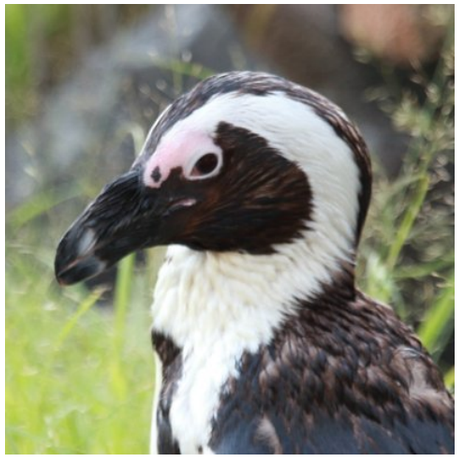
# Prometheusで始めるNetwiserVEの監視 on さくらのクラウド

さくらインターネット 株式会社 仲亀 拓馬

# 自己紹介



- 仲亀 拓馬(@kameneko1004)
- さくらインターネット エバンジェリストチーム
- ストレージ, Kubernetes, Prometheus

## NetwiserVEとは？

- Netwiser Virtual Edition
- セイコーソリューションズ様よりご提供いただいている仮想ロードバランサーアプライアンス
- さくらのクラウドのパブリックアーカイブより展開可能
- 詳細は Netwiser Virtual Edition | さくらのクラウド ドキュメント

## Prometheusとは

- SoundCloudのエンジニアによって開発された監視アプリケーション
  - OSS
  - メトリクス型
  - Pull型
- PromQL
- Golangで書かれており、シングルバイナリで簡単に動かせる
- 動的なターゲットが得意
  - EC2インスタンスのスケールアウト
  - Kubernetes

## Exporter

- Exporter(Agentのようなもの)をサービス毎に展開するアーキテクチャ
  - Node Exporter（サーバのハードウェア）
  - Nginx Exporter
  - Apache Exporter
  - MySQL Exporter
  - Oracle Exporter
  - BlackBox Exporter（エンドポイント監視）

# このセッションの目的

- Prometheusでのネットワーク機器のモニタリングを知る
  - SNMP Exporter

# NetwiserVEをPrometheusでモニタリングする方法

## Prometheusでのターゲット追加

- Exporterの利用が大前提
- Exporterで取得してきたターゲットからラベルを使ってフィルターする
  - すべてのPod → `env:Prod` なラベルのPodだけフィルター
  - すべてのInstance → `Status:Ready` なラベルのInstanceだけフィルター

## Prometheusでのターゲット追加

- まずは「どんなExporterでターゲットの情報を手に入れるか」
- 情報 → **メトリクス**
  - ターゲットの状態（Up/Down）
  - トラフィック
  - プロセッサ/メモリの使用率
  - エラーのカウント
  - etc
- HTTPのプレーンテキストで吐かれる
  - http://[PROMETHEUS-SERVER]:9090/metrics

## どんなExporterを使うか

- SNMP Exporter ← 今回はコッチ
- Custom Exporter

## SNMP Exporter

- SNMPを出力する機器を監視できるExporter
  - ネットワーク機器
  - サーバ
  - ファシリティ
- `snmpwalk` したものをHTTPに変換する
- どのOIDの情報を転送するかコンフィグに書く必要がある
  - SNMP Generatorで生成可能

## Custom Exporter

- **Exporter**を自分で作る
  - めっちゃ難しそう

## Custom Exporter

- HTTPで特定のフォーマットで値が取得できればOK
- 必要なモノ
  - ターゲットのAPIなどを叩く仕組み
  - 取得した値をテキストに変換する仕組み
  - Webサーバ(取得した値を出力)
- ライブラリがサポートされている言語もある
  - Client libraries | Prometheus
    - Go
    - Java or Scala
    - Python
    - Ruby

# 実際に監視をしてみる

## 必要な手順

1. NetwiserVEにSNMPの設定
2. MIBの取得
3. SNMP Exporterのデプロイ
4. コンフィグの生成
5. Prometheusのデプロイ

# NetwiserVEにSNMPの設定

- 前提：インターフェイスの設定がされている

## NetwiserVEにSNMPの設定

```
netwiser> config
netwiser(config)# snmp community public
netwiser(config)# snmp contact contact@example
netwiser(config)# snmp location sakura-cloud
netwiser(config)# snmp host 192.168.1.2
```

- `community` は統一のもの
- `contact` / `location` はなんでもOK
- `host` にSNMP Exporterのサーバ のIP

## NetwiserVEにSNMPの設定

```
ubuntu@monitoring:~$ snmpwalk -v2c -c public 192.168.0.253
iso.3.6.1.2.1.1.1.0 = STRING: "netwiser 3498185076 Netwiser v7.7.10"
iso.3.6.1.2.1.1.2.0 = OID: iso.3.6.1.4.1.955.1.21
iso.3.6.1.2.1.1.3.0 = Timeticks: (43040) 0:07:10.40
iso.3.6.1.2.1.1.4.0 = STRING: "contact@example"
iso.3.6.1.2.1.1.5.0 = STRING: "netwiser"
iso.3.6.1.2.1.1.6.0 = STRING: "sakura-cloud"
iso.3.6.1.2.1.1.7.0 = INTEGER: 76
...
```

- 手動で`snmpwalk`して、問題なく値が取得できればOK
- しかし、このままだとOIDが表示されていてどれが何の値なのかわからない

## MIBの取得

- OID → 項目名に変換するためのファイル
- NetwiserVEのMIBはNetwiser自身から取得可能

## MIBの取得 - CLIでの取得

```
netwiser> export mib tftp
Ready to TFTP send 'mib.zip'.
Press 'q[ENTER]' to cancel: Transfer is completed.
netwiser>
```

- TFTP/ZMODEMが利用可能
  - ZMODEMを使うときは↑のtftpをzmodemに
- SNMP Exporterのサーバから TFTPで取得する
- （今回はなんか上手くいきませんでした

# MIBの取得 - WebGUIでの取得

設定 > システム > 機器情報 > 設定エクスポート > mib.zip


- mib.zipのリンクからファイルをダウンロード可能

## MIBをインポートする

```
/usr/share/snmp/mibs
```

- ↑にzipを解凍したtxtファイルを配置
- 必要に応じて `snmp.conf` の設定変更

```
ubuntu@monitoring:~$ vim /etc/snmp/snmp.conf
# As the snmp packages come without MIB files due to license reasons, loading
# of MIBs is disabled by default. If you added the MIBs you can reenable
# loading them by commenting out the following line.
# mibs :  ← ここをコメントアウトする
```

## MIBをインポートする

コメントアウト前

```
ubuntu@monitoring:~$ snmptranslate -Tp
+--iso(1)
ubuntu@monitoring:~$
```

## MIBをインポートする

コメントアウト後

```
ubuntu@monitoring:~$ snmptranslate -Tp
+--iso(1)
   |
   +--org(3)
      |
      +--dod(6)
         |
         +--internet(1)
            |
            +--directory(1)
            |
            +--mgmt(2)
            |  |
            |  +--mib-2(1)
            |     |
            |     +--system(1)
            |     |  |
            |     |  +-- -R-- String    sysDescr(1)
            |     |  |        Textual Convention: DisplayString
...
ubuntu@monitoring:~$
```

## MIBをインポートする

```
ubuntu@monitoring:~$ snmpwalk -v2c -c public 192.168.0.253
SNMPv2-MIB::sysDescr.0 = STRING: netwiser 3498185076 Netwiser v7.7.10
SNMPv2-MIB::sysObjectID.0 = OID: SNMPv2-SMI::enterprises.955.1.21
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (162867) 0:27:08.67
SNMPv2-MIB::sysContact.0 = STRING: contact@example
SNMPv2-MIB::sysName.0 = STRING: netwiser
SNMPv2-MIB::sysLocation.0 = STRING: sakura-cloud
SNMPv2-MIB::sysServices.0 = INTEGER: 76
```

- 項目名が見えるようになった！
- この項目名からPrometheusで監視したいメトリクスを選別する

## SNMP Exporterのデプロイ

- 方法
  - シングルバイナリ ← 今回はコッチ
  - パッケージマネージャ
  - Dockerコンテナ

## SNMP Exporterのデプロイ

```
ubuntu@monitoring:~$ wget https://github.com/prometheus/snmp_exporter/releases/download/v
ubuntu@monitoring:~$ tar -xvf snmp_exporter-0.15.0.linux-amd64.tar.gz
ubuntu@monitoring:~$ cd snmp_exporter-0.15.0.linux-amd64/
ubuntu@monitoring:~/snmp_exporter-0.15.0.linux-amd64$ ./snmp_exporter
INFO[0000] Starting snmp exporter (version=0.15.0, branch=HEAD, revision=92a3da4467f8bc67
INFO[0000] Build context (go=go1.11.5, user=root@8c3a7c03d455, date=20190212-13:40:02)  s
INFO[0000] Listening on :9116                                source="main.go:226"
```

1. ファイルをダウンロード
2. tarを解凍
3. バイナリを実行

## SNMP Exporterのデプロイ

```
ubuntu@monitoring:~$ curl localhost:9116/metrics
# HELP go_gc_duration_seconds A summary of the GC invocation durations.
# TYPE go_gc_duration_seconds summary
go_gc_duration_seconds{quantile="0"} 6.1525e-05
go_gc_duration_seconds{quantile="0.25"} 6.1525e-05
go_gc_duration_seconds{quantile="0.5"} 7.3834e-05
go_gc_duration_seconds{quantile="0.75"} 9.537e-05
go_gc_duration_seconds{quantile="1"} 9.537e-05
go_gc_duration_seconds_sum 0.000230729
go_gc_duration_seconds_count 3
# HELP go_goroutines Number of goroutines that currently exist.
# TYPE go_goroutines gauge
```

- `localhost:9116` にアクセスしてメトリクスが帰ってくればOK

## SNMP Exporterのデプロイ

- 現在の設定（`snmp.yml`）にはNetwiserの設定は入っていない
- 設定を書く必要があるが、そのために**SNMP Generator**を利用する

## コンフィグの生成

- SNMP Generatorを利用
  - SNMP Exporterの複雑な設定を自動で生成してくれる

## コンフィグの生成

```
if_mib:
  walk:
  - 1.3.6.1.2.1.1.3
  - 1.3.6.1.2.1.2
  - 1.3.6.1.2.1.31.1.1
  metrics:
  - name: sysUpTime
    oid: 1.3.6.1.2.1.1.3
    type: gauge
    help: The time (in hundredths of a second) since the network management portion
      of the system was last re-initialized. - 1.3.6.1.2.1.1.3
  - name: ifNumber
    oid: 1.3.6.1.2.1.2.1
    type: gauge
    help: The number of network interfaces (regardless of their current state) present
      on this system. - 1.3.6.1.2.1.2.1
...
```

- 手で書くとこんなに大変

## コンフィグの生成

```
modules:
  if_mib:
    walk: [sysUpTime, interfaces, ifXTable]
```

- こっから生成してくれる
- Generatorはビルドしなければならない
- またはDockerコンテナ
- Generatorのデプロイ・ビルドは割愛
    - snmp_exporter/generator at master · prometheus/snmp_exporter

## コンフィグの生成

```
ubuntu@monitoring:~$ cat generator.yml
modules:
  netwiser:
    walk:
      - ifOutOctets
      - ifInOctets
    version: 2
    auth:
      community: public
ubuntu@monitoring:~$ snmp-generator generate
INFO[0000] Loading MIBs from $HOME/.snmp/mibs:/usr/share/snmp/mibs:/usr/...
WARN[0000] NetSNMP reported 2 parse errors                source="main.go:103"
INFO[0000] Generating config for module netwiser          source="main.go:49"
INFO[0000] Generated 2 metrics for module netwiser        source="main.go:60"
INFO[0000] Config written to /home/ubuntu/snmp-exporter/generator/snmp.yml  source="main.
```

- `generator.yml`を書いて、`generate`すると`snmp.yml`が作成される

## コンフィグの生成

```
netwiser:
  walk:
  - 1.3.6.1.2.1.2.2.1.10
  - 1.3.6.1.2.1.2.2.1.16
  metrics:
  - name: ifInOctets
    oid: 1.3.6.1.2.1.2.2.1.10
    type: counter
    help: The total number of octets received on the interface, including framing
      characters. - 1.3.6.1.2.1.2.2.1.10
    indexes:
    - labelname: ifIndex
      type: gauge
  - name: ifOutOctets
    oid: 1.3.6.1.2.1.2.2.1.16
    type: counter
    help: The total number of octets transmitted out of the interface, including framing
      characters. - 1.3.6.1.2.1.2.2.1.16
    indexes:
    - labelname: ifIndex
      type: gauge
  version: 2
  auth:
    community: public
```

## コンフィグの生成

- 生成した`snmp.yml`と元の`snmp.yml`を置き換えて、snmp-exporterを再起動

## コンフィグの生成

```
ubuntu@monitoring:~/snmp-exporter$
curl -sS "http://localhost:9116/snmp?target=192.168.0.253&module=netwiser" | grep -v "^#"
ifInOctets{ifIndex="1"} 1.266942e+06
ifInOctets{ifIndex="2"} 2.005737e+06
ifInOctets{ifIndex="3"} 0
ifInOctets{ifIndex="4"} 1.266942e+06
ifOutOctets{ifIndex="1"} 2.725738e+06
ifOutOctets{ifIndex="2"} 2.005685e+06
ifOutOctets{ifIndex="3"} 0
ifOutOctets{ifIndex="4"} 2.725738e+06
snmp_scrape_duration_seconds 0.002488775
snmp_scrape_pdus_returned 8
snmp_scrape_walk_duration_seconds 0.002408014
```

- `ifInOctets`などがインターフェイスのメトリクス
- 今回はインターフェイスのIn/Outのトラフィック量を取得

## Prometheusのデプロイ

```
ubuntu@monitoring:~$ wget https://github.com/prometheus/prometheus/releases/download/v2.9
ubuntu@monitoring:~$ tar -xvf prometheus-2.9.2.linux-amd64.tar.gz
ubuntu@monitoring:~$ cd prometheus-2.9.2.linux-amd64/
ubuntu@monitoring:~/prometheus-2.9.2.linux-amd64$ ./prometheus
level=info ts=2019-06-11T06:29:49.842Z caller=main.go:285 msg="no time or size retention
level=info ts=2019-06-11T06:29:49.842Z caller=main.go:321 msg="Starting Prometheus" versi
level=info ts=2019-06-11T06:29:49.842Z caller=main.go:322 build_context="(go=go1.12.4, us
level=info ts=2019-06-11T06:29:49.842Z caller=main.go:323 host_details="(Linux 4.15.0-45-
level=info ts=2019-06-11T06:29:49.843Z caller=main.go:324 fd_limits="(soft=1024, hard=409
level=info ts=2019-06-11T06:29:49.843Z caller=main.go:325 vm_limits="(soft=unlimited, har
level=info ts=2019-06-11T06:29:49.845Z caller=main.go:640 msg="Starting TSDB ..."
level=info ts=2019-06-11T06:29:49.846Z caller=web.go:416 component=web msg="Start listeni
level=info ts=2019-06-11T06:29:49.850Z caller=main.go:655 msg="TSDB started"
level=info ts=2019-06-11T06:29:49.850Z caller=main.go:724 msg="Loading configuration file
level=info ts=2019-06-11T06:29:49.853Z caller=main.go:751 msg="Completed loading of confi
level=info ts=2019-06-11T06:29:49.853Z caller=main.go:609 msg="Server is ready to receive
```

- バイナリをダウンロード
- 解凍して
- 実行するだけ

## Prometheusのデプロイ

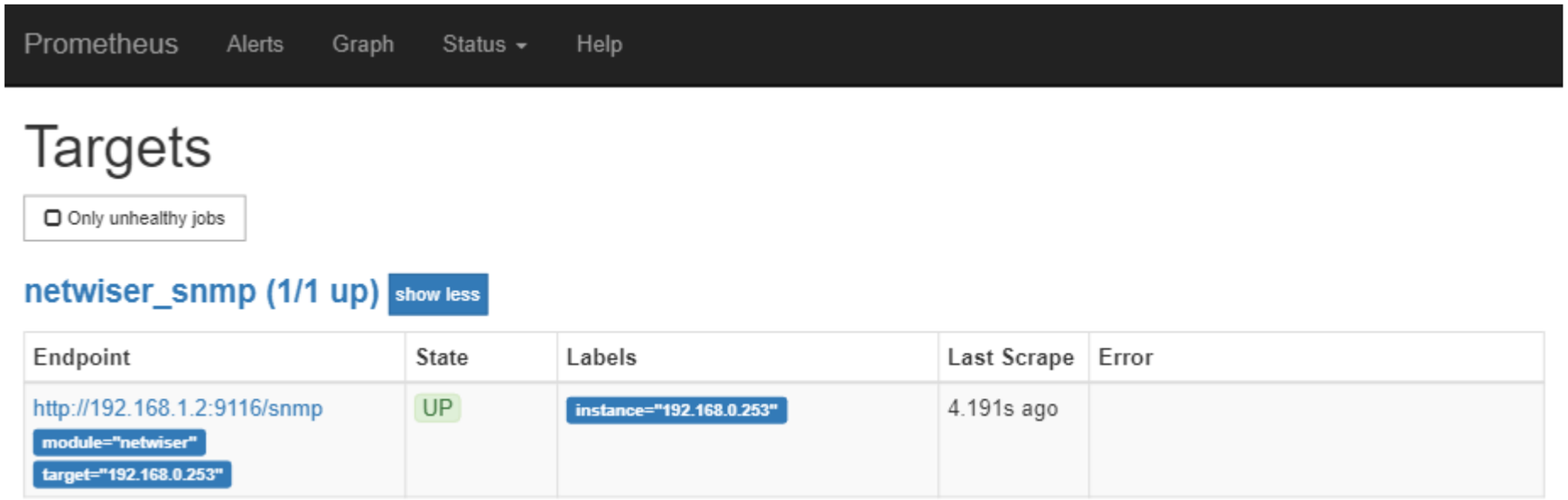
```
ubuntu@monitoring:~/prometheus-2.9.2.linux-amd64$ vim prometheus.yml
global:
  scrape_interval:     15s
  evaluation_interval: 15s
scrape_configs:
  - job_name: 'prometheus'
    static_configs:
      - targets: ['localhost:9090']

  - job_name: netwiser_snmp
    static_configs:
      - targets:
        - '192.168.0.253' # 監視対象を指定
    metrics_path: /snmp
    params:
      module: [netwiser] # Moduleを指定
    relabel_configs:
      - source_labels: [__address__]
        target_label: __param_target
      - source_labels: [__param_target]
        target_label: instance
      - target_label: __address__
        replacement: 192.168.1.2:9116
```

## Prometheusのデプロイ

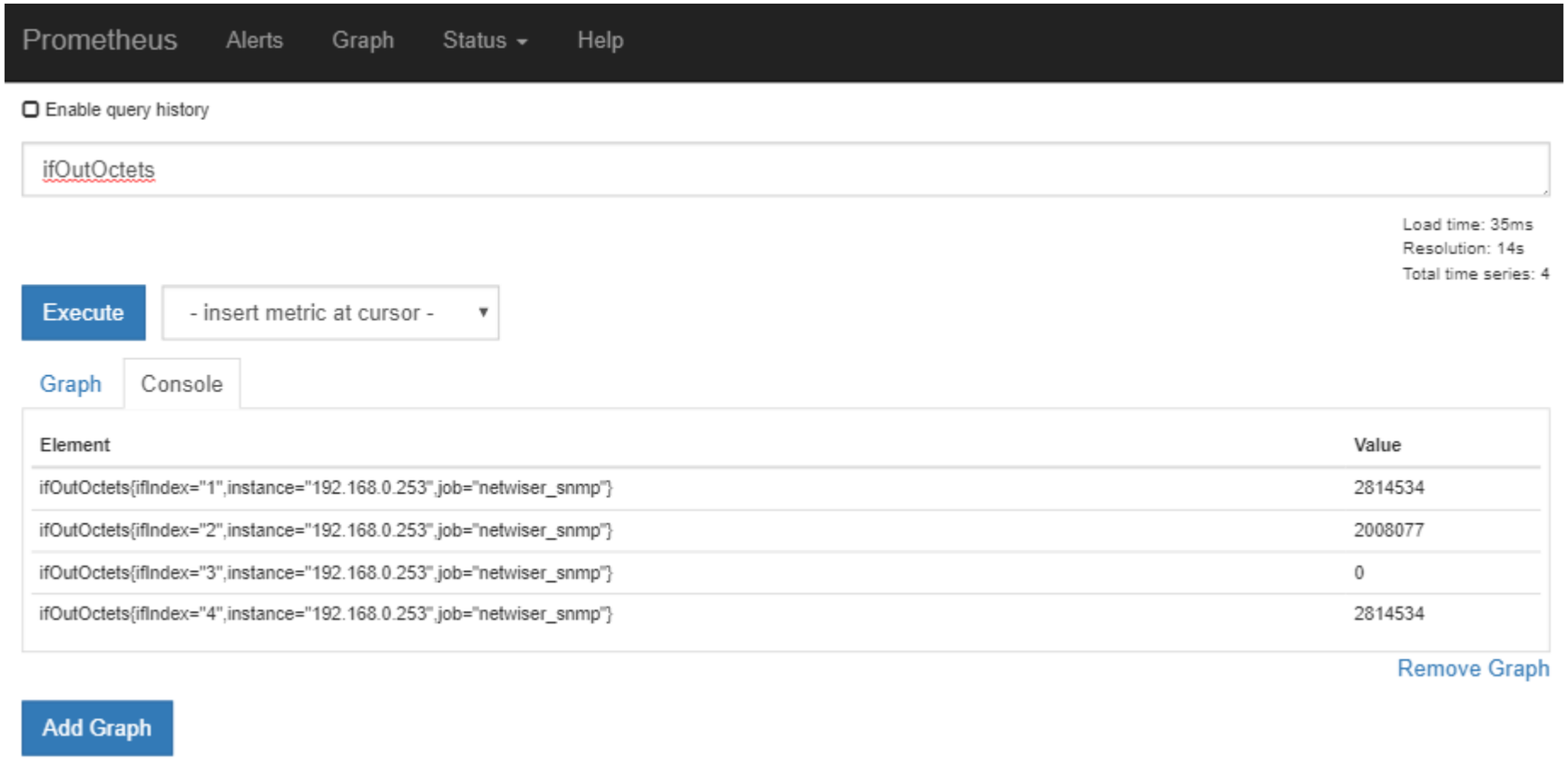
- `targets` にNetwiserのIPを指定する
- `module` にnetwiserのModuleを指定する
- 再起動

# Prometheusのデプロイ




- ターゲットに追加された

# Prometheusのデプロイ




- メトリクスも取得できた

## まとめ

- SNMP Exporterを使うことでNetwiserVEを監視できる
- インターフェイス以外にもSNMPで取得できるものであればOK
- 取得した値からアラートを作成する必要がある