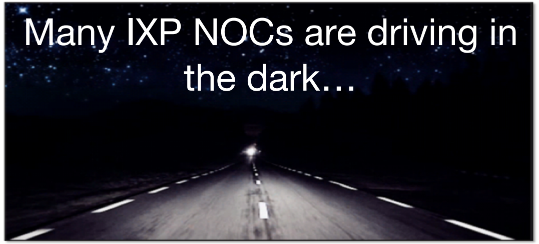
ITNO59- May 10, 2019

Davide Bassani <davide@arista.com>

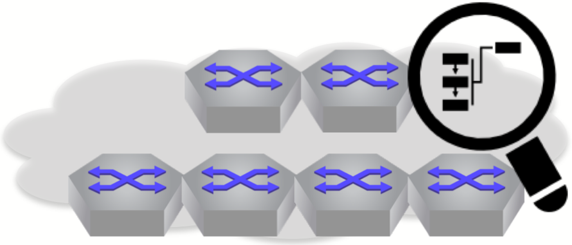
Real-time Streaming Telemetry

Is it really helpful or just another buzzword?

Today’s Telemetry Trends



Many IXP NOCs are driving in the dark…



Traditional / Legacy Approach	Cloud Telemetry Requirements
1990’s networking	Cloud DC Architectures
Polling Approach (5 min)	Real-time streaming
State scope limited to MIB definition	Complete state history
Per-Switch Per Device	Network-wide scope
Static, discrete events. Manually correlated	Dynamic event correlation

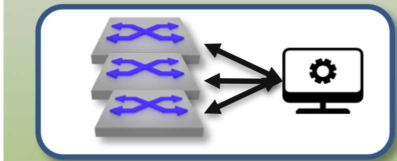
The Cloud has driven new telemetry approaches….

Telemetry Use-Cases

What is possible with a modern approach?

Real-time Monitoring

Instantaneous updates at new levels of granularity

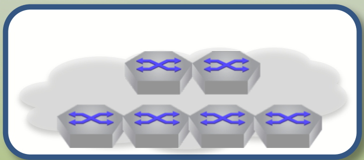


Forensic Troubleshooting

Recall historic network state for off-network analytics

Security

Real-time data for predictive security approaches



Event Correlation

Combining pieces of information to an enriched event for quick impact spotting

Improved visibility is broadly applicable

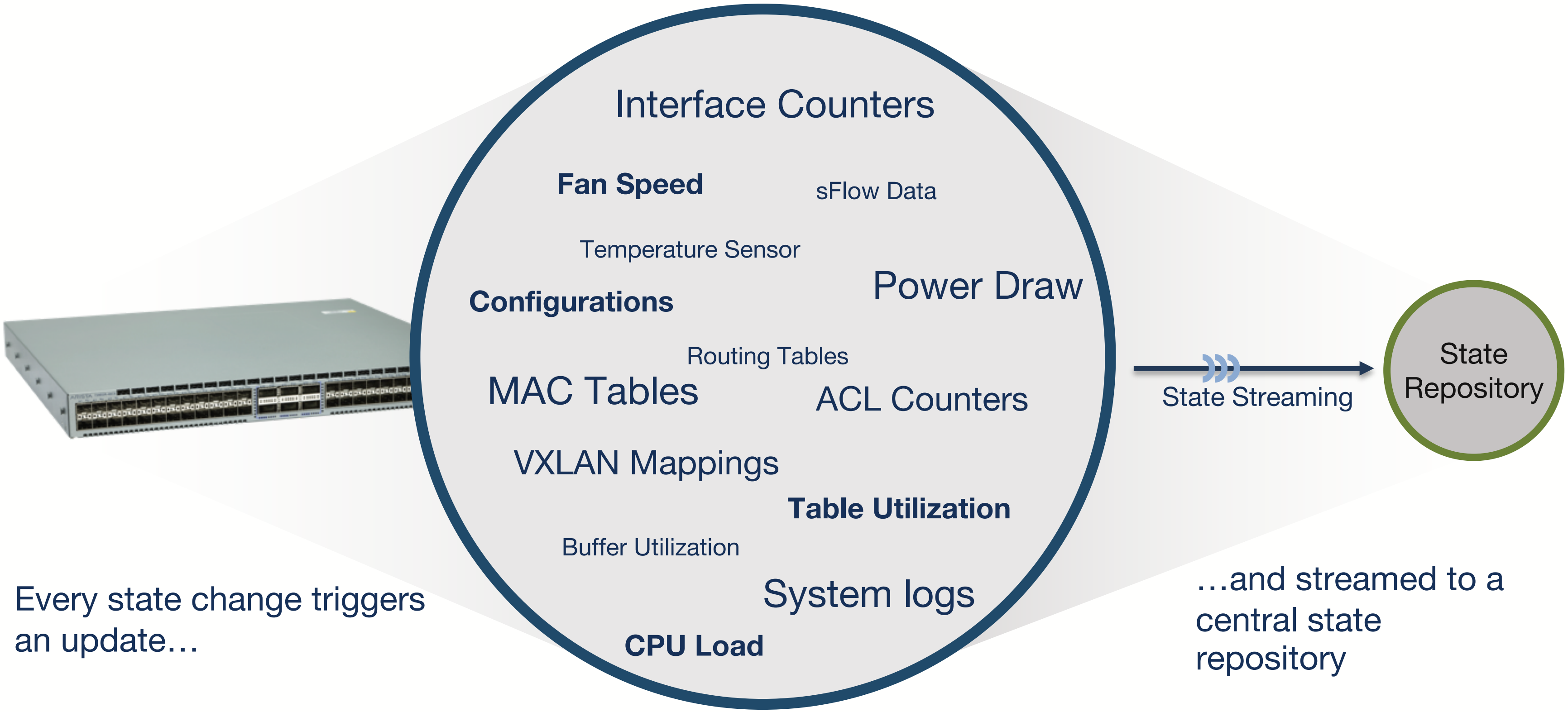
Streaming Telemetry and Analytics

1 **State Streaming Infrastructure**
Real-time streaming of events from devices w/ Open Standards

2 **Analytics Engine**
State repository providing analytics and API’s

3 **Telemetry Visualization**
Device, Event, Metric, Topology views

What is State Streaming?



Interface Counters

Fan Speed

sFlow Data

Temperature Sensor

Configurations

Power Draw

Routing Tables

MAC Tables

ACL Counters

VXLAN Mappings

Table Utilization

Buffer Utilization

System logs

CPU Load

State Streaming

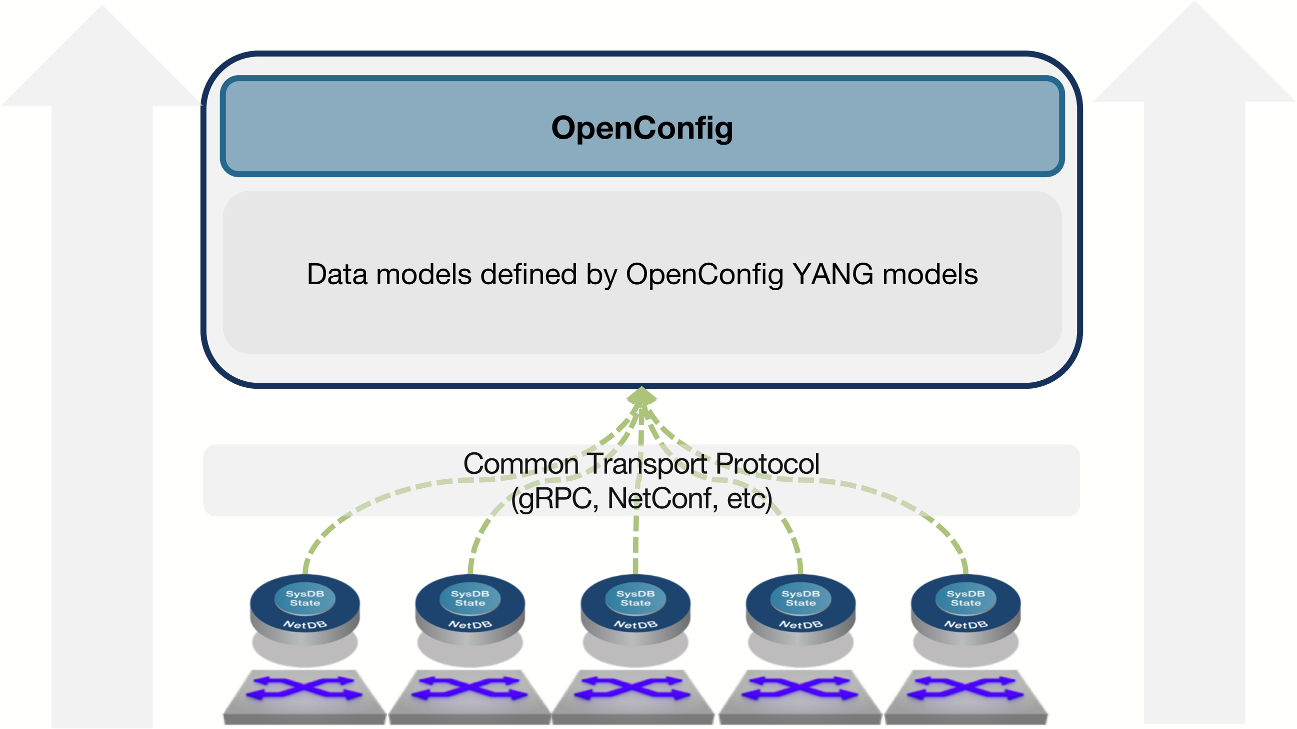
State Repository

Every state change triggers an update…

…and streamed to a central state repository

Every state change. From every device. Instantaneously.

1 Foundation for State Streaming



OpenConfig

Data models defined by OpenConfig YANG models

Common Transport Protocol
(gRPC, NetConf, etc)

Open & Standards-based APIs.

2 Analytics Engine

Three Components to the Backend Infrastructure

State Repository

High-throughput & Highly available pub/sub engine

Built on proven, scalable open source technology

Analytics Engine

Versions, aggregates, and filters raw state into actionable information:

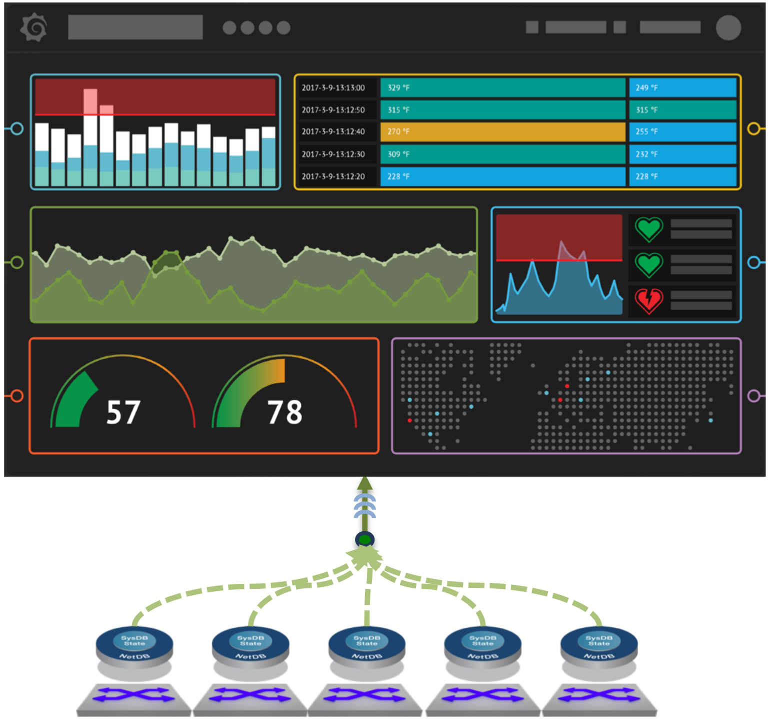
- Track trends
- Correlate data
- Detect anomalies

API Server

Standard APIs accessed via REST, Websocket, or gRPC

Query historical state and subscribe to streaming updates

3 Telemetry Visualization

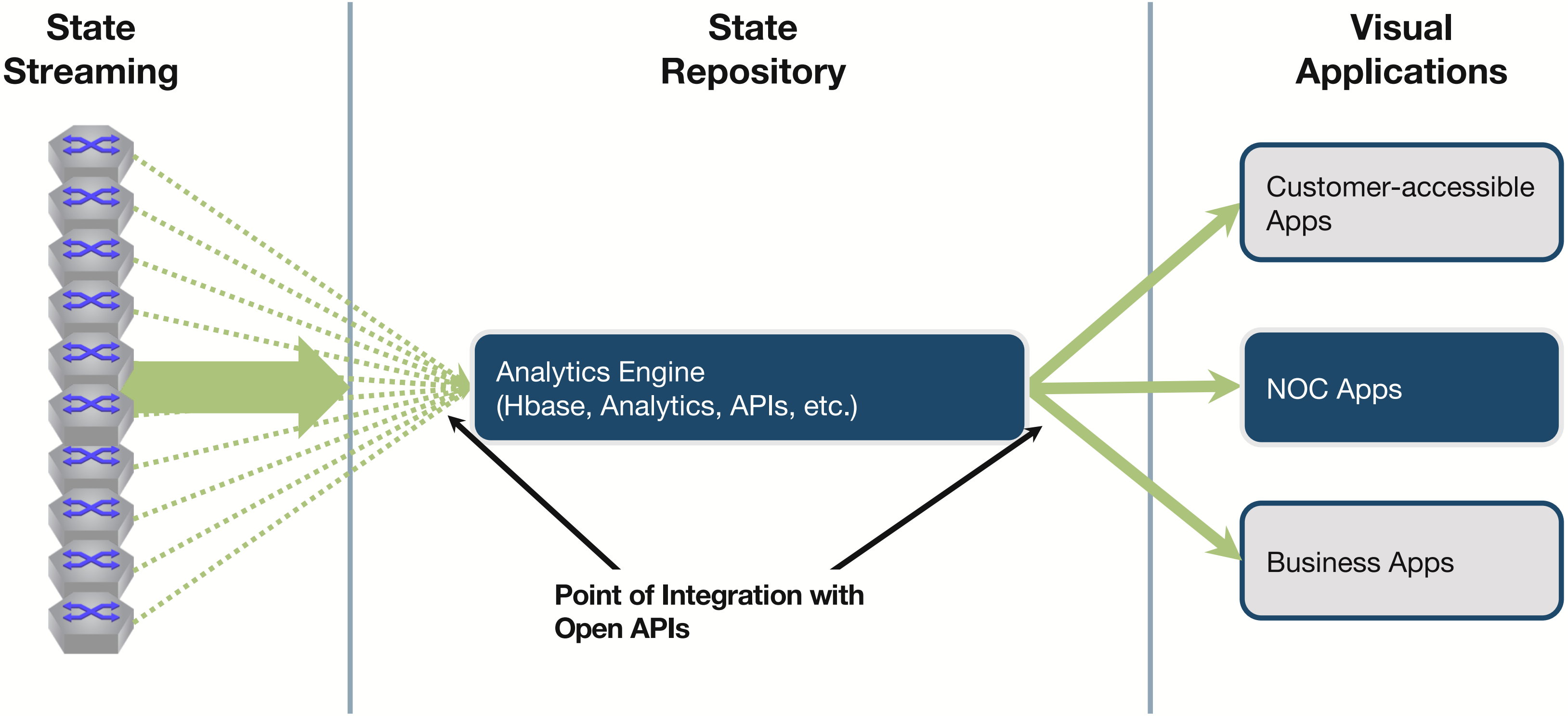


2017-3-9-13:13:00	329 °F	249 °F
2017-3-9-13:12:50	315 °F	315 °F
2017-3-9-13:12:40	270 °F	255 °F
2017-3-9-13:12:30	309 °F	232 °F
2017-3-9-13:12:20	228 °F	228 °F

Complete, real-time state streaming

- Telemetry Apps provide front-end for visibility network state
  - Correlation of network-wide data
  - Views: Event, Device, Metric, and more
  - Timeline view for better historic troubleshooting
  - APIs for customer & partner apps

3 Analytics Open Framework



State Streaming

State Repository

Visual Applications

Analytics Engine
(Hbase, Analytics, APIs, etc.)

Customer-accessible Apps

NOC Apps

Business Apps

Point of Integration with Open APIs

J

Building Your Own Telemetry System

(i.e. how a hyper-scale cloud operator might build a telemetry platform)

Component	Distributed Key-value Database	Queuing System	Analytics Pipeline	Search Engine	Visualization
Options	HBase, Cassandra, Kudu, Druid, Prometheus, etc	Kafka, ActiveMQ, ZeroMQ, RabbitMQ	Spark, Storm, Heron, etc	Elastic, Solr	Kibana, Grafana
Arista Telemetry	HBase	Kafka	CloudVision Turbines	New Elasticsearch	CloudVision Telemetry UI

Telemetry based on cloud scale approaches

Use Cases examples

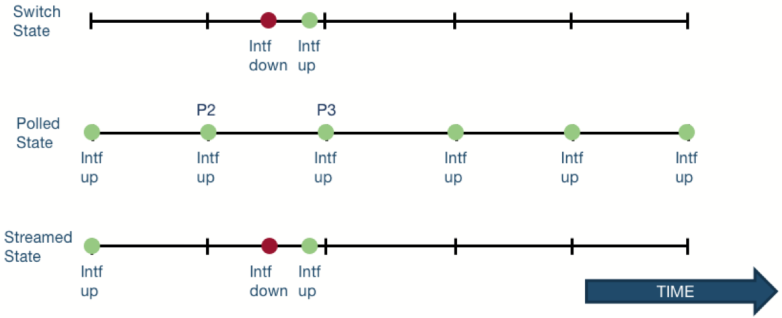
Data Collection

- Data being provided ‘near real-time’ (within seconds) instead of pre-defined polling intervals

- Retrieve all available data from the switch (or just the ones you like)
  - Device health (Temperature, fan, memory, CPU, power, etc.)
  - Network health (Optical levels, interface counters, ACL violations, QoS drops, etc.)

- Reduce load on collectors and network devices
  - No unnecessary information being repeatedly processed

- Keep historic values as detailed as you like
  - Aggregation of values of time is up to your collector/database, but not a must

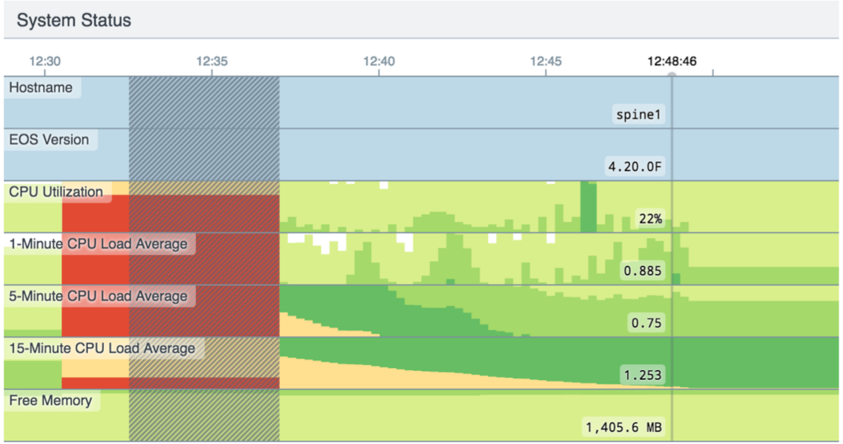


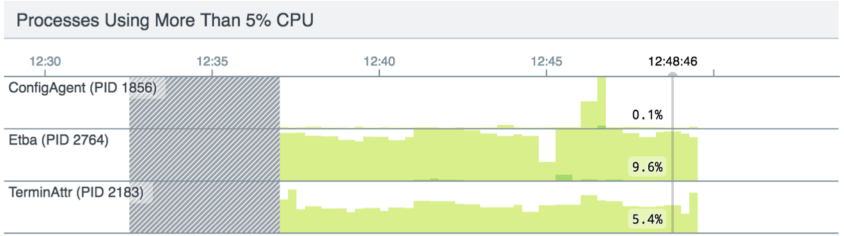
Data Collection

⚠ High CPU load average on spine1

Apr 10, 2018 12:48:46 CEST • a few seconds ago

Event on spine1: Device's 15 minute CPU load average exceeded threshold of 1.2


System Status


Processes Using More Than 5% CPU

Monitoring / ACL counters

- Use Case
  - Maintain a list of allowed/forbidden protocols and protect the shared infrastructure with ACLs

- Reality
  - Once the customer is out of quarantine, his connection will be ACL’d but increasing counters are only being looked at when an issue occurs. This is also not something being monitored by existing SNMP solutions.

- Approach
  - Being proactively informed when a Production Customer is violating the ACLs and automatically inform him about it

Monitoring / Microbursts

- Use Case
  - Especially with increased Content to Eyeball traffic you are likely to see more microbursts during ‘release’ windows.

- Reality
  - Interface counters (customer & backbone) are queried on a 1 to 5 minute average. Shorts bursts are flattened out and congestion of backbone interfaces might not be detected. This can cause severe impact to a large chunk of the customers.

- Approach
  - The Telemetry agent on the network device can provide more granular interface statistics. This can be brought down to 5 seconds per metric and enables operations to detect congestion quickly.

Monitoring / DDM/DOM monitoring

- Use Case
  - Over time optics may degrade on the transmit/receive side (’optic becomes blind’) leading to uncontrolled outages on either the backbone- or customer-facing side.

- Reality
  - Not all vendors provide implementation of DDM-MIB on SNMP. Also due to the aggregation of data with conventional tools the usefulness is not really given.

- Approach
  - Telemetry can be combined with Anomaly Detection and/or Machine Learning technologies to provide prediction mechanisms on when an issue could arise.

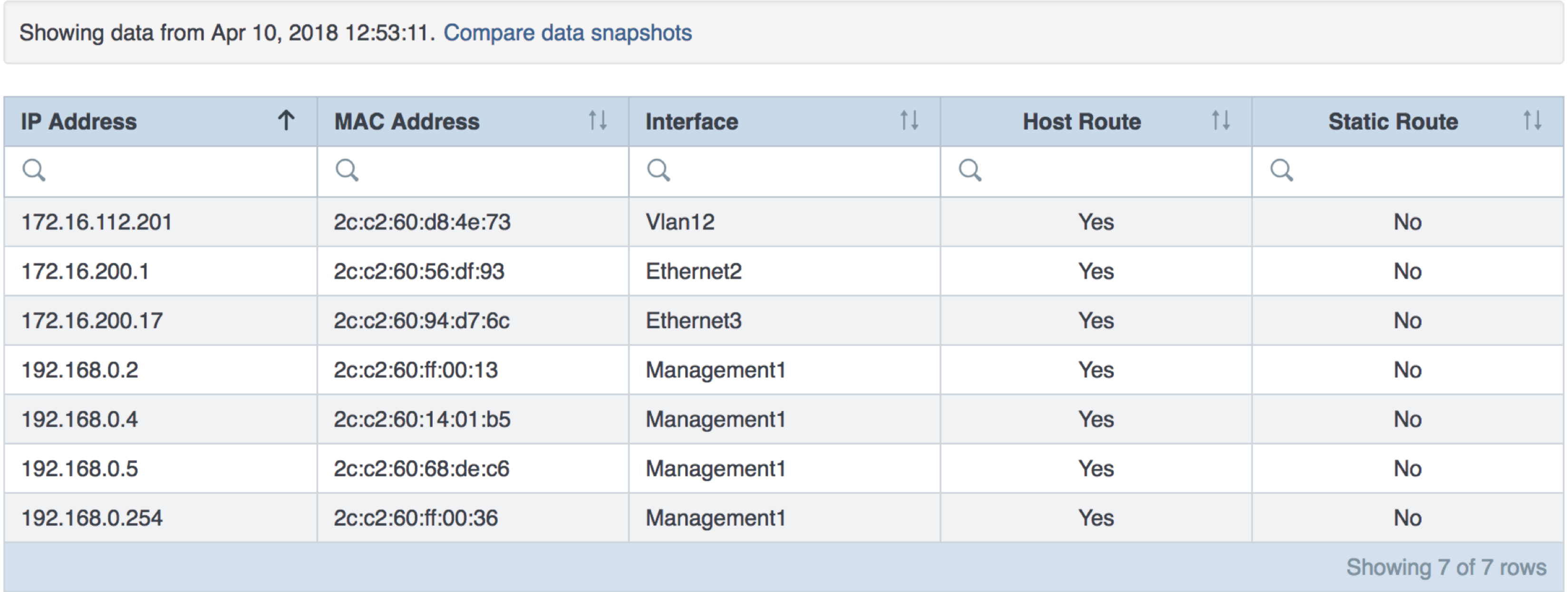
Monitoring / Proxy ARP detection

- Use Case
  - Misconfiguration of a customer interface with Proxy ARP can lead to network-wide issues and customers outages.

- Reality
  - It can be relatively easy to spot the misbehaving party, but it’s hard to spot the issue in arrears. This is the case when the ‘issue fixed itself’.

- Approach
  - With the historic information provided by the Telemetry database it is easy to ‘go back in time’ and pin down the rogue.

Monitoring / Proxy ARP detection



Showing data from Apr 10, 2018 12:53:11. Compare data snapshots

IP Address ↑	MAC Address ⇅	Interface ⇅	Host Route ⇅	Static Route ⇅
🔍	🔍	🔍	🔍	🔍
172.16.112.201	2c:c2:60:d8:4e:73	Vlan12	Yes	No
172.16.200.1	2c:c2:60:56:df:93	Ethernet2	Yes	No
172.16.200.17	2c:c2:60:94:d7:6c	Ethernet3	Yes	No
192.168.0.2	2c:c2:60:ff:00:13	Management1	Yes	No
192.168.0.4	2c:c2:60:14:01:b5	Management1	Yes	No
192.168.0.5	2c:c2:60:68:de:c6	Management1	Yes	No
192.168.0.254	2c:c2:60:ff:00:36	Management1	Yes	No

Showing 7 of 7 rows

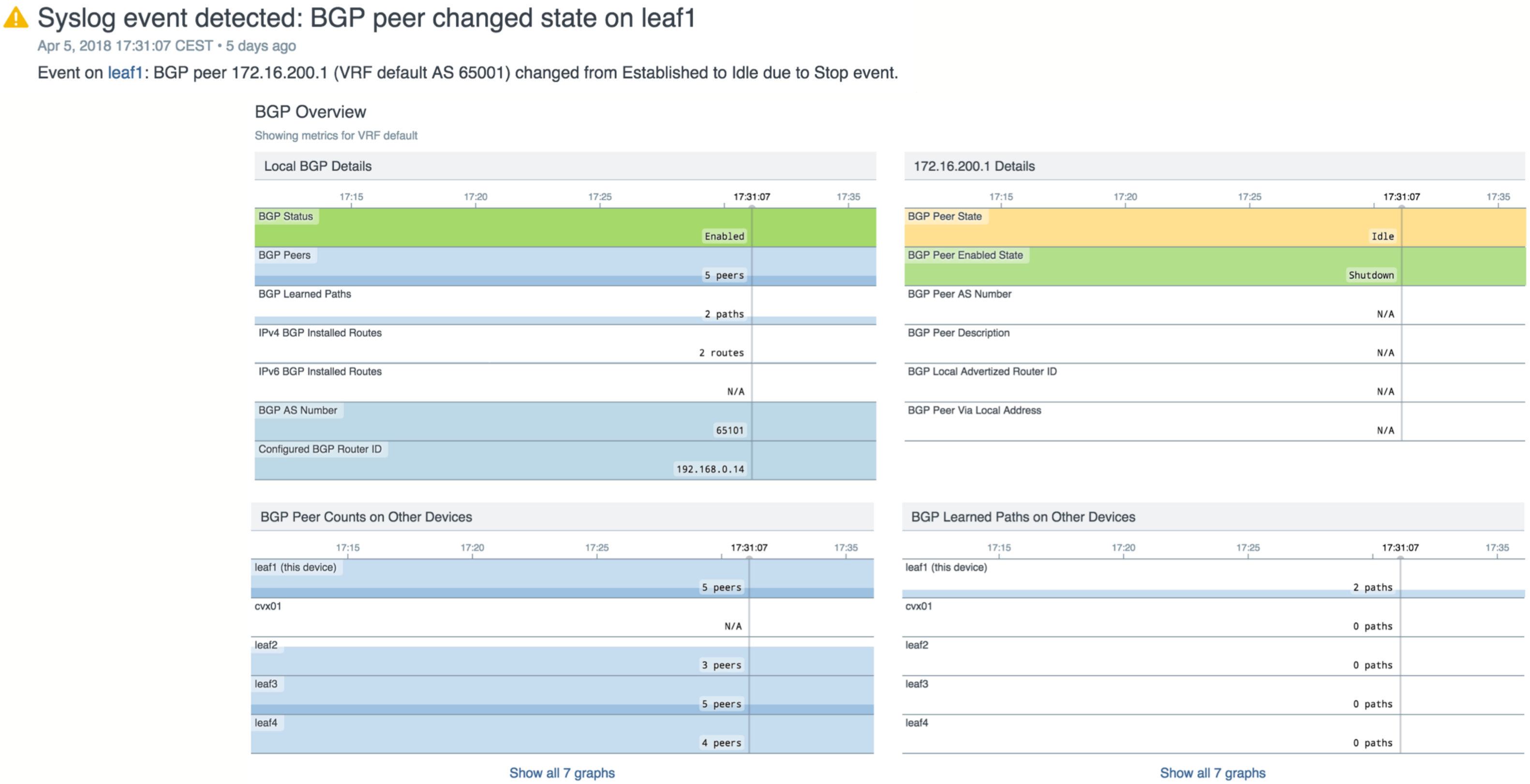
Event Correlation

- Use Case
  - Event generation can lead to an ‘overflow of information’ and takes an operator quite a while to actually find the root-cause and the customer impact.

- Reality
  - An event comes in, several commands are executed on the CLI to check customer impact and various other factors.

- Approach
  - Providing event-specific information (MAC addresses, optical levels of the interface, throughput, discards, etc.) around the device and network health with a timeline before and after the event helps to easily spot all relevant details for further troubleshooting and where to start.

Event Correlation



Syslog event detected: BGP peer changed state on leaf1

Apr 5, 2018 17:31:07 CEST • 5 days ago

Event on leaf1: BGP peer 172.16.200.1 (VRF default AS 65001) changed from Established to Idle due to Stop event.

BGP Overview

Showing metrics for VRF default

Local BGP Details

BGP Status	Enabled
BGP Peers	5 peers
BGP Learned Paths	2 paths
IPv4 BGP Installed Routes	2 routes
IPv6 BGP Installed Routes	N/A
BGP AS Number	65101
Configured BGP Router ID	192.168.0.14

172.16.200.1 Details

BGP Peer State	Idle
BGP Peer Enabled State	Shutdown
BGP Peer AS Number	N/A
BGP Peer Description	N/A
BGP Local Advertized Router ID	N/A
BGP Peer Via Local Address	N/A

BGP Peer Counts on Other Devices

leaf1 (this device)	5 peers
cvx01	N/A
leaf2	3 peers
leaf3	5 peers
leaf4	4 peers

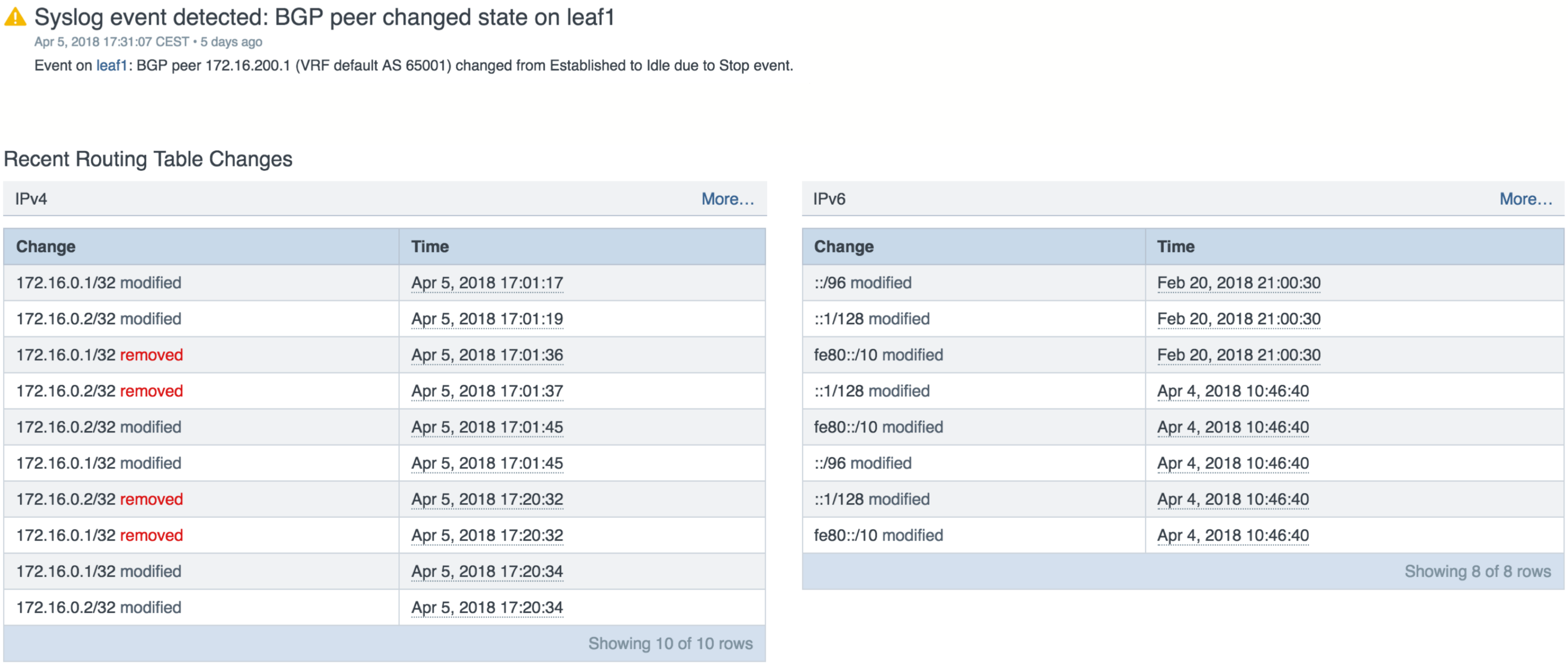
Show all 7 graphs

BGP Learned Paths on Other Devices

leaf1 (this device)	2 paths
cvx01	0 paths
leaf2	0 paths
leaf3	0 paths
leaf4	0 paths

Show all 7 graphs

Event Correlation



⚠ Syslog event detected: BGP peer changed state on leaf1

Apr 5, 2018 17:31:07 CEST • 5 days ago

Event on leaf1: BGP peer 172.16.200.1 (VRF default AS 65001) changed from Established to Idle due to Stop event.

Recent Routing Table Changes

IPv4 More…

Change	Time
172.16.0.1/32 modified	Apr 5, 2018 17:01:17
172.16.0.2/32 modified	Apr 5, 2018 17:01:19
172.16.0.1/32 removed	Apr 5, 2018 17:01:36
172.16.0.2/32 removed	Apr 5, 2018 17:01:37
172.16.0.2/32 modified	Apr 5, 2018 17:01:45
172.16.0.1/32 modified	Apr 5, 2018 17:01:45
172.16.0.2/32 removed	Apr 5, 2018 17:20:32
172.16.0.1/32 removed	Apr 5, 2018 17:20:32
172.16.0.1/32 modified	Apr 5, 2018 17:20:34
172.16.0.2/32 modified	Apr 5, 2018 17:20:34

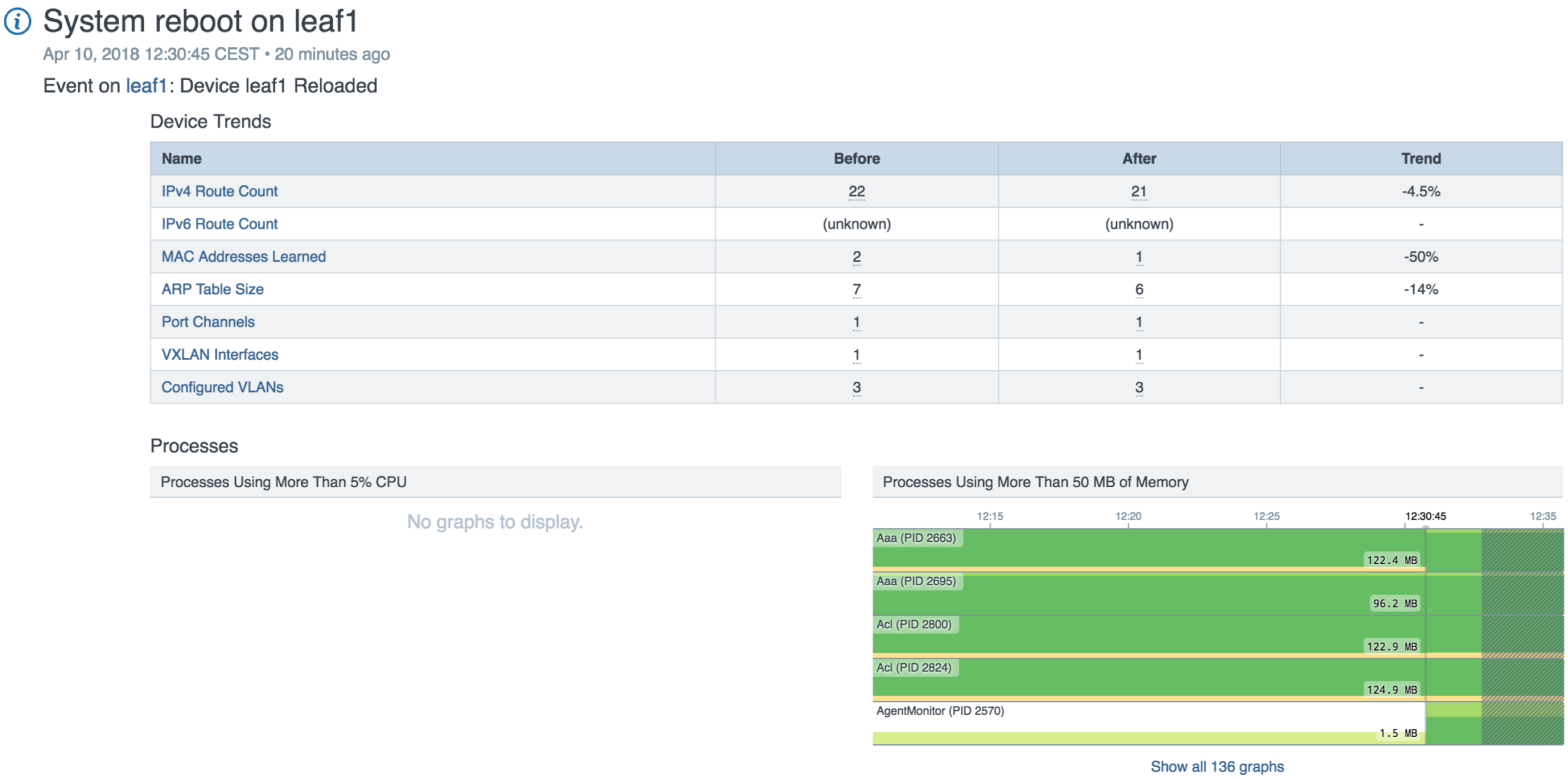
Showing 10 of 10 rows

IPv6 More…

Change	Time
::/96 modified	Feb 20, 2018 21:00:30
::1/128 modified	Feb 20, 2018 21:00:30
fe80::/10 modified	Feb 20, 2018 21:00:30
::1/128 modified	Apr 4, 2018 10:46:40
fe80::/10 modified	Apr 4, 2018 10:46:40
::/96 modified	Apr 4, 2018 10:46:40
::1/128 modified	Apr 4, 2018 10:46:40
fe80::/10 modified	Apr 4, 2018 10:46:40

Showing 8 of 8 rows

Event Correlation



ⓘ System reboot on leaf1

Apr 10, 2018 12:30:45 CEST • 20 minutes ago

Event on leaf1: Device leaf1 Reloaded

Device Trends

Name	Before	After	Trend
IPv4 Route Count	22	21	-4.5%
IPv6 Route Count	(unknown)	(unknown)	-
MAC Addresses Learned	2	1	-50%
ARP Table Size	7	6	-14%
Port Channels	1	1	-
VXLAN Interfaces	1	1	-
Configured VLANs	3	3	-

Processes

Processes Using More Than 5% CPU

No graphs to display.

Processes Using More Than 50 MB of Memory

Show all 136 graphs

How to build a (simple and open) Telemetry platform?

Simple and Open components

Component	Distributed Key-value Database	Queuing System	Analytics Pipeline	Visualization
Options	HBase, Cassandra, Kudu, Druid, Prometheus	Kafka, ActiveMQ, ZeroMQ, RabbitMQ	Spark, Storm, Heron, etc	Kibana, Grafana
Basic Telemetry	**Prometheus**	- skipped -	- skipped -	**Grafana**

Providing the metrics

- Prerequisites
  - You NEED a device/firmware which supports streaming in whatever way
  - Disk space and processing power on the collector
  - An idea what metrics you want to collect (KPIs)

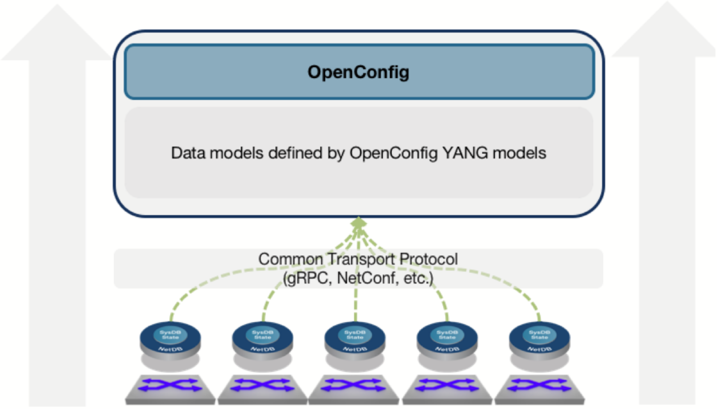
- Things to look out for
  - Inform your self about the implementation on your device/vendor of choice!
    - Some vendors ‘transform’ internal data from another format into streaming telemetry (CLI -> Streaming or SNMP -> Streaming), others support it ‘out of the box’ from the switch state database.
    - Data might be then just as ‘outdated’ as SNMP in those cases
  - Licensing fees
  - Load on the device (Telemetry can be CPU-hungry)

Providing the metrics

- Readable format to state repository
  - Convert the metrics to a format your solution can understand

- Push or Pull
  - Whilst ‘push’ would be the desired method, some monitoring solutions prefer ‘pull’ (like Prometheus)

- ’Source of Truth’ should be always the same
  - One Agent should provide the switch metrics to
    - A system who understands the metrics as they are
    - A converter (exporter) to a different format


OpenConfig
Data models defined by OpenConfig YANG models
Common Transport Protocol
(gRPC, NetConf, etc.)
SysDB State
NetDB

Converting the metrics to a Prometheus-readable format

- Only provide necessary metrics
  - Ability to define granular metrics you really need to not bloat your state repository

- Metrics will be provided via *http://<switch>:8080/metrics*.

```
subscriptions:
- /Sysdb/environment/archer/cooling/status
- /Sysdb/environment/archer/power/status
- /Sysdb/environment/archer/temperature/status
- /Smash/counters/ethIntf
- /Smash/interface/counter/lag/current/counter
- /Sysdb/hardware/archer/xcvr/status

metrics:
- name: intfCounter
  path: /Smash/counters/ethIntf/FocalPointV2/current/(counter)/(?P<intf>.+)/statistics/(?P<direction>(?:in|out))(Octets|Errors|Discards)
  help: Per-Interface Bytes/Errors/Discards Counters
- name: intfLagCounter
  path: /Smash/interface/counter/lag/current/(counter)/(?P<intf>.+)/statistics/(?P<direction>(?:in|out))(Octets|Errors|Discards)
  help: Per-PortChannel Bytes/Errors/Discards Counters
(...)
```

Deploying Prometheus / Grafana

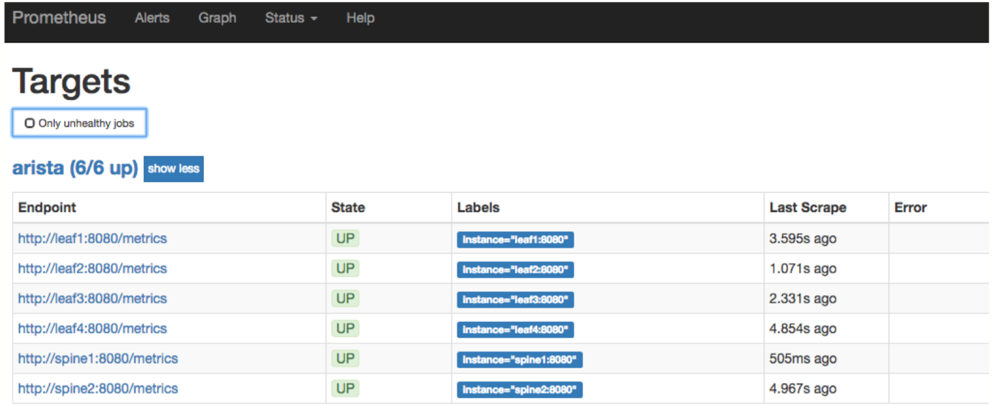
- This example uses a ‘ready-to-go’ Prometheus/Grafana docker stack
- Only need to edit ‘**prometheus/prometheus.yml**’

```
$ git clone https://github.com/vegasbrianc/prometheus.git
(...)
$ cd prometheus
$ vi prometheus/prometheus.yml
(...)
$ docker swarm init
(...)
$ HOSTNAME=$(hostname) docker stack deploy -c docker-compose.yml prom
(...)
$ docker stack ps prom | grep Run
ybxe20abekqd  prom_cadvisor.bpo4ex9k1pgdlknkkxvwh6qv0       google/cadvisor:latest     labvm  Running        Running 2 hours ago
q6x35kj8wuy9  prom_node-exporter.bpo4ex9k1pgdlknkkxvwh6qv0  prom/node-exporter:latest  labvm  Running        Running 2 hours ago
hoag8nj3gncv  prom_prometheus.1                             prom/prometheus:v2.1.0     labvm  Running        Running 2 hours ago
lcxocx172v2i  prom_alertmanager.1                           prom/alertmanager:latest   labvm  Running        Running 2 hours ago
sikfj95q1hmc  prom_grafana.1                                grafana/grafana:latest     labvm  Running        Running 2 hours ago
$ docker ps
CONTAINER ID        IMAGE                                                                                       COMMAND
CREATED             STATUS              PORTS               NAMES
888d3bd183f2        prom/prometheus@sha256:7b987901dbc44d17a88e7bda42dbbbb743c161e3152662959acd9f35aeefb9a3      "/bin/prometheus -..."   2
hours ago         Up 2 hours          9090/tcp            prom_prometheus.1.hoag8nj3gncv3lohrfqmdtrhb
(...)
```

Retrieving the metrics

- Define the targets (switches) in ‘**prometheus.yml**’
- Define scraping intervals
- Prometheus will connect to the switch and retrieve all defined metrics

```
scrape_configs:
 - job_name: 'arista’
   scrape_interval: 5s
   static_configs:
     - targets: [‘leaf1:8080', ‘leaf2:8080']
```

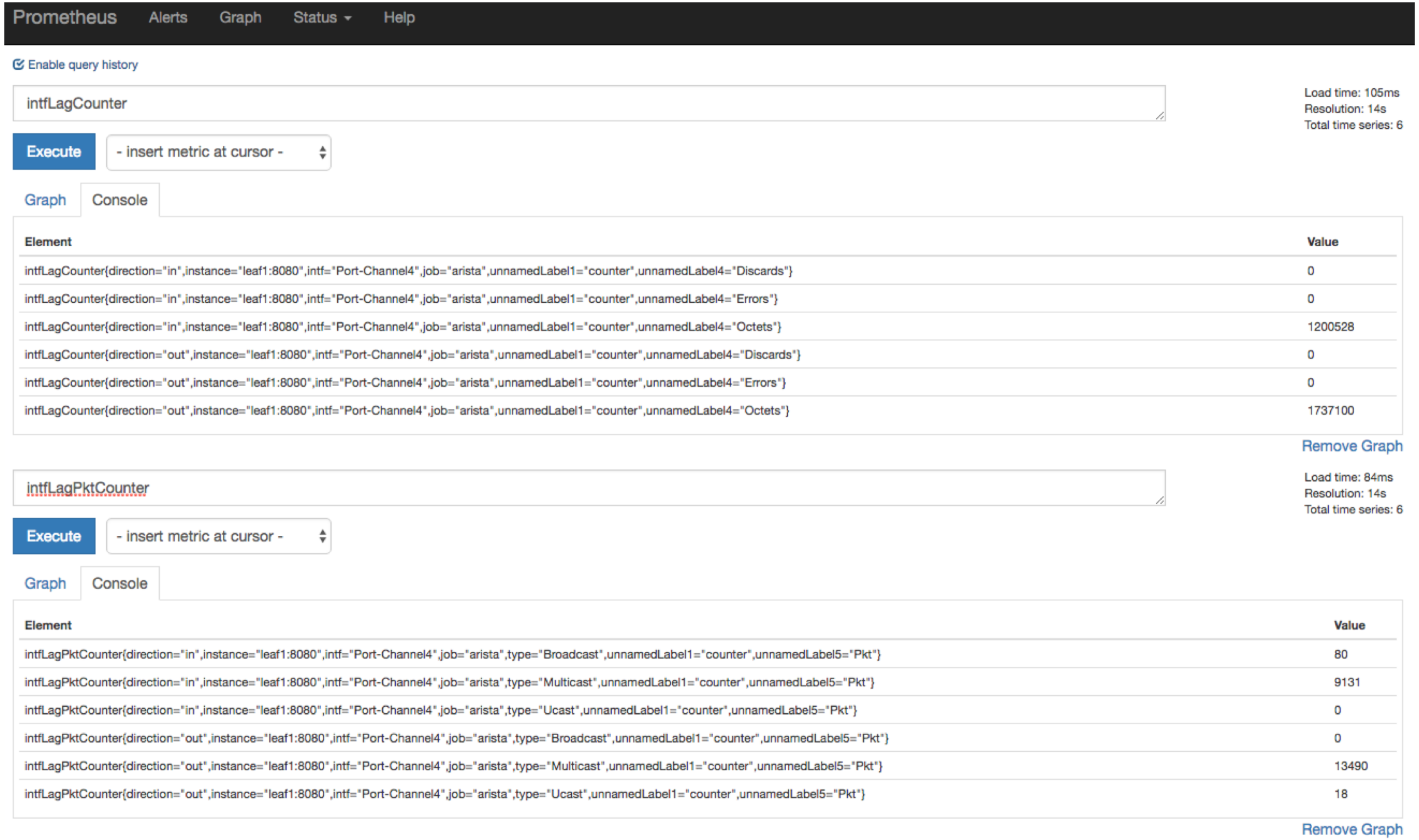


Targets

arista (6/6 up)

Endpoint	State	Labels	Last Scrape	Error
http://leaf1:8080/metrics	UP	instance="leaf1:8080"	3.595s ago	
http://leaf2:8080/metrics	UP	instance="leaf2:8080"	1.071s ago	
http://leaf3:8080/metrics	UP	instance="leaf3:8080"	2.331s ago	
http://leaf4:8080/metrics	UP	instance="leaf4:8080"	4.854s ago	
http://spine1:8080/metrics	UP	instance="spine1:8080"	505ms ago	
http://spine2:8080/metrics	UP	instance="spine2:8080"	4.967s ago	

Retrieving the metrics



intfLagCounter

Load time: 105ms
Resolution: 14s
Total time series: 6

Element	Value
intfLagCounter{direction="in",instance="leaf1:8080",intf="Port-Channel4",job="arista",unnamedLabel1="counter",unnamedLabel4="Discards"}	0
intfLagCounter{direction="in",instance="leaf1:8080",intf="Port-Channel4",job="arista",unnamedLabel1="counter",unnamedLabel4="Errors"}	0
intfLagCounter{direction="in",instance="leaf1:8080",intf="Port-Channel4",job="arista",unnamedLabel1="counter",unnamedLabel4="Octets"}	1200528
intfLagCounter{direction="out",instance="leaf1:8080",intf="Port-Channel4",job="arista",unnamedLabel1="counter",unnamedLabel4="Discards"}	0
intfLagCounter{direction="out",instance="leaf1:8080",intf="Port-Channel4",job="arista",unnamedLabel1="counter",unnamedLabel4="Errors"}	0
intfLagCounter{direction="out",instance="leaf1:8080",intf="Port-Channel4",job="arista",unnamedLabel1="counter",unnamedLabel4="Octets"}	1737100

intfLagPktCounter

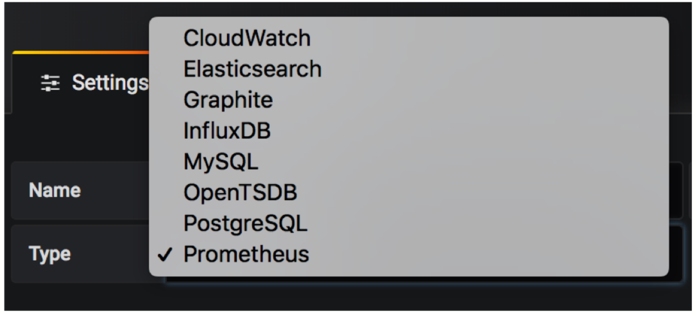
Load time: 84ms
Resolution: 14s
Total time series: 6

Element	Value
intfLagPktCounter{direction="in",instance="leaf1:8080",intf="Port-Channel4",job="arista",type="Broadcast",unnamedLabel1="counter",unnamedLabel5="Pkt"}	80
intfLagPktCounter{direction="in",instance="leaf1:8080",intf="Port-Channel4",job="arista",type="Multicast",unnamedLabel1="counter",unnamedLabel5="Pkt"}	9131
intfLagPktCounter{direction="in",instance="leaf1:8080",intf="Port-Channel4",job="arista",type="Ucast",unnamedLabel1="counter",unnamedLabel5="Pkt"}	0
intfLagPktCounter{direction="out",instance="leaf1:8080",intf="Port-Channel4",job="arista",type="Broadcast",unnamedLabel1="counter",unnamedLabel5="Pkt"}	0
intfLagPktCounter{direction="out",instance="leaf1:8080",intf="Port-Channel4",job="arista",type="Multicast",unnamedLabel1="counter",unnamedLabel5="Pkt"}	13490
intfLagPktCounter{direction="out",instance="leaf1:8080",intf="Port-Channel4",job="arista",type="Ucast",unnamedLabel1="counter",unnamedLabel5="Pkt"}	18

Visualizing the metrics

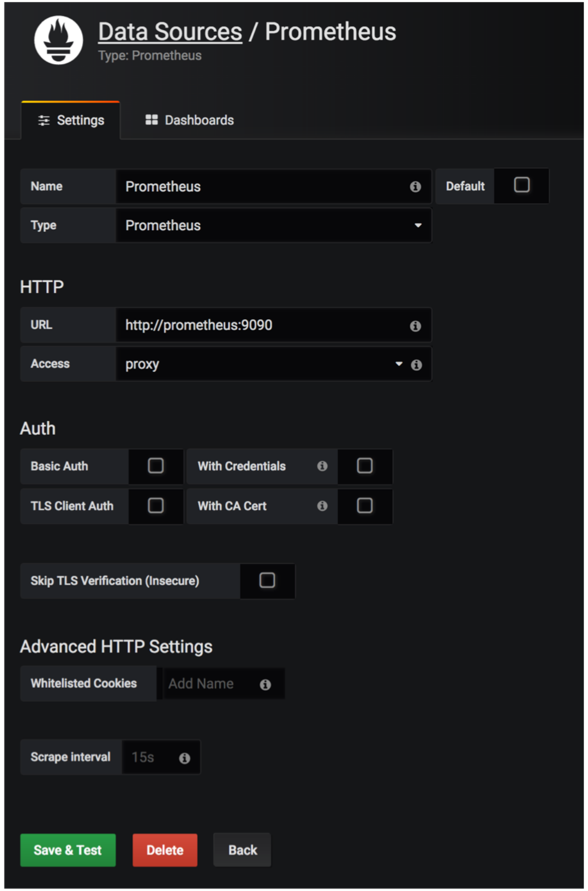
- Grafana supports Prometheus natively as a data source

- Besides Prometheus a lot of other Data Sources are supported by Grafana as well.



Name

Type: ✓ Prometheus



Data Sources / Prometheus

Type: Prometheus

Name: Prometheus

Default □

Type: Prometheus

HTTP

URL: http://prometheus:9090

Access: proxy

Auth

Basic Auth □

With Credentials □

TLS Client Auth □

With CA Cert □

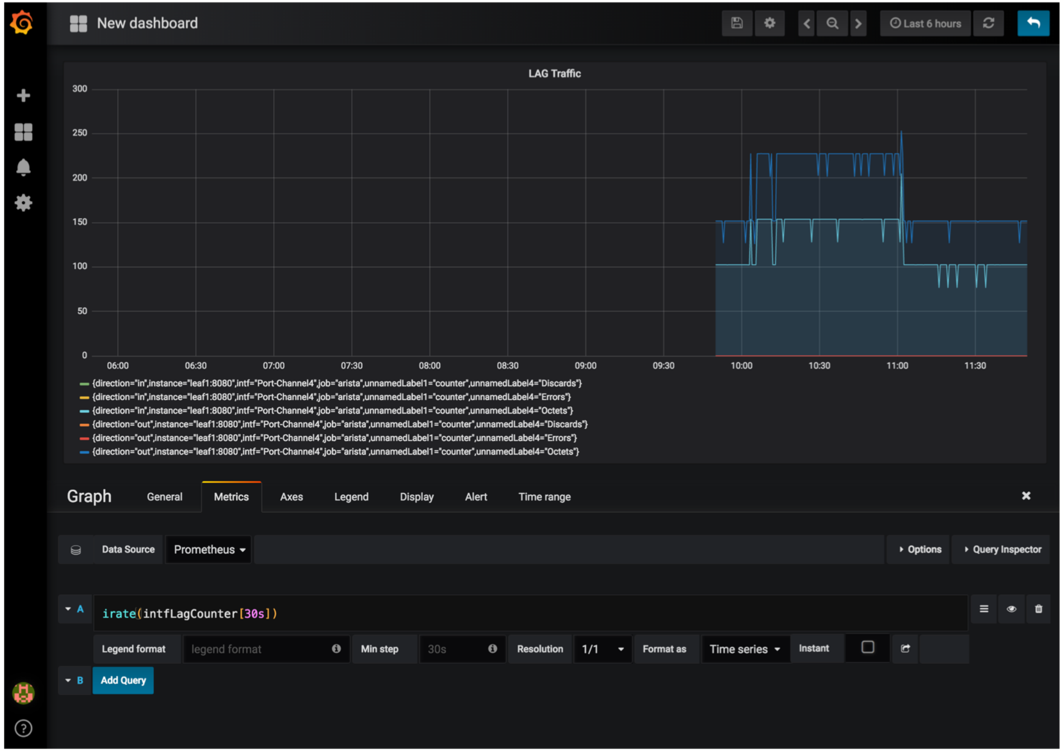
Skip TLS Verification (Insecure) □

Advanced HTTP Settings

Whitelisted Cookies: Add Name

Scrape interval: 15s

Visualizing the metrics



- {direction="in",instance="leaf1:8080",intf="Port-Channel4",job="arista",unnamedLabel1="counter",unnamedLabel4="Discards"}
- {direction="in",instance="leaf1:8080",intf="Port-Channel4",job="arista",unnamedLabel1="counter",unnamedLabel4="Errors"}
- {direction="in",instance="leaf1:8080",intf="Port-Channel4",job="arista",unnamedLabel1="counter",unnamedLabel4="Octets"}
- {direction="out",instance="leaf1:8080",intf="Port-Channel4",job="arista",unnamedLabel1="counter",unnamedLabel4="Discards"}
- {direction="out",instance="leaf1:8080",intf="Port-Channel4",job="arista",unnamedLabel1="counter",unnamedLabel4="Errors"}
- {direction="out",instance="leaf1:8080",intf="Port-Channel4",job="arista",unnamedLabel1="counter",unnamedLabel4="Octets"}

Data Source: Prometheus

A

```
irate(intfLagCounter[30s])
```

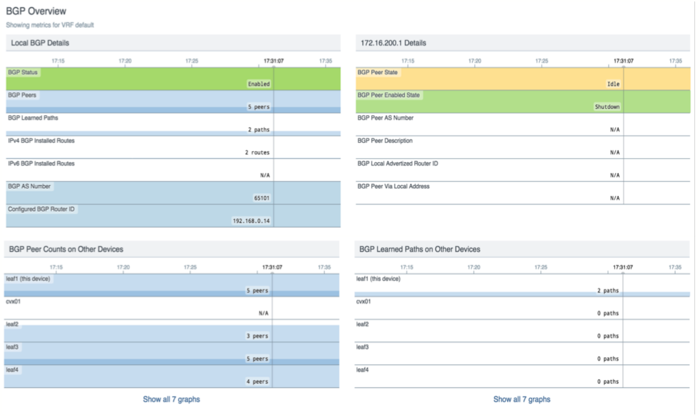
Legend format: legend format | Min step: 30s | Resolution: 1/1 | Format as: Time series | Instant: □

B

- Configure your dashboard(s) with the available metrics
- Auto-completion for metrics and functions is available
- If you have multiple vendors, make sure that the counters are named the same

Vendor solutions vs. Open Source

- Essentially it depends on the man power available
- Vendor solutions provide detailed and profound understanding of events for their own devices and can correlate them ‘out of the box’
- Open Source solutions can support multiple vendors in the same UI, but ‘intelligence’ on metrics and correlation has to be built by the customer itself.



BGP Overview

Showing metrics for VRF default

Local BGP Details

BGP Status: Enabled
BGP Peers: 5 peers
BGP Learned Paths: 2 paths
IPv4 BGP Installed Routes: 2 routes
IPv6 BGP Installed Routes: N/A
BGP AS Number: 65101
Configured BGP Router ID: 192.168.0.14

172.16.200.1 Details

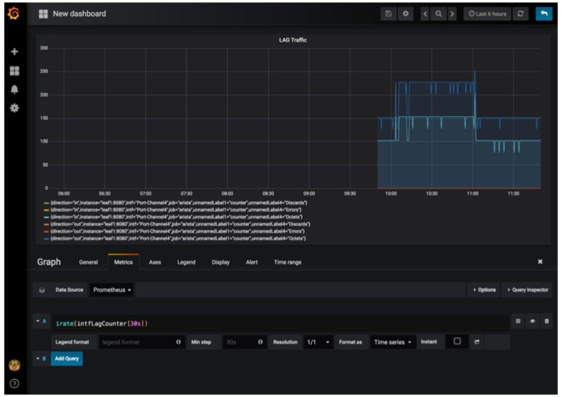
BGP Peer State: Idle
BGP Peer Enabled State: Shutdown
BGP Peer AS Number: N/A
BGP Peer Description: N/A
BGP Local Advertized Router ID: N/A
BGP Peer Via Local Address: N/A

BGP Peer Counts on Other Devices

leaf1 (this device): 5 peers
cvx01: N/A
leaf2: 3 peers
leaf3: 5 peers
leaf4: 4 peers

BGP Learned Paths on Other Devices

leaf1 (this device): 2 paths
cvx01: 0 paths
leaf2: 0 paths
leaf3: 0 paths
leaf4: 0 paths



```
{direction="in",instance="leaf1:8080",intf="Port-Channel4",job="arista",unnamedLabel1="counter",unnamedLabel4="Discards"}
{direction="in",instance="leaf1:8080",intf="Port-Channel4",job="arista",unnamedLabel1="counter",unnamedLabel4="Errors"}
{direction="in",instance="leaf1:8080",intf="Port-Channel4",job="arista",unnamedLabel1="counter",unnamedLabel4="Octets"}
{direction="out",instance="leaf1:8080",intf="Port-Channel4",job="arista",unnamedLabel1="counter",unnamedLabel4="Discards"}
{direction="out",instance="leaf1:8080",intf="Port-Channel4",job="arista",unnamedLabel1="counter",unnamedLabel4="Errors"}
{direction="out",instance="leaf1:8080",intf="Port-Channel4",job="arista",unnamedLabel1="counter",unnamedLabel4="Octets"}
```

Data Source: Prometheus

A: `irate(intfLagCounter[30s])`

Legend format: legend format · Min step: 30s · Resolution: 1/1 · Format as: Time series · Instant ☐

References

- OpenConfig ‘Streaming telemetry’ definition
  - http://www.openconfig.net/projects/telemetry/
- Database ‘connectors’
  - https://github.com/aristanetworks/goarista/tree/master/cmd
- Prometheus/Grafana Docker Stack
  - https://github.com/vegasbrianc/prometheus

Thank You

www.arista.com