どう使う？データセンターネットワーキング最前線 概要編

Shishio Tsuchiya
shtsuchi@arista.com

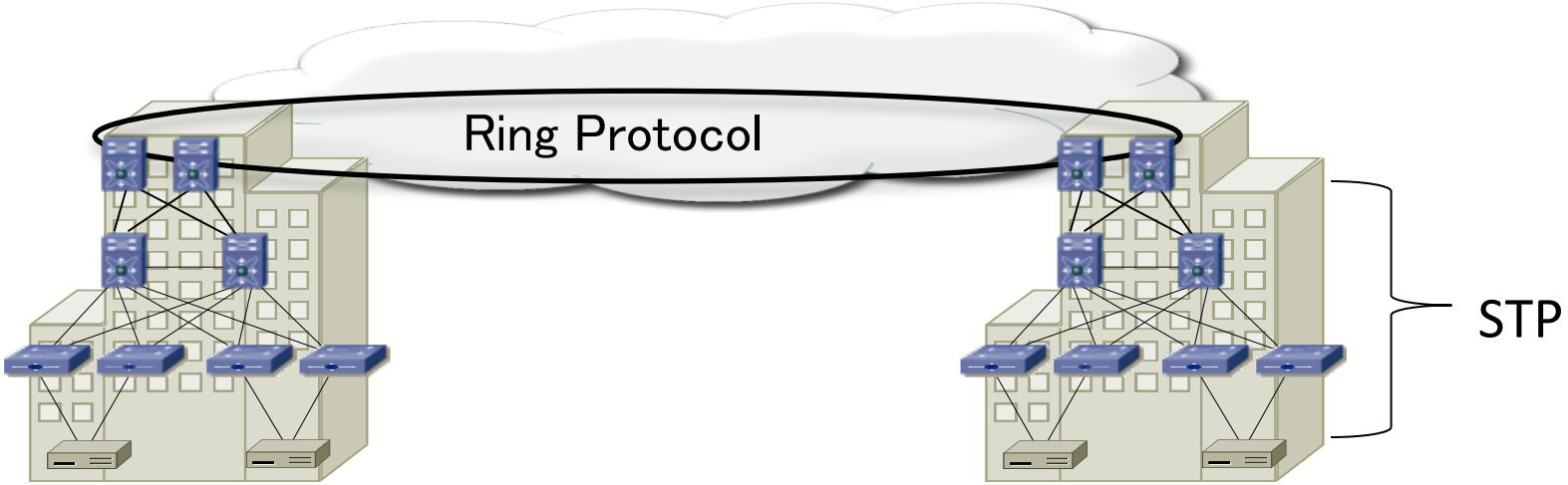
ARISTA

Agenda

- **データセンターネットワーキングデザイン**
- 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キングデザイン

2011年ごろのデータセンターネットワーク

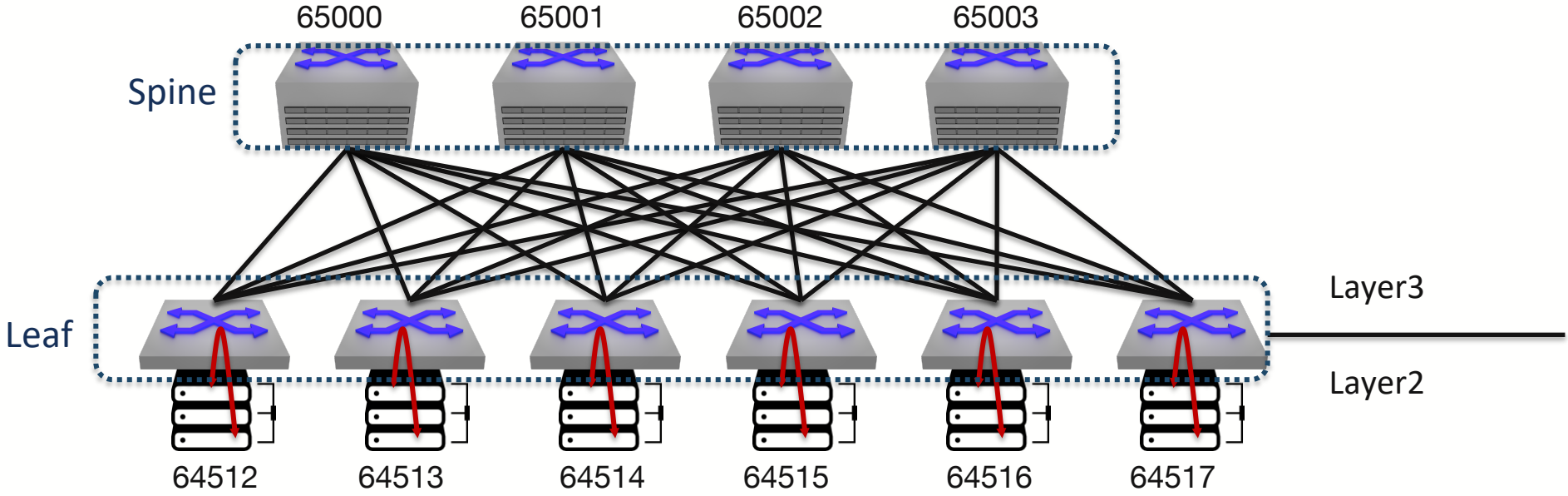


- フラットなレイヤー2のネットワークデザイン
- データセンター内ではスパニングツリーが使用
- ベンダー独自のリングプロトコル（またはG.8032)がデータセンター間で使用される
- VMライブマイグレーションは必要（GARPによって移動を通知）
- VLANがユーザ毎にアサインされる

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

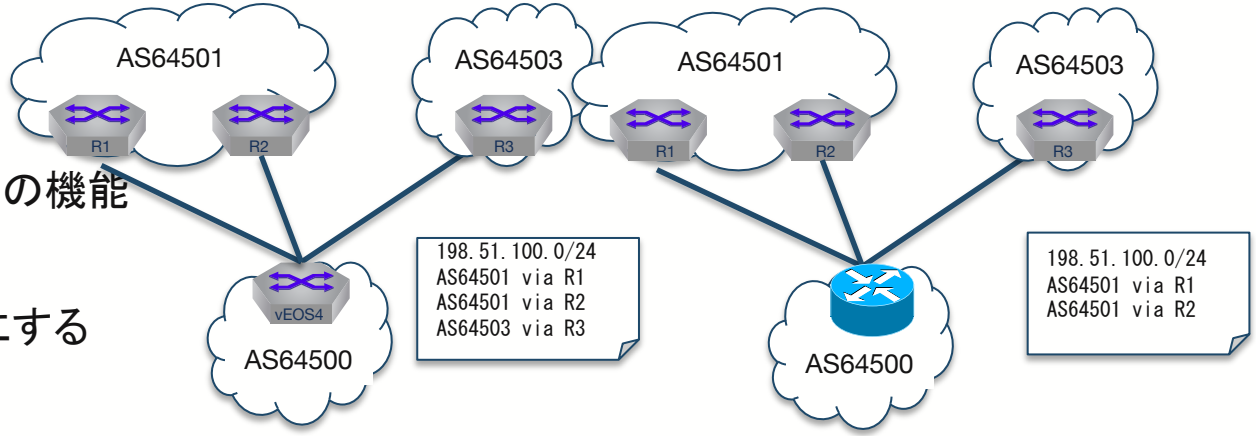


- スケールアウトするClosデザイン
- 安定した標準的なBGPプロトコルをToR/Leafスイッチに使用
- 安定性にフォーカスし、VMモビリティはラック内に留める

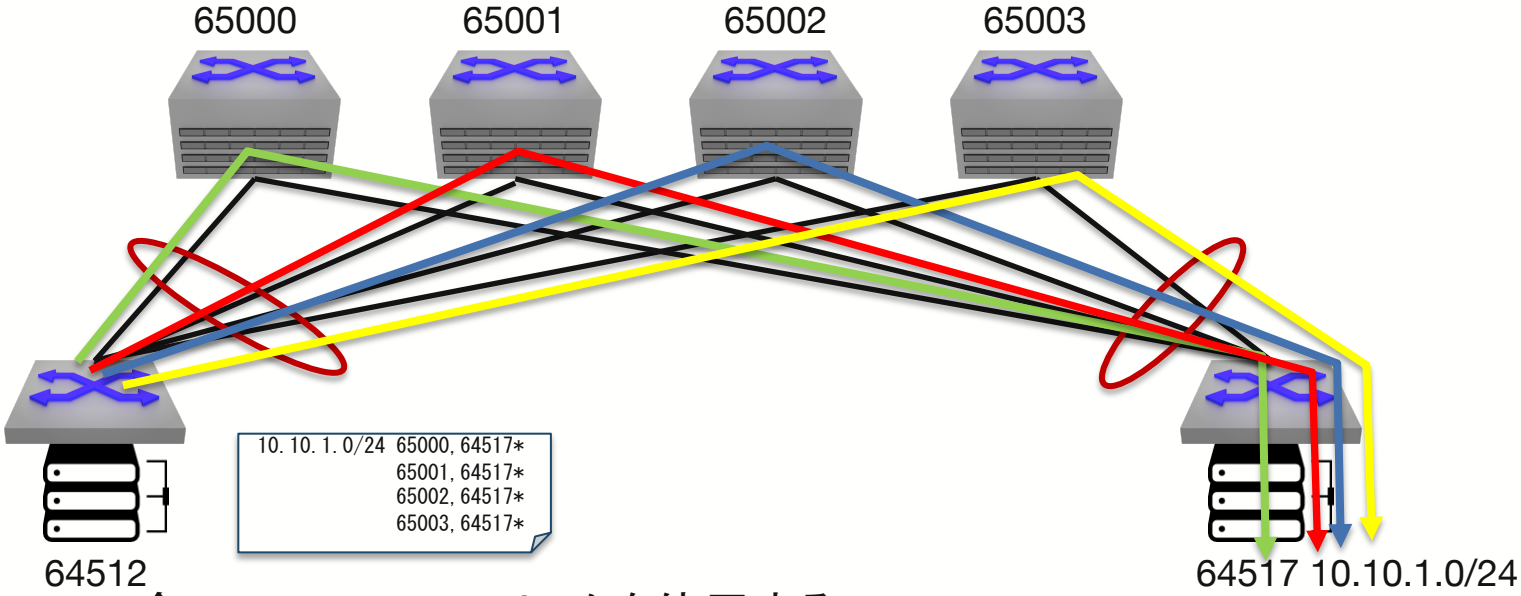
ARISTA

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**

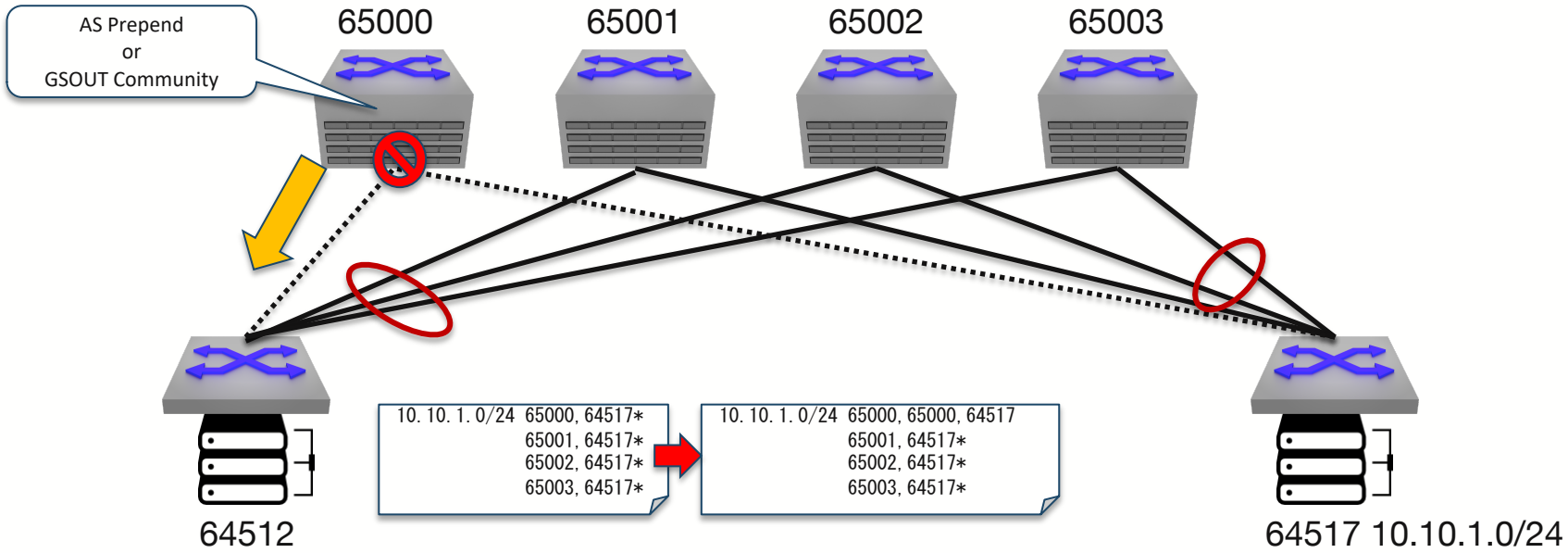


IP ECMPの利点



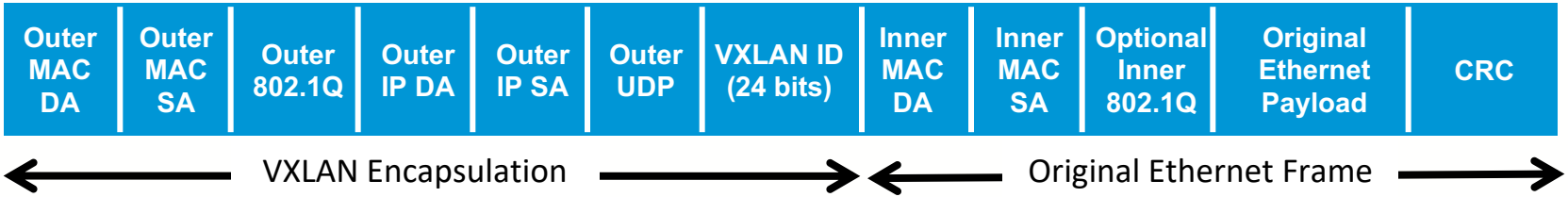
- ECMPで全てのLeaf-Spineリンクを使用する
- それぞれのフローはECMPハッシュにてバランスされる（既に実装されている）
- ルーティングパスはBGPパス属性により可視化される

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

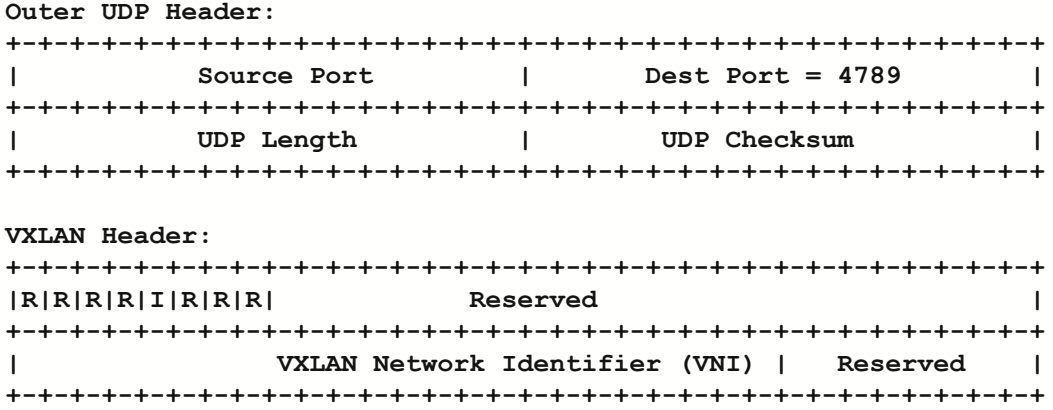
Virtual eXtensible LAN (VXLAN)



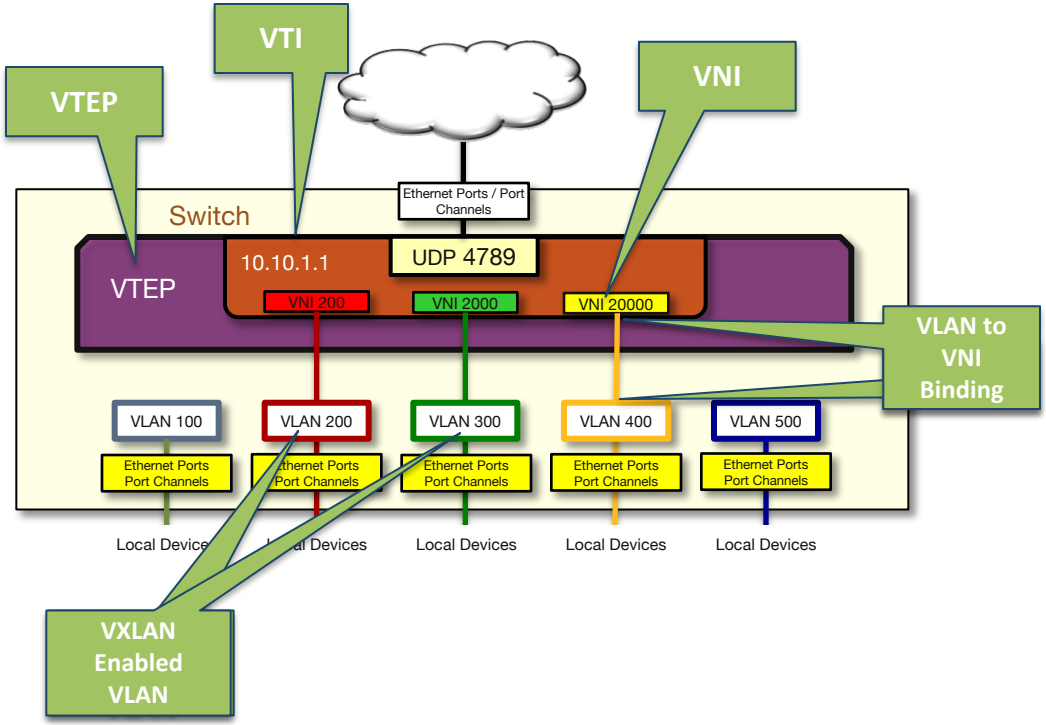
- レイヤー2フレームをIPでカプセルプロトコルを定義
- レイヤー3インフラストラクチャー上でオーバーレイのネットワークを作成する為に使用
- レイヤー2の接続性をユーザに提供

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ず、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



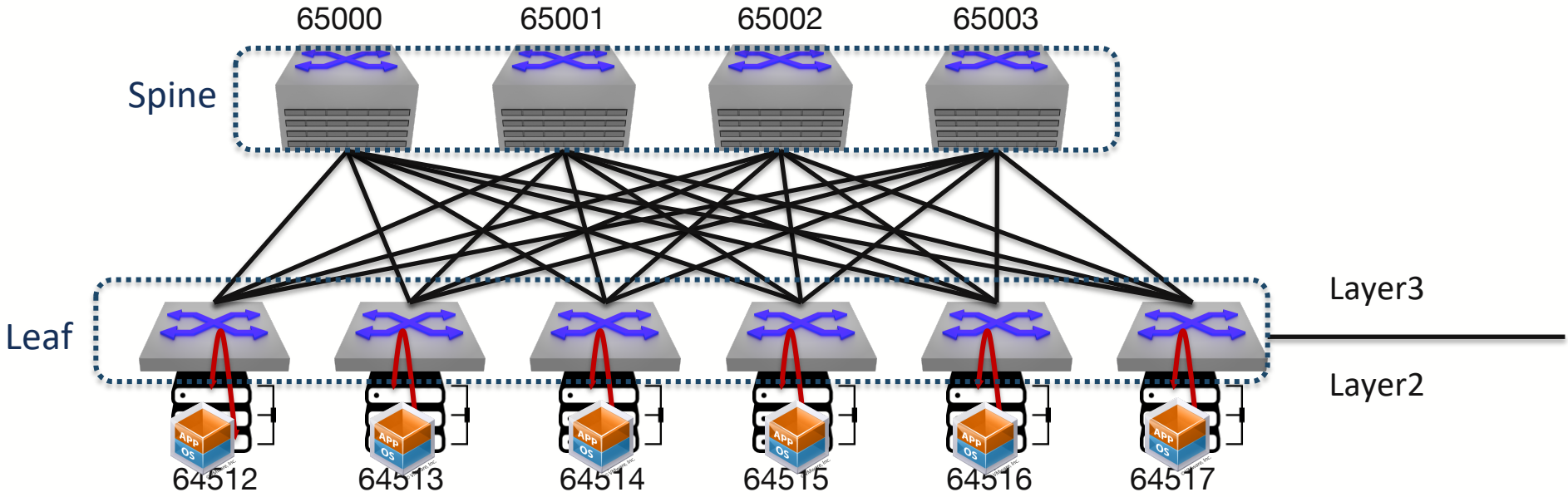
VXLAN用語集



- 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

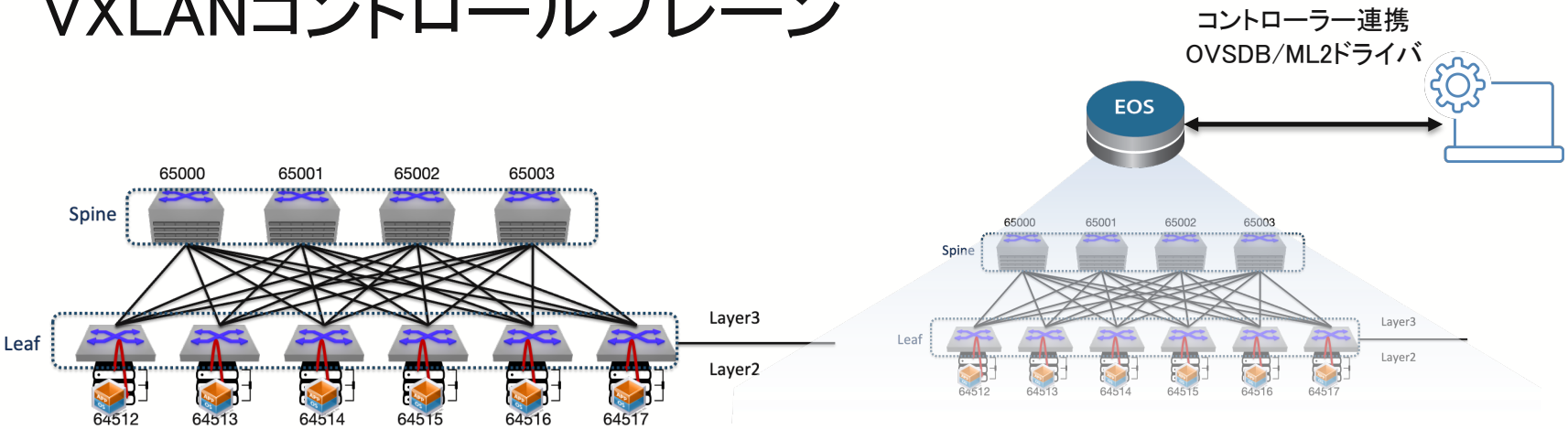
ARISTA

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 & Lean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

ARISTA

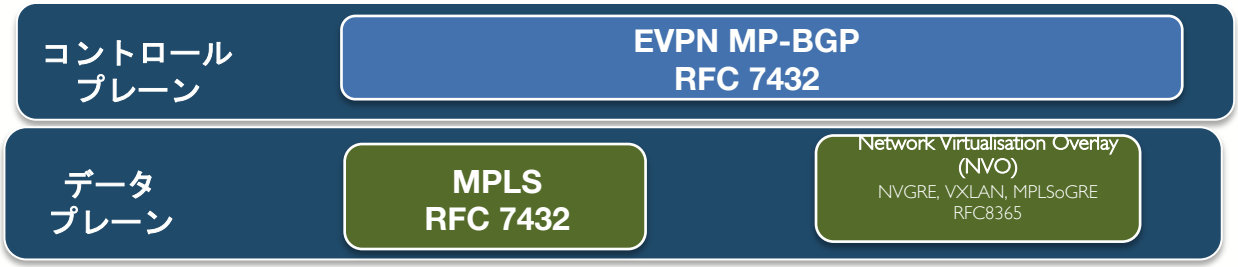
Requirements for Ethernet VPN (EVPN)

- 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

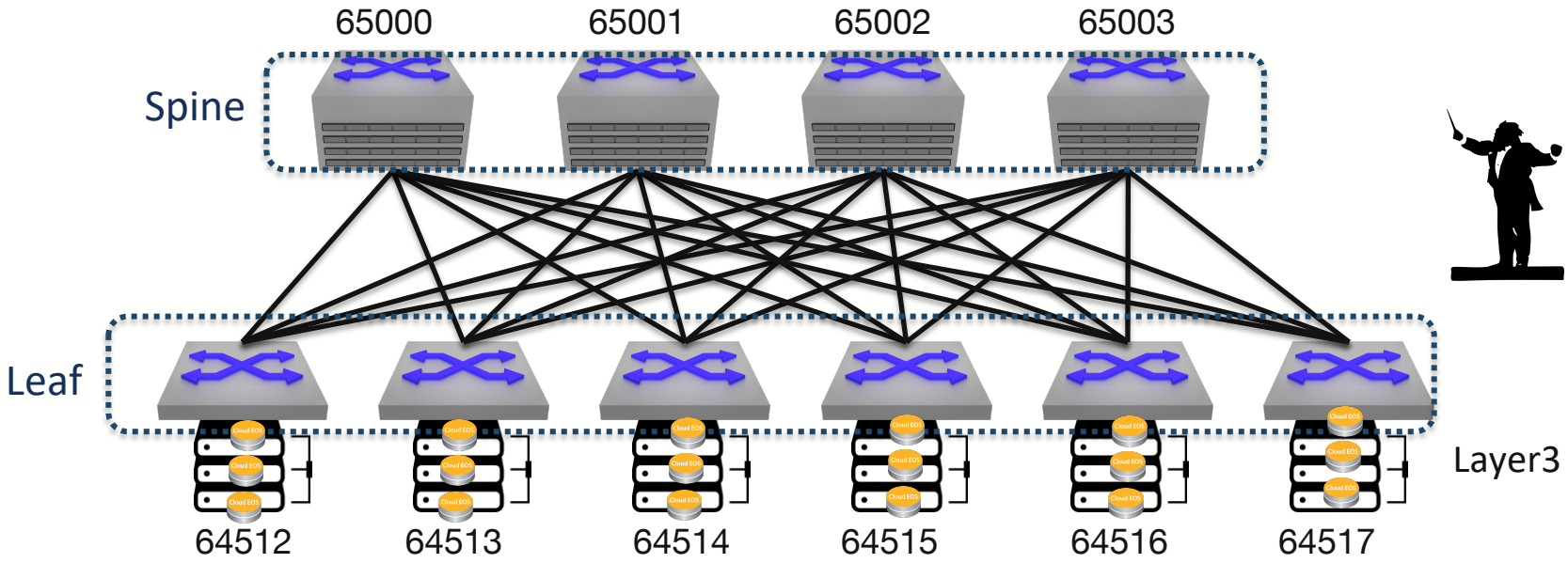
1. 冗長性の要求
   1. フローベースのロードバランシング
   2. フローベースのマルチパス
   3. PEノードの地理的冗長性
   4. 最適なトラフィックフォワーディング
   5. 最適なリダンダンシーグループ
   6. マルチホームネットワーク
2. マルチキャスト最適化
3. プロビジョニングの簡素化
4. 新しいサービスインターフェース
5. 高速コンバージェンス
6. フラッディング抑制
7. 柔軟なVPNトポロジーとポリシー
   - L2VPN/L3VPN/VPWS/E-TREE

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

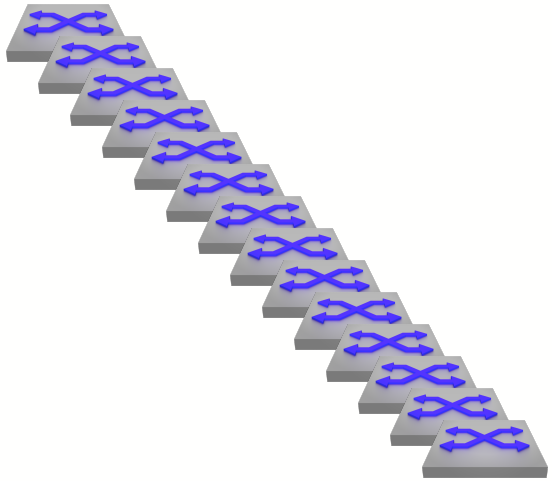
まとめ

- 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)

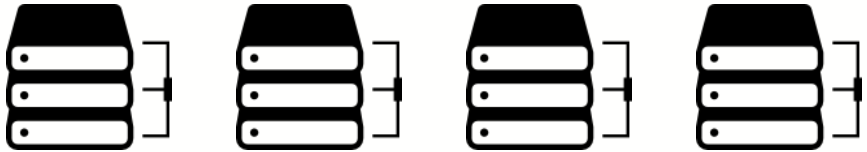
ZTPとは

- 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



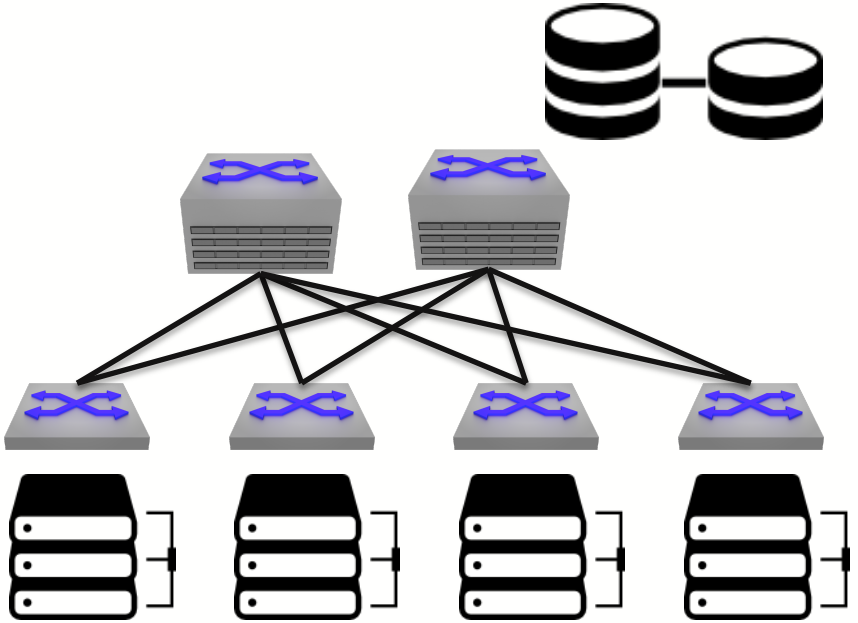
ネットワークブートの必要性

- 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



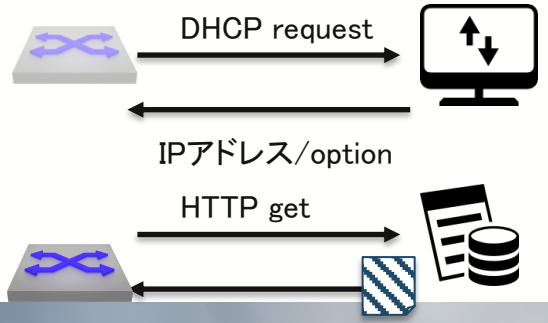
ネットワーク機器でのネットワークブートの必要性

- 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



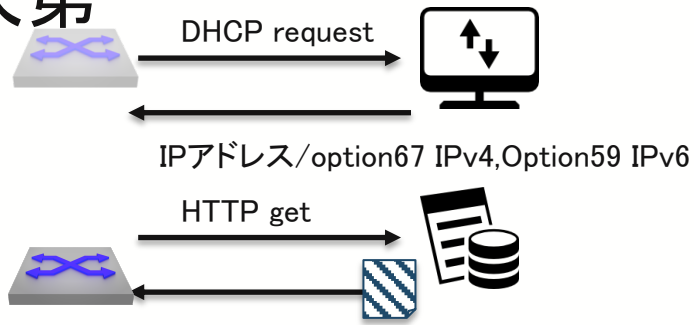
必要なもの

- ネットワークブートが可能なブートローダー
  - 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



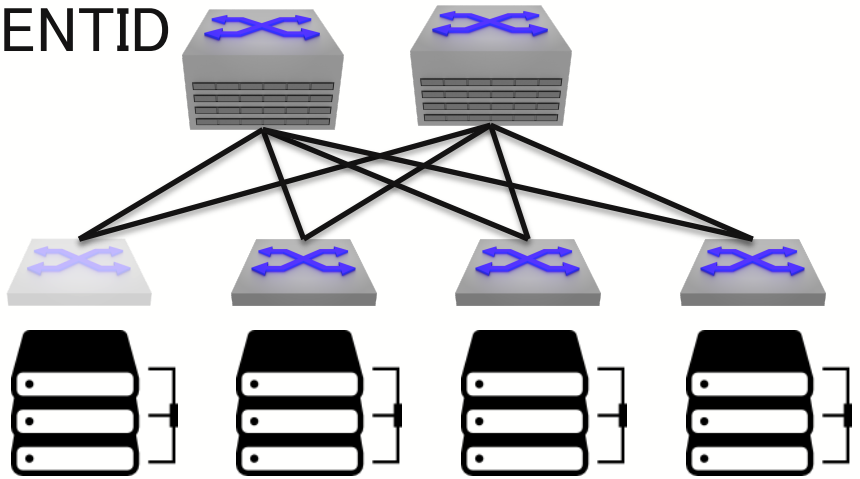
ZTPに使えるDHCPオプション

- 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

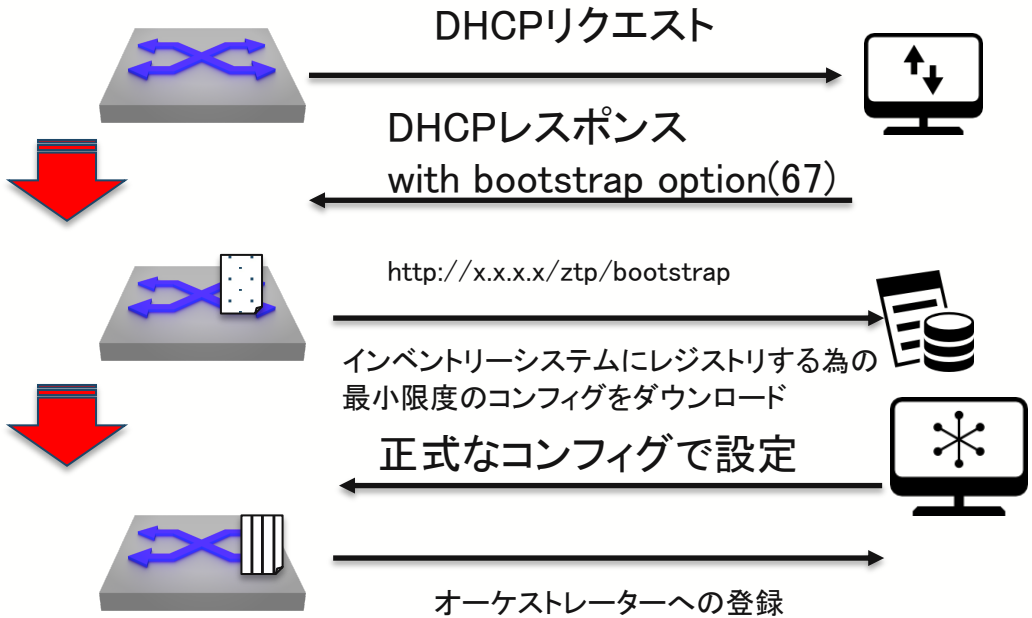


どの情報で機器固有情報を判断するか

- 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

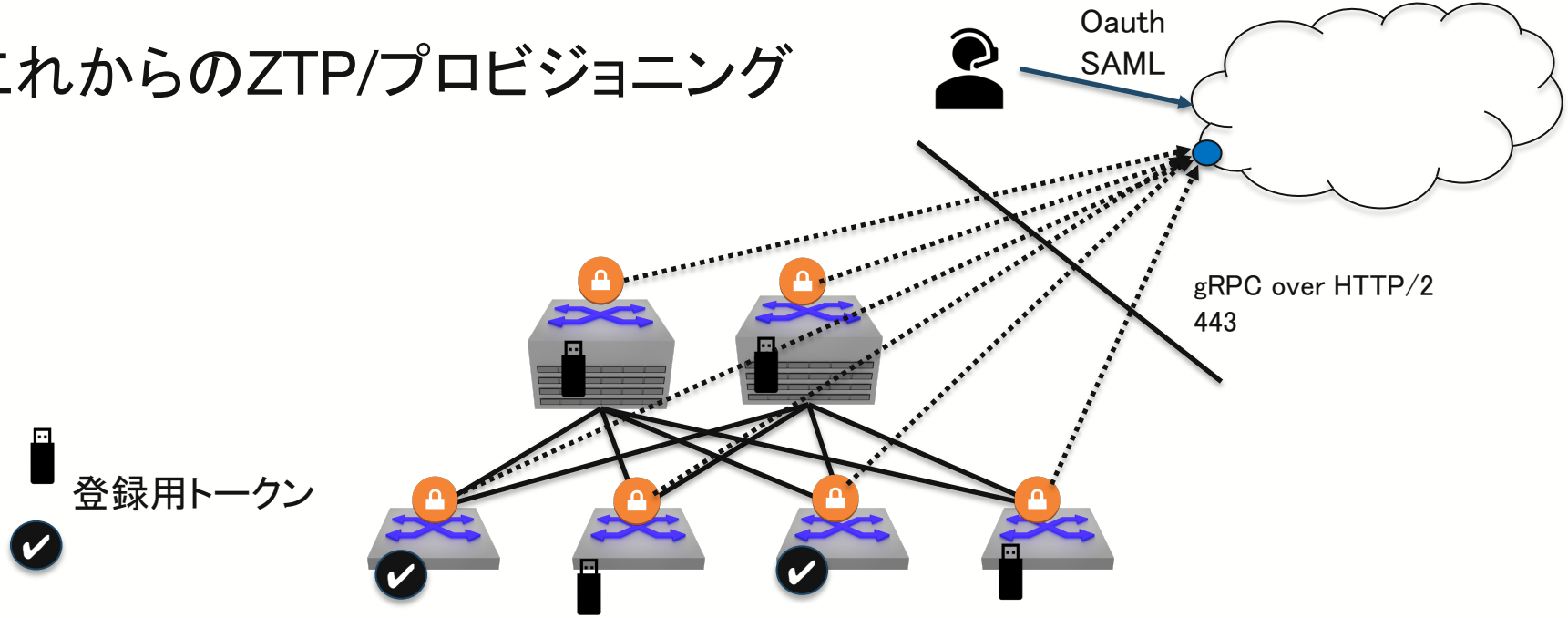


実際にクラウドタイタンで使われるZTP



- 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レジストしにいく

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

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

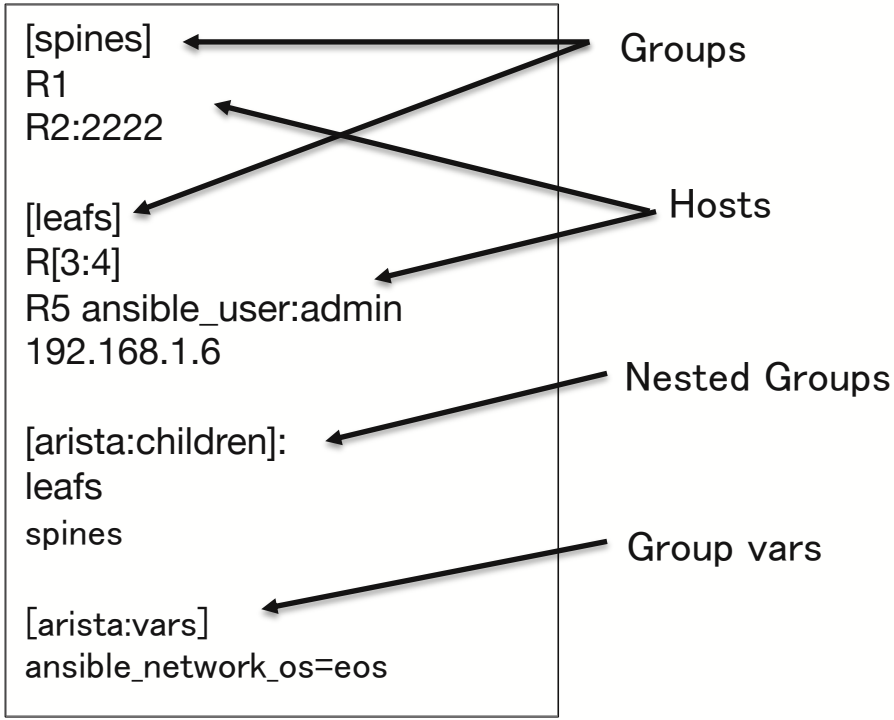
```
---

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
```

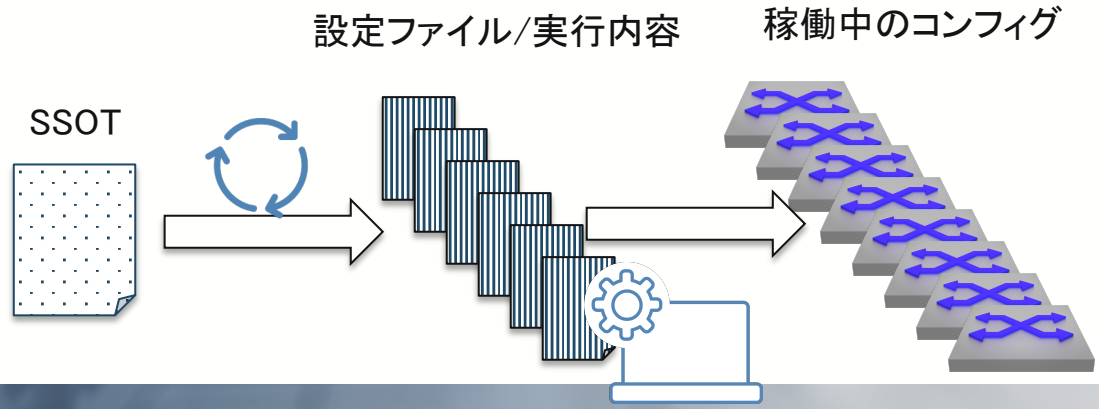
Inventory(インベントリ)

- 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

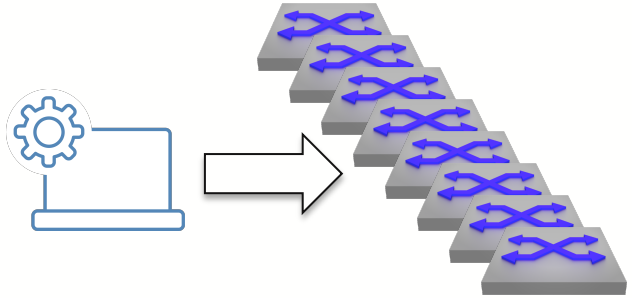


信頼できる唯一の情報源（SSOT：Single Source of Truth）

- 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

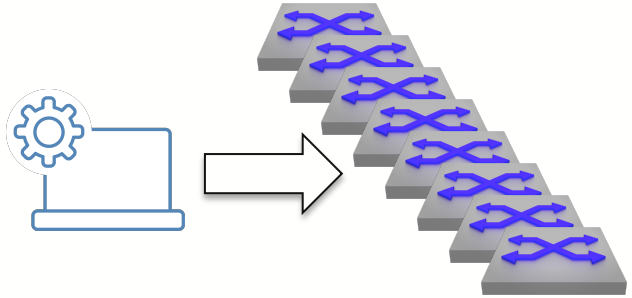


```
arista@ip-10-33-6-141:~/ATD$ ansible all -i hosts -m raw -a "show version" -u arista -k
SSH password:
192.168.0.14 | SUCCESS | rc=0 >>
```

- 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]



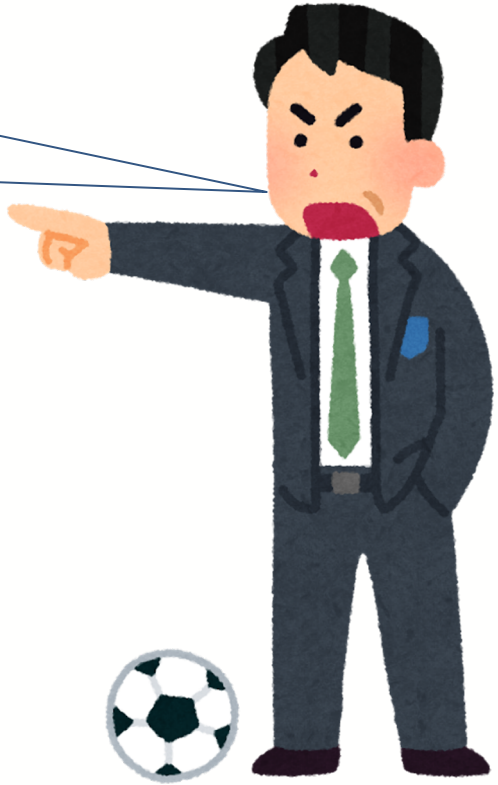
```
arista@ip-10-33-6-141:~/ATD$ ansible all -i hosts -m raw -a "show version" -u arista -k
SSH password:
192.168.0.14 | CHANGED | rc=0 >>
 vEOS
Hardware version:
Serial number:       leaf1
System MAC address:  04a8.0159.c467

Software image version: 4.23.0.1F
Architecture:           i686
Internal build version: 4.23.0.1F-13860745.42301F
Internal build ID:      6a1d05a3-2754-4ecf-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

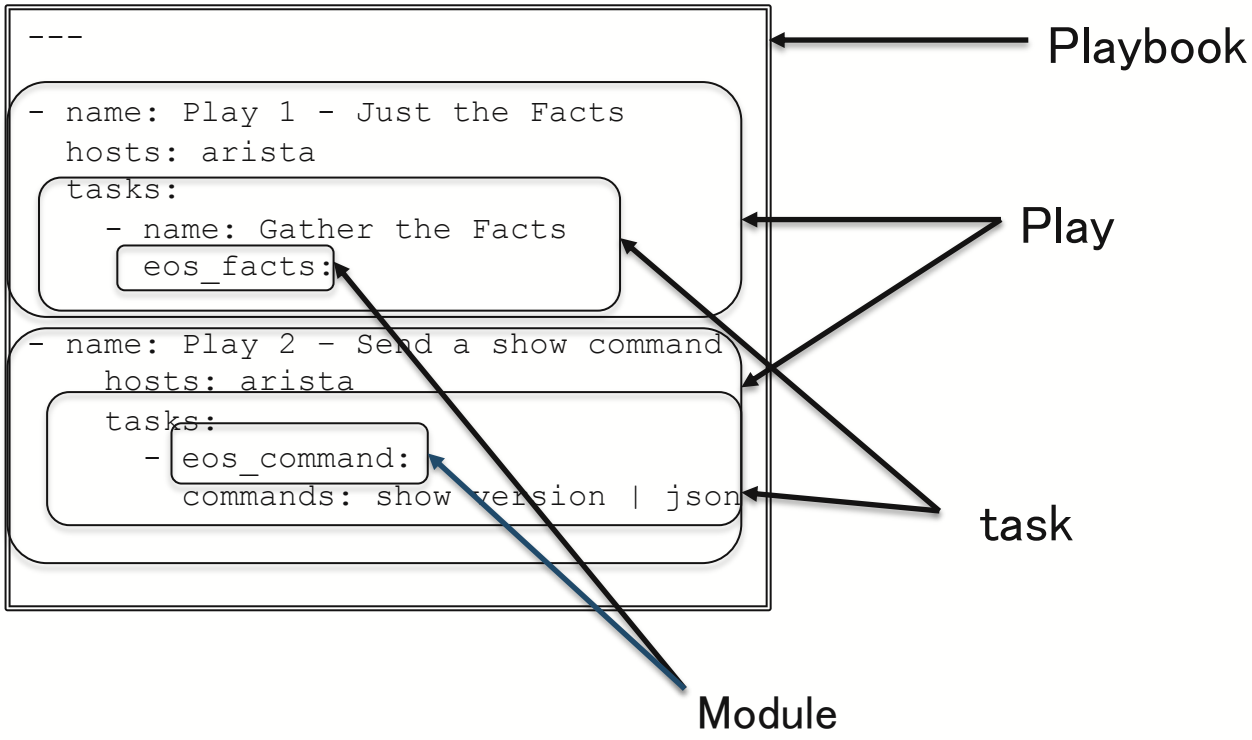
プレイブック

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
- 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に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

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Jinjya2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

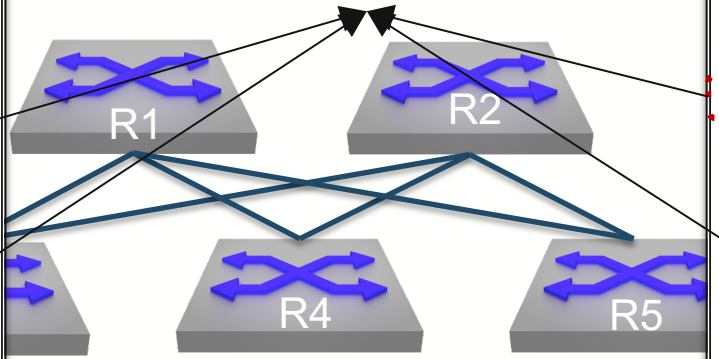
```
<title>{% block title %}{% endblock %}</title>
<ul>
{% for user in users %}
  <li><a href="{{ user.url }}">{{ user.username }}</a></li>
{% endfor %}
</ul>
```

http://docs.ansible.com/ansible/latest/playbooks_templating.html
http://jinja.pocoo.org/docs/2.10/

テンプレートの作り方

```
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
!
```

グローバルテンプレート

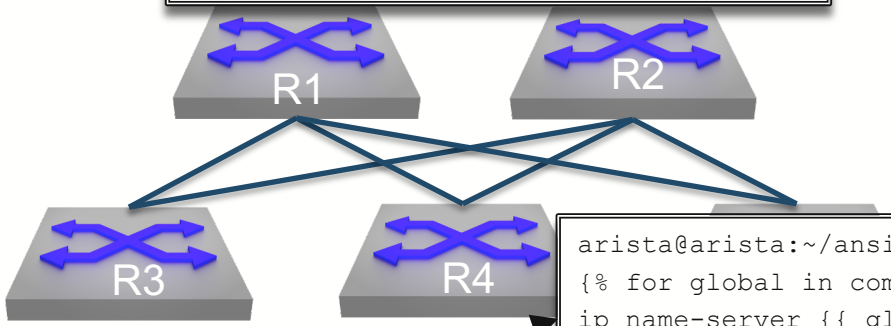


インターフェース
テンプレート

```
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$ cattemplates/global.j2
{% for global in common %}
ip name-server {{ global.dnsserver }}
switchport default mode {{ global.switchport }}
{% endfor %}
```

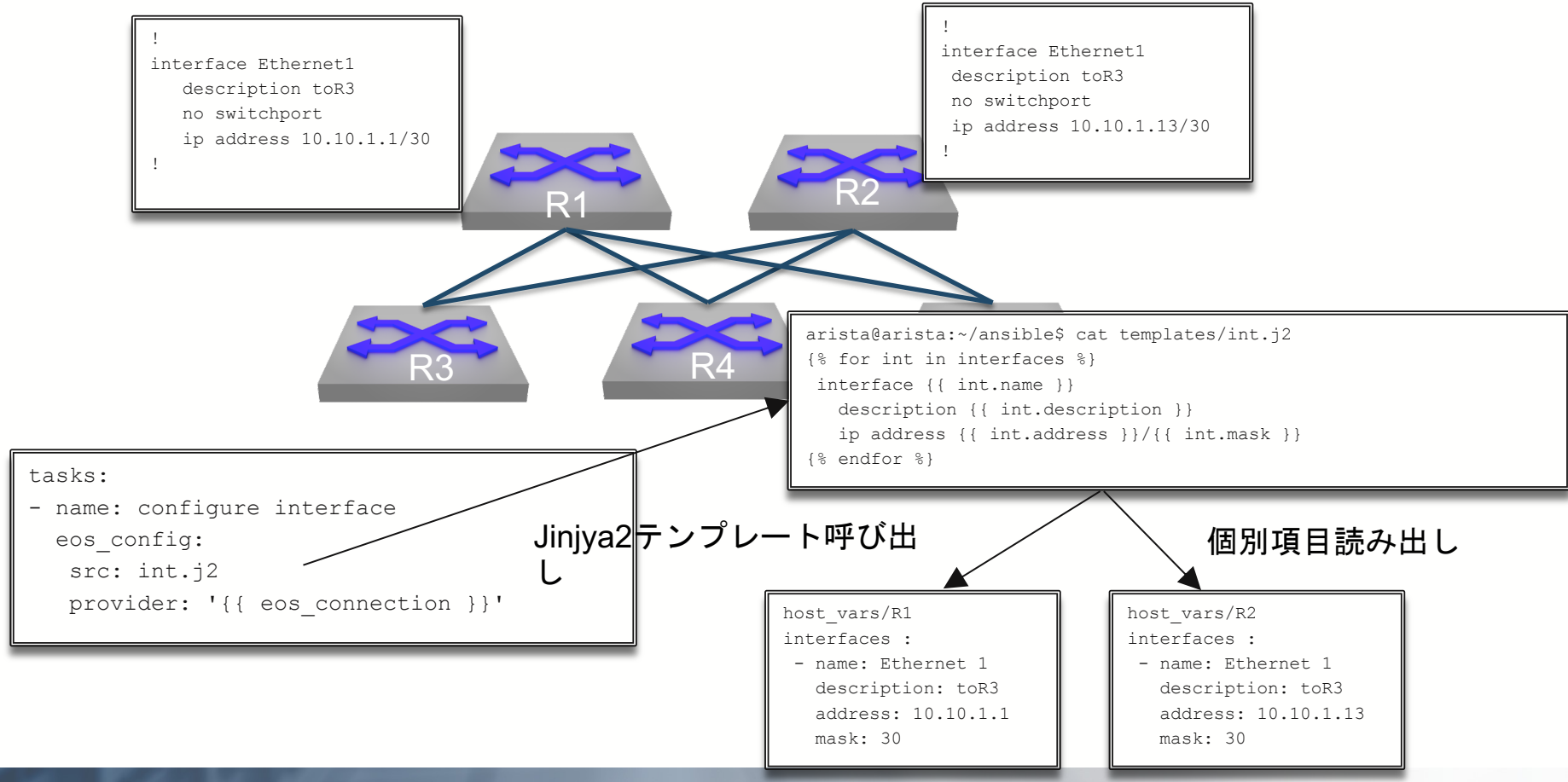
```
tasks:
  - name: configre global parameter
    eos_config:
     src: global.j2
     provider: '{{ eos_connection }}'
```

Jinjya2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



Playbook

誰がプレイをするのか？

groupには誰が所属するのか？

groupの変数

host毎の変数

taskを実行

```
.hosts
[Spine]
R1
R2
```

```
group_vars/Spine
common :
 - dnsserver: 8.8.8.8
   switchport: routed
```

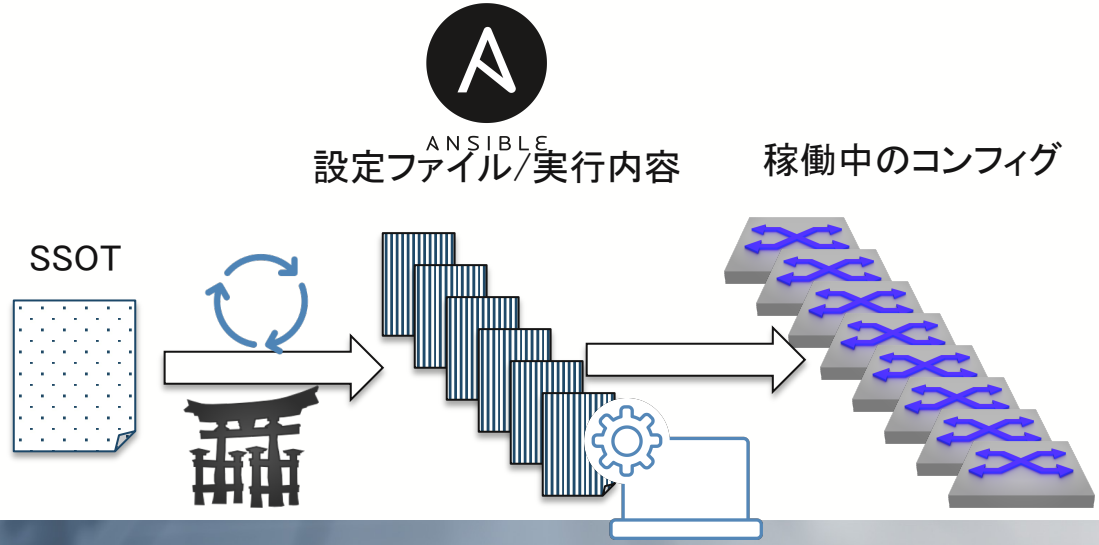
```
host_vars/R1
interfaces :
 - name: Ethernet 1
   description: toR3
   address: 10.10.1.1
   mask: 30
```

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

```
---

- hosts:  Spine
  connection: local
  gather_facts: yes

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

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

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

自動化をすすめる為に

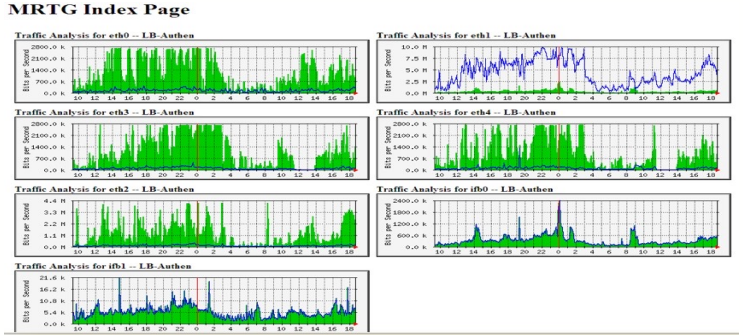
- SSOT(信頼できる唯一の情報源)を作成/まとめる
- アドホックなコマンドや簡単なワンラインのコマンドから始める
- 低リスクで高価値をものを最初に管理する: NTP, syslog, など
- 設定ファイルをテンプレートで活用出来るように考える



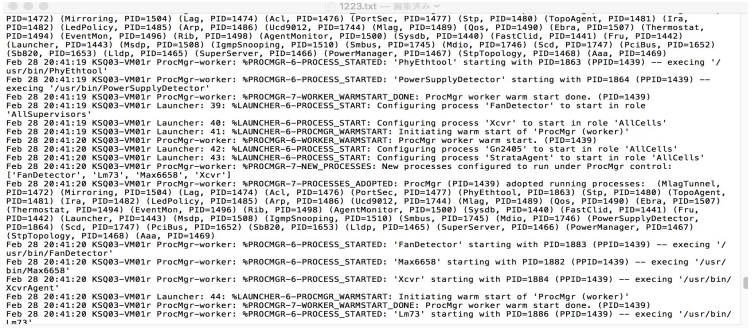
テレメトリー

従来のネットワーク運用の問題点

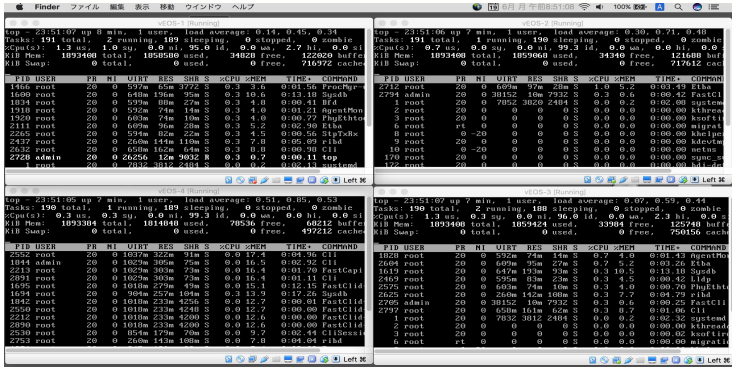
1.SNMPだと5分間隔ぐらいでしかデータを確認できない・・



2.Syslogの確認、装置が複数に及ぶとログも膨大に・・



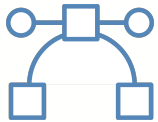
3.CLI画面たくさん同時に開き、CPU使用率などを比較・・

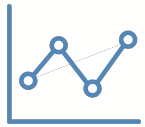


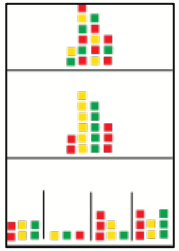
4.トラフィックフローの詳細を見るためにはflowコレクタが必要

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

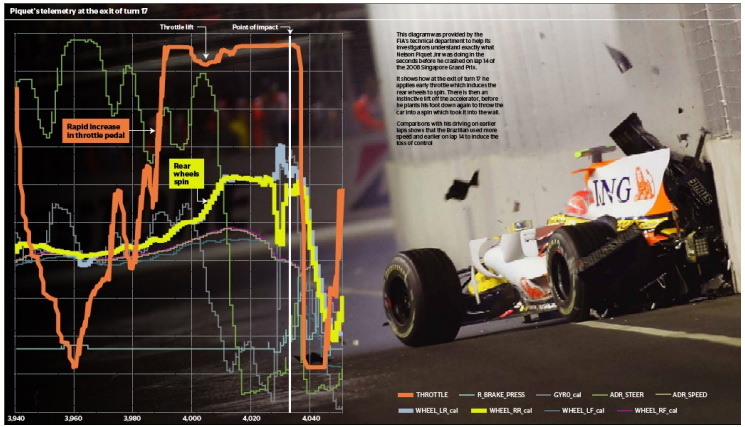






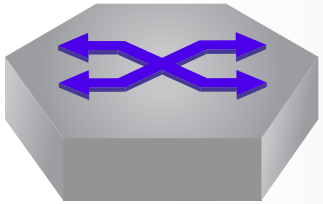
テレメトリーとは

- 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。テレメトリング。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。
複数のデータを相関関係で分析。

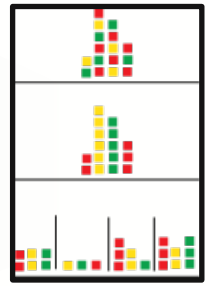
ストリーミングテレメトリー



すべての状態変更はアップデートの為のトリガーに

Interface Counters
LANZ Data
Fan Speed
sFlow Data
Temperature Sensor
Power Draw
Configurations
Routing Tables
MAC Tables
ACL Counters
VXLAN Mappings
Table Utilization
Buffer Utilization
System logs
CPU Load

State Streaming

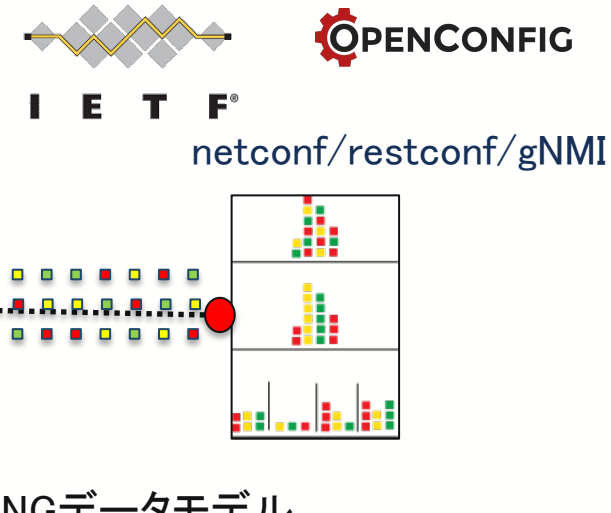


すべての状態を集中管理

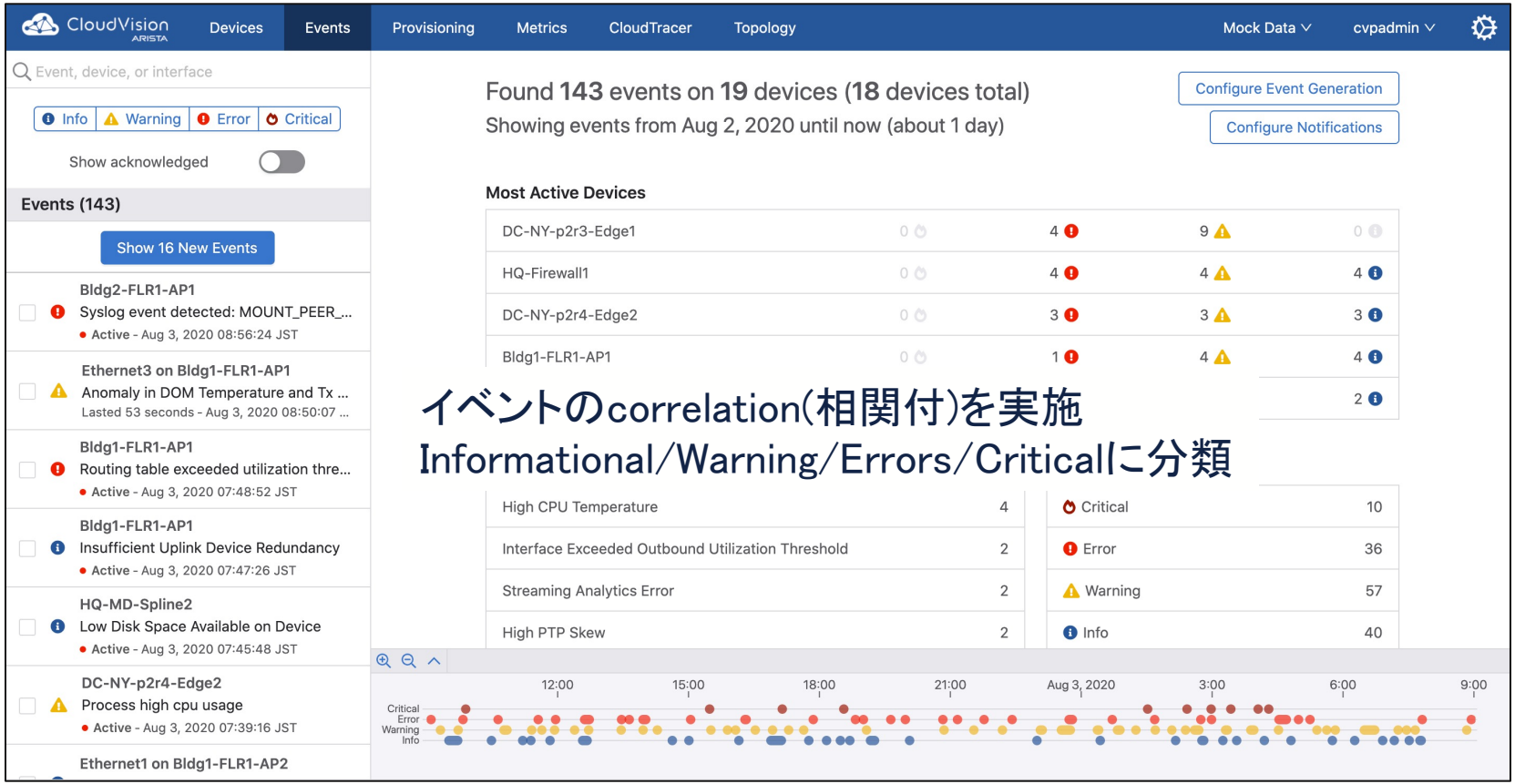
すべての状態変化をあらゆるデバイスから即座に

テレメトリーの標準化

- 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
    - 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

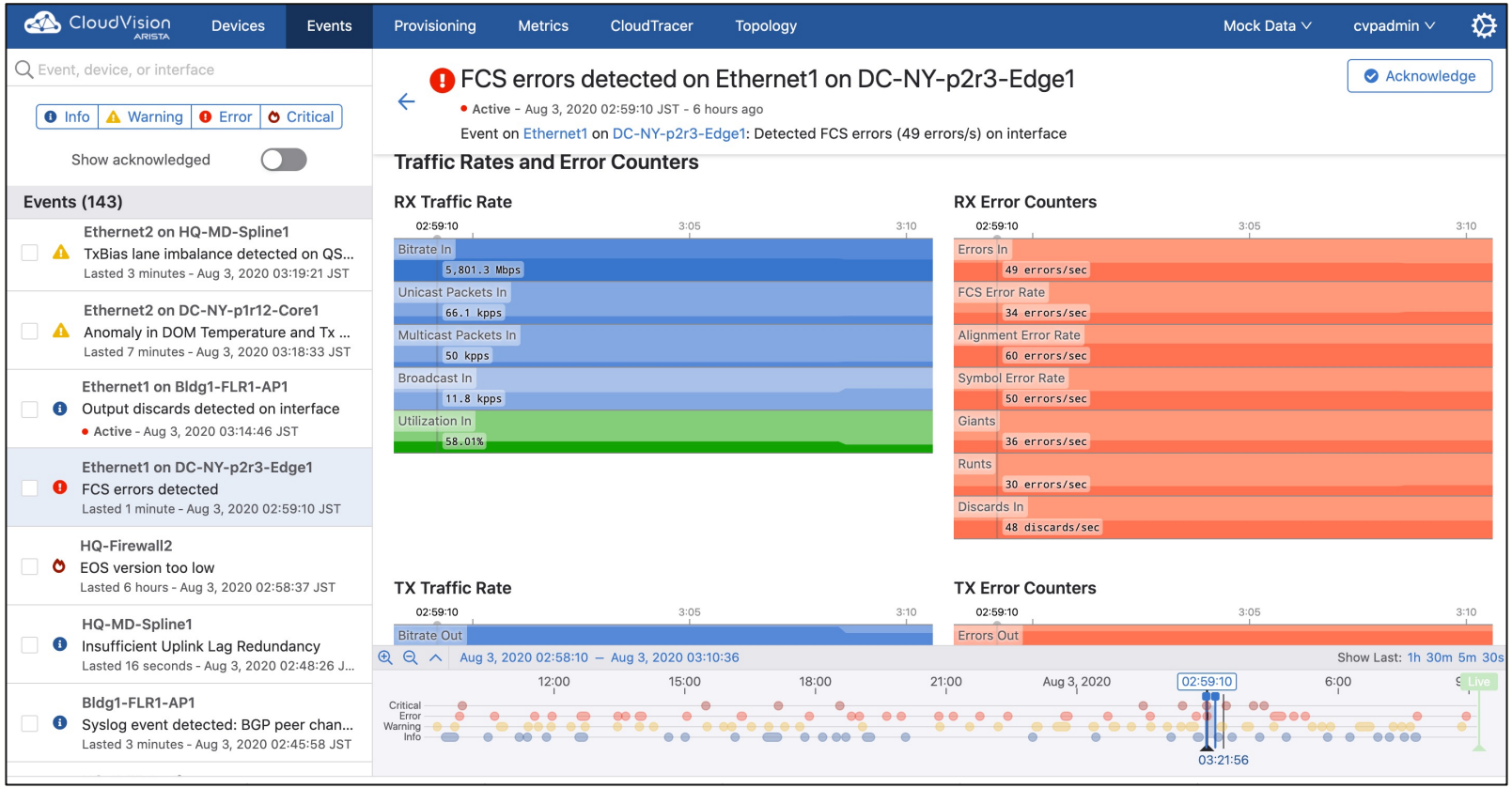


テレメトリーを利用したネットワークの見える化

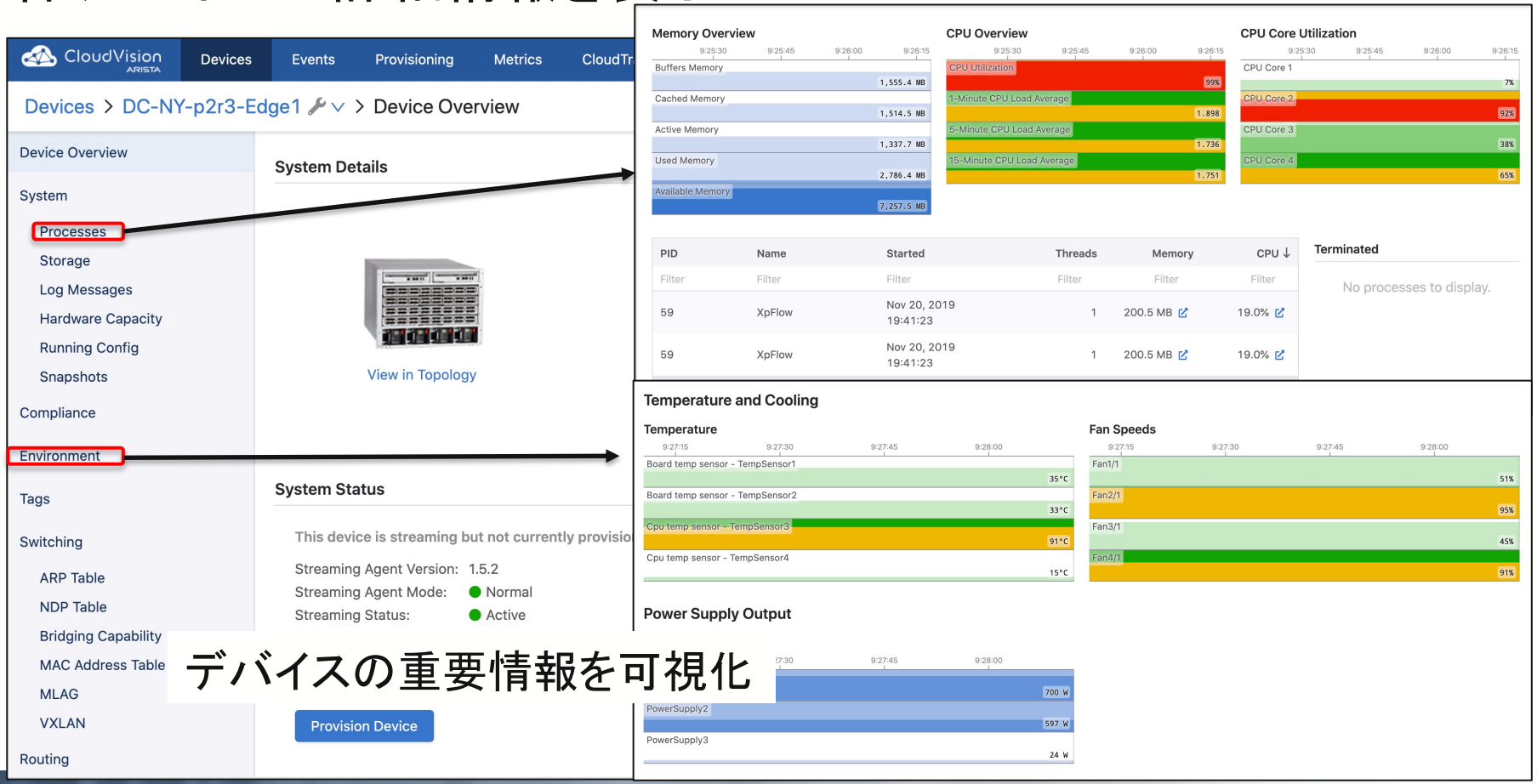


ARISTA

見つけにくいエラーも早期発見

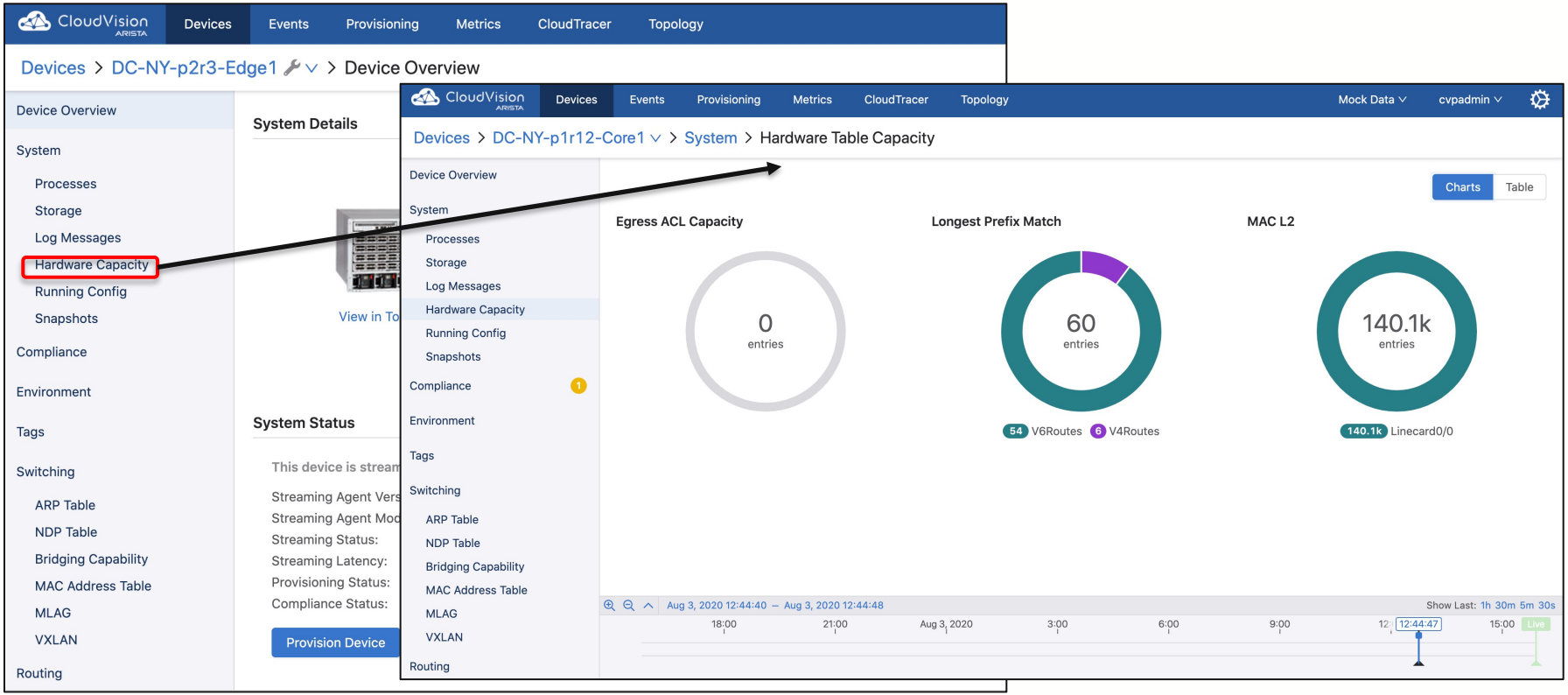


各デバイスの詳細情報を表示



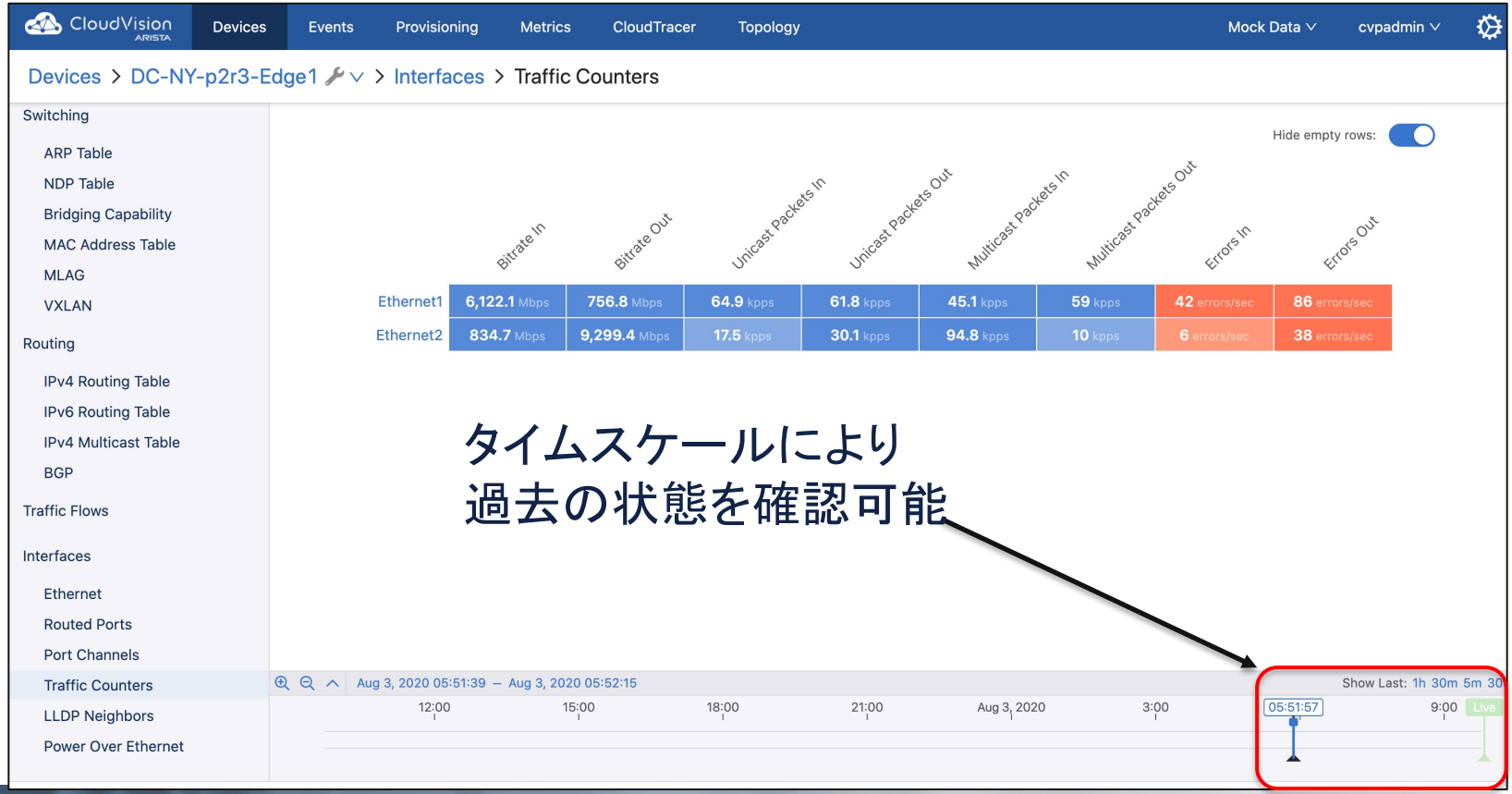
デバイスの重要情報を可視化

デバイスのハードウェアエントリーも確認可能

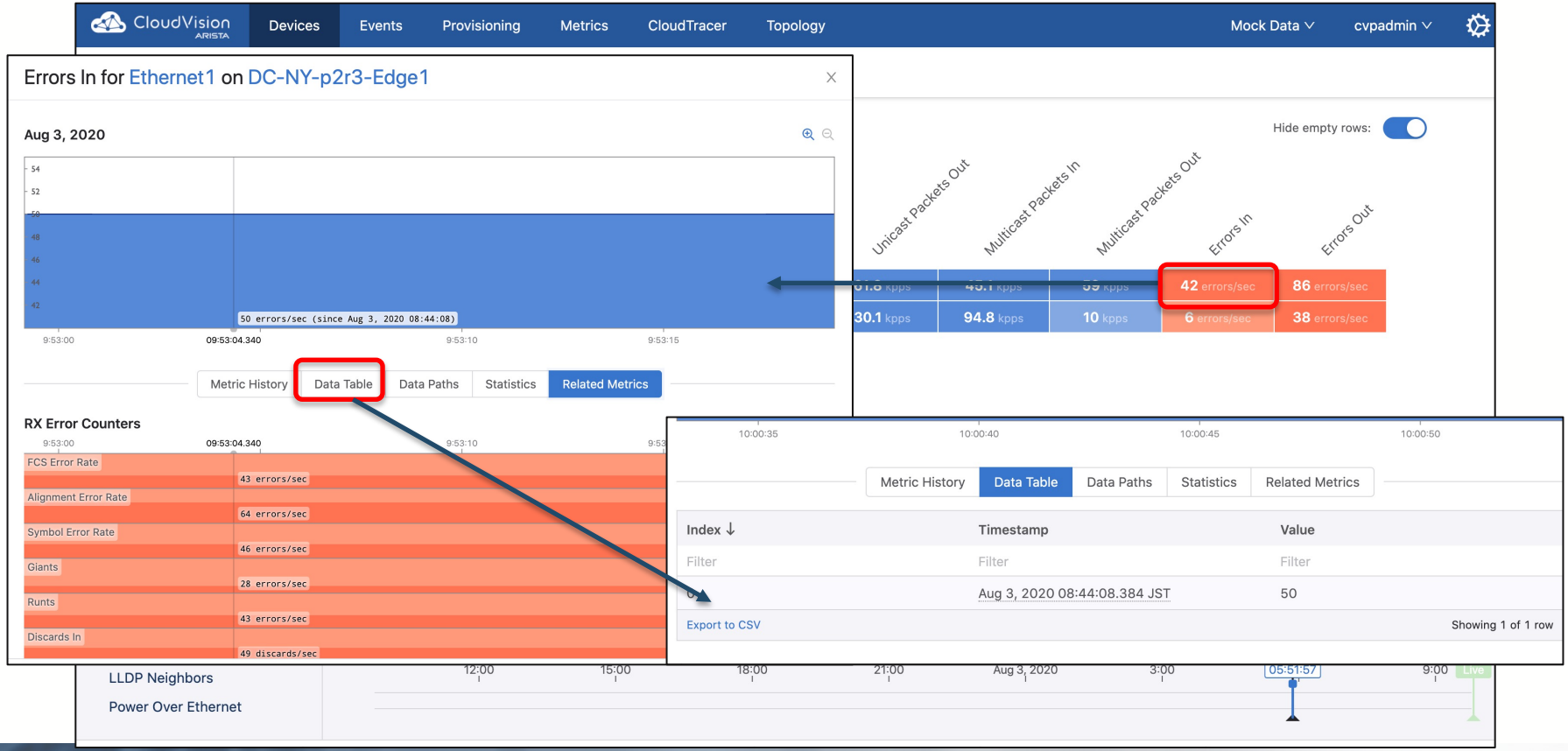


ARISTA

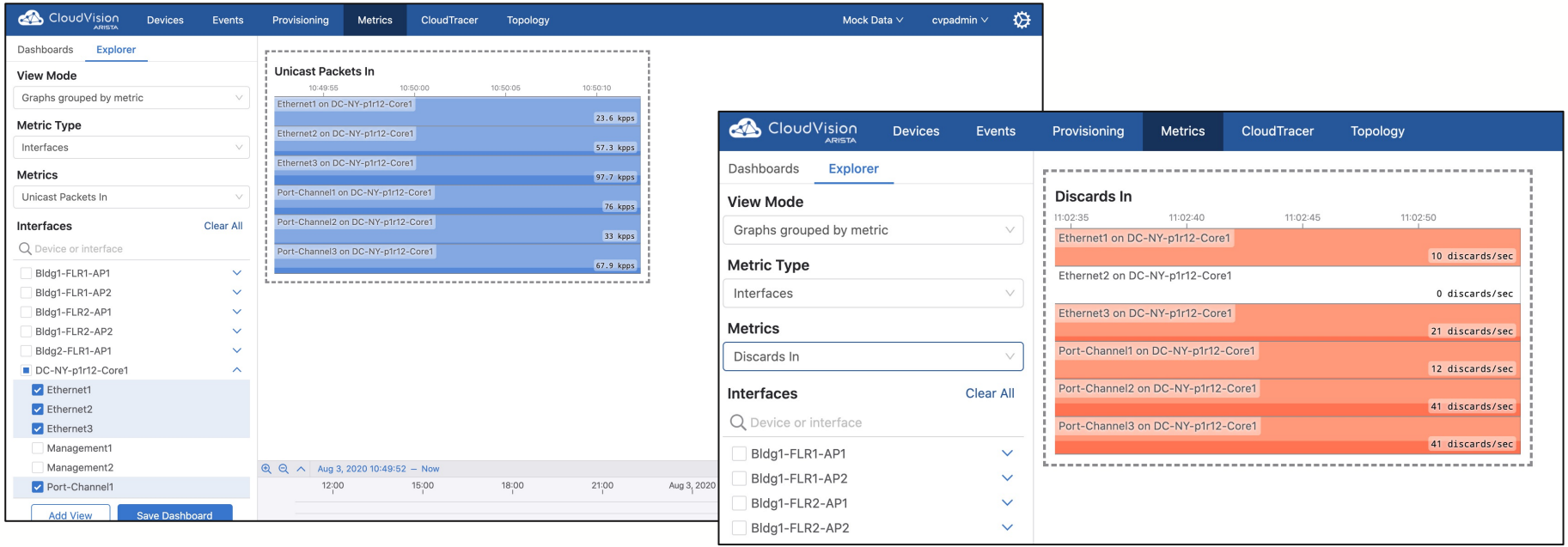
タイムスケールを用いた履歴管理



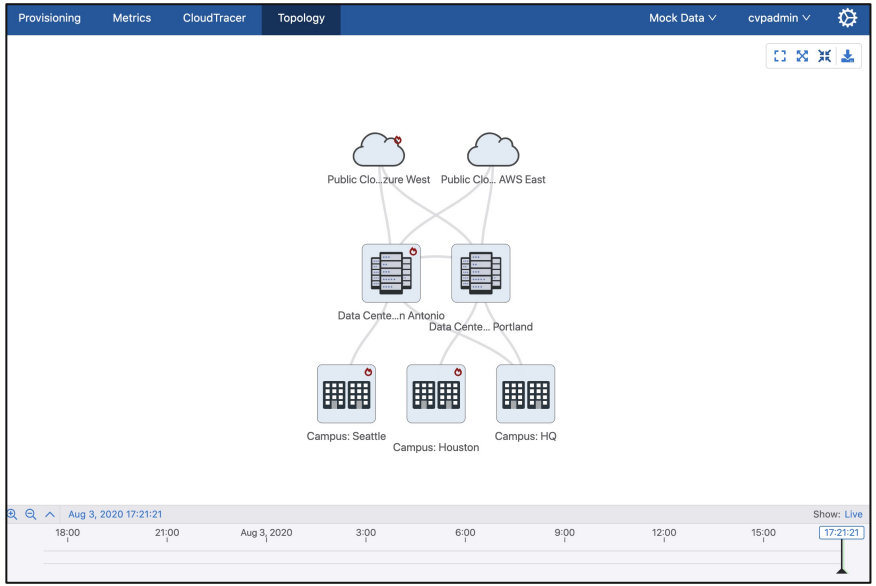
詳細履歴情報およびデータExport

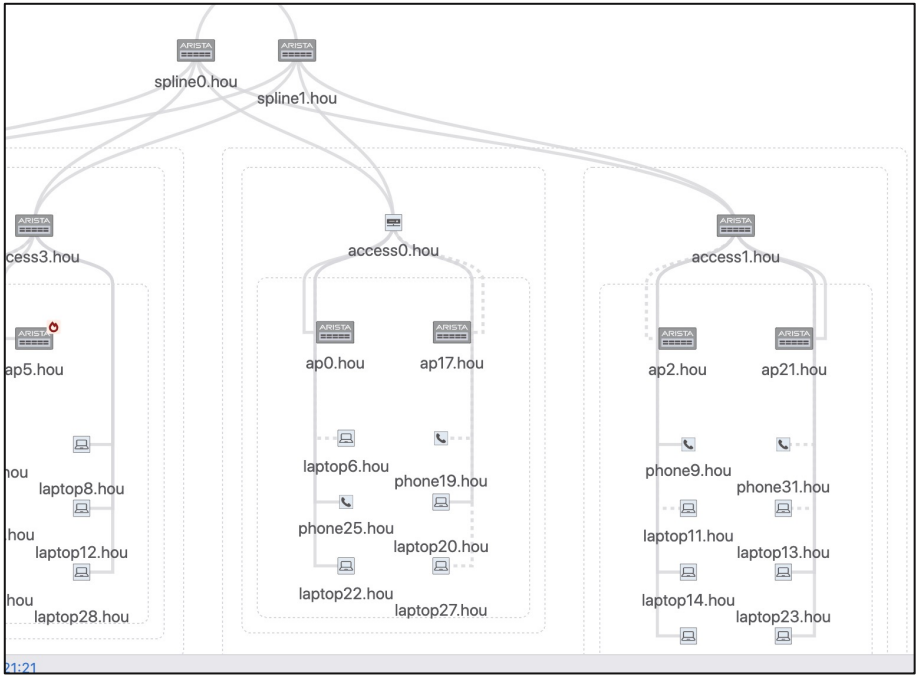


各メトリックに合わせたデバイス内/デバイス間の比較が可能



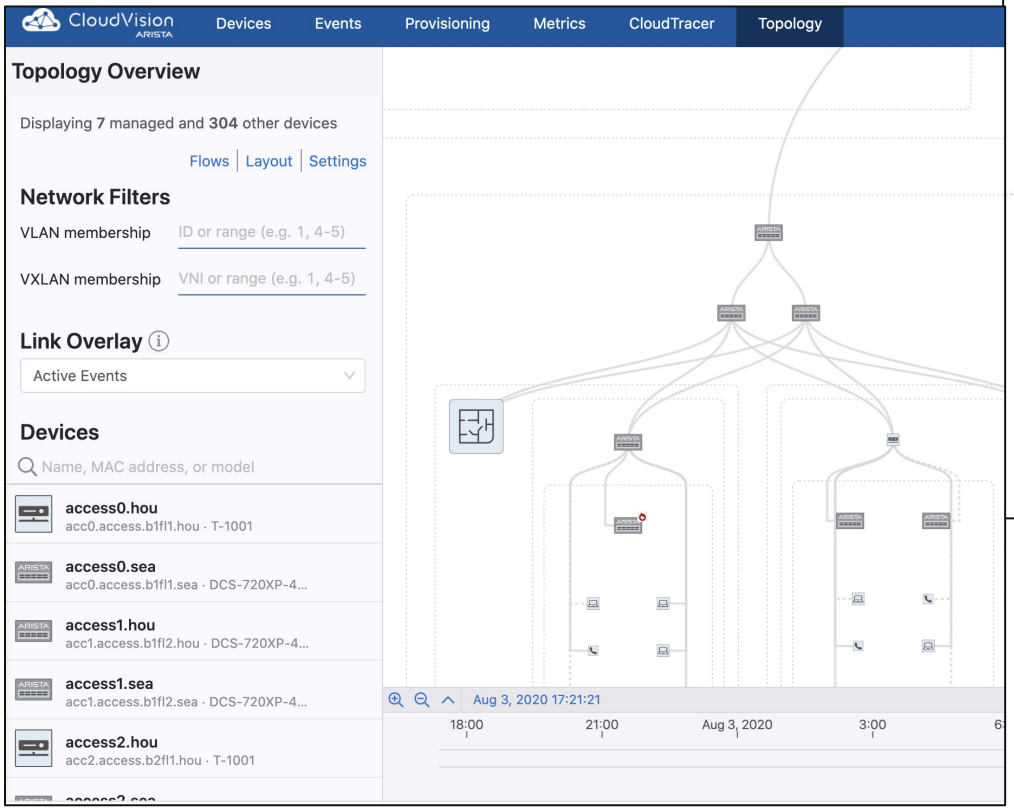
ネットワークトポロジー

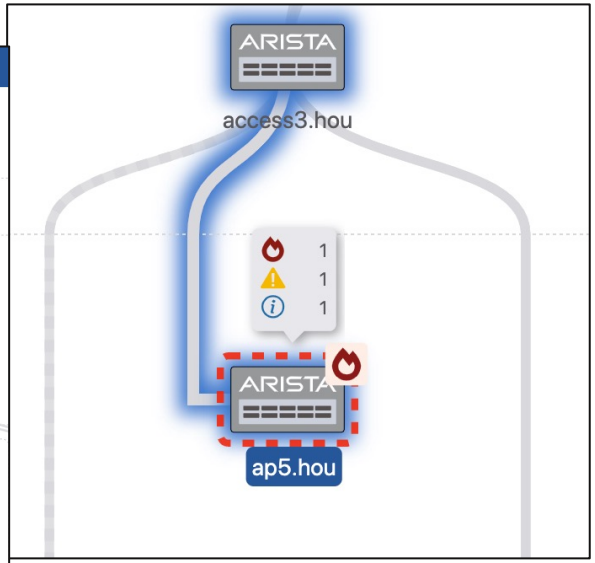




- LLDPの隣接情報を元にトポロジーを自動作成
- 標準プロトコルLLDPをサポートしていればタイプなどで判断し、トポロジーマップに登録される
- タイムスケールを持ち、時系列変更にも対応可能

Topology Overview with Active Events

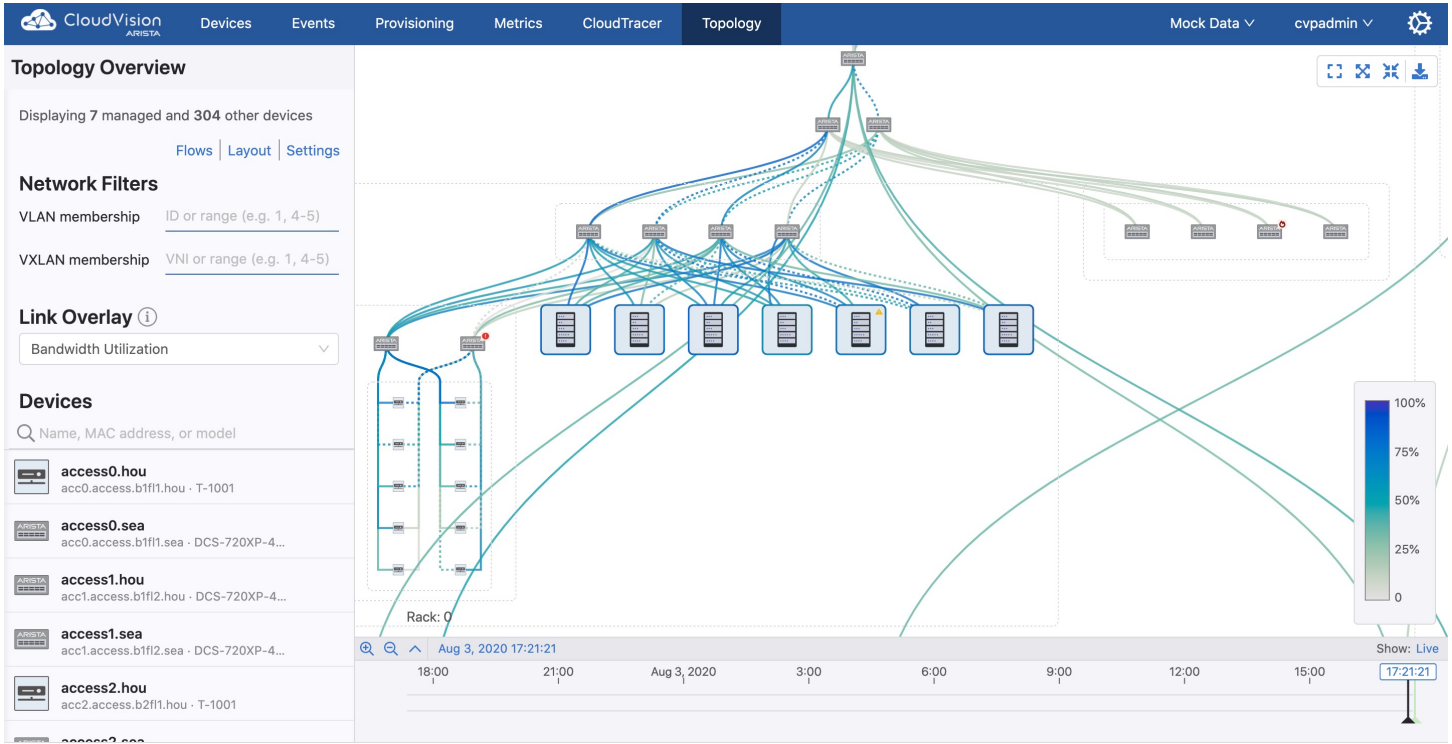




- イベントがどのノードで起きてるか早期発見可能

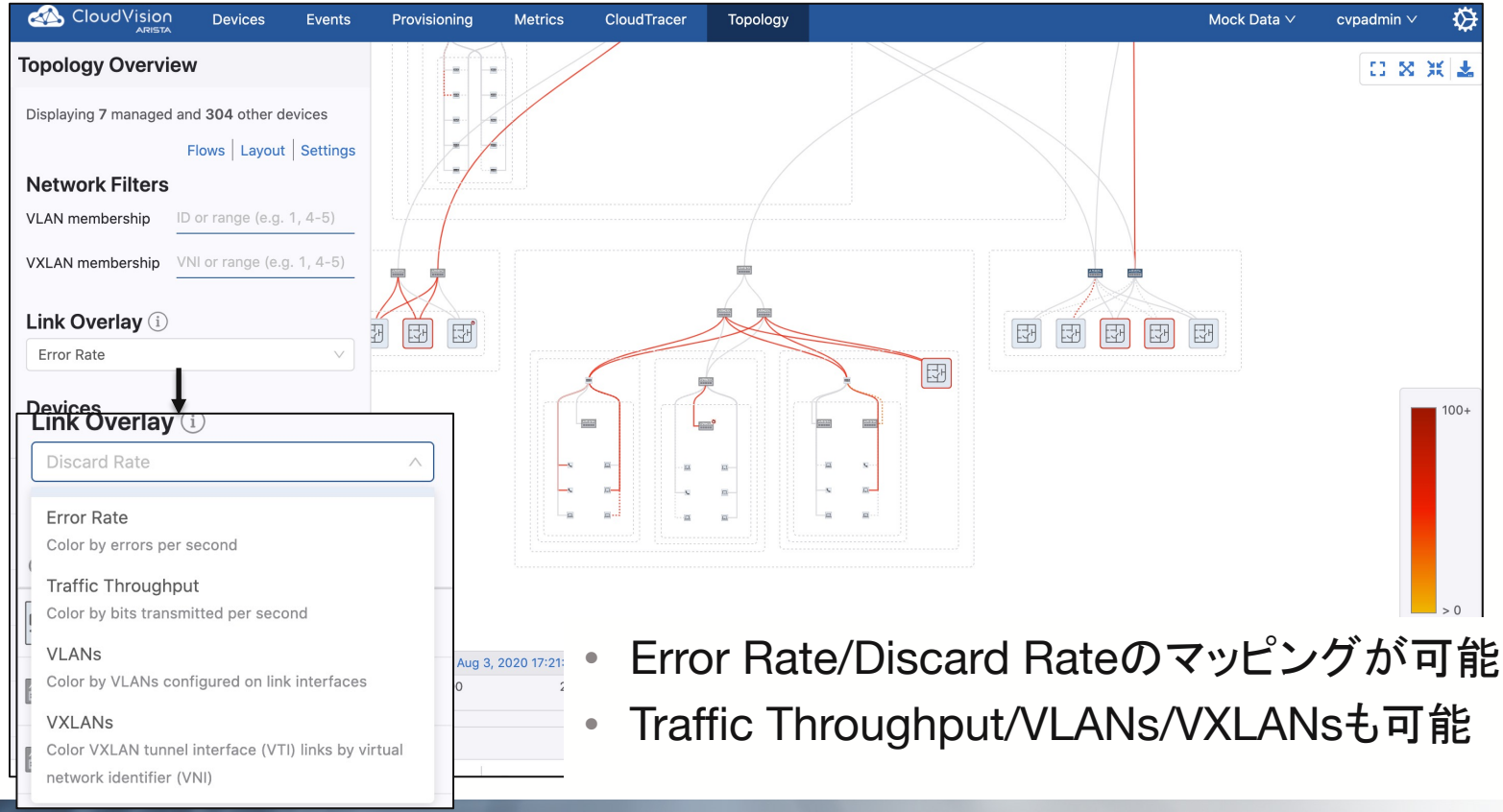
ARISTA

Topology Overview with Bandwidth Utilization



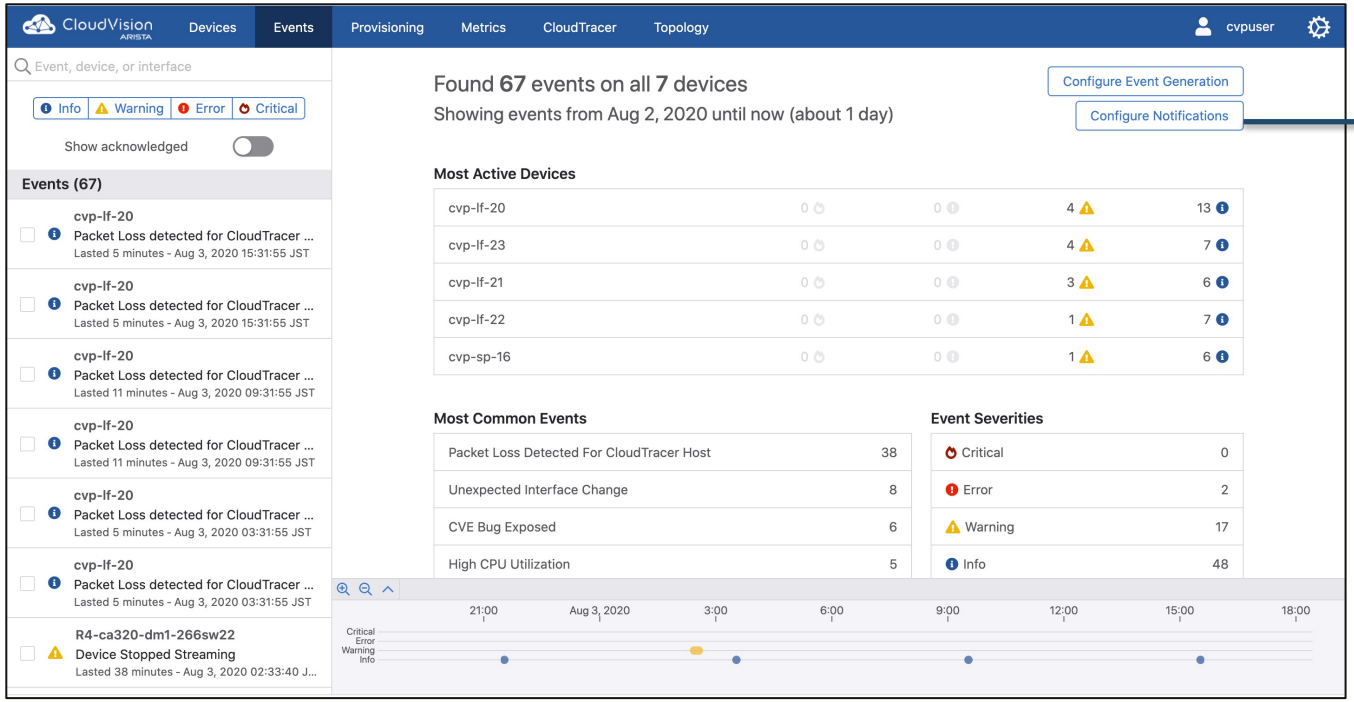
- トポロジーと使用帯域のマッピングが可能
- ネットワークウェザーマップ同様な使い方が可能

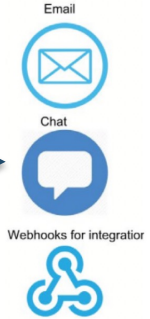
Topology Overview with Error Rate/Discard Rate



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

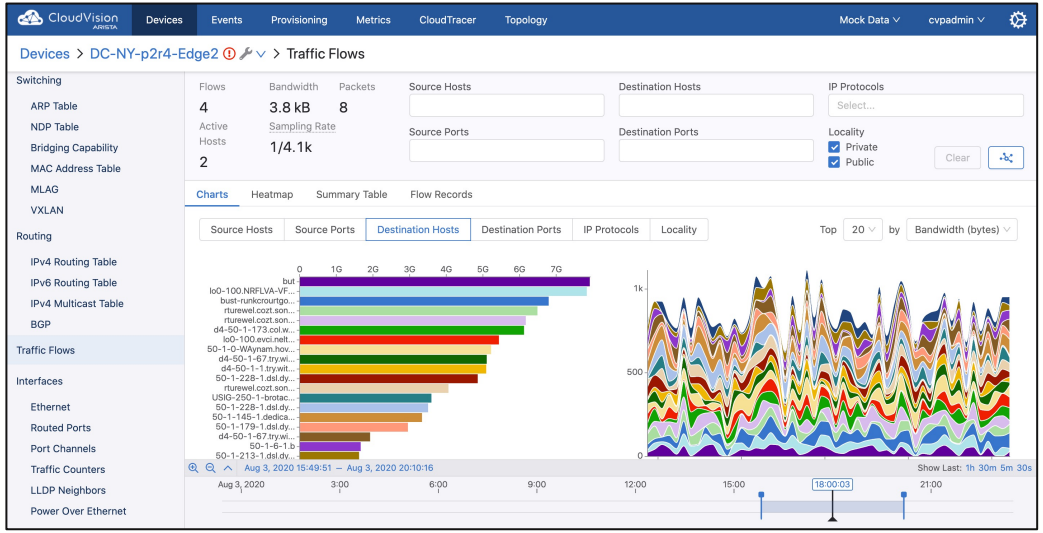
Event Alert

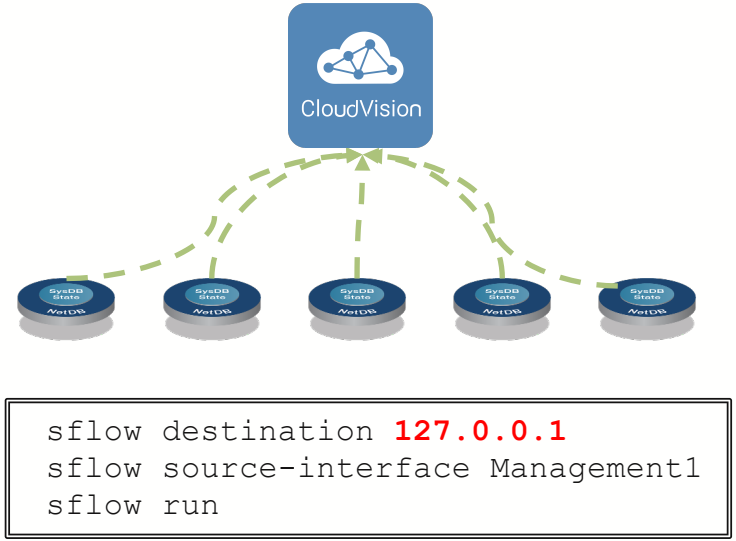




- Event情報をメール/Slackなどのツールに通知可能

Traffic Flow on テレメトリー





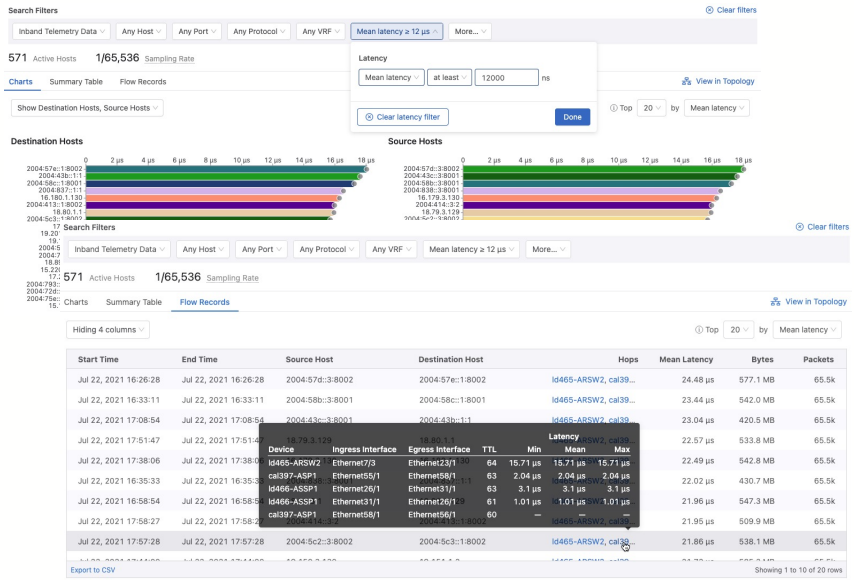

```
sflow destination 127.0.0.1
sflow source-interface Management1
sflow run
```

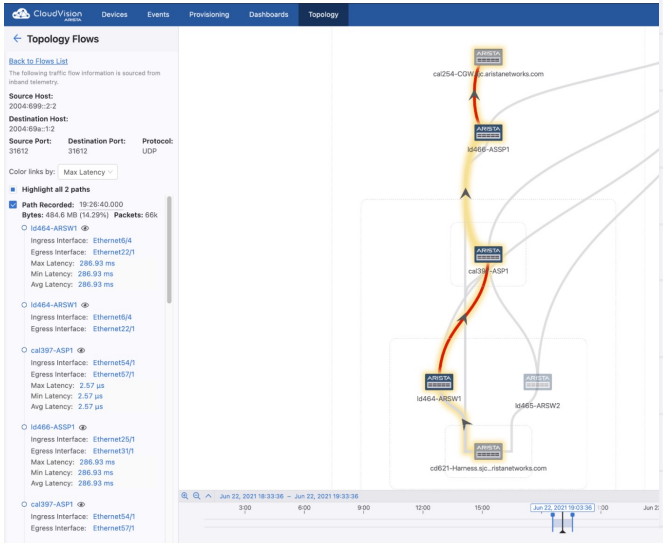
- sFlow/IPFIXでExportされたトラフィックフローデータをローカルに転送
- フロー情報をテレメトリーデータとして変換し、コレクタに送る

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

これからのテレメトリー





- データプレーンの転送時間を測定可能なin-band telemetryとの連動
- フロー情報やトポロジーマップと連動する事でより詳細のトラブルシューティング情報を得る事が出来る

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キング最前線　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キング最前線　LINE実用例
    - ≫ https://www.nic.ad.jp/iw2021/program/detail/#c6