Zabbixによるサーバー監視の基礎

株式会社SRA OSS
金　範起

概要

- 監視について
- Zabbixの概要
- Zabbix構築の流れ
- 基本的な監視設定

会社紹介

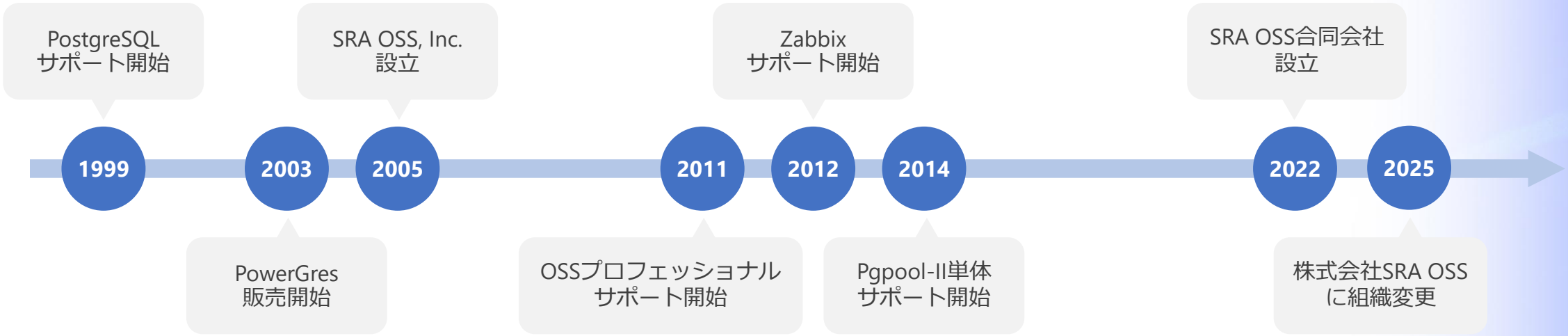
株式会社SRA OSS

所 在 地: 東京都豊島区南池袋2-32-8
設 立 日: 2022年6月17日
株　　主: 株式会社SRA
　　株式会社NTTデータグループ
資 本 金: 7,000万円
社　　長: 稲葉 香理

事業内容

- オープンソースソフトウェア (OSS) 関連のサポート、製品開発・販売、構築・コンサル
- OSSの教育、開発、コミュニティ運営支援
- ソフトウェアの研究開発

顧　　問: 石井 達夫
技術顧問: 増永 良文 (お茶の水女子大学名誉教授)



自己紹介

- 金（キム）　範起（バムキ）
- 株式会社SRA OSS　OSS 事業本部
  技術部 基盤技術グループ所属
- Zabbix の技術サポート、作業など担当
- 趣味：旅行、ウクレレ、アコギ

監視について

監視の必要性

- 本来稼働しているはずのサービスに障害が発生した時

監視をしていない場合

- どのサーバ・サービスが原因になっているか特定する時間がかかる
- 障害が発生していることに気づかない可能性も

結果として顧客にまで影響が及ぶことも

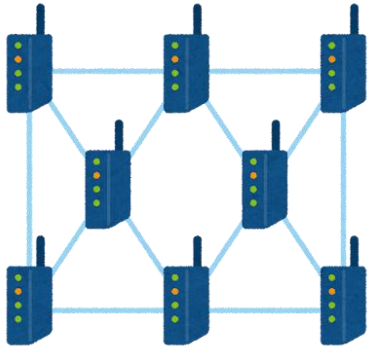
監視をしている場合

- 障害発生の原因特定が容易
- 障害の発生をリアルタイムで確認できる

障害発生時の迅速な対応が可能かつ
日頃の監視によって未然に防げることも

なにを監視する？

サーバリソース

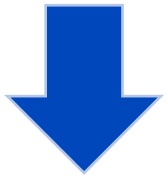


ネットワーク

アプリケーション

Web ページ

ログ



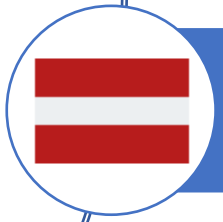
ZABBIX

Zabbixの概要

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

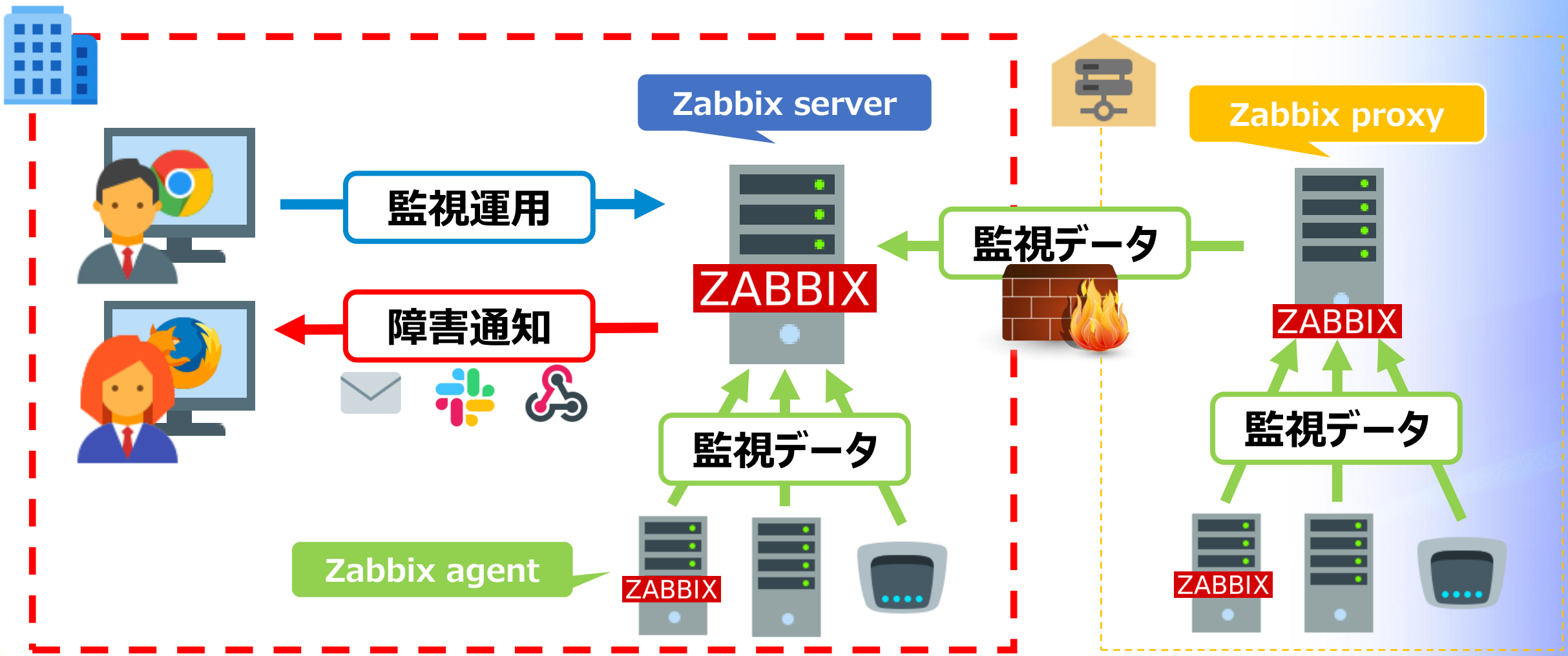
オープンソースソフトウェアとして開発・公開されており、無償で利用可能



開発元はラトビア共和国の Zabbix LLC

Zabbix 1.0 は 2004/3/23 リリースされ、7.4は2025/07/01　リリース

アーキテクチャ



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

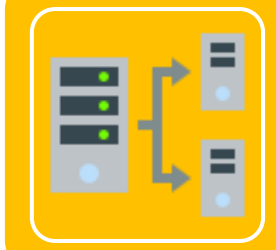
- Java アプリケーション監視用デーモン
- JMX を利用

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

対応OS,データベース

プラットフォーム	Zabbix server	Zabbix agent
Linux	✓	✓
IBM AIX	✓	✓
FreeBSD	✓	✓
NetBSD	✓	✓
OpenBSD	✓	✓
HP-UX	✓	✓
Mac OS X	✓	✓
Solaris	✓	✓
Windows	✗	✓

データベース	Zabbix server	Zabbix proxy
MySQL/Percona	✓	✓
MariaDB	✓	✓
Oracle(非推奨化)	✓	✓
PostgreSQL	✓	✓
TimescaleDB	✓	✓
SQLite	✗	✓

インストール要件

規模	監視メトリック※1	CPU	メモリ
小規模	1,000	2 コア	8GB
中規模	10,000	4 コア	16GB
大規模	100,000	16 コア	64GB
超大規模	1,000,000	32 コア	96GB

※1 (1 メトリック = 1 アイテム + 1 トリガー + 1 グラフです。)

また、Zabbix（7.4） の動作には以下のソフトウェアが必要になります。
（以下のソフトウェア構成は一例となります。）

RDBMS

PostgreSQL

13.0-17.X

WEBサーバー
（Apache＋PHP）

APACHE

2.4 or later

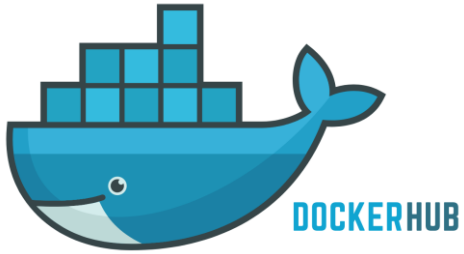
php

8.0.0 - 8.4.X

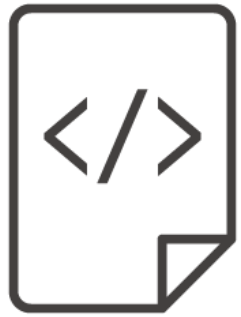
様々なZabbixのインストール方法

rpm/debパッケージ

クラウドイメージ



Dockerコンテナ
(docker hub)



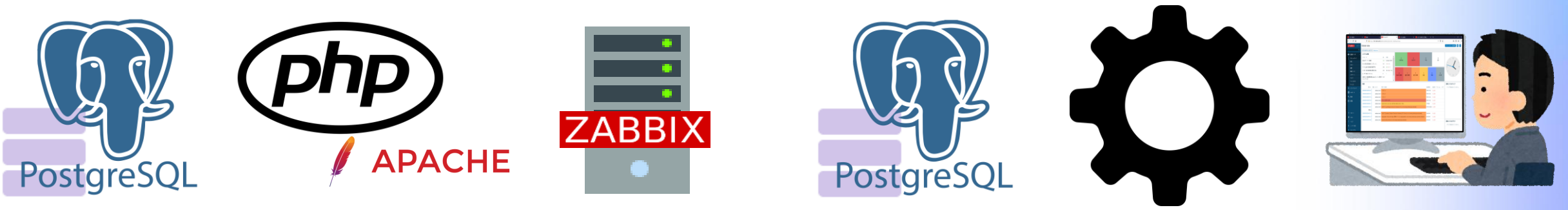
ソースファイル
（要コンパイル）

仮想アプライアンス

Zabbix構築の流れ

Zabbix構築の流れ

（以下のソフトウェア構成は一例となります。）



RDBMSインストール

PHP,APACHE
インストール

Zabbixインストール

RDBMS設定

Zabbixとその他設定

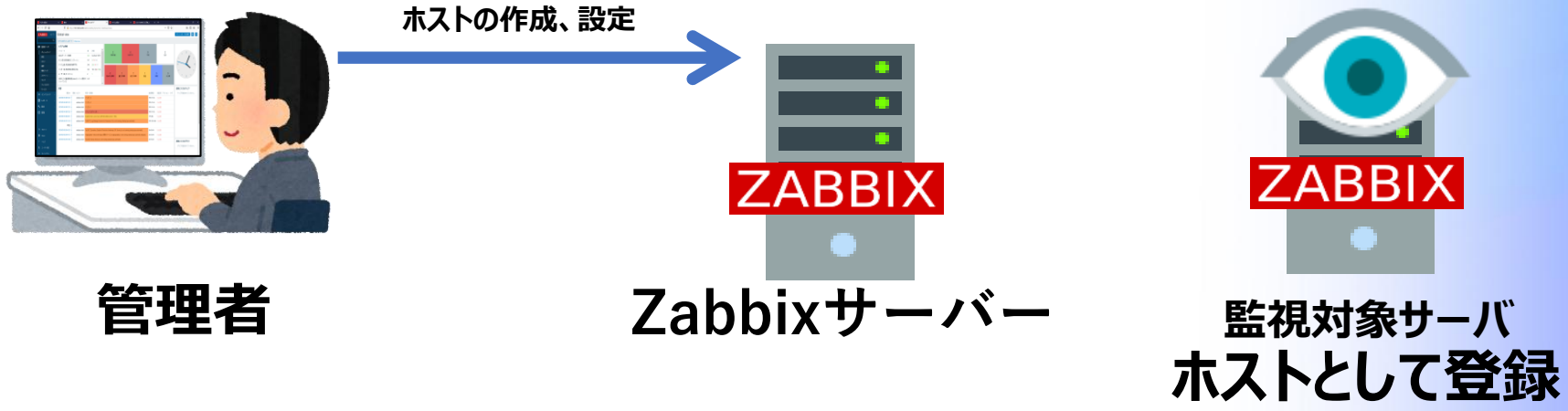
Web フロントエンドの
セットアップ

詳しいZabbixインストール手順や監視設定手順については
弊社のTech Blogを参照してください。
（URLは最終ページにあります。）

インストールされたZabbix確認

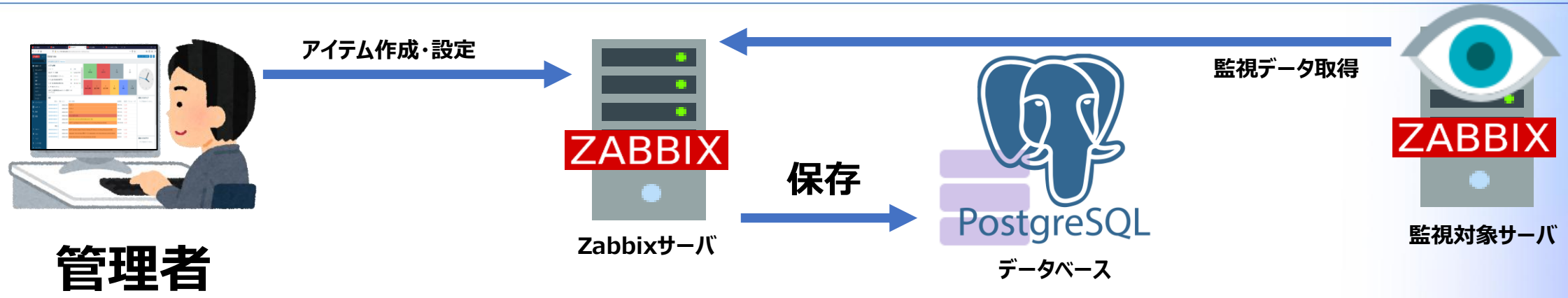
基本的な監視設定

ホストについて



- 「ホスト」とは、**Zabbixの監視対象**となる物理(仮想)ホストのこと
  (何らかの監視用に疑似的に作成もできる)
- 監視を行いたい対象（サーバやルーターなど）をホストとして登録・設定
  - 監視対象からどんなデータを収集するか（アイテム）
  - どのような状態になったら**ユーザーに通知**するか（トリガー）

アイテムについて



Zabbixは設定したアイテムを各ホストから収集しデータベースに保存
このデータを用いて、

- グラフの作成、表示
- 条件式（トリガー）と比較し、**ユーザーに通知**

監視項目（アイテム）

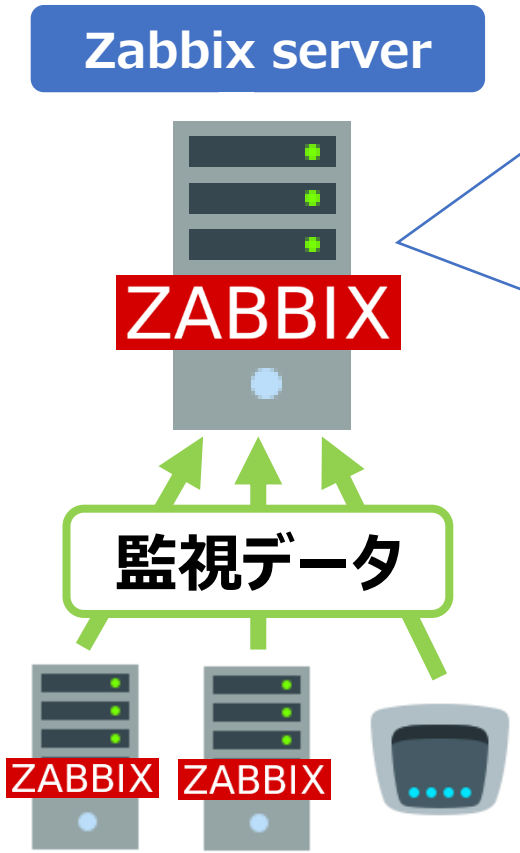
エージェント監視

- **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**

監視項目(アイテム)例



- サーバA
  - Ping 監視　⇒ 1 分間隔で疎通確認
  - CPU 監視　⇒ 5 分間隔で CPU 使用率確認
  - メモリ監視　⇒ 5 分間隔でメモリ使用率確認
  - ディスク監視　⇒ 60 分間隔で / と /var の使用率確認
- サーバB
  - ポート監視　⇒ 5 分間隔で 80 番への TCP 接続確認
  - プロセス監視　⇒ 5 分間隔で httpd プロセスの数を確認
  - ログ監視　⇒ httpd のエラーログを確認
- NW機器
  - SNMP 監視
    - トラフィック　⇒ 10 分間隔で IF の送受信 byte 確認
    - SNMP トラップ　⇒ すべてのトラップを確認

アイテムの新規作成

設定項目	説明
名前	アイテムの名前
キー	アイテムのタイプ（キー） システムからデータを 取得する為の様々な方法
ホストインターフェース	ホストで設定した、 どのインターフェースを 利用し監視を行うかを選択
監視間隔	監視間隔を設定
タイムアウト	アイテムデータ ポーリングリクエストのタイム アウト
ヒストリの保存期間	生データを保存する期間
トレンドの保存期間	グラフ用のサマリデータを保存 する期間

アイテムキー例

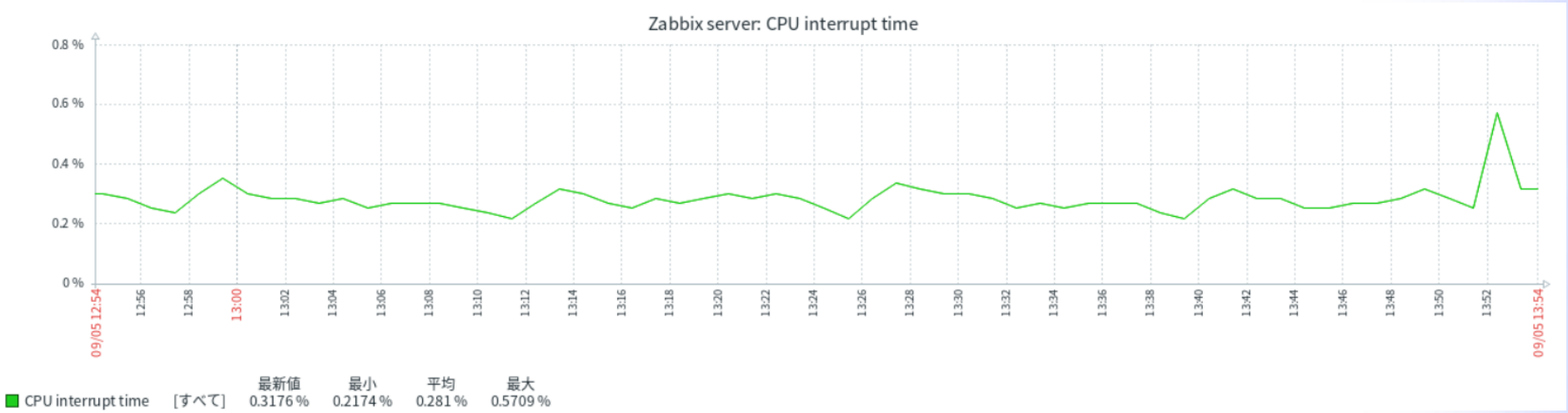
監視項目	キー	戻り値	説明
CPU 使用率	system.cpu.util[<cpu>,<type>,<mode>] (例) system.cpu.util[,user,avg5]	使用率 (%)	指定したCPUの特定の使用タイプ（例: ユーザーモードやカーネルモード）の使用率を監視します。
	system.cpu.load[<cpu>,<mode>] (例) system.cpu.load[,avg5]	浮動小数	CPUの負荷平均を取得します。
メモリ使用率	vm.memory.size[<mode>] (例) vm.memory.size[free]	バイト数 / %	メモリの使用状況を監視します。
ディスク使用率	vfs.fs.size[fs,<mode>] (例) vfs.fs.size[/,pfree]	バイト数 / %	ファイルシステムの空き容量を監視します。
ネットワーク使用率	net.if.in[if,<mode>] (例) net.if.in[eth0,bytes]	整数	指定したインターフェースの受信トラフィックを監視します。
	net.if.out[if,<mode>] (例) net.if.out[eth0,bytes]		指定したインターフェースの送信トラフィックを監視します。
プロセス起動数	proc.num[<name>,<user>,<state>,<cmdline>] (例) proc.num[httpd,apache]	プロセス数	特定のプロセスの起動数を監視します。
ポート	net.tcp.port[<ip>,port] (例) net.tcp.port[,80]	0(down) / 1(up)	指定したポートが開いているか（リッスンしているか）を監視します。
	net.tcp.service[service,<ip>,<port>] (例) net.tcp.service[http]		指定したサービスが正常に稼働しているかを監視します。
死活監視	icmpping[<target>,<packets>,<interval>,<size>,<timeout>] (例) icmpping	0 - ICMP ping 失敗 1 - ICMP ping 成功	サーバーやネットワークデバイスの死活監視を行います。
ログ監視	log[file,<regexp>,<encoding>,<maxlines>,<mode>,<output>,<maxdelay>,<options>,<persistent_dir>] (例) log[/var/log/messages]	ログ	指定したログファイルの内容を監視し、 正規表現で特定のパターンに一致するログを取得します。

アイテムの確認

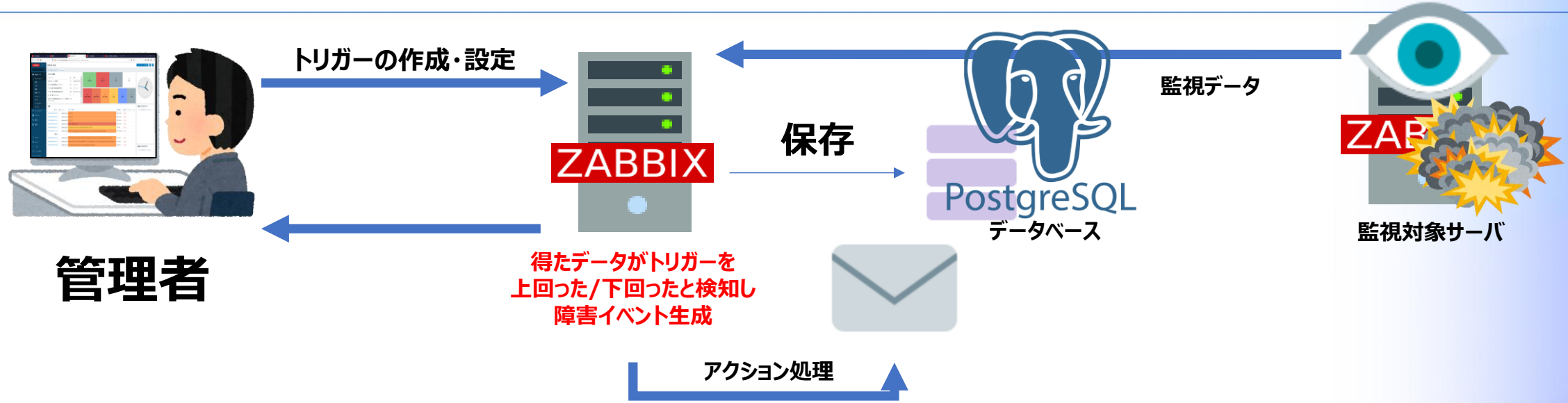
最新データ

ホスト	名前 ▲	最新のチェック時刻	最新の値
Zabbix server	Available memory	14s	538.62 MB
Zabbix server	Available memory in %	13s	30.5449 %
Zabbix server	Checksum of /etc/passwd	54m 15s	1057a763aea1d9b...
Zabbix server	Configuration cache, % used	24s	24.561 %
Zabbix server	Context switches per second	56s	873.2119

グラフ



トリガーについて

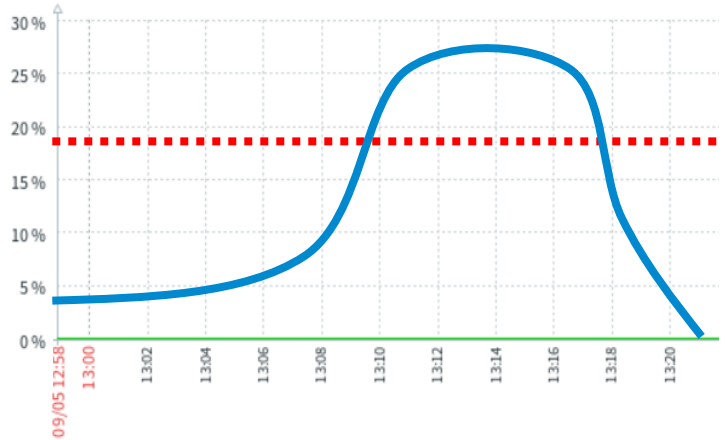


Zabbixは監視対象から受け取ったアイテム項目に対して、
それが異常であるかどうかを判定する（障害を検知する）

- １つのトリガーに複数のアイテムを設定したり、逆に1つのアイテムに対して複数のトリガー設定する。
- トリガーはホスト、アイテム、関数、演算子、数値を用いた条件式で設定されこれを満たした場合「イベント」というものを生成する。
- 例えば、障害の条件式が満たされた場合は「障害イベント」が発生し、発生している障害として表示したり、ユーザーに通知する。
- 復旧の条件式を別設定することも可能この場合は「障害の条件式を満たさない」かつ「復旧の条件式を満たす」時に「復旧イベント」が発生し、復旧したことを通知する。

障害検知（トリガー）

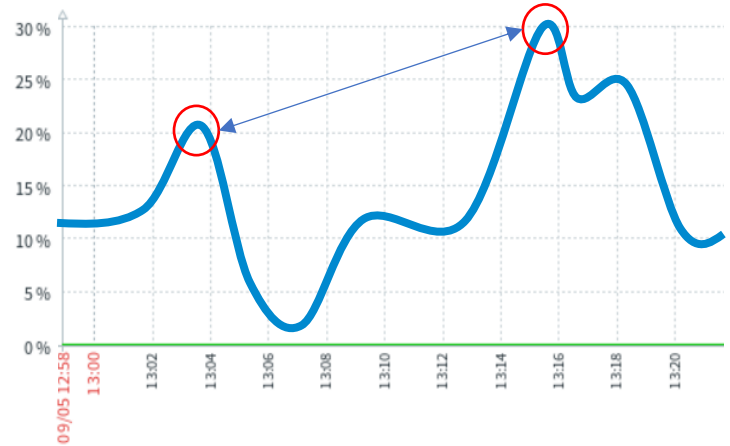
閾値



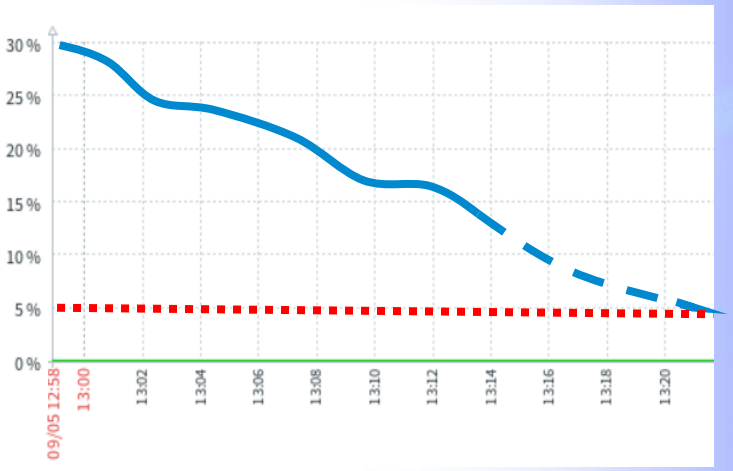
文字列

2025 09/05 14:13:25 [INFO] …
2025 09/05 14:15:10 [ERR] …
2025 09/05 14:21:47 [INFO] …

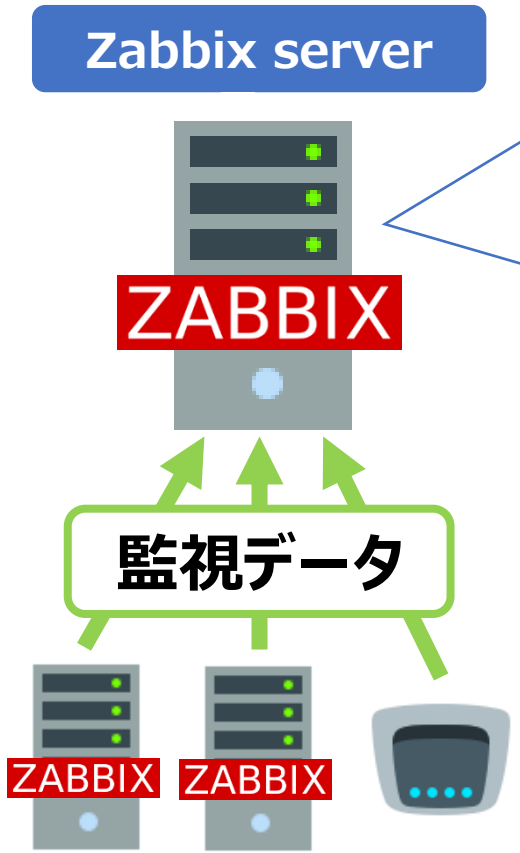
過去との比較



予測値



障害検知例



- サーバA
  - Ping 監視　⇒ 5 分間PING値がない場合障害
  - CPU 監視　⇒ 使用率 90% 以上で障害
  - メモリ監視　⇒ 10 分間の平均使用率が 80% 以上で障害
  - ディスク監視　⇒ 使用率 85% 以上が 3 回続いたら障害
- サーバB
  - ポート監視　⇒ 80 ポートに接続できなければ障害
  - プロセス監視　⇒ httpd プロセス数が 0 なら障害
  - ログ監視　⇒「too many connections」が出たら障害
- NW機器
  - SNMP 監視
    - トラフィック　⇒ 障害検知設定なし
    - SNMP トラップ　⇒「Link Down」を受信したら障害

トリガーの新規作成

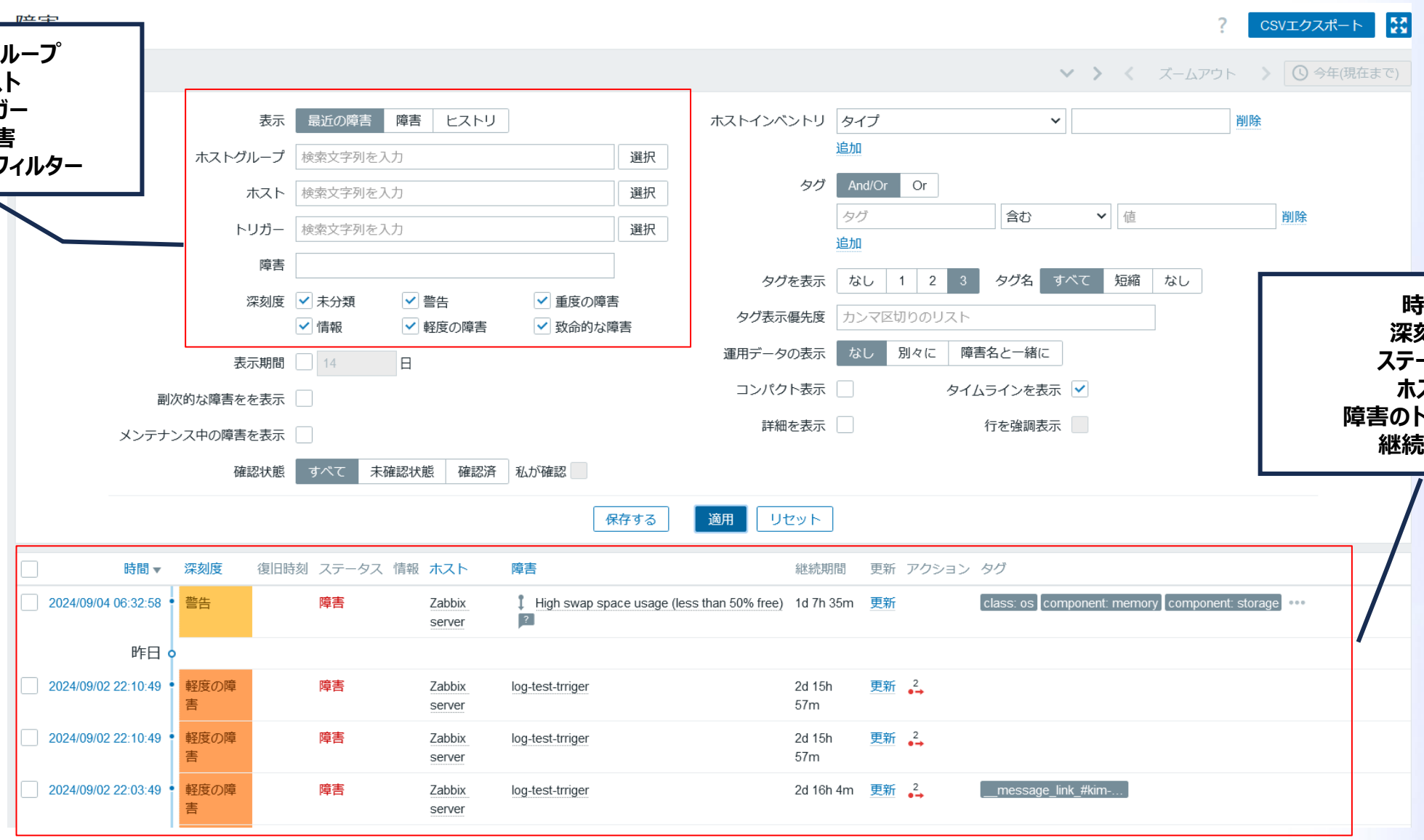
設定項目	説明
名前	トリガーの名前 マルチバイト文字も対応
深刻度	深刻度を6段階から選択 障害発生時には深刻度に応じて 色分けされて表示
条件式	障害となる閾値を論理式で設定
正常イベントの生成	障害が復旧する条件の 設定方法を選択
復旧条件式	復旧する閾値を論理式で設定
手動クローズを許可	障害生成時に Webインターフェースから 障害クローズ

トリガーの条件式例

監視項目	条件式	説明
CPU 使用率	min(/ホスト名/system.cpu.util[,user],5m)>90	関数: min(5m)： 過去5分間における最小値を取得します。 比較値: >90： CPU使用率が90%を超えた場合に障害にします。
メモリ使用率	last(/ホスト名/vm.memory.size[available])<500M	関数: last()：利用可能なメモリの最新値を取得します。 比較値: <500M：利用可能メモリが500MB未満になると障害にします
ディスク使用率	last(/ホスト名/vfs.fs.size[/,pfree])<10	関数: last()：ディスクの空き容量率の最新の値を取得します。 比較値: <10：空き容量率が10%未満になった場合に障害にします
ネットワーク使用率	min(/ホスト名/net.if.in[eth0,bytes],5m)>10000000	関数: min(5m)：過去5分間における最小の受信帯域を取得します。 比較値: >10000000：10MB/sを超える場合に障害にします
プロセス起動数	last(/ホスト名/proc.num[http,,running])=0	関数: last()：指定したプロセスの実行中のプロセス数の最新値を取得します。 比較値: =0：プロセス数が0の場合に障害にします
ポート	last(/ホスト名/net.tcp.port[127.0.0.1,80])=0	関数: last()：指定したIPアドレスとポートの応答状況の最新値を取得します。 比較値: =0：ポートが応答しない場合に障害にします
死活監視	last(/ホスト名/icmpping)=0	関数: last()：指定したサーバーに対するPingの応答状況の最新値を取得します。 比較値: =0：Pingが失敗した場合に障害にします
ログ監視	find(/ホスト名 /log[/var/log/messages,error],,"regexp","error")=1	関数: find()：指定したログファイルから、パータンに一致するログを検索します。 比較値: =1：一致するエントリが見つかった場合に障害にします

障害検知の確認

ホストグループ
ホスト
トリガー
障害
深刻度のフィルター



時間
深刻度
ステータス
ホスト
障害のトリガー名
継続時間

まとめ

監視について

ZABBIX

Zabbixとは

Zabbixのコンポーネント

Zabbixのホスト

Zabbixのアイテム

Zabbixのトリガー

ご清聴頂きありがとうございました！

▼技術情報配信中！

YouTube

https://www.youtube.com/c/sraoss-official

Tech Blog

Zabbix 7.0 インストール
https://wordpress.sraoss.jp/tech-blog/zabbix/zabbix70-install/
第 1 回 Zabbix を動かしてみよう
https://www.sraoss.co.jp/tech-blog/zabbix/zabbix-introduction-01/
第 2 回 Zabbix のさまざまな監視機能を試してみよう
https://www.sraoss.co.jp/tech-blog/zabbix/zabbix-introduction-02/
第 3 回 Zabbix API 入門
https://www.sraoss.co.jp/tech-blog/zabbix/zabbix-introduction-03/

SRA OSS

– RECRUIT –

【 Zabbix エンジニア】
【 インフラ系 OSS エンジニア 】

OSSのように、オープンで情熱的であれ。

全世界のエンジニア・OSSコミュニティの揺るぎない信念とたゆまぬ努力に敬意を。
私たちも情熱を持ち、会社の枠を超えてオープンであり続けます。

Our Value

- ❑ OSS貢献
- ❑ カスタマーファースト
- ❑ マルチジョブ/マルチキャリア

株式会社SRA OSS 人事担当
✉ personnel@sraoss.co.jp

業務内容、待遇、カジュアル面談などお気軽にお問合せください。

SRA OSS