Yahoo! JAPAN Site Operations

ソフトウェアエンジニアがネットワーク部隊に　ていろいろやってみた

ヤフー株式会社 サイトオペレーション本部
インフラ技術３部
安藤 格也

自己紹介

- 安藤 格也(あんどう かくや) @servak
- 2011年入社
- 決済チームで開発、運用
- 2015/10にNWチーム(現職)に異動

- ドラクエ11を楽しみに最近生きてます！

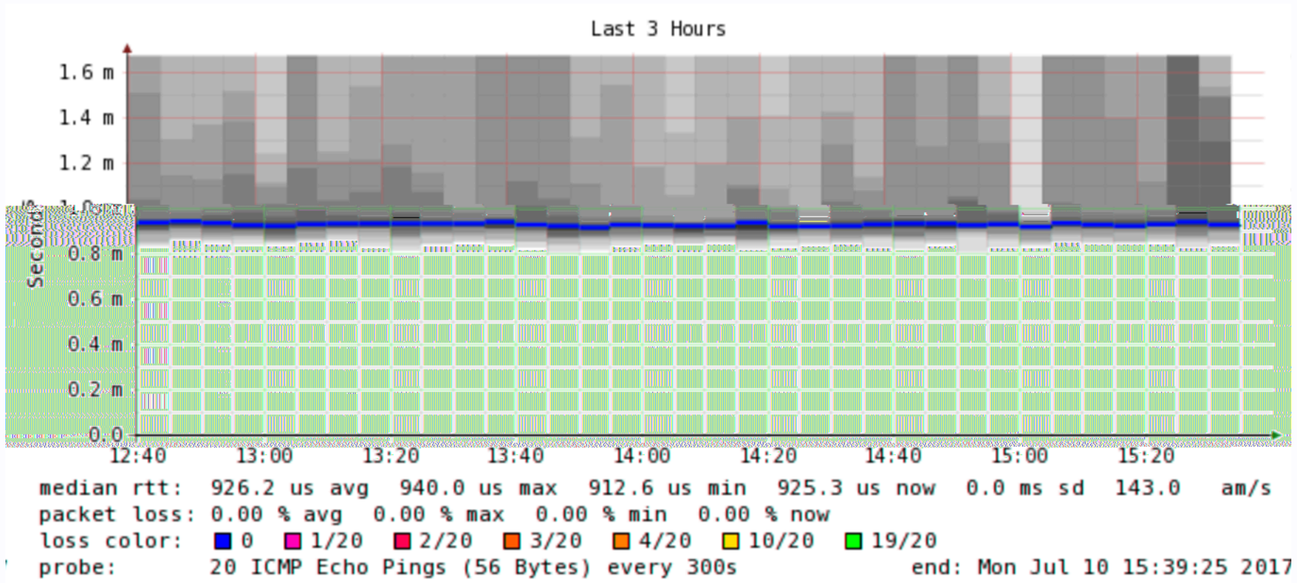
アジェンダ

- 可視化での取り組みについて
- 自動化での取り組みについて
- 可視化と自動化を合わせた取り組みについて
- 今後について

可視化

可視化の状況 (~2016/4)

MRTG
MULTI ROUTER TRAFFIC GRAPHER

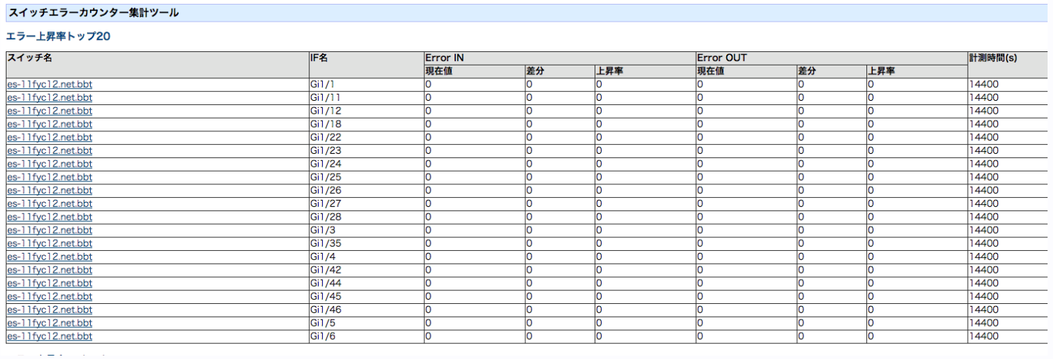


RRDtool
logging & graphing

SiteOPS
Yahoo! JAPAN Site Operations

可視化の状況(~2016/4)

- 問題点
  - MRTGを利用し、5分毎にトラフィックデータを取得
    - より細かく情報を見たい
  - 過去の情報を参照しようとしたときに詳細度が低い
  - トラフィック以外の見たい情報(ifErrorsなど)を見たい人がそれぞれツールを作成し収集し、情報が分散されていた。



スイッチエラーカウンター集計ツール

エラー上昇率トップ20

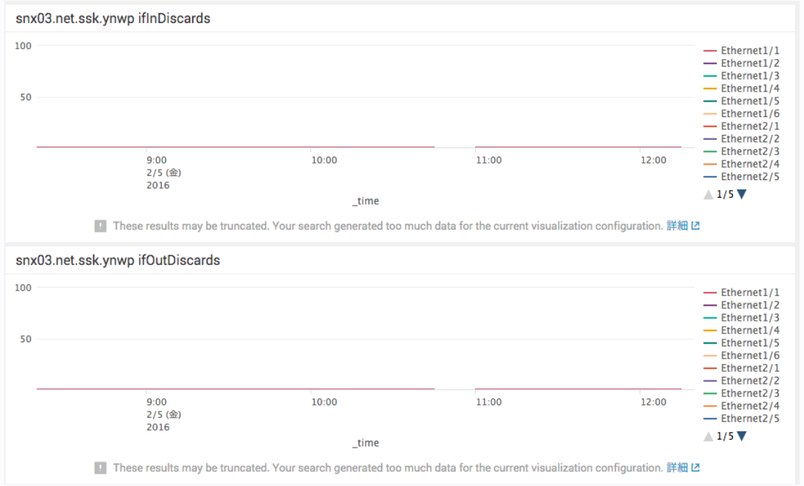
スイッチ名	IF名	Error IN			Error OUT			計測時間(s)
		現在値	差分	上昇率	現在値	差分	上昇率	
es-11fyc12.net.bbt	Gi1/1	0	0	0	0	0	0	14400
es-11fyc12.net.bbt	Gi1/11	0	0	0	0	0	0	14400
es-11fyc12.net.bbt	Gi1/12	0	0	0	0	0	0	14400
es-11fyc12.net.bbt	Gi1/18	0	0	0	0	0	0	14400
es-11fyc12.net.bbt	Gi1/22	0	0	0	0	0	0	14400
es-11fyc12.net.bbt	Gi1/23	0	0	0	0	0	0	14400
es-11fyc12.net.bbt	Gi1/24	0	0	0	0	0	0	14400
es-11fyc12.net.bbt	Gi1/25	0	0	0	0	0	0	14400
es-11fyc12.net.bbt	Gi1/26	0	0	0	0	0	0	14400
es-11fyc12.net.bbt	Gi1/27	0	0	0	0	0	0	14400
es-11fyc12.net.bbt	Gi1/28	0	0	0	0	0	0	14400
es-11fyc12.net.bbt	Gi1/3	0	0	0	0	0	0	14400
es-11fyc12.net.bbt	Gi1/35	0	0	0	0	0	0	14400
es-11fyc12.net.bbt	Gi1/4	0	0	0	0	0	0	14400
es-11fyc12.net.bbt	Gi1/42	0	0	0	0	0	0	14400
es-11fyc12.net.bbt	Gi1/44	0	0	0	0	0	0	14400
es-11fyc12.net.bbt	Gi1/45	0	0	0	0	0	0	14400
es-11fyc12.net.bbt	Gi1/46	0	0	0	0	0	0	14400
es-11fyc12.net.bbt	Gi1/5	0	0	0	0	0	0	14400
es-11fyc12.net.bbt	Gi1/6	0	0	0	0	0	0	14400

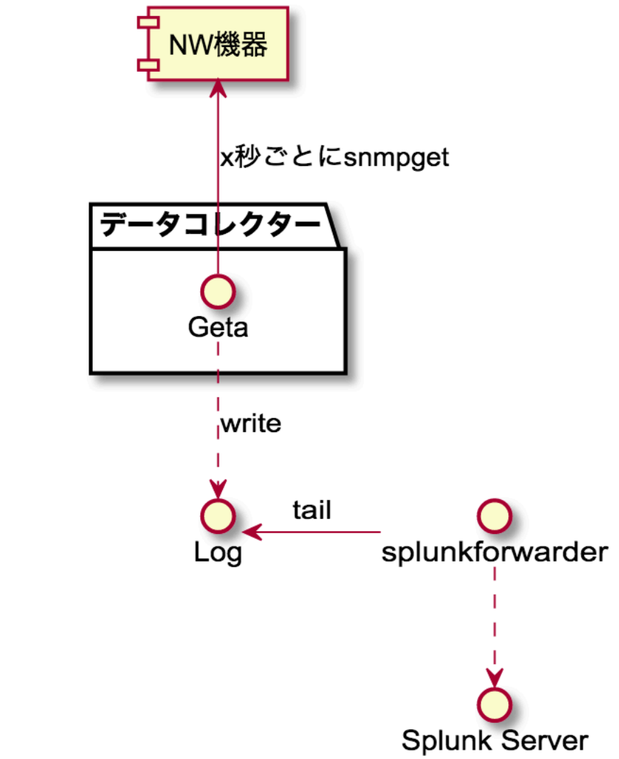
可視化の　件 (~2016/4)

- MRTGを利用し、5分毎にデータを取得
  - 短い範囲で情報を取得し、残すようにする。
- データが散らばってしまう問題
  - データの保存先を集約することを心がけるようにした。

可視化の解決案1（~2016/4）

- ログデータの保存、アラートにSplunk(全文検索エンジン)を利用していたため、Splunkにトラフィック情報を入れるようにしてみた。
- SNMPによるデータコレクター部分は自作
  - 機器へのデータ取得方法をOID毎に細かく設定
  - OID毎に情報の計算方法を変えたかったため



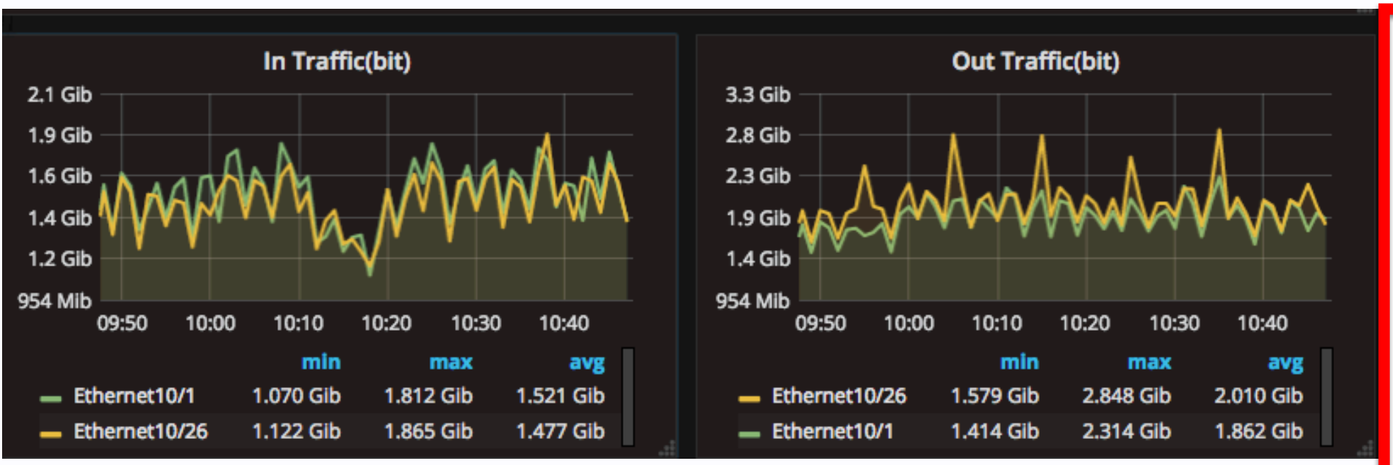


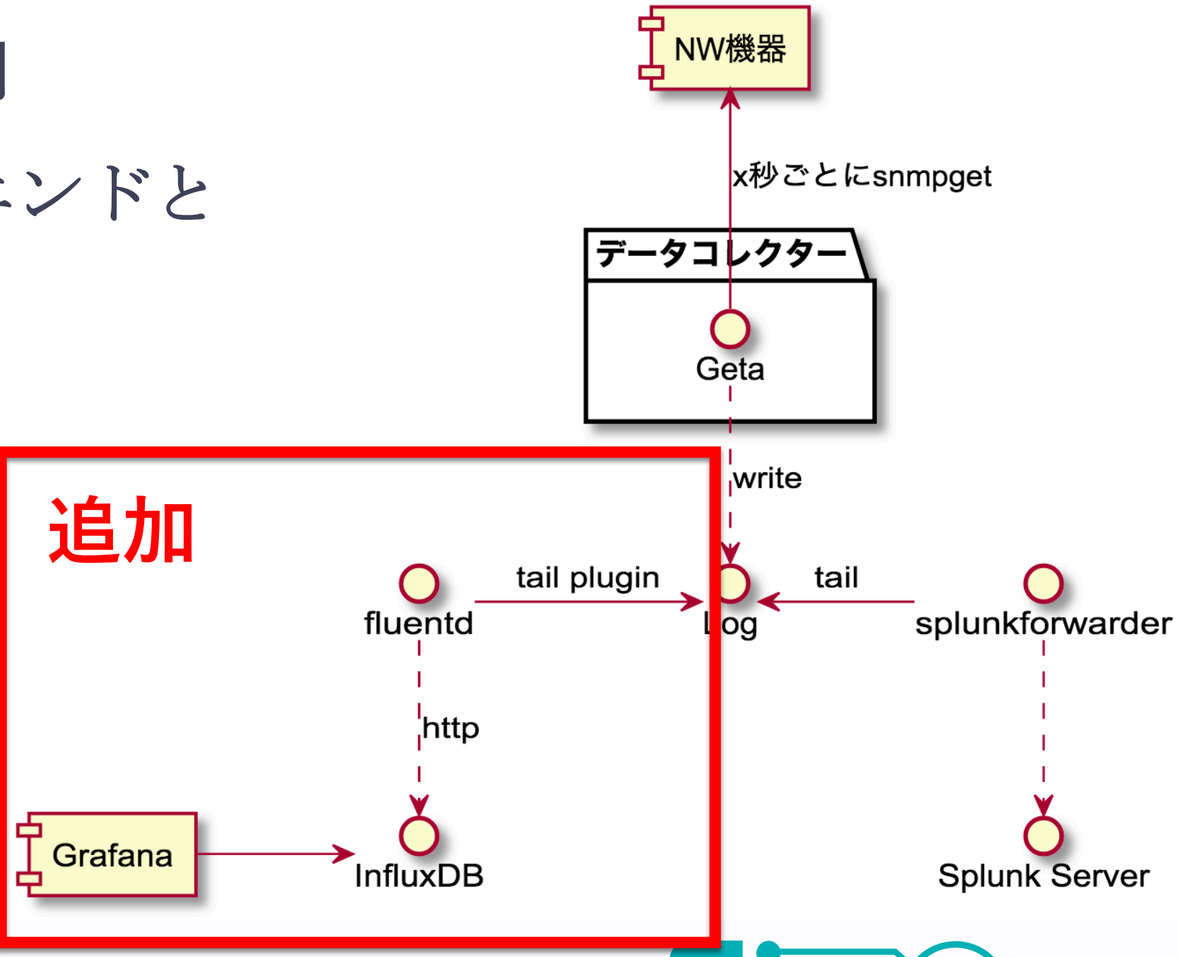
可視化の解決案1 (~2016/4)

- Good
  - Splunkを利用し、分析が行いやすい
  - 5min -> 20sになった
  - アラートを簡単に設定可能
  - 機器への負荷を見ながら、OID毎に取得タイミングを変えられた
- Bad
  - 可視化で描画するインターフェース数が多くなると、表示されない。
  - 個別のインターフェースに注力して見にくい。
  - アラート黙認ができないため、検索クエリでの除外が増えていく
  - どのホストへのデータ取得が失敗しているのか管理しづらい

可視化の解決案2（~2016/4）

- 可視化方法を変更し、Grafanaを利用
  - Grafanaで参照するためのバックエンドとしては時系列DBのinfluxDBを利用





可視化の解決案2 (~2016/4)

- Good
  - Grafanaは使いやすく、カッコイイ！
  - GrafanaのTemplate機能により、個別インターフェースの情報が見やすい
  - ダッシュボードの例が多いため、導入コストも少なくすんだ

可視化の解決案2 (~2016/4)

- Good
  - Grafanaは使いやすく、カッコイイ！
  - GrafanaのTemplate機能により、個別インターフェースの情報が見やすい
  - ダッシュボードの例が多いため、導入コストも少なくすんだ

解決したのはコアスイッチ(X00台)の情報のみ ...
ToRスイッチ(X000台)についても同じように可視化していきたい。

可視化の状況(2016/4~2017/1)

- **ToRスイッチからの情報も取得しようとしたところ問題点が多数**
  - 取得対象の増加とともに…
    - 管理しやすい設計でなかったため、設定更新が難しくなった
    - アラート管理(黙認)を持った機能が欲しくなった
    - **InfluxDBのOSS**からクラスタリング機能がなくなってしまった

可視化の状況(2016/4~2017/1)

- ToRスイッチからの情報も取得しようとしたところ問題点が多数
  - 取得対象の増加とともに…
    - 管理しやすい設計でなかったため、設定更新が難しくなった
    - アラート管理(黙認)を持った機能が欲しくなった
    - InfluxDBのOSSからクラスタリング機能がなくなってしまった

Prometheusを

Prometheusとは

- Pull型(HTTP)のメトリクス監視ツール
  - Inspired by Google’s Borgmon
- Alert管理機能を標準装備
  - Alertを発生させることが出来るし、管理ができる
- 多彩なService Discoveryに対応
  - OpenStack, Kubernetes, StaticFile ...
  - 監視対象を自動的に見つけてくれる
- 公式で様々なメトリクス取得方法を提供
  - snmp_exporter, blackbox_exporter, node_exporter ...

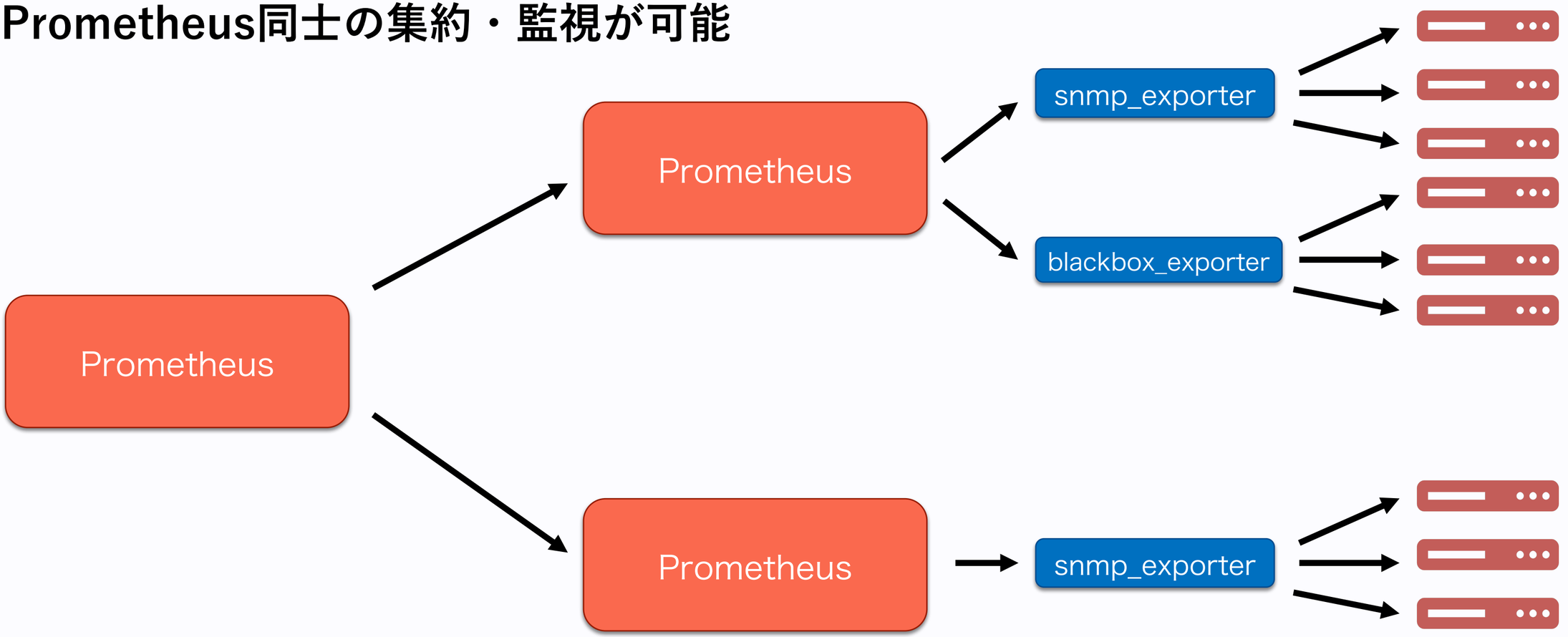
Exporterについて

: snmp_exporter

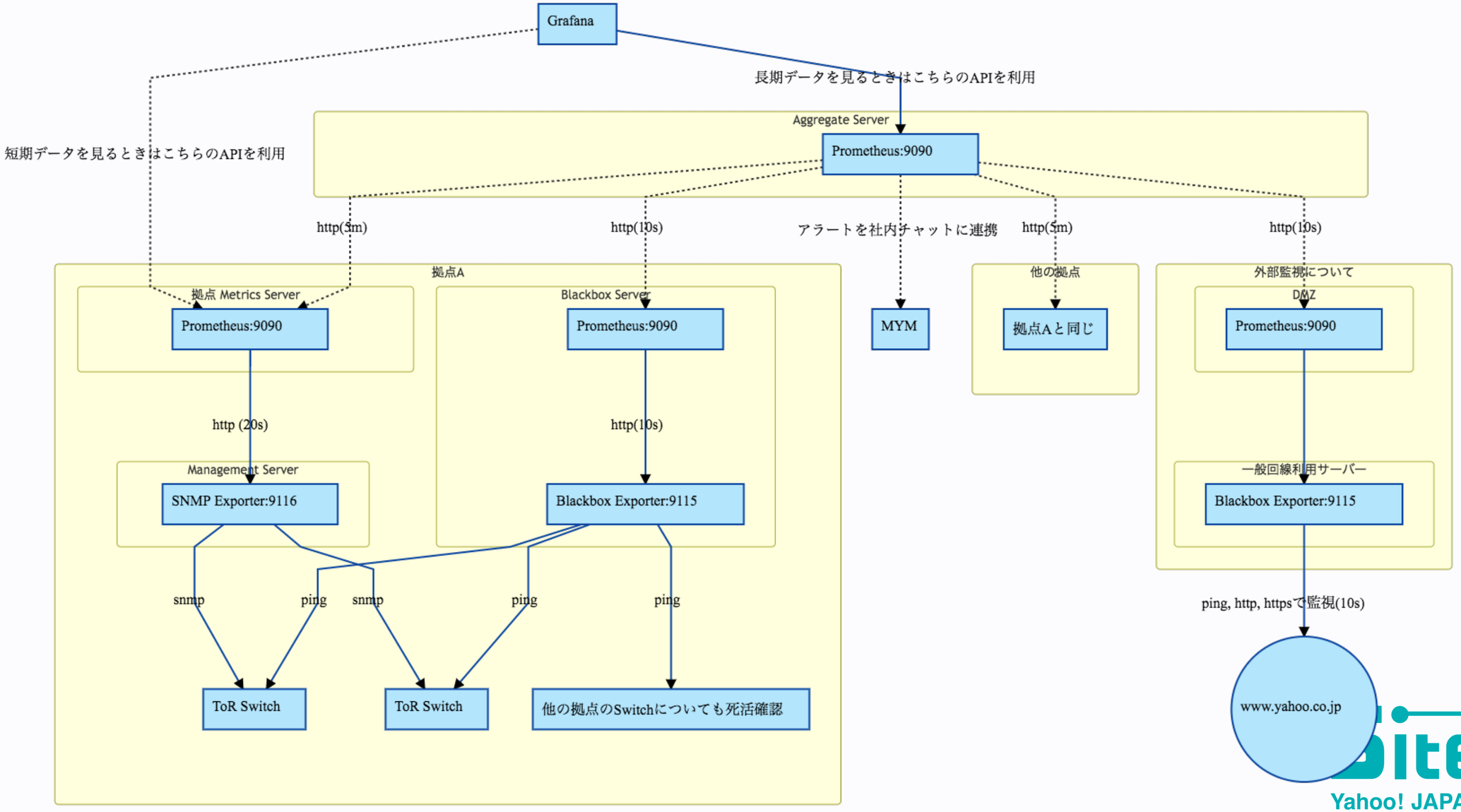
- snmp_exporter
  - SNMPによる情報取得が出来る
- node_exporter
  - *NIXのメトリクスを集めることが出来る
- blackbox_exporter
  - 外部監視をすることが出来る(pingなど)

Prometheusの集約について(Federation)

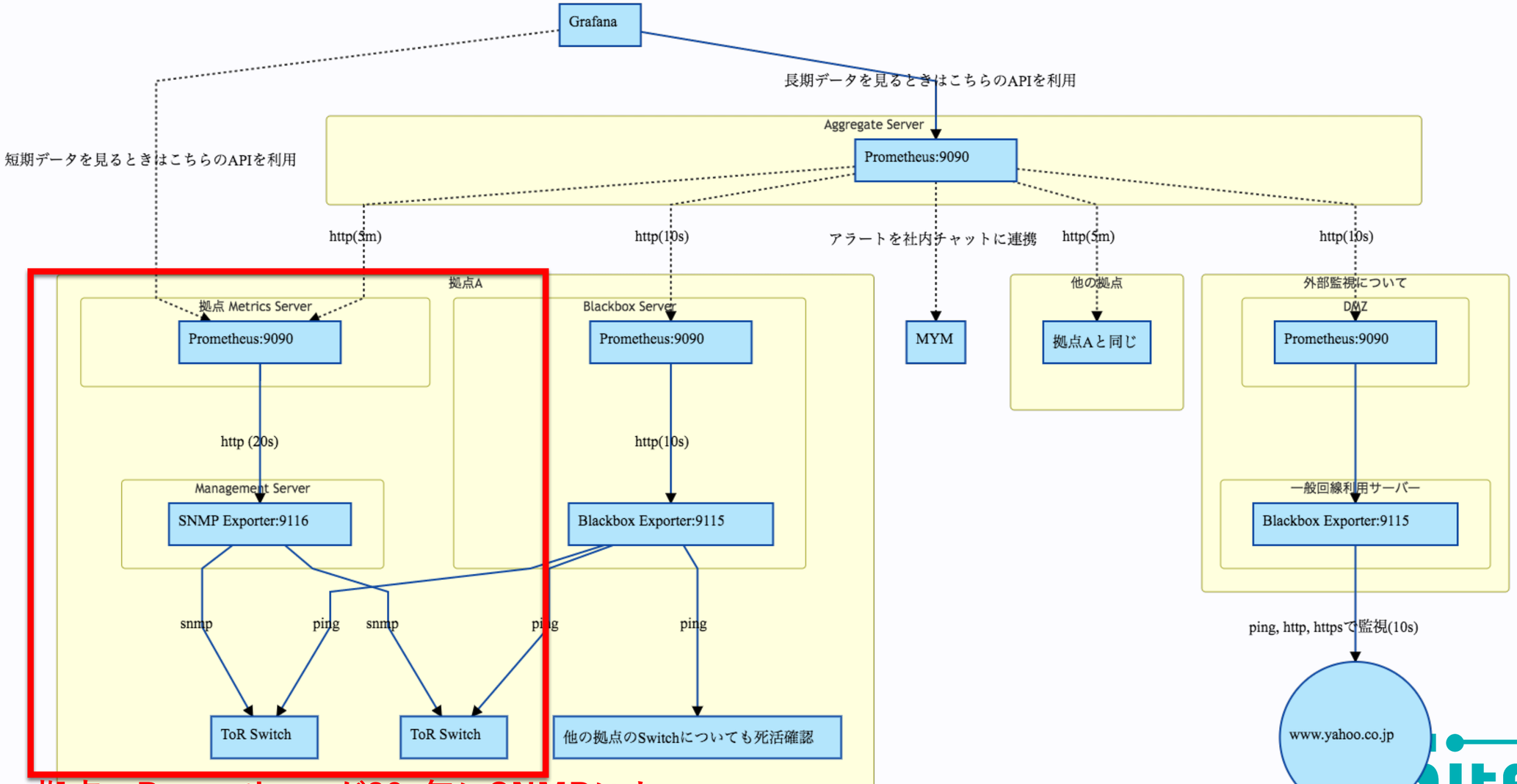
Prometheus同士の集約・監視が可能



Prometheus構成について

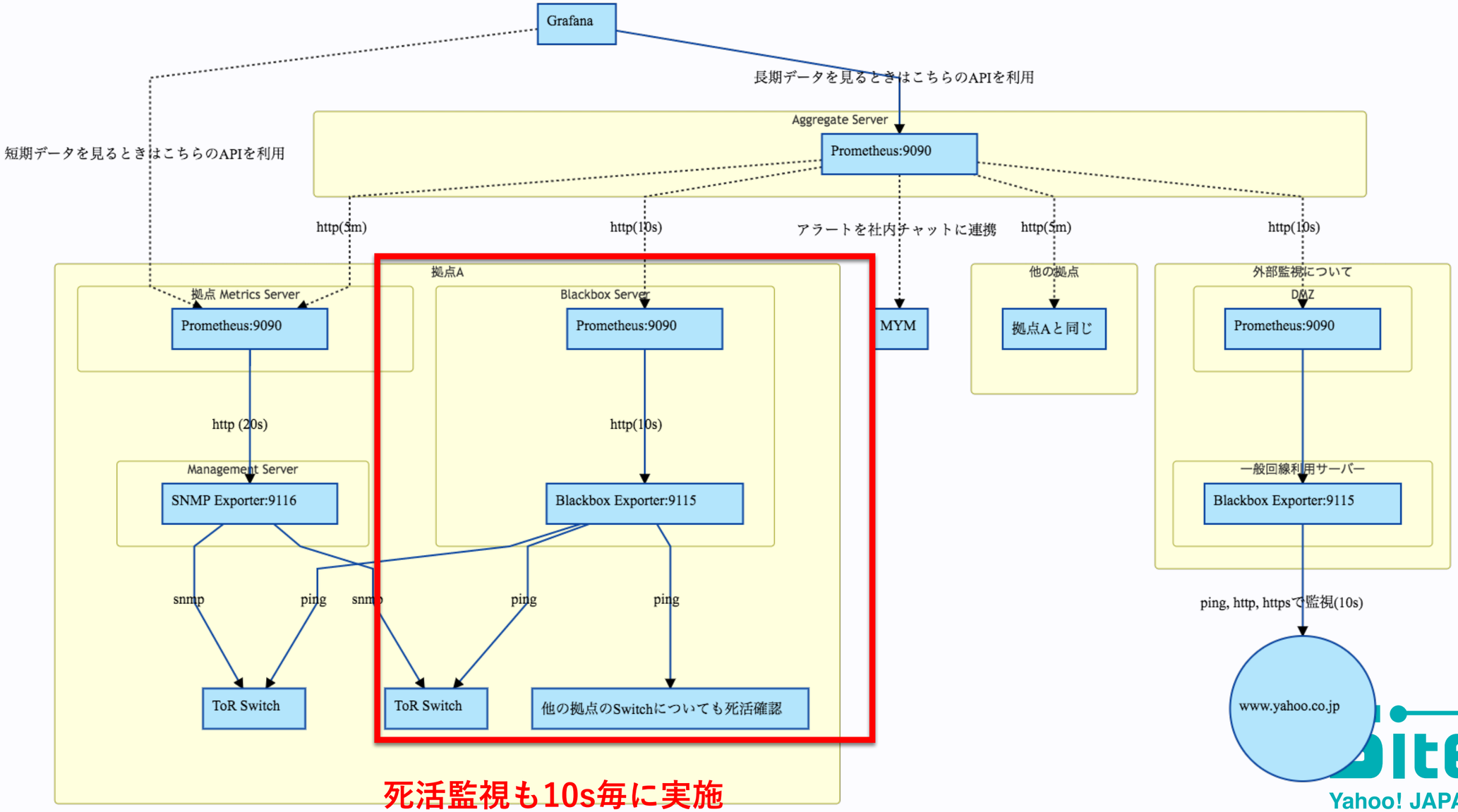


Prometheus構成について



拠点　Prometheusが20s毎にSNMPによりToRスイッチから情報取得

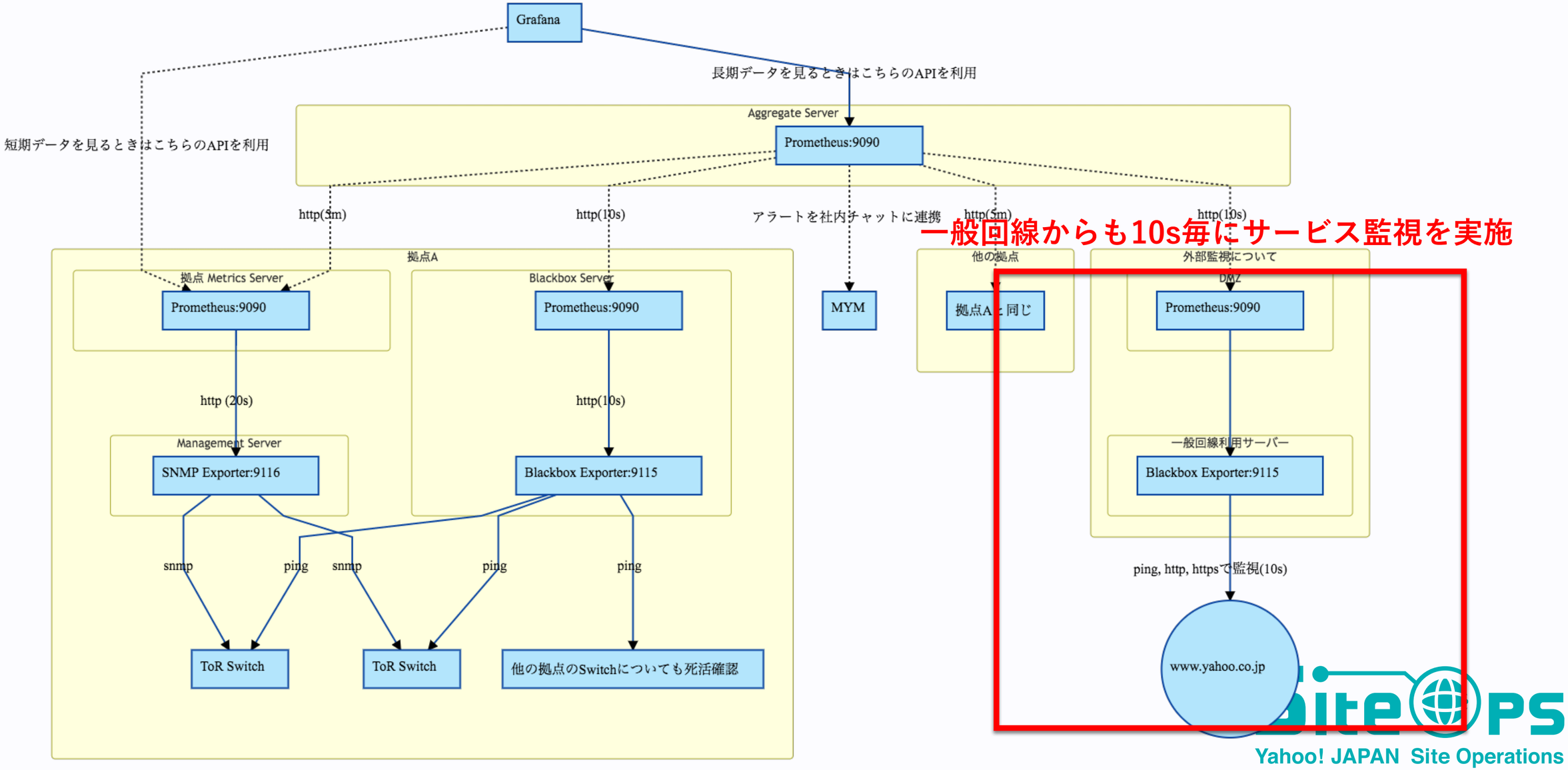
Prometheus構成について



死活監視も10s毎に実施

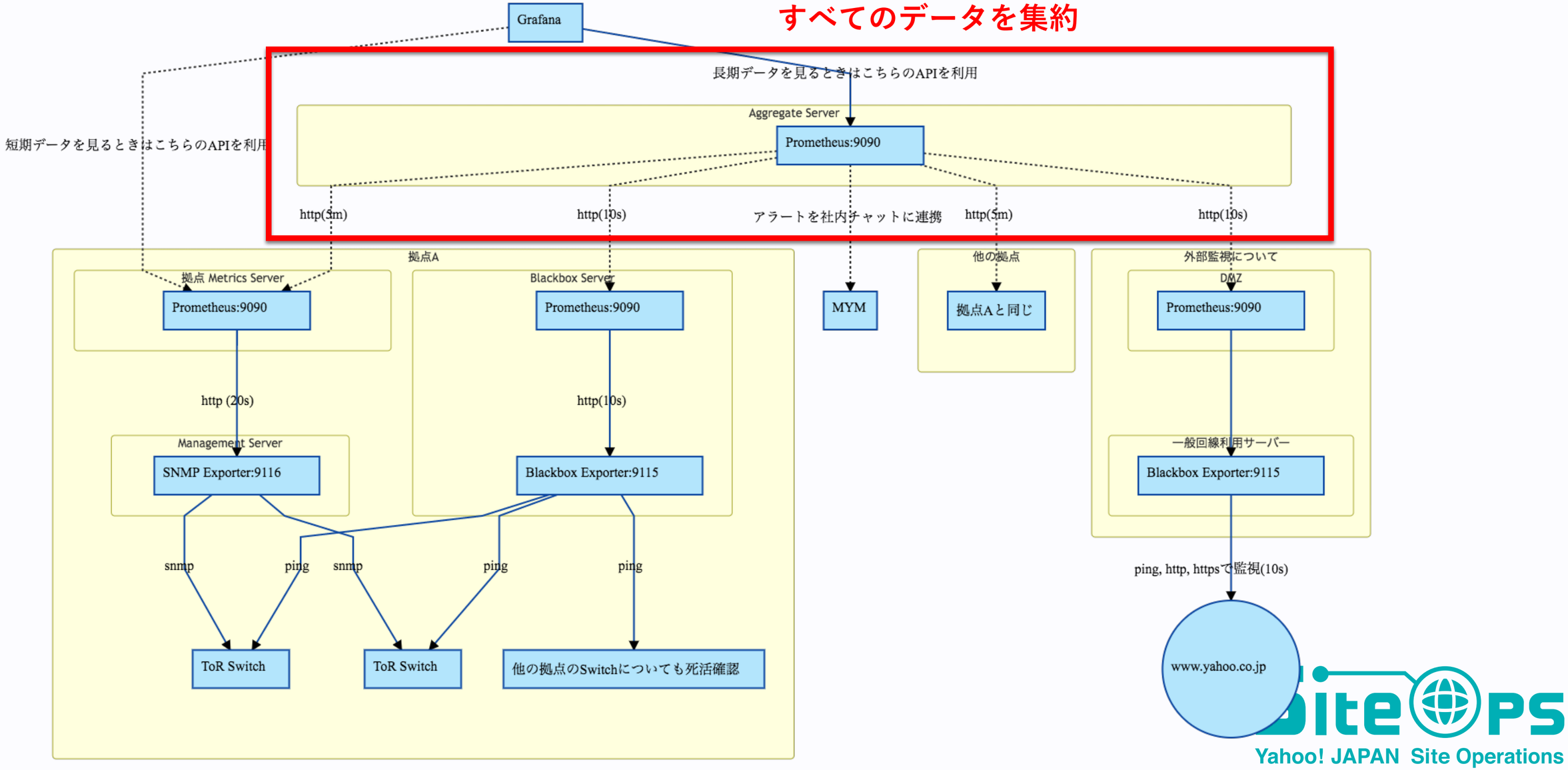
SiteOps
Yahoo! JAPAN Site Operations

Prometheus構成について



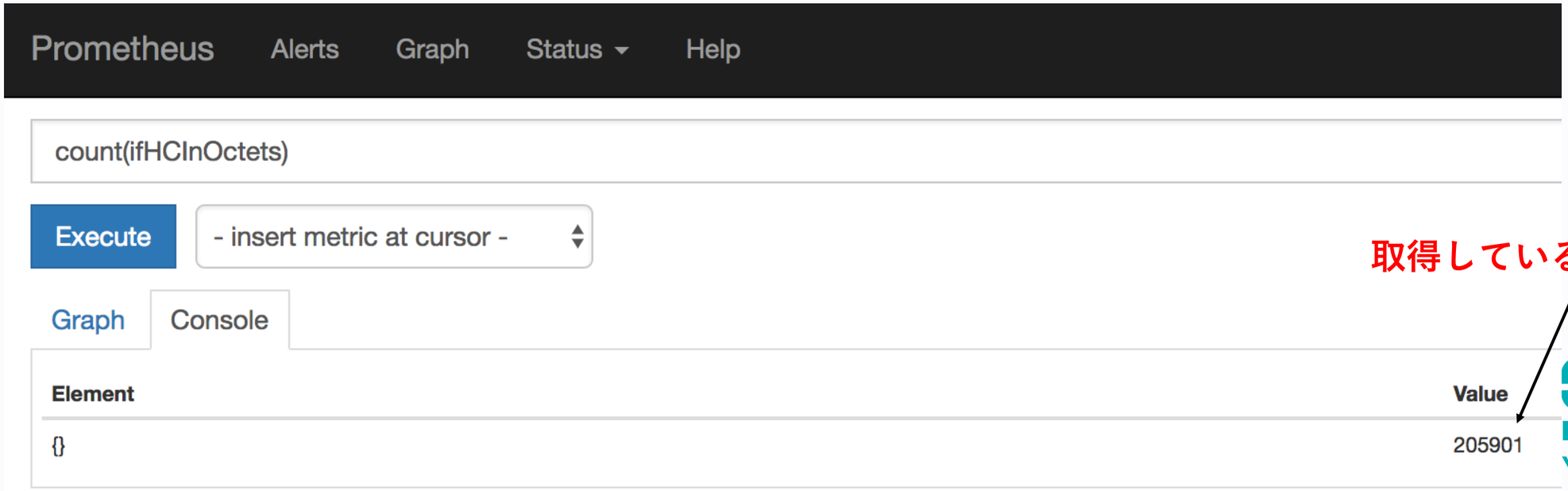
Prometheus構成について

すべてのデータを集約



可視化の解決（2016/4~2017/1）

- すべてのNW機器から情報を取得できるようになった
- 機器が増えても自動で情報取得・監視を実施
  - Jenkinsのジョブを常に動作させておく
- 機器に異常があり、取れなくなったらアラート機能ですぐ連携



Yahoo! JAPAN Site Operations

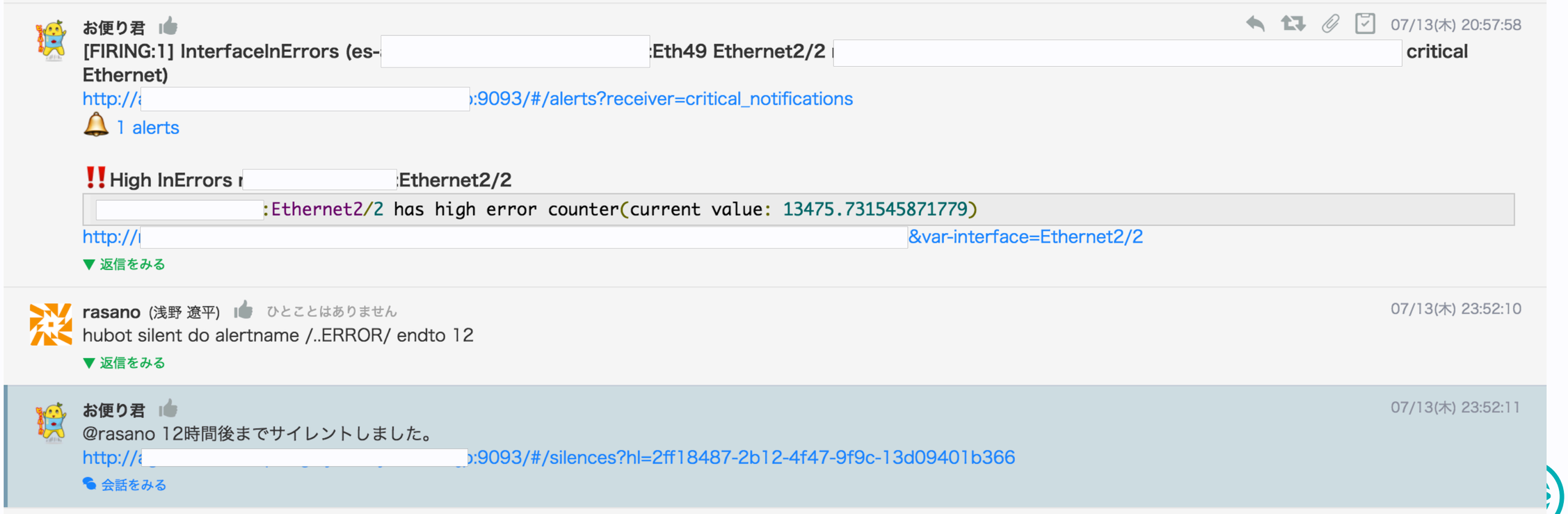
可視化の解決（2016/4~2017/1）

- すべてのNW機器から情報を取得できるようになった
- 機器が増えても自動で情報取得・監視を実施
  - Jenkinsのジョブを常に動作させておく
- 機器に異常があり、取れなくなったらアラート機能ですぐ連携

http://[redacted]:9116/snmp module="default" target="[redacted]"	UP	instance="[redacted]"	12.838s ago	
http://[redacted]:9116/snmp module="default" target="[redacted]"	UP	instance="[redacted]"	19.77s ago	
http://[redacted]:9116/snmp module="default" target="[redacted]"	UP	instance="[redacted]"	12.726s ago	
http://[redacted]:9116/snmp module="default" target="[redacted]"	UP	instance="[redacted]"	3.551s ago	
http://[redacted]:9116/snmp module="default" target="[redacted]"	DOWN	instance="[redacted]"	29.183s ago	context deadline exceeded
http://[redacted]:9116/snmp module="default" target="[redacted]"	UP	instance="[redacted]"	4.021s ago	

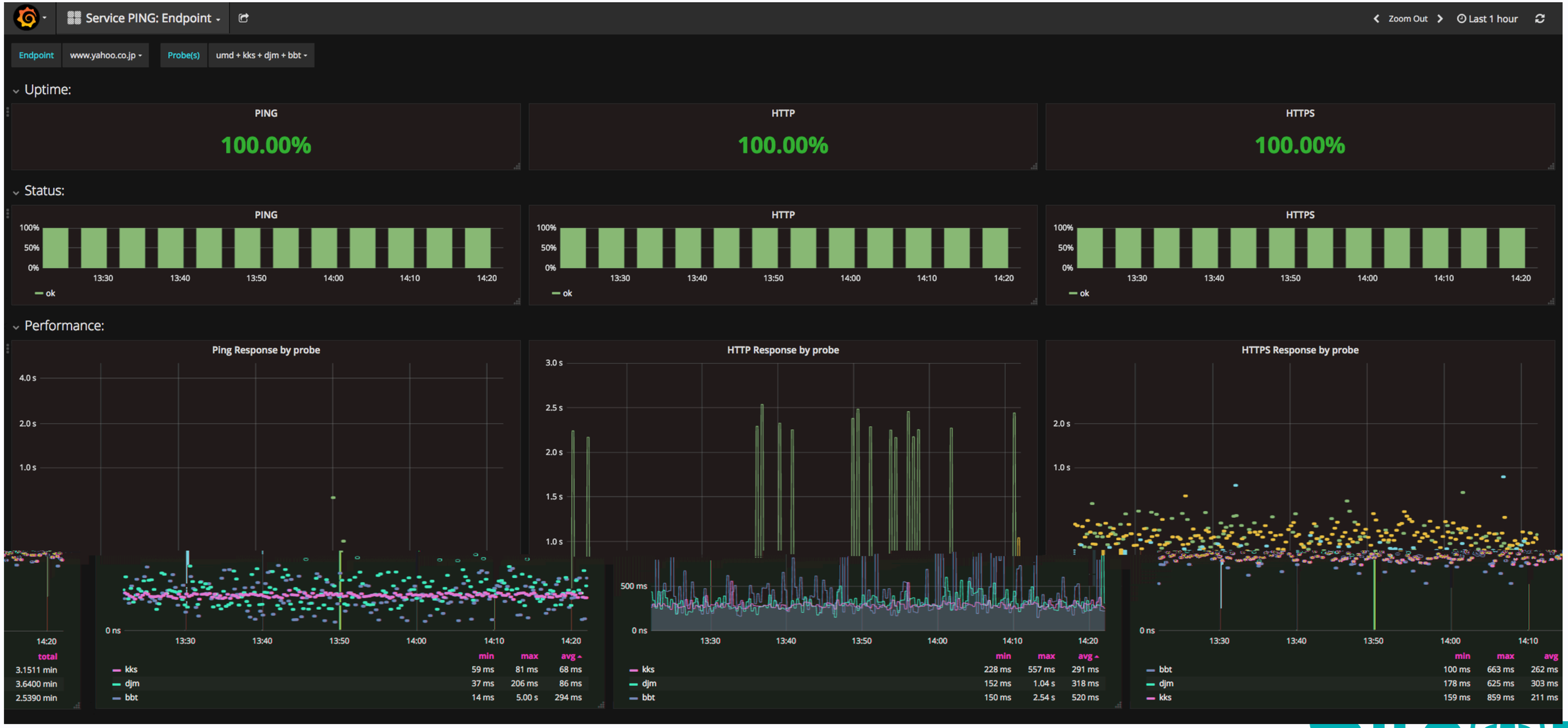
可視化の解決（2016/4~2017/1）

- メトリクスのアラートをチャットに連携し、把握しやすい形に
- 黙認もChatOps(hubot)で行うことで、簡単でわかりやすくしました。



可視化の解決（2016/4~2017/1）

可視化の解決（2016/4~2017/1）



SiteOPS
Yahoo! JAPAN Site Operations

可視化の解決 (2016/4~2017/1)

- Good
  - 監視対象を自動で増やしていくことが出来るので運用が楽
  - アラートを適切に上げることが出来る。
    - 何度もデータ取得に失敗したときだけアラート
    - ifDiscardsやifErrorsの値が何度も一定以上だとアラートなど
  - Exporterを増やすことでメトリクス以外の情報も集めることができた。
    - blackbox_exporterの利用
- Bad
  - Pull型であるため、テレメトリーと相性が悪い。

自動化

自動化の

- 効率化
- ミスの防止
  - 手オペによる運用だと人による作業のためミスが横行
    - 知識の属人化
    - コピペミスなど ...

プログラムから設定を　すようにしよう！

自動化の　（問題点）

- 文化的な話
  - NW運用者たちの中でプログラミングを実施したことある人が少なかった
    - そもそも自動化するためのプログラミング言語もチームで決まっていなかった。
  - ➢ チームのプログラミング言語をPythonに統一
- 技術的な話
  - NW機器を操作するためのAPIがCLIしか使えないNW機器が大多数
    - 一部WebAPI、Netconfを利用できるものもあるが、殆どは利用できない。
  - ➢ まずはOSレベルの方針を決めることに

OS毎のアクセス方法を整

OS	API
Cisco IOS	CLI
Cisco NXOS	Netconf
Juniper JUNOS	Netconf
Arista EOS	EAPI
Brocade IronOS	CLI

CLIが使えると何かと便　なため、すべての機器でCLIは必須　件に

CLIでのアクセスで必　なのは？

- PythonからCLIでNW機器を操作させるために必要なものは？

1. SSHセッションを張ること

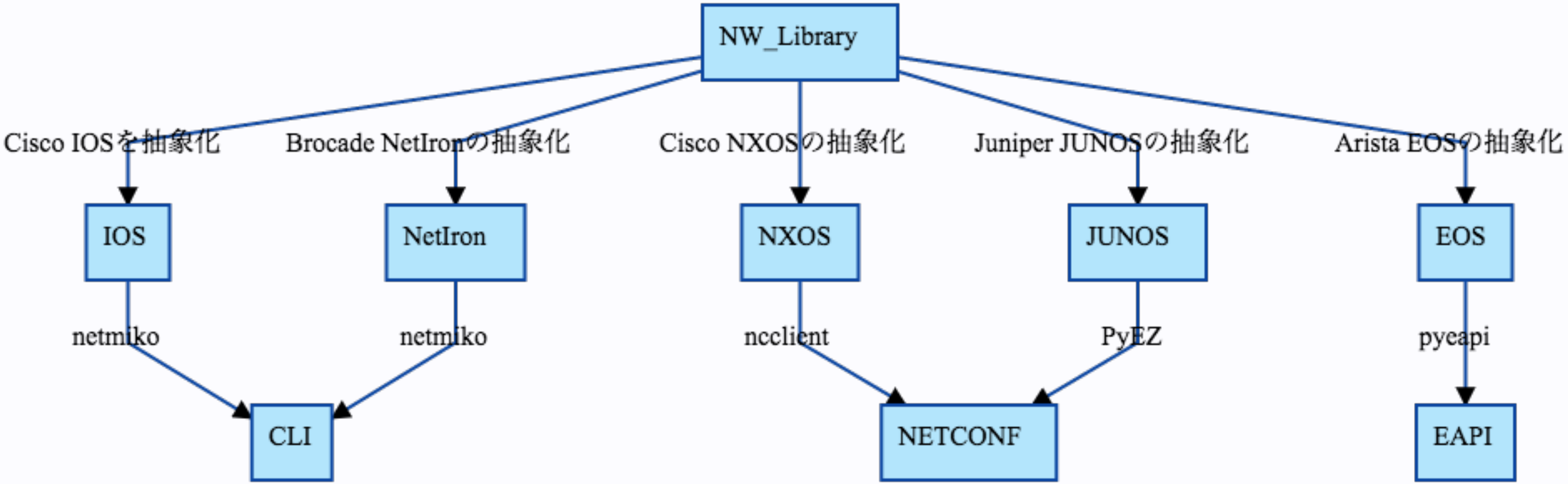
2. CLIでの操作を扱えること
   - コマンド操作後それを待ち、結果を返してくれること
   - 権限(enable, sudo)を上げて操作を行えること

各レイヤー毎に　できそうなOSS(Python)

- SSHの抽象化をしてくれるOSS
  - Paramiko(3513)
- CLIの抽象化をしてくれるOSS
  - netmiko(589)
  - Exscript(270)
    - どちらも主要NWベンダーのOSをサポートしている
- NETCONFの抽象化をしてくれるOSS
  - ncclient(183)

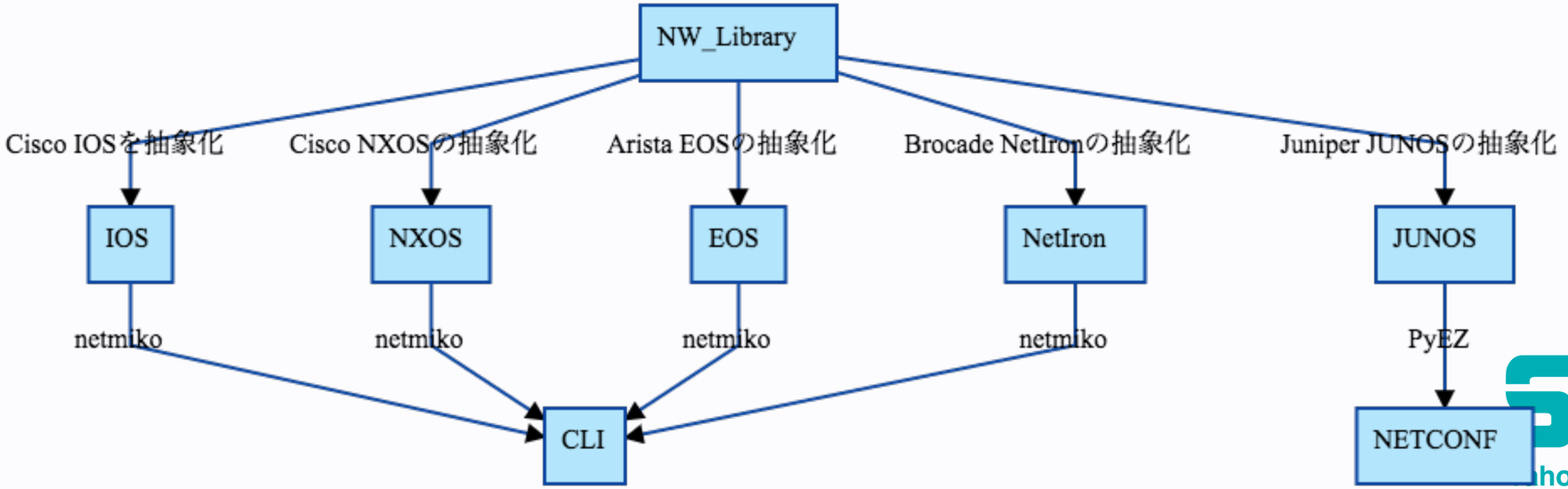
※()の数値はGitHubのスター数(2017/07)

OS毎のアクセス方法を整



しかし多くの問題が。。。

- **やっていくうちに出てくる問題たち(考慮漏れ)**
  - **NETCONF**で取れるデータ != 構造化されたデータ
  - 運用している機器すべてで**WebAPI**が利用できるわけでなかった
    - 運用している機器すべてのバージョンを上げることが難しい
  - などなど ...
- **取得方法を**CLIベースに変更



次のステップへ


```
In [1]: device = session_create(host, user, passwd, community, enable)

In [2]: device.run('show version')
Out[2]: u'Cisco IOS Software, C3750E Software (C3750E-UNIVERSALK9-M), Version 15.0(2)SE4, RELEASE SOFTWARE (fc1)\nTechnical Support: http://www.cisco.com/techsupport\nCopyright (c) 1986-2013 by Cisco Systems, Inc.\nCompiled Wed 26-Jun-13 01:40 by prod_rel_team\n\nROM: Bootstrap program is C3750E boot loader\nBOOTLDR: C3750E Boot Loader (C3750X-HBOOT-M) Version 12.2(58r)SE1, RELEASE SOFTWARE (fc1)\n\nlabcs01.nwlab.bbt uptime is 32 weeks, 2 days, 19 hours, 8 minutes\nSystem returned to ROM by power-on\nSystem restarted at 16:46:16 JST Sat Dec 3 2016\nSystem image file is "flash:/c3750e-universalk9-mz.150-2.SE4/c3750e-universalk9-mz.150-2.SE4.bin"\n\n\nThis product contains cryptographic features and is subject to United\nStates and local country laws governing import, export, transfer and\nuse. Delivery of Cisco cryptographic products does not imply\nthird-party authority to import, export, distribute or use encryption.\nImporters, exporters, distributors and users are responsible for\ncompliance with U.S. and local country laws. By using this product you\nagree to comply with applicable laws and regulations. If you are unable\nto comply with U.S. and local laws, return this product immediately.\n\nA summary of U.S. laws governing Cisco cryptographic products may be found at:\nhttp://www.cisco.com/wwl/export/crypto/tool/stqrg.html\n\nIf you require further assistance please contact us by sending email to\nexport@cisco.com.\n\nLicense Level: ipservices\nLicense Type: Permanent\nNext reload license Level: ipservices\n\ncisco WS-C3750X-24S (PowerPC405) processor (revision K0) with 524288K bytes of memory.\nProcessor board ID FDO1746Z0U1\nLast reset from power-on\n15 Virtual Ethernet interfaces\n1 FastEthernet interface\n28 Gigabit Ethernet interfaces\n2 Ten Gigabit Ethernet interfaces\nThe password-recovery mechanism is enabled.\n\n512K bytes of flash-simulated non-volatile configuration memory.\nBase ethernet MAC Address       : 24:E9:B3:CB:EE:80\nMotherboard assembly number     : 73-13061-05\nMotherboard serial number       : FDO17460JVT\nModel revision number           : K0\nMotherboard revision number     : B0\nModel number                    : WS-C3750X-24S-E\nDaughterboard assembly number   : 800-32727-03\nDaughterboard serial number     : FDO17460NC3\nSystem serial number            : FDO1746Z0U1\nTop Assembly Part Number        : 800-33746-05\nTop Assembly Revision Number    : D0\nVersion ID                      : V04\nCLEI Code Number                : CMMFF00ARD\nHardware Board Revision Number  : 0x05\n\n\nSwitch Ports Model              SW Version            SW Image                \n------ ----- -----              ----------            ----------              \n*    1 30    WS-C3750X-24S      15.0(2)SE4            C3750E-UNIVERSALK9-M    \n\n\nConfiguration register is 0xF\n'

In [3]:
```

- ログイン時にOSを意識する必要性は無くなった
- コマンド、コマンド結果は未だOSを意識する必要が残った

=> 抽象化は不完全

Yahoo! JAPAN Site Operations

共通モデルの定義

- 取得したい内容を共通モデル化
  - コマンド結果の定義化
  - コアとなる考えのみ定義
  - すべてのOSで扱えるもの

class **yj.dcnw.common.models.Interface**(*name, description, state, admin_state, speed*) [ソース]

ベースクラス: `object`

スイッチの物理、論理インターフェースを定義したクラス

パラメータ:
- **name** (*unicode*) – インターフェース名
- **description** (*unicode*) – description
- **state** (*unicode*) – up, downの状態
- **admin_state** (*unicode*) – 設定としてのup, down
- **speed** (*int*) – 速度を定義(bit)

channelgroup

LAGのメンバーであれば、LAGの名前を返す

戻り値の型: unicode or None

戻り値: LAGの名前を返す

in_metrics

IN側のメトリックス情報を返す

戻り値の型: yj.dcnw.common.models.Metric

戻り値: Metricクラスの情報を返却

is_admin_down() [ソース]

インターフェースを設定としてshutにしてるか判定

戻り値の型: bool

共通関数の定義

- 共通で利用する関数を用意
  - 共通モデルを取得できる関数
  - コマンドでのOSの意識を消す

yj.dcnw.common.commands.nxos module

class `yj.dcnw.common.commands.nxos.NXOSCommand`(*host, user, password, enable_password*
[ソース]

ベースクラス: `yj.dcnw.common.commands.command.CommandBase`

Cisco NXOSに対して操作を行うためのクラス

`get_arp()` [ソース]

ARP情報を取得する

戻り値の型: list[yj.dcnw.common.models.Arp]

戻り値: ARP情報を一括で返す

`get_cpuinfo()` [ソース]

CPU情報を取得する

戻り値の型: list[yj.dcnw.common.models.Cpuinfo]

戻り値: CPU情報を一括で返す

`get_feature()` [ソース]

Feature情報を取得する

戻り値の型: list[yj.dcnw.common.models.Feature]

戻り値: 有効機能を一括で返す

`get_interface`(*interface=u」, channelgroup_flg=True*) [ソース]

インターフェース情報を取得する

共通関数の定義

IOS

```
In [1]: device = session_create(host, user, passwd, community, enable)

In [2]: device.get_sysinfo()
Out[2]: <Sysinfo os: cisco_ios, os_version: 15.0(2)SE4>

In [3]: interfaces = device.get_interface()

In [4]: len(interfaces)
Out[4]: 49

In [5]: i = interfaces[16]

In [6]: i.name
Out[6]: u'GigabitEthernet1/0/1'

In [7]: i.description
Out[7]: u'es-11test'

In [8]: i.speed
Out[8]: 1000000000

In [9]:
```

EOS

```
In [1]: device = session_create(host, user, passwd, community, enable)

In [2]: device.get_sysinfo()
Out[2]: <Sysinfo os: arista_eos, os_version: 4.12.8.1>

In [3]: interfaces = device.get_interface()

In [4]: len(interfaces)
Out[4]: 66

In [5]: i = interfaces[0]

In [6]: i.name
Out[6]: u'Ethernet1'

In [7]: i.description
Out[7]: u''

In [8]: i.speed
Out[8]: 10000000000

In [9]:
```

同じ方法で、色んなOSから情報が取得できるように！

メンテナンスの自動化

- メンテナンスで行う内容
  - トラフィック寄せ(OSPF, VRRP)
  - インターフェースのダウンアップ

- YAMLファイルに上記の内容を記載することでその状態にしてくれるコマンドを作成
  - 設定を流すだけでなく、**Before, Afterの状態を確認**する
- YAMLファイルも機器から情報を取得し**自動生成**

メンテナンスの自動化

- 行いたいことを記載することで**OSを意識する必要なく**、インターフェースのup/downを実施することが出来るように！

janog.yml

```
1  - name: L2インターフェースを上げ下げする
2    target: minitest[redacted].yahoo.co.jp
3    tasks:
4    - interface: Ethernet2/19
5      memo: 元々上がっているから何もしない
6      state: up
7    - interface: Ethernet2/19
8      memo: インターフェースをshutする
9      state: down
10   - interface: Ethernet2/19
11     memo: インターフェースをupする
12     state: up
```

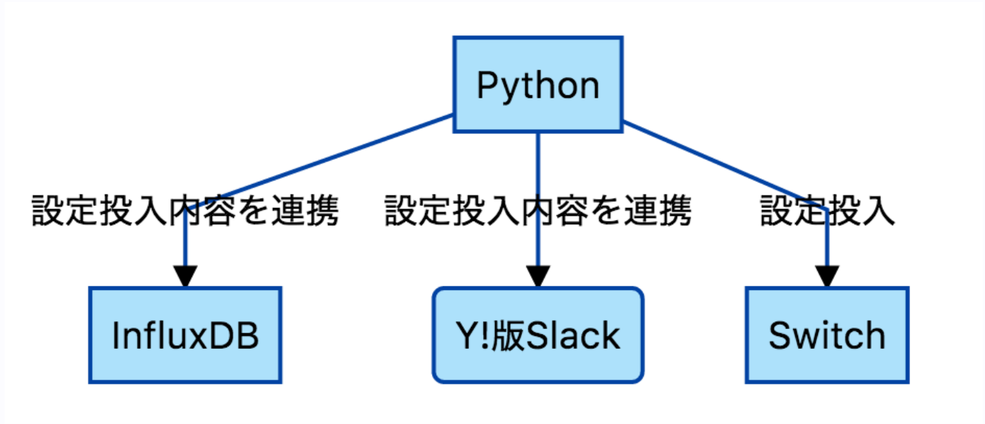
```
> otome switch playbook janog.yml --batch
SETUP ********************************
Connecting ALL Host...  [##############################
PLAY [L2インターフェースを上げ下げする] minitest[redacted]
OK: [interface Ethernet2/19] is already up
CHANGED: [interface Ethernet2/19] state: up => down
CHANGED: [interface Ethernet2/19] state: down => up
```

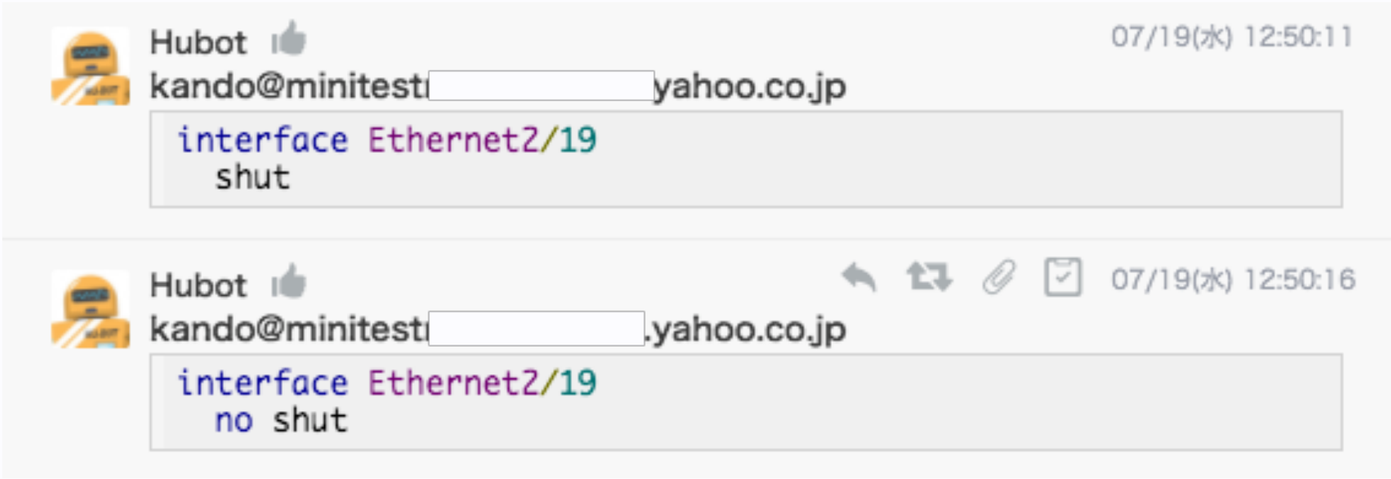
SiteOPS
Yahoo! JAPAN Site Operations

自動化 × 可視化

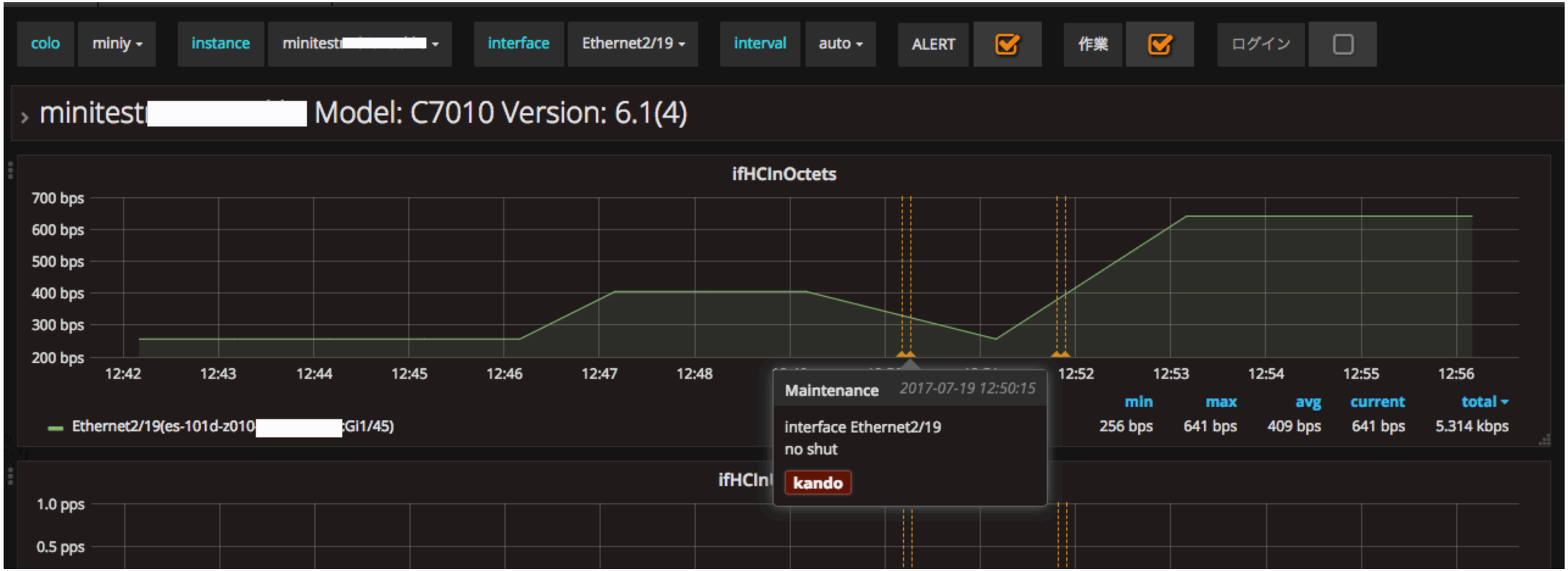
自動化の恩恵(オペレーションの可視化)

- 自動化により、設定投入がプログラムから行えるようになった為
  - 設定投入を可視化できるように





自動化の恩恵(オペレーションの可視化)



SiteOps
Yahoo! JAPAN Site Operations

デモ

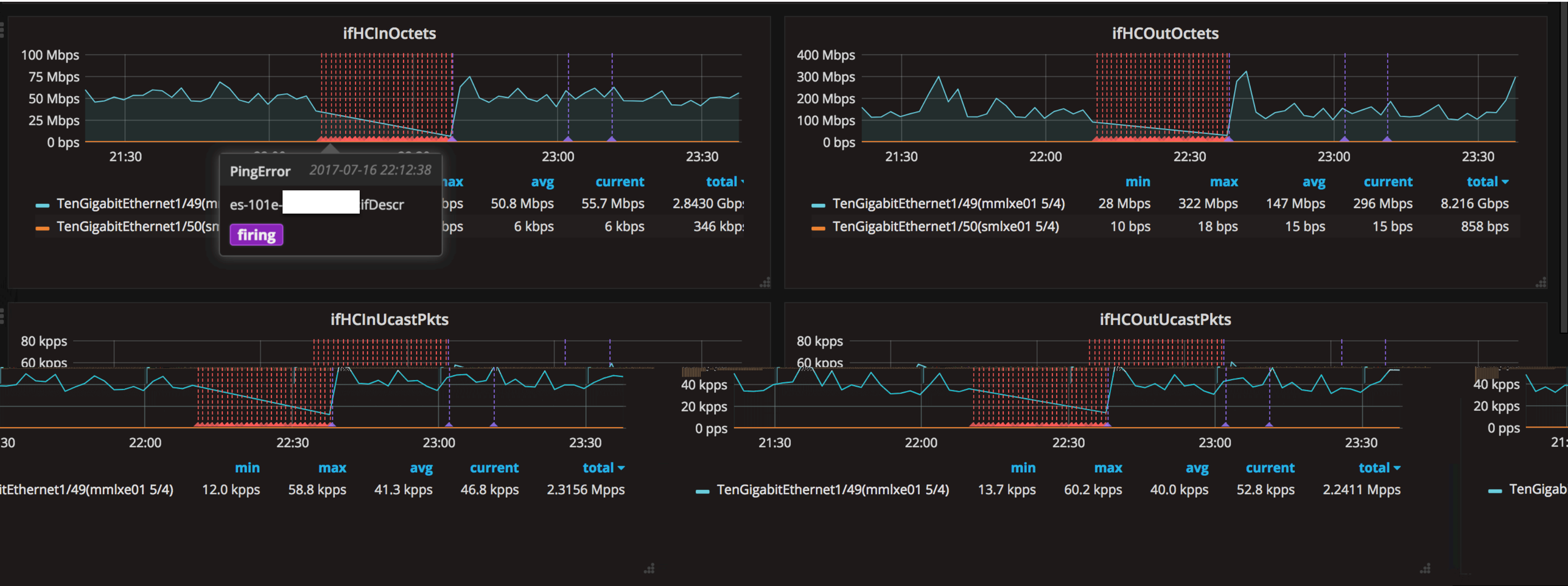
自動化の恩恵(メトリクスの詳細化)

- 自動化によりCLIから詳細情報を取得することが可能に
  - 既存のSNMPで取れていた情報はもちろん
  - モジュール
  - LLDP
  - SFPの光レベル
  - 温度など

自動化の恩恵(メトリクスの詳細化)

アラート×可視化

- Prometheusとも連携することでアラート発生とメトリクスの紐付けも行った。



今後について

◆今後やっていきたいこと

- **Telemetry**情報の活かし方を考えていく
- オペレーションの自動化をより推進する

ご清聴ありがとうございました

Yahoo! JAPAN Site Operations