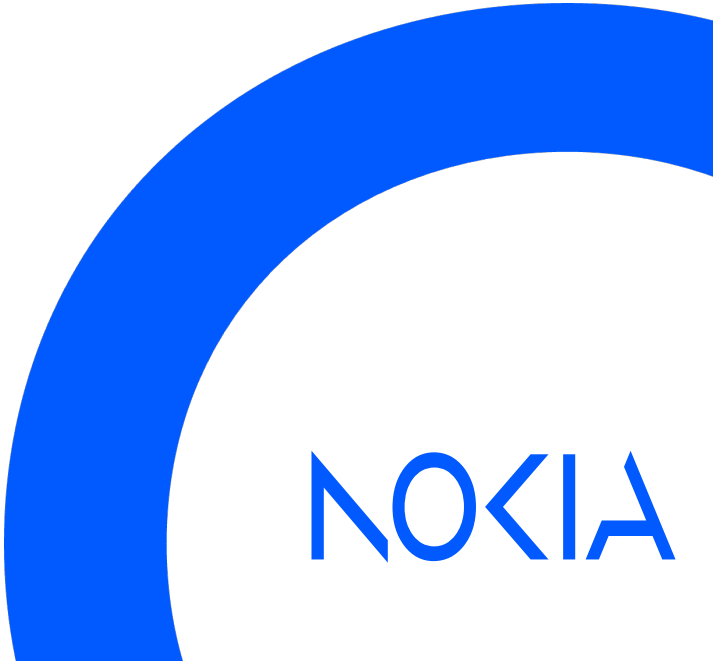
gNMIcを活用したマルチベンダー環境でのテレメトリ技術の実践

小川 怜 (Ogawa Satoshi)
ノキアソリューションズ＆ネットワークス

発表内容

1. 本発表について
2. 背景・課題
3. gNMIc とは？
4. gNMIc で解決する(できそうな)課題
5. デモ
6. まとめ・議論



1. 本発表について

発表概要

1. ストリーミング・テレメトリーの背景・課題を振り返り
2. gNMIc の解説
3. 諸課題について議論したい

議論・意見交換したいこと

1. テレメトリの導入課題って？ (そもそも使いたい動機は?)
2. gNMIc で解決できそうな課題
3. 潜在的問題 (実はその先にある課題？)

ぜひ率直なご感想なご意見をいただけると嬉しいです

NOKIA

1. 本発表について

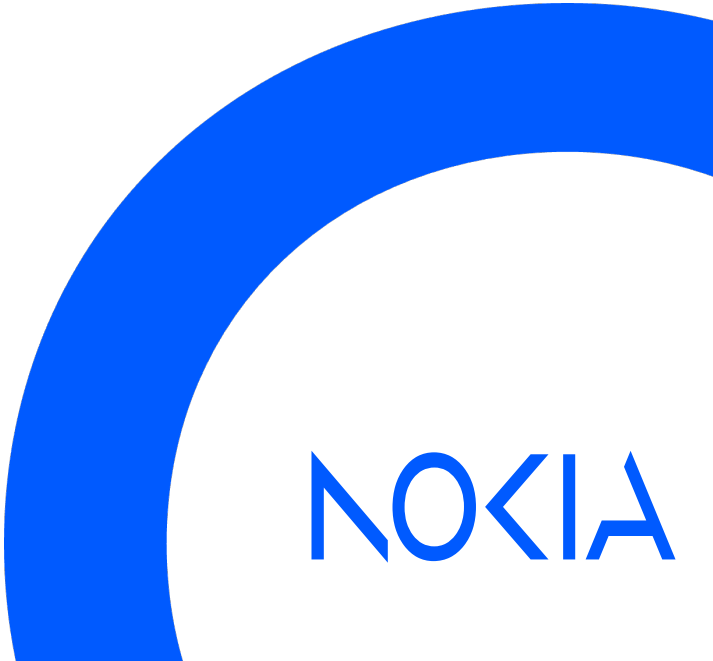
自己紹介

- 名前:　小川　怜　(Ogawa Satoshi)
- 所属:　ノキアソリューションズ&ネットワークス合同会社
  IP製品事業部 ソリューションアーキテクト
- 過去の発表:
  - JANOG26 - IPv6時代のIPv4を考える
  - ENOG76 - クイックスタートcontainerlab

✓ Email : satoshi.ogawa@nokia.com
✓ Slack(janog-meeting) : @Satoshi Ogawa
✓ Twitter : saogawa

発表内容

1. 本発表について
2. 背景・課題
3. gNMIc とは？
4. gNMIc で解決する(できそうな)課題
5. デモ
6. まとめ・議論



2. 背景・課題

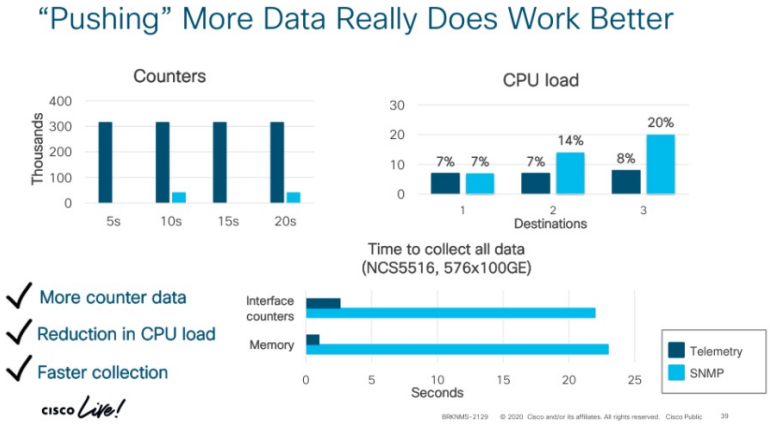
背景

- 過去のJANOGでもトラフィック可視化の課題、テレメトリー導入の重要性が議論されてきた。

- ストリーミング・テレメトリのメリット
  - リアルタイム性
  - ネットワーク装置の負荷軽減　etc

- でも意外とサービス設備で導入されていない？！

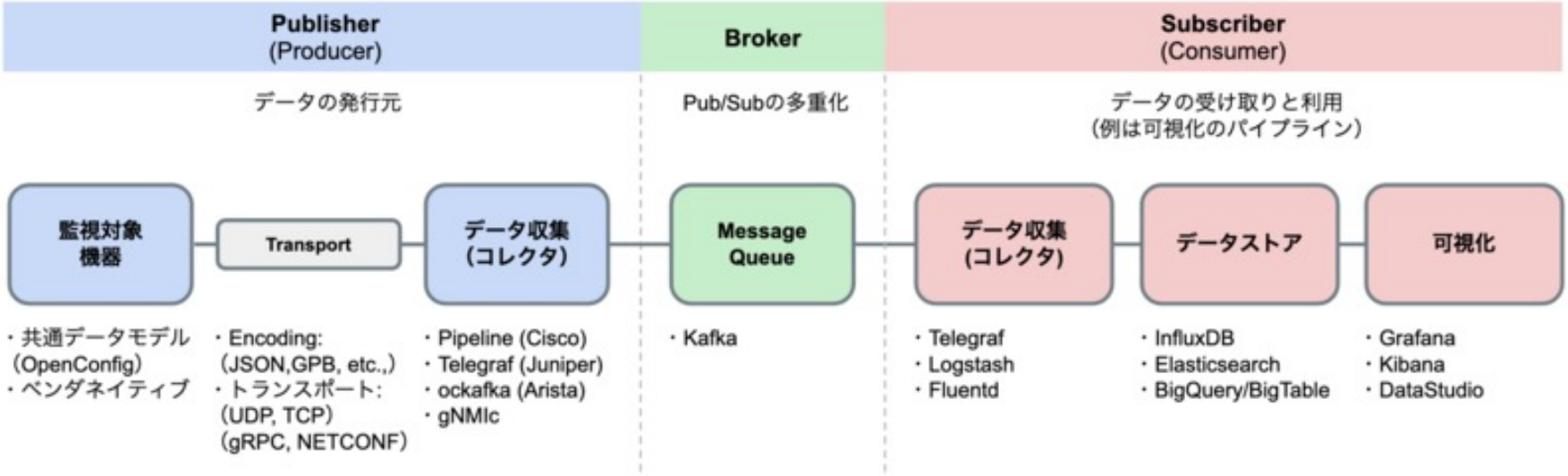
引用元：Discord



2. 背景・課題

テレメトリの基本構成

引用元: NTTコミュニケーションズ ENGINEER BLOG
次世代の監視技術 – Telemetry技術のご紹介



2. 背景・課題

課題要件 整理 (主にコレクター周辺)

JANOG Telemetry-WG：Telemetry デザインワークショップ報告求められる機能と理想像

コレクタに関しては、以下のような意見が挙がった。

- 様々なトランスポート、メッセージ形式で入力を受けられること
- 様々なデータベース、メッセージングシステムに出力できること
- データ整形（絞込み、フラット化、フィールドを使った計算[*1]）できること
- 複数の出力先に転送できること
- 柔軟にスケールできること
- 冗長構成が実現できること
- 再送処理ができること
- コレクタ自体のメトリクスを取得できること
- コレクタ自体の死活監視ができること

[*1] フィールドを使った計算には、同一レコード内でのフィールド間の計算（例えば、合計値と総数から平均値を求める等）や異なるレコード間で計算する（前のレコードとインクリメントする値の差分を求める等）が想定される

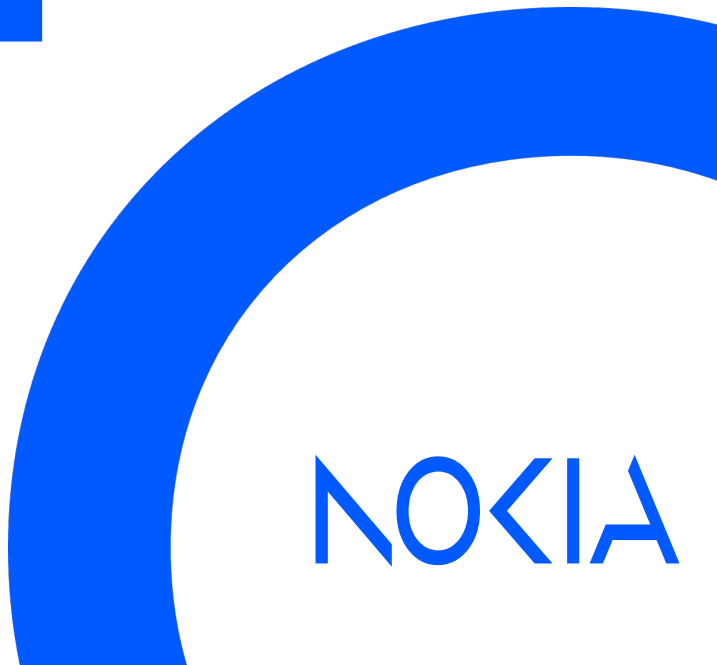
2. 背景・課題

課題

- インテグが複雑そう。そもそも使い方をよく知らない。
- マルチベンダー環境
- 冗長性・拡張性を考えると頭イタイ

発表内容

1. 本発表について
2. 背景・課題
3. gNMIc とは？
4. gNMIc で解決する(できそうな)課題
5. デモ
6. まとめ・議論



3. gNMIc とは？

gNMIc 概要

gNMIc
Get / Set / Subscribe / Collect

- 読み方: ジーエヌエムアイシー
- オープンソース・ソフトウェア。開発言語 Go。
- gNMIのクライアントとコレクター機能を持つ
- gRPC/gNMIのCapabilities、Get、Set、Subscribe RPC をフルサポート
- 初期リリースは2020年2月。本日時点で v0.31.0が最新リリース

  → 2回/月の頻度で機能追加している開発頻度

OPENCONFIG / gNMIc

openconfig/gnmic

2022年10月ノキアの開発からOpenConfigプロジェクトに寄贈

NOKIA

3. gNMIc とは？

3-1：基本的な使い方 - インストール

Linux/MAC環境：パッケージ

```
bash -c "$(curl -sL https://get-gnmic.openconfig.net)"
```

※WindowsはWSLが必要

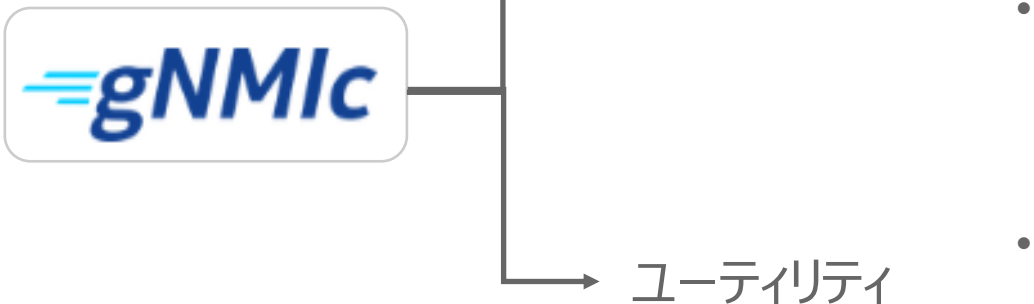
Docker

```
docker pull gnmic/gnmic:latest
```


NOKIA

3. gNMIc とは？

3-2：基本的な使い方 – コマンド体型



- Capability
- Get
- Set
- Subscribe

- Prompt
- Generate
- Diff

3. gNMIc とは？

3-2：基本的な使い方 – 設定 (flags/env/file)

フラグ

```
gnmic [global-flags] [command] [local-flags]
```

実行例

```
$ gnmic --address 10.0.0.1:57400 \
    --username admin --password admin\
    --skip-verify \
    get --path /interfaces/interface[name=mgmt0]
```


NOKIA

3. gNMIc とは？

3-2：基本的な使い方 – 設定 (flags/env/file)

env

```
export GNMIC_ADDRESS=10.0.0.1:57400
export GNMIC_USERNAME=admin
export GNMIC_PASSWORD=admin
export GNMIC_SKIP_VERIFY=true
```

実行例

```
$ gnmic [global-flags] get --path /interfaces/interface[name=mgmt0]
```


NOKIA

3. gNMIc とは？

3-2：基本的な使い方 – 設定 (flags/env/file)

file

```
$ cat ./gnmic.yaml
address: router1
username: admin
password: admin
insecure: true
encoding: json_ietf
get-path:
- /interfaces/interface[name=mgmt0]
```

実行例

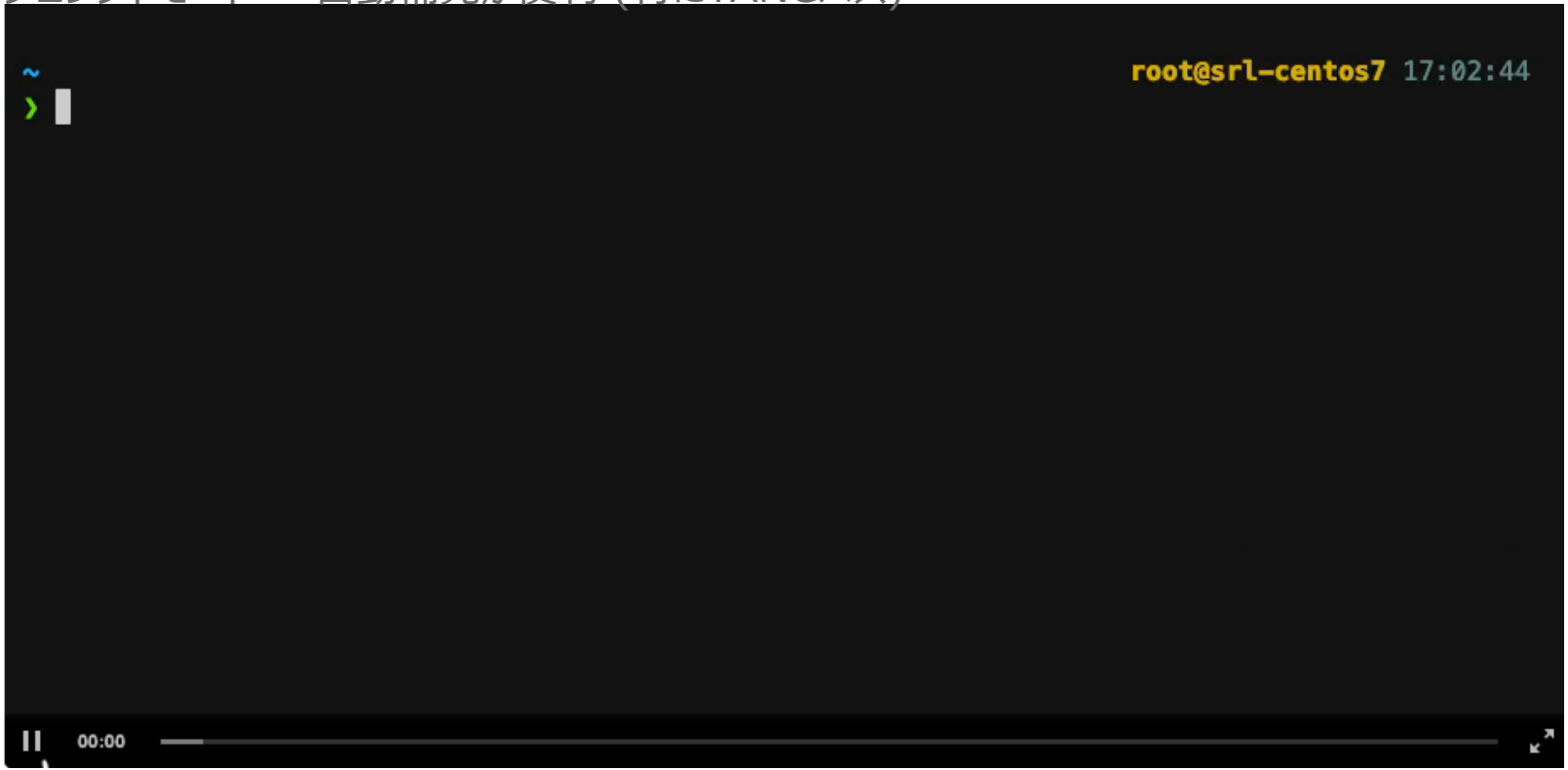
```
$ gnmic --config ./gnmic.yaml get
```


NOKIA

3. gNMIc とは？

3-2：基本的な使い方 (ユーティリティ：プロンプト)

プロンプトモード = 自動補完が便利 (特にYANGパス)



発表内容

1. 本発表について
2. 背景・課題
3. gNMIc とは？
4. gNMIc で解決する(できそうな)課題
5. デモ
6. まとめ・議論

4. gNMIc で解決する(できそうな)課題

課題

1. インテグが複雑そう。そもそも使い方をよく知らない。
2. マルチベンダー環境なのでしんどい
3. 冗長性も考えると頭イタイ

4. gNMIc で解決する(できそうな)課題

4-1 : 課題1. 複雑なインテグレーション

課題

- ベンダー推奨コレクターはベンダー毎に異なる = 複雑なインテグの要因の1つ?!

解決方法

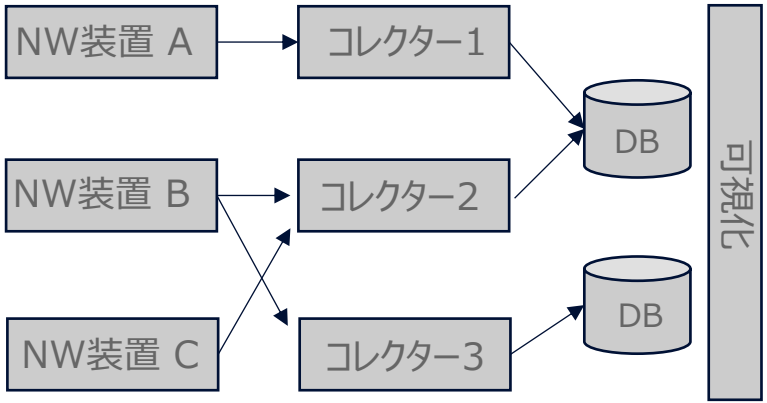
- 導入規模に合わせた構成で解決！

4. gNMIc で解決する(できそうな)課題

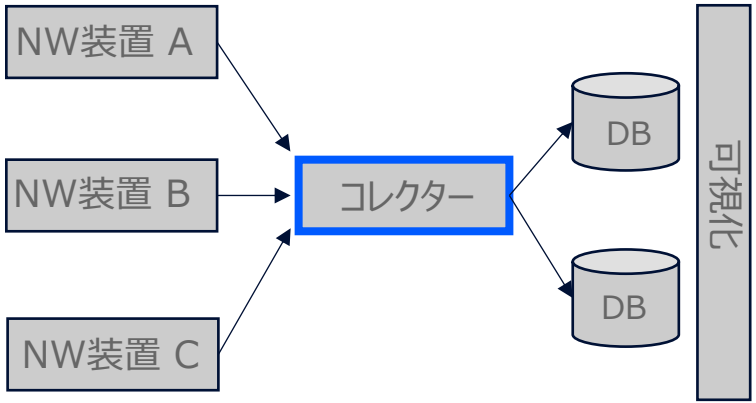
4-1 : 課題1. 複雑なインテグレーション

コレクターを「共通化」できたらインテグが簡素化できるはず

従来のテレメトリ構成



期待したい構成



4. gNMIc で解決する(できそうな)課題

4-1 : 課題1. 複雑なインテグレーション

様々なコレクタが存在
→ 複雑な構成の一因?

テレメトリ実装サマリ

ベンダー名	OS	gRPC対応	ベンダー コレクター	OSS実装	gNMIc 動作確認
Cisco	IOS-XR	○	Crosswork Health Insights	Pipeline Telegraf	○
Juniper	JUNOS	○	Paragon Insights	Jtimon Telegraf	○
Arista	EOS	○	Arista CloudVision	Ockafka Openconfigbeat	○
Nokia	SR-OS	○	NSP	gNMIc Telegraf	○
	SR-Linux	○	FSS/NSP	gNMIc Telegraf	○

4. gNMIc で解決する(できそうな)課題

4-2：課題2. マルチベンダー対応

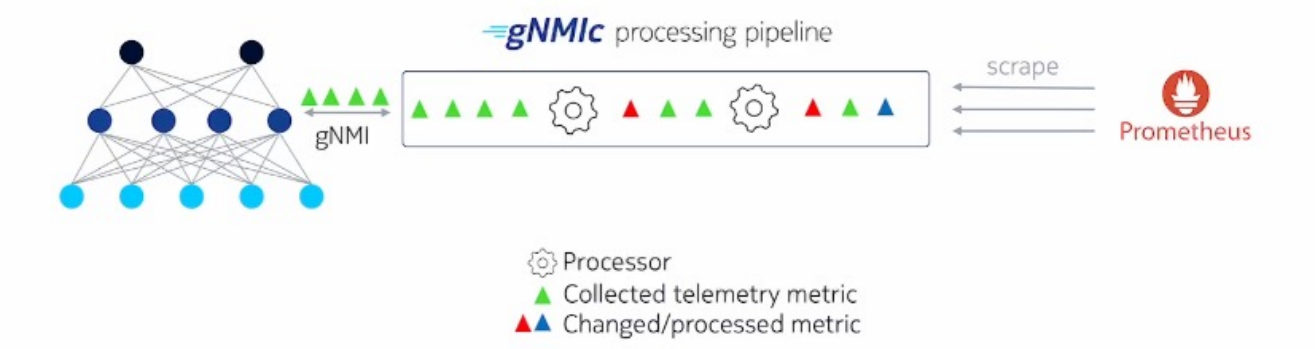
課題

・NW装置からの入力データ = データ構造がNOS毎に異なる

・コレクターからの出力データ = 柔軟に選択したい(送信先・データベース etc)

解決方法

・柔軟な入力・データ整形・出力を実現するパイプラインで解決！

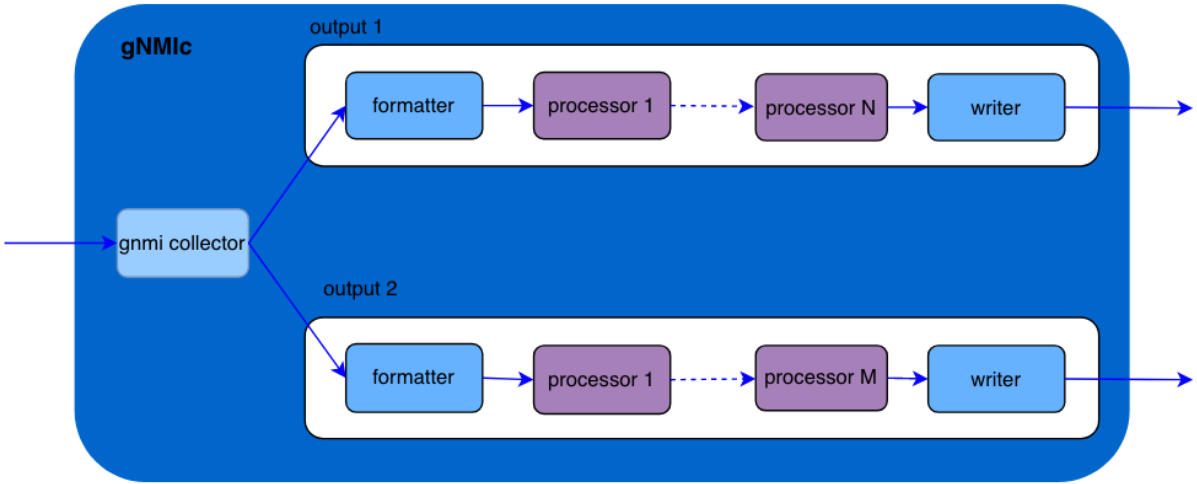


4. gNMIc で解決する(できそうな)課題

4-2：課題2. マルチベンダー対応

gNMIcのパイプライン　-「Processor」

- ネットワーク装置から収集するデータをデータベースに保存する前に
  データの整形・加工を行う一連のデータ処理



4. gNMIc で解決する(できそうな)課題

4-2：課題2. マルチベンダー対応

Processorの記述例

```
outputs:
  output1:
    type: influxdb
    url: http://localhost:8086
    bucket: telemetry
    token: srl:srl
    batch-size: 1000
    flush-timer: 10s
    event-processors:
      - proc-convert-integer
      - proc-delete-tag-name
      - proc-delete-value-name
```

```
processors:
  proc-convert-integer:
    event-convert:
      value-names:
        - ".*"
      type: int

  proc-delete-tag-name:
    event-delete:
      tag-names:
        - "^subscription-name"

  proc-delete-value-name:
    event-delete:
      value-names:
        - ".*out-unicast-packets"
```

4. gNMIc で解決する(できそうな)課題

4-2：課題2. マルチベンダー対応

Processorの主な機能

単位の正規化／変換：

利用率などの単位を、さまざまなベンダーから共通の単位に正規化することが可能
MB、KBをBytesに、または様々な時間フォーマットを共通のエポック・タイムまたはdatetimeフォーマットに変換する など

メトリクスのフィルタリング：

データベースの保存容量を最適化するために、必要なメトリクスを許可/削除し、何が書き込まれるかを制御することを可能にする

データ変換：

テレメトリのエンコーディングやベンダーの実装によっては、コレクタは整数である必要があるデータを文字列形式で受け取ることがある
変換プロセッサは、そのようなメトリクスを自動的に変換、ユーザーがデータベースでそのメトリクスに対して操作を実行できるようにする

タグ抽出：

gNMI を介して収集された特定のメトリクスについて、特定の値を抽出してメトリクスのタグに付与させ、
データベース内で適切なレイアウトを可能にする。

4. gNMIc で解決する(できそうな)課題

4-2：課題2. マルチベンダー対応

例1：単位の正規化/変換

```
processors:
  convert-data-unit:
    event-data-convert:
      value-names:
        - ".*-octets$"
      from: B
      to: KB
      keep: false
      old:
      new:
      debug: false
```

```
{
  "name": "default",
  "timestamp": 1607290633806716620,
  "tags": {
    "port_port-id": "A/1",
    "source": "172.17.0.100:57400",
    "subscription-name": "default"
  },
  "values": {
    "/state/port/ethernet/statistics/in-octets": "2048"
  }
}
```

```
"values": {
    "/state/port/ethernet/statistics/in-octets": 2
}
```

キロバイトに単位変換

4. gNMIc で解決する(できそうな)課題

4-2：課題2. マルチベンダー対応

例2：フィルタリング

```
processors:
  # processor name
  allow-processor:
    # processor type
    event-allow:
      condition: ".tags.interface_name == 1/1/1"
```

4. gNMIc で解決する(できそうな)課題

4-2：課題2. マルチベンダー対応

例3：データ整形

マルチベンダー対応で発生しそうな問題

- NW装置によって受信されるデータ構造が異なる
  - 例. 次スライド
- NW装置によっては値を持っていないケースがある。
  - 例. IF情報で送信レート(bits per sec)のバリュー

Starlark言語により柔軟なデータ整形が可能

NOKIA

4. gNMIc で解決する(できそうな)課題

4-2：課題2. マルチベンダー対応

例3：データ整形

Starlarkとは？

- 構成言語を目的としたプログラム言語の1つ
- Pythonの方言

```
processors:
  # プロセッサ名
  sample-processor:
    # プロセッサの指定
    event-starlark:
      # source以降、starlarkを記述
      source: |
        def apply(*events):
          # データ処理ロジックを記述
          return events
      # starlarkプログラムファイルへのパス
      script:
      # デバックON/OFF
      debug: false
```


NOKIA

4. gNMIc で解決する(できそうな)課題

4-2：課題2. マルチベンダー対応

例3：データ整形

NW装置が送信してくるデータ構造 例①

1つのアレイに**「単一」**のバリューが含まれる


```
[
{
  "name": "eos_cpu",
  "timestamp": 1680538251708431499,
  "tags": {
    "component_name": "CPU3",
    "source": "acm-lab-agg-sw2.custcbb.local:6030",
    "subscription-name": "eos_cpu"
  },
  "values": {
    "/components/component/cpu/utilization/state/avg": 21
  }
}
]
```

4. gNMIc で解決する(できそうな)課題

4-2 : 課題2. マルチベンダー対応

例3 :データ整形

NW装置が送信してくるデータ構造 例②

1つのアレイに**「複数」**のバリューが含まれる


```
[
{
  "name": "mx_cpu",
  "timestamp": 1680540418368000000,
  "tags": {
    "component_name": "FPC8:CPU0",
    "property_name": "mem-util-kernel-size",
    "source": "acm-lab-core1.custcbb.local:57400",
    "subscription-name": "mx_cpu"
  },
  "values": {
    "/components/component/properties/property/state/value": 1829480188
  }
},
{
  "name": "mx_cpu",
  "timestamp": 1680540418368000000,
  "tags": {
    "component_name": "FPC8:CPU0",
    "property_name": "mem-util-kernel-bytes-allocated",
    "source": "acm-lab-core1.custcbb.local:57400",
    "subscription-name": "mx_cpu"
  },
  "values": {
    "/components/component/properties/property/state/value": 211117736
  }
}
]
```


例①②の組み合わせだったり、もっと複雑だったりするかもしれない。。

4. gNMIc で解決する(できそうな)課題

4-2 : 課題2. マルチベンダー対応

例3 :データ整形

gNMIc Processor例: event-starlark

同じイベントメッセージの一部である値を「アングループ化」（分割）する

```
processors:
  ungroup_vals:
    event-starlark:
      source: |
        def apply(*events):
          ungrouped_events = []
          for e in events:
            for k, v in e.values.items():
              # 値のない新しいイベントを作成する
              new_event = Event(e.name, e.timestamp, e.tags)
              # 新しいイベントに1つの値を追加する
              new_event.values[k] = v
              # 新しいイベントを配列に追加する
              ungrouped_events.append(new_event)
          return ungrouped_events
```

4. gNMIc で解決する(できそうな)課題

4-2：課題2. マルチベンダー対応

例3 :データ整形 – トラフィックレートの追加

```
cache = {}

values_names=[
  '/interface/statistics/out-octets',
  '/interface/statistics/in-octets'
]

N=10

def apply(*events):
  for e in events:
    for value_name in values_names:
      v = e.values.get(value_name)
<SNIP>
      # calculate min, max and avg
      vals = [x[1] for x in val_list]
      if len(val_list) > 1:
        e.values[value_name+"_rate"] = rate(val_list[-2:])
  return events
```

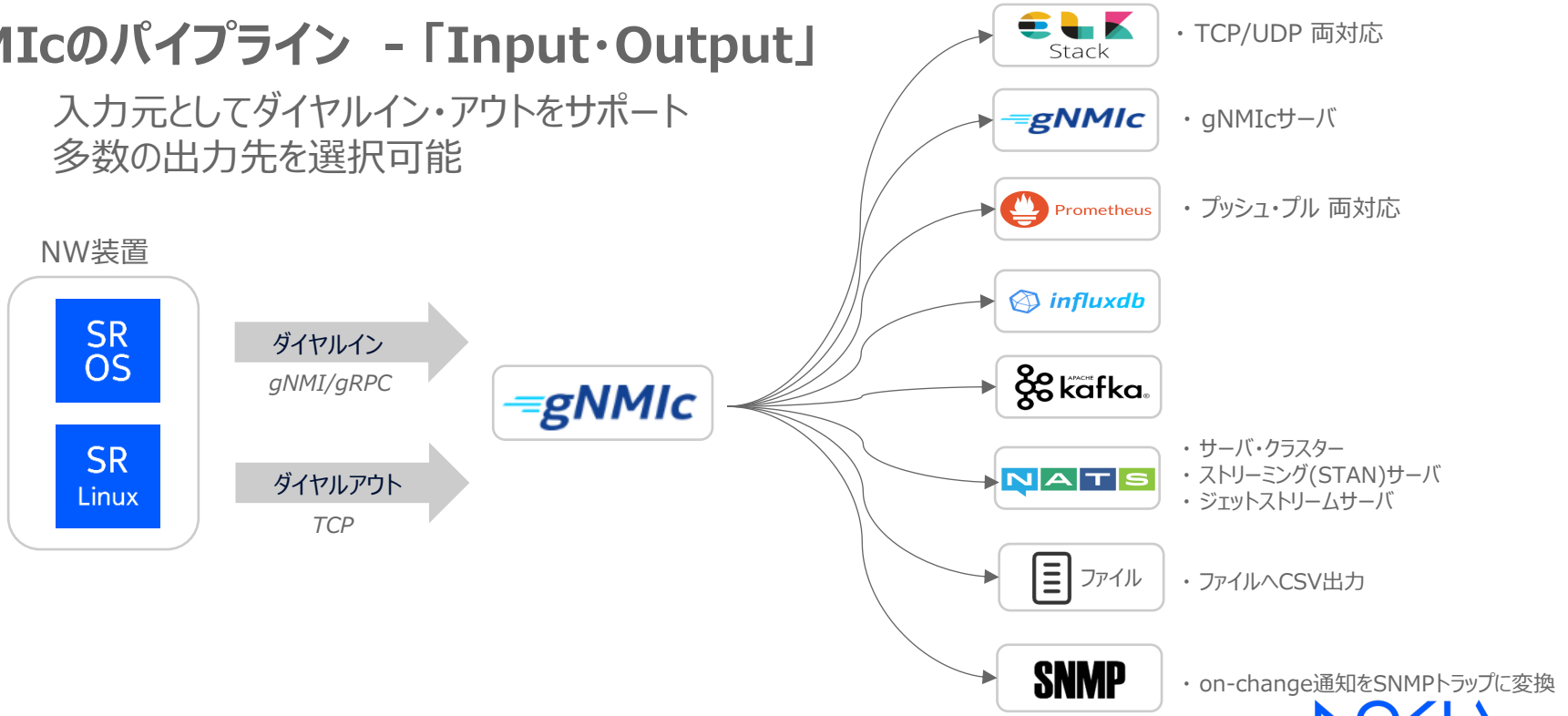
```
def rate(vals):
  period = (vals[1][0] - vals[0][0]) / 1000000000
  change = vals[1][1] - vals[0][1]
  return change / period
```

4. gNMIc で解決する(できそうな)課題

4-2：課題2. マルチベンダー対応

gNMIcのパイプライン　-「Input･Output」

- 入力元としてダイヤルイン･アウトをサポート
- 多数の出力先を選択可能



4. gNMIc で解決する(できそうな)課題

4-2 : 課題2. マルチベンダー対応

gNMIcのパイプライン -「Input･Output」

例: YAML設定ファイルでの記述

```
username: admin
password: admin
port: 57400
timeout: 10s
skip-verify: true
encoding: json_ietf

targets:
  leaf1:57400:
  spine1:57400:
  spine2:57400:

subscriptions:

  srl_if_stats:
    paths:
      - /interface[name=ethernet-1/*]/statistics
    mode: stream
    stream-mode: sample
    sample-interval: 1s

outputs:
  prom:
    type: prometheus
    listen: :9273
    path: /metrics
    metric-prefix: gnmic
    append-subscription-name: true
    export-timestamps: true
    debug: false
    event-processors:
      - trim-prefixes
      - up-down-map
  stdout:
    type: file
    file-type: stdout
    event-processors:
      - proc-convert-strings-to-int
      - trim-prefixes
```

マルチターゲット

YANGパス

複数出力に対応

出力毛データ構造の指定

出力先に対するデータ整形

デバッグ

4. gNMIc で解決する(できそうな)課題

4-3：課題3. システムの障害耐性

課題

・一旦見えてしまうと、見えなくなると困る(怒られる..)

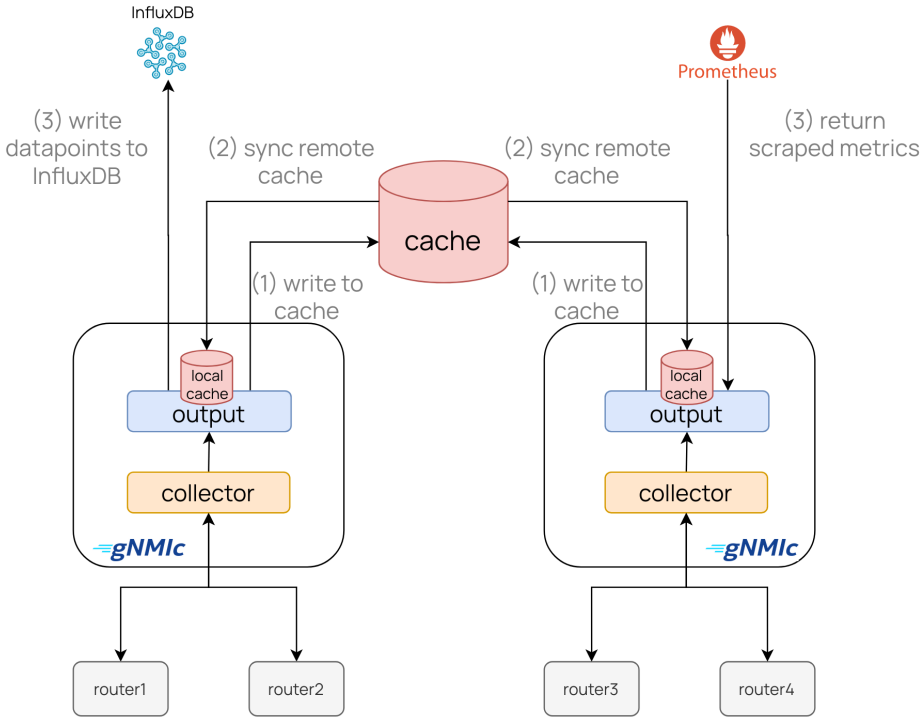
解決方法

・対障害を考慮した分散キャッシュ機能で解決！

4. gNMIc で解決する(できそうな)課題

4-3：課題3. システムの障害耐性

gNMIc クラスタリング機能 (分散キャッシュ)



- ローカルで受信したデータをキャッシュに保存
- キャッシュに保存してあるデータを同期
- サポートしている分散キャッシュ
  - NATS
  - JetStream
  - Redis

※Prometheus/influxdbはローカルキャッシュのみ対応

発表内容

1. 本発表について
2. 背景・課題
3. gNMIc とは？
4. gNMIc で解決する(できそうな)課題
5. デモ
6. まとめ・議論

5. デモ

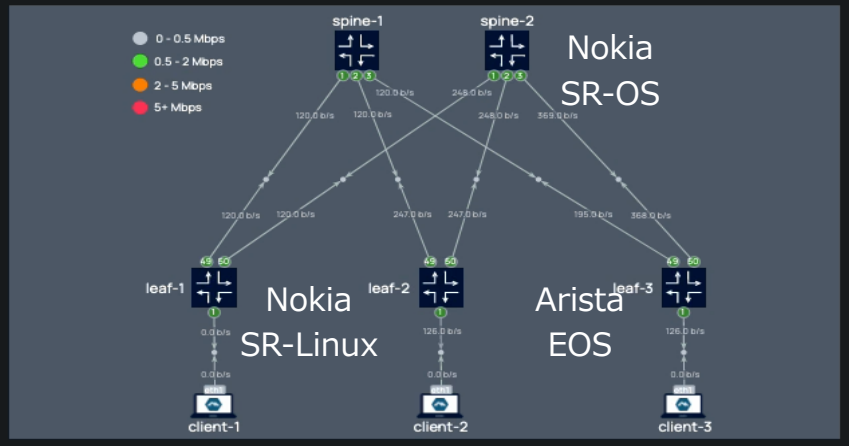
オリジナルデモ構成：SR Linux Streaming Telemetry Lab
→ マルチベンダー化してみました。

Home › Dashboards › SR Linux Telemetry

Add | Last 5 minutes | 5s

NE | spine1 | Services | default | Logs | sr_cli

Link Bandwidth



DC Fabric Throughput

spine1

Routing Stats

11
IPv4 Active Routes

11
IPv4 Total Routes

BGP Peer stats

6
Up Peers

6
Total Peers

0
Disabled Peers

BGP Route Stats

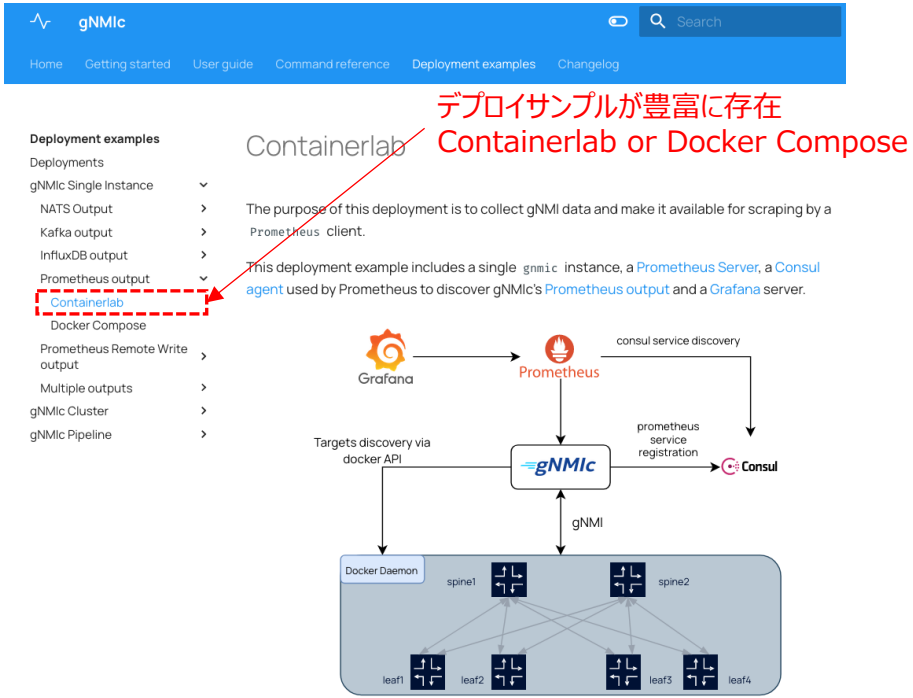
12
Total Received Ro...

6
Total Active Routes

Throughput

5. デモ – 小ネタ

- **gNMIc – Deployment example**



ContainerlabではLinuxノードも動作可能
→ コマンド2つで味見検証を開始！
(※他のNOSもYAML更新でお試し可能)

Deploy it with:

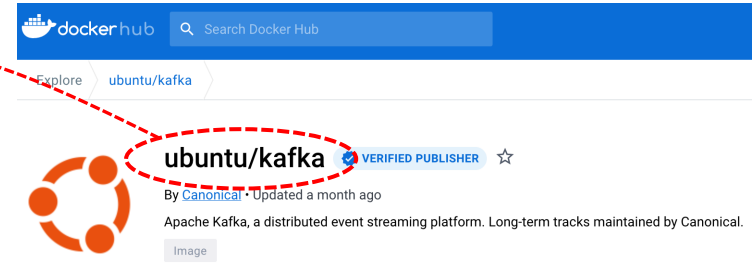
```
git clone https://github.com/openconfig/gnmic.git
cd gnmic/examples/deployments/1.single-instance/1.nats-output/containerlab
sudo clab deploy -t nats.clab.yaml
```

```
+---+--------------------+--------------+-------------------------------+-------+---
| # |        Name        | Container ID |             Image             | Kind  | Gr
+---+--------------------+--------------+-------------------------------+-------+---
| 1 | clab-lab11-gnmic   | 955eaa35b730 | ghcr.io/openconfig/gnmic:latest | linux |
| 2 | clab-lab11-leaf1   | f0f61a79124e | ghcr.io/nokia/srlinux         | srl   |
| 3 | clab-lab11-leaf2   | de714ee79856 | ghcr.io/nokia/srlinux         | srl   |
| 4 | clab-lab11-leaf3   | c674b7bbb898 | ghcr.io/nokia/srlinux         | srl   |
| 5 | clab-lab11-leaf4   | c37033f30e99 | ghcr.io/nokia/srlinux         | srl   |
| 6 | clab-lab11-nats    | ebbd346d2aee | nats:latest                   | linux |
| 7 | clab-lab11-spine1  | 0fe91271bdfe | ghcr.io/nokia/srlinux         | srl   |
| 8 | clab-lab11-spine2  | 6b05f4e42cc4 | ghcr.io/nokia/srlinux         | srl   |
+---+--------------------+--------------+-------------------------------+-------+---
```

5. デモ – 小ネタ

トポロジーYAML

```
  kafka:
    kind: linux
    mgmt_ipv4: 172.16.111.14
    image: ubuntu/kafka:latest
```




```
root@AF01-003:gNMIc$ clab inspect
INFO[0000] Parsing & checking topology file: gnmic-lab.clab.yml
+----+---------------------------+--------------+---------------------------------+----------+---------+-----------------+--------------+
| #  |           Name            | Container ID |              Image              |   Kind   |  State  |   IPv4 Address  | IPv6 Address |
+----+---------------------------+--------------+---------------------------------+----------+---------+-----------------+--------------+
|  1 | clab-gNMI-lab-ceos        | 54a235c324d7 | ceos:4.29.3M                    | ceos     | running | 172.16.111.4/24 | N/A          |
|  2 | clab-gNMI-lab-gnmic       | 4e0477ca6459 | gnmic:0.31.0                    | linux    | running | 172.16.111.12/24| N/A          |
|  3 | clab-gNMI-lab-iperf-client| 533c472d54da | ghcr.io/hellt/network-multitool | linux    | running | 172.16.111.11/24| N/A          |
|  4 | clab-gNMI-lab-iperf-server| 187018a77a14 | ghcr.io/hellt/network-multitool | linux    | running | 172.16.111.10/24| N/A          |
|  5 | clab-gNMI-lab-kafka       | 76e63746db76 | ubuntu/kafka:latest             | linux    | running | 172.16.111.14/24| N/A          |
|  6 | clab-gNMI-lab-prometheus  | d9cf96582545 | prom/prometheus:v2.44.0         | linux    | running | 172.16.111.13/24| N/A          |
|  7 | clab-gNMI-lab-srl         | deb81b236e1f | ghcr.io/nokia/srlinux:22.3.2    | srl      | running | 172.16.111.2/24 | N/A          |
|  8 | clab-gNMI-lab-vr-sros     | 3e13c8dbadfd | vrnetlab/vr-sros:23.3.R1        | vr-sros  | running | 172.16.111.3/24 | N/A          |
|  9 | clab-gNMI-lab-vr-vmx      | d7d070f0f68d | vrnetlab/vr-vmx:18.2R1.9        | vr-vmx   | running | 172.16.111.5/24 | N/A          |
| 10 | clab-gNMI-lab-vr-xrv9k    | e3ac37e3e0b2 | vrnetlab/vr-xrv9k:7.7.2         | vr-xrv9k | running | 172.16.111.6/24 | N/A          |
+----+---------------------------+--------------+---------------------------------+----------+---------+-----------------+--------------+
root@AF01-003:gNMIc$ docker exec -it clab-gNMI-lab-kafka bash
root@kafka:/#
root@kafka:/#
```

※本デモではKafkaは未使用

発表内容

1. 本発表について
2. 背景・課題
3. gNMIc とは？
4. gNMIc で解決する(できそうな)課題
5. デモ
6. まとめ・議論

6. まとめ・議論

まとめ

- 議論されてきた課題
- gNMIc概要
- 従来の課題を解決(できそうな) 機能を紹介

議論・意見交換したい点

- テレメトリ導入の課題となっている点は？
- マルチベンダーでテレメトリ導入されている方いらっしゃいますでしょうか？
- gNMIcで解決できる・できない課題は？

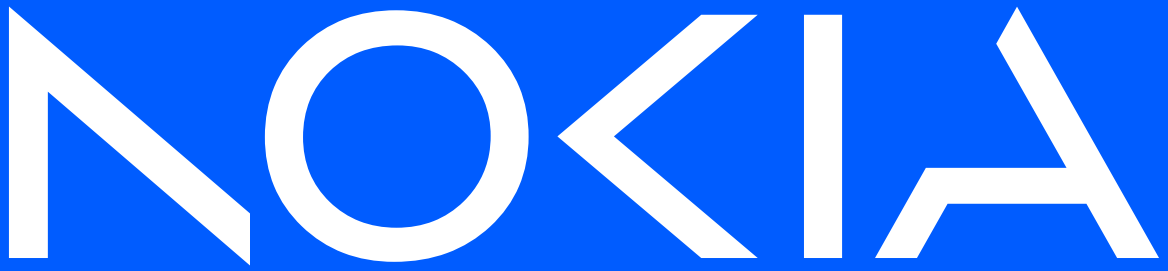
NOKIA

参考サイト

- JANOG Telemetryワーキンググループ
  JANOG-Telemetry-WG-情報整理Task-Group
- NTTコミュニケーションズ ENGINEER BLOG
  次世代の監視技術 – Telemetry技術のご紹介

- OpenConfig – gNMIc
- Discord containerlab #gnmic
- Tutorial: gNMIc - an intuitive gNMI CLI and a feature-rich telemetry collector

NOKIA


NOKIA