

# 第二回 JANOG Telemetry WG

～OSSツールで作る、Telemetry初めの一歩～

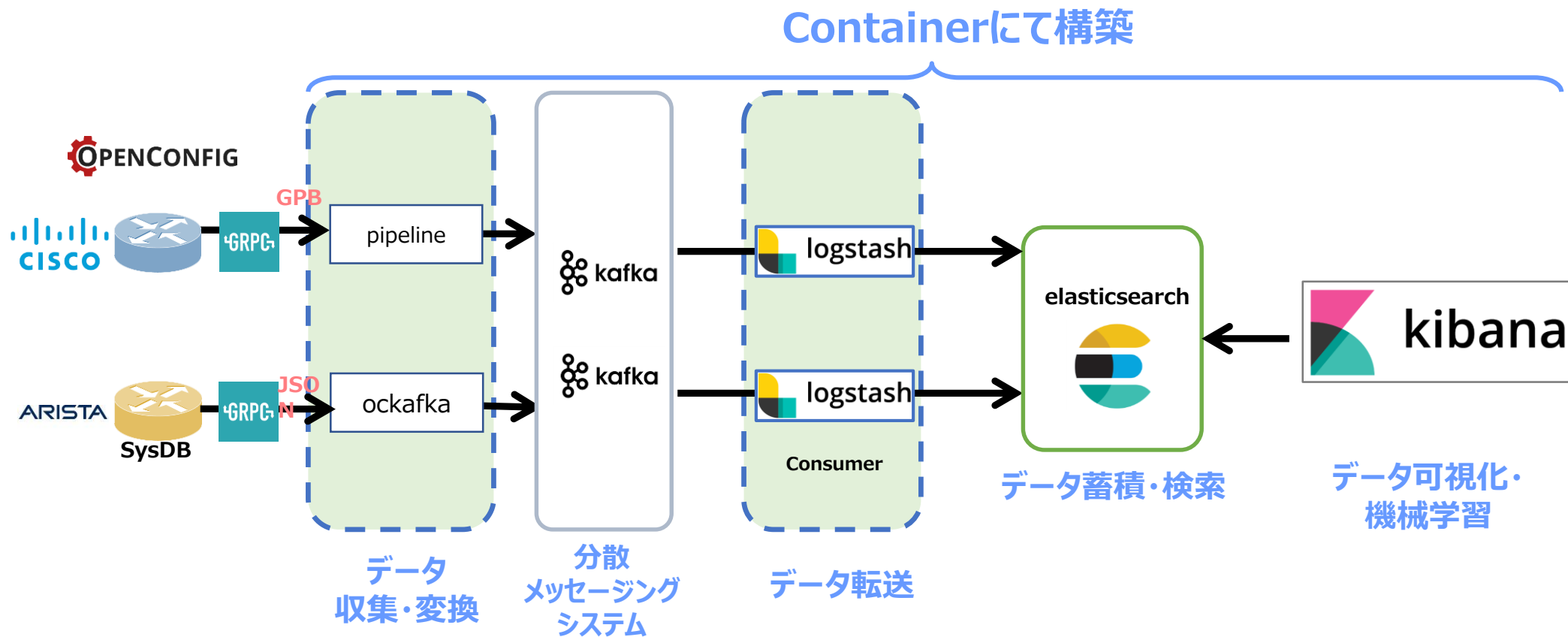
ネットワークシステムズ

## お願いと注意事項

1. 自己責任にてお願いします。  
万が一、本資料を基に実施した事柄により、何かしらの不利益が生じたとしても、情報提供者は責任を負いません。
2. 再配布ポリシー  
本WG内での利用とさせていただきます。
3. Feedback歓迎  
気づき、ツツコミ、、etc…

# 今日のGoalイメージ

Ciscoさま、Aristaさまが各々公開するCollectorツールを使い、RouterよりTelemetry Dataを収集する。そして、収集したデータを可視化する。



# 仮想Routerについて

## Cisco IOS XRv

- <http://bef-s-anne.hatenablog.com/entry/2016/12/14/233728> 入手からインストールまで書かれている
  - 概要のところでも、Telemetry の動作確認が動機の様
  - デモ版は200 kbps のレート制限。機能制限、利用期間の制限なし

## Juniper vMX

- <https://www.juniper.net/jp/jp/products-services/routing/mx-series/vmx/>
  - 会社のメールアカウントのみ有効（Gmail アカウントで苦労した方のブログ <https://www.ainoniwa.net/pelican/2016/0731a.html>）
  - デモ版は60日限定で利用可能。機能制限、帯域制限なし

## Arista vEOS-lab

- <https://qiita.com/souko2525/items/2e1478be0441f4c057e4> 入手からインストールまで書かれている
  - 制限なし（?）、ラボ利用限定

# ルータの設定 – Cisco(GRPC / OpenConfig/ GPB)

Cisco IOS XRv

!

grpc

port 10000

gRPCポートの指定

!

telemetry model-driven

sensor-group SGroup1

sensor-path openconfig-interfaces:interfaces/interface

送信するデータ

!

subscription Sub1

sensor-group-id SGroup1 sample-interval 30000

送信する周期(ミリ秒)

!

!

# 【参考】ルータの設定 – Juniper(GRPC / OpenConfig/ GPB)

## •OpenConfigパッケージのインストール

OpenConfigとJunos OSのコンフィグの変換  
OpenConfigをYANGデータモデルに設定

```
admin@vMX1> show version | match openconfig  
JUNOS Openconfig [0.0.0.3]
```

## •NetworkAgent パッケージのインストール

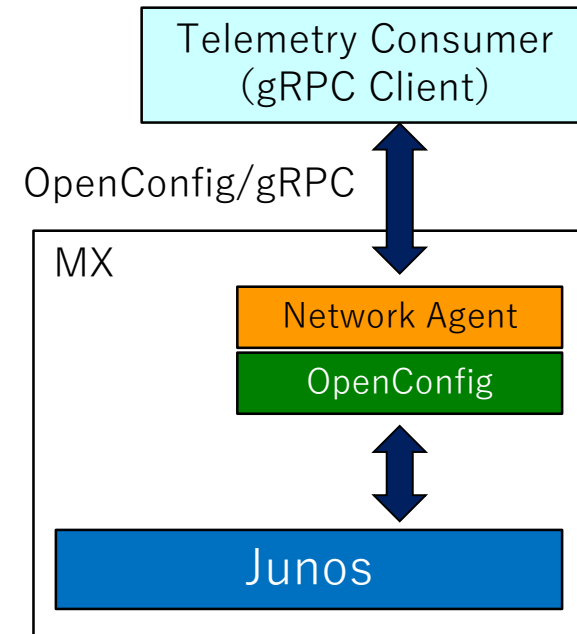
gRPCの終端、Telemetryデータの送信

```
admin@vMX1> show version | match na¥ telemetry  
JUNOS na telemetry [17.2R1.13-C1]
```

## •コンフィグレーション

以下のコンフィグを適用することで、gRPCの疎通が可能となる

```
# set system services extension-service request-response grpc clear-text
```



現 PoC では  
• Compact GPB にて送出

# ルータの設定 – Arista(GRPC /独自モデル/ JSON)

## •エージェントソフト(TerminAttr)のインストール

EOSからSysdbの収集

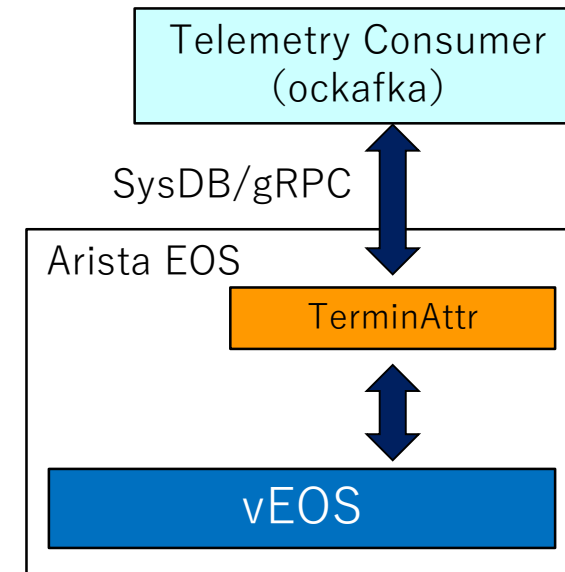
gRPCの終端、Telemetryデータの送信

```
[admin@vEOS2 ~]$ /mnt/flash/TerminAttr --version  
v0.18.0 go1.7.3
```

## •EOSコンフィグレーション

以下のコンフィグでTerminAttrのデーモンを起動させる

```
daemon TerminAttr  
exec /mnt/flash/TerminAttr -disableaaa -grpcaddr 10.44.101.81:6042  
no shutdown
```



# 環境準備（前提条件）

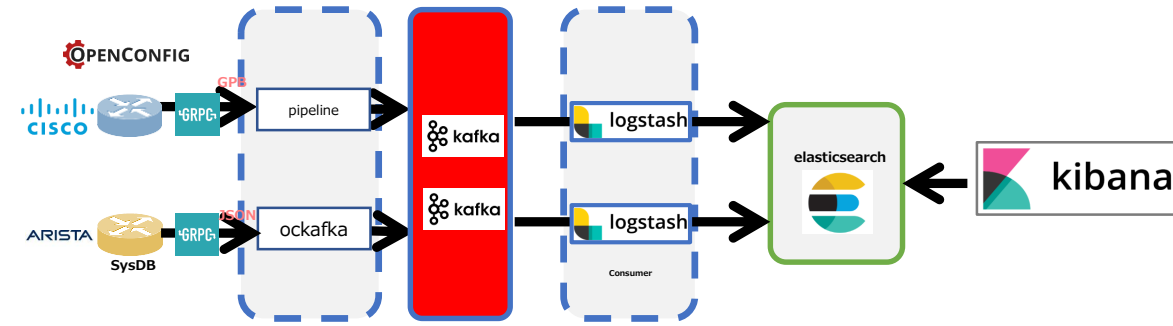
CentOS Linux release 7.5.1804 (Core)

Docker install, selinux off

- ① `systemctl stop firewalld.service`
- ② `yum install -y python-pip python-devel git epel-release`
- ③ `yum install -y docker`
- ④ `systemctl mask firewalld.service`
- ⑤ `systemctl enable docker.service`
- ⑥ `setenforce 0` (vi /etc/selinux/config -> SELINUX=disabled)
- ⑦ 再起動
- ⑧ `systemctl start docker.service`
- ⑨ `pip install docker-compose`
- ⑩ `yum install -y unzip wget curl jq coreutils`



# Kafka : Container作成方法



① KafkaをGitより取得する

```
git clone https://github.com/wurstmeister/kafka-docker.git  
cd kafka-docker/
```

② docker-compose.ymlの編集

③ 起動

```
docker-compose up もしくは docker-compose up -d
```

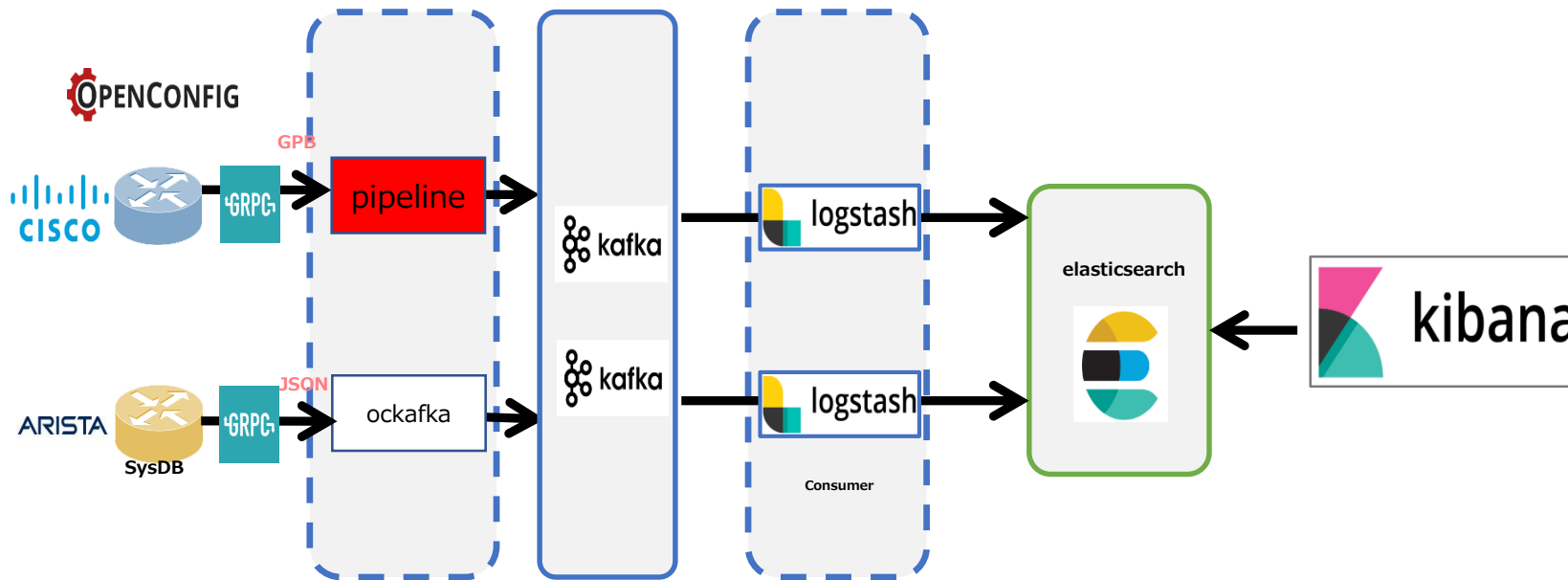
↑ 初回は少し時間かかったり

↑ 原因不明のエラーで悩ませられたり

```
version: '2'  
services:  
  zookeeper:  
    image: wurstmeister/zookeeper  
    ports:  
      - "2181:2181"  
  kafka:  
    build: . 追記  
    ports:  
      - "32768:9092"  ここを実アドレスへ変更  
    environment:  
      KAFKA_ADVERTISED_HOST_NAME: 10.44.160.97  
      KAFKA_ZOOKEEPER_CONNECT: zookeeper:2181  
    volumes:  
      - /var/run/docker.sock:/var/run/docker.sock
```

# cisco/bigmuddy-network-telemetry-pipeline

- Gitで公開されているFreeのOpenConfig / gRPC対応型Telemetry Collector  
(<https://github.com/cisco/bigmuddy-network-telemetry-pipeline>)
- Telemetry DataをGPBデシアライズし、人が読めるテキスト形式でシステムに出力する
- これから作るもの



# cisco/bigmuddy-network-telemetry-pipeline

## Collectorの設定と起動

### 1. Docker取得

```
# docker pull janogtelemetryworkinggroup/bigmuddy-network-telemetry-pipeline:001
```

### 2. Container起動

```
# docker run -it docker.io/janogtelemetryworkinggroup/bigmuddy-network-telemetry-pipeline:001
```

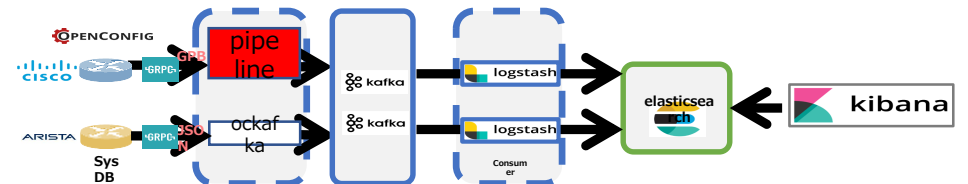
### 3. Configurationの変更

- RouterのIP Address/Port  
# sed -i -e s/"server = 10.44.101.71:10000"/"server = 1.2.3.4:10000"/ /data/pipeline.conf
- KafkaのIP Address/Port  
# sed -i -e s/"brokers = 10.44.160.98:32768"/"brokers = 5.6.7.8:32768"/ /data/pipeline.conf
- Routerのセンサーパス情報  
# sed -i -e s/"subscriptions = Sub1"/"subscriptions = Subx"/ /data/pipeline.conf

### 4. Telemetry Collector起動

```
# /pipeline -log=/data/pipeline.log -config=/data/pipeline.conf
```

※起動後、User/Passwordを入力します



# cisco/bigmuddy-network-telemetry-pipeline

## 起動後の確認（内部のDumpファイル）

1. （もう1つTerminalを立ち上げて、） Container ID確認

```
# docker ps | grep bigmu
```

```
4fd5f011cc8f      docker.io/janogtelemetryworkinggroup/bigmuddy-network-telemetry-  
pipeline:001     "/bin/sh -c /bin/bash"   13 minutes ago      Up 12 minutes  
optimistic_Jennings
```

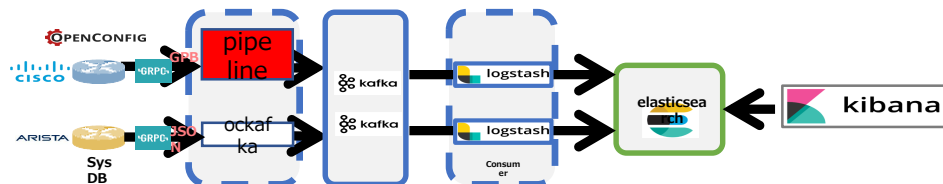
2. Containerへ接続

```
# docker exec -it 4fd5f011cc8f /bin/bash
```

3. 取得Telemetry Dataの確認

```
# tail -f /data/dump.txt
```

```
root@694e3f2135fa:/# tail -f /data/dump.txt  
----- 2018-06-27 03:34:57.831927049 +0000 UTC -----  
Summary: GPB(common) Message [10.44.101.71:10000(XRv1)/openconfig-  
interfaces:interfaces/interface msg len: 5821]  
{  
  "Source": "10.44.101.71:10000",  
  "Telemetry": {  
    "node_id_str": "XRv1",  
    "subscription_id_str": "Sub1",  
    "encoding_path": "openconfig-interfaces:interfaces/interface",  
    "collection_id": 26324,
```



# cisco/bigmuddy-network-telemetry-pipeline

## 起動後の確認 (Kafkaにて)

1. (もう1つTerminalを立ち上げて、) Container ID確認

```
# docker ps | grep kafka
24b9252916a8 kafkadocker_kafka "start-
kafka.sh" 46 hours ago Up 46 hours 0.0.0.0:32768->9092/tcp
kafkadocker_kafka
```

2. Containerへ接続

```
# docker exec -it 24b9252916a8 /bin/bash
```

3. 取得Telemetry Dataの確認

```
# kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic
telemetry_cisco_xrv
```

```
bash-4.4# kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic telemetry_cisco_xrv
[{"Source":"10.44.101.71:10000","Telemetry":{"node_id_str":"XRv1","subscription_id_str":"Sub1","encoding_path":"openconfig-
interfaces:interfaces/interface","collection_id":26396,"collection_start_time":1530080102572,"msg_timestamp":1530080102572,"collection_end_time":15
30080102634},"Rows":{"Timestamp":1530080102584,"Keys":{"name":"GigabitEthernet0/0/0/0"},"Content":{"subinterfaces":{"subinterface":{"index":0,"st
ate":{"admin-status":"DOWN","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":"2018-02-20T05:21:44Z","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":1530080102,"mtu":1514,"oper-status":"DOWN","type":"iana-if-
type:ethernetCsmacd"}}}}}}}]
```

# cisco/bigmuddy-network-telemetry-pipeline

## 【参考】 Default Configuration

```
[default]
id = pipeline

[mymdtrouter]
stage = xport_input
type = grpc
encoding = gpbkv
encap = gpb

## configure it according to your environments
server = 10.44.101.71:10000

subscriptions = Sub1
tls = false

[inspector]
stage = xport_output
type = tap
file = /data/dump.txt
datachanneldepth = 1000
```

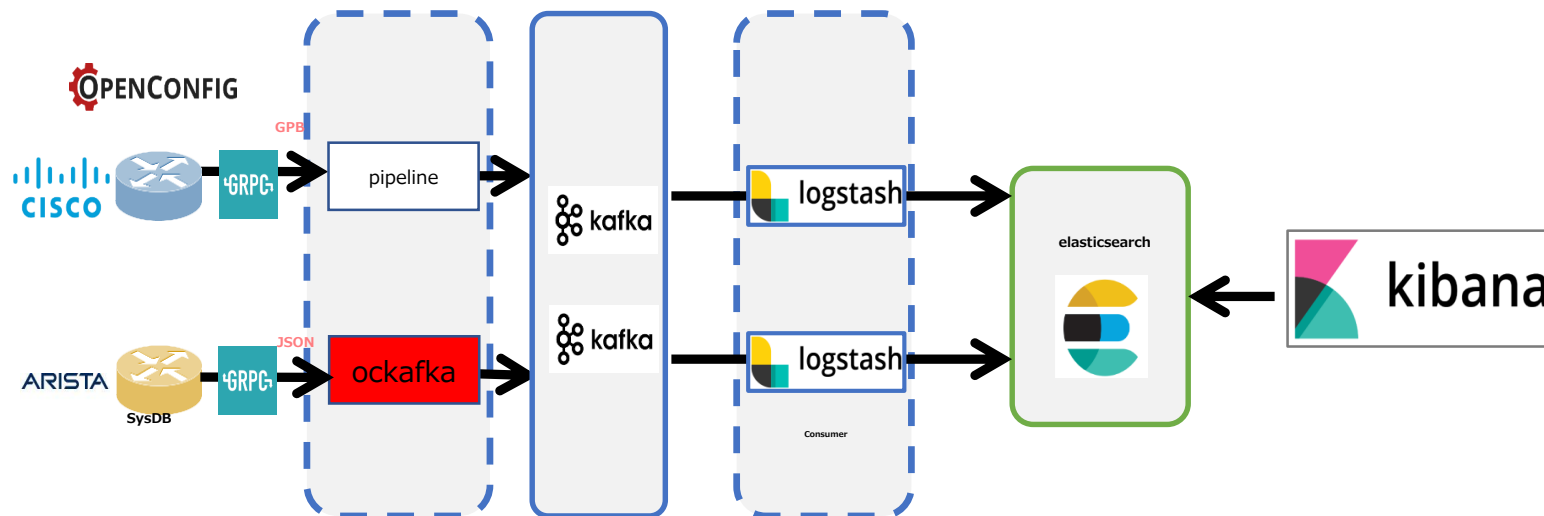
```
[mykafka]
stage = xport_output
type = kafka
encoding = json_events

## configure it according to your environments
brokers = 10.44.160.98:32768

topic = telemetry_cisco_xrv
```

# aristanetworks/goarista

- Gitで公開されているFreeのOpenConfig / gRPC対応型Telemetry Collector  
(<https://github.com/aristanetworks/goarista>)
- JSON 形式で、Telemetry Dataシステムに出力する
- これから作るもの  
※本日はOpenconfigでは無くベンダーNativeにて構築



# aristanetworks/goarista

## Collectorの設定と起動

### 1. Docker取得

```
# docker pull janogtelemetryworkinggroup/ockafka:latest
```

### 2. Container起動

```
# docker run -p 45968:45968 docker.io/janogtelemetryworkinggroup/ockafka:latest --  
kafkatopic telemetry_arista_ockafka -addr 10.44.101.81 -kafkaaddr  
10.44.160.189:32768 -subscribe
```

```
/Sysdb/sys/net/config,/Sysdb/inteace/counter/eth/slice/phy/1/intfCounterDir/Ethernet1  
/intfCounter/current,/Sysdb/sys/net/config,/Sysdb/interface/counter/eth/slice/phy/1/int  
fCounterDir/Ethernet2/intfCounter/current,/Sysdb/sys/net/config,/Sysdb/interface/coun  
ter/eth/slice/phy/1/intfCounterDir/Ethernet3/intfCounter/current
```

KafkaのAddress/port

KafkaのTopic

RouterのIP Address/Port

センサーパス



# aristanetworks/goarista

## 起動後の確認 (Kafkaにて)

1. (もう1つTerminalを立ち上げて、) Container ID確認

```
# docker ps | grep kafka
24b9252916a8      kafkadocker_kafka      "start-
kafka.sh"        46 hours ago          Up 46 hours          0.0.0.0:32768->9092/tcp
kafkadocker_kafka
```

2. Containerへ接続

```
# docker exec -it 24b9252916a8 /bin/bash
```

3. 取得Telemetry Dataの確認

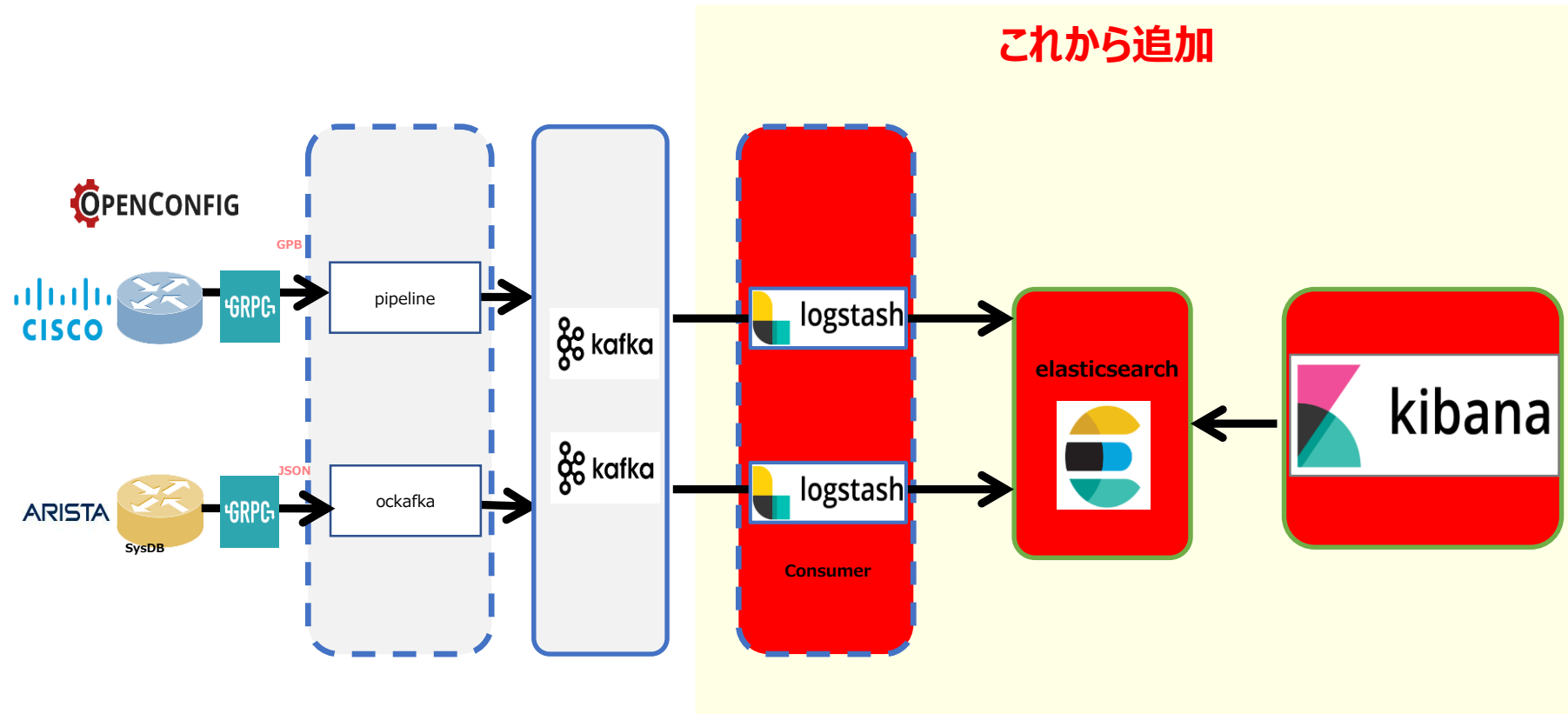
```
# kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic
telemetry_arista_ockafka --from-beginning
```

```
bash-4.4# kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic telemetry_arista_ockafka --from-beginning
{"dataset":"10.44.101.81","timestamp":1530085097880,"update":{"Sysdb":{"sys":{"net":{"config":{"domainList":{},"domainListMetadata":{"head":0,"tail":0
},"domainName":"nos.com","hostname":"vEOS2","hostnameTimeout":20,"name":"config","nameServer":{},"sourceIntf":{},"v6NameServer":{},"vrfName":{"
value":"default"}}}}}}}}
```

# Elastic Stack

## これからすること

Kafka に集約したTelemetry DataをElastic Stackへ展開する



# Elasticsearch

## 1) docker-compose.ymlの作成

```
version: '2.0'
services:
  elasticsearch:
    image: docker.elastic.co/elasticsearch/elasticsearch-platinum:6.1.3
    container_name: elasticsearch
    environment:
      - cluster.name=docker-cluster-test01
      - bootstrap.memory_lock=true
      - "ES_JAVA_OPTS=-Xms512m -Xmx512m"
      - ELASTIC_PASSWORD=changeme
    ulimits:
      memlock:
        soft: -1
        hard: -1
    volumes:
      - esdata1:/usr/share/elasticsearch/data
    ports:
      - 9200:9200
    networks:
      - esnet
```

```
elasticsearch2:
  image: docker.elastic.co/elasticsearch/elasticsearch-platinum:6.1.3
  container_name: elasticsearch2
  environment:
    - cluster.name=docker-cluster-test01
    - bootstrap.memory_lock=true
    - "ES_JAVA_OPTS=-Xms512m -Xmx512m"
    - "discovery.zen.ping.unicast.hosts=elasticsearch"
    - ELASTIC_PASSWORD=changeme
  ulimits:
    memlock:
      soft: -1
      hard: -1
  volumes:
    - esdata2:/usr/share/elasticsearch/data
  ports:
    - 9201:9200
  networks:
    - esnet

volumes:
  esdata1:
    driver: local
  esdata2:
    driver: local

networks:
  esnet:
```

# Elasticsearch

2) `$ sudo sysctl -w vm.max_map_count=262144`

Memo :

max virtual memory areas `vm.max_map_count` [65530] likely too low, increase to at least [262144].

BootstrapCheckというもので、設定が推奨値に達していなかったり、変な設定が入っているとエラーになって終了する。

今回は、「`vm.max_map_count`」が足りないと言われているので、変更。

3) `docker-compose up`

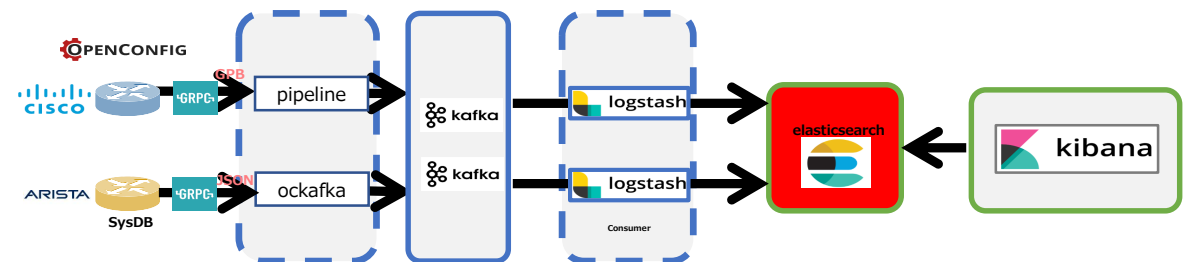
このコマンドで起動

※ `docker-compose.yml` のあるフォルダで実行する事

4) `docker-compose down`

このコマンドでContainerストップ（削除）⇒本デモでは引き続き利用するためそのまま動かす

※ `docker-compose.yml` のあるフォルダで実行する事



# Kibana

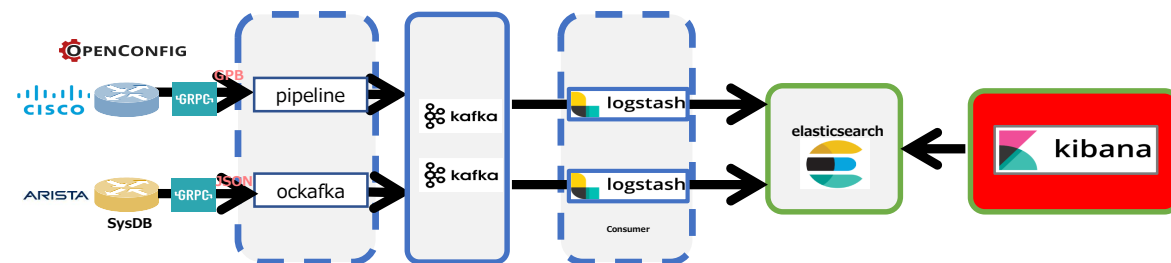
## 1) docker-compose.ymlの作成

```
version: '2'
services:
  kibana:
    image: docker.elastic.co/kibana/kibana:6.1.3
    volumes:
      - ./kibana.yml:/root/kibana_docker/kibana.yml
    ports:
      - 5601:5601
    networks:
      - esnet
    environment:
      SERVER_NAME: kibana.example.org
      ELASTICSEARCH_URL: http://10.44.160.97:9200
    networks:
      esnet:
```

Elastic Searchのアドレスを指定する事

## 2) kibana.ymlの作成

```
elasticsearch.username: "elastic"
elasticsearch.password: "changeme"
```



# Kibana

2) docker-compose up

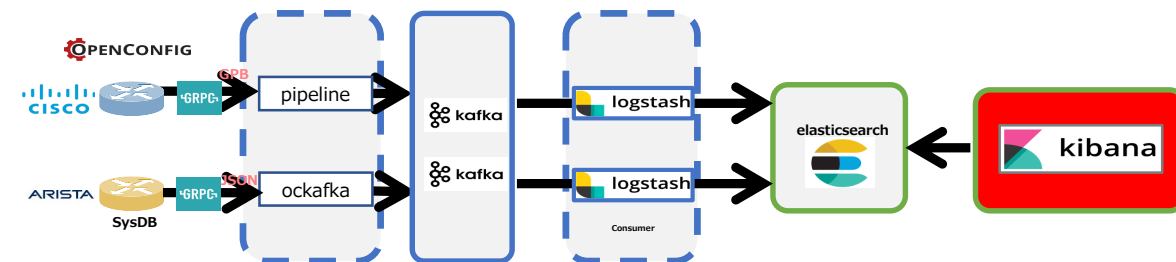
このコマンドで起動

※ docker-compose.yml のあるフォルダで実行する事

3) docker-compose down

このコマンドでContainerストップ（削除）

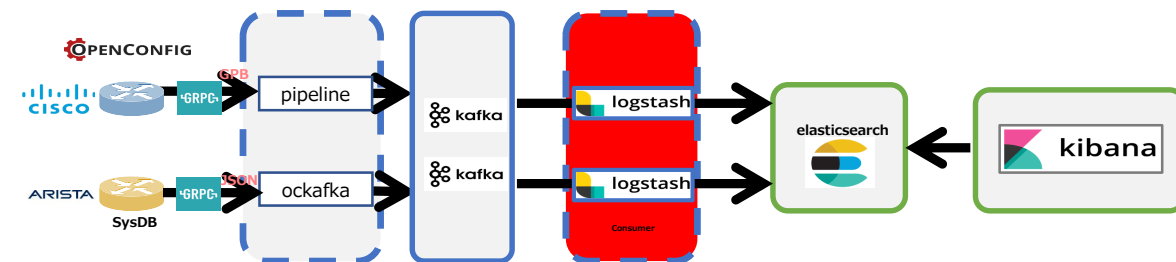
※ docker-compose.yml のあるフォルダで実行する事



# Logstash

本構成では、ベンダー単位で別TopicをKafkaに作成してTelemetry DataをProduceしているため、後段にあるLogstashもベンダー単位でMicro Service化する。（LogstashをCisco/Arista用に2つ起動）

- 1) Container用コンフィグレーション保存フォルダの作成  
mkdir /config-dir



# Logstash : Container作成方法

2) 各Logstash Configurationを作成し、Config-dirに保存。

## Cisco

```
input {
  kafka {
    group_id => "test01"
    bootstrap_servers => "10.44.160.189:32768"
    topics => "telemetry_cisco_xrv"
  }
}

filter {
  date {
    match => [ "timestamp", "UNIX_MS" ]
    remove_field => [ "timestamp" ]
  }
  json {
    source => "message"
    remove_field => "message"
  }
}

output {
  elasticsearch {
    hosts => "10.44.160.189:9200"
    user => elastic
    password => changeme
    index => "telemetry_cisco_sampling_xrv_%{+YYYY.MM}"
  }
}
```

Collectorで設定した  
Topicを指定する

## Arista

```
input {
  kafka {
    group_id => "test01"
    bootstrap_servers => "10.44.160.189:32768"
    topics => "telemetry_arista_ockafka"
  }
}

filter {
  date {
    match => [ "timestamp", "UNIX_MS" ]
    remove_field => [ "timestamp" ]
  }
  json {
    source => "message"
    remove_field => "message"
  }
}

output {
  elasticsearch {
    hosts => "10.44.160.189:9200"
    user => elastic
    password => changeme
    index => "telemetry_arista_sampling_veos_%{+YYYY.MM}"
  }
}
```

Collectorで設定した  
Topicを指定する



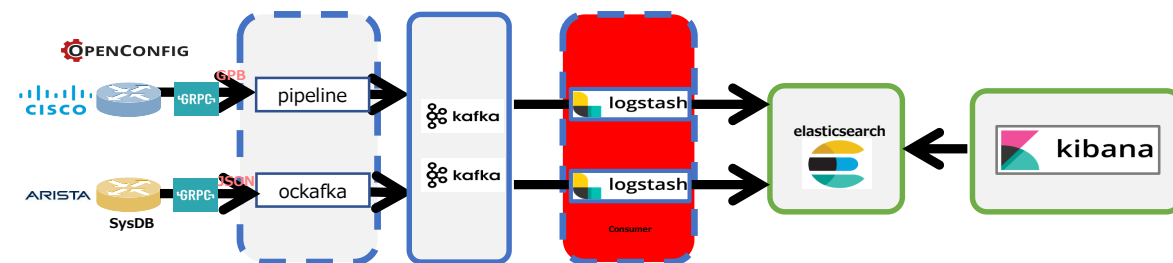
# Logstash : Container作成方法

## 3) Logstash Containerの実行

```
cd /config-dir/
```

```
docker run -d -v "$PWD":/config-dir logstash -f /config-dir/cisco_logstash.conf
```

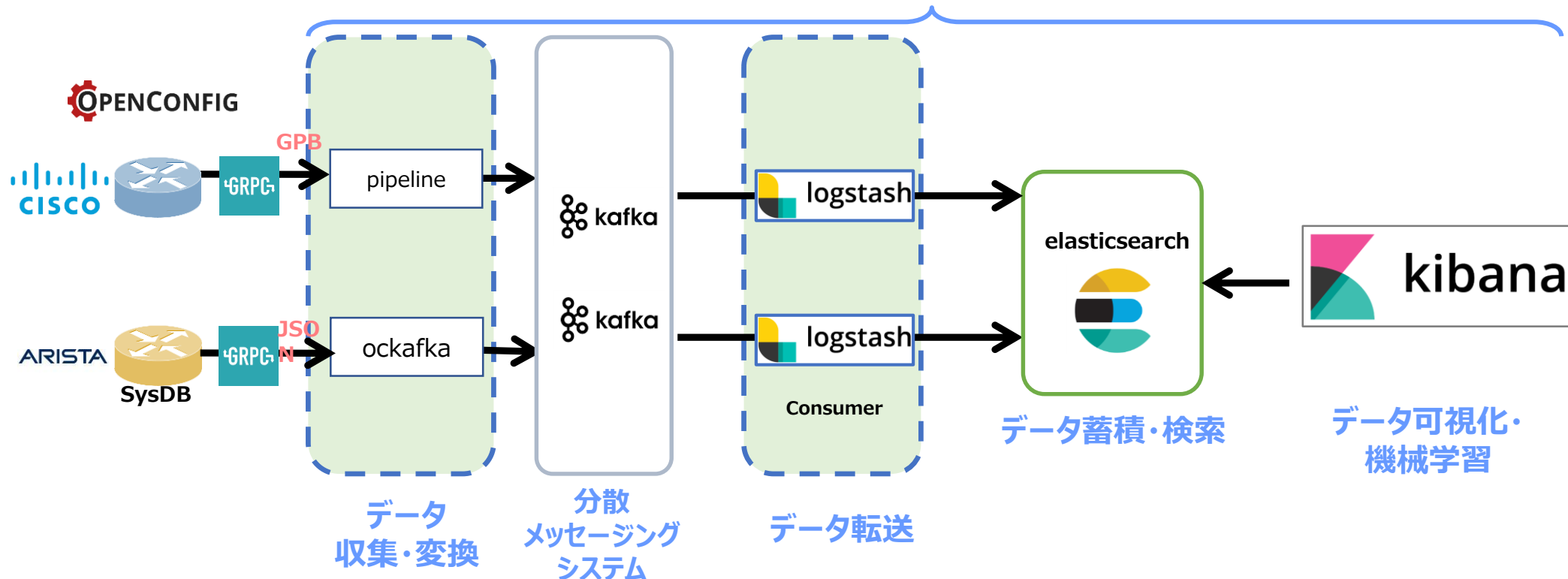
```
docker run -d -v "$PWD":/config-dir logstash -f /config-dir/arista_logstash.conf
```



# コンテナ環境 - 実行状態

```
[root@localhost config-dir]# docker ps
CONTAINER ID   IMAGE                                     COMMAND                  CREATED          STATUS          PORTS                               NAMES
5fbf6e392846   logstash                                  "/docker-entrypoint..." 3 seconds ago    Up 2 seconds    elastic_booth
2afe1eb68345   logstash                                  "/docker-entrypoint..." 5 seconds ago    Up 3 seconds    naughty_mestorf
05832bfe287c   docker.io/janogtelemetryworkinggroup/ockafka:latest "/go/bin/ockafka -..." 6 minutes ago    Up 6 minutes    0.0.0.0:45968->45968/tcp
determined_visvesvaraya
042bb24d888d   docker.io/janogtelemetryworkinggroup/bigmuddy-network-telemetry-pipeline:001 "/bin/sh -c /bin/bash" 12 minutes ago    Up 12 minutes    stupefied_shannon
c7841596549b   docker.elastic.co/elasticsearch/elasticsearch-platinum:6.1.3 "/usr/local/bin/do..." 32 minutes ago    Up 32 minutes    0.0.0.0:9200->9200/tcp, 9300/tcp    elasticsearch
405f52a5432a   docker.elastic.co/elasticsearch/elasticsearch-platinum:6.1.3 "/usr/local/bin/do..." 32 minutes ago    Up 32 minutes    9300/tcp, 0.0.0.0:9201->9200/tcp    elasticsearch2
4dcd838e2810   docker.elastic.co/kibana/kibana:6.1.3   "/bin/bash /usr/lo..." 35 minutes ago    Up 31 minutes    0.0.0.0:5601->5601/tcp             kibana_kibana_1
a5e30855f159   wurstmeister/zookeeper                  "/bin/sh -c '/usr/..." 2 hours ago      Up 2 hours      22/tcp, 2888/tcp, 3888/tcp, 0.0.0.0:2181->2181/tcp kafka-docker_zookeeper_1
ea48f6c92eca   kafka-docker_kafka                       "start-kafka.sh"        2 hours ago      Up 2 hours      0.0.0.0:32768->9092/tcp
```

## Containerにて構築

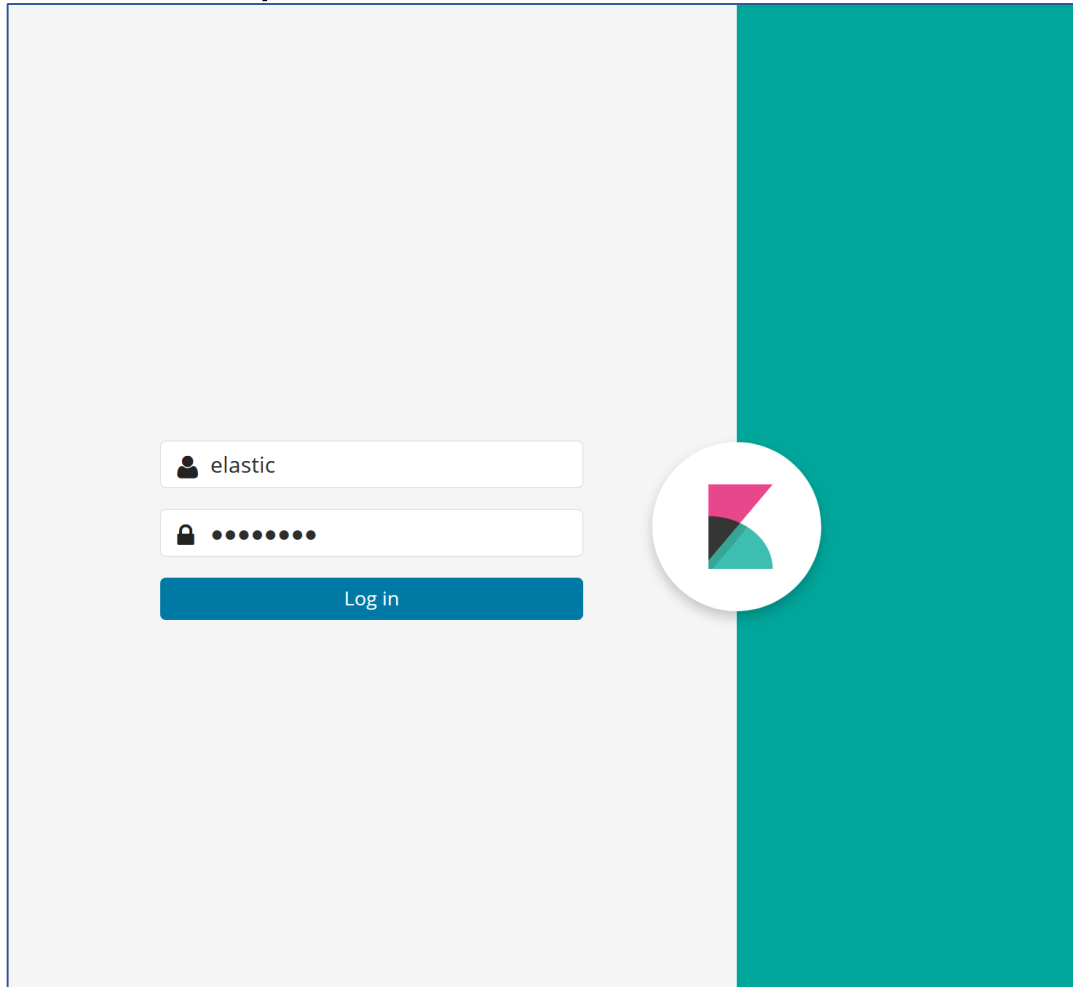


# KibanaからのTelemetry Data確認

1) <http://x.x.x.x:5061> にアクセス

User/Pass = elastic/changeme にてログイン

※最新版では、username/password でログイン可能



# KibanaからのTelemetry Data確認

## 2) Index Patternの作成

### Create index pattern

Kibana uses index patterns to retrieve data from Elasticsearch indices.

Step 1 of 2: Define index pattern

**Index pattern**

You can use a \* as a wildcard in your index pattern.  
You can't use empty spaces or the characters \ / ? " < > |.

✔ **Success!** Your index pattern matches **2 indices**.

- telemetry\_arista\_sampling\_veos\_2018.06
- telemetry\_cisco\_sampling\_xrv\_2018.06

### Create index pattern

Kibana uses index patterns to retrieve data from Elasticsearch indices for things like visualizations.

Step 2 of 2: Configure settings

You've defined **telemetry\*** as your index pattern. Now you can specify some settings before we create it.

**Time Filter field name** [Refresh](#)

The Time Filter will use this field to filter your data by time. You can choose not to have a time field, but you will.

▶ Show advanced options

# KibanaからのTelemetry Data確認

## 2) Index Patternの作成- 続き

★ telemetry\*

🕒 Time Filter field name: @timestamp

This page lists every field in the **telemetry\*** index and the field's associated core type as recorded by Elasticsearch. While this list allows you to view the core type, you can also view the field's details using Elasticsearch's [Mapping API](#)

fields (51)    scripted fields (0)    source filters (0)

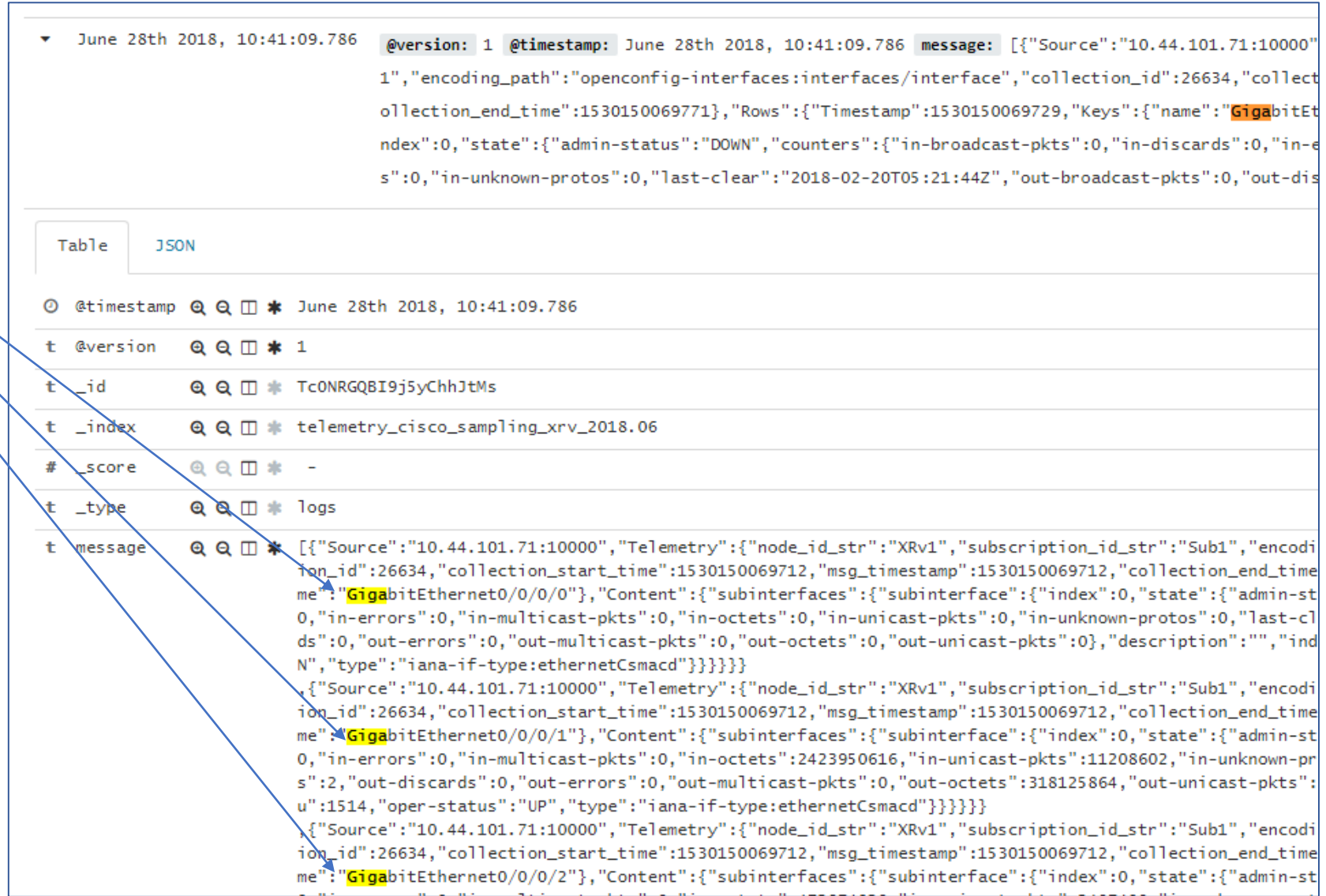
🔍 Filter

name ↕	type ↕	format ↕	searchable
@timestamp 🕒	date		✓
@version	string		✓
@version.keyword	string		✓
_id	string		✓
_index	string		✓
_score	number		
_source	_source		
_type	string		✓
dataset	string		✓
dataset.keyword	string		✓
message	string		✓
message.keyword	string		✓
tags	string		✓

# KibanaからのTelemetry Data確認

## 3) Telemetry Dataの確認

### 3-1) CiscoだとArrayしてしまう



▼ June 28th 2018, 10:41:09.786 @version: 1 @timestamp: June 28th 2018, 10:41:09.786 message: [{"Source":"10.44.101.71:10000", "encoding\_path":"openconfig-interfaces/interfaces/interface", "collection\_id":26634, "collection\_start\_time":1530150069771, "collection\_end\_time":1530150069771}, {"Rows":{"Timestamp":1530150069729, "Keys":{"name":"GigabitEthernet0/0/0/0", "index":0, "state":{"admin-status":"DOWN", "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":"2018-02-20T05:21:44Z", "out-broadcast-pkts":0, "out-discards":0, "out-errors":0, "out-multicast-pkts":0, "out-octets":0, "out-unicast-pkts":0, "type":"iana-if-type:ethernetCsmacd"}}}}}, {"Source":"10.44.101.71:10000", "Telemetry":{"node\_id\_str":"XRv1", "subscription\_id\_str":"Sub1", "encoding\_path":"openconfig-interfaces/interfaces/interface", "collection\_id":26634, "collection\_start\_time":1530150069712, "msg\_timestamp":1530150069712, "collection\_end\_time":1530150069712, "Content":{"subinterfaces":{"subinterface":{"index":0, "state":{"admin-status":"DOWN", "counters":{"in-broadcast-pkts":0, "in-discards":0, "in-errors":0, "in-multicast-pkts":0, "in-octets":2423950616, "in-unicast-pkts":11208602, "in-unknown-protos":2, "out-broadcast-pkts":0, "out-discards":0, "out-errors":0, "out-multicast-pkts":0, "out-octets":318125864, "out-unicast-pkts":1514, "oper-status":"UP", "type":"iana-if-type:ethernetCsmacd"}}}}}}}, {"Source":"10.44.101.71:10000", "Telemetry":{"node\_id\_str":"XRv1", "subscription\_id\_str":"Sub1", "encoding\_path":"openconfig-interfaces/interfaces/interface", "collection\_id":26634, "collection\_start\_time":1530150069712, "msg\_timestamp":1530150069712, "collection\_end\_time":1530150069712, "Content":{"subinterfaces":{"subinterface":{"index":0, "state":{"admin-status":"DOWN", "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":"2018-02-20T05:21:44Z", "out-broadcast-pkts":0, "out-discards":0, "out-errors":0, "out-multicast-pkts":0, "out-octets":0, "out-unicast-pkts":0, "type":"iana-if-type:ethernetCsmacd"}}}}}}}]

Table JSON

@timestamp	June 28th 2018, 10:41:09.786
@version	1
_id	Tc0NRGQBI9j5yChhJtMs
_index	telemetry_cisco_sampling_xrv_2018.06
_score	-
_type	logs
message	[{"Source":"10.44.101.71:10000", "Telemetry":{"node_id_str":"XRv1", "subscription_id_str":"Sub1", "encoding_path":"openconfig-interfaces/interfaces/interface", "collection_id":26634, "collection_start_time":1530150069712, "msg_timestamp":1530150069712, "collection_end_time":1530150069712, "Content":{"subinterfaces":{"subinterface":{"index":0, "state":{"admin-status":"DOWN", "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":"2018-02-20T05:21:44Z", "out-broadcast-pkts":0, "out-discards":0, "out-errors":0, "out-multicast-pkts":0, "out-octets":0, "out-unicast-pkts":0, "type":"iana-if-type:ethernetCsmacd"}}}}}}}, {"Source":"10.44.101.71:10000", "Telemetry":{"node_id_str":"XRv1", "subscription_id_str":"Sub1", "encoding_path":"openconfig-interfaces/interfaces/interface", "collection_id":26634, "collection_start_time":1530150069712, "msg_timestamp":1530150069712, "collection_end_time":1530150069712, "Content":{"subinterfaces":{"subinterface":{"index":0, "state":{"admin-status":"DOWN", "counters":{"in-broadcast-pkts":0, "in-discards":0, "in-errors":0, "in-multicast-pkts":0, "in-octets":2423950616, "in-unicast-pkts":11208602, "in-unknown-protos":2, "out-broadcast-pkts":0, "out-discards":0, "out-errors":0, "out-multicast-pkts":0, "out-octets":318125864, "out-unicast-pkts":1514, "oper-status":"UP", "type":"iana-if-type:ethernetCsmacd"}}}}}}}, {"Source":"10.44.101.71:10000", "Telemetry":{"node_id_str":"XRv1", "subscription_id_str":"Sub1", "encoding_path":"openconfig-interfaces/interfaces/interface", "collection_id":26634, "collection_start_time":1530150069712, "msg_timestamp":1530150069712, "collection_end_time":1530150069712, "Content":{"subinterfaces":{"subinterface":{"index":0, "state":{"admin-status":"DOWN", "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":"2018-02-20T05:21:44Z", "out-broadcast-pkts":0, "out-discards":0, "out-errors":0, "out-multicast-pkts":0, "out-octets":0, "out-unicast-pkts":0, "type":"iana-if-type:ethernetCsmacd"}}}}}}}]

# 【参考】データ整形後の情報

## Cisco の Array 情報をインターフェース毎に分割

- ▶ June 20th 2018, 18:01:19.720 @timestamp: June 20th 2018, 18:01:19.720 @version: 1 message: {"Source":"10.44.101.71:10000","Telemetry":{"node\_id\_str":"XRv1","subscription\_id\_str":"Sub 1","encoding\_path":"openconfig-interfaces:interfaces/interface","collection\_id":26310,"collection\_start\_time":1529485279624,"msg\_timestamp":1529485279624,"collection\_end\_time":1529485279688},"Rows":{"Timestamp":1529485279665,"Keys":{"name":"GigabitEthernet0/0/0/6"},"Content":{"subinterfaces":{"subinterface":{"index":0,"state":{"admin-status":"UP","counters":{"in-broadcast-pkts":16869849,"in-discards":0,"in-errors":0,"in-multicast-pkts":0,"in-octets":30613866078,"in-unicast-pkts":125342500,"in-unknown-protos":0,"last-clear":"2018-02-20T05:21:44Z","out-broadcast-pkts":3,"out-discards":0,"out-errors":0,"out-multicast-pkts":0,"out-octets":0,"out-unicast-pkts":0,"out-unknown-protos":0,"last-clear":"2018-02-20T05:21:44Z","out-broadcast-pkts":0,"out-discards":0,"out-errors":0,"out-multicast-pkts":0,"out-octets":0}}}}}}}
- ▶ June 20th 2018, 18:01:19.718 @timestamp: June 20th 2018, 18:01:19.718 @version: 1 message: {"Source":"10.44.101.71:10000","Telemetry":{"node\_id\_str":"XRv1","subscription\_id\_str":"Sub 1","encoding\_path":"openconfig-interfaces:interfaces/interface","collection\_id":26310,"collection\_start\_time":1529485279624,"msg\_timestamp":1529485279624,"collection\_end\_time":1529485279688},"Rows":{"Timestamp":1529485279661,"Keys":{"name":"GigabitEthernet0/0/0/5"},"Content":{"subinterfaces":{"subinterface":{"index":0,"state":{"admin-status":"DOWN","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":"2018-02-20T05:21:44Z","out-broadcast-pkts":0,"out-discards":0,"out-errors":0,"out-multicast-pkts":0,"out-octets":0}}}}}}}
- ▶ June 20th 2018, 18:01:19.717 @timestamp: June 20th 2018, 18:01:19.717 @version: 1 message: {"Source":"10.44.101.71:10000","Telemetry":{"node\_id\_str":"XRv1","subscription\_id\_str":"Sub 1","encoding\_path":"openconfig-interfaces:interfaces/interface","collection\_id":26310,"collection\_start\_time":1529485279624,"msg\_timestamp":1529485279624,"collection\_end\_time":1529485279688},"Rows":{"Timestamp":1529485279658,"Keys":{"name":"GigabitEthernet0/0/0/4"},"Content":{"subinterfaces":{"subinterface":{"index":0,"state":{"admin-status":"DOWN","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":"2018-02-20T05:21:44Z","out-broadcast-pkts":0,"out-discards":0,"out-errors":0,"out-multicast-pkts":0,"out-octets":0}}}}}}}
- ▶ June 20th 2018, 18:01:19.714 @timestamp: June 20th 2018, 18:01:19.714 @version: 1 message: {"Source":"10.44.101.71:10000","Telemetry":{"node\_id\_str":"XRv1","subscription\_id\_str":"Sub 1","encoding\_path":"openconfig-interfaces:interfaces/interface","collection\_id":26310,"collection\_start\_time":1529485279624,"msg\_timestamp":1529485279624,"collection\_end\_time":1529485279688},"Rows":{"Timestamp":1529485279653,"Keys":{"name":"GigabitEthernet0/0/0/3"},"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":1775840085,"in-unicast-pkts":3481370,"in-unknown-protos":0,"last-clear":"2018-02-20T05:21:37Z","out-broadcast-pkts":2,"out-discards":0,"out-errors":0,"out-multicast-pkts":0,"out-octets":0}}}}}}}

# KibanaからのTelemetry Data確認

## 3) Telemetry Dataの確認

### 3-2) Aristaは綺麗に見える

```
Time ▾ _source
▶ June 29th 2018, 15:42:00.403 @version: 1 update.Sysdb.interface.counter.eth.slice.phy.1.intfCounterDir.Ethernet2.intfCounter.current.rates.outBitsRate: 54.113
update.Sysdb.interface.counter.eth.slice.phy.1.intfCounterDir.Ethernet2.intfCounter.current.rates.inPktsRate: 0.027
update.Sysdb.interface.counter.eth.slice.phy.1.intfCounterDir.Ethernet2.intfCounter.current.rates.outPktsRate: 0.051
update.Sysdb.interface.counter.eth.slice.phy.1.intfCounterDir.Ethernet2.intfCounter.current.rates.statsUpdateTime: 11,151,611.514
update.Sysdb.interface.counter.eth.slice.phy.1.intfCounterDir.Ethernet2.intfCounter.current.rates.inBitsRate: 16.025

▶ June 29th 2018, 15:42:00.402 @version: 1 update.Sysdb.interface.counter.eth.slice.phy.1.intfCounterDir.Ethernet3.intfCounter.current.rates.outBitsRate: 51.687
update.Sysdb.interface.counter.eth.slice.phy.1.intfCounterDir.Ethernet3.intfCounter.current.rates.inPktsRate: 0.022
update.Sysdb.interface.counter.eth.slice.phy.1.intfCounterDir.Ethernet3.intfCounter.current.rates.outPktsRate: 0.046
update.Sysdb.interface.counter.eth.slice.phy.1.intfCounterDir.Ethernet3.intfCounter.current.rates.statsUpdateTime: 11,151,611.514
update.Sysdb.interface.counter.eth.slice.phy.1.intfCounterDir.Ethernet3.intfCounter.current.rates.inBitsRate: 11.087
```

Field	Type	Value	Count
@timestamp	date	June 28th 2018, 10:41:27.17	1
@version	number	1	1
_id	string	ZMONRQB9j5yChhatMP	1
_index	string	telemetry_arista_sampling_v	1
_score	number	-	1
_type	string	logs	1
dataset	string	10.44.101.81	1
timestamp	number	1,530,149,509,225	1
update.Sysdb.interface.counter.eth.slice.phy.1.intfCounterDir.Ethernet3.intfCounter.current.rates.inBitsRate	number	1.766	1
update.Sysdb.interface.counter.eth.slice.phy.1.intfCounterDir.Ethernet3.intfCounter.current.rates.inPktsRate	number	0.003	1
update.Sysdb.interface.counter.eth.slice.phy.1.intfCounterDir.Ethernet3.intfCounter.current.rates.outBitsRate	number	43.37	1
update.Sysdb.interface.counter.eth.slice.phy.1.intfCounterDir.Ethernet3.intfCounter.current.rates.outPktsRate	number	0.028	1
update.Sysdb.interface.counter.eth.slice.phy.1.intfCounterDir.Ethernet3.intfCounter.current.rates.statsUpdateTime	number	11,047,179.188	1
update.Sysdb.interface.counter.eth.slice.phy.1.intfCounterDir.Ethernet3.intfCounter.current.statistics.inBroadcastPkts	number	0	1
update.Sysdb.interface.counter.eth.slice.phy.1.intfCounterDir.Ethernet3.intfCounter.current.statistics.inDiscards	number	0	1
update.Sysdb.interface.counter.eth.slice.phy.1.intfCounterDir.Ethernet3.intfCounter.current.statistics.inErrors	number	0	1
update.Sysdb.interface.counter.eth.slice.phy.1.intfCounterDir.Ethernet3.intfCounter.current.statistics.inMulticastPkts	number	0	1
update.Sysdb.interface.counter.eth.slice.phy.1.intfCounterDir.Ethernet3.intfCounter.current.statistics.inOctets	number	1,846,891,283	1



# 【参考】データ整形後の情報

Arista デバイスからの情報を少しでも見やすく

```
Time ▾      _source
▼ June 29th 2018, 16:05:50.431  @timestamp: June 29th 2018, 16:05:50.431  newkeyword: Ethernet2  @version: 1
update.Sysdb.interface.counter.eth.slice.phy.1.intfCounterDir.Ethernet.intfCounter.current.rates.outBitsRate: 75.79864463974691
update.Sysdb.interface.counter.eth.slice.phy.1.intfCounterDir.Ethernet.intfCounter.current.rates.inPktsRate: 0.038527292983615936
update.Sysdb.interface.counter.eth.slice.phy.1.intfCounterDir.Ethernet.intfCounter.current.rates.outPktsRate: 0.07209252369387106
update.Sysdb.interface.counter.eth.slice.phy.1.intfCounterDir.Ethernet.intfCounter.current.rates.statsUpdateTime: 11153041.009580828
```

Table JSON

[View surrounding documents](#) [View single document](#)

@timestamp	June 29th 2018, 16:05:50.431
@version	1
_id	RB1cSmQBUIbM8K50w1xp
_index	telemetry_arista_sampling_veos_2018.06
_score	-
_type	logs
dataset	10.44.101.81
newkeyword	Ethernet2
timestamp	1530255371225
update.Sysdb.interface.counter.eth.slice.phy.1.intfCounterDir.Ethernet.intfCounter.current.rates.inBitsRate	22.548334513718164
update.Sysdb.interface.counter.eth.slice.phy.1.intfCounterDir.Ethernet.intfCounter.current.rates.inPktsRate	0.038527292983615936
update.Sysdb.interface.counter.eth.slice.phy.1.intfCounterDir.Ethernet.intfCounter.current.rates.outBitsRate	75.79864463974691
update.Sysdb.interface.counter.eth.slice.phy.1.intfCounterDir.Ethernet.intfCounter.current.rates.outPktsRate	0.07209252369387106

# 【参考】Juniper デバイスからの情報収集について

- Gitで公開されているFreeのOpenConfig / gRPC対応型Telemetry Collector (<https://github.com/nileshsimaria/jtimon>)
- Telemetry DataをGPBデシアライズし、人が読めるテキスト形式でシステムに出力する
- ネットワンのにて動作確認中
- Influxdbとprometheusに出力可能（Kafkaは未対応）

Juniper 独自のデータフォーマットであれば、Open-NTI というツールもある

- <https://github.com/Juniper/open-nti>

# 【参考】nileshsimaria/jtimon 実行例

複数のJSON  
に分割されている



且つ、



情報要素毎の  
JSONになっている



```
{
  "key": "__prefix__",
  "Value": {
    "StrValue": "/interfaces/interface[name='ge-0/0/1']/"
  }
},
{
  "key": "name",
  "Value": {
    "StrValue": "ge-0/0/1"
  }
},
{
  "key": "state/type",
  "Value": {
    "StrValue": "ethernetCsmacd"
  }
},
{
  "key": "state/mtu",
  "Value": {
    "UIntValue": 1514
  }
},
{
  "key": "state/name",
  "Value": {
    "StrValue": "ge-0/0/1"
  }
},
}
```

Key/Valueのペアが非効率

Key	Value
key	__prefix__
Value. StrValue	/interfaces/interface[name='ge-0/0/1
key	name
Value. StrValue	ge-0/0/1
key	state/type
Value. StrValue	ethernetCsmacd
key	state/mtu
Value. UIntValue	1514

このままでは利用不可か...

# 【参考】データ整形後の情報

Juniper デバイスからの情報を整形し、可視化

```
▶ June 22nd 2018, 10:07:53.139 component_id: 0 juniper_timestamp: 1529629101022 out-pkts: 7256882 system_id: vMX2 init-time: 1,519,197,607 out-multicast-pkts: 0 out-unicast-pkts: 7256882
in-pkts: 7295790 operational-state: up sequence_number: 73063 out-octets: 1794622286 path: sensor_1000_1_2:/junos/system/linecard/interface/logical/usage:/
junos/system/linecard/interface/logical/usage:/PFE in-octets: 565425415 @timestamp: June 22nd 2018, 10:07:53.139 in-multicast-pkts: 3603835 @version: 1 in-
unicast-pkts: 3691955 timestamp: 1529629099153 juniper_prefix: /interfaces/interface[name='ge-0/0/1']/subinterfaces/subinterface[index='0']/ _id: FBsIJWQBUIb
M8KS0hn07 _type: logs _index: telemetry_juniper_sampling_veos_2018.06 _score:

▶ June 22nd 2018, 10:07:52.183 component_id: 0 juniper_timestamp: 1529629100083 system_id: vMX2 init-time: 1,519,197,607 operational-state: up in-pkts: 1086503 sequence_number: 73060
path: sensor_1002_1_2:/junos/system/linecard/interface/logical/usage:/junos/system/linecard/interface/logical/usage:/PFE in-octets: 114983967 @timestamp: J
une 22nd 2018, 10:07:52.183 in-multicast-pkts: 0 @version: 1 in-unicast-pkts: 1086503 timestamp: 1529629098150 juniper_prefix: /interfaces/interface[name='ge
-0/0/2']/subinterfaces/subinterface[index='0']/ _id: ExsIJWQBUIbM8KS0gnP- _type: logs _index: telemetry_juniper_sampling_veos_2018.06 _score: -
```

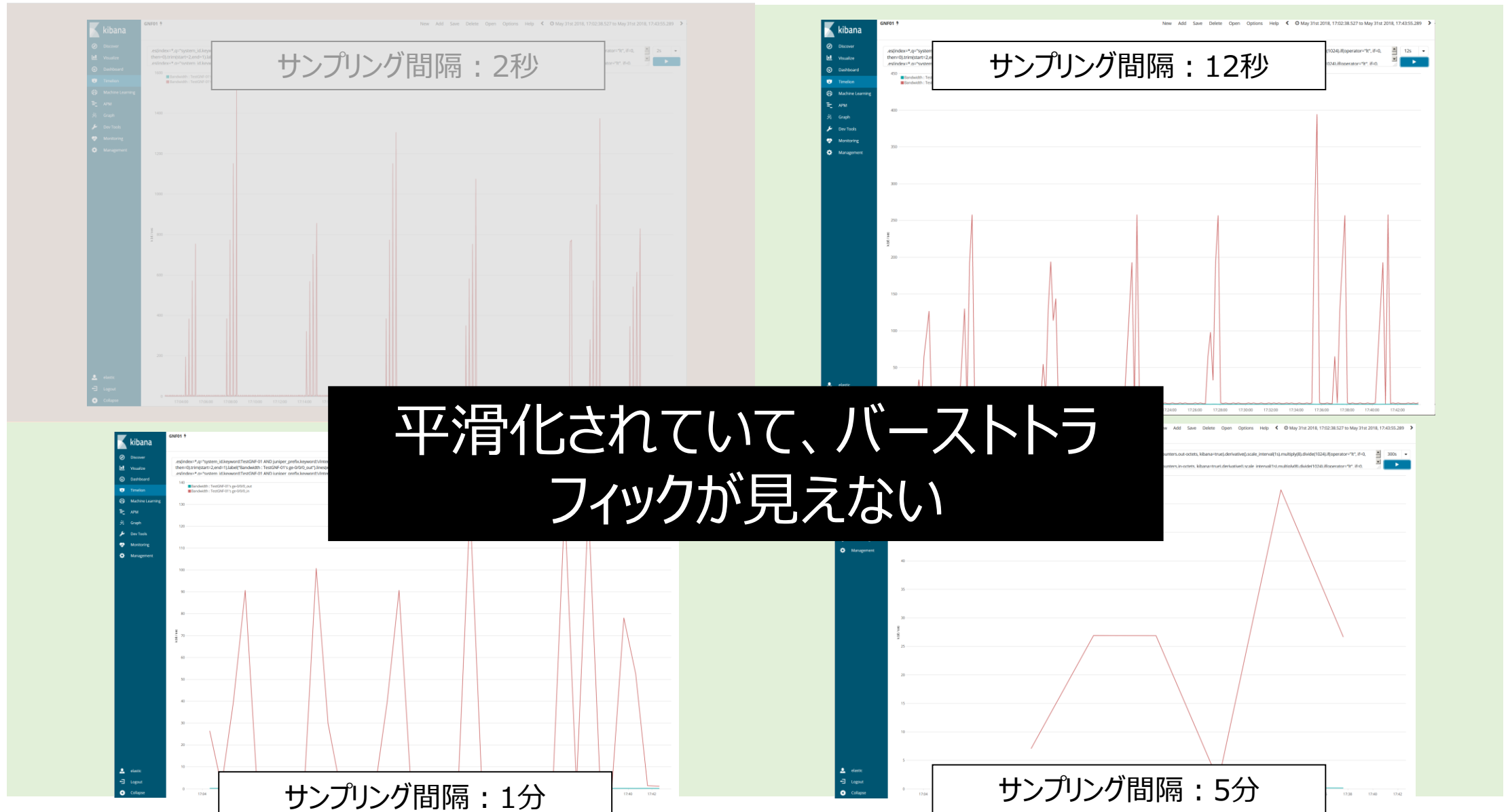
## まとめ

- 基本的にフリーなツールと仮想ルータ/スイッチの組み合わせで Telemetry 情報の確認は可能
- デバイスから送信される情報にはばらつきはあるため、加工は必要
- より多くの方に試していただき、Telemetry 情報の利活用方法を検討していく必要がある

# 【参考】テストシナリオ

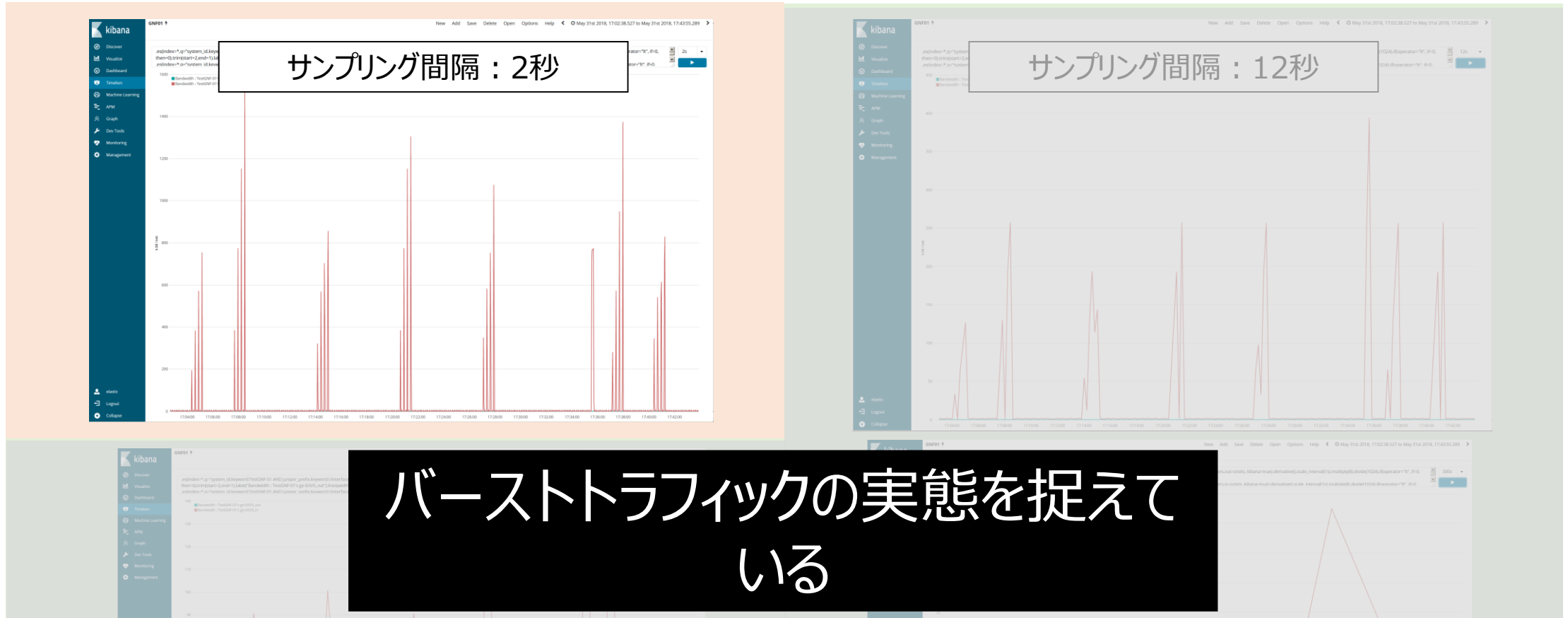
- ① バースト性トラフィックの検知
- ② Packet DropとInterface Queueの相関
- ③ 機械学習によるアノマリ検知

# 【参考】リアルタイム可視化 – Burst Traffic検知



試験条件 : 1000byte 100~400pps×1秒と0pps×600秒を繰り返すTrafficを印加

# 【参考】リアルタイム可視化 – Burst Traffic検知

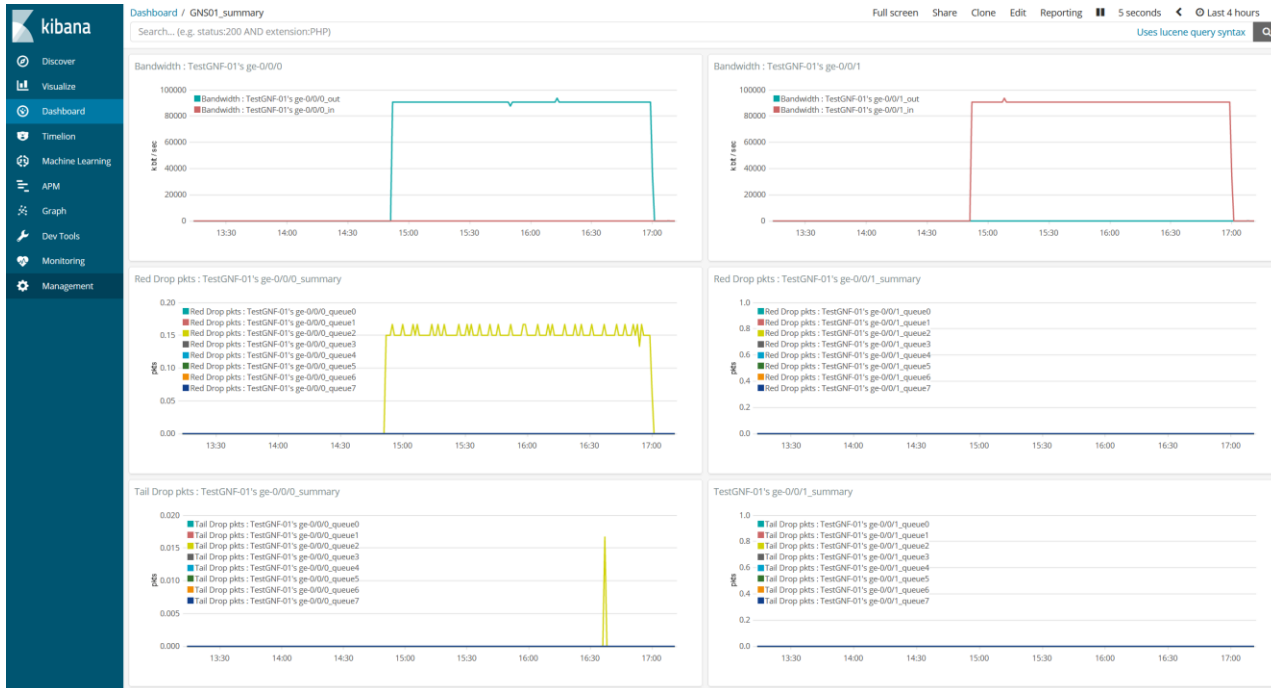


Telemetryを使用する事で、今まで見えていなかった事象  
(≡**サイレント障害**) を可視化、検知する事が出来る



# 【参考】リアルタイム可視化 - Packet DropとQueueの相関①

## Interface毎の使用帯域 / Dropカウンタ



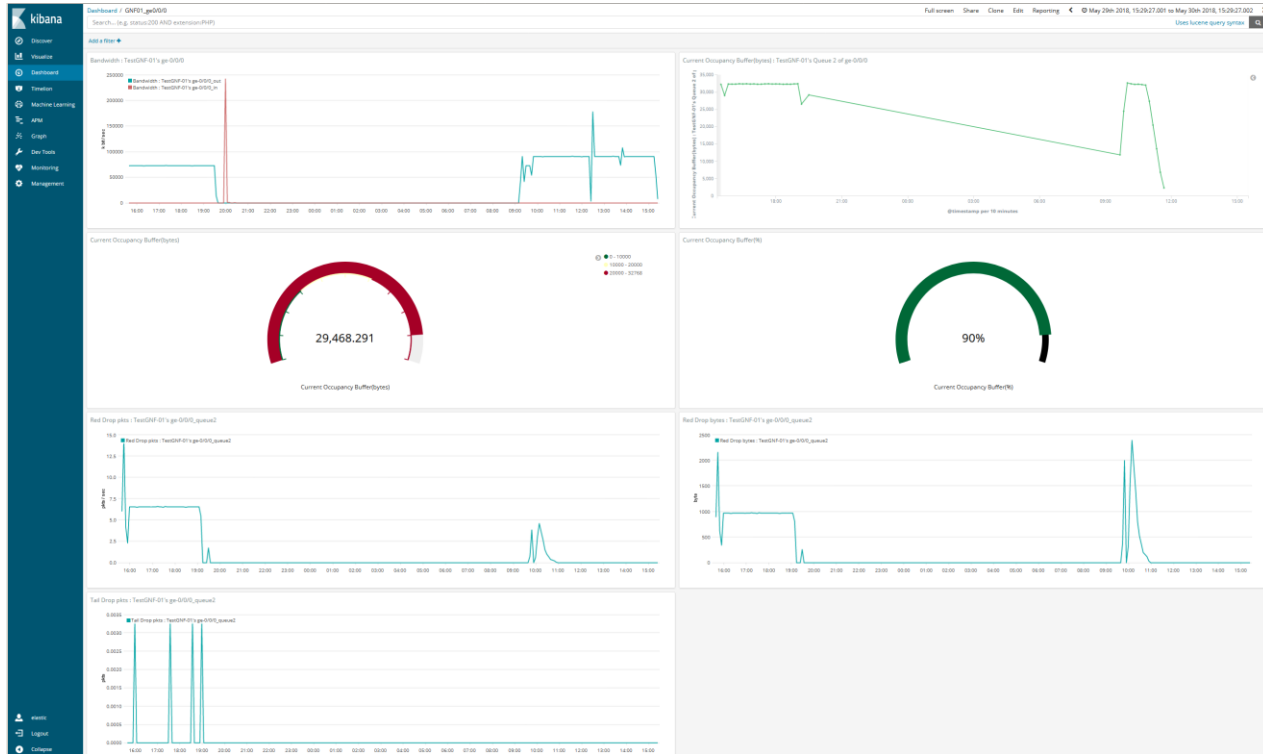
← Interface単位の帯域 (リアルタイム)

← Packet Drop状況 (リアルタイム)  
※この例では、ge 0/0/0 のqueue 2上での Red Drop, Tail Dropを把握

様々なデータを時系列データとして扱う事で、  
**各データ間の相関**を把握が可能となる

# 【参考】リアルタイム可視化 - Packet DropとQueueの相関②

## Interfaceの使用帯域 / Packet Drop / Interface queueの相関



Interfaceの使用帯域 (リアルタイム)  
Queue 2 の利用状態 (リアルタイム)

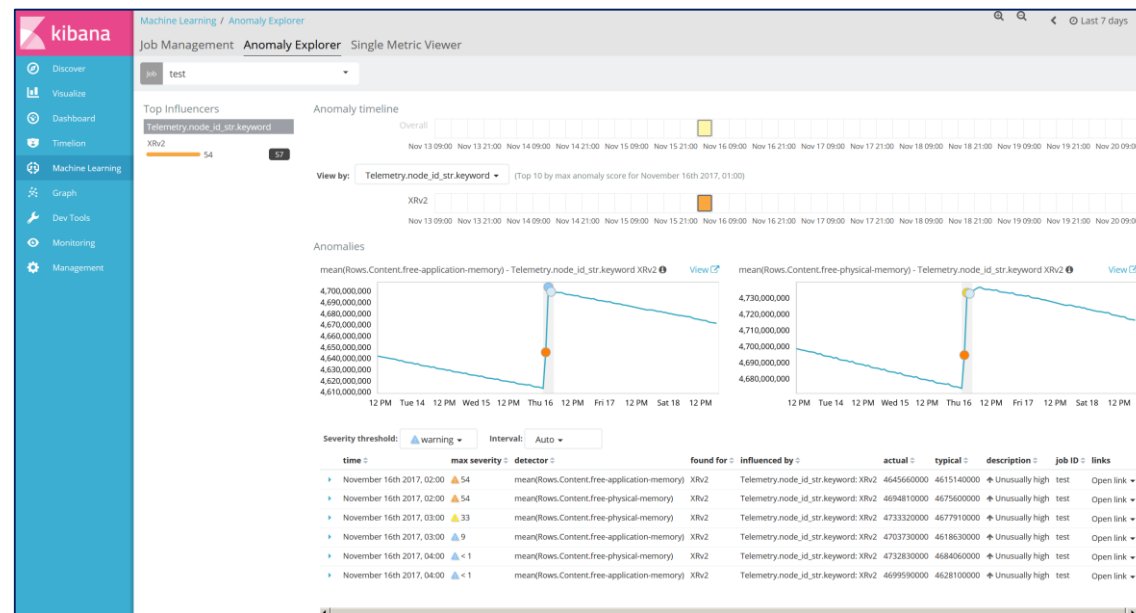
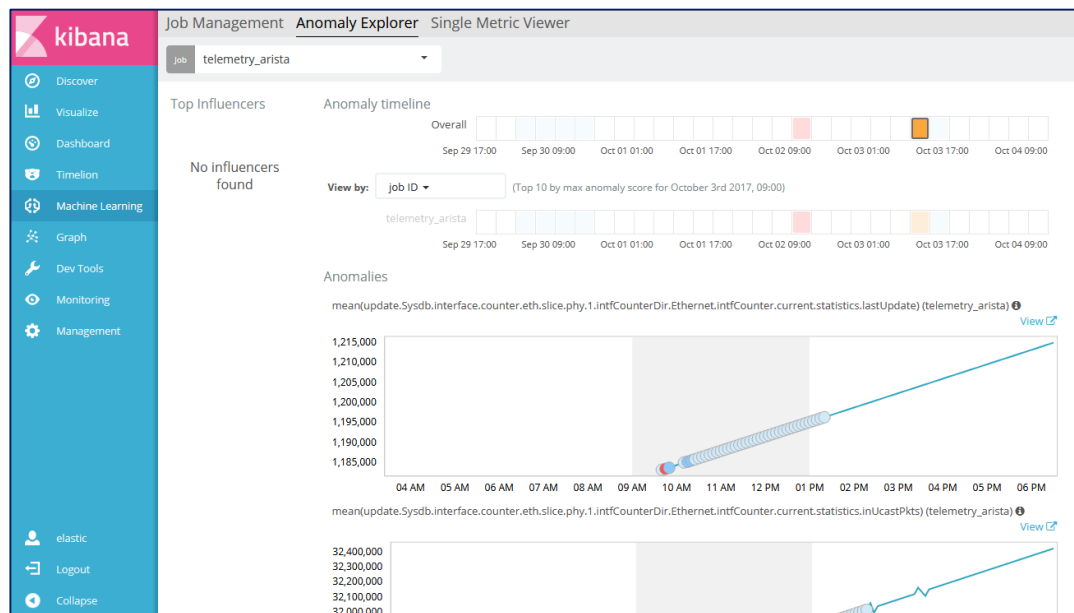
Queue 2 の利用状態 (%)

Queue 2のRed Drop bytes/pkts  
Tail Drop pkts (リアルタイム)

様々なデータを時系列データとして扱う事で、  
**各データ間の相関**を把握が可能となる

# 【参考】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。  
人の目では気づき難い微小な変化を検知 (Interfaceカウンタ増減・メモリ解放等、微小変化の検知)



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