SRA OSS, INC.

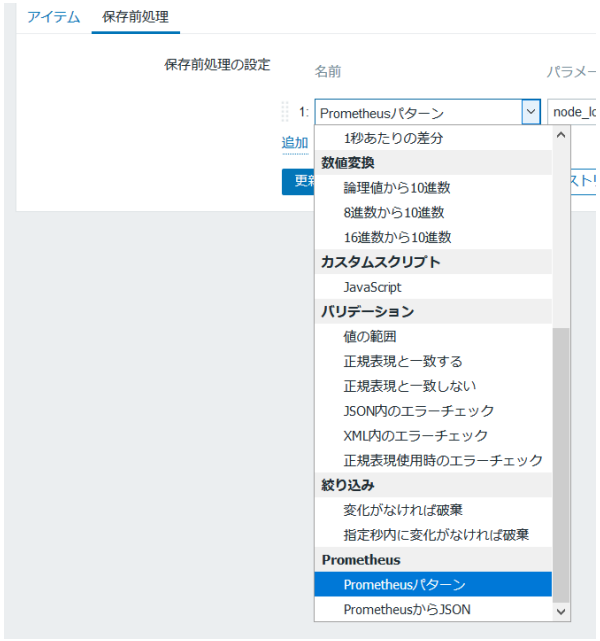
Zabbix 最新情報セミナー

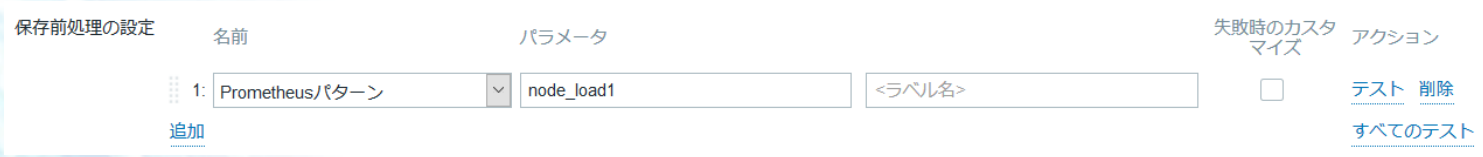
Prometheus を利用した Zabbix 応用監視

2019/7/10

Zabbix 4.2の新機能

アイテムの保存前処理で Prometheus との連携が可能に





内容

1. Prometheus？
2. Zabbix 4.2 Prometheusとの連携
3. Zabbix と Alertmanagerの連携

Alertmanagerを利用して重複通知の削除

Prometheus ?

Prometheus?

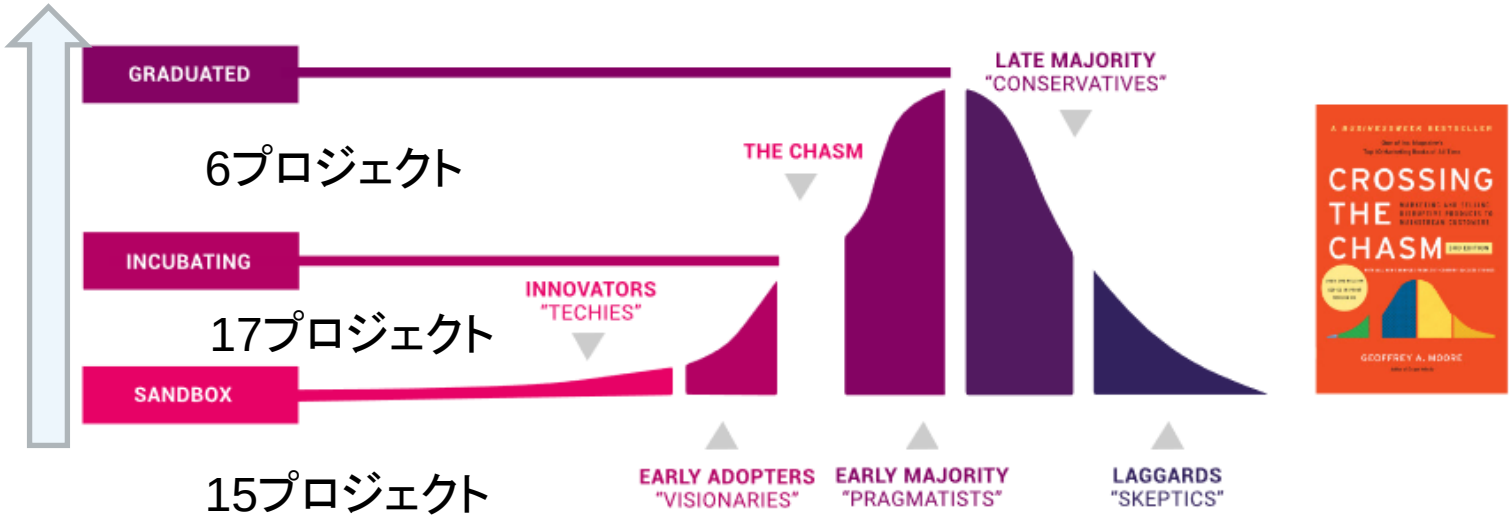
- メトリックスベースのモニタリングシステム
- Go言語で作成
- Apache ライセンス 2.0
- Cloud Native Computing Foundationで開発・メンテナンス

- CNCFの目的

オープンソースでベンダー中立プロジェクトのエコシステムを育成・維持して、このパラダイムの採用を促進したいと考えてます。私たちは最先端のパターンを民主化し、これらのイノベーションを誰もが利用できるようにします。

Project Services and Maturity Levels

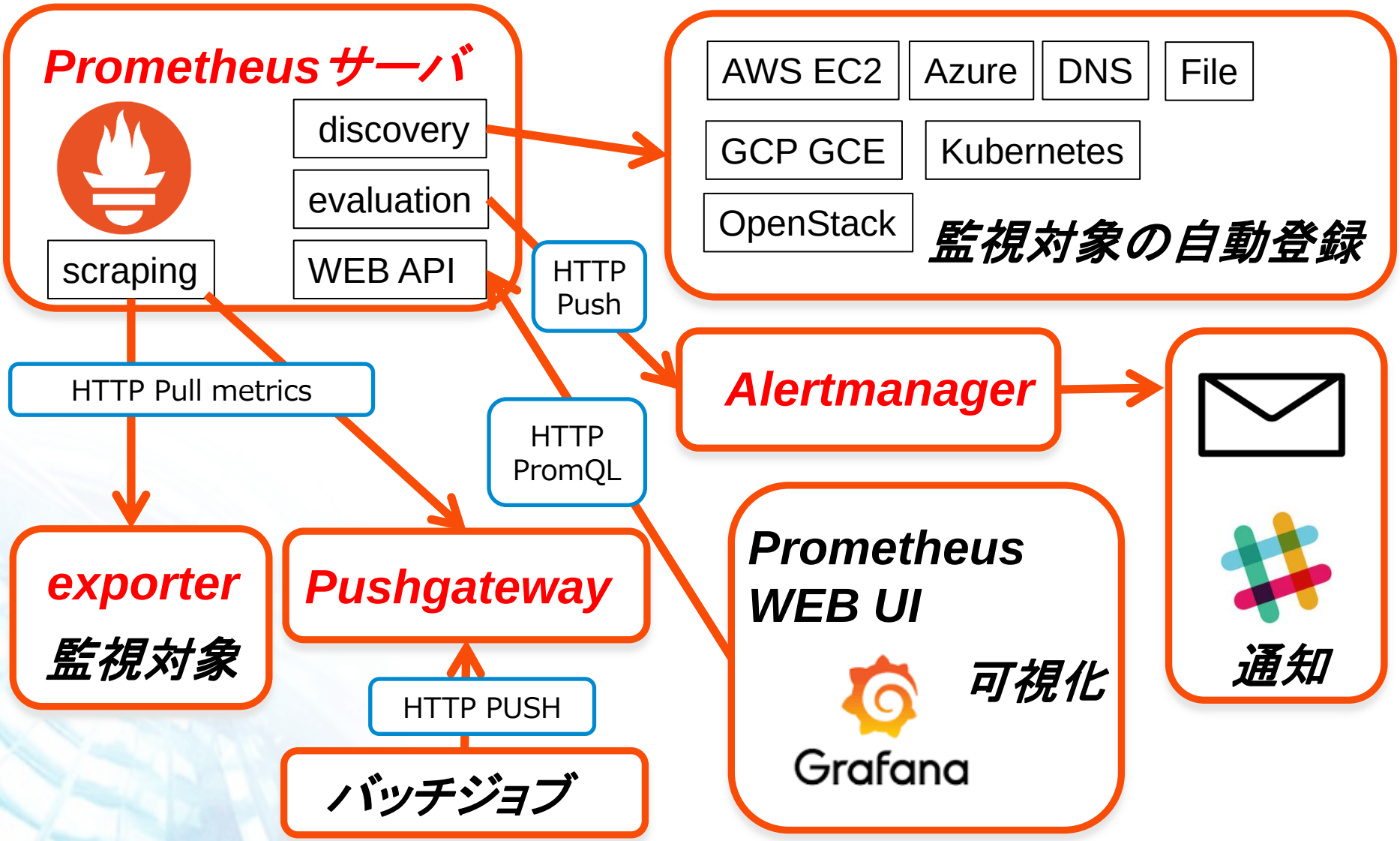
成熟度



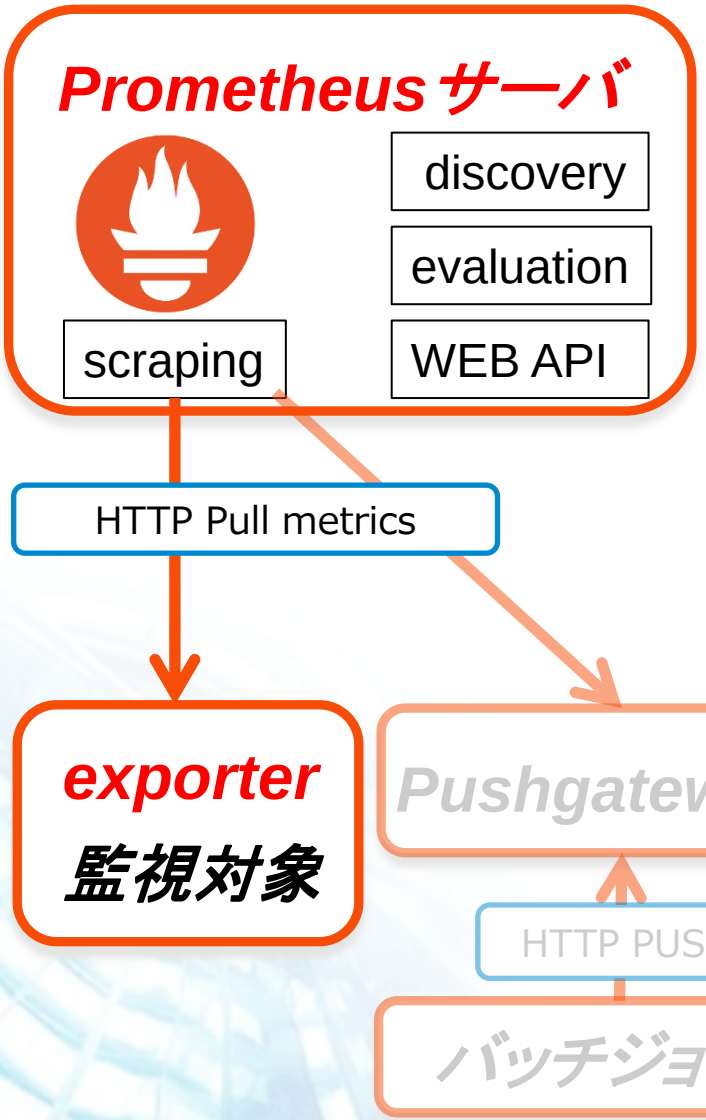
Graduated

Kubernetes	Prometheus	Envoy	CoreDNS	containerd	Fluentd
Orchestration	Monitoring	Service Proxy	Service Discovery	Container Runtime	Logging

Prometheusのアーキテクチャ



サーバとexporterについて



exporter　メトリックスを収集するエージェント

収集したいメトリックスごとにexporterを導入
Ex) http,postgresql,node

各exporterは異なるポート番号でHTTPリクエスト
を待ち受け

Prometheusサーバは定期的にHTTPリクエストを発行
してexporterで公開されているメトリックスを収集

exporter側の設定で公開するメトリックスを定義

exporterの種類

Databases

- Aerospike exporter
- ClickHouse exporter
- Consul exporter (**official**)
- Couchbase exporter
- CouchDB exporter
- ElasticSearch exporter
- EventStore exporter
- Memcached exporter (**officia**
- MongoDB exporter
- MSSQL server exporter
- MySQL server exporter (**official**)
- OpenTSDB Exporter
- Oracle DB Exporter
- PgBouncer exporter
- PostgreSQL exporter
- Presto exporter
- ProxySQL exporter
- RavenDB exporter
- Redis exporter
- RethinkDB exporter
- SQL exporter
- Tarantool metric library
- Twemproxy

Messaging systems

- Beanstalkd exporter
- EMQ exporter
- Gearman exporter
- IBM MQ exporter
- Kafka exporter
- NATS exporter
- NSQ exporter
- Mirth Connect exporter
- MQTT blackbox exporter
- RabbitMQ exporter
- RabbitMQ Management Plugin exporter
- RocketMQ exporter

HTTP

- Apache exporter
- HAProxy exporter (**official**)
- Nginx metric library
- Nginx VTS exporter
- Passenger exporter
- Squid exporter
- Tinyproxy exporter
- Varnish exporter
- WebDriver exporter

APIs

- AWS ECS exporter
- AWS Health exporter
- AWS SQS exporter
- Cloudflare exporter
- DigitalOcean exporter
- Docker Cloud exporter
- Docker Hub exporter
- GitHub exporter
- InstaClustr exporter
- Mozilla Observatory exporter
- OpenWeatherMap exporter
- Pagespeed exporter
- Rancher exporter
- Speedtest exporter

Issue trackers and continuous integration

- Bamboo exporter
- Bitbucket exporter
- Confluence exporter
- Jenkins exporter
- JIRA exporter

Hardware related

- apcupsd exporter
- BIG-IP exporter
- Collins exporter
- Dell Hardware OMSA exporter
- IBM Z HMC exporter
- IoT Edison exporter
- IPMI exporter
- knxd exporter
- Netgear Cable Modem Exporter
- Netgear Router exporter
- Node/system metrics exporter (**officia**
- NVIDIA GPU exporter
- ProSAFE exporter
- Ubiquiti UniFi exporter

Logging

- Fluentd exporter
- Google's mtail log data extractor
- Grok exporter

Other monitoring systems

- Akamai Cloudmonitor exporter
- Alibaba Cloudmonitor exporter
- AWS CloudWatch exporter (**official**)
- Azure Monitor exporter
- Cloud Foundry Firehose exporter
- Collectd exporter (**official**)
- Google Stackdriver exporter
- Graphite exporter (**official**)
- Heka dashboard exporter
- Heka exporter
- InfluxDB exporter (**official**)
- JavaMelody exporter
- JMX exporter (**official**)
- Munin exporter
- Nagios / Naemon exporter
- New Relic exporter
- NRPE exporter
- Osquery exporter
- OTC CloudEye exporter
- Pingdom exporter
- scollector exporter
- Sensu exporter
- SNMP exporter (**official**)
- StatsD exporter (**official**)
- TencentCloud monitor exporter

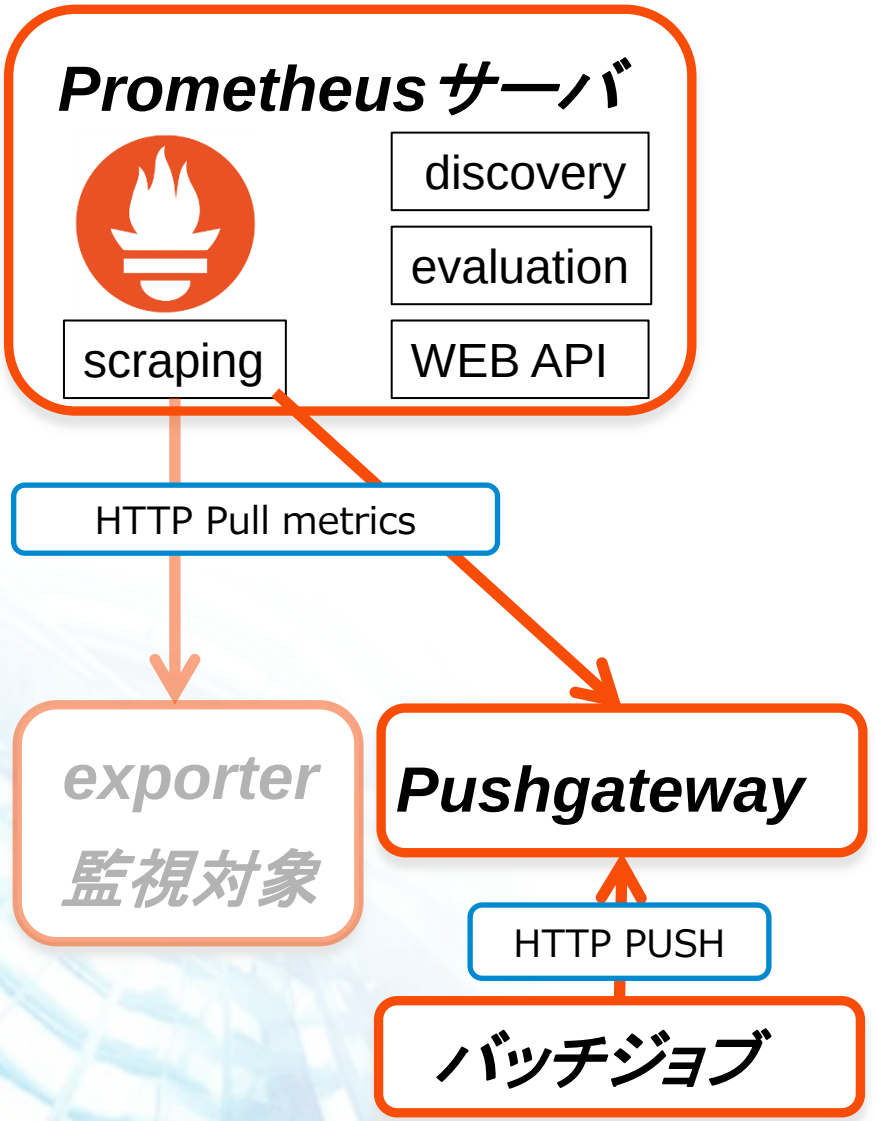
Storage

- Ceph exporter
- Ceph RADOSGW exporter
- Gluster exporter
- Hadoop HDFS FSImage exporter
- Lustre exporter
- ScaleIO exporter

Miscellaneous

- ACT Fibernet Exporter
- BIND exporter
- Blackbox exporter (**official**)
- BOSH exporter
- cAdvisor
- Cachet exporter
- ccache exporter
- Dovecot exporter
- Dnsmasq exporter
- eBPF exporter
- Ethereum Client exporter
- Hostapd Exporter
- JMeter plugin
- Kannel exporter
- Kemp LoadBalancer exporter
- Kibana Exporter
- kube-state-metrics
- Meteor JS web framework exporter
- Minecraft exporter module
- OpenStack exporter
- PHP-FPM exporter
- PowerDNS exporter
- Process exporter
- rTorrent exporter
- SABnzbd exporter
- Script exporter
- Shield exporter
- Smokeping prober
- SMTP/Maildir MDA blackbox prober
- SoftEther exporter
- Transmission exporter
- Unbound exporter
- WireGuard exporter
- Xen exporter

サーバとPushgateway



メトリックスをPUSHしたい場合に利用

バッチジョブ等の実行結果をPushgatewayにPUSH

PushgatewayではPUSHされた最新値を保存

PrometheusサーバからはPushgatewayに対して
データをPULLする

Prometheusサーバに対して直接メトリックスを
PUSHできない

サーバはメトリックスをPullするだけ
→ 安定性に貢献

WEB UI と PromQL

Prometheus サーバ

discovery

evaluation

scraping

WEB API

PrometheusサーバがWEB UIを提供しているが
1度きりのグラフ作成

ダッシュボードとしてはGrafanaを利用

GrafanaはHTTPを利用してサーバから
メトリックスを取得

HTTP
PromQL

PromQLでデータの選択、集計

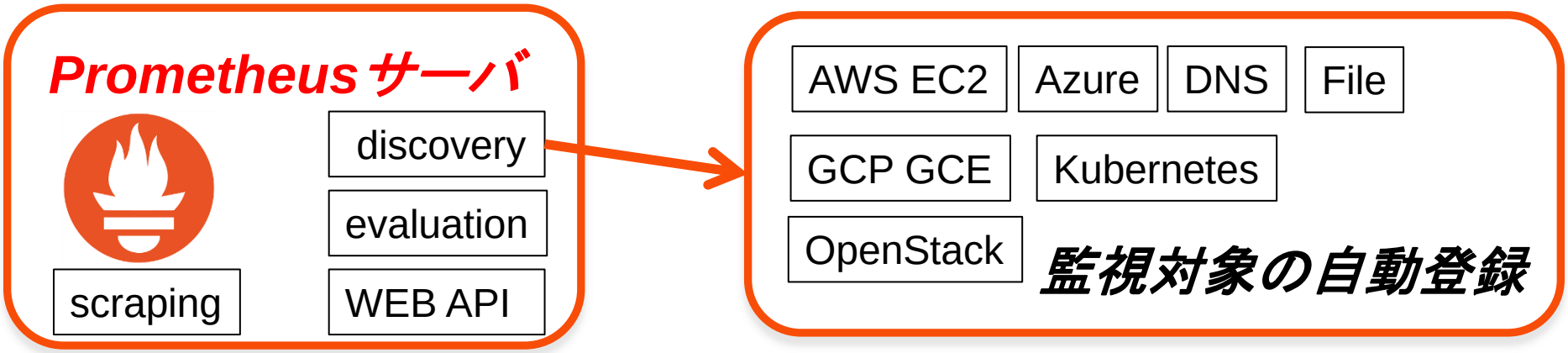
Prometheus WEB UI

Grafana

可視化

多数から絞り込んで少数のデータを参照
例) パーティションの利用率を知りたい
part_usage → 各パーティションの利用率
(/,/var,/home,/data)
part_usage{part='/var'}
→ /varの利用率

ディスカバリ 自動登録



定期的にPrometheusがサービスディスカバリを行い
監視対象を自動登録

EC2のインスタンスのディスカバリを行う設定

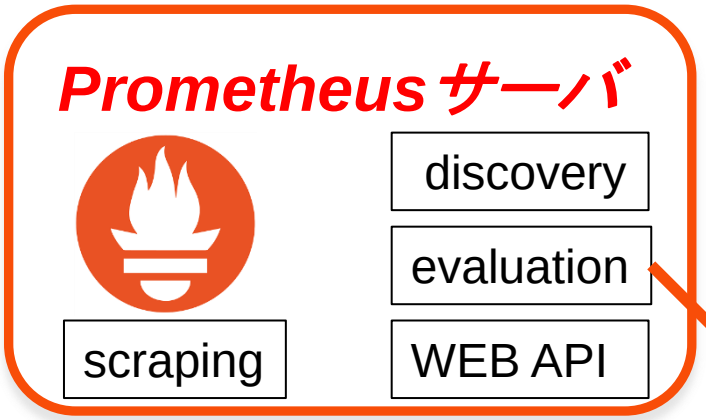
```
- job_name: ‘ec2sd'
    ec2_sd_configs:
      - region: xxx
        access_key: yyy
        secret_key:  zzz
        port: 9100
```

評価と通知

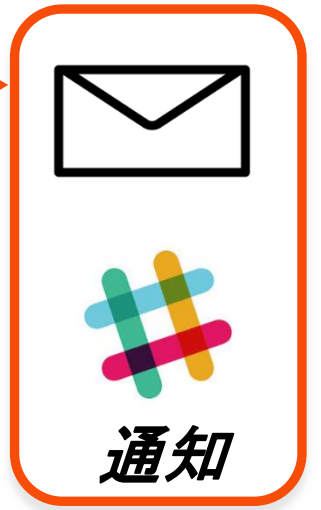
Scraping処理とは別処理でメトリックスの評価をおこなう

PromQLによりメトリックに対して計算式を定義し新たなメトリックスを定義することもできる

閾値を定義してAlertmanagerにHTTPでPush



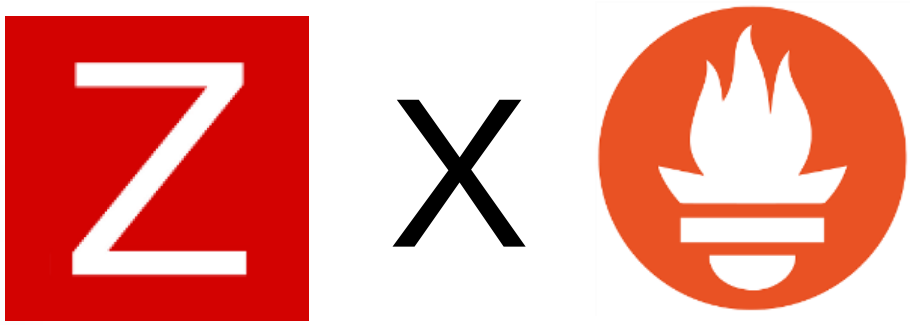
HTTP Push



Alertmanagerが受け取ったデータはラベルが定義されており、ラベル別に整理、集約ができる

ラベル“service”:”database”があるアラートは管理者に通知
ラベル“env”:”staging”は通知を行わない
ラベル“alertname”:”httpd_down”のアラートは5分ごとにまとめてWEB管理者へ通知

2. Zabbix 4.2 Prometheusとの連携



exporter から metrics を取得

exporter

HTTP

```
[root@dhcp-175-115 ~]# curl -Ss http://133.137.175.174:9100/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"} 8.257e-06
go_gc_duration_seconds{quantile="0.25"} 1.1438e-05
go_gc_duration_seconds{quantile="0.5"} 1.24e-05
go_gc_duration_seconds{quantile="0.75"} 1.3382e-05
go_gc_duration_seconds{quantile="1"} 8.6524e-05
go_gc_duration_seconds_sum 0.109053477
go_gc_duration_seconds_count 7411
# HELP go_goroutines Number of goroutines that currently exist.
# TYPE go_goroutines gauge
go_goroutines 6
# HELP go_info Information about the Go environment.
# TYPE go_info gauge
go_info{version="go1.12.3"} 1
# HELP go_memstats_alloc_bytes Number of bytes allocated and still in use.
# TYPE go_memstats_alloc_bytes gauge
go_memstats_alloc_bytes 2.53448e+06
```

メトリクス名{ラベル名="ラベル値",…} メトリクス値

Prometheus の metrics を HTTP エージェントで
取得・解析可能に

SRA OSS, INC. exporterの種類

Databases

- Aerospike exporter
- ClickHouse exporter
- Consul exporter (official)
- Couchbase exporter
- CouchDB exporter
- ElasticSearch exporter
- EventStore exporter
- Memcached exporter (officia
- MongoDB exporter
- MSSQL server exporter
- MySQL server exporter (official)
- OpenTSDB Exporter
- Oracle DB Exporter
- PgBouncer exporter
- PostgreSQL exporter
- Presto exporter
- ProxySQL exporter
- RavenDB exporter
- Redis exporter
- RethinkDB exporter
- SQL exporter
- Tarantool metric library
- Twemproxy

HTTP

- Apache exporter
- HAProxy exporter (official)
- Nginx metric library
- Nginx VTS exporter
- Passenger exporter
- Squid exporter
- Tinyproxy exporter
- Varnish exporter
- WebDriver exporter

APIs

- AWS ECS exporter
- AWS Health exporter
- AWS SQS exporter
- Cloudflare exporter
- DigitalOcean exporter
- Docker Cloud exporter
- Docker Hub exporter
- GitHub exporter
- InstaClustr exporter
- Mozilla Observatory exporter
- OpenWeatherMap exporter
- Pagespeed exporter
- Rancher exporter
- Speedtest exporter

Hardware related

- apcupsd exporter
- BIG-IP exporter
- Collins exporter
- Dell Hardware OMSA exporter
- IBM Z HMC exporter
- IoT Edison exporter
- IPMI exporter
- knxd exporter
- Netgear Cable Modem Exporter
- Netgear Router exporter
- Node/system metrics exporter (officia
- NVIDIA GPU exporter
- ProSAFE exporter
- Ubiquiti UniFi exporter

Logging

- Fluentd exporter
- Google's mtail log data extractor
- Grok exporter

Other monitoring systems

- Akamai Cloudmonitor exporter
- Alibaba Cloudmonitor exporter
- AWS CloudWatch exporter (official)
- Azure Monitor exporter
- Cloud Foundry Firehose exporter
- Collectd exporter (official)
- Google Stackdriver exporter
- Graphite exporter (official)
- Heka dashboard exporter
- Heka exporter
- InfluxDB exporter (official)
- JavaMelody exporter
- JMX exporter (official)
- Munin exporter
- Nagios / Naemon exporter
- New Relic exporter
- NRPE exporter
- Osquery exporter
- OTC CloudEye exporter
- Pingdom exporter
- scollector exporter
- Sensu exporter
- SNMP exporter (official)
- StatsD exporter (official)
- TencentCloud monitor exporter

Miscellaneous

- ACT Fibernet Exporter
- BIND exporter
- Blackbox exporter (official)
- BOSH exporter
- cAdvisor
- Cachet exporter
- ccache exporter
- Dovecot exporter
- Dnsmasq exporter
- eBPF exporter
- Ethereum Client exporter
- Hostapd Exporter
- JMeter plugin
- Kannel exporter
- Kemp LoadBalancer exporter
- Kibana Exporter
- kube-state-metrics
- Meteor JS web framework exporter
- Minecraft exporter module
- OpenStack exporter
- PHP-FPM exporter
- PowerDNS exporter
- Process exporter
- rTorrent exporter
- SABnzbd exporter
- Script exporter
- Shield exporter
- Smokeping prober
- SMTP/Maildir MDA blackbox prober
- SoftEther exporter
- Transmission exporter
- Unbound exporter
- WireGuard exporter
- Xen exporter

Messaging systems

- Beanstalkd exporter
- EMQ exporter
- Gearman exporter
- IBM MQ exporter
- Kafka exporter
- NATS exporter
- NSQ exporter
- Mirth Connect exporter
- MQTT blackbox exporter
- RabbitMQ exporter
- RabbitMQ Management Plugin exporter
- RocketMQ exporter

Issue trackers and continuous integration

- Bamboo exporter
- Bitbucket exporter
- Confluence exporter
- Jenkins exporter
- JIRA exporter

Storage

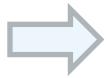
- Ceph exporter
- Ceph RADOSGW exporter
- Gluster exporter
- Hadoop HDFS FSImage exporter
- Lustre exporter
- ScaleIO exporter

Zabbixで使える！

Prometheus パターンの実例 1

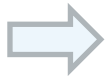
```
# HELP node_cpu_seconds_total Seconds the cpus spent in each mode.
# TYPE node_cpu_seconds_total counter
node_cpu_seconds_total{cpu="0",mode="idle"} 664924.22
node_cpu_seconds_total{cpu="0",mode="iowait"} 447.36
node_cpu_seconds_total{cpu="0",mode="irq"} 0
node_cpu_seconds_total{cpu="0",mode="nice"} 12.7
node_cpu_seconds_total{cpu="0",mode="softirq"} 250.5
node_cpu_seconds_total{cpu="0",mode="steal"} 0
node_cpu_seconds_total{cpu="0",mode="system"} 3964.57
node_cpu_seconds_total{cpu="0",mode="user"} 5382.81
```

名前	パラメータ		
1: Prometheusパターン	node_cpu_seconds_total{mode="idle"}	<ラベル名>	



結果
664924.22

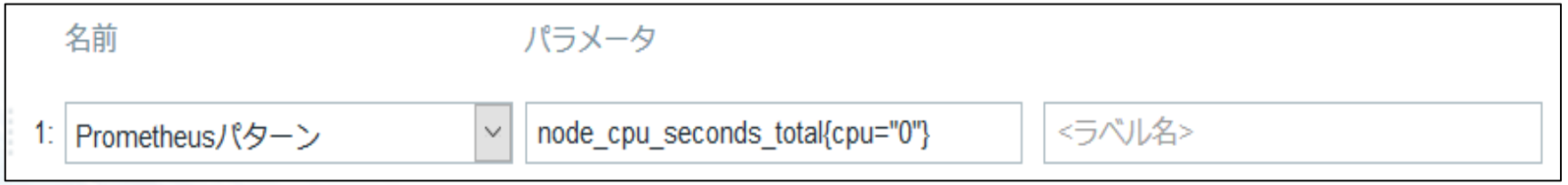
名前	パラメータ		
1: Prometheusパターン	node_cpu_seconds_total{mode="idle"}	mode	



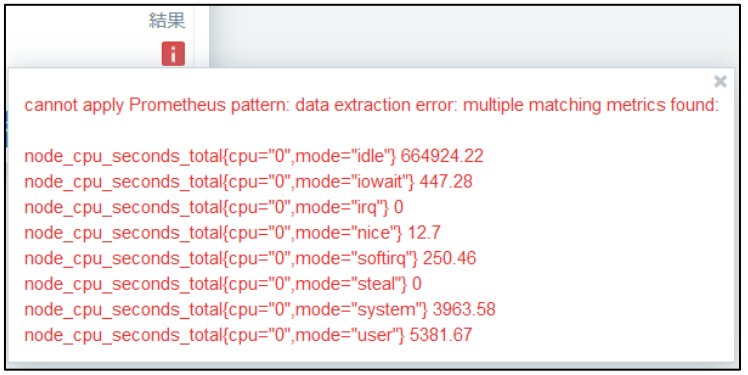
結果
idle

Prometheus パターンの実例 2

```
# HELP node_cpu_seconds_total Seconds the cpus spent in each mode.
# TYPE node_cpu_seconds_total counter
node_cpu_seconds_total{cpu="0",mode="idle"} 664924.22
node_cpu_seconds_total{cpu="0",mode="iowait"} 447.36
node_cpu_seconds_total{cpu="0",mode="irq"} 0
node_cpu_seconds_total{cpu="0",mode="nice"} 12.7
node_cpu_seconds_total{cpu="0",mode="softirq"} 250.5
node_cpu_seconds_total{cpu="0",mode="steal"} 0
node_cpu_seconds_total{cpu="0",mode="system"} 3964.57
node_cpu_seconds_total{cpu="0",mode="user"} 5382.81
```

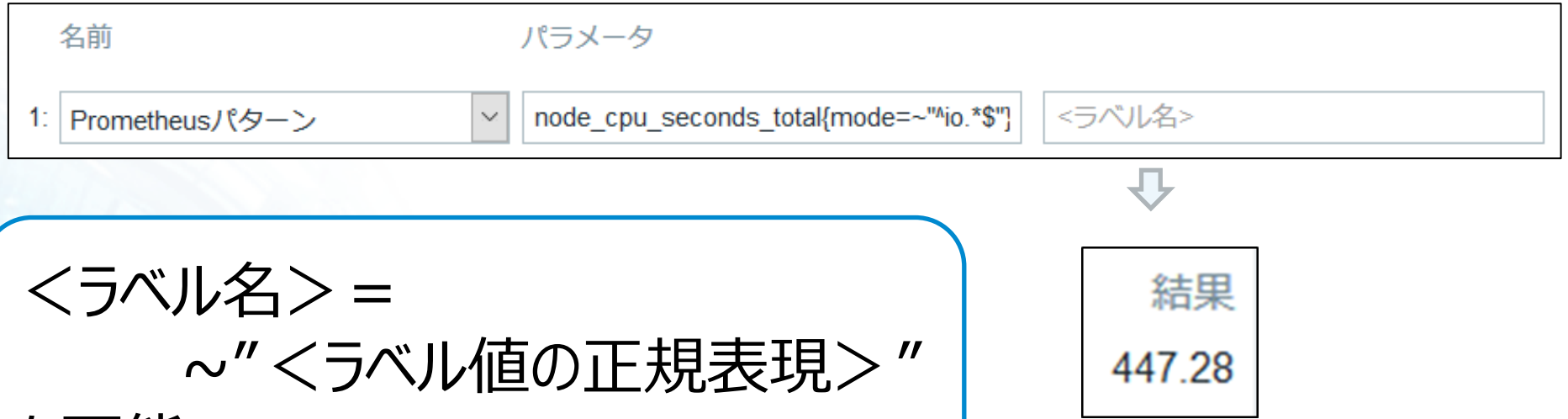


1 つのメトリクスを特定できるようなパラメータ設定が必須



Prometheus パターンの実例 3

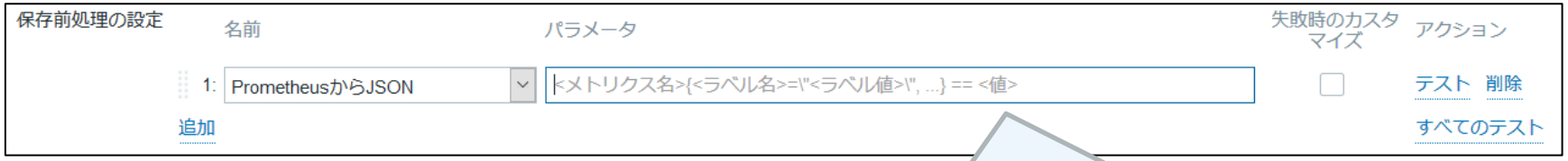
```
# HELP node_cpu_seconds_total Seconds the cpus spent in each mode.
# TYPE node_cpu_seconds_total counter
node_cpu_seconds_total{cpu="0",mode="idle"} 664924.22
node_cpu_seconds_total{cpu="0",mode="iowait"} 447.36
node_cpu_seconds_total{cpu="0",mode="irq"} 0
node_cpu_seconds_total{cpu="0",mode="nice"} 12.7
node_cpu_seconds_total{cpu="0",mode="softirq"} 250.5
node_cpu_seconds_total{cpu="0",mode="steal"} 0
node_cpu_seconds_total{cpu="0",mode="system"} 3964.57
node_cpu_seconds_total{cpu="0",mode="user"} 5382.81
```



＜ラベル名＞＝
～”＜ラベル値の正規表現＞”
も可能

Prometheus から JSON の設定

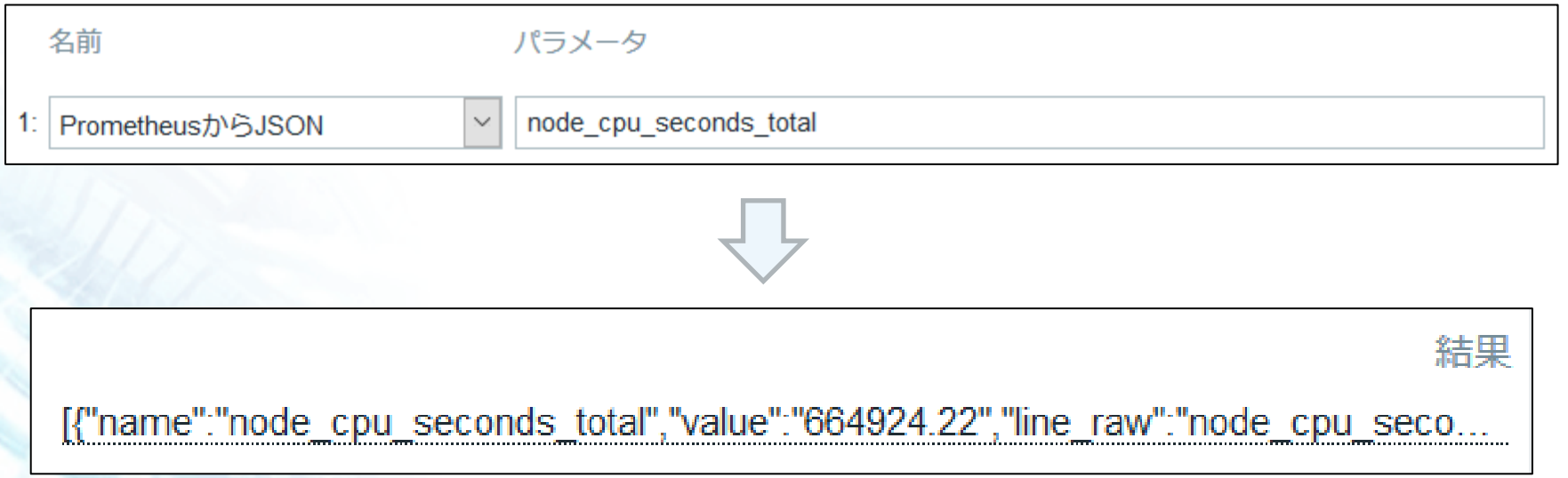
取得した Prometheus のメトリクスを JSON へ変換



JSON に変換したいメトリクスを
<メトリクス名>{<ラベル名>=<ラベル値>,…}==<値>
で指定（組み合わせ自由）

Prometheus から JSON の実例 1

```
# HELP node_cpu_seconds_total Seconds the cpus spent in each mode.
# TYPE node_cpu_seconds_total counter
node_cpu_seconds_total{cpu="0",mode="idle"} 664924.22
node_cpu_seconds_total{cpu="0",mode="iowait"} 447.36
node_cpu_seconds_total{cpu="0",mode="irq"} 0
node_cpu_seconds_total{cpu="0",mode="nice"} 12.7
node_cpu_seconds_total{cpu="0",mode="softirq"} 250.5
node_cpu_seconds_total{cpu="0",mode="steal"} 0
node_cpu_seconds_total{cpu="0",mode="system"} 3964.57
node_cpu_seconds_total{cpu="0",mode="user"} 5382.81
```



Prometheus から JSON の実例 2

```
# HELP node_cpu_seconds_total Seconds the cpus spent in each mode.
# TYPE node_cpu_seconds_total counter
node_cpu_seconds_total{cpu="0",mode="idle"} 664924.22
node_cpu_seconds_total{cpu="0",mode="iowait"} 447.36
node_cpu_seconds_total{cpu="0",mode="irq"} 0
node_cpu_seconds_total{cpu="0",mode="nice"} 12.7
node_cpu_seconds_total{cpu="0",mode="softirq"} 250.5
node_cpu_seconds_total{cpu="0",mode="steal"} 0
node_cpu_seconds_total{cpu="0",mode="system"} 3964.57
node_cpu
```

```
[
  {
    "name":"node_cpu_seconds_total",
    "value":"664924.22",
    “line_raw":"node_cpu_seconds_total{cpu=¥"0¥",mode=¥"idle¥"} 664924.22 ¥r",
    "labels":{"cpu":"0","mode":"idle"},
    "type":"counter",
    "help":"Seconds the cpus spent in each mode. ¥r“
  },
  …
]
```

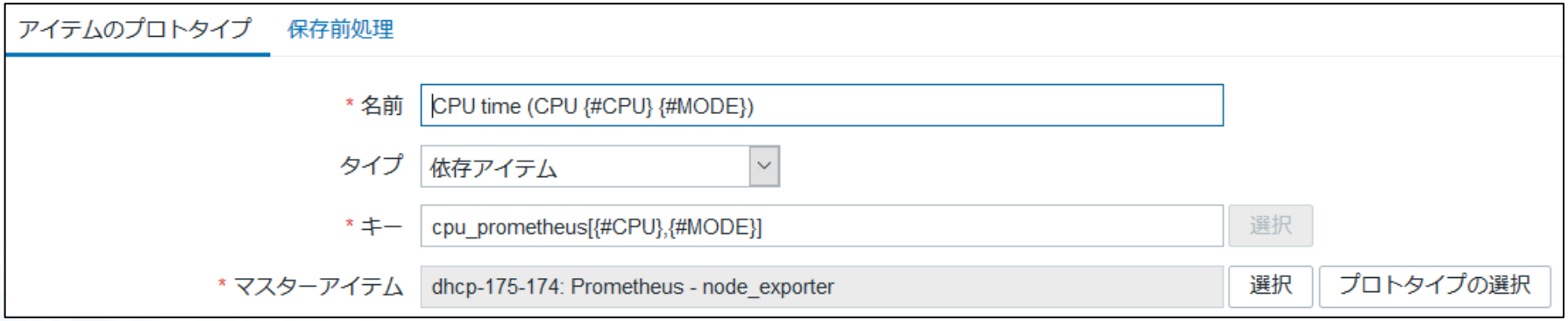
Prometheus から JSON の実例3-1

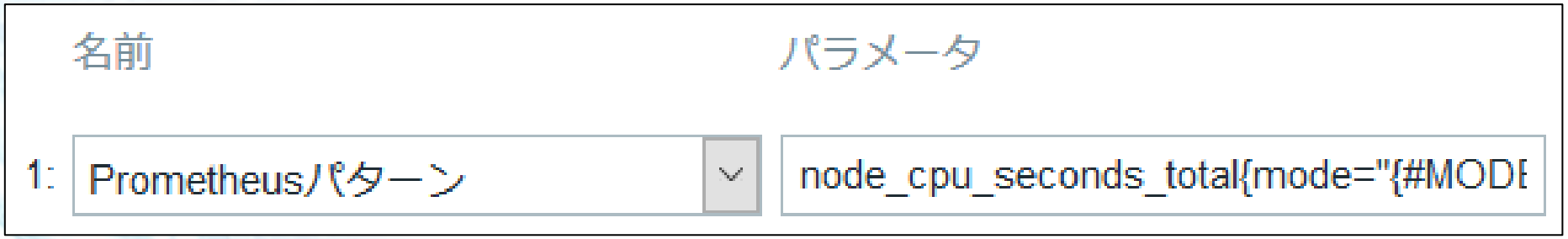
JSON へ変換して LLD ルールに活用

```
[
  {
    "name":"node_cpu_seconds_total",
    "value":"664924.22",
    “line_raw":"node_cpu_seconds_total{cpu=¥"0¥",mode=¥"idle¥"} 664924.22 ¥r",
    "labels":{"cpu":"0","mode":"idle"},
    "type":"counter",
    "help":"Seconds the cpus spent in each mode. ¥r“
  },
  …
]
```

Prometheus から JSON の実例3-2

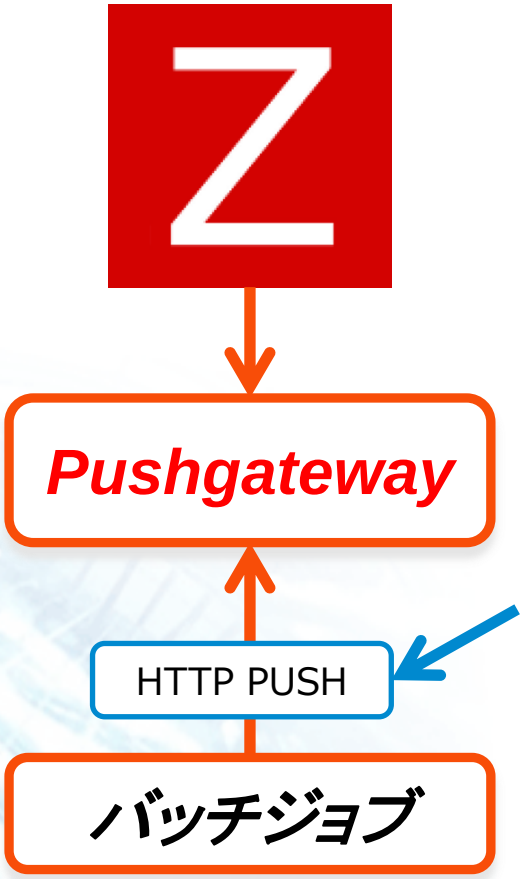
プロトタイプを依存アイテムで設定
保存前処理で LLD マクロを使用





Pushgatewayとの連携

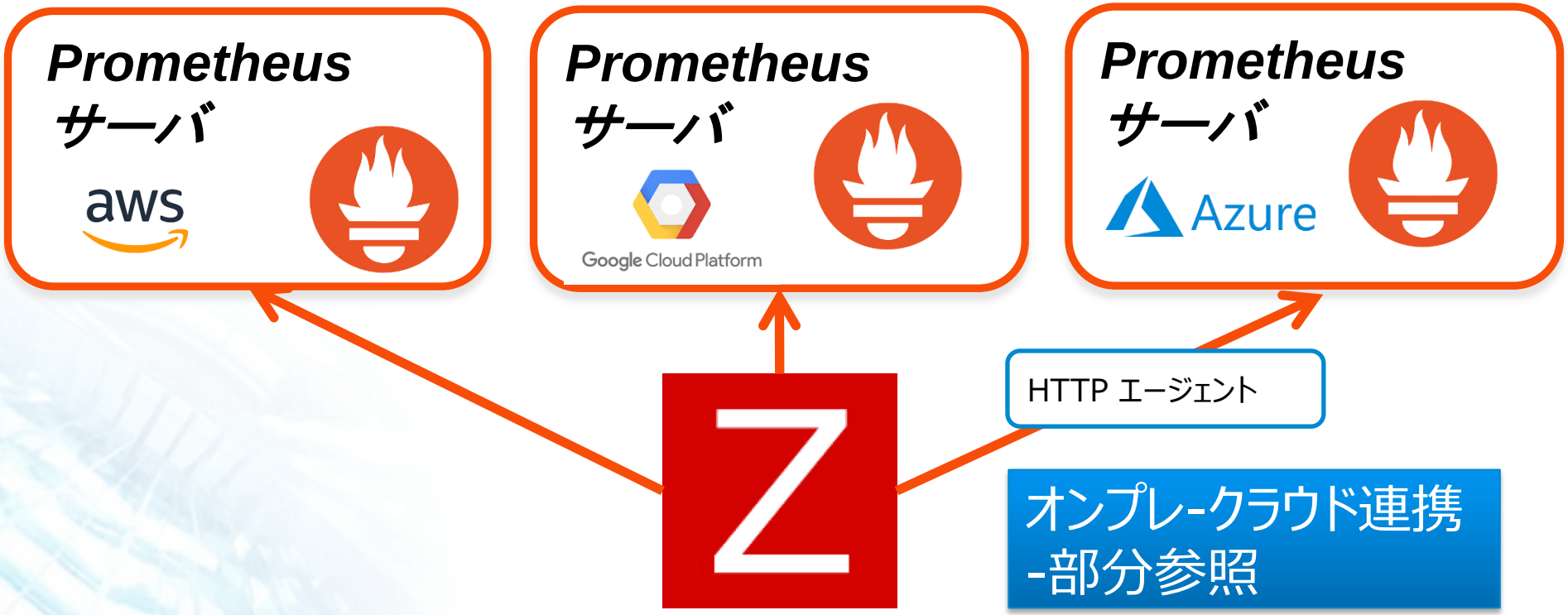
PushgatewayもExporterとして機能



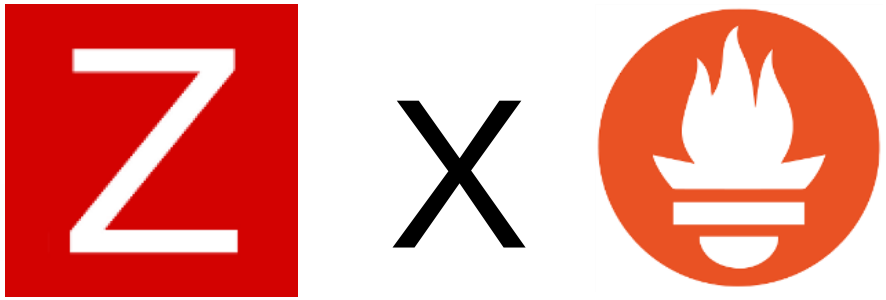

```
cat <<EOF | curl --data-binary @-
http://pushgateway.example.org:9091/metrics/job/some_job/instance/some_instance
# TYPE some_metric counter
some_metric{label="val1"} 42
# TYPE another_metric gauge
# HELP another_metric Just an example.
another_metric 2398.283
EOF
```

Prometheus Federationとの連携

Federation
Prometheusサーバ自身がExporterとして機能



3. Zabbix と Alertmanagerの連携



対象: Zabbix 3.2(障害の手動クローズ) 以降

Alertmanagerが受信するデータ 1

[障害発生] PrometheusサーバからAlertmanager(AM)へ次の形式のデータをPOST

```
[ { "labels":
      { “alertname”: “<requiredAlertName>”,
        “<labelname>”: “<labelvalue>”,
        ...
      },
      "annotations":
       { "<labelname>": "<labelvalue>", },
      “startsAt”: “<rfc3339>”,
      “endsAt”: “<rfc3339>”,
      "generatorURL": "<generator_url>"
}]
```

赤文字：必須

緑文字：自由定義

青文字：省略可

Labelsの内容が変わると違う通知データと解釈

Alertmanagerが受信するデータ 2

[障害復旧] PrometheusサーバからAlertmanager(AM)へ次の形式のデータをPOST

```
[ { "labels":
      { “alertname”: “<requiredAlertName>”,
        “<labelname>”: “<labelvalue>”,
        ...
      },
      "annotations":
       { "<labelname>": "<labelvalue>", },
      “startsAt”: “<rfc3339>”,
      “endsAt”: “2019-07-04T09:28:21.835576682+09:00”,
      "generatorURL": "<generator_url>"
}]
```

赤文字：必須
緑文字：自由定義
青文字：省略可

endsAtで指定した時刻に、同じlabelsのイベントが復旧する

ZabbixとAlertmanagerの連携　1

[障害発生] ZabbixのアクションからcurlでPOST

```
[{
   "labels":{
      "alertname":"{EVENT.NAME}",
      "hostname":"{HOST.NAME}",
      "severity":"{EVENT.SEVERITY}",
      “evnetid":"{EVENT.ID}",
      "itemkey":"{ITEM.KEY}",
    },
    "annotations":{ "lastvalue":"{ITEM.VALUE}“ },
    "generatorURL":
"http://ZabbixURL/zabbix/tr_events.php?triggerid={TRIGGER.ID}&eventid
={EVENT.ID}"
}]
```

Zabbixのマクロを利用

EVENT.IDを渡すことで
labelsがユニークになる

ITEM.VALUEなど変動する値
はannotaionsに入れる

[障害復旧] 復旧時のアクションからPOST

```
[{
   "labels":{
      "alertname":"{EVENT.NAME}",
      "hostname":"{HOST.NAME}",
      "severity":"{EVENT.SEVERITY}",
      “evnetid":"{EVENT.ID}",
      "itemkey":"{ITEM.KEY}",
    },
    "annotations":{ "lastvalue":"{ITEM.VALUE}“ },
    ”endsAt”:”アクション発報時刻を埋め込む”,
    "generatorURL":
"http://ZabbixURL/zabbix/tr_events.php?triggerid={TRIGGER.ID}&eventid
={EVENT.ID}"
}]
```

Zabbixの準備　Alertscriptの作成

/usr/lib/zabbix/alertscripts/sendtoalertmanager.sh

```
#!/bin/bash
url="$1"
messageorg="$2"
fixends=`date "+%Y-%m-%dT%H:%M:%S.%N%:z"`
proxy=${3-""}
if [[ "$proxy" != '' ]]; then
    proxy="-x $proxy"
fi
message=${messageorg/:endsAt:/$fixends}
return=$(curl $proxy -H "Content-Type: application/json" -d "${message}" $url)
```

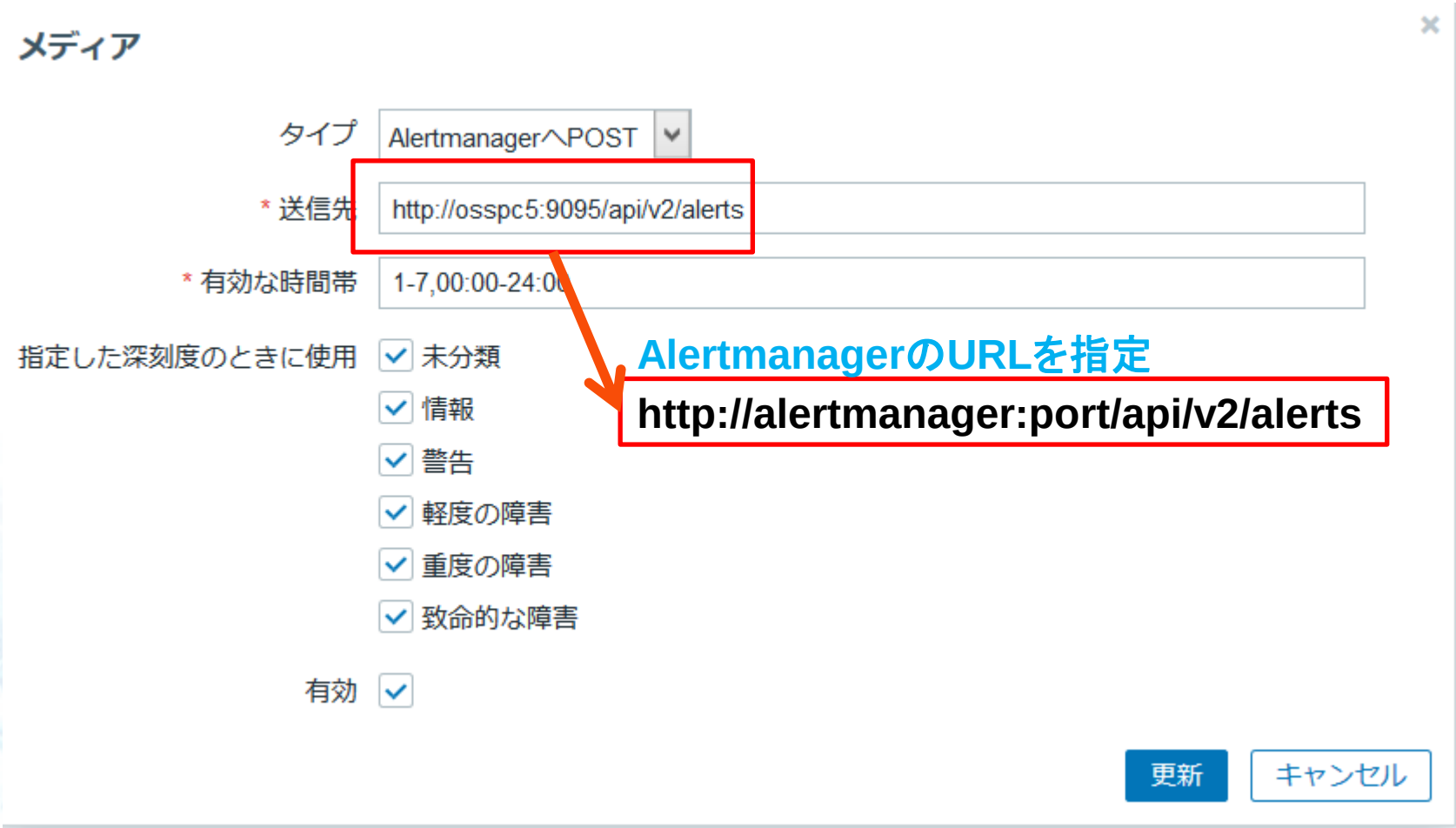
:endsAt:というパラメータがあれば
dateコマンドで作った日時を埋め込む

スクリプトの定義

第1引数　AlertmanagerのURL

第2引数「デフォルトのメッセージ」

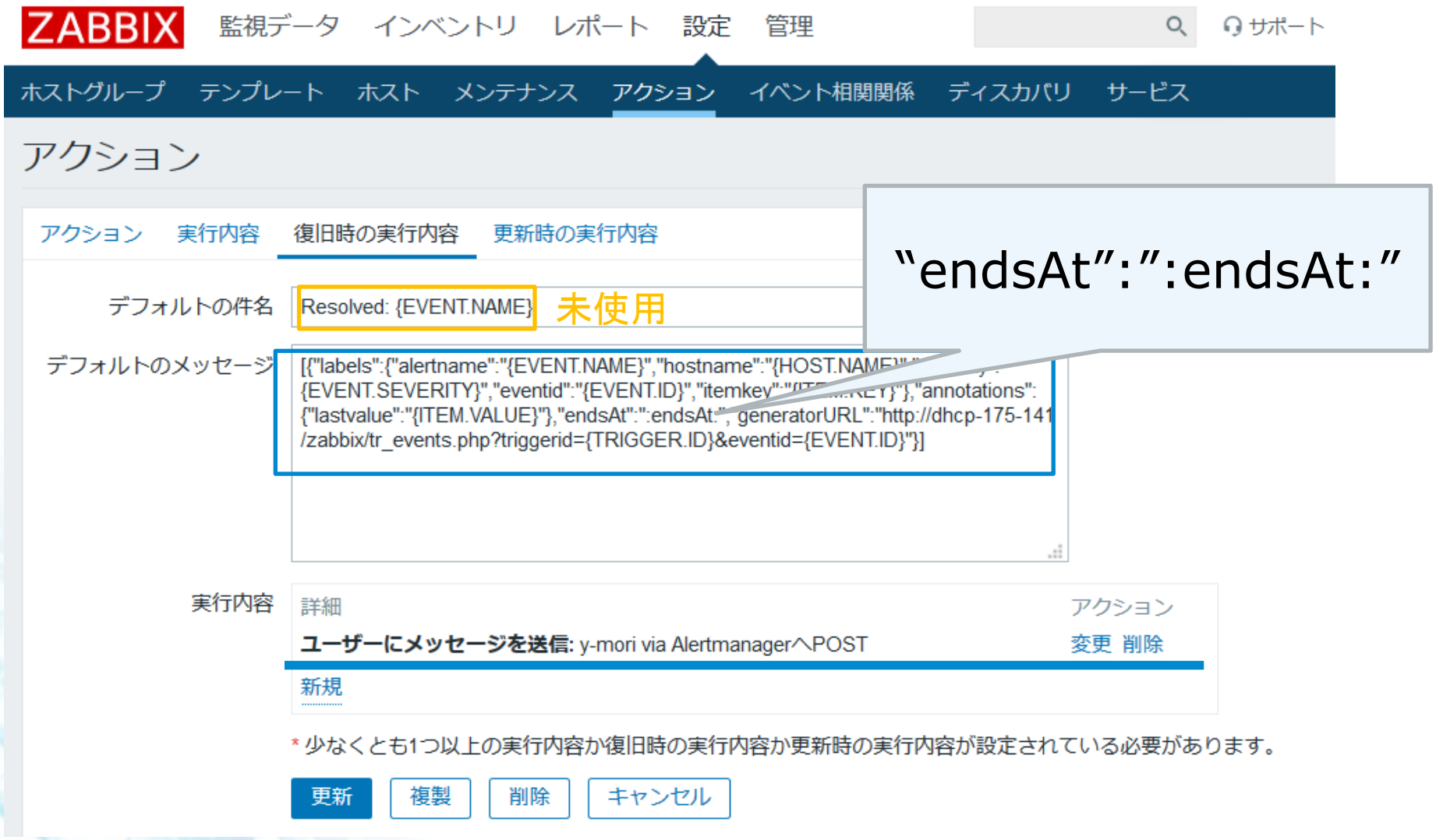
Zabbixの準備　ユーザにメディアを登録



Zabbixの準備 アクションの定義

Zabbixの準備 実行内容の定義

Zabbixの準備 アクションの定義



Alertmanagerの設定　全体

```
global:
  smtp_smarthost: ‘smtpserver:25'
  smtp_from: 'alertmanager@sraoss.co.jp'
  smtp_require_tls: false
  resolve_timeout: 5m

route:
  receiver: mailtouser
  group_wait: 30s
  group_interval: 5m
  repeat_interval: 4h
  group_by: [alertname,hostname]
  routes:
   - match:
       severity: Disaster
     group_wait: 5s

receivers:
  - name: mailtouser
    email_configs:
    - to: ‘someone@sraoss.co.jp'
      send_resolved: true
```

グローバル設定 SMTPの設定など

デフォルト設定

ルーティングの設定
どういう条件で
誰に通知するか？

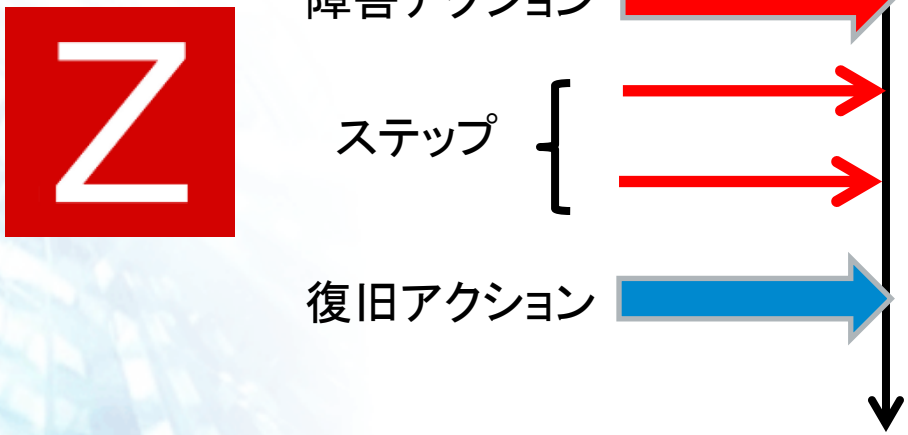
通知先の設定

Alertmanagerの設定 global

```
global:
  smtp_smarthost: ‘smtpserver:25'
  smtp_from: 'alertmanager@sraoss.co.jp'
  smtp_require_tls: false
resolve_timeout: 5m
```

- 障害が発生しPrometheusサーバがAMに通知データをPOSTしたあとは、同じラベルをもったデータをPrometheusが定期的にAMにデータを送る
- resolve_timeout時間(default:5m)データが届かないと回復したとみなす

zabbixからは「ステップ」を利用して定期的にAMにデータを送る



resolve_timeoutの時間を増やして、Zabbixのステップ実行間隔も増やすした方がよい

Alertmanager

Alertmanagerの設定 route

```
route:
  receiver: mailtouser
  repeat_interval: 4h
  group_wait: 30s
  group_interval: 5m
  group_by: [alertname,hostname]
  routes:
   - match:
       severity: Disaster
     group_wait: 5s
```

子ルートの定義

Zabbixの重要度**[致命的な障害]**のケースでは、group_waitを30秒から5秒に上書きしている例

group_wait（default:30s)
- 最初の通知を受信してから発報まで待つ時間
- しばらく待って通知をマージ

group_interval(default:5m)
- **通知をまとめる時間間隔**

group_by
- **指定したラベルで通知を集約**

repeat_interval(default:4h)
- **再通知を行う時間**

Alertmanagerの設定 receiver

```
receivers:
  - name: mailtouser
    email_configs:
    - to: ‘someone@sraoss.co.jp'
      send_resolved: true
```

send_resolved
(deafult: false)

- **障害回復通知を受け取るかどうか**

Slack通知例

```
 - name: sendtoslack
    slack_configs:
    - api_url: 'https://hooks.slack.com/services/XXXX/YYYY/ZZZZ'
      channel: '#notification'
      send_resolved: true
      title: '{{ if eq .Status "firing" }}[障害発生]{{else}}[障害回復]{{end}}
{{ .GroupLabels.alertname }}'
      http_config:
        proxy_url: 'http://proxyurl:proxy-port'
```

通知 時系列

Z

Labels{“alertname”:“DBエラー","hostname":”server1“,”eventid”: “X”}

Zabbixから発行された通知 障害 復旧

X＝1 X＝2 X＝3 X＝4

(注)図示していませんがステップ間隔で再送されています

時間

Alertmanager

group_wait

group_interval

1通目

X＝1 X＝2

2通目

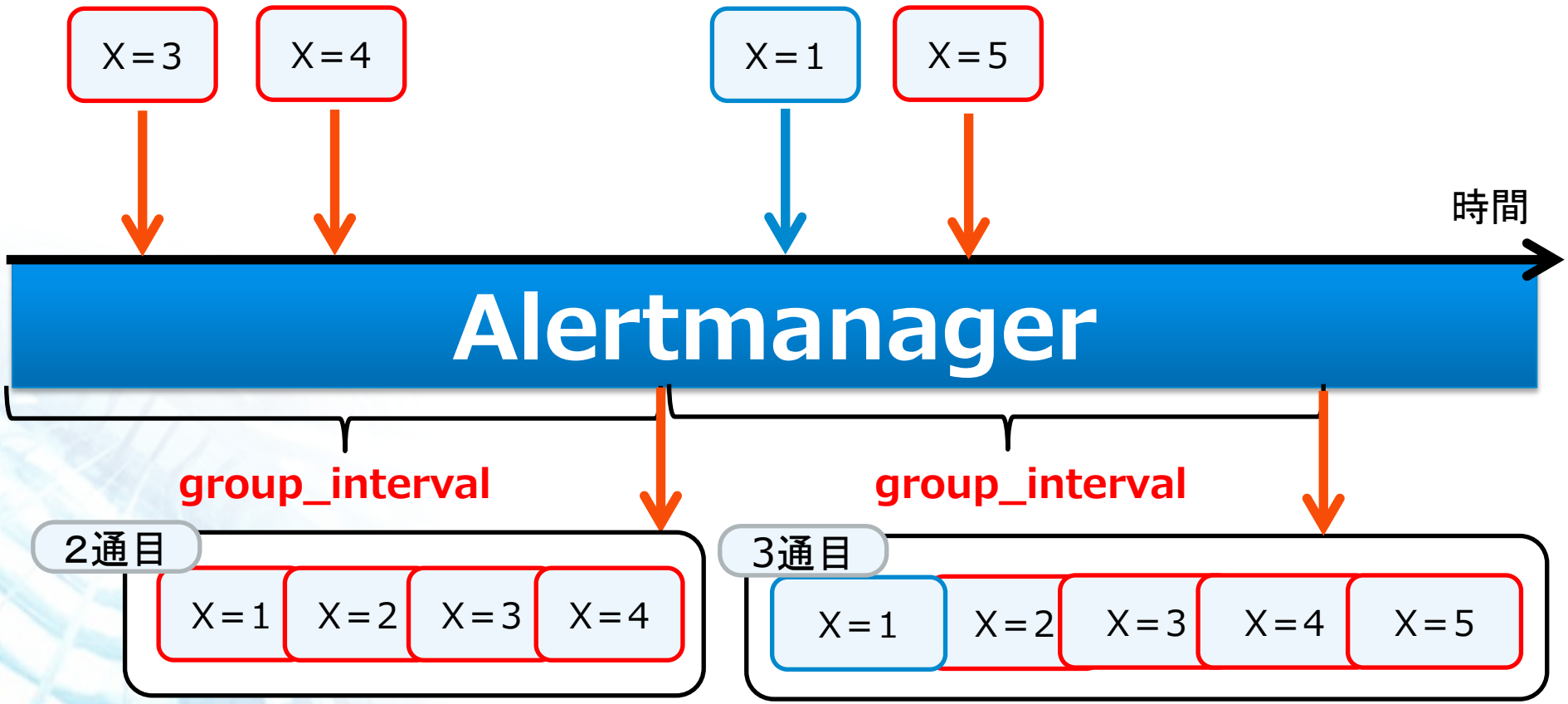
X＝1 X＝2 X＝3 X＝4

メール送信された内容

Z

Labels{“alertname”:“DBエラー","hostname":”server1“,”eventid”: “X”}

Zabbixから発行された通知 障害 復旧

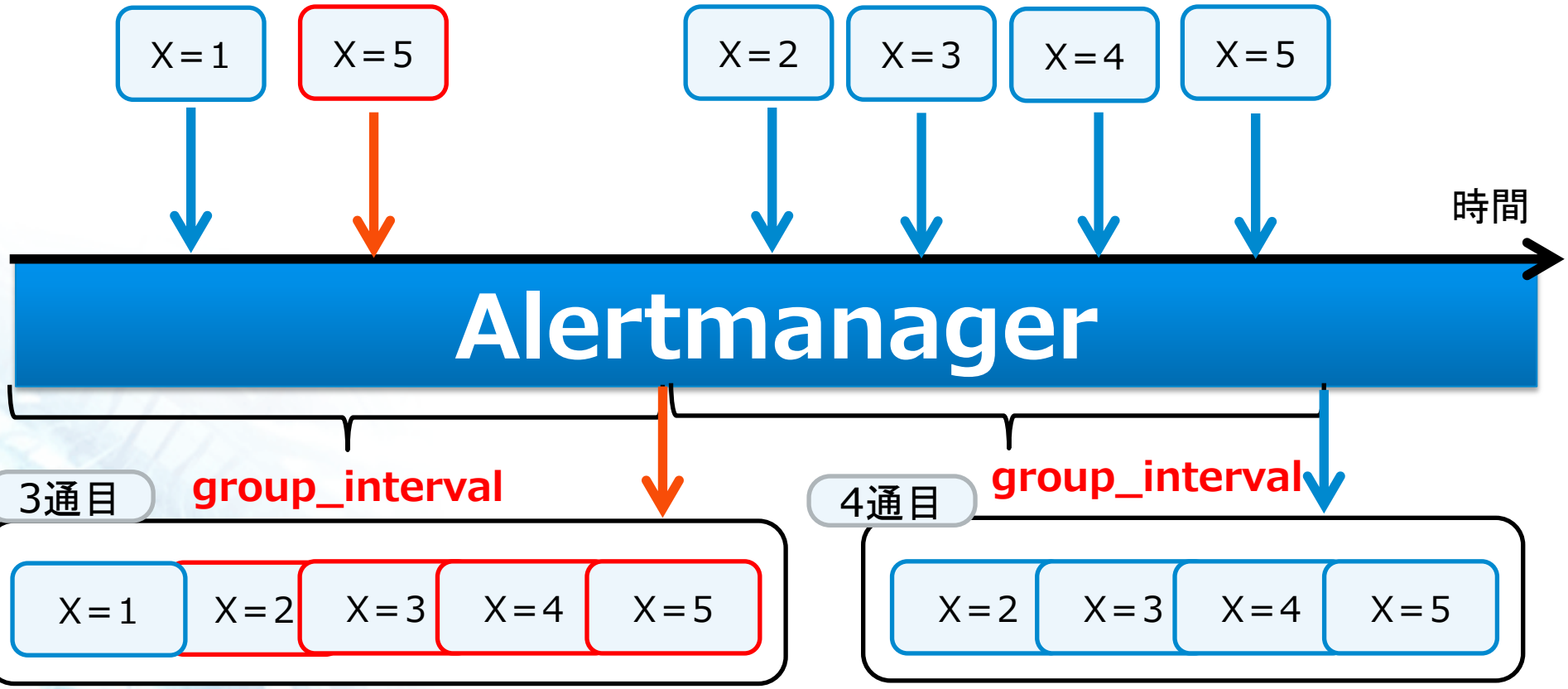


メール送信された内容

Z

Labels{“alertname”:“DBエラー","hostname":”server1“,”eventid”: “X”}

Zabbixから発行された通知　障害　復旧



メール送信された内容

実際のメール内容

To: y-mori@sraoss.co.jp
Subject: [FIRING:1] ログが出力されました Zabbix server (log[/tmp/date.log] Warning)
Date: Thu, 04 Jul 2019 10:33:09 +0900

text/html (13.7KB)

[FIRING:1] ログが出力されました Zabbix server (log[/tmp/date.log] Warning)
2 alerts for alertname=ログが出力されました hostname=Zabbix server
View in AlertManager
[1] Firing
Labels
alertname = ログが出力されました
eventid = 6167
hostname = Zabbix server
itemkey = log[/tmp/date.log]
severity = Warning
Annotations
lastvalue = 2019年 7月 4日 木曜日 10:24:53 JST xxx処理にエラーが発生しました。その2
Source

---

[1] Resolved
Labels
alertname = ログが出力されました
eventid = 6166
hostname = Zabbix server
itemkey = log[/tmp/date.log]
severity = Warning
Annotations
lastvalue = 2019年 7月 4日 木曜日 10:24:53 JST xxx処
Source

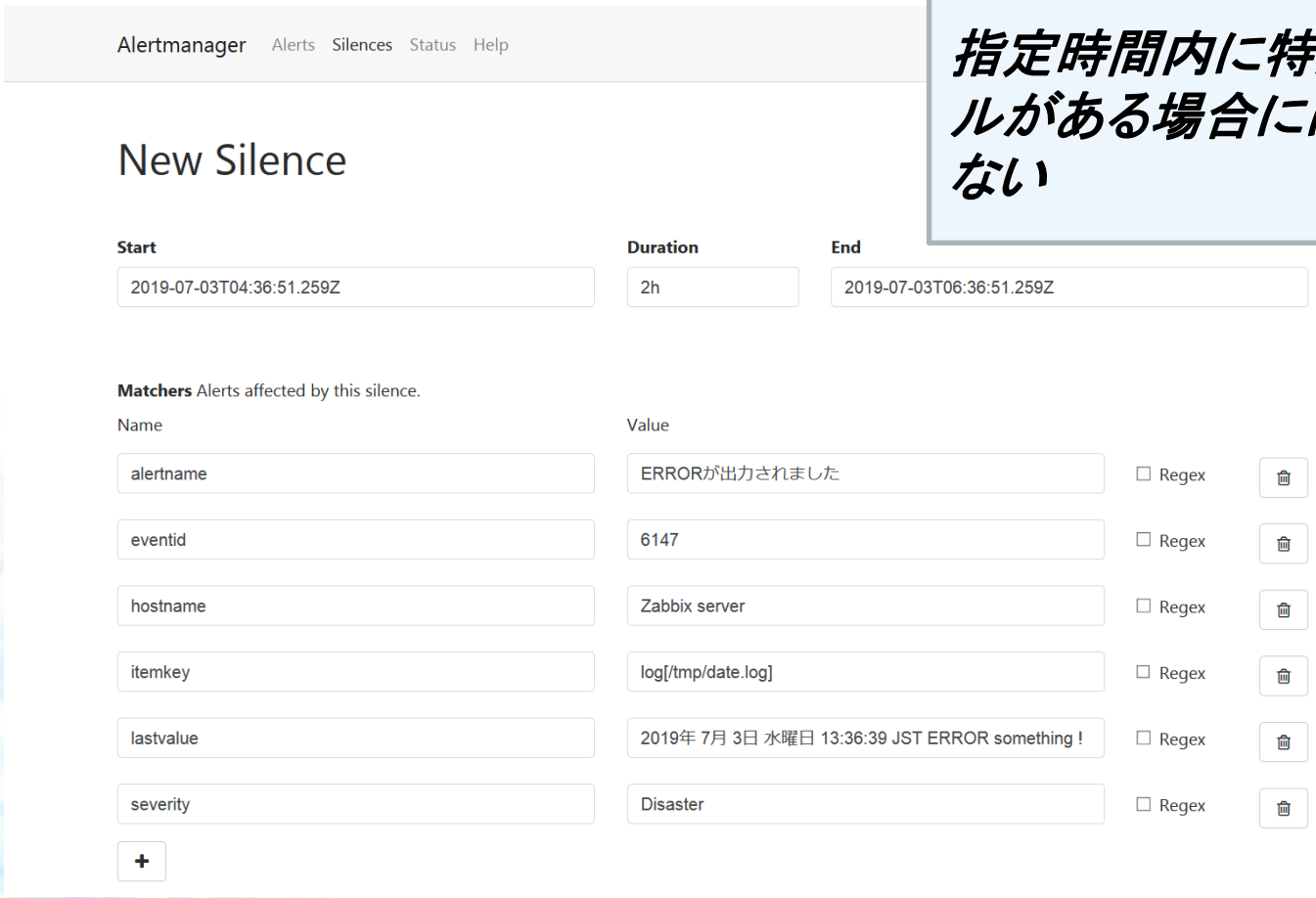
Sent by AlertManager

AlertManagerのUIへ

Zabbixのイベント障害画面へ

通知のSilence(ミュート)機能

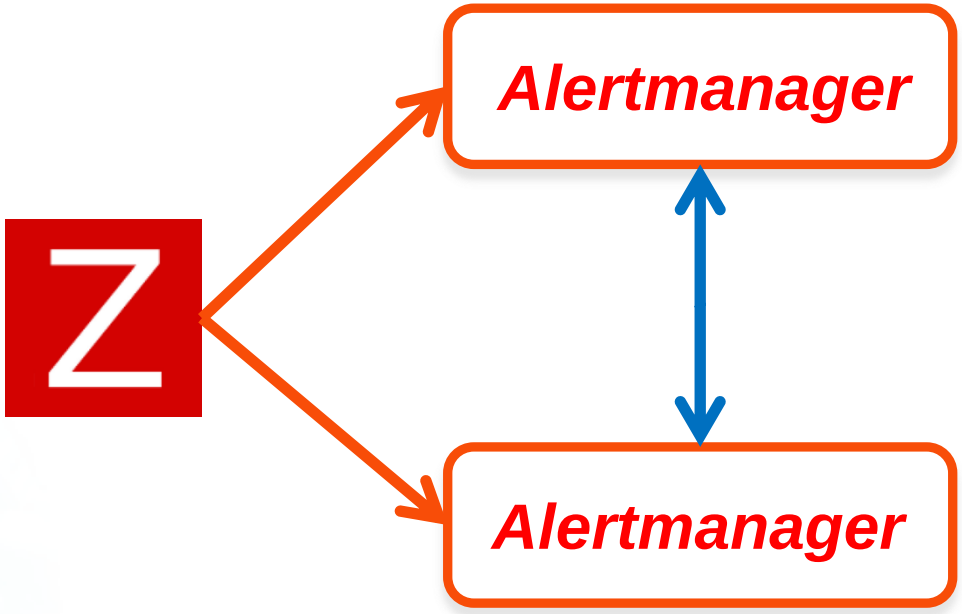
AlertmanagerのUIから通知の抑止ができる



指定時間内に特定のラベルがある場合には通知をしない

Alertmanager クラスタ

Labelsの内容が同じであれば同じ通知データ



互いのAlertmanagerが連携して通知を抑止

まとめ

- Prometheusと連携することで監視メトリックスを効率よく取得できる
- Alertmanagerを使うと通知の集約ができる

ご清聴ありがとうございました