正規表現の罠を回避せよ！ Zabbix ログ監視における 実践的フィルタリング術

株式会社SRA OSS
金　範起

会社紹介

株式会社SRA OSS

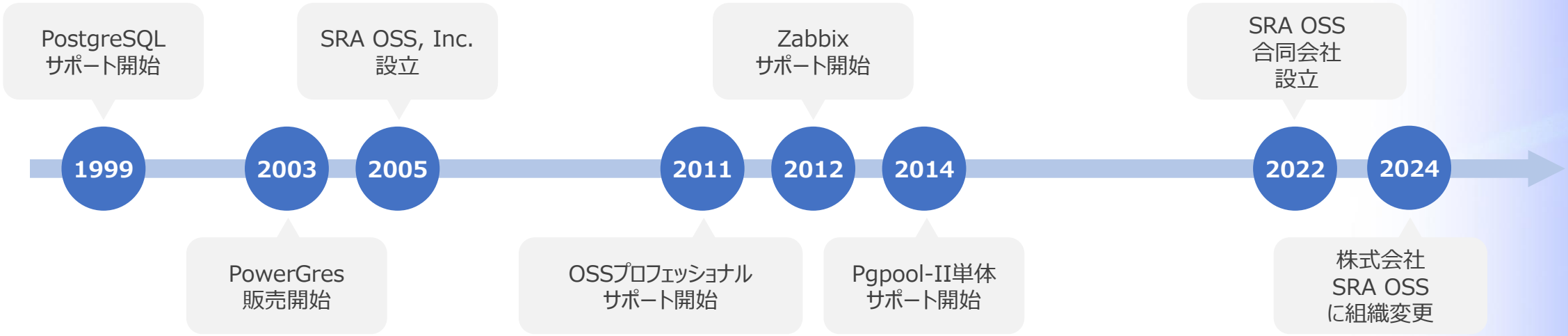
所 在 地: 東京都豊島区南池袋2-32-8
設 立 日: 2022年6月17日
株　　主: 株式会社SRA
　　株式会社NTTデータ
資 本 金: 7,000万円
社　　長: 稲葉 香理

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

アジェンダ

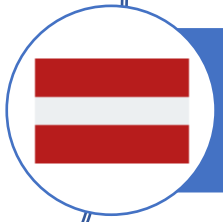
- Zabbix の概要
- 基本的な監視設定
- ログ監視の基礎
- ログ監視の TIPS
- グローバル正規表現
- ログ監視の最適化
- まとめ

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

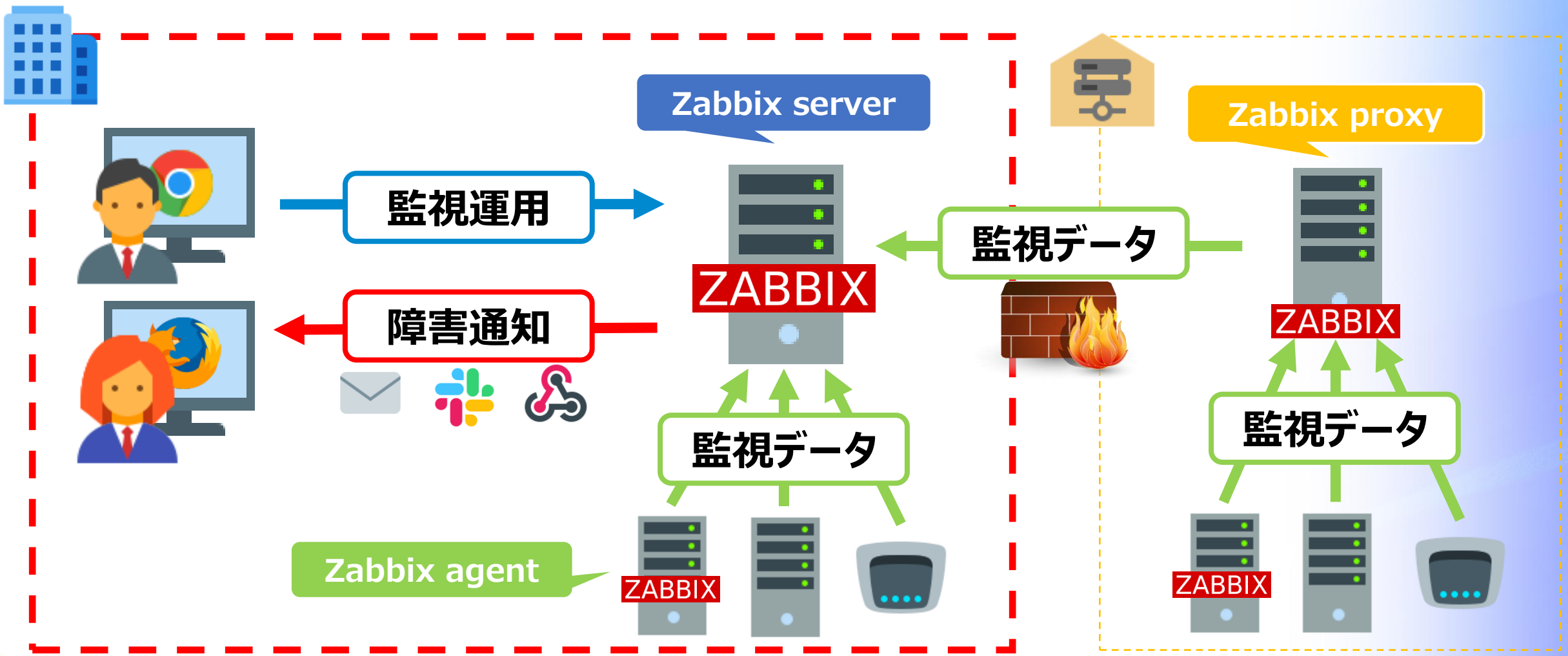
オープンソースソフトウェアとして開発・公開されており、無償で利用可能



開発元はラトビア共和国の Zabbix LLC

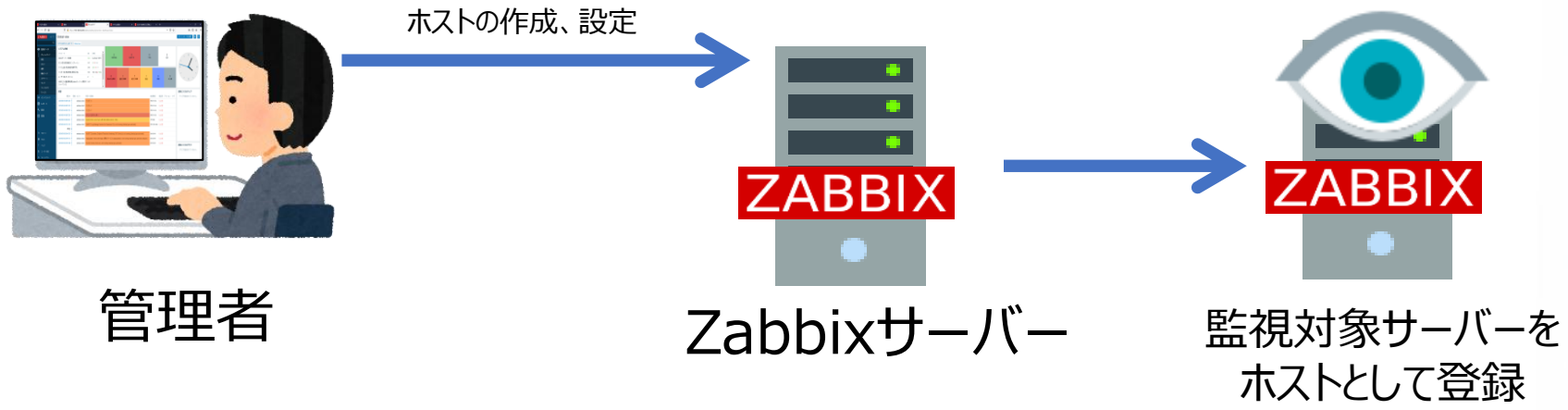
Zabbix 1.0は2004年にリリースされ、7.4は2025年7月にリリース

アーキテクチャ



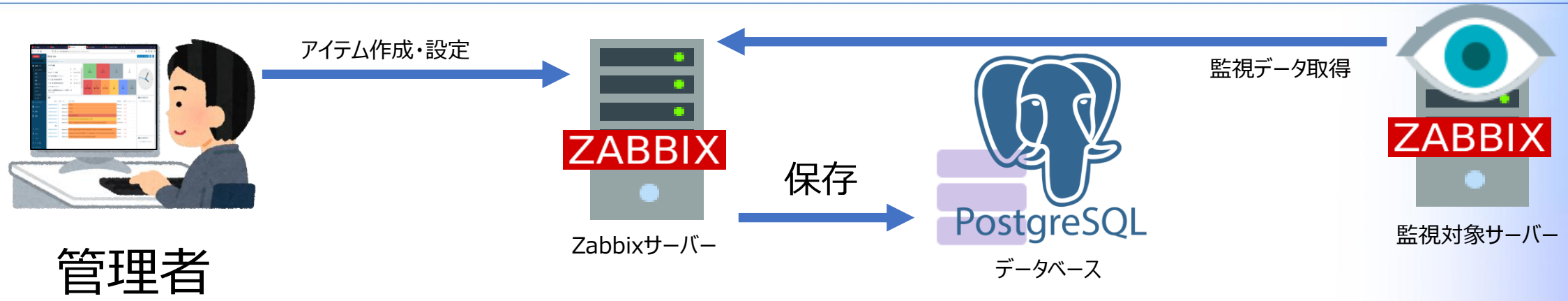
基本的な監視設定

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
- **監視を行いたい対象**（サーバーやルーターなど）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では、アイテムキーを用いて**「何を監視するか」**を指定

  このアイテムキーにより、取得するデータの種類が決定
- 設定されたアイテムのデータは**各ホストから収集され、データベースに保存**
- 保存されたデータをもとに、以下のような機能が提供
  - **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）

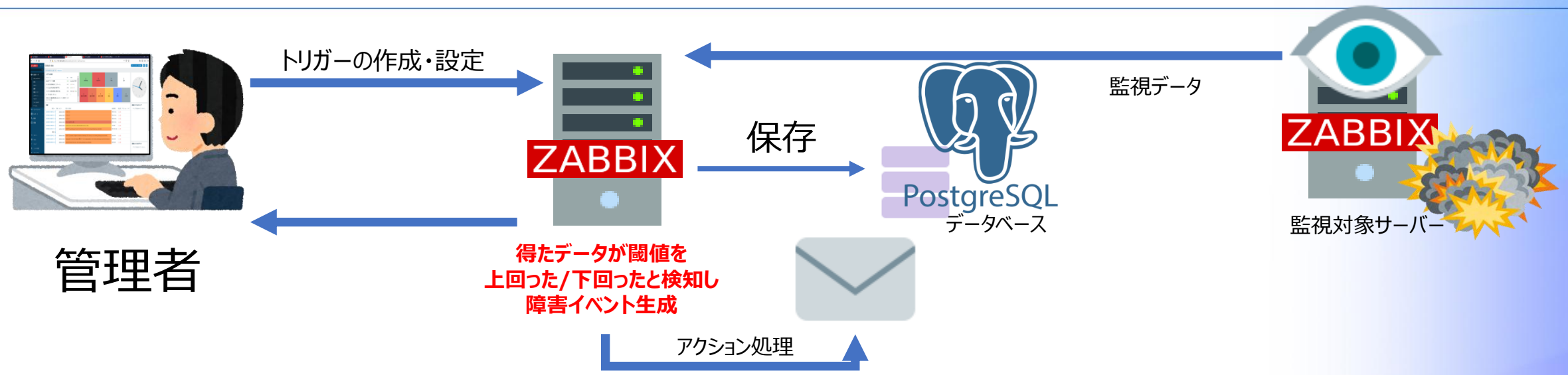
エージェント監視

- 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
- Web 監視
- プロセス/サービス監視
- **ログ監視**

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
- クラウド 監視

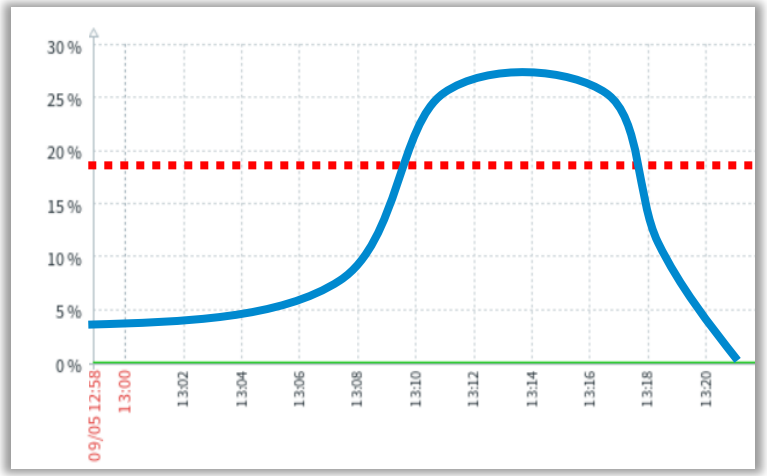
トリガーについて



- 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（障害を検知）
- 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、
  その条件を満たした場合に「イベント」が生成
- このイベントには、問題の重要度を表す**「深刻度（Severity）」**を設定できる
- イベントを条件に応じて「障害」だけでなく「復旧」も検知し、ユーザーへ通知することが可能

障害検知（トリガー）

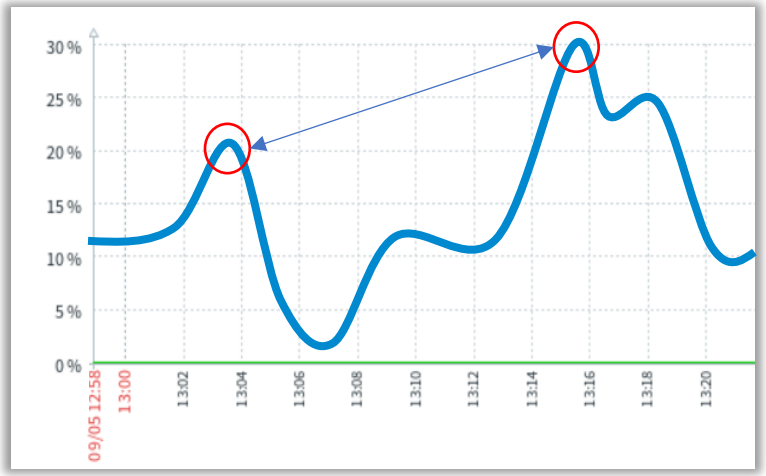
閾値



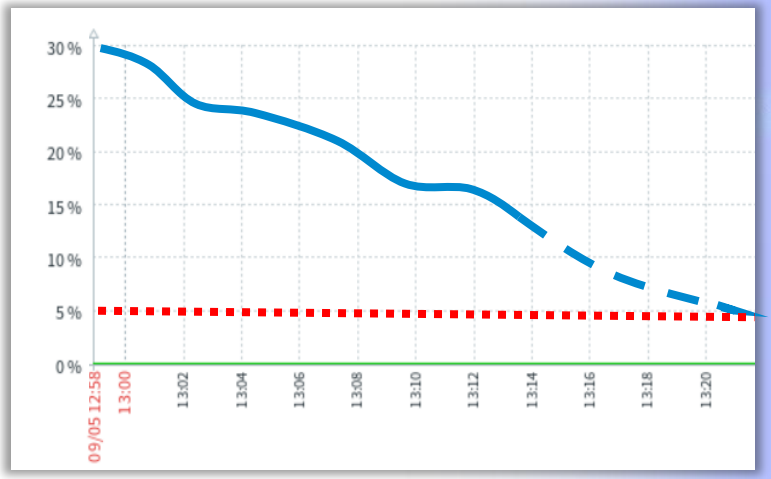
文字列

2026 09/05 14:13:25 [INFO] …
2026 09/05 14:15:10 [ERR] …
2026 09/05 14:21:47 [INFO] …

過去との比較

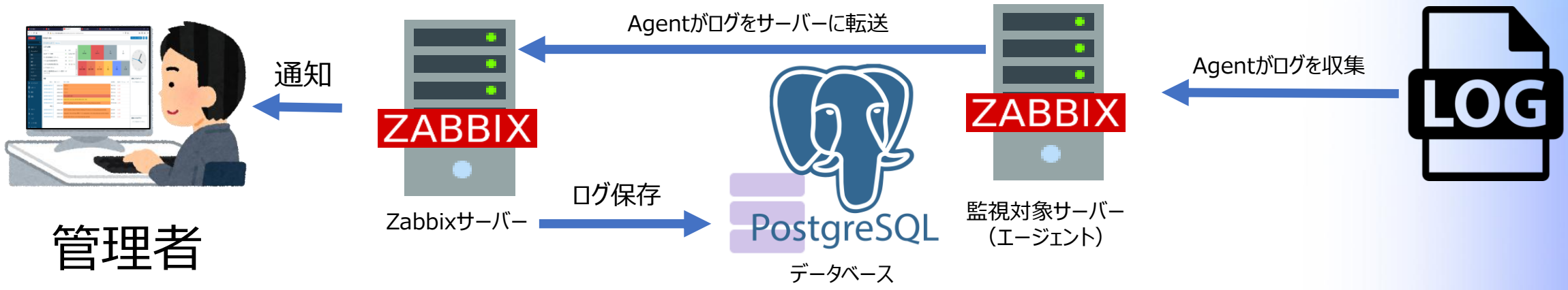


予測値



ログ監視の基礎

ログ監視について（1）



- Zabbixでは、**log や logrt キーを使用してログファイルを監視できる**
  (ログローテーションが設定されているログも監視可能)
- Error や Alert など特定の文字列を検知した際にアラートを出すことができる
  (ログの内容を**正規表現でフィルター処理**することも可能)
  - 例えば、/var/log/messages に ERROR が出力された場合に通知を送る
    例: logrt["/var/log/^messages(|-¥d{8})$", "(ERROR|FATAL|ALERT)“]

ログ監視について（2）

■代表的なログ監視アイテムキー

用途に合わせて適切なキーを選択（固定ログ、ローテーション、イベントログ等）

アイテムキー	対象	例
log[]	1つの固定ログファイル	log[/var/log/messages]
logrt[]	ローテーションされるログファイル	logrt[“/var/log/^messages(|-¥d{8})$”]
log.count[]	ログファイル内のマッチ行数カウント	log.count[/var/log/app.log,ERROR]
logrt.cont[]	ローテーションされる ログファイル内のマッチ行数カウント	logrt.count[/var/log/app.log,ERROR]
eventlog[]	Windowsイベントログの監視	eventlog[System]
eventlog.count[]	Windowsイベントログのカウント	eventlog.count[System,,,Error]

正規表現の基礎講座

- **正規表現とは、文字列のパターンを表現**
- **検索や一致判定を行うための仕組み**
- **条件に合う文字列を柔軟かつ簡潔に指定可能**

■基本メタ文字リファレンス

任意の1文字と一致

.	任意の1文字に一致	**¥d**	任意の数字に一致
[]	いずれかの1文字に一致	**[^]**	括弧内以外文字に一致

N回繰り返し

＊	0回以上の繰り返し	**？**	0回または1回の出現
＋	1回以上の繰り返し		

文字列の先頭/末尾に一致

^	文字列の先頭に一致	**$**	文字列の末尾に一致

文字例の先読み

(?=…) 先読み?=次に指定したパターンが後続する場合に一致
(?!…) 先読み?!次に指定したパターンが後続しない場合に一致

文字エスケープ

¥ メタ文字のエスケープ

日付付きログファイルへの一致

```
^messages(|-¥d{8})$
```

✔messages
✔messages-20230829

ドット＋数字のログファイルへの一致

```
^messages(¥.¥d+)?$
```

✔messages
✔messages.1, messages.123

文字列を含むか含まないか

```
^(?=.*ERROR)(?!.*(PostgreSQL|TempFile)).*$
```

✔ERROR
✔ERROR: Something went wrong

💡 **Tips:** Zabbixのログ監視では、
PCRE、PCRE2（Perl互換正規表現）が使用される
特にファイル名の指定には ^ と $ を組み合わせて
範囲を限定するのがコツ

Logrtアイテムキーについて

■Logrtのパラメータ

logrt[file regexp,<regexp>,<encoding>,<maxlines>,<mode>,<output>,<maxdelay>,<options>,<persistent dir>]

引数名	用途
file_regexp	監視対象ファイル名の特定
regexp	抽出したいログ本文パターン

■設定の記載例

例： app.log や、ローテーションされたファイルを対象に、ERROR または FATAL を検知

```
logrt["/var/log/^app¥.log(¥.[0-9]+)?$","ERROR|FATAL"]
```

運用の注意点

- **シェル（Linux）のワイルドカード形式ではない**点に注意
- logrtはログローテーションされるファイルの中で、実際に更新されているログファイルだけを読み取る

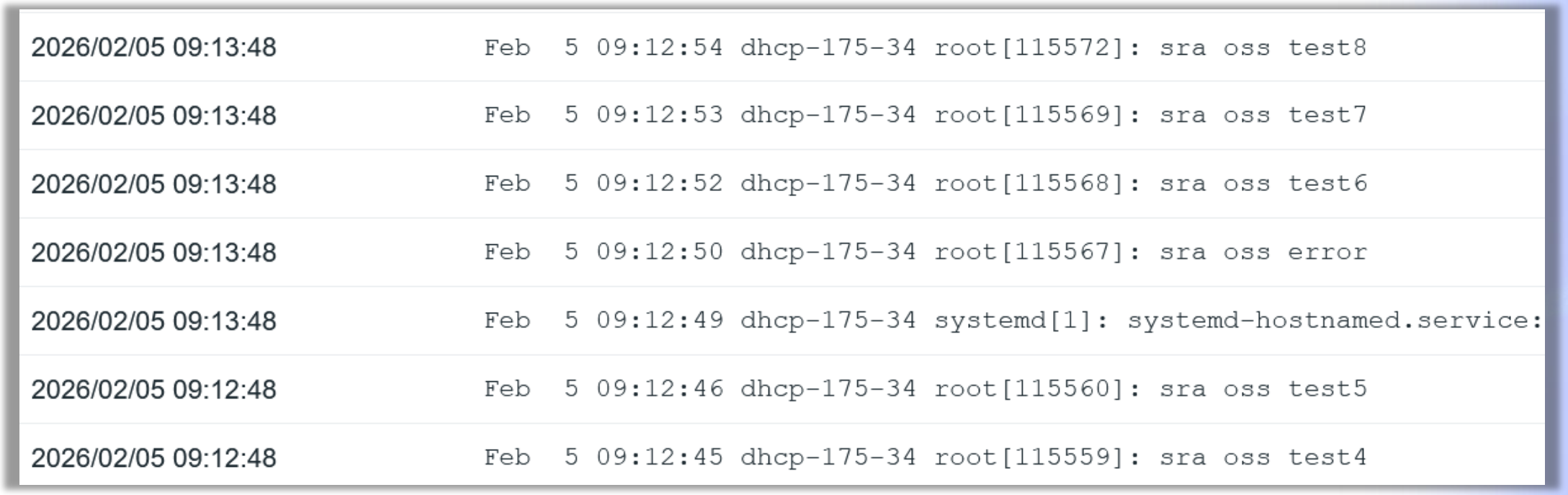
アイテムの新規作成

ログローテーションされる サーバーの messages ログを 監視するアイテム	
名前	log-test
タイプ	Zabbixエージェント（アクティブ）
キー	**logrt["/var/log/^messages(|-¥d{8})$"]**
データ型	ログ
監視間隔	1m

アイテムで取得したヒストリの確認

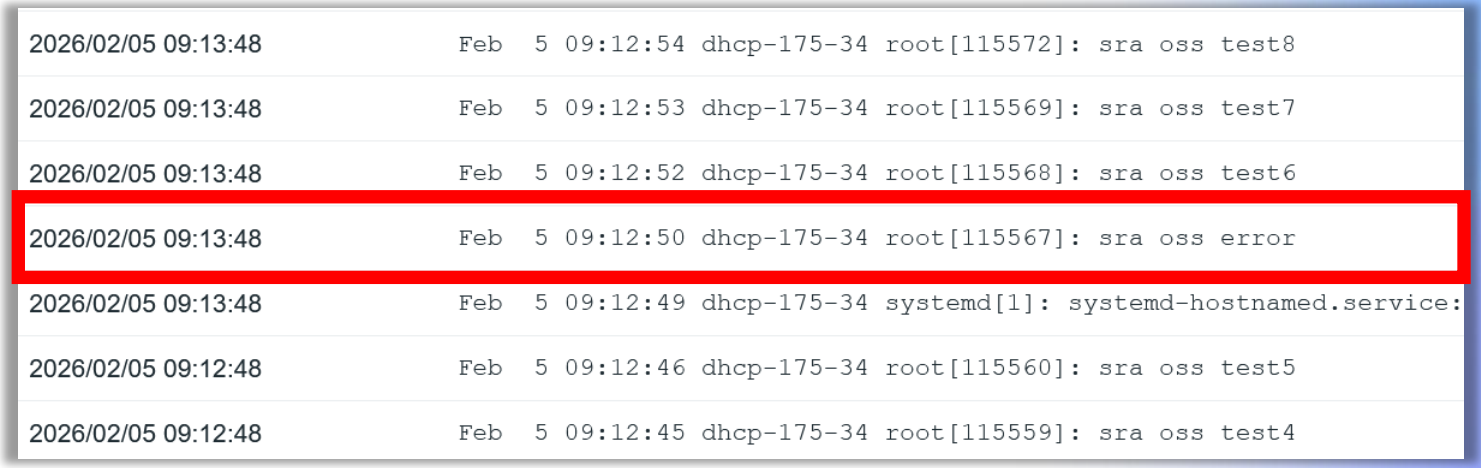
「監視データ」→「最新データ」→「ヒストリ」

名前 ▲	最新のチェック時刻	最新の値	変化	タグ	情報
log-test	43s	Feb 15 13:18:25 d...			ヒストリ



2026/02/05 09:13:48	Feb 5 09:12:54 dhcp-175-34 root[115572]: sra oss test8
2026/02/05 09:13:48	Feb 5 09:12:53 dhcp-175-34 root[115569]: sra oss test7
2026/02/05 09:13:48	Feb 5 09:12:52 dhcp-175-34 root[115568]: sra oss test6
2026/02/05 09:13:48	Feb 5 09:12:50 dhcp-175-34 root[115567]: sra oss error
2026/02/05 09:13:48	Feb 5 09:12:49 dhcp-175-34 systemd[1]: systemd-hostnamed.service:
2026/02/05 09:12:48	Feb 5 09:12:46 dhcp-175-34 root[115560]: sra oss test5
2026/02/05 09:12:48	Feb 5 09:12:45 dhcp-175-34 root[115559]: sra oss test4

トリガーの新規作成



実運用では、負荷軽減のため、
アイテムキー側で必要なログのみをフィルタリングする設計を推奨

サーバーの messages ログにerror を検知した場合発火するトリガー	
名前	log-error-trigger
条件式	**find(/Zabbix server/logrt["/var/log/^messages(|-¥d{8})$"] ,,"regexp","error")=1**
正常イベントの生成	条件式
障害イベント生成モード	複数

障害画面の確認

ログ監視のTIPS

ログ監視のTIPS（1-1）【権限とユーザー】

■監視の主体は「Zabbixエージェント実行ユーザー」である

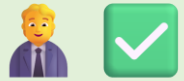
管理者(root)権限でログが参照できても、監視は成功しない

実際にログを読み取る**Zabbixエージェント（zabbixエージェント実行ユーザー）**に

権限があるかどうかが、ログ監視の最重要ポイント

ROOTから参照できてもNG

「rootで見えてるからヨシ！」は失敗の元



Zabbixエージェント実行ユーザーでの参照が必須

エージェント実行ユーザーに読み取り権限が必要

💡 **ミスを防ぐための「事前の検証」**

```
# sudo -u zabbix head -2 /path/to/example.log
```

※ ここで "Permission denied" が出る場合は設定修正が必要

ログ監視のTIPS（1-2）【権限とユーザー】

■権限不足の対策：推奨される方法と注意点

対策①：OS側で権限を付与 （非推奨）

```
# chmod 644 /path/to/example.log
```

対策②：Rootユーザーでのエージェント実行（非推奨）

```
# zabbix_agentd.conf

AllowRoot=1

# systemctl edit Zabbix-agent

[Service]
User=root
Group=root
```

※ 上記のAllowRoot設定を行うとAgentの再起動が必要

※ rootユーザーでのエージェントの実行はセキュリティ上非推奨

対策③：ACL で Zabbixユーザーに権限寄与（推奨）

```
# setfacl -m u:zabbix:r /path/to/example.log
```

※重要：ログローテーション対策

ログローテーション時に権限がデフォルトへ戻らないよう

logrotateの設定にも注意が必要

必ず logrotate の postrotate 等で、

再度 setfacl を実行できる設定が必要

ログ監視のTIPS（2）【注意点】

■ログ監視中やるべきこと

- **重複検知のない安定した監視体制の構築**
- **原因特定（デバッグ）はログ確認で実現**
- **最後に長さに注意！**

正しいファイル操作

エージェントは**inode番号**でファイルを識別
ファイル名が同じでも置き換え操作を行うと
再解析されLogを再読み込みする。

NGな操作

touchでの更新日時変更、
コピー上書き

重複アラートが発生！

エージェントログの活用

アイテムが**「取得不可」**になった場合、
その原因はZabbixエージェントの
ログファイルに記録される。

トラブル解決の鍵

- **DebugLevel=3でも**
  重要なエラーは出力される。
- **Agentログ自体の**
  監視設定も可能。

長さに注意！

1行のログが **256KBを超える場合**
Zabbix agent は **先頭256KBまで！**
そのため、非常に長いJSONログでは、
後半のエラーメッセージは検知不可能

ログ監視のTIPS（3）【ファイル名の指定】

■ローテーションへの対応と負荷のトレードオフ

logrt は正規表現でローテーションするファイルを監視できる強力な機能
一方、指定が曖昧だと予期せぬ負荷や誤動作を引き起こす

監視ファイル数と負荷の関係

正規表現が広範囲すぎると、エージェントがチェック
すべき対象が膨大になり、**CPU負荷が発生**

同一ディレクトリ内のファイル数が多い場合、
正規表現でマッチするファイルを制限する。

不要なファイルへの参照

Logrtの第1引数で、曖昧な指定（.* など）は、
監視に不要な一時ファイルやアーカイブファイルを拾ってしまい
不要なログを読み込み誤検知する可能性がある

NG: app¥.log.* では、想定しないログの参照
/var/log/app.log.swp
/var/log/app.log.gz
OK: ^app¥.log(¥.[0-9]+)?$ では、想定通りの参照
/var/log/app.log.2026

ログ監視のTIPS（4）【ドットの誤用・エスケープ漏れ】

問題

- ログの再読み込みが発生
- 文字化けしたログが読み込まれた

原因

正規表現において「.」は任意の1文字を意味するため、
エスケープ漏れに注意が必要
ドットをエスケープしない場合、想定外の文字列にもマッチ
また、**末尾に .* を指定すると、**
ローテートログ以外の予期せぬ拡張子まで一致

```
logrt["/var/log/app.log.*"]
```

解決方法

「.」を文字として扱う場合は ¥. と記述し先頭(^)と行末($)を明示しながらマッチングの範囲を厳密に制限で、
意図しないバックアップファイルを完全に除外することが可能

```
logrt["/var/log/^app¥.log(|-¥d{8})$"]
```

※日付なし、または日付あり(8桁)のみを許可し、.gz等を除外

【意図したターゲットログ】

/var/log/app.log

/var/log/app.log-20260201

【誤検知となるログの例】

/var/log/app.log-20260118.gz

/var/log/app.log-20260125.gz

/var/log/app.log-20260202.tmp

/var/log/.app.log-20260203.swp

ログ監視のTIPS（5）【複数ファイルを1アイテムで監視】

問題

- **過去のログデータが再読み込みされた**
- **同一アイテムに対して、同じアラートが複数回通知された**

原因

正規表現を用いて**個々の独立した複数のログファイルを1つのアイテムで一括監視**してログがランダムな更新タイミングにより、ファイルの切替、更新時にエージェントが再読込を起こしやすい構成

```
logrt["/var/log/app1¥.log|app2¥.log|app3¥.log"]
```

解決方法

監視対象はファイルごとにアイテムを分離する

個別監視により、各ファイルの読込状態を正確に管理でき、重複検知を根本から防げる

```
logrt["/var/log/app1¥.log"]
logrt["/var/log/app2¥.log"]
logrt["/var/log/app3¥.log"]
```

以下は個々に独立したログファイル

/var/log/app/app1.log

/var/log/app/app2.log

/var/log/app/app3.log

ログ監視のTIPS（6）【メタ文字の誤解】

問題

- **本来検知不要な正常ログに**
  一文字でも「E」「R」「O」などが含まれると、障害として通知された

原因

[] , () , | 等は正規表現のメタ文字で、[ERROR] と書くと
E, R, Oのうちどれか1文字という意味になり、誤検知発生

[ERROR]

解決方法

記号そのものを探したい場合はエスケープメタ文字 **¥** を付ける
これで正確に **[ERROR]** という文字列を特定

¥[ERROR¥]

【意図したターゲットログ】

2026-02-04 [ERROR] Connection refused.

【誤検知となるログの例】

※監視要件に関係なく「パターン一致」だけで誤検知される

SYSTEM OPERATION SUCCESSFUL

[RECOVERY] Service has restarted

USER LOGIN: Admin

ログ監視のTIPS（7）【長いログ行と .*? の使い方】

問題

- 1行が長いログで、マッチングに時間がかかりCPU負荷が発生した

原因

「.*」は**「可能な限り長く」**マッチしようとする特性
これは行の最後までをスキャンし、
パターン全体が成立するかを確認し続けるため、リソース浪費

`ERROR.*`

解決方法

.*? を使用することで、条件を満たした時点で、即座にスキャンを終了
これにより、**CPU負荷を最小限に抑え、高速な監視を実現**

`ERROR.*?`

監視対象ログファイルの文字例

例：長いJSONログ（巨大な1行）

【NGの場合】

```
{"level":"ERROR", "msg":"Failed to connect",
"meta":{"node":"01", "trace":"..."}
```

→行末まですべてスキャンしてしまう (高負荷)

【OKの場合】

```
{"level":"ERROR", "msg":"Failed to connect",
"meta":{"node":"01", "trace":"...",}
```

→ERRORを見つけた時点でスキャン終了

ログ監視のTIPS（8）【regsub による抽出】

問題

- **1障害名が固定だと、詳細（機器やエラー内容）が分からず判別が難しい**

原因

トリガー名が固定されていいるため

例：「SNMP Trapを受信しました」だけでは、
どのインターフェースがLinkDownしたか不明

解決方法

マクロ関数 regsub() を用いた動的命名
regsub("正規表現パターン", "置換文字列")

・regsubで正規表現マッチした箇所(識別情報)のみを抽出してトリガー名に表示

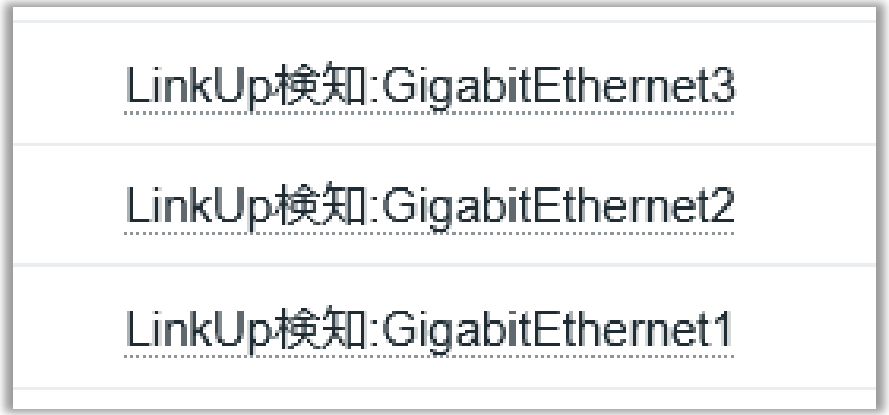
監視対象ログファイルの文字例

the SNMP entity, **GigabitEthernet 1~10** ・・・

{ITEM.VALUE}とregsub関数による抽出

設定例：ログからインターフェース名のみを抽出してトリガー名にする

```
LinkUp検知:{{ITEM.VALUE}.regsub(".*(GigabitEthernet¥[.*¥]).*","¥1")}
```



解説: regsubの仕組み

正規表現 .*(GigabitEthernet¥[.*¥]).* で
行全体にマッチさせ、¥1 (1つ目のカッコ内) の内容に置き換えることで、
長いログから必要な情報だけを切り出す

※regsub 関数内ではグローバル正規表現（@形式）は使用不可

グローバル正規表現

グローバル正規表現（1）

■仕組みとメリット

グローバル正規表現の定義
@SRAOSS → ERROR|FATAL…

ホストA
アイテムキー

ホストB
アイテムキー

ホストC
トリガー条件式

- 「@名前」形式で色んな設定で呼び出し可能
- 1つの設定を複数の監視設定で**使い回す**

■logrtでのグローバル正規表現のサポート範囲

監視アイテムキー logrt[file_regexp, regexp, ...] において

引数名	用途	グローバル正規表現
file_regexp	ファイル名の特定	利用不可 (No)
regexp	ログ本文パターン	利用可能 (Yes)

■設定の記載例

例1：インライン正規表現のみを使用

長くなりがち

```
logrt["/var/log/^app¥.log(¥.[0-9]+)?$","ERROR|FATAL…"]
```

例2：第2引数にグローバル正規表現を使用

スッキリ！

```
logrt["/var/log/^app¥.log(¥.[0-9]+)?$",“@SRAOSS"]
```

グローバル正規表現（2）：仕様と形式

条件式の形式（Expression Types）

形式	説明 / 特徴
文字列が含まれる	指定した部分文字列（テキスト）が含まれる場合に一致
いずれかの文字列が含まれる	カンマ(,)やドット(.)等の区切り文字で分けた文字列のいずれかに一致
文字列が含まれない	指定したテキストが含まれない場合に一致
結果が真 / 結果が偽	正規表現として評価、スラッシュ / をエスケープせずに使用可能

複数式の結合論理

複数の部分式を定義した場合、
Zabbixはこれらを **AND論理** で計算

条件式1 AND 条件式2 AND 条件式3

※ 1つでも結果が False の場合、
全体の結合結果も False と表示される

大文字と小文字の区別

各部分式ごとに**「大文字と小文字を区別」**

チェックボックスがあり、条件式ごとに文字の**区別が**可能

グローバル正規表現（3）：設定

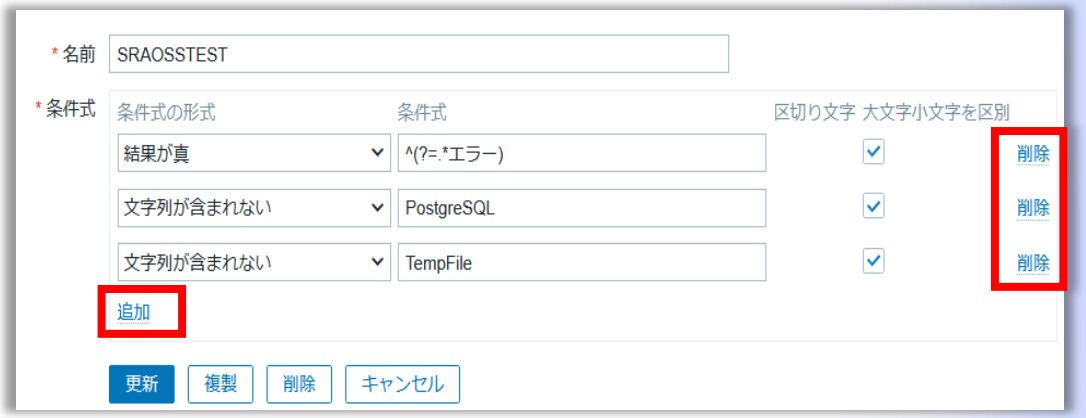
■ 複雑な正規表現をアイテムごとに一行で書くのは「大変」

また、一箇所のミスで監視が破綻するリスクがあり、メンテナンスが困難

```
^(?=.*エラー)(?!.*(PostgreSQL|TempFile)).*$
```

グローバル正規表現で整理、追加

- 条件を独立した「リスト」として管理できる
- **追加ボタン**で簡単に追加できる



TIPS：メンテナンスの劇的な効率化

運用中に、除外したいキーワードが増えても、
アイテム設定を編集する必要はない
グローバル正規表現を「一箇所」修正・追加するだけで、
その設定を参照している全てのアイテムに即時反映される

※条件を変更された場合は、適用前にテストすることが大切

ログ判定の例

[真] Zabbix ERROR エラー発生
[偽] **PostgreSQL**接続エラー（正しく除外）
[偽] **TempFile**のエラーです　（正しく除外）

グローバル正規表現 (4)：運用上のTIPS

テストタブの重要性

- **[テスト] タブでは、作成した条件式**が意図した論理で正しくフィルタリングされるかを**保存前に検証可能**

Zabbixバージョン管理

- Zabbixでは**PCRE／PCRE2が使用可能**だが、**古いagentはPCRE2に非対応**の場合がある
- そのため、Web UIのグローバル正規表現テストで正しく見えても、
  実際のログ監視では期待どおり動作しない可能性があるため注意が必要

255文字制限の回避策

- グローバル正規表現の個別の**「条件式」には255文字の制限**がある
- 255文字を超える条件式を扱うには**複数の条件式に分けて登録**

ログ監視の最適化

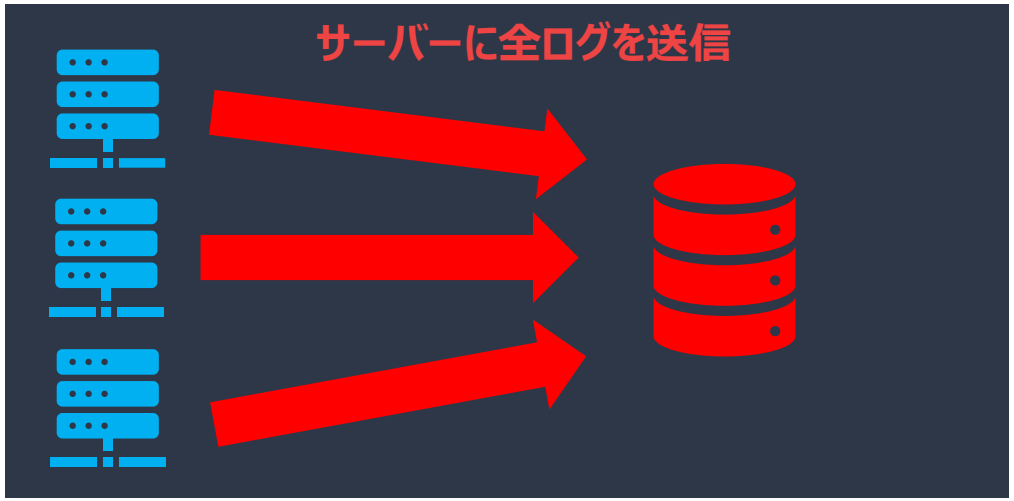
■ Zabbixは「保存」ではなく「監視」のためのサーバー

全てのログを転送する設計は非推奨

サーバーやDBのリソース保護のため、アイテムキー側での厳密な**フィルタリングが重要**

Zabbixサーバーの役割を再定義

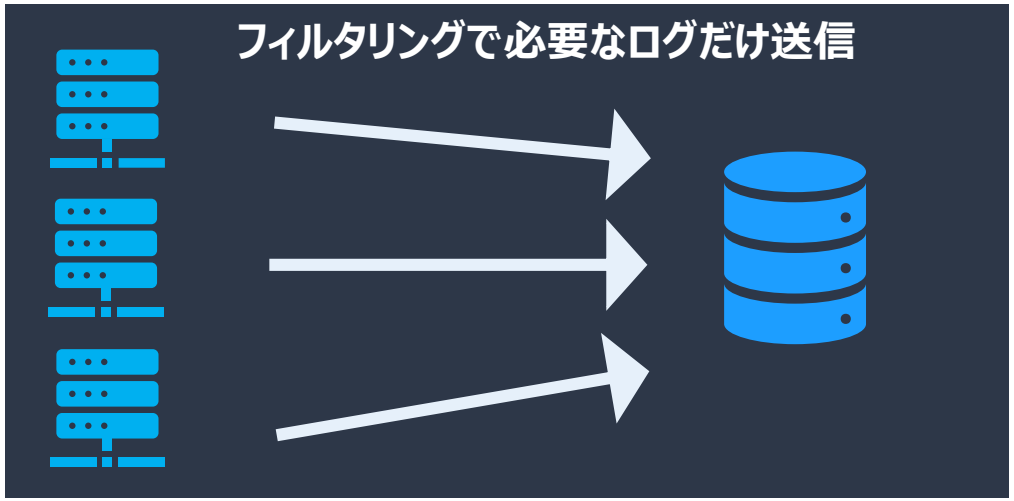
監視サーバーに大量のログ保存はDBの肥大化と
パフォーマンス低下を招く



DB肥大化やパフォーマンス低下の原因

アイテムキーでのフィルタリング

監視アイテムの第2引数（regexp）を使用して、
必要なログパターンのみをサーバーへ送信



フィルタリングによる改善

まとめ

ログ監視は「正しく読み取り」「賢く絞る」が成功の鍵

1. 監視成功の前提条件（正しく読む）

- **権限不足**は最大の落とし穴。Zabbixユーザーでの読み取り確認を徹底的に！
- **viやtouch**でファイルを触らない、**ログの長さ**に注意すべき

2. 正規表現と設定テクニック（正しく絞る）

- 誤検知を防ぐため、**正規表現の検証を行う**
- **長いログは .*?** (非貪欲マッチ) で負荷軽減
- **ファイル名は ^ と $ で厳密**に指定
- **複雑な条件は「グローバル正規表現」で整理**

3.設計思想：シンプル・イズ・ベスト

- 【基本原則】**1ログファイル ＝ 1アイテム**
- 【重要】**Zabbixはログ保存サーバーではない**（DB肥大化を防ぐ）
  → 必要なログだけをフィルタし、意味のあるデータのみを送る

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://www.sraoss.co.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