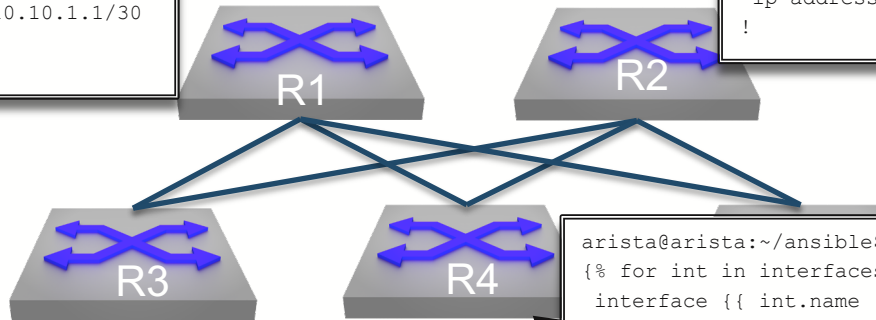
共通項目読み出し

```
arista@arista:~/ansible$ cat group_vars/Spine  
common :  
- dnsserver: 8.8.8.8  
  switchport: routed
```

# インターフェースパラメーター

```
!  
interface Ethernet1  
  description toR3  
  no switchport  
  ip address 10.10.1.1/30  
!
```

```
!  
interface Ethernet1  
  description toR3  
  no switchport  
  ip address 10.10.1.13/30  
!
```



```
arista@arista:~/ansible$ cat templates/int.j2  
{% for int in interfaces %}  
  interface {{ int.name }}  
    description {{ int.description }}  
    ip address {{ int.address }}/{{ int.mask }}  
{% endfor %}
```

```
tasks:  
- name: configure interface  
  eos_config:  
    src: int.j2  
    provider: '{{ eos_connection }}'
```

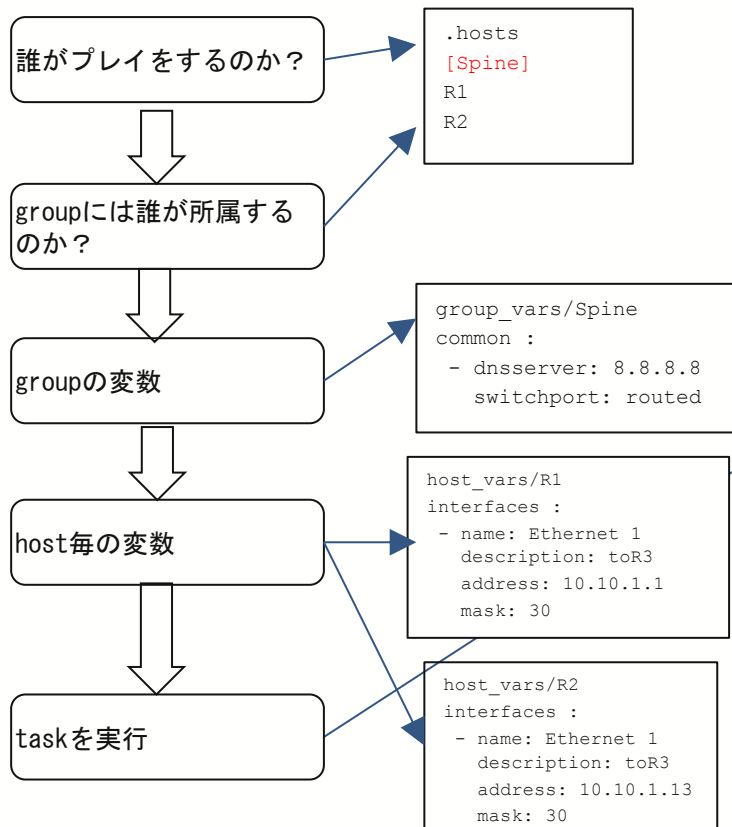
Jinja2テンプレート呼び出し

個別項目読み出し

```
host_vars/R1  
interfaces :  
- name: Ethernet 1  
  description: toR3  
  address: 10.10.1.1  
  mask: 30
```

```
host_vars/R2  
interfaces :  
- name: Ethernet 1  
  description: toR3  
  address: 10.10.1.13  
  mask: 30
```

# Playbook



```
---
- hosts: Spine
  connection: local
  gather_facts: yes

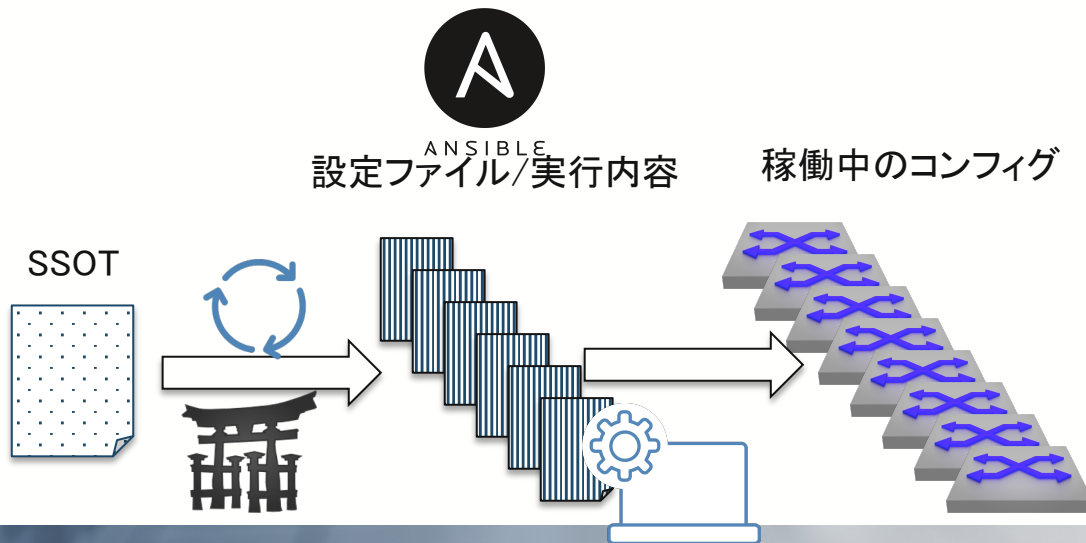
  tasks:
  - name: configure global parameter
    eos_config:
      src: global.j2
      provider: '{{ eos_connection }}'

  - name: configure interface
    eos_config:
      src: int.j2
      provider: '{{ eos_connection }}'

  - name: configure Spine BGP
    eos_config:
      src: spinebgp.j2
      provider: '{{ eos_connection }}'
      save_when: modified
```

# 自動化をすすめる為に

- SSOT(信頼できる唯一の情報源)を作成/まとめる
- アドホックなコマンドや簡単なワンラインのコマンドから始める
- 低リスクで高価値をものを最初に管理する: NTP, syslog, など
- 設定ファイルをテンプレートで活用出来るように考える



# テレメトリー

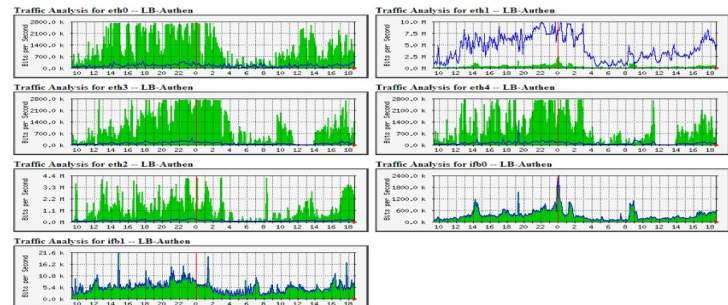


# 従来のネットワーク運用の問題点

1. SNMPだと5分間隔ぐらいでしかデータを確認できない・・

2. Syslogの確認、装置が複数に及ぶとログも膨大に・・

MRTG Index Page

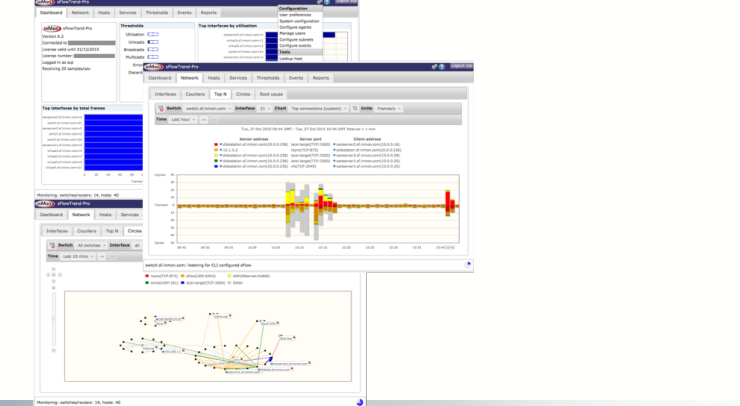
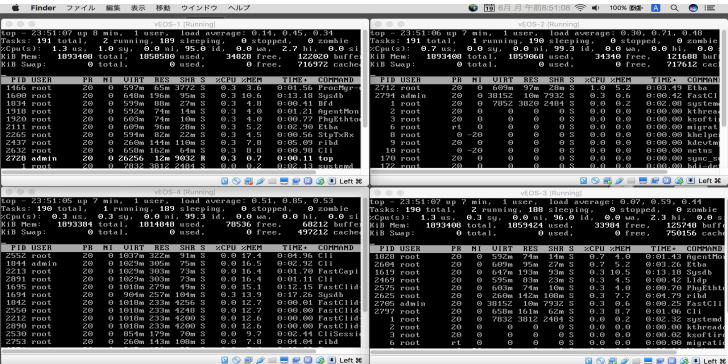


```

1223.txt - 詳細読み
PID=1472 (Mirroring, PID=1504) (Lag, PID=1474) (Acl, PID=1476) (PortSec, PID=1477) (Stp, PID=1480) (TopoAgent, PID=1481) (Ira, PID=1482) (LedPolicy, PID=1483) (Arp, PID=1486) (Icmp0012, PID=1441) (Mlag, PID=1480) (Dns, PID=1408) (Ebra, PID=1507) (Thermostat, PID=1494) (EventMon, PID=1496) (Rlb, PID=1498) (AgentMonitor, PID=1500) (Sysd, PID=1440) (FastClid, PID=1441) (Fru, PID=1442) (Launcher, PID=1441) (Mdp, PID=1508) (TempSnooping, PID=1511) (Smbus, PID=1745) (Mio, PID=1746) (Ssd, PID=1747) (PcBus, PID=1652) (Ssb28, PID=1653) (Lldp, PID=1465) (SuperServer, PID=1466) (PowerManager, PID=1467) (StpTopology, PID=1468) (Aaa, PID=1469) (Fru)
Feb 28 20:41:19 KS003-VM01r ProcMgr-worker: NPROCGR-6-PROCESS_STARTED: 'PowerSupplyDetector' starting with PID=1864 (PID=1439) --
executing /usr/bin/PowerSupplyDetector
Feb 28 20:41:19 KS003-VM01r ProcMgr-worker: NPROCGR-7-WORKER_WARMSTART_DONE: ProcMgr worker warm start done. (PID=1439)
Feb 28 20:41:19 KS003-VM01r Launcher: 39: %LAUNCHER-6-PROCESS_STARTED: Configuring process 'FanDetector' to start in role
'AllSupervisors'
Feb 28 20:41:19 KS003-VM01r Launcher: 40: %LAUNCHER-6-PROCESS_START: Configuring process 'Xcvr' to start in role 'AllCells'
Feb 28 20:41:19 KS003-VM01r Launcher: 41: %LAUNCHER-6-PROCGR_WARMSTART: Initiating warm start of 'ProcMgr (worker)'
Feb 28 20:41:20 KS003-VM01r ProcMgr-worker: NPROCGR-6-WORKER_WARMSTART: ProcMgr worker warm start. (PID=1439)
Feb 28 20:41:20 KS003-VM01r Launcher: 42: %LAUNCHER-6-PROCESS_START: Configuring process 'Q2485' to start in role 'AllCells'
Feb 28 20:41:20 KS003-VM01r Launcher: 43: %LAUNCHER-6-PROCESS_START: Configuring process 'StrataAgent' to start in role 'AllCells'
Feb 28 20:41:20 KS003-VM01r ProcMgr-worker: NPROCGR-7-NEW_PROCESSES: New processes configured to run under ProcMgr control:
['FanDetector', 'Ln3', 'Max6558', 'Xcvr']
Feb 28 20:41:20 KS003-VM01r ProcMgr-worker: NPROCGR-7-PROCESSES_ADOPTED: ProcMgr (PID=1439) adopted running processes: (MlagTunnel,
PID=1472) (Mirroring, PID=1504) (Lag, PID=1474) (Acl, PID=1476) (PortSec, PID=1477) (PhyEthtool, PID=1863) (Stp, PID=1480) (TopoAgent,
PID=1481) (Ira, PID=1482) (LedPolicy, PID=1483) (Arp, PID=1486) (Icmp0012, PID=1441) (Mlag, PID=1480) (Dns, PID=1408) (Ebra, PID=1507)
(Thermostat, PID=1494) (EventMon, PID=1496) (Rlb, PID=1498) (AgentMonitor, PID=1500) (Sysd, PID=1440) (FastClid, PID=1441) (Fru,
PID=1442) (Launcher, PID=1441) (Mdp, PID=1508) (TempSnooping, PID=1511) (Smbus, PID=1745) (Mio, PID=1746) (PowerSupplyDetector,
PID=1864) (Ssd, PID=1747) (PcBus, PID=1652) (Ssb28, PID=1653) (Lldp, PID=1465) (SuperServer, PID=1466) (PowerManager, PID=1467)
(StpTopology, PID=1468) (Aaa, PID=1469)
Feb 28 20:41:20 KS003-VM01r ProcMgr-worker: NPROCGR-6-PROCESS_STARTED: 'FanDetector' starting with PID=1883 (PID=1439) -- executing /
usr/bin/FanDetector
Feb 28 20:41:19 KS003-VM01r ProcMgr-worker: NPROCGR-6-PROCESS_STARTED: 'Max6558' starting with PID=1882 (PID=1439) -- executing /usr/
bin/Max6558
Feb 28 20:41:20 KS003-VM01r ProcMgr-worker: NPROCGR-6-PROCESS_STARTED: 'Xcvr' starting with PID=1884 (PID=1439) -- executing /usr/bin/
XcvrAgent
Feb 28 20:41:20 KS003-VM01r Launcher: 44: %LAUNCHER-6-PROCGR_WARMSTART: Initiating warm start of 'ProcMgr (worker)'
Feb 28 20:41:20 KS003-VM01r ProcMgr-worker: NPROCGR-7-WORKER_WARMSTART_DONE: ProcMgr worker warm start done. (PID=1439)
Feb 28 20:41:20 KS003-VM01r ProcMgr-worker: NPROCGR-6-PROCESS_STARTED: 'Ln3' starting with PID=1886 (PID=1439) -- executing /usr/bin/
ln3
    
```

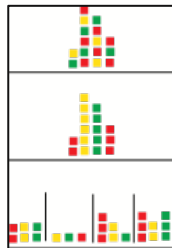
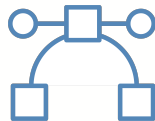
3. CLI画面たくさん同時に開き、CPU使用率などを比較・・

4. トラフィックフローの詳細を見るためにはflowコレクタが必要



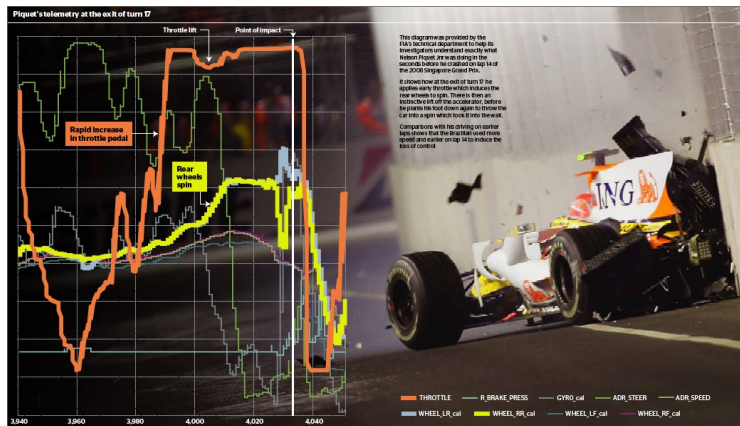
# 従来のネットワーク運用の問題点

- 精度が悪い/頻度が少ない
- 情報量が多すぎて整理ができない
- 全てのフォーマットが違う(SNMP/Syslog/xFlow/CLI)
  - 全てトランスポートが違う
  - 異なるタイムスタンプ
  - 解析ツールへの変換が難しい



# テレメトリーとは

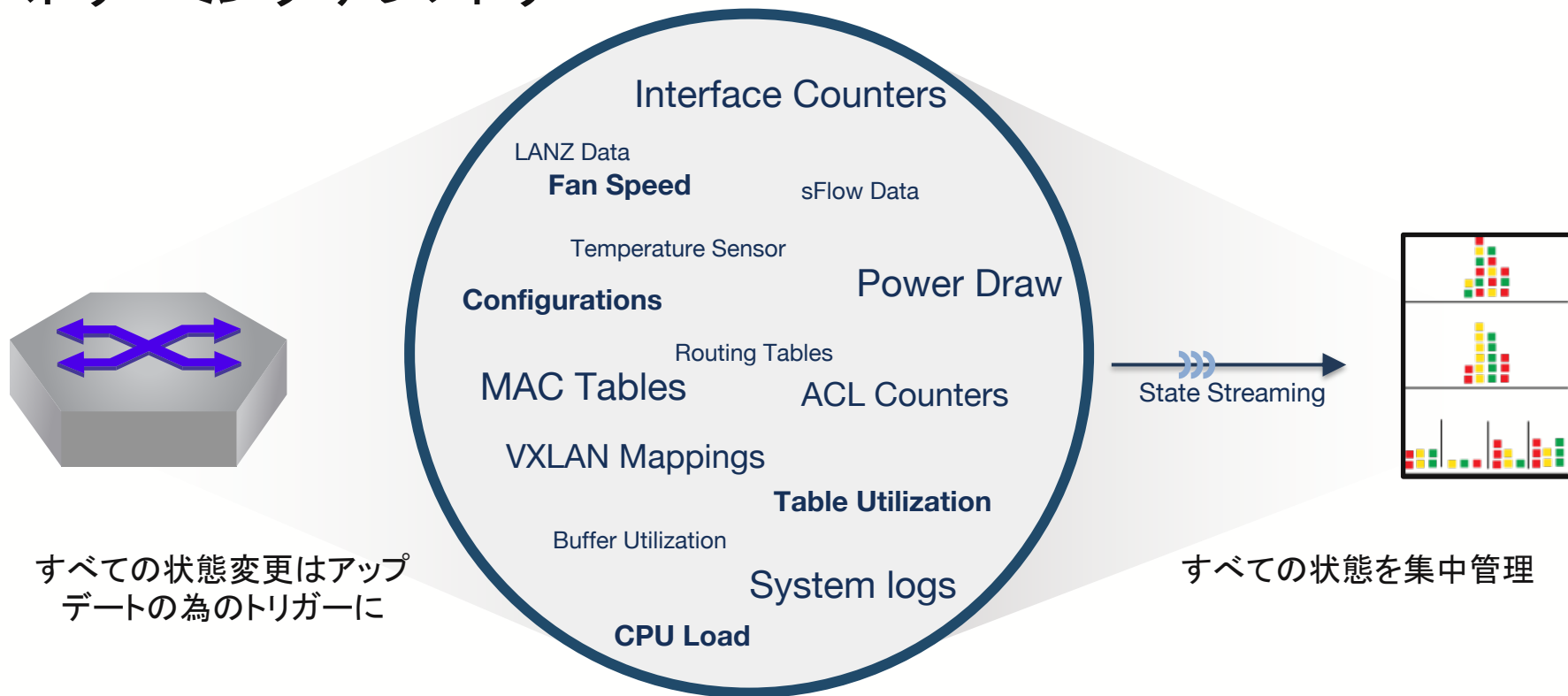
- テレメーター(遠隔計測装置)を使って、遠隔地の測定結果をコントロールセンターに送信すること。ガスや電気のメーター、自動販売機の売り上げ管理システムなどに用いられる。テレメリング。テレメータリング。遠隔測定法。\*デジタル大辞泉より



## 【例】F1でのテレメトリーデータ

ある選手が、レースでクラッシュする数秒前の行為を正確に理解するためにFIA技術部門が提供したものの、複数のデータを相関関係で分析。

# ストリーミングテレメトリー



すべての状態変化をあらゆるデバイスから即座に

# テレメリーの標準化

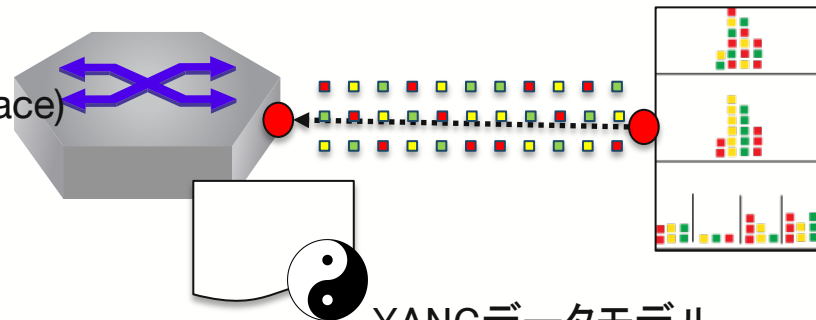
- 加工がしやすいデータモデル
  - YANGフォーマットで定義
- PublisherとSubscriberモデル
- トランスポート
  - NETCONF/RESTCONF
  - gNMI(gRPC Network Management Interface)
- 標準化
  - IETF
  - OpenConfig



I E T F®



netconf/restconf/gNMI



YANGデータモデル

- Google/AT&T/Microsoft/British Telecom/Facebook/Comcast/Level 3/Cox Communications/Yahoo!/Apple/Jive Communications/Deutsche Telekom/Bell Canada/SK Telecom/Bloomberg/Netflix/Cloudflare/Oracle/Tencent/Baidu/Alibaba/Telefonica/GoDaddy/Viasat/LinkedIn/Digital Ocean

# テレメトリーを利用したネットワークの見える化

The screenshot displays the CloudVision ARISTA interface with the following components:

- Navigation Bar:** CloudVision ARISTA, Devices, Events, Provisioning, Metrics, CloudTracer, Topology, Mock Data, cvpadmin, and a settings gear icon.
- Search Bar:** "Event, device, or interface" with filter buttons for Info, Warning, Error, and Critical, and a "Show acknowledged" toggle.
- Events (143):** A list of events with details such as "Syslog event detected: MOUNT\_PEER...", "Anomaly in DOM Temperature and Tx ...", "Routing table exceeded utilization thre...", "Insufficient Uplink Device Redundancy", "Low Disk Space Available on Device", "Process high cpu usage", and "Ethernet1 on Bldg1-FLR1-AP2".
- Summary:** "Found 143 events on 19 devices (18 devices total) Showing events from Aug 2, 2020 until now (about 1 day)". Includes buttons for "Configure Event Generation" and "Configure Notifications".
- Most Active Devices:** A table listing devices and their event counts by severity.
- Event Correlation:** A section titled "イベントのcorrelation(相関付)を実施 Informational/Warning/Errors/Criticalに分類" with a table showing event types and their counts.
- Timeline:** A horizontal timeline for "Aug 3, 2020" showing event occurrences from 12:00 to 9:00, with colored dots representing different event severities.

Device	Info	Warning	Error	Critical
DC-NY-p2r3-Edge1	0	9	4	0
HQ-Firewall1	0	4	4	4
DC-NY-p2r4-Edge2	0	3	3	3
Bldg1-FLR1-AP1	0	4	1	4

Event Type	Count	Event Type	Count
High CPU Temperature	4	Critical	10
Interface Exceeded Outbound Utilization Threshold	2	Error	36
Streaming Analytics Error	2	Warning	57
High PTP Skew	2	Info	40

# 見つけにくいエラーも早期発見

The screenshot displays the Arista CloudVision interface. The top navigation bar includes 'CloudVision ARISTA', 'Devices', 'Events', 'Provisioning', 'Metrics', 'CloudTracer', 'Topology', 'Mock Data', and 'cypadmin'. A search bar on the left contains the text 'Event, device, or interface'. Below the search bar are filters for 'Info', 'Warning', 'Error', and 'Critical', and a 'Show acknowledged' toggle.

The main content area shows an event titled 'FCS errors detected on Ethernet1 on DC-NY-p2r3-Edge1' with a red error icon. The event is active, dated Aug 3, 2020 02:59:10 JST, and occurred 6 hours ago. The event description states: 'Event on Ethernet1 on DC-NY-p2r3-Edge1: Detected FCS errors (49 errors/s) on interface'. An 'Acknowledge' button is visible in the top right of the event card.

Below the event card are four charts:

- RX Traffic Rate:** A stacked area chart showing traffic rates from 02:59:10 to 3:10. The categories are Bitrate In (5,801.3 Mbps), Unicast Packets In (66.1 kpps), Multicast Packets In (50 kpps), Broadcast In (11.8 kpps), and Utilization In (58.01%).
- RX Error Counters:** A horizontal bar chart showing error rates from 02:59:10 to 3:10. The categories and rates are: Errors In (49 errors/sec), FCS Error Rate (34 errors/sec), Alignment Error Rate (60 errors/sec), Symbol Error Rate (50 errors/sec), Giants (36 errors/sec), Runts (30 errors/sec), and Discards In (48 discards/sec).
- TX Traffic Rate:** A stacked area chart showing traffic rates from 02:59:10 to 3:10. The primary category is Bitrate Out.
- TX Error Counters:** A horizontal bar chart showing error rates from 02:59:10 to 3:10. The primary category is Errors Out.

At the bottom, a timeline view shows event activity from 12:00 to 6:00 on Aug 3, 2020. A vertical line marks the event time at 02:59:10, with a 'Live' indicator and a 'Show Last: 1h 30m 5m 30s' dropdown menu.

# 各デバイスの詳細情報を表示

CloudVision ARISTA

Devices Events Provisioning Metrics CloudTr

Devices > DC-NY-p2r3-Edge1 > Device Overview

Device Overview

System Details

System

Processes

Storage

Log Messages

Hardware Capacity

Running Config

Snapshots

View in Topology

Compliance

Environment

Tags

Switching

ARP Table

NDP Table

Bridging Capability

MAC Address Table

MLAG

VXLAN

Routing

Provision Device

Memory Overview

CPU Overview

CPU Core Utilization

PID	Name	Started	Threads	Memory	CPU ↓
59	XpFlow	Nov 20, 2019 19:41:23	1	200.5 MB	19.0%
59	XpFlow	Nov 20, 2019 19:41:23	1	200.5 MB	19.0%

Temperature and Cooling

Temperature

Fan Speeds

Power Supply Output

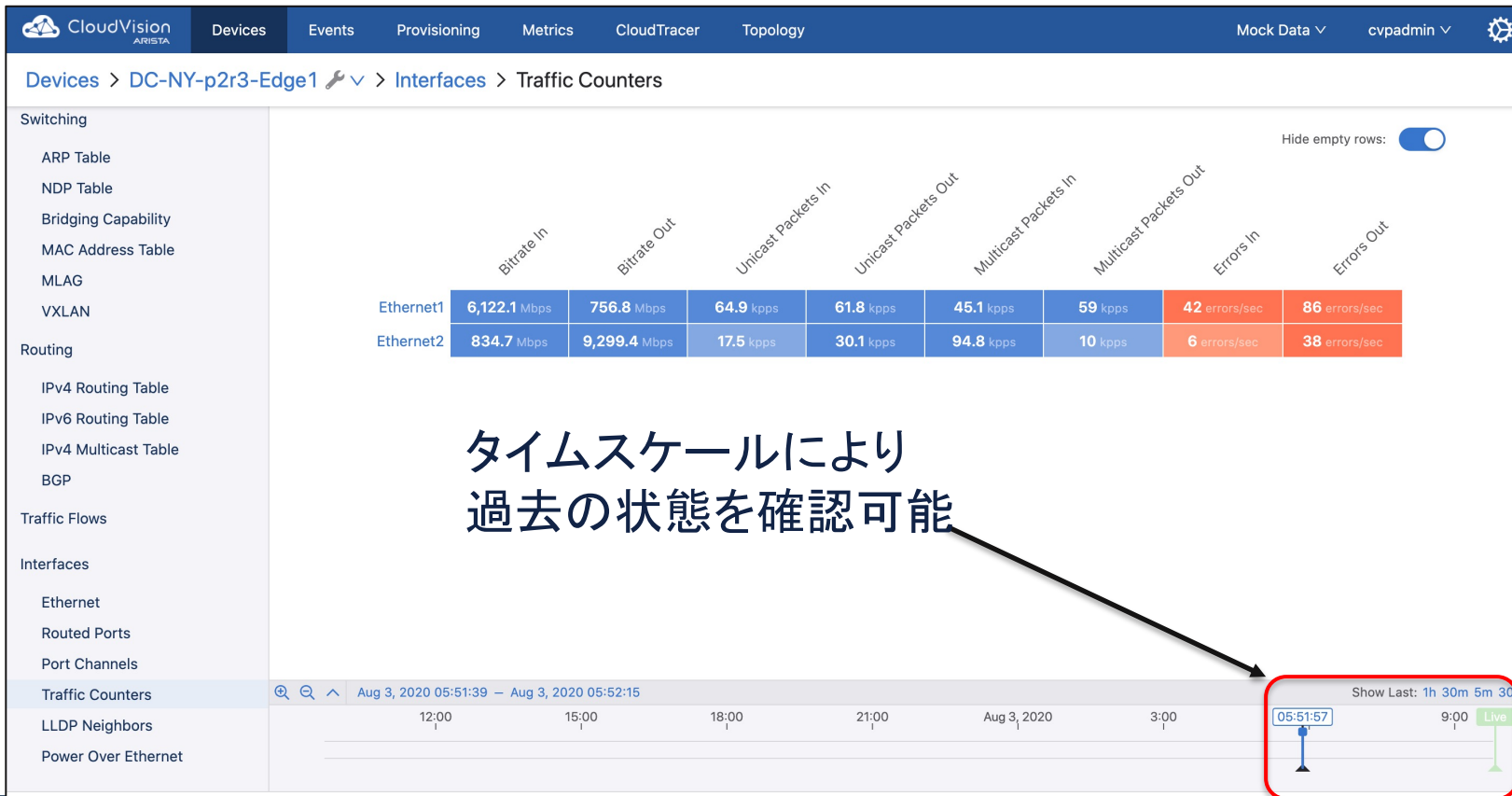
## デバイスの重要情報を可視化



# デバイスのハードウェアエントリーも確認可能

The screenshot displays the Arista CloudVision interface. The top navigation bar includes 'CloudVision ARISTA', 'Devices', 'Events', 'Provisioning', 'Metrics', 'CloudTracer', and 'Topology'. The main breadcrumb path is 'Devices > DC-NY-p2r3-Edge1 > Device Overview'. On the left sidebar, 'Hardware Capacity' is highlighted with a red box. An arrow points from this box to a detailed view of 'Hardware Table Capacity' for device 'DC-NY-p1r12-Core1'. This detailed view shows three donut charts: 'Egress ACL Capacity' with 0 entries, 'Longest Prefix Match' with 60 entries (54 V6Routes and 6 V4Routes), and 'MAC L2' with 140.1k entries (140.1k Linecard0/0). A 'System Status' section at the bottom indicates the device is streaming and provides a 'Provision Device' button. A timeline at the bottom shows the data was captured on Aug 3, 2020, at 12:44:47.

# タイムスケールを用いた履歴管理



# 詳細履歴情報およびデータExport

CloudVision ARISTA | Devices | Events | Provisioning | Metrics | CloudTracer | Topology | Mock Data | cvpadmin

## Errors In for Ethernet1 on DC-NY-p2r3-Edge1

Aug 3, 2020

50 errors/sec (since Aug 3, 2020 08:44:08)

Metric	Value
Unicast Packets Out	0.0 kpps
Multicast Packets In	40.1 kpps
Multicast Packets Out	0.0 kpps
Errors In	42 errors/sec
Errors Out	86 errors/sec
Unicast Packets In	30.1 kpps
Multicast Packets Out	10 kpps
Errors In	6 errors/sec
Errors Out	38 errors/sec

Metric History | **Data Table** | Data Paths | Statistics | Related Metrics

### RX Error Counters

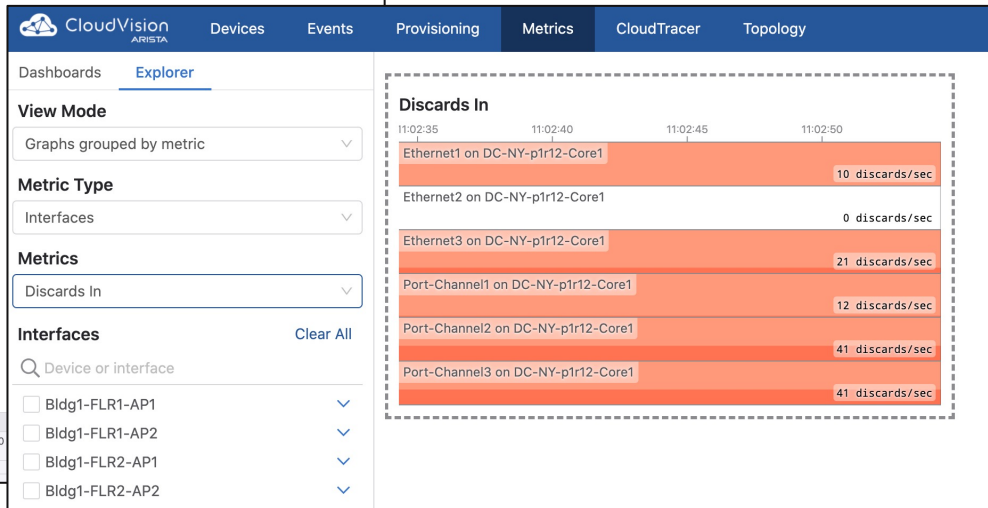
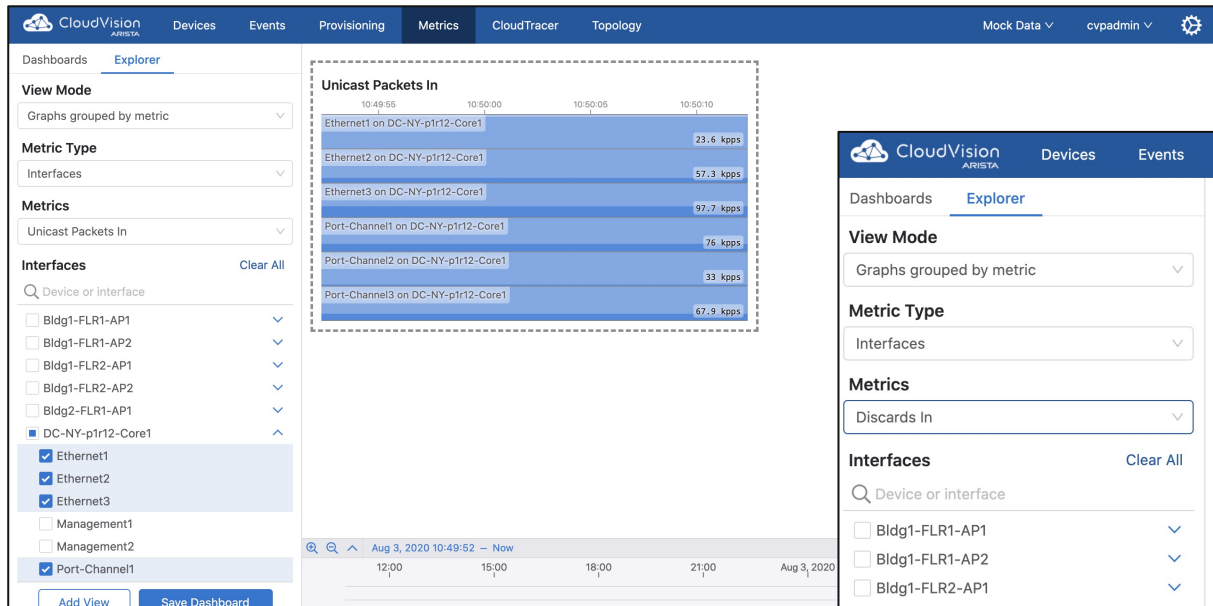
FCS Error Rate	43 errors/sec
Alignment Error Rate	64 errors/sec
Symbol Error Rate	46 errors/sec
Giants	28 errors/sec
Runts	43 errors/sec
Discards In	49 discards/sec

Index ↓	Timestamp	Value
Filter	Filter	Filter
	Aug 3, 2020 08:44:08.384 JST	50

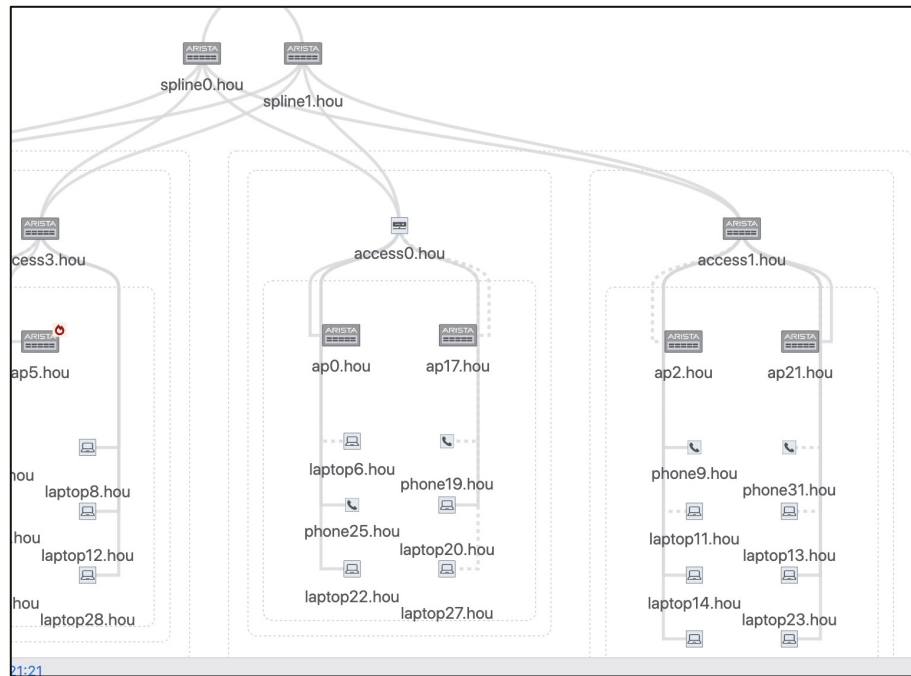
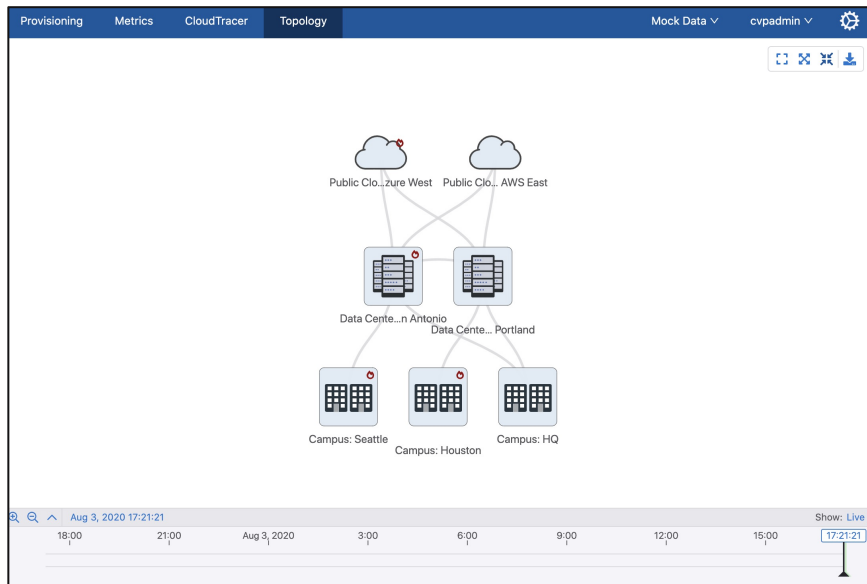
Export to CSV | Showing 1 of 1 row

LLDP Neighbors | Power Over Ethernet

# 各メトリックに合わせたデバイス内/デバイス間の比較が可能



# ネットワークトポロジー



- LLDPの隣接情報を元にトポロジーを自動作成
- 標準プロトコルLLDPをサポートしていればタイプなどで判断し、トポロジーマップに登録される
- タイムスケールを持ち、時系列変更にも対応可能

# Topology Overview with Active Events

CloudVision ARISTA

Devices Events Provisioning Metrics CloudTracer Topology

### Topology Overview

Displaying 7 managed and 304 other devices

Flows | Layout | Settings

### Network Filters

VLAN membership ID or range (e.g. 1, 4-5)

VXLAN membership VNI or range (e.g. 1, 4-5)

### Link Overlay (i)

Active Events

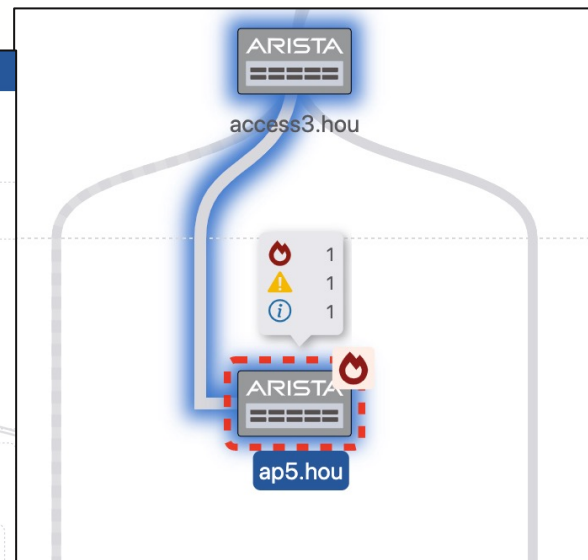
### Devices

Q Name, MAC address, or model

- access0.hou**  
acc0.access.b1f11.hou · T-1001
- access0.sea**  
acc0.access.b1f12.sea · DCS-720XP-4...
- access1.hou**  
acc1.access.b1f12.hou · DCS-720XP-4...
- access1.sea**  
acc1.access.b1f12.sea · DCS-720XP-4...
- access2.hou**  
acc2.access.b2f11.hou · T-1001
- access2.sea**  
acc2.access.b2f12.sea · T-1001

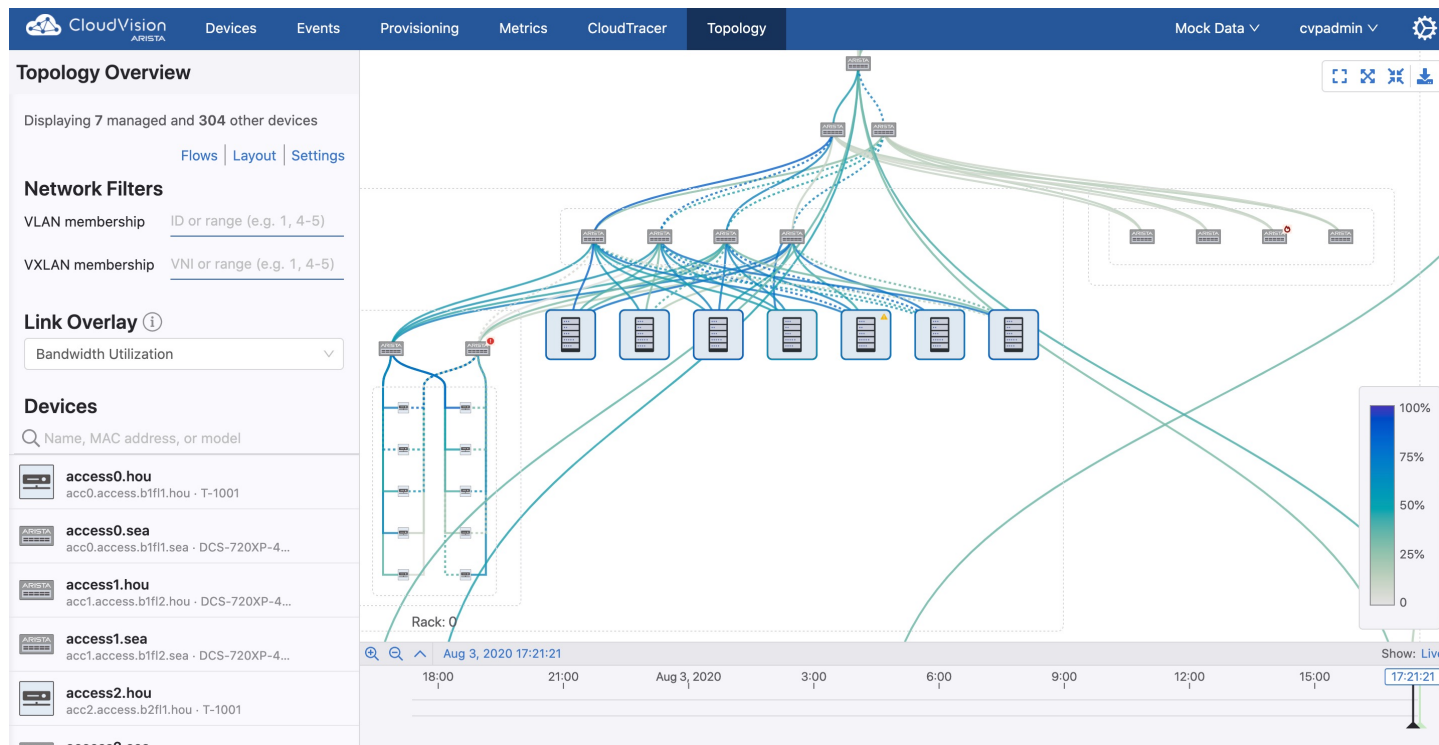
Aug 3, 2020 17:21:21

18:00 21:00 3:00 6



- イベントがどのノードで起きているか早期発見可能

# Topology Overview with Bandwidth Utilization



- トポロジーと使用帯域のマッピングが可能
- ネットワークウェザーマップ同様な使い方が可能

# Topology Overview with Error Rate/Discard Rate

CloudVision ARISTA

Devices Events Provisioning Metrics CloudTracer Topology Mock Data cvpadmin

### Topology Overview

Displaying 7 managed and 304 other devices

Flows | Layout | Settings

#### Network Filters

VLAN membership ID or range (e.g. 1, 4-5)

VXLAN membership VNI or range (e.g. 1, 4-5)

#### Link Overlay

Error Rate

#### Devices

##### Link Overlay

Discard Rate

Error Rate  
Color by errors per second

Traffic Throughput  
Color by bits transmitted per second

VLANs  
Color by VLANs configured on link interfaces

VXLANs  
Color VXLAN tunnel interface (VTI) links by virtual network identifier (VNI)

Aug 3, 2020 17:21:00

- Error Rate/Discard Rateのマッピングが可能
- Traffic Throughput/VLANs/VXLANsも可能

100+  
> 0



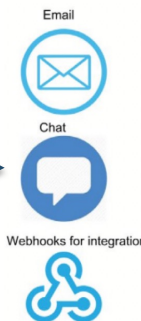
# Event Alert

The screenshot shows the CloudVision Events page. The top navigation bar includes 'CloudVision ARISTA', 'Devices', 'Events', 'Provisioning', 'Metrics', 'CloudTracer', and 'Topology'. The user is logged in as 'cvpuser'. The main content area displays 'Found 67 events on all 7 devices' and 'Showing events from Aug 2, 2020 until now (about 1 day)'. There are buttons for 'Configure Event Generation' and 'Configure Notifications'. The left sidebar shows a search bar and filters for 'Info', 'Warning', 'Error', and 'Critical'. The main area is divided into three sections: 'Most Active Devices', 'Most Common Events', and 'Event Severities'. A timeline at the bottom shows event occurrences from 21:00 on Aug 3, 2020, to 18:00.

Device	Info	Warning	Error	Critical
cvp-lf-20	0	0	4	13
cvp-lf-23	0	0	4	7
cvp-lf-21	0	0	3	6
cvp-lf-22	0	0	1	7
cvp-sp-16	0	0	1	6

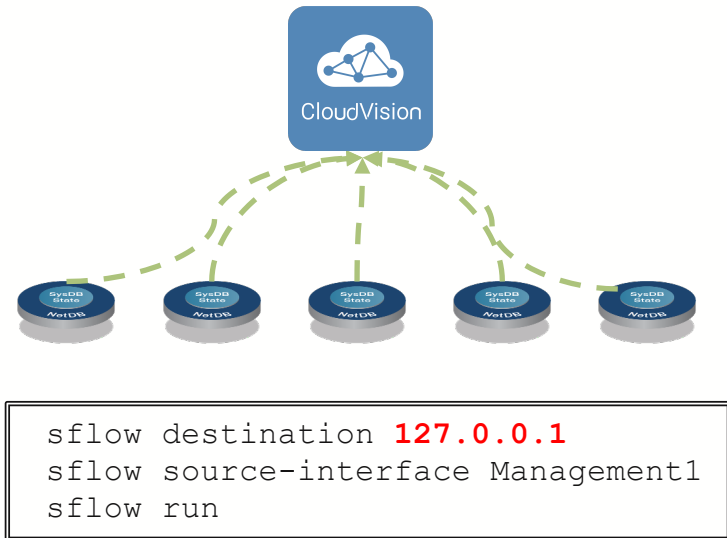
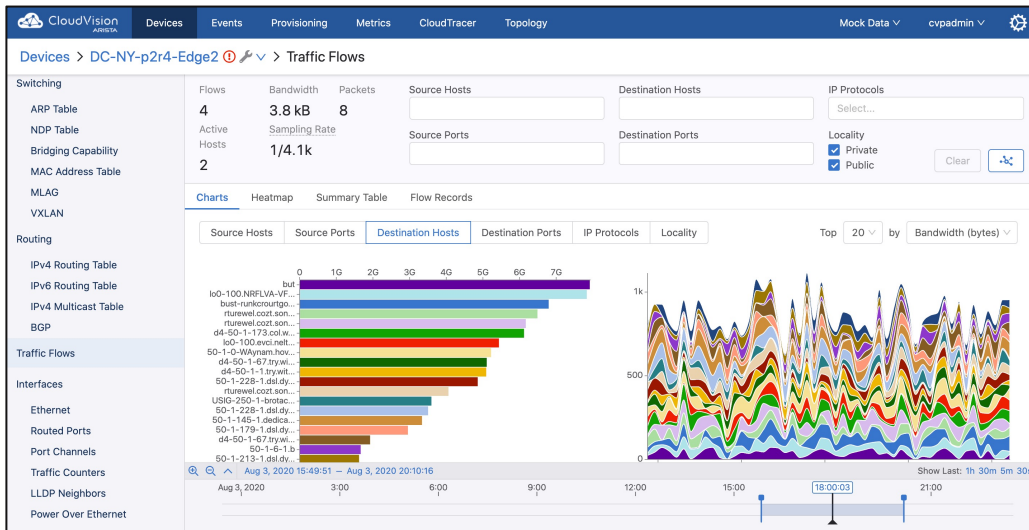
Event	Count
Packet Loss Detected For CloudTracer Host	38
Unexpected Interface Change	8
CVE Bug Exposed	6
High CPU Utilization	5

Severity	Count
Critical	0
Error	2
Warning	17
Info	48



- Event情報をメール/Slackなどのツールに通知可能

# Traffic Flow on テレメトリー



- sFlow/IPFIXでExportされたトラフィックフローデータをローカルに転送
- フロー情報をテレメトリーデータとして変換し、コレクタに送る

# テレメトリーまとめ

- 大容量トラフィック、高速インターフェース、多くのユーザを抱えるデータセンターでは従来のSNMP/Syslogなどの運用方法ではきめ細やかな情報収集や他システムとの連動が難しいケースがある
- テレメトリーは機器側より加工しやすいデータ・フォーマットでコレクタに送る仕組み
- 可視化をする事でトラブルの早期発見が可能に
- そのデータを用いてのトポロジーマップ作成/フロー情報表示などにも使える
- 機械学習(ML)などによる相関付けやアノマリー検出なども可能に



# まとめ

- 本セッションでは下記の概要を説明
  - IP ClosデザインのメリットおよびVXLANを使ったオーバーレイネットワーク
  - ZTPの技術実現方法および実際に使われている手法
  - Ansibleによる作業や設定の自動化および雛形化の重要性
  - テレメトリーを使ったネットワーク監視の最新動向
- これらを踏まえて実際のデータセンターネットワーク運用者がどの様な対応をしているかは下記セッションを受講
  - C6 どう使う？データセンターネットワーク最前線 Yahoo! JAPAN実用例
    - >> <https://www.nic.ad.jp/iw2021/program/detail/#c6>
  - C7 どう使う？データセンターネットワーク最前線 LINE実用例
    - >> <https://www.nic.ad.jp/iw2021/program/detail/#c6>



# Thank You

[www.arista.com](http://www.arista.com)