Streaming Telemetryで実現！！
サイレント障害における、予兆検知への道

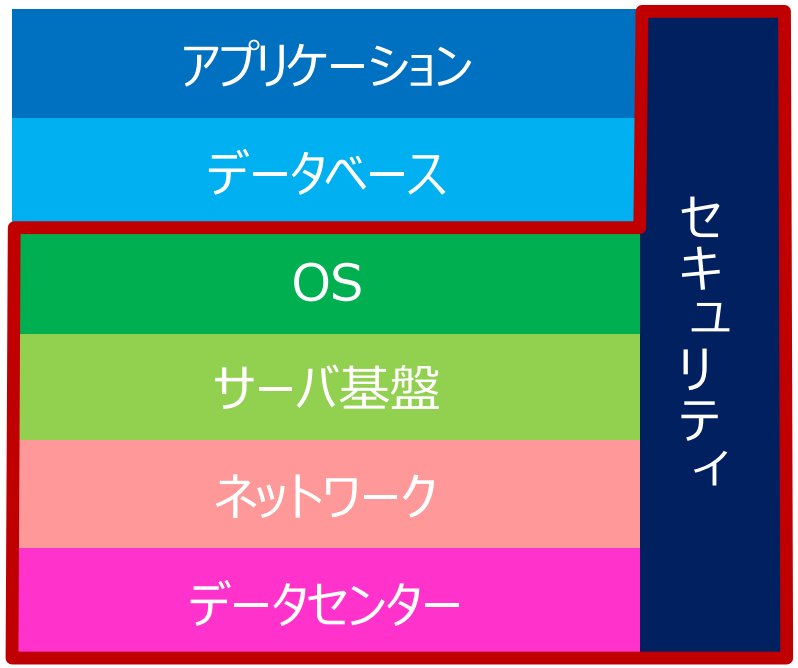
1/26(金)

ネットワンシステムズ株式会社　井上 勝晴、片野 祐
フューチャーアーキテクト株式会社　日比野 恒

自己紹介

名前：日比野 恒 (ひびの ひさし)
所属：フューチャーアーキテクト株式会社
Technology Innovation Group
セキュリティアーキテクト (CISA、登録セキスペ)

領域：

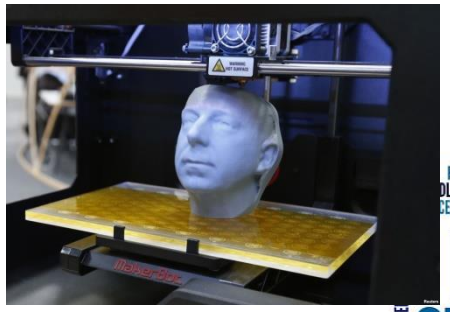


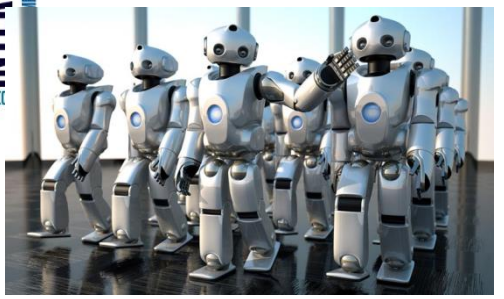
https://www.slideshare.net/hibinohisashi/presentations

自己紹介

名前：井上 勝晴（いのうえ まさはる）
所属：ビジネス推進本部　応用技術部
　　コアネットワークチーム
領域：ネットワーク、モバイル、クラウド

名前：片野 祐（かたの ゆう）
所属：ビジネス推進本部　応用技術部
　　クラウドデータインフラチーム
領域：データプラットフォーム
　　AI、機械学習、データ分析





ディスカッションしたいこと

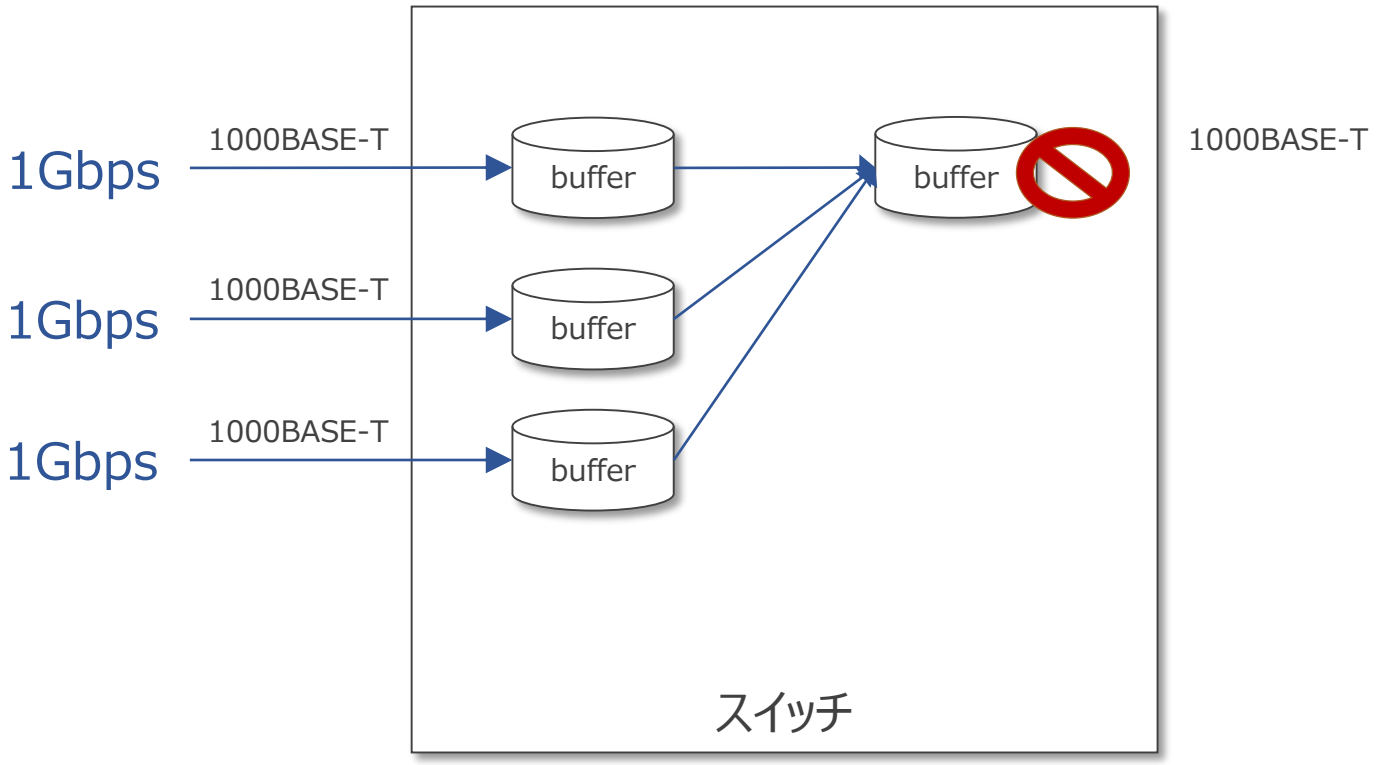
- ✓ サイレント障害による痛い経験
- ✓ 予兆検知による予防保守の抱える課題

目次

1. きっかけ

マイクロバーストって

1秒未満(マイクロ秒)のトラフィック量が**ポート帯域超過**してしまう事象のこと。



トラフィックがバッファから溢れ
ポートで**Discard**が起きる
(SNMP Trap検知)

SNMPの性能データでは
200-300Mbpsの通信しか
観測できず**原因特定が出来ない**
(サイレント障害の1つ)

ネットワーク性能管理の限界

サイレント障害は、**性能データ収集方法**の改善で減らせる。

- ✓ 秒単位で収集することで**平均化されない正確**な値が得られる。
- ✓ 秒単位で収集することで**リアルタイム性**が得られる。
- ✓ 多くの値を収集することでこれまで**気づけなかった事象**が相関分析できる。

※マイクロバースト検知は、秒単位のインターフェース情報やバッファ使用状況のデータで実現
※予兆検知は、機器情報(CPU、メモリ、トラフィック以外)のベースラインを確立することで実現

ハードディスクの故障原因って

温度ではなく、**相対湿度**が重要な故障原因であること、ご存知でしたか？？

DC Tag	Cooling	AFR	Increase wrt $AFR = 1.5\%$
CD1	*Chiller*	1.5%	0%
CD2	*Water-Side*	2.1%	40%
CD3	*Free-Cooled*	1.8%	20%
HD1	*Chiller*	2.0%	33%
HD2	*Water-Side*	2.3%	53%
HH1	*Free-Cooled*	3.1%	107%
HH2	*Free-Cooled*	5.1%	240%
HH3	*Free-Cooled*	5.1%	240%
HH4	*Free-Cooled*	5.4%	260%

Table 3: Disk AFRs. HH1-HH4 incur the highest rates.

<表3>ディスクの年間故障率（AFR）。HH1～HH4のAFRが特に高い。「DC Tag」欄のCはCold、DはDry、最初のHはHot、2つ目のHはHumidを表す（つまりHHならば「高温・多湿」）。

- ✓ 世界9か所 MSのデータセンター
- ✓ 100万台以上 ディスクドライブ
- ✓ 室温、相対湿度を含む環境データ

あらゆるデータを分析することで
相関関係から**新たな結論**を導き出す！

2. Telemetry 登場の背景と概要

Telemetryとは？

- Telemetry ＝ Tele(遠隔) ＋ Metry(測る)
  - Tele ＝ Remote (遠隔)とMetry ＝ Measure(測る)から成る造語

- 大雑把に言うと、次世代版 SNMP と捉える事も出来ます

何故、今、Telemetry??

Googleに代表されるHyper Scale Player達が、**SNMP**による自社Data Center運用に**限界**を感じている

1）Hyper Scale Player Topic
- 超大規模Data Center
- White Box Switch積極利用

→ 従来のOperationでは回らない

2）Operation変革
- NetworkをIntent Driven型へ
- Workflowの洗練化、Big data活用

→ その為にNetwork Resourceの
リアルタイム把握が必要

3）Telemetry登場
SNMPではマッチせず、新方式として
Telemetryが考案される

→ Open-Sourceで検討がなされ
ベンダー各社が製品リリース

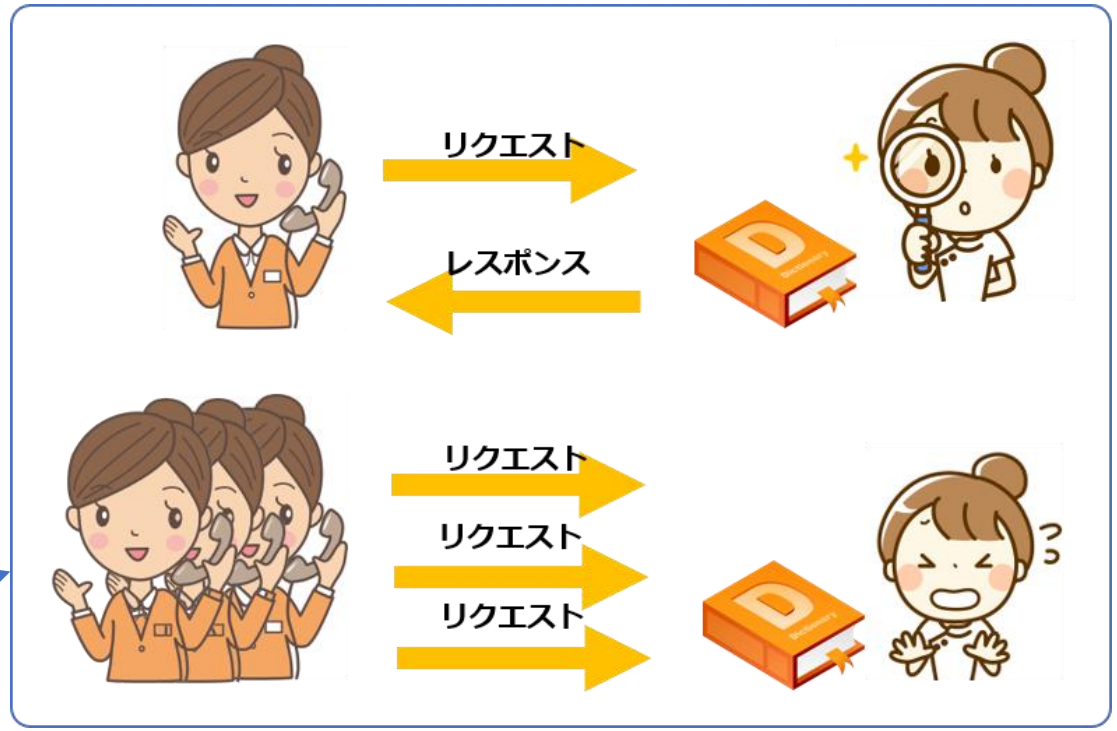
Operational Scale (e.g. Google)
- 30,000+ circuit operation
- Many tens of Network elements
- Dozen vendors
- 4M line of Configuration
- 30k configuration change per month
- 8M OID

SNMPの課題

SNMPはプロトコルの構造上、**リアルタイム性が低い**、レガシーな（古い）管理手法と言える　※SNMPv1 1988年 / SNMPv3 1999年

- 性能課題
  Polling間隔を狭める or 複数Pollerが存在すると、機器側で処理がスタック
- Low Protocol Flexibility
  YANG等の新Data Model適用不可、TransportもUDPのみ(gRPC非対応)

SNMPはリクエストベース
デバイスはリクエスト毎に、同じ辞書式順序で情報を取り出して返答している。
イメージ的には、電話通信（コール毎に都度応答、未完了状態では新規コールは受け付不可で保留）に相当し、scaleせずリアルタイム性も低い



SNMP の処理プロセスの概要

Telemetry概要

TelemetryはSubscription（購読宣言）ベースであり、
リアルタイム性が高い、新たな管理手法と言える

- 購読/配信型
  - ある情報に対するSubscriber（購読者）が複数存在したとしても、デバイスはリクエスト毎にコール（処理）することは無い
  - 自身が保持する情報を、購読者達に定期的に配信するのみ

TelemetryはSubscriptionベース
購読者は欲しい情報と購読頻度をデバイスに宣言する
デバイスは求められた情報を配信するのみ
故に軽いプロトコル構造であり、データのリアルタイム取得に適している

Telemetry の処理プロセスの概要

3. Telemetry 解説

Telemetry Framework Requirement by Google

- network elements stream data to collectors (push model)

  NW機器は**ストリーム形式**にてCollectorへデータを送信（Pushモデル）

- data populated based on vendor-neutral models whenever possible

  可能な限り**ベンダー非依存**なモデルにてデータを生成する事

以降のスライドで、これらRequirementを満たす上で重要となる技術要素を、幾つか見て行きたいと思います。

  - other protocols distribute load to hardware, so should telemetry

  向こう10年のデータ増加に対応出来る事 – HWに負荷分散出来る事(Telemetry)

- utilize modern transport mechanisms with active development communities – gRPC, protocol buffer

  最新Transportメカニズムを利用する事 – gRPC, protocol buffer over udp

技術要素 – gRPC as transport protocol

gRPC(Google Remote Procedure Call)はマルチプラットフォームに対応した RPC Framework

特徴 – 転送機能

- 負荷分散、APPレベルフロー制御、Callキャンセル機能
- 双方向Stream、Push型
- Stream Connectionの多重化

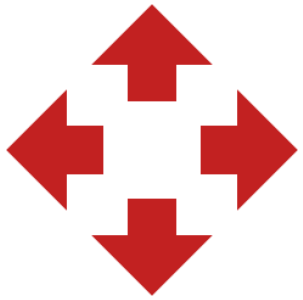
特徴 – 開発実行環境

- オープンソース
- マルチプラットフォーム（複数言語に対応

gRPCを使用する事で、**HTTP/2**を、**独立して動作する**
Transport Layerの様に利用する事が出来る

技術要素 – GPB as codec

GPB(Google Protocol Buffers)はGoogle が開発した構造データをシリアライズするためのメカニズム(SNMPにおけるASN.1 BER相当)



特徴 – 拡張性

- 言語非依存・プラットフォーム非依存で拡張性が高い

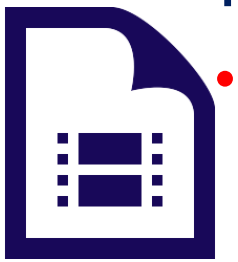
特徴 – 処理性能

- xmlと比べてデータサイズが小さく、高速、シンプル
- 単純処理を得意するASICでも動作可能、SNMPのASN.1 BERと比べると処理負荷は軽減される

CodecにGPBを利用する事で、Telemetry Dataを
小サイズで**高速・シンプル**な形で送信する事が出来る

技術要素 – OpenConfig as data modeling

OpenConfigはプログラマブルなネットワークへの移行を目指すオペレータ達で構成される非公式WG



特徴 – 共通データモデル

- **ベンダー非依存**な **Configuration(設定)**と **Operational State(運用状態)** の共通データモデルを策定
  → Multivendor NWの統一運用

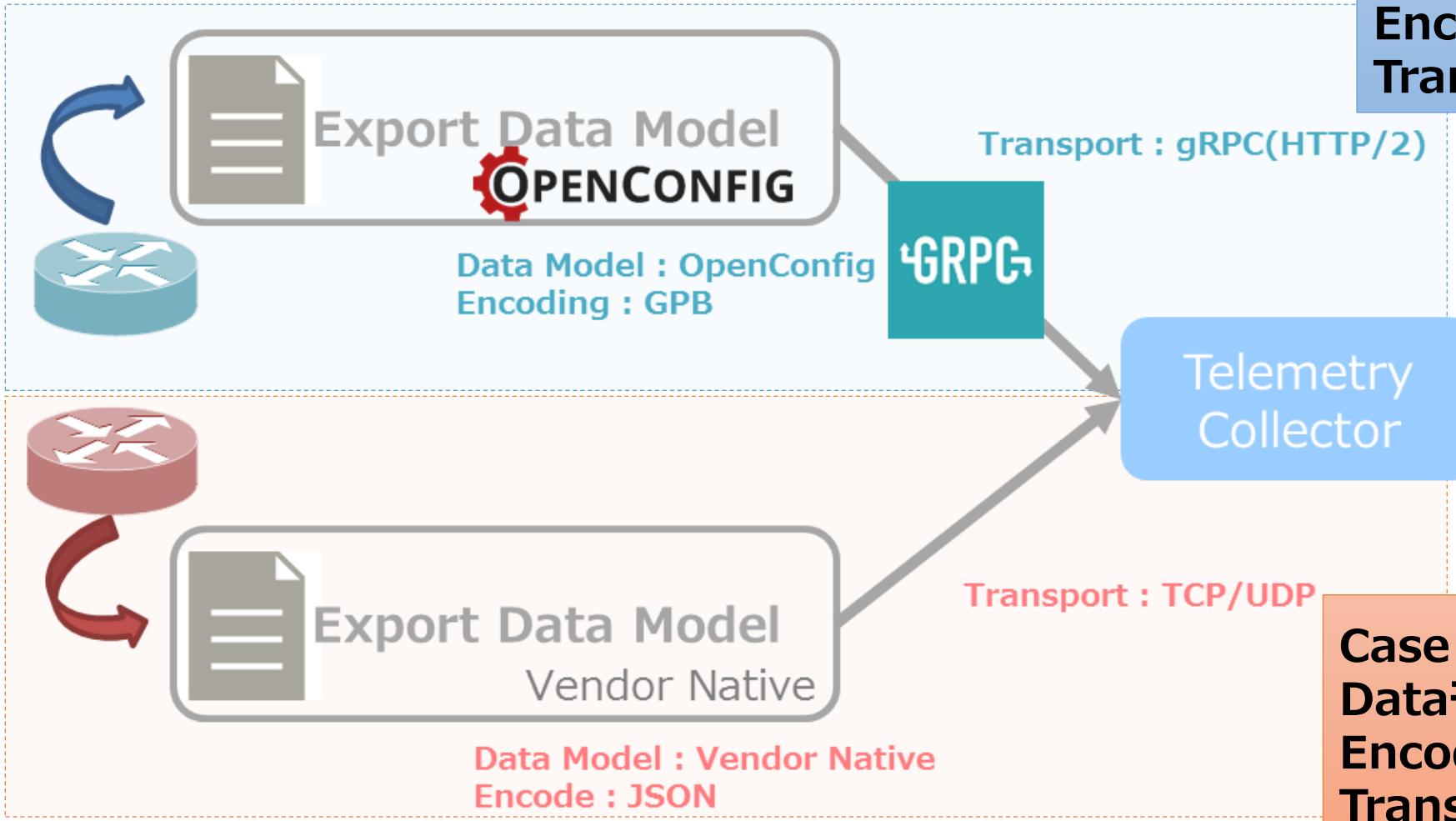
参画オペレータ

Google, AT&T, Microsoft, BT, facebook, COMCAST, verizon, YAHOO!, Apple

Interface Counter 等の機器ステータス取得時に、
OpenConfigで策定された**共通データモデル**を利用可能となる

技術要素のまとめ

- Data Model / Encoding / Transport技術



Case 1)
Dataモデル – OpenConfig
Encoding – GPB
Transport – gRPC

Case 2)
Dataモデル – Vendor Native
Encoding – Text(JSON)
Transport – TCP, UDP

取得データ例①

- OpenConfig Interface Counter（1/2）

ツリーパス	ノード名	説明	Interface MIB (RFC 2863)
+--rw interfaces +--rw interface* [name] +--rw subinterfaces +--rw subinterface* [index] +--ro state	index	サブインターフェースの番号	
	name	インターフェースの名前	
	description	インターフェースのディスクリプション	ifAlias
	enabled	インターフェースの状態	ifAdminStatus
	ifindex	インターフェースにアサインされる番号	The Interfaces Group MIB
	admin-status	インターフェースのアドミンステータス	ifAdminStatus
	oper-status	インターフェースのオペレーションステータス	ifOperStatus
	last-change	インターフェースのアップダウンなど、ステータスが最後に変わった時間	ifLastChange

取得データ例②

- OpenConfig Interface Counter（2/2）

ツリーパス	ノード名	説明	Interface MIB (RFC 2863)
+--rw interfaces +--rw interface* [name] +--rw subinterfaces +--rw subinterface* [index] +--ro state +--ro counters	in-octets	受信バイト数	ifHCInOctets
	in-unicast-pkts	ユニキャストの受信パケット数	ifHCInUcastPkts
	in-broadcast-pkts	ブロードキャストの受信パケット数	ifHCInBroadcastPkts
	in-multicast-pkts	マルチキャストの受信パケット数	ifHCInMulticastPkts
	in-discards	破棄された受信パケット数	ifInDiscards
	in-errors	エラーパケットの受信パケット数	ifInErrors
	in-unknown-protos	理解できない または 未サポートプロトコルにより破棄された受信パケット数	ifInUnknownProtos
	out-octets	送信バイト数	ifHCOutOctets
	out-unicast-pkts	ユニキャストの送信パケット数	ifHCOutUcastPkts
	out-broadcast-pkts	ブロードキャストの送信パケット数	ifHCOutBroadcastPkts

OpenConfig Interface Counterを例にすると、

Interface MIB(RFC 2863)相当の情報が取得可能

Telemetry/SNMP比較①

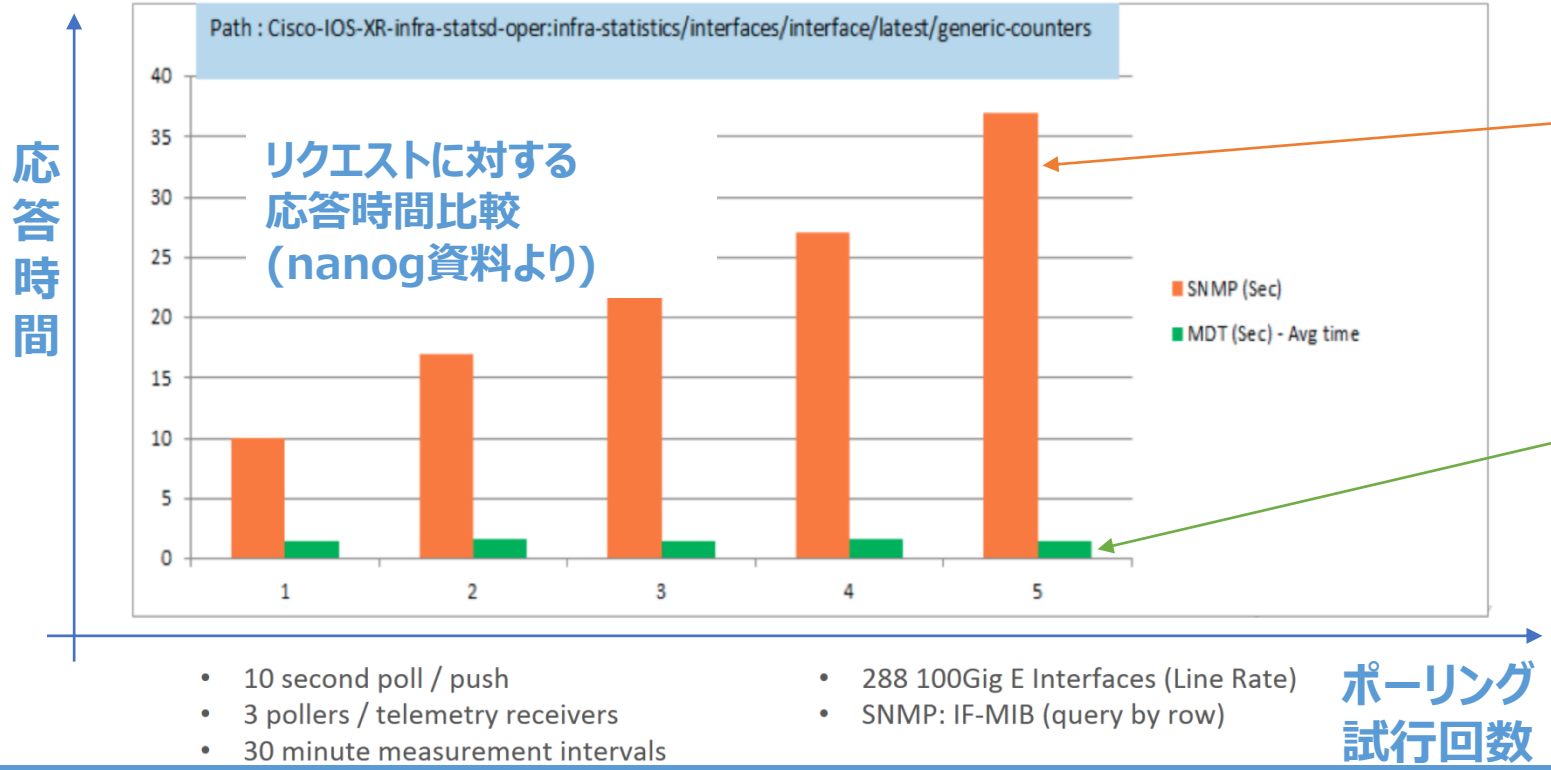
	SNMP	Telemetry	
取得方式	Polling	Subscription	Subscription
Push or Pull	Pull	Push	Push
Data Model	SMI ASN.1	OpenConfig	Vendor Native
Encoding	ASN.1 BER	GPB	GBP, JSON
Transport	UDP	gRCP(HTTP/2), TCP	TCP, UDP

Subscription/Push/GPB/gRPC → **リアルタイム性向上**

OpenConfig → 共通モデルによる **Operationの統一**

Telemetry/SNMP比較② - performance

Lesson 2: It's Not Hard to Beat SNMP



- 10 second poll / push
- 3 pollers / telemetry receivers
- 30 minute measurement intervals
- 288 100Gig E Interfaces (Line Rate)
- SNMP: IF-MIB (query by row)

SNMP：
ポーリング試行が重なる度に、機器からの応答時間が長くなる
→ **機器側で処理がスタック**

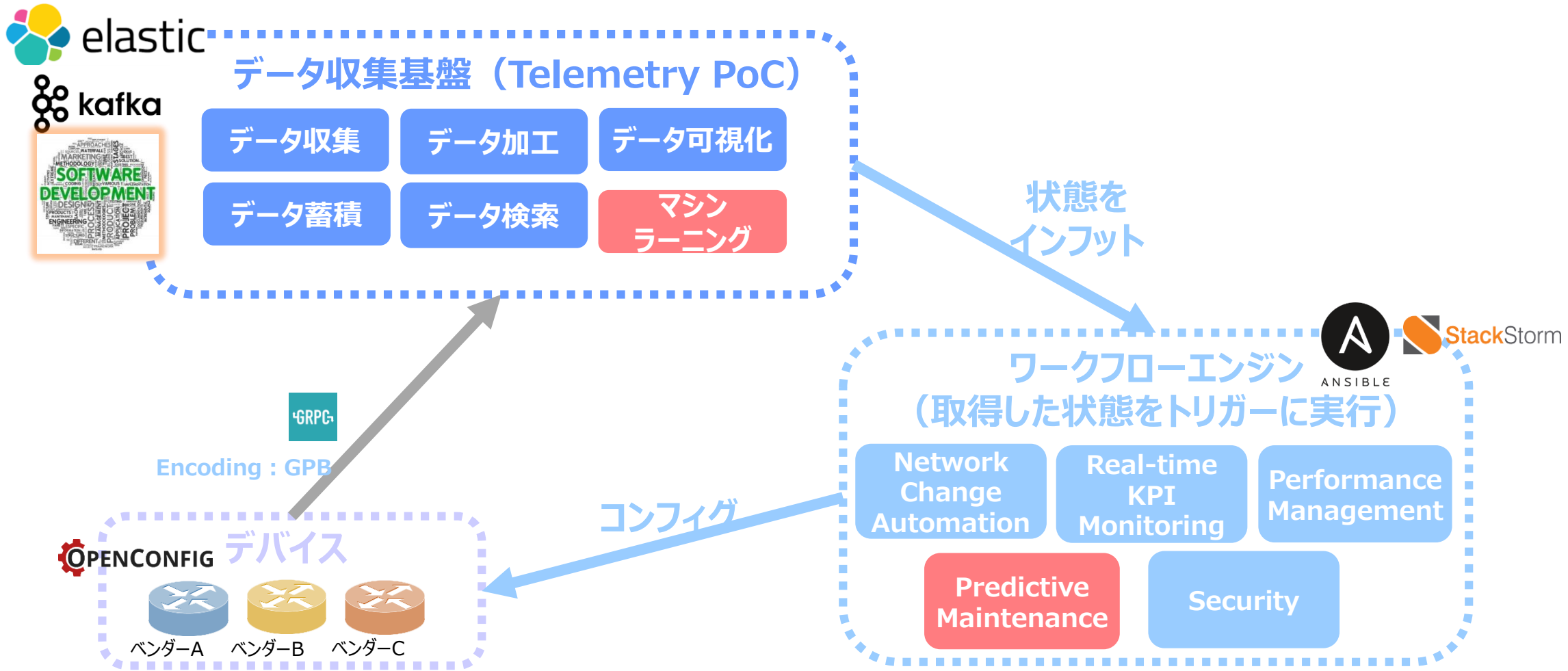
Telemetry：
ポーリング試行が増えても、一定した応答時間にて処理可能

Telemetry方式では一定した応答時間を達成しており、
機器負荷の面からもTelemetry方式が優れている事が解る

4. PoC 概要

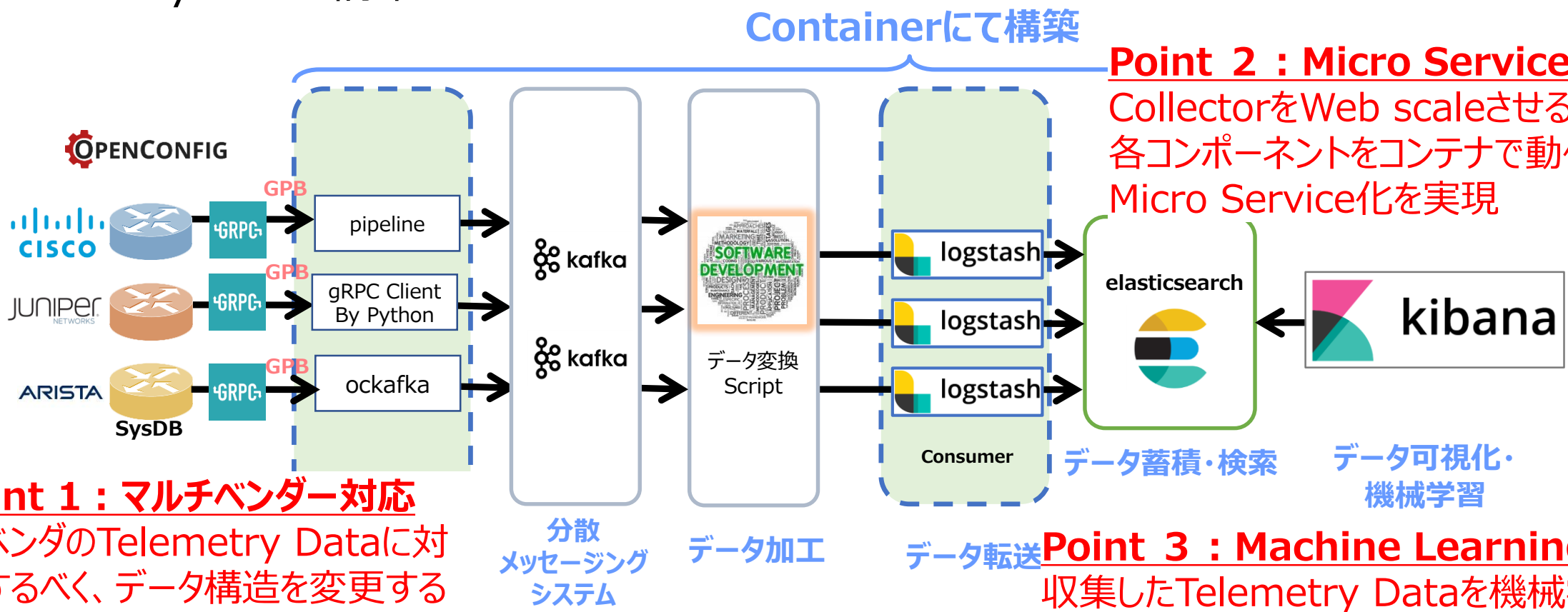
将来的な全体像

自動復旧・自動最適化を行う**自立型ネットワーク**
データ収集基盤（本Telemetry PoC）は、この自立型ネットワークへのコアコンポーネント



PoC概要

「Telemetry技術のキャッチアップ」と「リアルタイムデータの活用検討」を目的として、本Telemetry PoCを構築



Point 1：マルチベンダー対応

各ベンダのTelemetry Dataに対応するべく、データ構造を変更する仕組みをベンダー毎に構築

Point 2 ：Micro Service化

CollectorをWeb scaleさせるべく各コンポーネントをコンテナで動作させ、Micro Service化を実現

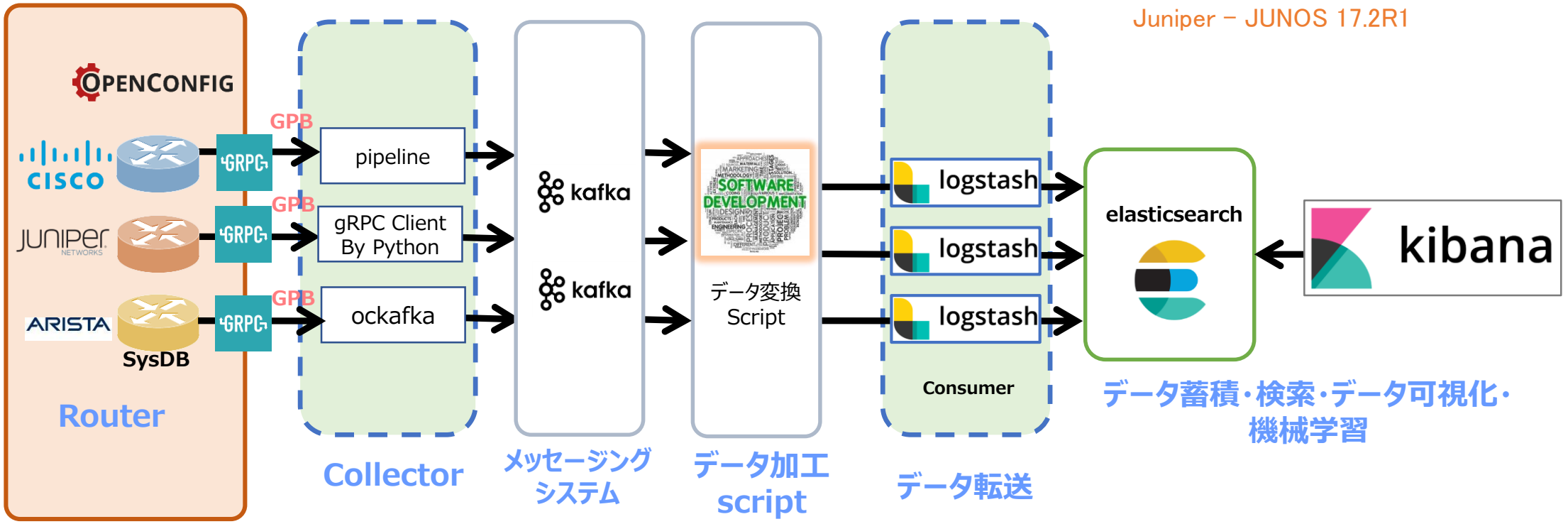
Point 3 ：Machine Learning

収集したTelemetry Dataを機械学習し（Elastic Stack X-packにて）、異常な振舞いを検知

5. PoC コンポーネント説明

各コンポーネント説明(1/5)

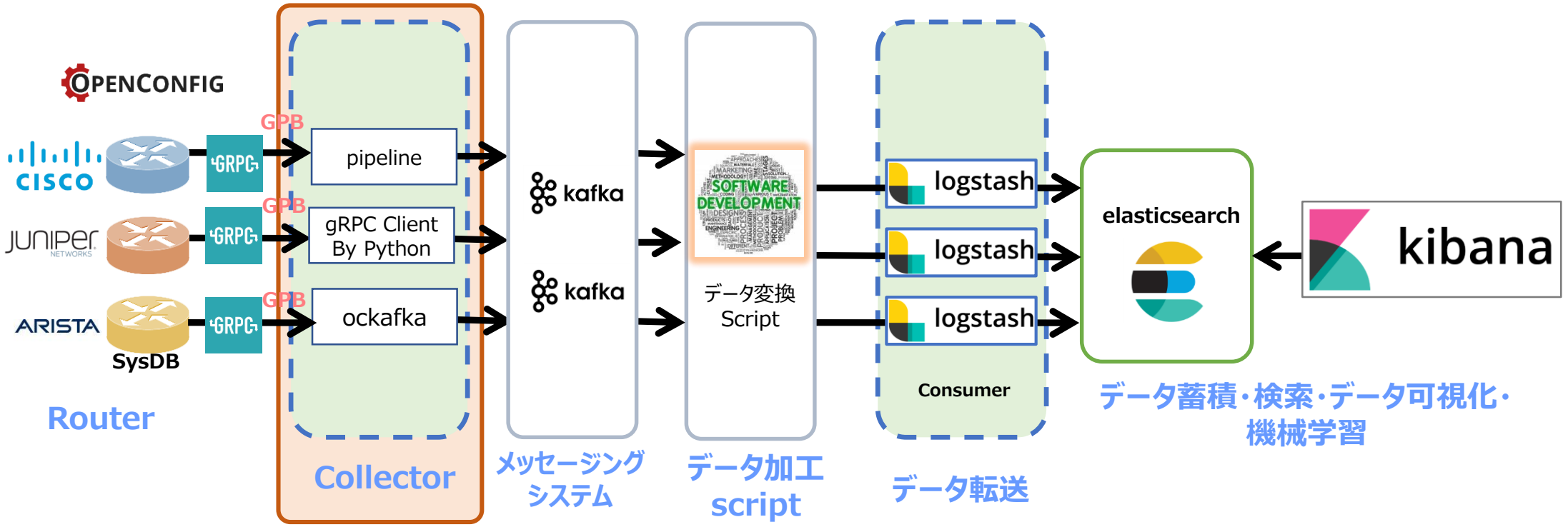
Arista – 4.17.5M-4414219.4175M
Cisco – Cisco IOS XR Software, Version 6.1.2
Juniper – JUNOS 17.2R1



Router

- Telemetry データを生成し、指定した Collector へ送信する
- データモデリング（OpenConfig や Native）、コーディング（GPB や JSON）、Transport（gRPC や TCP/UDP）を其々指定する事が出来る

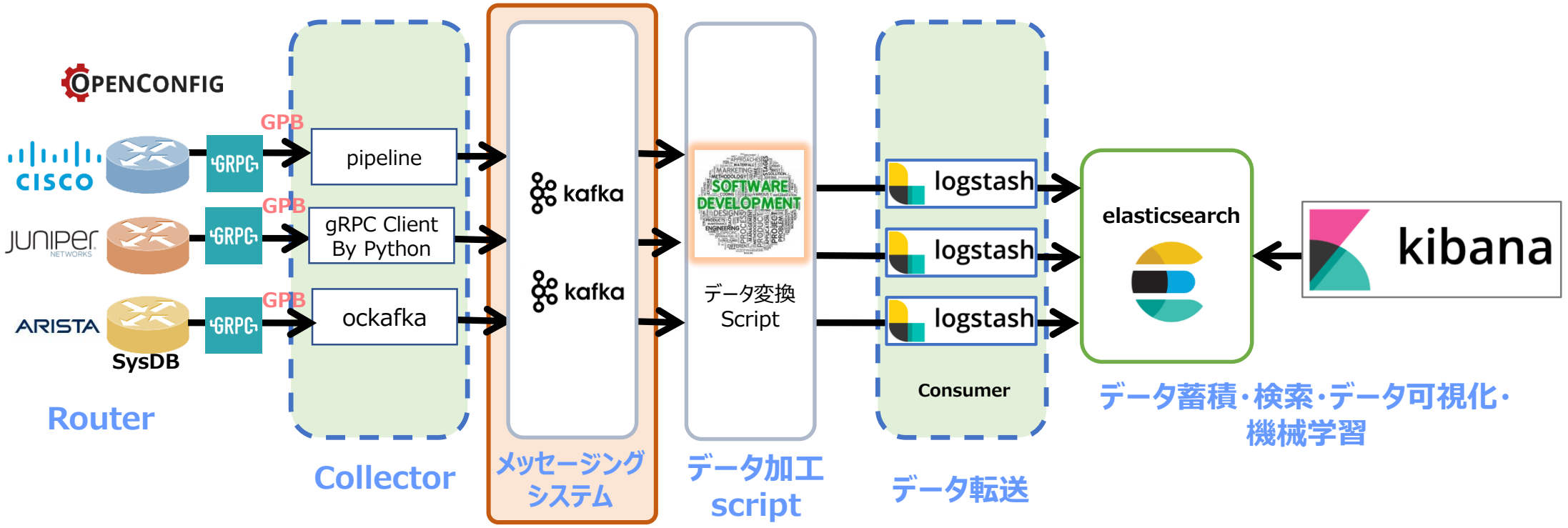
各コンポーネント説明(2/5)



Collector

- 機器からの Telemetry データを受ける Frontend コンポーネント
- gRPC の終端、GPB デシリアライズとデータの構造化（JSON 化）、外部コンポーネントへの送信等を担う
- 本PoCでは、各ネットワーク機器ベンダーが公開/提供する Collector ツールを利用

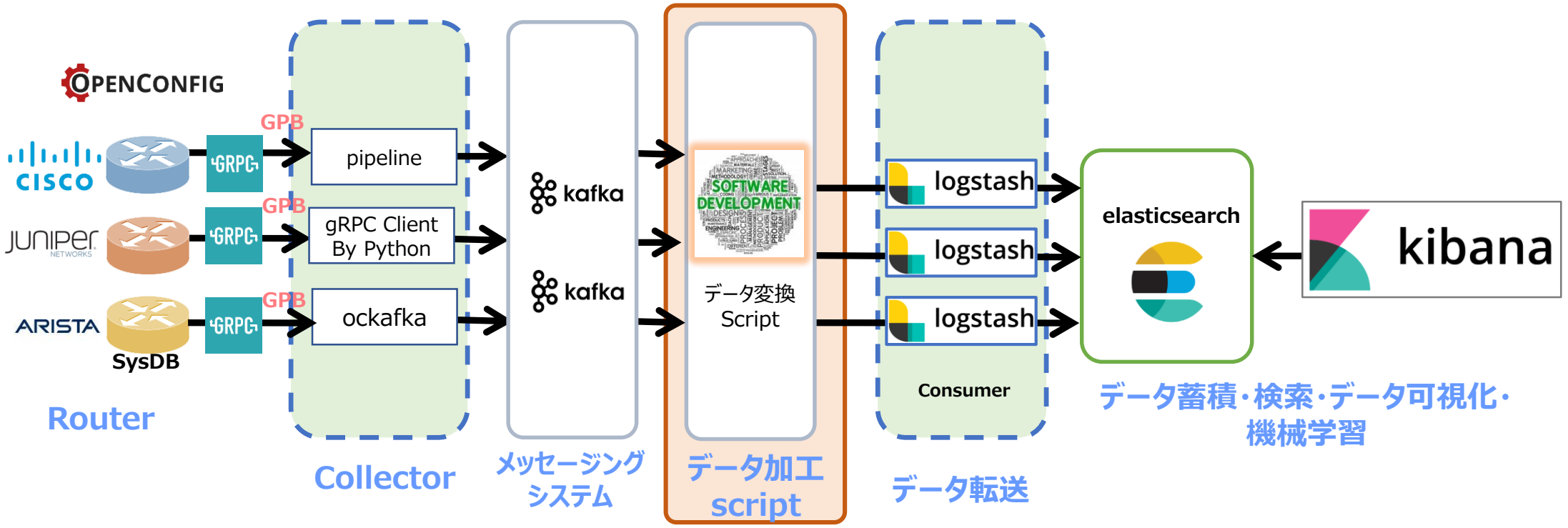
各コンポーネント説明(3/5)



メッセージングシステム

- Collector から送信されるリアルタイムデータを一時的に受け、外部コンポーネントへと展開
- 本PoC では Kafka を採用

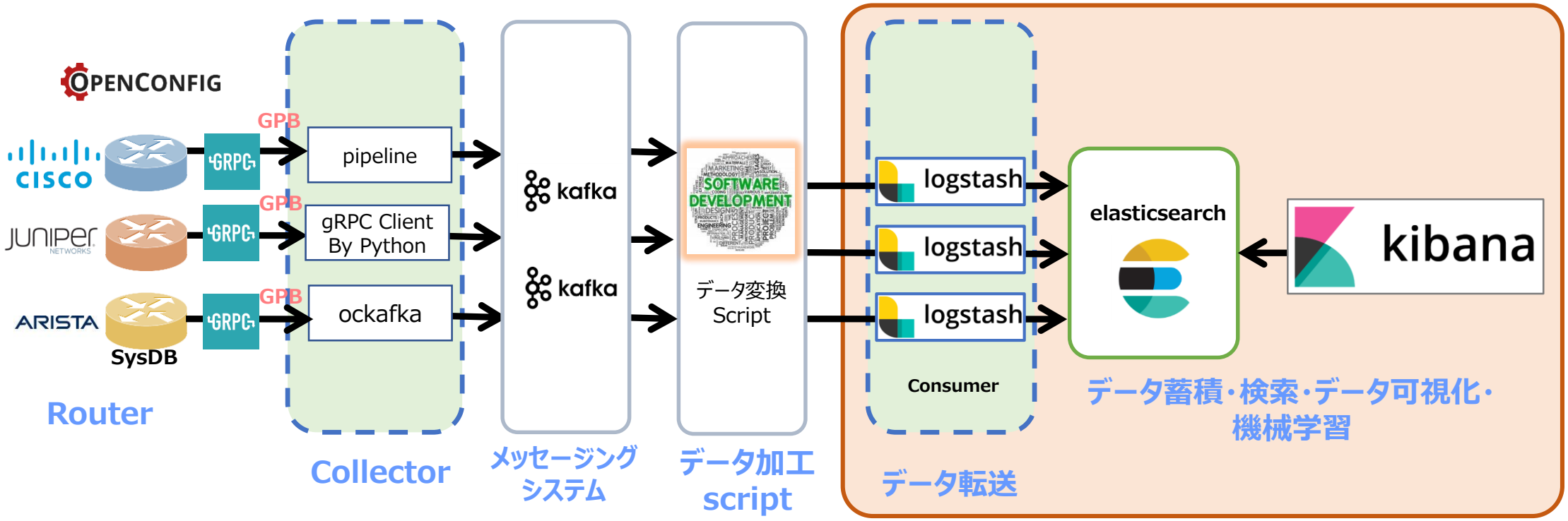
各コンポーネント説明(4/5)



データ加工 Script

- Elastic stack にて正確に可視化・解析する為に必要であった、基データへの加工処理を行うコンポーネント。（ Appendix にて詳細を説明）

各コンポーネント説明(5/5)



データ転送 / 蓄積 / 検索 / 可視化 / 機械学習

- Telemetryデータを取得・蓄積し、可視化及び機械学習を実行する
- 本PoC では、Elastic Stack (Logstash、Elasticsearch、Kibana)を使用

5. PoC 結果

取得情報と方法論（サマリ）

取得データ：Interface Counter / CPU Usage / Memory Usage

ベンダー	主な課題	理由	クリアした方法論
A社	• Array Object • 複数JSONメッセージ	• Routerの送信データ形式	• Logstash Filter(Grok filter / JSON Filter) • Python Script
B社	• Key field内の変数	• Routerの送信データ形式	• Logstash Filter(Grok filter / JSON Filter) • Python Script
C社	• データ形式（非JSON） • 外部エクスポート	• Routerの送信データ形式 • Collectorのエクスポート機能未実装	• Python Script

Elastic Stackを用いた可視化・解析には、

Routerから送信される基データの変換が必要！

データへのアプローチ方法(詳細はAppendix)

Routerが送信するTelemetry DataがElastic Stackと相性が悪く、そのままではKibanaで可視化/分析不可
⇒OpenConfigモデルは比較的きれいなデータ構造、Vendor Nativeモデルは課題あり
例：Array Object型のデータ構造
1つのメッセージに複数のJSONが内包

Routerから送信される基データの変換が必要！

データ変換方法論1:Logstash Filterにて

基データ
↓
grok filter
↓
json filter

結果：
途中でkeyの名称を変更する必要があり、実際の値を参照する際の作業が煩雑に…

データ変換方法論2:Python Scriptによる一括変換

基データ
↓
Python Script

結果：
データの構造をあらかじめ変更することで、自然な流れで値の参照ができるようになった

その他の苦労話…

予期していなかったデータ構造

- “” のような空欄のkeyが含まれていたことによるパースエラーの発生

取得内容によって異なる文字コード

- 基データをよく見ると、取得内容によりフォーマットや文字コードが異なっていた

valueのdata typeの不一致

- 同じ名前のkeyへ数値と文字列どちらも入る場合があり、mapping関連のエラーの発生

正規表現の試行錯誤

- 一括で変換できない部分はgrok filterを書いて検証の繰り返し

デフォルトで設定されている上限値

- keyが非常に長くなることがあり、デフォルトの上限値に達しておりエラーの発生

各種プラグインのIssue

- ドキュメントに書かれていない問題、仕様に直面

サマリ

当初の構想

- Telemetryの取得により、今までより粒度の細かい情報の可視化
- ベースラインの取得による、正常時と異常時の振る舞いの比較
- 各メトリックの相関性の確認

今回できたこと

- リアルタイム可視化
  - ✓ interface毎の使用帯域
  - ✓ 各種カウンター
- Burst Traffic検知
- 機械学習による異常性、特異性の検知

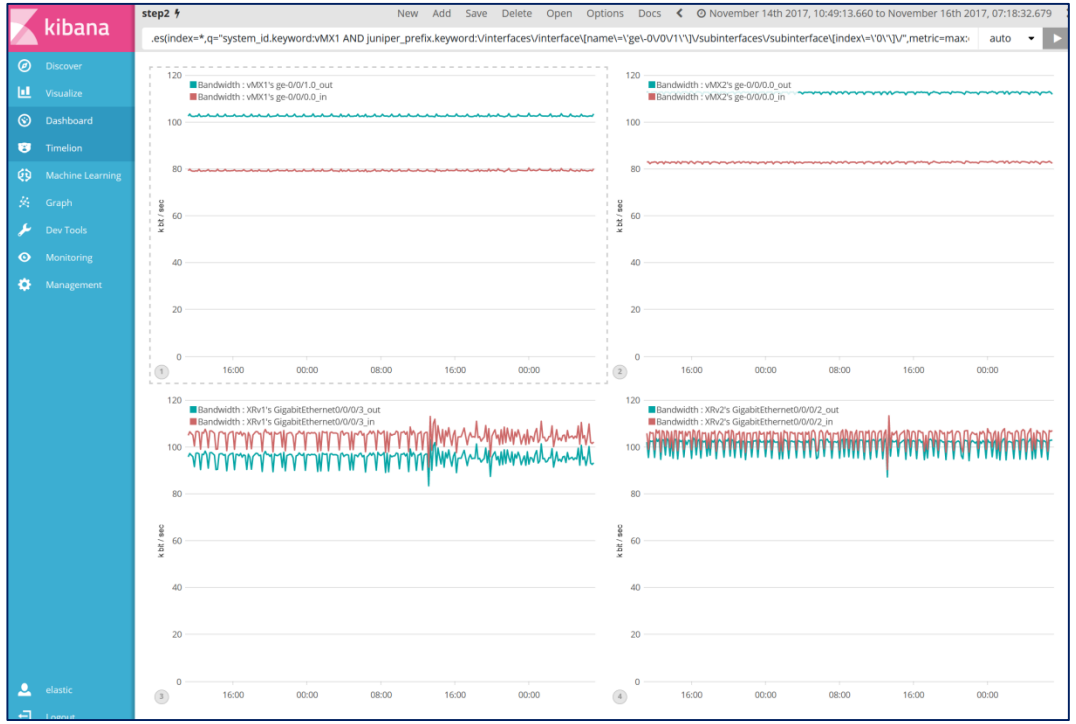
今後行うこと

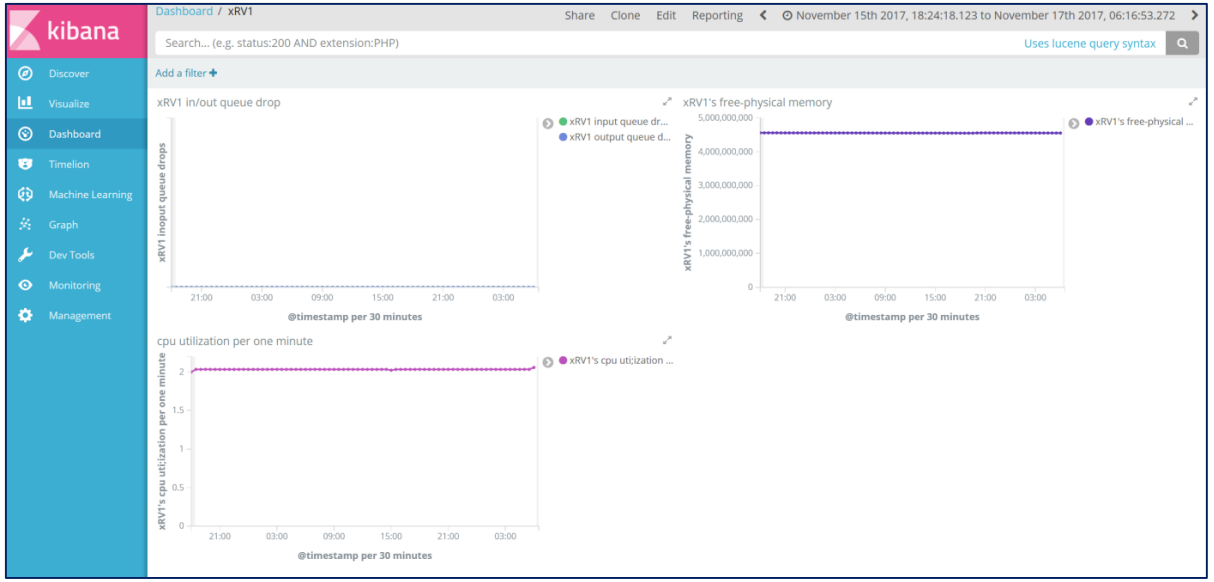
- ベースラインの取得による新しい発見
  - ✓ 予期していない異常値
  - ✓ 相関性
- 実環境でのテスト

今回可視化できたのは構想していた内容の限られたごく一部

リアルタイム可視化

Interface毎の使用帯域 / 各種カウンタ



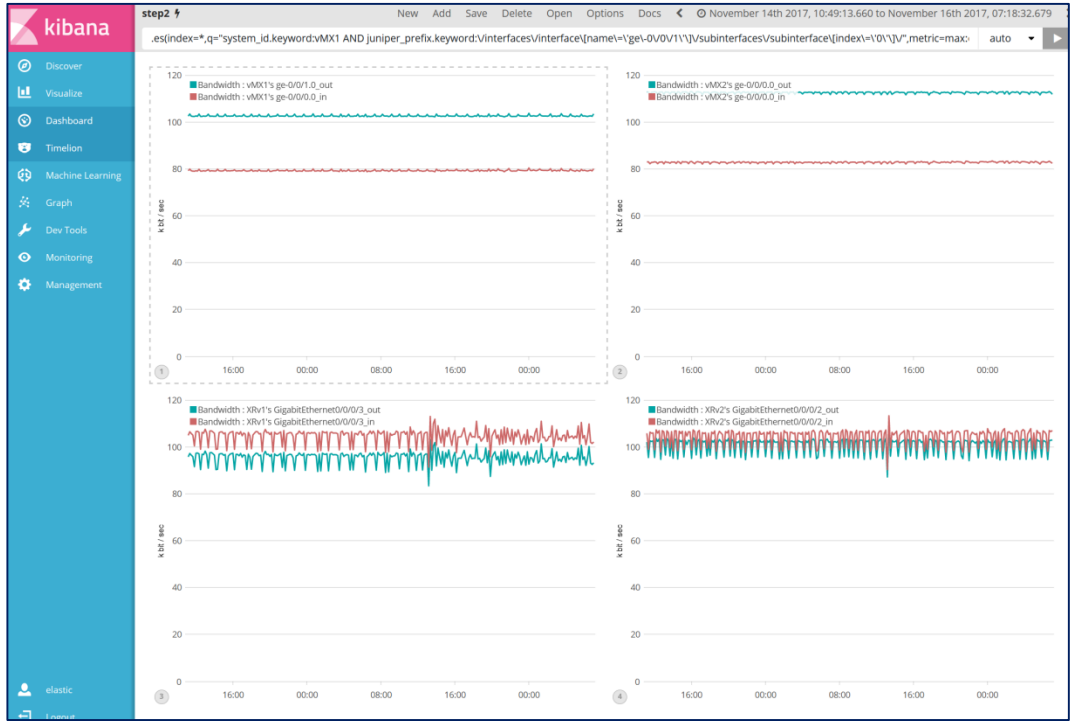


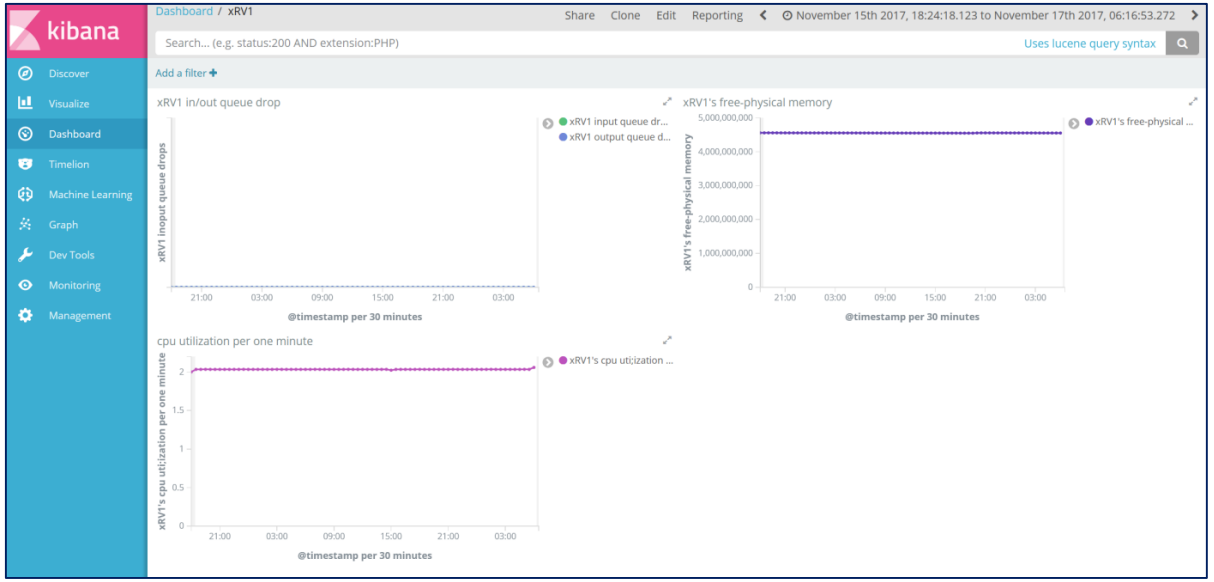
Interface帯域：
Interface Counter（Input / Output octets）を基に、Kibana Timelion 上で帯域計算を実行してグラフ化

その他カウンター：
CPU 使用率やメモリ使用率、Interface Queue drop カウンター等、Telemetry データで取得した値をグラフ化

リアルタイム可視化

Interface毎の使用帯域 / 各種カウンタ

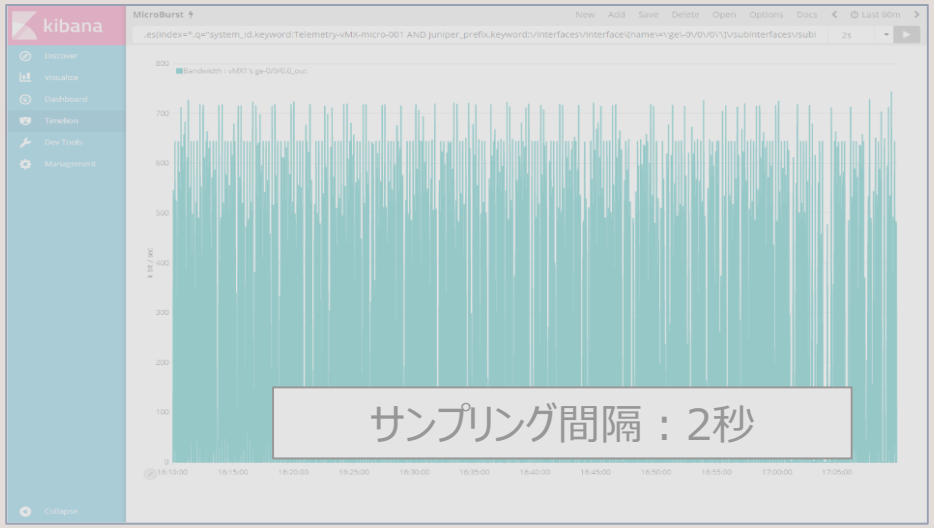


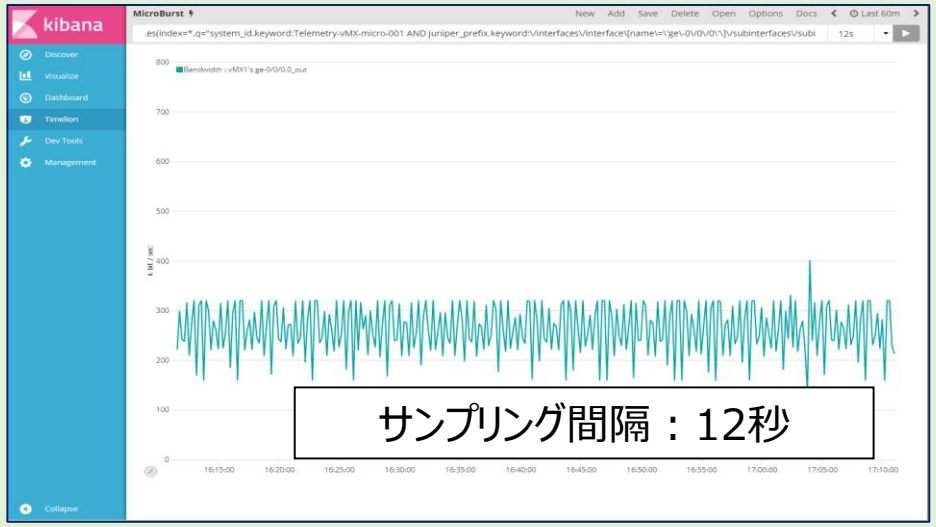


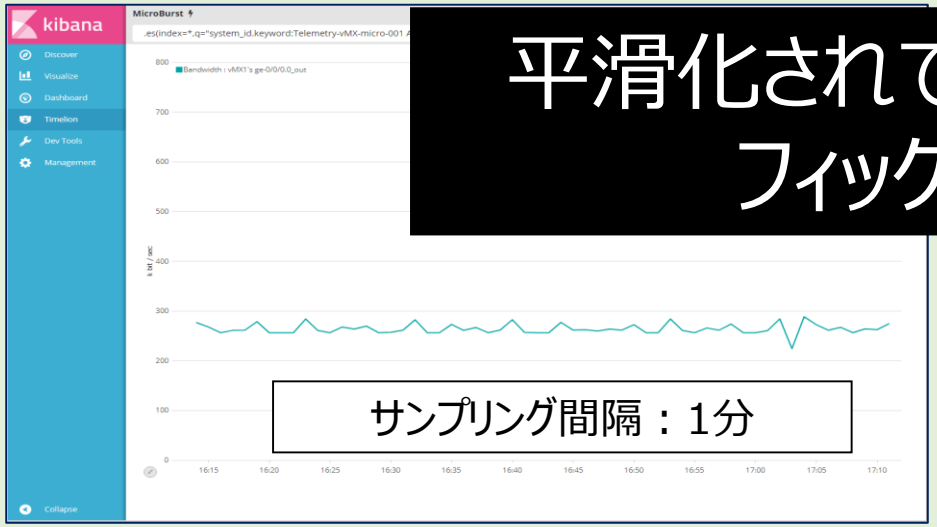
Telemetryを使用する事で、粒度の細かいデータを

可視化する事が出来る

リアルタイム可視化 – Burst Traffic検知



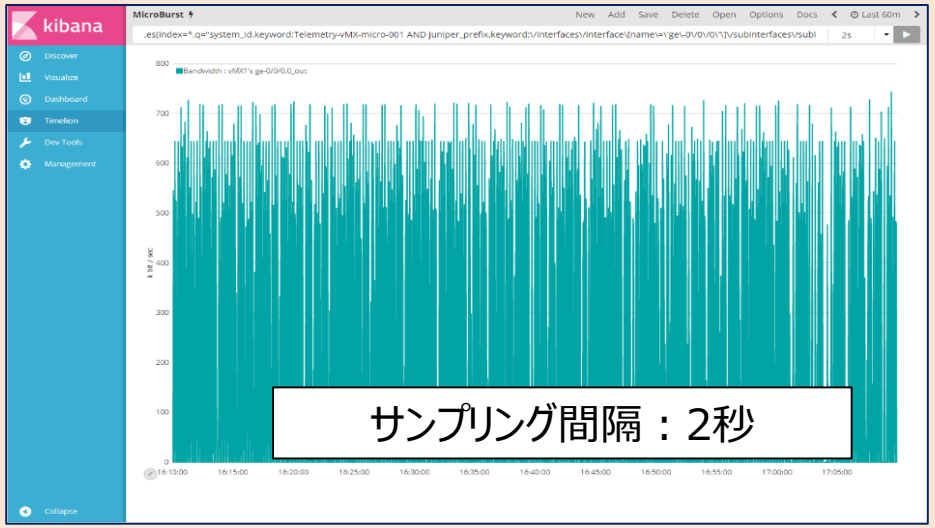


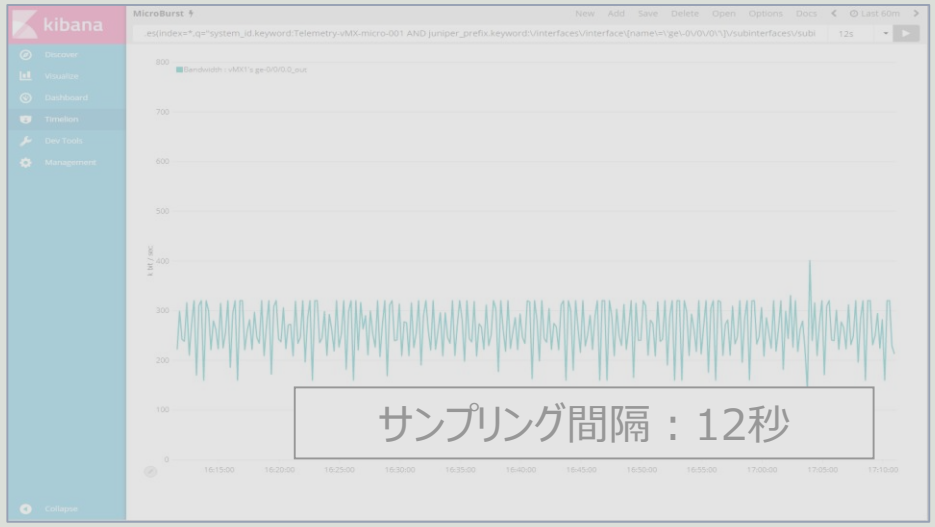


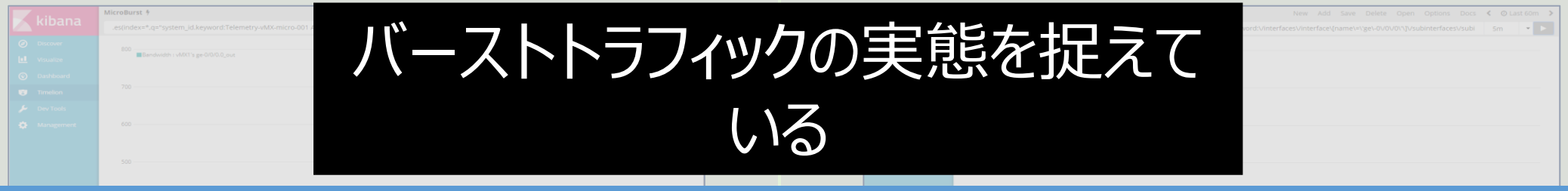
平滑化されていて、バーストトラフィックが見えない

試験条件：100byte 1000pps×3秒と0pps×4秒を繰り返すTrafficを印加

リアルタイム可視化 – Burst Traffic検知







バーストトラフィックの実態を捉えている

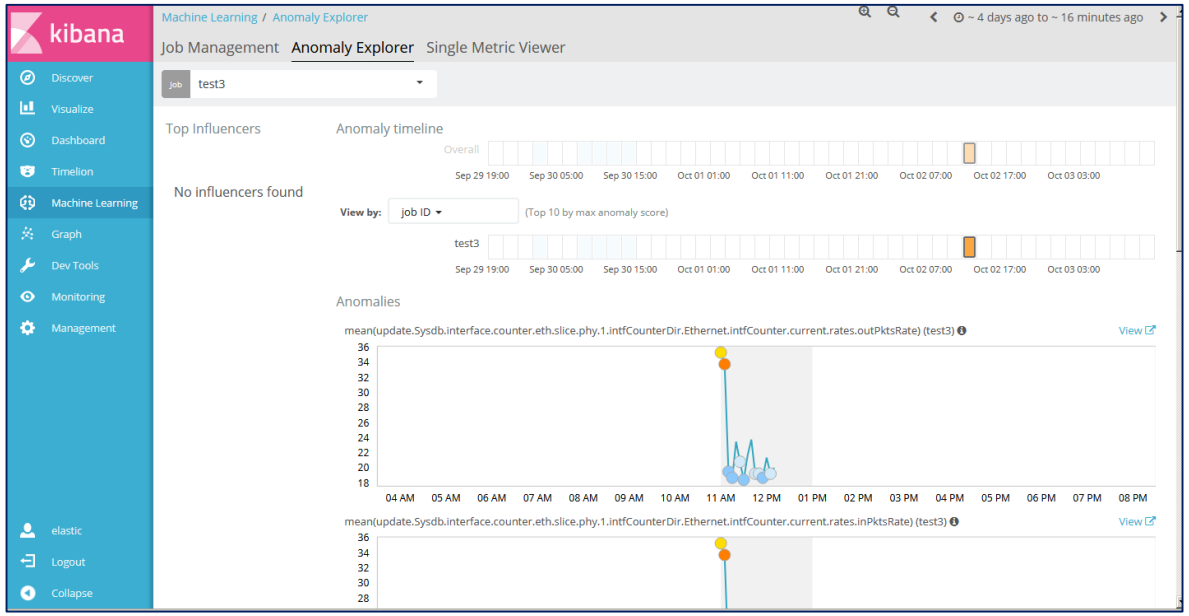
Telemetryを使用する事で、今まで見えていなかった事象
（≒**サイレント障害**）を可視化、検知する事が出来る

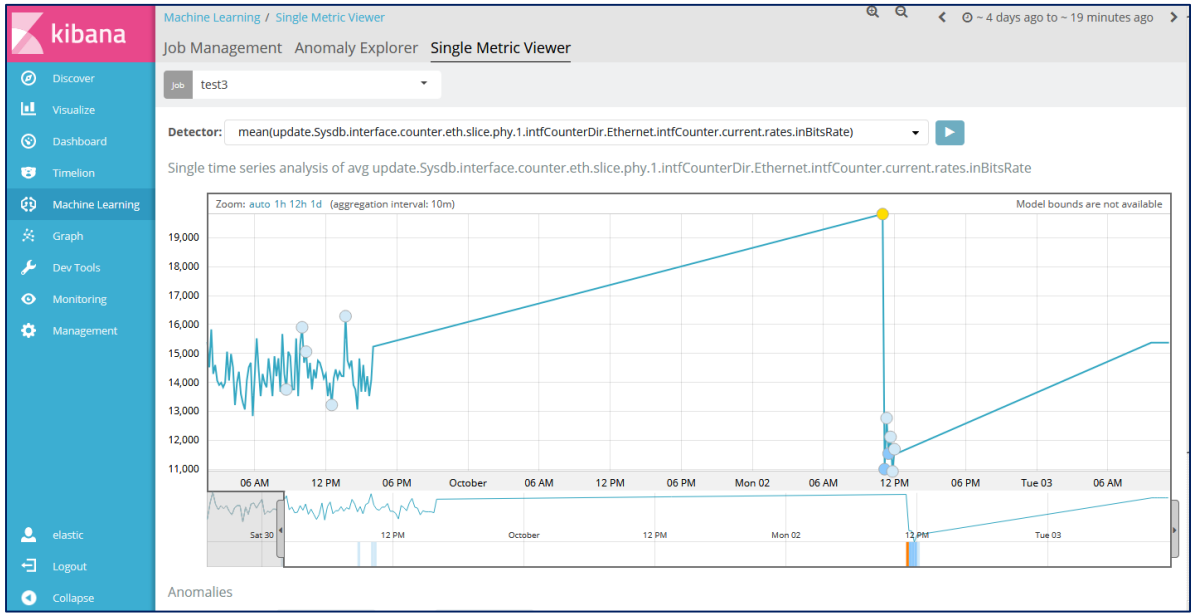
試験条件：100byte 1000pps×3秒と0pps×4秒を繰り返すTrafficを印加

効果:機械学習 -異常性, 特異性の検知

Interface カウンター等を Machine Learning に掛け、異常性/特異性の検知に成功しています。

Case 1）Interface閉塞に伴う明白な異常を検知

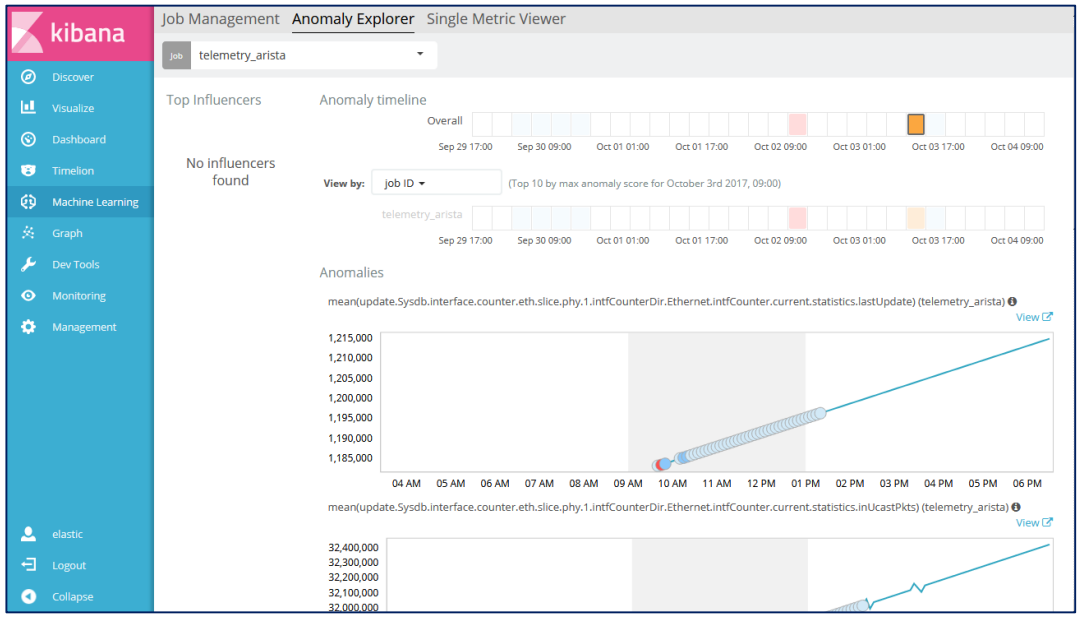


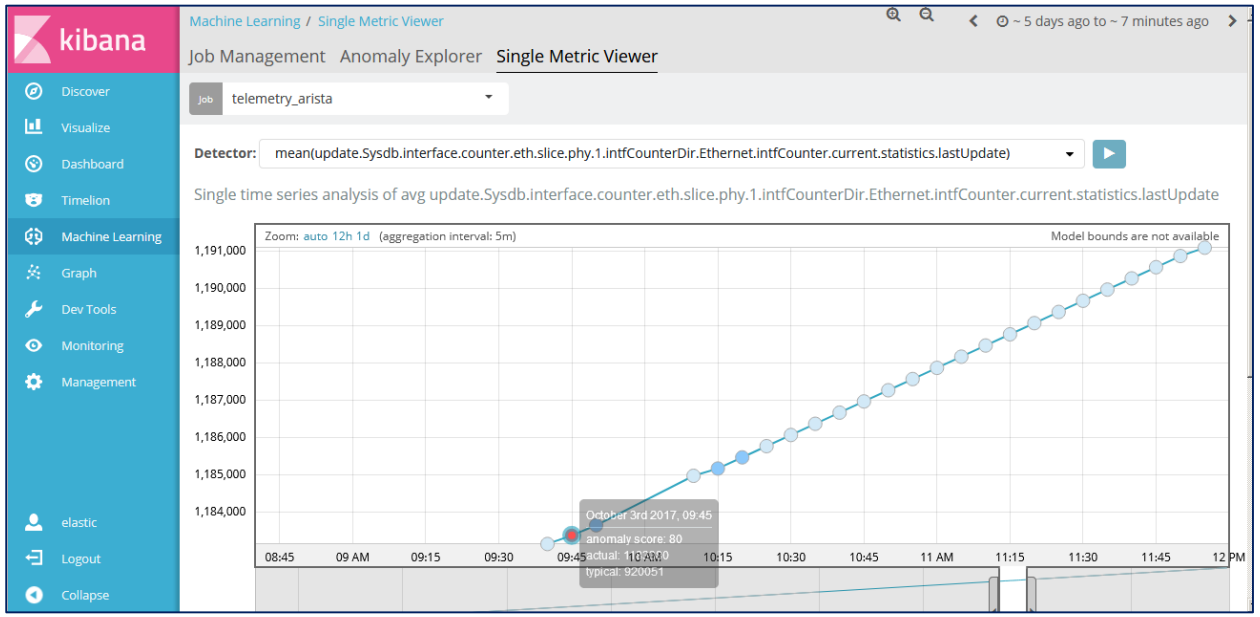


異常の検知をトリガーに、**サイレント障害**の通知や、設定修正のための**ワークフローエンジンを実行**する事が出来る

効果:機械学習 -異常性, 特異性の検知

Interface カウンター等を Machine Learning に掛け、異常性/特異性の検知に成功しています。
Case 2）人の目では気付き難い微小な変化を検知（Interfaceカウンタ増減の微小な変化を検知）

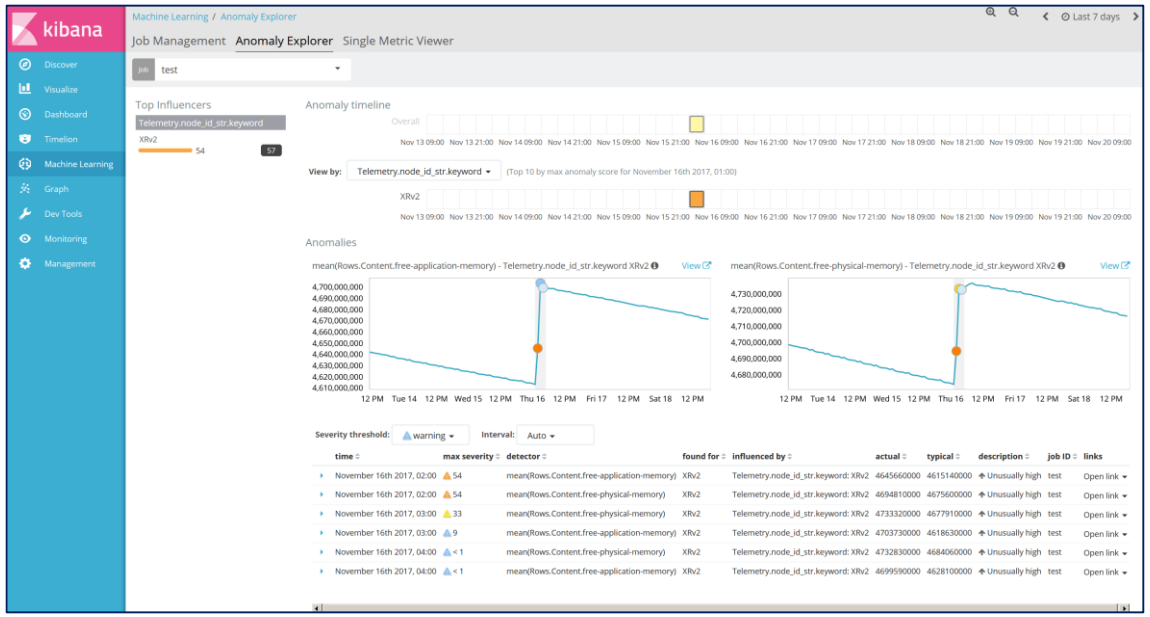




異常の検知をトリガーに、**サイレント障害**の通知や、設定
修正のための**ワークフローエンジンを実行**する事が出来る

効果:機械学習 -異常性, 特異性の検知

Interface カウンター等を Machine Learning に掛け、異常性/特異性の検知に成功しています。
Case 3）人の目では気付き難い微小な変化を検知（メモリ解放動作）



異常の検知をトリガーに、**サイレント障害**の通知や、設定
修正のための**ワークフローエンジンを実行**する事が出来る

PoC結果のまとめ

様々な角度からの可視化が可能になる

- Telemetryにより、これまでのネットワークインフラ上で把握が難しかった側面、サイレント障害への対応が可能になる

機械学習との連携により、従来では気付いていなかった微小な変化を捉えることが可能になる

- 今まで発生していたが気付くことのなかった障害（サイレント障害）に気づくことが可能になる

Collectorツールの選択も重要

- 目的により、最適なツールが異なる。データの柔軟性や機械学習の有無、上位レイヤとの連携を考えた上で、自作やElastic Stack系、ベンダーが提供するツールを選択する必要がある

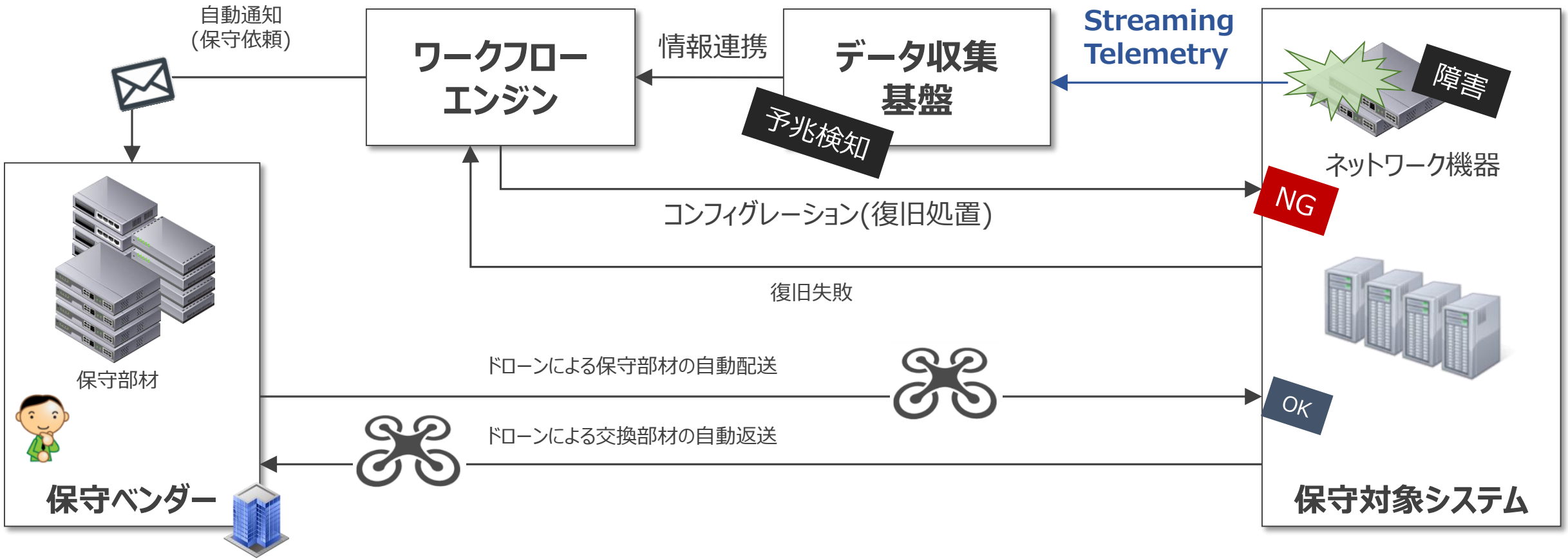
データを送信する技術の必要性 ⇔ データを変換する技術の必要性

- 予期していないエラーに引っかかることも多々あり。データを送信する際、変換する際のどちらにも課題が見えてきた。データを送るフォーマットや文字コード等は、是非統一して送信してほしい。

6. 総括

目指す未来のかたち

システム障害に対して、サービスに影響を与えずに**自動復旧**する世界の実現



解決すべき課題

実現に向けて、課題は**まだまだ山積み**！！(゜Д゜)

ディスカッションしたいこと

- ✓ サイレント障害による痛い経験
- ✓ 予兆検知による予防保守の抱える課題


ご清聴ありがとうございました！！

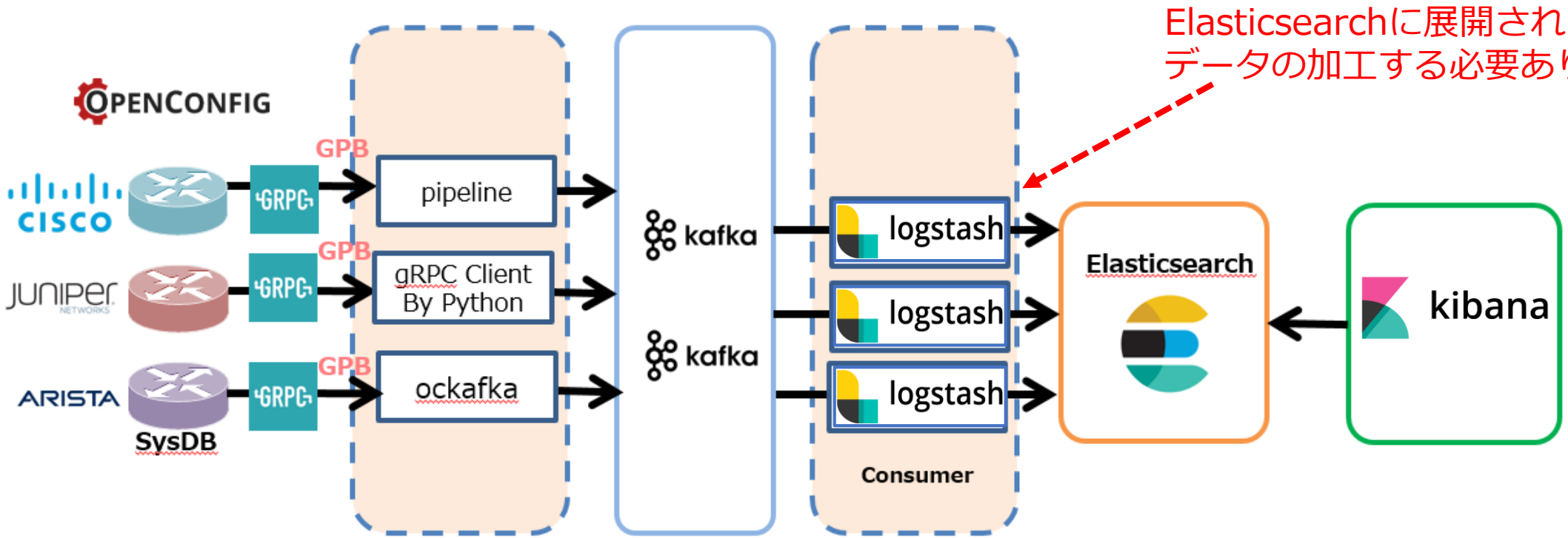
Appendix

PoC 構築時に気付いた事
データを変換する技術の必要性

PoC 構築時に気付いた事：データを加工する技術の必要性

PoCを作成して気付いた事

Routerが送信するTelemetry DataがElastic Stackと相性が悪く、そのままではKibanaで可視化 / 分析が出来ない事が判明。（下図にある、データ変更Script無し構成にて判明。）
OpenConfigモデルは比較的きれいなデータ構造であったが、あるベンダーのNativeモデルは課題が山積み。故にRouterから送信されるデータ（基データ）の変更が必要となった。



【補足】Elastic Stackを使ったデータのIndex化

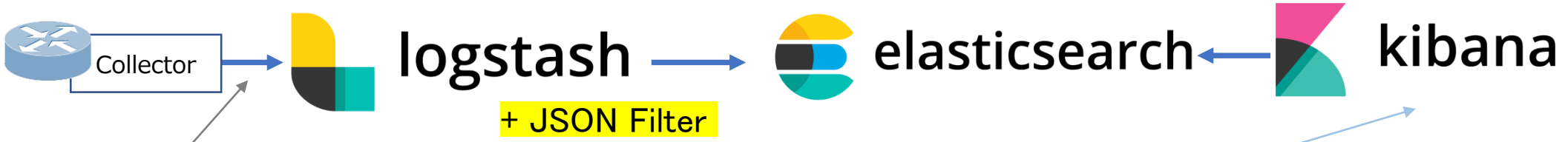
Elastic Stackの処理：通常


```
{
    "Source": "10.44.160.206:10000",
    "Telemetry": {
        "collection_end_time": 1510116522502,
        "collection_id": 202987,
        "collection_start_time": 1510116522413,
        "encoding_path": "openconfig-interfaces:interfaces/interface",
        "msg_timestamp": 1510116522413,
        "node_id_str": "XRv-Telemetry",
        "subscription_id_str": "Sub7"
    }
}
```


Key	Value
@timestamp	November 19th 2017, 05:25:19.319
_id	AV_QzpDoGi09dMQprp2D
_index	telemetry_cisco_sampling_xrv_2017.11
_type	logs
message	{"Source":"10.44.160.206:10000","Telemetry":{"node_id_str":"XRv-Telemetry","subscription_id_str":"Sub7","encoding_path":"openconfig-interfaces:interfaces/interface","collection_id":202987,"collection_start_time":1510116522413,"msg_timestamp":1510116522413,"collection_end_time":1510116522502}}

【補足】Elastic Stackを使ったデータのIndex化

Elastic Stackの処理：JSON Filterを用いたMessageのKey/Value化




```
{
    "Source": "10.44.160.206:10000",
    "Telemetry": {
        "collection_end_time": 1510116522502,
        "collection_id": 202987,
        "collection_start_time": 1510116522413,
        "encoding_path": "openconfig-
interfaces:interfaces/interface",
        "msg_timestamp": 1510116522413,
        "node_id_str": "XRv-Telemetry",
        "subscription_id_str": "Sub7"
    }
}
```


Key	Value
@timestamp	November 19th 2017, 05:25:19.319
_id	AV_QzpDoGi09dMQprp2D
_index	telemetry_cisco_sampling_xrv_2017.11
_type	logs
Source	10.44.160.206:10000
Telemetry. collection_end_time	1510116522502
Telemetry. collection_id	202987
Telemetry. collection_start_time	1510116522413
Telemetry. encoding_path	openconfig-interfaces:interfaces/interface
<snip>	

PoC 構築時に気付いた事：データを変換する技術の必要性

処置を要するデータ型（例）

① Kibana でデータとして取り扱うことが出来ない「Array Object 型」への処置
② 1つのメッセージに複数 JSON が内包されるケースへの処置
③ key 上の変数に対する処置
④ Kibana でデータとして認識するよう、Telemetry data の JSON 形式化

処理が必要なデータ型：Array Object型

Array Object型

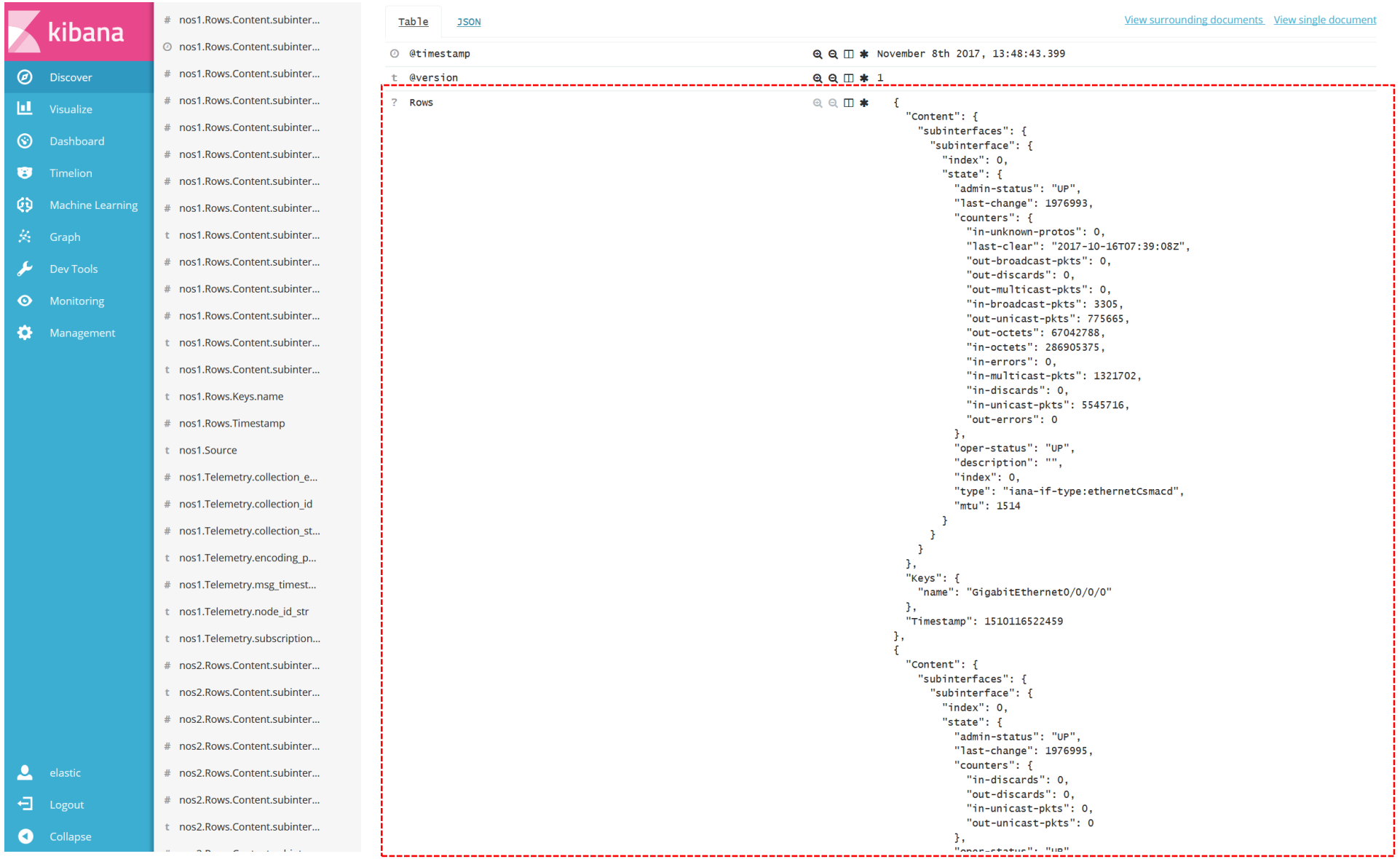
1つの Key に対し複数の Value が存在するケース。
Rows 以降のデータが Array 型となっており（黄色ハイライト部分）、Elastic Stackで取扱い不可。

{"Source":"10.44.160.206:10000","Telemetry":{"node_id_str":"XRv-Telemetry","subscription_id_str":"Sub7","encoding_path":"openconfig-interfaces:interfaces/interface","collection_id":202987,"collection_start_time":1510116522413,"msg_timestamp":1510116522413,"collection_end_time":1510116522502},"Rows":[{"Timestamp":1510116522459,"Keys":{"name":"GigabitEthernet0/0/0/0"},"Content":{"subinterfaces":{"subinterface":{"index":0,"state":{"admin-status":"UP","counters":{"in-broadcast-pkts":3305,"in-discards":0,"in-errors":0,"in-multicast-pkts":1321702,"in-octets":286905375,"in-unicast-pkts":5545716,"in-unknown-protos":0,"last-clear":"2017-10-16T07:39:08Z","out-broadcast-pkts":0,"out-discards":0,"out-errors":0,"out-multicast-pkts":0,"out-octets":67042788,"out-unicast-pkts":775665},"description":"","index":0,"last-change":1976993,"mtu":1514,"oper-status":"UP","type":"iana-if-type:ethernetCsmacd"}}}},{"Timestamp":1510116522470,"Keys":{"name":"Loopback0"},"Content":{"subinterfaces":{"subinterface":{"index":0,"state":{"admin-status":"UP","counters":{"in-discards":0,"in-unicast-pkts":0,"out-discards":0,"out-unicast-pkts":0},"description":"","index":0,"last-change":1976995,"mtu":1500,"oper-status":"UP","type":"iana-if-type:softwareLoopback"}}}},{"Timestamp":1510116522479,"Keys":{"name":"MgmtEth0/RP0/CPU0/0"},"Content":{"subinterfaces":{"subinterface":{"index":0,"state":{"admin-status":"UP","counters":{"in-broadcast-pkts":17952363,"in-discards":48003,"in-errors":0,"in-multicast-pkts":32947,"in-octets":1130726619,"in-unicast-pkts":36558948,"in-unknown-protos":0,"last-clear":"2017-10-16T07:36:34Z","out-broadcast-pkts":0,"out-discards":0,"out-errors":0,"out-multicast-pkts":0,"out-octets":363109046,"out-unicast-pkts":746754},"description":"","index":0,"last-change":1976816,"mtu":1514,"oper-status":"UP","type":"iana-if-type:ethernetCsmacd"}}}},{"Timestamp":1510116522490,"Keys":{"name":"Null0"},"Content":{"subinterfaces":{"subinterface":{"index":0,"state":{"admin-status":"UP","counters":{"in-broadcast-pkts":0,"in-discards":0,"in-errors":0,"in-multicast-pkts":0,"in-octets":0,"in-unicast-pkts":0,"in-unknown-protos":0,"last-clear":"2017-10-16T07:36:56Z","out-broadcast-pkts":0,"out-discards":0,"out-errors":0,"out-multicast-pkts":0,"out-octets":0,"out-unicast-pkts":0},"description":"","index":0,"last-change":1977175,"mtu":1500,"oper-status":"UP","type":"iana-if-type:other"}}}},{"Timestamp":1510116522501,"Keys":{"name":"tunnel-te1"},"Content":{"subinterfaces":{"subinterface":{"index":0,"state":{"admin-status":"UP","counters":{"in-broadcast-pkts":0,"in-discards":0,"in-errors":0,"in-multicast-pkts":0,"in-octets":0,"in-unicast-pkts":0,"in-unknown-protos":0,"last-clear":"2017-10-16T07:38:47Z","out-broadcast-pkts":0,"out-discards":0,"out-errors":0,"out-multicast-pkts":0,"out-octets":0,"out-unicast-pkts":0},"description":"","index":0,"last-change":1656104,"mtu":1500,"oper-status":"UP","type":"iana-if-type:mplsTunnel"}}}}]}

Key	Value
@timestamp	November 19th 2017, 05:25:19.319
_id	AV_QzpDoGi09dMQprp2D
_index	telemetry_cisco_sampling_xrv_2017.11
_type	logs
Source	10.44.160.206:10000
Telemetry.	1510116522502
<snip>	

Rows	"Content": {
	"subinterfaces": {
	"subinterface": {
	"index": 0,
	"state": {
	"admin-status": "UP",
	"counters": {
	"in-broadcast-pkts": 3305,
	<snip>

処理が必要なデータ型：Array Object型



処理が必要なデータ型：1つのメッセージに複数 JSON が内包

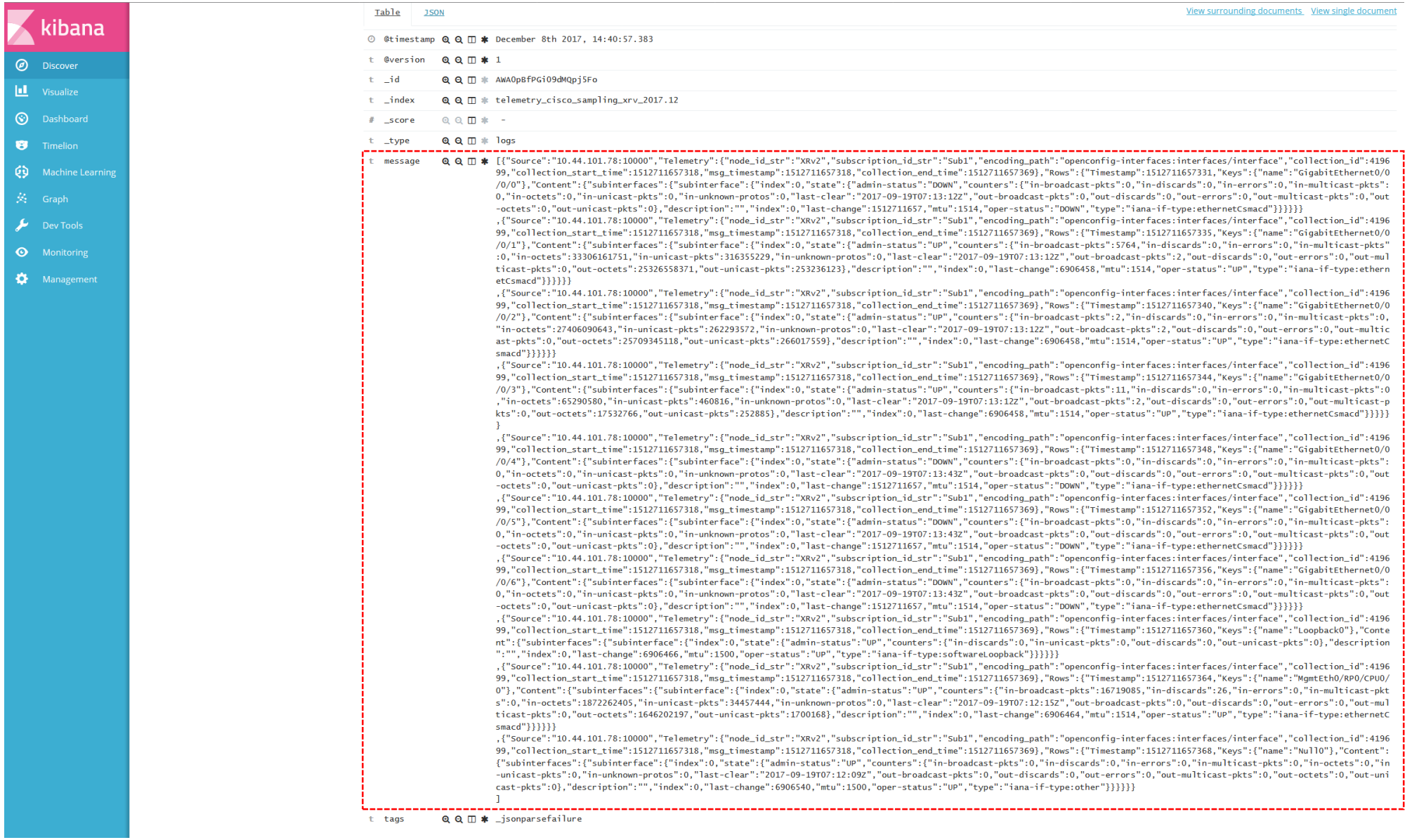
<u>複数JSON内包型</u>

1つのメッセージの中に複数の JSON が存在し、それら JSON が[]で括られている。
LogstashのJSON Filterが正常に動作しない。（JSON parse failure）

[{"Source":"10.44.160.206:10000","Telemetry":{"node_id_str":"XRv-Telemetry","subscription_id_str":"Sub7","encoding_path":"openconfig-interfaces:interfaces/interface","collection_id":199399,"collection_start_time":1510107352700,"msg_timestamp":1510107352700,"collection_end_time":1510107352774},"Rows":{"Timestamp":1510107352733,"Keys":{"name":"GigabitEthernet0/0/0/0"},"Content":{"subinterfaces":{"subinterface":{"index":0,"state":{"admin-status":"UP","counters":{"in-broadcast-pkts":3290,"in-discards":0,"in-errors":0,"in-multicast-pkts":1315573,"in-octets":285579805,"in-unicast-pkts":5520053,"in-unknown-protos":0,"last-clear":"2017-10-16T07:39:08Z","out-broadcast-pkts":0,"out-discards":0,"out-errors":0,"out-multicast-pkts":0,"out-octets":66739126,"out-unicast-pkts":772161},"description":"","index":0,"last-change":1967823,"mtu":1514,"oper-status":"UP","type":"iana-if-type:ethernetCsmacd"}}}}} ,{"Source":"10.44.160.206:10000","Telemetry":{"node_id_str":"XRv-Telemetry","subscription_id_str":"Sub7","encoding_path":"openconfig-interfaces:interfaces/interface","collection_id":199399,"collection_start_time":1510107352700,"msg_timestamp":1510107352700,"collection_end_time":1510107352774},"Rows":{"Timestamp":1510107352740,"Keys":{"name":"Loopback0"},"Content":{"subinterfaces":{"subinterface":{"index":0,"state":{"admin-status":"UP","counters":{"in-discards":0,"in-unicast-pkts":0,"out-discards":0,"out-unicast-pkts":0},"description":"","index":0,"last-change":1967825,"mtu":1500,"oper-status":"UP","type":"iana-if-type:softwareLoopback"}}}}} ,{"Source":"10.44.160.206:10000","Telemetry":{"node_id_str":"XRv-Telemetry","subscription_id_str":"Sub7","encoding_path":"openconfig-interfaces:interfaces/interface","collection_id":199399,"collection_start_time":1510107352700,"msg_timestamp":1510107352700,"collection_end_time":1510107352774},"Rows":{"Timestamp":1510107352749,"Keys":{"name":"MgmtEth0/RP0/CPU0/0"},"Content":{"subinterfaces":{"subinterface":{"index":0,"state":{"admin-status":"UP","counters":{"in-broadcast-pkts":17928814,"in-discards":47770,"in-errors":0,"in-multicast-pkts":32794,"in-octets":1128810186,"in-unicast-pkts":36505038,"in-unknown-protos":0,"last-clear":"2017-10-16T07:36:34Z","out-broadcast-pkts":0,"out-discards":0,"out-errors":0,"out-multicast-pkts":0,"out-octets":357083511,"out-unicast-pkts":739466},"description":"","index":0,"last-change":1967648,"mtu":1514,"oper-status":"UP","type":"iana-if-type:ethernetCsmacd"}}}}} ,{"Source":"10.44.160.206:10000","Telemetry":{"node_id_str":"XRv-Telemetry","subscription_id_str":"Sub7","encoding_path":"openconfig-interfaces:interfaces/interface","collection_id":199399,"collection_start_time":1510107352700,"msg_timestamp":1510107352700,"collection_end_time":1510107352774},"Rows":{"Timestamp":1510107352761,"Keys":{"name":"Null0"},"Content":{"subinterfaces":{"subinterface":{"index":0,"state":{"admin-status":"UP","counters":{"in-broadcast-pkts":0,"in-discards":0,"in-errors":0,"in-multicast-pkts":0,"in-octets":0,"in-unicast-pkts":0,"in-unknown-protos":0,"last-clear":"2017-10-16T07:36:56Z","out-broadcast-pkts":0,"out-discards":0,"out-errors":0,"out-multicast-pkts":0,"out-octets":0,"out-unicast-pkts":0},"description":"","index":0,"last-change":1968005,"mtu":1500,"oper-status":"UP","type":"iana-if-type:other"}}}}} ,{"Source":"10.44.160.206:10000","Telemetry":{"node_id_str":"XRv-Telemetry","subscription_id_str":"Sub7","encoding_path":"openconfig-interfaces:interfaces/interface","collection_id":199399,"collection_start_time":1510107352700,"msg_timestamp":1510107352700,"collection_end_time":1510107352774},"Rows":{"Timestamp":1510107352773,"Keys":{"name":"tunnel-te1"},"Content":{"subinterfaces":{"subinterface":{"index":0,"state":{"admin-status":"UP","counters":{"in-broadcast-pkts":0,"in-discards":0,"in-errors":0,"in-multicast-pkts":0,"in-octets":0,"in-unicast-pkts":0,"in-unknown-protos":0,"last-clear":"2017-10-16T07:38:47Z","out-broadcast-pkts":0,"out-discards":0,"out-errors":0,"out-multicast-pkts":0,"out-octets":0,"out-unicast-pkts":0},"description":"","index":0,"last-change":1646934,"mtu":1500,"oper-status":"UP","type":"iana-if-type:mplsTunnel"}}}}}]

Key	Value
@timestamp	November 19th 2017, 05:25:19.319
_id	AV_QzpDoGi09dMQprp2D
_index	telemetry_cisco_sampling_xrv_2017.11
_type	logs
Source	10.44.160.206:10000
Telemetry.	1510116522502
<snip>	
tags	_jsonparsefailure

処理が必要なデータ型：1つのメッセージに複数 JSON が内包



データ変換方法論1：Logstash Filter にて

データ変換方法論1：Logstash Filter にて

Elastic Stackの処理：Logstash grok filterを用いたデータ加工

grok Filter

```
{
“json_message1”:
“                                                    ",

“json_message2”:
“                                                    ",

“json_message3”:
“                                                    ",

“json_message4”:
“                                                    “,

“json_message5”:
“                                                    ",
}
```

JSON Filter

Key	Value
@timestamp	November 19th 05:25:19.319
<snip>	
nos1.Rows.Content.subinterface terface.index	0
nos1.Rows.Content.subinterface terface.state.admin-status	up
<snip>	
nos2.Rows.Content.subinterface terface.index	0
nos2.Rows.Content.subinterface terface.state.admin-status	up
<snip>	
nos3.Rows.Content.subinterface terface.index	0
nos3.Rows.Content.subinterface terface.state.admin-status	up
<snip>	

Grok filter で各要素を抜き出して新たなJSONを生成し、その後にJSON filterでKey / Value化する

データ変換方法論1：Logstash Filter にて

Elastic Stackの処理：Logstash grok filterを用いたデータ加工

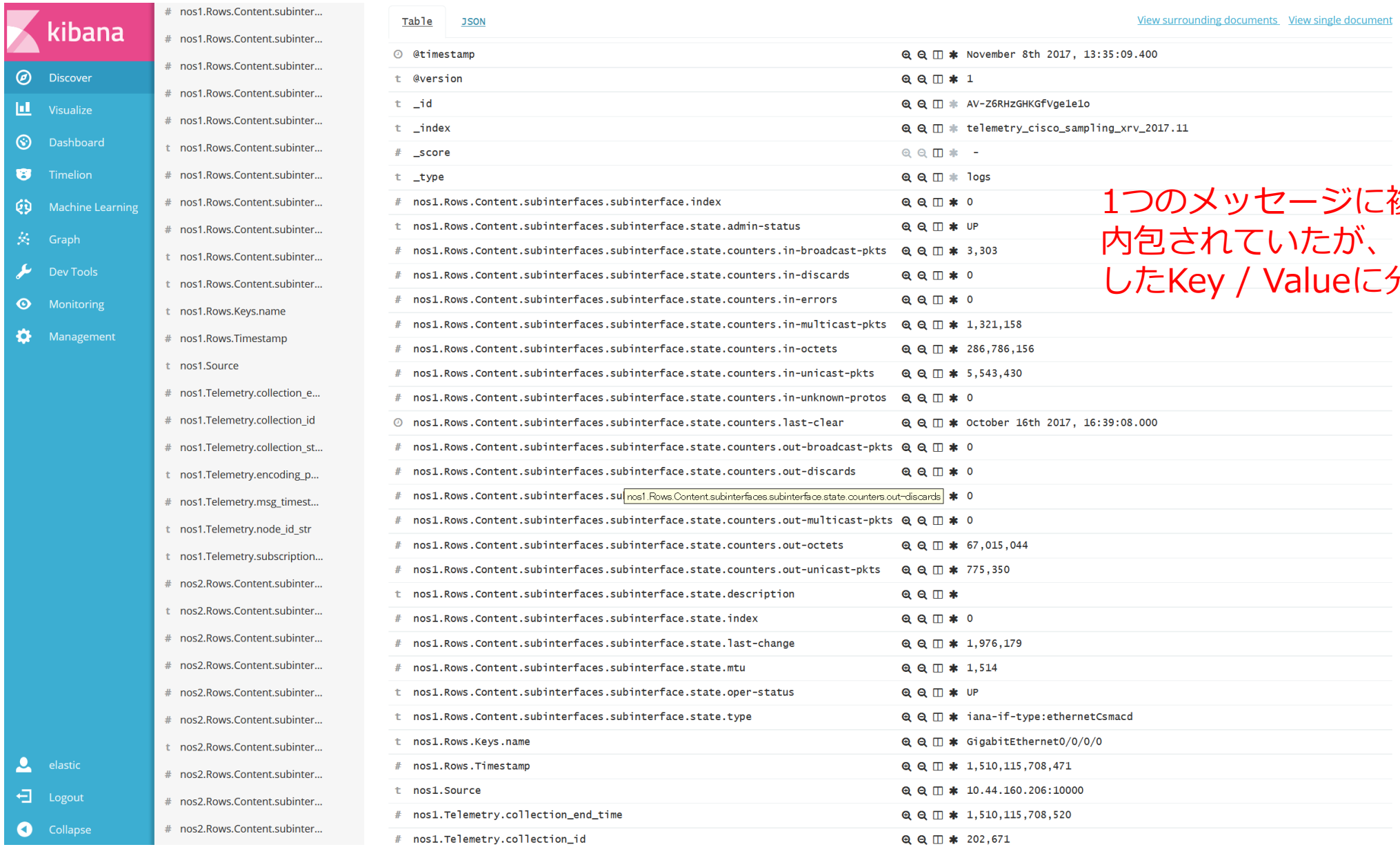
grok filter:
不要な[]を削除している

grok filter:
データを5つのBlockに分解し、各々を新定義のKeyへ設置

json filter:
Key要素の重複のため、Key先頭に新たな識別子を定義

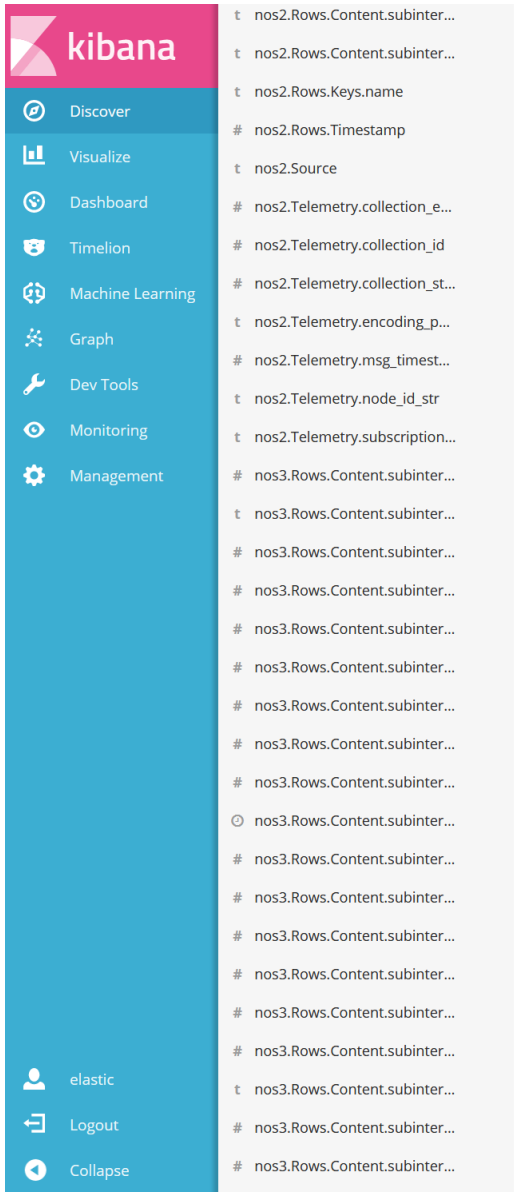
```
[root@localhost config-dir]# cat logstash.conf_jsonevent
input {
 kafka {
  bootstrap_servers  => "10.44.160.98:32768"
  topics => "telemetry_cisco_xrv"
 }
}
filter {
 date {
  match => [ "timestamp",  "UNIX_MS" ]
  remove_field  => [ "timestamp" ]
 }
 grok {
  match => { "message"  => "¥[%{GREEDYDATA:json_message0}¥]"}
 }

 grok {
  match => { "message"  =>
"¥[%{GREEDYDATA:json_message1}¥n,%{GREEDYDATA:json_message2}¥n,%{GREEDYDATA:json_message3}¥n,%{
GREEDYDATA:json_message4}¥n,%{GREEDYDATA:json_message5}¥]"}
 }

 json {
  source => "json_message0"
  skip_on_invalid_json  => "true"
  target=> nos0
  remove_field  => "json_message0"
 }
 json {
  source => "json_message1"
  target=> nos1
  remove_field  => "json_message1"
 }
 json {
  source => "json_message2"
  target=> nos2
  remove_field  => "json_message2"
 }
 json {
  source => "json_message3"
  target=> nos3
  remove_field  => "json_message3"
 }
 json {
  source => "json_message4"
  target=> nos4
  remove_field  => "json_message4"
 }
 json {
  source => "json_message5"
  target=> nos5
  remove_field  => "json_message5"
 }
 mutate { remove_field  => [ "json_message0"  ,"message" ] }
}

output {
 elasticsearch {
  hosts  => "10.44.160.98:9200"
  user => elastic
  password  => changeme
  index  => "telemetry_cisco_sampling_xrv_%{+YYYY.MM}"
 }
}
```

データ変換方法論1：Logstash Filter にて



1つのメッセージに複数JSONが内包されていたが、すべて独立したKey / Valueに分割された

データ変換方法論1：Logstash Filter にて

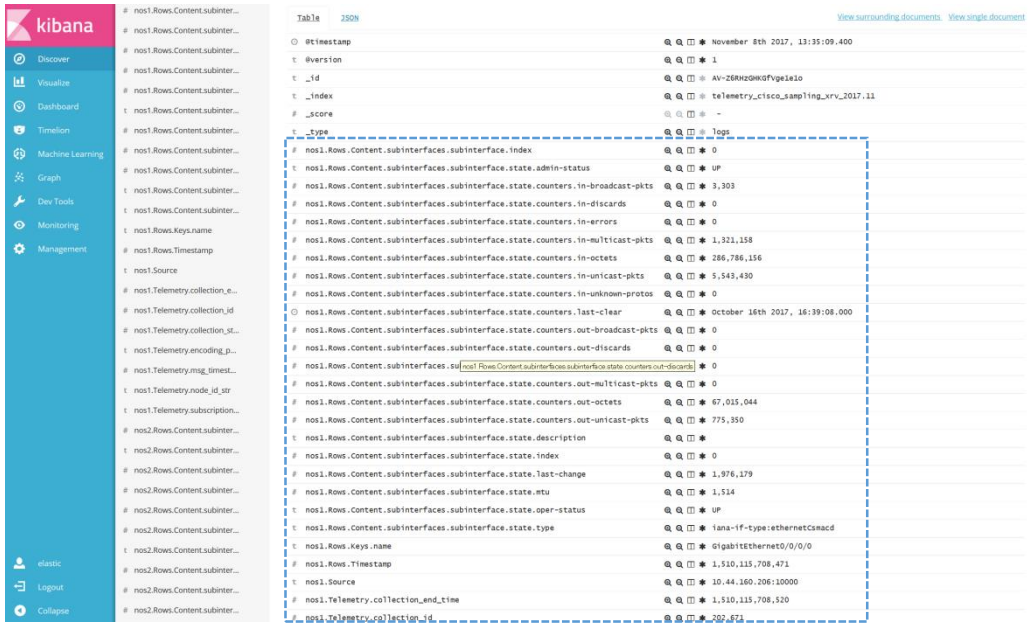


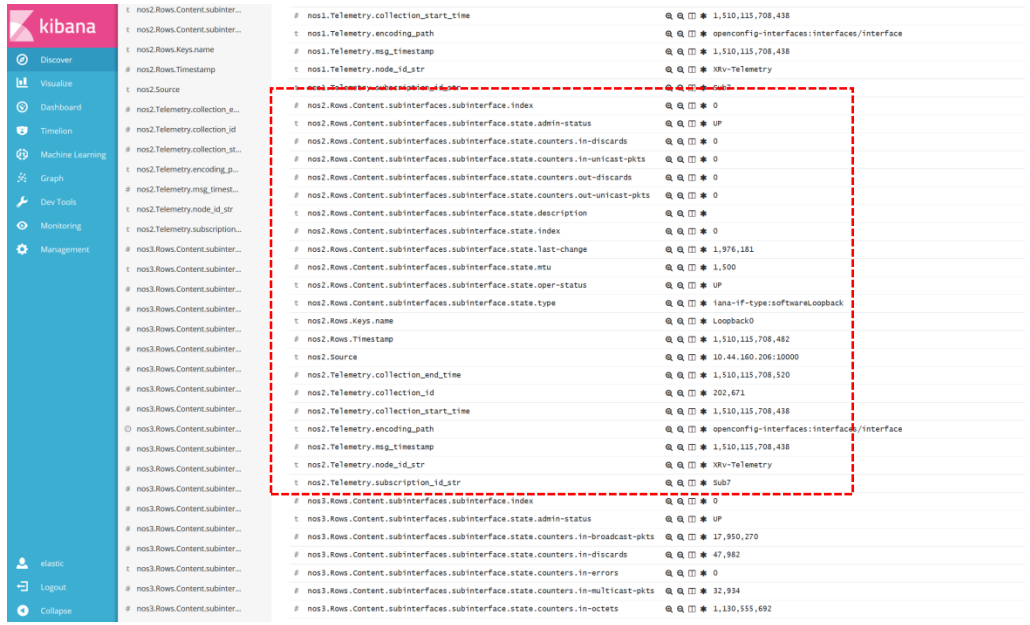
#	nos1.Telemetry.collection_start_time	1,510,115,708,438
t	nos1.Telemetry.encoding_path	openconfig-interfaces:interfaces/interface
#	nos1.Telemetry.msg_timestamp	1,510,115,708,438
t	nos1.Telemetry.node_id_str	XRv-Telemetry
t	nos1.Telemetry.subscription_id_str	Sub7
#	nos2.Rows.Content.subinterfaces.subinterface.index	0
t	nos2.Rows.Content.subinterfaces.subinterface.state.admin-status	UP
#	nos2.Rows.Content.subinterfaces.subinterface.state.counters.in-discards	0
#	nos2.Rows.Content.subinterfaces.subinterface.state.counters.in-unicast-pkts	0
#	nos2.Rows.Content.subinterfaces.subinterface.state.counters.out-discards	0
#	nos2.Rows.Content.subinterfaces.subinterface.state.counters.out-unicast-pkts	0
t	nos2.Rows.Content.subinterfaces.subinterface.state.description	
#	nos2.Rows.Content.subinterfaces.subinterface.state.index	0
#	nos2.Rows.Content.subinterfaces.subinterface.state.last-change	1,976,181
#	nos2.Rows.Content.subinterfaces.subinterface.state.mtu	1,500
t	nos2.Rows.Content.subinterfaces.subinterface.state.oper-status	UP
t	nos2.Rows.Content.subinterfaces.subinterface.state.type	iana-if-type:softwareLoopback
t	nos2.Rows.Keys.name	Loopback0
#	nos2.Rows.Timestamp	1,510,115,708,482
t	nos2.Source	10.44.160.206:10000
#	nos2.Telemetry.collection_end_time	1,510,115,708,520
#	nos2.Telemetry.collection_id	202,671
#	nos2.Telemetry.collection_start_time	1,510,115,708,438
t	nos2.Telemetry.encoding_path	openconfig-interfaces:interfaces/interface
#	nos2.Telemetry.msg_timestamp	1,510,115,708,438
t	nos2.Telemetry.node_id_str	XRv-Telemetry
t	nos2.Telemetry.subscription_id_str	Sub7
#	nos3.Rows.Content.subinterfaces.subinterface.index	0
t	nos3.Rows.Content.subinterfaces.subinterface.state.admin-status	UP
#	nos3.Rows.Content.subinterfaces.subinterface.state.counters.in-broadcast-pkts	17,950,270
#	nos3.Rows.Content.subinterfaces.subinterface.state.counters.in-discards	47,982
#	nos3.Rows.Content.subinterfaces.subinterface.state.counters.in-errors	0
#	nos3.Rows.Content.subinterfaces.subinterface.state.counters.in-multicast-pkts	32,934
#	nos3.Rows.Content.subinterfaces.subinterface.state.counters.in-octets	1,130,555,692

1つのメッセージに複数JSONが内包されていたが、すべて独立したKey / Valueに分割された

データ変換方法論1：Logstash Filter – 課題点

課題点
基データに複数の同一要素（Key）が存在していたため、各 Key の先頭に識別子（nos1, nos2, nos3）を追加する必要があった。





nos1.Rows.Content.subinterface.subinterface.index
nos1.Rows.Content.subinterface.subinterface.state.admin-status
nos1.Rows.Content.subinterface.subinterface.state.counters.in-broadcast-pkts
<snip>

nos2.Rows.Content.subinterface.subinterface.index
nos2.Rows.Content.subinterface.subinterface.state.admin-status
nos2.Rows.Content.subinterface.subinterface.state.counters.in-broadcast-pkts
<snip>

データ変換方法論1：Logstash Filter – 課題点

本来1つのデータ、例えば Router A の Interface eth0 の Octet カウンターを参照したい時、「Router ID」と「Interface name」と「Input Octet Counter」を識別子として該当データメッセージを検索する事がシンプルな方法。
この方法論は、1つのメッセージ内に「Interface name」等の識別子が重複しない前提の基で動作する。

基JSONメッセージ

Message 1

Router ID : Host A
Interface name : eth0
Input Octet Counter : 1024000
Output Octet Counter : 1000000
・
・

Message 2

Router Host ID : Host A
Interface name : eth1
Input Octet Counter : 2024000
Output Octet Counter : 2000000
・
・

Kibana表示

Key	Value
Router ID	Host A
Interface name	eth0
Input Octet Counter	1024000
Output Octet Counter	1000000

Key	Value
Router ID	Host A
Interface name	eth1
Input Octet Counter	2024000
Output Octet Counter	2000000

データ変換方法論1：Logstash Filter – 課題点

Logstash filter活用の例では、1つのメッセージ内に複数の「Interface name」や「Input Octet Counter」等が同じ Key として存在しており、故に、Logstash Filterで各 Key の先頭に識別子を入れる必要があった。
その為、同様に Router A の Interface eth0 の Input Octet を参照する際に、
①eth0 がどの nosx.interface_name として存在するかを確認し、②nosx.input_octet_counter を参照する、のような煩雑な作業となってしまう。

基JSONメッセージ

Message 1

Router ID : Host A
Interface name : eth0
Input Octet Counter : 1024000
Output Octet Counter : 1000000
・
・
・
Interface name : eth1
Input Octet Counter : 2024000
Output Octet Counter : 2000000
・
・
・

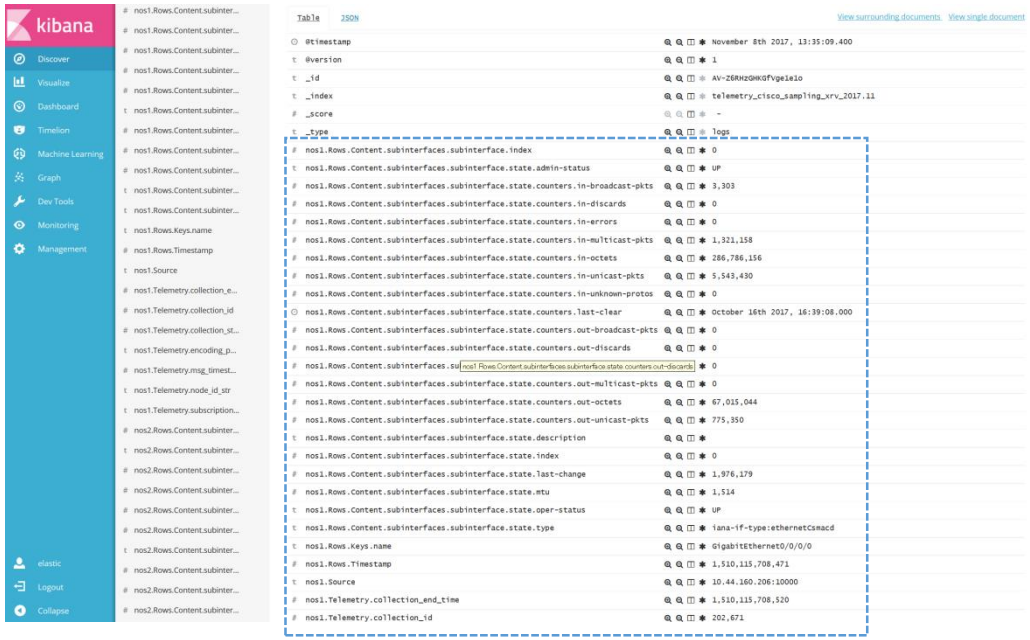
JSON Filterにて
target => “nosx”
の指定が、
重複数必要

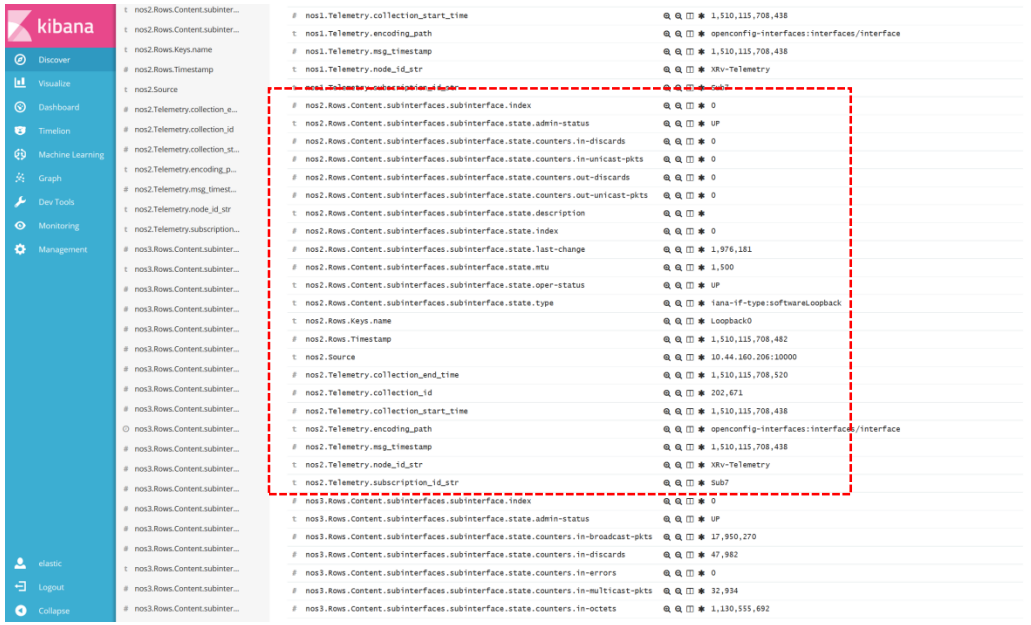
Kibana表示

Key	Value
Router ID	Host A
nos1.Interface name	eth0
nos1.Input Octet Counter	1024000
nos1.Output Octet Counter	1000000
・	
・	
・	
nos2.Interface name	eth1
nos2.Input Octet Counter	2024000
nos2.Output Octet Counter	2000000
・	
・	
・	

データ変換方法論1：Logstash Filter – 課題点（再掲）

課題点
基データに複数の同一要素（Key）が存在していたため、各 Key の先頭に識別子（nos1, nos2, nos3）を追加する必要があった。





nos1.Rows.Content.subinterface.subinterface.index
nos1.Rows.Content.subinterface.subinterface.state.admin-status
nos1.Rows.Content.subinterface.subinterface.state.counters.in-broadcast-pkts
<snip>

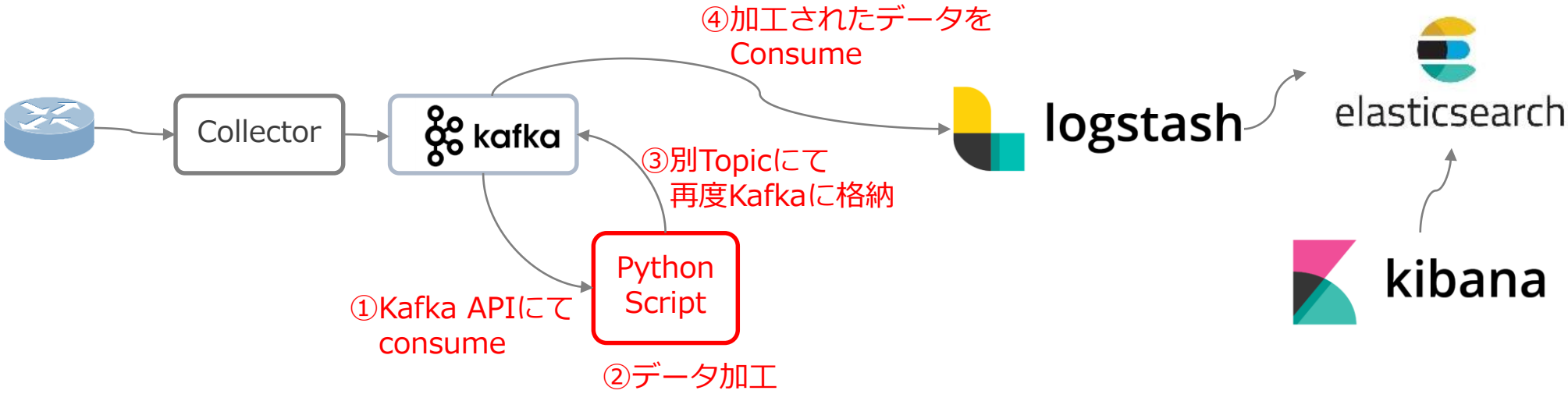
nos2.Rows.Content.subinterface.subinterface.index
nos2.Rows.Content.subinterface.subinterface.state.admin-status
nos2.Rows.Content.subinterface.subinterface.state.counters.in-broadcast-pkts
<snip>

データ変換方法論2：Python Script による一括変換

データ変換方法論2：Python Script にて

加工データ変換方法論2：Python Script による一括変換

先に紹介したLogstash Filterによるデータ可能手法では幾つか課題が残ってしまったため、Netone Telemetry PoCでは、下記の様な、Python Script を用いた一括データ変換の手法を採用した。



データ変換方法論2：Python Script にて

加工データ変換方法論2：Python Script による一括変換例

```
system_id: "Telemetry-vMX-17.2-001"
path:
"sensor_1000_1_1:/junos/system/linecard/interface/:/junos/system/linecard/int
erface/:PFE"
timestamp: 1499160812039
kv {
  key: "__timestamp__"
  uint_value: 1499160811706
}
kv {
  key: "__prefix__"
  str_value: "/interfaces/interface[name=¥'ge-0/0/0¥']/"
}
kv {
  key: "init_time"
  int_value: 1499136891
}
kv {
  key: "parent_ae_name"
  str_value: ""
}
kv {
  key: "oper-status"
  str_value: "UP"
}
kv {
  key: "carrier-transitions"
  int_value: 1
}
kv {
  key: "high-speed"
  int_value: 1000
}
kv {
  key: "counters/out-octets"
  int_value: 44656006
}
```

データ構造の変更(JSON化)
KafkaへのExport

```
{{
  "_index": "telemetry_juniper_sampling_vmx_2017.09",
  "_type": "logs",
  "_id": "AV6emC5yzJ1xIlaMhx4z",
  "_version": 1,
  "_score": null,
  "_source": {
    "carrier-transitions": 1,
    "component_id": 0,
    "juniper_timestamp": 1505899017500,
    "system_id": "vMX1",
    "oper-status": "UP",
    "high-speed": 1000,
    "sequence_number": 2,
    "out-octets": 5171059821,
    "path":
"sensor_1000_1_1:/junos/system/linecard/interface/:/junos/system/linecard/interface/:
PFE",
    "in-octets": 7456051690,
    "@timestamp": "2017-09-20T09:22:07.082Z",
    "in-multicast-pkts": 1715170,
    "@version": "1",
    "init_time": 1500620046,
    "parent_ae_name": "",
    "in-unicast-pkts": 112921259,
    "timestamp": 1505899017592,
    "juniper_prefix": "/interfaces/interface[name='ge-0/0/0']/"
  },
  "fields": {
    "@timestamp": [
      1505899327082
    ]
  },
  "sort": [
    1505899327082
  ]
}
```

データ変換方法論2：Python Script にて

加工データ変換方法論2：Python Script による一括変換例


```
{
  "dataset": "10.44.101.81",
  "timestamp": 1505895992822,
  "update": {
    "Sysdb": {
      "interface": {
        "counter": {
          "eth": {
            "slice": {
              "phy": {
                "1": {
                  "intfCounterDir": {
                    "Ethernet2": {
                      "intfCounter": {
                        "current": {
                          "rates": {
                            "inBitsRate": 7.852205286330623,
                            "inPktsRate": 0.013837285112298117,
                            "outBitsRate": 40.21825379454229,
                            "outPktsRate": 0.03351718607185202,
                            "statsUpdateTime": 88034.796056872
                          },
                          "statistics": {
                            "inBroadcastPkts": 0,
                            "inDiscards": 0,
                            "inErrors": 0,
                            "inMulticastPkts": 0,
                            "inOctets": 207926,
                            "inUcastPkts": 2947,
                            "lastUpdate": 88034.797429813,
                            "outBroadcastPkts": 0,
                            "outDiscards": 0,
                            "outErrors": 0,
                            "outMulticastPkts": 0,
                            "outOctets": 809057,
                            "outUcastPkts": 5880
                          }

    <Snip>
}
```


データ構造の変更
(Indexの変数値対応)


```
{
  "dataset": "10.44.101.81",
  "newkeyword": "Ethernet2",
  "timestamp": 1505895992822,
  "update": {
    "Sysdb": {
      "interface": {
        "counter": {
          "eth": {
            "slice": {
              "phy": {
                "1": {
                  "intfCounterDir": {
                    "Ethernet": {
                      "intfCounter": {
                        "current": {
                          "rates": {
                            "inBitsRate": 7.852205286330623,
                            "inPktsRate": 0.013837285112298117,
                            "outBitsRate": 40.21825379454229,
                            "outPktsRate": 0.03351718607185202,
                            "statsUpdateTime": 88034.796056872
                          },
                          "statistics": {
                            "inBroadcastPkts": 0,
                            "inDiscards": 0,
                            "inErrors": 0,
                            "inMulticastPkts": 0,
                            "inOctets": 207926,
                            "inUcastPkts": 2947,
                            "lastUpdate": 88034.797429813,
                            "outBroadcastPkts": 0,
                            "outDiscards": 0,
                            "outErrors": 0,
                            "outMulticastPkts": 0,
                            "outOctets": 809057,
                            "outUcastPkts": 5880
                          }

    <Snip>
}
```

データ変換方法論2：Python Script にて

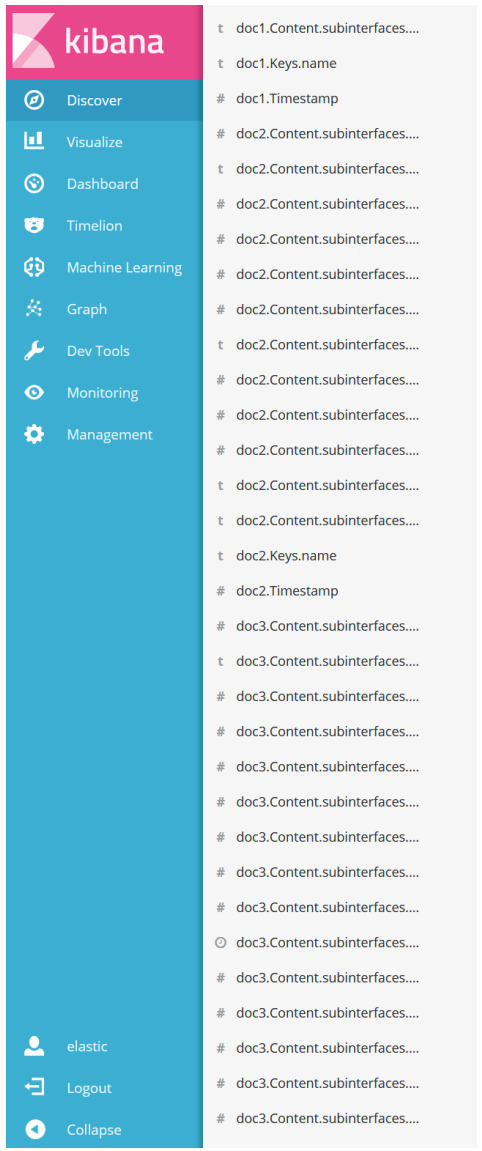


Table JSON

@timestamp	November 8th 2017, 14:59:58.389
@version	1
Rows.Content.subinterfaces.subinterface.index	0
Rows.Content.subinterfaces.subinterface.state.admin-status	UP
Rows.Content.subinterfaces.subinterface.state.counters.in-broadcast-pkts	17,963,315
Rows.Content.subinterfaces.subinterface.state.counters.in-discards	48,112
Rows.Content.subinterfaces.subinterface.state.counters.in-errors	0
Rows.Content.subinterfaces.subinterface.state.counters.in-multicast-pkts	33,019
Rows.Content.subinterfaces.subinterface.state.counters.in-octets	1,131,615,215
Rows.Content.subinterfaces.subinterface.state.counters.in-unicast-pkts	36,584,022
Rows.Content.subinterfaces.subinterface.state.counters.in-unknown-protos	0
Rows.Content.subinterfaces.subinterface.state.counters.last-clear	October 16th 2017, 16:36:34.000
Rows.Content.subinterfaces.subinterface.state.counters.out-broadcast-pkts	0
Rows.Content.subinterfaces.subinterface.state.counters.out-discards	0
Rows.Content.subinterfaces.subinterface.state.counters.out-errors	0
Rows.Content.subinterfaces.subinterface.state.counters.out-multicast-pkts	0
Rows.Content.subinterfaces.subinterface.state.counters.out-octets	365,915,083
Rows.Content.subinterfaces.subinterface.state.counters.out-unicast-pkts	750,148
Rows.Content.subinterfaces.subinterface.state.description	
Rows.Content.subinterfaces.subinterface.state.index	0
Rows.Content.subinterfaces.subinterface.state.last-change	1,981,090
Rows.Content.subinterfaces.subinterface.state.mtu	1,514
Rows.Content.subinterfaces.subinterface.state.oper-status	UP
Rows.Content.subinterfaces.subinterface.state.type	iana-if-type:ethernetCsmacd
Rows.Keys.name	MgmtEth0/RP0/CPU0/0
Rows.Timestamp	1,510,120,797,266
Source	10.44.160.206:10000
Telemetry.collection_end_time	1,510,120,797,290
Telemetry.collection_id	204,657
Telemetry.collection_start_time	1,510,120,797,214
Telemetry.encoding_path	openconfig-interfaces:interfaces/interface
Telemetry.msg_timestamp	1,510,120,797,214
Telemetry.node_id_str	XRv-Telemetry
Telemetry.subscription_id_str	Sub7
_id	AV-aNrjJGHKGfVge1kYs

データ変換方法論2：Python Script にて

Table JSON

View surrounding documents View single document

@timestamp	November 8th 2017, 14:59:58.386
@version	1
Rows.Content.subinterfaces.subinterface.index	0
Rows.Content.subinterfaces.subinterface.state.admin-status	UP
Rows.Content.subinterfaces.subinterface.state.counters.in-broadcast-pkts	3,311
Rows.Content.subinterfaces.subinterface.state.counters.in-discards	0
Rows.Content.subinterfaces.subinterface.state.counters.in-errors	0
Rows.Content.subinterfaces.subinterface.state.counters.in-multicast-pkts	1,324,560
Rows.Content.subinterfaces.subinterface.state.counters.in-octets	287,522,019
Rows.Content.subinterfaces.subinterface.state.counters.in-unicast-pkts	5,557,674
Rows.Content.subinterfaces.subinterface.state.counters.in-unknown-protos	0
Rows.Content.subinterfaces.subinterface.state.counters.last-clear	October 16th 2017, 16:39:08.000
Rows.Content.subinterfaces.subinterface.state.counters.out-broadcast-pkts	0
Rows.Content.subinterfaces.subinterface.state.counters.out-discards	0
Rows.Content.subinterfaces.subinterface.state.counters.out-errors	0
Rows.Content.subinterfaces.subinterface.state.counters.out-multicast-pkts	0
Rows.Content.subinterfaces.subinterface.state.counters.out-octets	67,183,750
Rows.Content.subinterfaces.subinterface.state.counters.out-unicast-pkts	777,294
Rows.Content.subinterfaces.subinterface.state.description	
Rows.Content.subinterfaces.subinterface.state.index	0
Rows.Content.subinterfaces.subinterface.state.last-change	1,981,268
Rows.Content.subinterfaces.subinterface.state.mtu	1,514
Rows.Content.subinterfaces.subinterface.state.oper-status	UP
Rows.Content.subinterfaces.subinterface.state.type	iana-if-type:ethernetCsmacd
Rows.Keys.name	GigabitEthernet0/0/0/0
Rows.Timestamp	1,510,120,797,249
Source	10.44.160.206:10000
Telemetry.collection_end_time	1,510,120,797,290
Telemetry.collection_id	204,657
Telemetry.collection_start_time	1,510,120,797,214
Telemetry.encoding_path	openconfig-interfaces:interfaces/interface
Telemetry.msg_timestamp	1,510,120,797,214
Telemetry.node_id_str	XRv-Telemetry
Telemetry.subscription_id_str	Sub7
_id	AV-aNrjJGHKGfVge1kYq