Driving Mercari with 50+ custom Plugins

Masahiro Nagano id:kazeburo

Mackerel Day! 2017/10/05

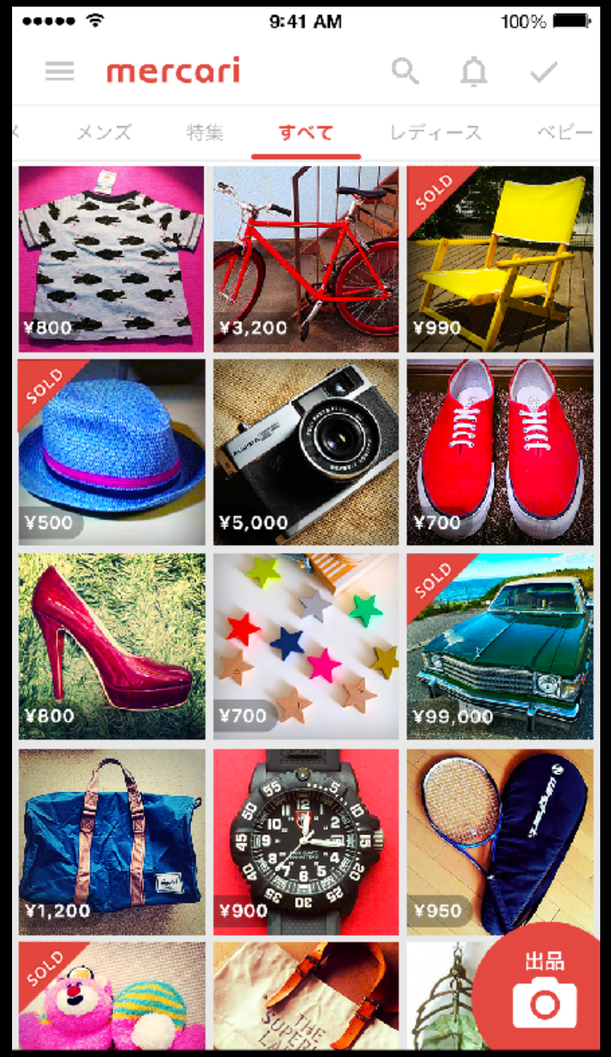
Me

- Masahiro Nagano / 長野雅広
- id:kazeburo
- Mercari, Inc
  Principal Engineer
  Site Reliability Engineering (SRE) Team
- BASE, Inc Technical Advisor

Agenda

- メルカリ & インフラストラクチャの簡単な紹介
- Service/Role設計 & Deploy
- Mackerelによる監視とPlugins
- その他の取り組み

Mercari



- フリマアプリ
- スマホで写真をとって簡単に出品
- 安心・安全な決済
- 日本・US・UKで展開

インフラストラクチャ

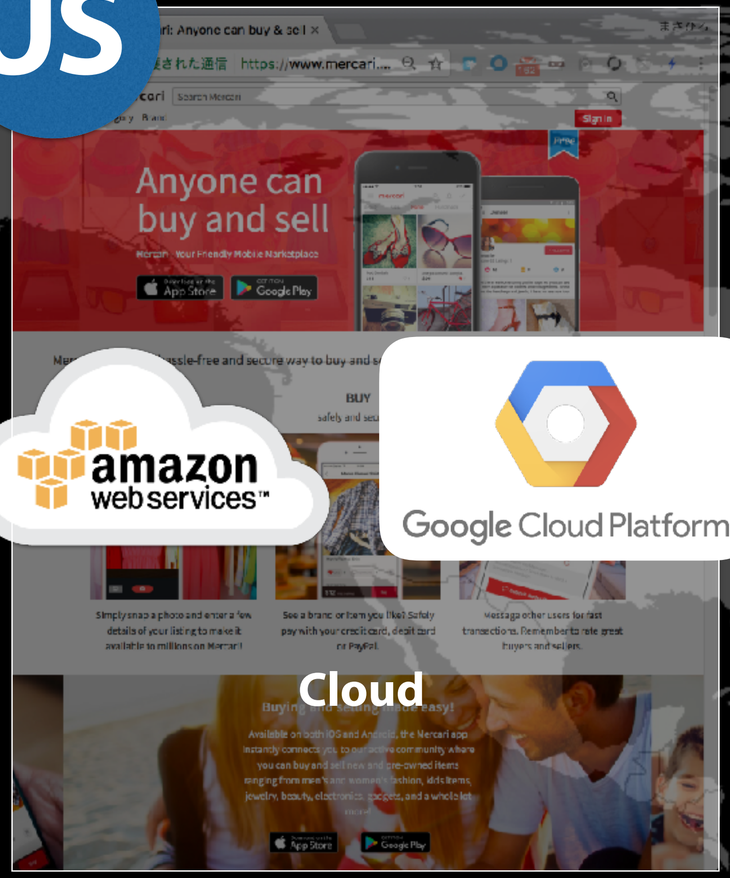
JP

SAKURA internet

石狩DC

専用サーバ

US

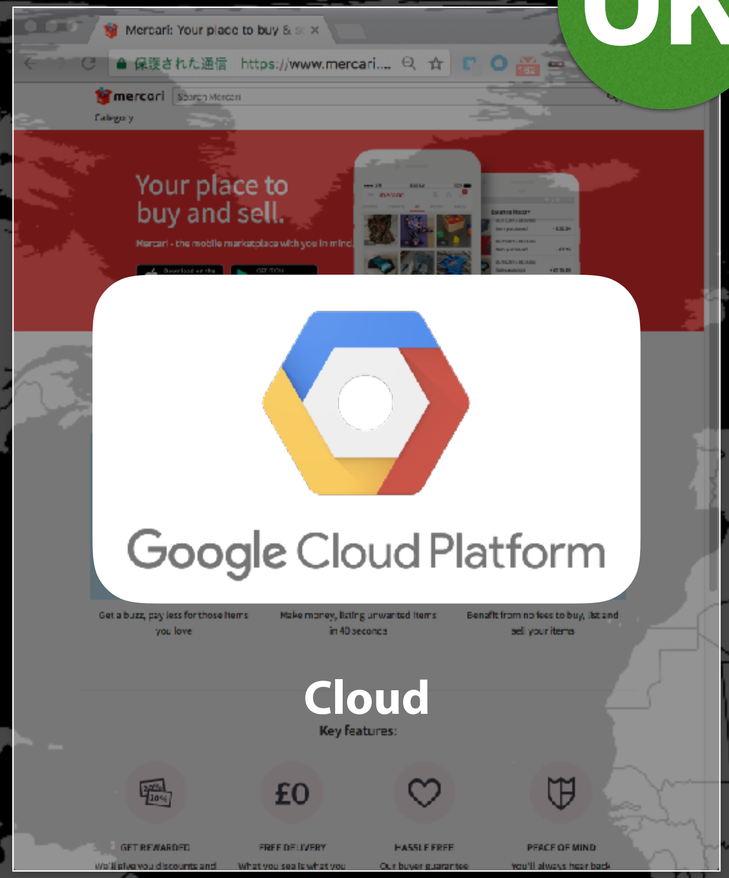


amazon web services™

Google Cloud Platform

Cloud

UK



Google Cloud Platform

Cloud

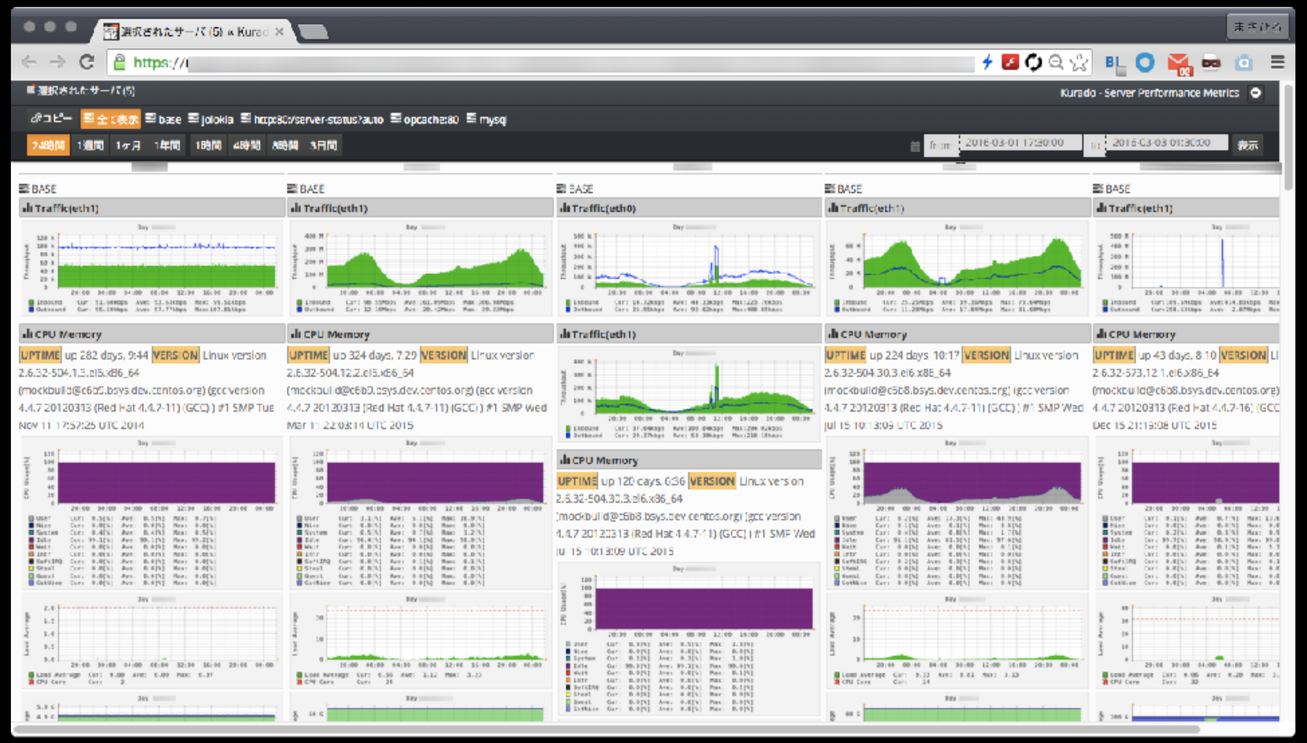
インフラストラクチャ


JP
US
UK
SAKURA internet
石狩DC
専用サーバ
amazon web services
Google Cloud Platform
Cloud
Google Cloud Platform
Cloud
mackerel

Mackerel 導入理由

- 異なるインフラストラクチャの監視項目・内容を共通化する
  - 以前はRegionごとにzabbixを利用。バージョンがずれたり監視内容の差が発生
  - Service/Roleを利用することで管理
- サービスメトリクスの柔軟な使い勝手
- Nagios互換のPlugin

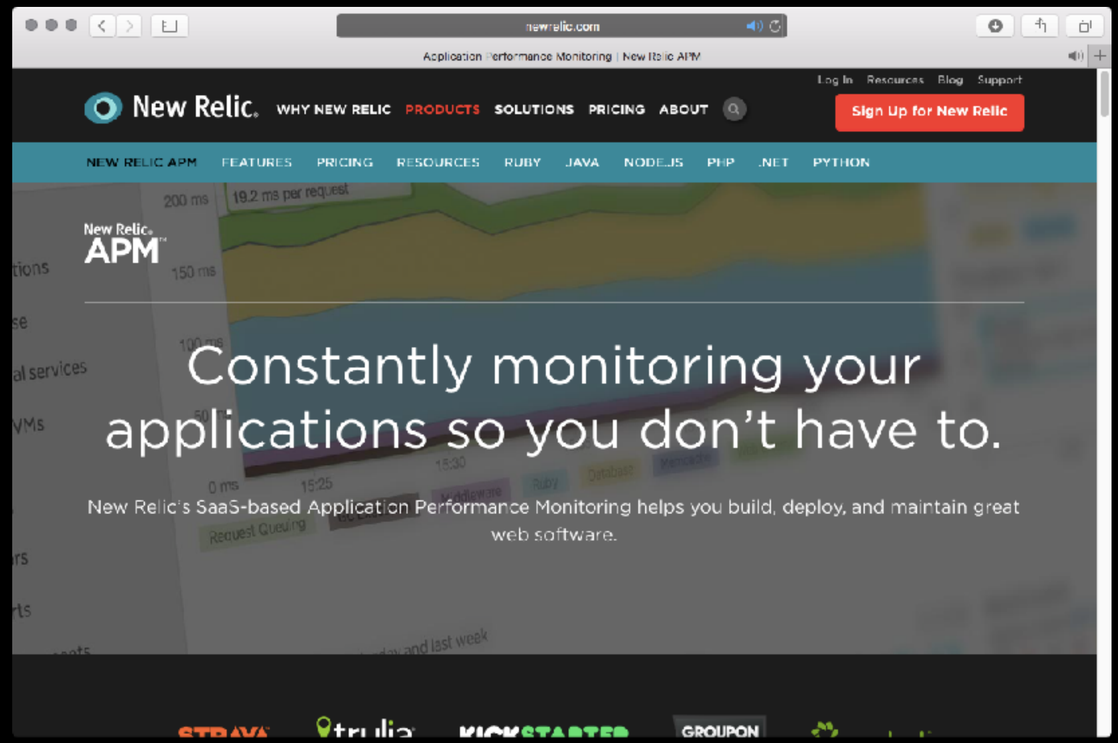
Mackerel以外の監視



Kurado

グラフ画像を一気に見れるので便利

基本的なメトリクスはこちらで見る



New Relic

PHPの内部のトレース

アプリケーションのチューニングの参考

Service/Role設計&Deploy

Service設計

- サービスを行うRegionごとにServiceを分ける
  - mercari, mercari-us, mercari-gb
- 外形監視は別Service
  - mercari-jp-exetenal, mercari-us-external, mercari-gb-external
  - 通知チャンネルを分けるため
- QA環境・マイクロサービス

Role設計

- Role名のPrefixに意味を持たせる
  - role- サーバの基本的な役割。role-mysql、role-applicationなど
  - z- 共通の役割。多くのサーバはz-commonに属し、phpが入っているサーバはz-phpを持つ
  - x- 監視上のフラグ。role-mysqlはレプリケーション監視を行うが、x-mysql-masterを追加することで監視除外する
    - x- は手作業で追加を行うことが多い

⚙mackerel-agent.conf

```
$ cat mackerel-agent.conf
apikey = “AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAaa="
include = “/etc/mackerel-agent/conf.d/*.conf"
```

=> mackerel-agent.conf には特に設定は書かない

```
$ ls -1 conf.d/
role-mysql.conf
z-common-jp.conf
z-postfix.conf
```

サーバによって配布するconfが異なる
role名とほぼ一致だが完全に一緒ではない

サーバに付与するRoleをどこかで自動で設定したい

Roleの自動付与

```
$ cat /etc/sysconfig/mackerel-agent
ROLES=$(grep -h role-def: /etc/mackerel-agent/conf.d/*.conf \
  |awk -F: '{printf "-role=mercari:" $2  " “}')
OTHER_OPTS=$ROLES
```

conf.d 以下のファイルに #role-def:ロール名 を追加すると

起動時に読み込み、agentの起動オプションとして利用

```
/usr/bin/mackerel-agent --pidfile=/var/run/mackerel-agent.pid --root=/var/lib/mackerel-agent \
          -role=mercari:role-mysql -role=mercari:z-common -role=mercari:z-postfix
```

Roleの実際例

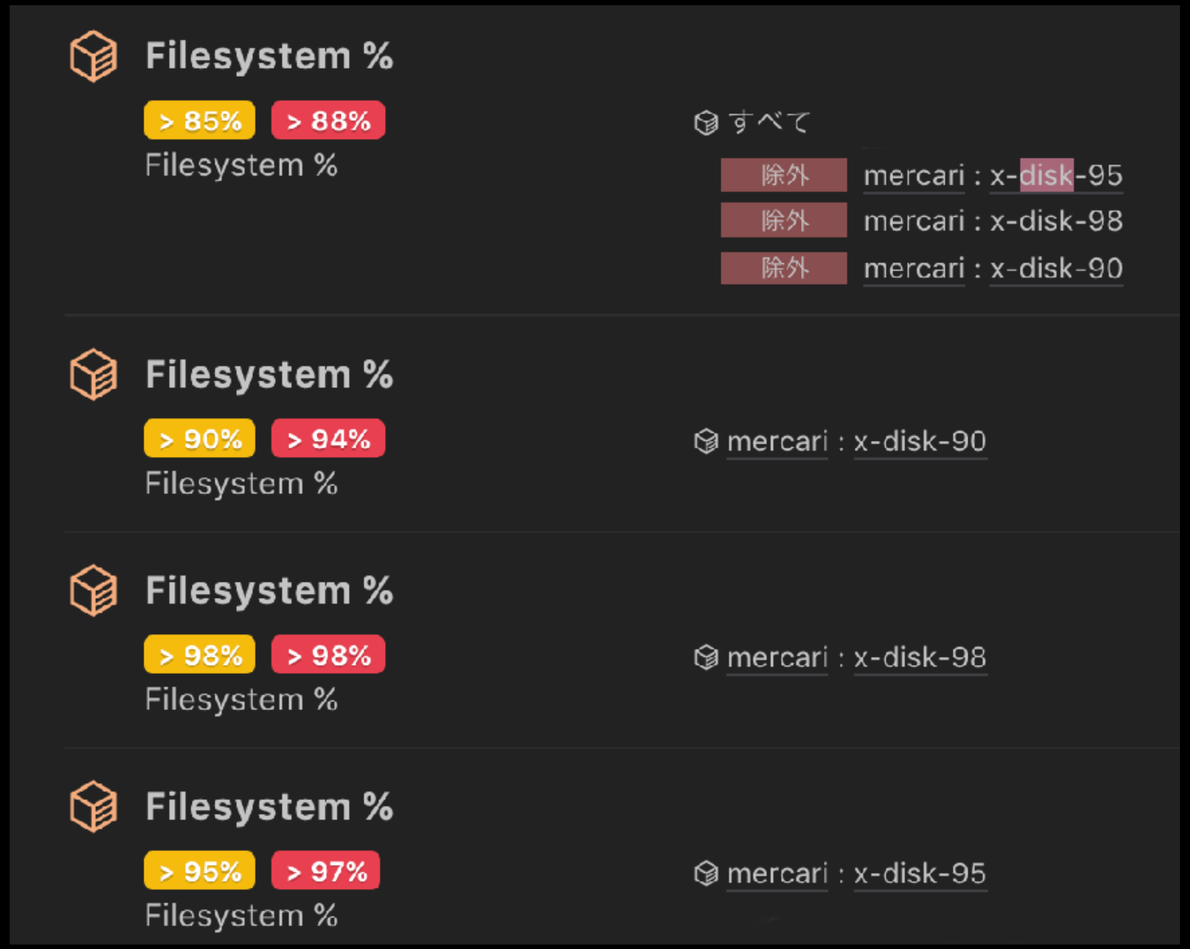
```
$ head -1 role-mysql.conf z-common-jp.conf z-postfix.conf
==> role-mysql.conf <==
## role-def:role-mysql

==> z-common-jp.conf <==
#role-def:z-common

==> z-postfix.conf <==
#role-def:z-postfix
```

実際にはrole-defは複数行でも構わない

x-Role の利用例



- disk監視の閾値をRoleで切り替える
- (運用で河馬に利用)

Service/Role設計 & Deploy

- **Roleのprefixに意味。conf内にRole名を書いて自動で設定**
  - systemdになると動かないという説
- **confはAnsibleで配布**
  - playbookにどのconfを配るのか記述
  - pluginも同じAnsible roleで管理し、confとともに配る

Mackerelによる監視とPlugins

監視にまつわる数字

- 監視ルール数: **265**
- **Host毎の監視ルール数**
  - MySQL: **34**
  - Application: **39**
  - Search: **36**
- **Custom Plugin: 50+ (check + metrics + utils)**

z-common-jp で行う監視

- unbound のプロセス監視
- unbound を使った名前解決
- crond のプロセス監視
- sshd のポート監視
- /etc/passwd ファイルの変更監視
- Global IPとiptable
- unameの変更監視
- hostnameの変更監視
- uptime監視
- inode監視
- メモリエラー
- HW-RAID監視
- [metrics] NTP
- [metrics] Linux Lite (CPU, Load avg, Process, Memory)
- [標準] File System
- [標準] Swap

Custom Pluginによる監視

- unbound のプロセス監視
- unbound を使った名前解決
- resolv.confを読んで名前解決
- crond のプロセス監視
- sshd のポート監視
- /etc/passwd ファイルの変更監視
- Global IPとiptable
- unameの変更監視
- hostnameの変更監視
- uptime監視
- inode監視
- メモリエラー
- HW-RAID監視
- [metrics] NTP
- [metrics] Linux Lite (CPU, Load avg, Process, Memory)
- [標準] File System
- [標準] Swap

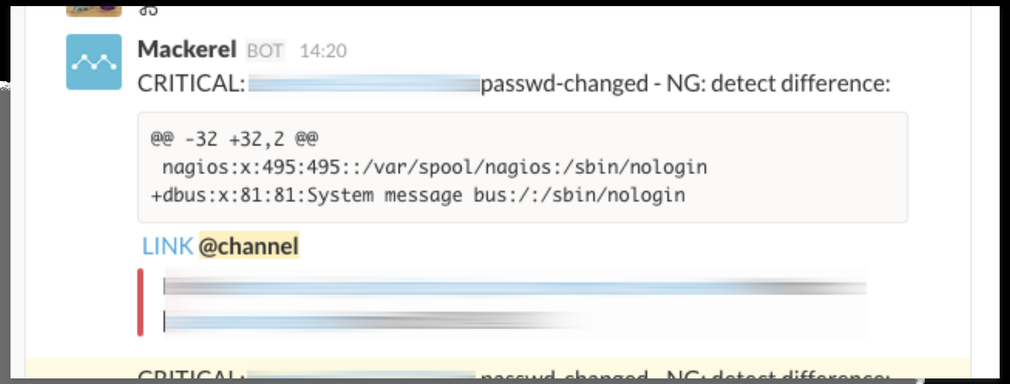
🔍 check_resolver

```
$ /etc/mackerel-agent/commands/check-resolver --host alive.local -w 2 -c 2
OK: elapsed_time 0.000888 sec (alive.local IN A 10.0.0.1)
```

- resolv.conf を独自に読み込んで名前解決する
  - resolv.conf が書き換わって名前解決に失敗する事故を防ぐ
- @kazuhoさんの Net::DNS::Lite を利用(pure-perlなDNSクライアント)

🔍 diff-detector

```
$ diff-detector -- date
NG: detect difference: ```@@ -1 +1 @@
-Tue May 10 08:11:42 UTC 2016
+Tue May 10 08:11:43 UTC 2016```
```



- コマンド結果の変化があるとアラート
- `cat /etc/passwd`、`uname -a`、`hostname` を見ている
- https://github.com/kazeburo/diff-detector

🔍 check-iptables

```
$ /etc/mackerel-agent/commands/check-iptables
OK: does not have global-ip and iptables(iptable_filter) is disabled
```

- さくらの専用サーバは全てglobal ipを持つ。不必要なサーバはdisableにして運用
  - global ipが有効: ip6?tables_filterがloadされてなければアラート
  - global ipが無効: ip6?tables_filterがloadされているとアラート
- 不用意な iptables --list でiptables_filterが読み込まれ、パフォーマンスに影響するのを発見

🔍 check-iptables

```
#!/bin/sh
set -e

if ( ip addr | grep 'inet ' | fgrep -v 'inet 127.0.0.1' | grep -v -E '^ *inet (10\.|
192\.168|10\.|172\.1[6789]\.|172\.2[0-9]\.|172\.3[01]\.)' > /dev/null 2>&1 ); then
  if ( lsmod |egrep 'ip6?table_filter' > /dev/null 2>&1 ); then
    echo "OK: have global-ip and iptables(iptable_filter) is enabled"
    exit 0
  else
    echo "NG: have global-ip. but iptables(iptable_filter) is disabled"
    exit 2
  fi
else
  ...
fi
```

🔍 check-uptime

```
$ /etc/mackerel-agent/commands/check-uptime -w 110 -c 110
OK: up 59 days,  8:21
```

- 不意な再起動を検知
- 閾値は2分-10秒。1回アラートが来てすぐに復旧する
- MySQLやmemcachedでも行なっている

🔍 check-inode

```
$ /etc/mackerel-agent/commands/check-inode -w 80 -c 90
OK: /:1%, /dev:1%, /dev/shm:1%, /run:1%, /sys/fs/cgroup:1%, /boot:1%, /run/user/1037:1%
```

- inode 枯渇防止
- OS、Roleによってpartitionのサイズが若干異なる。割合で監視し、一つでも閾値を上回るとアラート
- mackerel-plugin-inode は監視対象メトリックに wildcard が使えない

🔍 check-machine-exceptions

```
$ /etc/mackerel-agent/commands/check-machine-exceptions
OK: No Machine Check Exception found
```

- メモリ異常を検知した際のログを監視
- 検出後はハードウェアの保守を依頼する

🔍 check-machine-exceptions

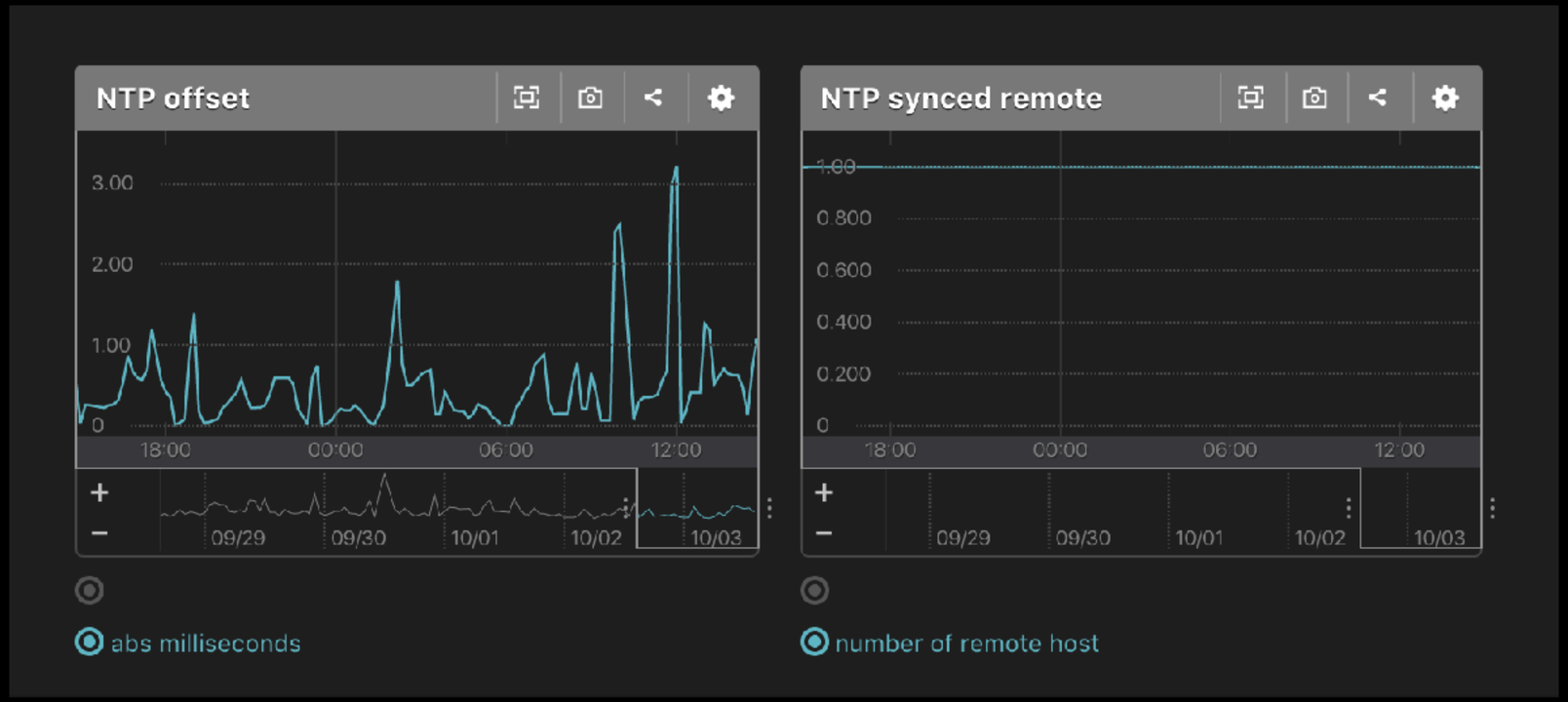
```
...
sbridge: HANDLING MCE MEMORY ERROR
CPU 0: Machine Check Exception: 0 Bank 8: cc0427c000010090
TSC 0 ADDR 37805ac0 MISC 45048ce86 PROCESSOR 0:406f1 TIME 1495654896 SOCKET
0 APIC 0
[Hardware Error]: Machine check events logged
EDAC MC1: CE row 0, channel 0, label "CPU_SrcID#0_Ha#0_Channel#0_DIMM":
4255 Unknown error(s): memory read on FATAL area OVERFLOW: cpu=0
Err=0001:0090 (ch=0), addr = 0x37805ac0 => socket=0, ha=1,
Channel=0(mask=1), rank=0
...
```

🔍 check-raid-disk (MegaRAID)

```
$ /etc/mackerel-agent/commands/check-raid-disk
Firmware state: Online, Spun Up
Firmware state: Online, Spun Up
```

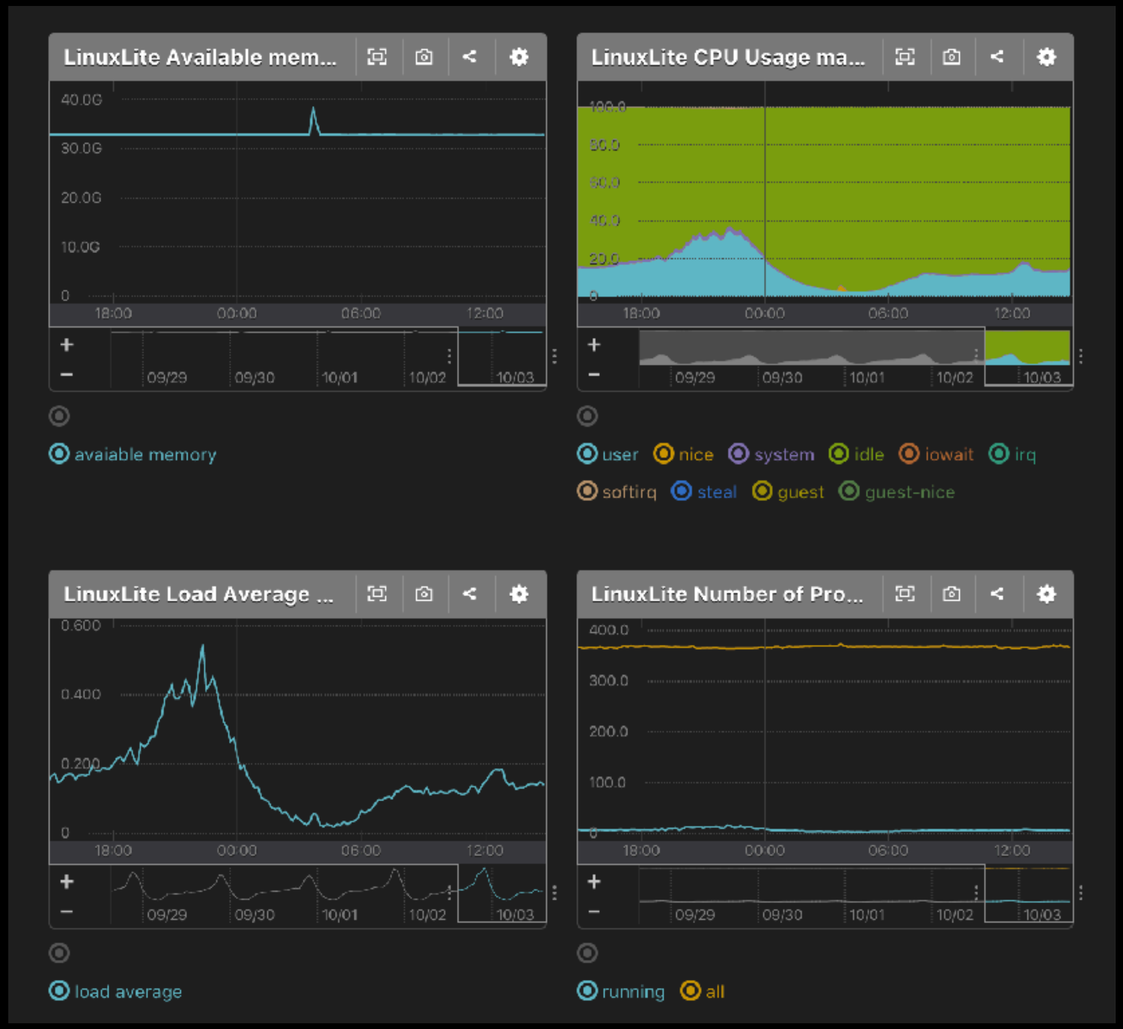
- MegaCLI を使い各物理Diskの状態を監視 Spun Upでなければアラート
- さくらの専用サーバのStorageは必ずRAID構成で提供され、Global側のさくらの専用サーバからSNMPで監視されている
  - Globalを閉じていたり、SNMPをfilterしていると監視されない。自前で監視し問題があれば保守を依頼
- SSDは壊れたことない

mackerel-plugin-ntpq



- offset(絶対値)とリモートとのSync状況の可視化
  - Sync < 0.1、offset > 300 でアラート
- 徐々に時間がずれていくサーバがあり、ntp.confの調整に可視化が便利

mackerel-plugin-linux-lite



- 2Coreから56Coreまでサーバがある
- 一貫した監視閾値を設けやすいように可視化
  - 空きメモリ
  - 100% 上限のCPU
  - コアあたりのロードアベレージ
  - プロセス数

z-common以外で利用するplugin

✂️ periodic-checker

```
$ periodic-checker --range 00:00-06:00,06:30-24:00 \
      -- check-tcp -p 80
```

- 特定の時間のみ監視を行う
- Daily メンテナンスなどへの対応
- mackerel-agentで欲しいなー

🔍 check-dns-rr

```
$ /etc/mackerel-agent/commands/check-dns-rr \
    --host=production.app.service.dc.consul -w 2 -c 2
OK: 3 dns-rr hosts found
```

- consul+DNSで稼働しているサービスの監視
- check-resolver同様、独自にresolv.confを読み込む
- 閾値以下の台数になるとアラート

🔍 check-spf-and-reserve-lookup

```
$ /usr/local/bin/check-spf-and-reserve-lookup 192.168.1.1 mercari.jp
NG: spf check failed: result=SoftFail
NG: reverse lookup dig failed: no result
```

- メール配信にて利用
- 該当IPがSPFレコードに含まれているか、逆引きした時にドメインが含まれているかを確認

🔍 check-spf-and-reserve-lookup-all

```
#!/bin/bash
set -e
for ip in $(ip addr | grep 'inet ' | fgrep -v 'inet 127.0.0.1' | grep -v -E
'^ *inet (10\.|192\.168|10\.|172\.1[6789]\.|172\.2[0-9]\.|172\.3[01]\.)' |
sort | awk '{print $2}' | awk -F / '{print $1}')
do
    /usr/local/bin/check-spf-and-reserve-lookup $ip mercari.jp
done
echo "OK: ALL"
```

- サーバが持っているGlobal IP全て確認

🔍 check-mysql-slave-sql-error

```
$ /usr/local/bin/check-mysql-slave-sql-error --user=monitor --password=xxx
mysql-slave-sql-error - MySQL slave SQL error CRITICAL: Last_SQL_Error
found: Error 'Table 'tmp_replication_stop' already exists' on query.
Default database: 'mercari'. Query: 'CREATE TABLE tmp_replication_stop ...
```

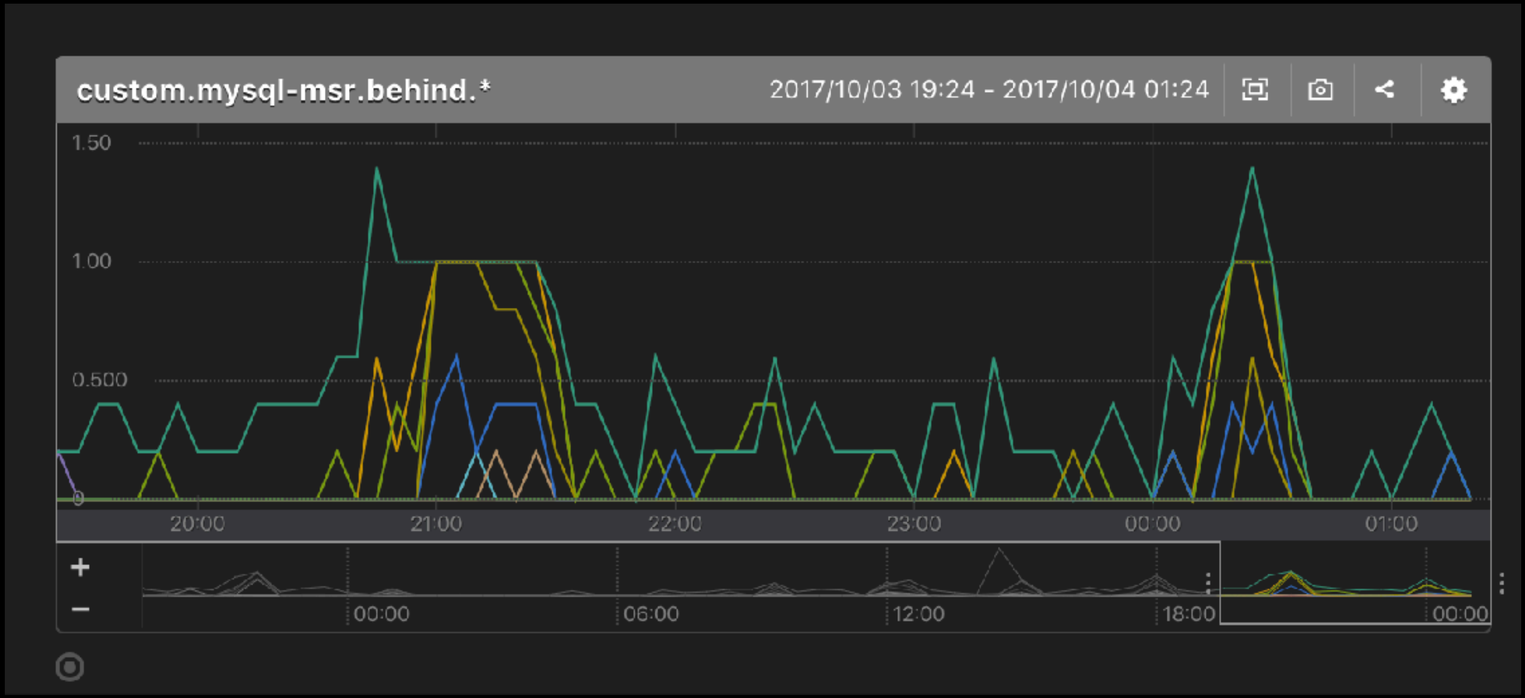
- 「レプリケーションが止まった時に、その理由も通知してくれると便利」で作ったplugin
- Multi Source Replication対応

🔍 check-mysql-msr

```
$ /usr/local/bin/check-mysql-msr --host=127.0.0.1 --port=3306 --
user=monitor --password=xxx -w 1 -c 1
MySQL Multi Source Replication OK: [0]
admin-db=io:Yes,sql:Yes,behind:0
main-db=io:Yes,sql:Yes,behind:0
web-db=io:Yes,sql:Yes,behind:0 (実際は1行)
```

- MySQLのMulti Source Replicationの監視
- 1つでも止まっていたり、閾値より遅延していたらアラート

📈 mackerel-plugin-msr



- check-mysql-msrの可視化
- mysql-msr.behind.* に対して監視が行えればcheck-mysql-msrは必要ない...？

Open Source!

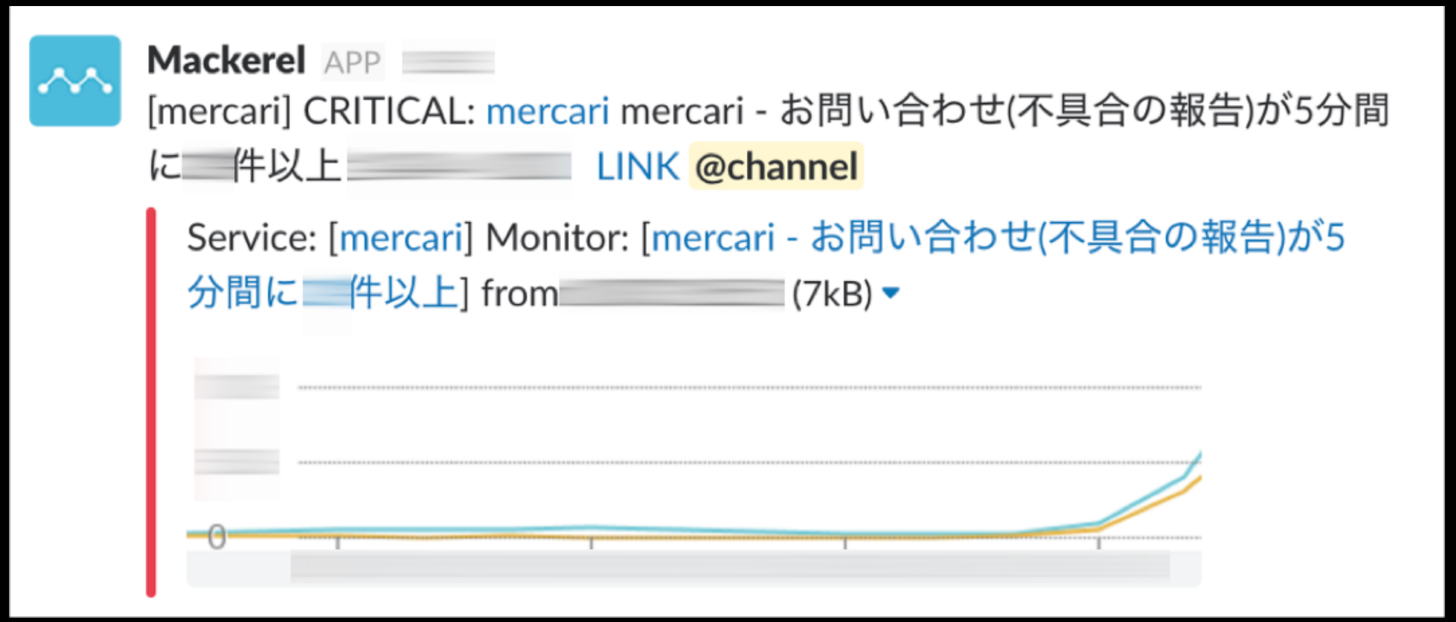
https://github.com/kazeburo/

https://github.com/kazeburo/custom-mackerel-plugins

足りないもの、見たいものがあれば教えてください!

その他の取り組み

問い合わせ数の監視



- 多くの人が参加するChannelへ通知
- 障害の検知、影響範囲の把握
- SREだけではなく、全チームが関わり対応の迅速化

監視されてないサーバの自動抽出

- mackerl APIとIPアドレス一覧を比較すれば出せる
- AWS/GCE ではprivate ipの一覧は各クラウドのAPIを使うことで取得可能
- さくら専用サーバ/さくらのクラウドでは取得できない
  - private networkを自前で管理するのが仕様

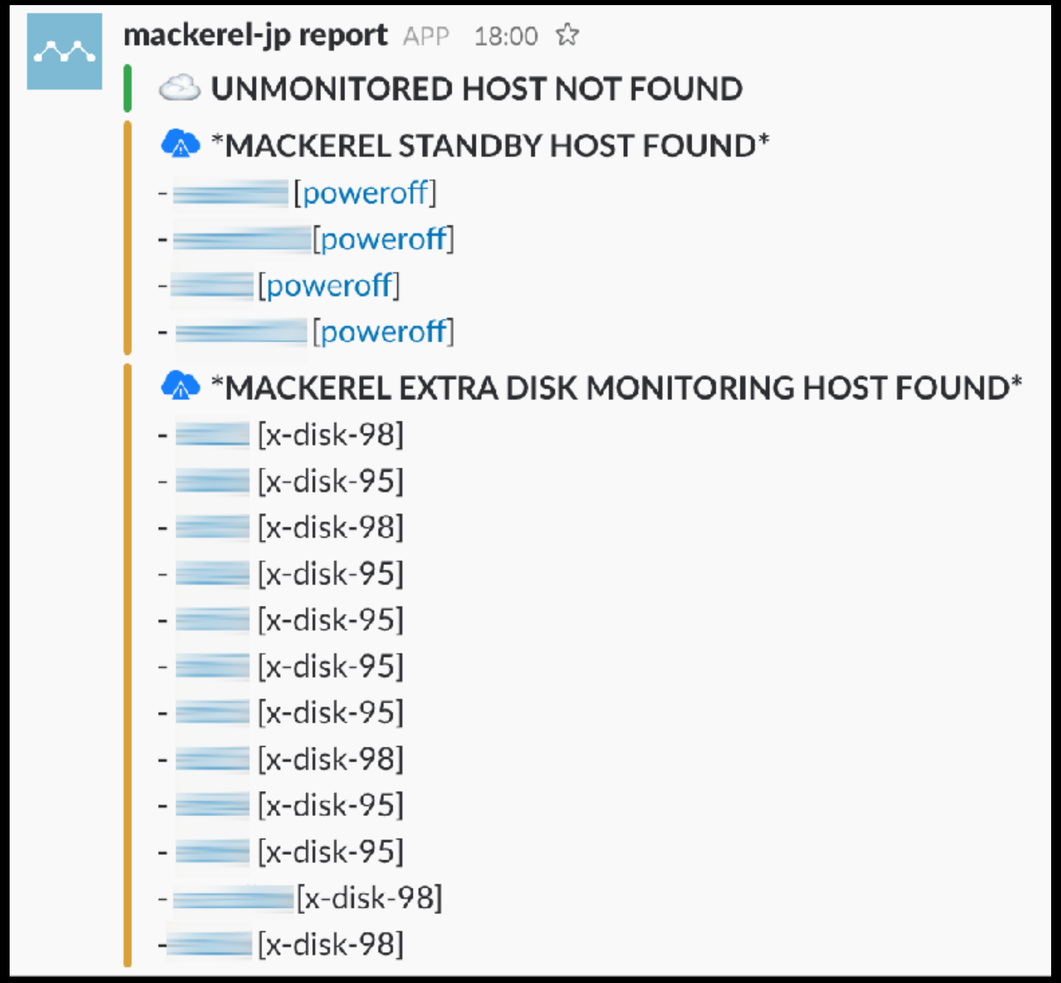
fpingによる疎通確認

```
$ seq 0 255 | xargs -I{} -P 16 fping -q -a -r 2 -i 10 -g x.y.{}.0/24
```

* xargsはイメージ

- fpingは並列して疎通確認するコマンド
- /24 のnetworkごとにfpingを実行しIP一覧を作成
- fpingをさらに並列化することで時間短縮

Slackへの通知



- 1日2回slackへ投稿
  - 監視されてないサーバ
  - standby、poweroff となっているサーバ
  - disk監視を緩めているサーバ
- JP/US/UKで実施

まとめ

コードを書いて問題を解決する
Mackerelは使いがいのあるツール

We're Hiring!

世界に挑む、メルカリ

多層多重多面インフラストラクチャ・信頼性で支えるSRE

www.mercari.com/jp/jobs/