SRA OSS, INC.

Zabbix 6.0 徹底解説

SRA OSS, Inc. 日本支社
OSS 事業本部　基盤技術グループ
赤松　俊弘

ZABBIX
PREMIUM PARTNER

whoami

- ☑ 赤松 俊弘（Toshihiro Akamatsu）
- ☑ SRA OSS, Inc. 日本支社
  OSS 事業本部 基盤技術グループ
- ☑ Zabbix 認定プロフェッショナル

職務

- ☑ PostgreSQL 以外の OSS 全般の技術サポート、構築
- ☑ 主に Zabbix を担当

アジェンダ

1. Zabbix とは

2. Zabbix 6.0 新機能解説
    1. Zabbix 6.0 について
    2. Zabbix 6.0 の新機能

3. アップグレード時の注意点

Zabbix とは

Zabbix とは

IT インフラやサービス、アプリケーションの可用性や性能を監視するための
エンタープライズ向け統合監視ソフトウェア

オープンソースソフトウェアとして開発・公開されており、無償で利用可能

開発元はラトビア共和国の Zabbix LLC

Zabbix 1.0 は 2004/3/23 リリース

コンポーネント

Zabbix server

- 全データ集約・保存
- 障害検知・通知

Zabbix Java gateway

- Java アプリケーション監視用デーモン
- JMX を利用

Zabbix agent

- 監視データ収集
- パッシブ・アクティブチェック

Zabbix sender

- コマンドラインユーティリティ
- コマンドラインから監視データ送信

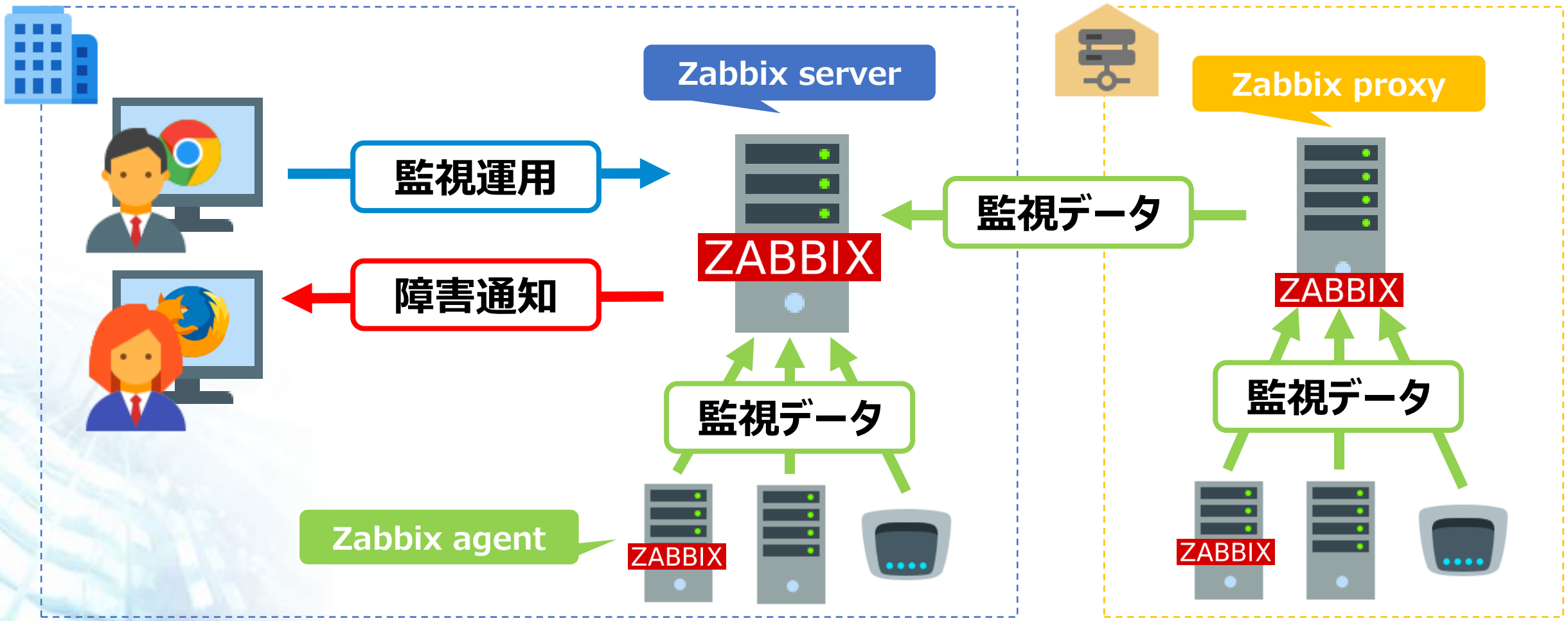
Zabbix proxy

- 監視データ収集の中継
- 負荷分散・集中監視

Zabbix get

- コマンドラインユーティリティ
- エージェントから任意の監視データを取得

アーキテクチャ



対応プラットフォーム

Zabbix server/proxy

- Linux
- IBM AIX
- FreeBSD
- NetBSD
- OpenBSD
- HP-UX
- Mac OS X
- Solaris

Zabbix agent

- Linux
- IBM AIX
- FreeBSD
- NetBSD
- OpenBSD
- HP-UX
- Mac OS X
- Solaris
- Windows

Zabbix agent 2

- Linux
- Windows

監視メトリクス（アイテム）

エージェント監視

リソース監視
- CPU
- メモリ
- ネットワーク
- ファイルシステム

ポート監視

ログ監視

プロセス/サービス監視

Web 監視

エージェントレス監視

ICMP 監視

DB 監視

SNMP 監視

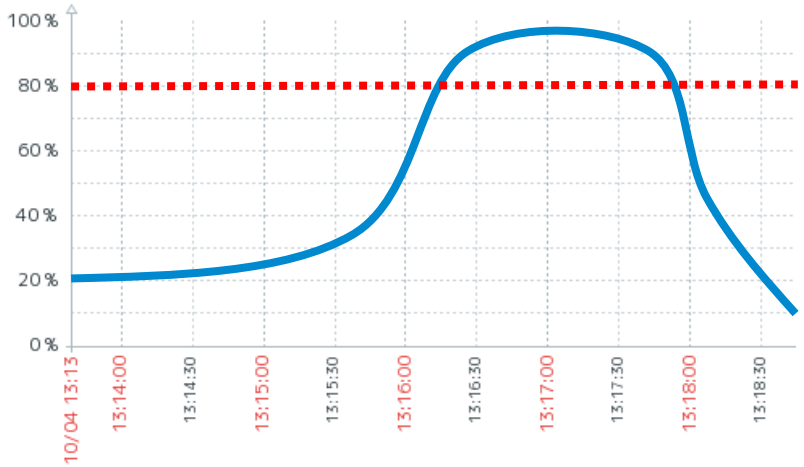
Java 監視

SSH/Telnet 監視

VMware 監視

障害検知（トリガー）

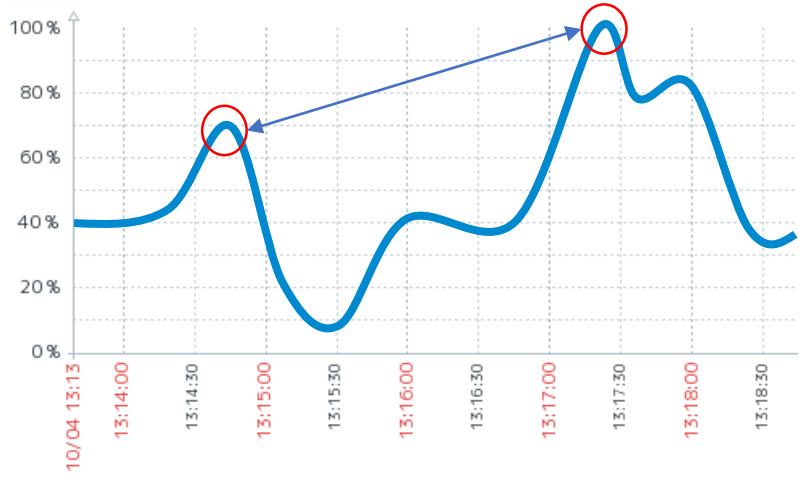
閾値



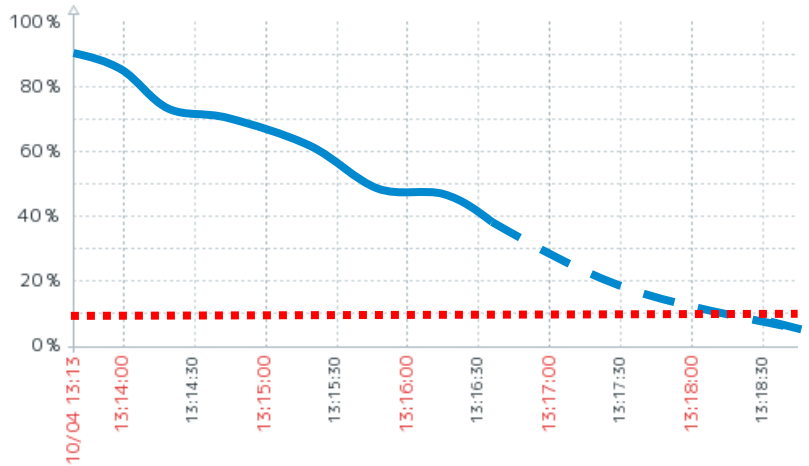
文字列

```
2022 02/14 09:13:25 [INFO] …
2022 02/14 09:15:10 [ERR] …
2022 02/14 09:21:47 [INFO] …
```

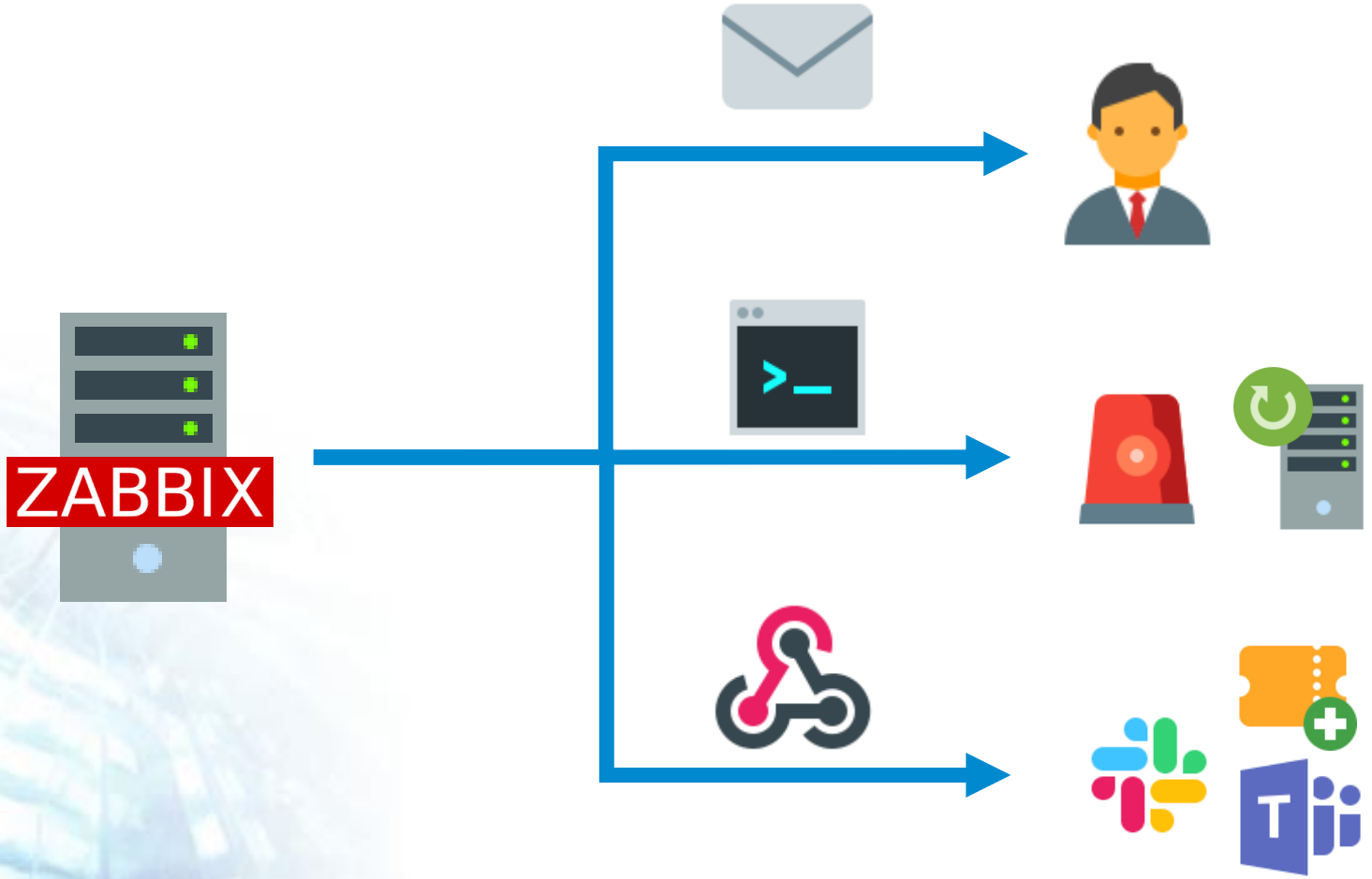
過去との比較



予測値



障害通知（アクション）



担当者にメール

HTML メールにも対応

コマンド実行

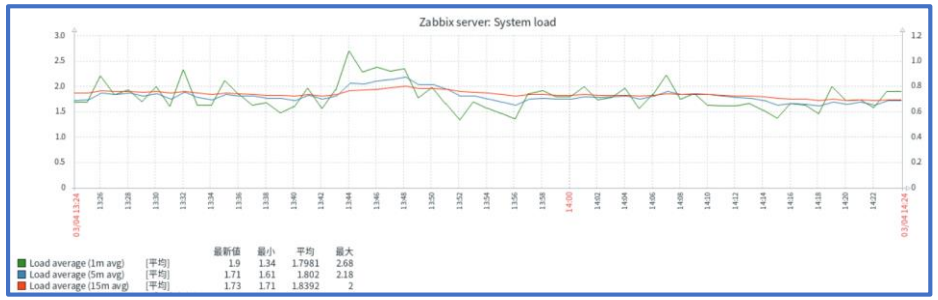
サーバやプロセスの再起動
パトランプの点灯

Webhook

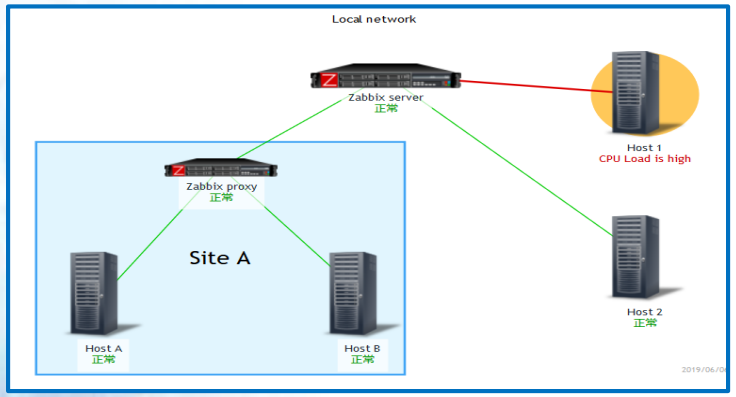
外部アプリとの連携
Slack, Teams, Redmine etc.

データの可視化

グラフ



ネットワークマップ

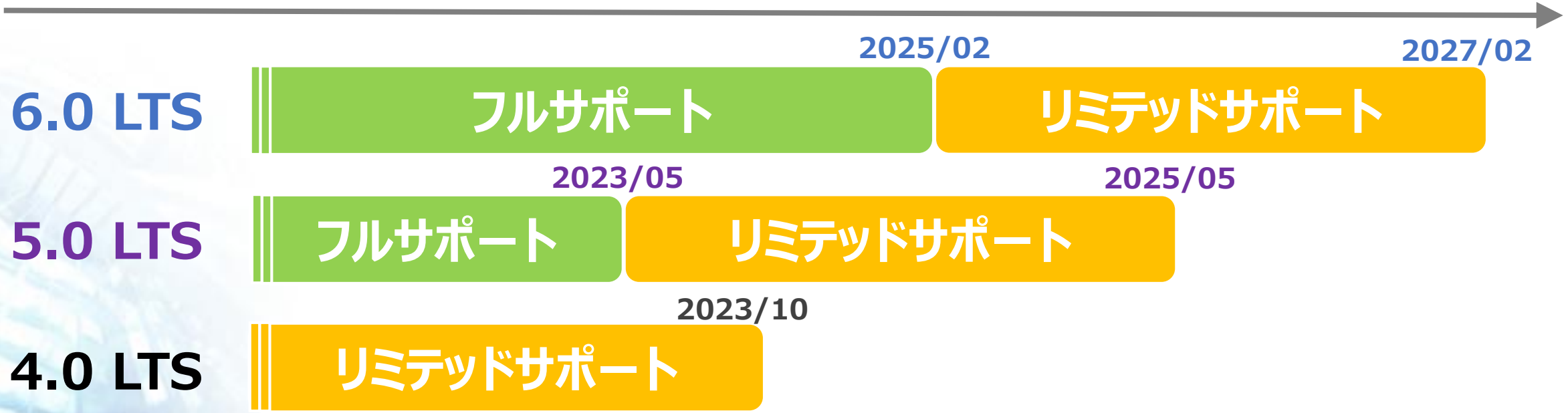


ダッシュボード

Zabbix 6.0 新機能解説

Zabbix 6.0

- 2022/02/15 リリース
- 長期サポート（LTS）バージョン
  - フルサポート 3 年、リミテッドサポート 2 年



Zabbix 6.0 対応データベース

DBMS	バージョン
MySQL/Percona	8.0.X
MariaDB	10.5.X ～ 10.6.X
PostgreSQL	13.X
Oracle	19.c ～ 21.c
TimescaleDB	2.0.1 ～ 2.3
SQLite	3.3.5 ～ 3.34.X

Zabbix 6.0 の新機能

ビジネスサービス監視

Zabbix server の
HA クラスタ

アノマリー検知と
ベースライン監視

Kubernetes 監視

新規ウィジェット

ETC　その他の新機能

ビジネスサービス監視

Zabbix のサービス監視とは

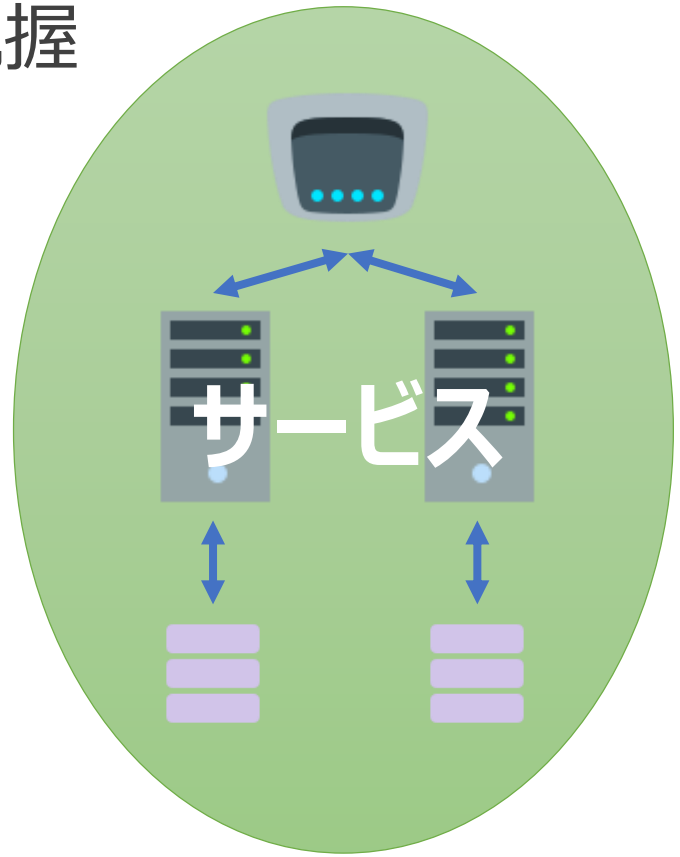
- 監視対象のインフラをハイレベル（ビジネスレベル）で把握

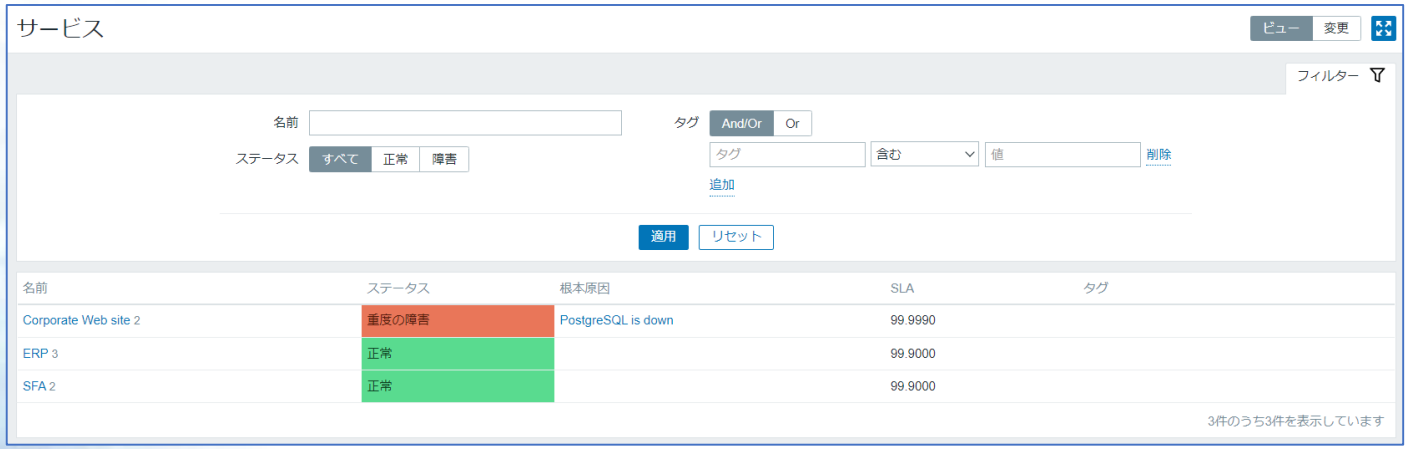
IT サービスの可用性

IT サービスの SLA

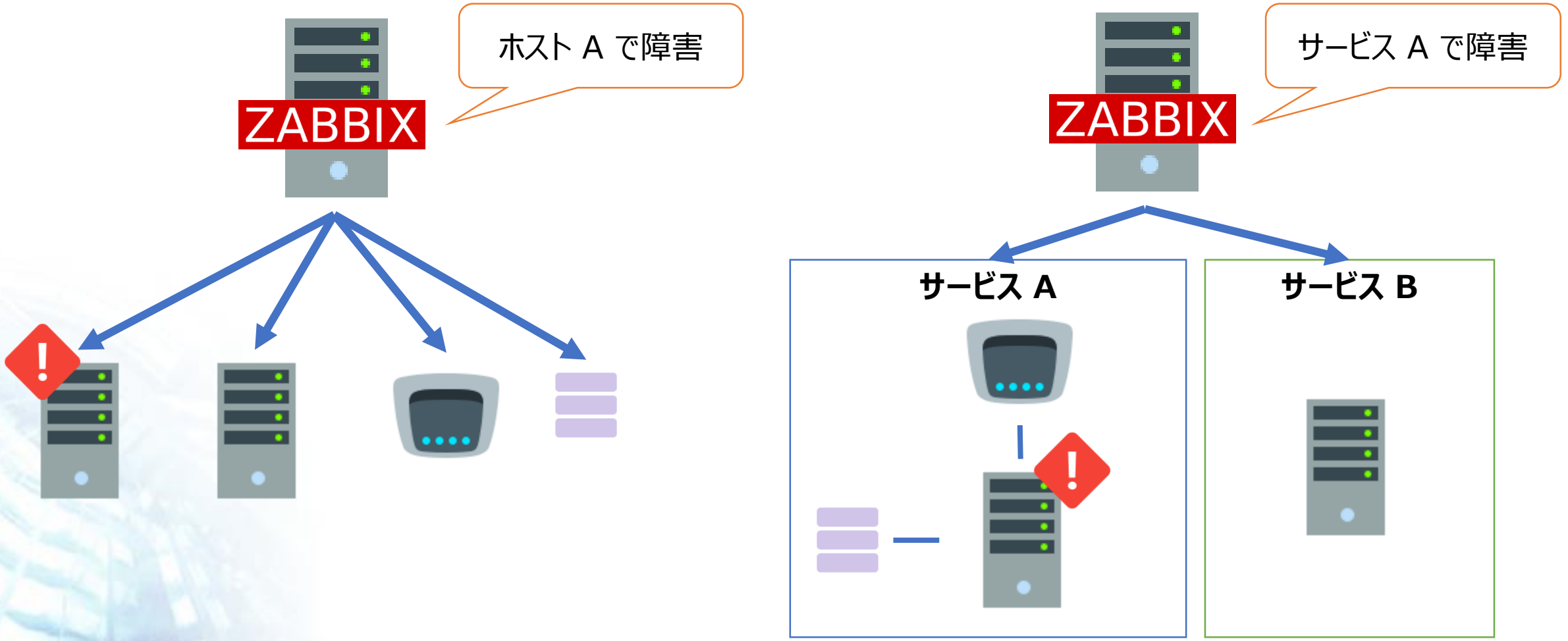
IT インフラの構造

IT インフラの弱点特定

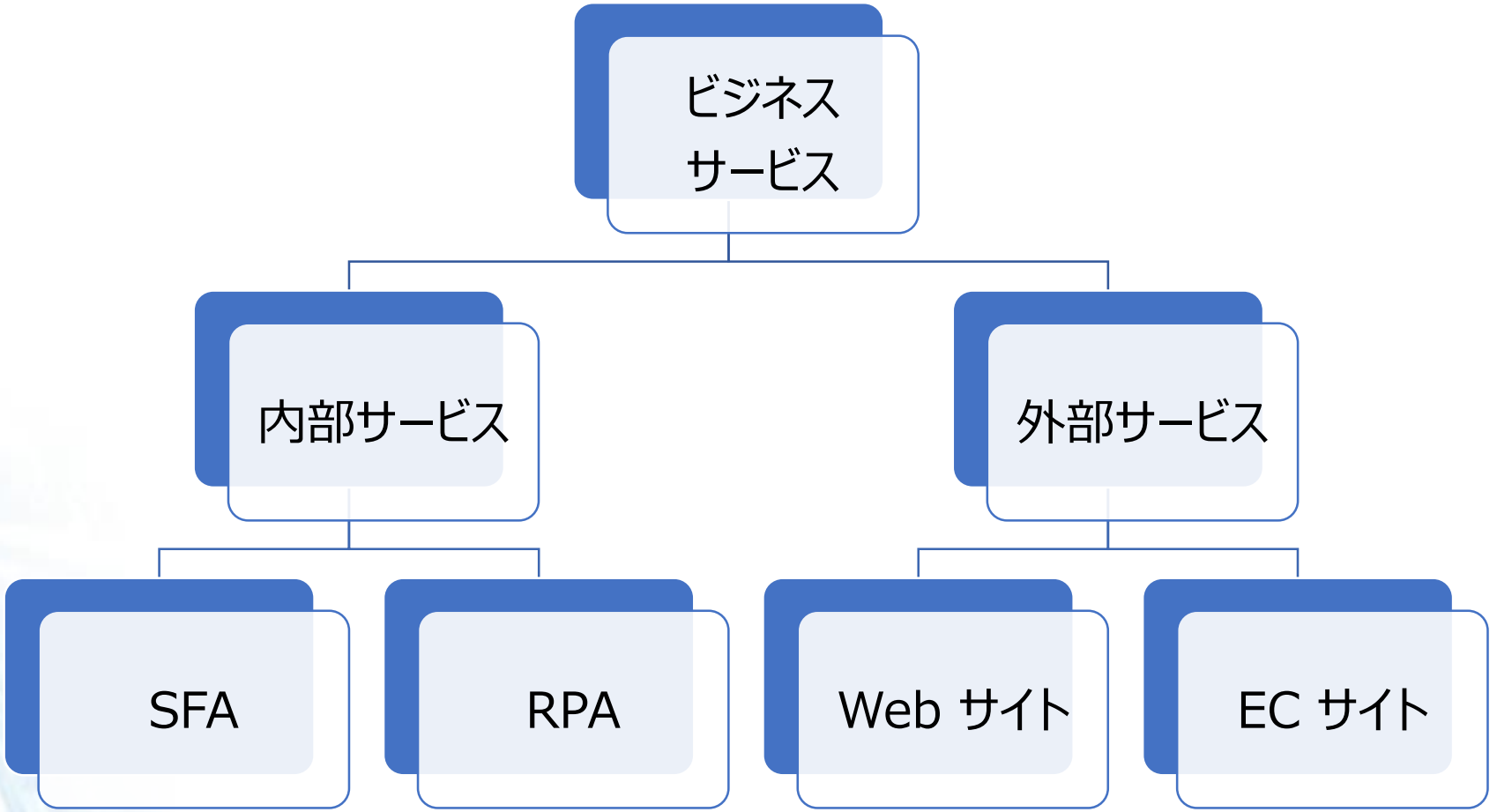




サービス監視のアーキテクチャ



サービスの階層化

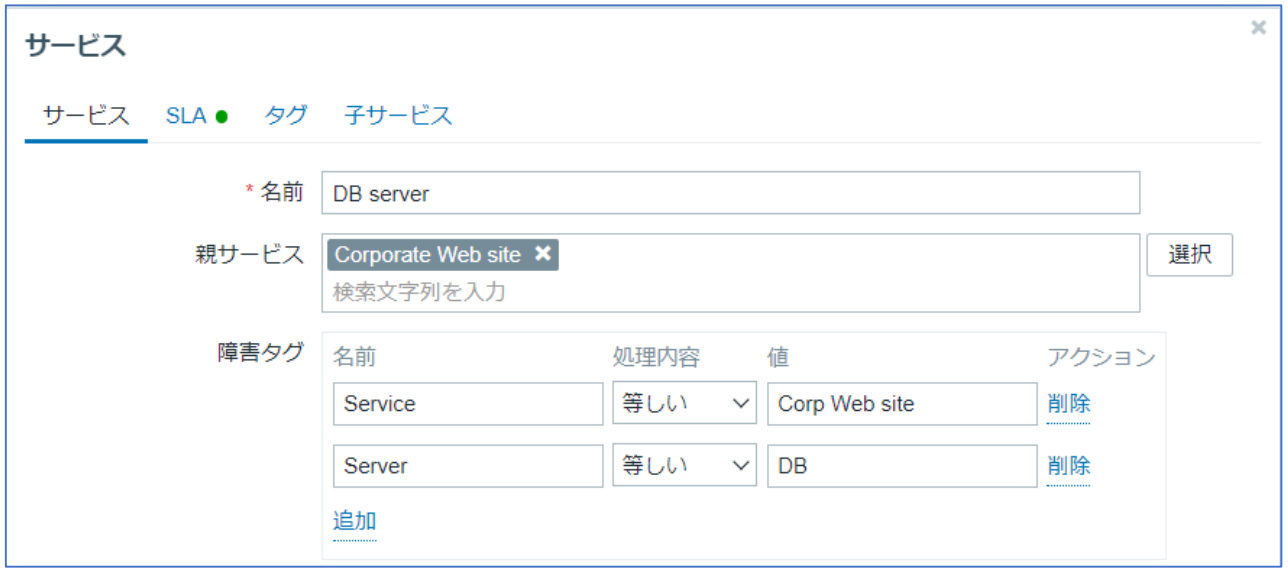


サービス監視の機能更新

- ビジネスレベルでのサービス監視を実現
  - タグによる障害の紐づけ
  - 柔軟なステータス計算および伝播
  - 根本原因の表示
  - サービスのステータス変化によるアラート通知
  - サービスへの権限付与

タグによる障害の紐づけ

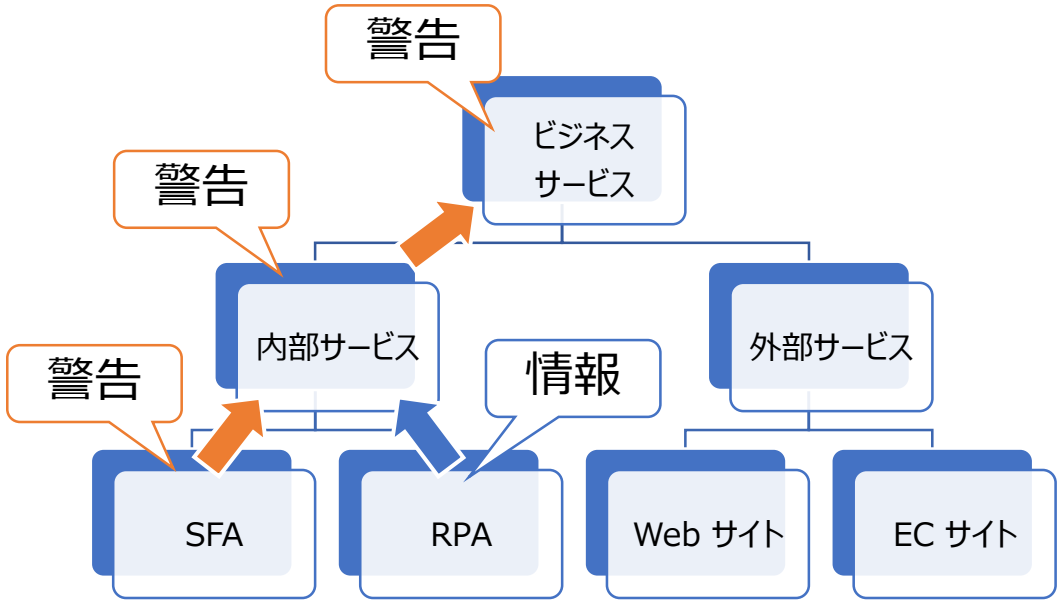
- サービスへの障害の紐づけがトリガーからタグに変更
- サービスと障害の関係が 1:1 から 1:N に
  - 冗長化されているサービスなど複雑な構成に対応可



柔軟なステータス計算および伝播（1）

- 子サービスのステータスから親サービスへのステータスを計算する方法

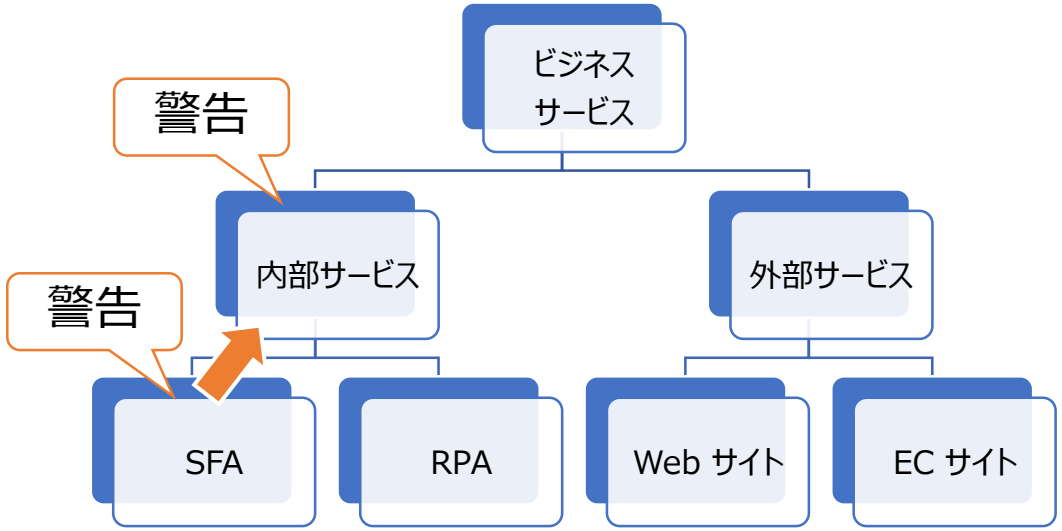
ステータス計算ルール	説明
子サービスで最も重要	子サービスの障害のうち最も高い深刻度を親サービスのステータスとする
すべての子が障害であったときに最も重要	すべての子サービスが障害であった場合に、その障害のうち最も高い深刻度を親サービスのステータスとする
ステータスを正常に設定	ステータスは計算せず常に正常



柔軟なステータス計算および伝播（2）

- さらに柔軟なステータス計算のための追加ルール
  - ルールに当てはまる場合に親に伝播する深刻度も合わせて指定

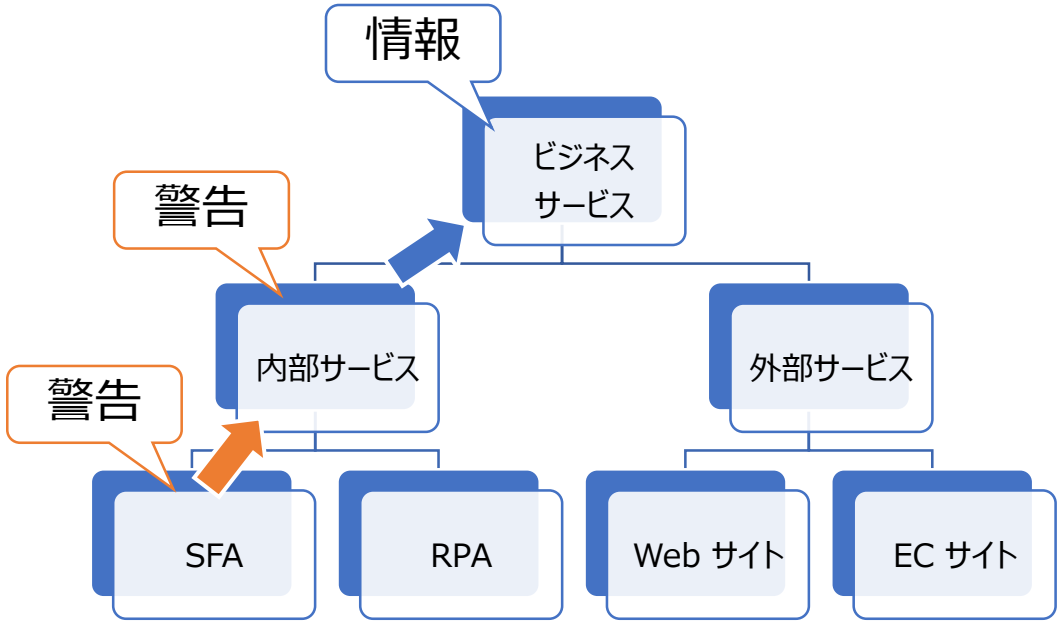
追加のルール
少なくとも **N** 個の子サービスの状態が**ステータス**以上
少なくとも **N%** の子サービスの状態が**ステータス**以上
N 個未満の子サービスのステータスが**ステータス**以下
子サービスの **N%** 未満のステータスが**ステータス**以下
ステータスが**ステータス**以上の子サービスの重みが少なくとも **W**
ステータスが**ステータス**以上の子サービスの重みが少なくとも **N%**
ステータスがステータス以下の子サービスの重みが **W** 未満
ステータスがステータス以下の子サービスの重みが **N%** 未満



柔軟なステータス計算および伝播（3）

- ステータス伝播時のルールも詳細に設定可能

ステータス伝播ルール	説明
そのまま	子サービスのステータスをそのまま伝播
深刻度を増やす	深刻度を 1 ～ 5 増やして伝播
深刻度を減らす	深刻度を 1 ～ 5 減らして伝播
このサービスを無視	ステータスを伝播しない
固定ステータス	常に同じステータスを伝播



根本原因の表示

- サービスのステータスに影響を与えている障害のリストを「根本原因」列に表示

名前	ステータス	根本原因	SLA
Corporate Web site 2	重度の障害	PostgreSQL is down	99.9990
ERP 3	正常		99.9000
SFA 2	正常		99.9000

根本原因に表示される障害は {SERVICE.ROOTCAUSE} マクロで取得可能

サービスのステータス変化によるアラート通知

- トリガーによる障害と同様にサービスのステータス変化時にもアクションによる通知が可能

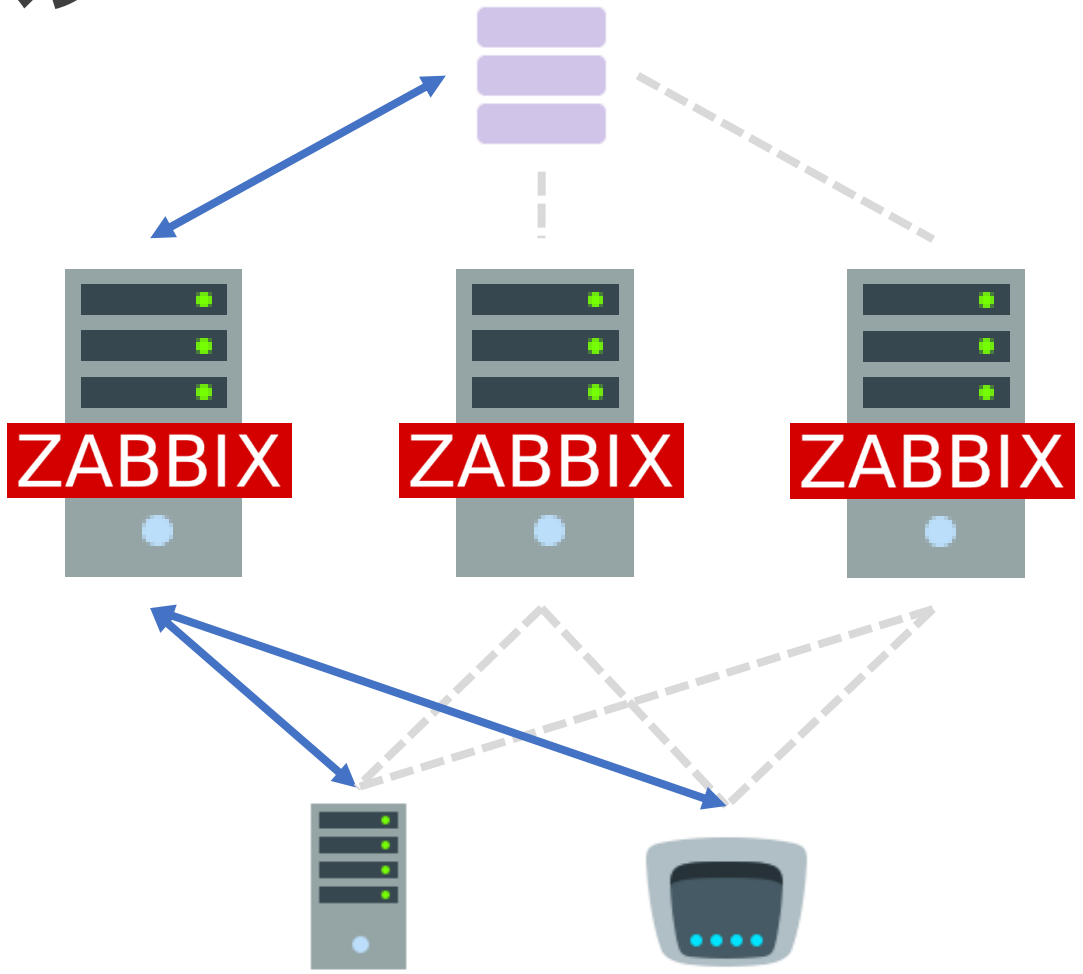
サービスへの権限付与

- ユーザロール単位でサービスへの Read-Write 権限が付与可能
  - 個々のサービス、またはタグによる指定が可能
  - サービスのマルチテナント対応

Zabbix server の HA クラスタ

Zabbix server の HA クラスタ

- Active/Standby
- 1 つの DB を共有
- マイナーバージョン間での互換性あり



HA クラスタの仕組み

- ha manager プロセスがクラスタを制御
  - アクティブ状態の Zabbix server では、全ての子プロセスが起動
  - スタンバイ状態の Zabbix server では、ha manager のみが起動
  - ha manager は DB にハートビートを送る

- クラスタの情報は DB に保存

```
zabbix=# select * from ha_node;
        ha_nodeid         |       name       |     address     | port  | lastaccess | status |       ha_sessionid
--------------------------+------------------+-----------------+-------+------------+--------+--------------------------
 ckvupwi8n0001rwnxvdlwfv0s | Zabbix server #3 | 133.137.175.151 | 10051 | 1636696278 |      0 | ckvvx4pae0000f9nxezi9hczg
 ckvupsmeq0001egpvksl6c5ih | Zabbix server #2 | 133.137.175.28  | 10051 | 1636710640 |      1 | ckvupys780000temnjy04t2pb
 ckvupihk70001z8mkpw5cg0u3 | Zabbix server #1 | 133.137.175.61  | 10051 | 1646708964 |      3 | cl08vj7000000r6mkoj1ie0k3
```

フェイルオーバ

- 全ノードが 5 秒おきに状態を更新
  - アクティブノードに異常（DB 上でハートビート未受信）があると、最初に検知したスタンバイノードが引き継ぐ
- ノードが時間内に応答しない場合
  - フェイルオーバ遅延時間 (デフォルト1分) に達するまで待機
  - 遅延時間経過後にいずれかのスタンバイノードが引き継ぐ
- フェイルオーバ遅延時間は、ランタイムコマンドから設定

```
# zabbix_server -R ha_set_failover_delay=2m
```

HA クラスタの設定 1

- Zabbix server (/etc/zabbix/zabbix_server.conf)
  - HANodeName にノード名を設定
    - このパラメータが設定されてると Zabbix server は HA モードになる
  - NodeAddress にノードアドレスを設定
    - フロントエンドがアクティブノードを判断するのに利用
  - DBHost に同じ DB を設定
    - 全ノードに同一の DB を参照させる
- フロントエンド (/etc/zabbix/web/zabbix.conf.php)
  - $ZBX_SERVER と $ZBX_SERVER_PORT をコメントアウト

HA クラスタの設定 2

- Zabbix agent (/etc/zabbix/zabbix_agent.conf)
  - パッシブ
    - Server パラメータに全ノードのアドレスをカンマ区切りで設定
  - アクティブ
    - ServerActive パラメータに全ノードのアドレスをセミコロン区切りで設定
- Zabbix proxy (/etc/zabbix/web/zabbix_proxy.conf)
  - パッシブ
    - Server パラメータに全ノードのアドレスをカンマ区切りで設定
  - アクティブ
    - Server パラメータに全ノードのアドレスをセミコロン区切りで設定

HA クラスタの状態確認および監視

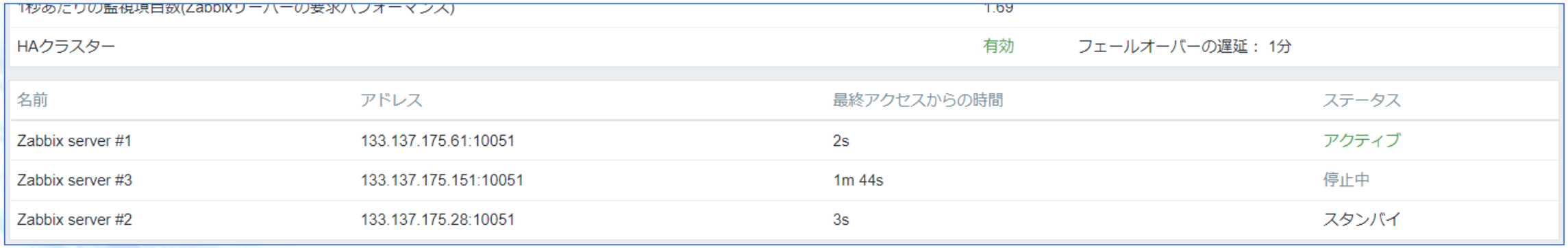
- 「システム情報」ウィジェット
  - HA　クラスタの状態
- 「レポート」 ⇒ 「システム情報」画面
  - HA　クラスタの状態と各ノードの状態
- ha_status ランタイムコントロールオプション
  - 各ノードの状態（zabbix_server.log に出力）
- zabbix[cluster,discovery,nodes] インターナルアイテム
  - 各ノードの状態（JSON 形式）

「システム情報」ウィジェット

- HA　クラスタの有効/無効
- フェイルオーバの遅延時間

「レポート」⇒「システム情報」画面

- ノード名
- アドレス
- 最終アクセスからの時間
- ステータス（アクティブ/スタンバイ/停止中/利用不可）



1秒あたりの監視項目数(Zabbixサーバーの要求パフォーマンス)		1.69	
HAクラスター		有効	フェールオーバーの遅延： 1分

名前	アドレス	最終アクセスからの時間	ステータス
Zabbix server #1	133.137.175.61:10051	2s	アクティブ
Zabbix server #3	133.137.175.151:10051	1m 44s	停止中
Zabbix server #2	133.137.175.28:10051	3s	スタンバイ

ha_status ランタイムコントロールオプション

- zabbix_server.log に各ノードの状態を出力

```
# zabbix_server -R ha_status
```

zabbix_server.log

```
#  ID                        Name              Address                Status   Last Access
1.  ckvupihk70001z8mkpw5cg0u3  Zabbix server #1  133.137.175.61:10051   active   0s
2.  ckvupsmeq0001egpvksl6c5ih  Zabbix server #2  133.137.175.28:10051   standby  1s
3.  ckvupwi8n0001rwnxvdlwfv0s  Zabbix server #3  133.137.175.151:10051  stopped  8m 12s
```

zabbix[cluster,discovery,nodes]

- 各ノードの情報を JSON 形式で取得
  - id
  - name
  - status
    - 0: standby
    - 1: stopped
    - 2: unavailable
    - 3: active
  - lastaccess (最終アクセス時刻 Unixtime)
  - address
  - db_timestamp（現在時刻 Unixtime）
  - lastaccess_age（最終アクセスからの経過秒）

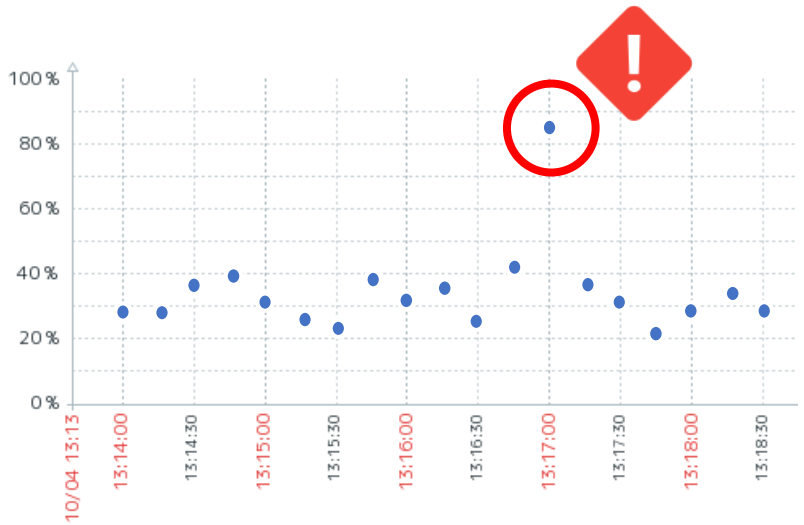
```
[
  {
    "id":"ckvupihk70001z8mkpw5cg0u3",
    "name":"Zabbix server #1",
    "status":3,
    "lastaccess":1636623048,
    "address":"133.137.175.61:10051",
    "db_timestamp":1636623050,
    "lastaccess_age":2
  },
  ...
]
```

アノマリー検知とベースライン監視

アノマリー検知とベースライン監視

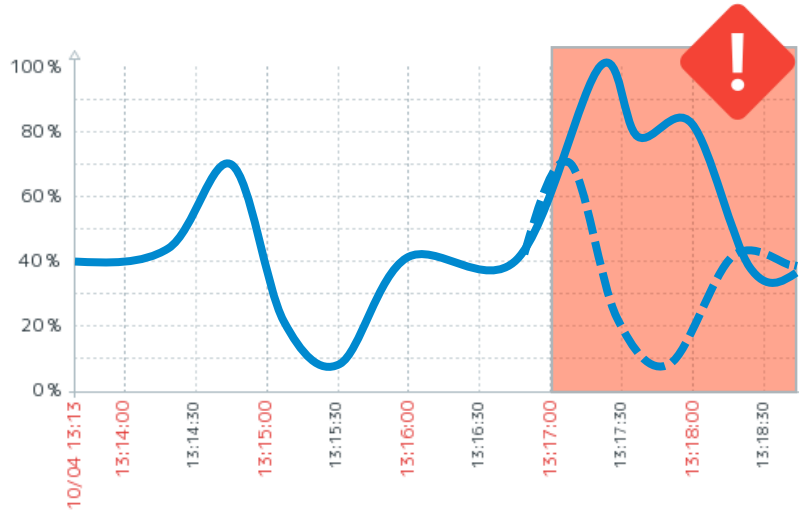
アノマリー検知

過去のデータと比べて大きく異なる値を検知



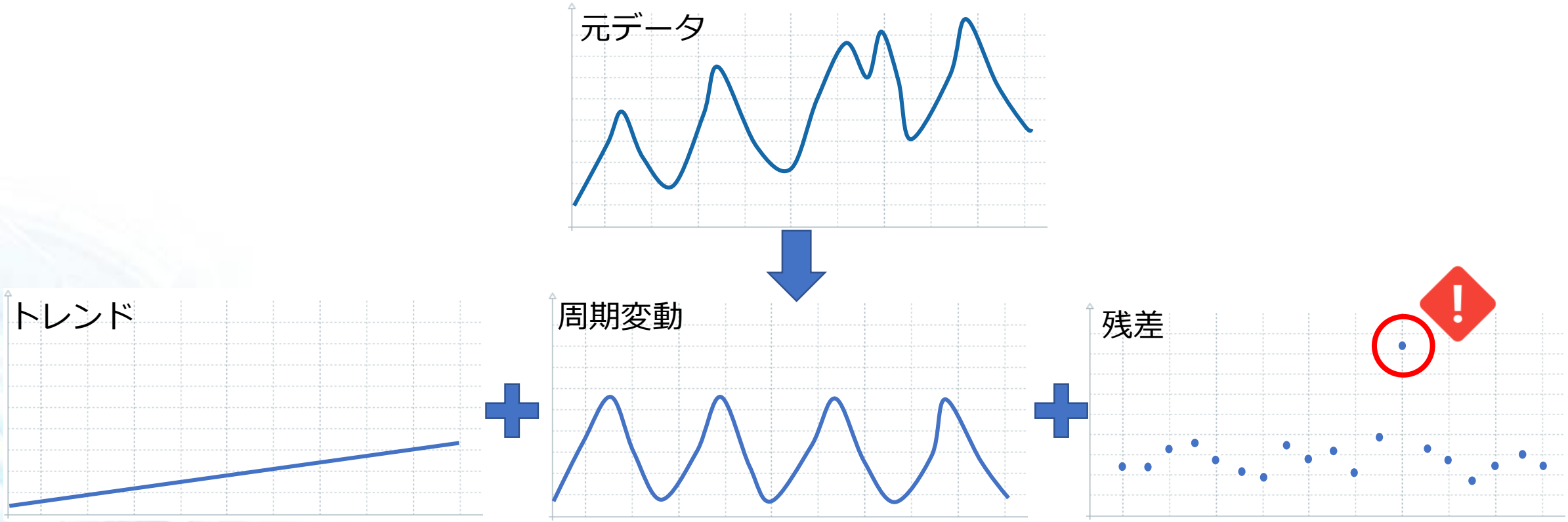
ベースライン監視

過去のデータの周期性から外れたデータを検知



アノマリー検知

STL 分解で元データをトレンド、周期変動、残差に分解し異常値を検知



アノマリー検知のトリガー関数

- trendstl(/host/key,eval period:time shift,
  detection period,season,<deviations>,<devalg>,<s_window>)
  - 全体のデータのうち異常値の数の割合を0～1の間の値で返す

パラメータ	説明
/host/key	ホストおよびアイテム
eval period	STL 分解に利用するデータの期間
detection period	アノマリー（異常値）の検知対象期間
season	周期の期間
deviations	異常値とする偏差値（デフォルト：3）
devalg	偏差アルゴリズム（デフォルト：mad）
s_windiw	loess ウィンドウのスパン （デフォルト：10 * 評価期間中のエントリ数 + 1）

アノマリー検知の設定例

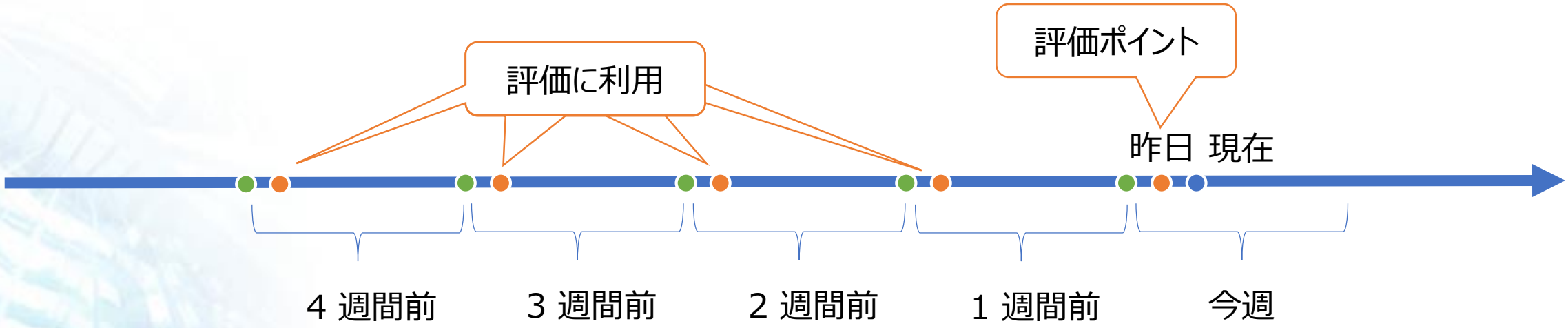
- trendstl(/host/key,100h:now/h,10h,2h,2.1,”mad”)
  - 過去 100 時間のトレンドデータを元に 2 時間を周期として
    直近 10 時間の異常率を返す
    ただし評価期間中の残りの系列の値が、その残りの系列の MAD (平均絶対偏差) の
    偏差値 2.1 に達した場合、異常とみなす

- trendstl(/host/key,1M:now/M-1y,1d,2h,,"stddevsamp")
  - 1 年前を起点に過去 1 ヶ月のトレンドデータを元に 2 時間を周期として
    直近 1 日の異常率を返す
    ただし評価期間中の残りの系列の値が、その残りの系列のサンプル標準偏差の
    偏差値 3 に達した場合、異常とみなす

ベースライン監視

- 過去のデータの指定期間の繰り返しから得られる値を監視に利用
  - トレンドデータから計算

- 期間: 1日、周期: 4週間の場合
  - 現在が火曜日の場合、4 週間の月曜日のデータを元に前日の月曜日を評価



ベースライン監視のトリガー関数

- baselinedev(/host/key,data period:time shift,season_unit,num_seasons)
  - 周期の中の指定した期間の標準偏差を返す
- baselinewma(/host/key,data period:time shift,season_unit,num_seasons)
  - 周期の中の指定した期間の平均値を返す

パラメータ	説明
/host/key	ホストおよびアイテム
data period	データ期間
season_unit	周期単位
num_seasons	周期数

ベースライン監視の設定例

- baselinedev(/host/key,1d:now/d,"M",6)
  - 前日と過去 6 ヶ月間の同じ日の標準偏差数（母数）を計算
- baselinewma(/host/key,1h:now/h,"d",3)
  - 昨日までの3日間のうち、最後の1時間を基準として平均値を計算

アノマリー検知とベースライン監視の評価方法

トリガー関数の計算結果を基準としてどう評価するかは
ユーザ次第

Kuberbetes 監視

Kubernetes 監視

- ノードと Pod の自動検出と監視
- ノードや Pod に関する情報をエージェントレスで収集
- ノードホストから詳細な情報を取得可能

- 以下のテンプレートが追加
  - Kubernetes nodes by HTTP
  - Kubernetes cluster state by HTTP
  - Kubernetes API server by HTTP
  - Kubernetes Controller manager by HTTP
  - Kubernetes Scheduler by HTTP
  - Kubernetes kubelet by HTTP

Kubernetes 監視導入ツール

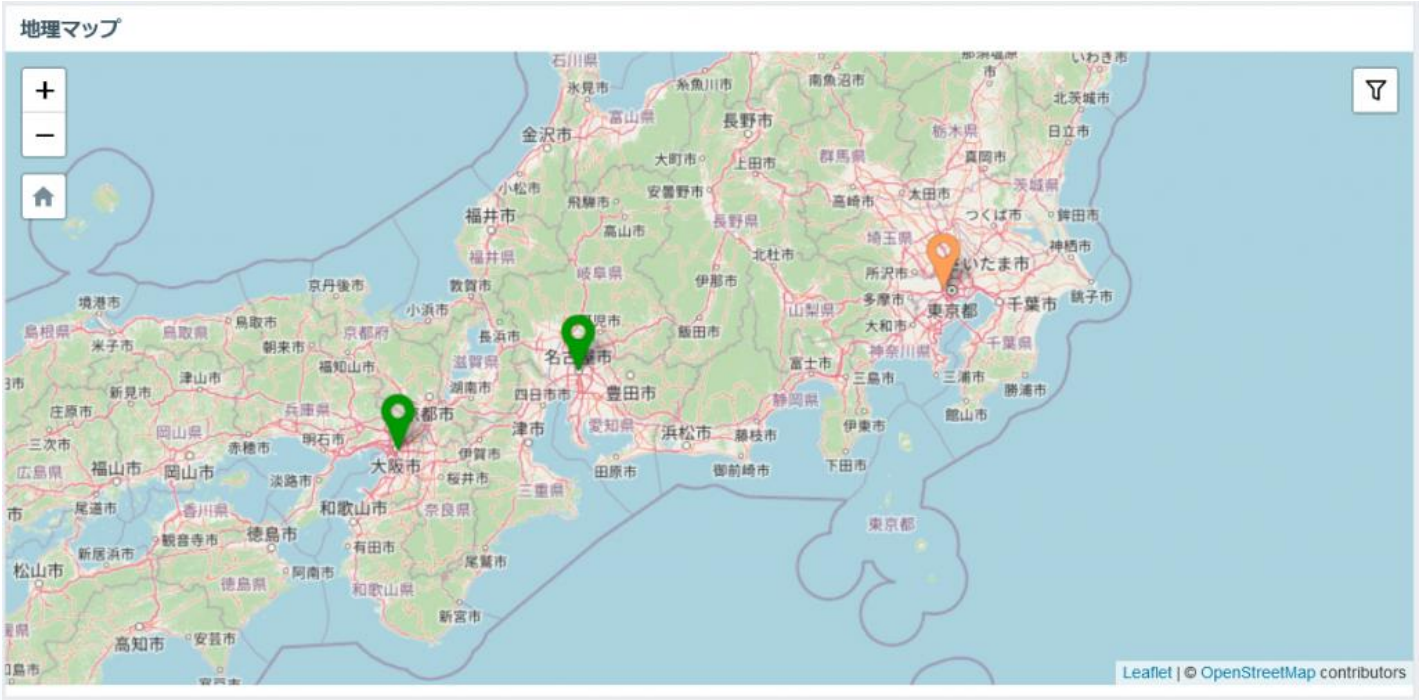
- Kubernetes 監視をフルに活用するためには
  Zabbix proxy と Zabbix agent を Kubernetes クラスタに導入する必要あり

- そのためのツールである Zabbix Helm Chart を提供
  - https://git.zabbix.com/projects/ZT/repos/kubernetes-helm/browse?at=refs%2Fheads%2Frelease%2F6.0
  - 要件
    - Kubernetes cluster 1.18 以上
    - Helm 3.0 以上
    - Zabbix server 6.0 以上
    - kube-state-metrics 2.13.2 以上

新規ウィジェット

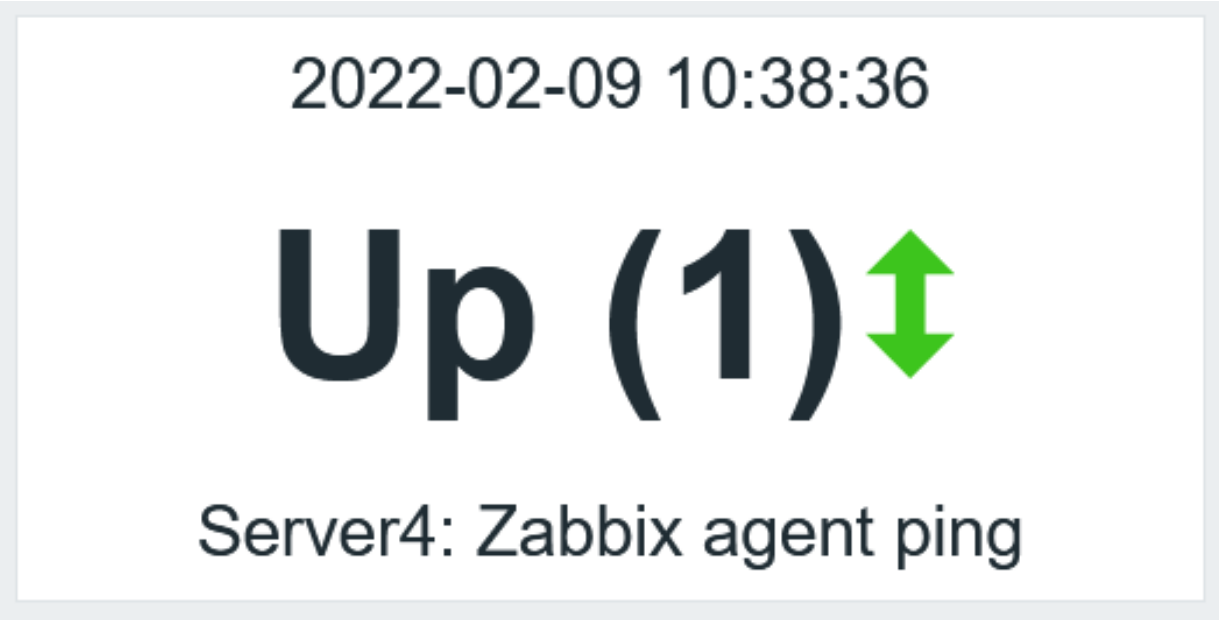
地理マップウィジェット

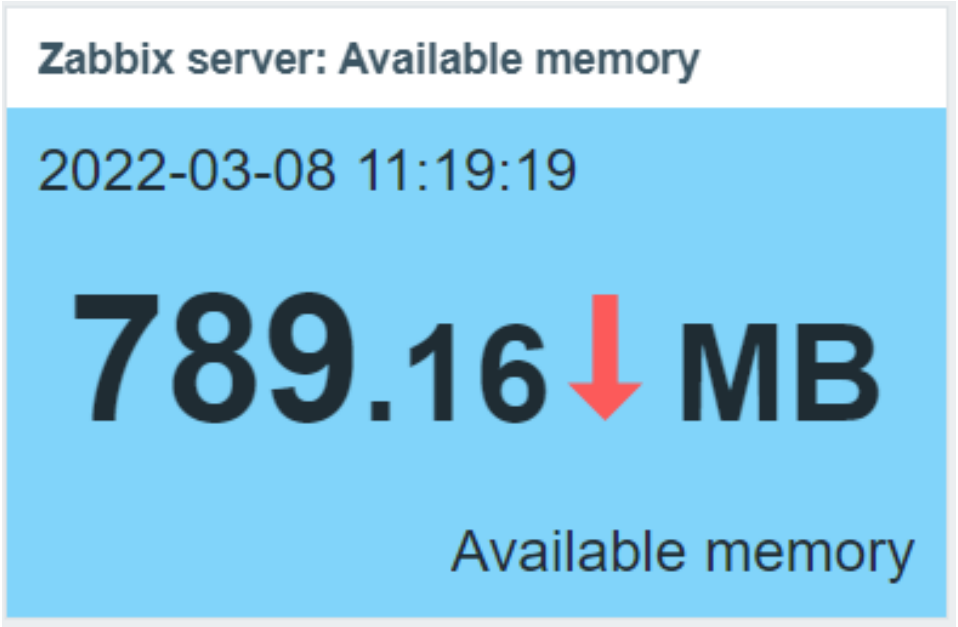
- 地図に監視対象の場所を表示
- 発生している障害によって色分け
- インベントリで緯度、経度の設定が必要



アイテムの値ウィジェット

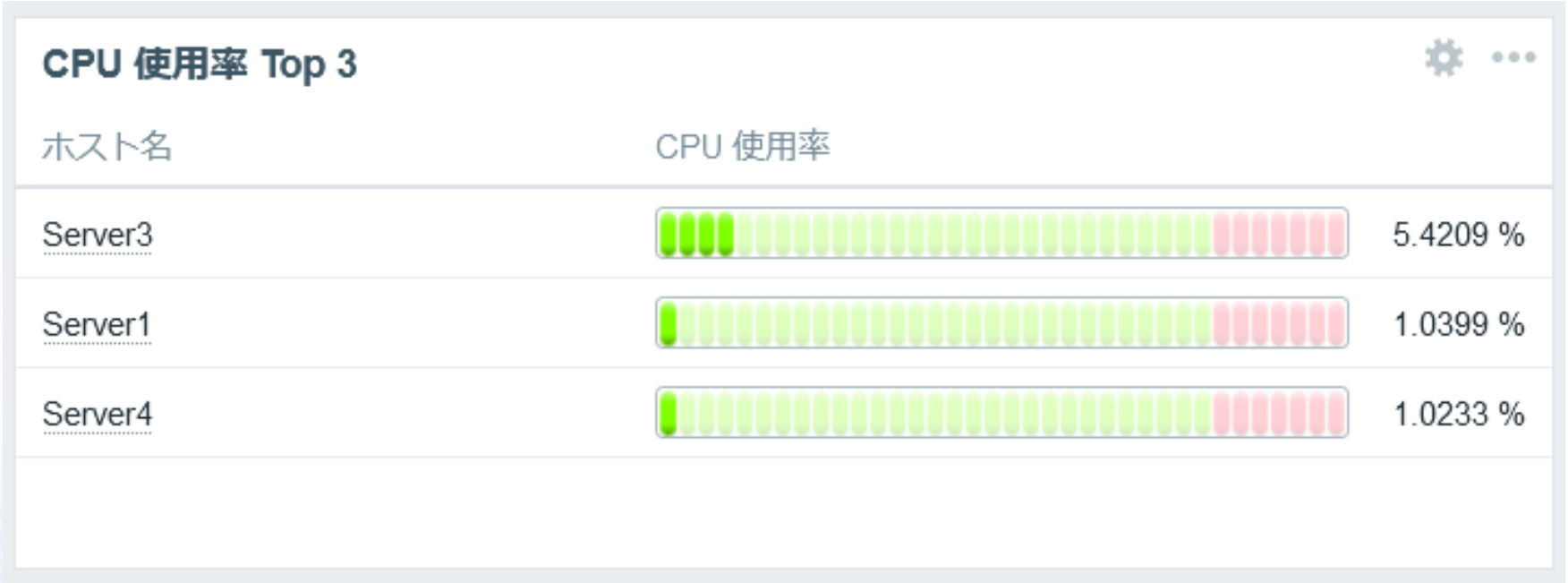
- 単一のアイテムの現在値を表示
- 表示スタイル、アイコンも変更可能





上位ホストウィジェット

- ユーザがカスタマイズ可能なデータ概要テーブルを表示
- 特定アイテムの TOP N やバーゲージ表示が可能



その他の新機能

データベースのバージョンチェック機能

- Zabbix server/proxy 起動時 DB のバージョンをチェック
- 対応バージョン外なら起動しない

zabbix_server.log

```
Unable to start Zabbix server due to unsupported PostgreSQL database
server version (10.17)

Must be at least (13.0)

Use of supported database version is highly recommended.

Override by setting AllowUnsupportedDBVersions=1 in Zabbix server
configuration file at your own risk.
```

データベースのバージョンチェック無効化（非推奨）

- 設定ファイルで「AllowUnsupportedDBVersions＝1」を設定

zabbix_server.conf

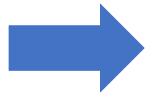
```
### Option: AllowUnsupportedDBVersions
#       Allow server to work with unsupported database versions.
#       0 - do not allow
#       1 - allow
#
# Mandatory: no
# Default:
# AllowUnsupportedDBVersions=0

AllowUnsupportedDBVersions=1
```

どのような障害が
発生するか
分からないので、
サポートバージョン内の
DB を使用することを
強く推奨

新規追加されたアイテムキー

- agent.hostmetadata : エージェントのホストメタデータ
- kernel.openfiles : 現在開いているファイルディスクリプタ数
- net.tcp.socket.count[] : パラメータに一致する TCP ソケット数
- net.udp.socket.count[] : パラメータに一致する UDP ソケット数
- vfs.dir.get[] : JSON 形式のディレクトリ情報
- vfs.file.get[] : JSON 形式のファイル情報
- vfs.file.owner[] : ファイルの所有者情報
- vfs.file.permissions[] : ファイルのパーミッション



詳細は Appendix II へ

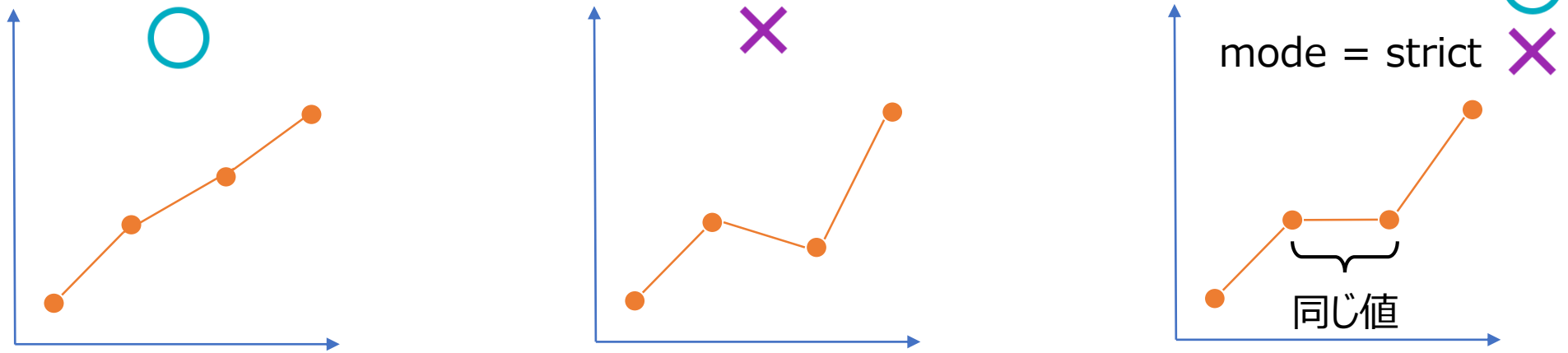
更新されたアイテムキー

- vfs.file.cksum[] : 第 2 パラメータ <mode> が追加（crc32 (デフォルト), md5, sha256）
- vfs.file.size[] : 第 2 パラメータ <mode> が追加（bytes (デフォルト), lines）
- vfs.fs.discovery : Windows の場合、ボリューム名を {#FSLABEL} マクロで取得
- vfs.fs.get : Windows の場合、ボリューム名を {#FSLABEL} マクロで取得

単調変化の検知

- monoinc() 関数　　：任意期間の単調増加を検知
- monodec() 関数　　：任意期間の単調減少を検知

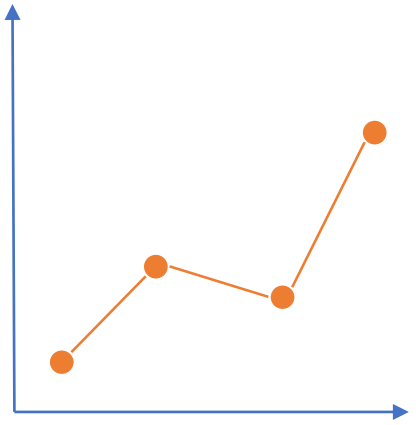
monoinc() の場合



変化回数の検知

- changecount() 関数 ：任意期間に隣接する値の間で
  変化があった回数を返す

mode
- all ：増減問わずカウント ⇒ 3
- inc ：増加のみカウント ⇒ 2
- dec ：減少のみカウント ⇒ 1



Prometheus パターンの保存前処理の演算子追加

- !=　：指定した文字列と等しくないラベルを選択
- !~　：指定した正規表現と等しくないラベルを選択

例
- node_filesystem_avail_bytes{device!="tmpfs"}
  ：device ラベルが tmpfs 以外

- wmi_service_state{name!~"^dhcp"}==1
  ：name ラベルが dhcp から始まるもの以外

Prometheus histogram 対応の関数

Prometheus histogram タイプのメトリクスは解析が困難だった

NEW
- rate() : 指定期間の 1 秒ごとの増加率を計算
- histogram_quantile() : 指定したヒストグラムのバケットのクォンタイルを計算
  PromQL の histogram_quantile に相当

上記 2 関数の補完用として以下関数も追加

- bucket_rate_foreach() : histogram_quantile のパラメータに利用
- bucket_percentile() : ヒストグラムのバケットからパーセンタイルを計算

新規追加された関数

- count()　　：foreach 関数によって返される配列の値を集計
- item_count()　：フィルタ条件に合うアイテム数を集計
  count(exists_foreach(item_filter)) に相当
- exists_foreach：フィルタ条件に合う有効なアイテム数を配列で返す

例
count(max_foreach(/*/proc.num[*], 1h))
⇒ 過去 1 時間でデータ収集が行われた
　proc.num アイテム数

count と履歴系の foreach関数 (max_foreach, avg_foreach など) を併用すると性能に影響が出る可能性があるが、exists_foreach では影響はない

トリガー関数デバッグ用マクロ

- トリガー条件式/復旧条件式を実際の値で解決した結果
  - {TRIGGER.EXPRESSION.EXPLAIN}
  - {TRIGGER.EXPRESSION.RECOVERY.EXPLAIN}
- トリガー条件式/復旧条件式の N 番目の関数を実際の値で解決した結果
  - {FUNCTION.VALUE<1-9>}
  - {FUNCTION.RECOVERY.VALUE<1-9>}

例
last(/host/system.cpu.util)>80 and monoinc(host/system.cpu.util, #2)=1

{TRIGGER.EXPRESSION.EXPLAIN} ⇒ 95>80 and 1=1

Github メディアタイプの追加

- Webhook で Github の issue を登録可能

監査ログ

- ディスカバリルールやアクション、スクリプトの実行を含む
  すべての処理が記録されるように改良
- 1 つの操作の結果として複数のログレコードが作成された場合、
  それらのレコードは同じレコード ID を持つように改良
- レコード ID でのフィルタ機能を追加
- 監査ログの有効/無効、保存期間の設定が
  一般設定の 1 セクションに分離

Zabbix ユーザのパスワードポリシー設定

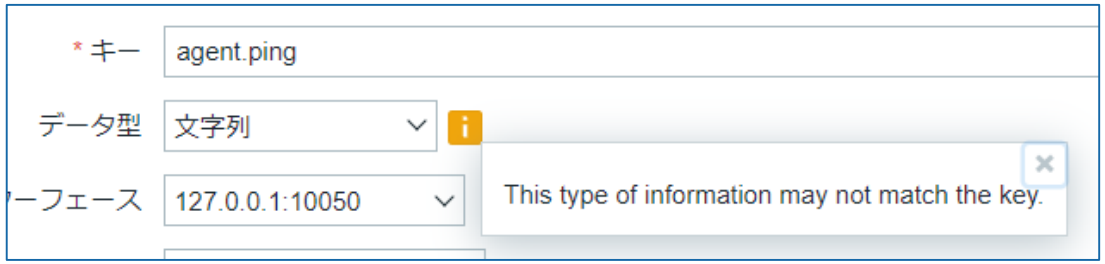
- 最小パスワード長
- 必須項目
  - 大文字小文字
  - 数字
  - 記号
- 推測が容易な文字列の禁止
  - よくあるパスワード
  - ユーザ名を含む

アイテムのデータ型の自動選択

- アイテムキーの選択時に
  対応するデータ型を自動選択
  - agent.ping ⇒ 数値（整数）
  - proc.cpu.util ⇒ 数値（浮動小数）
  - logrt ⇒ ログ

キーが複数のデータ型を返す
可能性がある場合は
自動選択はされない

誤ったデータ型を選択すると
警告が表示



フロントエンドのその他の変更点

- 「監視」 ⇒ 「ホスト」画面からホストの作成が可能
- ホスト編集画面がポップアップに変更
- 「監視」 ⇒ 「概要」画面が削除
  - データの概要、トリガーの概要ウィジェットで代替
- 言語からイギリス英語 (en_GB) が削除、
  デフォルトがアメリカ英語 (en_US) に変更
- メニューの Share のリンクがインテグレーション
  （Zabbix 公式サイトのインテグレーションページへのリンク）に変更
- フロントエンドのロゴをカスタマイズ可能

Zabbix get、Zabbix sender のタイムアウト

zabbix_get、zabbix_sender コマンドで
タイムアウトを指定するパラメータをサポート

```
-t <seconds>
--timeout <seconds>
```

- Zabbix get　　: 1 ～ 30 秒 (デフォルト: 30 秒)
- Zabbix sender : 1 ～ 300 秒 (デフォルト: 60 秒)

Zabbix agent 2 のプラグイン設定ファイルの分離

- プラグインの設定ファイルが /etc/zabbix/zabbix_agentd2.d/plugins.d 以下にプラグイン毎に分離して配置

```
# ls /etc/zabbix/zabbix_agent2.d/plugins.d/
ceph.conf  docker.conf  memcached.conf  modbus.conf  mongodb.conf
mqtt.conf  mysql.conf  oracle.conf  postgres.conf  redis.conf  smart.conf
```

その他

- Escalation Cancelled の通知の無効化
- MikroTik のテンプレートの追加
- Web 監視で圧縮コンテンツをサポート
- 保存前処理等で使用できる JavaScript で PATCH、HEAD、OPTIONS、TRACE、CONNECT メソッドおよび customRequest が利用可能
- TCP キューの最大サイズが設定可能
- シンプルマクロがエクスプレッションマクロに変更
- アイテムのポジショナルマクロ ($1, $2…) の削除
- MySQL での utf8mb4 のサポート

アップグレード時の注意点

アップグレードの前に

- 公式ドキュメントの新機能および
  アップグレードに関するページを確認
  - 1. Introduction – 5 What's new in Zabbix 6.0.0
  - 4. Installation – 7 Upgrade procedure
  - 4. Installation – 10 Upgrade notes for 6.0.0

Zabbix 6.0 アップグレード時の注意点 1

- RHEL で Zabbix server のパッケージは 8 系のみ提供
  - Zabbix proxy は RHEL 7 のパッケージも提供
  - Zabbix agent は RHEL 5、agent 2 は RHEL 6 以上
- Zabbix server と Zabbix proxy は要同一メジャーバージョン
  - Zabbix agent は後方互換性あり
- DB のサポートバージョンに注意
  - サポート外だとデフォルトでは起動しない
  - RHEL 8 の公式リポジトリの PostgreSQL、MySQL を利用するには RHEL 8.4 以上が必要

Zabbix 6.0 アップグレード時の注意点 2

- history テーブルへの主キー追加は別途スクリプトを実行
  - スクリプトは zabbix-sql-scripts パッケージに同梱
  - テーブルの変換には history テーブル分の空き容量が必要

- PCRE が PCRE2 に変更
  - 正規表現の記述方法に一部違い
  - ^[¥w-¥.] ⇒ ^[-¥¥w¥¥.]

- 監査ログの履歴は移行不可
  - 6.0 からの監査ログのスキーマの変更による

Zabbix 6.0 アップグレード時の注意点 3

- アイテム名でのポジショナルマクロ ($1, $2 ...) のサポート削除
  - アップグレードで自動変換はされない
- アイテム名でのユーザマクロのサポート削除

参考

- Zabbix 6.0 manual
  - https://www.zabbix.com/documentation/6.0/en/manual
- What's new in Zabbix 6.0.0
  - https://www.zabbix.com/documentation/6.0/manual/introduction/whatsnew600
- Zabbix roadmap
  - https://www.zabbix.com/jp/roadmap

SRA OSS Tech Blog

- 弊社 Tech Blog では Zabbix を始め、
  様々な OSS の技術情報を掲載
- Zabbix 6.0 に関する技術情報も随時掲載予定

- SRA OSS Tech Blog
  - https://www.sraoss.co.jp/tech-blog/

オープンソースとともに

Appendix I

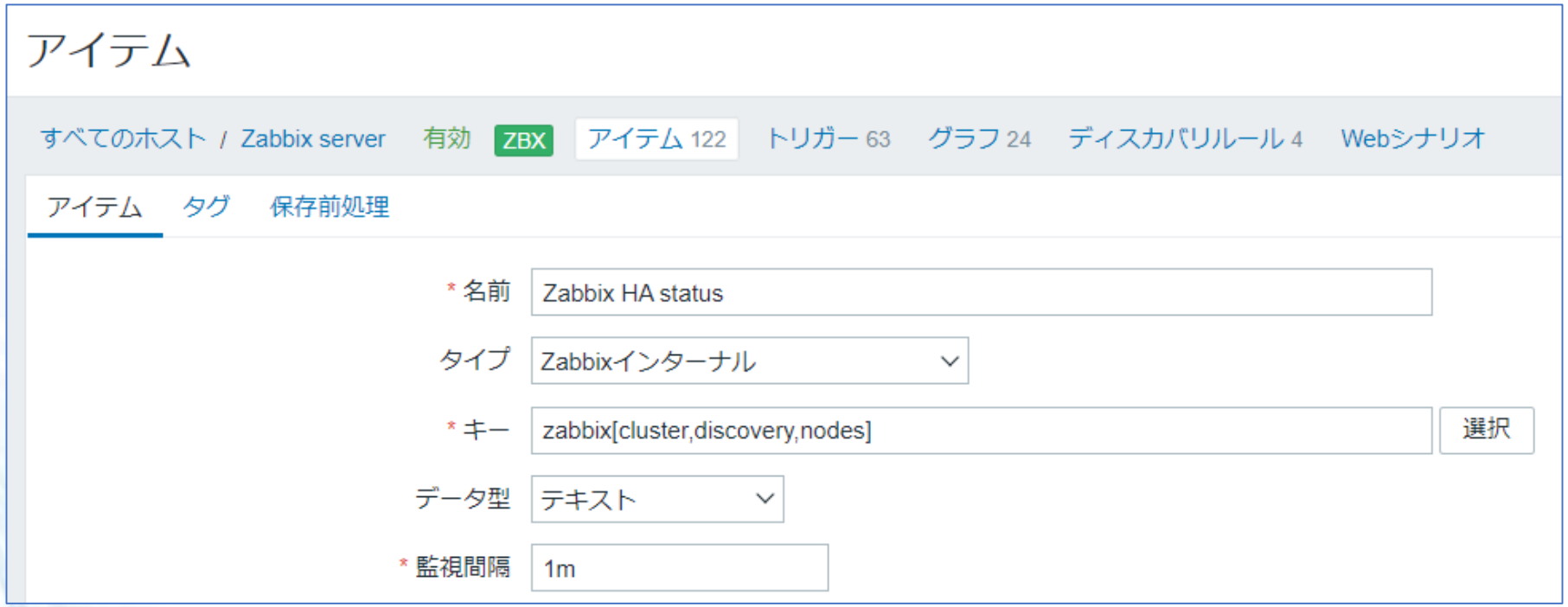
HA クラスタの状態確認と監視

HA の状態をアイテムで取得してみよう（1）

- 値のマッピング
  - status の数値と対応する状態をマッピング

HA の状態をアイテムで取得してみよう（2）

- アイテム
  - zabbix[cluster,discovery,nodes] で各ノードの状態一覧を取得

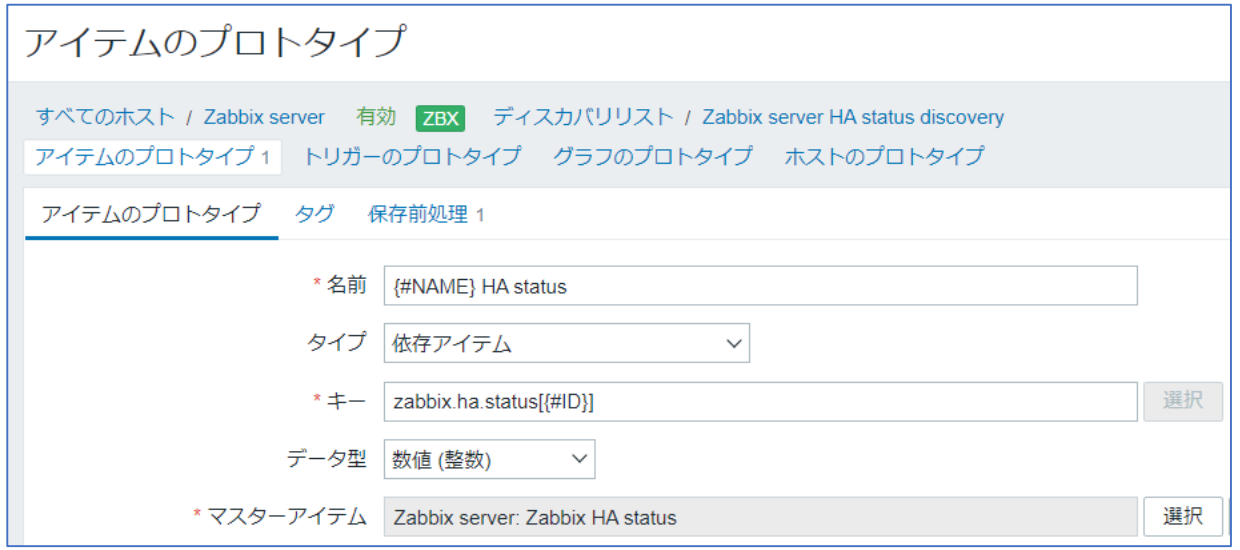


HA の状態をアイテムで取得してみよう（3）

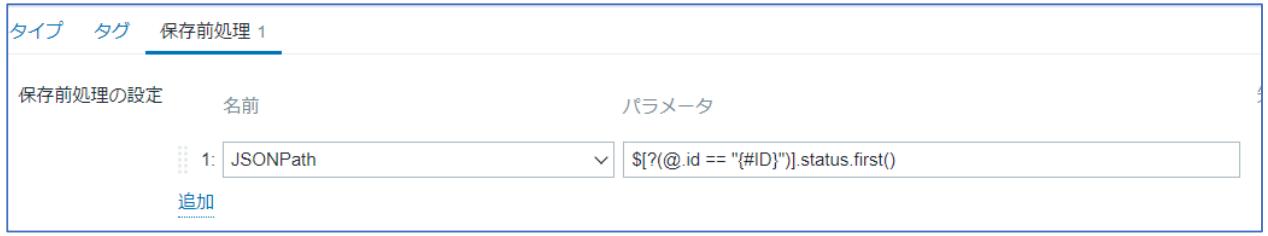
- LLD ルール
  - Zabbix HA Status の依存アイテムとして作成
  - LLD マクロで id と name を抽出

HA の状態をアイテムで取得してみよう（4）

- アイテムのプロトタイプ
  - Zabbix HA Status の依存アイテムとして作成
  - アイテム名に {#NAME} キーに {#ID} を使用



  - 保存前処理に JSONPath で $[?(@.id == “{#ID}”)].status.first() を設定



HA の状態をアイテムで取得してみよう（5）

- 取得結果
  - トリガー設定はよしなに

最新データ

ホスト	名前 ▲	最新のチェック時刻	最新の値
Zabbix server	Zabbix server #1 HA status	2021/11/12 14:51:51	Active (3)
Zabbix server	Zabbix server #2 HA status	2021/11/12 14:51:51	Standby (0)
Zabbix server	Zabbix server #3 HA status	2021/11/12 14:51:51	Stopped (1)

Appendix II

新規追加されたアイテムキー

agent.hostmetadata

- エージェントの**ホストメタデータ**を返す
  - タイプ　　：Zabbix エージェント
    Zabbix エージェント（アクティブ）
  - データ型　：文字列
  - パラメータ：なし

ホストメタデータ

1. zabbix_agentd.conf の HostMetadata に設定した文字列
2. HostMetadataItem に設定したアイテムキーの取得結果
3. 両方設定していなければ空文字列

kernel.openfiles

- 現在開いている**ファイルディスクリプタ数**を返す
  - タイプ　　：Zabbix エージェント
    Zabbix エージェント（アクティブ）
  - データ型　：数値（整数）
  - パラメータ：なし

net.tcp.socket.count[]

- パラメータに一致する **TCP ソケット数**を返す
  - タイプ　　：Zabbix エージェント
    Zabbix エージェント（アクティブ）
  - データ型　：数値（整数）
  - パラメータ：<laddr>　ローカルの IPv4/v6 アドレス（CIDR 表記可）
    <lport>　ローカルのポート番号もしくはサービス名
    <raddr>　リモートの IPv4/v6 アドレス (CIDR 表記可)
    <rport>　リモートのポート番号もしくはサービス名
    <state>　接続状態（established, syn_sent, syn_recv, fin_wait1, fin_wait2, time_wait, close, close_wait, last_ack, listen, closing）

net.udp.socket.count[]

- パラメータに一致する **UDP ソケット数**を返す
  - タイプ　　：Zabbix エージェント
    Zabbix エージェント（アクティブ）
  - データ型　：数値（整数）
  - パラメータ：<laddr>　ローカルの IPv4/v6 アドレス（CIDR 表記可）
    <lport>　ローカルのポート番号もしくはサービス名
    <raddr>　リモートの IPv4/v6 アドレス (CIDR 表記可)
    <rport>　リモートのポート番号もしくはサービス名
    <state>　接続状態（established, unconn）

vfs.dir.get[]

- ディレクトリの情報を JSON 形式で返す
  - タイプ　　: Zabbix エージェント
    Zabbix エージェント（アクティブ）
  - データ型　　: テキスト
  - パラメータ　　: dir　　ディレクトリのパス
    <regex_incl>　　含めるエンティティの正規表現
    <regex_excl>　　除外するエンティティの正規表現
    <types_incl>　　リストする種類 (file, dir, sym, sock, bdev, cdev, fifo, dev, all)
    <types_excl>　　リストしない種類
    <max_depth>　　探索するサブディレクトリの深さ
    <min_size>　　リストする最小 byte 数
    <max_size>　　リストする最大 byte 数
    <min_age>　　リストする最小タイムスタンプ
    <max_age>　　リストする最大タイムスタンプ
    <regex_excl_dir>除外するディレクトリの正規表現


```
[
  {
    "basename":"zabbix_agentd.d",
    "pathname":"/etc/zabbix/zabbix_agentd.d",
    "dirname":"/etc/zabbix",
    "type":"dir",
    "user":"root",
    "group":"root",
    "permissions":"0755",
    "uid":0,
    "gid":0,
    "size":4096,
    "time": {
      "access":"2021-11-10T04:16:58+0900",
      "modify":"2021-11-10T00:50:35+0900",
      "change":"2021-11-10T04:16:56+0900“
    },
    "timestamp": {
      "access":1636485418,
      "modify":1636473035,
      "change":1636485416
    }
  },
]
```

vfs.file.get[]

- ファイルの情報を JSON 形式で返す
  - タイプ　　：Zabbix エージェント
    Zabbix エージェント（アクティブ）
  - データ型　：テキスト
  - パラメータ：file　ファイルのフルパス


```
{
  "type": "file",
  "user": "root",
  "group": "zabbix",
  "permissions": "0600",
  "uid": 0,
  "gid": 975,
  "size": 25903,
  "time": {
    "access": "2021-11-09T11:22:32+0900",
    "modify": "2021-11-09T11:22:28+0900",
    "change": "2021-11-09T11:22:28+0900"
  },
  "timestamp": {
    "access": 1636424552,
    "modify": 1636424548,
    "change": 1636424548
  }
}
```

vfs.file.owner[]

- ファイルの所有者情報を返す
  - タイプ　　：Zabbix エージェント
Zabbix エージェント（アクティブ）
  - データ型　：文字列
  - パラメータ：file　　ファイルのフルパス
<ownertype>　所有者の種類（user, group）
<resulttype>　所有者の表示形式（name, id）

vfs.file.permissions[]

- ファイルのパーミッション情報（8 進数 4 桁）を返す
  - タイプ　　：Zabbix エージェント
    Zabbix エージェント（アクティブ）
  - データ型　：文字列
  - パラメータ：file　　ファイルのフルパス

Windows の Zabbix agent ではサポートされない