WhaTap Monitoring

ログ監視の「トイル(Toil)」をなくす

WhaTap Log Monitoring活用実践ガイド

～ Zabbixユーザー向けの『コンテキスト基盤オブザーバビリティ』入門 ～

本書の対象読者

Zabbixを中心としたインフラ監視を運用中のシステムエンジニア・DevOps担当者・IT運用管理者の皆様

目次

本書の構成

本書では、日本のIT現場でエンジニアの皆様が直面しがちな4つの具体的な課題シナリオを取り上げ、WhaTap Log Monitoringの機能がどのようにこれらを解決するかを、実践ガイドを交えてご説明します。

章	テーマ	解決する課題
第1章	SSHを使わず、Webからログをリアルタイムに監視する	SSH手動調査・ログの非構造化
第2章	アラートノイズを減らし、誤報疲弊を防ぐ	アラート疲労（Alert Fatigue）
第3章	セキュリティとログ保存問題を解決する	セキュリティ・コンプライアンス
第4章	主要ログ監視製品の機能比較	導入検討の材料

はじめに：貴社のシステム監視は、アラート受信だけで止まっていませんか？

多くのITエンジニアの皆様は、長年にわたりZabbixを心強いパートナーとして活用してこられました。その理由は極めて合理的です。ライセンス費用が要らないオープンソースであること、充実した日本国内の日本語サポート体制、そして大規模環境における長年の安定稼働実績―これらはZabbixが日本市場において揺るぎないシェアを維持し続ける正当な根拠です。

しかしながら、運用の現場では以下のような声をしばしば耳にします。

> 「Zabbixからアラートは通知されたが、そこから原因を特定するまでに1時間以上を費やしてしまった。」

> 「深夜に同一のエラーアラートが100通も届いて、眠れなかった。」

> 「アラート発生時のログを確認したいが、収集されていない。」

重要な点ですが、これらの課題は「Zabbixの設定や使い方の問題」ではありません。Zabbixが**「単一指標（点の指標）の監視」**に最適化されているのに対し、現代の複雑なシステム障害対応で求められるのは**「コンテキスト（文脈）基盤の連携分析」**だからです。

CPU使用率やレスポンスタイムといった**点の指標**だけでなく、その背後にあるアプリケーションログ・DBログ・トレース情報を**文脈（コンテキスト）として有機的に連携させて分析すること**が、現代のインフラ運

用においては不可欠となっています。

本書では、日本のIT現場でエンジニアの皆様が直面しがちな4つの具体的なシナリオを例として取り上げ、WhaTap Monitoringのログ監視機能がどのようにこれらの課題解決に役立つかを、実際の設定手順を交えてご説明します。

第1章：SSHを使わず、Webからログをリアルタイムに監視する

～ ライブテール(Live Tail)とログパース(Log Parse) ～

1.1 現場でよく遭遇するシナリオ

状況：ECサイトの大規模セール初日、午前11時。Zabbixから「Webサーバーのレスポンスタイムがしきい値を超過」というアラートが発報されました。しかし、サーバーのCPU・メモリ・ネットワーク帯域はいずれも正常範囲内です。一方、カスタマーサポートには「購入ボタンを押してもエラーになる」というクレームが相次いでいます。

Zabbixの管理画面が提示するのは、あくまで「応答が遅延している」という事実のみであり、「なぜ遅延しているのか」という**詳細内容**までは示してくれません。そのため、運用担当者は必然的に以下のような行動を余儀なくされます。

1. まず1台目のサーバーにSSH接続し、アプリケーションログを調査
2. 異常が見当たらなければ2台目、3台目へと移動して同様の調査を繰り返す
3. 各サーバーのログタイムラインを手動で突き合わせながら、障害の全体像を把握しようとする

この「複数サーバーへの手動SSH調査」を終えるまでに、熟練のエンジニアであっても30〜60分の時間を費やすことになります。障害の根本原因が判明する頃には、多くのお客様がすでにエラー画面の前で長時間を過ごした後です。

1.2 Zabbixのみで運用した場合の対応フロー

これはZabbixの本質的なアーキテクチャに起因する構造的な限界です。Zabbixは数値データである**「メトリクス指標」**の収集・通知に優れていますが、テキストデータである**「アプリケーションログ」**とメトリクスを同一画面上でコンテキストを持って関連付けられるデータモデルを持っていません。

技術的には、メトリクスは history テーブルに、ログは history_log テーブルという別々のRDB（関係型データベース）構造に格納されます。そのため、同じ時刻に発生した事象を一画面で結びつけて分析することが構造的に難しく、エンジニアが手作業でタイムラインを突き合わせる以外に方法がないのが実情です。

1.3 WhaTap Monitoringのライブテールによる解決

WhaTap Monitoringのライブテール(Live Tail)機能は、SSH接続なしにすべてのサーバーログをWeb画面上でリアルタイム(2秒間隔)にストリーミング表示する機能です。

各サーバーへ個別にSSH接続する必要は一切ありません。障害発生と同時にWhaTapのWeb画面を開くだけで、全サーバーのログが一元的に確認できます。

📌 ライブテールで解決できる問題

- 複数サーバーへの個別SSH接続が不要になり、調査時間を大幅短縮（30〜60分 → 数分）
- 全サーバーのログを単一画面でリアルタイムに並べて比較できる
- 障害発生の瞬間から即座に監視を開始でき、ログの見逃しがない
- 深夜・休日・どこからでもWebアクセスさえできれば対応可能

🛠️ 実践ガイド① ：フィルタの設定

フィルタはタグ形式で入力します。「検索キー」「オペレーター」「検索値」の順に指定するだけで、複雑な条件抽出が即座に実行されます。

例：HTTPステータス500のエラーログのみに絞り込む場合

```
検索キー    : status_code
オペレーター: ==
検索値      : 500
```

例：PaymentServiceのERRORログのみを表示する場合

```
タグ1 : level == ERROR
タグ2 : category == payment
```

※ 同一キー内はOR（||）、異なるキー間はAND（&&）で自動的に結合されます。

1.4 パーサ機能によるログの構造化

テキスト形式のログをそのままストリーミングするだけでは、「エラーが起きている」という事実しか把握できません。障害の詳細内容までを把握するには、非構造化テキストをデータベースのようにクエリ可能な**構造化データ（フィールド形式）**に整理する必要があります。

ログを構造化することで何ができるか？

- 特定フィールド（例：exit_code、exception_msg）を条件にした高速検索・絞り込み
- フィールド値を集計した統計処理（例：「ERRORが何件、WARNが何件」）
- 構造化フィールドを条件とした精緻なアラート設定（第2章）
- 個人情報フィールドを指定した自動マスキング（第3章）

WhaTap Monitoringの**ログパーサ（Log Parser）**機能を活用することで、プログラミングや設定ファイルの変更がなくても、**Web画面上の操作だけ**でログを構造化できます。

設定メニューへのアクセス：ログ > ログの設定 > ログプライマリパーサ設定

WhaTap Monitoringは、システムの特性に沿った2種類のパーサを提供しています。

タイプ	適用対象	特徴
GROKパーサ	任意のテキスト形式の非定形ログ	named regular expressions（名前付き正規表現）を使い、複雑なパターンも簡潔に記述できる
JSONパーサ	JSONフォーマットのログ	特別な設定なしに自動でパース処理が行われる

⚠️ **運用上の注意点：**同一カテゴリに複数のパーサが登録されている場合は、上位に配置された（最初にマッチした）パーサのみが適用されます。

🛠️ 実践ガイド② ：GROKパーサの設定

WhaTap Monitoringが提供するAI機能(WhaTap AI)を使ったGROKパーサの自動生成

GROKパターンの作成は、WhaTap AIを通じて自動生成することができます。ログの内容をそのまま貼り付けて、必要なフィールドを指定するだけで、最適なGROKパターンを提案してくれます。

WhaTap AIへの質問例：

```
ログの内容を以下に示します。
[2026-06-10 03:24:39] [ERROR] [Backup_new_customer_JOB] job_name=Backup_new_customer_JOB proce
ss=BACKUP_PROCESS exit_code=107 exception=I/O Exception: Customer infomation insert aborted -
customer No. 40465367, mobile 07095029589, name Hanako Suzuki, city No. 218, email h.suzuki@so
ftbank.ne.jp

上記ログのGROKパーサを作成してください。
必要なデータは、ログ収集時間、job_name、process、exit_code、exception です。
個人情報のマスキング処理のため、次のデータも一緒にパースします：mobile、name、city No.、email
```

生成したGROKパーサの設定

WhaTap MonitoringのWeb画面上で、以下のようにGROKパターンを入力します。以下のようなアプリケーションログを構造化する場合を例に挙げます。

```
# パース前の正常ログ
[2025-12-01 11:03:22] [NORMAL] [Systemwalker_販売系_JOB] job_name=Systemwalker_販売系_JOB process=SUB_PROCESS exit_code=0

# パース前のエラーログ
[2025-12-01 11:03:22] [ERROR] [BI_Schedule_Status] job_name=BI_Schedule_Status process=SUB_PROCESS exit_code=105 exception=SchedulerException: BI report generation aborted - DB connection pool exhausted
```

GROKパターン：

```
\[%{TIMESTAMP_ISO8601:timestamp}\] \[%{WORD:level}\] \[%{DATA:job_name}\] job_name=%{DATA:job_id} process=%{WORD:process} exit_code=%{INT:exit_code}( exception=%{GREEDYDATA:exception_msg})?
```

パース シミュレーション処理が完了すると、テキストログは以下のように整然としたフィールドデータに変換されます。

```
log_timestamp : 2025-12-01 11:03:22
level         : ERROR
job_name      : BI_Schedule_Status
job_id        : BI_Schedule_Status
process       : SUB_PROCESS
exit_code     : 105
exception_msg : SchedulerException: BI report generation ...
```

構造化されたデータは、ログエクスプローラー（Apache Lucene 8.11.1準拠）において、以下のような高度な検索条件として即座に活用できます。

```
job_name:Systemwalker_JDE送信_JOB and exception_msg:*SQLException*Timeout*
→ SystemwalkerのJDE送信ジョブでSQL接続タイムアウト例外が発生したログのみをピンポイント検索
```

ログパース機能を提供する主要ツールとの比較

ツール	パースの手法	運用上の課題
Zabbix	単純な正規表現マッチングが中心	特定文字列の有無は検知できるが、個別の値を抽出して検索・集計する機能が弱い
ELK Stack	Logstash等の設定ファイルにコードを記述	ログフォーマットが変更されるたびに、パイプラインコードの修正と再デプロイが必要
WhaTap	Web画面でのノーコード設定	専門知識やコード記述は不要。設定内容は即時に反映される

1.5 WhaTap Log Monitoringのログ検索性能

WhaTapのログ検索（Apache Lucene 8.11.1準拠）では、1回の照会で**最大10,000件**のマッチングログを**瞬時**に取得でき、スクロールによって過去データを途切れなく追加照会することも可能です。

また、頻繁に使用する検索クエリ条件は**「フィルターお気に入り」に最大50件まで保存**できます。チームで共通の調査パターン（例：「決済ERRORのみ」「認証WARNのみ」）をあらかじめ登録しておくことで、障害発生時にワンクリックで対応画面を即座に呼び出せます。

第2章：アラートノイズを減らし、誤報疲弊を防ぐ

～ 複合ログイベントとスマートアラート設定 ～

本章の範囲について：本章では、**ログ監視の一環としてのアラート設定**に範囲を限定して解説します。より本格的なアラート設定・ノイズ対策・アラート起点の原因分析については、別途「アラート実践ガイド」編で詳述します。

2.1 「アラート疲労」というインフラ運用の現実

伝統的なインフラ監視ツールでよく見られる単純なキーワードマッチング―「ログに『ERROR』という文字列が記録されたら即時通知する」―という設定は、現場エンジニアの疲弊を招く原因となります。

例えば、夜間のバッチ処理中に一時的なネットワーク瞬断が発生し、DBコネクションエラーログが1分間に60回記録されたとします。従来の方式では、管理者のメールボックスやSlackチャンネルに60通ものアラートが押し寄せます。深夜に起こされた担当者がシステムに接続すると、すでに2分程度で自動回復（セルフヒーリング）しているという状況が繰り返されます。

このような現象が常態化すると、担当者は心理的に「どうせまた自然に回復する」とアラートを軽視するようになります。結果として、**本当に深刻なクリティカル障害が発生した際に、膨大なアラートノイズの中に埋もれて見逃されてしまう**という致命的なリスクに繋がります。これをIT業界では**「アラートの狼少年化（Alert Fatigue）」**と呼び、システム監視体制における最大の盲点として警鐘が鳴らされています。

2.2 3種類のアラートイベント活用ガイド

WhaTapは、このアラート疲労問題を根本から解決するため、用途に応じた3種類のスマートアラート機能を提供しています。

イベントタイプ	検知対象	最適な活用シナリオ
リアルタイムログイベント	特定の単一ログ本文のキーワードマッチング	1件発生した時点で即座に対応が必要な致命的エラー
メトリクスイベント	ログ発生件数・数値フィールドの統計的しきい値	「5分間で特定のエラー率が平時より20%急増した」といった量的異常の検知
複合ログイベント	時間軸・連続性・複数条件のインテリジェントな結合検査	一時的な誤検知を排除し、持続的かつ本質的な障害のみを精緻に検出する最上位のアラート

アラートノイズを制御し、担当者の夜間待機における負荷を軽減する鍵が、上記の**「複合ログイベント（Log Composite Event）」**です。

2.3 複合ログイベントの設定ガイド

設定メニューへのアクセス：ホーム ＞ プロジェクト選択 ＞ アラート通知 ＞ イベント設定 ＞ 複合ログ ＞ ＋アラート通知追加

複合ログイベントは、複雑なスクリプトのコーディングなしに、Web画面上の操作だけで「時間」と「件数」を組み合わせた条件を設定できます。設定項目は4つです。

① イベント名とカテゴリを決める
管理者が識別しやすいタイトル（例：「決済DBコネクションエラー連続検知」）を設定し、監視対象のログカテゴリを選択します。

② 間隔（データ照会範囲、時間軸）を設定する
「何分間以内に」を定義する時間軸の設定です（例：5分）。

③ 件数条件を設定する
設定した時間範囲内で「何件以上累積したらアラートを発報するか」を指定します（例：4件以上）。

④ フィルター条件で対象ログを絞り込む
パースされた構造化フィールドを活用し、アラート対象を精緻に絞り込みます（例：level == ERROR AND job_name == BI_Schedule_Status）。

設定後の動作イメージ

```
【設定例】
タイトル ： BIスケジュールエラー
カテゴリ ： \log\job_execute.log
照会範囲 ： 5分以内
条件 ： 4件以上発生した場合にアラート送信
フィルター ： level : "ERROR" and job_name : "BI_Schedule_Status"

【動作結果】
・5分以内に該当ERRORが3件 → アラートなし（自動回復を待機）
・5分以内に該当ERRORが4件以上 → アラート1通を送信
```

Zabbixで1時間当たり10通以上送っていたものが、**1〜2通**に減ります。

🛠️ 実践ガイド③ ：運用シナリオ別のアラート設定

複合ログイベントは「どの条件を組み合わせるか」が分かれば、ほとんどのケースに応用できます。以下に代表的な3パターンをご紹介します。

シナリオA：夜間バッチの間欠エラー対策

```
タイトル ： 夜間バッチDB接続失敗（継続判定）
カテゴリ ： BatchLog
照会範囲 ： 10分
件数条件 ： 15件以上
フィルター ： exception_msg == *connection*pool* AND level == ERROR

→ 10分間に同一エラーが15件以上継続した場合のみアラート
→ 2〜3分の間欠エラー（10件未満）は通知しない
```

シナリオB：不正アクセス疑いの自動検知

```
タイトル ： ログイン失敗連続（ブルートフォース疑い）
カテゴリ ： AuthLog
照会範囲 ： 5分
件数条件 ： 10件以上
フィルター ： action == login_failed AND level == WARN

→ 5分間に同一カテゴリのWARNが10件以上 → セキュリティアラートを送信
```

シナリオC：外部API連続タイムアウト検知

```
タイトル ： 外部決済API障害（連続タイムアウト）
カテゴリ ： AppLog
照会範囲 ： 3分
件数条件 ： 5件以上
フィルター ： service == ExternalPaymentAPI AND exception_msg == *timeout*

→ 3分以内に外部APIタイムアウトが5件以上 → Criticalアラートを送信
```

第3章：セキュリティとログ保存問題を解決する

～ 非識別化と保存期間の設定 ～

3.1 厳格なセキュリティ・コンプライアンスの壁

サービスの開発・デバッグの過程において、エンジニアはログファイル内にユーザーのメールアドレス・電話番号・会員IDなどを記録することがあります。これは事後の障害追跡には有用である反面、日本の「改正個人情報保護法（2022年）」や各種セキュリティガイドラインを遵守する企業のコンプライアンス部門にとっては重大な問題となります。

「外部クラウド上のSaaSサーバーへ顧客の個人情報を生データのまま転送することはセキュリティ規程上問題であり、ログ収集を全面的に制限する」という声を耳にするケースも少なくありません。結果として、現場チームがサーバーへのSSH接続による手動分析という非効率な方式に逆戻りしてしまう状況が生じています。

3.2 ログ内個人情報の非識別化（フィールド単位のマスキング）

ケーススタディ：新規顧客情報バックアップジョブのPII問題

新規顧客情報のバックアップを行うバッチジョブ（Backup_new_customer_JOB）を例に挙げます。このジョブは、顧客データのDBインサート処理に失敗すると、デバッグのために顧客の個人情報を含むエラーログを出力します。

```
# バックアップ中、エラー発生時のログ（マスキング前）
[2025-12-01 11:03:22] [ERROR] [Backup_new_customer_JOB] job_name=Backup_new_customer_JOB proce
ss=BACKUP_PROCESS exit_code=107 exception=I/O Exception: Customer infomation insert aborted -
customer No. 00279978, mobile 07014142525, name Taro Hayashi, city No. 107, email t.hayashi@gm
ail.com

# バッチ完了時の結果ログ（エラーあり）
[2025-12-01 11:03:22] [FINISH] [Backup_new_customer_JOB] job_name=Backup_new_customer_JOB proc
ess=BACKUP_PROCESS exit_code=1 result=total 1027, pass 1025, error 2

# バッチ完了時の結果ログ（エラーなし）
[2025-12-01 11:03:22] [FINISH] [Backup_new_customer_JOB] job_name=Backup_new_customer_JOB proc
ess=BACKUP_PROCESS exit_code=0 result=total 1027, pass 1027, error 0
```

このログには氏名・携帯番号・メールアドレスという複数の個人情報が含まれています。しかし**障害調査の観点**では、顧客番号（customer No.）さえ分かれば、DBやCRMシステムで該当レコードを特定できます。つまり、顧客番号だけを残して他のPIIをマスキングすることが、セキュリティとデバッグ効率の両立にとって最善の設計です。

🛠️ 実践ガイド④ ：ログ内個人情報の非識別化

WhaTap Monitoringでの個人情報非識別化は、**「パース」と「フィールド指定」の2ステップ**で実現します。

```
Step 1: GROKパーサでフィールドを抽出
    customer_no / mobile / name / city_no / email などを個別フィールドとして構造化

Step 2: 個人情報非識別化設定でマスキング対象フィールドを選択
    mobile / name / city_no / email → WhaTap画面上で *** に装飾して表示
    customer_no → 選択しない（検索・調査に使用可能なまま残す）
```

Step 1：GROKパーサ設定

まず、ログから各フィールドを抽出するGROKパターンを登録します。

> **設定メニューへのアクセス：ログの設定 > ログパーシング > GROKパーサ追加**

本バッチジョブのログには3種類のパターンが存在するため、以下の3パターンを優先順位の順に登録します。

パターン①：バックアップ ERROR ログ

```
\[%{TIMESTAMP_ISO8601:log_timestamp}\] \[ERROR\] \[Backup_new_customer_JOB\] job_name=Backup_new_customer_JOB process=%{WORD:process} exit_code=%{INT:exit_code} exception=I/O Exception: Customer infomation insert aborted - customer No\. %{INT:customer_no}, mobile (?<mobile>[^,]+), name (?<name>[^,]+), city No\. %{INT:city_no}, email (?<email>\S+)
```

抽出されるフィールド：

フィールド	値の例	備考
log_timestamp	2025-12-01 11:03:22	検索・アラートの時間軸
exit_code	107	アラート条件に使用
customer_no	00279978	PIIなし。CRM/DB検索キーとして活用
mobile	07014142525	Step 2でマスキング対象に指定
name	Taro Hayashi	Step 2でマスキング対象に指定
city_no	107	Step 2でマスキング対象に指定
email	t.hayashi@gmail.com	Step 2でマスキング対象に指定

パターン②：バックアップ FINISH ログ（バッチ完了集計）

```
\[%{TIMESTAMP_ISO8601:log_timestamp}\] \[FINISH\] \[Backup_new_customer_JOB\] job_name=Backup_new_customer_JOB process=%{WORD:process} exit_code=%{INT:exit_code} result=total %{INT:result_total}, pass %{INT:result_pass}, error %{INT:result_error}
```

result_error フィールドを複合ログイベントの条件に使うことで「バッチ1回で10件以上の登録エラーが発生した場合のみ通知」という精緻なアラート設定が可能です。

Step 2：個人情報非識別化設定

GROKパースが完了したら、次に非識別化設定でマスキング対象のフィールドを指定します。

設定メニューへのアクセス：ログ設定 > 個人情報の非識別化

カテゴリとマスキング対象フィールドを選択するだけで設定完了です。customer_no はここに含めないため、WhaTap画面上でそのまま参照・検索できます。

フィールド	非識別化設定	WhaTap画面での表示
customer_no	対象外	00279978（そのまま表示・検索可能）
mobile	✓対象	***
name	✓対象	***
city_no	✓対象	***
email	✓対象	***

マスキング後のログは以下のように変換されます。customer No. の値は残り、障害調査に必要な最低限の情報が維持されます。

```
# マスキング後（WhaTapクラウドに届くログ）
[2025-12-01 11:03:22] [ERROR] [Backup_new_customer_JOB] job_name=Backup_new_customer_JOB process=BACKUP_PROCESS exit_code=107 exception=I/O Exception: Customer infomation insert aborted - customer No. 00279978, mobile ***, name ***, city No. ***, email ***
```

他ツールとの比較

ツール	個人情報マスキング	運用上の課題
Zabbix	標準機能なし	ソースコードレベルでの対応が必要
ELK Stack	Logstashフィルタで実装可能	パイプラインのコード管理が負担
WhaTap	UIで正規表現を登録するだけ。収集前に自動適用	コード不要・コンプライアンス対応が容易

3.3 コスト最適化：ログ長期保管統計機能

法的規制や監査対応のために6か月から1年以上のログ保管義務が課せられる一方で、膨大なログデータをそのまま保持し続けるためのストレージコストに課題を抱える企業は少なくありません。

WhaTapは、この問題を解決する**「ログ長期保管統計設定」**を提供しています。

この機能は、すべての生テキストデータ（Raw Log）を高額なストレージにそのまま蓄積するのではなく、事後の監査に必要な核心的な統計情報（例：「何月何日何時に、どのサービスでERRORログが何件発生したか」というメタデータ）へと圧縮・構造化し、**最大1年間、低コストで長期保管**するアーキテクチャです。

> **設定メニューへのアクセス：ログの設定 > ログ長期保管統計設定**

全量保管と統計保管の比較

```
【従来の全量ログ保管】
実データ : 2025-12-01 11:03:22 ERROR PaymentService user_id=1023 ...
           2025-12-01 11:03:22 ERROR PaymentService user_id=1024 ...
           （以下、同様のレコードが数千〜数百万行）
→ ストレージコスト：大

【長期保管統計設定】
統計データ :
  2025-12-01 11:00→11:05 | category=AppLog, level=ERROR, service=PaymentService | 件数: 187件
→ 集計後の統計情報のみを1年間保管
→ ストレージコスト：極小
```

不要なストレージコストを大幅に抑えながら、監査・コンプライアンス要件（ISO27001、PCI DSS等）が求める履歴の提出証跡を充足できます。

第4章：主要ログ監視製品の機能比較

4.1 ログ監視機能比較表

比較項目	Zabbix	ELK Stack	Splunk	WhaTap
リアルタイムログ表示	× SSHが必要	△ 設定が複雑	○	◎ 約2秒周期のLive Tail
ログパース（構造化）	× マッチングのみ	× コーディング必要	△	◎ UIでノーコード設定
インテリジェント複合アラート	× 単純マッチング	△ 外部ツール依存	○ 高習得コスト	◎ GUIで複合条件を設定
AI生成型運用支援	×	△ 外部LLM連携必要	△ 専用アドオン必要	◎ WhaTap AI を標準搭載
個人情報の非識別化	×	△ 専門知識必要	△ Heavy Forwarder設定必要	◎ エージェント側で自動マスキング
長期保管（統計）	△ 別途設計必要	△ 運用が複雑	○ 追加費用	◎ 最大1年間の統計アーカイブ
導入・維持管理の工数	中 手動チューニング	大 クラスター管理	小 ただし高額	◎ 5分でインストール、完全管理型SaaS

4.2 WhaTap Log Monitoringの主な仕様

スペック項目	数値・仕様	参照ドキュメント
ライブテール更新周期	約2秒（リアルタイムストリーミング）	docs.whatap.io/ja/log/log-lt
1回の検索最大表示件数	最大10,000件/回（追加照会可能）	docs.whatap.io/ja/log/log-search
フィルターお気に入り 最大保存数	最大50件	docs.whatap.io/ja/log/log-search
ログエクスプローラー 対応クエリエンジン	Apache Lucene 8.11.1 完全準拠	docs.whatap.io/en/mssql/log-explorer
長期保管統計のデータ保存期間	最大1年間	docs.whatap.io/ja/log/log-long-term-archive
WhaTap AI 対応言語	日本語・英語・韓国語	docs.whatap.io/ja/ai/introduction

おわりに：インフラ監視のパラダイムを変える第一歩

Zabbixは今日においても、多くの日本企業のシステム安定性を支え続けている信頼性の高いインフラ監視ツールです。WhaTap Monitoringの目指すところは、既存のZabbixをすべて廃棄・リプレイスすることではありません。

Zabbixが従来のメトリクスアラートを通じて伝えてくれる「システム状態の異常の有無」という一次情報はそのまま維持しながら、その先でエンジニアの貴重な時間を消耗させてきた「SSH接続後の手作業によるログ分析」というトイル(Toil)を、**WhaTapがスマートに代替する**ことが本来の役割です。

本ガイドでご紹介した中核的な機能―リアルタイムのライブテール、ノーコードのログパース、インテリジェントな複合アラート、コンプライアンス対応のための非識別化―は、監視サーバーを構築したり、ストレージを設置することなく、WhaTapエージェントを追加するだけで、今日からご活用いただけます。

運用担当者の夜間対応負荷を軽減し、システムへの洞察（インサイト）を次のステージへと引き上げるWhaTap Log Monitoringを、ぜひご体験ください。

参考リンク（日本語版のWhaTap公式ドキュメント）

ドキュメント	URL
ログモニタリング 概要	https://docs.whatap.io/ja/log/introduction
ライブテール	https://docs.whatap.io/ja/log/log-lt
ログエクスプローラー	https://docs.whatap.io/ja/log/log-explorer
WhaTap log search query 文法	https://docs.whatap.io/ja/log/log-search-query
ログの検索	https://docs.whatap.io/ja/log/log-search
ログパーシング	https://docs.whatap.io/ja/log/log-parser
複合ログイベント設定	https://docs.whatap.io/ja/log/alert/event-configuration/log-composite
ログ個人情報非識別化設定	https://docs.whatap.io/ja/log/log-deidentification
ログ長期保管統計設定	https://docs.whatap.io/ja/log/log-long-term-archive
WhaTap AI	https://docs.whatap.io/ja/ai/introduction
MySQL ログ収集設定	https://docs.whatap.io/ja/mysql/log-db